

玄铁 C906 R1S0 用户手册

2020 年 10 月 30 日

Copyright © 2020 平头哥半导体有限公司，保留所有权利。

本档的产权属于平头哥半导体有限公司（下称“平头哥”）。本档仅能分布给：(i) 拥有合法雇佣关系，并需要本档的信息的平头哥员工，或 (ii) 非平头哥组织但拥有合法合作关系，并且其需要本档的信息的合作方。对于本档，禁止任何在专利、版权或商业秘密过程中，授予或暗示的可以使用该文档。在没有得到平头哥半导体有限公司的书面许可前，不得复制本档的任何部分，传播、转录、储存在检索系统中或翻译成任何语言或计算机语言。

商标申明

平头哥的 LOGO 和其它所有商标归平头哥半导体有限公司及其关联公司所有，未经平头哥半导体有限公司的书面同意，任何法律实体不得使用平头哥的商标或者商业标识。

注意

您购买的产品、服务或特性等应受平头哥商业合同和条款的约束，本档中描述的全部或部分产品、服务或特性可能不在您的购买或使用范围之内。除非合同另有约定，平头哥对本档内容不做任何明示或默示的声明或保证。由于产品版本升级或其他原因，本档内容会不定期进行更新。除非另有约定，本档仅作为使用指导，本档中的所有陈述、信息和建议不构成任何明示或暗示的担保。平头哥半导体有限公司不对任何第三方使用本档产生的损失承担任何法律责任。

Copyright © 2020 T-HEAD Semiconductor Co.,Ltd. All rights reserved.

This document is the property of T-HEAD Semiconductor Co.,Ltd. This document may only be distributed to: (i) a T-HEAD party having a legitimate business need for the information contained herein, or (ii) a non-T-HEAD party having a legitimate business need for the information contained herein. No license, expressed or implied, under any patent, copyright or trade secret right is granted or implied by the conveyance of this document. No part of this document may be reproduced, transmitted, transcribed, stored in a retrieval system, translated into any language or computer language, in any form or by any means, electronic, mechanical, magnetic, optical, chemical, manual, or otherwise without the prior written permission of T-HEAD Semiconductor Co.,Ltd.

Trademarks and Permissions

The T-HEAD Logo and all other trademarks indicated as such herein are trademarks of Hangzhou T-HEAD Semiconductor Co.,Ltd. All other products or service names are the property of their respective owners.

Notice

The purchased products, services and features are stipulated by the contract made between T-HEAD and the customer. All or part of the products, services and features described in this document may not be within the purchase scope or the usage scope. Unless otherwise specified in the contract, all statements, information, and recommendations in this document are provided "AS IS" without warranties, guarantees or representations of any kind, either express or implied. The information in this document is subject to change without notice. Every effort has been made in the preparation of this document to ensure accuracy of the contents, but all statements, information, and recommendations in this document do not constitute a warranty of any kind, express or implied.

平头哥半导体有限公司 T-HEAD Semiconductor Co.,LTD

地址: 杭州市余杭区向往街 1122 号欧美金融城 (EFC) 英国中心西楼 T6

邮编: 311121

网址: www.t-head.cn

版本历史

版本	描述	日期
01	第一次正式发布。	2020.07.10
02	增加 TLB 访问延时信息描述。	2020.08.21
03	修复部分描述错误。	2020.10.19

买头哥半身体

目录

第一章 概述	1
1.1 简介	1
1.2 特点	1
1.3 可配置选项	1
1.4 命名规则	2
1.4.1 符号	2
1.4.2 术语	3
1.5 版本说明	3
第二章 处理器简介	4
2.1 结构框图	4
2.2 单元介绍	4
2.2.1 指令提取单元	5
2.2.2 指令译码单元	5
2.2.3 执行单元	5
2.2.4 存储载入单元	5
2.2.5 虚拟内存管理单元	6
2.2.6 物理内存保护单元	6
2.2.7 主设备接口	6
2.2.8 平台级中断控制器	6
2.2.9 计时器	6
2.3 接口	6
第三章 编程模型	8
3.1 工作模式及寄存器视图	8
3.2 通用寄存器	9
3.3 浮点寄存器	9
3.3.1 与通用寄存器传输数据	10
3.3.2 维护寄存器精度的一致	10
3.4 矢量寄存器	10
3.4.1 与通用寄存器传输数据	11
3.4.2 与浮点寄存器传输数据	11
3.5 系统控制寄存器	11
3.5.1 机器模式控制寄存器	11
3.5.2 超级用户模式控制寄存器	13
3.5.3 用户模式控制寄存器	14

3.6	异常处理	15
3.7	数据格式	17
3.7.1	浮点数据格式	17
3.7.2	整型数据格式	18
3.7.3	矢量数据格式	18
3.7.4	大小端	19
3.8	内存模型	19
第四章	指令集	21
4.1	RV64GCV 指令	21
4.1.1	RV64I 整型指令集	21
4.1.2	RV64M 乘除法指令集	24
4.1.3	RV64A 原子指令集	25
4.1.4	RV64F 单精度浮点指令集	25
4.1.5	RV64D 双精度浮点指令集	27
4.1.6	RVC 压缩指令集	28
4.1.7	RVV 矢量指令集	30
4.2	平头哥扩展指令集	40
4.2.1	Cache 指令子集	40
4.2.2	同步指令子集	41
4.2.3	算术运算指令子集	41
4.2.4	位操作指令子集	42
4.2.5	存储指令子集	42
4.2.6	半精度浮点指令子集	43
第五章	虚拟内存管理	46
5.1	MMU 概览	46
5.2	编程模型和地址转换	46
5.2.1	MMU 控制寄存器	46
5.2.1.1	MMU 控制寄存器 (SMCIR)	46
5.2.1.2	MMU Index 寄存器 (SMIR)	47
5.2.1.3	MMU EntryHi 寄存器 (SMEH)	48
5.2.1.4	MMU EntryLo 寄存器 (SMEL)	49
5.2.1.5	MMU 地址转换寄存器 (SATP)	51
5.2.2	地址转换流程	51
5.2.2.1	页表结构	52
5.2.2.2	地址转换流程	52
5.3	TLB 组织形式	53
第六章	物理内存保护	54
6.1	PMP 概览	54
6.2	PMP 控制寄存器	54
6.2.1	物理内存保护设置寄存器 (PMPCFG)	54
6.2.2	物理内存保护地址寄存器 (PMPADDR)	56
第七章	内存子系统	58
7.1	指令高速缓存子系统	58

7.1.1	指令预取	58
7.1.2	分支历史表	58
7.1.3	分支跳转目标预测器	59
7.1.4	返回地址预测器	59
7.2	数据高速缓存子系统	59
7.2.1	数据预取	60
7.2.2	自适应的写分配机制	60
7.2.3	独占式访问	61
7.3	L1 高速缓存操作	61
7.3.1	L1 高速缓存扩展寄存器	61
7.3.2	L1 高速缓存扩展指令	61
第八章	总线接口协议	63
8.1	主设备接口	63
8.1.1	特点	63
8.1.2	协议内容	63
8.1.2.1	支持传输类型	63
8.1.2.2	支持响应类型	64
8.1.3	不同总线响应下的行为	64
第九章	处理器核局部中断 CLINT	65
9.1	寄存器地址映射	65
9.2	软件中断	65
9.3	计时器中断	66
第十章	中断控制器 PLIC	68
10.1	中断处理机制	68
10.1.1	中断仲裁	68
10.1.2	中断请求与响应	69
10.1.3	中断完成	69
10.2	PLIC 寄存器地址映射	69
10.3	中断优先级配置寄存器 (PLIC_PRIO)	70
10.4	中断等待寄存器 (PLIC_IP)	71
10.5	中断使能寄存器 (PLIC_IE)	71
10.6	PLIC 权限控制寄存器 (PLIC_CTRL)	72
10.7	中断阈值寄存器 (PLIC_TH)	72
10.8	中断响应/完成寄存器 (PLIC_CLAIM)	73
第十一章	调试接口	74
11.1	概述	74
11.2	接口描述	75
第十二章	性能监测单元	77
12.1	性能监测控制寄存器	77
12.1.1	机器模式计数器访问授权寄存器 (MCOUNTEREN)	77
12.1.2	超级用户模式计数器访问授权寄存器 (SCOUNTEREN)	78
12.1.3	机器模式计数禁止寄存器 (MCOUNTINHIBIT)	79

12.1.4	机器模式计数器写使能授权寄存器 (MCOUNTERWEN)	80
12.2	性能监测事件选择寄存器	80
12.3	事件计数器	81
12.4	性能检测单元事件溢出中断	82
第十三章	程序示例	85
13.1	MMU 设置示例	85
13.2	PMP 设置示例	88
13.3	高速缓存设置示例	89
13.3.1	高速缓存的开启示例	89
13.3.2	指令高速缓存与数据高速缓存的同步示例	90
13.3.3	TLB 与数据高速缓存的同步示例	90
13.4	PLIC 设置示例	90
13.5	HPM 设置示例	91
第十四章	附录 A 标准指令术语	93
14.1	附录 A-1 I 指令术语	93
14.1.1	ADD——有符号加法指令	93
14.1.2	ADDI——有符号立即数加法指令	93
14.1.3	ADDIW——低 32 位有符号立即数加法指令	94
14.1.4	ADDW——低 32 位有符号加法指令	94
14.1.5	AND——按位与指令	95
14.1.6	ANDI——立即数按位与指令	95
14.1.7	AUIPC——PC 高位立即数加法指令	96
14.1.8	BEQ——相等分支指令	96
14.1.9	BGE——有符号大于等于分支指令	97
14.1.10	BGEU——无符号大于等于分支指令	98
14.1.11	BLT——有符号小于分支指令	98
14.1.12	BLTU——无符号小于分支指令	99
14.1.13	BNE——不等分支指令	100
14.1.14	CSRRC——控制寄存器清零传送指令	100
14.1.15	CSRRCI——控制寄存器立即数清零传送指令	101
14.1.16	CSRRS——控制寄存器置位传送指令	101
14.1.17	CSRRSI——控制寄存器立即数置位传送指令	102
14.1.18	CSRROW——控制寄存器读写传送指令	103
14.1.19	CSRROWI——控制寄存器立即数读写传送指令	103
14.1.20	EBREAK——断点指令	104
14.1.21	ECALL——环境异常指令	104
14.1.22	FENCE——存储同步指令	105
14.1.23	FENCE.I——指令流同步指令	105
14.1.24	JAL——直接跳转子程序指令	106
14.1.25	JALR——寄存器跳转子程序指令	106
14.1.26	LB——有符号扩展字节加载指令	107
14.1.27	LBU——无符号扩展字节加载指令	107
14.1.28	LD——双字加载指令	108
14.1.29	LH——有符号扩展半字加载指令	108

14.1.30	LHU——无符号扩展半字加载指令	109
14.1.31	LUI——高位立即数装载指令	109
14.1.32	LW——有符号扩展字加载指令	110
14.1.33	LWU——无符号扩展字加载指令	110
14.1.34	MRET——机器模式异常返回指令	111
14.1.35	OR——按位或指令	111
14.1.36	ORI——立即数按位或指令	112
14.1.37	SB——字节存储指令	112
14.1.38	SD——双字存储指令	113
14.1.39	SFENCE.VMA——虚拟内存同步指令	113
14.1.40	SH——半字存储指令	114
14.1.41	SLL——逻辑左移指令	114
14.1.42	SLLI——立即数逻辑左移指令	115
14.1.43	SLLIW——低 32 位立即数逻辑左移指令	115
14.1.44	SLLW——低 32 位逻辑左移指令	116
14.1.45	SLT——有符号比较小于置位指令	116
14.1.46	SLTI——有符号立即数比较小于置位指令	117
14.1.47	SLTIU——无符号立即数比较小于置位指令	117
14.1.48	SLTU——无符号比较小于置位指令	118
14.1.49	SRA——算数右移指令	118
14.1.50	SRAI——立即数算数右移指令	119
14.1.51	SRAIW——低 32 位立即数算数右移指令	119
14.1.52	SRAW——低 32 位算数右移指令	120
14.1.53	SRET——超级用户模式异常返回指令	120
14.1.54	SRL——逻辑右移指令	121
14.1.55	SRLI——立即数逻辑右移指令	121
14.1.56	SRLIW——低 32 位立即数逻辑右移指令	122
14.1.57	SRLW——低 32 位逻辑右移指令	122
14.1.58	SUB——有符号减法指令	123
14.1.59	SUBW——低 32 位有符号减法指令	123
14.1.60	SW——字存储指令	124
14.1.61	WFI——进入低功耗模式指令	124
14.1.62	XOR——按位异或指令	125
14.1.63	XORI——立即数按位异或指令	125
14.2	附录 A-2 M 指令术语	125
14.2.1	DIV——有符号除法指令	126
14.2.2	DIVU——无符号除法指令	126
14.2.3	DIVUW——低 32 位无符号除法指令	127
14.2.4	DIVW——低 32 位有符号除法指令	127
14.2.5	MUL——有符号乘法指令	128
14.2.6	MULH——有符号乘法取高位指令	128
14.2.7	MULHSU——有符号无符号乘法取高位指令	129
14.2.8	MULHU——无符号乘法取高位指令	129
14.2.9	MULW——低 32 位有符号乘法指令	130
14.2.10	REM——有符号取余指令	130

14.2.11	REMU——无符号取余指令	131
14.2.12	REMUW——低 32 位无符号取余指令	131
14.2.13	REMW——低 32 位有符号取余指令	132
14.3	附录 A-3 A 指令术语	132
14.3.1	AMOADD.D——原子加法指令	132
14.3.2	AMOADD.W——低 32 位原子加法指令	133
14.3.3	AMOAND.D——原子按位与指令	134
14.3.4	AMOAND.W——低 32 位原子按位与指令	135
14.3.5	AMOMAX.D——原子有符号取最大值指令	135
14.3.6	AMOMAX.W——低 32 位原子有符号取最大值指令	136
14.3.7	AMOMAXU.D——原子无符号取最大值指令	137
14.3.8	AMOMAXU.W——低 32 位原子无符号取最大值指令	138
14.3.9	AMOMIN.D——原子有符号取最小值指令	139
14.3.10	AMOMIN.W——低 32 位原子有符号取最小值指令	139
14.3.11	AMOMINU.D——原子无符号取最小值指令	140
14.3.12	AMOMINU.W——低 32 位原子无符号取最小值指令	141
14.3.13	AMOOR.D——原子按位或指令	142
14.3.14	AMOOR.W——低 32 位原子按位或指令	143
14.3.15	AMOSWAP.D——原子交换指令	143
14.3.16	AMOSWAP.W——低 32 位原子交换指令	144
14.3.17	AMOXOR.D——原子按位异或指令	145
14.3.18	AMOXOR.W——低 32 位原子按位异或指令	146
14.3.19	LR.D——双字加载保留指令	146
14.3.20	LR.W——字加载保留指令	147
14.3.21	SC.D——双字条件存储指令	148
14.3.22	SC.W——字条件存储指令	149
14.4	附录 A-4 F 指令术语	149
14.4.1	FADD.S——单精度浮点加法指令	150
14.4.2	FCLASS.S——单精度浮点分类指令	150
14.4.3	FCVT.L.S——单精度浮点转换成有符号长整型指令	151
14.4.4	FCVT.LU.S——单精度浮点转换成无符号长整型指令	152
14.4.5	FCVT.S.L——有符号长整型转换成单精度浮点数指令	153
14.4.6	FCVT.S.LU——无符号长整型转换成单精度浮点数指令	154
14.4.7	FCVT.S.W——有符号整型转换成单精度浮点数指令	155
14.4.8	FCVT.S.WU——无符号整型转换成单精度浮点数指令	155
14.4.9	FCVT.W.S——单精度浮点转换成有符号整型指令	156
14.4.10	FCVT.WU.S——单精度浮点转换成无符号整型指令	157
14.4.11	FDIV.S——单精度浮点除法指令	158
14.4.12	FEQ.S——单精度浮点比较相等指令	159
14.4.13	FLE.S——单精度浮点比较小于等于指令	159
14.4.14	FLT.S——单精度浮点比较小于指令	160
14.4.15	FLW——单精度浮点加载指令	161
14.4.16	FMADD.S——单精度浮点乘累加指令	161
14.4.17	FMAX.S——单精度浮点取最大值指令	162
14.4.18	FMIN.S——单精度浮点取最小值指令	163

14.4.19	FMSUB.S	—单精度浮点乘累减指令	163
14.4.20	FMUL.S	—单精度浮点乘法指令	164
14.4.21	FMV.W.X	—单精度浮点写传送指令	165
14.4.22	FMV.X.W	—单精度浮点寄存器读传送指令	165
14.4.23	FNMADD.S	—单精度浮点乘累加取负指令	166
14.4.24	FNMSUB.S	—单精度浮点乘累减取负指令	167
14.4.25	FSGNJS	—单精度浮点符号注入指令	168
14.4.26	FSGNJS	—单精度浮点符号取反注入指令	168
14.4.27	FSGNJS	—单精度浮点符号异或注入指令	169
14.4.28	FSQRT.S	—单精度浮点开方指令	169
14.4.29	FSUB.S	—单精度浮点减法指令	170
14.4.30	FSW	—单精度浮点存储指令	171
14.5	附录 A-5 D 指令术语		171
14.5.1	FADD.D	—双精度浮点加法指令	172
14.5.2	FCLASS.D	—双精度浮点分类指令	172
14.5.3	FCVT.D.L	—有符号长整型转换成双精度浮点数指令	173
14.5.4	FCVT.D.LU	—无符号长整型转换成双精度浮点数指令	174
14.5.5	FCVT.D.S	—单精度浮点转换成双精度浮点指令	175
14.5.6	FCVT.D.W	—有符号整型转换成双精度浮点数指令	176
14.5.7	FCVT.D.WU	—无符号整型转换成双精度浮点数指令	176
14.5.8	FCVT.L.D	—双精度浮点转换成有符号长整型指令	177
14.5.9	FCVT.LU.D	—双精度浮点转换成无符号长整型指令	177
14.5.10	FCVT.S.D	—双精度浮点转换成单精度浮点指令	178
14.5.11	FCVT.W.D	—双精度浮点转换成有符号整型指令	179
14.5.12	FCVT.WU.D	—双精度浮点转换成无符号整型指令	180
14.5.13	FDIV.D	—双精度浮点除法指令	181
14.5.14	FEQ.D	—双精度浮点比较相等指令	182
14.5.15	FLD	—双精度浮点加载指令	182
14.5.16	FLE.D	—双精度浮点比较小于等于指令	183
14.5.17	FLT.D	—双精度浮点比较小于指令	183
14.5.18	FMADD.D	—双精度浮点乘累加指令	184
14.5.19	FMAX.D	—双精度浮点取最大值指令	185
14.5.20	FMIN.D	—双精度浮点取最小值指令	185
14.5.21	FMSUB.D	—双精度浮点乘累减指令	186
14.5.22	FMUL.D	—双精度浮点乘法指令	187
14.5.23	FMV.D.X	—双精度浮点写传送指令	188
14.5.24	FMV.X.D	—双精度浮点读传送指令	188
14.5.25	FNMADD.D	—双精度浮点乘累加取负指令	189
14.5.26	FNMSUB.D	—双精度浮点乘累减取负指令	190
14.5.27	FSD	—双精度浮点存储指令	190
14.5.28	FSGNJD	—双精度浮点符号注入指令	191
14.5.29	FSGNJD	—双精度浮点符号取反注入指令	191
14.5.30	FSGNJD	—双精度浮点符号异或注入指令	192
14.5.31	FSQRT.D	—双精度浮点开方指令	193
14.5.32	FSUB.D	—双精度浮点减法指令	193

14.6	附录 A-6 C 指令术语	194
14.6.1	C.ADD——有符号加法指令	194
14.6.2	C.ADDI——有符号立即数加法指令	195
14.6.3	C.ADDIW——低 32 位有符号立即数加法指令	196
14.6.4	C.ADDI4SPN——4 倍立即数和堆栈指针相加指令	196
14.6.5	C.ADDI16SP——加 16 倍立即数到堆栈指针指令	197
14.6.6	C.ADDW——低 32 位有符号加法指令	197
14.6.7	C.AND——按位与指令	198
14.6.8	C.ANDI——立即数按位与指令	199
14.6.9	C.BEQZ——等于零分支指令	200
14.6.10	C.BNEZ——不等于零分支指令	201
14.6.11	C.EBREAK——断点指令	201
14.6.12	C.FLD——浮点双字加载指令	202
14.6.13	C.FLDSP——浮点双字堆栈加载指令	203
14.6.14	C.FSD——浮点双字存储指令	203
14.6.15	C.FSDSP——浮点双字堆栈存储指令	204
14.6.16	C.J——无条件跳转指令	205
14.6.17	C.JALR——寄存器跳转子程序指令	206
14.6.18	C.JR——寄存器跳转指令	206
14.6.19	C.LD——双字加载指令	207
14.6.20	C.LDSP——双字堆栈加载指令	208
14.6.21	C.LI——立即数传送指令	208
14.6.22	C.LUI——高位立即数传送指令	209
14.6.23	C.LW——字加载指令	209
14.6.24	C.LWSP——字堆栈加载指令	210
14.6.25	C.MV——数据传送指令	211
14.6.26	C.NOP——空指令	211
14.6.27	C.OR——按位或指令	212
14.6.28	C.SD——双字存储指令	213
14.6.29	C.SDSP——双字堆栈存储指令	213
14.6.30	C.SLLI——立即数逻辑左移指令	214
14.6.31	C.SRAI——立即数算数右移指令	214
14.6.32	C.SRLI——立即数逻辑右移指令	215
14.6.33	C.SW——字存储指令	216
14.6.34	C.SWSP——字堆栈存储指令	217
14.6.35	C.SUB——有符号减法指令	217
14.6.36	C.SUBW——低 32 位有符号减法指令	218
14.6.37	C.XOR——按位异或指令	219
14.7	附录 A-7 V 指令术语	220
14.7.1	VAADD.VI——矢量立即数整型加法取平均数指令	220
14.7.2	VAADD.VV——矢量整型加法取平均数指令	221
14.7.3	VAADD.VX——矢量标量整型加法取平均数指令	221
14.7.4	VADC.VIM——矢量立即数整型带进位加法指令	222
14.7.5	VADC.VVM——矢量整型带进位加法指令	223
14.7.6	VADC.VXM——矢量标量整型带进位加法指令	223

14.7.7	VADD.VI——矢量立即数整型加法指令	224
14.7.8	VADD.VV——矢量整型加法指令	224
14.7.9	VADD.VX——矢量标量整型加法指令	225
14.7.10	VAND.VI——矢量立即数按位与指令	226
14.7.11	VAND.VV——矢量按位与指令	226
14.7.12	VAND.VX——矢量标量按位与指令	227
14.7.13	VASUB.VV——矢量整型减法取平均数指令	228
14.7.14	VASUB.VX——矢量标量整型减法取平均数指令	228
14.7.15	VCOMPRESS.VM——矢量整型元素压缩指令	229
14.7.16	VDIV.VV——矢量整型有符号除法指令	230
14.7.17	VDIV.VX——矢量标量整型有符号除法指令	230
14.7.18	VDIVU.VV——矢量整型无符号除法指令	231
14.7.19	VDIVU.VX——矢量标量整型无符号除法指令	232
14.7.20	VEXT.X.V——矢量整型提取元素指令	232
14.7.21	VFADD.VF——矢量标量浮点加法指令	233
14.7.22	VFADD.VV——矢量浮点加法指令	234
14.7.23	VFCLASS.V——矢量浮点分类指令	234
14.7.24	VFCVT.F.X.V——矢量有符号整型转换成浮点指令	236
14.7.25	VFCVT.F.XU.V——矢量无符号整型转换成浮点指令	236
14.7.26	VFCVT.X.F.V——矢量浮点转换成有符号整型指令	237
14.7.27	VFCVT.XU.F.V——矢量浮点转换成无符号整型指令	238
14.7.28	VFDIV.VF——矢量标量浮点除法指令	239
14.7.29	VFDIV.VV——矢量浮点除法指令	240
14.7.30	VFMACC.VF——覆盖加数矢量标量浮点乘累加指令	241
14.7.31	VFMACC.VV——覆盖加数矢量浮点乘累加指令	242
14.7.32	VFMAADD.VF——覆盖被乘数矢量标量浮点乘累加指令	242
14.7.33	VFMAADD.VV——覆盖被乘数矢量浮点乘累加指令	243
14.7.34	VFMAX.VF——矢量标量浮点取最大值指令	244
14.7.35	VFMAX.VV——矢量浮点取最大值指令	245
14.7.36	VFMERGE.VFM——矢量浮点元素选择指令	246
14.7.37	VFMIN.VF——矢量标量浮点取最小值指令	246
14.7.38	VFMIN.VV——矢量浮点取最小值指令	247
14.7.39	VFMSAC.VF——覆盖减数矢量标量浮点乘累减指令	248
14.7.40	VFMSAC.VV——覆盖减数矢量浮点乘累减指令	248
14.7.41	VFMSUB.VF——覆盖被乘数矢量标量浮点乘累减指令	249
14.7.42	VFMSUB.VV——覆盖被乘数矢量浮点乘累减指令	250
14.7.43	VFMUL.VF——矢量标量浮点乘法指令	251
14.7.44	VFMUL.VV——矢量浮点乘法指令	252
14.7.45	VFMV.F.S——矢量首元素传送至浮点指令	253
14.7.46	VFMV.S.F——浮点传送至矢量首元素指令	253
14.7.47	VFMV.V.F——浮点传送至矢量指令	254
14.7.48	VFNCVT.F.F.V——矢量浮点缩位转换指令	254
14.7.49	VFNCVT.F.XU.V——矢量无符号整型缩位转换成浮点指令	255
14.7.50	VFNCVT.F.X.V——矢量有符号整型缩位转换成浮点指令	256
14.7.51	VFNCVT.X.F.V——矢量浮点缩位转换成有符号整型指令	257

14.7.52	VFNCVT.XU.F.V	— 矢量浮点缩位转换成无符号整型指令	258
14.7.53	VFNMACC.VF	— 覆盖减数矢量标量浮点乘累加取负指令	258
14.7.54	VFNMACC.VV	— 覆盖减数矢量浮点乘累加取负指令	259
14.7.55	VFNMADD.VF	— 覆盖被乘数矢量标量浮点乘累加取负指令	260
14.7.56	VFNMADD.VV	— 覆盖被乘数矢量浮点乘累加取负指令	261
14.7.57	VFNMSAC.VF	— 覆盖被减数矢量标量浮点乘累减取负指令	262
14.7.58	VFNMSAC.VV	— 覆盖被减数矢量浮点乘累减取负指令	263
14.7.59	VFNMSUB.VF	— 覆盖被乘数矢量标量浮点乘累减取负指令	264
14.7.60	VFNMSUB.VV	— 覆盖被乘数矢量浮点乘累减取负指令	264
14.7.61	VFRDIV.VF	— 标量矢量浮点除法指令	265
14.7.62	VFREDMAX.VS	— 矢量浮点元素缩减取最大值指令	266
14.7.63	VFREDMIN.VS	— 矢量浮点缩减取最小值指令	267
14.7.64	VFREDOSUM.VS	— 矢量浮点缩减顺序求和指令	268
14.7.65	VFREDSUM.VS	— 矢量浮点缩减乱序求和指令	268
14.7.66	VFRSUB.VF	— 标量矢量浮点减法指令	269
14.7.67	VFSGNJ.VF	— 矢量标量浮点符号注入指令	270
14.7.68	VFSGNJ.VV	— 矢量浮点符号注入指令	270
14.7.69	VFSGNJN.VF	— 矢量标量浮点符号取反注入指令	271
14.7.70	VFSGNJN.VV	— 矢量浮点符号取反注入指令	272
14.7.71	VFSGNJX.VF	— 矢量标量浮点符号异或注入指令	272
14.7.72	VFSGNJX.VV	— 矢量浮点符号异或注入指令	273
14.7.73	VFSQRT.V	— 矢量浮点开方指令	274
14.7.74	VFSUB.VF	— 矢量标量浮点减法指令	274
14.7.75	VFSUB.VV	— 矢量浮点减法指令	275
14.7.76	VFWADD.VF	— 矢量标量浮点扩位加法指令	276
14.7.77	VFWADD.VV	— 矢量浮点扩位加法指令	277
14.7.78	VFWADD.WF	— 扩位矢量标量浮点扩位加法指令	278
14.7.79	VFWADD.WV	— 扩位矢量浮点扩位加法指令	279
14.7.80	VFWCVT.F.F.V	— 矢量浮点扩位转换指令	280
14.7.81	VFWCVT.F.X.V	— 矢量有符号整型扩位转换成浮点指令	280
14.7.82	VFWCVT.F.XU.V	— 矢量无符号整型扩位转换成浮点指令	281
14.7.83	VFWCVT.X.F.V	— 矢量浮点扩位转换成有符号整型指令	282
14.7.84	VFWCVT.XU.F.V	— 矢量浮点扩位转换成无符号整型指令	283
14.7.85	VFWMACC.VF	— 覆盖加数矢量标量浮点扩位乘累加指令	284
14.7.86	VFWMACC.VV	— 覆盖加数矢量浮点扩位乘累加指令	285
14.7.87	VFWMSAC.VF	— 覆盖加数矢量浮点扩位乘累减指令	285
14.7.88	VFWMSAC.VV	— 覆盖加数矢量浮点扩位乘累减指令	286
14.7.89	VFWMUL.VF	— 矢量标量扩位浮点乘法指令	287
14.7.90	VFWMUL.VV	— 矢量浮点扩位乘法指令	288
14.7.91	VFWNMACC.VF	— 覆盖加数矢量标量浮点扩位乘累加取负指令	289
14.7.92	VFWNMACC.VV	— 覆盖加数矢量浮点扩位乘累加取负指令	290
14.7.93	VFWNMSAC.VF	— 覆盖加数矢量标量浮点扩位乘累减取负指令	291
14.7.94	VFWNMSAC.VV	— 覆盖加数矢量浮点扩位乘累减取负指令	291
14.7.95	VFWREDOSUM.VS	— 矢量扩位浮点缩减顺序求和指令	292
14.7.96	VFWREDSUM.VS	— 矢量扩位浮点缩减乱序求和指令	293

14.7.97 VFWSUB.VF——矢量标量浮点扩位减法指令	294
14.7.98 VFWSUB.VV——矢量浮点扩位减法指令	295
14.7.99 VFWSUB.WF——扩位矢量标量浮点扩位减法指令	295
14.7.100 VFWSUB.WV——扩位矢量浮点扩位减法指令	296
14.7.101 VID.V——矢量 index 求解指令	297
14.7.102 VIOTA.M——矢量 active 元素统计指令	298
14.7.103 VLB.V——矢量有符号字节加载指令	299
14.7.104 VLBBF.V——矢量 FOF 有符号字节加载指令	299
14.7.105 VLBU.V——矢量无符号字节加载指令	300
14.7.106 VLBUFF.V——矢量 FOF 无符号字节加载指令	301
14.7.107 VLE.V——矢量元素加载指令	301
14.7.108 VLEFF.V——矢量 FOF 无符号元素加载指令	302
14.7.109 VLH.V——矢量有符号半字加载指令	303
14.7.110 VLHFF.V——矢量 FOF 有符号半字加载指令	303
14.7.111 VLHU.V——矢量无符号半字加载指令	304
14.7.112 VLHUFF.V——矢量 FOF 无符号半字加载指令	305
14.7.113 VLSB.V——矢量步进有符号字节加载指令	305
14.7.114 VLSBU.V——矢量步进无符号字节加载指令	306
14.7.115 VLSE.V——矢量步进元素加载指令	307
14.7.116 VLSEG<NF>B.V——矢量 SEGMENT 有符号字节加载指令	307
14.7.117 VLSEG<NF>BBF.V——矢量 FOF SEGMENT 有符号字节加载指令	308
14.7.118 VLSEG<NF>BU.V——矢量 SEGMENT 无符号字节加载指令	309
14.7.119 VLSEG<NF>BUFF.V——矢量 FOF SEGMENT 无符号字节加载指令	310
14.7.120 VLSEG<NF>E.V——矢量 SEGMENT 元素加载指令	311
14.7.121 VLSEG<NF>EFF.V——矢量 FOF SEGMENT 元素加载指令	311
14.7.122 VLSEG<NF>H.V——矢量 SEGMENT 有符号半字加载指令	312
14.7.123 VLSEG<NF>HFF.V——矢量 FOF SEGMENT 有符号半字加载指令	313
14.7.124 VLSEG<NF>HU.V——矢量 SEGMENT 无符号半字加载指令	314
14.7.125 VLSEG<NF>HUFF.V——矢量 FOF SEGMENT 无符号半字加载指令	315
14.7.126 VLSEG<NF>W.V——矢量 SEGMENT 有符号字加载指令	316
14.7.127 VLSEG<NF>WFF.V——矢量 FOF SEGMENT 有符号字加载指令	317
14.7.128 VLSEG<NF>WU.V——矢量 SEGMENT 无符号字加载指令	317
14.7.129 VLSEG<NF>WUFF.V——矢量 FOF SEGMENT 无符号字加载指令	318
14.7.130 VLSSEG<NF>B.V——矢量步进 SEGMENT 有符号字节加载指令	319
14.7.131 VLSSEG<NF>BU.V——矢量步进 SEGMENT 无符号字节加载指令	320
14.7.132 VLSSEG<NF>E.V——矢量步进 SEGMENT 元素加载指令	321
14.7.133 VLSSEG<NF>H.V——矢量步进 SEGMENT 有符号半字加载指令	322
14.7.134 VLSSEG<NF>HU.V——矢量步进 SEGMENT 无符号半字加载指令	323
14.7.135 VLSSEG<NF>W.V——矢量步进 SEGMENT 有符号字加载指令	323
14.7.136 VLSSEG<NF>WU.V——矢量步进 SEGMENT 无符号字加载指令	324
14.7.137 VLSH.V——矢量步进有符号半字加载指令	325
14.7.138 VLSHU.V——矢量步进无符号半字加载指令	326
14.7.139 VLSW——矢量步进有符号字加载指令	326
14.7.140 VLSWU.V——矢量步进无符号字加载指令	327
14.7.141 VLW.V——矢量有符号字加载指令	328

14.7.142	VLWFF.V——矢量 FOF 有符号字加载指令	328
14.7.143	VLWU.V——矢量无符号字加载指令	329
14.7.144	VLWUFF.V——矢量 FOF 无符号字加载指令	330
14.7.145	VLXB.V——矢量索引有符号字节加载指令	330
14.7.146	VLXBU.V——矢量索引无符号字节加载指令	331
14.7.147	VLXE.V——矢量索引元素加载指令	332
14.7.148	VLXH.V——矢量索引有符号半字加载指令	332
14.7.149	VLXHU.V——矢量索引无符号半字加载指令	333
14.7.150	VLXSEG<NF>B.V——矢量索引 SEGMENT 有符号字节加载指令	334
14.7.151	VLXSEG<NF>BU.V——矢量索引 SEGMENT 无符号字节加载指令	334
14.7.152	VLXSEG<NF>E.V——矢量索引 SEGMENT 元素加载指令	335
14.7.153	VLXSEG<NF>H.V——矢量索引 SEGMENT 有符号半字加载指令	336
14.7.154	VLXSEG<NF>HU.V——矢量索引 SEGMENT 无符号半字加载指令	337
14.7.155	VLXSEG<NF>W.V——矢量索引 SEGMENT 有符号字加载指令	338
14.7.156	VLXSEG<NF>WU.V——矢量索引 SEGMENT 无符号字加载指令	339
14.7.157	VLXW.V——矢量索引有符号字加载指令	340
14.7.158	VLXWU.V——矢量索引无符号字加载指令	340
14.7.159	VMACC.VV——覆盖加数矢量低位乘累加指令	341
14.7.160	VMACC.VX——覆盖加数矢量标量低位乘累加指令	342
14.7.161	VMADC.VVM——矢量整型带进位加法取进位指令	342
14.7.162	VMADC.VXM——矢量标量整型带进位加法取进位指令	343
14.7.163	VMADC.VIM——矢量立即数整型带进位加法取进位指令	343
14.7.164	VMADD.VV——覆盖被乘数矢量低位乘累加指令	344
14.7.165	VMADD.VX——覆盖被乘数矢量标量低位乘累加指令	344
14.7.166	VMAND.MM——矢量掩码运算与指令	345
14.7.167	VMANDNOT.MM——矢量掩码运算非与指令	346
14.7.168	VMAX.VV——矢量整型有符号取最大值指令	346
14.7.169	VMAX.VX——矢量标量整型有符号取最大值指令	347
14.7.170	VMAXU.VV——矢量整型无符号取最大值指令	347
14.7.171	VMAXU.VX——矢量标量整型无符号取最大值指令	348
14.7.172	VMERGE.VVM——矢量元素选择指令	349
14.7.173	VMERGE.VXM——矢量标量元素选择指令	349
14.7.174	VMERGE.VIM——矢量立即数元素选择指令	350
14.7.175	VMFEQ.VF——矢量标量浮点比较相等指令	350
14.7.176	VMFEQ.VV——矢量浮点比较相等指令	351
14.7.177	VMFGE.VF——矢量标量浮点比较大于等于指令	352
14.7.178	VMFGT.VF——矢量标量浮点比较大于指令	353
14.7.179	VMFIRST——矢量掩码运算快速查找 1 指令	353
14.7.180	VMFLE.VF——矢量标量浮点比较小于等于指令	354
14.7.181	VMFLE.VV——矢量浮点比较小于等于指令	355
14.7.182	VMFLT.VF——矢量标量浮点比较小于指令	355
14.7.183	VMFLT.VV——矢量浮点比较小于指令	356
14.7.184	VMFNE.VF——矢量标量浮点比较不等指令	357
14.7.185	VMFNE.VV——矢量浮点比较不等指令	358
14.7.186	VMFORD.VF——矢量标量浮点 NaN 判断指令	358

14.7.187 VMFORD.VV——矢量浮点 NaN 判断指令	359
14.7.188 VMIN.VV——矢量整型有符号取最小值指令	360
14.7.189 VMIN.VX——矢量标量整型有符号取最小值指令	360
14.7.190 VMINU.VV——矢量整型无符号取最小值指令	361
14.7.191 VMINU.VX——矢量标量整型无符号取最小值指令	362
14.7.192 VMNAND.MM——矢量掩码运算与非指令	362
14.7.193 VMNOR.MM——矢量掩码运算或非指令	363
14.7.194 VMOR.MM——矢量掩码运算或指令	363
14.7.195 VMORNOT.MM——矢量掩码运算非或指令	364
14.7.196 VMPOPC.M——矢量运算元素个数统计指令	365
14.7.197 VMSBC.VVM——矢量整型带进位减法取借位指令	365
14.7.198 VMSBC.VXM——矢量标量整型带进位减法取借位指令	366
14.7.199 VMSBF.M——矢量掩码找 1 置位不包含指令	366
14.7.200 VMSIF.M——矢量掩码找 1 置位包含指令	367
14.7.201 VMSOF.M——矢量掩码找 1 单独置位指令	368
14.7.202 VMSEQ.VV——矢量整型比较相等指令	369
14.7.203 VMSEQ.VX——矢量标量整型比较相等指令	369
14.7.204 VMSEQ.VI——矢量立即数整型比较相等指令	370
14.7.205 VMSGT.VX——矢量标量整型有符号比较大于指令	371
14.7.206 VMSGT.VI——矢量立即数整型有符号比较大于指令	371
14.7.207 VMSGTU.VX——矢量标量整型无符号比较大于指令	372
14.7.208 VMSGTU.VI——矢量立即数整型无符号比较大于指令	373
14.7.209 VMSLE.VV——矢量整型有符号比较小于等于指令	374
14.7.210 VMSLE.VX——矢量标量整型有符号比较小于等于指令	374
14.7.211 VMSLE.VI——矢量立即数整型有符号比较小于等于指令	375
14.7.212 VMSLEU.VV——矢量整型无符号比较小于等于指令	376
14.7.213 VMSLEU.VX——矢量标量整型无符号比较小于等于指令	376
14.7.214 VMSLEU.VI——矢量立即数整型无符号比较小于等于指令	377
14.7.215 VMSLT.VV——矢量整型有符号比较小于指令	378
14.7.216 VMSLT.VX——矢量标量整型有符号比较小于指令	379
14.7.217 VMSLTU.VV——矢量整型有无符号比较小于指令	379
14.7.218 VMSLTU.VX——矢量标量整型无符号比较小于指令	380
14.7.219 VMSNE.VV——矢量整型比较不等指令	381
14.7.220 VMSNE.VX——矢量标量整型比较不等指令	381
14.7.221 VMSNE.VI——矢量立即数整型比较不等指令	382
14.7.222 VMUL.VV——矢量整型乘法取低位指令	383
14.7.223 VMUL.VX——矢量标量整型乘法取低位指令	383
14.7.224 VMULH.VV——矢量整型有符号乘法取高位指令	384
14.7.225 VMULH.VX——矢量标量有符号乘法取高位指令	385
14.7.226 VMULHU.VV——矢量整型无符号乘法取高位指令	385
14.7.227 VMULHU.VX——矢量标量整型无符号乘法取高位指令	386
14.7.228 VMULHSU.VV——矢量整型有符号乘无符号取高位指令	387
14.7.229 VMULHSU.VX——矢量标量整型有符号乘无符号取高位指令	387
14.7.230 VMV.V.V——矢量元素传送指令	388
14.7.231 VMV.V.X——整型传送至矢量指令	388

14.7.232 VMV.V.I——立即数传送至矢量指令	389
14.7.233 VMV.S.X——整型传送至矢量首元素指令	389
14.7.234 VMXOR.MM——矢量掩码运算异或指令	390
14.7.235 VMXNOR.MM——矢量掩码运算同或指令	391
14.7.236 VNCLIP.VV——矢量带饱和有符号缩位算术右移指令	391
14.7.237 VNCLIP.VX——矢量带饱和有符号按标量缩位算术右移指令	392
14.7.238 VNCLIP.VI——矢量带饱和有符号按立即数缩位算术右移指令	393
14.7.239 VNCLIPU.VV——矢量带饱和无符号缩位算术右移指令	393
14.7.240 VNCLIPU.VX——矢量带饱和无符号缩位算术右移指令	394
14.7.241 VNCLIPU.VI——矢量带饱和按立即数无符号缩位算术右移指令	395
14.7.242 VNMSAC.VV——覆盖被减数矢量低位乘累减取负指令	396
14.7.243 VNMSAC.VX——覆盖被减数矢量标量低位乘累减取负指令	396
14.7.244 VNMSUB.VV——覆盖被乘数矢量低位乘累减取负指令	397
14.7.245 VNMSUB.VX——覆盖被乘数矢量标量低位乘累减取负指令	397
14.7.246 VNSRA.VV——矢量缩位算术右移指令	398
14.7.247 VNSRA.VX——矢量标量缩位算术右移指令	399
14.7.248 VNSRA.VI——矢量立即数缩位算术右移指令	399
14.7.249 VNSRL.VV——矢量缩位逻辑右移指令	400
14.7.250 VNSRL.VX——矢量标量缩位逻辑右移指令	401
14.7.251 VNSRL.VI——矢量立即数缩位逻辑右移指令	401
14.7.252 VOR.VV——矢量按位或指令	402
14.7.253 VOR.VX——矢量标量按位或指令	403
14.7.254 VOR.VI——矢量立即数按位或指令	403
14.7.255 VREDAND.VS——矢量缩减按位与指令	404
14.7.256 VREDMAX.VS——矢量缩减有符号最大值指令	405
14.7.257 VREDMAXU.VS——矢量缩减无符号最大值指令	405
14.7.258 VREDMIN.VS——矢量缩减有符号最小值指令	406
14.7.259 VREDMINU.VS——矢量缩减无符号最小值指令	407
14.7.260 VREDOR.VS——矢量缩减按位或指令	408
14.7.261 VREDXOR.VS——矢量缩减按位异或指令	408
14.7.262 VREDSUM.VS——矢量缩减累加指令	409
14.7.263 VREM.VV——矢量整型有符号取余指令	410
14.7.264 VREM.VX——矢量标量整型有符号取余指令	411
14.7.265 VREMU.VV——矢量整型无符号取余指令	411
14.7.266 VREMU.VX——矢量标量整型无符号取余指令	412
14.7.267 VRGATHER.VV——矢量整型元素根据索引聚合指令	413
14.7.268 VRGATHER.VX——矢量标量整型元素根据索引聚合指令	413
14.7.269 VRGATHER.VI——矢量立即数整型元素根据索引聚合指令	414
14.7.270 VRSUB.VX——标量矢量整型减法指令	415
14.7.271 VRSUB.VI——立即数矢量整型减法指令	415
14.7.272 VSADD.VV——矢量整型加法有符号取饱和指令	416
14.7.273 VSADD.VX——矢量标量整型加法有符号取饱和指令	417
14.7.274 VSADD.VI——矢量立即数整型加法有符号取饱和指令	417
14.7.275 VSADDU.VV——矢量整型加法无符号取饱和指令	418
14.7.276 VSADDU.VX——矢量标量整型加法无符号取饱和指令	419

14.7.277 VSADDDU.VI——矢量立即数整型加法无符号取饱和指令	419
14.7.278 VSB.V——矢量字节存储指令	420
14.7.279 VSBC.VVM——矢量整型带进位减法指令	421
14.7.280 VSBC.VXM——矢量标量整型带进位减法指令	421
14.7.281 VSE.V——矢量元素存储指令	422
14.7.282 VSETVL——寄存器设置 vl 和 vtype 指令	422
14.7.283 VSETVLI——立即数设置 vl 和 vtype 指令	423
14.7.284 VSH.V——矢量半字存储指令	424
14.7.285 VSLIDEDOWN.VX——矢量元素向下偏移指令	424
14.7.286 VSLIDEDOWN.VI——矢量元素按立即数向下偏移指令	425
14.7.287 VSLIDE1DOWN.VX——矢量元素向下偏移 1 个索引指令	426
14.7.288 VSLIDEUP.VX——矢量元素向上偏移指令	426
14.7.289 VSLIDEUP.VI——矢量元素按立即数向上偏移指令	427
14.7.290 VSLIDE1UP.VX——矢量元素向上偏移 1 个索引指令	428
14.7.291 VSLL.VV——矢量逻辑左移指令	428
14.7.292 VSLL.VX——矢量按标量逻辑左移指令	429
14.7.293 VSLL.VI——矢量按立即数逻辑左移指令	430
14.7.294 VSMUL.VV——矢量带饱和乘法指令	430
14.7.295 VSMUL.VX——矢量标量带饱和乘法指令	431
14.7.296 VSRA.VV——矢量算术右移指令	432
14.7.297 VSRA.VX——矢量按标量算术右移指令	432
14.7.298 VSRA.VI——矢量按立即数算术右移指令	433
14.7.299 VSRL.VV——矢量逻辑右移指令	434
14.7.300 VSRL.VX——矢量按标量逻辑右移指令	434
14.7.301 VSRL.VI——矢量按立即数逻辑右移指令	435
14.7.302 VSSB.V——矢量步进字节存储指令	436
14.7.303 VSSE.V——矢量步进元素存储指令	436
14.7.304 VSSEG<NF>B.V——矢量 SEGMENT 字节存储指令	437
14.7.305 VSSEG<NF>E.V——矢量 SEGMENT 元素存储指令	438
14.7.306 VSSEG<NF>H.V——矢量 SEGMENT 半字存储指令	438
14.7.307 VSSEG<NF>W.V——矢量 SEGMENT 字存储指令	439
14.7.308 VSSSEG<NF>B.V——矢量步进 SEGMENT 字节存储指令	440
14.7.309 VSSSEG<NF>E.V——矢量步进 SEGMENT 元素存储指令	441
14.7.310 VSSSEG<NF>H.V——矢量步进 SEGMENT 半字存储指令	442
14.7.311 VSSSEG<NF>W.V——矢量步进 SEGMENT 字存储指令	443
14.7.312 VSXSEG<NF>B.V——矢量索引 SEGMENT 字节存储指令	443
14.7.313 VSXSEG<NF>E.V——矢量索引 SEGMENT 元素存储指令	444
14.7.314 VSXSEG<NF>H.V——矢量索引 SEGMENT 半字存储指令	445
14.7.315 VSXSEG<NF>W.V——矢量索引 SEGMENT 字存储指令	446
14.7.316 VSSH.V——矢量步进半字存储指令	447
14.7.317 VSSRA.VV——矢量带近似算术右移指令	447
14.7.318 VSSRA.VX——矢量标量带近似算术右移指令	448
14.7.319 VSSRA.VI——矢量立即数带近似算术右移指令	449
14.7.320 VSSRL.VV——矢量带近似逻辑右移指令	450
14.7.321 VSSRL.VX——矢量标量带近似逻辑右移指令	450

14.7.322	VSSRL.VI——矢量立即数带近似逻辑右移指令	451
14.7.323	VSSUB.VV——矢量整型减法有符号取饱和指令	452
14.7.324	VSSUB.VX——矢量标量整型减法有符号取饱和指令	453
14.7.325	VSSUBU.VV——矢量整型减法无符号取饱和指令	453
14.7.326	VSSUBU.VX——矢量标量整型减法无符号取饱和指令	454
14.7.327	VSSW.V——矢量步进字存储指令	455
14.7.328	VSUB.VV——矢量整型减法指令	455
14.7.329	VSUB.VX——矢量标量整型减法指令	456
14.7.330	VSUXB.V——矢量索引乱序字节存储指令	457
14.7.331	VSUXE.V——矢量索引乱序元素存储指令	457
14.7.332	VSUXH.V——矢量索引乱序半字存储指令	458
14.7.333	VSUXW.V——矢量索引乱序字存储指令	459
14.7.334	VSW.V——矢量字存储指令	459
14.7.335	VSXB.V——矢量索引按序字节存储指令	460
14.7.336	VSXE.V——矢量索引按序元素存储指令	461
14.7.337	VSXH.V——矢量索引按序半字存储指令	461
14.7.338	VSXW.V——矢量索引按序字存储指令	462
14.7.339	VWADD.VV——矢量整型有符号扩位加法指令	463
14.7.340	VWADD.VX——矢量标量整型有符号扩位加法指令	463
14.7.341	VWADDU.VV——矢量整型无符号扩位加法指令	464
14.7.342	VWADDU.VX——矢量标量整型无符号扩位加法指令	465
14.7.343	VWADD.WV——扩位矢量整型有符号扩位加法指令	465
14.7.344	VWADD.WX——扩位矢量标量整型有符号扩位加法指令	466
14.7.345	VWADDU.WV——扩位矢量整型无符号扩位加法指令	467
14.7.346	VWADDU.WX——扩位矢量标量整型无符号扩位加法指令	467
14.7.347	VWMACC.VV——覆盖加数矢量有符号扩位乘累加指令	468
14.7.348	VWMACC.VX——覆盖加数矢量标量有符号扩位乘累加指令	469
14.7.349	VWMACCSU.VV——覆盖加数矢量有符号无符号扩位乘累加指令	469
14.7.350	VWMACCSU.VX——覆盖加数矢量标量有符号无符号扩位乘累加指令	470
14.7.351	VWMACCU.VV——覆盖加数矢量无符号扩位乘累加指令	471
14.7.352	VWMACCU.VX——覆盖加数矢量标量无符号扩位乘累加指令	471
14.7.353	VWMACCUS.VX——覆盖加数矢量标量无符号有符号扩位乘累加指令	472
14.7.354	VWMUL.VV——矢量有符号扩位乘法指令	473
14.7.355	VWMUL.VX——矢量标量有符号扩位乘法指令	473
14.7.356	VWMULSU.VV——矢量有符号无符号扩位乘法指令	474
14.7.357	VWMULSU.VX——矢量标量有符号无符号扩位乘法指令	475
14.7.358	VWMULU.VV——矢量无符号扩位乘法指令	475
14.7.359	VWMULU.VX——矢量标量无符号扩位乘法指令	476
14.7.360	VWREDSUM.VS——矢量缩减有符号扩位累加指令	477
14.7.361	VWREDSUMU.VS——矢量缩减无符号扩位累加指令	477
14.7.362	VWSMACC.VV——矢量有符号带饱和扩位乘累加指令	478
14.7.363	VWSMACC.VX——矢量标量有符号带饱和扩位乘累加指令	479
14.7.364	VWSMACCSU.VV——矢量带饱和和取负有符号无符号扩位乘累减取负指令	480
14.7.365	VWSMACCSU.VX——矢量标量带饱和和取负有符号无符号扩位乘累减取负指令	480
14.7.366	VWSMACCU.VV——矢量无符号带饱和扩位乘累加指令	481

14.7.367	VWSMACCU.VX——矢量标量无符号带饱和和扩位乘累加指令	482
14.7.368	VWSMACCUS.VX——矢量标量带饱和和取负有符号无符号扩位乘累减指令	483
14.7.369	VWSUB.VV——矢量整型有符号扩位减法指令	484
14.7.370	VWSUB.VX——矢量标量整型有符号扩位减法指令	484
14.7.371	VWSUB.WV——扩位矢量整型有符号扩位减法指令	485
14.7.372	VWSUB.WX——扩位矢量标量整型有符号扩位减法指令	486
14.7.373	VWSUBU.VV——矢量整型无符号扩位减法指令	486
14.7.374	VWSUBU.VX——矢量标量整型无符号扩位减法指令	487
14.7.375	VWSUBU.WV——扩位矢量整型无符号扩位减法指令	488
14.7.376	VWSUBU.WX——扩位矢量标量整型无符号扩位减法指令	488
14.7.377	VXOR.VV——矢量按位异或指令	489
14.7.378	VXOR.VX——矢量标量按位异或指令	490
14.7.379	VXOR.VI——矢量立即数按位异或指令	490
14.8	附录 A-8 伪指令列表	491
第十五章 附录 B 平头哥扩展指令术语		494
15.1	附录 B-1 Cache 指令术语	494
15.1.1	DCACHE.CALL——DCACHE 清全部脏表项指令	494
15.1.2	DCACHE.CVA——DCACHE 指定虚拟地址清脏表项指令	495
15.1.3	DCACHE.CPA——DCACHE 指定物理地址清脏表项指令	495
15.1.4	DCACHE.CSW——DCACHE 指定 way/set 清脏表项并无效指令	496
15.1.5	DCACHE.IALL——DCACHE 无效全部表项指令	496
15.1.6	DCACHE.IVA——DCACHE 指定虚拟地址无效表项指令	497
15.1.7	DCACHE.IPA——DCACHE 指定物理地址无效表项指令	498
15.1.8	DCACHE.ISW——DCACHE 指定 set/way 无效表项指令	498
15.1.9	DCACHE.CIALL——DCACHE 清全部脏表项并无效指令	499
15.1.10	DCACHE.CIVA——DCACHE 指定虚拟地址清脏表项并无效指令	499
15.1.11	DCACHE.CIPA——DCACHE 指定物理地址清脏表项并无效指令	500
15.1.12	DCACHE.CISW——DCACHE 指定 way/set 清脏表项并无效指令	501
15.1.13	ICACHE.IALL——ICACHE 无效全部表项指令	501
15.1.14	ICACHE.IALLS——ICACHE 广播无效全部表项指令	502
15.1.15	ICACHE.IVA——ICACHE 指定虚拟地址无效表项指令	502
15.1.16	ICACHE.IPA——ICACHE 指定物理地址无效表项指令	503
15.2	附录 B-2 同步指令术语	503
15.2.1	SYNC——同步指令	504
15.2.2	SYNC.I——同步并清空指令	504
15.3	附录 B-3 算术运算指令术语	505
15.3.1	ADDSL——寄存器移位相加指令	505
15.3.2	MULA——乘累加指令	505
15.3.3	MULAH——低 16 位乘累加指令	506
15.3.4	MULAW——低 32 位乘累加指令	506
15.3.5	MULS——乘累减指令	507
15.3.6	MULSH——低 16 位乘累减指令	507
15.3.7	MULSW——低 32 位乘累减指令	508
15.3.8	MVEQZ——寄存器为 0 传送指令	508
15.3.9	MVNEZ——寄存器非 0 传送指令	509

15.3.10	SRRI——循环右移指令	509
15.3.11	SRRIW——低 32 位循环右移指令	510
15.4	附录 B-3 位操作指令术语	510
15.4.1	EXT——寄存器连续位提取符号位扩展指令	510
15.4.2	EXTU——寄存器连续位提取零扩展指令	511
15.4.3	FF0——快速找 0 指令	511
15.4.4	FF1——快速找 1 指令	512
15.4.5	REV——字节倒序指令	512
15.4.6	REW——低 32 位字节倒序指令	513
15.4.7	TST——比特为 0 测试指令	513
15.4.8	TSTNBZ——字节为 0 测试指令	514
15.5	附录 B-4 存储指令术语	514
15.5.1	FLRD——浮点寄存器移位双字加载指令	515
15.5.2	FLRW——浮点寄存器移位字加载指令	515
15.5.3	FLURD——浮点寄存器低 32 位移位双字加载指令	516
15.5.4	FLURW——浮点寄存器低 32 位移位字加载指令	516
15.5.5	FSRD——浮点寄存器移位双字存储指令	517
15.5.6	FSRW——浮点寄存器移位字存储指令	517
15.5.7	FSURD——浮点寄存器低 32 位移位双字存储指令	518
15.5.8	FSURW——浮点寄存器低 32 位移位字存储指令	518
15.5.9	LBIA——符号位扩展字节加载基地址自增指令	519
15.5.10	LBIB——基地址自增符号位扩展字节加载指令	519
15.5.11	LBUA——零扩展字节加载基地址自增指令	520
15.5.12	LBUIB——基地址自增零扩展字节加载指令	520
15.5.13	LDD——双寄存器加载指令	521
15.5.14	LDIA——符号位扩展双字加载基地址自增指令	522
15.5.15	LDIB——基地址自增符号位扩展双字加载指令	522
15.5.16	LHIA——符号位扩展半字加载基地址自增指令	523
15.5.17	LHIB——基地址自增符号位扩展半字加载指令	523
15.5.18	LHUIA——零扩展半字加载基地址自增指令	524
15.5.19	LHUIB——基地址自增零扩展半字加载指令	524
15.5.20	LRB——寄存器移位符号位扩展字节加载指令	525
15.5.21	LRBU——寄存器移位零扩展扩展字节加载指令	525
15.5.22	LRD——寄存器移位双字加载指令	526
15.5.23	LRH——寄存器移位符号位扩展半字加载指令	526
15.5.24	LRHU——寄存器移位零扩展扩展半字加载指令	527
15.5.25	LRW——寄存器移位符号位扩展字加载指令	527
15.5.26	LRWU——寄存器移位零扩展扩展字加载指令	528
15.5.27	LURB——寄存器低 32 位移位符号位扩展字节加载指令	528
15.5.28	LURBU——寄存器低 32 位移位零扩展字节加载指令	529
15.5.29	LURD——寄存器低 32 位移位双字加载指令	529
15.5.30	LURH——寄存器低 32 位移位符号位扩展半字加载指令	530
15.5.31	LURHU——寄存器低 32 位移位零扩展半字加载指令	530
15.5.32	LURW——寄存器低 32 位移位符号位扩展字加载指令	531
15.5.33	LURWU——寄存器低 32 位移位零扩展字加载指令	531

15.5.34	LWD——符号位扩展双寄存器字加载指令	532
15.5.35	LWIA——符号位扩展字加载基地址自增指令	532
15.5.36	LWIB——基地址自增符号位扩展字加载指令	533
15.5.37	LWUD——零扩展双寄存器字加载指令	533
15.5.38	LWUIA——零扩展字加载基地址自增指令	534
15.5.39	LWUIB——基地址自增零扩展字加载指令	535
15.5.40	SBIA——字节存储基地址自增指令	535
15.5.41	SBIB——基地址自增字节存储指令	536
15.5.42	SDD——双寄存器存储指令	536
15.5.43	SDIA——双字存储基地址自增指令	537
15.5.44	SDIB——基地址自增双字存储指令	537
15.5.45	SHIA——半字存储基地址自增指令	538
15.5.46	SHIB——基地址自增半字存储指令	538
15.5.47	SRB——寄存器移位字节存储指令	539
15.5.48	SRD——寄存器移位双字存储指令	539
15.5.49	SRH——寄存器移位半字存储指令	540
15.5.50	SRW——寄存器移位字存储指令	540
15.5.51	SURB——寄存器低 32 位移位字节存储指令	541
15.5.52	SURD——寄存器低 32 位移位双字存储指令	541
15.5.53	SURH——寄存器低 32 位移位半字存储指令	542
15.5.54	SURW——寄存器低 32 位移位字存储指令	542
15.5.55	SWIA——字存储基地址自增指令	543
15.5.56	SWIB——基地址自增字存储指令	543
15.5.57	SWD——双寄存器低 32 位存储指令	544
15.6	附录 B-5 半精度浮点指令术语	544
15.6.1	FADD.H——半精度浮点加法指令	544
15.6.2	FCLASS.H——半精度浮点分类指令	545
15.6.3	FCVT.D.H——半精度浮点转换成双精度浮点指令	546
15.6.4	FCVT.H.D——双精度浮点转换成半精度浮点指令	547
15.6.5	FCVT.H.L——有符号长整型转换成半精度浮点数指令	547
15.6.6	FCVT.H.LU——无符号长整型转换成半精度浮点数指令	548
15.6.7	FCVT.H.S——单精度浮点转换成半精度浮点指令	549
15.6.8	FCVT.H.W——有符号整型转换成半精度浮点数指令	550
15.6.9	FCVT.H.WU——无符号整型转换成半精度浮点数指令	551
15.6.10	FCVT.L.H——半精度浮点转换成有符号长整型指令	552
15.6.11	FCVT.LU.H——半精度浮点转换成无符号长整型指令	552
15.6.12	FCVT.S.H——半精度浮点转换成单精度浮点指令	553
15.6.13	FCVT.W.H——半精度浮点转换成有符号整型指令	554
15.6.14	FCVT.WU.H——半精度浮点转换成无符号整型指令	555
15.6.15	FDIV.H——半精度浮点除法指令	555
15.6.16	FEQ.H——半精度浮点比较相等指令	556
15.6.17	FLE.H——半精度浮点比较小于等于指令	557
15.6.18	FLH——半精度浮点加载指令	558
15.6.19	FLT.H——半精度浮点比较小于指令	558
15.6.20	FMADD.H——半精度浮点乘累加指令	559

15.6.21	FMAX.H——半精度浮点取最大值指令	560
15.6.22	FMIN.H——半精度浮点取最小值指令	560
15.6.23	FMSUB.H——半精度浮点乘累减指令	561
15.6.24	FMUL.H——半精度浮点乘法指令	562
15.6.25	FMV.H.X——半精度浮点写传送指令	562
15.6.26	FMV.X.H——半精度浮点寄存器读传送指令	563
15.6.27	FNMADD.H——半精度浮点乘累加取负指令	564
15.6.28	FNMSUB.H——半精度浮点乘累减取负指令	564
15.6.29	FSGNJ.H——半精度浮点符号注入指令	565
15.6.30	FSGNJN.H——半精度浮点符号取反注入指令	566
15.6.31	FSGNJX.H——半精度浮点符号异或注入指令	566
15.6.32	FSH——半精度浮点存储指令	567
15.6.33	FSQRT.H——半精度浮点开方指令	567
15.6.34	FSUB.H——半精度浮点减法指令	568
第十六章 附录 C 控制寄存器		570
16.1	附录 C-1 机器模式控制寄存器	570
16.1.1	机器模式信息寄存器组	570
16.1.1.1	机器模式供应商编号寄存器 (MVENDORID)	570
16.1.1.2	机器模式架构编号寄存器 (MARCHID)	570
16.1.1.3	机器模式硬件实现编号寄存器 (MIMPID)	570
16.1.1.4	机器模式逻辑内核编号寄存器 (MHARTID)	571
16.1.2	机器模式异常配置寄存器组	571
16.1.2.1	机器模式处理器状态寄存器 (MSTATUS)	571
16.1.2.2	机器模式处理器指令集架构寄存器 (MISA)	574
16.1.2.3	机器模式异常降级控制寄存器 (MEDELEG)	574
16.1.2.4	机器模式中断降级控制寄存器 (MIDELEG)	574
16.1.2.5	机器模式中断使能控制寄存器 (MIE)	575
16.1.2.6	机器模式向量基址寄存器 (MTVEC)	576
16.1.2.7	机器模式计数器访问授权寄存器 (MCOUNTEREN)	577
16.1.3	机器模式异常处理寄存器组	577
16.1.3.1	机器模式异常临时数据备份寄存器 (MSCRATCH)	577
16.1.3.2	机器模式异常保留程序计数器寄存器 (MEPC)	577
16.1.3.3	机器模式异常事件向量寄存器 (MCAUSE)	577
16.1.3.4	机器模式中断等待状态寄存器 (MIP)	578
16.1.4	机器模式内存保护寄存器组	579
16.1.4.1	机器模式物理内存保护配置寄存器 (PMPCFG)	579
16.1.4.2	机器模式物理内存地址寄存器 (PMPADDR)	580
16.1.5	机器模式计数器寄存器组	580
16.1.5.1	机器模式周期计数器 (MCYCLE)	580
16.1.5.2	机器模式退休指令计数器 (MINSTRET)	580
16.1.5.3	机器模式事件计数器 (MHPMCOUNTER _n)	580
16.1.6	机器模式计数器配置寄存器组	580
16.1.6.1	机器模式事件选择器 (MHPMEVENT _n)	580
16.1.7	机器模式处理器控制和状态扩展寄存器组	581
16.1.7.1	机器模式扩展状态寄存器 (MXSTATUS)	581

16.1.7.2	机器模式硬件配置寄存器 (MHCR)	583
16.1.7.3	机器模式硬件操作寄存器 (MCOR)	584
16.1.7.4	机器模式隐式操作寄存器 (MHINT)	585
16.1.7.5	机器模式复位向量基址寄存器 (MRVBR)	586
16.1.7.6	机器模式计数器写使能授权寄存器 (MCOUNTERWEN)	587
16.1.7.7	机器模式事件中断使能寄存器 (MCOUNTERINTEN)	587
16.1.7.8	机器模式事件计数器上溢出标注寄存器 (MCOUNTEROF)	587
16.1.7.9	机器模式外设地址高位寄存器 (MAPBADDR)	588
16.1.8	机器模式 Cache 访问扩展寄存器组	588
16.1.8.1	机器模式 Cache 指令寄存器 (MCINS)	588
16.1.8.2	机器模式 Cache 访问索引寄存器 (MCINDEX)	588
16.1.8.3	机器模式 Cache 数据寄存器 (MCDATA0/1)	589
16.1.9	机器模式处理器型号寄存器组	590
16.1.9.1	机器模式处理器型号寄存器 (MCPUID)	590
16.2	附录 C-2 超级用户模式控制寄存器	591
16.2.1	超级用户模式异常配置寄存器组	591
16.2.1.1	超级用户模式处理器状态寄存器 (SSTATUS)	591
16.2.1.2	超级用户模式中断使能控制寄存器 (SIE)	591
16.2.1.3	超级用户模式向量基址寄存器 (STVEC)	592
16.2.1.4	超级用户模式计数器访问授权寄存器 (SCOUNTEREN)	592
16.2.2	超级用户模式异常处理寄存器组	592
16.2.2.1	超级用户模式异常临时数据备份寄存器 (SSCRATCH)	592
16.2.2.2	超级用户模式异常保留程序计数器寄存器 (SEPC)	593
16.2.2.3	超级用户模式异常事件向量寄存器 (SCAUSE)	593
16.2.2.4	超级用户模式中断等待状态寄存器 (SIP)	593
16.2.2.5	超级用户模式异常事件原因寄存器 (STVAL)	593
16.2.3	超级用户模式地址转换寄存器组	594
16.2.3.1	超级用户模式地址转换寄存器 (SATP)	594
16.2.4	超级用户模式处理器控制和状态扩展寄存器组	594
16.2.4.1	超级用户模式扩展状态寄存器 (SXSTATUS)	594
16.2.4.2	超级用户模式硬件控制寄存器 (SHCR)	594
16.2.4.3	超级用户模式事件溢出中断使能寄存器 (SCOUNTERINTEN)	594
16.2.4.4	超级用户模式事件上溢出标注寄存器 (SCOUNTEROF)	595
16.2.4.5	超级用户模式周期计数器 (SCYCLE)	595
16.2.4.6	超级用户模式退休指令计数器 (SINSTRET)	595
16.2.4.7	超级用户模式事件计数器 (SHPMCOUNTER _n)	595
16.2.5	超级用户模式 MMU 扩展寄存器组	595
16.2.5.1	超级用户模式 MMU 控制寄存器 (SMCIR)	595
16.2.5.2	超级用户模式 MMU 控制寄存器 (SMIR)	596
16.2.5.3	超级用户模式 MMU 控制寄存器 (SMEH)	596
16.2.5.4	超级用户模式 MMU 控制寄存器 (SMEL)	596
16.3	附录 C-3 用户模式控制寄存器	596
16.3.1	用户模式浮点控制寄存器组	596
16.3.1.1	浮点异常累积状态寄存器 (FFLAGS)	596
16.3.1.2	浮点动态舍入模式寄存器 (FRM)	596

16.3.1.3	浮点控制状态寄存器 (FCSR)	596
16.3.2	用户模式事件计数器寄存器组	598
16.3.2.1	用户模式周期计数器 (CYCLE)	598
16.3.2.2	用户模式时间计数器 (TIME)	598
16.3.2.3	用户模式退休指令计数器 (INSTRET)	598
16.3.2.4	用户模式事件计数器 (HPMCOUNTER _n)	598
16.3.3	用户模式扩展浮点控制寄存器组	599
16.3.3.1	用户模式浮点扩展控制寄存器 (FXCR)	599
16.3.4	矢量扩展寄存器组	600
16.3.4.1	矢量起始位置寄存器 (VSTART)	600
16.3.4.2	定点溢出标志位寄存器 (VXSAT)	600
16.3.4.3	定点舍入模式寄存器 (VXRM)	600
16.3.4.4	矢量长度寄存器 (VL)	600
16.3.4.5	矢量数据类型寄存器 (VTYPE)	601
16.3.4.6	矢量寄存器字节数寄存器 (VLENB)	602

买头哥半身体

第一章 概述

1.1 简介

C906 是基于 RISC-V 指令架构的 64 位超高效处理器，主要面向安防监控、智能音箱、扫码/刷脸支付等领域。

1.2 特点

C906 处理器体系结构的主要特点如下：

- RV64IMA[FD]C[V] 指令架构；
- 5 级单发按序执行流水线；
- 一级哈佛结构的指令和数据缓存，大小为 8KB/16KB/32KB/64KB 可配置，缓存行为 64B；
- Sv39 内存管理单元，实现虚实地址转换与内存管理；
- 支持 AXI4.0 128 比特 Master 接口；
- 支持核内中断 CLINT 和中断控制器 PLIC；

矢量计算单元的主要特征点如下：

- 遵循 RISC-V V 矢量扩展标准 (revision 0.7.1)；
- 算力可达 4GFlops (@1GHz)；
- 支持配置矢量执行单元；
- 支持 INT8/INT16/INT32/FP16/FP32 矢量运算；

1.3 可配置选项

C906 可配置选项如表 1.1 所示。

表 1.1: C906 可配置选项

可配置单元	配置选项	详细
矢量单元	无/有	如果配置矢量执行单元，默认带浮点单元。
浮点单元	无/半 + 单 + 双精度浮点	半/单/双精度浮点单元可以作为整体进行配置。
L1 ICache	8KB/16KB/32KB/64KB	可以配置 8KB、16KB、32KB、64KB。
L1 DCache	8KB/16KB/32KB/64KB	可以配置 8KB、16KB、32KB、64KB。
MMU 表项数	128/256/512	jTLB 的表项数。
PMP 表项数	8/16	PMP 表项数可以配置 8 和 16。
BHT	8Kb/16Kb	分支历史表大小
中断数量	32-1024	PLIC 可配置接入的中断源数量。

1.4 命名规则

1.4.1 符号

本文档用到的标准符号和操作符如图 1.1 所示。

符号	功能
+	加
-	减
*	乘
/	除
>	大于
<	小于
=	等于
≥	大于或等于
≤	小于或等于
≠	不等于
&	与
	或
⊕	异或
NOT	取反
:	连接
⇒	传输
↔	交换
±	误差
0b0011	二进制数
0x0F	十六进制数
rd	整型目的寄存器
rs1	整型源寄存器1
rs2	整型源寄存器2
rs3	整型源寄存器3
fd	浮点目的寄存器
fs1	浮点源寄存器1
fs2	浮点源寄存器2
fs3	浮点源寄存器3
vd	矢量目的寄存器
vs1	矢量源寄存器1
vs2	矢量源寄存器2
vs3	矢量源寄存器3

图 1.1: 符号列表

1.4.2 术语

- 逻辑 1：指对应于布尔逻辑真的电平值。
- 逻辑 0：指对应于布尔逻辑伪的电平值。
- 置位：指使得某个或某几个位达到逻辑 1 对应的电平值。
- 清除：指使得某个或某几个位达到逻辑 0 对应的电平值。
- 保留位：为功能的扩展而预留的，没有特殊说明时其值为 0。
- 信号：指通过它的状态或状态间的转换来传递信息的电气值。
- 引脚：表示一种外部电气物理连接，同一个引脚可以连接多个信号。
- 使能：指使某个离散信号处在有效的状态：
 - 低电平有效信号从高电平切换到低电平；
 - 高电平有效信号从低电平切换到高电平。
- 禁止：指使某个处在使能状态的信号状态改变：
 - 低电平有效信号从低电平切换到高电平；
 - 高电平有效信号从高电平切换到低电平。
- LSB：最低有效位。
- MSB：最高有效位。
- 信号、位域、控制位的表示都使用一种通用的规则。
- 标识符跟着表示范围的数字，从高位到低位表示一组信号，比如：
 - `addr[4:0]`：表示一组地址总线；
 - `addr[4]` 表示最高位，`addr[0]` 表示最低位。
- 单个的标识符就表示单个信号，比如：`pad_cpu_rst_b` 就表示单独的一个信号。
- 有时候会在标识符后加上数字表示一定的意义，比如：`addr15` 就表示一组总线中的第 16 位。

1.5 版本说明

C906 兼容 RISC-V 标准，具体版本为：

- *The RISC-V Instruction Set Manual, Volume I: RISC-V User-Level ISA, Version 2.2.*
- *The RISC-V Instruction Set Manual, Volume II: Privileged Architecture, Version 1.10.*
- *RISC-V “V” Vector Extension, Version 0.7.1-20190610-Workshop-Release.*
- 性能监测单元 (HPM) 增加了 *The RISC-V Instruction Set Manual, Volume II: Privileged Architecture, Version 20190125-Public-Review-draft* 中的 MCOUNTINHIBIT 寄存器。

第二章 处理器简介

2.1 结构框图

C906 结构框图如图 2.1 所示。

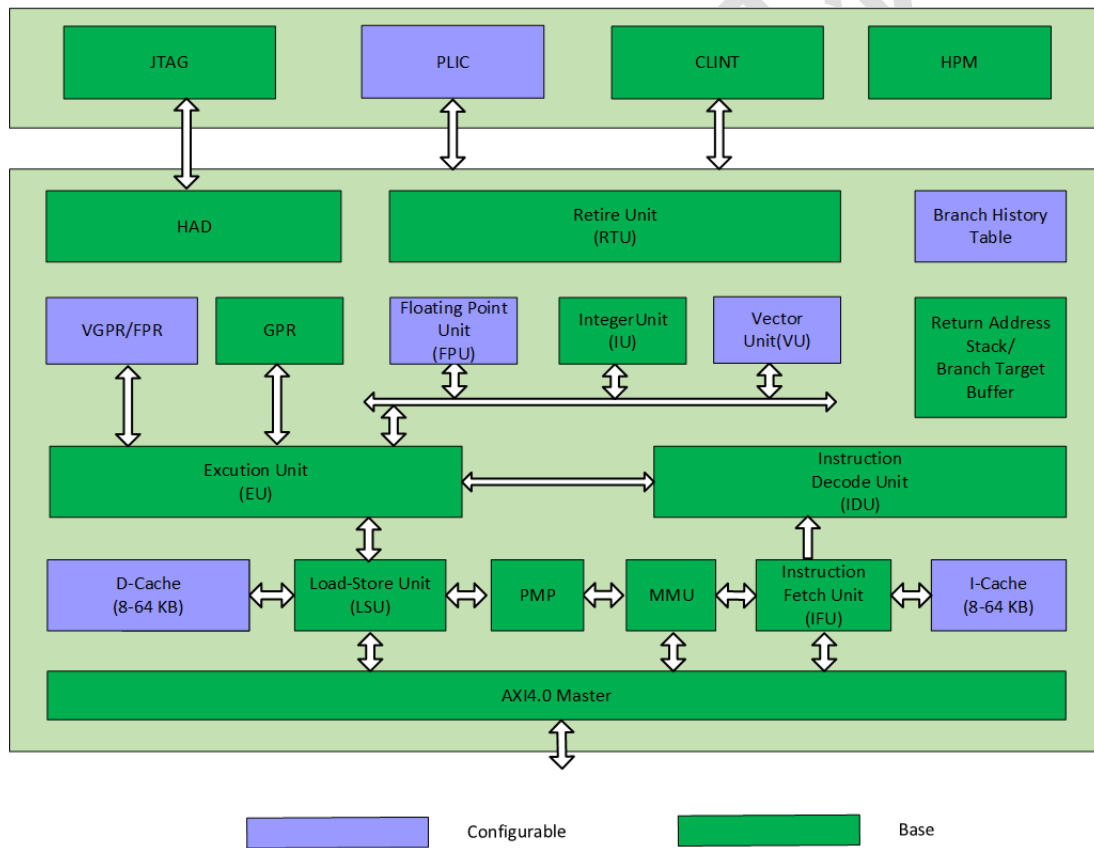


图 2.1: C906 微体系结构图

2.2 单元介绍

C906 核内子系统主要包含：指令提取单元 (IFU)、指令译码单元 (IDU)、整型执行单元 (IU)、浮点单元 (FPU)、可配的矢量执行单元 (VPU)、存储载入单元 (LSU)、指令退休单元 (RTU)、虚拟内存管理单元 (MMU)、物理内存

保护单元 (PMP)、主设备接口单元 (AXI Master IF) 等。

2.2.1 指令提取单元

指令提取单元 (IFU)，一次可最多提取 2 条指令并对其并行处理；配备高速缓存，在高速缓存缺失时采用关键指令优先获取技术；配备指令暂存器缓存预取指令；采用低成本高预测准确度的 Gshared 分支指令跳转预测器；整个指令提取单元拥有低功耗，高分支预测准确率，高指令预取效率的特点。

2.2.2 指令译码单元

指令译码单元 (IDU) 负责单条指令的译码和数据相关性检测。依据后级流水线执行情况，更新并解决指令的数据相关性信息，并将指令发送至下级流水线执行。

2.2.3 执行单元

执行单元包含整型单元 (IU)、可配置的浮点单元 (FPU) 以及可配置的矢量执行单元 (VPU)。当配置矢量执行单元时，浮点单元在支持原有标量浮点基础上，扩展成为矢量浮点单元，并支持矢量浮点计算，同时新增矢量整型单元，并支持矢量整型计算。

IU 含若干算术逻辑单元 (ALU)、乘法单元 (MULT)、除法单元 (DIV) 和跳转单元 (BJU)。其中，ALU 执行标准的 64 位整数操作，大部分常用指令单周期产生运算结果，如加减法、移位、逻辑运算等。ALU 通过操作数前馈减少数据真相关，单周期 ALU 指令不存在数据真相关停顿延时。MULT 支持 16*16、32*32、64*64 整数乘法。除法执行周期 2~36 不等。BJU 单周期内完成分支预测错误处理，加速处理器性能。

FPU 含浮点算术逻辑单元 (FALU)、浮点乘累加单元 (FMAU) 和一个浮点除法开方单元 (FDSU)。浮点单元支持半精度、单精度和双精度运算。浮点算术逻辑单元 (FALU) 负责加减、比较、转换、寄存器传输、符号注入、分类等操作。FMAU 负责普通乘法、融合乘累加等操作。浮点除法开方单元 (FDSU) 负责浮点除法、浮点开方等操作。浮点除法开方单元采用基 4 的 SRT 算法，执行周期 4~17 不等。

矢量浮点单元包括矢量浮点算术逻辑单元 (VFALU)、矢量浮点乘累加单元 (VFMAU) 和矢量浮点除法开方单元 (VFDSU)。矢量浮点单元支持不同位宽的矢量浮点运算。VFALU 负责矢量浮点加减法、寄存器传输、符号注入指令、分类指令、转换指令等操作。VFMAU 负责矢量乘法、矢量融合乘累加等操作。VFDSU 负责矢量浮点除法、矢量浮点开方等操作。矢量浮点除法开方单元采用基 4 的 SRT 算法，执行周期 5~33 不等。

矢量整型单元包含矢量加法单元 (VALU)、矢量移位单元 (VSHIFT)、矢量乘法单元 (VMUL)、矢量除法单元 (VDIVU)、矢量置换单元 (VPERM)、矢量缩减单元 (VREDU) 以及矢量逻辑操作单元 (VMISC)。VALU 负责加减法、比较、取最大最小等操作；VSHIFT 负责矢量移位操作。VMUL 负责矢量乘法、乘累加操作，8bit 及 16bit 乘法及乘累加可在 3 周期内完成；32bit 矢量乘法、乘累加和部分 reduction 加法指令在第 4 周期产生执行结果；VDIVU 单元采用基 4 的 SRT 算法，执行周期 3~21 不等。

2.2.4 存储载入单元

存储载入单元 (LSU) 支持标量存储/加载指令、矢量存储/加载指令的执行，支持高速缓存的非阻塞访问。支持字节、半字、字、双字和四字的存储/载入指令，并支持字节/半字/字载入指令的符号位和零扩展。具有内部前馈机制，消除存储指令回写数据的相关性。存储/加载指令可以流水执行，数据吞吐量达到一个周期存取一个数据。支持多路数据流硬件预取技术，提前将内存中数据预取回 L1 数据高缓存中。

2.2.5 虚拟内存管理单元

虚拟内存管理单元 (MMU) 遵从 RISC-V Sv39 标准, 将 39 位虚拟地址转换为 40 位物理地址。MMU 在 Sv39 定义的硬件回填标准基础上, 扩展了软件回填方式和地址属性。

具体信息参考[虚拟内存管理](#)。

2.2.6 物理内存保护单元

物理内存保护单元 (PMP) 遵从 RISC-V 标准, 支持配置 8 或 16 个表项, 最小粒度为 4KB, 不支持 NA4 模式。

具体信息参考[物理内存保护](#)。

2.2.7 主设备接口

主设备接口单元, 支持 AXI4.0 协议, 支持关键字优先的地址访问, 可以在不同的系统时钟与 CPU 时钟比例 (1:1, 1:2, 1:3, 1:4, 1:5, 1:6, 1:7, 1:8) 下工作。

2.2.8 平台级中断控制器

平台级中断控制器 (PLIC) 最多支持 1023 个外部中断源采样和分发, 支持电平和脉冲中断, 可以设置 32 个级别的中断优先级。

具体信息参考[中断控制器 PLIC](#)

2.2.9 计时器

RISC-V 定义了一个 SoC 系统共用的 64 位系统计时器, C906 内部拥有私有的计时器比较值寄存器, CPU 通过采集系统计时器的数值与软件设置的私有计时器比较值寄存器进行比较, 产生计时器中断信号。

具体信息参考[计时器中断](#)。

2.3 接口

C906 接口按照功能划分主要有以下几类: 时钟信号、复位信号、总线接口信号、中断信号、调试系统接口信号、低功耗指示信号、DFT 信号和 CPU 运行观测信号, C906 主要的接口 [图 2.2](#) 所示。

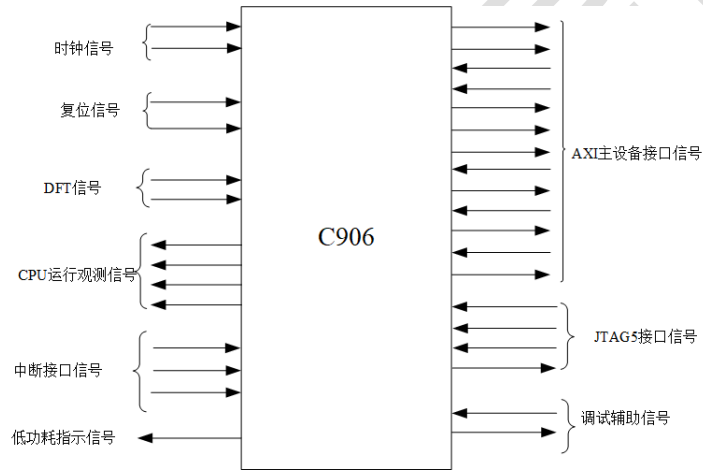


图 2.2: C906 接口概览

第三章 编程模型

3.1 工作模式及寄存器视图



图 3.1: 编程模型

C906 支持 RISC-V 三种特权模式：机器模式、超级用户模式和用户模式。处理器复位后在机器模式下执行程序，三种运行模式对应不同的操作权限，区别主要体现在以下几个方面：

1. 对寄存器的访问；
2. 特权指令的使用；
3. 对内存空间的访问。

用户模式权限最低，普通用户程序只允许访问用户模式寄存器避免普通用户程序接触特权信息。在普通用户程序访问用户模式寄存器时，操作系统通过协调与普通用户程序访问的行为来为普通用户程序提供管理和服务，用户程序对内存空间的访问受到 PMP 的保护。

超级用户模式权限比用户模式高，比机器模式低。超级用户模式下运行的程序可以访问用户模式和超级用户模式寄存器，不可以访问机器模式控制寄存器，对内存空间的访问受到 PMP 的保护。

机器模式拥有最高的权限，在机器模式下运行的程序对内存，I/O 和一些对于启动和配置系统来说必要的底层应用程序有着完全的访问权限。默认情况下（异常中断没有被降级处理），任何模式下发生的异常和中断都会切换到机器模式进行响应。

大多数指令在三种模式下都能执行，但是一些对系统产生重大影响的特权指令只能工作在超级用户模式或机器模式下执行。具体信息可参考附录 A 标准指令术语 和附录 B 平头哥扩展指令术语，查看指令的执行权限。

处理器的工作模式在异常响应时发生变化，处理器进入更高特权模式响应异常，异常响应完之后再回到低特权模式。

3.2 通用寄存器

C906 拥有 32 个 64 位的通用寄存器，功能定义与 RISC-V 一致，如表 3.1 所示。

表 3.1: 通用寄存器

寄存器	ABI 名称	描述
x0	zero	零值寄存器。
x1	ra	返回地址。
x2	sp	堆栈指针。
x3	gp	全局指针。
x4	tp	线程指针。
x5	t0	临时/备用链接寄存器。
x6-7	t1-2	临时寄存器。
x8	s0/fp	保留寄存器/帧指针。
x9	s1	保留寄存器。
x10-11	a0-1	函数参数/返回值。
x12-17	a2-7	函数参数。
x18-27	s2-11	保留寄存器。
x28-31	t3-6	临时寄存器。

通用寄存器用于保存指令操作数、指令执行结果和地址等信息。

3.3 浮点寄存器

C906 浮点单元除支持标准 RV64FD 指令集外，扩展支持半精度浮点计算，拥有 32 个独立的 64 位浮点寄存器，可在用户模式、超级用户模式和机器模式下被访问。

表 3.2: 浮点寄存器

寄存器	ABI 名称	描述
f0-7	ft0-7	浮点临时寄存器。
f8-9	fs0-1	浮点保留寄存器。
f10-11	fa0-1	浮点参数/返回值。
f12-17	fa2-7	浮点参数。
f18-27	fs2-11	浮点保留寄存器。
f28-31	ft8-11	浮点临时寄存器。

浮点寄存器 f0 和通用寄存器 x0 不同，并不是零值寄存器，而是和其他浮点寄存器一样，值是可变的。单精度浮点数仅使用 64 位浮点寄存器的低 32 位，高 32 位必须全为 1，否则会被当做非数处理；半精度浮点数仅使用 64 位浮点寄存器的低 16 位，高 48 位必须全为 1，否则会被当作非数处理。

增加单独的浮点寄存器可以增大寄存器容量和带宽，进而提高处理器性能。

3.3.1 与通用寄存器传输数据

通用寄存器跟浮点寄存器之间的数据传输可以通过浮点寄存器传送指令实现。浮点寄存器传送指令包括：

- FMV.X.H/FMV.H.X 浮点寄存器半精度传送指令。
- FMV.X.W/FMV.W.X 浮点寄存器单精度传送指令。
- FMV.X.D/FMV.D.X 浮点寄存器双精度传送指令。

从通用寄存器传送一个半/单/双精度浮点数据到浮点寄存器中，数据格式不会因为传输而改变，所以程序可以直接使用这些寄存器而不必对数据格式做类型转换。

具体指令说明和定义可以参考附录 A-4 F 指令术语

3.3.2 维护寄存器精度的一致

浮点寄存器可存储半精度浮点数、单精度浮点数、双精度浮点数和整型数据。比如在浮点寄存器 f1 中所存的数据类型，取决于上一次的写操作，所以数据类型可以是四种数据类型中的任何一种。

浮点单元在硬件上不对数据类型做任何数据格式上的检测，硬件对浮点寄存器中数据格式的解析只取决于执行的浮点指令本身，而不关心这个寄存器上次的写操作的数据格式。这完全是靠编译器或者程序本身来保证寄存器中的数据精度的一致性。

3.4 矢量寄存器

当配置矢量执行单元时，C906 拥有 32 个独立的 128 位矢量寄存器，可在用户模式、超级用户模式和机器模式下被访问。矢量寄存器通过矢量传送指令实现与整型通用寄存器/浮点寄存器的数据交换。

3.4.1 与通用寄存器传输数据

矢量寄存器与通用寄存器之间的数据传输可以通过矢量整型寄存器传送指令实现。矢量整型寄存器传送指令包括：

- VMV.V.X 整型传送至矢量指令；
- VMV.S.X 整型传送至矢量首元素指令；
- VEXT.X.V 矢量整型提取元素指令。

3.4.2 与浮点寄存器传输数据

矢量寄存器与浮点寄存器之间的数据传输可以通过矢量浮点寄存器传送指令实现。矢量浮点寄存器传送指令包括：

- VFMV.V.F 浮点传送至矢量指令；
- VFMV.F.S 矢量首元素传送至浮点指令；
- VFMV.S.F 浮点传送至矢量首元素指令。

3.5 系统控制寄存器

本节分别列出机器模式、超级用户模式和用户模式的控制寄存器并简要说明。

3.5.1 机器模式控制寄存器

C906 中实现的 RISC-V 标准定义的机器模式控制寄存器如表 3.3 所示。

表 3.3: RISC-V 标准机器模式控制寄存器

名称	读写权限	寄存器编号	描述
机器模式信息寄存器组			
MVENDORID	机器模式只读	0xF11	供应商编号寄存器
MARCHID	机器模式只读	0xF12	架构编号寄存器
MIMPID	机器模式只读	0xF13	机器模式硬件实现编号寄存器
MHARTID	机器模式只读	0xF14	机器模式逻辑内核编号寄存器
机器模式异常配置寄存器组			
MSTATUS	机器模式读写	0x300	机器模式处理器状态寄存器
MISA	机器模式读写	0x301	机器模式处理器指令集架构寄存器
MEDELEG	机器模式读写	0x302	机器模式异常降级控制寄存器
MIDELEG	机器模式读写	0x303	机器模式中断降级控制寄存器
MIE	机器模式读写	0x304	机器模式中断使能控制寄存器
MTVEC	机器模式读写	0x305	机器模式向量基址寄存器
MCOUNTEREN	机器模式读写	0x306	机器模式计数器授权控制寄存器
机器模式异常处理寄存器组			
MSCRATCH	机器模式读写	0x340	机器模式异常临时数据备份寄存器

下页继续

表 3.3 – 续上页

名称	读写权限	寄存器编号	描述
MEPC	机器模式读写	0x341	机器模式异常保留程序计数器
MCAUSE	机器模式读写	0x342	机器模式异常事件原因寄存器
MTVAL	机器模式读写	0x343	机器模式异常事件向量寄存器
MIP	机器模式读写	0x344	机器模式中断等待状态寄存器
机器模式内存保护寄存器组			
PMPCFG0	机器模式读写	0x3A0	物理内存保护配置寄存器 0
PMPCFG2	机器模式读写	0x3A2	物理内存保护配置寄存器 2
PMPADDR0	机器模式读写	0x3B0	物理内存保护基址寄存器 0
.....
PMPADDR15	机器模式读写	0x3BF	物理内存保护基址寄存器 15
机器模式计数器/计时器			
MCYCLE	机器模式读写	0xB00	机器模式周期计数器
MINSTRET	机器模式读写	0xB02	机器模式退休指令计数器
MHPMCOUNTER3	机器模式读写	0xB03	机器模式计数器 3
.....
MHPMCOUNTER17	机器模式读写	0xB11	机器模式计数器 17
机器模式计数器配置寄存器组			
MHPMEVENT3	机器模式读写	0x323	机器模式事件选择寄存器 3
.....
MHPMEVENT17	机器模式读写	0x331	机器模式事件选择寄存器 17

C906 中扩展的机器模式控制寄存器如 表 3.4 所示。

表 3.4: C906 扩展机器模式控制寄存器

名称	读写权限	寄存器编号	描述
机器模式处理器控制和状态扩展寄存器组			
MCOUNTINHIBIT	机器模式读写	0x320	机器模式计数禁止寄存器
MXSTATUS	机器模式读写	0x7C0	机器模式扩展状态寄存器
MHCR	机器模式读写	0x7C1	机器模式硬件配置寄存器
MCOR	机器模式读写	0x7C2	机器模式硬件操作寄存器
MHINT	机器模式读写	0x7C5	机器模式隐式操作寄存器
MRVBR	机器模式读写	0x7C7	机器模式复位向量基址寄存器
MCER	机器模式读写	0x7C8	机器模式 L1Cache ECC 寄存器
MCOUNTERWEN	机器模式读写	0x7C9	超级用户模式计数器写使能寄存器
MCOUNTERINTEN	机器模式读写	0x7CA	机器模式事件中断使能寄存器
MCOUNTEROF	机器模式读写	0x7CB	机器模式事件上溢出标注寄存器
MAPBADDR	机器模式只读	0xFC1	机器模式外设地址高位寄存器
机器模式 Cache 访问扩展寄存器组			
MCINS	机器模式读写	0x7D2	机器模式 Cache 指令寄存器
MCINDEX	机器模式读写	0x7D3	机器模式 Cache 访问索引寄存器
MCDATA0	机器模式读写	0x7D4	机器模式 Cache 数据寄存器 0
MCDATA1	机器模式读写	0x7D5	机器模式 Cache 数据寄存器 1
机器模式处理器型号扩展寄存器组			
MCPUID	机器模式只读	0xFC0	机器模式处理器型号寄存器

具体寄存器的定义和功能，请参考附录 C-1 机器模式控制寄存器。

3.5.2 超级用户模式控制寄存器

C906 中实现的 RISC-V 标准定义的超级用户模式控制寄存器如表 3.5 所示。

表 3.5: RISC-V 标准超级用户模式控制寄存器

名称	读写权限	寄存器编号	描述
超级用户模式异常配置寄存器组			
SSTATUS	超级用户模式读写	0x100	超级用户模式处理器状态寄存器
SIE	超级用户模式读写	0x104	超级用户模式中断使能控制寄存器
STVEC	超级用户模式读写	0x105	超级用户模式向量基址寄存器
SCOUNTEREN	超级用户模式读写	0x106	超级用户模式计数器使能控制寄存器
超级用户模式异常处理寄存器组			
SSCRATCH	超级用户模式读写	0x140	超级用户模式异常临时数据备份寄存器
SEPC	超级用户模式读写	0x141	超级用户模式异常保留程序计数器
SCAUSE	超级用户模式读写	0x142	超级用户模式异常事件原因寄存器
STVAL	超级用户模式读写	0x143	超级用户模式异常事件向量寄存器
SIP	超级用户模式读写	0x144	超级用户模式中断等待状态寄存器
超级用户模式地址转换寄存器组			
SATP	超级用户模式读写	0x180	超级用户虚拟地址转换和保护寄存器

C906 中扩展的超级用户模式控制寄存器如 表 3.6 所示。

表 3.6: C906 扩展超级用户模式控制寄存器

名称	读写权限	寄存器编号	描述
超级用户模式处理器控制和状态扩展寄存器组			
SXSTATUS	超级用户模式读写	0x5C0	超级用户模式扩展状态寄存器
SHCR	超级用户模式读写	0x5C1	超级用户模式硬件控制寄存器
SCOUNTERINTEN	超级用户模式读写	0x5C4	超级用户模式事件中断使能寄存器
SCOUNTEROF	超级用户模式读写	0x5C5	超级用户模式事件上溢出标注寄存器
SCYCLE	超级用户模式读写	0x5E0	超级用户模式周期计数器
SINSTRET	超级用户模式读写	0x5E2	超级用户模式退休指令计数器
SHPMCOUNTER3	超级用户模式读写	0x5E3	超级用户模式计数器 3
.....
SHPMCOUNTER17	超级用户模式读写	0x5F1	超级用户模式计数器 17
超级用户模式 MMU 扩展寄存器组			
SMIR	超级用户模式读写	0x9C0	超级用户模式 MMU Index 寄存器
SMEL	超级用户模式读写	0x9C1	超级用户模式 MMU EntryLo 寄存器
SMEH	超级用户模式读写	0x9C2	超级用户模式 MMU EntryHi 寄存器
SMCIR	超级用户模式读写	0x9C3	超级用户模式 MMU 控制寄存器

具体寄存器的定义和功能，请参考附录 C-2 超级用户模式控制寄存器

3.5.3 用户模式控制寄存器

C906 中实现的 RISC-V 标准定义的用户模式控制寄存器如 RISC-V 标准用户模式控制寄存器 所示。

表 3.7: RISC-V 标准用户模式控制寄存器

名称	读写权限	寄存器编号	描述
用户模式浮点控制寄存器组			
FFLAGS	用户模式读写	0x001	浮点异常累积状态寄存器
FRM	用户模式读写	0x002	浮点动态舍入模式控制寄存器
FCSR	用户模式读写	0x003	浮点控制状态寄存器
用户模式计数/计时器			
CYCLE	用户模式只读	0xC00	用户模式周期计数器
TIME	用户模式只读	0xC01	用户模式时间计数器
INSTRET	用户模式只读	0xC02	用户模式退休指令计数器
HPMCOUNTER3	用户模式只读	0xC03	用户模式计数器 3
.....
HPMCOUNTER17	用户模式只读	0xC1F	用户模式计数器 17
矢量扩展寄存器组			
VSTART	用户模式读写	0x008	矢量起始位置寄存器
VXSAT	用户模式读写	0x009	定点溢出标志位寄存器
VXRM	用户模式读写	0x00A	定点舍入模式寄存器
VL	用户模式只读	0xC20	矢量长度寄存器
VTYPE	用户模式只读	0xC21	矢量数据类型寄存器
VLENB	用户模式只读	0xC22	矢量寄存器字节数寄存器

C906 中扩展的用户模式控制寄存器如表 3.8 所示。

表 3.8: C906 扩展用户模式控制寄存器

名称	读写权限	寄存器编号	描述
用户模式扩展浮点控制寄存器组			
FXCR	用户模式读写	0x800	用户模式扩展浮点控制寄存器

具体寄存器的定义和功能，请参考附录 C-3 用户模式控制寄存器。

3.6 异常处理

异常处理 (包括指令异常和外部中断) 是处理器的一项重要功能。在某些异常事件产生时, 异常处理功能用来使处理器转入对这些异常事件的处理。这些异常事件包括硬件错误、指令执行错误、用户程序请求服务等。

异常处理功能的关键是在异常事件发生时, 可以保存处理器当前运行的状态, 在处理器退出异常处理后将处理器恢复为异常处理前的运行状态。异常能够在流水线的各个阶段被识别, 硬件会保证触发异常的后续指令不会改变处理器的状态。异常在指令的边界上被处理, 即处理器在指令退休时响应异常, 并保存退出异常处理并返回执行的指令地址。即使异常指令退休前被识别, 异常也要在相应的指令退休时才会被处理。为了程序功能的正确性, 处理器在异常处理结束后要避免重复执行已执行完成的指令。

以在机器模式响应异常为例, 具体步骤为:

第一步: 处理器保存异常指令 PC 到 MEPC 中。

第二步：根据发生的异常类型设置 MCAUSE，并更新 MTVAL 为出错的取指地址、存储/加载地址或者指令码。

第三步：将 MSTATUS 的中断使能位域 MIE 保存到 MPIE 域中，将 MIE 域的值清零，禁止响应中断。

第四步：将发生异常之前的权限模式保存到 MSTATUS 的 MPP 域中，切换到机器模式（没有做异常降级响应处理的话）。

第五步：根据 MTVEC 中的基址和模式，得到异常服务程序入口地址。处理器从异常服务程序的第一条指令处开始执行，进行异常的处理。

C906 遵从 RISC-V 标准的异常向量表如表 3.9 所示。

表 3.9: 异常和中断向量分配

中断标记	异常向量号	描述
1	0	未实现
1	1	超级用户模式软件中断
1	2	保留
1	3	机器模式软件中断
1	4	未实现
1	5	超级用户模式计时器中断
1	6	保留
1	7	机器模式计时器中断
1	8	未实现
1	9	超级用户模式外部中断
1	10	保留
1	11	机器模式外部中断
1	17	性能监测溢出中断（如配置性能监测单元）
1	其他	保留
0	0	未实现
0	1	取指令访问错误异常
0	2	非法指令异常
0	3	调试断点异常
0	4	加载指令非对齐访问异常
0	5	加载指令访问错误异常
0	6	存储/原子指令非对齐访问异常
0	7	存储/原子指令访问错误异常
0	8	用户模式环境调用异常
0	9	超级用户模式环境调用异常
0	10	保留
0	11	机器模式环境调用异常
0	12	取指令页面错误异常
0	13	加载指令页面错误异常
0	14	保留
0	15	存储/原子指令页面错误异常
0	≥ 16	保留

当同时发生多个中断请求时优先级固定：机器模式外部中断 > 机器模式软件中断 > 机器模式计时器中断 > 超级用户模式外部中断 > 超级用户模式软件中断 > 超级用户模式外部中断 > 性能监测溢出中断。其中，机器模式外部中断和超级用户模式外部中断均由 PLIC 传递到核内，所有外部中断在 PLIC 内的优先级仲裁由 PLIC 控制寄存器决定。

发生异常或者中断，且在机器模式响应时，处理器会更新 pc 到 MEPC，并根据异常类型更新 MTVAL。处理器响应中断时，MEPC 更新为下一条指令的 pc，MTVAL 值更新为 0。处理器响应异常时，MEPC 更新为触发异常的指令 pc，MTVAL 根据不同异常类型进行更新，表 3.10 为为机器模式响应异常时，MTVAL 的更新值。

表 3.10: 异常发生时 MTVAL 的更新

异常向量号	异常类型	MTVAL 更新值
1	取指令访问错误异常	取指访问的虚拟地址
2	非法指令异常	指令码
3	调试断点异常	0
4	加载指令非对齐访问异常	加载访问的虚拟地址
5	加载指令访问错误异常	见下文注释
6	存储/原子指令非对齐访问异常	存储/原子访问的虚拟地址
7	存储/原子指令访问错误异常	见下文注释
8	用户模式环境调用异常	0
9	超级用户模式环境调用异常	0
11	机器模式环境调用异常	0
12	取指令页面错误异常	取指访问的虚拟地址
13	加载指令页面错误异常	加载访问的虚拟地址
15	存储/原子指令页面错误异常	存储/原子访问的虚拟地址

注释：当访问错误异常由 PMP 鉴权失败导致时，MTVAL 保存内存访问的虚拟地址；当访问错误异常由硬件总线返回错误响应导致时，MTVAL 保存内存访问的物理地址。

C906 支持异常和中断的降级响应 (Delegation)。在超级用户模式发生异常或者中断时，处理器需要切换到机器模式响应，模式切换会造成处理器性能损失。Delegation 机制支持配置中断和异常在超级用户模式响应。其中，机器模式下发生的异常不受 Delegation 控制，只在机器模式响应。机器模式外部中断、机器模式软件中断、机器模式计时器中断不支持降级到超级用户模式响应，其他中断均可以被降级到超级用户模式态响应。在机器模式下不响应被降级的中断。

在超级用户模式和用户模式下均可响应所有符合条件的中断和异常。对于未被降级的中断和异常，进入机器模式进行处理，更新机器模式异常处理寄存器组。对于被降级的中断和异常均在超级用户模式响应，更新超级用户模式异常处理寄存器组。

3.7 数据格式

3.7.1 浮点数据格式

C906 浮点单元遵从 RISC-V 标准，兼容 ANSI/IEEE 754-2008 浮点标准，支持半精度、单精度和双精度浮点运算。数据格式如 图 3.2 所示，其中单精度数据仅使用 64 位浮点寄存器的低 32 位，高 32 位需要全为 1，否则会被当作非数处理；半精度数据仅使用 64 位浮点寄存器的低 16 位，高 48 位需要全为 1，否则会被当作非数处理。

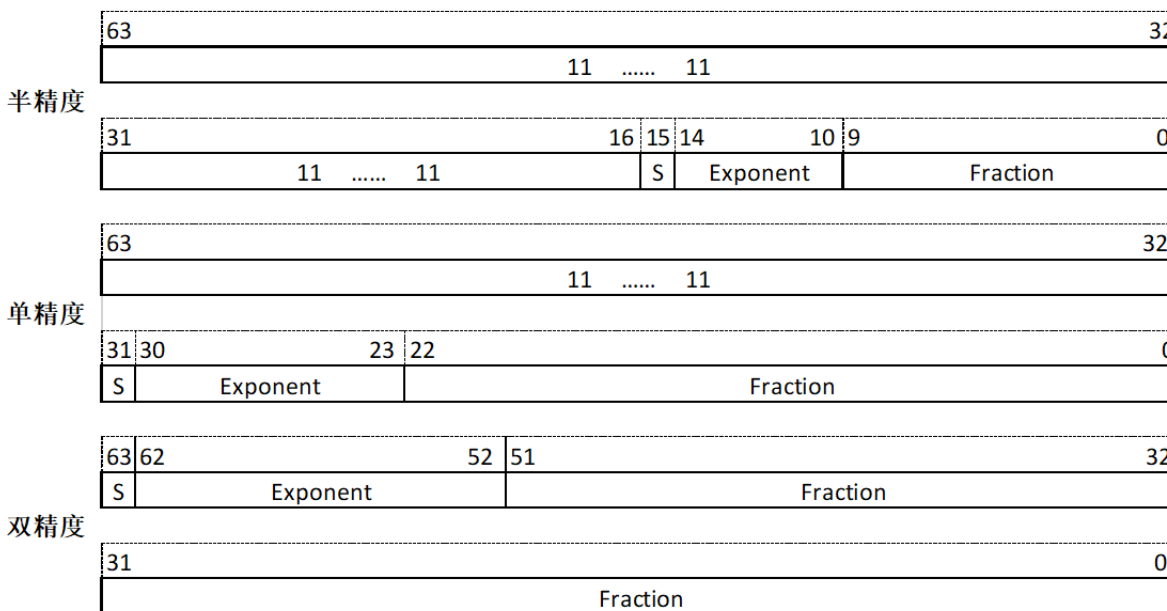


图 3.2: 浮点寄存器中的浮点数据格式

3.7.2 整型数据格式

整型寄存器内部的数值并没有大小端之分，只有有符号和无符号的区别。其格式均为从右至左表示逻辑低位到高位

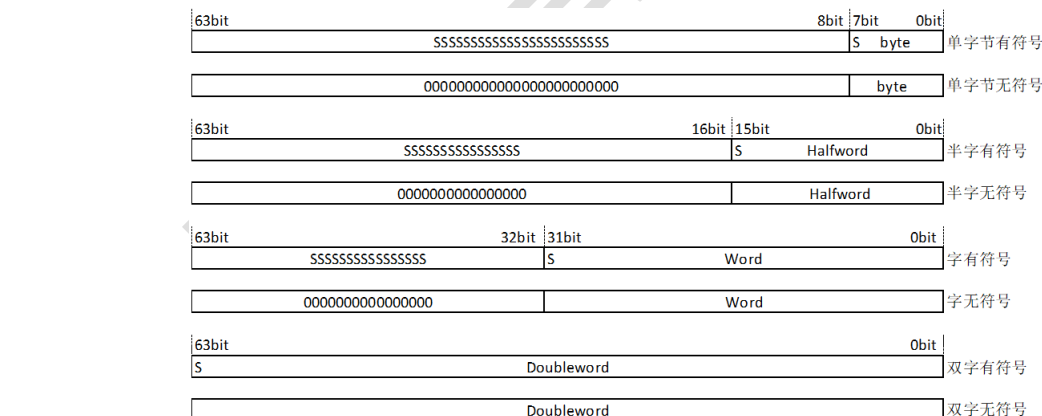


图 3.3: 整型寄存器中的整型数据格式

3.7.3 矢量数据格式

C906 矢量寄存器为 128 位，矢量寄存器中包含元素个数由当前矢量元素位宽决定。矢量寄存器支持矢量元素位宽为 8 位、16 位、32 位。不同矢量元素位宽时矢量寄存器数据排布如 图 3.4 所示。

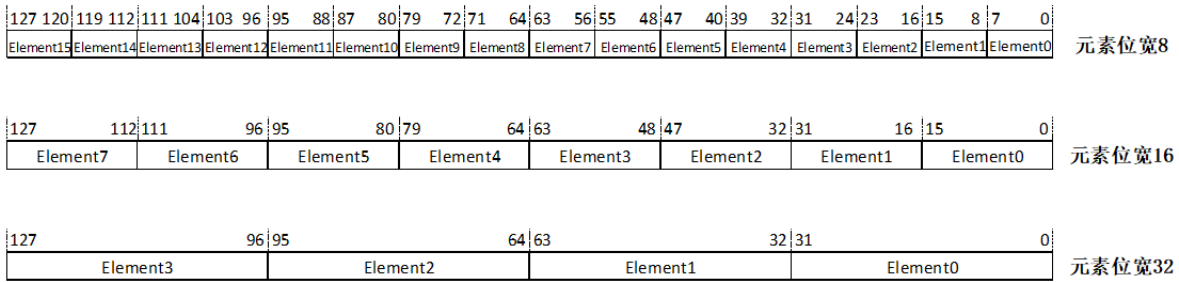


图 3.4: 矢量寄存器中的矢量数据格式

3.7.4 大小端

大小端是相对于存储器数据存储的格式而提出的，高地址字节存放至物理内存的低位被定义为大端；高地址字节存放至物理内存的高位被定义小端。小端模式的数据格式如图 3.5 所示。

A+7	A+6	A+5	A+4	A+3	A+2	A+1	A	
Byte7	Byte6	Byte5	Byte4	Byte3	Byte2	Byte1	Byte0	Double word at A
Byte7	Byte6	Byte5	Byte4	Byte3	Byte2	Byte1	Byte0	Word at A
Byte7	Byte6	Byte5	Byte4	Byte3	Byte2	Byte1	Byte0	Half word at A
Byte7	Byte6	Byte5	Byte4	Byte3	Byte2	Byte1	Byte0	Byte at A

图 3.5: 内存中的数据组织形式

C906 仅支持小端模式，支持标准补码的 2 进制整数。每个指令操作数的长度即可以显式编码在程序中 (load/store 指令)，也可以隐含在指令操作中 (index operation, byte extraction)。通常，指令接收 64 位操作数，产生 64 位结果。

3.8 内存模型

C906 支持两种内存存储类型，分别是内存 (memory) 和外设 (device) (由 SO 位区分)。其中，memory 类型根据是否可高缓 (Cacheable, C) 进一步分为可高缓内存 (Cacheable memory) 和不可高缓内存 (Non-cacheable memory)。device 类型的特点为不允许投机执行，因此 device 一定带有不可高缓的属性。device 根据是否可缓存 (Bufferable, B) 分为可缓存外设 (Bufferable device) 和不可缓存外设 (Non-bufferable device)。其中可缓存表示 slave 允许在某个中间节点快速返回写响应；不可缓存表示只有在从设备真正写完成后才返回写响应。

表 3.11 给出了各个内存类型对应的页面属性。页面属性的配置有两种方式：

1. 在所有不进行虚拟地址和物理地址转换的情况下：机器模式权限或者 MMU 关闭，地址的页面属性由 sysmap.h 文件内的宏定义决定。sysmap.h 是 C906 扩展的地址属性配置文件，对用户开放，用户可以根据自身需求，定义不同地址段的页面属性，地址区域个数上限为 8。
2. 在所有进行虚拟地址和物理地址转换的情况下：非机器模式权限且 MMU 打开时，地址的页面属性有两种配置方式：sysmap.h 文件和 C906 在 pte 中扩展的页面属性。使用哪种配置方式取决于 C906 扩展寄存器 MXSTATUS 中的 MAEE 域的值。如果 MAEE 域值为 1，地址的页面属性由对应 pte 中扩展的页面属性决定。如果 MAEE 域值为 0，地址的页面属性由 sysmap.h 决定。

表 3.11: 内存类型分类

内存类型	SO	C	B
可高缓存内存	0	1	1
不可高缓存内存	0	0	1
可缓存外设	1	0	1
不可缓存外设	1	0	0

sysmap.h 文件支持对 8 个内存地址空间的属性设定, 第 i (i 从 0 到 7) 个地址空间地址上限 (不包含) 由宏 `SYSMAP_BASE_ADDRi` (i 从 0 到 7) 定义, 地址下限 (包含) 由 `SYSMAP_BASE_ADDR(i-1)` 定义, 具体为:

$$\text{SYSMAP_BASE_ADDR}(i-1) \leq \text{第 } i \text{ 个地址空间地址} < \text{SYSMAP_BASE_ADDR}i$$

第 0 个地址空间下限是 `0x0`, 内存地址不在 `sysmap.h` 文件设定的 8 个地址区间的地址属性默认为 `Strong order/Non-cacheable/Non-bufferable`。每个地址空间上下边界是 4KB 对齐, 因此宏 `SYSMAP_BASE_ADDRi` 定义的是地址的高 28 位。

落在第 i (i 从 0 到 7) 个地址空间内的地址的属性由宏 `SYSMAP_FLAGi` (i 从 0 到 7) 定义, 属性的排布如图 3.6 所示:

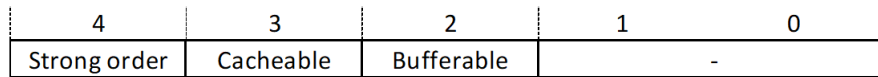


图 3.6: sysmap.h 地址属性格式

需要注意的是在 `sysmap` 和页表配置时需要将 `CLINT` 和 `PLIC` 地址空间设置为不可缓存外设。

第四章 指令集

C906 支持 RV64GCV 指令包集合，并在此基础上拓展了部分自定义指令。其中，矢量指令需要硬件配置有矢量单元才可使用，否则执行这类指令会触发非法指令异常。C906 拓展指令集中半精度浮点指令集可以直接使用，除了浮点半精度之外的所有 C906 拓展指令集需要在机器模式扩展状态寄存器 (MXSTATUS) 中打开扩展指令集使能位 (THEAD-ISAEE) 才能正常使用，否则执行后会触发非法指令异常。

4.1 RV64GCV 指令

本节主要介绍了 C906 中实现的 RV64GCV 指令集。

RV64GCV(RV64IMAFDCV) 指令集包括标准的整型指令集 (RV64I)、乘除法指令集 (RV64M)、原子指令集 (RV64A)、单精度浮点指令集 (RV64F)、双精度浮点指令集 (RV64D)、压缩指令集 (RVC) 和矢量指令集 (RVV)。

4.1.1 RV64I 整型指令集

RV64I 是 64 位基本整型指令。其中 RV64I 基本整型指令集按功能可以分为以下类型：

- 加减法指令
- 逻辑操作指令
- 移位指令
- 比较指令
- 数据传输指令
- 分支跳转指令
- 内存存取指令
- 控制寄存器操作指令
- 低功耗指令
- 异常返回指令
- 特殊功能指令

表 4.1: 整型指令 (RVI) 指令列表

指令名称	指令描述	执行延时
------	------	------

下页继续

表 4.1 – 续上页

加减法指令		
ADD	有符号加法指令	1
ADDW	低 32 位有符号加法指令	1
ADDI	有符号立即数加法指令	1
ADDIW	低 32 位有符号立即数加法指令	1
SUB	有符号减法指令	1
SUBW	低 32 位有符号减法指令	1
逻辑操作指令		
AND	按位与指令	1
ANDI	立即数按位与指令	1
OR	按位或指令	1
ORI	立即数按位或指令	1
XOR	按位异或指令	1
XORI	立即数按位异或指令	1
移位指令		
SLL	逻辑左移指令	1
SLLW	低 32 位逻辑字左移指令	1
SLLI	立即数逻辑左移指令	1
SLLIW	低 32 位立即数逻辑左移指令	1
SRL	逻辑右移指令	1
SRLW	低 32 位逻辑右移指令	1
SRLI	立即数逻辑右移指令	1
SRLIW	低 32 位立即数逻辑右移指令	1
SRA	算术右移指令	1
SRAW	低 32 位算术右移指令	1
SRAI	立即数算术右移指令	1
SRAIW	低 32 位立即数算术右移指令	1
比较指令		
SLT	有符号比较小于置位指令	1
SLTU	无符号比较小于置位指令	1
SLTI	有符号立即数比较小于置位指令	1
SLTIU	无符号立即数比较小于置位指令	1
数据传输指令		
LUI	高位立即数装载指令	1
AUIPC	PC 高位立即数加法指令	1
分支跳转指令		
BEQ	相等分支指令	1
BNE	不等分支指令	1
BLT	有符号小于分支指令	1
BGE	有符号大于等于分支指令	1
BLTU	无符号小于分支指令	1
BGEU	无符号大于等于分支指令	1

下页继续

表 4.1 – 续上页

JAL	直接跳转子程序指令	1
JALR	寄存器跳转子程序指令	1
内存存取指令		
LB	有符号扩展字节加载指令	Cacheable LOAD: ≥ 2 Cacheable STORE:1 Non-Cacheable 不定周期
LBU	无符号扩展字节加载指令	同上
LH	有符号扩展半字加载指令	同上
LHU	无符号扩展半字加载指令	同上
LW	有符号扩展字加载指令	同上
LWU	无符号扩展字加载指令	同上
LD	双字加载指令	同上
SB	字节存储指令	同上
SH	半字存储指令	同上
SW	字存储指令	同上
SD	双字存储指令	同上
控制寄存器操作指令		
CSR RW	控制寄存器读写传送指令	阻塞执行 不定周期
CSR RS	控制寄存器置位传送指令	同上
CSR RC	控制寄存器清零传送指令	同上
CSR RWI	控制寄存器立即数读写传送指令	同上
CSR RSI	控制寄存器立即数置位传送指令	同上
CSR RCI	控制寄存器立即数清零传送指令	同上
低功耗指令		
WFI	进入低功耗等待模式指令	不定周期
异常返回指令		
MRET	机器模式异常返回指令	阻塞执行 不定周期
SRET	超级用户模式异常返回指令	同上
特殊功能指令		
FENCE	存储同步指令	不定周期
FENCE.I	指令流同步指令	阻塞执行 不定周期
SFENCE.VMA	虚拟内存同步指令	同上
ECALL	环境异常指令	1
EBREAK	断点指令	1

具体指令说明和定义，请参考附录 A-1 I 指令术语

4.1.2 RV64M 乘除法指令集

表 4.2 为 RV64M 乘除法指令。

表 4.2: 整型乘除法 (RVM) 指令列表

指令名称	指令描述	执行延时
MUL	有符号乘法指令	3/6
MULW	低 32 位有符号乘法指令	3
MULH	有符号乘法取高位指令	3/6
MULHSU	有符号无符号乘法取高位指令	3/6
MULHU	无符号乘法取高位指令	3/6
DIV	有符号除法指令	5-36
DIVW	低 32 位有符号除法指令	5-20
DIVU	无符号除法指令	5-36
DIVUW	低 32 位无符号除法指令	5-20
REM	有符号取余指令	5-36
REMW	低 32 位有符号取余指令	5-20
REMU	无符号取余指令	5-36
REMUW	低 32 位无符号取余指令	5-20

具体指令说明和定义，请参考附录 A-2 M 指令术语

4.1.3 RV64A 原子指令集

表 4.3: 原子指令 (RVA) 指令列表

指令名称	指令描述	执行延时
LR.W	字加载保留指令	拆分为多条原子指令执行。 可能拆分出阻塞执行，指令延时不可预期。
LR.D	双字加载保留指令	
SC.W	字条件存储指令	
SC.D	双字条件存储指令	
AMOSWAP.W	低 32 位原子交换指令	
AMOSWAP.D	原子交换指令	
AMOADD.W	低 32 位原子加法指令	
AMOADD.D	原子加法指令	
AMOXOR.W	低 32 位原子按位异或指令	
AMOXOR.D	原子按位异或指令	
AMOAND.W	低 32 位原子按位与指令	
AMOAND.D	原子按位与指令	
AMOOR.W	低 32 位原子按位或指令	
AMOOR.D	原子按位或指令	
AMOMIN.W	低 32 位原子有符号取最小值指令	
AMOMIN.D	原子有符号取最小值指令	
AMOMAX.W	低 32 位原子有符号取最大值指令	
AMOMAX.D	原子有符号取最大值指令	
AMOMINU.W	低 32 位原子无符号取最小值指令	
AMOMINU.D	原子无符号取最小值指令	
AMOMAXU.W	低 32 位原子无符号取最大值指令	
AMOMAXU.D	原子无符号取最大值指令	

具体指令说明和定义，请参考附录 A-3 A 指令术语。

4.1.4 RV64F 单精度浮点指令集

本节主要介绍单精度浮点指令。

单精度浮点指令集按功能可以分为以下类型：

- 运算指令
- 符号注入指令
- 数据传输指令
- 比较指令
- 数据类型转换指令
- 内存存储指令
- 浮点数分类指令

表 4.4: 单精度浮点 (RVF) 指令列表

指令名称	指令描述	执行延时
运算指令		
FADD.S	单精度浮点加法指令	3
FSUB.S	单精度浮点减法指令	3
FMUL.S	单精度浮点乘法指令	3
FMADD.S	单精度浮点乘累加指令	4
FMSUB.S	单精度浮点乘累减指令	4
FNMADD.S	单精度浮点乘累加取负指令	4
FNMSUB.S	单精度浮点乘累减取负指令	4
FDIV.S	单精度浮点除法指令	4-17
FSQRT.S	单精度浮点开方指令	4-17
符号注入指令		
FSGNJ.S	单精度浮点符号注入指令	3
FSGNJN.S	单精度浮点符号取反注入指令	3
FSGNJX.S	单精度浮点符号异或注入指令	3
数据传输指令		
FMV.X.W	单精度浮点读传送指令	3
FMV.W.X	单精度浮点写传送指令	3
比较指令		
FMIN.S	单精度浮点取最小值指令	3
FMAX.S	单精度浮点取最大值指令	3
FEQ.S	单精度浮点比较相等指令	3
FLT.S	单精度浮点比较小于指令	3
FLE.S	单精度浮点比较小于等于指令	3
数据类型转换指令		
FCVT.W.S	单精度浮点转换成有符号整型指令	3
FCVT.WU.S	单精度浮点转换成无符号整型指令	3
FCVT.S.W	有符号整型转换成单精度浮点指令	3
FCVT.S.WU	无符号整型转换成单精度浮点指令	3
FCVT.L.S	单精度浮点转换成有符号长整型指令	3
FCVT.LU.S	单精度浮点转换成无符号长整型指令	3
FCVT.S.L	有符号长整型转换成单精度浮点指令	3
FCVT.S.LU	无符号长整型转换成单精度浮点指令	3
内存存储指令		
FLW	单精度浮点加载指令	Cacheable LOAD: >=2 Cacheable STORE:1 Non-Cacheable 不定周期
FSW	单精度浮点存储指令	同上
浮点数分类指令		
FCLASS.S	单精度浮点分类指令	3

具体指令说明和定义，请参考附录 A-4 F 指令术语。

4.1.5 RV64D 双精度浮点指令集

本节主要介绍双精度浮点指令。

双精度浮点指令集按功能可以分为以下类型：

- 运算指令
- 符号注入指令
- 数据传输指令
- 比较指令
- 数据类型转换指令
- 内存存储指令

表 4.5: 双精度浮点 (RVD) 指令列表

指令名称	指令描述	执行延时
运算指令		
FADD.D	双精度浮点加法指令	4
FSUB.D	双精度浮点减法指令	4
FMUL.D	双精度浮点乘法指令	4
FMADD.D	双精度浮点乘累加指令	5
FMSUB.D	双精度浮点乘累减指令	5
FNMSUB.D	双精度浮点乘累加取反指令	5
FNMADD.D	双精度浮点乘累减取反指令	5
FDIV.D	双精度浮点除法指令	4-31
FSQRT.D	双精度浮点开方指令	4-31
符号注入指令		
FSGNJ.D	双精度浮点符号注入指令	3
FSGNJN.D	双精度浮点符号取反注入指令	3
FSGNJX.D	双精度浮点符号异或注入指令	3
数据传输指令		
FMV.X.D	双精度浮点读传送指令	3
FMV.D.X	双精度浮点写传送指令	3
比较指令		
FMIN.D	双精度浮点取最小值指令	3
FMAX.D	双精度浮点取最大值指令	3
FEQ.D	双精度浮点比较相等指令	3
FLT.D	双精度浮点比较小于指令	3
FLE.D	双精度浮点比较小于等于指令	3
数据类型转换指令		
FCVT.S.D	双精度浮点转换成单精度浮点指令	3
FCVT.D.S	单精度浮点转换成双精度浮点指令	3
FCVT.W.D	双精度浮点转换成有符号整型指令	3
FCVT.WU.D	双精度浮点转换成无符号整型指令	3

下页继续

表 4.5 – 续上页

FCVT.D.W	有符号整型转换成双精度浮点指令	3
FCVT.D.WU	无符号整型转换成双精度浮点指令	3
FCVT.L.D	双精度浮点转换成有符号长整型指令	3
FCVT.LU.D	双精度浮点转换成无符号长整型指令	3
FCVT.D.L	有符号长整型转换成双精度浮点指令	3
FCVT.D.LU	无符号长整型转换成双精度浮点指令	3
内存存储指令		
FLD	双精度浮点加载指令	Cacheable LOAD: >=2 Cacheable STORE: 1 Non-Cacheable 不定周期
FSD	双精度浮点存储指令	同上
浮点数分类指令		
FCLASS.D	双精度浮点分类指令	3

具体指令说明和定义，请参考附录 A-5 D 指令术语

4.1.6 RVC 压缩指令集

本节主要介绍 16 位压缩指令。

压缩指令集按功能可以分为以下类型：

- 加减法指令
- 逻辑操作指令
- 移位指令
- 数据传输指令
- 分支跳转指令
- 立即数偏移存取指令
- 其他特殊指令

表 4.6: 压缩指令 (RVC) 指令列表

指令名称	指令描述	执行延时
加减法指令		
C.ADD	有符号加法指令	1
C.ADDW	低 32 位有符号加法指令	1
C.ADDI	有符号立即数加法指令	1
C.ADDIW	低 32 位有符号立即数加法指令	1
C.SUB	有符号减法压缩指令	1
C.SUBW	低 32 位有符号减法指令	1
C.ADDI16SP	加 16 倍立即数到堆栈指针	1

下页继续

表 4.6 – 续上页

C.ADDI4SPN	4 倍立即数和堆栈指针相加	1
逻辑操作指令		
C.AND	按位与指令	1
C.ANDI	立即数按位与指令	1
C.OR	按位或指令	1
C.XOR	按位异或指令	1
移位指令		
C.SLLI	立即数逻辑左移指令	1
C.SRLI	立即数逻辑右移指令	1
C.SRAI	立即数算术右移指令	1
数据传输指令		
C.MV	数据传送指令	1
C.LI	低位立即数传送指令	1
C.LUI	高位立即数传送指令	1
分支跳转指令		
C.BEQZ	等于零分支指令	1
C.BNEZ	不等于零分支指令	1
C.J	无条件跳转指令	1
C.JR	寄存器跳转指令	1
C.JALR	寄存器跳转子程序指令	1
立即数偏移存取指令		
C.LW	字加载指令	Cacheable LOAD: ≥ 2 Cacheable STORE: 1 Non-Cacheable 不定周期
C.SW	字存储指令	同上
C.LWSP	字堆栈加载指令	同上
C.SWSP	字堆栈存储指令	同上
C.LD	双字加载指令	同上
C.SD	双字存储指令	同上
C.LDSP	双字堆栈加载指令	同上
C.SDSP	双字堆栈存储指令	同上
C.FLD	双精度加载指令	同上
C.FSD	双精度存储指令	同上
C.FLDSP	双精度堆栈存储指令	同上
C.FSDSP	双精度堆栈加载指令	同上
特殊指令		
C.NOP	空指令	1
C.EBREAK	断点指令	1

具体指令说明和定义，请参考附录 A-6 C 指令术语

4.1.7 RVV 矢量指令集

本节介绍了矢量指令集，矢量指令集按功能可以分为以下类型：

- 矢量控制指令
- 矢量 MISC 指令
- 矢量缩减指令
- 矢量乘法乘累加指令
- 矢量移位指令
- 矢量整型加减法指令
- 矢量整型比较指令
- 矢量整型最大值/最小值指令
- 矢量整型除法/取余指令
- 矢量元素排序操作指令
- 矢量整型定点加减法指令
- 浮点运算指令
- 矢量浮点比较指令
- 浮点传送指令
- 浮点数据类型转换指令
- 浮点缩减指令
- 浮点分类指令
- 矢量加载存储指令
- 矢量原子指令
- 矢量 SEGMENT 指令

表 4.7: 矢量 (RVV) 指令列表

指令名称	指令描述	执行延时 (LMUL=1)
矢量控制指令		
VSETVL	寄存器设置 VL 和 VTYPE 指令	1
VSETVLI	立即数设置 VL 和 VTYPE 指令	
矢量 MISC 指令		
VAND.VV	矢量按位与指令	3
VAND.VI	矢量立即数按位与指令	
VOR.VV	矢量按位或指令	
VOR.VI	矢量立即数按位或指令	
VXOR.VV	矢量按位异或指令	
VXOR.VI	矢量立即数按位异或指令	
VMERGE.VVM	矢量元素选择指令	
VMERGE.VXM	矢量标量元素选择指令	

下页继续

表 4.7 – 续上页

指令名称	指令描述	执行延时 (LMUL=1)	
VMERGE.VIM	矢量立即数元素选择指令	3	
VMV.V.V	矢量元素传送指令		
VMV.V.X	整型传送至矢量指令		
VMV.V.I	立即数传送至矢量指令		
VMAND.MM	矢量掩码运算与指令		
VMNAND.MM	矢量掩码运算与非指令		
VMANDNOT.MM	矢量掩码运算非与指令		
VMXOR.MM	矢量掩码运算异或指令	3	
VMOR.MM	矢量掩码运算或指令		
VMNOR.MM	矢量掩码运算或非指令		
VMORNOT.MM	矢量掩码运算非或指令		
VMXNOR.MM	矢量掩码运算同或指令		
VMPOPC.M	矢量运算元素个数统计指令		
VMFIRST.M	矢量掩码运算快速查找 1 指令		
VMSBF.M	矢量掩码找 1 置位不包含指令		
VMSIF.M	矢量掩码找 1 置位包含指令		
VMSOF.M	矢量掩码找 1 单独置位指令		
VIOTA.M	矢量 active 元素统计指令		
VID.V	矢量 active 元素 index 求解指令		
VAND.VX	矢量标量按位与指令		
VOR.VX	矢量标量按位或指令		
VXOR.VX	矢量标量按位异或指令		
矢量缩减指令			
VREDSUM.VS	矢量缩减累加指令		4
VREDMAXU.VS	矢量缩减无符号最大值指令		
VREDMAX.VS	矢量缩减有符号最大值指令		
VREDMINU.VS	矢量缩减无符号最小值指令		
VREDMIN.VS	矢量缩减有符号最小值指令		
VREDAND.VS	矢量缩减按位与指令	3	
VREDOR.VS	矢量缩减按位或指令		
VREDXOR.VS	矢量缩减按位异或指令		
VWREDSUMU.VS	矢量缩减无符号扩位累加指令	4	
VWREDSUM.VS	矢量缩减有符号扩位累加指令		
矢量乘法乘累加指令			
VMUL.VV	矢量整型乘法取低位指令	sew<=16 时: 3 sew>16 时: 4	
VMULH.VV	矢量整型有符号乘法取高位指令		
VMULHU.VV	矢量整型无符号乘法取高位指令		
VMULHSU.VV	矢量整型有符号乘无符号取高位指令		
VSMUL.VV	矢量带饱和乘法指令		
VWMULU.VV	矢量无符号扩位乘法指令		
VWMUL.VV	矢量有符号扩位乘法指令		
VWMULSU.VV	矢量有符号无符号扩位乘法指令		

下页继续

表 4.7 – 续上页

指令名称	指令描述	执行延时 (LMUL=1)
VMACC.VV	覆盖加数矢量低位乘累加指令	sew<=16 时: 3 sew>16 时: 4
VNMSAC.VV	覆盖被减数矢量低位乘累减取负指令	
VMADD.VV	覆盖被乘数矢量低位乘累加指令	
VNMSUB.VV	覆盖被乘数矢量低位乘累减取负指令	
VWMACCU.VV	覆盖加数矢量无符号扩位乘累加指令	
VWMACC.VV	覆盖加数矢量有符号扩位乘累加指令	sew<=16 时: 3 sew>16 时: 4
VWMACCSU.VV	覆盖加数矢量有符号无符号扩位乘累加指令	
VWMACCUS.VV	覆盖加数矢量标量无符号有符号扩位乘累加指令	
VWSMACCU.VV	矢量无符号带饱和和扩位乘累加指令	
VWSMACC.VV	矢量有符号带饱和和扩位乘累加指令	
VWSMACCSU.VV	矢量带饱和有符号无符号扩位乘累减取负指令	sew<=16 时: 3 sew>16 时: 4
VWSMACCUS.VX	矢量标量带饱和有符号无符号扩位乘累减取负指令	
VMUL.VX	矢量标量整型乘法取低位指令	
VMULH.VX	矢量标量有符号乘法取高位指令	
VMULHU.VX	矢量标量无符号乘法取高位指令	
VMULHSU.VX	矢量标量有符号无符号乘法取高位指令	
VWMULU.VX	矢量标量无符号扩位乘法指令	
VWMUL.VX	矢量标量有符号扩位乘法指令	
VWMULSU.VX	矢量标量有符号无符号扩位指令	
VSMUL.VX	矢量标量带饱和乘法指令	
VMACC.VX	覆盖加数矢量标量低位乘累加指令	
VNMSAC.VX	覆盖被减数矢量标量低位乘累减取负指令	
VMADD.VX	覆盖被乘数矢量标量低位乘累加指令	
VNMSUB.VX	覆盖被乘数矢量标量低位乘累减取负指令	
VWMACCU.VX	矢量标量无符号乘累加指令	
VWMACC.VX	矢量标量有符号乘累加指令	
VWMACCSU.VX	矢量标量有符号无符号乘累加指令	
VWMACCUS.VX	矢量标量无符号有符号乘累加指令	
VWSMACCU.VX	矢量标量无符号带饱和乘累加指令	
VWSMACC.VX	矢量标量有符号带饱和乘累加指令	
VWSMACCSU.VX	矢量标量带饱和有符号无符号扩位乘累减取负指令	
VWSMACCUS.VX	矢量标量带饱和有符号无符号扩位乘累减取负指令	
矢量移位指令		
VSLL.VV	矢量逻辑左移指令	3
VSLL.VI	矢量按立即数逻辑左移指令	
VSRL.VV	矢量逻辑右移指令	
VSRL.VI	矢量按立即数逻辑右移指令	

下页继续

表 4.7 – 续上页

指令名称	指令描述	执行延时 (LMUL=1)
VSRA.VX	矢量算术右移指令	
VSRA.VI	矢量按立即数算数右移指令	
VNSRL.VV	矢量缩位逻辑右移指令	
VNSRL.VI	矢量按立即数缩位逻辑右移指令	
VNSRA.VV	矢量缩位算术右移指令	
VNSRA.VI	矢量按立即数缩位算术右移指令	
VSSRL.VV	矢量带近似逻辑右移指令	
VSSRL.VI	矢量带近似按立即数逻辑右移指令	
VSSRA.VV	矢量带近似算术右移指令	
VSSRA.VI	矢量带近似按立即数算术右移指令	
VNCLIPU.VV	矢量带饱和和无符号缩位算术右移指令	
VNCLIPU.VI	矢量带饱和和按立即数无符号缩位算术右移指令	
VNCLIP.VV	矢量带饱和和有符号缩位算术右移指令	
VNCLIP.VI	矢量带饱和和有符号按立即数缩位算术右移指令	
VSLL.VX	矢量按标量逻辑左移指令	
VSRL.VX	矢量按标量逻辑右移指令	
VSRA.VX	矢量按标量算术右移指令	
VNSRL.VX	矢量按标量缩位逻辑右移指令	
VNSRA.VX	矢量按标量缩位算术右移指令	
VSSRL.VX	矢量带近似按标量逻辑右移指令	
VSSRA.VX	矢量带近似按标量算术右移指令	
VNCLIPU.VX	矢量带饱和和无符号按标量缩位算术右移指令	
VNCLIP.VX	矢量带饱和和有符号按标量缩位算术右移指令	
矢量整型加减法指令		
VADD.VV	矢量整型加法指令	3
VADD.VI	矢量立即数整型加法指令	
VSUB.VV	矢量整型减法指令	
VRSUB.VI	立即数矢量整型减法指令	
VADC.VIM	矢量立即数整型带进位加法指令	
VMADC.VIM	矢量立即数整型带进位加法取进位指令	
VWADDU.VV	矢量整型无符号扩位加法指令	
VWADD.VV	矢量整型有符号扩位加法指令	
VWSUBU.VV	矢量整型无符号扩位减法指令	3
VWSUB.VV	矢量整型有符号扩位减法指令	
VWADDU.WV	扩位矢量整型无符号扩位加法指令	
VWADD.WV	扩位矢量整型有符号扩位加法指令	
VWSUBU.WV	扩位矢量整型无符号扩位减法指令	
VWSUB.WV	扩位矢量整型有符号扩位减法指令	
VADC.VVM	矢量整型带进位加法指令	
VMADC.VV	矢量整型带进位加法取进位指令	

下页继续

表 4.7 – 续上页

指令名称	指令描述	执行延时 (LMUL=1)
VSBC.VVM	矢量整型带进位减法指令	
VMSBC.VVM	矢量整型带进位减法取借位指令	
VADD.VX	矢量整型加法指令	
VSUB.VX	矢量整型减法指令	
VRSUB.VX	矢量整型反减法指令	
VWADDU.VX	矢量整型无符号扩位加法指令	
VWADD.VX	矢量整型有符号扩位加法指令	
VWSUBU.VX	矢量整型无符号扩位减法指令	
VWSUB.VX	矢量整型有符号扩位减法指令	
VWADDU.WX	扩位矢量整型无符号扩位加法指令	
VWADD.WX	扩位矢量整型有符号扩位加法指令	
VWSUBU.WX	扩位矢量整型无符号扩位减法指令	
VWSUB.WX	扩位矢量整型有符号扩位减法指令	
VADC.VXM	矢量整型带进位加法指令	
VMADC.VXM	矢量整型带进位加法取进位指令	
VSBC.VXM	矢量整型带进位减法指令	
VMSBC.VXM	矢量整型带进位减法取借位指令	
矢量整型比较指令		
VMSEQ.VX	矢量标量整型比较相等指令	3
VMSNE.VX	矢量标量整型比较不等指令	
VMSLTU.VX	矢量标量整型无符号比较小于指令	
VMSLT.VX	矢量标量整型有符号比较小于指令	
VMSLEU.VX	矢量整型无符号比较小于等于指令	
VMSLE.VX	矢量标量整型有符号比较小于等于指令	
VMSGTU.VX	矢量标量整型无符号比较大于指令	
VMSGT.VX	矢量标量整型有符号比较大于指令	
VMSEQ.VV	矢量整型比较相等指令	
VMSEQ.VI	矢量立即数整型比较相等指令	
VMSNE.VV	矢量整型比较不等指令	
VMSNE.VI	矢量立即数整型比较不等指令	
VMSLTU.VV	矢量整型无符号比较小于指令	
VMSLT.VV	矢量整型有符号比较小于指令	
VMSLEU.VV	矢量整型无符号比较小于等于指令	3
VMSLEU.VI	矢量立即数整型无符号比较小于等于指令	
VMSLE.VV	矢量整型有符号比较小于等于指令	
VMSLE.VI	矢量立即数整型有符号比较小于等于指令	
VMSGT.VI	矢量立即数整型有符号比较大于指令	
VMSGTU.VI	矢量立即数整型无符号比较大于指令	
矢量整型最大值/最小值指令		
VMINU.VV	矢量整型无符号取最小值指令	3
VMIN.VV	矢量整型有符号取最小值指令	
VMAXU.VV	矢量整型无符号取最大值指令	

下页继续

表 4.7 – 续上页

指令名称	指令描述	执行延时 (LMUL=1)
VMAX.VV	矢量整型有符号取最大值指令	
VMINU.VX	矢量标量整型无符号取最小值指令	
VMIN.VX	矢量标量整型有符号取最小值指令	
VMAXU.VX	矢量标量整型无符号取最大值指令	
VMAX.VX	矢量标量整型有符号取最大值指令	
矢量整型除法/取余指令		
VDIVU.VV	矢量整型无符号除法指令	3-21
VDIV.VV	矢量整型有符号除法指令	
VREMU.VV	矢量整型无符号取余指令	
VREM.VV	矢量整型有符号取余指令	
VDIVU.VX	矢量标量整型无符号除法指令	
VDIV.VX	矢量标量整型有符号除法指令	
VREMU.VX	矢量标量整型无符号取余指令	
VREM.VX	矢量标量整型有符号取余指令	
矢量元素排序操作指令		
VEXT.X.V	矢量整型提取元素指令	3
VMV.S.X	整型传送至矢量首元素指令	
VSLIDEUP.VX	矢量元素按标量向上偏移指令	
VSLIDEDOWN.VX	矢量元素按立即数向下偏移指令	
VSLIDE1UP.VX	矢量整型元素向上偏移 1 个索引指令	
VSLIDE1DOWN.VX	矢量整型元素向下偏移 1 个索引指令	
VRGATHER.VX	矢量标量整型元素根据索引聚合指令	
VSLIDEUP.VI	矢量元素按立即数向上偏移指令	
VSLIDEDOWN.VI	矢量元素按标量向下偏移指令	
VRGATHER.VI	矢量立即数整型元素根据索引聚合指令	
VRGATHER.VV	矢量整型元素根据索引聚合指令	
VCOMPRESS.VM	矢量整型元素压缩指令	
矢量整型定点加减法指令		
VSADDU.VX	矢量标量整型加法无符号取饱和指令	3
VSSUBU.VX	矢量标量整型减法无符号取饱和指令	
VSSUB.VX	矢量标量整型减法有符号取饱和指令	3
VAADD.VX	矢量标量整型加法取平均数指令	
VASUB.VX	矢量标量整型减法取平均数指令	
VSADD.VV	矢量整型加法有符号取饱和指令	
VSADDU.VI	矢量立即数整型加法无符号取饱和指令	
VSSUBU.VV	矢量整型减法无符号取饱和指令	
VSSUB.VV	矢量整型减法有符号取饱和指令	
VAADD.VV	矢量整型加法取平均数指令	
VAADD.VI	矢量立即数整型加法取平均数指令	
VASUB.VV	矢量整型减法取平均数指令	
浮点运算指令		
VFADD.VV	矢量浮点加法指令	3

下页继续

表 4.7 – 续上页

指令名称	指令描述	执行延时 (LMUL=1)
VFSUB.VV	矢量浮点减法指令	
VFWADD.VV	矢量浮点扩位加法指令	
VFWSUB.VV	矢量浮点扩位减法指令	
VFADD.VF	矢量标量浮点加法指令	
VFSUB.VF	矢量标量浮点减法指令	
VFRSUB.VF	标量矢量浮点减法指令	
VFWADD.VF	矢量标量浮点扩位加法指令	
VFWSUB.VF	矢量标量浮点扩位减法指令	
VFWADD.WV	扩位矢量浮点扩位加法指令	
VFWADD.WF	扩位矢量标量浮点扩位加法指令	
VFWSUB.WV	扩位矢量浮点扩位减法指令	
VFWSUB.WF	扩位矢量标量浮点扩位减法指令	
VFMUL.VV	矢量浮点乘法指令	
VFMUL.VF	矢量标量浮点乘法指令	
VFDIV.VV	矢量浮点除法指令	4-17
VFDIV.VF	矢量标量浮点除法指令	
VFRDIV.VF	标量矢量浮点除法指令	
VFSQRT.V	矢量浮点开方指令	
VFWMUL.VV	矢量浮点扩位乘法指令	3
VFWMUL.VF	矢量标量浮点扩位乘法指令	3
VFMACC.VV	覆盖加数矢量浮点乘累加指令	4
VFNMACC.VV	覆盖减数矢量浮点乘累加取负指令	
VFMSAC.VV	覆盖减数矢量浮点乘累减指令	
VFNMSAC.VV	覆盖被减数矢量浮点乘累减取负指令	
VFMADD.VV	覆盖被乘数矢量浮点乘累加指令	
VFNMADD.VV	覆盖被乘数矢量浮点乘累加取负指令	
VFMSUB.VV	覆盖被乘数矢量浮点乘累减指令	
VFNMSUB.VV	覆盖被乘数矢量浮点乘累减取负指令	
VFMACC.VF	覆盖加数矢量标量浮点乘累加指令	3
VFNMACC.VF	覆盖减数矢量标量浮点乘累加取负指令	
VFMSAC.VF	覆盖减数矢量标量浮点乘累减指令	
VFNMSAC.VF	覆盖被减数矢量标量浮点乘累减取负指令	
VFMADD.VF	覆盖被乘数矢量标量浮点乘累加指令	
VFNMADD.VF	覆盖被乘数矢量标量浮点乘累加取负指令	
VFMSUB.VF	覆盖被乘数矢量标量浮点乘累减指令	
VFNMSUB.VF	覆盖被乘数矢量标量浮点乘累减取负指令	
VFWMACC.VV	覆盖加数矢量浮点扩位乘累加指令	
VFWMACC.VF	扩位矢量标量覆盖加数浮点乘累加指令	
VFWNMACC.VV	扩位矢量矢量覆盖加数浮点乘累加取负指令	
VFWNMACC.VF	扩位矢量标量覆盖加数浮点乘累加取负指令	
VFWMSAC.VV	扩位矢量矢量覆盖加数浮点乘累减指令	
VFWMSAC.VF	扩位矢量标量覆盖加数浮点乘累减指令	

下页继续

表 4.7 – 续上页

指令名称	指令描述	执行延时 (LMUL=1)
VFWNMSAC.VV	扩位矢量矢量覆盖加数浮点乘累减取负指令	
VFWNMSAC.VF	扩位矢量标量覆盖加数浮点乘累减取负指令	
矢量浮点比较指令		
VFMIN.VV	矢量浮点取最大值指令	3
VFMAX.VV	矢量浮点取最小值指令	
VMFEQ.VV	矢量浮点比较相等指令	
VMFNE.VV	矢量浮点比较不等指令	
VMFLT.VV	矢量浮点比较小于指令	
VMFLE.VV	矢量浮点比较小于等于指令	
VMFORD.VV	矢量浮点 NaN 判断指令	
VFMIN.VF	矢量标量浮点取最大值指令	3
VFMAX.VF	矢量标量浮点取最小值指令	
VMFEQ.VF	矢量标量浮点比较相等指令	
VMFNE.VF	矢量标量浮点比较不等指令	
VMFLT.VF	矢量标量浮点比较小于指令	
VMFLE.VF	矢量标量浮点比较小于等于指令	
VMFGT.VF	矢量标量浮点比较大于指令	
VMFGE.VF	矢量标量浮点比较大于等于指令	
VMFORD.VF	矢量标量浮点 NaN 判断指令	
浮点传送指令		
VFMERGE.VFM	矢量浮点元素选择指令	3
VFMV.V.F	浮点传送至矢量指令	
VFMV.F.S	矢量首元素传送至浮点指令	
VFMV.S.F	浮点传送至矢量首元素指令	
浮点数据类型转换指令		
VFCVT.XU.F.V	矢量浮点转换成无符号整型指令	3
VFCVT.X.F.V	矢量浮点转换成有符号整型指令	
VFCVT.F.XU.V	矢量无符号整型转换成浮点指令	
VFCVT.F.X.V	矢量有符号整型转换成浮点指令	
VFWCVT.XU.F.V	矢量浮点扩位转换成无符号整型指令	3
VFWCVT.X.F.V	矢量浮点扩位转换成有符号整型指令	
VFWCVT.F.XU.V	矢量无符号整型扩位转换成浮点指令	
VFWCVT.F.X.V	矢量有符号整型扩位转换成浮点指令	
VFWCVT.F.F.V	矢量浮点扩位转换成浮点指令	
VFNCVT.XU.F.V	矢量浮点缩位转换成无符号整型指令	
VFNCVT.X.F.V	矢量浮点缩位转换成有符号整型指令	
VFNCVT.F.XU.V	矢量无符号整型缩位转换成浮点指令	
VFNCVT.F.X.V	矢量有符号整型缩位转换成浮点指令	
VFNCVT.F.F.V	矢量浮点缩位转换成浮点指令	
浮点缩减指令		
VFREDOSUM.VS	矢量浮点缩减顺序求和指令	拆分执行 3* (VLEN/SEW)

下页继续

表 4.7 – 续上页

指令名称	指令描述	执行延时 (LMUL=1)
VFREDSUM.VS	矢量浮点缩减乱序求和指令	拆分执行 $3 * \log_2(VLEN/SEW)$
VFREDMAX.VS	矢量浮点缩减取最大值指令	
VFREDMIN.VS	矢量浮点缩减取最小值指令	
VFWREDOSUM.VS	矢量浮点缩减顺序扩位求和指令	拆分执行 $3 * (VLEN/SEW)$
VFWREDSUM.VS	矢量浮点缩减乱序扩位求和指令	拆分执行 $3 * \log_2(VLEN/SEW)$
浮点分类指令		
VFCLASS.V	矢量浮点分类指令	3
矢量加载存储指令		
VLB.V	矢量有符号字节加载指令	LOAD: ≥ 2 STORE: ≥ 1
VLH.V	矢量有符号半字加载指令	
VLW.V	矢量有符号节加载指令	
VLBU.V	矢量无符号字节加载指令	
VLHU.V	矢量无符号半字加载指令	
VLWU.V	矢量无符号字加载指令	
VLE.V	矢量元素加载指令	
VSBU.V	矢量字节存储指令	
VSH.V	矢量半字存储指令	
VSW.V	矢量字存储指令	
VSE.V	矢量元素存储指令	
VLSB.V	矢量步进有符号字节加载指令	
VLSH.V	矢量步进有符号半字加载指令	
VLSW.V	矢量步进有符号节加载指令	
VLSBU.V	矢量步进无符号字节加载指令	
VLSHU.V	矢量步进无符号半字加载指令	
VLSWU.V	矢量步进无符号字加载指令	
VLSE.V	矢量步进元素加载指令	
VSSB.V	矢量步进字节存储指令	
VSSH.V	矢量步进半字存储指令	
VSSHW.V	矢量步进字存储指令	
VSSE.V	矢量步进元素存储指令	
VLXB.V	矢量索引有符号字节加载指令	
VLXH.V	矢量索引有符号半字加载指令	
VLXW.V	矢量索引有符号字加载指令	
VLXBU.V	矢量索引无符号字节加载指令	
VLXHU.V	矢量索引无符号半字加载指令	
VLXWU.V	矢量索引无符号字加载指令	
VLXE.V	矢量索引元素加载指令	
VSXB.V	矢量索引按序字节存储指令	
V SXH.V	矢量索引按序半字存储指令	
V SXW.V	矢量索引按序字存储指令	

下页继续

表 4.7 – 续上页

指令名称	指令描述	执行延时 (LMUL=1)	
VSXE.V	矢量索引按序元素存储指令		
VSUXB.V	矢量索引乱序字节存储指令		
VSUXH.V	矢量索引乱序半字存储指令		
VSUXW.V	矢量索引乱序字存储指令		
VSUXE.V	矢量索引乱序元素存储指令		
VLBFF.V	矢量 F OF 有符号字节加载指令	LOAD: ≥ 2 STORE: ≥ 1	
VLHFF.V	矢量 F OF 有符号半字加载指令		
VLWFF.V	矢量 FOF 有符号字加载指令		
VLBUFF.V	矢量 F OF 无符号字节加载指令		
VLHUFF.V	矢量 F OF 无符号半字加载指令		
VLWUFF.V	矢量 FOF 无符号字加载指令		
VLEFF.V	矢量 FOF 元素加载指令		
矢量 SEGMENT 指令			
VLSEG<NF>B.V	矢量 SEGMENT 有符号字节加载指令	Unit stride 拆分执行 $\geq 2+nf$ 其他拆分执行 $\geq VLEN/SEW$	
VLSEG<NF>BFF.V	矢量 FOF SEGMENT 有符号字节加载指令		
VLSEG<NF>BU.V	矢量 SEGMENT 无符号字节加载指令		
VLSEG<NF>BUFF.V	矢量 FOF SEGMENT 无符号字节加载指令		
VLSEG<NF>E.V	矢量 SEGMENT 元素加载指令		
VLSEG<NF>EFF.V	矢量 FOF SEGMENT 元素加载指令		
VLSEG<NF>H.V	矢量 SEGMENT 有符号半字加载指令		
VLSEG<NF>HFF.V	矢量 FOF SEGMENT 有符号半字加载指令		
VLSEG<NF>HU.V	矢量 SEGMENT 无符号半字加载指令		
VLSEG<NF>HUFF.V	矢量 FOF SEGMENT 无符号半字加载指令		
VLSEG<NF>W.V	矢量 SEGMENT 有符号字加载指令		
VLSEG<NF>WFF.V	矢量 FOF SEGMENT 有符号字加载指令		
VLSEG<NF>WU.V	矢量 SEGMENT 无符号字加载指令		
VLSEG<NF>WUFF.V	矢量 FOF SEGMENT 无符号字加载指令		
VLSSEG<NF>B.V	矢量步进 SEGMENT 有符号字节加载指令		
VLSSEG<NF>BU.V	矢量步进 SEGMENT 无符号字节加载指令		
VLSSEG<NF>E.V	矢量步进 SEGMENT 元素加载指令		
VLSSEG<NF>H.V	矢量步进 SEGMENT 有符号半字加载指令		
VLSSEG<NF>HU.V	矢量步进 SEGMENT 无符号半字加载指令		
VLSSEG<NF>W.V	矢量步进 SEGMENT 有符号字加载指令		
VLSSEG<NF>WU.V	矢量步进 SEGMENT 无符号字加载指令		
VLXSEG<NF>B.V	矢量索引 SEGMENT 有符号字节加载指令		
VLXSEG<NF>BU.V	矢量索引 SEGMENT 无符号字节加载指令		
VLXSEG<NF>E.V	矢量索引 SEGMENT 元素加载指令		
VLXSEG<NF>H.V	矢量索引 SEGMENT 有符号半字加载指令		
VLXSEG<NF>HU.V	矢量索引 SEGMENT 无符号半字加载指令		
VLXSEG<NF>W.V	矢量索引 SEGMENT 有符号字加载指令		
VLXSEG<NF>WU.V	矢量索引 SEGMENT 无符号字加载指令		
VSSEG<NF>B.V	矢量 SEGMENT 字节存储指令		Unit stride 拆分执行 $\geq 1+nf$ 其他拆分执行 $\geq VLEN/SEW$

表 4.7 – 续上页

指令名称	指令描述	执行延时 (LMUL=1)
VSSEG<NF>E.V	矢量 SEGMENT 元素存储指令	
VSSEG<NF>H.V	矢量 SEGMENT 半字存储指令	
VSSEG<NF>W.V	矢量 SEGMENT 字存储指令	
VSSSEG<NF>B.V	矢量步进 SEGMENT 字节存储指令	
VSSSEG<NF>E.V	矢量步进 SEGMENT 元素存储指令	
VSSSEG<NF>H.V	矢量步进 SEGMENT 半字存储指令	
VSSSEG<NF>W.V	矢量步进 SEGMENT 字存储指令	
VSXSEG<NF>B.V	矢量索引 SEGMENT 字节存储指令	
VSXSEG<NF>E.V	矢量索引 SEGMENT 元素存储指令	
VSXSEG<NF>H.V	矢量索引 SEGMENT 半字存储指令	
VSXSEG<NF>W.V	矢量索引 SEGMENT 字存储指令	

具体指令说明和定义，请参考附录 A-7 V 指令术语

4.2 平头哥扩展指令集

4.2.1 Cache 指令子集

表 4.8: Cache 指令列表

Cache 指令子集	指令描述	执行延时
DCACHE.CALL	DCACHE 清全部脏表项指令	不定周期
DCACHE.CVA	DCACHE 指定虚拟地址清脏表项指令	不定周期
DCACHE.CPA	DCACHE 指定物理地址清脏表项指令	不定周期
DCACHE.CSW	DCACHE 指定 way/set 清脏表项指令	不定周期
DCACHE.IALL	DCACHE 无效全部表项指令	不定周期
DCACHE.IVA	DCACHE 指定虚拟地址无效表项指令	不定周期
DCACHE.IPA	DCACHE 指定物理地址无效表项指令	不定周期
DCACHE.ISW	DCACHE 指定 way/set 无效表项指令	不定周期
DCACHE.CIALL	DCACHE 清全部脏表项并无效指令	不定周期
DCACHE.CIVA	DCACHE 指定虚拟地址清脏表项并无效指令	不定周期
DCACHE.CIPA	DCACHE 指定物理地址清脏表项并无效指令	不定周期
DCACHE.CISW	DCACHE 指定 way/set 清脏表项并无效指令	不定周期
ICACHE.IALL	ICACHE 无效全部表项指令	不定周期
ICACHE.IALLS	ICACHE 广播无效全部表项指令	不定周期
ICACHE.IVA	ICACHE 指定虚拟地址无效表项指令	不定周期
ICACHE.IPA	ICACHE 指定物理地址无效表项指令	不定周期

具体指令说明和定义，请参考附录 B-1 Cache 指令术语。

4.2.2 同步指令子集

表 4.9: 同步指令子集

Cache 指令子集	指令描述	执行延时
SYNC	同步指令	不定周期
SYNC.I	同步并清空指令	不定周期

具体指令说明和定义，请参考附录 B-2 同步指令术语。

4.2.3 算术运算指令子集

表 4.10: 算术运算指令集

指令名称	指令描述	执行延时
加减法指令		
ADDSL	寄存器移位相加指令	1
MULA	乘累加指令	3/6
MULS	乘累减指令	3/6
MULAW	低 32 位乘累加指令	3
MULSW	低 32 位乘累减指令	3
MULAH	低 16 位乘累加指令	3
MULSH	低 16 位成累减指令	3
位移指令		
SRRI	循环右移指令	1
SRRIW	低 32 位循环右移指令	1
传送指令		
MVEQZ	寄存器为 0 传送指令	1
MVNEZ	寄存器非 0 传送指令	1

具体指令说明和定义，请参考附录 B-3 算术运算指令术语。

4.2.4 位操作指令子集

表 4.11: 位操指令集

指令名称	指令描述	执行延时
位操作指令		
TST	比特为 0 测试指令	1
TSTNBZ	字节为 0 测试指令	1
REV	字节倒序指令	1
REVV	低 32 位字节倒序指令	1
FF0	快速找 0 指令	1
FF1	快速找 1 指令	1
EXT	寄存器连续位提取符号位扩展指令	1
EXTU	寄存器连续位提取零扩展指令	1

具体指令说明和定义，请参考附录 B-3 位操作指令术语。

4.2.5 存储指令子集

表 4.12: 存储指令集

存储指令	执行延时	
FLRD	浮点寄存器移位双字加载指令	Cacheable LOAD ≥ 2 NonCacheable 不定周期
FLRW	浮点寄存器移位字加载指令	
FLURD	浮点寄存器低 32 位移位双字加载指令	
FLURW	浮点寄存器低 32 位移位字加载指令	
LRB	寄存器移位符号位扩展字节加载指令	
LRH	寄存器移位符号位扩展半字加载指令	
LRW	寄存器移位符号位扩展字加载指令	
LRD	寄存器移位双字加载指令	
LRBU	寄存器移位零扩展字节加载指令	
LRHU	寄存器移位零扩展半字加载指令	
LRWU	寄存器移位零扩展字加载指令	
LURB	寄存器低 32 位移位符号位扩展字节加载指令	
LURH	寄存器低 32 位移位符号位扩展半字加载指令	
LURW	寄存器低 32 位移位符号位扩展字加载指令	
LURD	寄存器低 32 位移位双字加载指令	
LURBU	寄存器低 32 位移位零扩展字节加载指令	
LURHU	寄存器低 32 位移位零扩展半字加载指令	
LURWU	寄存器低 32 位移位零扩展字加载指令	
LBIA	符号位扩展字节加载基地址自增指令	
LBIB	基地址自增符号位扩展字节加载指令	

下页继续

表 4.12 – 续上页

LHIA	符号位扩展半字加载基地址自增指令	
LHIB	基地址自增符号位扩展半字加载指令	
LWIA	符号位扩展字加载基地址自增指令	
LWIB	基地址自增符号位扩展字加载指令	
LDIA	符号位扩展双字加载基地址自增指令	
LDIB	基地址自增符号位扩展双字加载指令	
LBUIA	零扩展字节加载基地址自增指令	
LBUIB	基地址自增零扩展字节加载指令	
LHUIA	零扩展半字加载地址自增指令	
LHUIB	基地址自增零扩展半字加载指令	
LWUIA	零拓展字加载地址自增指令	
LWUIB	基地址自增零扩展字加载指令	
LDD	双寄存器加载指令	
LWD	符号位扩展双寄存器字加载指令	Cacheable1 Noncacheable 不定周期
LWUD	零扩展双寄存器字加载指令	
FSRD	浮点寄存器移位双字存储指令	
FSRW	浮点寄存器移位字存储指令	
FSURD	浮点寄存器低 32 位移位双字存储指令	
FSURW	浮点寄存器低 32 位移位字存储指令	
SRB	寄存器移位字节存储指令	
SRH	寄存器移位半字存储指令	
SRW	寄存器移位字存储指令	
SRD	寄存器移位双字存储指令	
SURB	寄存器低 32 位移位字节存储指令	
SURH	寄存器低 32 位移位半字存储指令	
SURW	寄存器低 32 位移位字存储指令	
SURD	寄存器低 32 位移位双字存储指令	
SBIA	字节存储基地址自增指令	
SBIB	基地址自增字节存储指令	
SHIA	半字存储基地址自增指令	
SHIB	基地址自增半字存储指令	
SWIA	字存储基地址自增指令	
SWIB	基地址自增字存储指令	
SDIA	双字存储基地址自增指令	
SDIB	基地址自增双字存储指令	
SDD	双寄存器存储指令	拆分为两条 store 执行
SWD	双寄存器低 32 位存储指令	

具体指令说明和定义，请参考附录 B-4 存储指令术语。

4.2.6 半精度浮点指令子集

表 4.13: 半精度浮点指令集

指令名称	指令描述	执行延时
运算指令		
FADD.H	半精度浮点加法指令	3
FSUB.H	半精度浮点减法指令	3
FMUL.H	半精度浮点乘法指令	3
FMADD.H	半精度浮点乘累加指令	4
FMSUB.H	半精度浮点乘累减指令	4
FNMADD.H	半精度浮点乘累加取负指令	4
FNMSUB.H	半精度浮点乘累减取负指令	4
FDIV.H	半精度浮点除法指令	4-11
FSQRT.H	半精度浮点开方指令	4-11
符号注入指令		
FSGNJ.H	半精度浮点符号注入指令	3
FSGNJN.H	半精度浮点符号取反注入指令	3
FSGNJX.H	半精度浮点符号异或注入指令	3
数据传输指令		
FMV.X.H	半精度浮点读传送指令	3
FMV.H.X	半精度浮点写传送指令	3
比较指令		
FMIN.H	半精度浮点取最小值指令	3
FMAX.H	半精度浮点取最大值指令	3
FEQ.H	半精度浮点比较相等指令	3
FLT.H	半精度浮点比较小于指令	3
FLE.H	半精度浮点比较小于等于指令	3
数据类型转换指令		
FCVT.S.H	半精度浮点转换成单精度浮点指令	3
FCVT.H.S	单精度浮点转换成半精度浮点指令	3
FCVT.D.H	半精度浮点转换成双精度浮点指令	3
FCVT.H.D	双精度浮点转换成半精度浮点指令	3
FCVT.W.H	半精度浮点转换成有符号整型指令	3
FCVT.WU.H	半精度浮点转换成无符号整型指令	3
FCVT.H.W	有符号整型转换成半精度浮点指令	3
FCVT.H.WU	无符号整型转换成半精度浮点指令	3
FCVT.L.H	半精度浮点转换成有符号长整型指令	3
FCVT.LU.H	半精度浮点转换成无符号长整型指令	3
FCVT.H.L	有符号长整型转换成半精度浮点指令	3
FCVT.H.LU	无符号长整型转换成半精度浮点指令	3
内存存储指令		
FLH	半精度浮点加载指令	Cacheable LOAD: ≥ 2 Cacheable STORE: 1 Non-Cacheable 不定周期

下页继续

表 4.13 – 续上页

FSH	半精度浮点存储指令	同上
浮点数分类指令		
FCLASS.H	单精度浮点分类指令	3

具体指令说明和定义，请参考附录 B-5 半精度浮点指令术语。

平头哥半身体

第五章 虚拟内存管理

5.1 MMU 概览

C906 虚拟内存管理 MMU (Memory Management Unit) 兼容 RISC-V Sv39 虚拟内存系统, 其功能主要有: 地址转换、页面保护和页面属性管理。

- **地址转换**: 将虚拟地址转换成物理地址, 39 位虚拟地址转换为 40 位物理地址。
- **页面保护**: 通过对页面的访问者进行读写执行权限检查来进行保护。
- **页面属性管理**: 扩展地址属性位, 根据访问地址, 获取页面对应属性, 供系统进一步使用。

MMU 主要利用 TLB (Translation Look-aside Buffer) 来实现上述功能。TLB 将 CPU 访存所使用的虚拟地址作为输入, 转换前检查 TLB 中的页面属性, 再输出该虚拟地址所对应的物理地址。MMU 采用两级 TLB, 第一级为 uTLB(分 I-uTLB 和 D-uTLB), 第二级为 jTLB。uTLB 为全相联结构, 包含 20 个表项 (10 个 I 和 10 个 D); jTLB 为两路组相联的 RAM, 每路有至多 256 个表项。

5.2 编程模型和地址转换

C906 MMU 在 SATP 寄存器外扩展了 SMCIR、SMIR、SMEH 和 SMEL 控制寄存器, 用户可以通过对这几个扩展寄存器读写, 实现对 TLB 进行读写、查询和无效的操作。

5.2.1 MMU 控制寄存器

MMU 控制寄存器读写权限为超级用户模式和机器模式。

5.2.1.1 MMU 控制寄存器 (SMCIR)

SMCIR 寄存器实现对 MMU 进行多种操作, 包括 TLB 查找、TLB 读写、TLB 无效等操作。

TLBP – TLB 查询

根据 SMEH 寄存器去查询 TLB, 当查询命中时, 用 TLB 的序号去更新 SMIR 寄存器。

TLBR – TLB 读

根据 SMIR 寄存器索引, 读出对应的 TLB 表项的值, 并用这些值更新 SMEH 和 SMEL 寄存器。

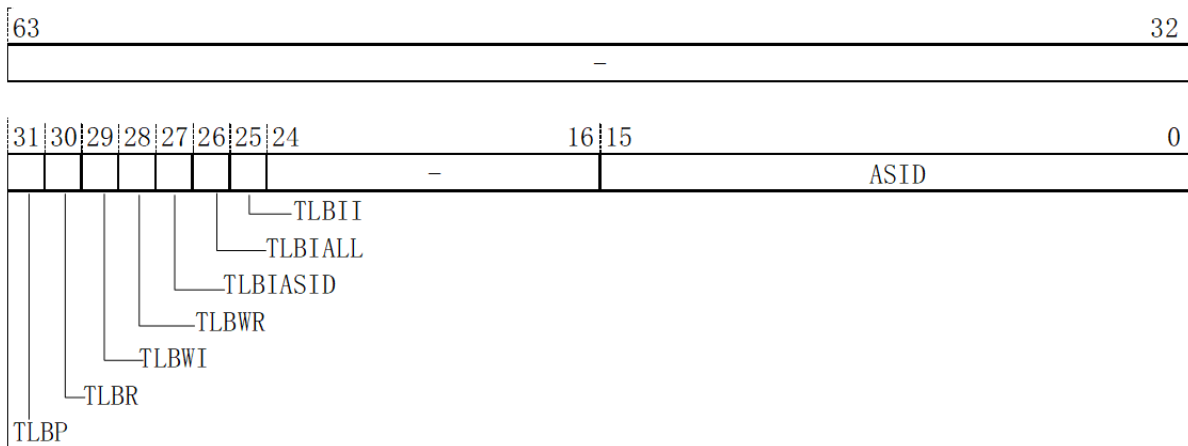


图 5.1: SMCIR 寄存器说明

TLBWI – TLB 索引写

根据 SMIR 寄存器索引值，将寄存器 SMEH, SMEL 写入 TLB 对应表项。

TLBWR – TLB 随机写

根据 Random 寄存器索引值，将寄存器 SMEH, SMEL 写入 TLB 对应表项。

TLBIASID – TLB 根据 ASID 无效

所有匹配 ASID 的 TLB 表项全部无效。

TLBIALL – TLB 初始化

将所有 TLB 表项无效，初始化。

TLBII – TLB 根据索引无效

根据 SMIR 寄存器索引值，将对应的 TLB 表项无效。

ASID – ASID 号

TLBIASID 操作使用该 ASID 做匹配。

5.2.1.2 MMU Index 寄存器 (SMIR)

MMU 索引寄存器，用于索引 TLB。在进行 TLB 查询时，将命中表项的 index 更新 SMIR 的 Index 域。在 TLB 写索引时，通过写 SMIR 的 index 域，可以将 SMEHSMEL 寄存器所示映射关系写入 jTLB 中对应 index 处。

P – Probe Failure

- 1' b0: TLBP 查询匹配命中;
- 1' b1: TLBP 查询没命中。

Tfatal – Probe multiple

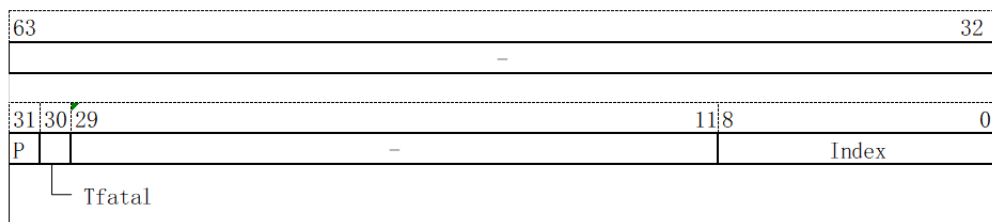


图 5.2: SMIR 寄存器说明

执行 TLBP 指令时，是否发生多重匹配。

- 1' b0: 没有多重匹配；
- 1' b1: 发生多重匹配。

Index – TLB Index

- Index[6] 用于 way 选择，Index[5:0] 用于 Set 选择（128 个 entry 配置）；
- Index[7] 用于 way 选择，Index[6:0] 用于 Set 选择（256 个 entry 配置）；
- Index[8] 用于 way 选择，Index[7:0] 用于 Set 选择（512 个 entry 配置）；

5.2.1.3 MMU EntryHi 寄存器 (SMEH)

SMEH 寄存器有两个功能包括：TLB 访问的虚拟地址信息和 TLB 异常时当前 VPN 的值。

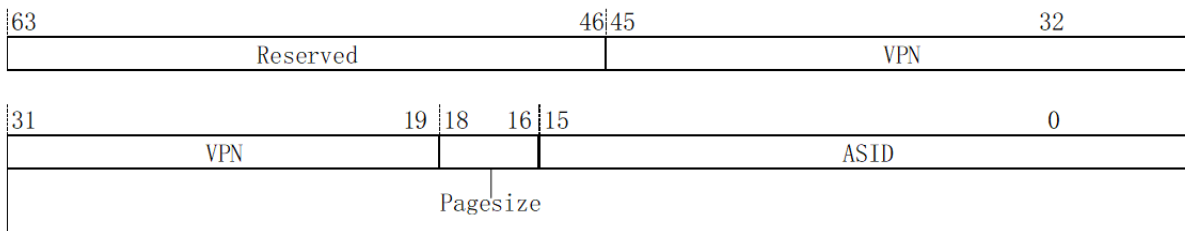


图 5.3: SMEH 寄存器说明

VPN – 虚拟页帧号

该域在 TLB 读和发生页面错误异常时硬件更新，在软件写 TLB 表项之前由软件预先写入。

Pagesize – 页面尺寸

Oneshot 从低到高表示 4K, 2M, 1G 页面大小。
该域在 TLB 读时硬件更新，在软件写 TLB 表项之前由软件预先写入。

ASID – 地址空间标识

D – Dirty

D 位为 1 时，表明该页是否被改写。

1' b0: 当前页未被写/不可写；

1' b1: 当前页已经被写/可写。

此位在 C906 的硬件实现与 W 属性类似。当 D 位为 0 时，对此页面进行写操作会触发 Page Fault (store) 异常，通过在异常服务程序中配置 D 位来维护该页面是否已经被改写/可写的定义。该位复位为 0。

A – Accessed

A 位为 1 时，表明该页可访问。为 0 时不可访问，否则会触发 Page Fault (对应访问类型) 异常。该位复位为 0。

G – Global

全局页面标识，当前页可供多个进程共享，该位复位为 0。

1' b0: 非共享页面，进程号 ASID 私有；

1' b1: 共享页面。

U – User

用户模式可访问，该位复位为 0。

1' b0: 用户模式不可访问，当用户模式访问，出 page fault 异常；

1' b1: 用户模式可访问。

X : 可执行；**W** : 可写；**R** : 可读。

表 5.1: XRW 权限说明

X	W	R	Meaning
0	0	0	Pointer to next level of page table
0	0	1	Read-only page
0	1	0	Reserved for future use
0	1	1	Read-write page
1	0	0	Execute-only page
1	0	1	Read-execute page
1	1	0	Reserved for future page
1	1	1	Read-write-execute page

违反 XWR 权限时将会触发 Page Fault 异常。

V – Valid

表明物理页在内存中是否分配好，访问一个 V 为 0 的页面，将触发 Page Fault 异常。该位复位为 0。

1' b0: 当前页没有分配好；

1' b1: 当前页已分配好。

5.2.1.5 MMU 地址转换寄存器 (SATP)

SATP 为 Sv39 规范的 MMU 控制寄存器。

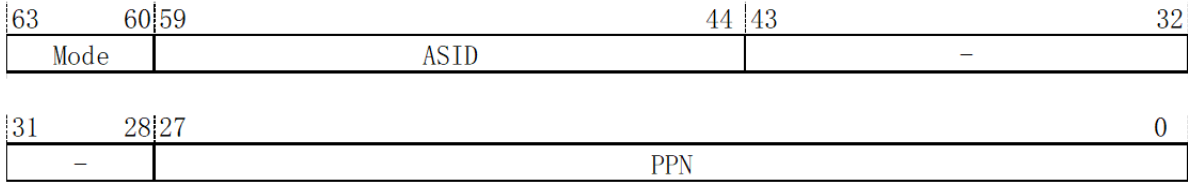


图 5.5: SATP 寄存器说明

Mode - MMU 地址翻译模式

表 5.2: MMU 地址翻译模式

Value	Name	Description
0	Bare	No translation or protection
1-7	-	Reserved
8	Sv39	Page-based 39-bit virtual addressing
9	Sv48	Page-based 48-bit virtual addressing
10	Sv57	Reserved for page-based 57-bit virtual addressing
11	Sv64	Reserved for page-based 64-bit virtual addressing
12-15	-	Reserved

当 Mode 为 0 时，MMU 关闭。C906 只支持 MMU 关闭和 Sv39 两种模式。

ASID – 当前 ASID

表示当前程序的 ASID 号。

PPN – 硬件回根 PPN

第一级硬件回根使用的 PPN。

5.2.2 地址转换流程

MMU 的主要功能是将虚拟地址转换为物理地址并进行相应的权限检查，具体的地址映射关系和相应权限由操作系统进行配置，存放于地址转换页表中，C906 采用最多三级页表索引的方式实现地址转换：访问第一级页表得到第二级页表的基地址和相应的权限属性；访问第二级页表得到第三级页表的基地址和相应的权限属性；访问第三级页表得到最终物理地址和相应的权限属性。每一级访问都有可能得到最终的物理地址，即叶子页表。虚拟页面号 VPN 有 27-bit，等分为三个 9-bit 的 $VPN[i] (\{VPN[2], VPN[1], VPN[0]\})$ ，每次访问使用一部分 VPN 进行索引，顺序为 $VPN[2] > VPN[1] > VPN[0]$ 。当页表大小配置为 4KB/2MB/1GB 时，页表索引的次数分别为 3/2/1 次。

叶子页表的表项内容（即由虚拟地址转换得到的物理地址和相应的权限属性）被缓存于 TLB 内以加速地址转换，TLB 包括两级：第一级 uTLB 分为指令、数据两个部分，均为 10 个表项，用于进一步加速 TLB 访问；第二级 jTLB 指令数据共用，两路组相联。

若 uTLB 失配则访问 jTLB，若 jTLB 进一步失配，MMU 则会启动 Hardware Page Table Walk，访问内存得到最终的地址转换结果。

5.2.2.1 页表结构

页表用于存储下级页表的入口地址或者最终页表的物理信息，其结构如 图 5.6 所示。

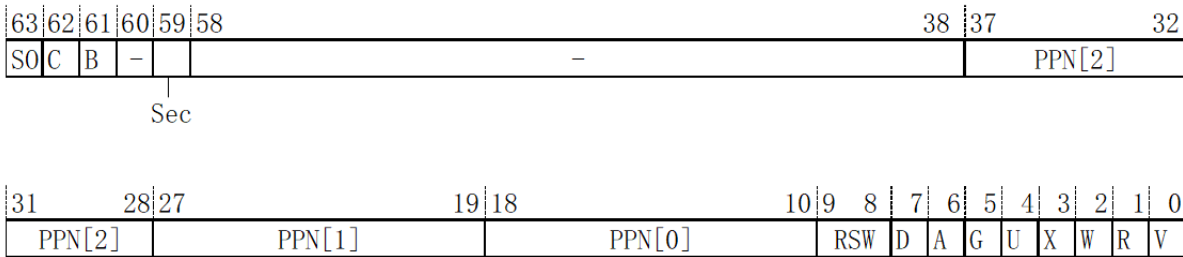


图 5.6: 页表结构说明

Flags – 9:0 位页面属性

功能如 *MMU EntryLo* 寄存器 (*SMEL*) 中所述。

Flags – 63:59 位页面属性

C906 扩展页面属性，当 *MXSTATUS* 寄存器的 *MAEE* 位为 1 时存在，功能如 *MMU EntryLo* 寄存器 (*SMEL*) 中所述。

PPN – 页表物理地址

PPN[i] 分别代表三级页表转换时所对应的 PPN 值。

5.2.2.2 地址转换流程

若 TLB 命中，则从 TLB 中直接获取物理地址及相关属性；若 TLB 缺失，则地址的转换具体步骤为：

1. 根据 SATP.PPN 和 VPN[2] 得到一级页表访存地址 {SATP.PPN, VPN[2], 3' b0}，使用该地址访问 Dcache/内存，得到 64-bit 一级页表 PTE。
2. 检查 PTE 是否符合 PMP 权限，若不符合则产生相应 access error 异常；若符合则根据 表 5.1 所示的规则判断 X/W/R-bit 是否符合叶子页表条件，若符合叶子页表条件则说明已经找到最终物理地址，到第 3 步；若不符合则到第 1 步，使用 PTE.PPN 拼接下一级 VPN[]，再拼接 3' b0 得到下一级页表访存地址继续访问 Dcache/内存。
3. 找到了叶子页表，结合 PMP 中的 X/W/R/L 位和 PTE 中的 X/W/R 位得到两者的最小权限进行权限检查，并将 PTE 的内容回填到 JTLB 中。
4. 在任何一步的 PMP 检查中，如果有权限违反，则根据访问类型产生对应的 access error 异常。
5. 若得到叶子页表，但：访问类型违反 A/D/X/W/R/U-bit 的设置，产生对应的 page fault 异常；若三次访问结束仍未得到叶子页表，则产生对应的 page fault 异常；若访问 Dcache/内存过程中得到 access error 响应，产生 page fault 异常。

6. 若得到叶子页表，但访问次数少于 3 次，则说明得到大页表。检查大页表的 PPN 是否按照页表尺寸对齐，若未对齐，则产生 page fault 异常。

uTLB 缺失时最快耗时 4 个处理器时钟周期可从下级缓存中获取页表。若 jTLB 缺失，从核内取指等请求发给 MMU 时开始最快耗时 16 个处理器时钟周期可从下级缓存中获取页表。

5.3 TLB 组织形式

C906 MMU 采用两级 TLB，第一级为 uTLB，分别为指令 I-uTLB 和数据 D-uTLB，第二级为 jTLB。在处理器 reset 后，硬件会将 uTLB 和 jTLB 的所有表项进行无效化操作，软件无需初始化操作。

I-uTLB 有 10 个全相联表项，每个表项可以混合存储 4K、2M 和 1G 三种大小的页面，取指请求命中 I-uTLB 时，当拍可以得到物理地址和相应权限属性。

D-uTLB 有 10 个全相联表项，可以混合存储 4K、2M 和 1G 三种大小的页面，加载和存储请求命中 D-uTLB 时，当拍可以得到物理地址和相应权限属性。

jTLB 为指令和数据共用，两路组相联结构，表项大小 128/256/512 可配，可以混合存储 4K、2M 和 1G 三种大小页面。uTLB 缺失，jTLB 命中时，最快 3 个 cycle 返回物理地址和相应权限属性。

第六章 物理内存保护

6.1 PMP 概览

C906 物理内存保护 (Physical Memory Protection, 以下简称 PMP) 遵从 RISC-V 标准。在受保护的系统中, 主要有两类资源的访问需要被监视: 存储器系统和外围设备。PMP 单元负责对系统内存 (包括存储器系统和外围设备) 的访问合法性进行检查, 其主要是判定当前工作模式下处理器是否具备对内存地址的读/写/执行访问权限。

C906 PMP 单元的主要特征有:

- 可配置 8/16 个 PMP 表项, 16 个表项通过 0-15 的编号来索引;
- 地址划分最小粒度为 4KB;
- 支持 OFF、TOR、NAPOT 三种地址匹配模式, 不支持 NA4 匹配模式;
- 支持可读、可写、可执行三种权限的配置;
- 每个表项支持软件 Lock 操作;

6.2 PMP 控制寄存器

PMP 表项主要由一个 8 比特的设置寄存器 (PMPCFG) 和一个 64 比特的地址寄存器 (PMPADDR) 构成, 所有 PMP 控制寄存器都只能在机器模式下访问, 其他模式访问将产生非法指令异常。

6.2.1 物理内存保护设置寄存器 (PMPCFG)

每个 64 位的 PMPCFG 提供 8 个表项的权限设置, 整体分布如 图 6.1 所示。

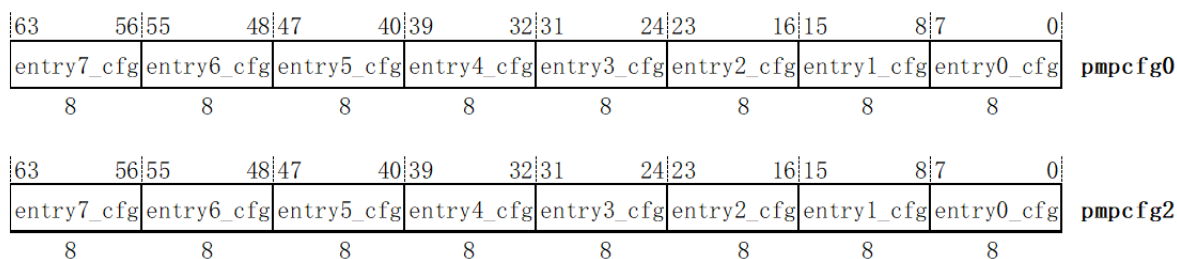


图 6.1: 物理内存保护设置寄存器整体分布

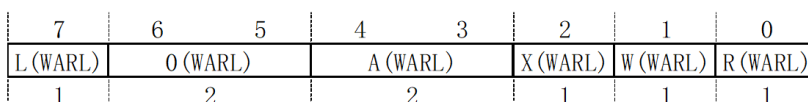


图 6.2: 物理内存保护设置寄存器

每个表项的 8 比特设置寄存器的排布如 图 6.2 所示。

PMP 控制寄存器描述其具体的描述如 表 6.1 所示。

表 6.1: PMP 控制寄存器描述

位	名称	描述
0	R	0: 表项匹配地址不可读; 1: 表项匹配地址可读。
1	W	0: 表项匹配地址不可写; 1: 表项匹配地址可写。
2	X	0: 表项匹配地址不可执行; 1: 表项匹配地址可执行。
4:3	A	表项的地址匹配模式: 00: OFF, 无效表项; 01: TOR (Top of range), 使用相邻表项的地址作为匹配区间的模式; 10: NA4 (Naturally aligned four-byte region), 区间大小为 4 字节的匹配模式, 该模式不支持; 11: NAPOT (Naturally aligned power-of-2 regions), 区间大小为 2 的幂次方的匹配模式, 至少为 4KB。
7	L	表项的 Lock 使能位: 0: 机器模式的访问都将成功。 超级用户模式/用户模式的访问根据 R/W/X 判定是否成功。 1: 表项被锁住, 无法对相关表项进行修改。 当配置 TOR 模式, 其索引号减一的表项的地址寄存器也无法被修改。 所有模式都需要根据 R/W/X 判定是否访问成功。

对于 TOR 模式, 假设访问地址为 a, 则其命中表项 i 的条件为: $PMPADDR_{i-1} \leq a < MPADDR_i$ 。对于 0 号表项, 其使用 0 作为下边界。

对于 NAPOT 模式, 其地址与区间大小的关系 表 6.2 所示。

表 6.2: 保护区间编码

PMPADDR	PMPCFG.A	保护区大小	备注
yy_yyyy_yyyy_yy yy_yyyy_yyyy_yyy y_yyyy_yyyy_yyyy	NA4	4B	该模式不支持
yy_yyyy_yyyy_yy yy_yyyy_yyyy_yyy y_yyyy_yyyy_yyy0	NAPOT	8B	该模式不支持
yy_yyyy_yyyy_yy yy_yyyy_yyyy_yyy y_yyyy_yyyy_yy01	NAPOT	16B	该模式不支持
yy_yyyy_yyyy_yy yy_yyyy_yyyy_yyy y_yyyy_yyyy_y011	NAPOT	32B	该模式不支持
yy_yyyy_yyyy_yy yy_yyyy_yyyy_yyy y_yyyy_yyyy_0111	NAPOT	64B	该模式不支持

下页继续

表 6.2 – 续上页

PMPADDR	PMPCFG.A	保护区大小	备注
yy_yyyy_yyyy_yy yy_yyyy_yyyy_yyy y_yyyy_yyy0_1111	NAPOT	128B	该模式不支持
yy_yyyy_yyyy_yy yy_yyyy_yyyy_yyy y_yyyy_yy01_1111	NAPOT	256B	该模式不支持
yy_yyyy_yyyy_yy yy_yyyy_yyyy_yyy y_yyyy_y011_1111	NAPOT	512B	该模式不支持
yy_yyyy_yyyy_yy yy_yyyy_yyyy_yyy y_yyyy_0111_1111	NAPOT	1KB	该模式不支持
yy_yyyy_yyyy_yy yy_yyyy_yyyy_yyy y_yyy0_1111_1111	NAPOT	2KB	该模式不支持
yy_yyyy_yyyy_yy yy_yyyy_yyyy_yyy y_yy01_1111_1111	NAPOT	4KB	支持
yy_yyyy_yyyy_yy yy_yyyy_yyyy_yyy y_y011_1111_1111	NAPOT	8KB	支持
yy_yyyy_yyyy_yy yy_yyyy_yyyy_yyy y_0111_1111_1111	NAPOT	16KB	支持
yy_yyyy_yyyy_yy yy_yyyy_yyyy_yyy 0_1111_1111_1111	NAPOT	32KB	支持
yy_yyyy_yyyy_yy yy_yyyy_yyyy_yy0 1_1111_1111_1111	NAPOT	64KB	支持
yy_yyyy_yyyy_yy yy_yyyy_yyyy_y01 1_1111_1111_1111	NAPOT	128KB	支持
yy_yyyy_yyyy_yy yy_yyyy_yyyy_011 1_1111_1111_1111	NAPOT	256KB	支持
yy_yyyy_yyyy_yy yy_yyyy_yyy0_111 1_1111_1111_1111	NAPOT	512KB	支持
yy_yyyy_yyyy_yy yy_yyyy_yy01_111 1_1111_1111_1111	NAPOT	1MB	支持
yy_yyyy_yyyy_yy yy_yyyy_y011_111 1_1111_1111_1111	NAPOT	2MB	支持
yy_yyyy_yyyy_yy yy_yyyy_0111_111 1_1111_1111_1111	NAPOT	4MB	支持
yy_yyyy_yyyy_yy yy_yyy0_1111_111 1_1111_1111_1111	NAPOT	8MB	支持
yy_yyyy_yyyy_yy yy_yy01_1111_111 1_1111_1111_1111	NAPOT	16MB	支持
yy_yyyy_yyyy_yy yy_y011_1111_111 1_1111_1111_1111	NAPOT	32MB	支持
yy_yyyy_yyyy_yy yy_0111_1111_111 1_1111_1111_1111	NAPOT	64MB	支持
yy_yyyy_yyyy_yy y0_1111_1111_111 1_1111_1111_1111	NAPOT	128MB	支持
yy_yyyy_yyyy_yy 01_1111_1111_111 1_1111_1111_1111	NAPOT	256MB	支持
yy_yyyy_yyyy_y0 11_1111_1111_111 1_1111_1111_1111	NAPOT	512MB	支持
yy_yyyy_yyyy_01 11_1111_1111_111 1_1111_1111_1111	NAPOT	1GB	支持
yy_yyyy_yyy0 11 11_1111_1111_111 1_1111_1111_1111	NAPOT	2GB	支持
yy_yyyy_yy01 11 11_1111_1111_111 1_1111_1111_1111	NAPOT	4GB	支持
yy_yyyy_y011 11 11_1111_1111_111 1_1111_1111_1111	NAPOT	8GB	支持
yy_yyyy_0111 11 11_1111_1111_111 1_1111_1111_1111	NAPOT	16GB	支持
yy_yyy0_1111 11 11_1111_1111_111 1_1111_1111_1111	NAPOT	32GB	支持
yy_yy01_1111 11 11_1111_1111_111 1_1111_1111_1111	NAPOT	64GB	支持
yy_y011_1111 11 11_1111_1111_111 1_1111_1111_1111	NAPOT	128GB	支持
yy_0111_1111 11 11_1111_1111_111 1_1111_1111_1111	NAPOT	256GB	支持
y0_1111_1111 11 11_1111_1111_111 1_1111_1111_1111	NAPOT	512GB	支持
01_1111_1111 11 11_1111_1111_111 1_1111_1111_1111	NAPOT	1TB	支持
11_1111_1111 11 11_1111_1111_111 1_1111_1111_1111	NAPOT	2TB	支持

需要说明的是，C906 PMP 支持的地址匹配区间最小粒度为 4KB。

6.2.2 物理内存保护地址寄存器 (PMPADDR)

PMP 共实现了 8/16 个地址寄存器 PMPADDR0-PMPADDR7/15，存放表项的物理地址。

RISC-V 规定 PMPADDR 存放的是物理地址的 [39:2] 比特，因为 C906 PMP 表项粒度最低支持 4KB，因此 PMPADDR[8:0] 不会用于地址鉴权逻辑。

	63	38	37	9	8	0
	0		Physical address [39:11] (WARL)		0 (WARL)	
Reset	0		0		0	

图 6.3: PMP 地址寄存器

平头哥半导体

第七章 内存子系统

C906 采用哈佛结构的一级高速缓存，包含独立的指令高速缓存和数据高速缓存。

7.1 指令高速缓存子系统

L1 指令高速缓存的主要特征如下：

- 指令高速缓存大小硬件可配置，支持 8KB/16KB/32KB/64KB；
- 2 路组相联，缓存行大小为 64B；
- 虚拟地址索引，物理地址标记 (VIPT)；
- 读取宽度为 32 比特，回写宽度为 128 比特；
- 采用先进先出的替换策略；
- 支持对整个指令高速缓存的无效操作，支持对单条缓存行的无效操作；
- 支持指令预取功能。

7.1.1 指令预取

L1 指令高速缓存支持指令预取功能，通过配置隐式操作寄存器 MHINT.IPLD 实现。在当前缓存行访问缺失时，开启下一条连续缓存行的预取，并将预取结果缓存到指令高速缓存中。

指令预取额外要求预取的缓存行与当前访问缓存行位为同一个页面，否则指令预取功能关闭，从而保证取指地址的安全。此外，读敏感的外设地址空间也禁止被分配到指令区空间。

7.1.2 分支历史表

C906 采用分支历史表对条件分支的跳转方向进行预测。分支历史表容量为 8Kb/16Kb 硬件可配，使用 GSHARE 预测器作为预测机制，每周支持一条分支结果预测。

分支历史表进行预测的条件分支指令包括：

- BEQ、BNE、BLT、BLTU、BGE、BGEU、C.BEQZ、C.BNEZ。

7.1.3 分支跳转目标预测器

C906 使用分支跳转目标预测器对分支指令的跳转目标地址进行预测。分支跳转目标预测器对分支指令历史目标地址进行记录，如果当前分支指令命中分支跳转目标预测器，则将记录目标地址作为当前分支指令预测目标地址。

分支跳转目标预测器主要特征包括：

- 支持 16 个表项，全相联结构；
- 使用当前分支指令部分 PC 进行索引；
- 分支历史表命中即可发起跳转，无额外跳转损失。

分支跳转目标预测器进行预测的分支指令包括：

- BEQ、BNE、BLT、BLTU、BGE、BGEU、C.BEQZ、C.BNEZ；
- JAL、C.J。

7.1.4 返回地址预测器

返回地址预测器用于函数调用结束时，返回地址的快速准确预测。当取指单元译码得到有效的函数调用指令时，将函数返回地址压栈存入返回地址预测器；当取指单元译码得到有效的函数返回指令时，则从返回地址预测器弹栈，获取函数返回目标地址。

返回地址预测器最多支持 4 层函数调用嵌套，超出嵌套次数会导致目标地址预测错误。

函数调用指令包括：JAL、JALR、C.JALR。

函数返回指令包括：JALR、C.JR、C.JALR。

指令功能的具体划分可以如表 7.1。

表 7.1: 指令功能具体划分

rd	rs1	rs1=rd	RAS action
!link	!link	-	none
!link	link	-	pop
link	!link	-	push
link	link	0	push and pop
link	link	1	push

7.2 数据高速缓存子系统

L1 数据高速缓存的主要特征如下：

- 数据高速缓存大小硬件可配置，支持 8KB/16KB/32KB/64KB；
- 4 路组相联，缓存行大小为 64B；
- 虚拟地址索引，物理地址标记 (VIPT)；
- 每次读访问的最大宽度为 64 比特，支持字节/半字/字/双字访问；

- 每次写访问的最大宽度为 128 比特，支持任意字节组合的访问；
- 写策略支持 write_back 以及 write_allocate/write_nonallocate 的模式组合配置；
- 采用先进先出的替换策略；
- 支持对整个数据高速缓存的无效和清除操作，支持对单条缓存行的无效和清除操作；
- 支持数据预取功能；

7.2.1 数据预取

为了减少 DDR 等大内存的存储访问延时，C906 支持数据预取功能。LSU 单元通过检测数据高速缓存的缺失，匹配出固定的访问模式，然后硬件自动预取缓存行并回填 L1 数据高速缓存。C906 实现了连续预取和间隔预取 (stride≤16 个缓存行) 这 2 种不同的预取方式。此外，在间隔预取模式下实现了正向预取和反向预取 (即 stride 为负数)，从而支持各种可能的访问模式。

在处理器执行数据高速缓存无效和清除操作时，停止数据预取功能。用户通过设置隐式寄存器 MHINT.DPLD 使能数据预取功能，并通过设置 MHINT.DPLD_DIS 决定一次预取的缓存行数量。

支持数据预取的指令如下：

- LB、LBU、LH、LHU、LW、LWU、LD；
- FLH、FLW、FLD；
- LRB、LRH、LRW、LRD、LRBU、LRHU、LRWU、LURB、LURH、LURW、LURD、LURBU、LURHU、LURWU、LBI、LHI、LWI、LDI、LBUI、LHUI、LWUI、LDD、LWD、LWUD；
- VLB、VLH、VLW、VLBU、VLHU、VLWU、VLE、VLSB、VLSH、VLSW、VLSBU、VLSHU、VLSWU、VLSE、VLXB、VLXH、VLXW、VLXBU、VLXHU、VLXWU、VLXE、VLBFF、VLHFF、VLWFF、VLBUFF、VLHUFF、VLWUFF、VLEFF；
- VLSEG、VLSSEG、VLXSEG；

7.2.2 自适应的写分配机制

C906 提供自适应的写分配机制，当处理器检测到连续的内存写入操作时，页面的写分配属性会自动关闭。用户可通过设置隐式寄存器 HINT.AMRL1 决定连续存储多少条缓存行后关闭 L1 数据高缓的写分配策略。

当执行数据高缓的无效或者清除操作时，处理器的自适应写分配机制会被关闭，在高速缓存操作完成后，重新检测内存连续写入行为。

支持自适应写分配的指令如下：

- SB、SH、SW、SD；
- FSH、FSW、FSD；
- SRB、SRH、SRW、SRD、SURB、SURH、SURW、SURD、SBI、SHI、SWI、SDI、SDD、SWD；
- VSB、VSH、VSW、VSBU、VSHU、VSWU、VSE、VSSB、VSSH、VSSW、VSSBU、VSSHU、VSSWU、VSSE、VSXB、VSXH、VSXW、VSXBU、VSXHU、VSXWU、VSXE、VSBFF、VSHFF、VSWFF、VSBUFF、VSHUFF、VSWUFF、VSEFF；
- VSSEG、VSSSEG、VSXSEG；

7.2.3 独占式访问

C906 支持独占式的内存访问指令 LR 和 SC。用户可以使用这两条指令构成原子锁等同步原语实现同一个核不同进程之间的同步。通过 LR 指令标记需要独占访问的地址，SC 指令判断被标记的地址是否被其他进程抢占。C906 设置了一个位于 L1 数据高缓的局部监测器。该监测器由一个状态机和一个地址缓存器组成，其中，状态机包含两个状态：IDLE 和 EXCLUSIVE。

LR 指令在执行过程中设置局部监测器的状态机为 EXCLUSIVE 态并将访问的地址和 Size 保存到缓存器中；SC 指令在执行过程中读取局部监测器的状态、地址和 Size，如果状态为 EXCLUSIVE 并且地址和 Size 完全匹配，那么执行该写操作，返回写成功，并清除状态机回到 IDLE 态；否则如果状态或者地址/Size 有一项不满足条件，或者数据高速缓存未使能时，不执行该写操作，返回写失败，并清除状态机回到 IDLE 态。此外，在进程切换时需要清除局部监测器。

7.3 L1 高速缓存操作

在处理器复位后，指令和数据高速缓存会自动进行无效化操作，且默认关闭指令和数据高速缓存。C906 扩展了高速缓存操作相关的控制寄存器和指令，支持高速缓存的开关、清脏表项、无效化和读操作。

7.3.1 L1 高速缓存扩展寄存器

C906 L1 高速缓存相关扩展寄存器，按功能主要分为：

- 高速缓存使能和模式配置：机器模式硬件配置寄存器 (mhcr) 可以实现对指令和数据高速缓存的开关以及写分配和写回模式的配置。超级用户模式硬件配置寄存器 (shcr) 是 mhcr 的映射，为只读寄存器。
- 脏表项清除和无效化操作：机器模式高速缓存操作寄存器 (mcor) 可以对指令和数据高速缓存进行脏表项和无效化操作。
- 高速缓存读操作：机器模式高速缓存访问指令寄存器 (mcins)、高速缓存访问索引寄存器 (mcindex) 和高速缓存访问数据寄存器 0/1 (mcdata0/1)，通过这三个寄存器可以实现对指令和数据高速缓存的读操作。

具体控制寄存器说明可以参考机器模式处理器控制和状态扩展寄存器组。

7.3.2 L1 高速缓存扩展指令

C906 扩展了高速缓存相关操作的指令，包括：按地址进行无效化、无效化全部、按地址清脏表项、清全部脏表项、按地址清脏表项并无效化和清全部脏表项并无效化，具体如表 7.2 所示。

表 7.2: L1Cache 扩展指令

DCACHE.CALL	DCACHE 清全部脏表项指令
DCACHE.CVA	DCACHE 指定虚拟地址清脏表项指令
DCACHE.CPA	DCACHE 指定物理地址清脏表项指令
DCACHE.CSW	DCACHE 指定 way/set 清脏表项指令
DCACHE.IALL	DCACHE 无效全部表项指令
DCACHE.IVA	DCACHE 指定虚拟地址无效表项指令
DCACHE.IPA	DCACHE 指定物理地址无效表项指令
DCACHE.ISW	DCACHE 指定 way/set 无效表项指令
DCACHE.CIALL	DCACHE 清全部脏表项并无效指令
DCACHE.CIVA	DCACHE 指定虚拟地址清脏表项并无效指令
DCACHE.CIPA	DCACHE 指定物理地址清脏表项并无效指令
DCACHE.CISW	DCACHE 指定 way/set 清脏表项并无效指令
ICACHE.IALL	ICACHE 无效全部表项指令
ICACHE.IALLS	ICACHE 广播无效全部表项指令
ICACHE.IVA	ICACHE 指定虚拟地址无效表项指令
ICACHE.IPA	ICACHE 指定物理地址无效表项指令

第八章 总线接口协议

本章描述了 C906 处理器的接口总线协议。C906 的总线支持在处理器和其他外围设备之间同步传输数据，包括数据存取、地址控制和总线控制。

8.1 主设备接口

C906 的主设备接口支持 AMBA 4.0 AXI 协议（请参考 AMBA 规格说明—AMBA AXI and ACE Protocol Specification）

8.1.1 特点

主设备接口负责 C906 和 AXI 系统总线之间的地址控制和数据传输，其基本特点包括：

- 支持 AMBA 4.0 AXI 总线协议；
- 支持 128 位总线宽度；
- 支持系统动态变频；
- 所有的总线输入和输出信号都经过寄存器打拍，以获得较好的时序。

8.1.2 协议内容

8.1.2.1 支持传输类型

主设备接口支持的传输特性如下：

- BURST 支持 INCR1, INCR4, WRAP4 传输，其它突发类型均不支持；
- BURST 支持传输长度为 1 或者 4，其他传输长度均不支持；
- 支持独占式访问；
- SIZE 支持四字、双字、字、半字和字节传输，其它传输大小不支持；
- 支持读和写操作。
- 读通道 outstanding 能力为 10，写通道 outstanding 能力为 16。

8.1.2.2 支持响应类型

主设备接口接收从设备的响应类型为：

- OKAY
- EXOKAY
- SLVERR
- DECERR

8.1.3 不同总线响应下的行为

总线上出现不同总线响应时 CPU 的行为如表 8.1。

表 8.1: 总线异常处理

RRESP/BRESP	结果
OKAY	普通传输访问成功，或 exclusive 传输访问失败；读传输 exclusive 访问失败代表总线不支持 exclusive 传输，产生访问错误异常，写传输 exclusive 访问失败仅代表抢锁失败，不会返回异常。
EXOKAY	exclusive 访问成功。
SLVERR/DECERR	访问出错，读传输产生访问错误异常，写传输忽略此错误。

第九章 处理器核局部中断 CLINT

C906 实现了处理器核局部中断控制器 (以下简称 CLINT), 是一个内存地址映射的模块, 用于处理软件中断和计时器中断。

9.1 寄存器地址映射

CLINT 占据 64KB 内存空间, 其高 13 位地址由 SoC 集成 C906 时硬件指定, 低 27 位地址映射如表 9.1 所示。所有寄存器仅支持字对齐的访问。

表 9.1: CLINT 寄存器存储器映射地址

地址	名称	属性	初始值	描述
0x4000000	MSIP0	读/写	0x00000000	机器模式软件中断配置寄存器 高位绑 0, bit[0] 有效
Reserved	-	-	-	-
0x4004000	MTIMECMPL0	读/写	0xFFFFFFFF	机器模式系统计时器 比较值寄存器 (低 32 位)
0x4004004	MTIMECMPH0	读/写	0xFFFFFFFF	机器模式系统计时器 比较值寄存器 (高 32 位)
Reserved	-	-	-	-
0x400C000	SSIP0	读/写	0x00000000	超级用户模式软件中断配置寄存器 高位绑 0, bit[0] 有效
Reserved	-	-	-	-
0x400D000	STIMECMPL0	读/写	0xFFFFFFFF	超级用户模式系统计时器 比较值寄存器 (低 32 位)
0x400D004	STIMECMPH0	读/写	0xFFFFFFFF	超级用户模式系统计时器 比较值寄存器 (高 32 位)
Reserved	-	-	-	-

9.2 软件中断

CLINT 可用于设置软件中断。软件中断通过配置地址映射的软件中断配置寄存器进行控制。其中机器模式软件中断由机器模式软件中断配置寄存器 (MSIPR) 控制, 超级用户模式软件中断由超级用户模式软件中断配置寄存器 (SSIPR) 控制。

用户可通过将 MSIPR/SSIPR 寄存器的值置 1 的方式，产生软件中断；可通过将 MSIPR/SSIPR 寄存器的值清 0 的方式，清除软件中断。其中 CLINT 超级用户模式软件中断请求，仅在 MXSTATUS 寄存器 CLINTEE 域的值为 1 时有效。

机器模式下拥有访问/修改所有软件中断相关寄存器的权限；超级用户模式下仅具有访问/修改 SSIPR 的权限；用户模式没有权限。

两组寄存器的结构相同，其寄存器位分布和位定义如 图 9.1 所示。

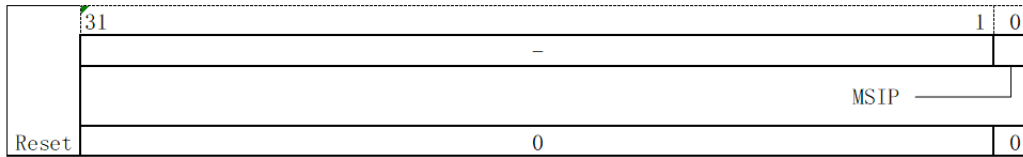


图 9.1: 机器模式软件中断配置寄存器 (MSIPR)

• **MSIP: 机器模式软件中断等待位**

该位表示机器模式软件中断的中断状态：

- 当 MSIP 位置 1，当前有有效的机器模式软件中断请求；
- 当 MSIP 位置 0，当前没有有效的机器模式软件中断请求。

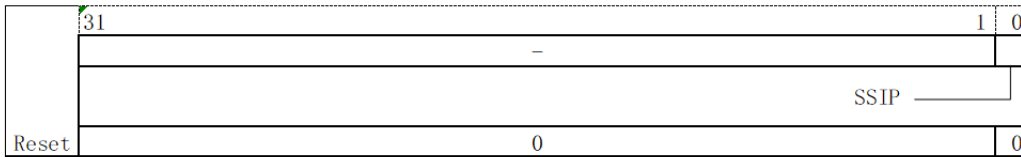


图 9.2: 超级用户模式软件中断配置寄存器 (SSIPR)

• **SSIP: 超级用户模式软件中断等待位**

该位表示超级用户模式软件中断的中断状态：

- 当 SSIP 位置 1，当前有有效的超级用户软件中断请求；
- 当 SSIP 位置 0，当前没有有效的超级用户软件中断请求。

9.3 计时器中断

按照 RISC-V 定义，在系统中需要实现一个 64 位系统计时器 MTIME，该实时计时器要求工作在常开的时钟域下。因为 CPU 在低功耗模式下可由 SoC 决定是否关闭工作时钟，因此该计时器的计数寄存器需要实现在 SoC 常开的时钟域下，并可在 SoC 对 MTIME 寄存器分配地址空间，通过加载或者存储指令对其进行读写操作，该寄存器可复位清零。另外，该计数寄存器的当前值需要传递给 CPU 内部，CPU 可通过 CSR 指令读取 TIME 这一 MTIME 的镜像 CSR 获取其值，CPU 不可通过 CSR 指令改写 MTIME 寄存器的值。

该系统计时器定义有一组 64 位的机器模式计时器比较值寄存器 (MTIMECMPH, MTIMECMPL) 和一组 64 位的超级用户模式计时器比较值寄存器 (STIMECMPH, STIMECMPL)。这些寄存器均可以通过地址字对齐访问的方式分别访问或修改。这两组寄存器实现在 C906 内部的 CLINT 模块内，地址映射如 表 9.1 所示。

通过比较系统计时器的当前值 (MTIME) 与比较值寄存器 {M/STIMECMPH[31:0],M/STIMECMPL[31:0]} 的值来确定是否产生计时器中断。当系统计时器的值小于等于 {M/STIMECMPH[31:0],M/STIMECMPL[31:0]} 的值时不产生中断；当系统计时器的值大于 {M/STIMECMPH[31:0],M/STIMECMPL[31:0]} 的值时 CLINT 产生对应的计时器中断。软件可通过改写 M/STIMECMP 的值来清除对应的计时器中断。其中超级用户模式计时器中断请求，仅在 MXSTATUS 寄存器 CLINTEE 域的值为 1 时有效。

机器模式下拥有访问/修改所有计时器中断相关寄存器的权限；超级用户模式下仅具有访问修改扩展的超级用户模式计时器比较值寄存器的权限；普通用户模式没有权限。

每组寄存器结构相同，其寄存器位分布和位定义如 图 9.3 所示。

	63	32	31	0
	MTIMECMPH		MTIMECMPL	
Reset	0xffffffff		0xffffffff	

图 9.3: 机器模式计时器中断比较值寄存器 (高/低)

• **MTIMECMPH/MTIMECMPL: 机器模式计时器中断比较值寄存器高位/低位**

该寄存器存储了计时器比较值。

- MTIMECMPH: 计时器比较值高 32 位;
- MTIMECMPL: 计时器比较值低 32 位。

	63	32	31	0
	STIMECMPH		STIMECMPL	
Reset	0xffffffff		0xffffffff	

图 9.4: 超级用户模式计时器中断比较值寄存器 (高/低)

• **STIMECMPH/STIMECMPL: 超级用户模式计时器中断比较值寄存器高位/低位**

该寄存器存储了计时器比较值，为平头哥扩展定义的寄存器。

- STIMECMPH: 计时器比较值高 32 位;
- STIMECMPL: 计时器比较值低 32 位。

第十章 中断控制器 PLIC

平台级中断控制器（以下简称 PLIC），仅用于对外部中断源进行采样，优先级仲裁和分发。

C906 实现的 PLIC 单元基本功能如下：

- 中断可配置为机器模式中断和超级用户模式中断；
- 最多 1023 个中断源采样，支持电平中断，脉冲中断；
- 32 个级别的中断优先级；
- 每个中断模式（机器/超级用户）的中断使能独立维护；
- 每个中断模式（机器/超级用户）的中断阈值独立维护；
- PLIC 寄存器访问权限可配置。

10.1 中断处理机制

10.1.1 中断仲裁

在 PLIC 中只有符合条件的中断源才会参与对某个中断目标模式的仲裁。满足的条件如下：

- 中断源处于等待状态（IP =1）；
- 中断优先级大于 0（优先级为 0 的中断为无效中断）；
- 对于该中断目标模式的使能位（PLIC_H0_MIE_n/SIE_n 寄存器）打开。

注释：中断目标模式是指该中断为超级用户模式中断或者机器模式中断，不同中断目标模式的响应情况参照表 10.1。

当 PLIC 中某个中断目标模式有多个中断处于 Pending 状态时，PLIC 仲裁出该中断目标模式下优先级最高（优先级寄存器值越大，优先级越高）的中断。C906 的 PLIC 实现中机器模式中断优先级始终高于超级用户模式中断，即处理器核心会优先响应 PLIC 仲裁出的机器模式有效中断。当模式相同的情况下，优先级配置寄存器的值越大，优先级越高，优先级为 0 的中断无效；如多个中断拥有相同的优先级并等待处理，则 ID 较小的优先被处理。

PLIC 会将仲裁结果以中断 ID 的形式更新入对应中断目标的中断响应/完成寄存器。

表 10.1 是依据 PLIC 仲裁出的中断目标模式和处理器当前工作模式等列出的中断响应条件及中断响应处理的模式。“更新寄存器”列所列出的“MCAUSE 等”表明处理器会切换到机器模式响应该中断，“SCAUSE 等”表明处理器会切换到超级用户模式下响应该中断。“Mie/Sie 寄存器”列对应为 MSTATUS 寄存器中的 MIE 和 SIE 寄存器的值是否为 1。“Delegation 寄存器”则对应 MIDELEG 寄存器中对应中断的位域。

表 10.1: 不同 PLIC 中断目标模式与处理器工作模式下中断的响应处理

处理器模式	中断模式	Mie/Sie 寄存器	Delegation 寄存器	响应中断	更新寄存器
M 态	M 态	Mie=1	不可配	响应	MCAUSE 等
	S 态	Mie=1	0	响应	MCAUSE 等
	S 态	Don' t care	1	不响应	不更新
S 态	M 态	Mie=1	不可配	响应	MCAUSE 等
	S 态	Don' t care	0	响应	MCAUSE 等
		Sie=1	1	响应	SCAUSE 等
U 态	M 态	Don' t care	不可配	响应	MCAUSE 等
	S 态	Don' t care	0	响应	MCAUSE 等
		Don' t care	1	响应	SCAUSE 等

10.1.2 中断请求与响应

当 PLIC 对特定中断目标存在有效中断请求，且优先级大于该中断目标的中断阈值时，会向该中断目标发起中断请求。

当该中断目标收到中断请求，且可响应该中断请求时，需要向 PLIC 发送中断响应消息。

中断响应机制如下：

- 中断目标向其对应的中断响应/完成寄存器发起一个读操作。该读操作将返回一个 ID，表示当前 PLIC 仲裁出的中断 ID。中断目标根据所获得的 ID 进行下一步处理。如果获得的中断 ID 为 0，表示没有有效中断请求，中断目标结束中断处理。
- 当 PLIC 收到中断目标发起的读操作，且返回相应 ID 后，会将该 ID 对应的中断源 IP 位清 0，且在中断处理完成之前屏蔽该中断源的后续采样。

10.1.3 中断完成

中断完成机制为：中断目标在完成中断处理后需要向 PLIC 发送中断完成消息，这个过程如下：

- 中断目标向中断响应/完成寄存器发起写操作，写操作的值为本次完成的中断 ID。如果中断类型为电平中断，需要在此之前清除外设中断源的有效中断信息。
- PLIC 收到该中断完成请求后，更新中断响应/完成寄存器，解除 ID 对应的中断源采样屏蔽，结束整个中断处理过程。

10.2 PLIC 寄存器地址映射

C906 为 PLIC 中断控制器保留分配了 64MB 的内存空间。其高 13 位地址由 SoC 集成使用 C906 时配置决定，低 27 位地址映射如表 10.2 所示。所有寄存器仅支持地址字对齐的访问。（PLIC 的寄存器要通过字访问指令（Load word 指令）访问，访问结果放在 64 位 GPR 的低 32 位。）

表 10.2: PLIC 寄存器地址空间映射

地址	名称	类型	初始值	描述
0x0000004	PLIC_PRIO1	R/W	0x0	1-10 23 号中断的优先级配置寄存器
0x0000008	PLIC_PRIO2	R/W	0x0	
0x000000C	PLIC_PRIO3	R/W	0x0	
...	
0x0000FFC	PLIC_PRIO1023	R/W	0x0	
0x0001000	PLIC_IP0	R/W	0x0	1-31 号中断的中断等待寄存器
0x0001004	PLIC_IP1	R/W	0x0	32-63 号中断的中断等待寄存器
...
0x000107C	PLIC_IP31	R/W	0x0	992- 1023 号中断的中断等待寄存器
Reserved	-	-	-	-
0x0002000	PLIC_H0_MIE0	R/W	0x0	1-31 号中断的机器模式中断使能寄存器
0x0002004	PLIC_H0_MIE1	R/W	0x0	32-63 号中断的机器模式中断使能寄存器
...
0x000207C	PLIC_H0_MIE31	R/W	0x0	992-1023 号中断的机器模式中断使能寄存器
0x0002080	PLIC_H0_SIE0	R/W	0x0	1-31 号中断的超级用户模式中断使能寄存器
0x0002084	PLIC_H0_SIE1	R/W	0x0	32-63 号中断的超级用户模式中断使能寄存器
...
0x00020FC	PLIC_H0_SIE31	R/W	0x0	992 -1023 号中断的超级用户模式中断使能寄存器
Reserved	-	-	-	-
0x01FFFFC	PLIC_CTRL	R/W	0x0	PLIC 权限控制寄存器
0x0200000	PLIC_H0_MTH	R/W	0x0	机器模式中断阈值寄存器
0x0200004	PLIC_H0_MCLAIM	R/W	0x0	机器模式中断响应/完成寄存器
Reserved	-	-	-	-
0x0201000	PLIC_H0_STH	R/W	0x0	超级用户模式中断阈值寄存器
0x0201004	PLIC_H0_SCLAIM	R/W	0x0	超级用户模式中断响应/完成寄存器
Reserved	-	-	-	-

10.3 中断优先级配置寄存器 (PLIC_PRIO)

每一个中断源均可通过修改地址映射的 32 位寄存器配置其优先级。寄存器当前值即为该中断源优先级。寄存器读写权限参考权限控制寄存器 (PLIC_CTRL) 描述。寄存器位分布和位定义如图 10.1 所示。

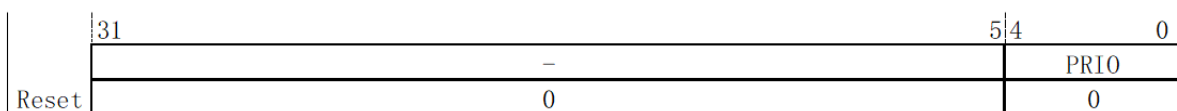


图 10.1: 中断优先级配置寄存器 (PLIC_PRIO)

PRIO: 中断优先级

优先级配置寄存器低 5 位可写，支持 32 个不同级别的优先级。其中优先级设置为 0 表示该中断无效。机器模式中中断优先级无条件高于超级用户模式中中断。当中断目标模式相同时，优先级 1 为最低优先级，优先级 31 为最高。当多个优先级相同的中断等待仲裁时，进一步比较中断源 ID，ID 较小的有高优先级。

10.4 中断等待寄存器 (PLIC_IP)

每一个中断源的等待状态都可以通过读取中断等待寄存器中的信息获取。对于中断 ID 为 N 的中断，其中断信息存储于 PLIC_IP x (x=N/32) 寄存器中的 IP y 上 (y = N mod 32)。其中 PLIC_IP0 寄存器的第 0 位固定绑 0。寄存器读写权限参考权限控制寄存器 (PLIC_PER) 描述。寄存器位分布和位定义如图 10.2 所示。

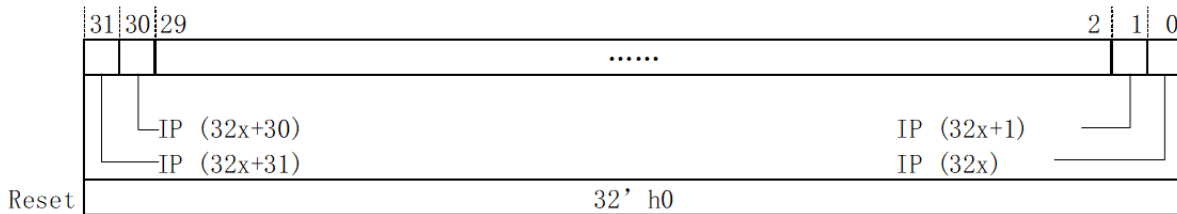


图 10.2: X 号中断等待寄存器 (PLIC_IPx)

IP——中断等待状态:

该位表示对应中断源的中断等待状态。

- 当 IP 位为 1 时，表示当前该外部中断源存在等待响应的中断。该位可通过内存存储指令 (SW 指令) 置 1。在对应中断源采样逻辑采样到有效电平或脉冲中断后也会将该位置 1。
- 当 IP 位为 0 时，表示当前该外部中断源没有等待响应的中断。该位可通过内存存储指令 (SW 指令) 清 0。当中断被响应后，PLIC 会将对应中断的 IP 位清除。

10.5 中断使能寄存器 (PLIC_IE)

每个中断目标对每个中断源均有一个中断使能位 (机器模式中中断使能和超级用户模式中中断使能)，可用于使能对应中断。其中机器模式中中断使能寄存器用于使能机器模式中中断，超级用户模式中中断使能寄存器用于使能超级用户模式中中断。

对于中断 ID 为 N 的中断，其中断使能信息存储于 PLIC_IE x (x=N/32) 寄存器中的 IE y 上 (y = N mod 32)。其中 ID0 对应的 IE 位固定绑 0。寄存器读写权限参考权限控制寄存器 (PLIC_PER) 描述。寄存器位分布和位定义如图 10.3 所示。

IE 中断使能:

该位表示对应中断源的中断使能状态。

- 当 IE 位为 1 时，表示中断对该目标使能。
- 当 IE 位为 0 时，表示中断对该目标屏蔽。

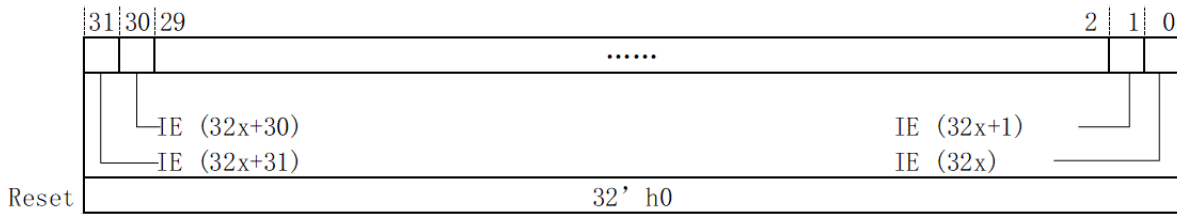


图 10.3: x 号中断使能寄存器 (PLIC_IEx)

10.6 PLIC 权限控制寄存器 (PLIC_CTRL)

PLIC 权限控制寄存器用于控制超级用户模式对 PLIC 部分寄存器的访问权限。

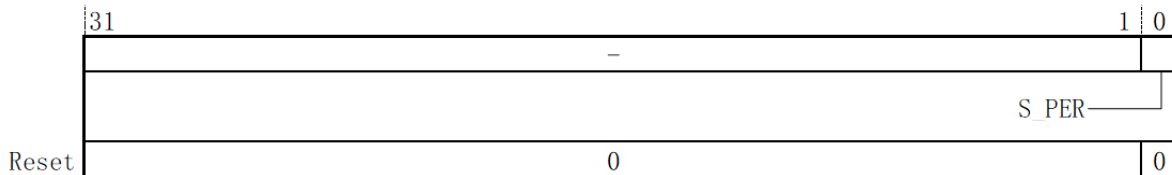


图 10.4: PLIC 权限控制寄存器 (PLIC_CTRL)

S_PER——访问权限控制位:

在 S_PER 为 0 时，仅机器模式拥有访问 PLIC 所有寄存器的权限。超级用户模式没有 PLIC_CTRL, PLIC_PRIO, PLIC_IP 和 PLIC_IE 寄存器的访问权限，仅能访问超级用户模式中断阈值寄存器和超级用户模式中断响应/完成寄存器。普通用户模式没有任何 PLIC 寄存器的访问权限。

在 S_PER 为 1 时，机器模式拥有访问 PLIC 所有寄存器的权限。超级用户模式拥有除 PLIC_CTRL 以外的所有 PLIC 寄存器访问权限。普通用户模式没有任何 PLIC 寄存器的访问权限。

10.7 中断阈值寄存器 (PLIC_TH)

每一个中断模式均有一个对应的中断阈值寄存器。仅有优先级大于中断阈值的有效中断才会向核内发起中断请求。寄存器读写权限参考 PLIC_CTRL 描述。寄存器位分布和位定义如 图 10.5 所示。

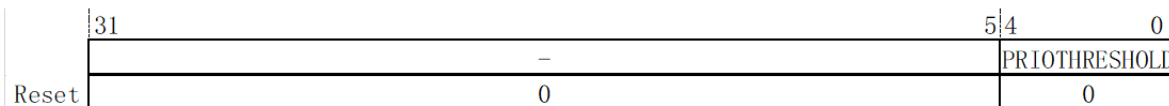


图 10.5: 中断阈值寄存器 (PLIC_TH)

PRIOTHRESHOLD——优先级阈值:

指示当前中断模式的中断阈值。阈值配置为 0，表示允许所有中断。

10.8 中断响应/完成寄存器 (PLIC_CLAIM)

每一个中断模式均有一个对应的中断响应/完成寄存器。该寄存器在 PLIC 完成仲裁时更新，更新值为 PLIC 本次仲裁结果的中断 ID。寄存器读写权限参考 PLIC_CTRL 描述。寄存器位分布和位定义如 图 10.6 所示。

	31	10	9	0
	-		CLAIM ID	
Reset	0		0	

图 10.6: 中断响应/完成寄存器 (PLIC_CLAIM)

CLAIM_ID——中断请求 ID:

对该寄存器的读操作：返回寄存器当前存储的 ID 值。该读操作表示对应 ID 的中断已开始处理。PLIC 开始中断响应处理参见中断请求与响应。

对该寄存器的写操作：表示对应 ID 的中断已完成处理，该写操作不会更新中断响应响应/完成寄存器。PLIC 开始中断完成处理中断完成。

第十一章 调试接口

11.1 概述

调试接口是软件与处理器交互的通道。用户可以通过调试接口获取 CPU 的寄存器和存储器里的内容等信息，包括其他的片上设备信息。此外，程序下载等操作也可以通过调试接口完成。

C906 支持兼容 IEEE-1149.1 标准的 JTAG 通信协议（通称 JTAG5），可以同已有的 JTAG 部件或独立的 JTAG 控制器集成在一起。

调试接口的主要特性如下：

- 使用标准的 JTAG 接口进行调试；
- 支持同步调试和异步调试，保证在极端恶劣情况下使处理器进入调试模式；
- 支持软断点；
- 可以设置两个内存断点和硬断点；
- 检查和设置 CPU 寄存器的值；
- 检查和设置内存值；
- 可进行指令单步执行或多步执行；
- 快速下载程序；

C906 的调试工作是调试软件，调试代理服务程序，调试器和调试接口一起配合完成的，调试接口在整个 CPU 调试环境中的位置如 图 11.1 所示。其中，调试软件和调试代理服务程序通过网络互联，调试代理服务程序与调试器通过 USB 连接，调试器与 CPU 的调试接口以 JTAG 模式通信。

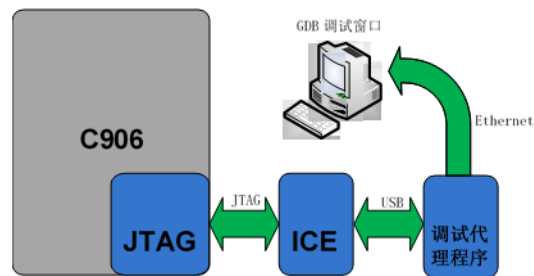


图 11.1: 调试接口在整个 CPU 调试环境中的位置

11.2 接口描述

调试模块与外部的接口主要是与 JTAG 相关的接口信号和调试相关的接口信号。表 11.1 列出了与调试相关的接口信号。

表 11.1: 调试模块与外部的接口信号

信号名	方向
pad_core0_dbgrq_b	输入
core0_pad_jdb_pm[1:0]	输出
pad_had_jtg_tclk	输入
pad_had_jtg_trst_b	输入
pad_had_jtg_tdi	输入
had_pad_jtg_tdo	输出
had_pad_jtg_tdo_en	输出
pad_had_jtg_tms	输入

1. pad_core0_dbgrq_b

pad_core_dbgrq_b 为同步进入调试模式请求信号，对低电平有效。该信号是 CORE 的外部输入信号，在 CORE 中被系统时钟同步之后输入到 HAD 中，HAD 使用该信号欲让 CORE 同步进入调试模式。拉低该信号与设置 HAD 寄存器 HCR 的 DR 位有相同的作用。

2. core0_pad_jdb_pm[1:0]

core_pad_jdb_pm[1:0] 信号指示对应 CORE 当前工作模式，可以通过该信号确定 CPU 是否已进入调试模式。具体如表 11.2 所示。

表 11.2: PM 指示的当前 CPU 状态

core_pad_jdb_pm[1:0]	说明
00	普通模式
01	低功耗模式
10	调试模式
11	保留

3. pad_had_jtg_tclk

pad_had_jtg_tclk 为 JTAG 时钟信号。该信号为外部输入信号，一般为调试器产生的时钟信号，要保证该时钟信号的频率低于 CPU 时钟信号的频率 1/2 才能保证调试模块与 CORE 之间的正常工作。

4. pad_had_jtg_trst_b

pad_had_jtg_trst_b 信号为 JTAG 复位信号，可以复位 TAP 状态机及其他相关控制信号。

5. JTAG5 相关信号

pad_had_jtg_tdi 信号为 HAD 端 JTAG 串行输入信号，HAD 端在 JTAG 时钟信号 tclk 的上升沿对其采样，而外部调试器在 JTAG 时钟的下降沿设置该信号；

pad_had_jtg_tms 信号为 JTAG 模式选择信号，该信号由调试器发出，用于控制 HAD 中 TAP 状态机的运作。

had_pad_jtg_tdo 信号为 HAD 端 JTAG 串行输出信号，HAD 端在 JTAG 时钟信号 tclk 的下降沿对其设置，而外部调试器在 JTAG 时钟的上升沿对其采样；

had_pad_jtg_tdo_en 信号表示 had_pad_jtg_tdo 信号有效，通常外部调试器监视该信号以确定 had_pad_jtg_tdo 信号是否有效，调试器也可以不观察该信号而通过调试器内部的 TAP 状态机的状态以决定对 had_pad_jtg_tdo 信号的采样。该信号主要用作在实现多个 TAP 状态机时，确定是哪个 JTAG 的输出。

平头哥半导体

第十二章 性能监测单元

C906 性能监测单元 (HPM) 遵从 RISC-V 标准, 用于统计程序运行中的软件信息和部分硬件信息, 供软件开发人员进行程序优化。

性能监测单元统计的软硬件信息分为以下几类:

- 处理器运行时钟周期数和时间;
- 指令信息统计;
- 处理器关键部件信息统计。

12.1 性能监测控制寄存器

HPM 控制寄存器主要为: 机器模式计数器访问授权寄存器 (MCOUNTEREN)、超级用户模式计数器访问授权寄存器 (SCOUNTEREN)、机器模式计数禁止寄存器 (MCOUNTINHIBIT)、机器模式计数器写使能授权寄存器 (MCOUNTERWEN)。

12.1.1 机器模式计数器访问授权寄存器 (MCOUNTEREN)

机器模式计数器访问授权寄存器 (MCOUNTEREN), 用于授权超级用户模式是否可以访问机器模式计数器的镜像寄存器。

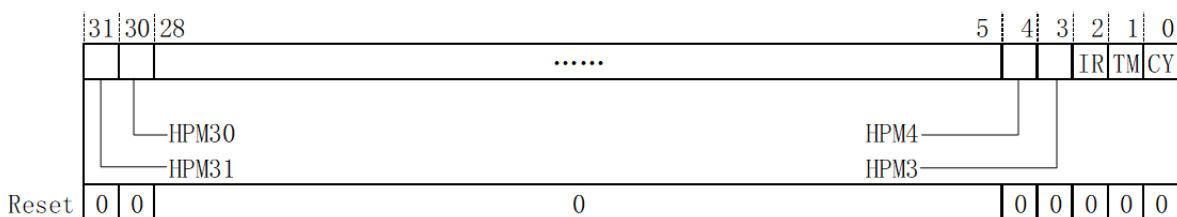


图 12.1: 机器模式计数器访问授权寄存器 (MCOUNTEREN)

机器模式计数器访问授权寄存器说明如表 12.1 所示。

表 12.1: 机器模式计数器访问授权寄存器说明

位	读写	名称	介绍
31: 18	读写	NA	保留
17: 3	读写	HPM n	SHPMCOUNTER n 寄存器 S-mode 访问位: 0: S-mode 访问 SHPMCOUNTER n 将触发非法指令异常; 1: S-mode 能正常访问 SHPMCOUNTER n 。
2	读写	IR	SINSTRET 寄存器 S-mode 访问位: 0: S-mode 访问 SINSTRET 寄存器将触发非法指令异常; 1: S-mode 能正常访问 SINSTRET 寄存器。
1	读写	TM	TIME 寄存器 S-mode 访问位: 0: S-mode 访问 TIME 寄存器将触发非法指令异常; 1: S-mode 能正常读取 TIME 寄存器。
0	读写	CY	SCYCLE 寄存器 S-mode 访问位: 0: S-mode 访问 SCYCLE 寄存器将触发非法指令异常; 1: S-mode 能正常访问 SCYCLE 寄存器。

12.1.2 超级用户模式计数器访问授权寄存器 (SCOUNTEREN)

超级用户模式计数器访问授权寄存器 (SCOUNTEREN)，用于授权用户模式是否可以访问机器模式计数器的镜像寄存器。

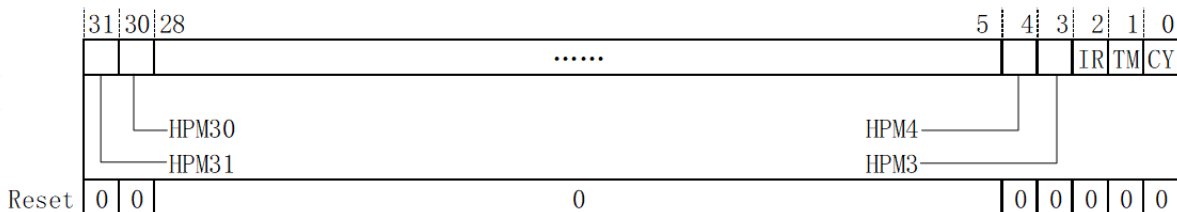


图 12.2: 超级用户模式计数器访问授权寄存器 (SCOUNTEREN)

超级用户模式计数器访问授权寄存器说明如表 12.2 所示。

表 12.2: 超级用户模式计数器访问授权寄存器说明

位	读写	名称	介绍
31: 18	读写	NA	保留
17: 3	读写	HPM n	HPMCOUNTER n 寄存器 U-mode 访问位: 0: U-mode 访问 HPMCOUNTER n 将触发非法指令异常; 1: U-mode 能正常访问 HPMCOUNTER n 。
2	读写	IR	INSTRET 寄存器 U-mode 访问位: 0: U-mode 访问 INSTRET 寄存器将触发非法指令异常; 1: U-mode 能正常访问 INSTRET 寄存器。
1	读写	TM	TIME 寄存器 U-mode 访问位: 0: U-mode 访问 TIME 寄存器将触发非法指令异常; 1: U-mode 能正常访问 TIME 寄存器。
0	读写	CY	CYCLE 寄存器 U-mode 访问位: 0: U-mode 访问 CYCLE 寄存器将触发非法指令异常; 1: U-mode 能正常访问 CYCLE 寄存器。

12.1.3 机器模式计数禁止寄存器 (MCOUNTINHIBIT)

机器模式禁止计数寄存器 (MCOUNTINHIBIT)，可以禁止机器模式计数器计数。在不需性能分析的场景下，关闭计数器，可以降低处理器功耗。

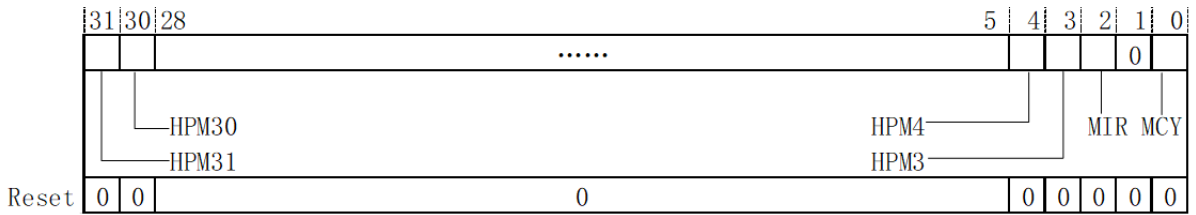


图 12.3: 机器模式计数禁止寄存器 (MCOUNTINHIBIT)

机器模式禁止计数寄存器说明如表 12.3 所示。按照 RISC-V 标准，MTIME 是系统中实现的计时器，因此 C906 无权禁止它的计时，所以 MCOUNTINHIT[1] 按照标准未实现。

表 12.3: 机器模式计数禁止授权寄存器寄存器说明

位	读写	名称	介绍
31: 18	读写	NA	保留
17: 3	读写	MHPM _n	MHPMCOUNTER _n 寄存器禁止计数位: 0: 正常计数; 1: 禁止计数。
2	读写	MIR	MINSTRET 寄存器禁止计数位: 0: 正常计数; 1: 禁止计数。
1	-	-	-
0	读写	MCY	MCYCLE 寄存器禁止计数位: 0: 正常计数; 1: 禁止计数。

12.1.4 机器模式计数器写使能授权寄存器 (MCOUNTERWEN)

机器模式计数器写使能授权寄存器 (MCOUNTERWEN)，用于授权超级用户模式是否可以写机器模式镜像事件计数器。该寄存器为 C906 机器模式扩展实现寄存器。

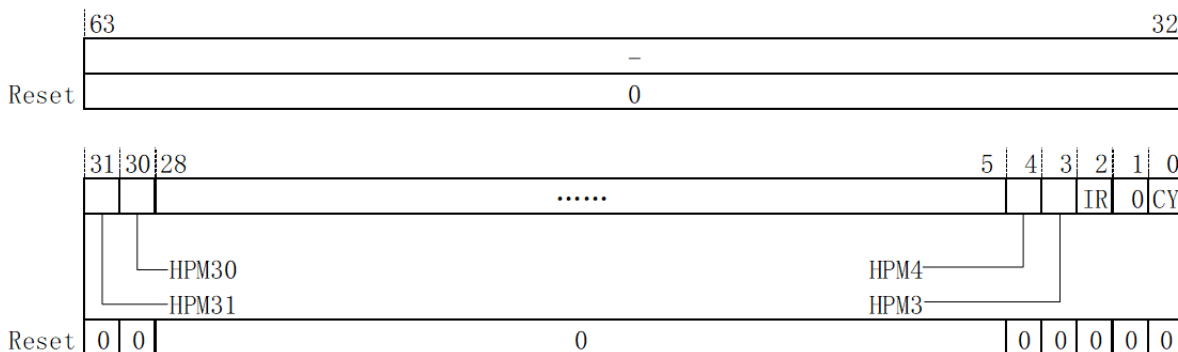


图 12.4: 机器模式计数器扩展写使能寄存器 (MCOUNTERWEN)

当 MCOUNTERWEN.bit[n] 为 1 时，允许超级用户模式下写对应 SHPMCOUNTER_n、SINSTERT、SCYCLE，对这些寄存器的写操作会直接改写 MHPMCOUNTER_n、MINSTRET、MCYCLE；

当 MCOUNTERWEN.bit[n] 为 0 时，不允许超级用户模式下写对应的 SHPMCOUNTER_n、SINSTERT、SCYCLE，否则会触发非法指令异常。

12.2 性能监测事件选择寄存器

性能监测事件选择器 (MHPMEVENT3-17)，用于选择每个计数器对应的计数事件。C906 中每个计数器对应一个事件，对应关系固定，不能修改，因此每个事件选择器只能写入对应的事件索引值。只有当事件选择器写入对应事件的索引值后才能正常计数。

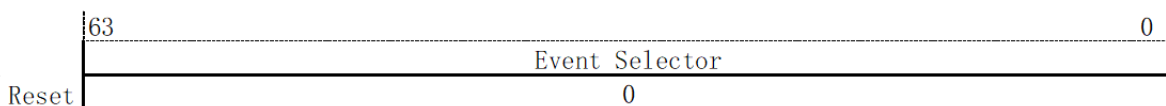


图 12.5: 机器模式性能监测事件选择寄存器 (MHPMEVENT)

机器模式性能监测事件选择寄存器说明如表 12.4 所示。

表 12.4: 机器模式性能监测事件选择寄存器说明

位	读写	名称	介绍
63:0	读写	事件索引号	性能监测事件索引: 0: 没有事件; 0x1~0xF: 硬件实现的性能监测事件, 具体见表 12.5; >=0x10: 硬件未定义实现的性能监测事件。

表 12.5 事件选择器和事件以及事件计数器之间的对应关系。

表 12.5: 计数器事件对应列表

事件选择器	索引	事件	计数器
MHPMEVENT3	0x1	L1 ICACHE Access Counter	MHPMCOUNTER3
MHPMEVENT4	0x2	L1 ICACHE Miss Counter	MHPMCOUNTER4
MHPMEVENT5	0x3	I-uTLB Miss Counter	MHPMCOUNTER5
MHPMEVENT6	0x4	D-uTLB Miss Counter	MHPMCOUNTER6
MHPMEVENT7	0x5	jTLB Miss Counter	MHPMCOUNTER7
MHPMEVENT8	0x6	Conditional Branch Mispredict Counter	MHPMCOUNTER8
MHPMEVENT9	0x7	Conditional Branch instruction counter	MHPMCOUNTER9
MHPMEVENT10-12	0x9	未定义实现	MHPMCOUNTER10-12
MHPMEVENT13	0xB	Store Instruction Counter	MHPMCOUNTER13
MHPMEVENT14	0xC	L1 DCACHE read access Counter	MHPMCOUNTER14
MHPMEVENT15	0xD	L1 DCACHE read miss Counter	MHPMCOUNTER15
MHPMEVENT16	0xE	L1 DCACHE write access Counter	MHPMCOUNTER16
MHPMEVENT17	0xF	L1 DCACHE write miss Counter	MHPMCOUNTER17
MHPMEVENT18~31	≥0x10	暂未定义实现	MHPMCOUNTER18~31

12.3 事件计数器

事件计数器有三组, 分别为: 机器模式事件计数器、用户模式事件计数器和 C906 扩展的超级用户模式事件计数器。具体如表 12.6 所示。

表 12.6: 机器模式事件计数器列表

名称	索引	读写	初始值	介绍
MCYCLE	0xB00	MRW	0x0	cycle counter
MINSTRET	0xB02	MRW	0x0	instructions-retired counter
MHPMCOUNTER3	0xB03	MRW	0x0	performance-monitoring counter
MHPMCOUNTER4	0xB04	MRW	0x0	performance-monitoring counter
...
MHPMCOUNTER17	0xB11	MRW	0x0	performance-monitoring counter

用户模式事件计数器列表如表 12.7 所示。

表 12.7: 用户模式事件计数器列表

名称	索引	读写	初始值	介绍
CYCLE	0xC00	URO	0x0	cycle counter
TIME	0xC01	URO	0x0	timer
INSTRET	0xC02	URO	0x0	instructions-retired counter
HPMCOUNTER3	0xC03	URO	0x0	performance-monitoring counter
HPMCOUNTER4	0xC04	URO	0x0	performance-monitoring counter
...
HPMCOUNTER17	0xC11	URO	0x0	performance-monitoring counter

超级用户模式事件计数器列表如表 12.8 所示。

表 12.8: 超级用户模式事件计数器列表

名称	索引	读写	初始值	介绍
SCYCLE	0x5E0	SRW	0x0	cycle counter
SINSTRET	0x5E2	SRW	0x0	instructions-retired counter
SHPMCOUNTER3	0x5E3	SRW	0x0	performance-monitoring counter3
SHPMCOUNTER4	0x5E4	SRW	0x0	performance-monitoring counter4
...
SHPMCOUNTER17	0x5F1	SRW	0x0	performance-monitoring counter17

其中，用户模式的 CYCLE、INSTRET 和 HPMCOUNTER_n 是对应机器模式事件计数器的只读映射，TIMER 计数器是 MTIME 寄存器的只读映射；超级用户模式的 SCYCLE、SINSTRET 和 SHPMCOUNTER_n 是对应机器模式事件计数器的映射。

12.4 性能检测单元事件溢出中断

C906 扩展实现了性能监测单元机器模式事件计数器上溢出标注寄存器 (MCOUNTEROF) 和机器模式事件计数器溢出中断使能寄存器 (MCOUNTERINTEN)。该寄存器机器模式可读写，非机器模式访问将触发非法指令异常。

机器模式事件上溢寄存器 (MCOUNTEROF) 中各个比特与事件计数器一一对应, 分别用于表示各事件计数是否发生了上溢出; 机器模式事件中断使能寄存器 (MCOUNTERINTEN) 中各个比特与事件计数器一一对应, 分别用于控制对应事件计数器发生上溢出情况下是否会发起中断请求。

由性能监测单元发起的溢出中断统一中断向量号为 17。中断模式为超级用户模式中断, 可以被 Delegate 到超级用户模式响应, 详见异常处理。

平头哥半导体

第十三章 程序示例

本附录主要介绍多种程序示例，包含：MMU 设置示例、PMP 设置示例、高速缓存设置示例、PLIC 设置示例和 HPM 设置示例。

13.1 MMU 设置示例

```
/******  
  
* Function: An example of set C906 MMU.  
* Memory space: Virtual address <-> physical address.  
*  
* Pagesize 4K: vpn: {vpn2,vpn1,vpn0} <-> ppn: {ppn2,ppn1,ppn0}  
* Pagesize 2M: vpn: {vpn2,vpn1}<-> ppn:{ppn2,ppn1}  
* Pagesize 1G: vpn: {vpn2} <-> ppn: {ppn2}  
*  
*****/  
  
/*C906 will invalid all MMU TLB Entry automatically when reset*/  
/*you can use sfence.vma to invalid all MMU TLB Entry if necessarily*/  
sfence.vma x0, x0  
  
/* Pagesize 4K: vpn: {vpn2, vpn1, vpn0} <-> ppn: {ppn2, ppn1, ppn0}*/  
/*first-level page addr base: PPN (defined in satp)*/  
/*second-level page addr base: BASE2 (self define)*/  
/*third-level page addr base: BASE3 (self define)*/  
/*1. get first-level page addr base: PPN and vpn*/  
/* get PPN*/  
csrr x3, satp  
li x4, 0xffffffff  
and x3, x3, x4  
  
/*2. cfig first-level page*/  
/*first-level page addr: {PPN, vpn2, 3' b0}, first-level page pte:{ 44' b BASE2, 10' b1} */  
/*get first-level page addr*/
```

(下页继续)

(续上页)

```
slli x3, x3, 12
/*get vpn2*/
li x4, VPN
li x5, 0x7fc0000
and x4, x4, x5
srli x4, x4, 15
and x5, x3, x4
/*store pte at first-level page addr*/
li x6, {44' b BASE2, 10' b1}
sd x6, 0(x5)

/*3. cfig second-level page*/
/*second-level page addr: {BASE2, vpn1, 3' b0}, second-level page pte:{ 44' b BASE3, 10' b1} */
/*get second-level page addr*/
/\* VPN1*/
li x4, VPN
li x5, 0x3fe00
and x4, x4, x5
srli x4, x4, 9
/*BASE2*/
li x5, BASE2
srli x5, x5, 12
and x5, x5, x4
/*store pte at second-level page addr*
li x6, {44' b BASE3, 10' b1}
sd x6, 0(x5)

/*4. cfig third-level page*/
/*third-level page addr: {BASE3, vpn0, 3' b0}, third-level page pte:{
theadflag, ppn2, ppn1, ppn0, 9' b flags, 1' b1} */
/*get second-level page addr*/
/* VPN0*/
li x4, VPN
li x5, 0x1fff
and x4, x4, x5
srli x4, x4, 3
/*BASE3*/
li x5, BASE3
srli x5, x5, 12
and x5, x5, x4
/*store pte at second-level page addr*/
li x6, { theadflag, ppn2, ppn1, ppn0, 9' b flags, 1' b1}
sd x6, 0(x5)
```

(下页继续)

(续上页)

```
/* Pagesize 2M: vpn: {vpn2, vpn1} <-> ppn: {ppn2, ppn1}*/
/*first-level page addr base: PPN (defined in satp)*/
/*second-level page addr base: BASE2 (self define)*/

/*1. get first-level page addr base: PPN and vpn*/
/* get PPN*/
csrr x3, satp
li x4, 0xffffffff
and x3, x3, x4

/*2. cfig first-level page*/
/*first-level page addr: {PPN, vpn2, 3' b0}, first-level page pte:{ 44' b
BASE2, 10' b1}*/
/*get first-level page addr*/
slli x3, x3, 12
/*get vpn2*/
li x4, VPN
li x5, 0x7fc0000
and x4, x4, x5
srli x4, x4, 15
and x5, x3, x4
/*store pte at first-level page addr*/
li x6, {44' b BASE2, 10' b1}
sd x6, 0(x5)

/*3. cfig second-level page*/
/*second-level page addr: {BASE2, vpn1, 3' b0}, second-level page pte:{
theadflag, ppn2, ppn1, 9' b0, 9' b flags,1' b1} */
/*get second-level page addr*/
/* VPN1*/
li x4, VPN
li x5, 0x3fe00
and x4, x4, x5
srli x4, x4, 9
/*BASE2*/
li x5, BASE2
srli x5, x5, 12
and x5, x5, x4
/*store pte at second-level page addr*/
li x6, { theadflag, ppn2, ppn1, 9' b0, 9' b flags,1' b1}
sd x6, 0(x5)
```

(下页继续)

(续上页)

```

/* Pagesize 1G: vpn: {vpn2} <-> ppn: {ppn2}*/
/*first-level page addr base: PPN (defined in satp)*/
/*1. get first-level page addr base: PPN and vpn*/
/* get PPN*/
csrr x3, satp
li x4, 0xffffffff
and x3, x3, x4

/*2. cfig first-level page*/
/*first-level page addr: {PPN, vpn2, 3' b0}, first-level page pte:{
theadflag, ppn2, 9' b0, 9' b0, 9' b flags,1' b1}*/
/*get first-level page addr*/
slli x3, x3, 12
/*get vpn2*/
li x4, VPN
li x5, 0x7fc0000
and x4, x4, x5
srli x4, x4, 15
and x5, x3, x4
/*store pte at first-level page addr*/
li x6, { theadflag, ppn2, 9' b0, 9' b0, 9' b flags,1' b1}
sd x6, 0(x5)
    
```

13.2 PMP 设置示例

```

/*****
* Function: An example of set C906 PMP.
* 0x0 ~ 0xf0000000, TOR 模式, RWX
* 0xf0000000 ~ 0xf8000000, NAPOT 模式, RW
*0xfff73000 ~ 0xfff74000, NAPOT 模式, RW
*0xfffc0000 ~ 0xfffc2000, NAPOT 模式, RW
* 以上 4 片区域配置了不同的执行权限。另外, 为了防止 CPU 在不同模式下投机执行到不支持的地址区域, 尤其是在
默认拥有全部执行权限的机器模式下, 需要对 PMP 进行相关配置。具体来说, 就
是配置完需要执行权限的地址区域后, 将剩余地址区域配置成无任何权限, 如下例所示。
*****/

# pmpaddr0,0x0 ~ 0xf0000000, TOR 模式, 读写可执行权限
li x3, (0xf0000000 >> 2)
csrw pmpaddr0, x3
# pmpaddr1,0xf0000000 ~ 0xf8000000, NAPOT 模式, 读写权限
li x3, ( 0xf0000000 >> 2 | (0x8000000-1) >> 3))
csrw pmpaddr1, x3
    
```

(下页继续)

(续上页)

```

# pmpaddr2,0xffff73000 ~ 0xffff74000, NAPOT 模式, 读写权限
li x3, ( 0xffff73000 >> 2 | (0x1000-1) >> 3))
csrw pmpaddr2, x3
# pmpaddr3,0xfffc0000 ~ 0xfffc2000, NAPOT 模式, 读写权限
li x3, ( 0xfffc0000 >> 2 | (0x2000-1) >> 3))
csrw pmpaddr3, x3
# pmpaddr4,0xf0000000 ~ 0x100000000, NAPOT 模式, 无任何权限
li x3, ( 0xf0000000 >> 2 | (0x10000000-1) >> 3))
csrw pmpaddr4, x3
# pmpaddr5,0x100000000 ~ 0xffffffff, TOR 模式, 无任何权限
li x3, (0xffffffff >> 2)
csrw pmpaddr5, x3
# PMPCFG0, 配置各表项执行权限/模式/lock 位,
lock 为 1 时, 该表项在机器模式下才有效
li x3,0x88989b9b9b8f
csrw pmpcfg0, x3
# pmpaddr5,0x100000000 ~ 0xffffffff, TOR 模式, 0x100000000 <= addr <
0xffffffff 时都会命中 pmpaddr5, 但是 0xffffffff000 ~
0xffffffff 地址区间无法命中 pmpaddr5 (C906 中 PMP 的最小粒度为 4K), 如果需要屏蔽 1T 空间的最后一个
4K 空间, 需要再配置一个 NAPOT 模式的表项。

```

13.3 高速缓存设置示例

13.3.1 高速缓存的开启示例

```

/*C906 will invalid all I-cache automatically when reset*/
/*you can invalid I-cache by yourself if necessarily*/
/*invalid I-cache*/
li x3, 0x33
csrc mcor, x3
li x3, 0x11
csrs mcor, x3
// it can also use icache instrucion to replace the invalid sequence if theadisae is enabled.
//icache.iall
//sync.is
/*enable I-cache*/
li x3, 0x1
csrs mhcr, x3
/*C906 will invalid all D-cache automatically when reset*/
/*you can invalid D-cache by yourself if necessarily*/
/*invalid D-cache*/

```

(下页继续)

(续上页)

```
li x3, 0x33
csrc mcor, x3
li x3, 0x12
csrs mcor, x3
// it can also use dcache instrucion to replace the invalid sequence if theadisaee is enabled.
// dcache.iall
// sync.is
/*enable D-cache*/
li x3, 0x2
csrs mhcr, x3
/*C906 will invalidate all D-cache automatically when reset*/
/*you can invalid D-cache by yourself if necessarily*/
```

13.3.2 指令高速缓存与数据高速缓存的同步示例

CPU0

```
sd x3,0(x4) // a new instruction defined in x3
           // is stored to program memory address defined in x4.
dcache.cval1 r0 // clean the new instrcution.
sync.s         // ensure completion of clean operation.
icache.iva r0 // invalid icache according to shareable configuraiton.
sync.s/fence.i // ensure completion in all CPUs.
sd x5,0(x6)    // set flag to signal operation completion.
sync.is
jr x4 // jmp to new code
```

13.3.3 TLB 与数据高速缓存的同步示例

CPU0

```
sd x4,0(x3) // update a new translation table entry
sync.is/fence.i // ensure completion of update operation.
sfence.vma x5,x0 // invalid the TLB by va
sync.is/fence.i // ensure completion of TLB invalidation and
                // synchronises context
```

13.4 PLIC 设置示例

```
//Init id 1 machine mode int for hart 0
/*1.set hart threshold if needed*/
li x3, (plic_base_addr + 0x200000) // h0 mthreshold addr
li x4, 0xa //threshold value
sw x4,0x0(x3) // set hart0 threshold as 0xa

/*2.set priority for int id 1*/
li x3, (plic_base_addr + 0x0) // int id 1 prio addr
li x4, 0x1f // prio value
sw x4,0x4(x3) // init id1 priority as 0x1f

/*3.enable m-mode int id1 to hart*/
li x3, (plic_base_addr + 0x2000) // h0 mie0 addr
li x4, 0x2
sw x4,0x0(x3) // enable int id1 to hart0

/*4.set ip or wait external int*/
/*following code set ip*/
li x3, (plic_base_addr + 0x1000) // h0 mthreshold addr
li x4, 0x2 // id 1 pending
sw x4, 0x0(x3) // set int id1 pending

/*5.core enters interrupt handler, read PLIC_CLAIM and get ID*/

/*6.core takes interrupt*/

/*7.core needs to clear external interrupt source if LEVEL(not PULSE)
configured, then core writes ID to PLIC_CLAIM and exits interrupt*/
```

13.5 HPM 设置示例

```
/*1.inhibit counters counting*/
li x3, 0xffffffff
csw mcountinhibit, x3

/*2.C906 will initial all HPM counters when reset*/
/*you can initial HPM counters manually if necessarily*/
csw mcycle, x0
csw minstret, x0
csw mhpcounter3, x0
.....
csw mhpcounter31, x0
```

(下页继续)

(续上页)

```
/*3.configure mhpmevent*/  
li x3, 0x1  
csrw mhpmevent3, x3 // mhpcounter3 count event: L1 ICache Access Counter  
li x3, 0x2  
csrw mhpmevent4, x3 // mhpcounter4 count event: L1 ICache Miss Counter  
.....  
  
/*4. configure mcounteren and scounteren*/  
li x3, 0xffffffff  
csrw mcounteren, x3 // enable super mode to read hpmcounter  
li x3, 0xffffffff  
csrw scounteren, x3 // enable user mode to read hpmcounter  
  
/*5. enable counters to count when you want*/  
csrw mcountinhibit, x0
```

第十四章 附录 A 标准指令术语

C906 实现了 RV64IMAFC 指令集包，以下各章节按照不同指令集对每条指令做具体描述。

14.1 附录 A-1 I 指令术语

以下是对 C906 实现的 RISC-V I 指令集的具体描述，指令按英文字母顺序排列。

本节指令位宽默认为 32 位，但是系统在特定情况下会将某些指令汇编成 16 位的压缩指令，关于压缩指令具体描述请见附录 A-6 C 指令术语。

14.1.1 ADD——有符号加法指令

语法：

add rd, rs1, rs2

操作：

$rd \leftarrow rs1 + rs2$

执行权限：

M mode/S mode/U mode

异常：

无

指令格式：

31	25	24	20	19	15	14	12	11	7	6	0
0000000			rs2		rs1		000		rd		0110011

14.1.2 ADDI——有符号立即数加法指令

语法：

addi rd, rs1, imm12

操作:

$$rd \leftarrow rs1 + \text{sign_extend}(imm12)$$

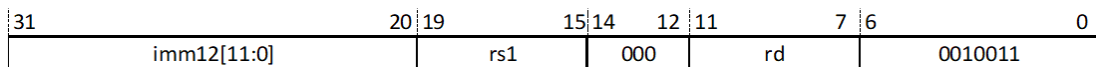
执行权限:

M mode/S mode/U mode

异常:

无

指令格式:



14.1.3 ADDIW——低 32 位有符号立即数加法指令

语法:

addiw rd, rs1, imm12

操作:

$$\begin{aligned} tmp[31:0] &\leftarrow rs1[31:0] + \text{sign_extend}(imm12)[31:0] \\ rd &\leftarrow \text{sign_extend}(tmp[31:0]) \end{aligned}$$

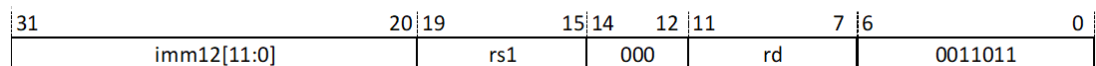
执行权限:

M mode/S mode/U mode

异常:

无

指令格式:



14.1.4 ADDW——低 32 位有符号加法指令

语法:

addw rd, rs1, rs2

操作:

$$\begin{aligned} tmp[31:0] &\leftarrow rs1[31:0] + rs2[31:0] \\ rd &\leftarrow \text{sign_extend}(tmp[31:0]) \end{aligned}$$

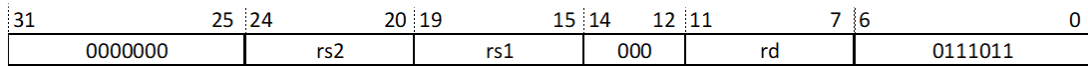
执行权限:

M mode/S mode/U mode

异常:

无

指令格式:



14.1.5 AND——按位与指令

语法:

and rd, rs1, rs2

操作:

$rd \leftarrow rs1 \& rs2$

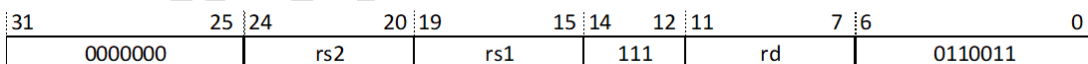
执行权限:

M mode/S mode/U mode

异常:

无

指令格式:



14.1.6 ANDI——立即数按位与指令

语法:

andi rd, rs1, imm12

操作:

$rd \leftarrow rs1 \& \text{sign_extend}(\text{imm12})$

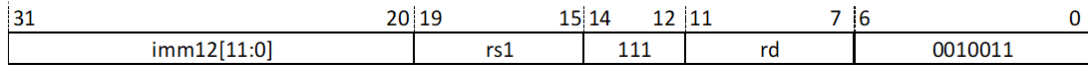
执行权限:

M mode/S mode/U mode

异常:

无

指令格式:



14.1.7 AUIPC——PC 高位立即数加法指令

语法:

auipc rd, imm20

操作:

$rd \leftarrow \text{current pc} + \text{sign_extend}(\text{imm20} \ll 12)$

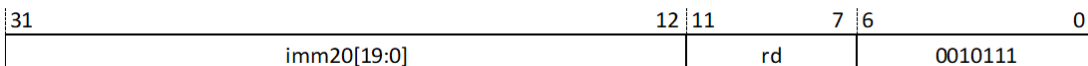
执行权限:

M mode/S mode/U mode

异常:

无

指令格式:



14.1.8 BEQ——相等分支指令

语法:

beq rs1, rs2, label

操作:

if (rs1 == rs2)
 next pc = current pc + sign_extend(imm12 << 1)
 else
 next pc = current pc + 4

执行权限:

M mode/S mode/U mode

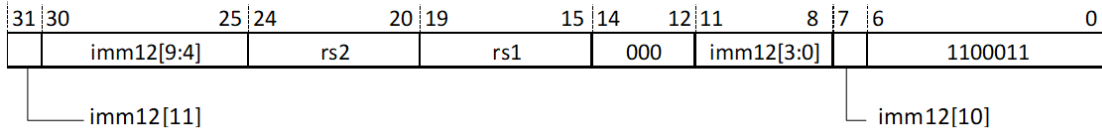
异常:

无

说明:

汇编器根据 label 算出 imm12
指令跳转范围为±4KB 地址空间

指令格式:



14.1.9 BGE——有符号大于等于分支指令

语法:

bge rs1, rs2, label

操作:

```
if (rs1 >= rs2)
    next pc = current pc + sign_extend(imm12 <<1)
else
    next pc = current pc + 4
```

执行权限:

M mode/S mode/U mode

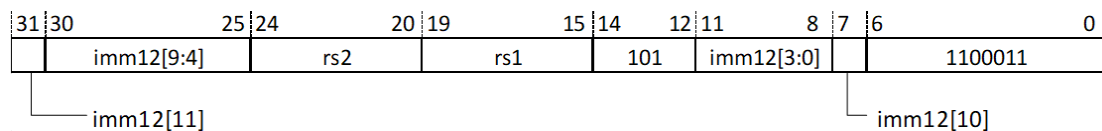
异常:

无

说明:

汇编器根据 label 算出 imm12
指令跳转范围为±4KB 地址空间

指令格式:



14.1.10 BGEU——无符号大于等于分支指令

语法:

bgeu rs1, rs2, label

操作:

```
if (rs1 >= rs2)
    next pc = current pc + sign_extend(imm12<<1)
else
    next pc = current pc + 4
```

执行权限:

M mode/S mode/U mode

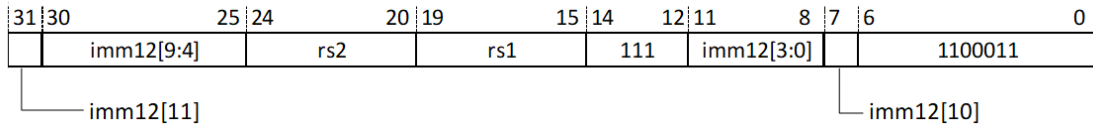
异常:

无

说明:

汇编器根据 label 算出 imm12
指令跳转范围为±4KB 地址空间

指令格式:



14.1.11 BLT——有符号小于分支指令

语法:

blt rs1, rs2, label

操作:

```
if (rs1 < rs2)
    next pc = current pc + sign_extend(imm12<<1)
else
    next pc = current pc + 4
```

执行权限:

M mode/S mode/U mode

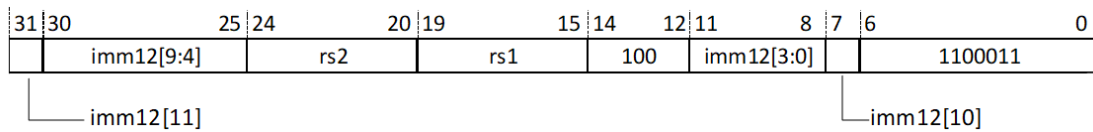
异常:

无

说明:

汇编器根据 label 算出 imm12
指令跳转范围为±4KB 地址空间

指令格式:



14.1.12 BLTU——无符号小于分支指令

语法:

bltu rs1, rs2, label

操作:

```
if (rs1 < rs2)
    next pc = current pc + sign_extend(imm12<<1)
else
    next pc = current pc + 4
```

执行权限:

M mode/S mode/U mode

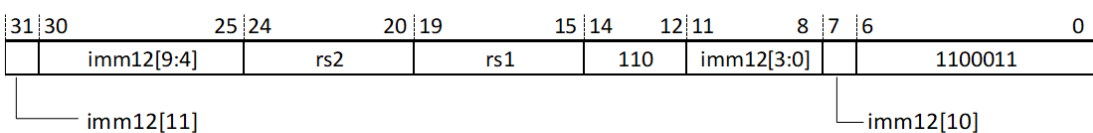
异常:

无

说明:

汇编器根据 label 算出 imm12
指令跳转范围为±4KB 地址空间

指令格式:



14.1.13 BNE——不等分支指令

语法:

bne rs1, rs2, label

操作:

```
if (rs1 != rs2)
    next pc = current pc + sign_extend(imm12<<1)
else
    next pc = current pc + 4
```

执行权限:

M mode/S mode/U mode

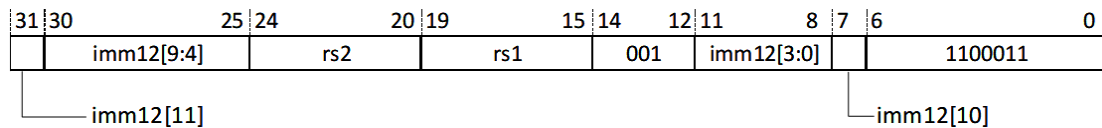
异常:

无

说明:

汇编器根据 label 算出 imm12
指令跳转范围为±4KB 地址空间

指令格式:



14.1.14 CSRRC——控制寄存器清零传送指令

语法:

csrcc rd, csr, rs1

操作:

```
rd ← csr
csr ← csr & (~rs1)
```

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

说明：

各权限下允许访问的控制寄存器不同，具体请参考控制寄存器章节。
当 $rs1=x0$ 时，该指令不产生写操作，不会产生写行为引发的异常。

指令格式：

31	20	19	15	14	12	11	7	6	0	
csr			rs1		011		rd		1110011	

14.1.15 CSRRCI——控制寄存器立即数清零传送指令

语法：

`csrrci rd, csr, imm5`

操作：

$rd \leftarrow csr$
 $csr \leftarrow csr \& \sim zero_extend(imm5)$

执行权限：

M mode/S mode/U mode

异常：

非法指令异常

说明：

各权限下允许访问的控制寄存器不同，具体请参考控制寄存器章节。

指令格式：

31	20	19	15	14	12	11	7	6	0	
csr			imm5		111		rd		1110011	

14.1.16 CSRRS——控制寄存器置位传送指令

语法：

`csrrs rd, csr, rs1`

操作：

$rd \leftarrow csr$
 $csr \leftarrow csr | rs1$

执行权限：

M mode/S mode/U mode

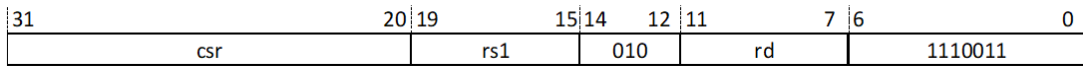
异常:

非法指令异常

说明:

各权限下允许访问的控制寄存器不同，具体请参考控制寄存器章节。
当 $rs1=x0$ 时，该指令不产生写操作，不会产生写行为引发的异常。

指令格式:



14.1.17 CSRRSI——控制寄存器立即数置位传送指令

语法:

`csrrsi rd, csr, imm5`

操作:

$rd \leftarrow csr$
 $csr \leftarrow csr \mid zero_extend(imm5)$

执行权限:

M mode/S mode/U mode

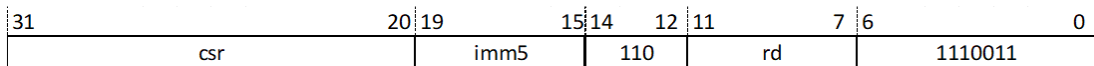
异常:

非法指令异常

说明:

各权限下允许访问的控制寄存器不同，具体请参考控制寄存器章节。

指令格式:



14.1.18 CSRRAW——控制寄存器读写传送指令

语法:

csrrw rd, csr, rs1

操作:

rd ← csr

csr ← rs1

执行权限:

M mode/S mode/U mode

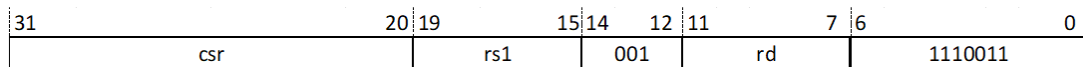
异常:

非法指令异常

说明:

各权限下允许访问的控制寄存器不同，具体请参考控制寄存器章节。

指令格式:



14.1.19 CSRRAWI——控制寄存器立即数读写传送指令

语法:

csrrwi rd, csr, imm5

操作:

rd ← csr

csr[4:0] ← imm5

csr[63:5] ← csr[63:5]

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

说明:

各权限下允许访问的控制寄存器不同，具体请参考控制寄存器章节。

指令格式:

31	20	19	15	14	12	11	7	6	0	
csr			imm5		101		rd		1110011	

14.1.20 EBREAK——断点指令

语法:

ebreak

操作:

产生断点异常或者进入调试模式

执行权限:

M mode/S mode/U mode

异常:

断点异常

指令格式:

31	20	19	15	14	12	11	7	6	0	
000000000001			00000		000		00000		1110011	

14.1.21 ECALL——环境异常指令

语法:

ecall

操作:

产生环境异常

执行权限:

M mode/S mode/U mode

异常:

用户模式环境调用异常、超级用户模式环境调用异常、机器模式环境调用异常

指令格式:

31	20	19	15	14	12	11	7	6	0	
000000000000			00000		000		00000		1110011	

14.1.22 FENCE——存储同步指令

语法:

fence iorw, iorw

操作:

保证该指令前序所有读写存储器或外设指令比该指令后序所有读写存储器或外设指令更早被观察到。

执行权限:

M mode/S mode/U mode

异常:

无

说明:

pi=1, so=1, 指令语法为 fence i,o, 以此类推

指令格式:

31	28	27	26	25	24	23	22	21	20	19	15	14	12	11	7	6	0
0000	pi	po	pr	pw	si	so	sr	sw	00000		000		00000				0001111

14.1.23 FENCE.I——指令流同步指令

语法:

fence.i

操作:

清空 icache, 保证该指令前序所有数据访存结果能够被指令后的取指操作访问到。

执行权限:

M mode/S mode/U mode

异常:

无

指令格式:

31	28	27	24	23	20	19	15	14	12	11	7	6	0
0000		0000		0000		00000		001		00000			0001111

14.1.24 JAL——直接跳转子程序指令

语法:

jal rd, label

操作:

next pc \leftarrow current pc + sign_extend(imm20 \ll 1)
rd \leftarrow current pc + 4

执行权限:

M mode/S mode/U mode

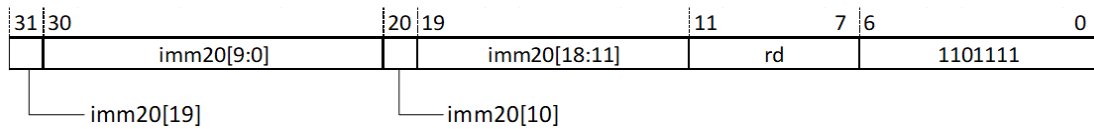
异常:

无

说明:

汇编器根据 label 算出 imm20
指令跳转范围为 $\pm 1\text{MB}$ 地址空间

指令格式:



14.1.25 JALR——寄存器跳转子程序指令

语法:

jalr rd, rs1, imm12

操作:

next pc \leftarrow (rs1 + sign_extend(imm12)) & 64'hfffffffffe
rd \leftarrow current pc + 4

执行权限:

M mode/S mode/U mode

异常:

无

说明:

机器模式或者 MMU 关闭时，指令跳转范围为全部 1TB 地址空间

非机器模式且 MMU 打开时，指令跳转范围为全部 512GB 地址空间

指令格式：

31	20	19	15	14	12	11	7	6	0
imm12[11:0]			rs1		000		rd		1100111

14.1.26 LB——有符号扩展字节加载指令

语法：

lb rd, imm12(rs1)

操作：

address ← rs1 + sign_extend(imm12)

rd ← sign_extend(mem[address])

执行权限：

M mode/S mode/U mode

异常：

加载指令非对齐访问异常、加载指令访问错误异常、加载指令页面错误异常

指令格式：

31	20	19	15	14	12	11	7	6	0
imm12[11:0]			rs1		000		rd		0000011

14.1.27 LBU——无符号扩展字节加载指令

语法：

lbu rd, imm12(rs1)

操作：

address ← rs1 + sign_extend(imm12)

rd ← zero_extend(mem[address])

执行权限：

M mode/S mode/U mode

异常：

加载指令非对齐访问异常、加载指令访问错误异常、加载指令页面错误异常

指令格式:

31	20	19	15	14	12	11	7	6	0
imm12[11:0]			rs1		100		rd		0000011

14.1.28 LD——双字加载指令

语法:

ld rd, imm12(rs1)

操作:

address ← rs1 + sign_extend(imm12)

rd ← mem[(address+7):address]

执行权限:

M mode/S mode/U mode

异常:

加载指令非对齐访问异常、加载指令访问错误异常、加载指令页面错误异常

指令格式:

31	20	19	15	14	12	11	7	6	0
imm12[11:0]			rs1		011		rd		0000011

14.1.29 LH——有符号扩展半字加载指令

语法:

lh rd, imm12(rs1)

操作:

address ← rs1 + sign_extend(imm12)

rd ← sign_extend(mem[(address+1):address])

执行权限:

M mode/S mode/U mode

异常:

加载指令非对齐访问异常、加载指令访问错误异常、加载指令页面错误异常

指令格式:

31	20	19	15	14	12	11	7	6	0
imm12[11:0]			rs1		001		rd		0000011

14.1.30 LHU——无符号扩展半字加载指令

语法:

lhu rd, imm12(rs1)

操作:

address ← rs1 + sign_extend(imm12)
rd ← zero_extend(mem[(address+1):address])

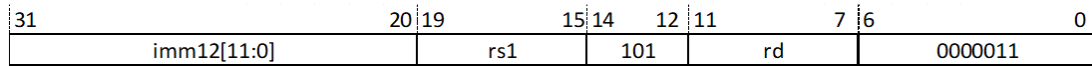
执行权限:

M mode/S mode/U mode

异常:

加载指令非对齐访问异常、加载指令访问错误异常、加载指令页面错误异常

指令格式:



14.1.31 LUI——高位立即数装载指令

语法:

lui rd, imm20

操作:

rd ← sign_extend(imm20 << 12)

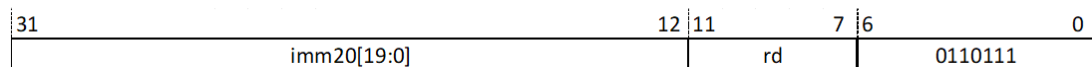
执行权限:

M mode/S mode/U mode

异常:

无

指令格式:



14.1.32 LW——有符号扩展字加载指令

语法:

lw rd, imm12(rs1)

操作:

address ← rs1 + sign_extend(imm12)
rd ← sign_extend(mem[(address+3):address])

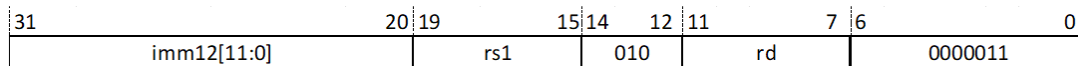
执行权限:

M mode/S mode/U mode

异常:

加载指令非对齐访问异常、加载指令访问错误异常、加载指令页面错误异常

指令格式:



14.1.33 LWU——无符号扩展字加载指令

语法:

lwu rd, imm12(rs1)

操作:

address ← rs1 + sign_extend(imm12)
rd ← zero_extend(mem[(address+3):address])

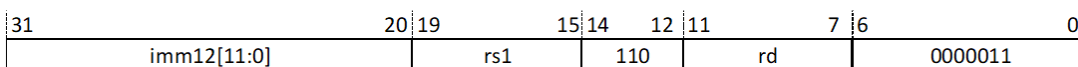
执行权限:

M mode/S mode/U mode

异常:

加载指令非对齐访问异常、加载指令访问错误异常、加载指令页面错误异常

指令格式:



14.1.34 MRET——机器模式异常返回指令

语法:

mret

操作:

next pc ← mepc

mstatus.mie ← mstatus.mpie

mstatus.mpie ← 1

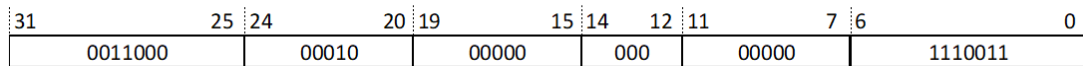
执行权限:

M mode

异常:

非法指令异常

指令格式:



14.1.35 OR——按位或指令

语法:

or rd, rs1, rs2

操作:

rd ← rs1 | rs2

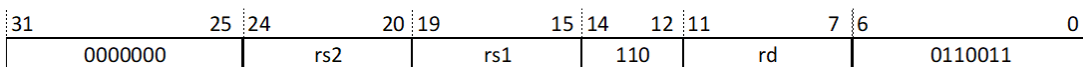
执行权限:

M mode/S mode/U mode

异常:

无

指令格式:



14.1.36 ORI——立即数按位或指令

语法:

ori rd, rs1, imm12

操作:

$rd \leftarrow rs1 \mid \text{sign_extend}(imm12)$

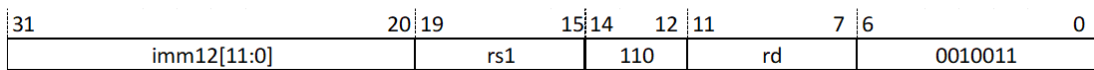
执行权限:

M mode/S mode/U mode

异常:

无

指令格式:



14.1.37 SB——字节存储指令

语法:

sb rs2, imm12(rs1)

操作:

$\text{address} \leftarrow rs1 + \text{sign_extend}(imm12)$

$\text{mem}[:\text{address}] \leftarrow rs2[7:0]$

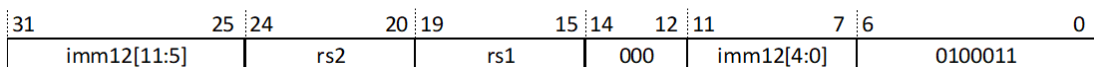
执行权限:

M mode/S mode/U mode

异常:

存储指令非对齐访问异常、存储指令访问错误异常、存储指令页面错误异常

指令格式:



14.1.38 SD——双字存储指令

语法:

sd rs2, imm12(rs1)

操作:

address ← rs1 + sign_extend(imm12)

mem[(address+7):address] ← rs2

执行权限:

M mode/S mode/U mode

异常:

存储指令非对齐访问异常、存储指令访问错误异常、存储指令页面错误异常

指令格式:

31	25	24	20	19	15	14	12	11	7	6	0
imm12[11:5]		rs2		rs1		011		imm12[4:0]		0100011	

14.1.39 SFENCE.VMA——虚拟内存同步指令

语法:

sfence.vma rs1,rs2

操作:

用于虚拟内存的无效和同步操作

执行权限:

M mode/S mode

异常:

非法指令异常

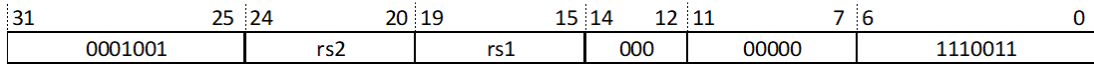
说明:

mstatus.tvm=1, 在超级用户模式下执行该指令引起非法指令异常。

rs1: 虚拟地址, rs2: asid

- rs1=x0, rs2=x0 时, 无效 TLB 中所有的表项。
- rs1!=x0, rs2=x0 时, 无效 TLB 中所有命中 rs1 虚拟地址的表项。
- rs1=x0, rs2!=x0 时, 无效 TB 中所有命中 rs2 进程号的表项。
- rs1!=x0, rs2!=x0 时, 无效 TLB 中所有命中 rs1 虚拟地址和 rs2 进程号的表项。

指令格式:



14.1.40 SH——半字存储指令

语法:

sh rs2, imm12(rs1)

操作:

address ← rs1 + sign_extend(imm12)
 mem[(address+1):address] ← rs2[15:0]

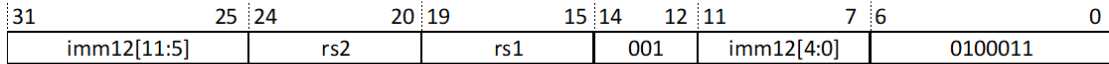
执行权限:

M mode/S mode/U mode

异常:

存储指令非对齐访问异常、存储指令访问错误异常、存储指令页面错误异常

指令格式:



14.1.41 SLL——逻辑左移指令

语法:

sll rd, rs1, rs2

操作:

rd ← rs1 << rs2[5:0]

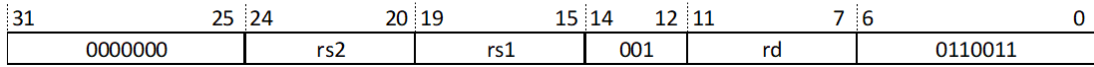
执行权限:

M mode/S mode/U mode

异常:

无

指令格式:



14.1.42 SLLI——立即数逻辑左移指令

语法:

slli rd, rs1, shamt6

操作:

$rd \leftarrow rs1 \ll shamt6$

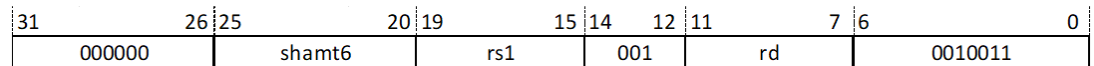
执行权限:

M mode/S mode/U mode

异常:

无

指令格式:



14.1.43 SLLIW——低 32 位立即数逻辑左移指令

语法:

slliw rd, rs1, shamt5

操作:

$tmp[31:0] \leftarrow (rs1[31:0] \ll shamt5)[31:0]$

$rd \leftarrow sign_extend(tmp[31:0])$

执行权限:

M mode/S mode/U mode

异常:

无

指令格式:

31	25	24	20	19	15	14	12	11	7	6	0	
0000000			shamt5		rs1		001		rd		0011011	

14.1.44 SLLW——低 32 位逻辑左移指令

语法:

sllw rd, rs1, rs2

操作:

tmp[31:0] ← (rs1[31:0] << rs2[4:0])[31:0]
rd ← sign_extend(tmp[31:0])

执行权限:

M mode/S mode/U mode

异常:

无

指令格式:

31	25	24	20	19	15	14	12	11	7	6	0	
0000000			rs2		rs1		001		rd		0111011	

14.1.45 SLT——有符号比较小于置位指令

语法:

slt rd, rs1, rs2

操作:

if (rs1 < rs2)
rd ← 1
else
rd ← 0

执行权限:

M mode/S mode/U mode

异常:

无

指令格式:

31	25	24	20	19	15	14	12	11	7	6	0
0000000			rs2		rs1		010		rd		0110011

14.1.46 SLTI——有符号立即数比较小于置位指令

语法:

slti rd, rs1, imm12

操作:

```
if (rs1 <sign_extend(imm12))
    rd ← 1
else
    rd ← 0
```

执行权限:

M mode/S mode/U mode

异常:

无

指令格式:

31	20	19	15	14	12	11	7	6	0
imm12[11:0]			rs1		010		rd		0010011

14.1.47 SLTIU——无符号立即数比较小于置位指令

语法:

sltiu rd, rs1, imm12

操作:

```
if (rs1 <zero_extend(imm12))
    rd ← 1
else
    rd ← 0
```

执行权限:

M mode/S mode/U mode

异常:

无

指令格式:

31	20	19	15	14	12	11	7	6	0	
imm12[11:0]			rs1		011		rd		0010011	

14.1.48 SLTU——无符号比较小于置位指令

语法:

sltu rd, rs1, rs2

操作:

```
if (rs1 < rs2)
    rd ← 1
else
    rd ← 0
```

执行权限:

M mode/S mode/U mode

异常:

无

指令格式:

31	25	24	20	19	15	14	12	11	7	6	0	
0000000			rs2		rs1		011		rd		0110011	

14.1.49 SRA——算数右移指令

语法:

sra rd, rs1, rs2

操作:

rd ← rs1 >>> rs2[5:0]

执行权限:

M mode/S mode/U mode

异常:

无

指令格式:

31	25	24	20	19	15	14	12	11	7	6	0
0100000			rs2		rs1		101		rd		0110011

14.1.50 SRAI——立即数算数右移指令

语法:

srai rd, rs1, shamt6

操作:

$rd \leftarrow rs1 \ggg shamt6$

执行权限:

M mode/S mode/U mode

异常:

无

指令格式:

31	26	25	20	19	15	14	12	11	7	6	0
010000		shamt6			rs1		101		rd		0010011

14.1.51 SRAIW——低 32 位立即数算数右移指令

语法:

sraiw rd, rs1, shamt5

操作:

$tmp[31:0] \leftarrow (rs1[31:0] \ggg shamt5)[31:0]$

$rd \leftarrow sign_extend(tmp[31:0])$

执行权限:

M mode/S mode/U mode

异常:

无

指令格式:

31	25	24	20	19	15	14	12	11	7	6	0
0100000			shamt5		rs1		101		rd		0011011

14.1.52 SRAW——低 32 位算数右移指令

语法:

sraw rd, rs1, rs2

操作:

tmp ← (rs1[31:0] >>> rs2[4:0])[31:0]
rd ← sign_extend(tmp)

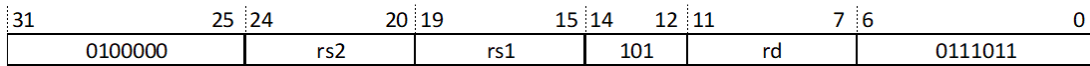
执行权限:

M mode/S mode/U mode

异常:

无

指令格式:



14.1.53 SRET——超级用户模式异常返回指令

语法:

sret

操作:

next pc ← sepc
sstatus.sie ← sstatus.spie
sstatus.spie ← 1

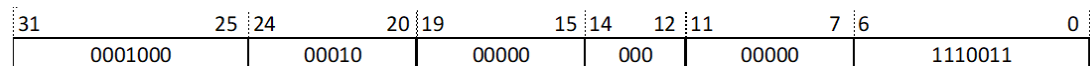
执行权限:

S mode

异常:

非法指令异常

指令格式:



14.1.54 SRL——逻辑右移指令

语法:

srl rd, rs1, rs2

操作:

$rd \leftarrow rs1 \gg rs2[5:0]$

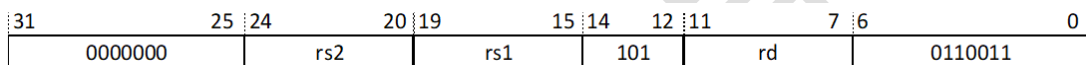
执行权限:

M mode/S mode/U mode

异常:

无

指令格式:



14.1.55 SRLI——立即数逻辑右移指令

语法:

srli rd, rs1, shamt6

操作:

$rd \leftarrow rs1 \gg shamt6$

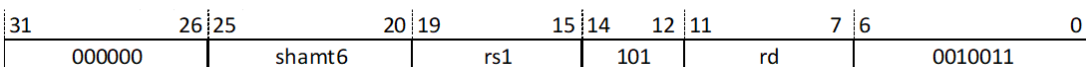
执行权限:

M mode/S mode/U mode

异常:

无

指令格式:



14.1.56 SRLIW——低 32 位立即数逻辑右移指令

语法:

srlw rd, rs1, shamt5

操作:

tmp[31:0] ← (rs1[31:0] >> shamt5)[31:0]
rd ← sign_extend(tmp[31:0])

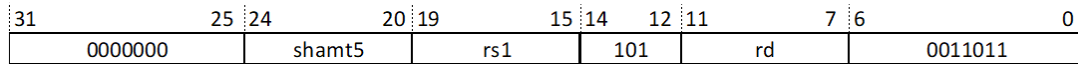
执行权限:

M mode/S mode/U mode

异常:

无

指令格式:



14.1.57 SRLW——低 32 位逻辑右移指令

语法:

srlw rd, rs1, rs2

操作:

tmp ← (rs1[31:0] >> rs2[4:0])[31:0]
rd ← sign_extend(tmp)

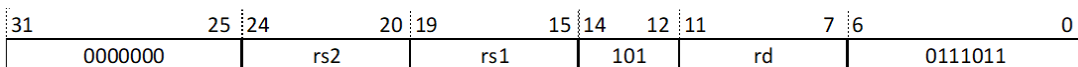
执行权限:

M mode/S mode/U mode

异常:

无

指令格式:



14.1.58 SUB——有符号减法指令

语法:

sub rd, rs1, rs2

操作:

$rd \leftarrow rs1 - rs2$

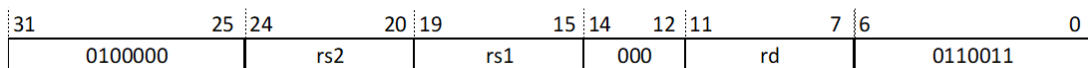
执行权限:

M mode/S mode/U mode

异常:

无

指令格式:



14.1.59 SUBW——低 32 位有符号减法指令

语法:

subw rd, rs1, rs2

操作:

$tmp[31:0] \leftarrow rs1[31:0] - rs2[31:0]$

$rd \leftarrow sign_extend(tmp[31:0])$

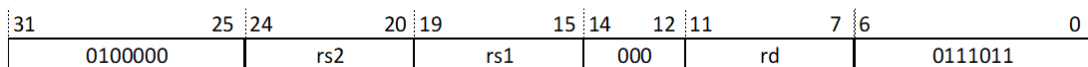
执行权限:

M mode/S mode/U mode

异常:

无

指令格式:



14.1.60 SW——字存储指令

语法:

sw rs2, imm12(rs1)

操作:

address ← rs1 + sign_extend(imm12)
 mem[(address+3):address] ← rs2[31:0]

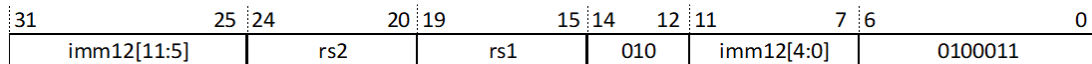
执行权限:

M mode/S mode/U mode

异常:

存储指令非对齐访问异常、存储指令访问错误异常、存储指令页面错误异常

指令格式:



14.1.61 WFI——进入低功耗模式指令

语法:

wfi

操作:

处理器进入低功耗模式，此时 CPU 时钟关闭，大部分外设时钟也关闭

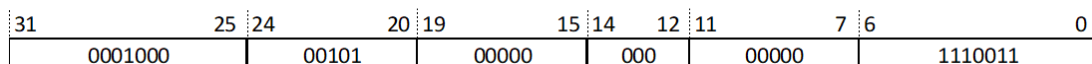
执行权限:

M mode/S mode/U mode

异常:

无

指令格式:



14.1.62 XOR——按位异或指令

语法:

xor rd, rs1, rs2

操作:

$rd \leftarrow rs1 \hat{\ } rs2$

执行权限:

M mode/S mode/U mode

异常:

无

指令格式:

31	25	24	20	19	15	14	12	11	7	6	0
0000000			rs2		rs1		100		rd		0110011

14.1.63 XORI——立即数按位异或指令

语法:

xori rd, rs1, imm12

操作:

$rd \leftarrow rs1 \& \text{sign_extend}(imm12)$

执行权限:

M mode/S mode/U mode

异常:

无

指令格式:

31	20	19	15	14	12	11	7	6	0	
imm12[11:0]				rs1		100		rd		0010011

14.2 附录 A-2 M 指令术语

以下是对 C906 实现的 RISC-V M 指令集的具体描述, 本节指令位宽为 32 位, 指令按英文字母顺序排列,

14.2.1 DIV——有符号除法指令

语法:

div rd, rs1, rs2

操作:

$rd \leftarrow rs1 / rs2$

执行权限:

M mode/S mode/U mode

异常:

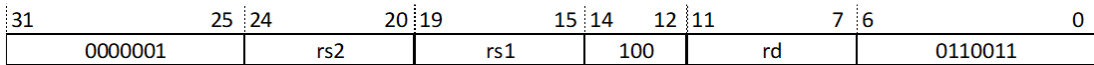
无

说明:

除数是 0 时, 除法结果为 0xffffffffffff

产生 overflow 时, 除法结果为 0x8000000000000000

指令格式:



14.2.2 DIVU——无符号除法指令

语法:

divu rd, rs1, rs2

操作:

$rd \leftarrow rs1 / rs2$

执行权限:

M mode/S mode/U mode

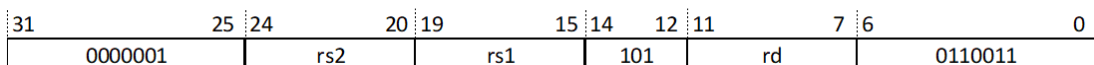
异常:

无

说明:

除数是 0 时, 除法结果为 0xffffffffffff

指令格式:



14.2.3 DIVUW——低 32 位无符号除法指令

语法:

divuw rd, rs1, rs2

操作:

tmp[31:0] ← (rs1[31:0] / rs2[31:0])[31:0]
rd ← sign_extend(tmp[31:0])

执行权限:

M mode/S mode/U mode

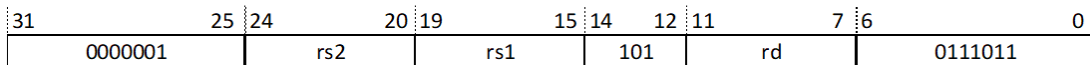
异常:

无

说明:

除数是 0 时, 除法结果为 0xffffffff

指令格式:



14.2.4 DIVW——低 32 位有符号除法指令

语法:

divw rd, rs1, rs2

操作:

tmp[31:0] ← (rs1[31:0] / rs2[31:0])[31:0]
rd ← sign_extend(tmp[31:0])

执行权限:

M mode/S mode/U mode

异常:

无

说明:

除数是 0 时, 除法结果为 0xffffffff

产生 overflow 时, 除法结果为 0xfffff80000000

指令格式:

31	25	24	20	19	15	14	12	11	7	6	0
0000001			rs2		rs1		100		rd		0111011

14.2.5 MUL——有符号乘法指令

语法:

mul rd, rs1, rs2

操作:

$rd \leftarrow (rs1 * rs2)[63:0]$

执行权限:

M mode/S mode/U mode

异常:

无

指令格式:

31	25	24	20	19	15	14	12	11	7	6	0
0000001			rs2		rs1		000		rd		0110011

14.2.6 MULH——有符号乘法取高位指令

语法:

mulh rd, rs1, rs2

操作:

$rd \leftarrow (rs1 * rs2)[127:64]$

执行权限:

M mode/S mode/U mode

异常:

无

指令格式:

31	25	24	20	19	15	14	12	11	7	6	0
0000001			rs2		rs1		001		rd		0110011

14.2.7 MULHSU——有符号无符号乘法取高位指令

语法:

mulusu rd, rs1, rs2

操作:

$rd \leftarrow (rs1 * rs2)[127:64]$

执行权限:

M mode/S mode/U mode

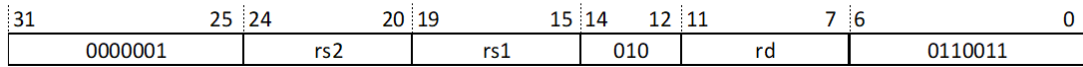
异常:

无

说明:

rs1 有符号数, rs2 无符号数

指令格式:



14.2.8 MULHU——无符号乘法取高位指令

语法:

mulhu rd, rs1, rs2

操作:

$rd \leftarrow (rs1 * rs2)[127:64]$

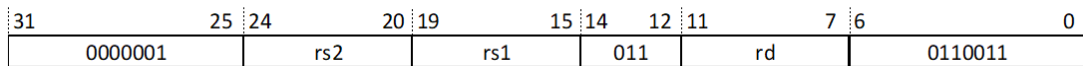
执行权限:

M mode/S mode/U mode

异常:

无

指令格式:



14.2.9 MULW——低 32 位有符号乘法指令

语法:

`mulw rd, rs1, rs2`

操作:

$tmp \leftarrow (rs1[31:0] * rs2[31:0])[31:0]$

$rd \leftarrow sign_extend(tmp[31:0])$

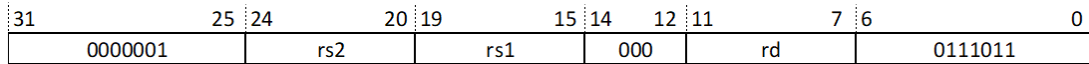
执行权限:

M mode/S mode/U mode

异常:

无

指令格式:



14.2.10 REM——有符号取余指令

语法:

`rem rd, rs1, rs2`

操作:

$rd \leftarrow rs1 \% rs2$

执行权限:

M mode/S mode/U mode

异常:

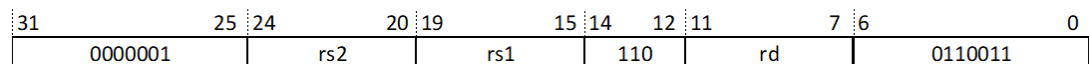
无

说明:

除数是 0 时, 求余结果为被除数

产生 overflow 时, 余数结果为 0x0

指令格式:



14.2.11 REMU——无符号取余指令

语法:

remu rd, rs1, rs2

操作:

$rd \leftarrow rs1 \% rs2$

执行权限:

M mode/S mode/U mode

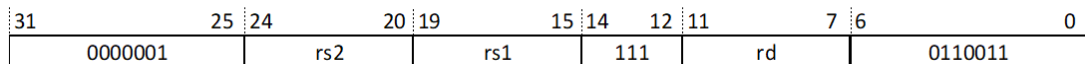
异常:

无

说明:

除数是 0 时, 求余结果为被除数

指令格式:



14.2.12 REMUW——低 32 位无符号取余指令

语法:

remw rd, rs1, rs2

操作:

$tmp \leftarrow (rs1[31:0] \% rs2[31:0])[31:0]$

$rd \leftarrow sign_extend(tmp)$

执行权限:

M mode/S mode/U mode

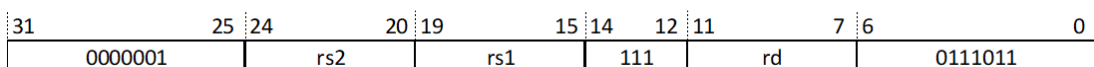
异常:

无

说明:

除数是 0 时, 求余结果是对被除数 [31] 位符号位扩展后的结果

指令格式:



14.2.13 REMW——低 32 位有符号取余指令

语法:

remw rd, rs1, rs2

操作:

tmp[31:0] ← (rs1[31:0] % rs2[31:0])[31:0]
rd ← sign_extend(tmp[31:0])

执行权限:

M mode/S mode/U mode

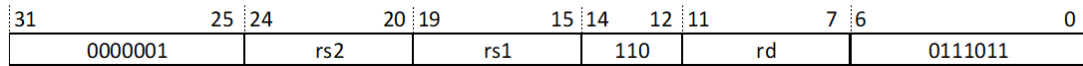
异常:

无

说明:

除数是 0 时, 求余结果是对被除数 [31] 位符号位扩展后的结果
产生 overflow 时, 余数结果为 0x0

指令格式:



14.3 附录 A-3 A 指令术语

以下是对 C906 实现的 RISC-V A 指令的具体描述, 本节指令位宽为 32 位, 指令按英文字母顺序排列。

14.3.1 AMOADD.D——原子加法指令

语法:

amoadd.d.aqrl rd, rs2, (rs1)

操作:

rd ← mem[rs1+7: rs1]
mem[rs1+7:rs1] ← mem[rs1+7:rs1] + rs2

执行权限:

M mode/S mode/U mode

异常:

原子指令非对齐访问异常、原子指令访问错误异常、原子指令页面错误异常

影响标志位:

无

说明:

aq 位和 rl 位决定该指令前序和后序访问存储器指令的执行顺序:

- aq=0,rl=0: 对应的汇编指令 amoadd.d rd, rs2, (rs1)。
- aq=0,rl=1: 对应的汇编指令 amoadd.d.rl rd, rs2, (rs1), 该指令前序所有访问存储的指令的结果必须在该指令执行之前被观察到。
- aq=1,rl=0: 对应的汇编指令 amoadd.d.aq rd, rs2, (rs1), 该指令后序所有访问存储的指令必须等该指令执行完成后才开始执行。
- aq=1,rl=1: 对应的汇编指令 amoadd.d.aqrl rd, rs2, (rs1), 该指令前序所有访问存储的指令的结果必须在该指令之前被观察到, 该指令后序所有访问存储的指令必须等该指令执行完成后才开始执行。

指令格式:

31	27	26	25	24	20	19	15	14	12	11	7	6	0	
00000			aq	rl	rs2		rs1		011		rd		0101111	

14.3.2 AMOADD.W——低 32 位原子加法指令

语法:

amoadd.w.aqrl rd, rs2, (rs1)

操作:

$rd \leftarrow \text{sign_extend}(\text{mem}[\text{rs1}+3:\text{rs1}])$
 $\text{mem}[\text{rs1}+3:\text{rs1}] \leftarrow \text{mem}[\text{rs1}+3:\text{rs1}] + \text{rs2}[31:0]$

执行权限:

M mode/S mode/U mode

异常:

原子指令非对齐访问异常、原子指令访问错误异常、原子指令页面错误异常

影响标志位:

无

说明:

aq 位和 rl 位决定该指令前序和后序访问存储器指令的执行顺序:

- aq=0,rl=0: 对应的汇编指令 amoadd.w rd, rs2, (rs1)。

- aq=0,rl=1: 对应的汇编指令 amoadd.w.rl rd, rs2, (rs1), 该指令前序所有访问存储的指令的结果必须在该指令执行之前被观察到。
- aq=1,rl=0: 对应的汇编指令 amoadd.w.aq rd, rs2, (rs1), 该指令后序所有访问存储的指令必须等该指令执行完成后才开始执行。
- aq=1,rl=1: 对应的汇编指令 amoadd.w.aqrl rd, rs2, (rs1), 该指令前序所有访问存储的指令的结果必须在该指令之前被观察到, 该指令后序所有访问存储的指令必须等该指令执行完成后才开始执行。

指令格式:

31	27	26	25	24	20	19	15	14	12	11	7	6	0
00000		aq	rl	rs2		rs1		010	rd		0101111		

14.3.3 AMOAND.D——原子按位与指令

语法:

amoand.d.aqrl rd, rs2, (rs1)

操作:

rd ← mem[rs1+7: rs1]
 mem[rs1+7:rs1] ← mem[rs1+7:rs1] & rs2

执行权限: M mode/S mode/U mode

异常:

原子指令非对齐访问异常、原子指令访问错误异常、原子指令页面错误异常

影响标志位:

无

说明:

aq 位和 rl 位决定该指令前序和后序访问存储器指令的执行顺序:

- aq=0,rl=0: 对应的汇编指令 amoand.d rd, rs2, (rs1)。
- aq=0,rl=1: 对应的汇编指令 amoand.d.rl rd, rs2, (rs1), 该指令前序所有访问存储的指令的结果必须在该指令执行之前被观察到。
- aq=1,rl=0: 对应的汇编指令 amoand.d.aq rd, rs2, (rs1), 该指令后序所有访问存储的指令必须等该指令执行完成后才开始执行。
- aq=1,rl=1: 对应的汇编指令 amoand.d.aqrl rd, rs2, (rs1), 该指令前序所有访问存储的指令的结果必须在该指令之前被观察到, 该指令后序所有访问存储的指令必须等该指令执行完成后才开始执行。

指令格式:

31	27	26	25	24	20	19	15	14	12	11	7	6	0
01100		aq	rl	rs2		rs1		011	rd		0101111		

14.3.4 AMOAND.W——低 32 位原子按位与指令

语法:

amoand.w.aqrl rd, rs2, (rs1)

操作:

$rd \leftarrow \text{sign_extend}(\text{mem}[\text{rs1}+3:\text{rs1}])$
 $\text{mem}[\text{rs1}+3:\text{rs1}] \leftarrow \text{mem}[\text{rs1}+3:\text{rs1}] \& \text{rs2}[31:0]$

执行权限:

M mode/S mode/U mode

异常:

原子指令非对齐访问异常、原子指令访问错误异常、原子指令页面错误异常

影响标志位:

无

说明:

aq 位和 rl 位决定该指令前序和后序访问存储器指令的执行顺序:

- aq=0,rl=0: 对应的汇编指令 amoand.w rd, rs2, (rs1)。
- aq=0,rl=1: 对应的汇编指令 amoand.w.rl rd, rs2, (rs1), 该指令前序所有访问存储的指令的结果必须在该指令执行之前被观察到。
- aq=1,rl=0: 对应的汇编指令 amoand.w.aq rd, rs2, (rs1), 该指令后序所有访问存储的指令必须等该指令执行完成后才开始执行。
- aq=1,rl=1: 对应的汇编指令 amoand.w.aqrl rd, rs2, (rs1), 该指令前序所有访问存储的指令的结果必须在该指令之前被观察到, 该指令后序所有访问存储的指令必须等该指令执行完成后才开始执行。

指令格式:

31	27	26	25	24	20	19	15	14	12	11	7	6	0		
01100				aq	rl	rs2			rs1			010	rd	0101111	

14.3.5 AMOMAX.D——原子有符号取最大值指令

语法:

amomax.d.aqrl rd, rs2, (rs1)

操作:

$rd \leftarrow \text{mem}[\text{rs1}+7:\text{rs1}]$
 $\text{mem}[\text{rs1}+7:\text{rs1}] \leftarrow \text{max}(\text{mem}[\text{rs1}+7:\text{rs1}], \text{rs2})$

执行权限:

M mode/S mode/U mode

异常:

原子指令非对齐访问异常、原子指令访问错误异常、原子指令页面错误异常

影响标志位:

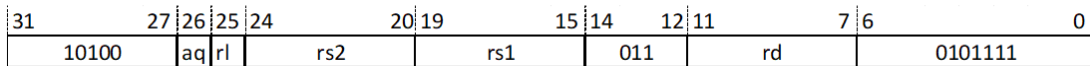
无

说明:

aq 位和 rl 位决定该指令前序和后序访问存储器指令的执行顺序:

- aq=0,rl=0: 对应的汇编指令 amomax.d rd, rs2, (rs1)。
- aq=0,rl=1: 对应的汇编指令 amomax.d.rl rd, rs2, (rs1), 该指令前序所有访问存储的指令的结果必须在该指令执行之前被观察到。
- aq=1,rl=0: 对应的汇编指令 amomax.d.aq rd, rs2, (rs1), 该指令后序所有访问存储的指令必须等该指令执行完成后才开始执行。
- aq=1,rl=1: 对应的汇编指令 amomax.d.aqrl rd, rs2, (rs1), 该指令前序所有访问存储的指令的结果必须在该指令之前被观察到, 该指令后序所有访问存储的指令必须等该指令执行完成后才开始执行。

指令格式:



14.3.6 AMOMAX.W——低 32 位原子有符号取最大值指令

语法:

amomax.w.aqrl rd, rs2, (rs1)

操作:

$rd \leftarrow \text{sign_extend}(\text{mem}[\text{rs1}+3:\text{rs1}])$
 $\text{mem}[\text{rs1}+3:\text{rs1}] \leftarrow \max(\text{mem}[\text{rs1}+3:\text{rs1}], \text{rs2}[31:0])$

执行权限:

M mode/S mode/U mode

异常:

原子指令非对齐访问异常、原子指令访问错误异常、原子指令页面错误异常

影响标志位:

无

说明:

aq 位和 rl 位决定该指令前序和后序访问存储器指令的执行顺序:

- aq=0,rl=0: 对应的汇编指令 amomax.w rd, rs2, (rs1)。
- aq=0,rl=1: 对应的汇编指令 amomax.w.rl rd, rs2, (rs1), 该指令前序所有访问存储的指令的结果必须在该指令执行之前被观察到。
- aq=1,rl=0: 对应的汇编指令 amomax.w.aq rd, rs2, (rs1), 该指令后序所有访问存储的指令必须等该指令执行完成后才开始执行。
- aq=1,rl=1: 对应的汇编指令 amomax.w.aqrl rd, rs2, (rs1), 该指令前序所有访问存储的指令的结果必须在该指令之前被观察到, 该指令后序所有访问存储的指令必须等该指令执行完成后才开始执行。

指令格式:

31	27	26	25	24	20	19	15	14	12	11	7	6	0
10100			aq	rl	rs2		rs1		010	rd		0101111	

14.3.7 AMOMAXU.D——原子无符号取最大值指令

语法:

amomaxu.d.aqrl rd, rs2, (rs1)

操作:

rd ← mem[rs1+7: rs1]
 mem[rs1+7:rs1] ← maxu(mem[rs1+7:rs1], rs2)

执行权限:

M mode/S mode/U mode

异常:

原子指令非对齐访问异常、原子指令访问错误异常、原子指令页面错误异常

影响标志位:

无

说明:

aq 位和 rl 位决定该指令前序和后序访问存储器指令的执行顺序:

- aq=0,rl=0: 对应的汇编指令 amomaxu.d rd, rs2, (rs1)。
- aq=0,rl=1: 对应的汇编指令 amomaxu.d.rl rd, rs2, (rs1), 该指令前序所有访问存储的指令的结果必须在该指令执行之前被观察到。
- aq=1,rl=0: 对应的汇编指令 amomaxu.d.aq rd, rs2, (rs1), 该指令后序所有访问存储的指令必须等该指令执行完成后才开始执行。

- aq=1,rl=1: 对应的汇编指令 amomaxu.d.aqrl rd, rs2, (rs1), 该指令前序所有访问存储的指令的结果必须在该指令之前被观察到, 该指令后序所有访问存储的指令必须等该指令执行完成后才开始执行。

指令格式:

31	27	26	25	24	20	19	15	14	12	11	7	6	0
11100				aq	rl	rs2			rs1		011	rd	0101111

14.3.8 AMOMAXU.W——低 32 位原子无符号取最大值指令

语法:

amomaxu.w.aqrl rd, rs2, (rs1)

操作:

rd ← sign_extend(mem[rs1+3: rs1])
 mem[rs1+3:rs1] ← maxu(mem[rs1+3:rs1], rs2[31:0])

执行权限:

M mode/S mode/U mode

异常:

原子指令非对齐访问异常、原子指令访问错误异常、原子指令页面错误异常

影响标志位:

无

说明:

aq 位和 rl 位决定该指令前序和后序访问存储器指令的执行顺序:

- aq=0,rl=0: 对应的汇编指令 amomaxu.w rd, rs2, (rs1)。
- aq=0,rl=1: 对应的汇编指令 amomaxu.w.rl rd, rs2, (rs1), 该指令前序所有访问存储的指令的结果必须在该指令执行之前被观察到。
- aq=1,rl=0: 对应的汇编指令 amomaxu.w.aq rd, rs2, (rs1), 该指令后序所有访问存储的指令必须等该指令执行完成后才开始执行。
- aq=1,rl=1: 对应的汇编指令 amomaxu.w.aqrl rd, rs2, (rs1), 指令前序所有访问存储的指令的结果必须在该指令执行之前被观察到, 该指令后序所有访问存储的指令必须等该指令执行完成后才开始执行。

指令格式:

31	27	26	25	24	20	19	15	14	12	11	7	6	0
11100				aq	rl	rs2			rs1		010	rd	0101111

14.3.9 AMOMIN.D——原子有符号取最小值指令

语法:

amomin.d.aqrl rd, rs2, (rs1)

操作:

$rd \leftarrow \text{mem}[\text{rs1}+7: \text{rs1}]$
 $\text{mem}[\text{rs1}+7:\text{rs1}] \leftarrow \text{minu}(\text{mem}[\text{rs1}+7:\text{rs1}], \text{rs2})$

执行权限:

M mode/S mode/U mode

异常:

原子指令非对齐访问异常、原子指令访问错误异常、原子指令页面错误异常

影响标志位:

无

说明:

aq 位和 rl 位决定该指令前序和后序访问存储器指令的执行顺序:

- aq=0,rl=0: 对应的汇编指令 amomin.d rd, rs2, (rs1)。
- aq=0,rl=1: 对应的汇编指令 amomin.d.rl rd, rs2, (rs1), 该指令前序所有访问存储的指令的结果必须在该指令执行之前被观察到。
- aq=1,rl=0: 对应的汇编指令 amomin.d.aq rd, rs2, (rs1), 该指令后序所有访问存储的指令必须等该指令执行完成后才开始执行。
- aq=1,rl=1: 对应的汇编指令 amomin.d.aqrl rd, rs2, (rs1), 指令前序所有访问存储的指令的结果必须在该指令执行之前被观察到, 该指令后序所有访问存储的指令必须等该指令执行完成后才开始执行。

指令格式:

31	27	26	25	24	20	19	15	14	12	11	7	6	0	
10000			aq	rl	rs2		rs1		011		rd		0101111	

14.3.10 AMOMIN.W——低 32 位原子有符号取最小值指令

语法:

amomin.w.aqrl rd, rs2, (rs1)

操作:

$rd \leftarrow \text{sign_extend}(\text{mem}[\text{rs1}+3: \text{rs1}])$
 $\text{mem}[\text{rs1}+3:\text{rs1}] \leftarrow \text{minu}(\text{mem}[\text{rs1}+3:\text{rs1}], \text{rs2}[\text{31}:0])$

执行权限:

M mode/S mode/U mode

异常:

原子指令非对齐访问异常、原子指令访问错误异常、原子指令页面错误异常

影响标志位:

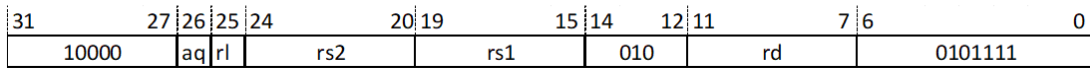
无

说明:

aq 位和 rl 位决定该指令前序和后序访问存储器指令的执行顺序:

- aq=0,rl=0: 对应的汇编指令 amomin.w rd, rs2, (rs1)。
- aq=0,rl=1: 对应的汇编指令 amomin.w.rl rd, rs2, (rs1), 该指令前序所有访问存储的指令的结果必须在该指令执行之前被观察到。
- aq=1,rl=0: 对应的汇编指令 amomin.w.aq rd, rs2, (rs1), 该指令后序所有访问存储的指令必须等该指令执行完成后才开始执行。
- aq=1,rl=1: 对应的汇编指令 amomin.w.aqrl rd, rs2, (rs1), 指令前序所有访问存储的指令的结果必须在该指令执行之前被观察到, 该指令后序所有访问存储的指令必须等该指令执行完成后才开始执行。

指令格式:



14.3.11 AMOMINU.D——原子无符号取最小值指令

语法:

amominu.d.aqrl rd, rs2, (rs1)

操作:

rd ← mem[rs1+7: rs1]
 mem[rs1+7:rs1] ← minu(mem[rs1+7:rs1], rs2)

执行权限:

M mode/S mode/U mode

异常:

原子指令非对齐访问异常、原子指令访问错误异常、原子指令页面错误异常

影响标志位:

无

说明:

aq 位和 rl 位决定该指令前序和后序访问存储器指令的执行顺序:

- aq=0,rl=0: 对应的汇编指令 amominu.d rd, rs2, (rs1)。
- aq=0,rl=1: 对应的汇编指令 amominu.d.rl rd, rs2, (rs1), 该指令前序所有访问存储的指令的结果必须在该指令执行之前被观察到。
- aq=1,rl=0: 对应的汇编指令 amominu.d.aq rd, rs2, (rs1), 该指令后序所有访问存储的指令必须等该指令执行完成后才开始执行。
- aq=1,rl=1: 对应的汇编指令 amominu.d.aqrl rd, rs2, (rs1), 指令前序所有访问存储的指令的结果必须在该指令执行之前被观察到, 该指令后序所有访问存储的指令必须等该指令执行完成后才开始执行。

指令格式:

31	27	26	25	24	20	19	15	14	12	11	7	6	0	
11000			aq	rl	rs2		rs1		011		rd		0101111	

14.3.12 AMOMINU.W——低 32 位原子无符号取最小值指令

语法:

amominu.w.aqrl rd, rs2, (rs1)

操作:

rd ← sign_extend(mem[rs1+3: rs1])
 mem[rs1+3:rs1] ← minu(mem[rs1+3:rs1], rs2[31:0])

执行权限:

M mode/S mode/U mode

异常:

原子指令非对齐访问异常、原子指令访问错误异常、原子指令页面错误异常

影响标志位:

无

说明:

aq 位和 rl 位决定该指令前序和后序访问存储器指令的执行顺序:

- aq=0,rl=0: 对应的汇编指令 amominu.w rd, rs2, (rs1)。
- aq=0,rl=1: 对应的汇编指令 amominu.w.rl rd, rs2, (rs1), 该指令前序所有访问存储的指令的结果必须在该指令执行之前被观察到。
- aq=1,rl=0: 对应的汇编指令 amominu.w.aq rd, rs2, (rs1), 该指令后序所有访问存储的指令必须等该指令执行完成后才开始执行。
- aq=1,rl=1: 对应的汇编指令 amominu.w.aqrl rd, rs2, (rs1), 指令前序所有访问存储的指令的结果必须在该指令执行之前被观察到, 该指令后序所有访问存储的指令必须等该指令执行完成后才开始执行。

指令格式:

31	27	26	25	24	20	19	15	14	12	11	7	6	0
11000		aq	rl	rs2		rs1		010		rd		0101111	

14.3.13 AMOOR.D——原子按位或指令

语法:

amoor.d.aqrl rd, rs2, (rs1)

操作:

rd \leftarrow mem[rs1+7: rs1]
 mem[rs1+7:rs1] \leftarrow mem[rs1+7:rs1] | rs2

执行权限:

M mode/S mode/U mode

异常:

原子指令非对齐访问异常、原子指令访问错误异常、原子指令页面错误异常

影响标志位:

无

说明:

aq 位和 rl 位决定该指令前序和后序访问存储器指令的执行顺序:

- aq=0,rl=0: 对应的汇编指令 amoor.d rd, rs2, (rs1)。
- aq=0,rl=1: 对应的汇编指令 amoor.d.rl rd, rs2, (rs1), 该指令前序所有访问存储的指令的结果必须在该指令执行之前被观察到。
- aq=1,rl=0: 对应的汇编指令 amoor.d.aq rd, rs2, (rs1), 该指令后序所有访问存储的指令必须等该指令执行完成后才开始执行。
- aq=1,rl=1: 对应的汇编指令 amoor.d.aqrl rd, rs2, (rs1), 指令前序所有访问存储的指令的结果必须在该指令执行之前被观察到, 该指令后序所有访问存储的指令必须等该指令执行完成后才开始执行。

指令格式:

31	27	26	25	24	20	19	15	14	12	11	7	6	0
01000		aq	rl	rs2		rs1		011		rd		0101111	

14.3.14 AMOOR.W——低 32 位原子按位或指令

语法:

amoor.w.aqrl rd, rs2, (rs1)

操作:

$rd \leftarrow \text{sign_extend}(\text{mem}[\text{rs1}+3: \text{rs1}])$
 $\text{mem}[\text{rs1}+3:\text{rs1}] \leftarrow \text{mem}[\text{rs1}+3:\text{rs1}] \mid \text{rs2}[31:0]$

执行权限:

M mode/S mode/U mode

异常:

原子指令非对齐访问异常、原子指令访问错误异常、原子指令页面错误异常

影响标志位:

无

说明:

aq 位和 rl 位决定该指令前序和后序访问存储器指令的执行顺序:

- aq=0,rl=0: 对应的汇编指令 amoor.w rd, rs2, (rs1)。
- aq=0,rl=1: 对应的汇编指令 amoor.w.rl rd, rs2, (rs1), 该指令前序所有访问存储的指令的结果必须在该指令执行之前被观察到。
- aq=1,rl=0: 对应的汇编指令 amoor.w.aq rd, rs2, (rs1), 该指令后序所有访问存储的指令必须等该指令执行完成后才开始执行。
- aq=1,rl=1: 对应的汇编指令 amoor.w.aqrl rd, rs2, (rs1), 指令前序所有访问存储的指令的结果必须在该指令执行之前被观察到, 该指令后序所有访问存储的指令必须等该指令执行完成后才开始执行。

指令格式:

31	27	26	25	24	20	19	15	14	12	11	7	6	0
01000		aq	rl	rs2		rs1		010		rd		0101111	

14.3.15 AMOSWAP.D——原子交换指令

语法:

amoswap.d.aqrl rd, rs2, (rs1)

操作:

$rd \leftarrow \text{mem}[\text{rs1}+7: \text{rs1}]$
 $\text{mem}[\text{rs1}+7:\text{rs1}] \leftarrow \text{rs2}$

执行权限:

M mode/S mode/U mode

异常:

原子指令非对齐访问异常、原子指令访问错误异常、原子指令页面错误异常

影响标志位: 无

说明:

aq 位和 rl 位决定该指令前序和后序访问存储器指令的执行顺序:

- aq=0,rl=0: 对应的汇编指令 amoswap.d rd, rs2, (rs1)。
- aq=0,rl=1: 对应的汇编指令 amoswap.d.rl rd, rs2, (rs1), 该指令前序所有访问存储的指令的结果必须在该指令执行之前被观察到。
- aq=1,rl=0: 对应的汇编指令 amoswap.d.aq rd, rs2, (rs1), 该指令后序所有访问存储的指令必须等该指令执行完成后才开始执行。
- aq=1,rl=1: 对应的汇编指令 amoswap.d.aqrl rd, rs2, (rs1), 指令前序所有访问存储的指令的结果必须在该指令执行之前被观察到, 该指令后序所有访问存储的指令必须等该指令执行完成后才开始执行。

指令格式:

31	27	26	25	24	20	19	15	14	12	11	7	6	0	
00001			aq	rl	rs2			rs1		011		rd	0101111	

14.3.16 AMOSWAP.W——低 32 位原子交换指令

语法:

amoswap.w.aqrl rd, rs2, (rs1)

操作:

$rd \leftarrow \text{sign_extend}(\text{mem}[\text{rs1}+3:\text{rs1}])$
 $\text{mem}[\text{rs1}+3:\text{rs1}] \leftarrow \text{rs2}[31:0]$

执行权限:

M mode/S mode/U mode

异常:

原子指令非对齐访问异常、原子指令访问错误异常、原子指令页面错误异常

影响标志位: 无

说明:

aq 位和 rl 位决定该指令前序和后序访问存储器指令的执行顺序:

- aq=0,rl=0: 对应的汇编指令 amoswap.w rd, rs2, (rs1)。
- aq=0,rl=1: 对应的汇编指令 amoswap.w.rl rd, rs2, (rs1), 该指令前序所有访问存储的指令的结果必须在该指令执行之前被观察到。
- aq=1,rl=0: 对应的汇编指令 amoswap.w.aq rd, rs2, (rs1), 该指令后序所有访问存储的指令必须等该指令执行完成后才开始执行。
- aq=1,rl=1: 对应的汇编指令 amoswap.w.aqrl rd, rs2, (rs1), 指令前序所有访问存储的指令的结果必须在该指令执行之前被观察到, 该指令后序所有访问存储的指令必须等该指令执行完成后才开始执行。

指令格式:

31	27	26	25	24	20	19	15	14	12	11	7	6	0
00001		aq	rl	rs2	rs1	010		rd	0101111				

14.3.17 AMOXOR.D——原子按位异或指令

语法:

amosxor.d.aqrl rd, rs2, (rs1)

操作:

rd ← mem[rs1+7: rs1]
 mem[rs1+7:rs1] ← mem[rs1+7:rs1] ^ rs2

执行权限:

M mode/S mode/U mode

异常:

原子指令非对齐访问异常、原子指令访问错误异常、原子指令页面错误异常

影响标志位:

无

说明:

aq 位和 rl 位决定该指令前序和后序访问存储器指令的执行顺序:

- aq=0,rl=0: 对应的汇编指令 amoxor.d rd, rs2, (rs1)。
- aq=0,rl=1: 对应的汇编指令 amoxor.d.rl rd, rs2, (rs1), 该指令前序所有访问存储的指令的结果必须在该指令执行之前被观察到。
- aq=1,rl=0: 对应的汇编指令 amoxor.d.aq rd, rs2, (rs1), 该指令后序所有访问存储的指令必须等该指令执行完成后才开始执行。
- aq=1,rl=1: 对应的汇编指令 amoxor.d.aqrl rd, rs2, (rs1), 指令前序所有访问存储的指令的结果必须在该指令执行之前被观察到, 该指令后序所有访问存储的指令必须等该指令执行完成后才开始执行。

指令格式:

31	27	26	25	24	20	19	15	14	12	11	7	6	0
00100		aq	rl	rs2	rs1	011		rd	0101111				

14.3.18 AMOXOR.W——低 32 位原子按位异或指令

语法:

```
amoxor.w.aqrl rd, rs2, (rs1)
```

操作:

```
rd ← sign_extend(mem[rs1+3: rs1])
mem[rs1+3:rs1] ← mem[rs1+3:rs1] ^ rs2[31:0]
```

执行权限:

M mode/S mode/U mode

异常:

原子指令非对齐访问异常、原子指令访问错误异常、原子指令页面错误异常

影响标志位:

无

说明:

aq 位和 rl 位决定该指令前序和后序访问存储器指令的执行顺序:

- aq=0,rl=0: 对应的汇编指令 amoxor.w rd, rs2, (rs1)。
- aq=0,rl=1: 对应的汇编指令 amoxor.w.rl rd, rs2, (rs1), 该指令前序所有访问存储的指令的结果必须在该指令执行之前被观察到。
- aq=1,rl=0: 对应的汇编指令 amoxor.w.aq rd, rs2, (rs1), 该指令后序所有访问存储的指令必须等该指令执行完成后才开始执行。
- aq=1,rl=1: 对应的汇编指令 amoxor.w.aqrl rd, rs2, (rs1), 指令前序所有访问存储的指令的结果必须在该指令执行之前被观察到, 该指令后序所有访问存储的指令必须等该指令执行完成后才开始执行。

指令格式:

31	27	26	25	24	20	19	15	14	12	11	7	6	0
00100		aq	rl	rs2		rs1		010	rd		0101111		

14.3.19 LR.D——双字加载保留指令

语法:

```
lr.d.aqrl rd, (rs1)
```

操作:

```
rd ← mem[rs1+7: rs1]
mem[rs1+7:rs1] is reserved
```

执行权限:

M mode/S mode/U mode

异常:

原子指令非对齐访问异常、原子指令访问错误异常、原子指令页面错误异常

影响标志位:

无

说明:

aq 位和 rl 位决定该指令前序和后序访问存储器指令的执行顺序:

- aq=0,rl=0: 对应的汇编指令 lr.d rd, (rs1)。
- aq=0,rl=1: 对应的汇编指令 lr.d.rl rd, (rs1), 该指令前序所有访问存储的指令的结果必须在该指令执行之前被观察到。
- aq=1,rl=0: 对应的汇编指令 lr.d.aq rd, (rs1), 该指令后序所有访问存储的指令必须等该指令执行完成后才开始执行。
- aq=1,rl=1: 对应的汇编指令 lr.d.aqrl rd, (rs1), 指令前序所有访问存储的指令的结果必须在该指令执行之前被观察到, 该指令后序所有访问存储的指令必须等该指令执行完成后才开始执行。

指令格式:

31	27	26	25	24	20	19	15	14	12	11	7	6	0	
00010		aq	rl	00000			rs1		011		rd		0101111	

14.3.20 LR.W——字加载保留指令

语法:

lr.w.aqrl rd, (rs1)

操作:

$rd \leftarrow \text{sign_extend}(\text{mem}[\text{rs1}+3:\text{rs1}])$
 mem[rs1+3:rs1] is reserved

执行权限:

M mode/S mode/U mode

异常: 原子指令非对齐访问异常、原子指令访问错误异常、原子指令页面错误异常

影响标志位: 无

说明: aq 位和 rl 位决定该指令前序和后序访问存储器指令的执行顺序:

- aq=0,rl=0: 对应的汇编指令 lr.w rd, (rs1)。
- aq=0,rl=1: 对应的汇编指令 lr.w.rl rd, (rs1), 该指令前序所有访问存储的指令的结果必须在该指令执行之前被观察到。

- aq=1,rl=0: 对应的汇编指令 lr.w.aq rd, (rs1), 该指令后序所有访问存储的指令必须等该指令执行完成后才开始执行。
- aq=1,rl=1: 对应的汇编指令 lr.w.aqrl rd, (rs1), 指令前序所有访问存储的指令的结果必须在该指令执行之前被观察到, 该指令后序所有访问存储的指令必须等该指令执行完成后才开始执行。

指令格式:

31	27	26	25	24	20	19	15	14	12	11	7	6	0
00010				aq	rl	00000			rs1	010	rd	0101111	

14.3.21 SC.D——双字条件存储指令

语法:

sc.d.aqrl rd, rs2, (rs1)

操作:

```
If(mem[rs1+7:rs1] is reserved)
    mem[rs1+7:rs1] ← rs2
    rd ← 0
else
    rd ← 1
```

执行权限:

M mode/S mode/U mode

异常:

原子指令非对齐访问异常、原子指令访问错误异常、原子指令页面错误异常

影响标志位:

无

说明:

aq 位和 rl 位决定该指令前序和后序访问存储器指令的执行顺序:

- aq=0,rl=0: 对应的汇编指令 sc.d rd, rs2, (rs1)。
- aq=0,rl=1: 对应的汇编指令 sc.d.rl rd, rs2, (rs1), 该指令前序所有访问存储的指令的结果必须在该指令执行之前被观察到。
- aq=1,rl=0: 对应的汇编指令 sc.d.aq rd, rs2, (rs1), 该指令后序所有访问存储的指令必须等该指令执行完成后才开始执行。
- aq=1,rl=1: 对应的汇编指令 sc.d.aqrl rd, rs2, (rs1), 指令前序所有访问存储的指令的结果必须在该指令执行之前被观察到, 该指令后序所有访问存储的指令必须等该指令执行完成后才开始执行。

指令格式:

31	27	26	25	24	20	19	15	14	12	11	7	6	0
00011				aq	rl	rs2	rs1	011	rd	0101111			

14.3.22 SC.W——字条件存储指令

语法:

sc.w.aqrl rd, rs2, (rs1)

操作:

```
if(mem[rs1+3:rs1] is reserved)
    mem[rs1+3:rs1] ← rs2[31:0]
    rd ← 0
else
    rd ← 1
```

执行权限:

M mode/S mode/U mode

异常:

原子指令非对齐访问异常、原子指令访问错误异常、原子指令页面错误异常

影响标志位:

无

说明:

aq 位和 rl 位决定该指令前序和后序访问存储器指令的执行顺序:

- aq=0,rl=0: 对应的汇编指令 sc.w rd, rs2, (rs1)。
- aq=0,rl=1: 对应的汇编指令 sc.w.rl rd, rs2, (rs1), 该指令前序所有访问存储的指令的结果必须在该指令执行之前被观察到。
- aq=1,rl=0: 对应的汇编指令 sc.w.aq rd, rs2, (rs1), 该指令后序所有访问存储的指令必须等该指令执行完成后才开始执行。
- aq=1,rl=1: 对应的汇编指令 sc.w.aqrl rd, rs2, (rs1), 指令前序所有访问存储的指令的结果必须在该指令执行之前被观察到, 该指令后序所有访问存储的指令必须等该指令执行完成后才开始执行。

指令格式:

31	27	26	25	24	20	19	15	14	12	11	7	6	0	
00011			aq	rl	rs2		rs1		010		rd		0101111	

14.4 附录 A-4 F 指令术语

以下是对 C906 实现的 RISC-V F 指令集的具体描述, 本节指令位宽为 32 位, 指令按英文字母顺序排列,

对于单精度浮点指令, 如果源寄存器的高 32 位不全为 1, 则该单精度数据当作 qNaN 处理。

当 mstatus.fs==2' b00 时, 执行本节所有指令会产生非法指令异常, 当 mstatus.fs != 2' b00 时, 执行本节任意指令后 mstatus.fs 置位为 2' b11。

14.4.1 FADD.S——单精度浮点加法指令

语法:

fadd.s fd, fs1, fs2, rm

操作:

frd \leftarrow fs1 + fs2

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

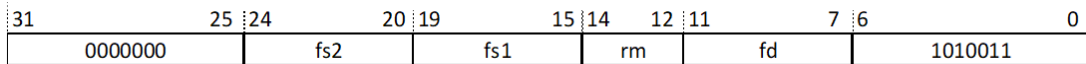
浮点状态位 NV/OF/NX

说明:

rm 决定舍入模式:

- 3' b000: 就近向偶数舍入, 对应的汇编指令 fadd.s fd, fs1,fs2,rne。
- 3' b001: 向零舍入, 对应的汇编指令 fadd.s fd, fs1,fs2,rtz。
- 3' b010: 向负无穷舍入, 对应的汇编指令 fadd.s fd, fs1,fs2,rdn。
- 3' b011: 向正无穷舍入, 对应的汇编指令 fadd.s fd, fs1,fs2,rup。
- 3' b100: 就近向大值舍入, 对应的汇编指令 fadd.s fd, fs1,fs2,rmm。
- 3' b101: 暂未使用, 不会出现该编码。
- 3' b110: 暂未使用, 不会出现该编码。
- 3' b111: 动态舍入, 根据浮点控制寄存器 fcsr 中的 rm 位来决定舍入模式, 对应的汇编指令 fadd.s fd, fs1,fs2。

指令格式:



14.4.2 FCLASS.S——单精度浮点分类指令

语法:

fclass.s rd, fs1

操作:

if (fs1 = -inf)

```

rd ← 64' h1
if ( fs1 = -norm)
    rd ← 64' h2
if ( fs1 = -subnorm)
    rd ← 64' h4
if ( fs1 = -zero)
    rd ← 64' h8
if ( fs1 = +zero)
    rd ← 64' h10
if ( fs1 = +subnorm)
    rd ← 64' h20
if ( fs1 = +norm)
    rd ← 64' h40
if ( fs1 = +Inf)
    rd ← 64' h80
if ( fs1 = sNaN)
    rd ← 64' h100
if ( fs1 = qNaN)
    rd ← 64' h200

```

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

无

指令格式:

31	25	24	20	19	15	14	12	11	7	6	0
1110000			00000		fs1		001		rd		1010011

14.4.3 FCVT.L.S——单精度浮点转换成有符号长整型指令

语法:

fcvt.l.s rd, fs1, rm

操作:

rd ← single_convert_to_signed_long(fs1)

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

浮点状态位 NV/NX

说明:

rm 决定舍入模式:

- 3' b000: 就近向偶数舍入, 对应的汇编指令 fcvt.l.s rd,fs1,rne。
- 3' b001: 向零舍入, 对应的汇编指令 fcvt.l.s rd,fs1,rtz。
- 3' b010: 向负无穷舍入, 对应的汇编指令 fcvt.l.s rd,fs1,rdn。
- 3' b011: 向正无穷舍入, 对应的汇编指令 fcvt.l.s rd,fs1,rup。
- 3' b100: 就近向大值舍入, 对应的汇编指令 fcvt.l.s rd,fs1,rmm。
- 3' b101: 暂未使用, 不会出现该编码。
- 3' b110: 暂未使用, 不会出现该编码。
- 3' b111: 动态舍入, 根据浮点控制寄存器 fcsr 中的 rm 位来决定舍入模式, 对应的汇编指令 fcvt.l.s rd, fs1。

指令格式:

31	25	24	20	19	15	14	12	11	7	6	0
1100000		00010		fs1		rm		rd		1010011	

14.4.4 FCVT.LU.S——单精度浮点转换成无符号长整型指令

语法:

fcvt.lu.s rd, fs1, rm

操作:

rd ← single_convert_to_unsigned_long(fs1)

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

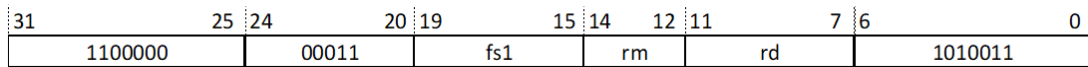
浮点状态位 NV/NX

说明:

rm 决定舍入模式:

- 3' b000: 就近向偶数舍入, 对应的汇编指令 fcvt.lu.s rd,fs1,rne。
- 3' b001: 向零舍入, 对应的汇编指令 fcvt.lu.s rd,fs1,rtz。
- 3' b010: 向负无穷舍入, 对应的汇编指令 fcvt.lu.s rd,fs1,rdn。
- 3' b011: 向正无穷舍入, 对应的汇编指令 fcvt.lu.s rd,fs1,rup。
- 3' b100: 就近向大值舍入, 对应的汇编指令 fcvt.lu.s rd,fs1,rmm。
- 3' b101: 暂未使用, 不会出现该编码。
- 3' b110: 暂未使用, 不会出现该编码。
- 3' b111: 动态舍入, 根据浮点控制寄存器 fcsr 中的 rm 位来决定舍入模式, 对应的汇编指令 fcvt.lu.s rd, fs1。

指令格式:



14.4.5 FCVT.S.L——有符号长整型转换成单精度浮点数指令

语法:

fcvt.s.l fd, rs1, rm

操作:

fd ← signed_long_convert_to_single(fs1)

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

浮点状态位 NX

说明:

rm 决定舍入模式:

- 3' b000: 就近向偶数舍入, 对应的汇编指令 fcvt.s.l fd,rs1,rne。
- 3' b001: 向零舍入, 对应的汇编指令 fcvt.s.l fd,rs1,rtz。
- 3' b010: 向负无穷舍入, 对应的汇编指令 fcvt.s.l fd,fs1,rdn。

- 3' b011: 向正无穷舍入, 对应的汇编指令 fcvt.s.l fd,fs1,rup。
- 3' b100: 就近向大值舍入, 对应的汇编指令 fcvt.s.l fd,fs1,rmm。
- 3' b101: 暂未使用, 不会出现该编码。
- 3' b110: 暂未使用, 不会出现该编码。
- 3' b111: 动态舍入, 根据浮点控制寄存器 fcsr 中的 rm 位来决定舍入模式, 对应的汇编指令 fcvt.s.l fd, fs1。

指令格式:

31	25	24	20	19	15	14	12	11	7	6	0
1101000			00010		rs1		rm		fd		1010011

14.4.6 FCVT.S.LU——无符号长整型转换成单精度浮点数指令

语法:

fcvt.s.lu fd, fs1, rm

操作:

fd ← unsigned_long_convert_to_single(fs1)

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

浮点状态位 NX

说明:

rm 决定舍入模式:

- 3' b000: 就近向偶数舍入, 对应的汇编指令 fcvt.s.lu fd,fs1,rne。
- 3' b001: 向零舍入, 对应的汇编指令 fcvt.s.lu fd, fs1,rtz。
- 3' b010: 向负无穷舍入, 对应的汇编指令 fcvt.s.lu fd, fs1,rdn。
- 3' b011: 向正无穷舍入, 对应的汇编指令 fcvt.s.lu fd, fs1,rup。
- 3' b100: 就近向大值舍入, 对应的汇编指令 fcvt.s.lu fd, fs1,rmm。
- 3' b101: 暂未使用, 不会出现该编码。
- 3' b110: 暂未使用, 不会出现该编码。
- 3' b111: 动态舍入, 根据浮点控制寄存器 fcsr 中的 rm 位来决定舍入模式, 对应的汇编指令 fcvt.s.lu fd, fs1。

指令格式:

31	25	24	20	19	15	14	12	11	7	6	0
1101000			00011		rs1		rm		fd		1010011

14.4.7 FCVT.S.W——有符号整型转换成单精度浮点数指令

语法:

`fcvt.s.w fd, rs1, rm`

操作:

$fd \leftarrow \text{signed_int_convert_to_single}(fs1)$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

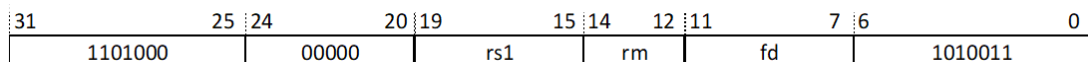
浮点状态位 NX

说明:

rm 决定舍入模式:

- 3' b000: 就近向偶数舍入, 对应的汇编指令 `fcvt.s.w fd,rs1,rne`。
- 3' b001: 向零舍入, 对应的汇编指令 `fcvt.s.w fd,rs1,rtz`。
- 3' b010: 向负无穷舍入, 对应的汇编指令 `fcvt.s.w fd,rs1,rdn`。
- 3' b011: 向正无穷舍入, 对应的汇编指令 `fcvt.s.w fd,rs1,rup`。
- 3' b100: 就近向大值舍入, 对应的汇编指令 `fcvt.s.w fd,rs1,rmm`。
- 3' b101: 暂未使用, 不会出现该编码。
- 3' b110: 暂未使用, 不会出现该编码。
- 3' b111: 动态舍入, 根据浮点控制寄存器 `fcsr` 中的 `rm` 位来决定舍入模式, 对应的汇编指令 `fcvt.s.w fd, rs1`。

指令格式:



14.4.8 FCVT.S.WU——无符号整型转换成单精度浮点数指令

语法:

`fcvt.s.wu fd, rs1, rm`

操作:

$fd \leftarrow \text{unsigned_int_convert_to_single}(fs1)$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

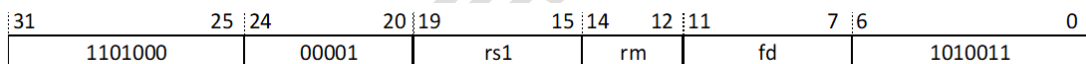
浮点状态位 NX

说明:

rm 决定舍入模式:

- 3' b000: 就近向偶数舍入, 对应的汇编指令 fcvt.s.wu fd,rs1,rne。
- 3' b001: 向零舍入, 对应的汇编指令 fcvt.s.wu fd,rs1,rtz。
- 3' b010: 向负无穷舍入, 对应的汇编指令 fcvt.s.wu fd,rs1,rdn。
- 3' b011: 向正无穷舍入, 对应的汇编指令 fcvt.s.wu fd,rs1,rup。
- 3' b100: 就近向大值舍入, 对应的汇编指令 fcvt.s.wu fd,rs1,rmm。
- 3' b101: 暂未使用, 不会出现该编码。
- 3' b110: 暂未使用, 不会出现该编码。
- 3' b111: 动态舍入, 根据浮点控制寄存器 fcsr 中的 rm 位来决定舍入模式, 对应的汇编指令 fcvt.s.wu fd, rs1。

指令格式:



14.4.9 FCVT.W.S——单精度浮点转换成有符号整型指令

语法:

fcvt.w.s rd, fs1, rm

操作:

```
tmp[31:0] ← single_convert_to_signed_int(fs1)
rd ← sign_extend(tmp[31:0])
```

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

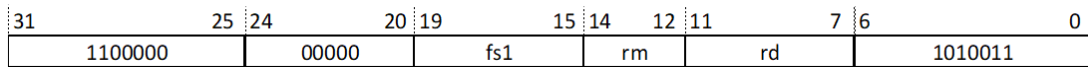
浮点状态位 NV/NX

说明:

rm 决定舍入模式:

- 3' b000: 就近向偶数舍入, 对应的汇编指令 fcvt.w.s rd,fs1,rne。
- 3' b001: 向零舍入, 对应的汇编指令 fcvt.w.s rd,fs1,rtz。
- 3' b010: 向负无穷舍入, 对应的汇编指令 fcvt.w.s rd,fs1,rdn。
- 3' b011: 向正无穷舍入, 对应的汇编指令 fcvt.w.s rd,fs1,rup。
- 3' b100: 就近向大值舍入, 对应的汇编指令 fcvt.w.s rd,fs1,rmm。
- 3' b101: 暂未使用, 不会出现该编码。
- 3' b110: 暂未使用, 不会出现该编码。
- 3' b111: 动态舍入, 根据浮点控制寄存器 fcsr 中的 rm 位来决定舍入模式, 对应的汇编指令 fcvt.w.s rd, fs1。

指令格式:



14.4.10 FCVT.WU.S——单精度浮点转换成无符号整型指令

语法:

fcvt.wu.s rd, fs1, rm

操作:

```
tmp ← single_convert_to_unsigned_int(fs1)
rd ← sign_extend(tmp)
```

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

浮点状态位 NV/NX

说明:

rm 决定舍入模式:

- 3' b000: 就近向偶数舍入, 对应的汇编指令 fcvt.wu.s rd,fs1,rne。
- 3' b001: 向零舍入, 对应的汇编指令 fcvt.wu.s rd,fs1,rtz。

- 3' b010: 向负无穷舍入, 对应的汇编指令 fcvt.wu.s rd,fs1,rdn。
- 3' b011: 向正无穷舍入, 对应的汇编指令 fcvt.wu.s rd,fs1,rup。
- 3' b100: 就近向大值舍入, 对应的汇编指令 fcvt.wu.s rd,fs1,rmm。
- 3' b101: 暂未使用, 不会出现该编码。
- 3' b110: 暂未使用, 不会出现该编码。
- 3' b111: 动态舍入, 根据浮点控制寄存器 fcsr 中的 rm 位来决定舍入模式, 对应的汇编指令 fcvt.wu.s rd, fs1。

指令格式:

31	25	24	20	19	15	14	12	11	7	6	0
1100000			00001		fs1		rm		rd		1010011

14.4.11 FDIV.S——单精度浮点除法指令

语法:

fdiv.s fd, fs1, fs2, rm

操作:

fd ← fs1 / fs2

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

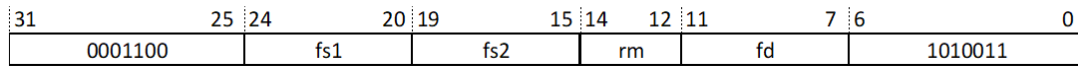
浮点状态位 NV/DZ/OF/UF/NX

说明:

rm 决定舍入模式:

- 3' b000: 就近向偶数舍入, 对应的汇编指令 fdiv.s fs1,fs2,rne。
- 3' b001: 向零舍入, 对应的汇编指令 fdiv.s fd fs1,fs2,rtz。
- 3' b010: 向负无穷舍入, 对应的汇编指令 fdiv.s fd, fs1,fs2,rdn。
- 3' b011: 向正无穷舍入, 对应的汇编指令 fdiv.s fd, fs1,fs2,rup。
- 3' b100: 就近向大值舍入, 对应的汇编指令 fdiv.s fd, fs1,fs2,rmm。
- 3' b101: 暂未使用, 不会出现该编码。
- 3' b110: 暂未使用, 不会出现该编码。
- 3' b111: 动态舍入, 根据浮点控制寄存器 fcsr 中的 rm 位来决定舍入模式, 对应的汇编指令 fdiv.s fd, fs1,fs2。

指令格式:



14.4.12 FEQ.S——单精度浮点比较相等指令

语法:

feq.s rd, fs1, fs2

操作:

if(fs1 == fs2)

rd ← 1

else

rd ← 0

执行权限:

M mode/S mode/U mode

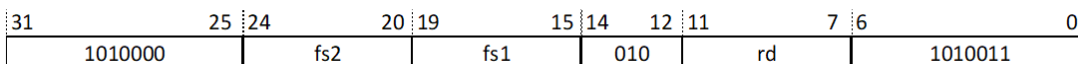
异常:

非法指令异常

影响标志位:

浮点状态位 NV

指令格式:



14.4.13 FLE.S——单精度浮点比较小于等于指令

语法:

fle.s rd, fs1, fs2

操作:

if(fs1 <= fs2)

rd ← 1

else

rd ← 0

执行权限:

M mode/S mode/U mode

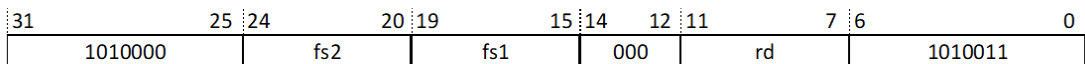
异常:

非法指令异常

影响标志位:

浮点状态位 NV

指令格式:



14.4.14 FLT.S——单精度浮点比较小于指令

语法:

flt.s rd, fs1, fs2

操作:

```
if(fs1 < fs2)
    rd ← 1
else
    rd ← 0
```

执行权限:

M mode/S mode/U mode

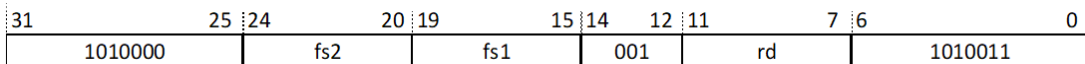
异常:

非法指令异常

影响标志位:

浮点状态位 NV

指令格式:



14.4.15 FLW——单精度浮点加载指令

语法:

`flw fd, imm12(rs1)`

操作:

$address \leftarrow rs1 + sign_extend(imm12)$
 $fd[31:0] \leftarrow mem[(address+3):address]$
 $fd[63:32] \leftarrow 32' hfffffff$

执行权限:

M mode/S mode/U mode

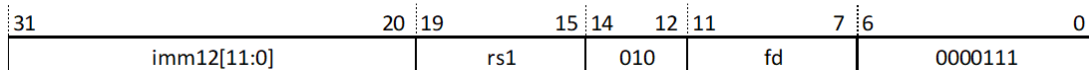
异常:

加载指令非对齐访问异常、加载指令访问错误异常、加载指令页面错误异常、非法指令异常

影响标志位:

无

指令格式:



14.4.16 FMADD.S——单精度浮点乘累加指令

语法:

`fmadd.s fd, fs1, fs2, fs3, rm`

操作:

$fd \leftarrow fs1 * fs2 + fs3$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

浮点状态位 NV/OF/UF/IX

说明:

rm 决定舍入模式:

- 3' b000: 就近向偶数舍入, 对应的汇编指令 `fmadd.s fd,fs1, fs2, fs3, rne`。
- 3' b001: 向零舍入, 对应的汇编指令 `fmadd.s fd,fs1, fs2, fs3, rtz`。
- 3' b010: 向负无穷舍入, 对应的汇编指令 `fmadd.s fd,fs1, fs2, fs3, rdn`。
- 3' b011: 向正无穷舍入, 对应的汇编指令 `fmadd.s fd,fs1, fs2, fs3, rup`。
- 3' b100: 就近向大值舍入, 对应的汇编指令 `fmadd.s fd,fs1, fs2, fs3, rmm`。
- 3' b101: 暂未使用, 不会出现该编码。
- 3' b110: 暂未使用, 不会出现该编码。
- 3' b111: 动态舍入, 根据浮点控制寄存器 `fcsr` 中的 `rm` 位来决定舍入模式, 对应的汇编指令 `fmadd.s fd,fs1, fs2, fs3`。

指令格式:

31	27	26	25	24	20	19	15	14	12	11	7	6	0
fs3			00		fs2		fs1		rm		fd		1000011

14.4.17 FMAX.S——单精度浮点取最大值指令

语法:

`fmax.s fd, fs1, fs2`

操作:

```
if(fs1 >= fs2)
    fd ← fs1
else
    fd ← fs2
```

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

浮点状态位 NV

指令格式:

31	25	24	20	19	15	14	12	11	7	6	0
0010100			fs2		fs1		001		fd		1010011

14.4.18 FMIN.S——单精度浮点取最小值指令

语法:

fmin.s fd, fs1, fs2

操作:

```
if(fs1 < fs2)
    fd ← fs1
else
    fd ← fs2
```

执行权限:

M mode/S mode/U mode

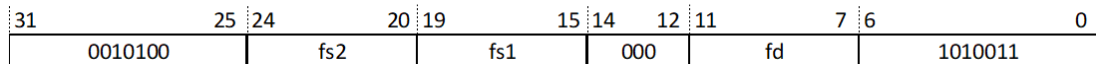
异常:

非法指令异常

影响标志位:

浮点状态位 NV

指令格式:



14.4.19 FMSUB.S——单精度浮点乘累减指令

语法:

fmsub.s fd, fs1, fs2, fs3, rm

操作:

fd ← fs1*fs2 - fs3

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

浮点状态位 NV/OF/UF/IX

说明:

rm 决定舍入模式:

- 3' b000: 就近向偶数舍入, 对应的汇编指令 fmsub.s fd,fs1, fs2, fs3, rne。
- 3' b001: 向零舍入, 对应的汇编指令 fmsub.s fd,fs1, fs2, fs3, rtz。
- 3' b010: 向负无穷舍入, 对应的汇编指令 fmsub.s fd,fs1, fs2, fs3, rdn。
- 3' b011: 向正无穷舍入, 对应的汇编指令 fmsub.s fd,fs1, fs2, fs3, rup。
- 3' b100: 就近向大值舍入, 对应的汇编指令 fmsub.s fd, fs1, fs2, fs3,rmm。
- 3' b101: 暂未使用, 不会出现该编码。
- 3' b110: 暂未使用, 不会出现该编码。
- 3' b111: 动态舍入, 根据浮点控制寄存器 fcsr 中的 rm 位来决定舍入模式, 对应的汇编指令 fmsub.s fd,fs1, fs2, fs3。

指令格式:

31	27	26	25	24	20	19	15	14	12	11	7	6	0		
fs3			00		fs2			fs1		rm		fd		1000111	

14.4.20 FMUL.S——单精度浮点乘法指令

语法:

fmul.s fd, fs1, fs2, rm

操作:

fd ← fs1 * fs2

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

浮点状态位 NV/OF/UF/NX

说明:

rm 决定舍入模式:

- 3' b000: 就近向偶数舍入, 对应的汇编指令 fmul.s fd, fs1, fs2, rne。
- 3' b001: 向零舍入, 对应的汇编指令 fmul.s fd, fs1, fs2, rtz。
- 3' b010: 向负无穷舍入, 对应的汇编指令 fmul.s fd, fs1, fs2, rdn。
- 3' b011: 向正无穷舍入, 对应的汇编指令 fmul.s fd, fs1, fs2, rup。
- 3' b100: 就近向大值舍入, 对应的汇编指令 fmul.s fd, fs1,fs2, rmm。

- 3' b101: 暂未使用, 不会出现该编码。
- 3' b110: 暂未使用, 不会出现该编码。
- 3' b111: 动态舍入, 根据浮点控制寄存器 fcsr 中的 rm 位来决定舍入模式, 对应的汇编指令 fmul.s fs1,fs2。

指令格式:

31	25	24	20	19	15	14	12	11	7	6	0
0001000			fs2		fs1		rm		fd		1010011

14.4.21 FMV.W.X——单精度浮点写传送指令

语法:

fmv.w.x fd, rs1

操作:

fd[31:0] ← rs[31:0]

fd[63:32] ← 32' hfffffff

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

无

指令格式:

31	25	24	20	19	15	14	12	11	7	6	0
1111000			00000		rs1		000		fd		1010011

14.4.22 FMV.X.W——单精度浮点寄存器读传送指令

语法:

fmv.x.w rd, fs1

操作:

tmp[31:0] ← fs1[31:0]

rd ← sign_extend(tmp[31:0])

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

无

指令格式:

31	25	24	20	19	15	14	12	11	7	6	0
1110000			00000		fs1		000		rd		1010011

14.4.23 FNMADD.S——单精度浮点乘累加取负指令

语法:

fnmadd.s fd, fs1, fs2, fs3, rm

操作:

$fd \leftarrow -(fs1 * fs2 + fs3)$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

浮点状态位 NV/OF/UF/IX

说明:

rm 决定舍入模式:

- 3' b000: 就近向偶数舍入, 对应的汇编指令 `fnmadd.s fd,fs1, fs2, fs3, rne`。
- 3' b001: 向零舍入, 对应的汇编指令 `fnmadd.s fd,fs1, fs2, fs3, rtz`。
- 3' b010: 向负无穷舍入, 对应的汇编指令 `fnmadd.s fd,fs1, fs2, fs3, rdn`。
- 3' b011: 向正无穷舍入, 对应的汇编指令 `fnmadd.s fd,fs1, fs2, fs3, rup`。
- 3' b100: 就近向大值舍入, 对应的汇编指令 `fnmadd.s fd,fs1, fs2, fs3, rmm`。
- 3' b101: 暂未使用, 不会出现该编码。
- 3' b110: 暂未使用, 不会出现该编码。
- 3' b111: 动态舍入, 根据浮点控制寄存器 `fcsr` 中的 `rm` 位来决定舍入模式, 对应的汇编指令 `fnmadd.s fd,fs1, fs2, fs3`。

指令格式:

31	27	26	25	24	20	19	15	14	12	11	7	6	0
fs3			00		fs2		fs1		rm		fd		1001111

14.4.24 FNMSUB.S——单精度浮点乘累减取负指令

语法:

fnmsub.s fd, fs1, fs2, fs3, rm

操作:

$fd \leftarrow -(fs1 * fs2 - fs3)$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

浮点状态位 NV/OF/UF/IX

说明:

rm 决定舍入模式:

- 3' b000: 就近向偶数舍入, 对应的汇编指令 fnmsub.s fd,fs1, fs2, fs3, rne。
- 3' b001: 向零舍入, 对应的汇编指令 fnmsub.s fd,fs1, fs2, fs3, rtz。
- 3' b010: 向负无穷舍入, 对应的汇编指令 fnmsub.s fd,fs1, fs2, fs3, rdn。
- 3' b011: 向正无穷舍入, 对应的汇编指令 fnmsub.s fd,fs1, fs2, fs3, rup。
- 3' b100: 就近向大值舍入, 对应的汇编指令 fnmsub.s fd,fs1, fs2, fs3, rmm。
- 3' b101: 暂未使用, 不会出现该编码。
- 3' b110: 暂未使用, 不会出现该编码。
- 3' b111: 动态舍入, 根据浮点控制寄存器 fcsr 中的 rm 位来决定舍入模式, 对应的汇编指令 fnmsub.s fd,fs1, fs2, fs3。

指令格式:

31	27	26	25	24	20	19	15	14	12	11	7	6	0
fs3			00		fs2		fs1		rm		fd		1001011

14.4.25 FSGNJ.S——单精度浮点符号注入指令

语法:

fsgnj.s fd, fs1, fs2

操作:

fd[30:0] ← fs1[30:0]

fd[31] ← fs2[31]

fd[63:32] ← 32' hfffffff

执行权限:

M mode/S mode/U mode

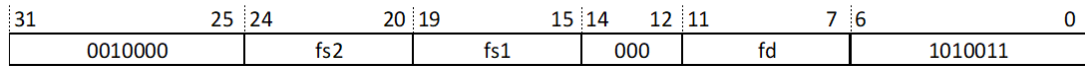
异常:

非法指令异常

影响标志位:

无

指令格式:



14.4.26 FSGNJN.S——单精度浮点符号取反注入指令

语法:

fsgnjn.s fd, fs1, fs2

操作:

fd[30:0] ← fs1[30:0]

fd[31] ← ! fs2[31]

fd[63:32] ← 32' hfffffff

执行权限:

M mode/S mode/U mode

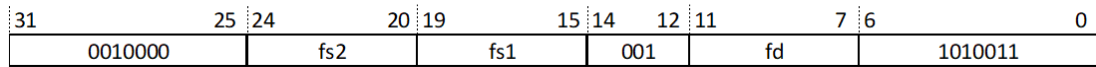
异常:

非法指令异常

影响标志位:

无

指令格式:



14.4.27 FSGNJX.S——单精度浮点符号异或注入指令

语法:

fsgnjx.s fd, fs1, fs2

操作:

fd[30:0] ← fs1[30:0]
 fd[31] ← fs1[31] ^ fs2[31]
 fd[63:32] ← 32' hfffffff

执行权限:

M mode/S mode/U mode

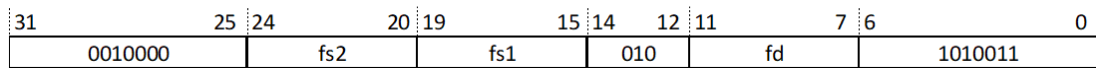
异常:

非法指令异常

影响标志位:

无

指令格式:



14.4.28 FSQRT.S——单精度浮点开方指令

语法:

fsqrt.s fd, fs1, rm

操作:

fd ← sqrt(fs1)

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

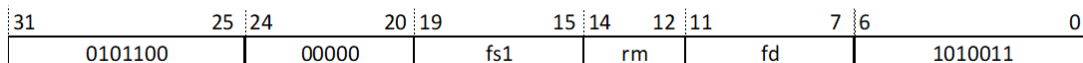
浮点状态位 NV/NX

说明:

rm 决定舍入模式:

- 3' b000: 就近向偶数舍入, 对应的汇编指令 fsqrt.s fd, fs1,rne
- 3' b001: 向零舍入, 对应的汇编指令 fsqrt.s fd, fs1,rtz
- 3' b010: 向负无穷舍入, 对应的汇编指令 fsqrt.s fd, fs1,rdn
- 3' b011: 向正无穷舍入, 对应的汇编指令 fsqrt.s fd, fs1,rup
- 3' b100: 就近向大值舍入, 对应的汇编指令 fsqrt.s fd, fs1,rmm
- 3' b101: 暂未使用, 不会出现该编码。
- 3' b110: 暂未使用, 不会出现该编码。
- 3' b111: 动态舍入, 根据浮点控制寄存器 fcsr 中的 rm 位来决定舍入模式, 对应的汇编指令 fsqrt.s fd, fs1。

指令格式:



14.4.29 FSUB.S——单精度浮点减法指令

语法:

fsub.s fd, fs1, fs2, rm

操作:

fd ← fs1 - fs2

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

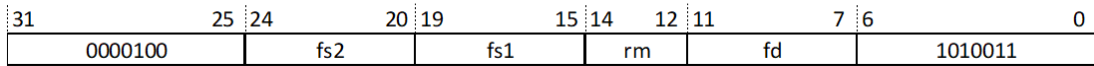
浮点状态位 NV/OF/NX

说明:

rm 决定舍入模式:

- 3' b000: 就近向偶数舍入, 对应的汇编指令 fsub.f_d, fs1,fs2,rne
- 3' b001: 向零舍入, 对应的汇编指令 fsub.s fd, fs1,fs2,rtz
- 3' b010: 向负无穷舍入, 对应的汇编指令 fsub.s fd, fs1,fs2,rdn
- 3' b011: 向正无穷舍入, 对应的汇编指令 fsub.s fd, fs1,fs2,rup
- 3' b100: 就近向大值舍入, 对应的汇编指令 fsub.s fd, fs1,fs2,rmm
- 3' b101: 暂未使用, 不会出现该编码。
- 3' b110: 暂未使用, 不会出现该编码。
- 3' b111: 动态舍入, 根据浮点控制寄存器 fcsr 中的 rm 位来决定舍入模式, 对应的汇编指令 fsub.s fd, fs1,fs2。

指令格式:



14.4.30 FSW——单精度浮点存储指令

语法:

fsw fs2, imm12(rs1)

操作:

address ← rs1 + sign_extend(imm12)
 mem[(address+31):address] ← fs2[31:0]

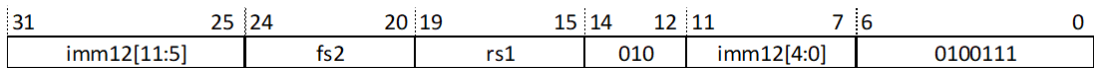
执行权限:

M mode/S mode/U mode

异常:

存储指令非对齐访问异常、存储指令访问错误异常、存储指令页面错误异常、非法指令异常

指令格式:



14.5 附录 A-5 D 指令术语

以下是对 C906 实现的 RISC-V D 指令集的具体描述, 本节指令位宽为 32 位, 指令按英文字母顺序排列。

当 mstatus.fs = 2' b00 时, 执行本节所有指令会产生非法指令异常, 当 mstatus.fs != 2' b00 时, 执行本节任意指令后 mstatus.fs 置位为 2' b11。

14.5.1 FADD.D——双精度浮点加法指令

语法:

fadd.d fd, fs1, fs2, rm

操作:

$fd \leftarrow fs1 + fs2$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

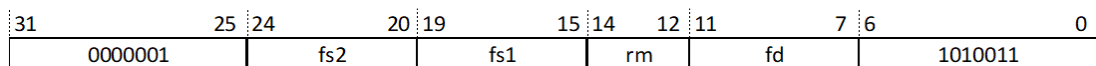
浮点状态位 NV/OF/NX

说明:

rm 决定舍入模式:

- 3' b000: 就近向偶数舍入, 对应的汇编指令 fadd.d fd, fs1,fs2,rne
- 3' b001: 向零舍入, 对应的汇编指令 fadd.d fd, fs1,fs2,rtz
- 3' b010: 向负无穷舍入, 对应的汇编指令 fadd.d fd, fs1,fs2,rdn
- 3' b011: 向正无穷舍入, 对应的汇编指令 fadd.d fd, fs1,fs2,rup
- 3' b100: 就近向大值舍入, 对应的汇编指令 fadd.d fd, fs1,fs2,rmm
- 3' b101: 暂未使用, 不会出现该编码。
- 3' b110: 暂未使用, 不会出现该编码。
- 3' b111: 动态舍入, 根据浮点控制寄存器 fcsr 中的 rm 位来决定舍入模式, 对应的汇编指令 fadd.d fd, fs1,fs2。

指令格式:



14.5.2 FCLASS.D——双精度浮点分类指令

语法:

fclass.d rd, fs1

操作: if (fs1 = -Inf)

$rd \leftarrow 64' h1$

if (fs1 = -norm)

```

rd ← 64' h2
if ( fs1 = -subnorm)
    rd ← 64' h4
if ( fs1 = -zero)
    fd ← 64' h8
if ( fs1 = +Zero)
    rd ← 64' h10
if ( fs1 = +subnorm)
    rd ← 64' h20
if ( fs1 = +norm)
    rd ← 64' h40
if ( fs1 = +Inf)
    rd ← 64' h80
if ( fs1 = sNaN)
    rd ← 64' h100
if ( fs1 = qNaN)
    rd ← 64' h200
    
```

执行权限:

M mode/S mode/U mode

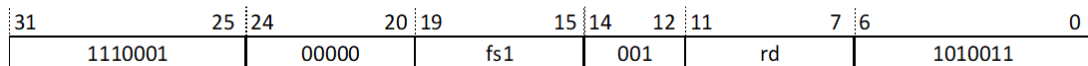
异常:

非法指令异常

影响标志位:

无

指令格式:



14.5.3 FCVT.D.L——有符号长整型转换成双精度浮点数指令

语法:

fcvt.d.l fd, rs1, rm

操作:

$fd \leftarrow \text{signed_long_convert_to_double}(fs1)$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

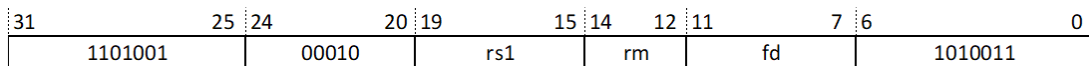
浮点状态位 NX

说明:

rm 决定舍入模式:

- 3' b000: 就近向偶数舍入, 对应的汇编指令 fcvt.d.l fd,rs1,rne。
- 3' b001: 向零舍入, 对应的汇编指令 fcvt.d.l fd,rs1,rtz。
- 3' b010: 向负无穷舍入, 对应的汇编指令 fcvt.d.l fd,rs1,rdn。
- 3' b011: 向正无穷舍入, 对应的汇编指令 fcvt.d.l fd,rs1,rup。
- 3' b100: 就近向大值舍入, 对应的汇编指令 fcvt.d.l fd,rs1,rmm。
- 3' b101: 暂未使用, 不会出现该编码。
- 3' b110: 暂未使用, 不会出现该编码。
- 3' b111: 动态舍入, 根据浮点控制寄存器 fcsr 中的 rm 位来决定舍入模式, 对应的汇编指令 fcvt.d.l fd, rs1。

指令格式:



14.5.4 FCVT.D.LU——无符号长整型转换成双精度浮点数指令

语法:

fcvt.d.lu fd, rs1, rm

操作:

fd ← unsigned_long_convert_to_double(fs1)

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

浮点状态位 NX

说明:

rm 决定舍入模式:

- 3' b000: 就近向偶数舍入, 对应的汇编指令 fcvt.d.lu fd,rs1,rne。
- 3' b001: 向零舍入, 对应的汇编指令 fcvt.d.lu fd,rs1,rtz。
- 3' b010: 向负无穷舍入, 对应的汇编指令 fcvt.d.lu fd,rs1,rdn。
- 3' b011: 向正无穷舍入, 对应的汇编指令 fcvt.d.lu fd,rs1,rup。
- 3' b100: 就近向大值舍入, 对应的汇编指令 fcvt.d.lu fd,rs1,rmm。
- 3' b101: 暂未使用, 不会出现该编码。
- 3' b110: 暂未使用, 不会出现该编码。
- 3' b111: 动态舍入, 根据浮点控制寄存器 fcsr 中的 rm 位来决定舍入模式, 对应的汇编指令 fcvt.d.lu fd, rs1。

指令格式:

31	25	24	20	19	15	14	12	11	7	6	0
1101001			00011		rs1		rm		fd		1010011

14.5.5 FCVT.D.S——单精度浮点转换成双精度浮点指令

语法:

fcvt.d.s fd, fs1

操作:

fd ← single_convert_to_double(fs1)

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

浮点状态位 NV

指令格式:

31	25	24	20	19	15	14	12	11	7	6	0
0100001			00000		fs1		000		fd		1010011

14.5.6 FCVT.D.W——有符号整型转换成双精度浮点数指令

语法:

fcvt.d.w fd, rs1

操作:

$fd \leftarrow \text{signed_int_convert_to_double}(fs1)$

执行权限:

M mode/S mode/U mode

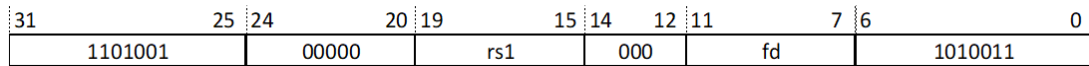
异常:

非法指令异常

影响标志位:

无

指令格式:



14.5.7 FCVT.D.WU——无符号整型转换成双精度浮点数指令

语法:

fcvt.d.wu fd, rs1

操作:

$fd \leftarrow \text{unsigned_int_convert_to_double}(fs1)$

执行权限:

M mode/S mode/U mode

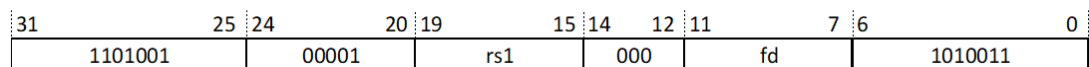
异常:

非法指令异常

影响标志位:

无

指令格式:



14.5.8 FCVT.L.D——双精度浮点转换成有符号长整型指令

语法:

`fcvt.l.d rd, fs1, rm`

操作:

`rd ← double_convert_to_signed_long(fs1)`

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

浮点状态位 NV/NX

说明:

rm 决定舍入模式:

- 3' b000: 就近向偶数舍入, 对应的汇编指令 `fcvt.l.d rd,fs1,rne`。
- 3' b001: 向零舍入, 对应的汇编指令 `fcvt.l.d rd,fs1,rtz`。
- 3' b010: 向负无穷舍入, 对应的汇编指令 `fcvt.l.d rd,fs1,rdn`。
- 3' b011: 向正无穷舍入, 对应的汇编指令 `fcvt.l.d rd,fs1,rup`。
- 3' b100: 就近向大值舍入, 对应的汇编指令 `fcvt.l.d rd,fs1,rmm`。
- 3' b101: 暂未使用, 不会出现该编码。
- 3' b110: 暂未使用, 不会出现该编码。
- 3' b111: 动态舍入, 根据浮点控制寄存器 `fcsr` 中的 `rm` 位来决定舍入模式, 对应的汇编指令 `fcvt.l.d rd, fs1`。

指令格式:

31	25	24	20	19	15	14	12	11	7	6	0
1100001		00010		fs1		rm		rd		1010011	

14.5.9 FCVT.LU.D——双精度浮点转换成无符号长整型指令

语法:

`fcvt.lu.d rd, fs1, rm`

操作:

`rd ← double_convert_to_unsigned_long(fs1)`

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

浮点状态位 NV/NX

说明:

rm 决定舍入模式:

- 3' b000: 就近向偶数舍入, 对应的汇编指令 fcvt.lu.d rd,fs1,rne。
- 3' b001: 向零舍入, 对应的汇编指令 fcvt.lu.d rd,fs1,rtz。
- 3' b010: 向负无穷舍入, 对应的汇编指令 fcvt.lu.d rd,fs1,rdn。
- 3' b011: 向正无穷舍入, 对应的汇编指令 fcvt.lu.d rd,fs1,rup。
- 3' b100: 就近向大值舍入, 对应的汇编指令 fcvt.lu.d rd,fs1,rmm。
- 3' b101: 暂未使用, 不会出现该编码。
- 3' b110: 暂未使用, 不会出现该编码。
- 3' b111: 动态舍入, 根据浮点控制寄存器 fcsr 中的 rm 位来决定舍入模式, 对应的汇编指令 fcvt.lu.d rd, fs1。

指令格式:

31	25	24	20	19	15	14	12	11	7	6	0
1100001		00011		fs1		rm		rd		1010011	

14.5.10 FCVT.S.D——双精度浮点转换成单精度浮点指令

语法:

fcvt.s.d fd, fs1, rm

操作:

fd ← double_convert_to_single(fs1)

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

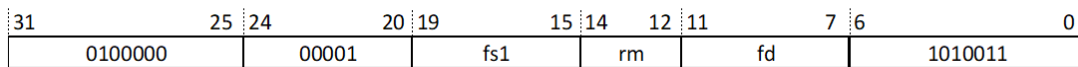
浮点状态位 NV/OF/UF/NX

说明:

rm 决定舍入模式:

- 3' b000: 就近向偶数舍入, 对应的汇编指令 fcvt.s.d fd,fs1,rne。
- 3' b001: 向零舍入, 对应的汇编指令 fcvt.s.d fd,fs1,rtz。
- 3' b010: 向负无穷舍入, 对应的汇编指令 fcvt.s.d fd,fs1,rdn。
- 3' b011: 向正无穷舍入, 对应的汇编指令 fcvt.s.d fd,fs1,rup。
- 3' b100: 就近向大值舍入, 对应的汇编指令 fcvt.s.d fd,fs1,rmm。
- 3' b101: 暂未使用, 不会出现该编码。
- 3' b110: 暂未使用, 不会出现该编码。
- 3' b111: 动态舍入, 根据浮点控制寄存器 fcsr 中的 rm 位来决定舍入模式, 对应的汇编指令 fcvt.s.d fd, fs1。

指令格式:



14.5.11 FCVT.W.D——双精度浮点转换成有符号整型指令

语法:

fcvt.w.d rd, fs1, rm

操作:

```
tmp ← double_convert_to_signed_int(fs1)
rd ← sign_extend(tmp)
```

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

浮点状态位 NV/NX

说明:

rm 决定舍入模式:

- 3' b000: 就近向偶数舍入, 对应的汇编指令 fcvt.w.d rd,fs1,rne。
- 3' b001: 向零舍入, 对应的汇编指令 fcvt.w.d rd,fs1,rtz。

- 3' b010: 向负无穷舍入, 对应的汇编指令 fcvt.w.d rd,fs1,rdn。
- 3' b011: 向正无穷舍入, 对应的汇编指令 fcvt.w.d rd,fs1,rup。
- 3' b100: 就近向大值舍入, 对应的汇编指令 fcvt.w.d rd,fs1,rm。
- 3' b101: 暂未使用, 不会出现该编码。
- 3' b110: 暂未使用, 不会出现该编码。
- 3' b111: 动态舍入, 根据浮点控制寄存器 fcsr 中的 rm 位来决定舍入模式, 对应的汇编指令 fcvt.w.d rd, fs1。

指令格式:

31	25	24	20	19	15	14	12	11	7	6	0
1100001			00000		fs1		rm		rd		1010011

14.5.12 FCVT.WU.D——双精度浮点转换成无符号整型指令

语法:

fcvt.wu.d rd, fs1, rm

操作:

tmp ← double_convert_to_unsigned_int(fs1)
rd ← sign_extend(tmp)

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

浮点状态位 NV/NX

说明:

rm 决定舍入模式:

- 3' b000: 就近向偶数舍入, 对应的汇编指令 fcvt.wu.d rd,fs1,rne。
- 3' b001: 向零舍入, 对应的汇编指令 fcvt.wu.d rd,fs1,rtz。
- 3' b010: 向负无穷舍入, 对应的汇编指令 fcvt.wu.d rd,fs1,rdn。
- 3' b011: 向正无穷舍入, 对应的汇编指令 fcvt.wu.d rd,fs1,rup。
- 3' b100: 就近向大值舍入, 对应的汇编指令 fcvt.wu.d rd,fs1,rm。
- 3' b101: 暂未使用, 不会出现该编码。
- 3' b110: 暂未使用, 不会出现该编码。
- 3' b111: 动态舍入, 根据浮点控制寄存器 fcsr 中的 rm 位来决定舍入模式, 对应的汇编指令 fcvt.wu.d rd, fs1。

指令格式:

31	25	24	20	19	15	14	12	11	7	6	0
1100001			00001		fs1		rm		rd		1010011

14.5.13 FDIV.D——双精度浮点除法指令

语法:

fdiv.d fd, fs1, fs2, rm

操作:

$fd \leftarrow fs1 / fs2$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

浮点状态位 NV/DZ/OF/UF/NX

说明:

rm 决定舍入模式:

- 3' b000: 就近向偶数舍入, 对应的汇编指令 fdiv.d fd, fs1,fs2,rne。
- 3' b001: 向零舍入, 对应的汇编指令 fdiv.d fd, fs1,fs2,rtz。
- 3' b010: 向负无穷舍入, 对应的汇编指令 fdiv.d fd, fs1,fs2,rdn。
- 3' b011: 向正无穷舍入, 对应的汇编指令 fdiv.d fd, fs1,fs2,rup。
- 3' b100: 就近向大值舍入, 对应的汇编指令 fdiv.d fd, fs1,fs2,rmm。
- 3' b101: 暂未使用, 不会出现该编码。
- 3' b110: 暂未使用, 不会出现该编码。
- 3' b111: 动态舍入, 根据浮点控制寄存器 fcsr 中的 rm 位来决定舍入模式, 对应的汇编指令 fdiv.d fd, fs1,fs2。

指令格式:

31	25	24	20	19	15	14	12	11	7	6	0
0001101			fs2		fs1		rm		fd		1010011

14.5.14 FEQ.D——双精度浮点比较相等指令

语法:

feq.d rd, fs1, fs2

操作:

```
if(fs1 == fs2)
    rd ← 1
else
    rd ← 0
```

执行权限:

M mode/S mode/U mode

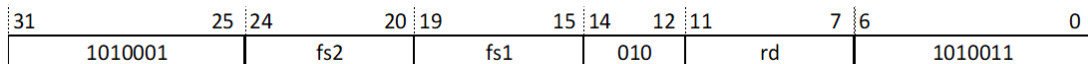
异常:

非法指令异常

影响标志位:

浮点状态位 NV

指令格式:



14.5.15 FLD——双精度浮点加载指令

语法:

fld fd, imm12(rs1)

操作:

```
address ← rs1 + sign_extend(imm12)
fd[63:0] ← mem[(address+7):address]
```

执行权限:

M mode/S mode/U mode

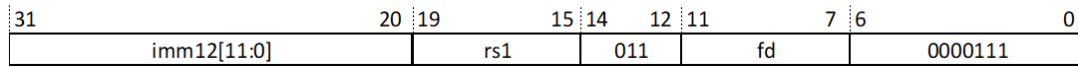
异常:

加载指令非对齐访问异常、加载指令访问错误异常、加载指令页面错误异常、非法指令异常

影响标志位:

无

指令格式:



14.5.16 FLE.D——双精度浮点比较小于等于指令

语法:

fle.d rd, fs1, fs2

操作:

```
if(fs1 <= fs2)
    rd ← 1
else
    rd ← 0
```

执行权限:

M mode/S mode/U mode

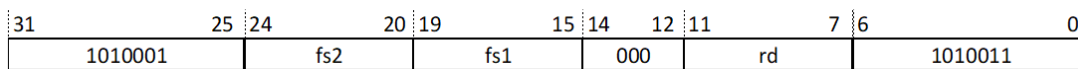
异常:

非法指令异常

影响标志位:

浮点状态位 NV

指令格式:



14.5.17 FLT.D——双精度浮点比较小于指令

语法:

flt.d rd, fs1, fs2

操作:

```
if(fs1 < fs2)
    rd ← 1
else
    rd ← 0
```


执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

浮点状态位 NV

指令格式:

31	25	24	20	19	15	14	12	11	7	6	0
1010001			fs2		fs1		001		rd		1010011

14.5.18 FMADD.D——双精度浮点乘累加指令

语法:

fmadd.d fd, fs1, fs2, fs3, rm

操作:

$fd \leftarrow fs1 * fs2 + fs3$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

浮点状态位 NV/OF/UF/IX

说明:

rm 决定舍入模式:

- 3' b000: 就近向偶数舍入, 对应的汇编指令 fmadd.d fd,fs1, fs2, fs3, rne。
- 3' b001: 向零舍入, 对应的汇编指令 fmadd.d fd,fs1, fs2, fs3, rtz。
- 3' b010: 向负无穷舍入, 对应的汇编指令 fmadd.d fd,fs1, fs2, fs3, rdn。
- 3' b011: 向正无穷舍入, 对应的汇编指令 fmadd.d fd,fs1, fs2, fs3, rup。
- 3' b100: 就近向大值舍入, 对应的汇编指令 fmadd.d fd,fs1, fs2, fs3, rmm。
- 3' b101: 暂未使用, 不会出现该编码。
- 3' b110: 暂未使用, 不会出现该编码。

- 3' b111: 动态舍入, 根据浮点控制寄存器 fcsr 中的 rm 位来决定舍入模式, 对应的汇编指令 fmadd.d fd, fs1, fs2, fs3。

指令格式:

31	27	26	25	24	20	19	15	14	12	11	7	6	0		
fs3			01		fs2			fs1		rm		fd		1000011	

14.5.19 FMAX.D——双精度浮点取最大值指令

语法:

fmax.d fd, fs1, fs2

操作:

```
if(fs1 >= fs2)
    fd ← fs1
else
    fd ← fs2
```

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

浮点状态位 NV

指令格式:

31	25	24	20	19	15	14	12	11	7	6	0		
0010101			fs2			fs1		001		fd		1010011	

14.5.20 FMIN.D——双精度浮点取最小值指令

语法:

fmin.d fd, fs1, fs2

操作:

```
if(fs1 < fs2)
    fd ← fs1
else
```

$fd \leftarrow fs2$

执行权限:

M mode/S mode/U mode

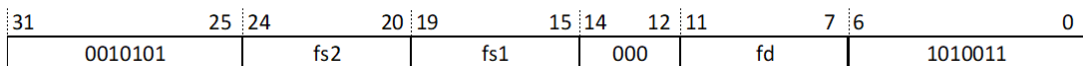
异常:

非法指令异常

影响标志位:

浮点状态位 NV

指令格式:



14.5.21 FMSUB.D——双精度浮点乘累减指令

语法:

fmsub.d fd, fs1, fs2, fs3, rm

操作:

$fd \leftarrow fs1 * fs2 - fs3$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

浮点状态位 NV/OF/UF/IX

说明:

rm 决定舍入模式:

- 3' b000: 就近向偶数舍入, 对应的汇编指令 fmsub.d fd, fs1, fs2, fs3, rne。
- 3' b001: 向零舍入, 对应的汇编指令 fmsub.d fd, fs1, fs2, fs3, rtz。
- 3' b010: 向负无穷舍入, 对应的汇编指令 fmsub.d fd, fs1, fs2, fs3, rdn。
- 3' b011: 向正无穷舍入, 对应的汇编指令 fmsub.d fd, fs1, fs2, fs3, rup。
- 3' b100: 就近向大值舍入, 对应的汇编指令 fmsub.d fd, fs1, fs2, fs3, rmm。
- 3' b101: 暂未使用, 不会出现该编码。

- 3' b110: 暂未使用, 不会出现该编码。
- 3' b111: 动态舍入, 根据浮点控制寄存器 fcsr 中的 rm 位来决定舍入模式, 对应的汇编指令 fmsub.d fd, fs1, fs2, fs3。

指令格式:

31	27	26	25	24	20	19	15	14	12	11	7	6	0
fs3			01		fs2		fs1		rm		fd		1000111

14.5.22 FMUL.D——双精度浮点乘法指令

语法:

fmul.d fd, fs1, fs2, rm

操作:

fd ← fs1 * fs2

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

浮点状态位 NV/OF/UF/NX

说明:

rm 决定舍入模式:

- 3' b000: 就近向偶数舍入, 对应的汇编指令 fmul.d fd, fs1, fs2, rne。
- 3' b001: 向零舍入, 对应的汇编指令 fmul.d fd, fs1, fs2, rtz。
- 3' b010: 向负无穷舍入, 对应的汇编指令 fmul.d fd, fs1, fs2, rdn。
- 3' b011: 向正无穷舍入, 对应的汇编指令 fmul.d fd, fs1, fs2, rup。
- 3' b100: 就近向大值舍入, 对应的汇编指令 fmul.d fd, fs1, fs2, rmm。
- 3' b101: 暂未使用, 不会出现该编码。
- 3' b110: 暂未使用, 不会出现该编码。
- 3' b111: 动态舍入, 根据浮点控制寄存器 fcsr 中的 rm 位来决定舍入模式, 对应的汇编指令 fmul. fd, fs1,fs2。

指令格式:

31	25	24	20	19	15	14	12	11	7	6	0
0001001			fs2		fs1		rm		fd		1010011

14.5.23 FMV.D.X——双精度浮点写传送指令

语法:

fmv.d.x fd, rs1

操作:

fd ← rs1

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

无

说明:

整型寄存器搬运到浮点寄存器

指令格式:

31	25	24	20	19	15	14	12	11	7	6	0
1111001		00000		rs1		000		fd		1010011	

14.5.24 FMV.X.D——双精度浮点读传送指令

语法:

fmv.x.d rd, fs1

操作:

rd ← fs1

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

无

说明:

浮点寄存器搬运到整型寄存器

指令格式:

31	25	24	20	19	15	14	12	11	7	6	0
1110001			00000		fs1		000		rd		1010011

14.5.25 FNMADD.D——双精度浮点乘累加取负指令

语法:

fnmadd.d fd, fs1, fs2, fs3, rm

操作:

$fd \leftarrow -(fs1 * fs2 + fs3)$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

浮点状态位 NV/OF/UF/IX

说明:

rm 决定舍入模式:

- 3' b000: 就近向偶数舍入, 对应的汇编指令 fnmadd.d fd, fs1, fs2, fs3, rne。
- 3' b001: 向零舍入, 对应的汇编指令 fnmadd.d fd, fs1, fs2, fs3, rtz。
- 3' b010: 向负无穷舍入, 对应的汇编指令 fnmadd.d fd, fs1, fs2, fs3, rdn。
- 3' b011: 向正无穷舍入, 对应的汇编指令 fnmadd.d fd, fs1, fs2, fs3, rup。
- 3' b100: 就近向大值舍入, 对应的汇编指令 fnmadd.d fd, fs1, fs2, fs3, rmm。
- 3' b101: 暂未使用, 不会出现该编码。
- 3' b110: 暂未使用, 不会出现该编码。
- 3' b111: 动态舍入, 根据浮点控制寄存器 fcsr 中的 rm 位来决定舍入模式, 对应的汇编指令 fnmadd.d fd, fs1, fs2, fs3。

指令格式:

31	27	26	25	24	20	19	15	14	12	11	7	6	0
fs3			01		fs2		fs1		rm		fd		1001111

14.5.26 FNMSUB.D——双精度浮点乘累减取负指令

语法:

fnmsub.d fd, fs1, fs2, fs3, rm

操作:

$fd \leftarrow -(fs1 * fs2 - fs3)$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

浮点状态位 NV/OF/UF/IX

说明:

rm 决定舍入模式:

- 3' b000: 就近向偶数舍入, 对应的汇编指令 fnmsub.d fd, fs1, fs2, fs3, rne。
- 3' b001: 向零舍入, 对应的汇编指令 fnmsub.d fd, fs1, fs2, fs3, rtz。
- 3' b010: 向负无穷舍入, 对应的汇编指令 fnmsub.d fd, fs1, fs2, fs3, rdn。
- 3' b011: 向正无穷舍入, 对应的汇编指令 fnmsub.d fd, fs1, fs2, fs3, rup。
- 3' b100: 就近向大值舍入, 对应的汇编指令 fnmsub.d fd, fs1, fs2, fs3, rmm。
- 3' b101: 暂未使用, 不会出现该编码。
- 3' b110: 暂未使用, 不会出现该编码。
- 3' b111: 动态舍入, 根据浮点控制寄存器 fcsr 中的 rm 位来决定舍入模式, 对应的汇编指令 fnmsub.d fd, fs1, fs2, fs3。

指令格式:

31	27	26	25	24	20	19	15	14	12	11	7	6	0
fs3			01		fs2		fs1		rm		fd		1001011

14.5.27 FSD——双精度浮点存储指令

语法:

fsd fs2, imm12(rs1)

操作:

address ← rs1 + sign_extend(imm12)

mem[(address+63):address] ← fs2[63:0]

执行权限:

M mode/S mode/U mode

异常:

存储指令非对齐访问异常、存储指令访问错误异常、存储指令页面错误异常、非法指令异常

指令格式:

31	25	24	20	19	15	14	12	11	7	6	0
imm12[11:5]			fs2		rs1		011		imm12[4:0]		0100111

14.5.28 FSGNJ.D——双精度浮点符号注入指令

语法:

fsgnj.d fd, fs1, fs2

操作:

fd[62:0] ← fs1[62:0]

fd[63] ← fs2[63]

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

无

指令格式:

31	25	24	20	19	15	14	12	11	7	6	0
0010001			fs2		fs1		000		fd		1010011

14.5.29 FSGNJN.D——双精度浮点符号取反注入指令

语法:

fsgnjn.d fd, fs1, fs2

操作:

fd[62:0] ← fs1[62:0]

fd[63] ← !fs2[63]

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

无

指令格式:

31	25	24	20	19	15	14	12	11	7	6	0
0010001		fs2		fs1		001		fd		1010011	

14.5.30 FSGNJX.D——双精度浮点符号异或注入指令

语法:

fsgnjx.d fd, fs1, fs2

操作:

$fd[62:0] \leftarrow fs1[62:0]$

$fd[63] \leftarrow fs1[63] \wedge fs2[63]$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

无

指令格式:

31	25	24	20	19	15	14	12	11	7	6	0
0010001		fs2		fs1		010		fd		1010011	

14.5.31 FSQRT.D——双精度浮点开方指令

语法:

fsqrt.d fd, fs1, rm

操作:

$fd \leftarrow \text{sqrt}(fs1)$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

浮点状态位 NV/NX

说明:

rm 决定舍入模式:

- 3' b000: 就近向偶数舍入, 对应的汇编指令 fsqrt.d fd, fs1, rne。
- 3' b001: 向零舍入, 对应的汇编指令 fsqrt.d fd, fs1, rtz。
- 3' b010: 向负无穷舍入, 对应的汇编指令 fsqrt.d fd, fs1, rdn。
- 3' b011: 向正无穷舍入, 对应的汇编指令 fsqrt.d fd, fs1, rup。
- 3' b100: 就近向大值舍入, 对应的汇编指令 fsqrt.d fd, fs1, rmm。
- 3' b101: 暂未使用, 不会出现该编码。
- 3' b110: 暂未使用, 不会出现该编码。
- 3' b111: 动态舍入, 根据浮点控制寄存器 fcsr 中的 rm 位来决定舍入模式, 对应的汇编指令 fsqrt.d fd, fs1。

指令格式:

31	25	24	20	19	15	14	12	11	7	6	0	
0101101			00000			fs1		rm		fd		1010011

14.5.32 FSUB.D——双精度浮点减法指令

语法:

fsub.d fd, fs1, fs2, rm

操作:

$fd \leftarrow fs1 - fs2$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

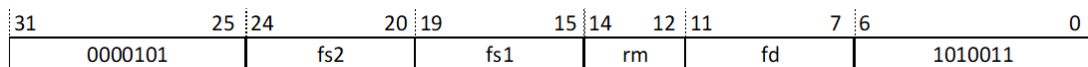
浮点状态位 NV/OF/NX

说明:

rm 决定舍入模式:

- 3' b000: 就近向偶数舍入, 对应的汇编指令 fsub.f_d, fs1, fs2, rne。
- 3' b001: 向零舍入, 对应的汇编指令 fsub.d fd, fs1, fs2, rtz。
- 3' b010: 向负无穷舍入, 对应的汇编指令 fsub.d fd, fs1, fs2, rdn。
- 3' b011: 向正无穷舍入, 对应的汇编指令 fsub.d fd, fs1, fs2, rup。
- 3' b100: 就近向大值舍入, 对应的汇编指令 fsub.d fd, fs1, fs2, rmm。
- 3' b101: 暂未使用, 不会出现该编码。
- 3' b110: 暂未使用, 不会出现该编码。
- 3' b111: 动态舍入, 根据浮点控制寄存器 fcsr 中的 rm 位来决定舍入模式, 对应的汇编指令 fsub.dfd, fs1, fs2。

指令格式:



14.6 附录 A-6 C 指令术语

以下是对 C906 实现的 RISC-V C 指令的具体描述, 每条指令位宽为 16 位, 指令按英文字母顺序排列。

14.6.1 C.ADD——有符号加法指令

语法:

c.add rd, rs2

操作:

rd ← rs1 + rs2

执行权限:

M mode/S mode/U mode

异常:

无

说明:

$rs1 = rd \neq 0$

$rs2 \neq 0$

指令格式:

15	13	12	11	7	6	2	1	0
100	1	rs1/rd			rs2		10	

14.6.2 C.ADDI——有符号立即数加法指令

语法:

`c.addi rd, nzimm6`

操作:

$rd \leftarrow rs1 + \text{sign_extend}(nzimm6)$

执行权限:

M mode/S mode/U mode

异常:

无

说明:

$rs1 = rd \neq 0$

$nzimm6 \neq 0$

指令格式:

15	13	12	11	7	6	2	1	0
000		rs1/rd			nzimm6[4:0]		01	
		└─── nzimm6[5]						

14.6.3 C.ADDIW——低 32 位有符号立即数加法指令

语法:

c.addiw rd, imm6

操作:

tmp[31:0] ← rs1[31:0] + sign_extend(imm6)
rd ← sign_extend(tmp[31:0])

执行权限:

M mode/S mode/U mode

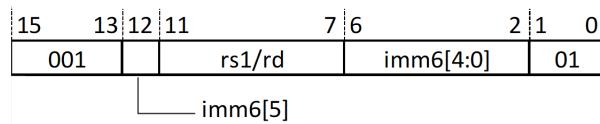
异常:

无

说明:

rs1 = rd != 0

指令格式:



14.6.4 C.ADDI4SPN——4 倍立即数和堆栈指针相加指令

语法:

c.addi4spn rd, sp, nzuimm8<<2

操作:

rd ← sp + zero_extend(nzuimm8<<2)

执行权限:

M mode/S mode/U mode

异常:

无

说明:

nzuimm8 != 0

rd 编码代表寄存器如下:

- 000 x8
- 001 x9
- 010 x10
- 011 x11
- 100 x12
- 101 x13
- 110 x14
- 111 x15

指令格式:

15	13	12	5	4	2	1	0
000		nzuimm8[3:2 7:4 0 1]			rd		00

14.6.5 C.ADDI16SP——加 16 倍立即数到堆栈指针指令

语法:

c.addi16sp sp, nzuimm6<<4

操作:

sp ← sp + sign_extend(nzuimm6<<4)

执行权限:

M mode/S mode/U mode

异常:

无

指令格式:

15	13	12	11	7	6	2	1	0
011		00010						01
					nzimm6[5]		nzimm6[0 2 4:3 1]	

14.6.6 C.ADDW——低 32 位有符号加法指令

语法:

c.addw rd, rs2

操作:

tmp[31:0] ← rs1[31:0] + rs2[31:0]
rd ← sign_extend(tmp[31:0])

执行权限:

M mode/S mode/U mode

异常:

无

说明:

rs1 = rd

rd/rs1, rs2 编码代表寄存器如下:

- 000: x8
- 001: x9
- 010: x10
- 011: x11
- 100: x12
- 101: x13
- 110: x14
- 111: x15

指令格式:

15	13	12	11	10	9	7	6	5	4	2	1	0
100	1	11	rs1/rd	01	rs2	01						

14.6.7 C.AND——按位与指令

语法:

c.and rd, rs2

操作:

rd ← rs1 & rs2

执行权限:

M mode/S mode/U mode

异常:

无

说明:

$rs1 = rd$

rd/rs1, rs2 编码代表寄存器如下:

- 000: x8
- 001: x9
- 010: x10
- 011: x11
- 100: x12
- 101: x13
- 110: x14
- 111: x15

指令格式:

15	13	12	11	10	9	7	6	5	4	2	1	0
100	0	11	rs1/rd			11	rs2			01		

14.6.8 C.ANDI——立即数按位与指令

语法:

c.andi rd, imm6

操作:

$rd \leftarrow rs1 \& \text{sign_extend}(imm6)$

执行权限:

M mode/S mode/U mode

异常:

无

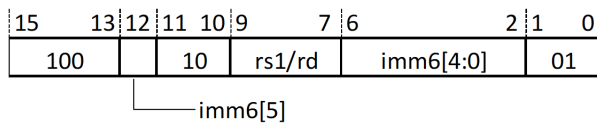
说明:

$rs1 = rd$

rd/rs1 编码代表寄存器如下:

- 000: x8
- 001: x9
- 010: x10
- 011: x11
- 100: x12
- 101: x13
- 110: x14
- 111: x15

指令格式:



14.6.9 C.BEQZ——等于零分支指令

语法:

c.beqz rs1, label

操作:

```
if (rs1 == 0)
    next pc = current pc + sign_ext(imm8<<1);
else
    next pc = current pc + 2;
```

执行权限:

M mode/S mode/U mode

异常:

无

说明:

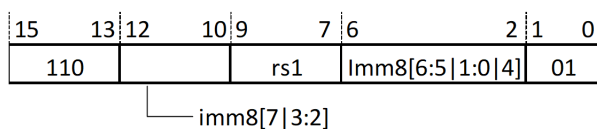
rs1 编码代表寄存器如下:

- 000: x8
- 001: x9
- 010: x10
- 011: x11
- 100: x12
- 101: x13
- 110: x14
- 111: x15

汇编器根据 label 算出 imm8

指令跳转范围为±256B 地址空间

指令格式:



14.6.10 C.BNEZ——不等于零分支指令

语法:

```
c.bnez rs1, label
```

操作:

```
if (rs1 != 0)
    next pc = current pc + sign_ext(imm8<<1);
else
    next pc = current pc + 2;
```

执行权限:

M mode/S mode/U mode

异常:

无

说明:

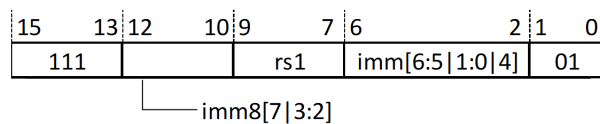
rs1 编码代表寄存器如下:

- 000: x8
- 001: x9
- 010: x10
- 011: x11
- 100: x12
- 101: x13
- 110: x14
- 111: x15

汇编器根据 label 算出 imm12

指令跳转范围为±256B 地址空间

指令格式:



14.6.11 C.EBREAK——断点指令

语法:

```
c.ebreak
```

操作:

产生断点异常或者进入调试模式

执行权限:

M mode/S mode/U mode

异常:

断点异常

指令格式:

15	13	12	11	7	6	2	1	0
100	1	00000			00000		10	

14.6.12 C.FLD——浮点双字加载指令

语法:

c.fld fd, uimm5<<3(rs1)

操作:

address \leftarrow rs1+ zero_extend(uimm5<<3)

fd \leftarrow mem[address+7:address]

执行权限:

M mode/S mode/U mode

异常:

加载指令非对齐访问异常、加载指令访问错误异常、加载指令页面错误异常

说明:

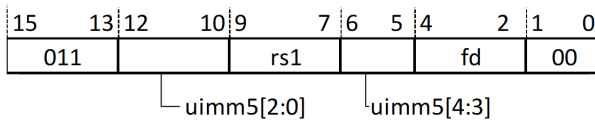
rs1 编码代表寄存器如下:

- 000: x8
- 001: x9
- 010: x10
- 011: x11
- 100: x12
- 101: x13
- 110: x14
- 111: x15

fd 编码代表寄存器如下:

- 000: f8
- 001: f9
- 010: f10
- 011: f11
- 100: f12
- 101: f13
- 110: f14
- 111: f15

指令格式:



14.6.13 C.FLDSP——浮点双字堆栈加载指令

语法:

c.fldsp fd, uimm6<<3(sp)

操作:

address \leftarrow sp+ zero_extend(uimm6<<3)

fd \leftarrow mem[address+7:address]

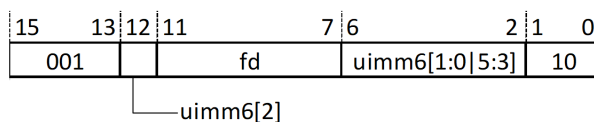
执行权限:

M mode/S mode/U mode

异常:

加载指令非对齐访问异常、加载指令访问错误异常、加载指令页面错误异常

指令格式:



14.6.14 C.FSD——浮点双字存储指令

语法:

c.fsd fs2, uimm5<<3(rs1)

操作:

address \leftarrow rs1 + zero_extend(uimm5 << 3)
 mem[address+7:address] \leftarrow fs2

执行权限:

M mode/S mode/U mode

异常:

存储指令非对齐访问异常、存储指令访问错误异常、存储指令页面错误异常

说明:

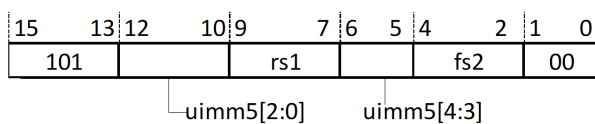
fs1 编码代表寄存器如下:

- 000: x8
- 001: x9
- 010: x10
- 011: x11
- 100: x12
- 101: x13
- 110: x14
- 111: x15

rs2 编码代表寄存器如下:

- 000: f8
- 001: f9
- 010: f10
- 011: f11
- 100: f12
- 101: f13
- 110: f14
- 111: f15

指令格式:



14.6.15 C.FSDSP——浮点双字堆栈存储指令

语法:

c.fsdsp fs2, uimm6 << 3(sp)

操作:

address \leftarrow sp + zero_extend(uimm6 << 3)
 mem[address+7:address] \leftarrow fs2

执行权限:

M mode/S mode/U mode

异常:

存储指令非对齐访问异常、存储指令访问错误异常、存储指令页面错误异常

指令格式:

15	13	12	7	6	2	1	0
101		uimm6[2:0 5:3]			fs2		10

14.6.16 C.J——无条件跳转指令

语法:

c.j label

操作:

next pc \leftarrow current pc + sign_extend(imm << 1);

执行权限:

M mode/S mode/U mode

异常:

无

说明:

汇编器根据 label 算出 imm11
 指令跳转范围为 ± 2 KB 地址空间

指令格式:

15	13	12	2	1	0
101		imm11[10 3 8:7 9 5 6 2:0 4]			01

14.6.17 C.JALR——寄存器跳转子程序指令

语法:

c.jalr rs1

操作:

next pc ← rs1;
x1 ← current pc + 2;

执行权限:

M mode/S mode/U mode

异常:

无

说明:

rs1 != 0。
MMU 打开时, 跳转范围是全部 512GB 地址空间。
MMU 关闭时, 跳转范围是全部 1TB 地址空间。

指令格式:

15	13	12	11	7	6	2	1	0
100	1	rs1			00000		10	

14.6.18 C.JR——寄存器跳转指令

语法:

c.jr rs1

操作:

next pc = rs1;

执行权限:

M mode/S mode/U mode

异常:

无

说明:

rs1 != 0。

MMU 打开时，跳转范围是全部 512GB 地址空间。

MMU 关闭时，跳转范围是全部 1TB 地址空间。

指令格式：

15	13	12	11	7	6	2	1	0
100	0	rs1			00000		10	

14.6.19 C.LD——双字加载指令

语法：

c.ld rd, uimm5<<3(rs1)

操作：

address ← rs1+ zero_extend(uimm5<<3)

rd ← mem[address+7:address]

执行权限：

M mode/S mode/U mode

异常：

加载指令非对齐访问异常、加载指令访问错误异常、加载指令页面错误异常

说明：

rs1/rd 编码代表寄存器如下：

- 000: x8
- 001: x9
- 010: x10
- 011: x11
- 100: x12
- 101: x13
- 110: x14
- 111: x15

指令格式：

15	13	12	10	9	7	6	5	4	2	1	0
011		rs1			rd		00				
└───uimm5[2:0]						└───uimm5[4:3]					

14.6.20 C.LDSP——双字堆栈加载指令

语法:

`c.ldsp rd, uimm6<<3(sp)`

操作:

$address \leftarrow sp + zero_extend(uimm6 \ll 3)$

$rd \leftarrow mem[address+7:address]$

执行权限:

M mode/S mode/U mode

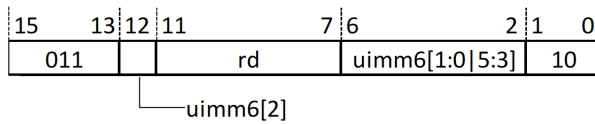
异常:

加载指令非对齐访问异常、加载指令访问错误异常、加载指令页面错误异常

说明:

$rd \neq 0$

指令格式:



14.6.21 C.LI——立即数传送指令

语法:

`c.li rd, imm6`

操作:

$rd \leftarrow sign_extend(imm6)$

执行权限:

M mode/S mode/U mode

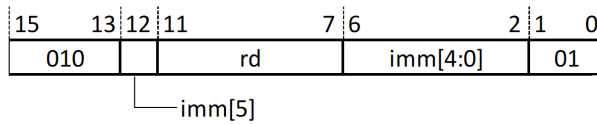
异常:

无

说明:

$rd \neq 0$ 。

指令格式:



14.6.22 C.LUI——高位立即数传送指令

语法:

c.lui rd, nzimm6

操作:

$rd \leftarrow \text{sign_extend}(nzimm6 \ll 12)$

执行权限:

M mode/S mode/U mode

异常:

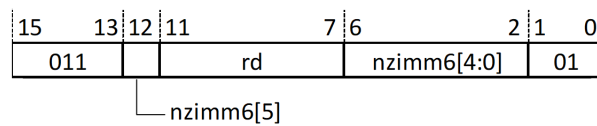
无

说明:

rd != 0。

Nzimm6 != 0。

指令格式:



14.6.23 C.LW——字加载指令

语法:

c.lw rd, uimm5<<2(rs1)

操作:

$\text{address} \leftarrow rs1 + \text{zero_extend}(uimm5 \ll 2)$

$\text{tmp}[31:0] \leftarrow \text{mem}[\text{address}+3:\text{address}]$

$rd \leftarrow \text{sign_extend}(\text{tmp}[31:0])$

执行权限:

M mode/S mode/U mode

异常:

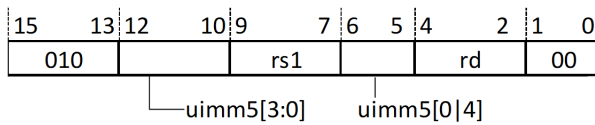
加载指令非对齐访问异常、加载指令访问错误异常、加载指令页面错误异常

说明:

rs1/rd 编码代表寄存器如下:

- 000: x8
- 001: x9
- 010: x10
- 011: x11
- 100: x12
- 101: x13
- 110: x14
- 111: x15

指令格式:



14.6.24 C.LWSP——字堆栈加载指令

语法:

c.lwsp rd, uimm6<<2(sp)

操作:

```
address ← sp + zero_extend(uimm6<<2)
tmp[31:0] ← mem[address+3:address]
rd ← sign_extend(tmp[31:0])
```

执行权限:

M mode/S mode/U mode

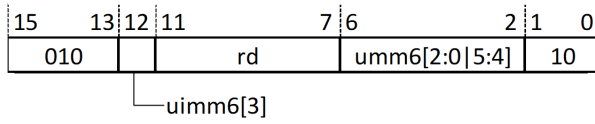
异常:

加载指令非对齐访问异常、加载指令访问错误异常、加载指令页面错误异常

说明:

rd != 0

指令格式:



14.6.25 C.MV——数据传送指令

语法:

c.mv rd, rs2

操作:

rd ← rs2;

执行权限:

M mode/S mode/U mode

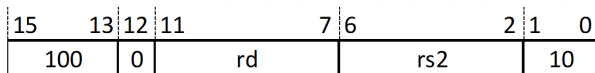
异常:

无

说明:

rs2 != 0, rd != 0.

指令格式:



14.6.26 C.NOP——空指令

语法:

c.nop

操作:

无操作

执行权限:

M mode/S mode/U mode

异常:

无

指令格式:

15	13	12	11	7	6	2	1	0
000	0	00000	00000	01				

14.6.27 C.OR——按位或指令

语法:

c.or rd, rs2

操作:

$rd \leftarrow rs1 \mid rs2$

执行权限:

M mode/S mode/U mode

异常:

无

说明:

rs1 = rd

rd/rs1 编码代表寄存器如下:

- 000: x8
- 001: x9
- 010: x10
- 011: x11
- 100: x12
- 101: x13
- 110: x14
- 111: x15

指令格式:

15	13	12	11	10	9	7	6	5	4	2	1	0
100	0	11	rs1/rd	10	rs2	01						

14.6.28 C.SD——双字存储指令

语法:

`c.sd rs2, uimm5<<3(rs1)`

操作:

$address \leftarrow rs1 + zero_extend(uimm5 \ll 3)$
 $mem[address+7:address] \leftarrow rs2$

执行权限:

M mode/S mode/U mode

异常:

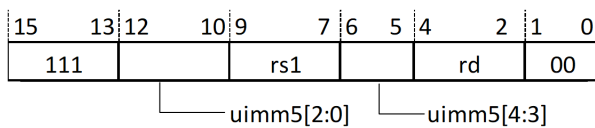
存储指令非对齐访问异常、存储指令访问错误异常、存储指令页面错误异常

说明:

rs1/rd 编码代表寄存器如下:

- 000: x8
- 001: x9
- 010: x10
- 011: x11
- 100: x12
- 101: x13
- 110: x14
- 111: x15

指令格式:



14.6.29 C.SDSP——双字堆栈存储指令

语法:

`c.fsdsp rs2, uimm6<<3(sp)`

操作:

$address \leftarrow sp + zero_extend(uimm6 \ll 3)$
 $mem[address+7:address] \leftarrow rs2$

执行权限:

M mode/S mode/U mode

异常:

存储指令非对齐访问异常、存储指令访问错误异常、存储指令页面错误异常

指令格式:

15	13	12	7	6	2	1	0
111		uimm6[2:0 5:3]			rs2		10

14.6.30 C.SLLI——立即数逻辑左移指令

语法:

c.slli rd, nzuimm6

操作:

$rd \leftarrow rs1 \ll nzuimm6$

执行权限:

M mode/S mode/U mode

异常:

无

说明:

$rs1 == rd$

$rd/rs1 \neq 0, nzuimm6 \neq 0$

指令格式:

15	13	12	11	7	6	2	1	0
000		rs1/rd			nzuimm6[4:0]		10	

└─ nzuimm6[5]

14.6.31 C.SRAI——立即数算数右移指令

语法:

c.srli rd, nzuimm6

操作:

$rd \leftarrow rs1 \ggg nzuimm6$

执行权限:

M mode/S mode/U mode

异常:

无

说明:

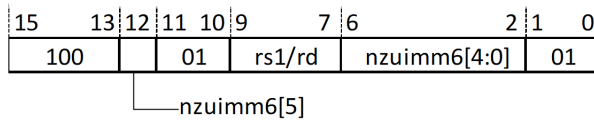
$nzuimm6 \neq 0$

$rs1 == rd$

rs1/rd 编码代表寄存器如下:

- 000: x8
- 001: x9
- 010: x10
- 011: x11
- 100: x12
- 101: x13
- 110: x14
- 111: x15

指令格式:



14.6.32 C.SRLI——立即数逻辑右移指令

语法:

`c.srli rd, nzuimm6`

操作:

$rd \leftarrow rs1 \gg nzuimm6$

执行权限:

M mode/S mode/U mode

异常:

无

说明:

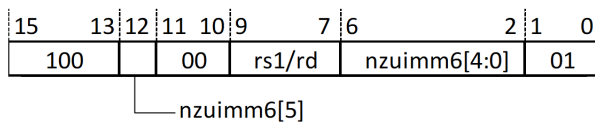
$nzuimm6 \neq 0$

$rs1 == rd$

rs1/rd 编码代表寄存器如下:

- 000: x8
- 001: x9
- 010: x10
- 011: x11
- 100: x12
- 101: x13
- 110: x14
- 111: x15

指令格式:



14.6.33 C.SW——字存储指令

语法:

`c.sw rs2, uimm5<<2(rs1)`

操作:

$address \leftarrow rs1 + zero_extend(uimm5 \ll 2)$

$mem[address+3:address] \leftarrow rs2$

执行权限:

M mode/S mode/U mode

异常:

存储指令非对齐访问异常、存储指令访问错误异常、存储指令页面错误异常

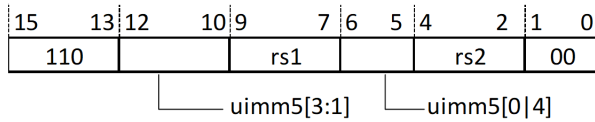
说明:

rs1/rs2 编码代表寄存器如下:

- 000: x8
- 001: x9
- 010: x10
- 011: x11

- 100: x12
- 101: x13
- 110: x14
- 111: x15

指令格式:



14.6.34 C.SWSP——字堆栈存储指令

语法:

c.swsp rs2, uimm6<<2(sp)

操作:

address ← sp+ zero_extend(uimm6<<2)
 mem[address+3:address] ←rs2

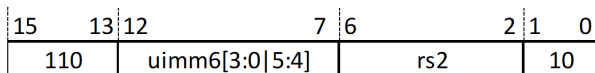
执行权限:

M mode/S mode/U mode

异常:

存储指令非对齐访问异常、存储指令访问错误异常、存储指令页面错误异常

指令格式:



14.6.35 C.SUB——有符号减法指令

语法:

c.sub rd, rs2

操作:

rd ← rs1 - rs2

执行权限:

M mode/S mode/U mode

异常:

无

说明:

rs1 == rd

rs1/rd 编码代表寄存器如下:

- 000: x8
- 001: x9
- 010: x10
- 011: x11
- 100: x12
- 101: x13
- 110: x14
- 111: x15

指令格式:

15	13	12	11	10	9	7	6	5	4	2	1	0
100	0	11	rs1/rd			00	rs2			01		

14.6.36 C.SUBW——低 32 位有符号减法指令

语法:

c.subw rd, rs2

操作:

tmp[31:0] ← rs1[31:0] - rs2[31:0]

rd ← sign_extend(tmp)

执行权限:

M mode/S mode/U mode

异常:

无

说明:

rs1 == rd

rs1/rd 编码代表寄存器如下:

- 000: x8
- 001: x9

- 010: x10
- 011: x11
- 100: x12
- 101: x13
- 110: x14
- 111: x15

指令格式

15	13	12	11	10	9	7	6	5	4	2	1	0
100	1	11	rs1/rd	00	rs2	01						

14.6.37 C.XOR——按位异或指令

语法:

c.xor rd, rs2

操作:

$rd \leftarrow rs1 \wedge rs2$

执行权限:

M mode/S mode/U mode

异常:

无

说明:

rs1 == rd

rs1/rd 编码代表寄存器如下:

- 000: x8
- 001: x9
- 010: x10
- 011: x11
- 100: x12
- 101: x13
- 110: x14
- 111: x15

指令格式:

15	13	12	11	10	9	7	6	5	4	2	1	0
100	0	11	rs1/rd	01	rs2	01						

14.7 附录 A-7 V 指令术语

V 指令集下的所有指令需要在配置 VECTOR_SIMD 并且 mstatus.vs != 2' b00 时才正常执行，否则产生非法指令异常。符合以上条件执行任意矢量指令，mstatus.vs 置位成 2' b11。

C906 的 V 指令集支持的元素位宽为 SEW=8/16/32。普通矢量整型指令可支持 SEW=8/16/32 的计算，而矢量整型扩位/缩位指令，只支持 SEW=8/16 的元素位宽，否则产生非法指令异常；对于矢量浮点指令，普通矢量浮点指令只支持 SEW=16/32 的计算，只有矢量浮点扩位转换指令和矢量浮点缩位转换指令可在 SEW=8 时正常执行，其他矢量浮点扩位/缩位指令必须在 SEW=16 时执行，否则产生非法指令异常。

以下是对 C906 实现的 RISC-V V 指令的具体描述，每条指令位宽为 32 位，指令按英文字母顺序排列。

本章节中用到的缩写含义如下：

1. VLEN：矢量寄存器位宽，C906 中为 128。
2. VLMAX：当前矢量寄存器组最大元素个数，VLMAX=VLEN/SEW*LMUL。

14.7.1 VAADD.VI——矢量立即数整型加法取平均数指令

语法：

```
vaadd.vi vd, vs2, imm, vm
```

操作：

$$vd[i] \leftarrow (vs2[i] + \text{sign_extend}(imm) + \text{round}) \gg 1$$

执行权限：

M mode/S mode/U mode

异常：

非法指令异常

影响标志位：

无

说明：

根据定点舍入模式寄存器 vxrm 确定舍入模式：

- 2' b00: 就近向大值舍入。
- 2' b01: 就近向偶数舍入。
- 2' b10: 向零舍入。
- 2' b11: 向奇数舍入。

当 vm=1 时，指令未被 mask，对应汇编指令 vaadd.vi vd, vs2, imm。

当 vm=0 时，指令被 mask，对应汇编指令 vaadd.vi vd, vs2, imm,v0.t。

指令格式：

31	26	25	24	20	19	15	14	12	11	7	6	0
100100			vm	vs2		imm		011		vd		1010111

14.7.2 VAADD.VV——矢量整型加法取平均数指令

语法:

vaadd.vv vd, vs2, vs1, vm

操作:

$vd[i] \leftarrow (vs2[i] + vs1[i] + \text{round}) \gg 1$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

无

说明:

根据定点舍入模式寄存器 vxrm 确定舍入模式:

- 2' b00: 就近向大值舍入。
- 2' b01: 就近向偶数舍入。
- 2' b10: 向零舍入。
- 2' b11: 向奇数舍入。

当 vm=1 时, 指令未被 mask, 对应汇编指令 vaadd.vv vd, vs2, vs1。

当 vm=0 时, 指令被 mask, 对应汇编指令 vaadd.vv vd, vs2, vs1,v0.t。

指令格式:

31	26	25	24	20	19	15	14	12	11	7	6	0
100100			vm	vs2		vs1		000		vd		1010111

14.7.3 VAADD.VX——矢量标量整型加法取平均数指令

语法:

vaadd.vx vd, vs2, rs1, vm

操作:

$$vd[i] \leftarrow (vs2[i] + rs1 + round) \gg 1$$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

无

说明:

根据定点舍入模式寄存器 vxrm 确定舍入模式:

- 2' b00: 就近向大值舍入。
- 2' b01: 就近向偶数舍入。
- 2' b10: 向零舍入。
- 2' b11: 向奇数舍入。

当 vm=1 时, 指令未被 mask, 对应汇编指令 vaadd.vx vd, vs2, rs1。

当 vm=0 时, 指令被 mask, 对应汇编指令 vaadd.vx vd, vs2, rs1,v0.t。

指令格式:

31		26	25	24		20	19		15	14	12	11		7	6		0
	100100		vm		vs2			rs1		100			vd			1010111	

14.7.4 VADC.VIM——矢量立即数整型带进位加法指令

语法:

vadc.vim vd, vs2, imm, v0

操作:

$$vd[i] \leftarrow vs2[i] + sign_extend(imm) + v0[i].LSB$$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

无

指令格式:

31	26	25	24	20	19	15	14	12	11	7	6	0
010000		1	vs2	imm		011		vd		1010111		

14.7.5 VADC.VVM——矢量整型带进位加法指令

语法:

vadc.vvm vd, vs2, vs1, v0

操作:

$vd[i] \leftarrow vs2[i] + vs1[i] + v0[i].LSB$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

无

指令格式:

31	26	25	24	20	19	15	14	12	11	7	6	0
010000		1	vs2	vs1		000		vd		1010111		

14.7.6 VADC.VXM——矢量标量整型带进位加法指令

语法:

vadc.vxm vd, vs2, rs1, v0

操作:

$vd[i] \leftarrow vs2[i] + rs1 + v0[i].LSB$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

无

指令格式:

31	26	25	24	20	19	15	14	12	11	7	6	0
010000			1	vs2		rs1		100		vd		1010111

14.7.7 VADD.VI——矢量立即数整型加法指令

语法:

vadd.vi vd, vs2, imm, vm

操作:

$vd[i] \leftarrow vs2[i] + \text{sign_extend}(imm)$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

无

说明:

当 $vm=1$ 时, 指令未被 mask, 对应汇编指令 vadd.vi vd, vs2, imm。

当 $vm=0$ 时, 指令被 mask, 对应汇编指令 vadd.vi, vs2, imm, v0.t。

指令格式:

31	26	25	24	20	19	15	14	12	11	7	6	0
000000			vm	vs2		imm		011		vd		1010111

14.7.8 VADD.VV——矢量整型加法指令

语法:

vadd.vv vd, vs2, vs1, vm

操作:

$vd[i] \leftarrow vs2[i] + vs1[i]$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

无

说明:

当 vm=1 时, 指令未被 mask, 对应汇编指令 vadd.vv vd, vs2, vs1。

当 vm=0 时, 指令被 mask, 对应汇编指令 vadd.vv vd, vs2, vs1,v0.t

指令格式:

31	26	25	24	20	19	15	14	12	11	7	6	0
000000	vm		vs2		vs1		000		vd			1010111

14.7.9 VADD.VX——矢量标量整型加法指令

语法:

vadd.vx vd, vs2, rs1, vm

操作:

$vd[i] \leftarrow vs2[i] + rs1$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

无

说明:

当 vm=1 时, 指令未被 mask, 对应汇编指令 vadd.vx vd, vs2, rs1。

当 vm=0 时, 指令被 mask, 对应汇编指令 vadd.vx vd, vs2, rs1,v0.t

指令格式:

31	26	25	24	20	19	15	14	12	11	7	6	0
000000	vm		vs2		rs1		100		vd			1010111

14.7.10 VAND.VI——矢量立即数按位与指令

语法:

vand.vi vd, vs2, imm, vm

操作:

$vd[i] \leftarrow vs2[i] \& \text{sign_extend}(imm)$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

无

说明:

当 vm=1 时, 指令未被 mask, 对应汇编指令 vand.vi vd, vs2, imm。

当 vm=0 时, 指令被 mask, 对应汇编指令 vand.vi vd, vs2, imm, v0.t。

指令格式:

31	26	25	24	20	19	15	14	12	11	7	6	0
001001			vm	vs2		imm		011		vd		1010111

14.7.11 VAND.VV——矢量按位与指令

语法:

vand.vv vd, vs2, vs1, vm

操作:

$vd[i] \leftarrow vs2[i] \& vs1[i]$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

无

说明:

当 $vm=1$ 时, 指令未被 mask, 对应汇编指令 `vand.vv vd, vs2, vs1`。

当 $vm=0$ 时, 指令被 mask, 对应汇编指令 `vand.vv vd, vs2, vs1, v0.t`。

指令格式:

31	26	25	24	20	19	15	14	12	11	7	6	0
001001	vm		vs2		vs1		000		vd			1010111

14.7.12 VAND.VX——矢量标量按位与指令

语法:

`vand.vx vd, vs2, rs1, vm`

操作:

$vd[i] \leftarrow vs2[i] \& rs1$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

无

说明:

当 $vm=1$ 时, 指令未被 mask, 对应汇编指令 `vand.vx vd, vs2, rs1`。

当 $vm=0$ 时, 指令被 mask, 对应汇编指令 `vand.vx vd, vs2, rs1, v0.t`。

指令格式:

31	26	25	24	20	19	15	14	12	11	7	6	0
001001	vm		vs2		rs1		100		vd			1010111

14.7.13 VASUB.VV——矢量整型减法取平均数指令

语法:

vasub.vv vd, vs2, vs1, vm

操作:

$vd[i] \leftarrow (vs2[i] - vs1[i] + \text{round}) \gg 1$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

无

说明:

根据定点舍入模式寄存器 vxrm 确定舍入模式:

- 2' b00: 就近向大值舍入。
- 2' b01: 就近向偶数舍入。
- 2' b10: 向零舍入。
- 2' b11: 向奇数舍入。

当 vm=1 时, 指令未被 mask, 对应汇编指令 vasub.vv vd, vs2, vs1。

当 vm=0 时, 指令被 mask, 对应汇编指令 vasub.vv vd, vs2, vs1,v0.t

指令格式:

31	26 25 24	20 19	15 14	12 11	7 6	0
100110	vm	vs2	vs1	000	vd	1010111

14.7.14 VASUB.VX——矢量标量整型减法取平均数指令

语法:

vasub.vx vd, vs2, rs1, vm

操作:

$vd[i] \leftarrow (vs2[i] - rs1 + \text{round}) \gg 1$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

无

说明:

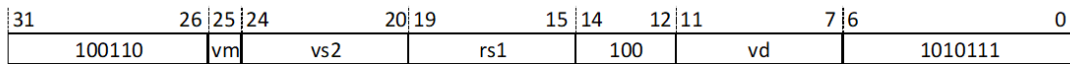
根据定点舍入模式寄存器 vxrm 确定舍入模式:

- 2' b00: 就近向大值舍入。
- 2' b01: 就近向偶数舍入。
- 2' b10: 向零舍入。
- 2' b11: 向奇数舍入。

当 vm=1 时, 指令未被 mask, 对应汇编指令 vasub.vx vd, vs2, rs1。

当 vm=0 时, 指令被 mask, 对应汇编指令 vasub.vx vd, vs2, rs1,v0.t

指令格式:



14.7.15 VCOMPRESS.VM——矢量整型元素压缩指令

语法:

vcompress.vm vd, vs2, vs1

操作:

```

j=0, vd[i]=0
for(i=0; i<v1; i=i+1){
    if(vs1[i]==1)
        vd[j] = vs2[i]
    j=j++
}
    
```

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

无

指令格式:

31	26	25	24	20	19	15	14	12	11	7	6	0
010111		1	vs2		vs1		010		vd		1010111	

14.7.16 VDIV.VV——矢量整型有符号除法指令

语法:

vdiv.vv vd, vs2, vs1, vm

操作:

$vd[i] \leftarrow \text{signed}(vs2[i]) / \text{signed}(vs1[i])$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

无

说明:

当 vm=1 时, 指令未被 mask, 对应汇编指令 vdiv.vv vd, vs2, vs1。

当 vm=0 时, 指令被 mask, 对应汇编指令 vdiv.vv vd, vs2, vs1,v0.t

指令格式:

31	26	25	24	20	19	15	14	12	11	7	6	0
100001		vm	vs2		vs1		010		vd		1010111	

14.7.17 VDIV.VX——矢量标量整型有符号除法指令

语法:

vdiv.vx vd, vs2, rs1, vm

操作:

$vd[i] \leftarrow \text{signed}(vs2[i]) / \text{signed}(rs1)$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

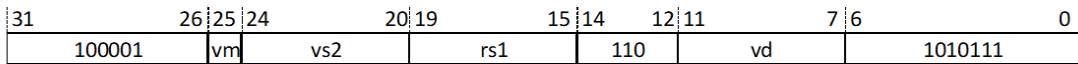
无

说明:

当 vm=1 时, 指令未被 mask, 对应汇编指令 vdiv.vx vd, vs2, rs1。

当 vm=0 时, 指令被 mask, 对应汇编指令 vdiv.vx vd, vs2, rs1,v0.t

指令格式:



14.7.18 VDIVU.VV——矢量整型无符号除法指令

语法:

vdivu.vv vd, vs2, vs1, vm

操作:

$vd[i] \leftarrow \text{unsigned}(vs2[i]) / \text{unsigned}(vs1[i])$

执行权限:

M mode/S mode/U mode

异常:

无

影响标志位:

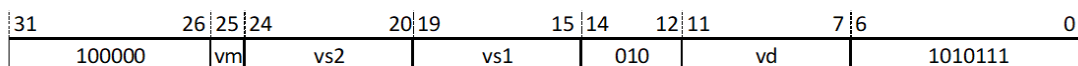
无

说明:

当 vm=1 时, 指令未被 mask, 对应汇编指令 vdivu.vv vd, vs2, vs1。

当 vm=0 时, 指令被 mask, 对应汇编指令 vdivu.vv vd, vs2, vs1,v0.t。

指令格式:



14.7.19 VDIVU.VX——矢量标量整型无符号除法指令

语法:

`vdivu.vx vd, vs2, rs1, vm`

操作:

$vd[i] \leftarrow \text{unsigned}(vs2[i]) / \text{unsigned}(rs1)$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

无

说明:

当 $vm=1$ 时, 指令未被 mask, 对应汇编指令 `vdivu.vx vd, vs2, rs1`。

当 $vm=0$ 时, 指令被 mask, 对应汇编指令 `vdivu.vx vd, vs2, rs1, v0.t`。

指令格式:

31	26	25	24	20	19	15	14	12	11	7	6	0
100000			vm	vs2		rs1		110		vd		1010111

14.7.20 VEXT.X.V——矢量整型提取元素指令

语法:

`vext.x.v rd, vs2, rs1`

操作:

if ($\text{unsigned}(rs1) \geq \text{VLEN}/\text{SEW}$)

$rd = 0$

else

$rd = vs2[\text{unsigned}(rs1)]$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

无

指令格式:

31	26	25	24	20	19	15	14	12	11	7	6	0
001100			1	vs2			rs1		010	vd		1010111

14.7.21 VFADD.VF——矢量标量浮点加法指令

语法:

vfadd.vf vd, vs2, fs1, vm

操作:

$rd[i] \leftarrow vs2[i] + fs1$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

浮点状态位 NV/OF/NX

说明:

根据浮点控制寄存器 fcsr 中的 rm 位确定舍入模式:

- 3' b000: 就近向偶数舍入。
- 3' b001: 向零舍入。
- 3' b010: 向负无穷舍入。
- 3' b011: 向正无穷舍入。
- 3' b100: 就近向大值舍入。
- 3' b101: 非法值, 当配置为该值时, 执行 vfadd.vf 产生非法指令异常。
- 3' b110: 非法值, 当配置为该值时, 执行 vfadd.vf 产生非法指令异常。
- 3' b111: 非法值, 当配置为该值时, 执行 vfadd.vf 产生非法指令异常。

当 vm=1 时, 指令未被 mask, 对应汇编指令 vfadd.vf vd, vs2, fs1。

当 vm=0 时, 指令被 mask, 对应汇编指令 vfadd.vf vd, vs2, fs1,v0.t。

指令格式:

31	26	25	24	20	19	15	14	12	11	7	6	0
000000			vm	vs2			fs1		101	vd		1010111

14.7.22 VFADD.VV——矢量浮点加法指令

语法:

`vfadd.vv vd, vs2, vs1, vm`

操作:

$rd[i] \leftarrow vs2[i] + vs1[i]$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

浮点状态位 NV/OF/NX

说明:

根据浮点控制寄存器 `fscr` 中的 `rm` 位确定舍入模式:

- 3' b000: 就近向偶数舍入。
- 3' b001: 向零舍入。
- 3' b010: 向负无穷舍入。
- 3' b011: 向正无穷舍入。
- 3' b100: 就近向大值舍入。
- 3' b101: 非法值, 当配置为该值时, 执行 `vfadd.vv` 产生非法指令异常。
- 3' b110: 非法值, 当配置为该值时, 执行 `vfadd.vv` 产生非法指令异常。
- 3' b111: 非法值, 当配置为该值时, 执行 `vfadd.vv` 产生非法指令异常。

当 `vm=1` 时, 指令未被 `mask`, 对应汇编指令 `vfadd.vv vd, vs2, vs1`。

当 `vm=0` 时, 指令被 `mask`, 对应汇编指令 `vfadd.vv vd, vs2, vs1,v0.t`。

指令格式:

31	26	25	24	20	19	15	14	12	11	7	6	0
000000			vm	vs2		vs1		001		vd		1010111

14.7.23 VFCLASS.V——矢量浮点分类指令

语法: `v`

`vfclass.v vd, vs2, vm`

操作:

```

if ( vs2[i] = -inf)
    vd[i] ← 64' h1
if ( vs2[i] = -norm)
    vd[i] ← 64' h2
if (vs2[i] = -subnorm)
    vd[i] ← 64' h4
if (vs2[i] = -zero)
    vd[i] ← 64' h8
if ( vs2[i] = +zero)
    vd[i] ← 64' h10
if (vs2[i] = +subnorm)
    vd[i] ← 64' h20
if ( vs2[i] = +norm)
    vd[i] ← 64' h40
if ( vs2[i] = +inf)
    vd[i] ← 64' h80
if ( vs2[i] = sNaN)
    vd[i] ← 64' h100
if ( vs2[i] = qNaN)
    vd[i] ← 64' h200
    
```

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

无

说明:

当 vm=1 时, 指令未被 mask, 对应汇编指令 vfclass.v vd, vs2。
 当 vm=0 时, 指令被 mask, 对应汇编指令 vfclass.v vd, vs2,v0.t。

指令格式:

31	26	25	24	20	19	15	14	12	11	7	6	0
100011		vm	vs2		10000		001	vd		1010111		

14.7.24 VFCVT.F.X.V——矢量有符号整型转换成浮点指令

语法:

vfcvt.f.x.v vd, vs2, vm

操作:

vd [i] ← integer_convert_to_float(vs2[i])

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

浮点状态位 NX/OF

说明:

根据浮点控制寄存器 fcsr 中的 rm 位确定舍入模式:

- 3' b000: 就近向偶数舍入。
- 3' b001: 向零舍入。
- 3' b010: 向负无穷舍入。
- 3' b011: 向正无穷舍入。
- 3' b100: 就近向大值舍入。
- 3' b101: 非法值, 当配置为该值时, 执行 vfcvt.f.x.v 产生非法指令异常。
- 3' b110: 非法值, 当配置为该值时, 执行 vfcvt.f.x.v 产生非法指令异常。
- 3' b111: 非法值, 当配置为该值时, 执行 vfcvt.f.x.v 产生非法指令异常。

当 vm=1 时, 指令未被 mask, 对应汇编指令 vfcvt.f.x.v vd, vs2。

当 vm=0 时, 指令被 mask, 对应汇编指令 vfcvt.f.x.v vd, vs2,v0.t

指令格式:

31	26	25	24	20	19	15	14	12	11	7	6	0
100010			vm	vs2		00011		001		vd		1010111

14.7.25 VFCVT.F.XU.V——矢量无符号整型转换成浮点指令

语法:

vfcvt.f.xu.v vd, vs2, vm

操作:

$vd[i] \leftarrow \text{unsigned_integer_convert_to_float}(vs2[i])$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

浮点状态位 NX/OF

说明:

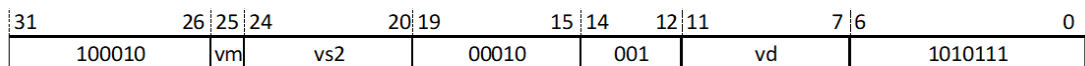
根据浮点控制寄存器 fcsr 中的 rm 位确定舍入模式:

- 3' b000: 就近向偶数舍入。
- 3' b001: 向零舍入。
- 3' b010: 向负无穷舍入。
- 3' b011: 向正无穷舍入。
- 3' b100: 就近向大值舍入。
- 3' b101: 非法值, 当配置为该值时, 执行 `vfcvt.f.xu.v` 产生非法指令异常。
- 3' b110: 非法值, 当配置为该值时, 执行 `vfcvt.f.xu.v` 产生非法指令异常。
- 3' b111: 非法值, 当配置为该值时, 执行 `vfcvt.f.xu.v` 产生非法指令异常。

当 `vm=1` 时, 指令未被 mask, 对应汇编指令 `vfcvt.f.xu.v vd, vs2`。

当 `vm=0` 时, 指令被 mask, 对应汇编指令 `vfcvt.f.xu.v vd, vs2,v0.t`

指令格式:



14.7.26 VFCVT.X.F.V——矢量浮点转换成有符号整型指令

语法:

`vfcvt.x.f.v vd, vs2, vm`

操作:

$vd[i] \leftarrow \text{float_convert_to_signed_integer}(vs2[i])$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

浮点状态位 NV/NX

说明:

根据浮点控制寄存器 fcsr 中的 rm 位确定舍入模式:

- 3' b000: 就近向偶数舍入。
- 3' b001: 向零舍入。
- 3' b010: 向负无穷舍入。
- 3' b011: 向正无穷舍入。
- 3' b100: 就近向大值舍入。
- 3' b101: 非法值, 当配置为该值时, 执行 vfcvt.x.f.v 产生非法指令异常。
- 3' b110: 非法值, 当配置为该值时, 执行 vfcvt.x.f.v 产生非法指令异常。
- 3' b111: 非法值, 当配置为该值时, 执行 vfcvt.x.f.v 产生非法指令异常。

当 vm=1 时, 指令未被 mask, 对应汇编指令 vfcvt.x.f.v vd, vs2。

当 vm=0 时, 指令被 mask, 对应汇编指令 vfcvt.x.f.v vd, vs2,v0.t。

指令格式:

31		26	25	24		20	19		15	14	12	11		7	6		0
100010		vm		vs2		00001		001		vd		1010111					

14.7.27 VFCVT.XU.F.V——矢量浮点转换成无符号整型指令

语法:

vfcvt.xu.f.v vd, vs2, vm

操作:

vd [i] ← float_convert_to_unsigned_integer(vs2[i])

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

浮点状态位 NV/NX

说明:

根据浮点控制寄存器 fcsr 中的 rm 位确定舍入模式。

- 3' b000: 就近向偶数舍入。
- 3' b001: 向零舍入。
- 3' b010: 向负无穷舍入。
- 3' b011: 向正无穷舍入。
- 3' b100: 就近向大值舍入。
- 3' b101: 非法值, 当配置为该值时, 执行 vfcvt.xu.f.v 产生非法指令异常。
- 3' b110: 非法值, 当配置为该值时, 执行 vfcvt.xu.f.v 产生非法指令异常。
- 3' b111: 非法值, 当配置为该值时, 执行 vfcvt.xu.f.v 产生非法指令异常。

当 vm=1 时, 指令未被 mask, 对应汇编指令 vfcvt.xu.f.v vd, vs2。

当 vm=0 时, 指令被 mask, 对应汇编指令 vfcvt.xu.f.v vd, vs2,v0.t

指令格式:

31	26	25	24	20	19	15	14	12	11	7	6	0
100010			vm	vs2		00000		001		vd		1010111

14.7.28 VFDIV.VF——矢量标量浮点除法指令

语法:

vfdiv.vf vd, vs2, fs1, vm

操作:

$vd[i] \leftarrow vs2[i] / fs1$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

浮点状态位 NV/DZ/OF/UF/NX

说明:

根据浮点控制寄存器 fcsr 中的 rm 位确定舍入模式:

- 3' b000: 就近向偶数舍入。
- 3' b001: 向零舍入。
- 3' b010: 向负无穷舍入。
- 3' b011: 向正无穷舍入。
- 3' b100: 就近向大值舍入。

- 3' b101: 非法值, 当配置为该值时, 执行 vfddiv.vf 产生非法指令异常。
- 3' b110: 非法值, 当配置为该值时, 执行 vfddiv.vf 产生非法指令异常。
- 3' b111: 非法值, 当配置为该值时, 执行 vfddiv.vf 产生非法指令异常。

当 vm=1 时, 指令未被 mask, 对应汇编指令 vfddiv.vf vd, vs2, fs1。

当 vm=0 时, 指令被 mask, 对应汇编指令 vfddiv.vf vd, vs2, fs1,v0.t。

指令格式:

31	26	25	24	20	19	15	14	12	11	7	6	0
100001			vm	vs2		fs1		101		vd		1010111

14.7.29 VFDIV.VV——矢量浮点除法指令

语法:

vfddiv.vv vd, vs2, vs1, vm

操作:

$rd[i] \leftarrow vs2[i] / vs1[i]$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

浮点状态位 NV/DZ/OF/UF/NX

说明:

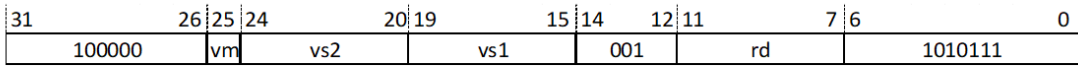
根据浮点控制寄存器 fcsr 中的 rm 位确定舍入模式:

- 3' b000: 就近向偶数舍入。
- 3' b001: 向零舍入。
- 3' b010: 向负无穷舍入。
- 3' b011: 向正无穷舍入。
- 3' b100: 就近向大值舍入。
- 3' b101: 非法值, 当配置为该值时, 执行 vfddiv.vv 产生非法指令异常。
- 3' b110: 非法值, 当配置为该值时, 执行 vfddiv.vv 产生非法指令异常。
- 3' b111: 非法值, 当配置为该值时, 执行 vfddiv.vv 产生非法指令异常。

当 vm=1 时, 指令未被 mask, 对应汇编指令 vfddiv.vv vd, vs2, vs1。

当 vm=0 时, 指令被 mask, 对应汇编指令 vfddiv.vv vd, vs2, vs1,v0.t。

指令格式:



14.7.30 VFMACC.VF——覆盖加数矢量标量浮点乘累加指令

语法:

vfmacc.vf vd, fs1, vs2, vm

操作:

$vd[i] \leftarrow fs1 * vs2[i] + vd[i]$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

浮点状态位 NV/OF/UF/IX

说明:

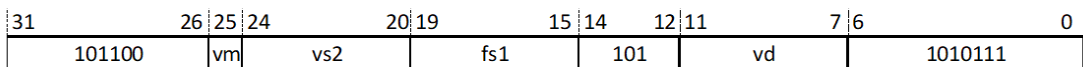
根据浮点控制寄存器 fcsr 中的 rm 位确定舍入模式:

- 3' b000: 就近向偶数舍入。
- 3' b001: 向零舍入。
- 3' b010: 向负无穷舍入。
- 3' b011: 向正无穷舍入。
- 3' b100: 就近向大值舍入。
- 3' b101: 非法值, 当配置为该值时, 执行 vfmacc.vf 产生非法指令异常。
- 3' b110: 非法值, 当配置为该值时, 执行 vfmacc.vf 产生非法指令异常。
- 3' b111: 非法值, 当配置为该值时, 执行 vfmacc.vf 产生非法指令异常。

当 vm=1 时, 指令未被 mask, 对应汇编指令 vfmacc.vf vd, fs1, vs2。

当 vm=0 时, 指令被 mask, 对应汇编指令 vfmacc.vf vd, fs1, vs2, v0.t。

指令格式:



14.7.31 VFMACC.VV——覆盖加数矢量浮点乘累加指令

语法:

`vfmacc.vv vd, vs1, vs2, vm`

操作:

$vd[i] \leftarrow vs1[i] * vs2[i] + vd[i]$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

浮点状态位 NV/OF/UF/IX

说明:

根据浮点控制寄存器 fcsr 中的 `rm` 位确定舍入模式:

- 3' b000: 就近向偶数舍入。
- 3' b001: 向零舍入。
- 3' b010: 向负无穷舍入。
- 3' b011: 向正无穷舍入。
- 3' b100: 就近向大值舍入。
- 3' b101: 非法值, 当配置为该值时, 执行 `vfmacc.vv` 产生非法指令异常。
- 3' b110: 非法值, 当配置为该值时, 执行 `vfmacc.vv` 产生非法指令异常。
- 3' b111: 非法值, 当配置为该值时, 执行 `vfmacc.vv` 产生非法指令异常。

当 `vm=1` 时, 指令未被 `mask`, 对应汇编指令 `vfmacc.vv vd, vs1, vs2`。

当 `vm=0` 时, 指令被 `mask`, 对应汇编指令 `vfmacc.vv vd, vs1, vs2,v0.t`。

指令格式:

31	26	25	24	20	19	15	14	12	11	7	6	0
101100	vm			vs2			vs1	001			rd	1010111

14.7.32 VFMADD.VF——覆盖被乘数矢量标量浮点乘累加指令

语法:

`vfmadd.vf vd, fs1, vs2, vm`

操作:

$$vd[i] \leftarrow fs1 * vd[i] + vs2[i]$$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

浮点状态位 NV/OF/UF/IX

说明:

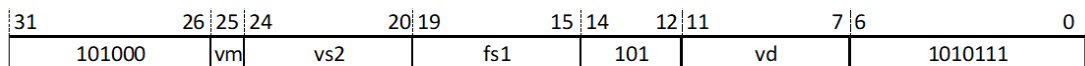
根据浮点控制寄存器 fcsr 中的 rm 位确定舍入模式:

- 3' b000: 就近向偶数舍入。
- 3' b001: 向零舍入。
- 3' b010: 向负无穷舍入。
- 3' b011: 向正无穷舍入。
- 3' b100: 就近向大值舍入。
- 3' b101: 非法值, 当配置为该值时, 执行 vfmadd.vf 产生非法指令异常。
- 3' b110: 非法值, 当配置为该值时, 执行 vfmadd.vf 产生非法指令异常。
- 3' b111: 非法值, 当配置为该值时, 执行 vfmadd.vf 产生非法指令异常。

当 vm=1 时, 指令未被 mask, 对应汇编指令 vfmadd.vf vd, fs1, vs2。

当 vm=0 时, 指令被 mask, 对应汇编指令 vfmadd.vf vd, fs1, vs2,v0.t。

指令格式:



14.7.33 VFMADD.VV——覆盖被乘数矢量浮点乘累加指令

语法:

vfmadd.vv vd, vs1, vs2, vm

操作:

$$vd[i] \leftarrow vs1[i] * vd[i] + vs2[i]$$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

浮点状态位 NV/OF/UF/IX

说明:

根据浮点控制寄存器 fcsr 中的 rm 位确定舍入模式。

- 3' b000: 就近向偶数舍入。
- 3' b001: 向零舍入。
- 3' b010: 向负无穷舍入。
- 3' b011: 向正无穷舍入。
- 3' b100: 就近向大值舍入。
- 3' b101: 非法值, 当配置为该值时, 执行 vfmadd.vv 产生非法指令异常。
- 3' b110: 非法值, 当配置为该值时, 执行 vfmadd.vv 产生非法指令异常。
- 3' b111: 非法值, 当配置为该值时, 执行 vfmadd.vv 产生非法指令异常。

当 vm=1 时, 指令未被 mask, 对应汇编指令 vfmadd.vv vd, vs1, vs2。

当 vm=0 时, 指令被 mask, 对应汇编指令 vfmadd.vv vd, vs1, vs2, v0.t。

指令格式:

31	26	25	24	20	19	15	14	12	11	7	6	0
101000			vm	vs2		vs1		101		vd		1010111

14.7.34 VFMAX.VF——矢量标量浮点取最大值指令

语法:

vfmax.vf vd, vs2, fs1, vm

操作:

```
if(fs1 >= vs2[i])
    vd[i] ← fs1
else
    vd[i] ← vs2[i]
```

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

浮点状态位 NV

说明:

当 $vm=1$ 时, 指令未被 mask, 对应汇编指令 `vfmax.vf vd, vs2, fs1`。

当 $vm=0$ 时, 指令被 mask, 对应汇编指令 `vfmax.vf vd, vs2, fs1, v0.t`。

指令格式:

31	26	25	24	20	19	15	14	12	11	7	6	0
000110	vm			vs2			fs1	101			vd	1010111

14.7.35 VFMAX.VV——矢量浮点取最大值指令

语法:

`vfmax.vv vd, vs2, vs1, vm`

操作: `if(vs1[i] >= vs2[i])`

`vd[i] ← vs1[i]`

else

`vd[i] ← vs2[i]`

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

浮点状态位 NV

说明:

当 $vm=1$ 时, 指令未被 mask, 对应汇编指令 `vfmax.vv vd, vs2, vs1`。

当 $vm=0$ 时, 指令被 mask, 对应汇编指令 `vfmax.vv vd, vs2, vs1, v0.t`。

指令格式:

31	26	25	24	20	19	15	14	12	11	7	6	0
000110	vm			vs2			vs1	001			vd	1010111

14.7.36 VFMERGE.VFM——矢量浮点元素选择指令

语法: `vfmerge.vfm vd, vs2, fs1, v0`

操作: `if(v0[i].LSB == 1)`
`vd[i] ← fs1`
 else
`vd[i] ← vs2[i]`

执行权限:

M mode/S mode/U mode

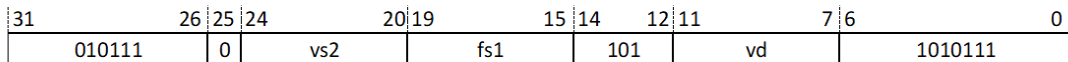
异常:

非法指令异常

影响标志位:

无

指令格式:



14.7.37 VFMIN.VF——矢量标量浮点取最小值指令

语法:

`vfmin.vf vd, vs2, fs1, vm`

操作:

`if(fs1 <= vs2[i])`
`vd[i] ← fs1`
 else
`vd[i] ← vs2[i]`

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

浮点状态位 NV

说明:

当 $vm=1$ 时, 指令未被 mask, 对应汇编指令 `vfmin.vf vd, vs2, fs1`。
 当 $vm=0$ 时, 指令被 mask, 对应汇编指令 `vfmin.vf vd, vs2, fs1, v0.t`。

指令格式:

31	26	25	24	20	19	15	14	12	11	7	6	0
000100	vm		vs2			fs1		101		vd		1010111

14.7.38 VFMIN.VV——矢量浮点取最小值指令

语法:

`vfmin.vv vd, vs2, vs1, vm`

操作:

```

if(vs1[i] <= vs2[i])
    vd[i] ← vs1[i]
else
    vd[i] ← vs2[i]
    
```

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

浮点状态位 NV

说明:

当 $vm=1$ 时, 指令未被 mask, 对应汇编指令 `vfmin.vv vd, vs2, vs1`。
 当 $vm=0$ 时, 指令被 mask, 对应汇编指令 `vfmin.vv vd, vs2, vs1, v0.t`。

指令格式:

31	26	25	24	20	19	15	14	12	11	7	6	0
000100	vm		vs2			vs1		001		vd		1010111

14.7.39 VFMSAC.VF——覆盖减数矢量标量浮点乘累减指令

语法:

vfmsac.vf vd, fs1, vs2, vm

操作:

$vd[i] \leftarrow fs1 * vs2[i] - vd[i]$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

浮点状态位 NV/OF/UF/IX

说明:

根据浮点控制寄存器 fcsr 中的 rm 位确定舍入模式:

- 3' b000: 就近向偶数舍入。
- 3' b001: 向零舍入。
- 3' b010: 向负无穷舍入。
- 3' b011: 向正无穷舍入。
- 3' b100: 就近向大值舍入。
- 3' b101: 非法值, 当配置为该值时, 执行 vfmsac.vf 产生非法指令异常。
- 3' b110: 非法值, 当配置为该值时, 执行 vfmsac.vf 产生非法指令异常。
- 3' b111: 非法值, 当配置为该值时, 执行 vfmsac.vf 产生非法指令异常。

当 vm=1 时, 指令未被 mask, 对应汇编指令 vfmsac.vf vd, fs1, vs2。

当 vm=0 时, 指令被 mask, 对应汇编指令 vfmsac.vf vd, fs1, vs2,v0.t。

指令格式:

31		26	25	24		20	19		15	14	12	11		7	6		0
	101110		vm		vs2			fs1		101		vd				1010111	

14.7.40 VFMSAC.VV——覆盖减数矢量浮点乘累减指令

语法:

vfmsac.vv vd, vs1, vs2, vm

操作:

$vd[i] \leftarrow vs1[i] * vs2[i] - vd[i]$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

浮点状态位 NV/OF/UF/IX

说明:

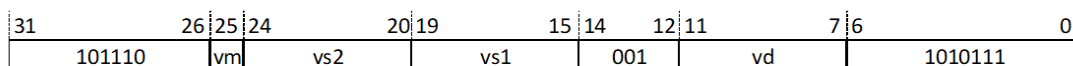
根据浮点控制寄存器 fcsr 中的 rm 位确定舍入模式。

- 3' b000: 就近向偶数舍入。
- 3' b001: 向零舍入。
- 3' b010: 向负无穷舍入。
- 3' b011: 向正无穷舍入。
- 3' b100: 就近向大值舍入。
- 3' b101: 非法值, 当配置为该值时, 执行 vfmsac.vv 产生非法指令异常。
- 3' b110: 非法值, 当配置为该值时, 执行 vfmsac.vv 产生非法指令异常。
- 3' b111: 非法值, 当配置为该值时, 执行 vfmsac.vv 产生非法指令异常。

当 vm=1 时, 指令未被 mask, 对应汇编指令 vfmsac.vv vd, vs1, vs2。

当 vm=0 时, 指令被 mask, 对应汇编指令 vfmsac.vv vd, vs1, vs2,v0.t。

指令格式:



14.7.41 VFMSUB.VF——覆盖被乘数矢量标量浮点乘累减指令

语法:

vfmsub.vf vd, fs1, vs2, vm

操作:

$vd[i] \leftarrow fs1 * vd[i] - vs2[i]$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

浮点状态位 NV/OF/UF/IX

说明:

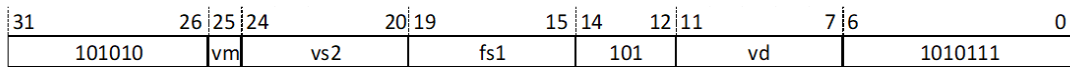
根据浮点控制寄存器 fcsr 中的 rm 位确定舍入模式。

- 3' b000: 就近向偶数舍入。
- 3' b001: 向零舍入。
- 3' b010: 向负无穷舍入。
- 3' b011: 向正无穷舍入。
- 3' b100: 就近向大值舍入。
- 3' b101: 非法值, 当配置为该值时, 执行 vfmsub.vf 产生非法指令异常。
- 3' b110: 非法值, 当配置为该值时, 执行 vfmsub.vf 产生非法指令异常。
- 3' b111: 非法值, 当配置为该值时, 执行 vfmsub.vf 产生非法指令异常。

当 vm=1 时, 指令未被 mask, 对应汇编指令 vfmsub.vf vd, fs1, vs2。

当 vm=0 时, 指令被 mask, 对应汇编指令 vfmsub.vf vd, fs1, vs2,v0.t。

指令格式:



14.7.42 VFMSUB.VV——覆盖被乘数矢量浮点乘累减指令

语法:

vfmsub.vv vd, vs1, vs2, vm

操作:

$vd[i] \leftarrow (vs1[i] * vd[i]) - vs2[i]$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

浮点状态位 NV/OF/UF/IX

说明:

根据浮点控制寄存器 fcsr 中的 rm 位确定舍入模式。

- 3' b000: 就近向偶数舍入。
- 3' b001: 向零舍入。
- 3' b010: 向负无穷舍入。
- 3' b011: 向正无穷舍入。
- 3' b100: 就近向大值舍入。
- 3' b101: 非法值, 当配置为该值时, 执行 vfmsub.vv 产生非法指令异常。
- 3' b110: 非法值, 当配置为该值时, 执行 vfmsub.vv 产生非法指令异常。
- 3' b111: 非法值, 当配置为该值时, 执行 vfmsub.vv 产生非法指令异常。

当 vm=1 时, 指令未被 mask, 对应汇编指令 vfmsub.vv vd, vs1, vs2。

当 vm=0 时, 指令被 mask, 对应汇编指令 vfmsub.vv vd, vs1, vs2,v0.t。

指令格式:

31	26	25	24	20	19	15	14	12	11	7	6	0
101010		vm	vs2	vs1		001		vd		1010111		

14.7.43 VFMUL.VF——矢量标量浮点乘法指令

语法:

vfmul.vf vd, vs2, fs1, vm

操作:

$vd[i] \leftarrow vs2[i] * fs1$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

浮点状态位 NV/OF/UF/IX

说明: 根据浮点控制寄存器 fcsr 中的 rm 位确定舍入模式。

- 3' b000: 就近向偶数舍入。
- 3' b001: 向零舍入。
- 3' b010: 向负无穷舍入。
- 3' b011: 向正无穷舍入。
- 3' b100: 就近向大值舍入。
- 3' b101: 非法值, 当配置为该值时, 执行 vfmul.vf 产生非法指令异常。

- 3' b110: 非法值, 当配置为该值时, 执行 vfmul.vf 产生非法指令异常。
- 3' b111: 非法值, 当配置为该值时, 执行 vfmul.vf 产生非法指令异常。

当 vm=1 时, 指令未被 mask, 对应汇编指令 vfmul.vf vd, vs2, fs1。

当 vm=0 时, 指令被 mask, 对应汇编指令 vfmul.vf vd, vs2, fs1,v0.t。

指令格式:

31	26	25	24	20	19	15	14	12	11	7	6	0
100100		vm	vs2		fs1		101		vd		1010111	

14.7.44 VFMUL.VV——矢量浮点乘法指令

语法:

vfmul.vv vd, vs2, vs1, vm

操作:

$vd[i] \leftarrow vs2[i] * vs1[i]$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

浮点状态位 NV/OF/UF/IX

说明:

根据浮点控制寄存器 fcsr 中的 rm 位确定舍入模式。

- 3' b000: 就近向偶数舍入。
- 3' b001: 向零舍入。
- 3' b010: 向负无穷舍入。
- 3' b011: 向正无穷舍入。
- 3' b100: 就近向大值舍入。
- 3' b101: 非法值, 当配置为该值时, 执行 vfmul.vv 产生非法指令异常。
- 3' b110: 非法值, 当配置为该值时, 执行 vfmul.vv 产生非法指令异常。
- 3' b111: 非法值, 当配置为该值时, 执行 vfmul.vv 产生非法指令异常。

当 vm=1 时, 指令未被 mask, 对应汇编指令 vfmul.vv vd, vs2, vs1。

当 vm=0 时, 指令被 mask, 对应汇编指令 vfmul.vv vd, vs2, vs1,v0.t。

指令格式:

31	26	25	24	20	19	15	14	12	11	7	6	0
100100		vm	vs2		vs1		001		vd		1010111	

14.7.45 VFMV.F.S——矢量首元素传送至浮点指令

语法:

`vfmv.f.s rd, vs2`

操作:

`rd = vs2[0]`

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

无

指令格式:

31	26	25	24	20	19	15	14	12	11	7	6	0
001100	1	vs2		00000		001		rd		1010111		

14.7.46 VFMV.S.F——浮点传送至矢量首元素指令

语法:

`vfmv.s.f vd, fs1`

操作:

`vd[0] = fs1`

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

无

指令格式:

31	26	25	24	20	19	15	14	12	11	7	6	0
001101	1	00000		fs1		101		vd		1010111		

14.7.47 VFMV.V.F——浮点传送至矢量指令

语法:

`vfmv.v.f vd, fs1`

操作:

$vd[i] = fs1$

执行权限:

M mode/S mode/U mode

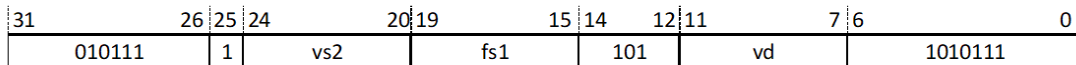
异常:

非法指令异常

影响标志位:

无

指令格式:



14.7.48 VFNCVT.F.F.V——矢量浮点缩位转换指令

语法:

`vfncvt.f.f.v vd, vs2, vm`

操作:

$vd[i] \leftarrow \text{double_width_float_convert_to_float}(vs2[i])$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

无

说明:

vs2 数据宽度为 2*sew, vd 数据宽度为 sew。

根据浮点控制寄存器 fcsr 中的 rm 位确定舍入模式。

- 3' b000: 就近向偶数舍入。
- 3' b001: 向零舍入。
- 3' b010: 向负无穷舍入。
- 3' b011: 向正无穷舍入。
- 3' b100: 就近向大值舍入。
- 3' b101: 非法值, 当配置为该值时, 执行 vfncvt.f.f.v 产生非法指令异常。
- 3' b110: 非法值, 当配置为该值时, 执行 vfncvt.f.f.v 产生非法指令异常。
- 3' b111: 非法值, 当配置为该值时, 执行 vfncvt.f.f.v 产生非法指令异常。

当 vm=1 时, 指令未被 mask, 对应汇编指令 vfncvt.f.f.v vd, vs2。

当 vm=0 时, 指令被 mask, 对应汇编指令 vfncvt.f.f.v vd, vs2,v0.t

指令格式:

31	26	25	24	20	19	15	14	12	11	7	6	0
100010			vm	vs2		10100		001		vd		1010111

14.7.49 VFNCVT.F.XU.V——矢量无符号整型缩位转换成浮点指令

语法:

vfncvt.f.xu.v vd, vs2, vm

操作:

vd [i] ← double_width_unsigned_integer_convert_to_float(vs2[i])

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

浮点状态位 NX/OF

说明:

vs2 数据宽度为 2*sew, vd 数据宽度为 sew。

根据浮点控制寄存器 fcsr 中的 rm 位确定舍入模式。

- 3' b000: 就近向偶数舍入。
- 3' b001: 向零舍入。
- 3' b010: 向负无穷舍入。

- 3' b011: 向正无穷舍入。
- 3' b100: 就近向大值舍入。
- 3' b101: 非法值, 当配置为该值时, 执行 `vfncvt.f.xu.v` 产生非法指令异常。
- 3' b110: 非法值, 当配置为该值时, 执行 `vfncvt.f.xu.v` 产生非法指令异常。
- 3' b111: 非法值, 当配置为该值时, 执行 `vfncvt.f.xu.v` 产生非法指令异常。

当 `vm=1` 时, 指令未被 `mask`, 对应汇编指令 `vfncvt.f.xu.v vd, vs2`。

当 `vm=0` 时, 指令被 `mask`, 对应汇编指令 `vfncvt.f.xu.v vd, vs2, v0.t`。

指令格式:

31	26	25	24	20	19	15	14	12	11	7	6	0	
100010			vm	vs2			10010		001	vd		1010111	

14.7.50 VFNCVT.F.X.V——矢量有符号整型缩位转换成浮点指令

语法:

`vfncvt.f.x.v vd, vs2, vm`

操作:

`vd [i] ← double_width_signed_integer_convert_to_float(vs2[i])`

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

浮点状态位 NX/OF

说明:

`vs2` 数据宽度为 $2 * \text{sew}$, `vd` 数据宽度为 `sew`。

根据浮点控制寄存器 `fcsr` 中的 `rm` 位确定舍入模式。

- 3' b000: 就近向偶数舍入。
- 3' b001: 向零舍入。
- 3' b010: 向负无穷舍入。
- 3' b011: 向正无穷舍入。
- 3' b100: 就近向大值舍入。
- 3' b101: 非法值, 当配置为该值时, 执行 `vfncvt.f.x.v` 产生非法指令异常。
- 3' b110: 非法值, 当配置为该值时, 执行 `vfncvt.f.x.v` 产生非法指令异常。
- 3' b111: 非法值, 当配置为该值时, 执行 `vfncvt.f.x.v` 产生非法指令异常。

当 $vm=1$ 时, 指令未被 mask, 对应汇编指令 `vfncvt.f.x.v vd,vs2`。

当 $vm=0$ 时, 指令被 mask, 对应汇编指令 `vfncvt.f.x.v vd, vs2,v0.t`。

指令格式:

31	26	25	24	20	19	15	14	12	11	7	6	0
100010	vm			vs2			10011	001			vd	1010111

14.7.51 VFNCVT.X.F.V——矢量浮点缩位转换成有符号整型指令

语法:

`vfncvt.x.f.v vd, vs2, vm`

操作:

$vd[i] \leftarrow \text{double_width_float_convert_to_signed_integer}(vs2[i])$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

浮点状态位 NV/NX

说明:

$vs2$ 数据宽度为 $2 * sew$, vd 数据宽度为 sew 。

根据浮点控制寄存器 `fcsr` 中的 `rm` 位确定舍入模式。

- 3' b000: 就近向偶数舍入。
- 3' b001: 向零舍入。
- 3' b010: 向负无穷舍入。
- 3' b011: 向正无穷舍入。
- 3' b100: 就近向大值舍入。
- 3' b101: 非法值, 当配置为该值时, 执行 `vfncvt.x.f.v` 产生非法指令异常。
- 3' b110: 非法值, 当配置为该值时, 执行 `vfncvt.x.f.v` 产生非法指令异常。
- 3' b111: 非法值, 当配置为该值时, 执行 `vfncvt.x.f.v` 产生非法指令异常。

当 $vm=1$ 时, 指令未被 mask, 对应汇编指令 `vfncvt.x.f.v vd, vs2`。

当 $vm=0$ 时, 指令被 mask, 对应汇编指令 `vfncvt.x.f.v vd, vs2,v0.t`。

指令格式:

31	26	25	24	20	19	15	14	12	11	7	6	0
100010	vm			vs2			10001	001			vd	1010111

14.7.52 VFNCVT.XU.F.V——矢量浮点缩位转换成无符号整型指令

语法:

`vfncvt.xu.f.v vd, vs2, vm`

操作:

`vd [i] ← double_width_float_convert_to_unsigned_integer(vs2[i])`

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

浮点状态位 NV/NX

说明:

vs2 数据宽度为 2*sew, vd 数据宽度为 sew。

根据浮点控制寄存器 fcsr 中的 rm 位确定舍入模式。

- 3' b000: 就近向偶数舍入。
- 3' b001: 向零舍入。
- 3' b010: 向负无穷舍入。
- 3' b011: 向正无穷舍入。
- 3' b100: 就近向大值舍入。
- 3' b101: 非法值, 当配置为该值时, 执行 `vfncvt.xu.f.v` 产生非法指令异常。
- 3' b110: 非法值, 当配置为该值时, 执行 `vfncvt.xu.f.v` 产生非法指令异常。
- 3' b111: 非法值, 当配置为该值时, 执行 `vfncvt.xu.f.v` 产生非法指令异常。

当 `vm=1` 时, 指令未被 mask, 对应汇编指令 `vfncvt.xu.f.v vd, vs2`。

当 `vm=0` 时, 指令被 mask, 对应汇编指令 `vfncvt.xu.f.v vd, vs2,v0.t`

指令格式:

31	26 25 24	20 19	15 14	12 11	7 6	0
	100010	vm	vs2	10000	001	vd
						1010111

14.7.53 VFNMACC.VF——覆盖减数矢量标量浮点乘累加取负指令

语法:

`vfnmacc.vf vd, fs1, vs2, vm`

操作:

$$vd[i] \leftarrow -(fs1 * vs2[i]) - vd[i]$$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

浮点状态位 NV/OF/UF/IX

说明:

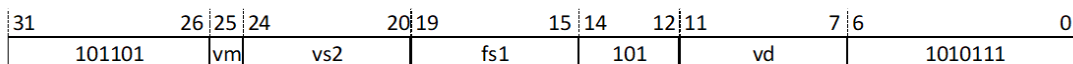
根据浮点控制寄存器 fcsr 中的 rm 位确定舍入模式。

- 3' b000: 就近向偶数舍入。
- 3' b001: 向零舍入。
- 3' b010: 向负无穷舍入。
- 3' b011: 向正无穷舍入。
- 3' b100: 就近向大值舍入。
- 3' b101: 非法值, 当配置为该值时, 执行 vfnmacc.vf 产生非法指令异常。
- 3' b110: 非法值, 当配置为该值时, 执行 vfnmacc.vf 产生非法指令异常。
- 3' b111: 非法值, 当配置为该值时, 执行 vfnmacc.vf 产生非法指令异常。

当 vm=1 时, 指令未被 mask, 对应汇编指令 vfnmacc.vf vd, fs1, vs2。

当 vm=0 时, 指令被 mask, 对应汇编指令 vfnmacc.vf vd, fs1, vs2,v0.t。

指令格式:



14.7.54 VFNMACC.VV——覆盖减数矢量浮点乘累加取负指令

语法:

vfnmacc.vv vd, vs1, vs2, vm

操作:

$$vd[i] \leftarrow -(vs1[i] * vs2[i]) - vd[i]$$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

浮点状态位 NV/OF/UF/IX

说明:

根据浮点控制寄存器 fcsr 中的 rm 位确定舍入模式。

- 3' b000: 就近向偶数舍入。
- 3' b001: 向零舍入。
- 3' b010: 向负无穷舍入。
- 3' b011: 向正无穷舍入。
- 3' b100: 就近向大值舍入。
- 3' b101: 非法值, 当配置为该值时, 执行 vfnmacc.vv 产生非法指令异常。
- 3' b110: 非法值, 当配置为该值时, 执行 vfnmacc.vv 产生非法指令异常。
- 3' b111: 非法值, 当配置为该值时, 执行 vfnmacc.vv 产生非法指令异常。

当 vm=1 时, 指令未被 mask, 对应汇编指令 vfnmacc.vv vd, vs1, vs2。

当 vm=0 时, 指令被 mask, 对应汇编指令 vfnmacc.vv vd, vs1, vs2,v0.t。

指令格式:

31	26	25	24	20	19	15	14	12	11	7	6	0
101101		vm	vs2		vs1		001		vd		1010111	

14.7.55 VFNMADD.VF——覆盖被乘数矢量标量浮点乘累加取负指令

语法:

vfnmadd.vf vd, fs1, vs2, vm

操作:

$vd[i] \leftarrow -(fs1 * vd[i]) - vs2[i]$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

浮点状态位 NV/OF/UF/IX

说明:

根据浮点控制寄存器 fcsr 中的 rm 位确定舍入模式。

- 3' b000: 就近向偶数舍入。
- 3' b001: 向零舍入。
- 3' b010: 向负无穷舍入。
- 3' b011: 向正无穷舍入。
- 3' b100: 就近向大值舍入。
- 3' b101: 非法值, 当配置为该值时, 执行 vfnmadd.vf 产生非法指令异常。
- 3' b110: 非法值, 当配置为该值时, 执行 vfnmadd.vf 产生非法指令异常。
- 3' b111: 非法值, 当配置为该值时, 执行 vfnmadd.vf 产生非法指令异常。

当 vm=1 时, 指令未被 mask, 对应汇编指令 vfnmadd.vf vd, fs1, vs2。

当 vm=0 时, 指令被 mask, 对应汇编指令 vfnmadd.vf vd, fs1, vs2,v0.t。

指令格式:

31	26	25	24	20	19	15	14	12	11	7	6	0
101001			vm	vs2		fs1		101		rd		1010111

14.7.56 VFNMADD.VV——覆盖被乘数矢量浮点乘累加取负指令

语法:

vfnmadd.vv vd, vs1, vs2, vm

操作:

$vd[i] \leftarrow -(vs1[i] * vd[i]) - vs2[i]$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

浮点状态位 NV/OF/UF/IX

说明:

根据浮点控制寄存器 fcsr 中的 rm 位确定舍入模式。

- 3' b000: 就近向偶数舍入。
- 3' b001: 向零舍入。
- 3' b010: 向负无穷舍入。
- 3' b011: 向正无穷舍入。
- 3' b100: 就近向大值舍入。

- 3' b101: 非法值, 当配置为该值时, 执行 `vfnmadd.vv` 产生非法指令异常。
- 3' b110: 非法值, 当配置为该值时, 执行 `vfnmadd.vv` 产生非法指令异常。
- 3' b111: 非法值, 当配置为该值时, 执行 `vfnmadd.vv` 产生非法指令异常。

当 `vm=1` 时, 指令未被 `mask`, 对应汇编指令 `vfnmadd.vv vd, vs1, vs2`。

当 `vm=0` 时, 指令被 `mask`, 对应汇编指令 `vfnmadd.vv vd, vs1, vs2,v0.t`。

指令格式:

31	26 25 24	20 19	15 14	12 11	7 6	0
101001	vm	vs2	fs1	001	vd	1010111

14.7.57 VFNMSAC.VF——覆盖被减数矢量标量浮点乘累减取负指令

语法:

`vfnmsac.vf vd, fs1, vs2, vm`

操作:

$vd[i] \leftarrow -(fs1 * vs2[i]) + vd[i]$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

浮点状态位 NV/OF/UF/IX

说明:

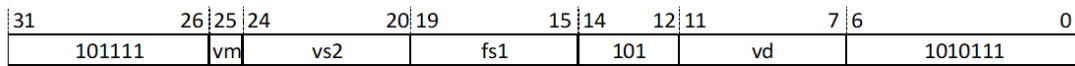
根据浮点控制寄存器 `fcsr` 中的 `rm` 位确定舍入模式。

- 3' b000: 就近向偶数舍入。
- 3' b001: 向零舍入。
- 3' b010: 向负无穷舍入。
- 3' b011: 向正无穷舍入。
- 3' b100: 就近向大值舍入。
- 3' b101: 非法值, 当配置为该值时, 执行 `vfnmsac.vf` 产生非法指令异常。
- 3' b110: 非法值, 当配置为该值时, 执行 `vfnmsac.vf` 产生非法指令异常。
- 3' b111: 非法值, 当配置为该值时, 执行 `vfnmsac.vf` 产生非法指令异常。

当 `vm=1` 时, 指令未被 `mask`, 对应汇编指令 `vfnmsac.vf vd, fs1, vs2`。

当 `vm=0` 时, 指令被 `mask`, 对应汇编指令 `vfnmsac.vf vd, fs1, vs2,v0.t`。

指令格式:



14.7.58 VFNMSAC.VV——覆盖被减数向量浮点乘累减取负指令

语法:

vfmsac.vv vd, vs1, vs2, vm

操作:

$vd[i] \leftarrow -(vs1[i] * vs2[i]) + vd[i]$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

浮点状态位 NV/OF/UF/IX

说明:

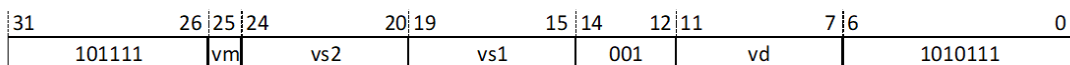
根据浮点控制寄存器 fcsr 中的 rm 位确定舍入模式。

- 3' b000: 就近向偶数舍入。
- 3' b001: 向零舍入。
- 3' b010: 向负无穷舍入。
- 3' b011: 向正无穷舍入。
- 3' b100: 就近向大值舍入。
- 3' b101: 非法值, 当配置为该值时, 执行 vfmsac.vv 产生非法指令异常。
- 3' b110: 非法值, 当配置为该值时, 执行 vfmsac.vv 产生非法指令异常。
- 3' b111: 非法值, 当配置为该值时, 执行 vfmsac.vv 产生非法指令异常。

当 vm=1 时, 指令未被 mask, 对应汇编指令 vfmsac.vv vd, vs1, vs2。

当 vm=0 时, 指令被 mask, 对应汇编指令 vfmsac.vv vd, vs1, vs2,v0.t。

指令格式:



14.7.59 VFNMSUB.VF——覆盖被乘数矢量标量浮点乘累减取负指令

语法:

`vfmsub.vf vd, fs1, vs2, vm`

操作:

$vd[i] \leftarrow -(fs1 * vd[i]) + vs2[i]$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

浮点状态位 NV/OF/UF/IX

说明:

根据浮点控制寄存器 `fcsr` 中的 `rm` 位确定舍入模式。

- 3' b000: 就近向偶数舍入。
- 3' b001: 向零舍入。
- 3' b010: 向负无穷舍入。
- 3' b011: 向正无穷舍入。
- 3' b100: 就近向大值舍入。
- 3' b101: 非法值, 当配置为该值时, 执行 `vfmsub.vf` 产生非法指令异常。
- 3' b110: 非法值, 当配置为该值时, 执行 `vfmsub.vf` 产生非法指令异常。
- 3' b111: 非法值, 当配置为该值时, 执行 `vfmsub.vf` 产生非法指令异常。

当 `vm=1` 时, 指令未被 `mask`, 对应汇编指令 `vfmsub.vf vd, fs1, vs2`。

当 `vm=0` 时, 指令被 `mask`, 对应汇编指令 `vfmsub.vf vd, fs1, vs2,v0.t`。

指令格式:

31	26 25 24	20 19	15 14	12 11	7 6	0
101011	vm	vs2	fs1	101	vd	1010111

14.7.60 VFNMSUB.VV——覆盖被乘数矢量浮点乘累减取负指令

语法:

`vfmsub.vv vd, vs1, vs2, vm`

操作:

$$vd[i] \leftarrow -(vs1[i] * vd[i]) + vs2[i]$$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

浮点状态位 NV/OF/UF/IX

说明:

根据浮点控制寄存器 fcsr 中的 rm 位确定舍入模式。

- 3' b000: 就近向偶数舍入。
- 3' b001: 向零舍入。
- 3' b010: 向负无穷舍入。
- 3' b011: 向正无穷舍入。
- 3' b100: 就近向大值舍入。
- 3' b101: 非法值, 当配置为该值时, 执行 vfnmsub.vv 产生非法指令异常。
- 3' b110: 非法值, 当配置为该值时, 执行 vfnmsub.vv 产生非法指令异常。
- 3' b111: 非法值, 当配置为该值时, 执行 vfnmsub.vv 产生非法指令异常。

当 vm=1 时, 指令未被 mask, 对应汇编指令 vfnmsub.vv vd, vs1, vs2。

当 vm=0 时, 指令被 mask, 对应汇编指令 vfnmsub.vv vd, vs1, vs2,v0.t。

指令格式:

31		26	25	24		20	19		15	14	12	11		7	6		0
101011		vm		vs2			vs1			001		vd		1010111			

14.7.61 VFRDIV.VF——标量矢量浮点除法指令

语法:

vfrdiv.vf vd, vs2, fs1, vm

操作:

$$vd[i] \leftarrow fs1 / vs2[i]$$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

浮点状态位 NV/DZ/OF/UF/NX

说明:

根据浮点控制寄存器 fcsr 中的 rm 位确定舍入模式。

- 3' b000: 就近向偶数舍入。
- 3' b001: 向零舍入。
- 3' b010: 向负无穷舍入。
- 3' b011: 向正无穷舍入。
- 3' b100: 就近向大值舍入。
- 3' b101: 非法值, 当配置为该值时, 执行 vfrdiv.vf 产生非法指令异常。
- 3' b110: 非法值, 当配置为该值时, 执行 vfrdiv.vf 产生非法指令异常。
- 3' b111: 非法值, 当配置为该值时, 执行 vfrdiv.vf 产生非法指令异常。

当 vm=1 时, 指令未被 mask, 对应汇编指令 vfrdiv.vf vd, vs2, fs1。

当 vm=0 时, 指令被 mask, 对应汇编指令 vfrdiv.vf vd, vs2, fs1,v0.t。

指令格式:

31		26	25	24		20	19		15	14	12	11		7	6		0
100001		vm		vs2		fs1		101		vd		1010111					

14.7.62 VFREDMAX.VS —— 矢量浮点元素缩减取最大值指令

语法:

vfredmax.vs. vd, vs2, vs1, vm

操作:

```
tmp = vs1[0]
for( i=0; i<v1; i++) {
    tmp = max(tmp, vs2[i])
}
vd[0]= tmp
vd[VLEN/SEW-1:1] = 0
```

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

浮点状态位 NV

说明:

当 $vm=1$ 时, 指令未被 mask, 对应汇编指令 `vfredmax.vs.vf vd, vs2, vs1`。

当 $vm=0$ 时, 指令被 mask, 对应汇编指令 `vfredmax.vs vd, vs2, vs1, v0.t`。

指令格式:

31	26 25 24	20 19	15 14	12 11	7 6	0
000111	vm	vs2	vs1	001	vd	1010111

14.7.63 VFREDMIN.VS —— 矢量浮点缩减取最小值指令

语法:

`vfredmin.vs. vd, vs2, vs1, vm`

操作:

```
tmp = vs1[0]
for( i=0; i< vl; i++) {
    tmp = min(vs2[i])
}
vd[0]= tmp
vd[VLEN/SEW-1:1] = 0
```

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

浮点状态位 NV

说明:

当 $vm=1$ 时, 指令未被 mask, 对应汇编指令 `vfredmin.vs vd, vs2, vs1`。

当 $vm=0$ 时, 指令被 mask, 对应汇编指令 `vfredmin.vs vd, vs2, vs1, v0.t`。

指令格式:

31	26 25 24	20 19	15 14	12 11	7 6	0
000101	vm	vs2	vs1	001	vd	1010111

14.7.64 VFREDOSUM.VS —— 矢量浮点缩减顺序求和指令

语法:

vfredosum.vs. vd, vs2, vs1, vm

操作:

```
tmp = vs1[0]
for( i=0; i< vl; i++) {
    tmp = tmp +vs2[i]
}
vd[0]= tmp
vd[VLEN/SEW-1:1] = 0
```

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

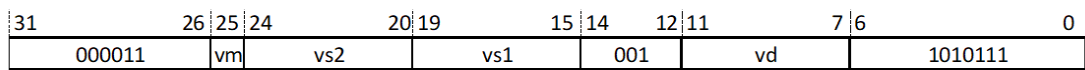
浮点状态位 NV/OF/IX

说明:

当 vm=1 时, 指令未被 mask, 对应汇编指令 vfredosum.vs vd, vs2, vs1。

当 vm=0 时, 指令被 mask, 对应汇编指令 vfredosum.vs vd, vs2, vs1, v0.t。

指令格式:



14.7.65 VFREDSUM.VS —— 矢量浮点缩减乱序求和指令

语法:

vfredsum.vs. vd, vs2, vs1, vm

操作:

```
tmp=unorder_sum(vs2[i])
vd[0]= tmp + vs1[0];
vd[VLEN/SEW-1:1] = 0
```

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

浮点状态位 NV/OF/IX

说明:

当 $vm=1$ 时, 指令未被 mask, 对应汇编指令 `vfredsum.vs vd, vs2, vs1`。

当 $vm=0$ 时, 指令被 mask, 对应汇编指令 `vfredsum.vs vd, vs2, vs1, v0.t`。

指令格式:

31	26	25	24	20	19	15	14	12	11	7	6	0
000001	vm		vs2		vs1			001		vd		1010111

14.7.66 VFRSUB.VF —— 标量矢量浮点减法指令

语法:

`vfrsub.vf. vd, vs2, fs1, vm`

操作:

$vd[i] = fs1 - vs2[i];$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

浮点状态位 NV/DZ/OF/UF/NX

说明: 根据浮点控制寄存器 `fcsr` 中的 `rm` 位确定舍入模式。3' b000: 就近向偶数舍入。3' b001: 向零舍入。3' b010: 向负无穷舍入。3' b011: 向正无穷舍入。3' b100: 就近向大值舍入。3' b101~3' b111: 非法值, 当配置为该值时, 执行 `vfrsub.vf` 产生非法指令异常。

当 $vm=1$ 时, 指令未被 mask, 对应汇编指令 `vfrsub.vf vd, vs2, fs1`。当 $vm=0$ 时, 指令被 mask, 对应汇编指令 `vfrsub.vf vd, vs2, fs1, v0.t`。

指令格式:

31	26	25	24	20	19	15	14	12	11	7	6	0
100111	vm		vs2		fs1		101		vd		1010111	

14.7.67 VFSGNJ.VF——矢量标量浮点符号注入指令

语法:

`vfsgnj.vf vd, vs2, fs1, vm`

操作:

$vd[i][MSB-1:0] \leftarrow vs2[i][MSB-1:0]$

$vd[i][MSB] \leftarrow fs1[MSB]$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

无

说明:

当 $vm=1$ 时, 指令未被 mask, 对应汇编指令 `vfsgnj.vf vd, vs2, fs1`。

当 $vm=0$ 时, 指令被 mask, 对应汇编指令 `vfsgnj.vf vd, vs2, fs1, v0.t`。

指令格式:

31	26	25	24	20	19	15	14	12	11	7	6	0
001000		vm	vs2		fs1		101		vd		1010111	

14.7.68 VFSGNJ.VV——矢量浮点符号注入指令

语法:

`vfsgnj.vv vd, vs2, vs1, vm`

操作:

$vd[i][MSB-1:0] \leftarrow vs2[i][MSB-1:0]$

$vd[i][MSB] \leftarrow vs1[i][MSB]$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

无

说明:

当 $vm=1$ 时, 指令未被 mask, 对应汇编指令 `vfsgnj.vv vd, vs2, vs1`。

当 $vm=0$ 时, 指令被 mask, 对应汇编指令 `vfsgnj.vv vd, vs2, vs1, v0.t`。

指令格式:

31	26 25 24	20 19	15 14	12 11	7 6	0
001000	vm	vs2	vs1	001	vd	1010111

14.7.69 VFSGNJN.VF——矢量标量浮点符号取反注入指令

语法:

`vfsgnjn.vf vd, vs2, fs1,vm`

操作:

$vd[i][MSB-1:0] \leftarrow vs2[i][MSB-1:0]$

$vd[i][MSB] \leftarrow !fs1[MSB]$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

无

说明:

当 $vm=1$ 时, 指令未被 mask, 对应汇编指令 `vfsgnjn.vf vd, vs2, fs1`。

当 $vm=0$ 时, 指令被 mask, 对应汇编指令 `vfsgnjn.vf vd, vs2, fs1, v0.t`。

指令格式:

31	26 25 24	20 19	15 14	12 11	7 6	0
001001	vm	vs2	fs1	101	vd	1010111

14.7.70 VFSGNJN.VV——矢量浮点符号取反注入指令

语法:

`vfsgnfn.vv vd, vs2, vs1,vm`

操作:

$vd[i][MSB-1:0] \leftarrow vs2[i][MSB-1:0]$

$vd[i][MSB] \leftarrow !vs1[i][MSB]$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

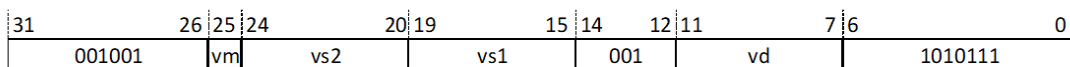
无

说明:

当 $vm=1$ 时, 指令未被 mask, 对应汇编指令 `vfsgnfn.vv vd, vs2, vs1`。

当 $vm=0$ 时, 指令被 mask, 对应汇编指令 `vfsgnfn.vv vd, vs2, vs1, v0.t`。

指令格式:



14.7.71 VFSGNJX.VF——矢量标量浮点符号异或注入指令

语法:

`vfsgnfx.vf vd, vs2, fs1,vm`

操作:

$vd[i][MSB-1:0] \leftarrow vs2[i][MSB-1:0]$

$vd[i][MSB] \leftarrow fs1[MSB] \wedge vs2[i][MSB]$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

无

说明:

当 vm=1 时, 指令未被 mask, 对应汇编指令 vfsgnjx.vf vd, vs2, fs1。

当 vm=0 时, 指令被 mask, 对应汇编指令 vfsgnjx.vf vd, vs2, fs1, v0.t。

指令格式:

31	26	25	24	20	19	15	14	12	11	7	6	0
001010	vm			vs2			fs1			vd		
												1010111

14.7.72 VFSGNJX.VV——矢量浮点符号异或注入指令

语法:

vfsgnjx.vv vd, vs2, vs1,vm

操作: $vd[i][MSB-1:0] \leftarrow vs2[i][MSB-1:0]$

$vd[i][MSB] \leftarrow vs1[i][MSB] \wedge vs2[i][MSB]$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

无

说明:

当 vm=1 时, 指令未被 mask, 对应汇编指令 vfsgnjx.vv vd, vs2, vs1。

当 vm=0 时, 指令被 mask, 对应汇编指令 vfsgnjx.vv vd, vs2, vs1, v0.t。

指令格式:

31	26	25	24	20	19	15	14	12	11	7	6	0	
001010	vm			vs2			vs1			001	vd		
												1010111	

14.7.73 VFSQRT.V——矢量浮点开方指令

语法:

`vfsqrt.v vd, vs2, vm`

操作:

$vd[i] \leftarrow \text{sqrt}(vs2[i])$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

浮点状态位 NV/NX

说明:

根据浮点控制寄存器 `fcsr` 中的 `rm` 位确定舍入模式。

- 3' b000: 就近向偶数舍入。
- 3' b001: 向零舍入。
- 3' b010: 向负无穷舍入。
- 3' b011: 向正无穷舍入。
- 3' b100: 就近向大值舍入。
- 3' b101: 非法值, 当配置为该值时, 执行 `vfsqrt.v` 产生非法指令异常。
- 3' b110: 非法值, 当配置为该值时, 执行 `vfsqrt.v` 产生非法指令异常。
- 3' b111: 非法值, 当配置为该值时, 执行 `vfsqrt.v` 产生非法指令异常。

当 `vm=1` 时, 指令未被 `mask`, 对应汇编指令 `vfsqrt.v vd, vs2`。

当 `vm=0` 时, 指令被 `mask`, 对应汇编指令 `vfsqrt.v vd, vs2,v0.t`。

指令格式:

31	26	25	24	20	19	15	14	12	11	7	6	0
100011	vm		vs2	00000		001		vd		1010111		

14.7.74 VFSUB.VF——矢量标量浮点减法指令

语法:

`vfsub.vf vd, vs2, fs1, vm`

操作:

$vd[i] \leftarrow vs2[i] - fs1$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

浮点状态位 NV/OF/NX

说明:

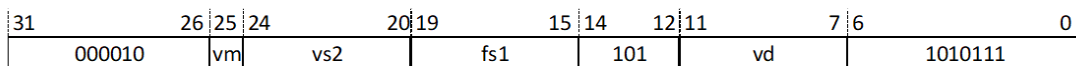
根据浮点控制寄存器 fcsr 中的 rm 位确定舍入模式。

- 3' b000: 就近向偶数舍入。
- 3' b001: 向零舍入。
- 3' b010: 向负无穷舍入。
- 3' b011: 向正无穷舍入。
- 3' b100: 就近向大值舍入。
- 3' b101: 非法值, 当配置为该值时, 执行 vsub.vf 产生非法指令异常。
- 3' b110: 非法值, 当配置为该值时, 执行 vsub.vf 产生非法指令异常。
- 3' b111: 非法值, 当配置为该值时, 执行 vsub.vf 产生非法指令异常。

当 vm=1 时, 指令未被 mask, 对应汇编指令 vsub.vf vd, vs2, fs1。

当 vm=0 时, 指令被 mask, 对应汇编指令 vsub.vf vd, vs2, fs1,v0.t。

指令格式:



14.7.75 VFSUB.VV——矢量浮点减法指令

语法:

vsub.vv vd, vs2, vs1, vm

操作:

$rd[i] \leftarrow vs2[i] - vs1[i]$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

浮点状态位 NV/OF/NX

说明:

根据浮点控制寄存器 fcsr 中的 rm 位确定舍入模式。

- 3' b000: 就近向偶数舍入。
- 3' b001: 向零舍入。
- 3' b010: 向负无穷舍入。
- 3' b011: 向正无穷舍入。
- 3' b100: 就近向大值舍入。
- 3' b101: 非法值, 当配置为该值时, 执行 vsub.vf 产生非法指令异常。
- 3' b110: 非法值, 当配置为该值时, 执行 vsub.vf 产生非法指令异常。
- 3' b111: 非法值, 当配置为该值时, 执行 vsub.vf 产生非法指令异常。

当 vm=1 时, 指令未被 mask, 对应汇编指令 vsub.vv vd, vs2, vs1。

当 vm=0 时, 指令被 mask, 对应汇编指令 vsub.vv vd, vs2, vs1,v0.t。

指令格式:

31		26	25	24		20	19		15	14	12	11		7	6		0
000010		vm		vs2		vs1		001		vd		1010111					

14.7.76 VFWADD.VF——向量标量浮点扩位加法指令

语法:

vfwadd.vf vd, vs2, fs1, vm

操作:

$vd[i] \leftarrow vs2[i] + fs1$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

浮点状态位 NV

说明:

vd 数据宽度为 2*sew, vs2 和 fs1 数据宽度为 sew。

根据浮点控制寄存器 fcsr 中的 rm 位确定舍入模式。

- 3' b000: 就近向偶数舍入。
- 3' b001: 向零舍入。
- 3' b010: 向负无穷舍入。
- 3' b011: 向正无穷舍入。
- 3' b100: 就近向大值舍入。
- 3' b101: 非法值, 当配置为该值时, 执行 vfwadd.vf 产生非法指令异常。
- 3' b110: 非法值, 当配置为该值时, 执行 vfwadd.vf 产生非法指令异常。
- 3' b111: 非法值, 当配置为该值时, 执行 vfwadd.vf 产生非法指令异常。

当 vm=1 时, 指令未被 mask, 对应汇编指令 vfwadd.vf vd, vs2, fs1。

当 vm=0 时, 指令被 mask, 对应汇编指令 vfwadd.vf vd, vs2, fs1,v0.t。

指令格式:

31	26	25	24	20	19	15	14	12	11	7	6	0
110000		vm	vs2	fs1		101		vd		1010111		

14.7.77 VFWADD.VV——矢量浮点扩位加法指令

语法:

vfwadd.vv vd, vs2, vs1, vm

操作:

$vd[i] \leftarrow vs2[i] + vs1[i]$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

浮点状态位 NV

说明:

vd 数据宽度为 2*sew, vs2 和 vs1 数据宽度为 sew。

浮点控制寄存器 fcsr 中的 rm 位确定舍入模式。

- 3' b000: 就近向偶数舍入。
- 3' b001: 向零舍入。
- 3' b010: 向负无穷舍入。

- 3' b011: 向正无穷舍入。
- 3' b100: 就近向大值舍入。
- 3' b101: 非法值, 当配置为该值时, 执行 vfwadd.vv 产生非法指令异常。
- 3' b110: 非法值, 当配置为该值时, 执行 vfwadd.vv 产生非法指令异常。
- 3' b111: 非法值, 当配置为该值时, 执行 vfwadd.vv 产生非法指令异常。

当 vm=1 时, 指令未被 mask, 对应汇编指令 vfwadd.vv vd, vs2, vs1。

当 vm=0 时, 指令被 mask, 对应汇编指令 vfwadd.vv vd, vs2, vs1,v0.t。

指令格式:

31	26 25 24	20 19	15 14	12 11	7 6	0
110000	vm	vs2	vs1	001	vd	1010111

14.7.78 VFWADD.WF——扩位矢量标量浮点扩位加法指令

语法:

vfwadd.wf vd, vs2, fs1, vm

操作:

$vd[i] \leftarrow vs2[i] + fs1$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

浮点状态位 NV/OF/IX

说明:

vd 和 vs2 数据宽度为 2*sew, fs1 数据宽度为 sew。

根据浮点控制寄存器 fcsr 中的 rm 位确定舍入模式。

- 3' b000: 就近向偶数舍入。
- 3' b001: 向零舍入。
- 3' b010: 向负无穷舍入。
- 3' b011: 向正无穷舍入。
- 3' b100: 就近向大值舍入。
- 3' b101: 非法值, 当配置为该值时, 执行 vfwadd.wf 产生非法指令异常。
- 3' b110: 非法值, 当配置为该值时, 执行 vfwadd.wf 产生非法指令异常。
- 3' b111: 非法值, 当配置为该值时, 执行 vfwadd.wf 产生非法指令异常。

当 $vm=1$ 时, 指令未被 mask, 对应汇编指令 `vfwadd.wf vd, vs2, fs1`。

当 $vm=0$ 时, 指令被 mask, 对应汇编指令 `vfwadd.wf vd, vs2, fs1, v0.t`。

指令格式:

31	26	25	24	20	19	15	14	12	11	7	6	0
110100	vm			vs2		fs1		101		vd		1010111

14.7.79 VFWADD.WV——扩位矢量浮点扩位加法指令

语法:

`vfwadd.wv vd, vs2, vs1, vm`

操作:

$vd[i] \leftarrow vs2[i] + vs1[i]$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

浮点状态位 NV/OF/IX

说明:

`vd` 和 `vs2` 数据宽度为 $2 * sew$, `vs1` 数据宽度为 `sew`。

根据浮点控制寄存器 `fcsr` 中的 `rm` 位确定舍入模式。

- 3' b000: 就近向偶数舍入。
- 3' b001: 向零舍入。
- 3' b010: 向负无穷舍入。
- 3' b011: 向正无穷舍入。
- 3' b100: 就近向大值舍入。
- 3' b101: 非法值, 当配置为该值时, 执行 `vfwadd.wv` 产生非法指令异常。
- 3' b110: 非法值, 当配置为该值时, 执行 `vfwadd.wv` 产生非法指令异常。
- 3' b111: 非法值, 当配置为该值时, 执行 `vfwadd.wv` 产生非法指令异常。

当 $vm=1$ 时, 指令未被 mask, 对应汇编指令 `vfwadd.wv vd, vs2, vs1`。

当 $vm=0$ 时, 指令被 mask, 对应汇编指令 `vfwadd.wv vd, vs2, vs1, v0.t`。

指令格式:

31	26	25	24	20	19	15	14	12	11	7	6	0
110100	vm			vs2		fs1		001		vd		1010111

14.7.80 VFWCVT.F.F.V——矢量浮点扩位转换指令

语法:

`vfwcvt.f.f.v vd, vs2, vm`

操作:

$vd[i] \leftarrow \text{float_convert_to_double_width_float}(vs2[i])$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

无

说明:

vd 数据宽度为 $2 * sew$, vs2 数据宽度为 sew 。

指令格式:

31	26	25	24	20	19	15	14	12	11	7	6	0
100010			vm	vs2		01100		001		vd		1010111

14.7.81 VFWCVT.F.X.V——矢量有符号整型扩位转换成浮点指令

语法:

`vfwcvt.f.x.v vd, vs2, vm`

操作:

$vd[i] \leftarrow \text{signed_integer_convert_to_double_width_float}(vs2[i])$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

无

说明:

vd 数据宽度为 2*sew, vs2 数据宽度为 sew。

根据浮点控制寄存器 fcsr 中的 rm 位确定舍入模式。

- 3' b000: 就近向偶数舍入。
- 3' b001: 向零舍入。
- 3' b010: 向负无穷舍入。
- 3' b011: 向正无穷舍入。
- 3' b100: 就近向大值舍入。
- 3' b101: 非法值, 当配置为该值时, 执行 vfwcvt.f.x.v 产生非法指令异常。
- 3' b110: 非法值, 当配置为该值时, 执行 vfwcvt.f.x.v 产生非法指令异常。
- 3' b111: 非法值, 当配置为该值时, 执行 vfwcvt.f.x.v 产生非法指令异常。

当 vm=1 时, 指令未被 mask, 对应汇编指令 vfwcvt.f.x.v vd, vs2。

当 vm=0 时, 指令被 mask, 对应汇编指令 vfwcvt.f.x.v vd, vs2,v0.t

指令格式:

31		26	25	24		20	19		15	14	12	11		7	6		0	
			vm		vs2			01011		001			vd					

14.7.82 VFWCVT.F.XU.V——矢量无符号整型扩位转换成浮点指令

语法:

vfwcvt.f.xu.v vd, vs2, vm

操作:

vd [i] ← unsigned_integer_convert_to_double_width_float(vs2[i])

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

无

说明:

vd 数据宽度为 2*sew, vs2 数据宽度为 sew。

根据浮点控制寄存器 fcsr 中的 rm 位确定舍入模式。

- 3' b000: 就近向偶数舍入。

- 3' b001: 向零舍入。
- 3' b010: 向负无穷舍入。
- 3' b011: 向正无穷舍入。
- 3' b100: 就近向大值舍入。
- 3' b101: 非法值, 当配置为该值时, 执行 vfwcvt.f.xu.v 产生非法指令异常。
- 3' b110: 非法值, 当配置为该值时, 执行 vfwcvt.f.xu.v 产生非法指令异常。
- 3' b111: 非法值, 当配置为该值时, 执行 vfwcvt.f.xu.v 产生非法指令异常。

当 vm=1 时, 指令未被 mask, 对应汇编指令 vfwcvt.f.xu.v vd, vs2。

当 vm=0 时, 指令被 mask, 对应汇编指令 vfwcvt.f.xu.v vd, vs2,v0.t

指令格式:

31	26	25	24	20	19	15	14	12	11	7	6	0
100010	vm		vs2	01010		001		vd		1010111		

14.7.83 VFWCVT.X.F.V——矢量浮点扩位转换成有符号整型指令

语法:

vfwcvt.x.f.v vd, vs2, vm

操作:

vd [i] ← float_convert_to_double_width_signed_integer(vs2[i])

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

浮点状态位 NV/NX

说明:

vd 数据宽度为 2*sew, vs2 数据宽度为 sew。

根据浮点控制寄存器 fcsr 中的 rm 位确定舍入模式。

- 3' b000: 就近向偶数舍入。
- 3' b001: 向零舍入。
- 3' b010: 向负无穷舍入。
- 3' b011: 向正无穷舍入。
- 3' b100: 就近向大值舍入。
- 3' b101: 非法值, 当配置为该值时, 执行 vfwcvt.x.f.v 产生非法指令异常。

- 3' b110: 非法值, 当配置为该值时, 执行 vfwcvt.x.f.v 产生非法指令异常。
- 3' b111: 非法值, 当配置为该值时, 执行 vfwcvt.x.f.v 产生非法指令异常。

当 vm=1 时, 指令未被 mask, 对应汇编指令 vfwcvt.x.f.v vd, vs2。

当 vm=0 时, 指令被 mask, 对应汇编指令 vfwcvt.x.f.v vd, vs2,v0.t

指令格式:

31	26 25 24	20 19	15 14	12 11	7 6	0
100010	vm	vs2	01001	001	vd	1010111

14.7.84 VFWCVT.XU.F.V——矢量浮点扩位转换成无符号整型指令

语法:

vfwcvt.xu.f.v vd, vs2, vm

操作:

vd [i] ← float_convert_to_double_width_unsigned_integer(vs2[i])

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

浮点状态位 NV/NX

说明:

vd 数据宽度为 2*sew, vs2 数据宽度为 sew。

根据浮点控制寄存器 fcsr 中的 rm 位确定舍入模式。

- 3' b000: 就近向偶数舍入。
- 3' b001: 向零舍入。
- 3' b010: 向负无穷舍入。
- 3' b011: 向正无穷舍入。
- 3' b100: 就近向大值舍入。
- 3' b101: 非法值, 当配置为该值时, 执行 vfwcvt.xu.f.v 产生非法指令异常。
- 3' b110: 非法值, 当配置为该值时, 执行 vfwcvt.xu.f.v 产生非法指令异常。
- 3' b111: 非法值, 当配置为该值时, 执行 vfwcvt.xu.f.v 产生非法指令异常。

当 vm=1 时, 指令未被 mask, 对应汇编指令 vfwcvt.xu.f.v vd, vs2。

当 vm=0 时, 指令被 mask, 对应汇编指令 vfwcvt.xu.f.v vd, vs2,v0.t

指令格式:

31	26	25	24	20	19	15	14	12	11	7	6	0
100010		vm	vs2		01000		001		vd		1010111	

14.7.85 VFWMACC.VF——覆盖加数矢量标量浮点扩位乘累加指令

语法:

`vfwmaccc.vf vd, fs1, vs2, vm`

操作:

$vd[i] \leftarrow fs1 * vs2[i] + vd[i]$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

浮点状态位 NV/OF/IX

说明:

vd 数据宽度为 $2 * sew$, fs1 和 vs2 数据宽度为 sew 。

根据浮点控制寄存器 fcsr 中的 rm 位确定舍入模式。

- 3' b000: 就近向偶数舍入。
- 3' b001: 向零舍入。
- 3' b010: 向负无穷舍入。
- 3' b011: 向正无穷舍入。
- 3' b100: 就近向大值舍入。
- 3' b101: 非法值, 当配置为该值时, 执行 `vfwmaccc.vf` 产生非法指令异常。
- 3' b110: 非法值, 当配置为该值时, 执行 `vfwmaccc.vf` 产生非法指令异常。
- 3' b111: 非法值, 当配置为该值时, 执行 `vfwmaccc.vf` 产生非法指令异常。

当 $vm=1$ 时, 指令未被 mask, 对应汇编指令 `vfwmaccc.vf vd, fs1, vs2`。

当 $vm=0$ 时, 指令被 mask, 对应汇编指令 `vfwmaccc.vf vd, fs1, vs2, v0.t`。

指令格式:

31	26	25	24	20	19	15	14	12	11	7	6	0
111100		vm	vs2		fs1		101		vd		1010111	

14.7.86 VFWMACC.VV——覆盖加数矢量浮点扩位乘累加指令

语法:

vfwmacc.vv vd, fs1, vs2, vm

操作:

$vd[i] \leftarrow vs1[i] * vs2[i] + vd[i]$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

浮点状态位 NV/OF/IX

说明:

vd 数据宽度为 2*sew, vs1 和 vs2 数据宽度为 sew。

根据浮点控制寄存器 fcsr 中的 rm 位确定舍入模式。

- 3' b000: 就近向偶数舍入。
- 3' b001: 向零舍入。
- 3' b010: 向负无穷舍入。
- 3' b011: 向正无穷舍入。
- 3' b100: 就近向大值舍入。
- 3' b101: 非法值, 当配置为该值时, 执行 vfwmacc.vv 产生非法指令异常。
- 3' b110: 非法值, 当配置为该值时, 执行 vfwmacc.vv 产生非法指令异常。
- 3' b111: 非法值, 当配置为该值时, 执行 vfwmacc.vv 产生非法指令异常。

当 vm=1 时, 指令未被 mask, 对应汇编指令 vfwmacc.vv vd, vs1, vs2,。

当 vm=0 时, 指令被 mask, 对应汇编指令 vfwmacc.vv vd, vs1, vs2, v0.t。

指令格式:

31	26 25 24	20 19	15 14	12 11	7 6	0
111100	vm	vs2	vs1	001	vd	1010111

14.7.87 VFWMSAC.VF——覆盖加数矢量浮点扩位乘累减指令

语法:

vfwmsac.vf vd, fs1, vs2, vm

操作:

$vd[i] \leftarrow fs1 * vs2[i] - vd[i]$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

浮点状态位 NV/OF/IX

说明:

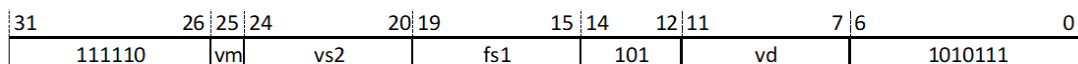
vd 数据宽度为 2*sew, fs1 和 vs2 数据宽度为 sew
根据浮点控制寄存器 fcsr 中的 rm 位确定舍入模式。

- 3' b000: 就近向偶数舍入。
- 3' b001: 向零舍入。
- 3' b010: 向负无穷舍入。
- 3' b011: 向正无穷舍入。
- 3' b100: 就近向大值舍入。
- 3' b101: 非法值, 当配置为该值时, 执行 vfwmsac.vf 产生非法指令异常。
- 3' b110: 非法值, 当配置为该值时, 执行 vfwmsac.vf 产生非法指令异常。
- 3' b111: 非法值, 当配置为该值时, 执行 vfwmsac.vf 产生非法指令异常。

当 vm=1 时, 指令未被 mask, 对应汇编指令 vfwmsac.vf vd, fs1, vs2。

当 vm=0 时, 指令被 mask, 对应汇编指令 vfwmsac.vf vd, fs1, vs2,v0.t。

指令格式:



14.7.88 VFWMSAC.VV——覆盖加数矢量浮点扩位乘累减指令

语法:

vfwmsac.vv vd, vs1, vs2, vm

操作:

$vd[i] \leftarrow vs1[i] * vs2[i] - vd[i]$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

浮点状态位 NV/OF/IX

说明:

vd 数据宽度为 2*sew, vs1 和 vs2 数据宽度为 sew。

根据浮点控制寄存器 fcsr 中的 rm 位确定舍入模式。

- 3' b000: 就近向偶数舍入。
- 3' b001: 向零舍入。
- 3' b010: 向负无穷舍入。
- 3' b011: 向正无穷舍入。
- 3' b100: 就近向大值舍入。
- 3' b101: 非法值, 当配置为该值时, 执行 vfwmsac.vv 产生非法指令异常。
- 3' b110: 非法值, 当配置为该值时, 执行 vfwmsac.vv 产生非法指令异常。
- 3' b111: 非法值, 当配置为该值时, 执行 vfwmsac.vv 产生非法指令异常。

当 vm=1 时, 指令未被 mask, 对应汇编指令 vfwmsac.vv vd, vs1, vs2。

当 vm=0 时, 指令被 mask, 对应汇编指令 vfwmsac.vv vd, vs1, vs2,v0.t。

指令格式:

31		26	25	24		20	19		15	14	12	11		7	6		0
111110		vm		vs2			vs1			001		vd		1010111			

14.7.89 VFWMUL.VF——向量标量扩位浮点乘法指令

语法:

vfmul.vf vd, vs2, fs1, vm

操作:

$vd[i] \leftarrow vs2[i] * fs1$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

浮点状态位 NV

说明:

vd 数据宽度为 $2 * sew$, fs1 和 vs2 数据宽度为 sew 。
 根据浮点控制寄存器 fcsr 中的 rm 位确定舍入模式。

- 3' b000: 就近向偶数舍入。
- 3' b001: 向零舍入。
- 3' b010: 向负无穷舍入。
- 3' b011: 向正无穷舍入。
- 3' b100: 就近向大值舍入。
- 3' b101: 非法值, 当配置为该值时, 执行 vfwmul.vf 产生非法指令异常。
- 3' b110: 非法值, 当配置为该值时, 执行 vfwmul.vf 产生非法指令异常。
- 3' b111: 非法值, 当配置为该值时, 执行 vfwmul.vf 产生非法指令异常。

当 $vm=1$ 时, 指令未被 mask, 对应汇编指令 vfwmul.vf vd, vs2, fs1。

当 $vm=0$ 时, 指令被 mask, 对应汇编指令 vfwmul.vf vd, vs2, fs1, v0.t。

指令格式:

31	26	25	24	20	19	15	14	12	11	7	6	0
111000		vm	vs2		fs1		101		vd		1010111	

14.7.90 VFWMUL.VV——矢量浮点扩位乘法指令

语法:

vfwmul.vv vd, vs2, vs1, vm

操作:

$vd[i] \leftarrow vs2[i] * vs1[i]$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

浮点状态位 NV

说明:

vd 数据宽度为 $2 * sew$, vs1 和 vs2 数据宽度为 sew 。
 根据浮点控制寄存器 fcsr 中的 rm 位确定舍入模式。

- 3' b000: 就近向偶数舍入。
- 3' b001: 向零舍入。
- 3' b010: 向负无穷舍入。

- 3' b011: 向正无穷舍入。
- 3' b100: 就近向大值舍入。
- 3' b101: 非法值, 当配置为该值时, 执行 vfwmul.vv 产生非法指令异常。
- 3' b110: 非法值, 当配置为该值时, 执行 vfwmul.vv 产生非法指令异常。
- 3' b111: 非法值, 当配置为该值时, 执行 vfwmul.vv 产生非法指令异常。

当 vm=1 时, 指令未被 mask, 对应汇编指令 vfwmul.vv vd, vs2, vs1。

当 vm=0 时, 指令被 mask, 对应汇编指令 vfwmul.vv vd, vs2, vs1,v0.t。

指令格式:

31	26 25 24	20 19	15 14	12 11	7 6	0
111000	vm	vs2	vs1	001	vd	1010111

14.7.91 VFWNMACC.VF——覆盖加数矢量标量浮点扩位乘累加取负指令

语法:

vfwnmacc.vf vd, fs1, vs2, vm

操作:

$vd[i] \leftarrow -(fs1 * vs2[i]) - vd[i]$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

浮点状态位 NV/OF/IX

说明:

vd 数据宽度为 2*sew, fs1 和 vs2 数据宽度为 sew。

根据浮点控制寄存器 fcsr 中的 rm 位确定舍入模式。

- 3' b000: 就近向偶数舍入。
- 3' b001: 向零舍入。
- 3' b010: 向负无穷舍入。
- 3' b011: 向正无穷舍入。
- 3' b100: 就近向大值舍入。
- 3' b101: 非法值, 当配置为该值时, 执行 vfwnmacc.vf 产生非法指令异常。
- 3' b110: 非法值, 当配置为该值时, 执行 vfwnmacc.vf 产生非法指令异常。
- 3' b111: 非法值, 当配置为该值时, 执行 vfwnmacc.vf 产生非法指令异常。

当 $vm=1$ 时，指令未被 mask，对应汇编指令 `vfwnmacc.vf vd, fs1, vs2`。

当 $vm=0$ 时，指令被 mask，对应汇编指令 `vfwnmacc.vf vd, fs1, vs2, v0.t`。

指令格式：

31	26 25 24	20 19	15 14	12 11	7 6	0
111101	vm	vs2	fs1	101	vd	1010111

14.7.92 VFWNMACC.VV——覆盖加数矢量浮点扩位乘累加取负指令

语法：

`vfwnmacc.vv vd, vs1, vs2, vm`

操作：

$vd[i] \leftarrow -(vs1[i] * vs2[i]) - vd[i]$

执行权限：

M mode/S mode/U mode

异常：

非法指令异常

影响标志位：

浮点状态位 NV/OF/IX

说明：

`vd` 数据宽度为 $2 * sew$ ，`vs1` 和 `vs2` 数据宽度为 `sew`。

根据浮点控制寄存器 `fcsr` 中的 `rm` 位确定舍入模式。

- 3' b000: 就近向偶数舍入。
- 3' b001: 向零舍入。
- 3' b010: 向负无穷舍入。
- 3' b011: 向正无穷舍入。
- 3' b100: 就近向大值舍入。
- 3' b101: 非法值，当配置为该值时，执行 `vfwnmacc.vv` 产生非法指令异常。
- 3' b110: 非法值，当配置为该值时，执行 `vfwnmacc.vv` 产生非法指令异常。
- 3' b111: 非法值，当配置为该值时，执行 `vfwnmacc.vv` 产生非法指令异常。

当 $vm=1$ 时，指令未被 mask，对应汇编指令 `vfwnmacc.vv vd, vs1, vs2`。

当 $vm=0$ 时，指令被 mask，对应汇编指令 `vfwnmacc.vv vd, vs1, vs2, v0.t`。

指令格式：

31	26 25 24	20 19	15 14	12 11	7 6	0
111101	vm	vs2	vs1	001	vd	1010111

14.7.93 VFWNMSAC.VF——覆盖加数矢量标量浮点扩位乘累减取负指令

语法:

`vfnwnmsac.vf vd, fs1, vs2, vm`

操作:

$vd[i] \leftarrow -(fs1 * vs2[i]) + vd[i]$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

浮点状态位 NV/OF/IX

说明:

vd 数据宽度为 $2 * sew$, fs1 和 vs2 数据宽度为 sew 。

根据浮点控制寄存器 fcsr 中的 rm 位确定舍入模式。

- 3' b000: 就近向偶数舍入。
- 3' b001: 向零舍入。
- 3' b010: 向负无穷舍入。
- 3' b011: 向正无穷舍入。
- 3' b100: 就近向大值舍入。
- 3' b101: 非法值, 当配置为该值时, 执行 `vfnwnmsac.vv` 产生非法指令异常。
- 3' b110: 非法值, 当配置为该值时, 执行 `vfnwnmsac.vf` 产生非法指令异常。
- 3' b111: 非法值, 当配置为该值时, 执行 `vfnwnmsac.vf` 产生非法指令异常。

当 $vm=1$ 时, 指令未被 mask, 对应汇编指令 `vfnwnmsac.vf vd, fs1, vs2`。

当 $vm=0$ 时, 指令被 mask, 对应汇编指令 `vfnwnmsac.vf vd, fs1, vs2, v0.t`。

指令格式:

31	26	25	24	20	19	15	14	12	11	7	6	0
111111			vm	vs2		fs1		101		vd		1010111

14.7.94 VFWNMSAC.VV——覆盖加数矢量浮点扩位乘累减取负指令

语法:

`vfnwnmsac.vv. vd, vs1, vs2, vm`

操作:

$$vd[i] \leftarrow -(vs1[i] * vs2[i]) + vd[i]$$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

浮点状态位 NV/OF/IX

说明:

vd 数据宽度为 2*sew, vs1 和 vs2 数据宽度为 sew。

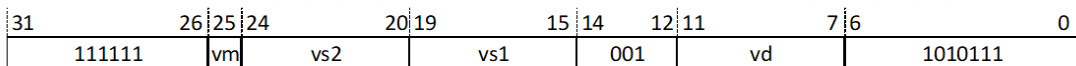
根据浮点控制寄存器 fcsr 中的 rm 位确定舍入模式。

- 3' b000: 就近向偶数舍入。
- 3' b001: 向零舍入。
- 3' b010: 向负无穷舍入。
- 3' b011: 向正无穷舍入。
- 3' b100: 就近向大值舍入。
- 3' b101: 非法值, 当配置为该值时, 执行 vfwmsac.vv 产生非法指令异常。
- 3' b110: 非法值, 当配置为该值时, 执行 vfwmsac.vv 产生非法指令异常。
- 3' b111: 非法值, 当配置为该值时, 执行 vfwmsac.vv 产生非法指令异常。

当 vm=1 时, 指令未被 mask, 对应汇编指令 vfwmsac.vv vd, vs1, vs2。

当 vm=0 时, 指令被 mask, 对应汇编指令 vfwmsac.vv vd, vs1, vs2,v0.t。

指令格式:



14.7.95 VFWREDOSUM.VS —— 矢量扩位浮点缩减顺序求和指令

语法:

vfwredosum.vs. vd, vs2, vs1, vm

操作:

```
tmp = vs1[0]
for( i=0; i< vl; i++) {
    tmp = tmp + widen_to_2sew(vs2[i])
}
vd[0]= tmp
vd[VLEN/SEW-1:1] = 0
```

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

浮点状态位 NV/OF/IX

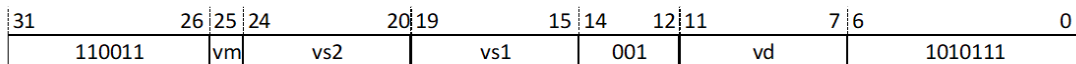
说明:

vd 和 vs1 数据宽度为 2*sew, vs2 数据宽度为 sew, vs2 元素被扩位至 2*sew 后进行计算。

当 vm=1 时, 指令未被 mask, 对应汇编指令 vfwredosum.vs vd, vs2, vs1。

当 vm=0 时, 指令被 mask, 对应汇编指令 vfwredosum.vs vd, vs2, vs1, v0.t。

指令格式:



14.7.96 VFWREDSUM.VS —— 矢量扩位浮点缩减乱序求和指令

语法:

vfwredsum.vs. vd, vs2, vs1, vm

操作:

tmp = unordered_sum(widen_to_2sew(vs2[i]))

vd[0]= vs1[0] + tmp

vd[VLEN/SEW-1:1] = 0

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

浮点状态位 NV/OF/IX

说明:

vd 和 vs1 数据宽度为 2*sew, vs2 数据宽度为 sew。vs2 元素被扩位至 2*sew 后进行计算。

当 vm=1 时, 指令未被 mask, 对应汇编指令 vfwredsum.vs vd, vs2, vs1。

当 vm=0 时, 指令被 mask, 对应汇编指令 vfwredsum.vs vd, vs2, vs1, v0.t。

指令格式:

31	26	25	24	20	19	15	14	12	11	7	6	0
110001			vm	vs2	vs1	001		vd	1010111			

14.7.97 VFWSUB.VF——矢量标量浮点扩位减法指令

语法:

vfwsb.vf vd, vs2, fs1, vm

操作:

vd[i] ← vs2[i] - fs1

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

浮点状态位 NV

说明:

vd 数据宽度为 2*sew, vs2 和 fs1 数据宽度为 sew。

根据浮点控制寄存器 fcsr 中的 rm 位确定舍入模式。

- 3' b000: 就近向偶数舍入。
- 3' b001: 向零舍入。
- 3' b010: 向负无穷舍入。
- 3' b011: 向正无穷舍入。
- 3' b100: 就近向大值舍入。
- 3' b101: 非法值, 当配置为该值时, 执行 vfwsb.vf 产生非法指令异常。
- 3' b110: 非法值, 当配置为该值时, 执行 vfwsb.vf 产生非法指令异常。
- 3' b111: 非法值, 当配置为该值时, 执行 vfwsb.vf 产生非法指令异常。

当 vm=1 时, 指令未被 mask, 对应汇编指令 vfwsb.vf vd, vs2, fs1。

当 vm=0 时, 指令被 mask, 对应汇编指令 vfwsb.vf vd, vs2, fs1, v0.t。

指令格式:

31	26	25	24	20	19	15	14	12	11	7	6	0
110010			vm	vs2	fs1	101		vd	1010111			

14.7.98 VFWSUB.VV——矢量浮点扩位减法指令

语法:

`vfwsb.vv vd, vs2, vs1, vm`

操作:

$vd[i] \leftarrow vs2[i] - vs1[i]$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

浮点状态位 NV

说明:

vd 数据宽度为 $2 * sew$, vs2 和 vs1 数据宽度为 sew 。

根据浮点控制寄存器 fcsr 中的 rm 位确定舍入模式。

- 3' b000: 就近向偶数舍入。
- 3' b001: 向零舍入。
- 3' b010: 向负无穷舍入。
- 3' b011: 向正无穷舍入。
- 3' b100: 就近向大值舍入。
- 3' b101: 非法值, 当配置为该值时, 执行 `vfwsb.vf` 产生非法指令异常。
- 3' b110: 非法值, 当配置为该值时, 执行 `vfwsb.vf` 产生非法指令异常。
- 3' b111: 非法值, 当配置为该值时, 执行 `vfwsb.vf` 产生非法指令异常。

当 $vm=1$ 时, 指令未被 mask, 对应汇编指令 `vfwsb.vv vd, vs2, vs1`。

当 $vm=0$ 时, 指令被 mask, 对应汇编指令 `vfwsb.vv vd, vs2, vs1, v0.t`。

指令格式:

31		26	25	24		20	19		15	14	12	11		7	6		0
110010		vm		vs2		vs1		001		vd		1010111					

14.7.99 VFWSUB.WF——扩位矢量标量浮点扩位减法指令

语法:

`vfwsb.wf vd, vs2, fs1, vm`

操作:

$vd[i] \leftarrow vs2[i] - fs1$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

浮点状态位 NV/OF

说明:

vd 和 vs2 数据宽度为 $2 * sew$, fs1 数据宽度为 sew 。

根据浮点控制寄存器 fcsr 中的 rm 位确定舍入模式。

- 3' b000: 就近向偶数舍入。
- 3' b001: 向零舍入。
- 3' b010: 向负无穷舍入。
- 3' b011: 向正无穷舍入。
- 3' b100: 就近向大值舍入。
- 3' b101: 非法值, 当配置为该值时, 执行 vfwsb.wf 产生非法指令异常。
- 3' b110: 非法值, 当配置为该值时, 执行 vfwsb.wf 产生非法指令异常。
- 3' b111: 非法值, 当配置为该值时, 执行 vfwsb.wf 产生非法指令异常。

当 $vm=1$ 时, 指令未被 mask, 对应汇编指令 vfwsb.wf vd, vs2, fs1。

当 $vm=0$ 时, 指令被 mask, 对应汇编指令 vfwsb.wf vd, vs2, fs1, v0.t。

指令格式:

31	26	25	24	20	19	15	14	12	11	7	6	0
110110		vm	vs2			rs1		101		vd		1010111

14.7.100 VFWSUB.WV——扩位矢量浮点扩位减法指令

语法:

vfwsb.wv vd, vs2, vs1, vm

操作:

$vd[i] \leftarrow vs2[i] - vs1[i]$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

浮点状态位 NV/OF

说明:

vd 和 vs2 数据宽度为 2*sew, vs1 数据宽度为 sew。

根据浮点控制寄存器 fcsr 中的 rm 位确定舍入模式。

- 3' b000: 就近向偶数舍入。
- 3' b001: 向零舍入。
- 3' b010: 向负无穷舍入。
- 3' b011: 向正无穷舍入。
- 3' b100: 就近向大值舍入。
- 3' b101: 非法值, 当配置为该值时, 执行 vfwsb.wv 产生非法指令异常。
- 3' b110: 非法值, 当配置为该值时, 执行 vfwsb.wv 产生非法指令异常。
- 3' b111: 非法值, 当配置为该值时, 执行 vfwsb.wv 产生非法指令异常。

当 vm=1 时, 指令未被 mask, 对应汇编指令 vfwsb.wv vd, vs2, vs1。

当 vm=0 时, 指令被 mask, 对应汇编指令 vfwsb.wv vd, vs2, vs1,v0.t。

指令格式:

31		26	25	24		20	19		15	14	12	11		7	6		0	
110110			vm	vs2			vs1			001			vd			1010111		

14.7.101 VID.V——矢量 index 求解指令

语法:

vid.v vd, vm

操作:

vd[i] = i

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

无

说明:

当 vm=1 时，指令未被 mask，对应汇编指令 vid.v vd。

当 vm=0 时，指令被 mask，对应汇编指令 vid.v vd, v0.t

指令格式：

31	26	25	24	20	19	15	14	12	11	7	6	0
010110	vm			vs2			10001			010	vd	1010111

14.7.102 VIOTA.M——矢量 active 元素统计指令

语法：

viota.m vd, vs2, vm

操作：

```

for(i=0;i<VLMAX;i++){
    if vs2[i] is active{
        count = 0
        for(j=0;j<i;j++){
            if vs2[i] is active
                count = count + 1
        }
        vd[i] = count
    }
    else
        if vs2[i] is inactive{
            vd[i] = vd[i]
        }
        else{
            vd[i] = 0
        }
}
    
```

执行权限：

M mode/S mode/U mode

异常：

非法指令异常

影响标志位：

无

说明：

当 $vm=1$ 时, 指令未被 mask, 对应汇编指令 `viota.m vd, vs2`。

当 $vm=0$ 时, 指令被 mask, 对应汇编指令 `viota.m vd, vs2, v0.t`。

指令格式:

31	26	25	24	20	19	15	14	12	11	7	6	0
010110	vm			vs2			10000	010			vd	1010111

14.7.103 VLB.V——矢量有符号字节加载指令

语法:

`vlb.v vd, (rs1), vm`

操作:

$vd[i] = \text{sign_extend}(\text{mem}[\text{rs1}+i])$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常、加载指令非对齐访问异常、加载指令访问错误异常、加载指令页面错误异常

影响标志位:

无

说明:

读取 byte 数据后有符号扩展至当前 sew 长度

当 $vm=1$ 时, 指令未被 mask, 对应汇编指令 `vlb.v vd, (rs1)`。

当 $vm=0$ 时, 指令被 mask, 对应汇编指令 `vlb.v vd, (rs1), v0.t`。

指令格式:

31	29	28	26	25	24	20	19	15	14	12	11	7	6	0	
000	100	vm			00000			rs1			000	vd			0000111

14.7.104 VLBF.V——矢量 FOF 有符号字节加载指令

语法:

`vlbff.v vd, (rs1), vm`

操作:

$vd[i] = \text{sign_extend}(\text{mem}[\text{rs1}+i])$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常, 加载指令非对齐访问异常, 加载指令访问错误异常, 加载指令页面错误异常

影响标志位:

无

说明:

读取 byte 数据后有符号扩展至当前 sew 长度, 当元素 0 出现 trap 时直接响应 trap, 在其他元素出现 trap 时后续元素将不会继续执行并且不响应当前 trap, 修改 vl 寄存器为出问题元素 index。

当 vm=1 时, 指令未被 mask, 对应汇编指令 vlbff.v vd, (rs1)。

当 vm=0 时, 指令被 mask, 对应汇编指令 vlbff.v vd, (rs1), v0.t

指令格式:

31	29	28	26	25	24	20	19	15	14	12	11	7	6	0
000		100	vm		10000		rs1		000		vd			0000111

14.7.105 VLBU.V——矢量无符号字节加载指令

语法:

vlbu.v vd, (rs1), vm

操作:

vd[i] = zero_extend(mem[rs1+i])

执行权限:

M mode/S mode/U mode

异常:

非法指令异常, 加载指令非对齐访问异常, 加载指令访问错误异常, 加载指令页面错误异常

影响标志位:

无

说明:

读取 byte 数据后无符号扩展至当前 sew 长度

当 vm=1 时, 指令未被 mask, 对应汇编指令 vlbu.v vd, (rs1)。

当 vm=0 时, 指令被 mask, 对应汇编指令 vlbu.v vd, (rs1), v0.t。

指令格式:

31	29	28	26	25	24	20	19	15	14	12	11	7	6	0
000	000	vm	00000			rs1		000		vd		0000111		

14.7.106 VLBUFF.V——矢量 FOF 无符号字节加载指令

语法:

vlbuff.v vd, (rs1), vm

操作:

$vd[i] = \text{zero_extend}(\text{mem}[\text{rs1}+i])$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常, 加载指令非对齐访问异常, 加载指令访问错误异常, 加载指令页面错误异常

影响标志位:

无

说明:

读取 byte 数据后无符号扩展至当前 sew 长度, 当元素 0 出现 trap 时直接响应 trap, 在其他元素出现 trap 时后续元素将不会继续执行并且不响应当前 trap, VL 寄存器被更新为出问题元素 index。

当 vm=1 时, 指令未被 mask, 对应汇编指令 vlbuff.v vd, (rs1)。

当 vm=0 时, 指令被 mask, 对应汇编指令 vlbuff.v vd, (rs1), v0.t

指令格式:

31	29	28	26	25	24	20	19	15	14	12	11	7	6	0
000	000	vm	10000			rs1		000		vd		0000111		

14.7.107 VLE.V——矢量元素加载指令

语法:

vle.v vd, (rs1), vm

操作:

$vd[i] = \text{mem}[\text{rs1}+i*\text{SEW}/8]$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常, 加载指令非对齐访问异常, 加载指令访问错误异常, 加载指令页面错误异常

影响标志位:

无

说明:

读取当前 sew 长度数据

当 vm=1 时, 指令未被 mask, 对应汇编指令 vle.v vd, (rs1)。

当 vm=0 时, 指令被 mask, 对应汇编指令 vle.v vd, (rs1), v0.t

指令格式:

31	29 28	26 25 24	20 19	15 14	12 11	7 6	0
000	000	vm	00000	rs1	111	vd	0000111

14.7.108 VLEFF.V——矢量 FOF 无符号元素加载指令

语法:

vleff.v vd, (rs1), vm

操作:

$vd[i] = mem[rs1+i*SEW/8]$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常, 加载指令非对齐访问异常, 加载指令访问错误异常, 加载指令页面错误异常

影响标志位:

无

说明:

当元素 0 出现 trap 时直接响应 trap, 在其他元素出现 trap 时后续元素将不会继续执行并且不响应当前 trap, vl 寄存器被更新为出问题元素 index。

当 vm=1 时, 指令未被 mask, 对应汇编指令 vleff.v vd, (rs1)。

当 vm=0 时, 指令被 mask, 对应汇编指令 vleff.v vd, (rs1), v0.t

指令格式:

31	29 28	26 25 24	20 19	15 14	12 11	7 6	0
000	000	vm	10000	rs1	111	vd	0000111

14.7.109 VLH.V——矢量有符号半字加载指令

语法:

vlh.v vd, (rs1), vm

操作:

$vd[i] = \text{sign_extend}(\text{mem}[\text{rs1}+2*i])$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常, 加载指令非对齐访问异常, 加载指令访问错误异常, 加载指令页面错误异常

影响标志位:

无

说明:

读取 half word 数据后有符号扩展至当前 sew 长度

当 vm=1 时, 指令未被 mask, 对应汇编指令 vlh.v vd, (rs1)。

当 vm=0 时, 指令被 mask, 对应汇编指令 vlh.v vd, (rs1), v0.t。

指令格式:

31	29 28	26 25 24	20 19	15 14	12 11	7 6	0
000	100	vm	00000	rs1	101	vd	0000111

14.7.110 VLHFF.V——矢量 FOF 有符号半字加载指令

语法:

vlhff.v vd, (rs1), vm

操作:

$vd[i] = \text{sign_extend}(\text{mem}[\text{rs1}+2*i])$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常, 加载指令非对齐访问异常, 加载指令访问错误异常, 加载指令页面错误异常

影响标志位:

无

说明:

读取 half word 数据后有符号扩展至当前 sew 长度, 当元素 0 出现 trap 时直接响应 trap, 在其他元素出现 trap 时后续元素将不会继续执行并且不响应当前 trap, vl 寄存器被更新为出问题元素 index。

当 vm=1 时, 指令未被 mask, 对应汇编指令 vlhff.v vd, (rs1)。

当 vm=0 时, 指令被 mask, 对应汇编指令 vlhff.v vd, (rs1), v0.t。

指令格式:

31	29	28	26	25	24	20	19	15	14	12	11	7	6	0
000	100	vm	10000			rs1			101	vd			0000111	

14.7.111 VLHU.V——矢量无符号半字加载指令

语法:

vlhu.v vd, (rs1), vm

操作:

$vd[i] = \text{zero_extend}(\text{mem}[\text{rs1}+2*i])$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常, 加载指令非对齐访问异常, 加载指令访问错误异常, 加载指令页面错误异常

影响标志位:

无

说明:

读取 half word 数据后无符号扩展至当前 sew 长度

当 vm=1 时, 指令未被 mask, 对应汇编指令 vlhu.v vd, (rs1)。

当 vm=0 时, 指令被 mask, 对应汇编指令 vlhu.v vd, (rs1), v0.t。

指令格式:

31	29	28	26	25	24	20	19	15	14	12	11	7	6	0
000	000	vm	00000			rs1			101	vd			0000111	

14.7.112 VLHUFF.V——矢量 FOF 无符号半字加载指令

语法:

`vlhuff.v vd, (rs1), vm`

操作:

$vd[i] = \text{zero_extend}(\text{mem}[\text{rs1}+2*i])$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常, 加载指令非对齐访问异常, 加载指令访问错误异常, 加载指令页面错误异常

影响标志位:

无

说明:

读取 half word 数据后无符号扩展至当前 SEW 长度, 当元素 0 出现 trap 时直接响应 trap, 在其他元素出现 trap 时后续元素将不会继续执行并且不响应当前 trap, vl 寄存器被更新为出问题元素 index。

当 $vm=1$ 时, 指令未被 mask, 对应汇编指令 `vlhuff.v vd, (rs1)`。

当 $vm=0$ 时, 指令被 mask, 对应汇编指令 `vlhuff.v vd, (rs1), v0.t`

指令格式:

31	29 28	26 25 24	20 19	15 14	12 11	7 6	0
000	000	vm	10000	rs1	101	vd	0000111

14.7.113 VLBSB.V——矢量步进有符号字节加载指令

语法:

`vlsb.v vd, (rs1), rs2, vm`

操作:

$vd[i] = \text{sign_extend}(\text{mem}[\text{rs1}+i*\text{rs2}])$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常, 加载指令非对齐访问异常, 加载指令访问错误异常, 加载指令页面错误异常

影响标志位:

无

说明:

读取 byte 数据后有符号扩展至当前 sew 长度

当 vm=1 时, 指令未被 mask, 对应汇编指令 vlsb.v vd, (rs1), rs2。

当 vm=0 时, 指令被 mask, 对应汇编指令 vlsb.v vd, (rs1), rs2, v0.t。

指令格式:

31 29 28 26 25 24 20 19 15 14 12 11 7 6 0

31	29	28	26	25	24	20	19	15	14	12	11	7	6	0
000	110	vm	rs2			rs1		000		vd				0000111

14.7.114 VLSBU.V——矢量步进无符号字节加载指令

语法:

vlsbu.v vd, (rs1), rs2, vm

操作:

$vd[i] = \text{zero_extend}(\text{mem}[\text{rs1} + i * \text{rs2}])$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常, 加载指令非对齐访问异常, 加载指令访问错误异常, 加载指令页面错误异常

影响标志位:

无

说明:

读取 byte 数据后无符号扩展至当前 sew 长度

当 vm=1 时, 指令未被 mask, 对应汇编指令 vlsbu.v vd, (rs1), rs2。

当 vm=0 时, 指令被 mask, 对应汇编指令 vlsbu.v vd, (rs1), rs2, v0.t

指令格式:

31	29	28	26	25	24	20	19	15	14	12	11	7	6	0
000	010	vm	rs2			rs1		000		vd				0000111

14.7.115 VLSE.V——矢量步进元素加载指令

语法:

vlse.v vd, (rs1), rs2, vm

操作:

$vd[i] = mem[rs1+i*rs2]$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常, 加载指令非对齐访问异常, 加载指令访问错误异常, 加载指令页面错误异常

影响标志位: 无

说明:

读取当前 sew 长度的数据

当 vm=1 时, 指令未被 mask, 对应汇编指令 vlse.v vd, (rs1), rs2。

当 vm=0 时, 指令被 mask, 对应汇编指令 vlse.v vd, (rs1), rs2, v0.t

指令格式:

31	29 28	26 25 24	20 19	15 14	12 11	7 6	0
000	010	vm	rs2	rs1	111	vd	0000111

14.7.116 VLSEG<NF>B.V——矢量 SEGMENT 有符号字节加载指令

语法:

vlseg<nf>b.v vd, (rs1), vm

操作:

for(k=0; k<=nf-1; k++) {

$V_{d+k}[i] = mem[rs1 + k + i * nf]$

}

nf 指定每个 segment 中 field 的数量和目的寄存器的数量, nf 取值范围为 2~8。如当 nf=4, vd=8, rs1=5, vm=1, lmul=1 时, 指令为 vlseg4b.v v8, (x5), 此时指令操作如下:

$v8[0]=mem[x5], v8[1]=mem[x5+4] \dots v8[i]=mem[x5+4*i]$

$v9[0]=mem[x5+1], v9[1]=mem[x5+5] \dots v9[i]=mem[x5+1+4*i]$

$v10[0]=mem[x5+2], v10[1]=mem[x5+6] \dots v10[i]=mem[x5+2+4*i]$

$v11[0]=mem[x5+3], v11[1]=mem[x5+7] \dots v11[i]=mem[x5+3+4*i]$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常, 加载指令非对齐访问异常, 加载指令访问错误异常, 加载指令页面错误异常

影响标志位:

无

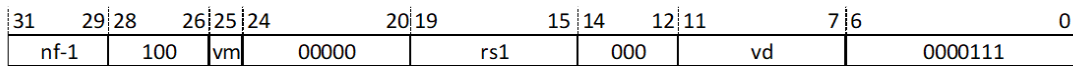
说明:

读取 byte 数据后有符号扩展至当前 sew 长度

当 vm=1 时, 指令未被 mask, 对应汇编指令 vlseg<nf>b.v vd, (rs1)

当 vm=0 时, 指令被 mask, 对应汇编指令 vlseg<nf>b.v vd, (rs1), v0.t

指令格式:



14.7.117 VLSEG<NF>BFF.V——矢量 FOF SEGMENT 有符号字节加载指令

语法:

vlseg<nf>bff.v vd, (rs1), vm

操作:

```
for( k=0; k<=nf-1; k++) {
    Vd+k[i] = mem[rs1 + k + i *nf]
}
```

nf 指定每个 segment 中 field 的数量和目的寄存器的数量, nf 取值范围为 2~8。如当 nf=4, vd=8, rs1=5, vm=1, lmul=1 时, 指令为 vlseg4bff.v v8, (x5), 此时指令操作如下:

```
v8[0]=mem[x5],v8[1]=mem[x5+4] ... v8[i]=mem[x5+4*i]
v9[0]=mem[x5+1],v9[1]=mem[x5+5] ... v9[i]=mem[x5+1+4*i]
v10[0]=mem[x5+2],v10[1]=mem[x5+6] ... v10[i]=mem[x5+2+4*i]
v11[0]=mem[x5+3],v11[1]=mem[x5+7] ... v11[i]=mem[x5+3+4*i]
```

执行权限:

M mode/S mode/U mode

异常:

非法指令异常, 加载指令非对齐访问异常, 加载指令访问错误异常, 加载指令页面错误异常

影响标志位:

无

说明:

读取 byte 数据后有符号扩展至当前 sew 长度, 当元素 0 出现 trap 时直接响应 trap, 在其他元素出现 trap 时后续元素将不会继续执行并且不响应当前 trap, 修改 vl 寄存器为出问题元素 index。

当 vm=1 时, 指令未被 mask, 对应汇编指令 vlseg<nf>bff.v vd, (rs1)

当 vm=0 时, 指令被 mask, 对应汇编指令 vlseg<nf>bff.v vd, (rs1), v0.t

指令格式:

31	29 28	26 25 24	20 19	15 14	12 11	7 6	0
nf-1	100	vm	10000	rs1	000	vd	0000111

14.7.118 VLSEG<NF>BU.V——矢量 SEGMENT 无符号字节加载指令

语法:

vlseg<nf>bu.v vd, (rs1), vm

操作:

```
for( k=0; k<=nf-1 ; k++) {
    Vd+k[i] = mem[rs1 + k + i *nf]
}
```

nf 指定每个 segment 中 field 的数量和目的寄存器的数量, nf 取值范围为 2~8。如当 nf=4, vd=8, rs1=5, vm=1, lmul=1 时, 指令为 vlseg4bu.v v8, (x5), 此时指令操作如下:

```
v8[0]=mem[x5],v8[1]=mem[x5+4] ... v8[i]=mem[x5+4*i]
v9[0]=mem[x5+1],v9[1]=mem[x5+5] ... v9[i]=mem[x5+1+4*i]
v10[0]=mem[x5+2],v10[1]=mem[x5+6] ... v10[i]=mem[x5+2+4*i]
v11[0]=mem[x5+3],v11[1]=mem[x5+7] ... v11[i]=mem[x5+3+4*i]
```

执行权限:

M mode/S mode/U mode

异常:

非法指令异常, 加载指令非对齐访问异常, 加载指令访问错误异常, 加载指令页面错误异常

影响标志位:

无

说明:

读取 byte 数据后无符号扩展至当前 sew 长度

当 vm=1 时, 指令未被 mask, 对应汇编指令 vlseg<nf>bu.v vd, (rs1)

当 vm=0 时, 指令被 mask, 对应汇编指令 vlseg<nf>bu.v vd, (rs1), v0.t

指令格式:

31	29	28	26	25	24	20	19	15	14	12	11	7	6	0
nf-1		000	vm		00000		rs1		000		vd		0000111	

14.7.119 VLSEG<NF>BUFF.V——矢量 FOF SEGMENT 无符号字节加载指令

语法:

vlseg<nf>buff.v vd, (rs1), vm

操作:

```
for( k=0; k<=nf-1; k++) {
    Vd+k[i] = mem[rs1 + k + i *nf]
}
```

nf 指定每个 segment 中 field 的数量和目的寄存器的数量, nf 取值范围为 2~8。如当 nf=4, vd=8, rs1=5, vm=1, lmul=1 时, 指令为 vlseg4buff.v v8, (x5), 此时指令操作如下:

```
v8[0]=mem[x5],v8[1]=mem[x5+4] ... v8[i]=mem[x5+4*i]
v9[0]=mem[x5+1],v9[1]=mem[x5+5] ... v9[i]=mem[x5+1+4*i]
v10[0]=mem[x5+2],v10[1]=mem[x5+6] ... v10[i]=mem[x5+2+4*i]
v11[0]=mem[x5+3],v11[1]=mem[x5+7] ... v11[i]=mem[x5+3+4*i]
```

执行权限:

M mode/S mode/U mode

异常:

非法指令异常, 加载指令非对齐访问异常, 加载指令访问错误异常, 加载指令页面错误异常

影响标志位:

无

说明:

读取 byte 数据后无符号扩展至当前 sew 长度, 当元素 0 出现 trap 时直接响应 trap, 在其他元素出现 trap 时后续元素将不会继续执行并且不响应当前 trap, 修改 vl 寄存器为出问题元素 index。

当 vm=1 时, 指令未被 mask, 对应汇编指令 vlseg<nf>buff.v vd, (rs1)

当 vm=0 时, 指令被 mask, 对应汇编指令 vlseg<nf>buff.v vd, (rs1), v0.t

指令格式:

31	29	28	26	25	24	20	19	15	14	12	11	7	6	0
nf-1		000	vm		10000		rs1		000		vd		0000111	

14.7.120 VLSEG<NF>E.V——矢量 SEGMENT 元素加载指令

语法:

vlseg<nf>e.v vd, (rs1), vm

操作:

```
offset=sew/8
for( k=0; k<=nf-1; k++) {
    Vd+k[i] = mem[rs1 + k*offset + i*nf*offset]
}
```

nf 指定每个 segment 中 field 的数量和目的寄存器的数量, nf 取值范围为 2~8。如当 nf=4, vd=8, rs1=5, vm=1, sew=32, lmul=1 时, 指令为 vlseg4e.v v8, (x5), offset=32/8=4, 此时指令操作如下:

```
v8[0]=mem[x5],v8[1]=mem[x5+4*4] ... v8[i]=mem[x5+16*i]
v9[0]=mem[x5+4],v9[1]=mem[x5+5*4] ... v9[i]=mem[x5+4+16*i]
v10[0]=mem[x5+2*4],v10[1]=mem[x5+6*4] ... v10[i]=mem[x5+8+16*i]
v11[0]=mem[x5+3*4],v11[1]=mem[x5+7*4] ... v11[i]=mem[x5+12+16*i]
```

执行权限:

M mode/S mode/U mode

异常:

非法指令异常, 加载指令非对齐访问异常, 加载指令访问错误异常, 加载指令页面错误异常

影响标志位:

无

说明:

读取当前 sew 长度的数据

当 vm=1 时, 指令未被 mask, 对应汇编指令 vlseg<nf>e.v vd, (rs1), rs2。

当 vm=0 时, 指令被 mask, 对应汇编指令 vlseg<nf>e.v vd, (rs1), rs2, v0.t

指令格式:

31	29	28	26	25	24	20	19	15	14	12	11	7	6	0
nf-1	000	vm	00000			rs1		111		vd		0000111		

14.7.121 VLSEG<NF>EFF.V——矢量 FOF SEGMENT 元素加载指令

语法:

vlseg<nf>eff.v vd, (rs1), vm

操作:

```
offset=sew/8
for( k=0; k<=nf-1; k++) {
    Vd+k[i] = mem[rs1 + k*offset + i*nf*offset]
}
```

nf+1 指定每个 segment 中 field 的数量和目的寄存器的数量, nf 取值范围为 2~8。如当 nf=4, vd=8, rs1=5, vm=1, sew=32, lmul=1 时, 指令为 vlseg4eff.v v8, (x5), offset=32/8=4, 此时指令操作如下:

```
v8[0]=mem[x5],v8[1]=mem[x5+4*4] ... v8[i]=mem[x5+16*i]
v9[0]=mem[x5+4],v9[1]=mem[x5+5*4] ... v9[i]=mem[x5+4+16*i]
v10[0]=mem[x5+2*4],v10[1]=mem[x5+6*4] ... v10[i]=mem[x5+8+16*i]
v11[0]=mem[x5+3*4],v11[1]=mem[x5+7*4] ... v11[i]=mem[x5+12+16*i]
```

执行权限:

M mode/S mode/U mode

异常:

非法指令异常, 加载指令非对齐访问异常, 加载指令访问错误异常, 加载指令页面错误异常

影响标志位:

无

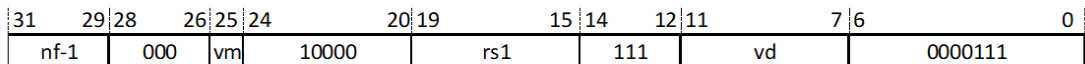
说明:

当元素 0 出现 trap 时直接响应 trap, 在其他元素出现 trap 时后续元素将不会继续执行并且不响应当前 trap, vl 寄存器被更新为出问题元素 index。

当 vm=1 时, 指令未被 mask, 对应汇编指令 vlseg<nf>eff.v vd, (rs1), rs2。

当 vm=0 时, 指令被 mask, 对应汇编指令 vlseg<nf>eff.v vd, (rs1), rs2, v0.t

指令格式:



14.7.122 VLSEG<NF>H.V——矢量 SEGMENT 有符号半字加载指令

语法:

```
vlseg<nf>h.v vd, (rs1), vm
```

操作:

```
for( k=0; k<=nf-1; k++) {
    Vd+k[i] = mem[rs1 + 2*k + 2*i *nf]
```

}

nf 指定每个 segment 中 field 的数量和目的寄存器的数量, nf 取值范围为 2~8。如当 nf=4, vd=8, rs1=5, vm=1, lmul=1 时, 指令为 vlseg4h.v v8, (x5), 此时指令操作如下:

```
v8[0]=mem[x5],v8[1]=mem[x5+8] ... v8[i]=mem[x5+8*i]
v9[0]=mem[x5+2],v9[1]=mem[x5+10] ... v9[i]=mem[x5+2+8*i]
v10[0]=mem[x5+4],v10[1]=mem[x5+12] ... v10[i]=mem[x5+4+8*i]
v11[0]=mem[x5+6],v11[1]=mem[x5+14] ... v11[i]=mem[x5+6+8*i]
```

执行权限:

M mode/S mode/U mode

异常:

非法指令异常, 加载指令非对齐访问异常, 加载指令访问错误异常, 加载指令页面错误异常

影响标志位:

无

说明:

读取 half word 数据后有符号扩展至当前 sew 长度
 当 vm=1 时, 指令未被 mask, 对应汇编指令 vlseg<nf>h.v vd, (rs1)
 当 vm=0 时, 指令被 mask, 对应汇编指令 vlseg<nf>h.v vd, (rs1), v0.t

指令格式:

31	29	28	26	25	24	20	19	15	14	12	11	7	6	0
nf-1	100	vm	00000	rs1	101	vd	0000111							

14.7.123 VLSEG<NF>HFF.V——矢量 FOF SEGMENT 有符号半字加载指令

语法:

vlseg<nf>hff.v vd, (rs1), vm

操作:

```
for( k=0; k<=nf-1; k++) {
    Vd+k[i] = mem[rs1 + 2*k + 2*i *nf]
}
```

nf 指定每个 segment 中 field 的数量和目的寄存器的数量, nf 取值范围为 2~8。如当 nf=4, vd=8, rs1=5, vm=1, lmul=1 时, 指令为 vlseg4hff.v v8, (x5), 此时指令操作如下:

```
v8[0]=mem[x5],v8[1]=mem[x5+8] ... v8[i]=mem[x5+8*i]
v9[0]=mem[x5+2],v9[1]=mem[x5+10] ... v9[i]=mem[x5+2+8*i]
v10[0]=mem[x5+4],v10[1]=mem[x5+12] ... v10[i]=mem[x5+4+8*i]
v11[0]=mem[x5+6],v11[1]=mem[x5+14] ... v11[i]=mem[x5+6+8*i]
```

执行权限:

M mode/S mode/U mode

异常:

非法指令异常, 加载指令非对齐访问异常, 加载指令访问错误异常, 加载指令页面错误异常

影响标志位:

无

说明:

读取 half word 数据后有符号扩展至当前 sew 长度, 当元素 0 出现 trap 时直接响应 trap, 在其他元素出现 trap 时后续元素将不会继续执行并且不响应当前 trap, vl 寄存器被更新为出问题元素 index。

当 vm=1 时, 指令未被 mask, 对应汇编指令 vlseg<nf>hff.v vd, (rs1), rs2。

当 vm=0 时, 指令被 mask, 对应汇编指令 vlseg<nf>hff.v vd, (rs1), rs2, v0.t

指令格式:

31	29	28	26	25	24	20	19	15	14	12	11	7	6	0
nf-1		100		vm	10000		rs1		101		vd		0000111	

14.7.124 VLSEG<NF>HU.V——矢量 SEGMENT 无符号半字加载指令

语法:

vlseg<nf>hu.v vd, (rs1), vm

操作:

```
for( k=0; k<=nf-1; k++) {
    Vd+k[i] = mem[rs1 + 2*k + 2*i *nf]
}
```

nf 指定每个 segment 中 field 的数量和目的寄存器的数量, nf 取值范围为 2~8。如当 nf=4, vd=8, rs1=5, vm=1, lmul=1 时, 指令为 vlseg4hu.v v8, (x5), 此时指令操作如下:

```
v8[0]=mem[x5],v8[1]=mem[x5+8] ... v8[i]=mem[x5+8*i]
v9[0]=mem[x5+2],v9[1]=mem[x5+10] ... v9[i]=mem[x5+2+8*i]
v10[0]=mem[x5+4],v10[1]=mem[x5+12] ... v10[i]=mem[x5+4+8*i]
v11[0]=mem[x5+6],v11[1]=mem[x5+14] ... v11[i]=mem[x5+6+8*i]
```

执行权限:

M mode/S mode/U mode

异常:

非法指令异常, 加载指令非对齐访问异常, 加载指令访问错误异常, 加载指令页面错误异常

影响标志位:

无

说明:

读取 half word 数据后无符号扩展至当前 sew 长度

当 $vm=1$ 时, 指令未被 mask, 对应汇编指令 $vlseg<nf>hu.v\ vd, (rs1)$

当 $vm=0$ 时, 指令被 mask, 对应汇编指令 $vlseg<nf>hu.v\ vd, (rs1), v0.t$

指令格式:

31	29	28	26	25	24	20	19	15	14	12	11	7	6	0
nf-1	000		vm	00000			rs1	101		vd		0000111		

14.7.125 VLSEG<NF>HUFF.V——矢量 FOF SEGMENT 无符号半字加载指令

语法:

$vlseg<nf>huff.v\ vd, (rs1), vm$

操作:

for($k=0; k \leq nf-1; k++$) {

$V_{d+k}[i] = mem[rs1 + 2*k + 2*i * nf]$

}

nf 指定每个 segment 中 field 的数量和目的寄存器的数量, nf 取值范围为 2~8。如当 $nf=4, vd=8, rs1=5, vm=1, lmul=1$ 时, 指令为 $vlseg4huff.v\ v8, (x5)$, 此时指令操作如下:

$v8[0]=mem[x5], v8[1]=mem[x5+8] \dots v8[i]=mem[x5+8*i]$

$v9[0]=mem[x5+2], v9[1]=mem[x5+10] \dots v9[i]=mem[x5+2+8*i]$

$v10[0]=mem[x5+4], v10[1]=mem[x5+12] \dots v10[i]=mem[x5+4+8*i]$

$v11[0]=mem[x5+6], v11[1]=mem[x5+14] \dots v11[i]=mem[x5+6+8*i]$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常, 加载指令非对齐访问异常, 加载指令访问错误异常, 加载指令页面错误异常

影响标志位:

无

说明:

读取 half word 数据后无符号扩展至当前 sew 长度，当元素 0 出现 trap 时直接响应 trap，在其他元素出现 trap 时后续元素将不会继续执行并且不响应当前 trap，vl 寄存器被更新为出问题元素 index。

当 vm=1 时，指令未被 mask，对应汇编指令 vlseg<nf>huff.v vd, (rs1), rs2。

当 vm=0 时，指令被 mask，对应汇编指令 vlseg<nf>huff.v vd, (rs1), rs2, v0.t

指令格式：

31	29 28	26 25 24	20 19	15 14	12 11	7 6	0
nf-1	000	vm	10000	rs1	101	vd	0000111

14.7.126 VLSEG<NF>W.V——矢量 SEGMENT 有符号字加载指令

语法：

vlseg<nf>w.v vd, (rs1), vm

操作：

```
for( k=0; k<=nf-1; k++) {
    Vd+k[i] = mem[rs1 + 4*k + 4*i *nf]
}
```

nf 指定每个 segment 中 field 的数量和目的寄存器的数量，nf 取值范围为 2~8。如当 nf=4, vd=8, rs1=5, vm=1, lmul=1 时，指令为 vlseg4w.v v8, (x5)，此时指令操作如下：

```
v8[0]=mem[x5],v8[1]=mem[x5+16] ...v8[i]=mem[x5+16*i]
v9[0]=mem[x5+4],v9[1]=mem[x5+20] ...v9[i]=mem[x5+4+16*i]
v10[0]=mem[x5+8],v10[1]=mem[x5+24] ...v10[i]=mem[x5+8+16*i]
v11[0]=mem[x5+12],v11[1]=mem[x5+28] ...v11[i]=mem[x5+12+16*i]
```

执行权限：

M mode/S mode/U mode

异常：

非法指令异常，加载指令非对齐访问异常，加载指令访问错误异常，加载指令页面错误异常

影响标志位：

无

说明：

读取 word 数据后有符号扩展至当前 sew 长度

当 vm=1 时，指令未被 mask，对应汇编指令 vlseg<nf>w.v vd, (rs1)

当 vm=0 时，指令被 mask，对应汇编指令 vlseg<nf>w.v vd, (rs1), v0.t

指令格式：

31	29 28	26 25 24	20 19	15 14	12 11	7 6	0
nf-1	100	vm	00000	rs1	110	vd	0000111

14.7.127 VLSEG<NF>WFF.V——矢量 FOF SEGMENT 有符号字加载指令

语法:

`vlseg<nf>wff.v vd, (rs1), vm`

操作:

```
for( k=0; k<=nf-1; k++) {
    Vd+k[i] = mem[rs1 + 2*k + 2*i *nf]
}
```

nf 指定每个 segment 中 field 的数量和目的寄存器的数量, nf 取值范围为 2~8。如当 nf=4, vd=8, rs1=5, vm=1, lmul=1 时, 指令为 `vlseg4wff.v v8, (x5)`, 此时指令操作如下:

```
v8[0]=mem[x5],v8[1]=mem[x5+8] ... v8[i]=mem[x5+8*i]
v9[0]=mem[x5+2],v9[1]=mem[x5+10] ... v9[i]=mem[x5+2+8*i]
v10[0]=mem[x5+4],v10[1]=mem[x5+12] ... v10[i]=mem[x5+4+8*i]
v11[0]=mem[x5+6],v11[1]=mem[x5+14] ... v11[i]=mem[x5+6+8*i]
```

执行权限:

M mode/S mode/U mode

异常:

非法指令异常, 加载指令非对齐访问异常, 加载指令访问错误异常, 加载指令页面错误异常

影响标志位:

无

说明:

读取 word 数据后有符号扩展至当前 sew 长度, 当元素 0 出现 trap 时直接响应 trap, 在其他元素出现 trap 时后续元素将不会继续执行并且不响应当前 trap, vl 寄存器被更新为出问题元素 index。

当 vm=1 时, 指令未被 mask, 对应汇编指令 `vlseg<nf>wff.v vd, (rs1), rs2`。

当 vm=0 时, 指令被 mask, 对应汇编指令 `vlseg<nf>wff.v vd, (rs1), rs2, v0.t`

指令格式:

31	29	28	26	25	24	20	19	15	14	12	11	7	6	0
nf-1	100	vm	10000		rs1	111		vd					0000111	

14.7.128 VLSEG<NF>WU.V——矢量 SEGMENT 无符号字加载指令

语法:

`vlseg<nf>wu.v vd, (rs1), vm`

操作:

```
for( k=0; k<=nf-1; k++) {
    Vd+k[i] = mem[rs1 + 4*k + 4*i *nf]
}
```

nf 指定每个 segment 中 field 的数量和目的寄存器的数量, nf 取值范围为 2~8。如当 nf=4, vd=8, rs1=5, vm=1, lmul=1 时, 指令为 vlseg4wu.v v8, (x5), 此时指令操作如下:

```
v8[0]=mem[x5],v8[1]=mem[x5+16] ...v8[i]=mem[x5+16*i]
v9[0]=mem[x5+4],v9[1]=mem[x5+20] ...v9[i]=mem[x5+4+16*i]
v10[0]=mem[x5+8],v10[1]=mem[x5+24] ...v10[i]=mem[x5+8+16*i]
v11[0]=mem[x5+12],v11[1]=mem[x5+28] ...v11[i]=mem[x5+12+16*i]
```

执行权限:

M mode/S mode/U mode

异常:

非法指令异常, 加载指令非对齐访问异常, 加载指令访问错误异常, 加载指令页面错误异常

影响标志位:

无

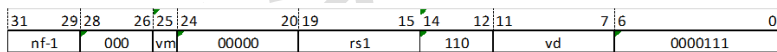
说明:

读取 word 数据后有符号扩展至当前 sew 长度。

当 vm=1 时, 指令未被 mask, 对应汇编指令 vlseg<nf>wu.v vd, (rs1)。

当 vm=0 时, 指令被 mask, 对应汇编指令 vlseg<nf>wu.v vd, (rs1), v0.t

指令格式:



14.7.129 VLSEG<NF>WUFF.V——矢量 FOF SEGMENT 无符号字加载指令

语法:

vlseg<nf>wuff.v vd, (rs1), vm

操作:

```
for( k=0; k<=nf-1; k++) {
    Vd+k[i] = mem[rs1 + 2*k + 2*i *nf]
}
```

nf 指定每个 segment 中 field 的数量和目的寄存器的数量, nf 取值范围为 2~8。如当 nf=4, vd=8, rs1=5, vm=1, lmul=1 时, 指令为 vlseg4wuff.v v8, (x5), 此时指令操作如下:

$$\begin{aligned} v8[0] &= \text{mem}[x5], v8[1] = \text{mem}[x5+8] \dots v8[i] = \text{mem}[x5+8*i] \\ v9[0] &= \text{mem}[x5+2], v9[1] = \text{mem}[x5+10] \dots v9[i] = \text{mem}[x5+2+8*i] \\ v10[0] &= \text{mem}[x5+4], v10[1] = \text{mem}[x5+12] \dots v10[i] = \text{mem}[x5+4+8*i] \\ v11[0] &= \text{mem}[x5+6], v11[1] = \text{mem}[x5+14] \dots v11[i] = \text{mem}[x5+6+8*i] \end{aligned}$$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常, 加载指令非对齐访问异常, 加载指令访问错误异常, 加载指令页面错误异常

影响标志位:

无

说明:

读取 word 数据后无符号扩展至当前 sew 长度, 当元素 0 出现 trap 时直接响应 trap, 在其他元素出现 trap 时后续元素将不会继续执行并且不响应当前 trap, vl 寄存器被更新为出问题元素 index。

当 vm=1 时, 指令未被 mask, 对应汇编指令 vlseg<nf>wuff.v vd, (rs1), rs2。

当 vm=0 时, 指令被 mask, 对应汇编指令 vlseg<nf>wuff.v vd, (rs1), rs2, v0.t

指令格式:

31	29 28	26 25	24	20 19	15 14	12 11	7 6	0
nf-1	000	vm	10000	rs1	111	vd	0000111	

14.7.130 VLSSEG<NF>B.V——矢量步进 SEGMENT 有符号字节加载指令

语法:

vlssseg<nf>b.v vd, (rs1), rs2, vm

操作:

```
for( k=0; k<=nf-1; k++) {
    Vd+k[i] = mem[rs1 + k + i *rs2]
}
```

nf 指定每个 segment 中 field 的数量和目的寄存器的数量, nf 取值范围为 2~8。如当 nf=4, vd=8, rs1=5, rs2=6, vm=1, lmul=1 时, 指令为 vlssseg4b.v v8, (x5), x6 那么指令操作如下:

$$\begin{aligned} v8[0] &= \text{mem}[x5], v8[1] = \text{mem}[x5+x6] \dots v8[i] = \text{mem}[x5+i*x6] \\ v9[0] &= \text{mem}[x5+1], v9[1] = \text{mem}[x5+x6+1] \dots v9[i] = \text{mem}[x5+i*x6+1] \\ v10[0] &= \text{mem}[x5+2], v10[1] = \text{mem}[x5+x6+2] \dots v10[i] = \text{mem}[x5+i*x6+2] \\ v11[0] &= \text{mem}[x5+3], v11[1] = \text{mem}[x5+x6+3] \dots v11[i] = \text{mem}[x5+i*x6+3] \end{aligned}$$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常, 加载指令非对齐访问异常, 加载指令访问错误异常, 加载指令页面错误异常

影响标志位:

无

说明:

读取 byte 数据后有符号扩展至当前 sew 长度

当 $vm=1$ 时, 指令未被 mask, 对应汇编指令 $vlsseg<nf>b.v\ vd, (rs1), rs2$

当 $vm=0$ 时, 指令被 mask, 对应汇编指令 $vlsseg<nf>b.v\ vd, (rs1), rs2, v0.t$

指令格式:

31	29	28	26	25	24	20	19	15	14	12	11	7	6	0
nf-1	110		vm	00000			rs1		000		vd		0000111	

14.7.131 VLSSEG<NF>BU.V——矢量步进 SEGMENT 无符号字节加载指令

语法:

$vlsseg<nf>bu.v\ vd, (rs1), rs2, vm$

操作:

```
for( k=0; k<=nf-1; k++) {
     $V_{d+k}[i] = mem[rs1 + k + i * rs2]$ 
}
```

nf 指定每个 segment 中 field 的数量和目的寄存器的数量, nf 取值范围为 2~8。如当 $nf=4, vd=8, rs1=5, rs2=6, vm=1, lmul=1$ 时, 指令为 $vlsseg4bu.v\ v8, (x5), x6$ 那么指令操作如下:

```
 $v8[0]=mem[x5 ], v8[1]=mem[x5+x6 ] \dots v8[i]=mem[x5+i*x6]$ 
 $v9[0]=mem[x5+1], v9[1]=mem[x5+x6+1] \dots v9[i]=mem[x5+i*x6+1]$ 
 $v10[0]=mem[x5+2], v10[1]=mem[x5+x6+2] \dots v10[i]=mem[x5+i*x6+2]$ 
 $v11[0]=mem[x5+3], v11[1]=mem[x5+x6+3] \dots v11[i]=mem[x5+i*x6+3]$ 
```

执行权限:

M mode/S mode/U mode

异常:

非法指令异常, 加载指令非对齐访问异常, 加载指令访问错误异常, 加载指令页面错误异常

影响标志位:

无

说明:

读取 byte 数据后无符号扩展至当前 sew 长度

当 vm=1 时, 指令未被 mask, 对应汇编指令 vlsseg<nf>bu.v vd, (rs1), rs2

当 vm=0 时, 指令被 mask, 对应汇编指令 vlsseg<nf>bu.v vd, (rs1), rs2, v0.t

指令格式:

31	29	28	26	25	24	20	19	15	14	12	11	7	6	0
nf-1		010	vm	00000			rs1		000		vd		0000111	

14.7.132 VLSSEG<NF>E.V——矢量步进 SEGMENT 元素加载指令

语法:

vlsseg<nf>e.v vd, (rs1), rs2, vm

操作:

offset=sew/8

for(k=0; k<=nf-1; k++) {

V_{d+k}[i] = mem[rs1 + k*offset + i*rs2]

}

nf 指定每个 segment 中 field 的数量和目的寄存器的数量, nf 取值范围为 2~8。如当 nf=4, vd=8, rs1=5, rs2=6, vm=1, sew=32, lmul=1 时, 指令为 vlsseg4e.v v8, (x5), x6, offset=32/8=4 那么指令操作如下:

v8[0]=mem[x5], v8[1]=mem[x5+x6] ... v8[i]=mem[x5+x6]

v9[0]=mem[x5+4], v9[1]=mem[x5+x6+4] ... v9[i]=mem[x5+i*x6+4]

v10[0]=mem[x5+8], v10[1]=mem[x5+x6+8] ... v10[i]=mem[x5+i*x6+8]

v11[0]=mem[x5+12], v11[1]=mem[x5+x6+12] ... v11[i]=mem[x5+i*x6+12]

执行权限:

M mode/S mode/U mode

异常:

非法指令异常, 加载指令非对齐访问异常, 加载指令访问错误异常, 加载指令页面错误异常

影响标志位:

无

说明:

读取当前 sew 长度的数据

当 vm=1 时, 指令未被 mask, 对应汇编指令 vlsseg<nf>e.v vd, (rs1), rs2。

当 vm=0 时, 指令被 mask, 对应汇编指令 vlsseg<nf>e.v vd, (rs1), rs2, v0.t

指令格式:

31	29 28	26 25 24	20 19	15 14	12 11	7 6	0
nf-1	010	vm	00000	rs1	111	vd	0000111

14.7.133 VLSSEG<NF>H.V——矢量步进 SEGMENT 有符号半字加载指令

语法:

vlsseg<nf>h.v vd, (rs1), rs2, vm

操作:

```
for( k=0; k<=nf-1; k++) {
    Vd+k[i] = mem[rs1 + 2*k + i *rs2]
}
```

nf 指定每个 segment 中 field 的数量和目的寄存器的数量, nf 取值范围为 2~8。如当 nf=4, vd=8, rs1=5, rs2=6, vm=1, lmul=1 时, 指令为 vlsseg4h.v v8, (x5), x6 那么指令操作如下:

```
v8[0]=mem[x5 ],v8[1]=mem[x5+x6 ] ... v8[i]=mem[x5+i*x6]
v9[0]=mem[x5+2],v9[1]=mem[x5+x6+2] ... v9[i]=mem[x5+i*x6+2]
v10[0]=mem[x5+4],v10[1]=mem[x5+x6+4] ... v10[i]=mem[x5+i*x6+4]
v11[0]=mem[x5+6],v11[1]=mem[x5+x6+6] ... v11[i]=mem[x5+i*x6+6]
```

执行权限:

M mode/S mode/U mode

异常:

非法指令异常, 加载指令非对齐访问异常, 加载指令访问错误异常, 加载指令页面错误异常

影响标志位:

无

说明:

读取 half word 数据后有符号扩展至当前 sew 长度

当 vm=1 时, 指令未被 mask, 对应汇编指令 vlsseg<nf>h.v vd, (rs1), rs2

当 vm=0 时, 指令被 mask, 对应汇编指令 vlsseg<nf>h.v vd, (rs1), rs2, v0.t

指令格式:

31	29 28	26 25 24	20 19	15 14	12 11	7 6	0
nf-1	110	vm	00000	rs1	101	vd	0000111

14.7.134 VLSSEG<NF>HU.V——矢量步进 SEGMENT 无符号半字加载指令

语法:

vlsseg<nf>hu.v vd, (rs1), rs2, vm

操作:

```
for( k=0; k<=nf-1; k++) {
    Vd+k[i] = mem[rs1 + k + i *rs2]
}
```

nf 指定每个 segment 中 field 的数量和目的寄存器的数量, nf 取值范围为 2~8。如当 nf=4, vd=8, rs1=5, rs2=6, vm=1, lmul=1 时, 指令为 vlsseg4hu.v v8, (x5), x6 那么指令操作如下:

```
v8[0]=mem[x5 ],v8[1]=mem[x5+x6 ] ... v8[i]=mem[x5+i*x6]
v9[0]=mem[x5+2],v9[1]=mem[x5+x6+2] ... v9[i]=mem[x5+i*x6+2]
v10[0]=mem[x5+4],v10[1]=mem[x5+x6+4] ... v10[i]=mem[x5+i*x6+4]
v11[0]=mem[x5+6],v11[1]=mem[x5+x6+6] ... v11[i]=mem[x5+i*x6+6]
```

执行权限:

M mode/S mode/U mode

异常:

非法指令异常, 加载指令非对齐访问异常, 加载指令访问错误异常, 加载指令页面错误异常

影响标志位:

无

说明:

读取 byte 数据后无符号扩展至当前 sew 长度

当 vm=1 时, 指令未被 mask, 对应汇编指令 vlsseg<nf>hu.v vd, (rs1), rs2

当 vm=0 时, 指令被 mask, 对应汇编指令 vlsseg<nf>hu.v vd, (rs1), rs2, v0.t

指令格式:

31	29	28	26	25	24	20	19	15	14	12	11	7	6	0
nf-1	010	vm	0000	rs1	101	vd	0000111							

14.7.135 VLSSEG<NF>W.V——矢量步进 SEGMENT 有符号字加载指令

语法:

vlsseg<nf>w.v vd, (rs1), rs2, vm

操作:

```
for( k=0; k<=nf-1; k++) {
    Vd+k[i] = mem[rs1 + 4*k + i *rs2]
}
```

nf 指定每个 segment 中 field 的数量和目的寄存器的数量, nf 取值范围为 2~8。如当 nf=4, vd=8, rs1=5, rs2=6, vm=1, lmul=1 时, 指令为 vlsseg4w.v v8, (x5), x6 那么指令操作如下:

```
v8[0]=mem[x5 ],v8[1]=mem[x5+x6 ] ... v8[i]=mem[x5+i*x6]
v9[0]=mem[x5+4],v9[1]=mem[x5+x6+4] ... v9[i]=mem[x5+i*x6+4]
v10[0]=mem[x5+8],v10[1]=mem[x5+x6+8] ... v10[i]=mem[x5+i*x6+8]
v11[0]=mem[x5+12],v11[1]=mem[x5+x6+12] ... v11[i]=mem[x5+i*x6+12]
```

执行权限:

M mode/S mode/U mode

异常:

非法指令异常, 加载指令非对齐访问异常, 加载指令访问错误异常, 加载指令页面错误异常

影响标志位:

无

说明:

读取 word 数据后有符号扩展至当前 sew 长度

当 vm=1 时, 指令未被 mask, 对应汇编指令 vlsseg<nf>w.v vd, (rs1), rs2

当 vm=0 时, 指令被 mask, 对应汇编指令 vlsseg<nf>w.v vd, (rs1), rs2, v0.t

指令格式:

31	29 28	26 25 24	20 19	15 14	12 11	7 6	0
nf-1	110	vm	00000	rs1	110	vd	0000111

14.7.136 VLSSEG<NF>WU.V——矢量步进 SEGMENT 无符号字加载指令

语法:

vlsseg<nf>wu.v vd, (rs1), rs2, vm

操作:

```
for( k=0; k<=nf-1; k++) {
    Vd+k[i] = mem[rs1 + 4*k + i *rs2]
}
```

nf 指定每个 segment 中 field 的数量和目的寄存器的数量, nf 取值范围为 2~8。如当 nf=4, vd=8, rs1=5, rs2=6, vm=1, lmul=1 时, 指令为 vlsseg4wu.v v8, (x5), x6 那么指令操作如下:

$v8[0]=mem[x5], v8[1]=mem[x5+x6] \dots v8[i]=mem[x5+i*x6]$
 $v9[0]=mem[x5+4], v9[1]=mem[x5+x6+4] \dots v9[i]=mem[x5+i*x6+4]$
 $v10[0]=mem[x5+8], v10[1]=mem[x5+x6+8] \dots v10[i]=mem[x5+i*x6+8]$
 $v11[0]=mem[x5+12], v11[1]=mem[x5+x6+12] \dots v11[i]=mem[x5+i*x6+12]$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常, 加载指令非对齐访问异常, 加载指令访问错误异常, 加载指令页面错误异常

影响标志位:

无

说明:

读取 word 数据后无符号扩展至当前 sew 长度
 当 $vm=1$ 时, 指令未被 mask, 对应汇编指令 $vlsseg<nf>wu.v vd, (rs1), rs2$
 当 $vm=0$ 时, 指令被 mask, 对应汇编指令 $vlsseg<nf>wu.v vd, (rs1), rs2, v0.t$

指令格式:

31	29	28	26	25	24	20	19	15	14	12	11	7	6	0
nf-1			010		vm	00000		rs1		110		vd		0000111

14.7.137 VLSH.V——矢量步进有符号半字加载指令

语法:

$vlsh.v vd, (rs1), rs2, vm$

操作:

$vd[i] = sign_extend(mem[rs1+i*rs2])$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常, 加载指令非对齐访问异常, 加载指令访问错误异常, 加载指令页面错误异常

影响标志位:

无

说明:

读取 half word 数据后有符号扩展至当前 sew 长度

当 vm=1 时, 指令未被 mask, 对应汇编指令 vlsh.v vd, (rs1), rs2。

当 vm=0 时, 指令被 mask, 对应汇编指令 vlsh.v vd, (rs1), rs2, v0.t

指令格式:

31	29	28	26	25	24	20	19	15	14	12	11	7	6	0
000	110	vm	rs2	rs1	101	vd	0000111							

14.7.138 VLSHU.V——矢量步进无符号半字加载指令

语法:

vlshu.v vd, (rs1), rs2, vm

操作:

$vd[i] = \text{zero_extend}(\text{mem}[\text{rs1} + i * \text{rs2}])$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常, 加载指令非对齐访问异常, 加载指令访问错误异常, 加载指令页面错误异常

影响标志位:

无

说明: 读取 half word 数据后无符号扩展至当前 sew 长度

当 vm=1 时, 指令未被 mask, 对应汇编指令 vlshu.v vd, (rs1), rs2。

当 vm=0 时, 指令被 mask, 对应汇编指令 vlshu.v vd, (rs1), rs2, v0.t

指令格式:

31	29	28	26	25	24	20	19	15	14	12	11	7	6	0
000	010	vm	rs2	rs1	101	vd	0000111							

14.7.139 VLSSH.V——矢量步进有符号半字加载指令

语法:

vlssh.v vd, (rs1), rs2, vm

操作:

$vd[i] = \text{sign_extend}(\text{mem}[\text{rs1} + i * \text{rs2}])$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常、加载指令非对齐访问异常、加载指令访问错误异常、加载指令页面错误异常

影响标志位:

无

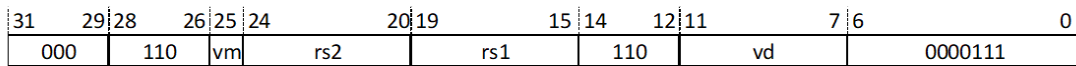
说明:

读取 word 数据后有符号扩展至当前 sew 长度

当 vm=1 时, 指令未被 mask, 对应汇编指令 vlsu.v vd, (rs1), rs2。

当 vm=0 时, 指令被 mask, 对应汇编指令 vlsu.v vd, (rs1), rs2, v0.t。

指令格式:



14.7.140 VLSWU.V——矢量步进无符号字加载指令

语法:

vlsu.v vd, (rs1), rs2, vm

操作:

$vd[i] = \text{zero_extend}(\text{mem}[\text{rs1} + i * \text{rs2}])$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常、加载指令非对齐访问异常、加载指令访问错误异常、加载指令页面错误异常

影响标志位:

无

说明:

读取 word 数据后无符号扩展至当前 sew 长度

当 vm=1 时, 指令未被 mask, 对应汇编指令 vlsu.v vd, (rs1), rs2。

当 vm=0 时, 指令被 mask, 对应汇编指令 vlsu.v vd, (rs1), rs2, v0.t。

指令格式:

31	29 28	26 25 24	20 19	15 14	12 11	7 6	0
000	010	vm	rs2	rs1	110	vd	0000111

14.7.141 VLW.V——矢量有符号字加载指令

语法:

vlw.v vd, (rs1), vm

操作:

$vd[i] = \text{sign_extend}(\text{mem}[rs1+4*i])$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常, 加载指令非对齐访问异常, 加载指令访问错误异常, 加载指令页面错误异常

影响标志位:

无

说明:

读取 word 数据后有符号扩展至当前 sew 长度

当 vm=1 时, 指令未被 mask, 对应汇编指令 vlw.v vd, (rs1)。

当 vm=0 时, 指令被 mask, 对应汇编指令 vlw.v vd, (rs1), v0.t。

指令格式:

31	29 28	26 25 24	20 19	15 14	12 11	7 6	0
000	100	vm	00000	rs1	110	vd	0000111

14.7.142 VLWFF.V——矢量 FOF 有符号字加载指令

语法:

vlwff.v vd, (rs1), vm

操作:

$vd[i] = \text{sign_extend}(\text{mem}[rs1+4*i])$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常, 加载指令非对齐访问异常, 加载指令访问错误异常, 加载指令页面错误异常

影响标志位:

无

说明:

读取 word 数据后有符号扩展至当前 sew 长度, 当元素 0 出现 trap 时直接响应 trap, 在其他元素出现 trap 时后续元素将不会继续执行并且不响应当前 trap, vl 寄存器被更新为出问题元素 index。

当 vm=1 时, 指令未被 mask, 对应汇编指令 vlwff.v vd, (rs1)。

当 vm=0 时, 指令被 mask, 对应汇编指令 vlwff.v vd, (rs1), v0.t。

指令格式:

31	29	28	26	25	24	20	19	15	14	12	11	7	6	0
000	100	vm	10000	rs1	110	vd	0000111							

14.7.143 VLWU.V——矢量无符号字加载指令

语法:

vlwu.v vd, (rs1), vm

操作:

vd[i] = zero_extend(mem[rs1+4*i])

执行权限:

M mode/S mode/U mode

异常:

非法指令异常, 加载指令非对齐访问异常, 加载指令访问错误异常, 加载指令页面错误异常

影响标志位:

无

说明:

读取单字数据后无符号扩展至当前 sew 长度。

当 vm=1 时, 指令未被 mask, 对应汇编指令 vlwu.v vd, (rs1)。

当 vm=0 时, 指令被 mask, 对应汇编指令 vlwu.v vd, (rs1), v0.t。

指令格式:

31	29	28	26	25	24	20	19	15	14	12	11	7	6	0
000	000	vm	00000			rs1		110		vd		0000111		

14.7.144 VLWUFF.V——矢量 FOF 无符号字加载指令

语法:

`vlwuff.v vd, (rs1), vm`

操作:

$vd[i] = \text{zero_extend}(\text{mem}[\text{rs1}+4*i])$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常, 加载指令非对齐访问异常, 加载指令访问错误异常, 加载指令页面错误异常

影响标志位:

无

说明:

读取 word 数据后无符号扩展至当前 sew 长度, 当元素 0 出现 trap 时直接响应 trap, 在其他元素出现 trap 时后续元素将不会继续执行并且不响应当前 trap, 修改 vl 寄存器为出问题元素 index。

当 vm=1 时, 指令未被 mask, 对应汇编指令 `vlwuff.v vd, (rs1)`。

当 vm=0 时, 指令被 mask, 对应汇编指令 `vlwuff.v vd, (rs1), v0.t`。

指令格式:

31	29	28	26	25	24	20	19	15	14	12	11	7	6	0
000	000	vm	10000			rs1		110		vd		0000111		

14.7.145 VLXB.V——矢量索引有符号字节加载指令

语法:

`vlxb.v vd, (rs1), vs2, vm`

操作:

$vd[i] = \text{sign_extend}(\text{mem}[\text{rs1}+\text{sign_extend}(\text{vs2}[i])])$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常, 加载指令非对齐访问异常, 加载指令访问错误异常, 加载指令页面错误异常

影响标志位:

无

说明:

读取 byte 数据后有符号扩展至当前 sew 长度。

当 vm=1 时, 指令未被 mask, 对应汇编指令 vlxb.v vd, (rs1), vs2。

当 vm=0 时, 指令被 mask, 对应汇编指令 vlxb.v vd, (rs1), vs2, v0.t。

指令格式:

31	29 28	26 25 24	20 19	15 14	12 11	7 6	0
000	111	vm	vs2	rs1	000	vd	0000111

14.7.146 VLXBU.V——矢量索引无符号字节加载指令

语法:

vlxbu.v vd, (rs1), vs2, vm

操作:

$vd[i] = \text{zero_extend}(\text{mem}[\text{rs1} + \text{sign_extend}(\text{vs2}[i])])$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常, 加载指令非对齐访问异常, 加载指令访问错误异常, 加载指令页面错误异常

影响标志位:

无

说明:

读取 byte 数据后无符号扩展至当前 sew 长度。

当 vm=1 时, 指令未被 mask, 对应汇编指令 vlxbu.v vd, (rs1), vs2。

当 vm=0 时, 指令被 mask, 对应汇编指令 vlxbu.v vd, (rs1), vs2, v0.t。

指令格式:

31	29 28	26 25 24	20 19	15 14	12 11	7 6	0
000	011	vm	vs2	rs1	000	vd	0000111

14.7.147 VLXE.V——矢量索引元素加载指令

语法:

`vlxe.v vd, (rs1), vs2, vm`

操作:

$vd[i] = mem[rs1 + sign_extend(vs2[i])]$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常, 加载指令非对齐访问异常, 加载指令访问错误异常, 加载指令页面错误异常

影响标志位:

无

说明:

读取当前 `sew` 长度的数据。

当 `vm=1` 时, 指令未被 `mask`, 对应汇编指令 `vlxe.v vd, (rs1), vs2`。

当 `vm=0` 时, 指令被 `mask`, 对应汇编指令 `vlxe.v vd, (rs1), vs2, v0.t`。

指令格式:

31	29 28	26 25 24	20 19	15 14	12 11	7 6	0
000	011	vm	vs2	rs1	111	vd	0000111

14.7.148 VLXH.V——矢量索引有符号半字加载指令

语法:

`vlxh.v vd, (rs1), vs2, vm`

操作:

$vd[i] = sign_extend(mem[rs1 + sign_extend(vs2[i])])$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常, 加载指令非对齐访问异常, 加载指令访问错误异常, 加载指令页面错误异常

影响标志位:

无

说明:

读取 half word 数据后有符号扩展至当前 sew 长度。

当 vm=1 时, 指令未被 mask, 对应汇编指令 vlxh.v vd, (rs1), vs2。

当 vm=0 时, 指令被 mask, 对应汇编指令 vlxh.v vd, (rs1), vs2, v0.t。

指令格式:

31	29	28	26	25	24	20	19	15	14	12	11	7	6	0
000		111	vm		vs2		rs1		101		vd			0000111

14.7.149 VLXHU.V——矢量索引无符号半字加载指令

语法:

vlxhu.v vd, (rs1), vs2, vm

操作:

$vd[i] = \text{zero_extend}(\text{mem}[\text{rs1} + \text{sign_extend}(\text{vs2}[i])])$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常, 加载指令非对齐访问异常, 加载指令访问错误异常, 加载指令页面错误异常

影响标志位:

无

说明:

读取 half word 数据后无符号扩展至当前 sew 长度。

当 vm=1 时, 指令未被 mask, 对应汇编指令 vlxhu.v vd, (rs1), vs2。

当 vm=0 时, 指令被 mask, 对应汇编指令 vlxhu.v vd, (rs1), vs2, v0.t。

指令格式:

31	29	28	26	25	24	20	19	15	14	12	11	7	6	0
000		011	vm		vs2		rs1		101		vd			0000111

14.7.150 VLXSEG<NF>B.V——矢量索引 SEGMENT 有符号字节加载指令

语法:

$vlxseg<nf>b.v\ vd, (rs1), vs2, vm$

操作:

```
for( k=0; k<=nf-1; k++) {
     $V_{d+k}[i] = mem[rs1 + k + vs2[i]]$ 
}
```

nf 指定每个 segment 中 field 的数量和目的寄存器的数量, nf 取值范围为 2~8。如当 nf=4, vd=8, rs1=5, vs2=16, vm=1, lmul=1 时, 指令为 $vlxseg4b.v\ v8, (x5), v16$ 那么指令操作如下:

```
 $v8[0]=mem[x5+v16[0]] \dots v8[i]=mem[x5+vs16[i]]$ 
 $v9[0]=mem[x5+v16[0]+1] \dots v9[i]=mem[x5+vs16[i]+1]$ 
 $v10[0]=mem[x5+v16[0]+2] \dots v10[i]=mem[x5+vs16[i]+2]$ 
 $v11[0]=mem[x5+v16[0]+3] \dots v11[i]=mem[x5+vs16[i]+3]$ 
```

执行权限:

M mode/S mode/U mode

异常:

非法指令异常, 加载指令非对齐访问异常, 加载指令访问错误异常, 加载指令页面错误异常

影响标志位:

无

说明:

读取 byte 数据后有符号扩展至当前 sew 长度

当 vm=1 时, 指令未被 mask, 对应汇编指令 $vlxseg<nf>b.v\ vd, (rs1), vs2$

当 vm=0 时, 指令被 mask, 对应汇编指令 $vlxseg<nf>b.v\ vd, (rs1), vs2, v0.t$

指令格式:

31	29	28	26	25	24	20	19	15	14	12	11	7	6	0
nf-1	111	vm	00000	rs1	000	vd	0000111							

14.7.151 VLXSEG<NF>BU.V——矢量索引 SEGMENT 无符号字节加载指令

语法:

$vlxseg<nf>bu.v\ vd, (rs1), vs2, vm$

操作:

```
for( k=0; k<=nf-1; k++) {
    Vd+k[i] = mem[rs1 + k + vs2[i]]
}
```

nf 指定每个 segment 中 field 的数量和目的寄存器的数量, nf 取值范围为 2~8。如当 nf=4, vd=8, rs1=5, vs2=16, vm=1, lmul=1 时, 指令为 vlxseg4bu.v v8, (x5), v16 那么指令操作如下:

```
v8[0]=mem[x5+v16[0]] ... v8[i]=mem[x5+vs16[i]]
v9[0]=mem[x5+v16[0]+1] ... v9[i]=mem[x5+vs16[i]+1]
v10[0]=mem[x5+v16[0]+2] ... v10[i]=mem[x5+vs16[i]+2]
v11[0]=mem[x5+v16[0]+3] ... v11[i]=mem[x5+vs16[i]+3]
```

执行权限:

M mode/S mode/U mode

异常:

非法指令异常, 加载指令非对齐访问异常, 加载指令访问错误异常, 加载指令页面错误异常

影响标志位:

无

说明:

读取 byte 数据后无符号扩展至当前 sew 长度

当 vm=1 时, 指令未被 mask, 对应汇编指令 vlxseg<nf>bu.v vd, (rs1), vs2

当 vm=0 时, 指令被 mask, 对应汇编指令 vlxseg<nf>bu.v vd, (rs1), vs2, v0.t

指令格式:

31	29	28	26	25	24	20	19	15	14	12	11	7	6	0
nf-1	011		vm	00000			rs1	000		vd	0000111			

14.7.152 VLXSEG<NF>E.V——矢量索引 SEGMENT 元素加载指令

语法:

vlxseg<nf>e.v vd, (rs1), vs2, vm

操作:

offset=sew/8

```
for( k=0; k<=nf-1; k++) {
    Vd+k[i] = mem[rs1 + vs2[i] + k*offset]
}
```

nf 指定每个 segment 中 field 的数量和目的寄存器的数量, nf 取值范围为 2~8。如当 nf=4, vd=8, rs1=5, vs2=6, vm=1, sew=32, lmul=1 时, 指令为 vlxseg4e.v v8, (x5), v16, offset=32/8=4 那么指令操作如下:

$v8[0]=mem[x5+v16[0]] \dots v8[i]=mem[x5+v16[i]]$
 $v9[0]=mem[x5+v16[0]+4] \dots v9[i]=mem[x5+v16[i]+4]$
 $v10[0]=mem[x5+v16[0]+8] \dots v10[i]=mem[x5+v16[i]+8]$
 $v11[0]=mem[x5+v16[0]+12] \dots v11[i]=mem[x5+v16[i]+12]$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常, 加载指令非对齐访问异常, 加载指令访问错误异常, 加载指令页面错误异常

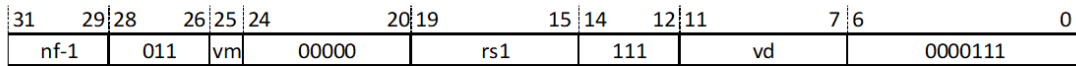
影响标志位:

无

说明:

读取当前 sew 长度的数据
 当 $vm=1$ 时, 指令未被 mask, 对应汇编指令 $vlxseg<nf>.v vd, (rs1), vs2$ 。
 当 $vm=0$ 时, 指令被 mask, 对应汇编指令 $vlxseg<nf>.v vd, (rs1), vs2, v0.t$

指令格式:



14.7.153 VLXSEG<NF>H.V——矢量索引 SEGMENT 有符号半字加载指令

语法:

$vlxseg<nf>.h.v vd, (rs1), vs2, vm$

操作:

```

for( k=0; k<=nf-1; k++) {
     $V_{d+k}[i] = mem[rs1 + vs2[i] + 2*k]$ 
}
    
```

nf 指定每个 segment 中 field 的数量和目的寄存器的数量, nf 取值范围为 2~8。如当 $nf=4, vd=8, rs1=5, vs2=16, vm=1, lmul=1$ 时, 指令为 $vlxseg4h.v v8, (x5), v16$ 那么指令操作如下:

$v8[0]=mem[x5+v16[0]] \dots v8[i]=mem[x5+vs16[i]]$
 $v9[0]=mem[x5+v16[0]+2] \dots v9[i]=mem[x5+vs16[i]+2]$
 $v10[0]=mem[x5+v16[0]+4] \dots v10[i]=mem[x5+vs16[i]+4]$
 $v11[0]=mem[x5+v16[0]+6] \dots v11[i]=mem[x5+vs16[i]+6]$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常, 加载指令非对齐访问异常, 加载指令访问错误异常, 加载指令页面错误异常

影响标志位:

无

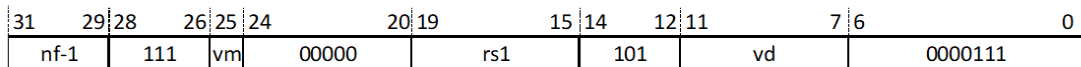
说明:

读取 half word 数据后有符号扩展至当前 sew 长度。

当 vm=1 时, 指令未被 mask, 对应汇编指令 `vlxseg<nf>h.v vd, (rs1), vs2`

当 vm=0 时, 指令被 mask, 对应汇编指令 `vlxseg<nf>h.v vd, (rs1), vs2, v0.t`

指令格式:



14.7.154 VLXSEG<NF>HU.V——矢量索引 SEGMENT 无符号半字加载指令

语法:

`vlxseg<nf>hu.v vd, (rs1), vs2, vm`

操作:

```
for( k=0; k<=nf-1; k++) {
    Vd+k[i] = mem[rs1 + vs2[i] + 2*k]
}
```

nf 指定每个 segment 中 field 的数量和目的寄存器的数量, nf 取值范围为 2~8。如当 nf=4, vd=8, rs1=5, vs2=16, vm=1, lmul=1 时, 指令为 `vlxseg4hu.v v8, (x5), v16` 那么指令操作如下:

$$\begin{aligned}
 v8[0] &= \text{mem}[x5 + v16[0]] \cdots v8[i] = \text{mem}[x5 + vs16[i]] \\
 v9[0] &= \text{mem}[x5 + v16[0] + 2] \cdots v9[i] = \text{mem}[x5 + vs16[i] + 2] \\
 v10[0] &= \text{mem}[x5 + v16[0] + 4] \cdots v10[i] = \text{mem}[x5 + vs16[i] + 4] \\
 v11[0] &= \text{mem}[x5 + v16[0] + 6] \cdots v11[i] = \text{mem}[x5 + vs16[i] + 6]
 \end{aligned}$$

执行权限:

M mode/S mode/U mode

异常: 非法指令异常, 加载指令非对齐访问异常, 加载指令访问错误异常, 加载指令页面错误异常

影响标志位:

无

说明:

读取 half word 数据后无符号扩展至当前 sew 长度。

当 vm=1 时, 指令未被 mask, 对应汇编指令 vlxseg<nf>hu.v vd, (rs1), vs2

当 vm=0 时, 指令被 mask, 对应汇编指令 vlxseg<nf>hu.v vd, (rs1), vs2, v0.t

指令格式:

31	29	28	26	25	24	20	19	15	14	12	11	7	6	0
nf-1			011	vm	00000			rs1	101	vd			0000111	

14.7.155 VLXSEG<NF>W.V——矢量索引 SEGMENT 有符号字加载指令

语法:

vlxseg<nf>w.v vd, (rs1), vs2, vm

操作:

```
for( k=0; k<=nf-1; k++) {
    Vd+k[i] = mem[rs1 + vs2[i] + 4*k]
}
```

nf 指定每个 segment 中 field 的数量和目的寄存器的数量, nf 取值范围为 2~8。如当 nf=4, vd=8, rs1=5, vs2=16, vm=1, lmul=1 时, 指令为 vlxseg4w.v v8, (x5), v16 那么指令操作如下:

```
v8[0]=mem[x5+v16[0] ] ... v8[i]=mem[x5+vs16[i] ]
v9[0]=mem[x5+v16[0]+4] ... v9[i]=mem[x5+vs16[i]+4]
v10[0]=mem[x5+v16[0]+8] ... v10[i]=mem[x5+vs16[i]+8]
v11[0]=mem[x5+v16[0]+12] ... v11[i]=mem[x5+vs16[i]+12]
```

执行权限:

M mode/S mode/U mode

异常:

非法指令异常, 加载指令非对齐访问异常, 加载指令访问错误异常, 加载指令页面错误异常

影响标志位:

无

说明:

读取 word 数据后有符号扩展至当前 sew 长度

当 vm=1 时, 指令未被 mask, 对应汇编指令 vlxseg<nf>w.v vd, (rs1), vs2

当 vm=0 时, 指令被 mask, 对应汇编指令 vlxseg<nf>w.v vd, (rs1), vs2, v0.t

指令格式:

31	29	28	26	25	24	20	19	15	14	12	11	7	6	0
nf-1		111	vm		00000		rs1		110		vd		0000111	

14.7.156 VLXSEG<NF>WU.V——矢量索引 SEGMENT 无符号字加载指令

语法:

vlxseg<nf>wu.v vd, (rs1), vs2, vm

操作:

```
for( k=0; k<=nf-1; k++) {
    Vd+k[i] = mem[rs1 + vs2[i] + 4*k]
}
```

nf 指定每个 segment 中 field 的数量和目的寄存器的数量, nf 取值范围为 2~8。如当 nf=4, vd=8, rs1=5, vs2=16, vm=1, lmul=1 时, 指令为 vlxseg4wu.v v8, (x5), v16 那么指令操作如下:

```
v8[0]=mem[x5+v16[0] ] ... v8[i]=mem[x5+vs16[i] ]
v9[0]=mem[x5+v16[0]+4] ... v9[i]=mem[x5+vs16[i]+4]
v10[0]=mem[x5+v16[0]+8] ... v10[i]=mem[x5+vs16[i]+8]
v11[0]=mem[x5+v16[0]+12] ... v11[i]=mem[x5+vs16[i]+12]
```

执行权限:

M mode/S mode/U mode

异常:

非法指令异常, 加载指令非对齐访问异常, 加载指令访问错误异常, 加载指令页面错误异常

影响标志位:

无

说明:

读取 word 数据后无符号扩展至当前 sew 长度

当 vm=1 时, 指令未被 mask, 对应汇编指令 vlxseg<nf>wu.v vd, (rs1), vs2

当 vm=0 时, 指令被 mask, 对应汇编指令 vlxseg<nf>wu.v vd, (rs1), vs2,v0.t

指令格式:

31	29	28	26	25	24	20	19	15	14	12	11	7	6	0
nf-1		011	vm		00000		rs1		110		vd		0000111	

14.7.157 VLXW.V——矢量索引有符号字加载指令

语法:

`vlxw.v vd, (rs1), vs2, vm`

操作:

$vd[i] = \text{sign_extend}(\text{mem}[\text{rs1} + \text{sign_extend}(\text{vs2}[i])])$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常, 加载指令非对齐访问异常, 加载指令访问错误异常, 加载指令页面错误异常

影响标志位:

无

说明:

读取 word 数据后有符号扩展至当前 sew 长度。

当 $vm=1$ 时, 指令未被 mask, 对应汇编指令 `vlxw.v vd, (rs1), vs2`。

当 $vm=0$ 时, 指令被 mask, 对应汇编指令 `vlxw.v vd, (rs1), vs2, v0.t`。

指令格式:

31	29	28	26	25	24	20	19	15	14	12	11	7	6	0
000	111	vm	vs2	rs1	110	vd	0000111							

14.7.158 VLXWU.V——矢量索引无符号字加载指令

语法:

`vlxwu.v vd, (rs1), vs2, vm`

操作:

$vd[i] = \text{zero_extend}(\text{mem}[\text{rs1} + \text{sign_extend}(\text{vs2}[i])])$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常, 加载指令非对齐访问异常, 加载指令访问错误异常, 加载指令页面错误异常

影响标志位:

无

说明:

加载 word 数据后无符号扩展至当前 sew 长度。

当 vm=1 时, 指令未被 mask, 对应汇编指令 vlxwu.v vd, (rs1), vs2。

当 vm=0 时, 指令被 mask, 对应汇编指令 vlxwu.v vd, (rs1), vs2, v0.t。

指令格式:

31	29	28	26	25	24	20	19	15	14	12	11	7	6	0
000	011	vm	vs2	rs1	110	vd	0000111							

14.7.159 VMACC.VV——覆盖加数矢量低位乘累加指令

语法:

vmacc.vv vd, vs1, vs2, vm

操作:

$vd[i] \leftarrow \text{low_half}(vs1[i] \times vs2[i]) + vd[i];$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

无

说明:

当 vm=1 时, 指令未被 mask, 对应汇编指令 vmacc.vv vd, vs2, vs1。

当 vm=0 时, 指令被 mask, 对应汇编指令 vmacc.vv vd, vs2, vs1, v0.t。

指令格式:

31	26	25	24	20	19	15	14	12	11	7	6	0
101101	vm	vs2	vs1	010	vd	1010111						

14.7.160 VMACC.VX——覆盖加数矢量标量低位乘累加指令

语法:

vmacc.vx vd, rs1, vs2, vm

操作:

$vd[i] \leftarrow \text{low_half}(rs1 \times vs2[i]) + vd[i]$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

无

说明:

当 vm=1 时, 指令未被 mask, 对应汇编指令 vmacc.vx vd, rs1, vs2。

当 vm=0 时, 指令被 mask, 对应汇编指令 vmacc.vx vd, rs1, vs2, v0.t。

指令格式:

31	26	25	24	20	19	15	14	12	11	7	6	0
101101			vm	vs2		rs1		110		vd		1010111

14.7.161 VMADC.VVM——矢量整型带进位加法取进位指令

语法:

vmadc.vvm vd, vs2, vs1, v0

操作:

$vd[i] \leftarrow \text{carry_out}(vs2[i] + vs1[i] + v0[i].\text{LSB})$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

无

指令格式:

31	26	25	24	20	19	15	14	12	11	7	6	0
010001	1	vs2		vs1		000		vd		1010111		

14.7.162 VMADC.VXM——矢量标量整型带进位加法取进位指令

语法:

vmadc.vxm vd, vs2, rs1, v0

操作:

$vd[i] \leftarrow \text{carry_out}(vs2[i] + rs1 + v0[i].\text{LSB})$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

无

指令格式:

31	26	25	24	20	19	15	14	12	11	7	6	0
010001	1	vs2		rs1		100		vd		1010111		

14.7.163 VMADC.VIM——矢量立即数整型带进位加法取进位指令

语法:

vmadc.vim vd, vs2, imm, v0

操作:

$vd[i] \leftarrow \text{carry_out}(vs2[i] + \text{sign_extend}(\text{imm}) + v0[i].\text{LSB})$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

无

指令格式:

31	26	25	24	20	19	15	14	12	11	7	6	0
010001		1	vs2		imm		011		vd		1010111	

14.7.164 VMADD.VV——覆盖被乘数矢量低位乘累加指令

语法:

vmadd.vv vd, vs1, vs2, vm

操作:

$vd[i] \leftarrow \text{low_half}(vs1[i] \times vs2[i] + vs2[i])$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

无

说明:

当 vm=1 时, 指令未被 mask, 对应汇编指令 vmadd.vv vd, vs2, vs1。

当 vm=0 时, 指令被 mask, 对应汇编指令 vmadd.vv vd, vs2, vs1, v0.t。

指令格式:

31	26	25	24	20	19	15	14	12	11	7	6	0
101001		vm	vs2		vs1		010		vd		1010111	

14.7.165 VMADD.VX——覆盖被乘数矢量标量低位乘累加指令

语法:

vmadd.vx vd, rs1, vs2, vm

操作:

$vd[i] \leftarrow \text{low_half}(rs1 \times vs2[i] + vs2[i])$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

无

说明:

当 vm=1 时, 指令未被 mask, 对应汇编指令 vmadd.vx vd, rs1, vs2。

当 vm=0 时, 指令被 mask, 对应汇编指令 vmadd.vx vd, rs1, vs2, v0.t。

指令格式:

31	26	25	24	20	19	15	14	12	11	7	6	0
101001	vm			vs2			rs1	110			vd	1010111

14.7.166 VMAND.MM——矢量掩码运算与指令

语法:

vmand.mm vd, vs2, vs1

操作:

$vd[i] \leftarrow \text{zero-extend}(vs2[i].\text{LSB} \&\& vs1[i].\text{LSB})$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

无

说明:

vs2,vs1 按照 mask 方式排放参与运算

指令格式:

31	26	25	24	20	19	15	14	12	11	7	6	0
011001	1			vs2			vs1	010			vd	1010111

14.7.167 VMANDNOT.MM——矢量掩码运算非与指令

语法:

vmandnot.mm vd, vs2, vs1

操作:

$vd[i] \leftarrow \text{zero-extend}(vs2[i].\text{LSB} \&\& !vs1[i].\text{LSB})$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

无

说明:

vs2,vs1 按照 mask 方式排放参与运算

指令格式:

31	26	25	24	20	19	15	14	12	11	7	6	0
011000	1	vs2	vs1	010	vd	1010111						

14.7.168 VMAX.VV——矢量整型有符号取最大值指令

语法:

vmax.vv vd, vs2, vs1, vm

操作:

$vd[i] \leftarrow \max(\text{signed}(vs2[i]), \text{signed}(vs1[i]))$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

无

说明:

当 $vm=1$ 时, 指令未被 mask, 对应汇编指令 `vmax.vv vd, vs2, vs1`。

当 $vm=0$ 时, 指令被 mask, 对应汇编指令 `vmax.vv vd, vs2, vs1, v0.t`。

指令格式:

31	26	25	24	20	19	15	14	12	11	7	6	0
000111	vm		vs2		vs1		000		vd			1010111

14.7.169 VMAX.VX——矢量标量整型有符号取最大值指令

语法:

`vmax.vx vd, vs2, rs1, vm`

操作:

$vd[i] \leftarrow \max(\text{signed}(vs2[i]), \text{signed}(rs1))$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

无

说明:

当 $vm=1$ 时, 指令未被 mask, 对应汇编指令 `vmax.vx vd, vs2, rs1`。

当 $vm=0$ 时, 指令被 mask, 对应汇编指令 `vmax.vx vd, vs2, rs1, v0.t`。

指令格式:

31	26	25	24	20	19	15	14	12	11	7	6	0
000111	vm		vs2		rs1		100		vd			1010111

14.7.170 VMAXU.VV——矢量整型无符号取最大值指令

语法:

`vmaxu.vv vd, vs2, vs1, vm`

操作:

$vd[i] \leftarrow \max(\text{unsigned}(vs2[i]), \text{unsigned}(vs1[i]))$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

无

说明:

当 $vm=1$ 时, 指令未被 mask, 对应汇编指令 `vmaxu.vv vd, vs2, vs1`。

当 $vm=0$ 时, 指令被 mask, 对应汇编指令 `vmaxu.vv vd, vs2, vs1, v0.t`。

指令格式:

31	26	25	24	20	19	15	14	12	11	7	6	0
000110	vm			vs2			vs1	000		vd	1010111	

14.7.171 VMAXU.VX——矢量标量整型无符号取最大值指令

语法:

`vmaxu.vx vd, vs2, rs1, vm`

操作:

$vd[i] \leftarrow \max(\text{unsigned}(vs2[i]), \text{unsigned}(rs1))$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

无

说明:

当 $vm=1$ 时, 指令未被 mask, 对应汇编指令 `vmaxu.vx vd, vs2, rs1`。

当 $vm=0$ 时, 指令被 mask, 对应汇编指令 `vmaxu.vx vd, vs2, rs1, v0.t`。

指令格式:

31	26	25	24	20	19	15	14	12	11	7	6	0
000110		vm	vs2		rs1		100		vd		1010111	

14.7.172 VMERGE.VVM——矢量元素选择指令

语法:

vmerge.vvm vd, vs2, vs1, v0

操作:

$vd[i] \leftarrow v0[i].LSB ? vs1[i] : vs2[i]$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

无

指令格式:

31	26	25	24	20	19	15	14	12	11	7	6	0
010111		0	vs2		vs1		000		vd		1010111	

14.7.173 VMERGE.VXM——矢量标量元素选择指令

语法:

vmerge.vxm vd, vs2, rs1, v0

操作:

$vd[i] \leftarrow v0[i].LSB ? rs1 : vs2[i]$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

无

说明:

该指令默认为 mask 类型指令

指令格式:

31	26	25	24	20	19	15	14	12	11	7	6	0
010111	0	vs2			rs1		100	vd		1010111		

14.7.174 VMERGE.VIM——矢量立即数元素选择指令

语法:

vmerge.vim vd, vs2, imm, v0

操作:

$vd[i] \leftarrow v0[i].LSB ? sign_extend(imm) : vs2[i]$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

无

指令格式:

31	26	25	24	20	19	15	14	12	11	7	6	0
010111	0	vs2			imm		011	vd		1010111		

14.7.175 VMFEQ.VF——矢量标量浮点比较相等指令

语法:

vmfeq.vf vd, vs2, fs1,vm

操作:

if(fs1 == vs2[i])

vd[i] ← 1

else

$vd[i] \leftarrow 0$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

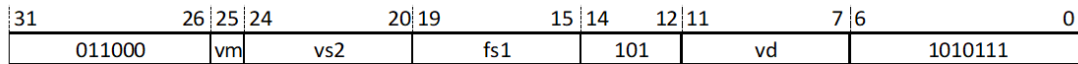
浮点状态位 NV

说明:

当 $vm=1$ 时, 指令未被 mask, 对应汇编指令 `vmfeq.vf vd, vs2, fs1`。

当 $vm=0$ 时, 指令被 mask, 对应汇编指令 `vmfeq.vf vd, vs2, fs1, v0.t`。

指令格式:



14.7.176 VMFEQ.VV——矢量浮点比较相等指令

语法:

`vmfeq.vv vd, vs2, vs1,vm`

操作:

if($vs1 == vs2[i]$)

$vd[i] \leftarrow 1$

else

$vd[i] \leftarrow 0$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

浮点状态位 NV

说明:

当 $vm=1$ 时, 指令未被 mask, 对应汇编指令 `vmfeq.vv vd, vs2, vs1`。

当 $vm=0$ 时, 指令被 mask, 对应汇编指令 `vmfeq.vv vd, vs2, vs1, v0.t`。

指令格式:

31	26	25	24	20	19	15	14	12	11	7	6	0
011000	vm		vs2		vs1		001		vd			1010111

14.7.177 VMFGE.VF——矢量标量浮点比较大于等于指令

语法:

`vmfge.vf vd, vs2, fs1, vm`

操作:

```
if(vs2[i] >= fs1)
    vd[i] ← 1
else
    vd[i] ← 0
```

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

浮点状态位 NV

说明:

当 $vm=1$ 时, 指令未被 mask, 对应汇编指令 `vmfge.vf vd, vs2, fs1`。

当 $vm=0$ 时, 指令被 mask, 对应汇编指令 `vmfge.vf vd, vs2, fs1, v0.t`。

指令格式:

31	26	25	24	20	19	15	14	12	11	7	6	0
011111	vm		vs2		fs1		101		vd			1010111

14.7.178 VMFGT.VF——矢量标量浮点比较大于指令

语法:

```
vmfgt.vf vd, vs2, fs1, vm
```

操作:

```
if(vs2[i] > fs1)
    vd[i] ← 1
else
    vd[i] ← 0
```

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

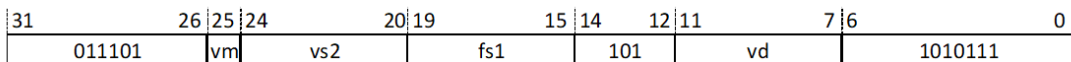
影响标志位:

浮点状态位 NV

说明:

当 vm=1 时, 指令未被 mask, 对应汇编指令 vmfgt.vf vd, vs2, fs1。
 当 vm=0 时, 指令被 mask, 对应汇编指令 vmfgt.vf vd, vs2, fs1, v0.t。

指令格式:



14.7.179 VMFIRST——矢量掩码运算快速查找 1 指令

语法:

```
vmfirst.m rd, vs2, vm
```

操作:

```
rd ← 0xffffffff
for (i=0; i < vl; i++){
    if(vs2[i].LSB == 1 && vs2[i] is active)
        rd ← i
    break
}
```

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

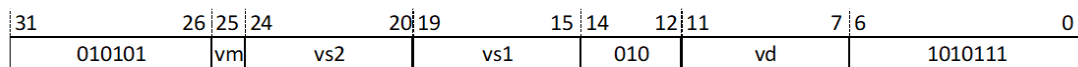
无

说明:

当 $vm=1$ 时, 指令未被 mask, 对应汇编指令 `vmfirst.m rd, vs2`。

当 $vm=0$ 时, 指令被 mask, 对应汇编指令 `vmfirst.m rd, vs2, v0.t`

指令格式:



14.7.180 VMFLE.VF——矢量标量浮点比较小于等于指令

语法:

`vmfle.vf vd, vs2, fs1, vm`

操作:

```

if(vs2[i] <= fs1)
    vd[i] ← 1
else
    vd[i] ← 0
    
```

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

浮点状态位 NV

说明:

当 $vm=1$ 时, 指令未被 mask, 对应汇编指令 `vmfle.vf vd, vs2, fs1`。

当 $vm=0$ 时, 指令被 mask, 对应汇编指令 `vmfle.vf vd, vs2, fs1, v0.t`。

指令格式:

31	26	25	24	20	19	15	14	12	11	7	6	0
011001	vm			vs2			fs1			101	vd	1010111

14.7.181 VMFLE.VV——矢量浮点比较小于等于指令

语法:

vmfle.vv vd, vs2, vs1,vm

操作:

```
if(vs2[i] <= vs1)
    vd[i] ← 1
else
    vd[i] ← 0
```

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

浮点状态位 NV

说明:

当 vm=1 时, 指令未被 mask, 对应汇编指令 vmfle.vv vd, vs2, vs1。

当 vm=0 时, 指令被 mask, 对应汇编指令 vmfle.vv vd, vs2, vs1, v0.t。

指令格式:

31	26	25	24	20	19	15	14	12	11	7	6	0
011001	vm			vs2			vs1			001	vd	1010111

14.7.182 VMFLT.VF——矢量标量浮点比较小于指令

语法:

vmflt.vf vd, vs2, fs1,vm

操作:

```
if(vs2[i] < fs1)
```

```

    vd[i] ← 1
else
    vd[i] ← 0

```

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

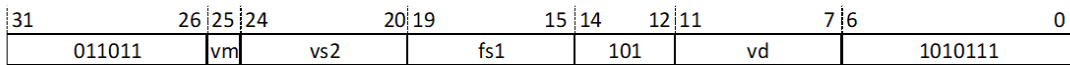
影响标志位:

浮点状态位 NV

说明:

当 $vm=1$ 时, 指令未被 mask, 对应汇编指令 `vmflt.vf vd, vs2, fs1`。
 当 $vm=0$ 时, 指令被 mask, 对应汇编指令 `vmflt.vf vd, vs2, fs1, v0.t`。

指令格式:



14.7.183 VMFLT.VV——矢量浮点比较小于指令

语法:

`vmflt.vv vd, vs2, vs1, vm`

操作:

```

if(vs2[i] < vs1)
    vd[i] ← 1
else
    vd[i] ← 0

```

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

浮点状态位 NV

说明:

当 $vm=1$ 时, 指令未被 mask, 对应汇编指令 `vmftt.vv vd, vs2, vs1`。

当 $vm=0$ 时, 指令被 mask, 对应汇编指令 `vmftt.vv vd, vs2, vs1, v0.t`。

指令格式:

31	26	25	24	20	19	15	14	12	11	7	6	0
011011	vm		vs2		vs1		001		vd			1010111

14.7.184 VMFNE.VF——矢量标量浮点比较不等指令

语法:

`vmfne.vf vd, vs2, fs1, vm`

操作:

```

if(fs1 != vs2[i])
    vd[i] ← 1
else
    vd[i] ← 0
    
```

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

浮点状态位 NV

说明:

当 $vm=1$ 时, 指令未被 mask, 对应汇编指令 `vmfne.vf vd, vs2, fs1`。

当 $vm=0$ 时, 指令被 mask, 对应汇编指令 `vmfne.vf vd, vs2, fs1, v0.t`。

指令格式:

31	26	25	24	20	19	15	14	12	11	7	6	0
011100	vm		vs2		fs1		101		vd			1010111

14.7.185 VMFNE.VV——矢量浮点比较不等指令

语法:

vmfne.vv vd, vs2, vs1,vm

操作:

```
if(vs1 != vs2[i])
    vd[i] ← 1
else
    vd[i] ← 0
```

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

浮点状态位 NV

说明:

当 vm=1 时, 指令未被 mask, 对应汇编指令 vmfne.vv vd, vs2, vs1。

当 vm=0 时, 指令被 mask, 对应汇编指令 vmfne.vv vd, vs2, vs1, v0.t。

指令格式:

31	26 25 24	20 19	15 14	12 11	7 6	0
011100	vm	vs2	vs1	001	vd	1010111

14.7.186 VMFORD.VF——矢量标量浮点 NaN 判断指令

语法:

vmford.vf vd, vs2, fs1,vm

操作:

```
if(fs1 != NaN && vs2[i] != NaN)
    vd[i] ← 1
else
    vd[i] ← 0
```

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

浮点状态位 NV

说明:

当 vm=1 时, 指令未被 mask, 对应汇编指令 vmford.vf vd, vs2, fs1。

当 vm=0 时, 指令被 mask, 对应汇编指令 vmford.vf vd, vs2, fs1, v0.t。

指令格式:

31	26	25	24	20	19	15	14	12	11	7	6	0
011010	vm		vs2		fs1		101		vd		1010111	

14.7.187 VMFORD.VV——矢量浮点 NaN 判断指令

语法:

vmford.vv vd, vs2, vs1, vm

操作:

```

if(vs1[i] != NaN && vs2[i] != NaN)
    vd[i] ← 1
else
    vd[i] ← 0
    
```

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

浮点状态位 NV

说明:

当 vm=1 时, 指令未被 mask, 对应汇编指令 vmford.vv vd, vs2, vs1。

当 vm=0 时, 指令被 mask, 对应汇编指令 vmford.vv vd, vs2, vs1, v0.t。

指令格式:

31	26	25	24	20	19	15	14	12	11	7	6	0
011010			vm	vs2		vs1		001		vd	1010111	

14.7.188 VMIN.VV——矢量整型有符号取最小值指令

语法:

vmin.vv vd, vs2, vs1, vm

操作:

$vd[i] \leftarrow \min(\text{signed}(vs2[i]), \text{signed}(vs1[i]))$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

无

说明:

当 vm=1 时, 指令未被 mask, 对应汇编指令 vmin.vv vd, vs2, vs1。

当 vm=0 时, 指令被 mask, 对应汇编指令 vmin.vv vd, vs2, vs1, v0.t。

指令格式:

31	26	25	24	20	19	15	14	12	11	7	6	0
000101			vm	vs2		vs1		000		vd	1010111	

14.7.189 VMIN.VX——矢量标量整型有符号取最小值指令

语法:

vmin.vx vd, vs2, rs1, vm

操作:

$vd[i] \leftarrow \min(\text{signed}(vs2[i]), \text{signed}(rs1))$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

无

说明:

当 $vm=1$ 时, 指令未被 mask, 对应汇编指令 `vmin.vx vd, vs2, rs1`。

当 $vm=0$ 时, 指令被 mask, 对应汇编指令 `vmin.vx vd, vs2, rs1, v0.t`。

指令格式:

31	26	25	24	20	19	15	14	12	11	7	6	0
000101	vm			vs2			rs1			100	vd	1010111

14.7.190 VMINU.VV——矢量整型无符号取最小值指令

语法:

`vminu.vv vd, vs2, vs1, vm`

操作:

$vd[i] \leftarrow \min(\text{unsigned}(vs2[i]), \text{unsigned}(vs1[i]))$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

无

说明:

当 $vm=1$ 时, 指令未被 mask, 对应汇编指令 `vminu.vv vd, vs2, vs1`。

当 $vm=0$ 时, 指令被 mask, 对应汇编指令 `vminu.vv vd, vs2, vs1, v0.t`。

指令格式:

31	26	25	24	20	19	15	14	12	11	7	6	0
000100	vm			vs2			vs1			000	vd	1010111

14.7.191 VMINU.VX——矢量标量整型无符号取最小值指令

语法:

`vminu.vx vd, vs2, rs1, vm`

操作:

$vd[i] \leftarrow \min(\text{unsigned}(vs2[i]), \text{unsigned}(rs1))$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

无

说明:

当 $vm=1$ 时, 指令未被 mask, 对应汇编指令 `vminu.vx vd, vs2, rs1`。

当 $vm=0$ 时, 指令被 mask, 对应汇编指令 `vminu.vx vd, vs2, rs1, v0.t`。

指令格式:

31	26	25	24	20	19	15	14	12	11	7	6	0
000100	vm		vs2	rs1		100		vd		1010111		

14.7.192 VMNAND.MM——矢量掩码运算与非指令

语法:

`vmnand.mm vd, vs2, vs1`

操作:

$vd[i] \leftarrow \text{zero-extend}(\!(vs2[i].\text{LSB} \&\& vs1[i].\text{LSB}))$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

无

说明:

vs2,vs1 按照 mask 方式排放参与运算

指令格式:

31	26 25 24	20 19	15 14	12 11	7 6	0
011101	1	vs2	vs1	010	vd	1010111

14.7.193 VMNOR.MM——矢量掩码运算或非指令

语法:

vmnor.mm vd, vs2, vs1

操作:

$vd[i] \leftarrow \text{zero-extend}(!vs2[i].\text{LSB} \parallel vs1[i].\text{LSB})$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

无

说明:

vs2,vs1 按照 mask 方式排放参与运算

指令格式:

31	26 25 24	20 19	15 14	12 11	7 6	0
011110	1	vs2	vs1	010	vd	1010111

14.7.194 VMOR.MM——矢量掩码运算或指令

语法:

vmor.mm vd, vs2, vs1

操作:

$vd[i] \leftarrow \text{zero-extend}(vs2[i].\text{LSB} \parallel vs1[i].\text{LSB})$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

无

说明:

vs2,vs1 按照 mask 方式排放参与运算

指令格式:

31	26	25	24	20	19	15	14	12	11	7	6	0
011010	1	vs2		vs1		010		vd		1010111		

14.7.195 VMORNOT.MM——矢量掩码运算非或指令

语法:

vmornot.mm vd, vs2, vs1

操作:

$vd[i] \leftarrow \text{zero-extend}(vs2[i].\text{LSB} \parallel !vs1[i].\text{LSB})$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

无

说明:

vs2,vs1 按照 mask 方式排放参与运算

指令格式:

31	26	25	24	20	19	15	14	12	11	7	6	0
011100	1	vs2		vs1		010		vd		1010111		

14.7.196 VMPOPC.M——矢量运算元素个数统计指令

语法:

vmpopc.m rd, vs2, vm

操作:

```

cnt = 0
for (i=0; i < vl; i++){
    if(vs2[i].LSB == 1 && vs2[i] is active)
        cnt←cnt+1
}
rd ←cnt
    
```

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

无

说明:

当 vm=1 时, 指令未被 mask, 对应汇编指令 vmpopc.m rd, vs2。

当 vm=0 时, 指令被 mask, 对应汇编指令 vmpopc.m rd, vs2, v0.t。

指令格式:

31	26	25	24	20	19	15	14	12	11	7	6	0
010100	vm	vs2		vs1		010		vd		1010111		

14.7.197 VMSBC.VVM——矢量整型带进位减法取借位指令

语法:

vmsbc.vvm vd, vs2, vs1, v0

操作:

$vd[i] \leftarrow borrow_out(vs2[i] - vs1[i] - v0[i].LSB)$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

无

指令格式:

31	26	25	24	20	19	15	14	12	11	7	6	0
010011	1	vs2		vs1		000		vd		1010111		

14.7.198 VMSBC.VXM——矢量标量整型带进位减法取借位指令

语法:

vmsbc.vxm vd, vs2, rs1, v0

操作:

$vd[i] \leftarrow borrow_out(vs2[i] - rs1 - v0[i].LSB)$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

无

指令格式:

31	26	25	24	20	19	15	14	12	11	7	6	0
010011	1	vs2		rs1		100		vd		1010111		

14.7.199 VMSBF.M——矢量掩码找 1 置位不包含指令

语法:

vmsbf.m vd, vs2, vm

操作:

index = active element && the first element in vs2 with LSB equals to 1
if (i < index)

```

    vd[i] = 1
else
    vd[i] = 0
}

```

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

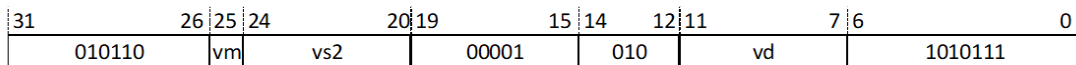
无

说明:

当 vm=1 时, 指令未被 mask, 对应汇编指令 vmsbf.m vd, vs2。

当 vm=0 时, 指令被 mask, 对应汇编指令 vmsbf.m vd, vs2, v0.t。

指令格式:



14.7.200 VMSIF.M——向量掩码找 1 置位包含指令

语法:

vmsif.m vd, vs2, vm

操作:

index = active element && the first element in vs2 with LSB equals to 1

```

if (i <=index)
    vd[i] = 1
else
    vd[i] = 0
}

```

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

无

说明:

当 vm=1 时, 指令未被 mask, 对应汇编指令 vms if.m vd, vs2。

当 vm=0 时, 指令被 mask, 对应汇编指令 vms if.m vd, vs2, v0.t。

指令格式:

31	26	25	24	20	19	15	14	12	11	7	6	0
010110	vm	vs2		00011		010		vd		1010111		

14.7.201 VMSOF.M——矢量掩码找 1 单独置位指令

语法:

vmsof.m vd, vs2, vm

操作:

index = active element && the first element in vs2 with LSB equals to 1

vd[index] = 1

vd[others] = 0

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

无

说明:

当 vm=1 时, 指令未被 mask, 对应汇编指令 vmsof.m vd, vs2。

当 vm=0 时, 指令被 mask, 对应汇编指令 vmsof.m vd, vs2, v0.t。

指令格式:

31	26	25	24	20	19	15	14	12	11	7	6	0
010110	vm	vs2		00010		010		vd		1010111		

14.7.202 VMSEQ.VV——矢量整型比较相等指令

语法:

vmseq.vv vd, vs2, vs1, vm

操作:

```
if(vs2[i] == vs1[i])
    vd[i] ← 1
else
    vd[i] ← 0
```

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

无

说明:

当 vm=1 时, 指令未被 mask, 对应汇编指令 vmseq.vv vd, vs2, vs1。

当 vm=0 时, 指令被 mask, 对应汇编指令 vmseq.vv vd, vs2, vs1, v0.t。

指令格式:

31	26	25	24	20	19	15	14	12	11	7	6	0
011000			vm	vs2		vs1		000		vd	1010111	

14.7.203 VMSEQ.VX——矢量标量整型比较相等指令

语法:

vmseq.vx vd, vs2, rs1, vm

操作:

```
if(vs2[i] == rs1)
    vd[i] ← 1
else
    vd[i] ← 0
```

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

无

说明:

当 $vm=1$ 时, 指令未被 mask, 对应汇编指令 `vmseq.vx vd, vs2, rs1`。

当 $vm=0$ 时, 指令被 mask, 对应汇编指令 `vmseq.vx vd, vs2, rs1, v0.t`。

指令格式:

31	26 25 24	20 19	15 14	12 11	7 6	0
011000	vm	vs2	rs1	100	vd	1010111

14.7.204 VMSEQ.VI——矢量立即数整型比较相等指令

语法:

`vmseq.vi vd, vs2, imm, vm`

操作:

`if(signed(vs2[i]) == signed(sign_extend(imm)))`

`vd[i] ← 1`

else

`vd[i] ← 0`

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

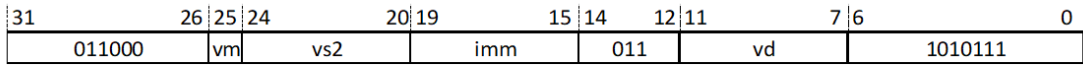
无

说明:

当 $vm=1$ 时, 指令未被 mask, 对应汇编指令 `vmseq.vi vd, vs2, imm`。

当 $vm=0$ 时, 指令被 mask, 对应汇编指令 `vmseq.vi vd, vs2, imm, v0.t`。

指令格式:



14.7.205 VMSGT.VX——矢量标量整型有符号比较大于指令

语法:

vmsgt.vx vd, vs2, rs1, vm

操作:

```
if(signed(vs2[i]) > signed(rs1))
    vd[i] ← 1
else
    vd[i] ← 0
```

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

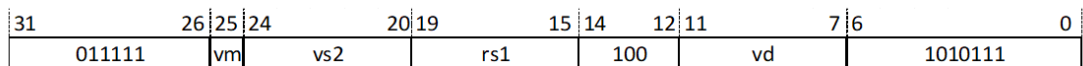
无

说明:

当 vm=1 时, 指令未被 mask, 对应汇编指令 vmsgt.vx vd, vs2, rs1。

当 vm=0 时, 指令被 mask, 对应汇编指令 vmsgt.vx vd, vs2, rs1, v0.t。

指令格式:



14.7.206 VMSGT.VI——矢量立即数整型有符号比较大于指令

语法:

vmsgt.vi vd, vs2, imm, vm

操作:

```
if(signed(vs2[i]) > signed(sign_extend(imm)))
    vd[i] ← 1
```

else
vd[i] ← 0

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

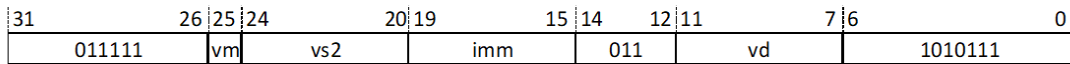
影响标志位:

无

说明:

当 vm=1 时, 指令未被 mask, 对应汇编指令 vmsgt.vi vd, vs2, imm。
当 vm=0 时, 指令被 mask, 对应汇编指令 vmsgt.vi vd, vs2, imm, v0.t

指令格式:



14.7.207 VMSGTU.VX——矢量标量整型无符号比较大于指令

语法:

vmsgtu.vx vd, vs2, rs1, vm

操作:

if(unsigned(vs2[i]) > unsigned(rs1))
vd[i] ← 1
else
vd[i] ← 0

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

无

说明:

当 $vm=1$ 时, 指令未被 mask, 对应汇编指令 `vmsgtu.vx vd, vs2, rs1`。

当 $vm=0$ 时, 指令被 mask, 对应汇编指令 `vmsgtu.vx vd, vs2, rs1, v0.t`。

指令格式:

31	26	25	24	20	19	15	14	12	11	7	6	0	
011110			vm	vs2		rs1		100		vd		1010111	

14.7.208 VMSGTU.VI——矢量立即数整型无符号比较大于指令

语法:

`vmsgtu.vi vd, vs2, imm, vm`

操作:

```
if(unsigned(vs2[i]) > unsigned(sign_extend(imm)))
    vd[i] ← 1
else
    vd[i] ← 0
```

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

无

说明:

当 $vm=1$ 时, 指令未被 mask, 对应汇编指令 `vmsgtu.vi vd, vs2, imm`。

当 $vm=0$ 时, 指令被 mask, 对应汇编指令 `vmsgtu.vi vd, vs2, imm, v0.t`。

指令格式:

31	26	25	24	20	19	15	14	12	11	7	6	0	
011110			vm	vs2		imm		011		vd		1010111	

14.7.209 VMSLE.VV——矢量整型有符号比较小于等于指令

语法:

vmsle.vv vd, vs2, vs1, vm

操作:

```
if(signed(vs2[i]) <= signed(vs1[i]))
    vd[i] ← 1
else
    vd[i] ← 0
```

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

无

说明:

当 vm=1 时, 指令未被 mask, 对应汇编指令 vmsle.vv vd, vs2, vs1。

当 vm=0 时, 指令被 mask, 对应汇编指令 vmsle.vv vd, vs2, vs1, v0.t。

指令格式:

31	26	25	24	20	19	15	14	12	11	7	6	0
011101	vm		vs2	vs1		000		vd		1010111		

14.7.210 VMSLE.VX——矢量标量整型有符号比较小于等于指令

语法:

vmsle.vx vd, vs2, rs1, vm

操作:

```
if(signed(vs2[i]) <= signed(rs1))
    vd[i] ← 1
else
    vd[i] ← 0
```

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

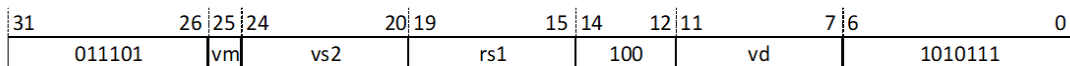
无

说明:

当 vm=1 时, 指令未被 mask, 对应汇编指令 vmsle.vx vd, vs2, rs1。

当 vm=0 时, 指令被 mask, 对应汇编指令 vmsle.vx vd, vs2, rs1, v0.t。

指令格式:



14.7.211 VMSLE.VI——矢量立即数整型有符号比较小于等于指令

语法:

vmsle.vi vd, vs2, imm, vm

操作:

if(signed(vs2[i]) <= signed(sign_extend(imm)))

vd[i] ← 1

else

vd[i] ← 0

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

无

说明:

当 vm=1 时, 指令未被 mask, 对应汇编指令 vmsle.vi vd, vs2, imm。

当 vm=0 时, 指令被 mask, 对应汇编指令 vmsle.vi vd, vs2, imm, v0.t

指令格式:

31	26	25	24	20	19	15	14	12	11	7	6	0
011101			vm	vs2			imm		011	vd		1010111

14.7.212 VMSLEU.VV——矢量整型无符号比较小于等于指令

语法:

vmsleu.vv vd, vs2, vs1, vm

操作:

```
if(unsigned(vs2[i]) <= unsigned(vs1[i]))
    vd[i] ← 1
else
    vd[i] ← 0
```

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

无

说明:

当 vm=1 时, 指令未被 mask, 对应汇编指令 vmsleu.vv vd, vs2, vs1。

当 vm=0 时, 指令被 mask, 对应汇编指令 vmsleu.vv vd, vs2, vs1, v0.t。

指令格式:

31	26	25	24	20	19	15	14	12	11	7	6	0
011100			vm	vs2	vs1	000		vd		1010111		

14.7.213 VMSLEU.VX——矢量标量整型无符号比较小于等于指令

语法:

vmsleu.vx vd, vs2, rs1, vm

操作:

```
if(unsigned(vs2[i]) <= unsigned(rs1))
    vd[i] ← 1
```

else
vd[i] ← 0

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

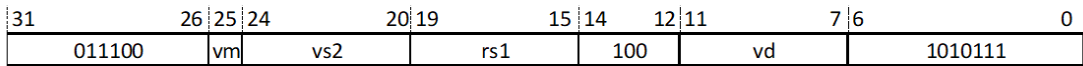
影响标志位:

无

说明:

当 vm=1 时, 指令未被 mask, 对应汇编指令 vmsleu.vx vd, vs2, rs1。
当 vm=0 时, 指令被 mask, 对应汇编指令 vmsleu.vx vd, vs2, rs1, v0.t。

指令格式:



14.7.214 VMSLEU.VI——矢量立即数整型无符号比较小于等于指令

语法:

vmsleu.vi vd, vs2, imm, vm

操作:

if(unsigned(vs2[i]) <= unsigned(sign_extend(imm)))
vd[i] ← 1
else
vd[i] ← 0

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

无

说明:

当 $vm=1$ 时, 指令未被 mask, 对应汇编指令 `vmsleu.vi vd, vs2, imm`。

当 $vm=0$ 时, 指令被 mask, 对应汇编指令 `vmsleu.vi vd, vs2, imm, v0.t`

指令格式:

31	26	25	24	20	19	15	14	12	11	7	6	0
011100		vm	vs2		imm		011		vd		1010111	

14.7.215 VMSLT.VV——矢量整型有符号比较小于指令

语法:

`vmslt.vv vd, vs2, vs1, vm`

操作:

`if(signed(vs2[i]) < signed(vs1[i]))`

`vd[i] ← 1`

else

`vd[i] ← 0`

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

无

说明:

当 $vm=1$ 时, 指令未被 mask, 对应汇编指令 `vmslt.vv vd, vs2, vs1`。

当 $vm=0$ 时, 指令被 mask, 对应汇编指令 `vmslt.vv vd, vs2, vs1, v0.t`。

指令格式:

31	26	25	24	20	19	15	14	12	11	7	6	0
011011		vm	vs2		vs1		000		vd		1010111	

14.7.216 VMSLT.VX——矢量标量整型有符号比较小于指令

语法:

vmslt.vx vd, vs2, rs1, vm

操作:

```
if(signed(vs2[i]) < signed(rs1))
    vd[i] ← 1
else
    vd[i] ← 0
```

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

无

说明:

当 vm=1 时, 指令未被 mask, 对应汇编指令 vmslt.vx vd, vs2, rs1。

当 vm=0 时, 指令被 mask, 对应汇编指令 vmslt.vx vd, vs2, rs1, v0.t。

指令格式:

31	26	25	24	20	19	15	14	12	11	7	6	0
011011	vm	vs2		rs1		100		vd		1010111		

14.7.217 VMSLTU.VV——矢量整型有无符号比较小于指令

语法:

vmsltu.vv vd, vs2, vs1, vm

操作:

```
if(unsigned(vs2[i]) < unsigned(vs1[i]))
    vd[i] ← 1
else
    vd[i] ← 0
```

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

无

说明:

当 $vm=1$ 时, 指令未被 mask, 对应汇编指令 `vmsltu.vv vd, vs2, vs1`。

当 $vm=0$ 时, 指令被 mask, 对应汇编指令 `vmsltu.vv vd, vs2, vs1, v0.t`。

指令格式:

31	26	25	24	20	19	15	14	12	11	7	6	0
011010			vm	vs2		vs1		000		vd		1010111

14.7.218 VMSLTU.VX——矢量标量整型无符号比较小于指令

语法:

`vmsltu.vx vd, vs2, rs1, vm`

操作:

`if(signed(vs2[i]) < signed(rs1))`

`vd[i] ← 1`

else

`vd[i] ← 0`

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

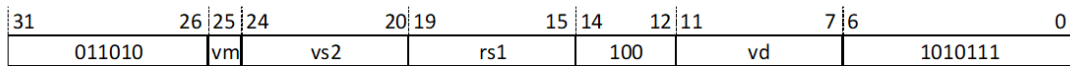
无

说明:

当 $vm=1$ 时, 指令未被 mask, 对应汇编指令 `vmsltu.vx vd, vs2, rs1`。

当 $vm=0$ 时, 指令被 mask, 对应汇编指令 `vmsltu.vx vd, vs2, rs1, v0.t`。

指令格式:



14.7.219 VMSNE.VV——矢量整型比较不等指令

语法:

vmsne.vv vd, vs2, vs1, vm

操作:

```
if(vs2[i] != vs1[i])
    vd[i] ← 1
else
    vd[i] ← 0
```

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

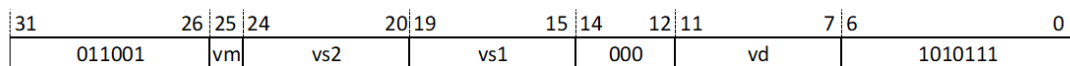
无

说明:

当 vm=1 时, 指令未被 mask, 对应汇编指令 vmsne.vv vd, vs2, vs1。

当 vm=0 时, 指令被 mask, 对应汇编指令 vmsne.vv vd, vs2, vs1, v0.t。

指令格式:



14.7.220 VMSNE.VX——矢量标量整型比较不等指令

语法:

vmsne.vx vd, vs2, rs1, vm

操作:

```
if(vs2[i] != rs1)
    vd[i] ← 1
```

else
 $vd[i] \leftarrow 0$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

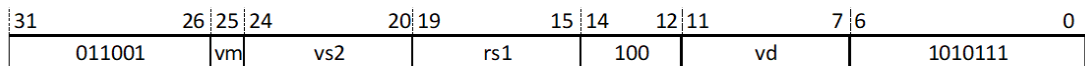
影响标志位:

无

说明:

当 $vm=1$ 时, 指令未被 mask, 对应汇编指令 `vmsne.vx vd, vs2, rs1`。
 当 $vm=0$ 时, 指令被 mask, 对应汇编指令 `vmsne.vx vd, vs2, rs1, v0.t`。

指令格式:



14.7.221 VMSNE.VI——矢量立即数整型比较不等指令

语法:

`vmsne.vi vd, vs2, imm, vm`

操作:

$if(\text{signed}(vs2[i]) \neq \text{signed}(\text{sign_extend}(\text{imm})))$
 $vd[i] \leftarrow 1$
 else
 $vd[i] \leftarrow 0$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

无

说明:

当 $vm=1$ 时, 指令未被 mask, 对应汇编指令 `vmsne.vi vd, vs2, imm`。

当 $vm=0$ 时, 指令被 mask, 对应汇编指令 `vmsne.vi vd, vs2, imm, v0.t`。

指令格式:

31	26	25	24	20	19	15	14	12	11	7	6	0
011001			vm	vs2		imm		011		vd		1010111

14.7.222 VMUL.VV——矢量整型乘法取低位指令

语法:

`vmul.vv vd, vs2, vs1, vm`

操作:

$vd[i] \leftarrow \text{low_half}(vs2[i] \times vs1[i])$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

无

说明:

当 $vm=1$ 时, 指令未被 mask, 对应汇编指令 `vmul.vv vd, vs2, vs1`。

当 $vm=0$ 时, 指令被 mask, 对应汇编指令 `vmul.vv vd, vs2, vs1, v0.t`

指令格式:

31	26	25	24	20	19	15	14	12	11	7	6	0
100101			vm	vs2		vs1		010		vd		1010111

14.7.223 VMUL.VX——矢量标量整型乘法取低位指令

语法:

`vmul.vx vd, vs2, rs1, vm`

操作:

$vd[i] \leftarrow \text{low_half}(vs2[i] \times rs1)$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

无

说明:

当 vm=1 时, 指令未被 mask, 对应汇编指令 vmul.vx vd, vs2, rs1。

当 vm=0 时, 指令被 mask, 对应汇编指令 vmul.vx vd, vs2, rs1, v0.t

指令格式:

31	26	25	24	20	19	15	14	12	11	7	6	0
100101		vm	vs2		rs1		110		vd		1010111	

14.7.224 VMULH.VV——矢量整型有符号乘法取高位指令

语法:

vmulh.vv vd, vs2, vs1, vm

操作:

$vd[i] \leftarrow \text{high_half}(\text{signed}(vs2[i]) \times \text{signed}(vs1[i]))$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

无

说明:

当 vm=1 时, 指令未被 mask, 对应汇编指令 vmulh.vv vd, vs2, vs1。

当 vm=0 时, 指令被 mask, 对应汇编指令 vmulh.vv vd, vs2, vs1, v0.t。

指令格式:

31	26	25	24	20	19	15	14	12	11	7	6	0
100111		vm	vs2		vs1		010		vd		1010111	

14.7.225 VMULH.VX——矢量标量有符号乘法取高位指令

语法:

`vmulh.vx vd, vs2, rs1, vm`

操作:

$vd[i] \leftarrow \text{high_half}(\text{signed}(vs2[i]) \times \text{signed}(rs1))$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

无

说明:

当 `vm=1` 时, 指令未被 mask, 对应汇编指令 `vmulh.vx vd, vs2, rs1`。

当 `vm=0` 时, 指令被 mask, 对应汇编指令 `vmulh.vx vd, vs2, rs1, v0.t`

指令格式:

31	26	25	24	20	19	15	14	12	11	7	6	0
100111	vm		vs2	rs1		110		vd		1010111		

14.7.226 VMULHU.VV——矢量整型无符号乘法取高位指令

语法:

`vmulhu.vv vd, vs2, vs1, vm`

操作:

$vd[i] \leftarrow \text{high_half}(\text{unsigned}(vs2[i]) \times \text{unsigned}(vs1[i]))$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

无

说明:

当 vm=1 时, 指令未被 mask, 对应汇编指令 vmulhu.vv vd, vs2, vs1。

当 vm=0 时, 指令被 mask, 对应汇编指令 vmulhu.vv vd, vs2, vs1, v0.t。

指令格式:

31	26	25	24	20	19	15	14	12	11	7	6	0
100100	vm		vs2	vs1		010		vd		1010111		

14.7.227 VMULHU.VX——矢量标量整型无符号乘法取高位指令

语法:

vmulhu.vx vd, vs2, rs1, vm

操作:

$vd[i] \leftarrow \text{high_half}(\text{unsigned}(vs2[i]) \times \text{unsigned}(rs1))$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

无

说明:

当 vm=1 时, 指令未被 mask, 对应汇编指令 vmulhu.vx vd, vs2, rs1。

当 vm=0 时, 指令被 mask, 对应汇编指令 vmulhu.vx vd, vs2, rs1, v0.t。

指令格式:

31	26	25	24	20	19	15	14	12	11	7	6	0
100100	vm		vs2	rs1		110		vd		1010111		

14.7.228 VMULHSU.VV——矢量整型有符号乘无符号取高位指令

语法:

`vmulhsu.vv vd, vs2, vs1, vm`

操作:

$vd[i] \leftarrow \text{high_half}(\text{signed}(vs2[i]) \times \text{unsigned}(vs1[i]))$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

无

说明:

当 `vm=1` 时, 指令未被 mask, 对应汇编指令 `vmulhsu.vv vd, vs2, vs1`。

当 `vm=0` 时, 指令被 mask, 对应汇编指令 `vmulhsu.vv vd, vs2, vs1, v0.t`。

指令格式:

31	26	25	24	20	19	15	14	12	11	7	6	0
100110			vm	vs2		vs1		010		vd		1010111

14.7.229 VMULHSU.VX——矢量标量整型有符号乘无符号取高位指令

语法:

`vmulhsu.vx vd, vs2, rs1, vm`

操作:

$vd[i] \leftarrow \text{high_half}(\text{signed}(vs2[i]) \times \text{unsigned}(rs1))$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

无

说明:

当 $vm=1$ 时, 指令未被 mask, 对应汇编指令 `vmulhsu.vx vd, vs2, rs1`。

当 $vm=0$ 时, 指令被 mask, 对应汇编指令 `vmulhsu.vx vd, vs2, rs1, v0.t`。

指令格式:

31	26	25	24	20	19	15	14	12	11	7	6	0
100110	vm		vs2		rs1		110		vd		1010111	

14.7.230 VMV.V.V——矢量元素传送指令

语法:

`vmv.vv vd, vs1`

操作:

$vd[i] \leftarrow vs1[i]$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

无

指令格式:

31	26	25	24	20	19	15	14	12	11	7	6	0
010111	1		00000		vs1		000		vd		1010111	

14.7.231 VMV.V.X——整型传送至矢量指令

语法:

`vmv.v.x vd, rs1`

操作:

$vd[i] \leftarrow rs1$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

无

指令格式:

31	26	25	24	20	19	15	14	12	11	7	6	0
010111	1	00000			rs1	100		vd		1010111		

14.7.232 VMV.V.I——立即数传送至矢量指令

语法:

vmv.v.i vd, imm

操作:

vd[i] ← sign_extend(imm)

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

无

指令格式:

31	26	25	24	20	19	15	14	12	11	7	6	0
010111	1	00000			imm	011		vd		1010111		

14.7.233 VMV.S.X——整型传送至矢量首元素指令

语法:

vmv.s.x vd, rs1

操作:

vd[0] = rs1

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

无

指令格式:

31	26	25	24	20	19	15	14	12	11	7	6	0
001101	1	00000		rs1		110		vd		1010111		

14.7.234 VMXOR.MM——矢量掩码运算异或指令

语法:

vmxor.mm vd, vs2, vs1

操作:

$vd[i] \leftarrow \text{zero-extend}(vs2[i].\text{LSB} \wedge vs1[i].\text{LSB})$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

无

说明:

vs2,vs1 按照 mask 方式排放参与运算

指令格式:

31	26	25	24	20	19	15	14	12	11	7	6	0
011011	1	vs2		vs1		010		vd		1010111		

14.7.235 VMXNOR.MM——矢量掩码运算同或指令

语法:

`vmxnor.mm vd, vs2, vs1`

操作:

$vd[i] \leftarrow \text{zero-extend}(!(\text{vs2}[i].\text{LSB} \wedge \text{vs1}[i].\text{LSB}))$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

无

说明:

`vs2,vs1` 按照 `mask` 方式排放参与运算

指令格式:

31	26	25	24	20	19	15	14	12	11	7	6	0
011111	1	vs2	vs1	010	vd	1010111						

14.7.236 VNCLIP.VV——矢量带饱和有符号缩位算术右移指令

语法:

`vnclip.vv vd, vs2, vs1, vm`

操作:

$vd[i] \leftarrow \text{clip}(\text{sign_extend}((\text{vs2}[i]) + \text{round})) \gg \text{vs1}[i])$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

VXSAT 位

说明:

根据定点舍入模式寄存器 vxrm 确定舍入模式:

- 2' b00: 就近向大值舍入。
- 2' b01: 就近向偶数舍入。
- 2' b10: 向零舍入。
- 2' b11: 向奇数舍入。

当 vm=1 时, 指令未被 mask, 对应汇编指令 vnclip.vv vd, vs2, vs1。

当 vm=0 时, 指令被 mask, 对应汇编指令 vnclip.vv vd, vs2, vs1,v0.t。

指令格式:

31	26	25	24	20	19	15	14	12	11	7	6	0
101111	vm			vs2		vs1		000		vd		1010111

14.7.237 VNCLIP.VX——矢量带饱和有符号按标量缩位算术右移指令

语法:

vnclip.vx vd, vs2, rs1, vm

操作:

vd[i] ← clip(sign_extend((vs2[i]) + round)) >> rs1

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

无

说明:

根据定点舍入模式寄存器 vxrm 确定舍入模式:

- 2' b00: 就近向大值舍入。
- 2' b01: 就近向偶数舍入。
- 2' b10: 向零舍入。
- 2' b11: 向奇数舍入。

当 vm=1 时, 指令未被 mask, 对应汇编指令 vnclip.vx vd, vs2, rs1。

当 vm=0 时, 指令被 mask, 对应汇编指令 vnclip.vx vd, vs2, rs1,v0.t。

指令格式:

31	26	25	24	20	19	15	14	12	11	7	6	0
101111	vm			vs2		rs1		100		vd		1010111

14.7.238 VNCLIP.VI——矢量带饱和有符号按立即数缩位算术右移指令

语法:

`vnclip.vi vd, vs2, imm, vm`

操作:

$vd[i] \leftarrow clip(sign_extend((vs2[i] + round)) \gg unsigned(imm))$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

无

说明:

根据定点舍入模式寄存器 `vxrm` 确定舍入模式:

- 2' b00: 就近向大值舍入。
- 2' b01: 就近向偶数舍入。
- 2' b10: 向零舍入。
- 2' b11: 向奇数舍入。

当 `vm=1` 时, 指令未被 `mask`, 对应汇编指令 `vnclip.vi vd, vs2, imm`。

当 `vm=0` 时, 指令被 `mask`, 对应汇编指令 `vnclip.vi vd, vs2, imm,v0.t`。

指令格式:

31	26	25	24	20	19	15	14	12	11	7	6	0
101111	vm	vs2		imm		011		vd		1010111		

14.7.239 VNCLIPU.VV——矢量带饱和和无符号缩位算术右移指令

语法:

`vnclipu.vv vd, vs2, vs1, vm`

操作:

$vd[i] \leftarrow clipu(zero_extend((vs2[i] + round)) \gg vs1[i])$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

VXSAT 位

说明:

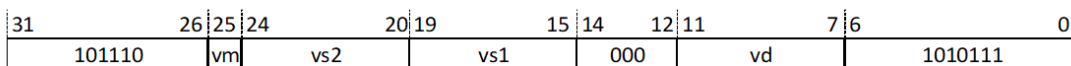
根据定点舍入模式寄存器 vxrm 确定舍入模式:

- 2' b00: 就近向大值舍入。
- 2' b01: 就近向偶数舍入。
- 2' b10: 向零舍入。
- 2' b11: 向奇数舍入。

当 vm=1 时, 指令未被 mask, 对应汇编指令 vnclipu.vv vd, vs2, vs1。

当 vm=0 时, 指令被 mask, 对应汇编指令 vnclipu.vv vd, vs2, vs1,v0.t。

指令格式:



14.7.240 VNCLIPU.VX——矢量带饱和和无符号缩位算术右移指令

语法:

vnclipu.vx vd, vs2, rs1, vm

操作:

$vd[i] \leftarrow clipu(zero_extend((vs2[i]) + round)) \gg rs1$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

VXSAT 位

说明:

根据定点舍入模式寄存器 vxrm 确定舍入模式:

- 2' b00: 就近向大值舍入。

- 2' b01: 就近向偶数舍入。
- 2' b10: 向零舍入。
- 2' b11: 向奇数舍入。

当 vm=1 时, 指令未被 mask, 对应汇编指令 vnclipu.vx vd, vs2, rs1。

当 vm=0 时, 指令被 mask, 对应汇编指令 vnclipu.vx vd, vs2, rs1,v0.t。

指令格式:

31	26	25	24	20	19	15	14	12	11	7	6	0
101110	vm			vs2			rs1			100	vd	1010111

14.7.241 VNCLIPU.VI——矢量带饱和按立即数无符号缩位算术右移指令

语法:

vnclipu.vi vd, vs2, imm, vm

操作:

$vd[i] \leftarrow clipu(zero_extend((vs2[i] + round)) \gg unsigned(imm))$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

VXSAT 位

说明:

根据定点舍入模式寄存器 vxrm 确定舍入模式:

- 2' b00: 就近向大值舍入。
- 2' b01: 就近向偶数舍入。
- 2' b10: 向零舍入。
- 2' b11: 向奇数舍入。

当 vm=1 时, 指令未被 mask, 对应汇编指令 vnclip.vi vd, vs2, imm。

当 vm=0 时, 指令被 mask, 对应汇编指令 vnclip.vi vd, vs2, imm,v0.t。

指令格式:

31	26	25	24	20	19	15	14	12	11	7	6	0
101110	vm			vs2			imm	011			vd	1010111

14.7.242 VNMSAC.VV——覆盖被减数矢量低位乘累减取负指令

语法:

`vnmsac.vv vd, vs1, vs2, vm`

操作:

$vd[i] \leftarrow -(low_half(vs1[i] \times vs2[i])) + vd[i]$

执行权限: M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

无

说明:

当 `vm=1` 时, 指令未被 mask, 对应汇编指令 `vnmsac.vv vd, vs2, vs1`。

当 `vm=0` 时, 指令被 mask, 对应汇编指令 `vnmsac.vv vd, vs2, vs1, v0.t`。

指令格式:

31	26	25	24	20	19	15	14	12	11	7	6	0
101111			vm	vs2		vs1		010		vd		1010111

14.7.243 VNMSAC.VX——覆盖被减数矢量标量低位乘累减取负指令

语法:

`vnmsac.vx vd, rs1, vs2, vm`

操作:

$vd[i] \leftarrow -(low_half(rs1 \times vs2[i])) + vd[i]$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

无

说明:

当 $vm=1$ 时, 指令未被 mask, 对应汇编指令 `vnmsac.vx vd, vs2, rs1`。

当 $vm=0$ 时, 指令被 mask, 对应汇编指令 `vnmsac.vx vd, vs2, rs1, v0.t`。

指令格式:

31	26	25	24	20	19	15	14	12	11	7	6	0
101111	vm		vs2		rs1		110		vd			1010111

14.7.244 VNMSUB.VV——覆盖被乘数矢量低位乘累减取负指令

语法:

`vnmsub.vv vd, vs1, vs2, vm`

操作:

$vd[i] \leftarrow -(low_half(vs1[i] \times vd[i]) + vs2[i])$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

无

说明:

当 $vm=1$ 时, 指令未被 mask, 对应汇编指令 `vnmsub.vv vd, vs2, vs1`。

当 $vm=0$ 时, 指令被 mask, 对应汇编指令 `vnmsub.vv vd, vs2, vs1, v0.t`。

指令格式:

31	26	25	24	20	19	15	14	12	11	7	6	0
101011	vm		vs2		vs1		010		vd			1010111

14.7.245 VNMSUB.VX——覆盖被乘数矢量标量低位乘累减取负指令

语法:

`vnmsub.vx vd, rs1, vs2, vm`

操作:

$vd[i] \leftarrow -(low_half(rs1 \times vd[i])) + vs2[i]$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

无

说明:

当 $vm=1$ 时, 指令未被 mask, 对应汇编指令 `vnmsub.vx vd, vs2, rs1`。

当 $vm=0$ 时, 指令被 mask, 对应汇编指令 `vnmsub.vx vd, vs2, rs1, v0.t`。

指令格式:

31	26	25	24	20	19	15	14	12	11	7	6	0
101011		vm	vs2		rs1		110		vd		1010111	

14.7.246 VNSRA.VV——矢量缩位算术右移指令

语法:

`vnsra.vv vd, vs2, vs1, vm`

操作:

$vd[i] \leftarrow sign_extend(vs2[i]) \gg vs1[i]$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

无

说明:

当 $vm=1$ 时, 指令未被 mask, 对应汇编指令 `vnsra.vv vd, vs2, vs1`。

当 $vm=0$ 时, 指令被 mask, 对应汇编指令 `vnsra.vv vd, vs2, vs1, v0.t`。

指令格式:

31	26 25 24	20 19	15 14	12 11	7 6	0
101101	vm	vs2	vs1	000	vd	1010111

14.7.247 VNSRA.VX——矢量标量缩位算术右移指令

语法:

vnsra.vx vd, vs2, rs1, vm

操作:

$vd[i] \leftarrow \text{sign_extend}(vs2[i]) \gg rs1$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

无

说明:

当 vm=1 时, 指令未被 mask, 对应汇编指令 vnsra.vx vd, vs2, rs1。

当 vm=0 时, 指令被 mask, 对应汇编指令 vnsra.vx vd, vs2, rs1, v0.t。

指令格式:

31	26 25 24	20 19	15 14	12 11	7 6	0
101101	vm	vs2	rs1	100	vd	1010111

14.7.248 VNSRA.VI——矢量立即数缩位算术右移指令

语法:

vnsra.vi vd, vs2, imm, vm

操作:

$vd[i] \leftarrow \text{sign_extend}(vs2[i]) \gg \text{unsigned}(imm)$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

无

说明:

当 vm=1 时, 指令未被 mask, 对应汇编指令 vnrsra.vi vd, vs2, imm。

当 vm=0 时, 指令被 mask, 对应汇编指令 vnsra.vi vd, vs2, imm, v0.t。

指令格式:

31	26	25	24	20	19	15	14	12	11	7	6	0
101101		vm	vs2	imm		011		vd		1010111		

14.7.249 VNSRL.VV——矢量缩位逻辑右移指令

语法:

vnsrl.vv vd, vs2, vs1, vm

操作:

$vd[i] \leftarrow \text{zero_extend}(vs2[i]) \gg\gg vs1[i]$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

无

说明:

当 vm=1 时, 指令未被 mask, 对应汇编指令 vnrsrl.vv vd, vs2, vs1。

当 vm=0 时, 指令被 mask, 对应汇编指令 vnsrl.vv vd, vs2, vs1, v0.t。

指令格式:

31	26	25	24	20	19	15	14	12	11	7	6	0
101100		vm	vs2	vs1		000		vd		1010111		

14.7.250 VNSRL.VX——矢量标量缩位逻辑右移指令

语法:

vnsrl.vx vd, vs2, rs1, vm

操作:

$vd[i] \leftarrow \text{zero_extend}(vs2[i]) \gg \gg rs1$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

无

说明:

当 vm=1 时, 指令未被 mask, 对应汇编指令 vnsrl.vx vd, vs2, rs1。

当 vm=0 时, 指令被 mask, 对应汇编指令 vnsrl.vx vd, vs2, rs1, v0.t。

指令格式:

31	26	25	24	20	19	15	14	12	11	7	6	0
101100			vm	vs2		rs1		100		vd		1010111

14.7.251 VNSRL.VI——矢量立即数缩位逻辑右移指令

语法:

vnsrl.vi vd, vs2, imm, vm

操作:

$vd[i] \leftarrow \text{zero_extend}(vs2[i]) \gg \gg \text{unsigned}(imm)$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

无

说明:

当 $vm=1$ 时, 指令未被 mask, 对应汇编指令 `vnrsl.vi vd, vs2, imm`。

当 $vm=0$ 时, 指令被 mask, 对应汇编指令 `vnsrl.vi vd, vs2, imm, v0.t`。

指令格式:

31	26	25	24	20	19	15	14	12	11	7	6	0
101100	vm	vs2			imm			011		vd		1010111

14.7.252 VOR.VV——矢量按位或指令

语法:

`vor.vv vd, vs2, vs1, vm`

操作:

$vd[i] \leftarrow vs2[i] | vs1[i]$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

无

说明:

当 $vm=1$ 时, 指令未被 mask, 对应汇编指令 `vor.vv vd, vs2, vs1`。

当 $vm=0$ 时, 指令被 mask, 对应汇编指令 `vor.vv vd, vs2, vs1, v0.t`。

指令格式:

31	26	25	24	20	19	15	14	12	11	7	6	0
001010	vm	vs2			vs1			000		vd		1010111

14.7.253 VOR.VX——矢量标量按位或指令

语法:

`vor.vx vd, vs2, rs1, vm`

操作:

$vd[i] \leftarrow vs2[i] \mid rs1$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

无

说明:

当 $vm=1$ 时, 指令未被 mask, 对应汇编指令 `vor.vx vd, vs2, rs1`。

当 $vm=0$ 时, 指令被 mask, 对应汇编指令 `vor.vx vd, vs2, rs1, v0.t`。

指令格式:

31	26	25	24	20	19	15	14	12	11	7	6	0	
001010			vm	vs2			rs1			100	vd	1010111	

14.7.254 VOR.VI——矢量立即数按位或指令

语法:

`vor.vi vd, vs2, imm, vm`

操作:

$vd[i] \leftarrow vs2[i] \mid \text{sign_extend}(imm)$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

无

说明:

当 $vm=1$ 时, 指令未被 mask, 对应汇编指令 `vor.vi vd, vs2, imm`。

当 $vm=0$ 时, 指令被 mask, 对应汇编指令 `vor.vi vd, vs2, imm, v0.t`。

指令格式:

31	26	25	24	20	19	15	14	12	11	7	6	0
001010	vm		vs2		imm		011		vd			1010111

14.7.255 VREDAND.VS——矢量缩减按位与指令

语法:

`vredand.vs vd, vs2, vs1, vm`

操作:

```
tmp = vs1[0]
for( i=0; i<vl; i++) {
    tmp = and(tmp, vs2[i])
}
vd[0]= tmp
vd[VLEN/SEW-1:1] = 0
```

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

无

说明:

当 $vm=1$ 时, 指令未被 mask, 对应汇编指令 `vredand.vs vd, vs2, vs1`。

当 $vm=0$ 时, 指令被 mask, 对应汇编指令 `vredand.vs vd, vs2, vs1, v0.t`。

指令格式:

31	26	25	24	20	19	15	14	12	11	7	6	0
000001	vm		vs2		vs1		010		vd			1010111

14.7.256 VREDMAX.VS——矢量缩减有符号最大值指令

语法:

vredmax.vs vd, vs2, vs1, vm

操作:

```
tmp = vs1[0]
for( i=0; i<vl; i++) {
    tmp = max(tmp, vs2[i])
}
vd[0]= tmp
vd[VLEN/SEW-1:1] = 0
```

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

无

说明:

当 vm=1 时, 指令未被 mask, 对应汇编指令 vredmax.vs vd, vs2, vs1。

当 vm=0 时, 指令被 mask, 对应汇编指令 vredmax.vs vd, vs2, vs1, v0.t。

指令格式:

31	26	25	24	20	19	15	14	12	11	7	6	0
000111	vm	vs2	vs1	010	vd	1010111						

14.7.257 VREDMAXU.VS——矢量缩减无符号最大值指令

语法:

vredmaxu.vs vd, vs2, vs1, vm

操作:

```
tmp = vs1[0]
for( i=0; i<vl; i++) {
    tmp = maxu(tmp, vs2[i])
}
```

```

}
vd[0]= tmp
vd[VLEN/SEW-1:1] = 0
    
```

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

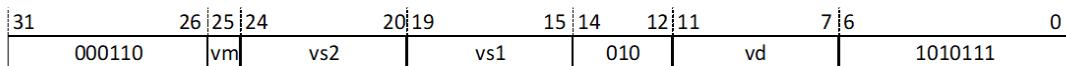
无

说明:

当 vm=1 时, 指令未被 mask, 对应汇编指令 vredmaxu.vs vd, vs2, vs1。

当 vm=0 时, 指令被 mask, 对应汇编指令 vredmaxu.vs vd, vs2, vs1, v0.t。

指令格式:



14.7.258 VREDMIN.VS——矢量缩减有符号最小值指令

语法:

vredmin.vs vd, vs2, vs1, vm

操作:

```

tmp = vs1[0]
for( i=0; i<vl; i++) {
    tmp = min(tmp, vs2[i])
}
vd[0]= tmp
vd[VLEN/SEW-1:1] = 0
    
```

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

无

说明:

当 vm=1 时, 指令未被 mask, 对应汇编指令 vredmin.vs vd, vs2, vs1。

当 vm=0 时, 指令被 mask, 对应汇编指令 vredmin.vs vd, vs2, vs1, v0.t。

指令格式:

31	26	25	24	20	19	15	14	12	11	7	6	0
000101	vm		vs2		vs1	010		vd				1010111

14.7.259 VREDMINU.VS——矢量缩减无符号最小值指令

语法:

vredmaxu.vs vd, vs2, vs1, vm

操作:

```
tmp = vs1[0]
for( i=0; i<vl; i++) {
    tmp = minu(tmp, vs2[i])
}
vd[0]= tmp
vd[VLEN/SEW-1:1] = 0
```

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

无

说明:

当 vm=1 时, 指令未被 mask, 对应汇编指令 vredminu.vs vd, vs2, vs1。

当 vm=0 时, 指令被 mask, 对应汇编指令 vredminu.vs vd, vs2, vs1, v0.t。

指令格式:

31	26	25	24	20	19	15	14	12	11	7	6	0
000101	vm		vs2		vs1	010		vd				1010111

14.7.260 VREDOR.VS——矢量缩减按位或指令

语法:

```
vredor.vs vd, vs2, vs1, vm
```

操作:

```
tmp = vs1[0]
for( i=0; i<vl; i++) {
    tmp = or(tmp, vs2[i])
}
vd[0]= tmp
vd[VLEN/SEW-1:1] = 0
```

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

无

说明:

当 vm=1 时, 指令未被 mask, 对应汇编指令 vredor.vs vd, vs2, vs1。

当 vm=0 时, 指令被 mask, 对应汇编指令 vredor.vs vd, vs2, vs1, v0.t。

指令格式:

31	26	25	24	20	19	15	14	12	11	7	6	0
000010	vm	vs2		vs1		010		vd		1010111		

14.7.261 VREDXOR.VS——矢量缩减按位异或指令

语法:

```
vredxor.vs vd, vs2, vs1, vm
```

操作:

```
tmp = vs1[0]
for( i=0; i<vl; i++) {
    tmp = xor(tmp, vs2[i])
}
```

```

}
vd[0]= tmp
vd[VLEN/SEW-1:1] = 0
    
```

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

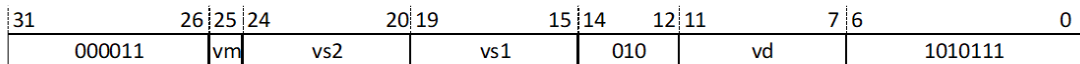
无

说明:

当 vm=1 时, 指令未被 mask, 对应汇编指令 vredxor.vs vd, vs2, vs1。

当 vm=0 时, 指令被 mask, 对应汇编指令 vredxor.vs vd, vs2, vs1, v0.t。

指令格式:



14.7.262 VREDSUM.VS——矢量缩减累加指令

语法:

vredsum.vs vd, vs2, vs1, vm

操作:

```

tmp = vs1[0]
for( i=0; i<vl; i++) {
    tmp = tmp +vs2[i]
}
vd[0]= tmp
vd[VLEN/SEW-1:1] = 0
    
```

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

无

说明:

当 $vm=1$ 时, 指令未被 mask, 对应汇编指令 `vredsum.vs vd, vs2, vs1`。

当 $vm=0$ 时, 指令被 mask, 对应汇编指令 `vredsum.vs vd, vs2, vs1, v0.t`。

指令格式:

31	26	25	24	20	19	15	14	12	11	7	6	0
000000			vm	vs2		vs1		010		vd		1010111

14.7.263 VREM.VV——矢量整型有符号取余指令

语法:

`vrem.vv vd, vs2, vs1, vm`

操作:

$vd[i] \leftarrow \text{signed}(vs2[i]) \% \text{signed}(vs1[i])$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

无

说明:

当 $vm=1$ 时, 指令未被 mask, 对应汇编指令 `vrem.vv vd, vs2, vs1`。

当 $vm=0$ 时, 指令被 mask, 对应汇编指令 `vrem.vv vd, vs2, vs1, v0.t`。

指令格式:

31	26	25	24	20	19	15	14	12	11	7	6	0
100011			vm	vs2		vs1		010		vd		1010111

14.7.264 VREM.VX——矢量标量整型有符号取余指令

语法:

vrem.vx vd, vs2, rs1, vm

操作:

$vd[i] \leftarrow \text{signed}(vs2[i]) \% \text{signed}(rs1)$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

无

说明:

当 vm=1 时, 指令未被 mask, 对应汇编指令 `vrem.vx vd, vs2, rs1`。

当 vm=0 时, 指令被 mask, 对应汇编指令 `vrem.vx vd, vs2, rs1, v0.t`。

指令格式:

31	26 25 24	20 19	15 14	12 11	7 6	0
100011	vm	vs2	rs1	110	vd	1010111

14.7.265 VREMU.VV——矢量整型无符号取余指令

语法:

vremu.vv vd, vs2, vs1, vm

操作:

$vd[i] \leftarrow \text{unsigned}(vs2[i]) \% \text{unsigned}(vs1[i])$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

无

说明:

当 $vm=1$ 时, 指令未被 mask, 对应汇编指令 `vremu.vv vd, vs2, vs1`。

当 $vm=0$ 时, 指令被 mask, 对应汇编指令 `vremu.vv vd, vs2, vs1, v0.t`。

指令格式:

31	26	25	24	20	19	15	14	12	11	7	6	0
100010	vm		vs2		vs1	010		vd		1010111		

14.7.266 VREMU.VX——矢量标量整型无符号取余指令

语法:

`vremu.vx vd, vs2, rs1, vm`

操作:

$vd[i] \leftarrow \text{unsigned}(vs2[i]) \% \text{unsigned}(rs1)$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

无

说明:

当 $vm=1$ 时, 指令未被 mask, 对应汇编指令 `vremu.vx vd, vs2, rs1`。

当 $vm=0$ 时, 指令被 mask, 对应汇编指令 `vremu.vx vd, vs2, rs1, v0.t`。

指令格式:

31	26	25	24	20	19	15	14	12	11	7	6	0
100010	vm		vs2		rs1	110		vd		1010111		

14.7.267 VRGATHER.VV——矢量整型元素根据索引聚合指令

语法:

vrgather.vv vd, vs2, vs1, vm

操作:

$vd[i] = (vs1[i] \text{ VLMAX}) ? 0 : vs2[vs1[i]]$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

无

说明:

当 vm=1 时, 指令未被 mask, 对应汇编指令 vrgather.vv vd, vs2, vs1。

当 vm=0 时, 指令被 mask, 对应汇编指令 vrgather.vv vd, vs2, vs1, v0.t。

指令格式:

31	26 25 24	20 19	15 14	12 11	7 6	0
001100	vm	vs2	vs1	000	vd	1010111

14.7.268 VRGATHER.VX——矢量标量整型元素根据索引聚合指令

语法:

vrgather.vx vd, vs2, rs1, vm

操作:

$vd[i] = (\text{unsigned}(rs1) \text{ VLAMX}) ? 0 : vs2[\text{unsigned}(rs1)]$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

无

说明:

当 vm=1 时, 指令未被 mask, 对应汇编指令 vrgather.vx vd, vs2, rs1。

当 vm=0 时, 指令被 mask, 对应汇编指令 vrgather.vx vd, vs2, rs1, v0.t。

指令格式:

31	26	25	24	20	19	15	14	12	11	7	6	0
001100	vm			vs2			rs1			100	vd	1010111

14.7.269 VRGATHER.VI——矢量立即数整型元素根据索引聚合指令

语法:

vrgather.vi vd, vs2, imm, vm

操作:

$vd[i] = (\text{unsigned}(\text{imm}) \text{ VLMAX}) ? 0 : vs2[\text{unsigned}(\text{imm})]$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

无

说明:

当 vm=1 时, 指令未被 mask, 对应汇编指令 vrgather.vi vd, vs2,imm, rs1。

当 vm=0 时, 指令被 mask, 对应汇编指令 vrgather.vi vd, vs2, imm, v0.t。

指令格式:

31	26	25	24	20	19	15	14	12	11	7	6	0
001100	vm			vs2			imm	011			vd	1010111

14.7.270 VRSUB.VX——标量矢量整型减法指令

语法:

vrsub.vx vd, vs2, rs1, vm

操作:

$vd[i] \leftarrow rs1 - vs2[i]$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

无

说明:

当 vm=1 时, 指令未被 mask, 对应汇编指令 vrsub.vx vd, vs2, rs1。

当 vm=0 时, 指令被 mask, 对应汇编指令 vrsub.vx vd, vs2, rs1, v0.t。

指令格式:

31	26	25	24	20	19	15	14	12	11	7	6	0	
000011			vm	vs2			rs1			100	vd	1010111	

14.7.271 VRSUB.VI——立即数矢量整型减法指令

语法:

vrsub.vi vd, vs2, imm, vm

操作:

$vd[i] \leftarrow \text{sign_extend}(\text{imm}) - vs2[i]$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

无

说明:

当 $vm=1$ 时, 指令未被 mask, 对应汇编指令 `vrsub.vi vd, vs2, imm`。

当 $vm=0$ 时, 指令被 mask, 对应汇编指令 `vrsub.vi vd, vs2, imm, v0.t`。

指令格式:

31	26	25	24	20	19	15	14	12	11	7	6	0
000011	vm			vs2			imm			011	vd	1010111

14.7.272 VSADD.VV——矢量整型加法有符号取饱和指令

语法:

`vsadd.vv vd, vs2, vs1, vm`

操作:

$vd[i] \leftarrow \text{sat}(vs2[i] + vs1[i])$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

VXSAT 位

说明:

当 $vm=1$ 时, 指令未被 mask, 对应汇编指令 `vsadd.vv vd, vs2, vs1`。

当 $vm=0$ 时, 指令被 mask, 对应汇编指令 `vsadd.vv vd, vs2, vs1, v0.t`。

指令格式:

31	26	25	24	20	19	15	14	12	11	7	6	0
100001	vm			vs2			vs1			000	vd	1010111

14.7.273 VSADD.VX——矢量标量整型加法有符号取饱和指令

语法:

`vsadd.vx vd, vs2, rs1, vm`

操作:

$vd[i] \leftarrow \text{sat}(vs2[i] + rs1)$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

VXSAT 位

说明:

当 $vm=1$ 时, 指令未被 mask, 对应汇编指令 `vsadd.vx vd, vs2, rs1`。

当 $vm=0$ 时, 指令被 mask, 对应汇编指令 `vsadd.vx vd, vs2, rs1, v0.t`。

指令格式:

31	26	25	24	20	19	15	14	12	11	7	6	0		
100001			vm	vs2			rs1			100		vd	1010111	

14.7.274 VSADD.VI——矢量立即数整型加法有符号取饱和指令

语法:

`vsadd.vi vd, vs2, imm, vm`

操作:

$vd[i] \leftarrow \text{sat}(vs2[i] + \text{sign_extend}(imm))$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

VXSAT 位

说明:

当 $vm=1$ 时, 指令未被 mask, 对应汇编指令 `vsadd.vi vd, vs2, imm`。

当 $vm=0$ 时, 指令被 mask, 对应汇编指令 `vsadd.vi vd, vs2, imm, v0.t`。

指令格式:

31	26	25	24	20	19	15	14	12	11	7	6	0
100001		vm	vs2		imm		011		vd		1010111	

14.7.275 VSADDU.VV——矢量整型加法无符号取饱和指令

语法:

`vsaddu.vv vd, vs2, vs1, vm`

操作:

$vd[i] \leftarrow \text{sat}(vs2[i] + vs1[i])$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

VXSAT 位

说明:

当 $vm=1$ 时, 指令未被 mask, 对应汇编指令 `vsaddu.vv vd, vs2, vs1`。

当 $vm=0$ 时, 指令被 mask, 对应汇编指令 `vsaddu.vv vd, vs2, vs1, v0.t`。

指令格式:

31	26	25	24	20	19	15	14	12	11	7	6	0
100000		vm	vs2		vs1		000		vd		1010111	

14.7.276 VSADDU.VX——矢量标量整型加法无符号取饱和指令

语法:

vsaddu.vx vd, vs2, rs1, vm

操作:

$vd[i] \leftarrow \text{sat}(vs2[i] + rs1)$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

VSAT 位

说明:

当 vm=1 时, 指令未被 mask, 对应汇编指令 vsaddu.vx vd, vs2, rs1。

当 vm=0 时, 指令被 mask, 对应汇编指令 vsaddu.vx vd, vs2, rs1, v0.t。

指令格式:

31	26	25	24	20	19	15	14	12	11	7	6	0
100000			vm	vs2		rs1		100		vd		1010111

14.7.277 VSADDU.VI——矢量立即数整型加法无符号取饱和指令

语法:

vsaddu.vi vd, vs2, imm, vm

操作:

$vd[i] \leftarrow \text{sat}(vs2[i] + \text{sign_extend}(\text{imm}))$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

VXSAT 位

说明:

当 $vm=1$ 时, 指令未被 mask, 对应汇编指令 `vsaddu.vi vd, vs2, imm`。

当 $vm=0$ 时, 指令被 mask, 对应汇编指令 `vsaddu.vi vd, vs2, imm, v0.t`。

指令格式:

31	26	25	24	20	19	15	14	12	11	7	6	0
100000	vm	vs2			imm		011		vd		1010111	

14.7.278 VSB.V——矢量字节存储指令

语法:

`vsb.v vs3, (rs1), vm`

操作:

$mem[rs1+i] = vs3[i][7:0]$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常, 存储指令非对齐访问异常、存储指令访问错误异常、存储指令页面错误异常

影响标志位:

无

说明:

当 $vm=1$ 时, 指令未被 mask, 对应汇编指令 `vsb.v vs3, (rs1)`。

当 $vm=0$ 时, 指令被 mask, 对应汇编指令 `vsb.v vs3, (rs1), v0.t`。

指令格式:

31	29	28	26	25	24	20	19	15	14	12	11	7	6	0
000	000		vm	00000			rs1		000		vs3		0100111	

14.7.279 VSBC.VVM——矢量整型带进位减法指令

语法:

vsbc.vvm vd, vs2, vs1, v0

操作:

$vd[i] \leftarrow vs2[i] - vs1[i] - v0[i].LSB$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

无

指令格式:

31	26	25	24	20	19	15	14	12	11	7	6	0		
010010			1	vs2			vs1			000		vd	1010111	

14.7.280 VSBC.VXM——矢量标量整型带进位减法指令

语法:

vsbc.vxm vd, vs2, rs1, v0

操作:

$vd[i] \leftarrow vs2[i] - rs1 - v0[i].LSB$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

无

指令格式:

31	26	25	24	20	19	15	14	12	11	7	6	0		
010010			1	vs2			rs1			100		vd	1010111	

14.7.281 VSE.V——矢量元素存储指令

语法:

vse.v vs3, (rs1), vm

操作:

mem[rs1+i*SEW/8] = vs3[i]

执行权限:

M mode/S mode/U mode

异常:

非法指令异常、存储指令非对齐访问异常、存储指令访问错误异常、存储指令页面错误异常

影响标志位:

无

说明:

当 vm=1 时, 指令未被 mask, 对应汇编指令 vse.v vs3, (rs1)。

当 vm=0 时, 指令被 mask, 对应汇编指令 vse.v vs3, (rs1), v0.t。

指令格式:

31	29	28	26	25	24	20	19	15	14	12	11	7	6	0
000	000	vm	00000			rs1		111		vd		0100111		

14.7.282 VSETVL——寄存器设置 vl 和 vtype 指令

语法:

vsetvl rd, rs1, rs2

操作:

rd =new vl, rs1=AVL, rs2=new vtype value

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

无

说明:

当 $rs1=x0$ 时, vl 将被设置成 VLMAX

当设置 SEW=128 或其他不支持的 $vtype$ 值时, VILL 位会置 1。

指令格式:

31	30	25	24	20	19	15	14	12	11	7	6	0
1	000000	rs2		rs1		111		rd		1010111		

14.7.283 VSETVLI——立即数设置 vl 和 $vtype$ 指令

语法:

`vsetvli rd, rs1, vtypei`

操作:

$rd = \text{new } vl, rs1 = \text{AVL}, vtypei = \text{new } vtype \text{ setting}$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

无

说明:

当 $rs1=x0$ 时, vl 将被设置成 VLMAX

当设置 SEW=128 或其他不支持的 $vtype$ 值时, VILL 位会置 1

指令格式:

31	30	20	19	15	14	12	11	7	6	0
1	imm			rs1		111		rd		1010111

14.7.284 VSH.V——矢量半字存储指令

语法:

vsh.v vs3, (rs1), vm

操作:

mem[rs1+2*i] = vs3[i][15:0]

执行权限:

M mode/S mode/U mode

异常:

非法指令异常、存储指令非对齐访问异常、存储指令访问错误异常、存储指令页面错误异常

影响标志位:

无

说明:

当 vm=1 时, 指令未被 mask, 对应汇编指令 vsh.v vs3, (rs1)。

当 vm=0 时, 指令被 mask, 对应汇编指令 vsh.v vs3, (rs1), v0.t。

指令格式:

31	29	28	26	25	24	20	19	15	14	12	11	7	6	0
000	000	vm	00000	rs1	101	vd	0100111							

14.7.285 VSLIDEDOWN.VX——矢量元素向下偏移指令

语法:

vslidedown.vx vd, vs2, rs1, vm

操作:

```
if( (i + unsigned(rs)) < VLMAX)
    vd[i] ← vs2[i + unsigned(rs1)]
else
    vd[i] ← 0
```

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

无

说明:

当 $vm=1$ 时, 指令未被 mask, 对应汇编指令 `vslidedown.vx vd, vs2, rs1`。

当 $vm=0$ 时, 指令被 mask, 对应汇编指令 `vslidedown.vx vd, vs2, rs1, v0.t`。

指令格式:

31	26	25	24	20	19	15	14	12	11	7	6	0
001111	vm	vs2		rs1		100		vd		1010111		

14.7.286 VSLIDEDOWN.VI——矢量元素按立即数向下偏移指令

语法:

`vslidedown.vi vd, vs2, imm, vm`

操作:

```
if( (i + unsigned(imm)) < VLMAX)
    vd[i] ← vs2[i + unsigned(imm)]
else
    vd[i] ← 0
```

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

无

说明:

当 $vm=1$ 时, 指令未被 mask, 对应汇编指令 `vslidedown.vx vd, vs2, imm`。

当 $vm=0$ 时, 指令被 mask, 对应汇编指令 `vslidedown.vx vd, vs2, imm, v0.t`。

指令格式:

31	26	25	24	20	19	15	14	12	11	7	6	0
001111	vm	vs2		imm		011		vd		1010111		

14.7.287 VSLIDE1DOWN.VX——矢量元素向下偏移 1 个索引指令

语法:

`vslide1down.vx vd, vs2, rs1, vm`

操作:

$vd[i] = vs2[i+1]$

$vd[vl-1] = rs1$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

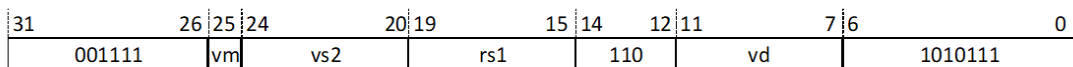
无

说明:

当 $vm=1$ 时, 指令未被 mask, 对应汇编指令 `vslide1down.vx vd, vs2, rs1`。

当 $vm=0$ 时, 指令被 mask, 对应汇编指令 `vslide1down.vx vd, vs2, rs1, v0.t`。

指令格式:



14.7.288 VSLIDEUP.VX——矢量元素向上偏移指令

语法:

`vslideup.vx vd, vs2, rs1, vm`

操作:

if($i < \text{unsigned}(rs1)$)

$vd[i] \leftarrow vd[i]$

else

$vd[i] \leftarrow vs2[i - \text{unsigned}(rs1)]$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

无

说明:

当 $vm=1$ 时, 指令未被 mask, 对应汇编指令 `vslideup.vx vd, vs2, rs1`。

当 $vm=0$ 时, 指令被 mask, 对应汇编指令 `vslideup.vx vd, vs2, rs1, v0.t`。

指令格式:

31	26	25	24	20	19	15	14	12	11	7	6	0
001110	vm	vs2			rs1			100	vd		1010111	

14.7.289 VSLIDEUP.VI——矢量元素按立即数向上偏移指令

语法:

`vslideup.vi vd, vs2, imm, vm`

操作:

```

if( (i < unsigned(imm) )
    vd[i] ← vd[i]
else
    vd[i] ← vs2[i - unsigned(imm)]
    
```

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

无

说明:

当 $vm=1$ 时, 指令未被 mask, 对应汇编指令 `vslideup.vi vd, vs2, imm`。

当 $vm=0$ 时, 指令被 mask, 对应汇编指令 `vslideup.vi vd, vs2, imm, v0.t`。

指令格式:

31	26	25	24	20	19	15	14	12	11	7	6	0
001110	vm	vs2			imm			011	vd		1010111	

14.7.290 VSLIDE1UP.VX——矢量元素向上偏移 1 个索引指令

语法:

`vslide1up.vx vd, vs2, rs1, vm`

操作:

$vd[0] = rs1$
 $vd[i+1] = vs2[i]$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

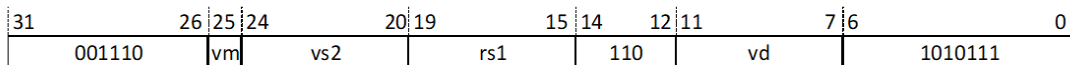
影响标志位:

无

说明:

当 $vm=1$ 时, 指令未被 mask, 对应汇编指令 `vslide1up.vx vd, vs2, rs1`。
 当 $vm=0$ 时, 指令被 mask, 对应汇编指令 `vslide1up.vx vd, vs2, rs1, v0.t`。

指令格式:



14.7.291 VSSL.VV——矢量逻辑左移指令

语法:

`vssl.vv vd, vs2, vs1, vm`

操作:

$vd[i] \leftarrow \text{unsigned}(vs2[i]) \ll vs1[i]$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

无

说明:

当 vm=1 时, 指令未被 mask, 对应汇编指令 vsll.vv vd, vs2, vs1。

当 vm=0 时, 指令被 mask, 对应汇编指令 vsll.vv vd, vs2, vs1, v0.t。

指令格式:

31	26	25	24	20	19	15	14	12	11	7	6	0
100101	vm		vs2		vs1	000		vd		1010111		

14.7.292 VSSL.VX——矢量按标量逻辑左移指令

语法:

vsll.vx vd, vs2, rs1, vm

操作:

$vd[i] \leftarrow \text{unsigned}(vs2[i]) \ll rs1$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

无

说明:

当 vm=1 时, 指令未被 mask, 对应汇编指令 vsll.vx vd, vs2, rs1。

当 vm=0 时, 指令被 mask, 对应汇编指令 vsll.vx vd, vs2, rs1, v0.t。

指令格式:

31	26	25	24	20	19	15	14	12	11	7	6	0
100101	vm		vs2		rs1	100		vd		1010111		

14.7.293 VSSL.VI——矢量按立即数逻辑左移指令

语法:

`vssl.vi vd, vs2, imm, vm`

操作:

$vd[i] \leftarrow \text{unsigned}(vs2[i]) \ll \text{unsigned}(imm)$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

无

说明:

当 `vm=1` 时, 指令未被 `mask`, 对应汇编指令 `vssl.vi vd, vs2, imm`。

当 `vm=0` 时, 指令被 `mask`, 对应汇编指令 `vssl.vi vd, vs2, imm, v0.t`。

指令格式:

31	26	25	24	20	19	15	14	12	11	7	6	0
100101			vm	vs2		imm		011		vd		1010111

14.7.294 VSMUL.VV——矢量带饱和乘法指令

语法:

`vsmul.vv vd, vs2, vs1, vm`

操作:

$vd[i] = \text{clip}((vs2[i] \times vs1[i] + \text{round}) \gg (\text{SEW}-1))$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

VXSAT 位

说明:

根据定点舍入模式寄存器 vxrm 确定舍入模式:

- 2' b00: 就近向大值舍入。
- 2' b01: 就近向偶数舍入。
- 2' b10: 向零舍入。
- 2' b11: 向奇数舍入。

当 vm=1 时, 指令未被 mask, 对应汇编指令 vsmul.vv vd, vs2, vs1。

当 vm=0 时, 指令被 mask, 对应汇编指令 vsmul.vv vd, vs2, vs1, v0.t。

指令格式:

31	26	25	24	20	19	15	14	12	11	7	6	0
100111	vm			vs2			vs1	000		vd	1010111	

14.7.295 VSMUL.VX——矢量标量带饱和乘法指令

语法:

vsmul.vx vd, vs2, rs1, vm

操作:

$vd[i] = clip((vs2[i] \times rs1 + round) >> (SEW-1))$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

VXSAT 位

说明:

根据定点舍入模式寄存器 vxrm 确定舍入模式:

- 2' b00: 就近向大值舍入。
- 2' b01: 就近向偶数舍入。
- 2' b10: 向零舍入。
- 2' b11: 向奇数舍入。

当 vm=1 时, 指令未被 mask, 对应汇编指令 vsmul.vx vd, vs2, rs1。

当 vm=0 时, 指令被 mask, 对应汇编指令 vsmul.vx vd, vs2, rs1, v0.t。

指令格式:

31	26	25	24	20	19	15	14	12	11	7	6	0
100111	vm			vs2			rs1	100			vd	1010111

14.7.296 VSRA.VV——矢量算术右移指令

语法:

vsra.vv vd, vs2, vs1, vm

操作:

$vd[i] \leftarrow \text{signed}(vs2[i]) \ggg vs1[i]$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

无

说明:

当 vm=1 时, 指令未被 mask, 对应汇编指令 vsra.vv vd, vs2, vs1。

当 vm=0 时, 指令被 mask, 对应汇编指令 vsra.vv vd, vs2, vs1, v0.t。

指令格式:

31	26	25	24	20	19	15	14	12	11	7	6	0
101001	vm			vs2			vs1	000			vd	1010111

14.7.297 VSRA.VX——矢量按标量算术右移指令

语法:

vsra.vx vd, vs2, rs1, vm

操作:

$vd[i] \leftarrow \text{signed}(vs2[i]) \ggg rs1$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

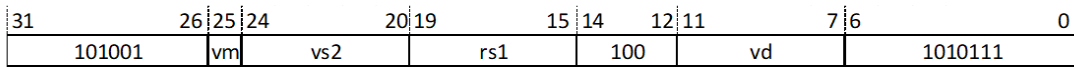
无

说明:

当 vm=1 时, 指令未被 mask, 对应汇编指令 vsra.vx vd, vs2, rs1。

当 vm=0 时, 指令被 mask, 对应汇编指令 vsra.vx vd, vs2, rs1, v0.t。

指令格式:



14.7.298 VSRA.VI——矢量按立即数算术右移指令

语法:

vsra.vi vd, vs2, imm, vm

操作:

$vd[i] \leftarrow \text{signed}(vs2[i]) \ggg \text{unsigned}(imm)$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

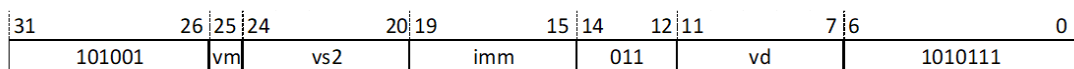
无

说明:

当 vm=1 时, 指令未被 mask, 对应汇编指令 vsra.vi vd, vs2, imm。

当 vm=0 时, 指令被 mask, 对应汇编指令 vsra.vi vd, vs2, imm, v0.t。

指令格式:



14.7.299 VSRL.VV——矢量逻辑右移指令

语法:

vsrl.vv vd, vs2, vs1, vm

操作:

$vd[i] \leftarrow \text{unsigned}(vs2[i]) \gg vs1[i]$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

无

说明:

当 vm=1 时, 指令未被 mask, 对应汇编指令 vsrl.vv vd, vs2, vs1。

当 vm=0 时, 指令被 mask, 对应汇编指令 vsrl.vv vd, vs2, vs1, v0.t。

指令格式:

31	26	25	24	20	19	15	14	12	11	7	6	0
101000	vm		vs2	vs1		000		vd		1010111		

14.7.300 VSRL.VX——矢量按标量逻辑右移指令

语法:

vsrl.vx vd, vs2, rs1, vm

操作:

$vd[i] \leftarrow \text{unsigned}(vs2[i]) \gg rs1$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

无

说明:

当 vm=1 时, 指令未被 mask, 对应汇编指令 vsrl.vx vd, vs2, rs1。

当 vm=0 时, 指令被 mask, 对应汇编指令 vsrl.vx vd, vs2, rs1, v0.t

指令格式:

31	26	25	24	20	19	15	14	12	11	7	6	0
101000	vm		vs2		rs1		100		vd			1010111

14.7.301 VSRL.VI——矢量按立即数逻辑右移指令

语法:

vsrl.vi vd, vs2, imm, vm

操作:

$vd[i] \leftarrow \text{unsigned}(vs2[i]) \gg \text{unsigned}(imm)$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

无

说明:

当 vm=1 时, 指令未被 mask, 对应汇编指令 vsrl.vi vd, vs2, imm。

当 vm=0 时, 指令被 mask, 对应汇编指令 vsrl.vi vd, vs2, imm, v0.t

指令格式:

31	26	25	24	20	19	15	14	12	11	7	6	0
101000	vm		vs2		imm		011		vd			1010111

14.7.302 VSSB.V——矢量步进字节存储指令

语法:

vssb.v vs3, (rs1), rs2, vm

操作:

mem[rs1+i*rs2] = vs3[i][7:0]

执行权限:

M mode/S mode/U mode

异常:

非法指令异常、存储指令非对齐访问异常、存储指令访问错误异常、存储指令页面错误异常

影响标志位:

无

说明:

当 vm=1 时, 指令未被 mask, 对应汇编指令 vssb.v vs3, (rs1), rs2。

当 vm=0 时, 指令被 mask, 对应汇编指令 vssb.v vs3, (rs1), rs2, v0.t。

指令格式:

31	29	28	26	25	24	20	19	15	14	12	11	7	6	0
000	010	vm	rs2			rs1			000	vd	0100111			

14.7.303 VSSE.V——矢量步进元素存储指令

语法:

vsse.v vs3, (rs1), rs2, vm

操作:

mem[rs1+i*rs2] = vs3[i][SEW-1:0]

执行权限:

M mode/S mode/U mode

异常:

非法指令异常, 存储指令非对齐访问异常、存储指令访问错误异常、存储指令页面错误异常

影响标志位:

无

说明:

当 $vm=1$ 时, 指令未被 mask, 对应汇编指令 $vsse.v\ vs3,\ (rs1),\ rs2$ 。
 当 $vm=0$ 时, 指令被 mask, 对应汇编指令 $vsse.v\ vs3,\ (rs1),\ rs2,\ v0.t$ 。

指令格式:

31	26	25	24	20	19	15	14	12	11	7	6	0
100101			vm	vs2		imm		011		vd		1010111

14.7.304 VSSEG<NF>B.V——矢量 SEGMENT 字节存储指令

语法:

$vsseg<nf>b.v\ vs3,\ (rs1),\ vm$

操作:

```
for( k=0; k<=nf-1; k++) {
    mem[rs1 + k +i *nf] = vs3d+k[i]
}
```

nf 指定每个 segment 中 field 的数量和目的寄存器的数量, nf 取值范围为 2~8。如当 $nf=4,vs3=8,rs1=5,vm=1,lmul=1$ 时, 指令为 $vsseg4b.v\ v8,\ (x5)$, 此时指令操作如下:

```
mem[x5]=v8[0], mem[x5+4]=v8[1] ... mem[x5+4*i]=v8[i]
mem[x5+1]=v9[0], mem[x5+5]=v9[1] ... mem[x5+1+4*i]=v9[i]
mem[x5+2]=v10[0], mem[x5+6]=v10[1] ... mem[x5+2+4*i]=v10[i]
mem[x5+3]=v11[0], mem[x5+7]=v11[1] ... mem[x5+3+4*i]=v11[i]
```

执行权限:

M mode/S mode/U mode

异常:

非法指令异常, 存储指令非对齐访问异常、存储指令访问错误异常、存储指令页面错误异常

影响标志位:

无

说明:

当 $vm=1$ 时, 指令未被 mask, 对应汇编指令 $vsseg<nf>b.v\ vs3,\ (rs1)$
 当 $vm=0$ 时, 指令被 mask, 对应汇编指令 $vsseg<nf>b.v\ vs3,\ (rs1),\ v0.t$

指令格式:

31	29	28	26	25	24	20	19	15	14	12	11	7	6	0
nf-1		000		vm	0000		rs1		000		vs3		0100111	

14.7.305 VSSEG<NF>E.V——矢量 SEGMENT 元素存储指令

语法:

vsseg<nf>e.v vs3, (rs1), vm

操作:

```
offset=sew/8
for( k=0; k<=nf-1; k++) {
    mem[rs1 + k*offset +i *nf*offset] = vs3d+k[i]
}
```

nf 指定每个 segment 中 field 的数量和目的寄存器的数量, nf 取值范围为 2~8。如当 nf=4,vs3=8,rs1=5,vm=1,sew=32, lmul=1 时, offset=32/8=4, 指令为 vsseg4e.v v8, (x5), 此时指令操作如下:

```
mem[x5]=v8[0], mem[x5+16]=v8[1] ... mem[x5+16*i]=v8[i]
mem[x5+4]=v9[0], mem[x5+20]=v9[1] ... mem[x5+4+16*i]=v9[i]
mem[x5+8]=v10[0], mem[x5+24]=v10[1] ... mem[x5+8+16*i]=v10[i]
mem[x5+12]=v11[0], mem[x5+28]=v11[1] ... mem[x5+12+16*i]=v11[i]
```

执行权限:

M mode/S mode/U mode

异常:

非法指令异常, 存储指令非对齐访问异常、存储指令访问错误异常、存储指令页面错误异常

影响标志位:

无

说明:

当 vm=1 时, 指令未被 mask, 对应汇编指令 vsseg<nf>e.v vs3, (rs1)

当 vm=0 时, 指令被 mask, 对应汇编指令 vsseg<nf>e.v vs3, (rs1), v0.t

指令格式:

31	29	28	26	25	24	20	19	15	14	12	11	7	6	0
nf-1		000	vm		00000		rs1		111		vs3		0100111	

14.7.306 VSSEG<NF>H.V——矢量 SEGMENT 半字存储指令

语法:

vsseg<nf>h.v vs3, (rs1), vm

操作:

```
for( k=0; k<=nf-1; k++) {
    mem[rs1 + k*2 + 2*i *nf] = vs3d+k[i]
}
```

nf 指定每个 segment 中 field 的数量和目的寄存器的数量, nf 取值范围为 2~8。如当 nf=4,vs3=8,rs1=5,vm=1,lmul=1 时, 指令为 vsseg4h.v v8, (x5), 此时指令操作如下:

```
mem[x5]=v8[0], mem[x5+8]=v8[1] ... mem[x5+8*i]=v8[i]
mem[x5+2]=v9[0], mem[x5+10]=v9[1] ... mem[x5+2+8*i]=v9[i]
mem[x5+4]=v10[0], mem[x5+12]=v10[1] ... mem[x5+4+8*i]=v10[i]
mem[x5+6]=v11[0], mem[x5+14]=v11[1] ... mem[x5+6+8*i]=v11[i]
```

执行权限:

M mode/S mode/U mode

异常:

非法指令异常, 存储指令非对齐访问异常、存储指令访问错误异常、存储指令页面错误异常

影响标志位:

无

说明:

当 vm=1 时, 指令未被 mask, 对应汇编指令 vsseg<nf>h.v vs3, (rs1)
 当 vm=0 时, 指令被 mask, 对应汇编指令 vsseg<nf>h.v vs3, (rs1), v0.t

指令格式:

31	29 28	26 25 24	20 19	15 14	12 11	7 6	0
nf-1	000	vm	00000	rs1	101	vs3	0100111

14.7.307 VSSEG<NF>W.V——矢量 SEGMENT 字存储指令

语法:

vsseg<nf>w.v vs3, (rs1), vm

操作:

```
for( k=0; k<=nf-1; k++) {
    mem[rs1 + k*4 + 4*i *nf] = vs3d+k[i]
}
```

nf 指定每个 segment 中 field 的数量和目的寄存器的数量, nf 取值范围为 2~8。如当 nf=4,vs3=8,rs1=5,vm=1,lmul=1 时, 指令为 vsseg4w.v v8, (x5), 此时指令操作如下:

mem[x5]=v8[0], mem[x5+16]=v8[1] ... mem[x5+8*i]=v8[i]
 mem[x5+4]=v9[0], mem[x5+20]=v9[1] ... mem[x5+4+8*i]=v9[i]
 mem[x5+8]=v10[0], mem[x5+24]=v10[1] ... mem[x5+8+8*i]=v10[i]
 mem[x5+12]=v11[0], mem[x5+28]=v11[1] ... mem[x5+12+8*i]=v11[i]

执行权限:

M mode/S mode/U mode

异常:

非法指令异常, 存储指令非对齐访问异常、存储指令访问错误异常、存储指令页面错误异常

影响标志位:

无

说明:

当 vm=1 时, 指令未被 mask, 对应汇编指令 vsseg<nf>.v vs3, (rs1)

当 vm=0 时, 指令被 mask, 对应汇编指令 vsseg<nf>.v vs3, (rs1), v0.t

指令格式:

31	29	28	26	25	24	20	19	15	14	12	11	7	6	0
nf-1			000		vm	00000		rs1		110		vs3		0100111

14.7.308 VSSSEG<NF>.B.V——矢量步进 SEGMENT 字节存储指令

语法:

vssseg<nf>.b.v vs3, (rs1),rs2, vm

操作:

```
for( k=0; k<=nf-1; k++) {
    mem[rs1 + k + i *nf] = vs3d+k[i]
}
```

nf 指定每个 segment 中 field 的数量和目的寄存器的数量, nf 取值范围为 2~8。如当 nf=4,vs3=8,rs1=5,rs2=6,vm=1,lmul=1 时, 指令为 vssseg4b.v v8, (x5), x6, 此时指令操作如下:

mem[x5]=v8[0], mem[x5+x6+1]=v8[1] ... mem[x5+i*x6+1]=v8[i]
 mem[x5+1]=v9[0], mem[x5+x6+2]=v9[1] ... mem[x5+i*x6+2]=v9[i]
 mem[x5+2]=v10[0], mem[x5+x6+3]=v10[1] ... mem[x5+i*x6+3]=v10[i]
 mem[x5+3]=v11[0], mem[x5+x6+4]=v11[1] ... mem[x5+i*x6+4]=v11[i]

执行权限:

M mode/S mode/U mode

异常:

非法指令异常, 存储指令非对齐访问异常、存储指令访问错误异常、存储指令页面错误异常

影响标志位:

无

说明:

当 $vm=1$ 时, 指令未被 mask, 对应汇编指令 $vssseg<nf>b.v\ vs3, (rs1),rs2$

当 $vm=0$ 时, 指令被 mask, 对应汇编指令 $vssseg<nf>b.v\ vs3, (rs1), rs2,v0.t$

指令格式:

31	29	28	26	25	24	20	19	15	14	12	11	7	6	0
nf-1		010		vm	rs2		rs1		000		vs3		0100111	

14.7.309 VSSSEG<NF>E.V——矢量步进 SEGMENT 元素存储指令

语法:

$vssseg<nf>e.v\ vs3, (rs1),rs2, vm$

操作:

$offset = sew / 32$

```
for( k=0; k<=nf-1; k++) {
    mem[rs1 + k*offset + i *nf*offset] = vs3d+k[i]
}
```

nf 指定每个 segment 中 field 的数量和目的寄存器的数量, nf 取值范围为 2~8。如当 $nf=4, vs3=8, rs1=5, rs2=6, vm=1, lmul=1, sew=32$ 时, $offset=32/8=4$, 指令为 $vssseg4e.v\ v8, (x5), x6$, 此时指令操作如下:

```
mem[x5]=v8[0], mem[x5+x6]=v8[1] ... mem[x5+i*x6]=v8[i]
mem[x5+4]=v9[0], mem[x5+x6+4]=v9[1] ... mem[x5+i*x6+4]=v9[i]
mem[x5+8]=v10[0], mem[x5+x6+8]=v10[1] ... mem[x5+i*x6+8]=v10[i]
mem[x5+12]=v11[0], mem[x5+x6+12]=v11[1] ... mem[x5+i*x6+12]=v11[i]
```

执行权限:

M mode/S mode/U mode

异常:

非法指令异常, 存储指令非对齐访问异常、存储指令访问错误异常、存储指令页面错误异常

影响标志位:

无

说明:

当 $vm=1$ 时, 指令未被 mask, 对应汇编指令 $vssseg<nf>e.v\ vs3, (rs1),rs2$

当 $vm=0$ 时, 指令被 mask, 对应汇编指令 $vssseg<nf>e.v\ vs3, (rs1), rs2,v0.t$

指令格式:

31	29	28	26	25	24	20	19	15	14	12	11	7	6	0
nf-1	010	vm	rs2			rs1		111		vs3				0100111

14.7.310 VSSSEG<NF>H.V——矢量步进 SEGMENT 半字存储指令

语法:

$vssseg<nf>h.v\ vs3, (rs1),rs2\ vm$

操作:

```
for( k=0; k<=nf-1; k++) {
    mem[rs1 +2* k +2*i *nf] = vs3d+k[i]
}
```

nf 指定每个 segment 中 field 的数量和目的寄存器的数量, nf 取值范围为 2~8。如当 $nf=4,vs3=8,rs1=5,rs2=6,vm=1,lmul=1$ 时, 指令为 $vssseg4h.v\ v8, (x5), x6$, 此时指令操作如下:

```
mem[x5]=v8[0], mem[x5+x6]=v8[1] ... mem[x5+i*x6]=v8[i]
mem[x5+2]=v9[0], mem[x5+x6+2]=v9[1] ... mem[x5+i*x6+2]=v9[i]
mem[x5+4]=v10[0], mem[x5+x6+4]=v10[1] ... mem[x5+i*x6+4]=v10[i]
mem[x5+6]=v11[0], mem[x5+x6+6]=v11[1] ... mem[x5+i*x6+6]=v11[i]
```

执行权限:

M mode/S mode/U mode

异常:

非法指令异常, 存储指令非对齐访问异常、存储指令访问错误异常、存储指令页面错误异常

影响标志位:

无

说明:

当 $vm=1$ 时, 指令未被 mask, 对应汇编指令 $vssseg<nf>h.v\ vs3, (rs1),rs2$

当 $vm=0$ 时, 指令被 mask, 对应汇编指令 $vssseg<nf>h.v\ vs3, (rs1), rs2,v0.t$

指令格式:

31	29	28	26	25	24	20	19	15	14	12	11	7	6	0
nf-1	010	vm	rs2			rs1		101		vs3				0100111

14.7.311 VSSSEG<NF>W.V——矢量步进 SEGMENT 字存储指令

语法:

vssseg<nf>w.v vs3, (rs1),rs2,vm

操作:

```
for( k=0; k<=nf-1; k++) {
    mem[rs1 +4* k +4*i *nf] = vs3d+k[i]
}
```

nf 指定每个 segment 中 field 的数量和目的寄存器的数量, nf 取值范围为 2~8。如当 nf=4,vs3=8,rs1=5,rs2=6,vm=1,lmul=1 时,指令为 vssseg4w.v v8, (x5), x6, 此时指令操作如下:

```
mem[x5]=v8[0], mem[x5+x6]=v8[1] ... mem[x5+i*x6]=v8[i]
mem[x5+4]=v9[0], mem[x5+x6+4]=v9[1] ... mem[x5+i*x6+4]=v9[i]
mem[x5+8]=v10[0], mem[x5+x6+8]=v10[1] ... mem[x5+i*x6+8]=v10[i]
mem[x5+12]=v11[0], mem[x5+x6+12]=v11[1] ... mem[x5+i*x6+12]=v11[i]
```

执行权限:

M mode/S mode/U mode

异常:

非法指令异常, 存储指令非对齐访问异常、存储指令访问错误异常、存储指令页面错误异常

影响标志位:

无

说明:

当 vm=1 时, 指令未被 mask, 对应汇编指令 vssseg<nf>w.v vs3, (rs1),rs2

当 vm=0 时, 指令被 mask, 对应汇编指令 vssseg<nf>w.v vs3, (rs1), rs2,v0.t

指令格式:

31	29	28	26	25	24	20	19	15	14	12	11	7	6	0
nf-1	010	vm	rs2	rs1	110	vs3	0100111							

14.7.312 VSXSEG<NF>B.V——矢量索引 SEGMENT 字节存储指令

语法:

vsxseg<nf>b.v vs3, (rs1),vs2, vm

操作:

```
for( k=0; k<=nf-1; k++) {
```

```

    mem[rs1 + k + vs2[i]] = vs3d+k[i]
}

```

nf 指定每个 segment 中 field 的数量和目的寄存器的数量, nf 取值范围为 2~8。如当 nf=4,vs3=8,rs1=5,vs2=16,vm=1,lmul=1 时,指令为 vsxseg4b.v v8, (x5), v16, 此时指令操作如下:

```

mem[x5+v16[0]]=v8[0] ... mem[x5+v16[i]]=v8[i]
mem[x5+v16[0]+1]=v9[0] ... mem[x5+v16[i]+1]=v9[i]
mem[x5+v16[0]+2]=v10[0] ... mem[x5+v16[i]+2]=v10[i]
mem[x5+v16[0]+3]=v11[0] ... mem[x5+v16[i]+3]=v11[i]

```

执行权限:

M mode/S mode/U mode

异常:

非法指令异常, 存储指令非对齐访问异常、存储指令访问错误异常、存储指令页面错误异常

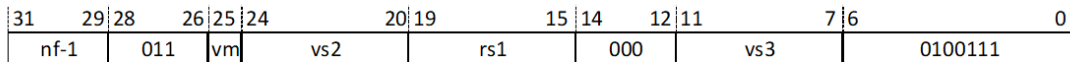
影响标志位:

无

说明:

当 vm=1 时, 指令未被 mask, 对应汇编指令 vsxseg<nf>b.v vs3, (rs1),vs2
 当 vm=0 时, 指令被 mask, 对应汇编指令 vsxseg<nf>b.v vs3, (rs1), vs2,v0.t

指令格式:



14.7.313 VSXSEG<NF>E.V——矢量索引 SEGMENT 元素存储指令

语法:

vsxseg<nf>e.v vs3, (rs1),vs2, vm

操作:

```

offset = sew/8
for( k=0; k<=nf-1; k++) {
    mem[rs1 + k*offset + vs2[i]] = vs3d+k[i]
}

```

nf 指定每个 segment 中 field 的数量和目的寄存器的数量, nf 取值范围为 2~8。如当 nf=4,vs3=8,rs1=5,vs2=16,vm=1,lmul=1,sew=32 时, offset=32/8=4, 指令为 vsxseg4e.v v8, (x5), v16, 此时指令操作如下:

```
mem[x5+v16[0]]=v8[0] ... mem[x5+v16[i]]=v8[i]
mem[x5+v16[0]+4]=v9[0] ... mem[x5+v16[i]+4]=v9[i]
mem[x5+v16[0]+8]=v10[0] ... mem[x5+v16[i]+8]=v10[i]
mem[x5+v16[0]+12]=v11[0] ... mem[x5+v16[i]+12]=v11[i]
```

执行权限:

M mode/S mode/U mode

异常:

非法指令异常, 存储指令非对齐访问异常、存储指令访问错误异常、存储指令页面错误异常

影响标志位:

无

说明:

当 vm=1 时, 指令未被 mask, 对应汇编指令 vsxseg<nf>.v vs3, (rs1), vs2

当 vm=0 时, 指令被 mask, 对应汇编指令 vsxseg<nf>.v vs3, (rs1), vs2, v0.t

指令格式:

31	29	28	26	25	24	20	19	15	14	12	11	7	6	0
nf-1			011		vm	vs2		rs1		111		vs3		0100111

14.7.314 VSXSEG<NF>H.V——矢量索引 SEGMENT 半字存储指令

语法:

vsxseg<nf>.h.v vs3, (rs1), vs2, vm

操作:

```
for( k=0; k<=nf-1; k++) {
    mem[rs1 + 2*k +vs2[i]] = vs3d+k[i]
}
```

nf 指定每个 segment 中 field 的数量和目的寄存器的数量, nf 取值范围为 2~8。如当 nf=4, vs3=8, rs1=5, vs2=16, vm=1, lmul=1 时, 指令为 vsxseg4h.v v8, (x5), v16, 此时指令操作如下:

```
mem[x5+v16[0]]=v8[0] ... mem[x5+v16[i]]=v8[i]
mem[x5+v16[0]+2]=v9[0] ... mem[x5+v16[i]+2]=v9[i]
mem[x5+v16[0]+4]=v10[0] ... mem[x5+v16[i]+4]=v10[i]
mem[x5+v16[0]+6]=v11[0] ... mem[x5+v16[i]+6]=v11[i]
```

执行权限:

M mode/S mode/U mode

异常:

非法指令异常, 存储指令非对齐访问异常、存储指令访问错误异常、存储指令页面错误异常

影响标志位:

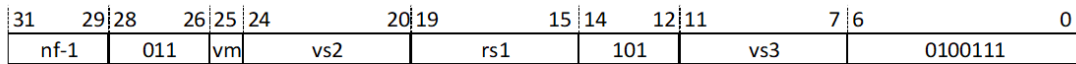
无

说明:

当 $vm=1$ 时, 指令未被 mask, 对应汇编指令 $vsxseg<nf>h.v\ vs3, (rs1),vs2$

当 $vm=0$ 时, 指令被 mask, 对应汇编指令 $vsxseg<nf>h.v\ vs3, (rs1), vs2,v0.t$

指令格式:



14.7.315 VSXSEG<NF>W.V——矢量索引 SEGMENT 字存储指令

语法:

$vsxseg<nf>w.v\ vs3, (rs1),vs2, vm$

操作:

```
for( k=0; k<=nf-1; k++) {
    mem[rs1 + 4*k +vs2[i]] = vs3a+k[i]
}
```

nf 指定每个 segment 中 field 的数量和目的寄存器的数量, nf 取值范围为 2~8。如当 $nf=4,vs3=8,rs1=5,vs2=16,vm=1,lmul=1$ 时, 指令为 $vsxseg4w.v\ v8, (x5), v16$, 此时指令操作如下:

```
mem[x5+v16[0]]=v8[0] ...mem[x5+v16[i]]=v8[i]
mem[x5+v16[0]+4]=v9[0] ...mem[x5+v16[i]+4]=v9[i]
mem[x5+v16[0]+8]=v10[0] ...mem[x5+v16[i]+8]=v10[i]
mem[x5+v16[0]+12]=v11[0] ...mem[x5+v16[i]+12]=v11[i]
```

执行权限:

M mode/S mode/U mode

异常:

非法指令异常, 存储指令非对齐访问异常、存储指令访问错误异常、存储指令页面错误异常

影响标志位:

无

说明:

当 $vm=1$ 时, 指令未被 mask, 对应汇编指令 $vsxseg<nf>w.v\ vs3, (rs1), vs2$

当 $vm=0$ 时, 指令被 mask, 对应汇编指令 $vsxseg<nf>w.v\ vs3, (rs1), vs2, v0.t$

指令格式:

31	29	28	26	25	24	20	19	15	14	12	11	7	6	0
nf-1	011	vm	vs2			rs1		110		vs3			0100111	

14.7.316 VSSH.V——矢量步进半字存储指令

语法:

$vssh.v\ vs3, (rs1), rs2, vm$

操作:

$mem[rs1+i*rs2] = vs3[i][15:0]$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常, 存储指令非对齐访问异常、存储指令访问错误异常、存储指令页面错误异常

影响标志位:

无

说明: 当 $vm=1$ 时, 指令未被 mask, 对应汇编指令 $vssh.v\ vs3, (rs1), rs2$ 。

当 $vm=0$ 时, 指令被 mask, 对应汇编指令 $vssh.v\ vs3, (rs1), rs2, v0.t$ 。

指令格式:

31	29	28	26	25	24	20	19	15	14	12	11	7	6	0
000	010	vm	rs2			rs1		101		vs3			0100111	

14.7.317 VSSRA.VV——矢量带近似算术右移指

语法:

$vssra.vv\ vd, vs2, vs1, vm$

操作:

$vd[i] \leftarrow (\text{signed}(vs2[i] + \text{round}) \gg \gg vs1[i])$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

无

说明:

根据定点舍入模式寄存器 vxrm 确定舍入模式:

- 2' b00: 就近向大值舍入。
- 2' b01: 就近向偶数舍入。
- 2' b10: 向零舍入。
- 2' b11: 向奇数舍入。

当 vm=1 时, 指令未被 mask, 对应汇编指令 vssra.vv vd, vs2, vs1。

当 vm=0 时, 指令被 mask, 对应汇编指令 vssra.vv vd, vs2, vs1,v0.t。

指令格式:

31		26	25	24		20	19		15	14	12	11		7	6		0
	101011		vm		vs2			vs1		000			vd				1010111

14.7.318 VSSRA.VX——矢量标量带近似算术右移指令

语法:

vssra.vx vd, vs2, rs1, vm

操作:

$vd[i] \leftarrow (\text{signed}(vs2[i] + \text{round}) \gg \gg rs1)$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

无

说明:

根据定点舍入模式寄存器 vxrm 确定舍入模式:

- 2' b00: 就近向大值舍入。
- 2' b01: 就近向偶数舍入。
- 2' b10: 向零舍入。
- 2' b11: 向奇数舍入。

当 vm=1 时, 指令未被 mask, 对应汇编指令 vssra.vx vd, vs2, rs1。

当 vm=0 时, 指令被 mask, 对应汇编指令 vssra.vx vd, vs2, rs1,v0.t。

指令格式:

31	26	25	24	20	19	15	14	12	11	7	6	0
101011	vm	vs2		vs1		100		vd		1010111		

14.7.319 VSSRA.VI——矢量立即数带近似算术右移指令

语法:

vssra.vi vd, vs2, imm, vm

操作:

$vd[i] \leftarrow (\text{signed}(vs2[i] + \text{round}) \gg \text{unsigned}(imm))$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

无

说明:

根据定点舍入模式寄存器 vxrm 确定舍入模式:

- 2' b00: 就近向大值舍入。
- 2' b01: 就近向偶数舍入。
- 2' b10: 向零舍入。
- 2' b11: 向奇数舍入。

当 vm=1 时, 指令未被 mask, 对应汇编指令 vssra.vi vd, vs2, imm。

当 vm=0 时, 指令被 mask, 对应汇编指令 vssra.vi vd, vs2, imm,v0.t。

指令格式:

31	26	25	24	20	19	15	14	12	11	7	6	0
101011	vm	vs2		vs1		011		vd		1010111		

14.7.320 VSSRL.VV——矢量带近似逻辑右移指令

语法:

vssrl.vv vd, vs2, vs1, vm

操作:

$vd[i] \leftarrow (\text{signed}(vs2[i] + \text{round}) \gg vs1[i])$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

无

说明:

根据定点舍入模式寄存器 vxrm 确定舍入模式:

- 2' b00: 就近向大值舍入。
- 2' b01: 就近向偶数舍入。
- 2' b10: 向零舍入。
- 2' b11: 向奇数舍入。

当 vm=1 时, 指令未被 mask, 对应汇编指令 vssrl.vv vd, vs2, vs1。

当 vm=0 时, 指令被 mask, 对应汇编指令 vssrl.vv vd, vs2, vs1,v0.t。

指令格式:

31	26	25	24	20	19	15	14	12	11	7	6	0
101010			vm	vs2		vs1		000		vd	1010111	

14.7.321 VSSRL.VX——矢量标量带近似逻辑右移指令

语法:

vssrl.vx vd, vs2, rs1, vm

操作:

$vd[i] \leftarrow (\text{signed}(vs2[i] + \text{round}) \gg rs1)$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

无

说明:

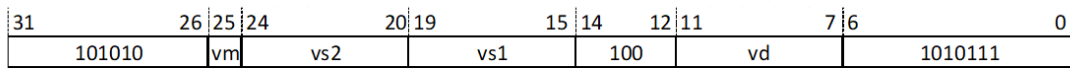
根据定点舍入模式寄存器 vxrm 确定舍入模式:

- 2' b00: 就近向大值舍入。
- 2' b01: 就近向偶数舍入。
- 2' b10: 向零舍入。
- 2' b11: 向奇数舍入。

当 vm=1 时, 指令未被 mask, 对应汇编指令 vssrl.vx vd, vs2, rs1。

当 vm=0 时, 指令被 mask, 对应汇编指令 vssrl.vx vd, vs2, rs1,v0.t。

指令格式:



14.7.322 VSSRL.VI——矢量立即数带近似逻辑右移指令

语法:

vssrl.vi vd, vs2, imm, vm

操作:

$vd[i] \leftarrow (\text{signed}(vs2[i] + \text{round}) \gg \text{unsigned}(imm))$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

无

说明:

根据定点舍入模式寄存器 vxrm 确定舍入模式:

- 2' b00: 就近向大值舍入。

- 2' b01: 就近向偶数舍入。
- 2' b10: 向零舍入。
- 2' b11: 向奇数舍入。

当 vm=1 时, 指令未被 mask, 对应汇编指令 vssrl.vi vd, vs2, imm。

当 vm=0 时, 指令被 mask, 对应汇编指令 vssrl.vi vd, vs2, imm,v0.t。

指令格式:

31	26	25	24	20	19	15	14	12	11	7	6	0
101010			vm	vs2		vs1		011		vd		1010111

14.7.323 VSSUB.VV——矢量整型减法有符号取饱和指令

语法:

vssub.vv vd, vs2, vs1, vm

操作:

$vd[i] \leftarrow \text{sat}(vs2[i] - vs1[i])$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

VXSAT 位

说明:

当 vm=1 时, 指令未被 mask, 对应汇编指令 vssub.vv vd, vs2, vs1。

当 vm=0 时, 指令被 mask, 对应汇编指令 vssub.vv vd, vs2, vs1,v0.t。

指令格式:

31	26	25	24	20	19	15	14	12	11	7	6	0
100011			vm	vs2		vs1		000		vd		1010111

14.7.324 VSSUB.VX——矢量标量整型减法有符号取饱和指令

语法:

vssub.vx vd, vs2, rs1, vm

操作:

$vd[i] \leftarrow \text{sat}(vs2[i] - rs1)$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

VXSAT 位

说明:

当 vm=1 时, 指令未被 mask, 对应汇编指令 vssub.vx vd, vs2, rs1。

当 vm=0 时, 指令被 mask, 对应汇编指令 vssub.vx vd, vs2, rs1,v0.t。

指令格式:

31	26	25	24	20	19	15	14	12	11	7	6	0
100011	vm		vs2		rs1		100		vd		1010111	

14.7.325 VSSUBU.VV——矢量整型减法无符号取饱和指令

语法:

vssubu.vv vd, vs2, vs1, vm

操作:

$vd[i] \leftarrow \text{sat}(vs2[i] - vs1[i])$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

VXSAT 位

说明:

当 $vm=1$ 时, 指令未被 mask, 对应汇编指令 `vssub.vv vd, vs2, vs1`。

当 $vm=0$ 时, 指令被 mask, 对应汇编指令 `vssub.vv vd, vs2, vs1, v0.t`。

指令格式:

31	26	25	24	20	19	15	14	12	11	7	6	0
100010	vm		vs2		vs1		000		vd			1010111

14.7.326 VSSUBU.VX——矢量标量整型减法无符号取饱和指令

语法:

`vssub.vx vd, vs2, rs1, vm`

操作:

$vd[i] \leftarrow \text{sat}(vs2[i] - rs1)$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

VXSAT 位

说明:

当 $vm=1$ 时, 指令未被 mask, 对应汇编指令 `vssub.vx vd, vs2, rs1`。

当 $vm=0$ 时, 指令被 mask, 对应汇编指令 `vssub.vx vd, vs2, rs1, v0.t`。

指令格式:

31	26	25	24	20	19	15	14	12	11	7	6	0
100010	vm		vs2		rs1		100		vd			1010111

14.7.327 VSSW.V——矢量步进字存储指令

语法:

vssw.v vs3, (rs1), rs2, vm

操作:

mem[rs1+i*rs2] = vs3[i][31:0]

执行权限:

M mode/S mode/U mode

异常:

非法指令异常、存储指令非对齐访问异常、存储指令访问错误异常、存储指令页面错误异常

影响标志位:

无

说明:

当 vm=1 时, 指令未被 mask, 对应汇编指令 vssw.v vs3, (rs1), rs2。

当 vm=0 时, 指令被 mask, 对应汇编指令 vssw.v vs3, (rs1), rs2, v0.t。

指令格式:

31	29	28	26	25	24	20	19	15	14	12	11	7	6	0
000	010	vm	rs2			rs1			110	vs3			0100111	

14.7.328 VSUB.VV——矢量整型减法指令

语法:

vsub.vv vd, vs2, vs1, vm

操作:

vd[i] ← vs2[i] - vs1[i]

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

无

说明:

当 $vm=1$ 时, 指令未被 mask, 对应汇编指令 `vsub.vv vd, vs2, vs1`。

当 $vm=0$ 时, 指令被 mask, 对应汇编指令 `vsub.vv vd, vs2, vs1, v0.t`。

指令格式:

31	26	25	24	20	19	15	14	12	11	7	6	0
000010	vm			vs2			vs1			000	vd	1010111

14.7.329 VSUB.VX——矢量标量整型减法指令

语法:

`vsub.vx vd, vs2, rs1, vm`

操作:

$vd[i] \leftarrow vs2[i] - rs1$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

无

说明:

当 $vm=1$ 时, 指令未被 mask, 对应汇编指令 `vsub.vx vd, vs2, rs1`。

当 $vm=0$ 时, 指令被 mask, 对应汇编指令 `vsub.vx vd, vs2, rs1, v0.t`。

指令格式:

31	26	25	24	20	19	15	14	12	11	7	6	0
000010	vm			vs2			rs1			100	vd	1010111

14.7.330 VSUXB.V——矢量索引乱序字节存储指令

语法:

`vsuxb.v vs3, (rs1), vs2, vm`

操作:

$\text{mem}[\text{rs1} + \text{sign_extend}(\text{vs2}[\text{i}])] = \text{vs3}[\text{i}][7:0]$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常、存储指令非对齐访问异常、存储指令访问错误异常、存储指令页面错误异常

影响标志位:

无

说明:

该指令将数据乱序存入存储器。

- 当 $\text{vm}=1$ 时, 指令未被 mask, 对应汇编指令 `vsuxb.v vs3, (rs1), vs2`。
- 当 $\text{vm}=0$ 时, 指令被 mask, 对应汇编指令 `vsuxb.v vs3, (rs1), vs2, v0.t`。

指令格式:

31	29	28	26	25	24	20	19	15	14	12	11	7	6	0
000	111	vm	vs2			rs1		000		vs3			0100111	

14.7.331 VSUXE.V——矢量索引乱序元素存储指令

语法:

`vsuxe.v vs3, (rs1), vs2, vm`

操作:

$\text{mem}[\text{rs1} + \text{sign_extend}(\text{vs2}[\text{i}])] = \text{vs3}[\text{i}][\text{SEW}-1:0]$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常、存储指令非对齐访问异常、存储指令访问错误异常、存储指令页面错误异常

影响标志位:

无

说明:

该指令将数据乱序存入存储器。

当 $vm=1$ 时, 指令未被 mask, 对应汇编指令 `vsuxe.v vs3, (rs1), vs2`。

当 $vm=0$ 时, 指令被 mask, 对应汇编指令 `vsuxe.v vs3, (rs1), vs2, v0.t`。

指令格式:

31	29	28	26	25	24	20	19	15	14	12	11	7	6	0
000	111	vm	vs2			rs1		111		vs3				0100111

14.7.332 VSUXH.V——矢量索引乱序半字存储指令

语法:

`vsuxh.v vs3, (rs1), vs2, vm`

操作:

$mem[rs1+sign_extend(vs2[i])] = vs3[i][15:0]$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常、存储指令非对齐访问异常、存储指令访问错误异常、存储指令页面错误异常

影响标志位:

无

说明:

该指令将数据乱序存入存储器。

当 $vm=1$ 时, 指令未被 mask, 对应汇编指令 `vsuxh.v vs3, (rs1), vs2`。

当 $vm=0$ 时, 指令被 mask, 对应汇编指令 `vsuxh.v vs3, (rs1), vs2, v0.t`。

指令格式:

31	29	28	26	25	24	20	19	15	14	12	11	7	6	0
000	111	vm	vs2			rs1		101		vs3				0100111

14.7.333 VSUXW.V——矢量索引乱序字存储指令

语法:

$vsuxw.v\ vs3,\ (rs1),\ vs2,\ vm$

操作:

$mem[rs1+sign_extend(vs2[i])] = vs3[i][31:0]$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常、存储指令非对齐访问异常、存储指令访问错误异常、存储指令页面错误异常

影响标志位:

无

说明:

该指令将数据乱序存入存储器。

当 $vm=1$ 时, 指令未被 mask, 对应汇编指令 $vsuxw.v\ vs3,\ (rs1),\ vs2$ 。

当 $vm=0$ 时, 指令被 mask, 对应汇编指令 $vsuxw.v\ vs3,\ (rs1),\ vs2,\ v0.t$ 。

指令格式:

31	29	28	26	25	24	20	19	15	14	12	11	7	6	0
000	111	vm	vs2			rs1		110		vs3		0100111		

14.7.334 VSW.V——矢量字存储指令

语法:

$vsw.v\ vs3,\ (rs1),\ vm$

操作:

$mem[rs1+4*i] = vs3[i][31:0]$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常、存储指令非对齐访问异常、存储指令访问错误异常、存储指令页面错误异常

影响标志位:

无

说明:

当 $vm=1$ 时, 指令未被 mask, 对应汇编指令 `vsw.v vs3, (rs1)`。

当 $vm=0$ 时, 指令被 mask, 对应汇编指令 `vsw.v vs3, (rs1), v0.t`。

指令格式:

31	29	28	26	25	24	20	19	15	14	12	11	7	6	0
000		000		vm		00000		rs1		110		vd		0100111

14.7.335 VSXB.V——矢量索引按序字节存储指令

语法:

`vsxb.v vs3, (rs1), vs2, vm`

操作:

$mem[rs1+sign_extend(vs2[i])] = vs3[i][7:0]$

将源寄存器每个元素最低的 byte 数据按序存入以 `rs1` 作为基址, `vs2[i]` 作为 offset 的内存地址上

执行权限:

M mode/S mode/U mode

异常:

非法指令异常, 存储指令非对齐访问异常、存储指令访问错误异常、存储指令页面错误异常

影响标志位:

无

说明:

该指令将数据按序存入存储器。

当 $vm=1$ 时, 指令未被 mask, 对应汇编指令 `vsxb.v vs3, (rs1), vs2`。

当 $vm=0$ 时, 指令被 mask, 对应汇编指令 `vsxb.v vs3, (rs1), vs2, v0.t`。

指令格式:

31	29	28	26	25	24	20	19	15	14	12	11	7	6	0
000		011		vm		vs2		rs1		000		vs3		0100111

14.7.336 VSXE.V——矢量索引按序元素存储指令

语法:

`vsxe.v vs3, (rs1), vs2, vm`

操作:

$\text{mem}[\text{rs1} + \text{sign_extend}(\text{vs2}[\text{i}])] = \text{vs3}[\text{i}][\text{SEW}-1:0]$

执行权限:

M mode/S mode/U mode

异常: 非法指令异常, 存储指令非对齐访问异常、存储指令访问错误异常、存储指令页面错误异常

影响标志位:

无

说明:

该指令将数据按序存入存储器。

当 $\text{vm}=1$ 时, 指令未被 mask, 对应汇编指令 `vsxe.v vs3, (rs1), vs2`。

当 $\text{vm}=0$ 时, 指令被 mask, 对应汇编指令 `vsxe.v vs3, (rs1), vs2, v0.t`。

指令格式:

31	29	28	26	25	24	20	19	15	14	12	11	7	6	0
000	011	vm	vs2			rs1			111	vs3			0100111	

14.7.337 VSXH.V——矢量索引按序半字存储指令

语法:

`vsxh.v vs3, (rs1), vs2, vm`

操作:

$\text{mem}[\text{rs1} + \text{sign_extend}(\text{vs2}[\text{i}])] = \text{vs3}[\text{i}][15:0]$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常, 存储指令非对齐访问异常、存储指令访问错误异常、存储指令页面错误异常

影响标志位:

无

说明:

该指令将数据按序存入存储器。

当 vm=1 时, 指令未被 mask, 对应汇编指令 vsxh.v vs3, (rs1), vs2。

当 vm=0 时, 指令被 mask, 对应汇编指令 vsxh.v vs3, (rs1), vs2, v0.t。

指令格式:

31	29	28	26	25	24	20	19	15	14	12	11	7	6	0
000	011	vm	vs2			rs1		101		vs3				0100111

14.7.338 VSXW.V——矢量索引按序字存储指令

语法:

vsxw.v vs3, (rs1), vs2, vm

操作:

$mem[rs1 + sign_extend(vs2[i])] = vs3[i][31:0]$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常, 存储指令非对齐访问异常、存储指令访问错误异常、存储指令页面错误异常

影响标志位:

无

说明:

该指令将数据按序存入存储器。

当 vm=1 时, 指令未被 mask, 对应汇编指令 vsxw.v vs3, (rs1), vs2。

当 vm=0 时, 指令被 mask, 对应汇编指令 vsxw.v vs3, (rs1), vs2, v0.t。

指令格式:

31	29	28	26	25	24	20	19	15	14	12	11	7	6	0
000	011	vm	vs2			rs1		110		vs3				0100111

14.7.339 VWADD.VV——矢量整型有符号扩位加法指令

语法:

`vwadd.vv vd, vs2, vs1, vm`

操作:

$vd[i] \leftarrow \text{sign_extend}(vs2[i]) + \text{sign_extend}(vs1[i])$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

无

说明:

vd 数据宽度为 $2 * \text{sew}$, vs2 和 vs1 数据宽度为 sew 。

当 $vm=1$ 时, 指令未被 mask, 对应汇编指令 `vwadd.vv vd, vs2, vs1`。

当 $vm=0$ 时, 指令被 mask, 对应汇编指令 `vwadd.vv vd, vs2, vs1, v0.t`。

指令格式:

31	26	25	24	20	19	15	14	12	11	7	6	0
110001		vm	vs2		vs1		010		vd		1010111	

14.7.340 VWADD.VX——矢量标量整型有符号扩位加法指令

语法:

`vwadd.vx vd, vs2, rs1, vm`

操作:

$vd[i] \leftarrow \text{sign_extend}(vs2[i]) + \text{sign_extend}(rs1)$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

无

说明:

vd 数据宽度为 2*sew, vs2 和 rs1 数据宽度为 sew。

当 vm=1 时, 指令未被 mask, 对应汇编指令 vwadd.vx vd, vs2, rs1。

当 vm=0 时, 指令被 mask, 对应汇编指令 vwadd.vx vd, vs2, rs1, v0.t。

指令格式:

31	26	25	24	20	19	15	14	12	11	7	6	0
110001		vm	vs2		rs1		110		vd		1010111	

14.7.341 VWADDU.VV——矢量整型无符号扩位加法指令

语法:

vwaddu.vv vd, vs2, vs1, vm

操作:

$vd[i] \leftarrow \text{zero_extend}(vs2[i]) - \text{zero_extend}(rs1)$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

无

说明:

vd 数据宽度为 2*sew, vs2 和 vs1 数据宽度为 sew。

当 vm=1 时, 指令未被 mask, 对应汇编指令 vwaddu.vv vd, vs2, vs1。

当 vm=0 时, 指令被 mask, 对应汇编指令 vwaddu.vv vd, vs2, vs1, v0.t。

指令格式:

31	26	25	24	20	19	15	14	12	11	7	6	0
110000		vm	vs2		vs1		110		vd		1010111	

14.7.342 VWADDU.VX——矢量标量整型无符号扩位加法指令

语法:

`vwaddu.vx vd, vs2, rs1, vm`

操作:

$vd[i] \leftarrow zero_extend(vs2[i]) - zero_extend(rs1)$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

无

说明:

vd 数据宽度为 $2 * sew$, vs2 和 rs1 数据宽度为 sew 。

当 $vm=1$ 时, 指令未被 mask, 对应汇编指令 `vwaddu.vx vd, vs2, rs1`。

当 $vm=0$ 时, 指令被 mask, 对应汇编指令 `vwaddu.vx vd, vs2, rs1, v0.t`。

指令格式:

31	26	25	24	20	19	15	14	12	11	7	6	0
110000		vm	vs2		rs1		110		vd		1010111	

14.7.343 VWADD.WV——扩位矢量整型有符号扩位加法指令

语法:

`vwadd.wv vd, vs2, vs1, vm`

操作:

$vd[i] \leftarrow vs2[i] + sign_extend(vs1[i])$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

无

说明:

当 $vm=1$ 时, 指令未被 mask, 对应汇编指令 `vwadd.wv vd, vs2, vs1`。

当 $vm=0$ 时, 指令被 mask, 对应汇编指令 `vwadd.wv vd, vs2, vs1, v0.t`。

指令格式:

31	26	25	24	20	19	15	14	12	11	7	6	0
110101	vm	vs2		vs1		010		vd		1010111		

14.7.344 VWADD.WX——扩位矢量标量整型有符号扩位加法指令

语法:

`vwadd.wx vd, vs2, rs1, vm`

操作:

$vd[i] \leftarrow vs2[i] + sign_extend(rs1)$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

无

说明:

`vd` 和 `vs2` 数据宽度为 $2*sew$, `rs1` 数据宽度为 `sew`。

当 $vm=1$ 时, 指令未被 mask, 对应汇编指令 `vwadd.wx vd, vs2, rs1`。

当 $vm=0$ 时, 指令被 mask, 对应汇编指令 `vwadd.wx vd, vs2, rs1, v0.t`。

指令格式:

31	26	25	24	20	19	15	14	12	11	7	6	0
110101	vm	vs2		rs1		110		vd		1010111		

14.7.345 VWADDU.WV——扩位矢量整型无符号扩位加法指令

语法:

`vwaddu.wv vd, vs2, vs1, vm`

操作:

$vd[i] \leftarrow vs2[i] + zero_extend(vs1[i])$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

无

说明:

vd 和 vs2 数据宽度为 $2 * sew$, vs1 数据宽度为 sew 。

当 $vm=1$ 时, 指令未被 mask, 对应汇编指令 `vwaddu.wv vd, vs2, vs1`。

当 $vm=0$ 时, 指令被 mask, 对应汇编指令 `vwaddu.wv vd, vs2, vs1, v0.t`。

指令格式:

31	26	25	24	20	19	15	14	12	11	7	6	0
110100		vm	vs2		vs1		010		vd		1010111	

14.7.346 VWADDU.WX——扩位矢量标量整型无符号扩位加法指令

语法:

`vwaddu.wx vd, vs2, rs1, vm`

操作:

$vd[i] \leftarrow vs2[i] + zero_extend(rs1)$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

无

说明:

vd 和 vs2 数据宽度为 2*sew, rs1 数据宽度为 sew。

当 vm=1 时, 指令未被 mask, 对应汇编指令 vwaddu.wx vd, vs2, rs1。

当 vm=0 时, 指令被 mask, 对应汇编指令 vwaddu.wx vd, vs2, rs1, v0.t。

指令格式:

31	26	25	24	20	19	15	14	12	11	7	6	0
110100		vm	vs2		rs1		110		vd		1010111	

14.7.347 VWMACC.VV——覆盖加数矢量有符号扩位乘累加指令

语法:

vwmacc.vv vd, vs1, vs2, vm

操作:

$vd[i] \leftarrow \text{signed}(vs1[i]) \times \text{signed}(vs2[i]) + vd[i]$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

无

说明:

vd 数据宽度为 2*sew, vs2 和 vs1 数据宽度为 sew。

当 vm=1 时, 指令未被 mask, 对应汇编指令 vwmacc.vv vd, vs1, vs2。

当 vm=0 时, 指令被 mask, 对应汇编指令 vwmacc.vv vd, vs1, vs2, v0.t。

指令格式:

31	26	25	24	20	19	15	14	12	11	7	6	0
111101		vm	vs2		vs1		010		vd		1010111	

14.7.348 VWMACC.VX——覆盖加数矢量标量有符号扩位乘累加指令

语法:

`vwmacc.vx vd, rs1, vs2, vm`

操作:

$vd[i] \leftarrow \text{signed}(rs1) \times \text{signed}(vs2[i]) + vd[i]$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

无

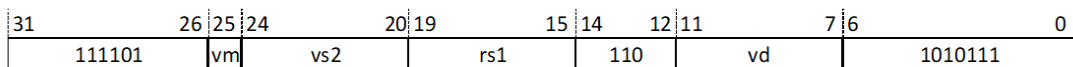
说明:

vd 数据宽度为 $2 \times \text{sew}$, vs2 和 rs1 数据宽度为 sew 。

当 $vm=1$ 时, 指令未被 mask, 对应汇编指令 `vwmacc.vx vd, rs1, vs2`。

当 $vm=0$ 时, 指令被 mask, 对应汇编指令 `vwmacc.vx vd, rs1, vs2, v0.t`。

指令格式:



14.7.349 VWMACCSU.VV——覆盖加数矢量有符号无符号扩位乘累加指令

语法:

`vwmaccsu.vv vd, vs1, vs2, vm`

操作:

$vd[i] \leftarrow \text{signed}(vs1[i]) \times \text{unsigned}(vs2[i]) + vd[i]$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

无

说明:

vd 数据宽度为 2*sew, vs2 和 vs1 数据宽度为 sew。

当 vm=1 时, 指令未被 mask, 对应汇编指令 vwmaccsu.vv vd, vs1, vs2。

当 vm=0 时, 指令被 mask, 对应汇编指令 vwmaccsu.vv vd, vs1, vs2, v0.t。

指令格式:

31	26	25	24	20	19	15	14	12	11	7	6	0
111110	vm			vs2			vs1	010			vd	1010111

14.7.350 VWMACCSU.VX——覆盖加数矢量标量有符号无符号扩位乘累加指令

语法:

vwmaccsu.vx vd, rs1, vs2, vm

操作:

$vd[i] \leftarrow \text{signed}(rs1) \times \text{unsigned}(vs2[i]) + vd[i]$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

无

说明:

vd 数据宽度为 2*sew, vs2 和 rs1 数据宽度为 sew。

当 vm=1 时, 指令未被 mask, 对应汇编指令 vwmaccsu.vx vd, rs1, vs2。

当 vm=0 时, 指令被 mask, 对应汇编指令 vwmaccsu.vx vd, rs1, vs2, v0.t。

指令格式:

31	26	25	24	20	19	15	14	12	11	7	6	0
111110	vm			vs2			rs1	110			vd	1010111

14.7.351 VWMACCU.VV——覆盖加数矢量无符号扩位乘累加指令

语法:

vwmaccu.vv vd, vs1, vs2, vm

操作:

$vd[i] \leftarrow \text{unsigned}(vs1[i]) \times \text{unsigned}(vs2[i]) + vd[i]$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

无

说明:

vd 数据宽度为 $2 \times \text{sew}$, vs2 和 vs1 数据宽度为 sew。

当 $vm=1$ 时, 指令未被 mask, 对应汇编指令 vwmaccu.vv vd, vs1, vs2。

当 $vm=0$ 时, 指令被 mask, 对应汇编指令 vwmaccu.vv vd, vs1, vs2, v0.t。

指令格式:

31	26	25	24	20	19	15	14	12	11	7	6	0
111100		vm	vs2		vs1		010		vd		1010111	

14.7.352 VWMACCU.VX——覆盖加数矢量标量无符号扩位乘累加指令

语法:

vwmaccu.vx vd, rs1, vs2, vm

操作:

$vd[i] \leftarrow \text{unsigned}(rs1) \times \text{unsigned}(vs2[i]) + vd[i]$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

无

说明:

vd 数据宽度为 2*sew, vs2 和 rs1 数据宽度为 sew。

当 vm=1 时, 指令未被 mask, 对应汇编指令 vwmaccu.vx vd, rs1, vs2。

当 vm=0 时, 指令被 mask, 对应汇编指令 vwmaccu.vx vd, rs1, vs2, v0.t。

指令格式:

31	26	25	24	20	19	15	14	12	11	7	6	0
111100		vm	vs2		rs1		110		vd		1010111	

14.7.353 VWMACCUS.VX——覆盖加数矢量标量无符号有符号扩位乘累加指令

语法:

vwmaccus.vx vd, rs1, vs2, vm

操作:

$vd[i] \leftarrow \text{unsigned}(rs1) \times \text{signed}(vs2[i]) + vd[i]$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

无

说明:

vd 数据宽度为 2*sew, vs2 和 rs1 数据宽度为 sew。

当 vm=1 时, 指令未被 mask, 对应汇编指令 vwmaccus.vx vd, rs1, vs2。

当 vm=0 时, 指令被 mask, 对应汇编指令 vwmaccus.vx vd, rs1, vs2, v0.t。

指令格式:

31	26	25	24	20	19	15	14	12	11	7	6	0
111111		vm	vs2		rs1		110		vd		1010111	

14.7.354 VWMUL.VV——矢量有符号扩位乘法指令

语法:

vwmul.vv vd, vs2, vs1, vm

操作:

$vd[i] \leftarrow \text{signed}(vs2[i]) \times \text{signed}(vs1[i])$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

无

说明:

vd 数据宽度为 $2 * sew$, vs2 和 vs1 数据宽度为 sew 。

当 $vm=1$ 时, 指令未被 mask, 对应汇编指令 vwmul.vv vd, vs2, vs1。

当 $vm=0$ 时, 指令被 mask, 对应汇编指令 vwmul.vv vd, vs2, vs1, v0.t。

指令格式:

31	26	25	24	20	19	15	14	12	11	7	6	0
111011		vm	vs2		vs1		010		vd		1010111	

14.7.355 VWMUL.VX——矢量标量有符号扩位乘法指令

语法:

vwmul.vx vd, vs2, rs1, vm

操作:

$vd[i] \leftarrow \text{signed}(vs2[i]) \times \text{signed}(rs1)$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

无

说明:

vd 数据宽度为 $2 \times \text{sew}$, vs2 和 rs1 数据宽度为 sew 。

当 $\text{vm}=1$ 时, 指令未被 mask, 对应汇编指令 `vwmul.vx vd, vs2, rs1`。

当 $\text{vm}=0$ 时, 指令被 mask, 对应汇编指令 `vwmul.vx vd, vs2, rs1, v0.t`。

指令格式:

31	26	25	24	20	19	15	14	12	11	7	6	0
111011		vm	vs2		rs1		110		vd		1010111	

14.7.356 VWMULSU.VV——矢量有符号无符号扩位乘法指令

语法:

`vwmulsu.vv vd, vs2, vs1, vm`

操作:

$\text{vd}[i] \leftarrow \text{signed}(\text{vs2}[i]) \times \text{unsigned}(\text{vs1}[i])$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

无

说明:

vd 数据宽度为 $2 \times \text{sew}$, vs2 和 vs1 数据宽度为 sew 。

当 $\text{vm}=1$ 时, 指令未被 mask, 对应汇编指令 `vwmulsu.vv vd, vs2, vs1`。

当 $\text{vm}=0$ 时, 指令被 mask, 对应汇编指令 `vwmulsu.vv vd, vs2, vs1, v0.t`。

指令格式:

31	26	25	24	20	19	15	14	12	11	7	6	0
111010		vm	vs2		vs1		010		vd		1010111	

14.7.357 VWMULSU.VX——矢量标量有符号无符号扩位乘法指令

语法:

vwmulsu.vx vd, vs2, rs1, vm

操作:

$vd[i] \leftarrow \text{signed}(vs2[i]) \times \text{unsigned}(rs1)$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

无

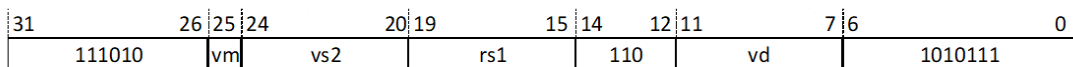
说明:

vd 数据宽度为 $2 * \text{sew}$, vs2 和 rs1 数据宽度为 sew 。

当 $vm=1$ 时, 指令未被 mask, 对应汇编指令 vwmulsu.vx vd, vs2, rs1。

当 $vm=0$ 时, 指令被 mask, 对应汇编指令 vwmulsu.vx vd, vs2, rs1, v0.t。

指令格式:



14.7.358 VWMULU.VV——矢量无符号扩位乘法指令

语法:

vwmulu.vv vd, vs2, vs1, vm

操作:

$vd[i] \leftarrow \text{unsigned}(vs2[i]) \times \text{unsigned}(vs1[i])$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

无

说明:

vd 数据宽度为 $2 * sew$, vs2 和 vs1 数据宽度为 sew 。

当 $vm=1$ 时, 指令未被 mask, 对应汇编指令 `vwmulu.vv vd, vs2, vs1`。

当 $vm=0$ 时, 指令被 mask, 对应汇编指令 `vwmulu.vv vd, vs2, vs1, v0.t`。

指令格式:

31	26	25	24	20	19	15	14	12	11	7	6	0
111000		vm	vs2		vs1		010		vd		1010111	

14.7.359 VWMULU.VX——矢量标量无符号扩位乘法指令

语法:

`vwmulu.vx vd, vs2, rs1, vm`

操作:

$vd[i] \leftarrow \text{unsigned}(vs2[i]) \times \text{unsigned}(rs1)$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

无

说明:

vd 数据宽度为 $2 * sew$, vs2 和 rs1 数据宽度为 sew 。

当 $vm=1$ 时, 指令未被 mask, 对应汇编指令 `vwmulu.vx vd, vs2, rs1`。

当 $vm=0$ 时, 指令被 mask, 对应汇编指令 `vwmulu.vx vd, vs2, rs1, v0.t`。

指令格式:

31	26	25	24	20	19	15	14	12	11	7	6	0
111000		vm	vs2		rs1		110		vd		1010111	

14.7.360 VWREDSUM.VS——矢量缩减有符号扩位累加指令

语法:

```
vwredsum.vs vd, vs2, vs1, vm
```

操作:

```
tmp = vs1[0]
for( i=0; i<vl; i++) {
    tmp = tmp + sign_extend(vs2[i])
}
vd[0]= tmp
vd[VLEN/SEW-1:1] = 0
```

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

无

说明:

vd 和 vs1 数据宽度为 2*sew, vs2 数据宽度为 sew。vs2 中的元素有符号扩位成 2*sew 后参与运算。
 当 vm=1 时, 指令未被 mask, 对应汇编指令 vwredsum.vs vd, vs2, vs1。
 当 vm=0 时, 指令被 mask, 对应汇编指令 vwredsum.vs vd, vs2, vs1, v0.t。

指令格式:

31	26	25	24	20	19	15	14	12	11	7	6	0		
110001			vm	vs2			vs1			000		vd	1010111	

14.7.361 VWREDSUMU.VS——矢量缩减无符号扩位累加指令

语法:

```
vwredsumu.vs vd, vs2, vs1, vm
```

操作:

```
tmp = vs1[0]
for( i=0; i<vl; i++) {
    tmp = tmp + unsign_extend(vs2[i])
}
```


vd 数据宽度为 2*sew, vs2 和 vs1 数据宽度为 sew。

根据定点舍入模式寄存器 vxrm 确定舍入模式:

- 2' b00: 向大数舍入。
- 2' b01: 向偶数舍入。
- 2' b10: 向零舍入。
- 2' b11: 向奇数舍入。

当 vm=1 时, 指令未被 mask, 对应汇编指令 vwsimacc.vv vd, vs2, vs1。

当 vm=0 时, 指令被 mask, 对应汇编指令 vwsimacc.vv vd, vs2, vs1, v0.t。

指令格式:

31	26	25	24	20	19	15	14	12	11	7	6	0
111101			vm	vs2		vs1		000		vd		1010111

14.7.363 VWSIMACC.VX——矢量标量有符号带饱和扩位乘累加指令

语法:

vwsimacc.vx vd, rs1, vs2, vm

操作:

$vd[i] \leftarrow \text{clip}(((rs1 \times vs2[i] + \text{round}) \gg SEW/2) + vd[i])$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

VXSAT 位

说明:

vd 数据宽度为 2*sew, vs2 和 rs1 数据宽度为 sew。

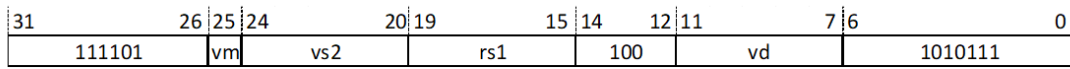
根据定点舍入模式寄存器 vxrm 确定舍入模式:

- 2' b00: 向大数舍入。
- 2' b01: 向偶数舍入。
- 2' b10: 向零舍入。
- 2' b11: 向奇数舍入。

当 vm=1 时, 指令未被 mask, 对应汇编指令 vwsimacc.vx vd, rs1, vs2。

当 vm=0 时, 指令被 mask, 对应汇编指令 vwsimacc.vx vd, rs1, vs2, v0.t。

指令格式:



14.7.364 VWSMACCSU.VV——矢量带饱和和取负有符号无符号扩位乘累减取负指令

语法:

vwsmaccsu.vv vd, vs1, vs2, vm

操作:

$vd[i] \leftarrow clip(-((signed(vs1[i]) \times unsigned(vs2[i]) + round) \gg SEW/2) + vd[i])$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

VXSAT 位

说明:

vd 数据宽度为 2*sew, vs2 和 vs1 数据宽度为 sew。

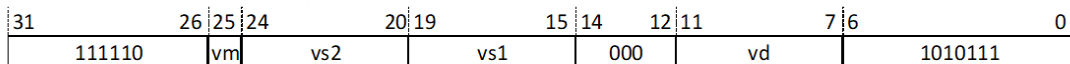
根据定点舍入模式寄存器 vxrm 确定舍入模式:

- 2' b00: 向大数舍入。
- 2' b01: 向偶数舍入。
- 2' b10: 向零舍入。
- 2' b11: 向奇数舍入。

当 vm=1 时, 指令未被 mask, 对应汇编指令 vwsmaccsu.vv vd, vs2, vs1。

当 vm=0 时, 指令被 mask, 对应汇编指令 vwsmaccsu.vv vd, vs2, vs1, v0.t。

指令格式:



14.7.365 VWSMACCSU.VX——矢量标量带饱和和取负有符号无符号扩位乘累减取负指令

语法:

vwsmaccsu.vx vd, rs1, vs2, vm

操作:

$$vd[i] \leftarrow \text{clip}(-((\text{signed}(rs1) \times \text{unsigned}(vs2[i]) + \text{round}) \gg \text{SEW}/2) + vd[i])$$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

VXSAT 位

说明:

vd 数据宽度为 2*sew, vs2 和 rs1 数据宽度为 sew。

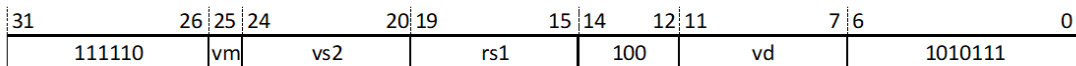
根据定点舍入模式寄存器 vxrm 确定舍入模式:

- 2' b00: 向大数舍入。
- 2' b01: 向偶数舍入。
- 2' b10: 向零舍入。
- 2' b11: 向奇数舍入。

当 vm=1 时, 指令未被 mask, 对应汇编指令 vwsmaccsu.vx vd, rs1, vs2。

当 vm=0 时, 指令被 mask, 对应汇编指令 vwsmaccsu.vx vd, rs1, vs2, v0.t。

指令格式:



14.7.366 VWSMACCU.VV——矢量无符号带饱和扩位乘累加指令

语法:

vwsmaccu.vv vd, vs1, vs2, vm

操作:

$$vd[i] \leftarrow \text{clipu}(((vs1[i] \times vs2[i] + \text{round}) \gg \text{SEW}/2) + vd[i])$$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

VXSAT 位

说明:

vd 数据宽度为 $2 * \text{sew}$, vs2 和 vs1 数据宽度为 sew 。

根据定点舍入模式寄存器 vxrm 确定舍入模式:

- 2' b00: 向大数舍入。
- 2' b01: 向偶数舍入。
- 2' b10: 向零舍入。
- 2' b11: 向奇数舍入。

当 $\text{vm}=1$ 时, 指令未被 mask, 对应汇编指令 `vwsmaccu.vv vd, vs1, vs2`。

当 $\text{vm}=0$ 时, 指令被 mask, 对应汇编指令 `vwsmaccu.vv vd, vs1, vs2, v0.t`。

指令格式:

31		26	25	24		20	19		15	14	12	11		7	6		0
	111100		vm		vs2			vs1		000		vd				1010111	

14.7.367 VWSMACCU.VX——矢量标量无符号带饱和扩位乘累加指令

语法:

`vwsmaccu.vx vd, rs1, vs2, vm`

操作:

$\text{vd}[i] \leftarrow \text{clipu}(((\text{rs1} \times \text{vs2}[i] + \text{round}) \gg \text{SEW}/2) + \text{vd}[i])$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

VXSAT 位

说明:

vd 数据宽度为 $2 * \text{sew}$, vs2 和 rs1 数据宽度为 sew 。

根据定点舍入模式寄存器 vxrm 确定舍入模式:

- 2' b00: 向大数舍入。
- 2' b01: 向偶数舍入。

- 2' b10: 向零舍入。
- 2' b11: 向奇数舍入。

当 vm=1 时, 指令未被 mask, 对应汇编指令 vwsmaacu.vv vd, rs1, vs2。

当 vm=0 时, 指令被 mask, 对应汇编指令 vwsmaacu.vv vd, rs1, vs2, v0.t。

指令格式:

31	26	25	24	20	19	15	14	12	11	7	6	0
111100		vm	vs2		rs1		100		vd		1010111	

14.7.368 VWSMACCUS.VX——矢量标量带饱和和取负有符号无符号扩位乘累减指令

语法:

vswmaccus.vx vd, rs1, vs2, vm

操作:

$vd[i] \leftarrow clip(-((unsigned(rs1) \times signed(vs2[i]) + round) \gg SEW/2) + vd[i])$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

VXSAT 位

说明:

vd 数据宽度为 2*sew, vs2 和 rs1 数据宽度为 sew。

根据定点舍入模式寄存器 vxrm 确定舍入模式:

- 2' b00: 向大数舍入。
- 2' b01: 向偶数舍入。
- 2' b10: 向零舍入。
- 2' b11: 向奇数舍入。

当 vm=1 时, 指令未被 mask, 对应汇编指令 vswmaccus.vx vd, rs1, vs2。

当 vm=0 时, 指令被 mask, 对应汇编指令 vswmaccus.vx vd, rs1, vs2, v0.t。

指令格式:

31	26	25	24	20	19	15	14	12	11	7	6	0
111111		vm	vs2		rs1		100		vd		1010111	

14.7.369 VWSUB.VV——矢量整型有符号扩位减法指令

语法:

vwsb.vv vd, vs2, vs1, vm

操作:

$vd[i] \leftarrow \text{sign_extend}(vs2[i]) - \text{sign_extend}(vs1[i])$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

无

说明:

vd 数据宽度为 $2 * \text{sew}$, vs2 和 vs1 数据宽度为 sew 。

当 $vm=1$ 时, 指令未被 mask, 对应汇编指令 vwsb.vv vd, vs2, vs1。

当 $vm=0$ 时, 指令被 mask, 对应汇编指令 vwsb.vv vd, vs2, vs1, v0.t。

指令格式:

31	26	25	24	20	19	15	14	12	11	7	6	0
110011		vm	vs2		vs1		010		vd		1010111	

14.7.370 VWSUB.VX——矢量标量整型有符号扩位减法指令

语法:

vwsb.vx vd, vs2, rs1, vm

操作:

$vd[i] \leftarrow \text{sign_extend}(vs2[i]) - \text{sign_extend}(rs1)$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

无

说明:

vd 数据宽度为 2*sew, vs2 和 rs1 数据宽度为 sew。

当 vm=1 时, 指令未被 mask, 对应汇编指令 vwsb.vx vd, vs2, rs1。

当 vm=0 时, 指令被 mask, 对应汇编指令 vwsb.vx vd, vs2, rs1, v0.t。

指令格式:

31	26	25	24	20	19	15	14	12	11	7	6	0	
110011			vm	vs2		rs1		110		vd		1010111	

14.7.371 VWSUB.WV——扩位矢量整型有符号扩位减法指令

语法:

vwsb.wv vd, vs2, vs1, vm

操作:

$vd[i] \leftarrow vs2[i] - sign_extend(vs1[i])$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

无

说明:

vd 和 vs2 数据宽度为 2*sew, vs1 数据宽度为 sew。

当 vm=1 时, 指令未被 mask, 对应汇编指令 vwsb.wv vd, vs2, vs1。

当 vm=0 时, 指令被 mask, 对应汇编指令 vwsb.wv vd, vs2, vs1, v0.t。

指令格式:

31	26	25	24	20	19	15	14	12	11	7	6	0	
110111			vm	vs2		vs1		010		vd		1010111	

14.7.372 VWSUB.WX——扩位矢量标量整型有符号扩位减法指令

语法:

vwsb.wx vd, vs2, rs1, vm

操作:

$vd[i] \leftarrow vs2[i] - \text{sign_extend}(rs1)$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

无

说明:

vd 和 vs2 数据宽度为 2*sew, rs1 数据宽度为 sew。

当 vm=1 时, 指令未被 mask, 对应汇编指令 vwsb.wx vd, vs2, rs1。

当 vm=0 时, 指令被 mask, 对应汇编指令 vwsb.wx vd, vs2, rs1, v0.t。

指令格式:

31	26	25	24	20	19	15	14	12	11	7	6	0
110111		vm	vs2		rs1		110		vd		1010111	

14.7.373 VWSUBU.VV——矢量整型无符号扩位减法指令

语法:

vwsbu.vv vd, vs2, vs1, vm

操作:

$vd[i] \leftarrow \text{unsign_extend}(vs2[i]) - \text{unsign_extend}(vs1[i])$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

无

说明:

vd 数据宽度为 2*sew, vs2 和 vs1 数据宽度为 sew。

当 vm=1 时, 指令未被 mask, 对应汇编指令 vwsubu.vv vd, vs2, vs1。

当 vm=0 时, 指令被 mask, 对应汇编指令 vwsubu.vv vd, vs2, vs1, v0.t。

指令格式:

31	26	25	24	20	19	15	14	12	11	7	6	0
110010		vm	vs2		vs1		010		vd		1010111	

14.7.374 VWSUBU.VX——矢量标量整型无符号扩位减法指令

语法:

vwsubu.vx vd, vs2, rs1, vm

操作:

$vd[i] \leftarrow \text{unsign_extend}(vs2[i]) - \text{unsign_extend}(rs1)$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

无

说明:

vd 数据宽度为 2*sew, vs2 和 rs1 数据宽度为 sew。

当 vm=1 时, 指令未被 mask, 对应汇编指令 vwsubu.vx vd, vs2, rs1。

当 vm=0 时, 指令被 mask, 对应汇编指令 vwsubu.vx vd, vs2, rs1, v0.t。

指令格式:

31	26	25	24	20	19	15	14	12	11	7	6	0
110010		vm	vs2		vs1		110		vd		1010111	

14.7.375 VWSUBU.WV——扩位矢量整型无符号扩位减法指令

语法:

vwsubu.wv vd, vs2, vs1, vm

操作:

$vd[i] \leftarrow vs2[i] - \text{unsign_extend}(vs1[i])$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

无

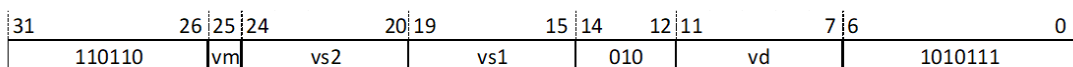
说明:

vd 和 vs2 数据宽度为 2*sew, vs1 数据宽度为 sew。

当 vm=1 时, 指令未被 mask, 对应汇编指令 vwsubu.wv vd, vs2, vs1。

当 vm=0 时, 指令被 mask, 对应汇编指令 vwsubu.wv vd, vs2, vs1, v0.t

指令格式:



14.7.376 VWSUBU.WX——扩位矢量标量整型无符号扩位减法指令

语法:

vwsubu.wx vd, vs2, rs1, vm

操作:

$vd[i] \leftarrow vs2[i] - \text{zero_extend}(rs1)$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

无

说明:

vd 和 vs2 数据宽度为 2*sew, rs1 数据宽度为 sew。

当 vm=1 时, 指令未被 mask, 对应汇编指令 vwsubu.wx vd, vs2, rs1。

当 vm=0 时, 指令被 mask, 对应汇编指令 vwsubu.wx vd, vs2, rs1, v0.t

指令格式:

31	26	25	24	20	19	15	14	12	11	7	6	0
110110		vm	vs2		rs1		110		vd		1010111	

14.7.377 VXOR.VV——矢量按位异或指令

语法:

vxor.vv vd, vs2, vs1, vm

操作:

$vd[i] \leftarrow vs2[i] \wedge vs1[i]$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

无

说明:

当 vm=1 时, 指令未被 mask, 对应汇编指令 vxor.vv vd, vs2, vs1。

当 vm=0 时, 指令被 mask, 对应汇编指令 vxor.vv vd, vs2, vs1, v0.t。

指令格式:

31	26	25	24	20	19	15	14	12	11	7	6	0
001011		vm	vs2		vs1		000		vd		1010111	

14.7.378 VXOR.VX——矢量标量按位异或指令

语法:

`vxor.vx vd, vs2, rs1, vm`

操作:

$vd[i] \leftarrow vs2[i] \wedge rs1$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

无

说明:

当 $vm=1$ 时, 指令未被 mask, 对应汇编指令 `vxor.vx vd, vs2, rs1`。

当 $vm=0$ 时, 指令被 mask, 对应汇编指令 `vxor.vx vd, vs2, rs1, v0.t`。

指令格式:

31	26 25 24	20 19	15 14	12 11	7 6	0
001011	vm	vs2	rs1	100	vd	1010111

14.7.379 VXOR.VI——矢量立即数按位异或指令

语法:

`vxor.vi vd, vs2, imm, vm`

操作:

$vd[i] \leftarrow vs2[i] \wedge \text{sign_extend}(imm)$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

无

说明:

当 $vm=1$ 时, 指令未被 mask, 对应汇编指令 $vxor.vi\ vd, vs2, imm$ 。

当 $vm=0$ 时, 指令被 mask, 对应汇编指令 $vxor.vi\ vd, vs2, imm, v0.t$ 。

指令格式:

31	26	25	24	20	19	15	14	12	11	7	6	0
001011	vm		vs2		imm		011		vd			1010111

14.8 附录 A-8 伪指令列表

RISC-V 实现了一系列的伪指令, 在此列出仅供参考, 按英文字母顺序排列。

伪指令	基础指令	含义
beqz rs, offset	beq rs, x0, offset	寄存器为零分支跳转
bnez rs, offset	bne rs, x0, offset	寄存器不为零分支跳转
blez rs, offset	bge x0,rs,offset	寄存器小于等于零跳转
bgez rs, offset	bge rs, x0, offset	寄存器大于等于零跳转
bltz rs, offset	blt rs, x0, offset	寄存器小于零跳转
bgtz rs, offset	blt x0, xs, offset	寄存器大于零跳转
bgt rs, rt, offset	blt rt, rs, offset	比较大于分支跳转
ble rs, rt, offset	bge rt, rs, offset	比较小于等于分支跳转
bgtu rs, rt, offset	bltu rt, rs, offset	无符号比较大于分支跳转
bleu rs, rt, offset	bgeu rt, rs, offset	无符号比较小于等于分支跳转
call offset	auipc x6, offset[31:12] jalr x1, x6, offset[11:0]	跳转 4KB-4GB 空间的函数
csrrc csr, rs	csrrc x0, csr, rs	清除控制寄存器中对应比特
csrci csr, imm	csrrci x0, csr, imm	清除控制寄存器低 6 位中对应比特
csrs csr, rs	csrrs x0, csr, rs	置位控制寄存器中对应比特
csrr rd, csr csrrs rd, csr, x0 读 控制寄存器中对应比特		
csrsi csr, imm csrrsi x0, csr, imm 置位控制寄 存器低 6 位中对应比特		
csrw csr, rs	csrrw x0, csr, rs	写控制寄存器中对应比特
csrwi csr, imm	csrrwi x0, csr, imm	写控制寄存器低 6 位中对应比特
fabs.d rd , rs	fsgnjx.d rd, rs,rs	双精度浮点数取绝对值
fabs.s rd , rs	fsgnjx.s rd, rs,rs	单精度浮点数取绝对值
fence	fence iorw, iorw	存储和外设同步指令
fl{w d} rd, symbol, rt	auipc rt, symbol[31:12] fl{w d} rd, symbol[11:0](rt)	4GB 地址空间浮点加载指令

下页继续

表 14.1 – 续上页

伪指令	基础指令	含义
fmv.d rd, rs	fsgnj.d rd, rs,rs	双精度浮点复制指令
fmv.s rd, rs	fsgnj.s rd, rs,rs	单精度浮点复制指令
fneg.d rd, rs	fsgnjn.d rd, rs,rs	双精度浮点取负指令
fneg.s rd, rs	fsgnjn.s rd, rs,rs	单精度浮点取负指令
frcsr rd	csrrs x0, fcsr, x0	浮点控制寄存器读取指令
frflags rd	csrrs rd, fflags,x0	浮点异常位读取指令
frfm rd	csrrs rd, frm,x0	浮点舍入位读取指令
fscsr rs	csrrw x0, fcsr,rs	写浮点控制寄存器指令
fscsr rd, rs	csrrs rd, fcsr, rs	浮点控制寄存器读写指令
fsflags rs	csrrw x0, fcsr,rs	写浮点异常位指令
fsflags rd, rs	csrrs rd, fcsr, rs	浮点异常位读写指令
fsflagsi imm	csrrwi x0, fflags,imm	立即数写浮点异常位指令
fsflagsi rd, imm	csrrwi rd, fflags, imm	浮点异常位立即数读写指令
fsrm rs	csrrw x0, frm,rs	写浮点舍入位指令
fsrm rd, rs	csrrs rd, frm, rs	浮点舍入位读写指令
fsrmi imm	csrrwi x0, frm,imm	立即数写浮点舍入位指令
fsrmi rd, imm	csrrwi rd, frm, imm	浮点舍入位立即数读写指令
fs{w d} rd, symbol, rt	auipc rt, symbol[31:12] fs{w d} rd, symbol[11:0](rt)	4GB 地址空间浮点存储指令
j offset	jal x0, offset	直接跳转指令
jal offset	jal x1, offset	子程序跳转和链接指令
jalr rs	jalr x1, rs, 0	子程序跳转寄存器和链接寄存器指令
jr rs	jalr x0, rs, 0	跳转寄存器指令
la rd, symbol	auipc rd, symbol[31:12] addi rd, rd, symbol[11:0]	指令地址加载指令
li rd, immediate	根据立即数大小拆分为多条指令	立即数加载指令
l{b h w d} rd, symbol, rt	auipc rt, symbol[31:12] l{b h w d} rd, symbol[11:0](rt)	4GB 地址空间加载指令
mv rd, rs	addi rd, rs, 0	数据传送指令
neg rd, rs	sub rd, x0, rs	寄存器取负指令
negw rd, rs	subw rd, x0, rs	寄存器低 32 位取负指令
nop	addi x0,x0,0	空指令
not rd, rs	xori rd, rs, -1	寄存器取反指令
rdcycle[h] rd	csrrs rd, cycle[h], x0	周期数读取指令
rdinstret[h] rd	csrrs rd, instret[h], x0	指令数读取指令
rdtime[h] rd	csrrs rd, time[h], x0	真实始终读取指令
ret	jalr x0, x1,0	子程序返回指令
s{b h w d} rd, symbol, rt	auipc rt, symbol[31:12] s{b h w d} rd, symbol[11:0](rt)	4GB 地址空间存储指令
seqz rd, rs	sltiu rd, rs, 1	寄存器为 0 置 1 指令
sext.w rd, rs	addiw rd, rs, 0	符号位扩展指令

下页继续

表 14.1 – 续上页

伪指令	基础指令	含义
sgtz rd, rs	slt rd, rs, x0, rs	寄存器大于 0 置 1 指令
sltz rd, rs	slt rd, rs, rs, x0	寄存器小于 0 置 1 指令
snez rd, rs	sltu rd, rs, x0, rs	寄存器不为 0 置 1 指令
tail offset	auipc x6, offset[31:12] jalr x0, x6, offset[11:0]	寄存器不链接跳转子程序指令

平头哥半导体

第十五章 附录 B 平头哥扩展指令术语

除了标准中定义的 GCV 指令集外，C906 实现了自定义的指令集，包括 Cache 指令子集，算术运算指令子集，位操作指令子集，存储指令子集以及浮点半精度指令子集。

其中，Cache 指令子集、同步指令子集、算术运算指令子集、位操作指令子集以及存储指令子集需要在 `mxstatus.theadisae == 1` 时方可正常执行，否则产生非法指令异常；浮点半精度指令子集需要在 `mstatus.fs != 2'b00` 时方可正常执行，否则产生非法指令异常。

以下按照不同指令子集扩展对每条指令做具体描述。

15.1 附录 B-1 Cache 指令术语

Cache 指令子集实现了对 cache 的操作，每条指令位宽为 32 位。

以下指令按英文字母顺序排列。

15.1.1 DCACHE.CALL——DCACHE 清全部脏表项指令

语法：

`dcache.call`

操作：

clear 所有 L1 dcache 表项，将所有 dirty 表项写回到下一级存储。

执行权限：

M mode/S mode

异常：

非法指令异常

说明：

`mxstatus.theadisae=0`，执行该指令产生非法指令异常。

`mxstatus.theadisae=1`，U mode 下执行该指令产生非法指令异常。

指令格式：

31	25	24	20	19	15	14	12	11	7	6	0
0000000		00001		00000		000		00000		0001011	

15.1.2 DCACHE.CVA——DCACHE 指定虚拟地址清脏表项指令

语法:

dcache.cva rs1

操作:

将 rs1 中虚拟地址所对应的 dcache 表项写回到下一级存储, 操作所有核 L1CACHE。

执行权限:

M mode/S mode/U mode

异常:

非法指令异常/加载指令页面错误异常

说明:

- mxstatus.theadisae=0, 执行该指令产生非法指令异常。
- mxstatus.theadisae=1, mxstatus.ucme =1, U mode 下可以执行该指令。
- mxstatus.theadisae=1, mxstatus.ucme =0, U mode 下执行该指令产生非法指令异常。

指令格式:

31	25	24	20	19	15	14	12	11	7	6	0
0000001		00100		rs1		000		00000		0001011	

15.1.3 DCACHE.CPA——DCACHE 指定物理地址清脏表项指令

语法:

dcache.cpa rs1

操作:

将 rs1 中物理地址所对应的 dcache 表项写回到下一级存储, 操作所有核 L1CACHE。

执行权限:

M mode/S mode

异常:

非法指令异常

说明:

mxstatus.theadisaee=0, 执行该指令产生非法指令异常。

mxstatus.theadisaee=1, U mode 下执行该指令产生非法指令异常。

指令格式:

31	25	24	20	19	15	14	12	11	7	6	0
0000001		01000		rs1		000		00000		0001011	

15.1.4 DCACHE.CSW——DCACHE 指定 way/set 清脏表项并无效指令

语法:

dcache.csw rs1

操作:

按照 rs1 中指定的 way/set 将 L1 dache dirty 表项写回到下一级存储。

执行权限:

M mode/S mode

异常:

非法指令异常

说明:

C906 dache 为四路组相联, rs1[31:30] 为 way 编码, rs1[w:6] 为 set 编码。当 dcache 为 32K 时, w 为 13, dcache 为 64K 时, w 为 14。

- mxstatus.theadisaee=0, 执行该指令产生非法指令异常。
- mxstatus.theadisaee=1, U mode 下执行该指令产生非法指令异常。

指令格式:

31	25	24	20	19	15	14	12	11	7	6	0
0000001		00001		rs1		000		00000		0001011	

15.1.5 DCACHE.IALL——DCACHE 无效全部表项指令

语法:

dcache.iall

操作:

无效所有 L1 dcache 表项。

执行权限:

M mode/S mode

异常:

非法指令异常

说明:

- mxstatus.theadisace=0, 执行该指令产生非法指令异常。
- mxstatus.theadisace=1, U mode 下执行该指令产生非法指令异常。

指令格式:

31	25	24	20	19	15	14	12	11	7	6	0
0000000		00010		00000		000		00000		0001011	

15.1.6 DCACHE.IVA ——DCACHE 指定虚拟地址无效表项指令

语法:

dcache.iva rs1

操作:

将 rs1 中虚拟地址所对应的 dcache 表项无效。

执行权限:

M mode/S mode

异常:

非法指令异常/加载指令页面错误异常

说明:

- mxstatus.theadisace=0, 执行该指令产生非法指令异常。
- mxstatus.theadisace=1, U mode 下执行该指令产生非法指令异常。

指令格式:

31	25	24	20	19	15	14	12	11	7	6	0
0000001		00110		rs1		000		00000		0001011	

15.1.7 DCACHE.IPA ——DCACHE 指定物理地址无效表项指令

语法:

dcache.ipa rs1

操作:

将 rs1 中物理地址所对应的 dcache 表项无效。

执行权限:

M mode/S mode

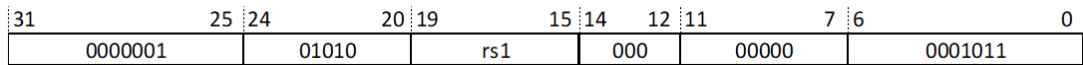
异常:

非法指令异常

说明:

- mxstatus.theadisaee=0, 执行该指令产生非法指令异常。
- mxstatus.theadisaee=1, U mode 下执行该指令产生非法指令异常。

指令格式:



15.1.8 DCACHE.ISW ——DCACHE 指定 set/way 无效表项指令

语法:

dcache.isw rs1

操作:

无效指定 SET 和 WAY 的 dcache 表项。

执行权限:

M mode/S mode

异常:

非法指令异常

说明:

C906 dcache 为两路组相联, rs1[31:30] 为 way 编码, rs1[w:6] 为 set 编码。当 dcache 为 32K 时, w 为 13, dcache 为 64K 时, w 为 14。

- mxstatus.theadisaee=0, 执行该指令产生非法指令异常。

- mxstatus.theadisae=1, U mode 下执行该指令产生非法指令异常。

指令格式:

31	25	24	20	19	15	14	12	11	7	6	0
0000001		00010		rs1		000		00000		0001011	

15.1.9 DCACHE.CIALL——DCACHE 清全部脏表项并无效指令

语法:

dcache.ciall

操作:

将所有 L1 dcache dirty 表项写回到下一级存储后, 无效所有表项。

执行权限:

M mode/S mode

异常:

非法指令异常

说明:

mxstatus.theadisae=0, 执行该指令产生非法指令异常。

mxstatus.theadisae=1, U mode 下执行该指令产生非法指令异常。

指令格式:

31	25	24	20	19	15	14	12	11	7	6	0
0000000		00011		00000		000		00000		0001011	

□

15.1.10 DCACHE.CIVA ——DCACHE 指定虚拟地址清脏表项并无效指令

语法:

dcache.civa rs1

操作:

将 rs1 指定虚拟地址所属的 dcache 表项写回到下级存储, 并无效该表项。

执行权限:

M mode/S mode/U mode

异常:

非法指令异常/加载指令页面错误异常

说明:

mxstatus.theadisae=0, 执行该指令产生非法指令异常。

mxstatus.theadisae=1, mxstatus.ucme =1, U mode 下可以执行该指令。

mxstatus.theadisae=1, mxstatus.ucme =0, U mode 下执行该指令产生非法指令异常。

指令格式:

31	25	24	20	19	15	14	12	11	7	6	0	
0000001			00111		rs1		000		00000		0001011	

15.1.11 DCACHE.CIPA ——DCACHE 指定物理地址清脏表项并无效指令

语法:

dcache.cipa rs1

操作:

将 rs1 中物理地址所属的 dcache 表项写回下级存储并无效该表项。

执行权限:

M mode/S mode

异常:

非法指令异常

说明:

mxstatus.theadisae=0, 执行该指令产生非法指令异常。

mxstatus.theadisae=1, U mode 下执行该指令产生非法指令异常。

指令格式:

31	25	24	20	19	15	14	12	11	7	6	0	
0000001			01011		rs1		000		00000		0001011	

15.1.12 DCACHE.CISW——DCACHE 指定 way/set 清脏表项并无效指令

语法:

dcache.cisw rs1

操作:

按照 rs1 中指定的 way/set 将 L1 dache dirty 表项写回到下一级存储并无效该表项。

执行权限:

M mode/S mode

异常:

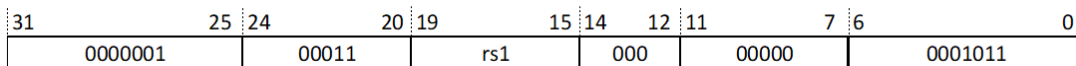
非法指令异常

说明:

C906dache 为四路组相联, rs1[31:30] 为 way 编码, rs1[w:6] 为 set 编码。当 dcache 为 32K 时, w 为 13, dcache 为 64K 时, w 为 14。

- mxstatus.theadisae=0, 执行该指令产生非法指令异常。
- mxstatus.theadisae=1, U mode 下执行该指令产生非法指令异常。

指令格式:



15.1.13 ICACHE.IALL——ICACHE 无效全部表项指令

语法:

icache.iall

操作:

无效所有 icache 表项。

执行权限:

M mode/S mode

异常:

非法指令异常

说明:

- mxstatus.theadisae=0, 执行该指令产生非法指令异常。
- mxstatus.theadisae=1, U mode 下执行该指令产生非法指令异常。

指令格式:

31	25	24	20	19	15	14	12	11	7	6	0
0000000			10000		00000		000		00000		0001011

15.1.14 ICACHE.IALLS——ICACHE 广播无效全部表项指令

语法:

icache.ialls

操作:

无效所有 icache 表项, 并广播其他核去无效各自所有 icache 表项, 操作所有核。

执行权限:

M mode/S mode

异常:

非法指令异常

说明:

mxstatus.theadisae=0, 执行该指令产生非法指令异常。

mxstatus.theadisae=1, U mode 下执行该指令产生非法指令异常。

指令格式:

31	25	24	20	19	15	14	12	11	7	6	0
0000000			10001		00000		000		00000		0001011

15.1.15 ICACHE.IVA——ICACHE 指定虚拟地址无效表项指令

语法:

icache.iva rs1

操作:

将 rs1 中虚拟地址所对应的 icache 表项无效。

执行权限:

M mode/S mode/U mode

异常:

非法指令异常/加载指令页面错误异常

说明:

mxstatus.theadisaee=0, 执行该指令产生非法指令异常。

mxstatus.theadisaee=1, mxstatus.ucme=1, U mode 下可以执行该指令。

mxstatus.theadisaee=1, mxstatus.ucme=0, U mode 下执行该指令产生非法指令异常。

指令格式:

31	25	24	20	19	15	14	12	11	7	6	0
0000001		10000		rs1		000		00000		0001011	

15.1.16 ICACHE.IPA——ICACHE 指定物理地址无效表项指令

语法:

icache.ipa rs1

操作:

将 rs1 中物理地址所对应的 icache 表项无效, 操作所有核。

执行权限:

M mode/S mode

异常:

非法指令异常

说明:

mxstatus.theadisaee=0, 执行该指令产生非法指令异常。

mxstatus.theadisaee=1, U mode 下执行该指令产生非法指令异常。

指令格式:

31	25	24	20	19	15	14	12	11	7	6	0
0000001		11000		rs1		000		00000		0001011	

15.2 附录 B-2 同步指令术语

同步指令扩展每条指令位宽为 32 位。

以下指令按英文字母顺序排列。

15.2.1 SYNC——同步指令

语法:

sync

操作:

该指令保证 sync 前序所有指令比该指令早退休, sync 后序所有指令比该指令晚退休。

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

指令格式:

31	25	24	20	19	15	14	12	11	7	6	0
0000000		11000		00000		000		00000		0001011	

15.2.2 SYNC.I——同步并清空指令

语法:

sync.i

操作:

该指令保证 sync.i 前序所有指令比该指令早退休, sync.i 后序所有指令比该指令晚退休。该指令退休时清空流水线。

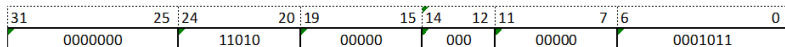
执行权限:

M mode/S mode/U mode

异常:

非法指令异常

指令格式:



15.3 附录 B-3 算术运算指令术语

算术运算指令子集实现了对算术指令的扩展，每条指令位宽为 32 位。

以下指令按英文字母顺序排列。

15.3.1 ADDSL——寄存器移位相加指令

语法：

addsl rd rs1, rs2, imm2

操作：

$rd \leftarrow rs1 + rs2 \ll imm2$

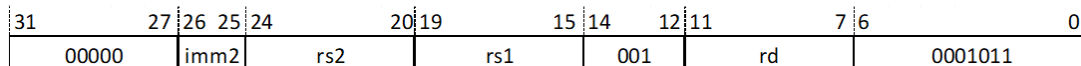
执行权限：

M mode/S mode/U mode

异常：

非法指令异常

指令格式：



15.3.2 MULA——乘累加指令

语法：

mula rd, rs1, rs2

操作：

$rd \leftarrow rd + (rs1 * rs2)[63:0]$

执行权限：

M mode/S mode/U mode

异常:

非法指令异常

指令格式:

31	27	26	25	24	20	19	15	14	12	11	7	6	0
00100	00			rs2			rs1	001			rd		0001011

15.3.3 MULAH——低 16 位乘累加指令

语法:

mulah rd, rs1, rs2

操作:

$tmp[31:0] \leftarrow rd[31:0] + (rs1[15:0] * rs[15:0])$

$rd \leftarrow sign_extend(tmp[31:0])$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

指令格式:

31	27	26	25	24	20	19	15	14	12	11	7	6	0
00101	00			rs2			rs1	001			rd		0001011

15.3.4 MULAW——低 32 位乘累加指令

语法:

mulaw rd, rs1, rs2

操作:

$tmp[31:0] \leftarrow rd[31:0] + (rs1[31:0] * rs[31:0])[31:0]$

$rd \leftarrow sign_extend(tmp[31:0])$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

指令格式:

31	27	26	25	24	20	19	15	14	12	11	7	6	0			
00100				10		rs2			rs1		001		rd		0001011	

15.3.5 MULS——乘累减指令

语法:

mul_s rd, rs1, rs2

操作:

$rd \leftarrow rd - (rs1 * rs2)[63:0]$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

指令格式:

31	27	26	25	24	20	19	15	14	12	11	7	6	0			
00100				01		rs2			rs1		001		rd		0001011	

15.3.6 MULSH——低 16 位乘累减指令

语法:

mul_{sh} rd, rs1, rs2

操作:

$tmp[31:0] \leftarrow rd[31:0] - (rs1[15:0] * rs[15:0])$

$rd \leftarrow \text{sign_extend}(tmp[31:0])$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

指令格式:

31	27	26	25	24	20	19	15	14	12	11	7	6	0
00101	01	rs2			rs1			001	rd			0001011	

15.3.7 MULSW——低 32 位乘累减指令

语法:

mulaw rd, rs1, rs2

操作:

tmp[31:0] ← rd[31:0] - (rs1[31:0] * rs[31:0])
rd ← sign_extend(tmp[31:0])

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

指令格式:

31	27	26	25	24	20	19	15	14	12	11	7	6	0
00100	11	rs2			rs1			001	rd			0001011	

15.3.8 MVEQZ——寄存器为 0 传送指令

语法:

mveqz rd, rs1, rs2

操作: if (rs2 == 0)

rd ← rs1

else

rd ← rd

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

指令格式:

31	27	26	25	24	20	19	15	14	12	11	7	6	0
01000	00	rs2			rs1			001	rd			0001011	

15.3.9 MVNEZ——寄存器非 0 传送指令

语法:

mvnez rd, rs1, rs2

操作:

```
if (rs2 != 0)
    rd ← rs1
else
    rd ← rd
```

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

指令格式:

31	27	26	25	24	20	19	15	14	12	11	7	6	0
01000		01		rs2		rs1		001		rd		0001011	

15.3.10 SRRI——循环右移指令

语法:

srrl rd, rs1, imm6

操作:

```
rd ← rs1 >>>> imm6
rs1 原值右移，左侧移入右侧移出位
```

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

指令格式:

31	26	25	20	19	15	14	12	11	7	6	0
000100		imm6		rs1		001		rd		0001011	

15.3.11 SRRIW——低 32 位循环右移指令

语法:

srriw rd, rs1, imm5

操作:

$rd \leftarrow \text{sign_extend}(rs1[31:0] \gg \gg \gg \text{imm5})$

rs1[31:0] 原值右移, 左侧移入右侧移出位

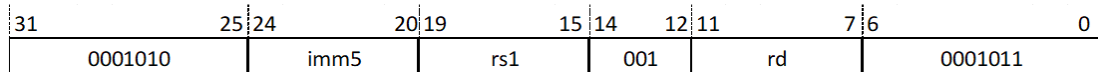
执行权限:

M mode/S mode/U mode

异常:

非法指令异常

指令格式:



15.4 附录 B-3 位操作指令术语

位操作指令子集实现了对位运算指令的扩展, 每条指令位宽为 32 位。

以下指令按英文字母顺序排列。

15.4.1 EXT——寄存器连续位提取符号位扩展指令

语法:

ext rd, rs1, imm1,imm2

操作:

$rd \leftarrow \text{sign_extend}(rs1[\text{imm1}:\text{imm2}])$

执行权限:

M mode/S mode/U mode

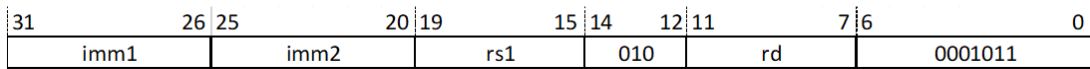
异常:

非法指令异常

说明:

若 $imm1 < imm2$ ，该指令行为不可预测

指令格式：



15.4.2 EXTU——寄存器连续位提取零扩展指令

语法：

`extu rd, rs1, imm1,imm2`

操作：

$rd \leftarrow \text{zero_extend}(rs1[imm1:imm2])$

执行权限：

M mode/S mode/U mode

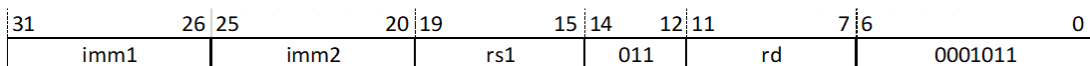
异常：

非法指令异常

说明：

若 $imm1 < imm2$ ，该指令行为不可预测

指令格式：



15.4.3 FFO——快速找 0 指令

语法：

`ff0 rd, rs1`

操作：

从 `rs1` 最高位开始查找第一个为 0 的位，结果写回 `rd`。如果 `rs1` 的最高位为 0，则结果为 0，如果 `rs1` 中没有 0，结果为 64

执行权限：

M mode/S mode/U mode

异常：

非法指令异常

指令格式:

31	27	26	25	24	20	19	15	14	12	11	7	6	0
10000	10	00000	rs1	001	rd	0001011							

15.4.4 FF1——快速找 1 指令

语法:

ff1 rd, rs1

操作:

从 rs1 最高位开始查找第一个为 1 的位, 将该位的索引写回 rd。如果 rs1 的最高位为 1, 则结果为 0, 如果 rs1 中没有 1, 结果为 64

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

指令格式:

31	27	26	25	24	20	19	15	14	12	11	7	6	0
10000	11	00000	rs1	001	rd	0001011							

15.4.5 REV——字节倒序指令

语法:

rev rd, rs1

操作:

rd[63:56] ←rs1[7:0]
 rd[55:48] ←rs1[15:8]
 rd[47:40] ←rs1[23:16]
 rd[39:32] ←rs1[31:24]
 rd[31:24] ←rs1[39:32]
 rd[23:16] ←rs1[47:40]
 rd[15:8] ←rs1[55:48]
 rd[7:0] ←rs1[63:56]

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

指令格式:

31	27	26	25	24	20	19	15	14	12	11	7	6	0
10000		01		00000		rs1		001		rd		0001011	

15.4.6 REVW——低 32 位字节倒序指令

语法:

revw rd, rs1

操作:

tmp[31:24] ←rs1[7:0]
 tmp [23:16] ←rs1[15:8]
 tmp [15:8] ←rs1[23:16]
 tmp [7:0] ←rs1[31:24]
 rd ←sign_extend(tmp[31:0])

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

指令格式:

31	27	26	25	24	20	19	15	14	12	11	7	6	0
10010		00		00000		rs1		001		rd		0001011	

15.4.7 TST——比特为 0 测试指令

语法:

tst rd, rs1, imm6

操作:

if(rs1[imm6] == 1)
 rd←-1
 else

$rd \leftarrow 0$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

指令格式:

31	26	25	20	19	15	14	12	11	7	6	0
100010		imm6		rs1		001		rd		0001011	

15.4.8 TSTNBZ——字节为 0 测试指令

语法:

tstnbz rd, rs1

操作:

$rd[63:56] \leftarrow (rs1[63:56] == 0) ? 8'hff : 8'h0$

$rd[55:48] \leftarrow (rs1[55:48] == 0) ? 8'hff : 8'h0$

$rd[47:40] \leftarrow (rs1[47:40] == 0) ? 8'hff : 8'h0$

$rd[39:32] \leftarrow (rs1[39:32] == 0) ? 8'hff : 8'h0$

$rd[31:24] \leftarrow (rs1[31:24] == 0) ? 8'hff : 8'h0$

$rd[23:16] \leftarrow (rs1[23:16] == 0) ? 8'hff : 8'h0$

$rd[15:8] \leftarrow (rs1[15:8] == 0) ? 8'hff : 8'h0$

$rd[7:0] \leftarrow (rs1[7:0] == 0) ? 8'hff : 8'h0$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

指令格式:

31	27	26	25	24	20	19	15	14	12	11	7	6	0
10000		00		00000		rs1		001		rd		0001011	

15.5 附录 B-4 存储指令术语

存储指令子集实现了对存储指令的扩展，每条指令位宽为 32 位。

以下指令按英文字母顺序排列。

15.5.1 FLRD——浮点寄存器移位双字加载指令

语法:

`fldr rd, rs1, rs2, imm2`

操作:

$rd \leftarrow \text{mem}[(rs1+rs2 \ll imm2)+7: (rs1+rs2 \ll imm2)]$

执行权限:

M mode/S mode/U mode

异常:

加载指令非对齐访问异常、加载指令访问错误异常、加载指令页面错误异常,非法指令异常

说明:

`mxstatus.theadisae=1' b0` 或 `mstatus.fs =2' b00` 时,该指令产生非法指令异常

指令格式:

31	27	26	25	24	20	19	15	14	12	11	7	6	0
01100		imm2		rs2		rs1		110		rd		0001011	

15.5.2 FLRW——浮点寄存器移位字加载指令

语法:

`flrw rd, rs1, rs2, imm2`

操作:

$rd \leftarrow \text{one_extend}(\text{mem}[(rs1+rs2 \ll imm2)+3: (rs1+rs2 \ll imm2)])$

执行权限:

M mode/S mode/U mode

异常:

加载指令非对齐访问异常、加载指令访问错误异常、加载指令页面错误异常,非法指令异常

说明:

`mxstatus.theadisae=1' b0` 或 `mstatus.fs =2' b00` 时,该指令产生非法指令异常

指令格式:

31	27	26	25	24	20	19	15	14	12	11	7	6	0
01000		imm2		rs2		rs1		110		rd		0001011	

15.5.3 FLURD——浮点寄存器低 32 位移位双字加载指令

语法:

flurd rd, rs1, rs2, imm2

操作:

$rd \leftarrow \text{mem}[(rs1+rs2[31:0] \ll imm2)+7: (rs1+rs2[31:0] \ll imm2)]$

执行权限:

M mode/S mode/U mode

异常:

加载指令非对齐访问异常、加载指令访问错误异常、加载指令页面错误异常、非法指令异常

说明:

rs2[31:0] 是无符号数, 高位 [63:32] 补 0 进行地址运算

mxstatus.theadisaee=1' b0 或 mstatus.fs = 2' b00 时, 该指令产生非法指令异常

指令格式:

31	27	26	25	24	20	19	15	14	12	11	7	6	0
01110		imm2		rs2		rs1		110		rd		0001011	

15.5.4 FLURW——浮点寄存器低 32 位移位字加载指令

语法:

flurw rd, rs1, rs2, imm2

操作:

$rd \leftarrow \text{one_extend}(\text{mem}[(rs1+rs2[31:0] \ll imm2)+3: (rs1+rs2[31:0] \ll imm2)])$

执行权限:

M mode/S mode/U mode

异常:

加载指令非对齐访问异常、加载指令访问错误异常、加载指令页面错误异常、非法指令异常

说明:

rs2[31:0] 是无符号数, 高位 [63:32] 补 0 进行地址运算

mxstatus.theadisaee=1' b0 或 mstatus.fs = 2' b00 时, 该指令产生非法指令异常

指令格式:

31	27	26	25	24	20	19	15	14	12	11	7	6	0
01010				imm2	rs2		rs1		110		rd		0001011

15.5.5 FSRD——浮点寄存器移位双字存储指令

语法:

fsrd rd, rs1, rs2, imm2

操作:

$\text{mem}[(\text{rs1}+\text{rs2}\ll\text{imm2})+7:(\text{rs1}+\text{rs2}\ll\text{imm2})] \leftarrow \text{rd}[63:0]$

执行权限:

M mode/S mode/U mode

异常:

存储指令非对齐访问异常、存储指令访问错误异常、存储指令页面错误异常、非法指令异常

说明:

mxstatus.theadisaee=1' b0 或 mstatus.fs =2' b00 时, 该指令产生非法指令异常

指令格式:

31	27	26	25	24	20	19	15	14	12	11	7	6	0
01100				imm2	rs2		rs1		111		rd		0001011

15.5.6 FSRW——浮点寄存器移位字存储指令

语法:

fsrw rd, rs1, rs2, imm2

操作:

$\text{mem}[(\text{rs1}+\text{rs2}\ll\text{imm2})+3:(\text{rs1}+\text{rs2}\ll\text{imm2})] \leftarrow \text{rd}[31:0]$

执行权限:

M mode/S mode/U mode

异常:

存储指令非对齐访问异常、存储指令访问错误异常、存储指令页面错误异常、非法指令异常

说明:

mxstatus.theadisaee=1' b0 或 mstatus.fs =2' b00 时, 该指令产生非法指令异常

指令格式:

31	27	26	25	24	20	19	15	14	12	11	7	6	0
01000		imm2		rs2		rs1		111		rd		0001011	

15.5.7 FSURD——浮点寄存器低 32 位移位双字存储指令

语法:

fsurd rd, rs1, rs2, imm2

操作:

$\text{mem}[(rs1+rs2[31:0] \ll imm2)+7: (rs1+rs2[31:0] \ll imm2)] \leftarrow rd[63:0]$

执行权限:

M mode/S mode/U mode

异常:

存储指令非对齐访问异常、存储指令访问错误异常、存储指令页面错误异常、非法指令异常

说明:

rs2[31:0] 是无符号数, 高位 [63:32] 补 0 进行地址运算

mxstatus.theadisae=1' b0 或 mstatus.fs = 2' b00 时, 该指令产生非法指令异常

指令格式:

31	27	26	25	24	20	19	15	14	12	11	7	6	0
01110		imm2		rs2		rs1		111		rd		0001011	

15.5.8 FSURW——浮点寄存器低 32 位移位字存储指令

语法:

fsurw rd, rs1, rs2, imm2

操作:

$\text{mem}[(rs1+rs2[31:0] \ll imm2)+3: (rs1+rs2[31:0] \ll imm2)] \leftarrow rd[31:0]$

执行权限:

M mode/S mode/U mode

异常:

存储指令非对齐访问异常、存储指令访问错误异常、存储指令页面错误异常、非法指令异常

说明:

rs2[31:0] 是无符号数, 高位 [63:32] 补 0 进行地址运算

mxstatus.theadisae=1' b0 或 mstatus.fs = 2' b00 时, 该指令产生非法指令异常

指令格式:

31	27	26	25	24	20	19	15	14	12	11	7	6	0
01010		imm2		rs2		rs1		111		rd		0001011	

15.5.9 LBIA——符号位扩展字节加载基地址自增指令

语法:

lbia rd, (rs1), imm5,imm2

操作:

$rd \leftarrow \text{sign_extend}(\text{mem}[\text{rs1}])$

$\text{rs1} \leftarrow \text{rs1} + \text{sign_extend}(\text{imm5} \ll \text{imm2})$

执行权限:

M mode/S mode/U mode

异常:

加载指令非对齐访问异常、加载指令访问错误异常、加载指令页面错误异常、非法指令异常。

说明:

rd 和 rs1 不可相等

指令格式:

31	27	26	25	24	20	19	15	14	12	11	7	6	0
00011		imm2		imm5		rs1		100		rd		0001011	

15.5.10 LBIB——基地址自增符号位扩展字节加载指令

语法:

lbib rd, (rs1), imm5,imm2

操作:

$\text{rs1} \leftarrow \text{rs1} + \text{sign_extend}(\text{imm5} \ll \text{imm2})$

$rd \leftarrow \text{sign_extend}(\text{mem}[\text{rs1}+7:\text{rs1}])$

执行权限:

M mode/S mode/U mode

异常:

加载指令非对齐访问异常、加载指令访问错误异常、加载指令页面错误异常、非法指令异常。

说明:

rd 和 rs1 不可相等

指令格式:

31	27	26	25	24	20	19	15	14	12	11	7	6	0			
00001				imm2		imm5			rs1		100		rd		0001011	

15.5.11 LBUIA——零扩展字节加载基地址自增指令

语法:

lbuia rd, (rs1), imm5,imm2

操作:

rd ← zero_extend(mem[rs1])
rs1 ← rs1 + sign_extend(imm5 << imm2)

执行权限:

M mode/S mode/U mode

异常:

加载指令非对齐访问异常、加载指令访问错误异常、加载指令页面错误异常、非法指令异常。

说明:

rd 和 rs1 不可相等

指令格式:

31	27	26	25	24	20	19	15	14	12	11	7	6	0			
10011				imm2		imm5			rs1		100		rd		0001011	

15.5.12 LBUIB——基地址自增零扩展字节加载指令

语法:

lbuib rd, (rs1), imm5,imm2

操作:

$rs1 \leftarrow rs1 + \text{sign_extend}(\text{imm5} \ll \text{imm2})$
 $rd \leftarrow \text{zero_extend}(\text{mem}[rs1])$

执行权限:

M mode/S mode/U mode

异常:

加载指令非对齐访问异常、加载指令访问错误异常、加载指令页面错误异常、非法指令异常。

说明:

rd 和 rs1 不可相等

指令格式:

31	27	26	25	24	20	19	15	14	12	11	7	6	0		
10001				imm2		imm5		rs1		100		rd		0001011	

15.5.13 LDD——双寄存器加载指令

语法:

ldd rd1,rd2, (rs1),imm2

操作:

$\text{address} \leftarrow rs1 + \text{zero_extend}(\text{imm2} \ll 4)$
 $rd1 \leftarrow \text{mem}[\text{address}+7:\text{address}]$
 $rd2 \leftarrow \text{mem}[\text{address}+15:\text{address}+8]$

执行权限:

M mode/S mode/U mode

异常:

加载指令非对齐访问异常、加载指令访问错误异常、加载指令页面错误异常、非法指令异常

说明:

rd1,rd2 ,rs1 互相不可相等

指令格式:

31	27	26	25	24	20	19	15	14	12	11	7	6	0		
11111				imm2		rd2		rs1		100		rd1		0001011	

15.5.14 LDIA——符号位扩展双字加载基地址自增指令

语法:

ldia rd, (rs1), imm5,imm2

操作:

rd ← sign_extend(mem[rs1+7:rs1])
rs1 ← rs1 + sign_extend(imm5 << imm2)

执行权限:

M mode/S mode/U mode

异常:

加载指令非对齐访问异常、加载指令访问错误异常、加载指令页面错误异常、非法指令异常。

说明:

rd 和 rs1 不可相等

指令格式:

31	27	26	25	24	20	19	15	14	12	11	7	6	0	
01111				imm2	imm5			rs1		100		rd	0001011	

15.5.15 LDIB——基地址自增符号位扩展双字加载指令

语法:

ldib rd, (rs1), imm5,imm2

操作:

rs1 ← rs1 + sign_extend(imm5 << imm2)
rd ← sign_extend(mem[rs1+7:rs1])

执行权限:

M mode/S mode/U mode

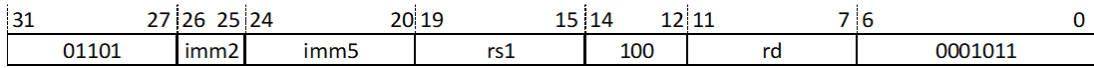
异常:

加载指令非对齐访问异常、加载指令访问错误异常、加载指令页面错误异常、非法指令异常。

说明:

rd 和 rs1 不可相等

指令格式:



15.5.16 LHIA——符号位扩展半字加载基地址自增指令

语法:

lhia rd, (rs1), imm5,imm2

操作:

$rd \leftarrow \text{sign_extend}(\text{mem}[\text{rs1}+1:\text{rs1}])$
 $\text{rs1} \leftarrow \text{rs1} + \text{sign_extend}(\text{imm5} \ll \text{imm2})$

执行权限:

M mode/S mode/U mode

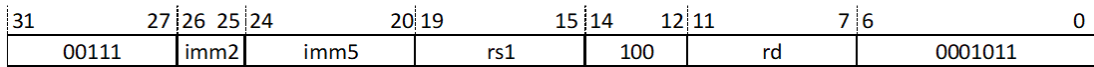
异常:

加载指令非对齐访问异常、加载指令访问错误异常、加载指令页面错误异常、非法指令异常。

说明:

rd 和 rs1 不可相等

指令格式:



15.5.17 LHIB——基地址自增符号位扩展半字加载指令

语法:

lhib rd, (rs1), imm5,imm2

操作:

$\text{rs1} \leftarrow \text{rs1} + \text{sign_extend}(\text{imm5} \ll \text{imm2})$
 $rd \leftarrow \text{sign_extend}(\text{mem}[\text{rs1}+1:\text{rs1}])$

执行权限:

M mode/S mode/U mode

异常:

加载指令非对齐访问异常、加载指令访问错误异常、加载指令页面错误异常、非法指令异常。

说明:

rd 和 rs1 不可相等

指令格式:

31	27	26	25	24	20	19	15	14	12	11	7	6	0
00101		imm2		imm5		rs1		100		rd		0001011	

15.5.18 LHUIA——零扩展半字加载基地址自增指令

语法:

lhua rd, (rs1), imm5,imm2

操作:

rd \leftarrow zero_extend(mem[rs1+1:rs1])
rs1 \leftarrow rs1 + sign_extend(imm5 << imm2)

执行权限:

M mode/S mode/U mode

异常:

加载指令非对齐访问异常、加载指令访问错误异常、加载指令页面错误异常、非法指令异常。

说明:

rd 和 rs1 不可相等

指令格式:

31	27	26	25	24	20	19	15	14	12	11	7	6	0
10111		imm2		imm5		rs1		100		rd		0001011	

15.5.19 LHUIB——基地址自增零扩展半字加载指令

语法:

lhuib rd, (rs1), imm5,imm2

操作:

rs1 \leftarrow rs1 + sign_extend(imm5 << imm2)
rd \leftarrow zero_extend(mem[rs1+1:rs1])

执行权限:

M mode/S mode/U mode

异常:

加载指令非对齐访问异常、加载指令访问错误异常、加载指令页面错误异常、非法指令异常。

说明:

rd 和 rs1 不可相等

指令格式:

31	27	26	25	24	20	19	15	14	12	11	7	6	0
10101		imm2		imm5		rs1		100		rd		0001011	

15.5.20 LRB——寄存器移位符号位扩展字节加载指令

语法:

lrb rd, rs1, rs2, imm2

操作:

$rd \leftarrow \text{sign_extend}(\text{mem}[(rs1+rs2 \ll imm2)])$

执行权限:

M mode/S mode/U mode

异常:

加载指令非对齐访问异常、加载指令访问错误异常、加载指令页面错误异常、非法指令异常

指令格式:

31	27	26	25	24	20	19	15	14	12	11	7	6	0
00000		imm2		rs2		rs1		100		rd		0001011	

15.5.21 LRBU——寄存器移位零扩展扩展字节加载指令

语法:

lrbu rd, rs1, rs2, imm2

操作:

$rd \leftarrow \text{zero_extend}(\text{mem}[(rs1+rs2 \ll imm2)])$

执行权限:

M mode/S mode/U mode

异常:

加载指令非对齐访问异常、加载指令访问错误异常、加载指令页面错误异常、非法指令异常

指令格式:

31	27	26	25	24	20	19	15	14	12	11	7	6	0
10000		imm2		rs2	rs1		100		rd		0001011		

15.5.22 LRD——寄存器移位双字加载指令

语法:

lrd rd, rs1, rs2, imm2

操作:

$rd \leftarrow \text{mem}[(rs1+rs2 \ll imm2)+7: (rs1+rs2 \ll imm2)]$

执行权限:

M mode/S mode/U mode

异常:

加载指令非对齐访问异常、加载指令访问错误异常、加载指令页面错误异常、非法指令异常

指令格式:

31	27	26	25	24	20	19	15	14	12	11	7	6	0
01100		imm2		rs2	rs1		100		rd		0001011		

15.5.23 LRH——寄存器移位符号位扩展半字加载指令

语法:

lrh rd, rs1, rs2, imm2

操作:

$rd \leftarrow \text{sign_extend}(\text{mem}[(rs1+rs2 \ll imm2)+1: (rs1+rs2 \ll imm2)])$

执行权限:

M mode/S mode/U mode

异常:

加载指令非对齐访问异常、加载指令访问错误异常、加载指令页面错误异常、非法指令异常

指令格式:

31	27	26	25	24	20	19	15	14	12	11	7	6	0
00100		imm2		rs2	rs1		100		rd		0001011		

15.5.24 LRHU——寄存器移位零扩展扩展半字加载指令

语法:

lrhu rd, rs1, rs2, imm2

操作:

$rd \leftarrow \text{zero_extend}(\text{mem}[(rs1+rs2 \ll imm2)+1: (rs1+rs2 \ll imm2)])$

执行权限:

M mode/S mode/U mode

异常:

加载指令非对齐访问异常、加载指令访问错误异常、加载指令页面错误异常、非法指令异常

指令格式:

31	27	26	25	24	20	19	15	14	12	11	7	6	0
10100		imm2		rs2		rs1		100		rd		0001011	

15.5.25 LRW——寄存器移位符号位扩展字加载指令

语法:

lrw rd, rs1, rs2, imm2

操作:

$rd \leftarrow \text{sign_extend}(\text{mem}[(rs1+rs2 \ll imm2)+3: (rs1+rs2 \ll imm2)])$

执行权限:

M mode/S mode/U mode

异常:

加载指令非对齐访问异常、加载指令访问错误异常、加载指令页面错误异常、非法指令异常

指令格式:

31	27	26	25	24	20	19	15	14	12	11	7	6	0
01000		imm2		rs2		rs1		100		rd		0001011	

15.5.26 LRWU——寄存器移位零扩展扩展字加载指令

语法:

lrwu rd, rs1, rs2, imm2

操作:

$rd \leftarrow \text{zero_extend}(\text{mem}[(rs1+rs2 \ll imm2)+3: (rs1+rs2 \ll imm2)])$

执行权限:

M mode/S mode/U mode

异常:

加载指令非对齐访问异常、加载指令访问错误异常、加载指令页面错误异常、非法指令异常

指令格式:

31	27	26	25	24	20	19	15	14	12	11	7	6	0
11000		imm2		rs2		rs1		100		rd		0001011	

15.5.27 LURB——寄存器低 32 位移位符号位扩展字节加载指令

语法:

lurb rd, rs1, rs2, imm2

操作:

$rd \leftarrow \text{sign_extend}(\text{mem}[(rs1+rs2[31:0] \ll imm2)])$

执行权限:

M mode/S mode/U mode

异常:

加载指令非对齐访问异常、加载指令访问错误异常、加载指令页面错误异常、非法指令异常

说明:

rs2[31:0] 是无符号数, 高位 [63:32] 补 0 进行地址运算

指令格式:

31	27	26	25	24	20	19	15	14	12	11	7	6	0
00010		imm2		rs2		rs1		100		rd		0001011	

15.5.28 LURBU——寄存器低 32 位移位零扩展字节加载指令

语法:

lurbu rd, rs1, rs2, imm2

操作:

$rd \leftarrow \text{zero_extend}(\text{mem}[(rs1+rs2[31:0] \ll imm2)])$

执行权限:

M mode/S mode/U mode

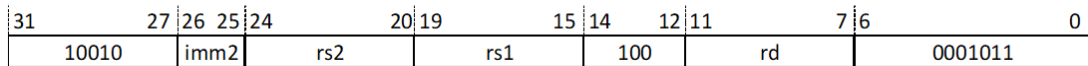
异常:

加载指令非对齐访问异常、加载指令访问错误异常、加载指令页面错误异常、非法指令异常

说明:

rs2[31:0] 是无符号数, 高位 [63:32] 补 0 进行地址运算

指令格式:



15.5.29 LURD——寄存器低 32 位移位双字加载指令

语法:

lurd rd, rs1, rs2, imm2

操作:

$rd \leftarrow \text{mem}[(rs1+rs2[31:0] \ll imm2)+7: (rs1+rs2[31:0] \ll imm2)]$

执行权限:

M mode/S mode/U mode

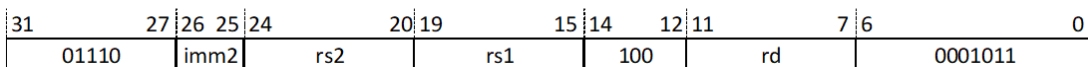
异常:

加载指令非对齐访问异常、加载指令访问错误异常、加载指令页面错误异常、非法指令异常

说明:

rs2[31:0] 是无符号数, 高位 [63:32] 补 0 进行地址运算

指令格式:



15.5.30 LURH——寄存器低 32 位移位符号位扩展半字加载指令

语法:

`lurh rd, rs1, rs2, imm2`

操作:

$$rd \leftarrow \text{sign_extend}(\text{mem}[(rs1+rs2[31:0] \ll imm2)+1: (rs1+rs2[31:0] \ll imm2)])$$

执行权限:

M mode/S mode/U mode

异常:

加载指令非对齐访问异常、加载指令访问错误异常、加载指令页面错误异常、非法指令异常

说明:

rs2[31:0] 是无符号数, 高位 [63:32] 补 0 进行地址运算

指令格式:

31	27	26	25	24	20	19	15	14	12	11	7	6	0
00110		imm2		rs2		rs1		100		rd		0001011	

15.5.31 LURHU——寄存器低 32 位移位零扩展半字加载指令

语法:

`lurhu rd, rs1, rs2, imm2`

操作:

$$rd \leftarrow \text{zero_extend}(\text{mem}[(rs1+rs2[31:0] \ll imm2)+1: (rs1+rs2[31:0] \ll imm2)])$$

执行权限:

M mode/S mode/U mode

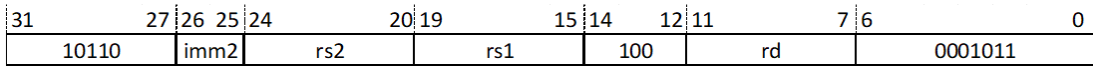
异常:

加载指令非对齐访问异常、加载指令访问错误异常、加载指令页面错误异常、非法指令异常

说明:

rs2[31:0] 是无符号数, 高位 [63:32] 补 0 进行地址运算

指令格式:



15.5.32 LURW——寄存器低 32 位移位符号位扩展字加载指令

语法:

lurw rd, rs1, rs2, imm2

操作:

$rd \leftarrow \text{sign_extend}(\text{mem}[(rs1+rs2[31:0] \ll imm2)+3: (rs1+rs2[31:0] \ll imm2)])$

执行权限:

M mode/S mode/U mode

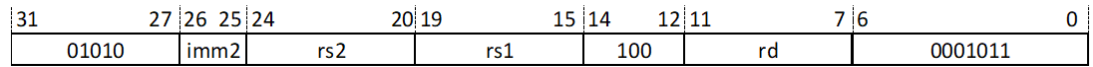
异常:

加载指令非对齐访问异常、加载指令访问错误异常、加载指令页面错误异常、非法指令异常

说明:

rs2[31:0] 是无符号数, 高位 [63:32] 补 0 进行地址运算

指令格式:



15.5.33 LURWU——寄存器低 32 位移位零扩展字加载指令

语法:

lwd rd1, rd2, (rs1),imm2

操作:

$\text{address} \leftarrow rs1 + \text{zero_extend}(imm2 \ll 3)$
 $rd1 \leftarrow \text{sign_extend}(\text{mem}[\text{address}+3: \text{address}])$
 $rd2 \leftarrow \text{sign_extend}(\text{mem}[\text{address}+7: \text{address}+4])$

执行权限:

M mode/S mode/U mode

异常:

加载指令非对齐访问异常、加载指令访问错误异常、加载指令页面错误异常、非法指令异常

说明:

rd1,rd2 ,rs1 互相不可相等

指令格式:

31	27	26	25	24	20	19	15	14	12	11	7	6	0		
11010				imm2		rs2		rs1		100		rd		0001011	

15.5.34 LWD——符号位扩展双寄存器字加载指令

语法:

lwd rd, imm7(rs1)

操作:

address ← rs1 + sign_extend(imm7)

rd ← sign_extend(mem[address+31: address])

rd+1 ← sign_extend(mem[address+63: address+32])

执行权限:

M mode/S mode/U mode

异常:

加载指令非对齐访问异常、加载指令访问错误异常、加载指令页面错误异常、非法指令异常

指令格式:

31	27	26	25	24	20	19	15	14	12	11	7	6	0		
11100				imm2		rd2		rs1		100		rd1		0001011	

15.5.35 LWIA——符号位扩展字加载基地址自增指令

语法:

lwia rd, (rs1), imm5,imm2

操作:

rd ← sign_extend(mem[rs1+3:rs1])

rs1 ← rs1 + sign_extend(imm5 << imm2)

执行权限:

M mode/S mode/U mode

异常:

加载指令非对齐访问异常、加载指令访问错误异常、加载指令页面错误异常、非法指令异常。

说明:

rd 和 rs1 不可相等

指令格式:

31	27	26	25	24	20	19	15	14	12	11	7	6	0
01011	imm2			imm5			rs1		100		rd		0001011

15.5.36 LWIB——基地址自增符号位扩展字加载指令

语法:

lwib rd, (rs1), imm5,imm2

操作:

$rs1 \leftarrow rs1 + \text{sign_extend}(imm5 \ll imm2)$

$rd \leftarrow \text{sign_extend}(\text{mem}[rs1+3:rs1])$

执行权限:

M mode/S mode/U mode

异常:

加载指令非对齐访问异常、加载指令访问错误异常、加载指令页面错误异常、非法指令异常。

说明:

rd 和 rs1 不可相等

指令格式:

31	27	26	25	24	20	19	15	14	12	11	7	6	0
01001	imm2			imm5			rs1		100		rd		0001011

15.5.37 LWUD——零扩展双寄存器字加载指令

语法:

lwud rd1,rd2, (rs1),imm2

操作:

$address \leftarrow rs1 + \text{zero_extend}(imm2 \ll 3)$

$rd1 \leftarrow \text{zero_extend}(\text{mem}[address+3: address])$

$rd2 \leftarrow \text{zero_extend}(\text{mem}[address+7: address+4])$

执行权限:

M mode/S mode/U mode

异常:

加载指令非对齐访问异常、加载指令访问错误异常、加载指令页面错误异常、非法指令异常

说明:

rd1,rd2 ,rs1 互相不可相等

指令格式:

31	27	26	25	24	20	19	15	14	12	11	7	6	0
11110		imm2		rd2		rs1		100		rd1		0001011	

15.5.38 LWUIA——零扩展字加载基地址自增指令

语法:

lwuia rd, (rs1), imm5,imm2

操作:

rd ← zero_extend(mem[rs1+3:rs1])
rs1 ← rs1 + sign_extend(imm5 << imm2)

执行权限:

M mode/S mode/U mode

异常:

加载指令非对齐访问异常、加载指令访问错误异常、加载指令页面错误异常、非法指令异常。

说明:

rd 和 rs1 不可相等

指令格式:

31	27	26	25	24	20	19	15	14	12	11	7	6	0
11011		imm2		imm5		rs1		100		rd		0001011	

15.5.39 LWUIB——基地址自增零扩展字加载指令

语法:

lwuib rd, (rs1), imm5,imm2

操作:

$rs1 \leftarrow rs1 + \text{sign_extend}(imm5 \ll imm2)$

$rd \leftarrow \text{zero_extend}(\text{mem}[rs1+3:rs1])$

执行权限:

M mode/S mode/U mode

异常:

加载指令非对齐访问异常、加载指令访问错误异常、加载指令页面错误异常、非法指令异常。

说明:

rd 和 rs1 不可相等

指令格式:

31	27	26	25	24	20	19	15	14	12	11	7	6	0	
11001				imm2	imm5			rs1		100		rd	0001011	

15.5.40 SBIA——字节存储基地址自增指令

语法:

sbia rs2, (rs1), imm5,imm2

操作:

$\text{mem}[rs1] \leftarrow rs2[7:0]$

$rs1 \leftarrow rs1 + \text{sign_extend}(imm5 \ll imm2)$

执行权限:

M mode/S mode/U mode

异常:

存储指令非对齐访问异常、存储指令访问错误异常、存储指令页面错误异常、非法指令异常

指令格式:

31	27	26	25	24	20	19	15	14	12	11	7	6	0	
00011				imm2	imm5			rs1		101		rs2	0001011	

15.5.41 SBIB——基地址自增字节存储指令

语法:

sbib rs2, (rs1), imm5,imm2

操作:

$rs1 \leftarrow rs1 + \text{sign_extend}(imm5 \ll imm2)$

$mem[rs1] \leftarrow rs2[7:0]$

执行权限:

M mode/S mode/U mode

异常:

存储指令非对齐访问异常、存储指令访问错误异常、存储指令页面错误异常、非法指令异常

指令格式:

31	27	26	25	24	20	19	15	14	12	11	7	6	0
00001	imm2			imm5			rs1		101		rs2		0001011

15.5.42 SDD——双寄存器存储指令

语法:

sdd rd1,rd2, (rs1),imm2

操作:

$address \leftarrow rs1 + \text{zero_extend}(imm2 \ll 4)$

$mem[address+7:address] \leftarrow rd1$

$mem[address+15:address+8] \leftarrow rd2$

执行权限:

M mode/S mode/U mode

异常:

存储指令非对齐访问异常、存储指令访问错误异常、存储指令页面错误异常、非法指令异常

指令格式:

31	27	26	25	24	20	19	15	14	12	11	7	6	0
11111	imm2			rd2			rs1		101		rd1		0001011

15.5.43 SDIA——双字存储基地址自增指令

语法:

`sdia rs2, (rs1), imm5,imm2`

操作:

$mem[rs1+7:rs1] \leftarrow rs2[63:0]$
 $rs1 \leftarrow rs1 + sign_extend(imm5 \ll imm2)$

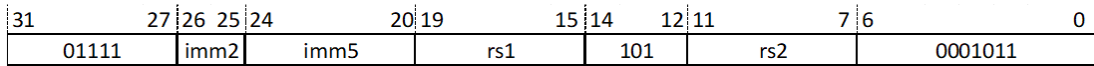
执行权限:

M mode/S mode/U mode

异常:

存储指令非对齐访问异常、存储指令访问错误异常、存储指令页面错误异常、非法指令异常

指令格式:



15.5.44 SDIB——基地址自增双字存储指令

语法:

`sdib rs2, (rs1), imm5,imm2`

操作:

$rs1 \leftarrow rs1 + sign_extend(imm5 \ll imm2)$
 $mem[rs1+7:rs1] \leftarrow rs2[63:0]$

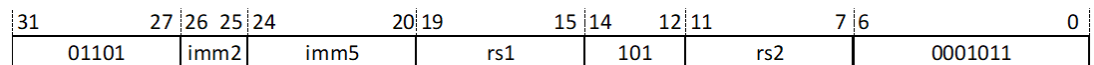
执行权限:

M mode/S mode/U mode

异常:

存储指令非对齐访问异常、存储指令访问错误异常、存储指令页面错误异常、非法指令异常

指令格式:



15.5.45 SHIA——半字存储基地址自增指令

语法:

shia rs2, (rs1), imm5,imm2

操作:

mem[rs1+1:rs1]←rs2[15:0]
rs1←rs1 + sign_extend(imm5 << imm2)

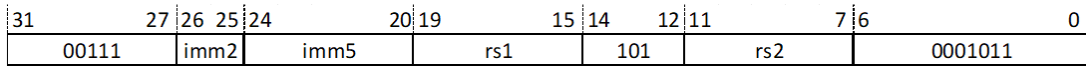
执行权限:

M mode/S mode/U mode

异常:

存储指令非对齐访问异常、存储指令访问错误异常、存储指令页面错误异常、非法指令异常

指令格式:



15.5.46 SHIB——基地址自增半字存储指令

语法:

shib rs2, (rs1), imm5,imm2

操作:

rs1←rs1 + sign_extend(imm5 << imm2)
mem[rs1+1:rs1] ←rs2[15:0]

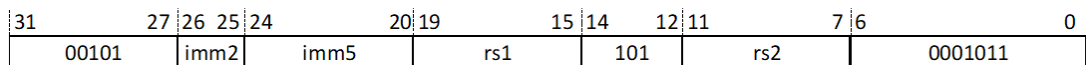
执行权限:

M mode/S mode/U mode

异常:

存储指令非对齐访问异常、存储指令访问错误异常、存储指令页面错误异常、非法指令异常

指令格式:



15.5.47 SRB——寄存器移位字节存储指令

语法:

srb rd, rs1, rs2, imm2

操作:

mem[(rs1+rs2<<imm2)] ←rd[7:0]

执行权限:

M mode/S mode/U mode

异常:

存储指令非对齐访问异常、存储指令访问错误异常、存储指令页面错误异常、非法指令异常

指令格式:

31	27	26	25	24	20	19	15	14	12	11	7	6	0
00000		imm2		imm5		rs1		101		rd		0001011	

15.5.48 SRD——寄存器移位双字存储指令

语法:

srd rd, rs1, rs2, imm2

操作:

mem[(rs1+rs2<<imm2)+7: (rs1+rs2<<imm2)] ←rd[63:0]

执行权限:

M mode/S mode/U mode

异常:

存储指令非对齐访问异常、存储指令访问错误异常、存储指令页面错误异常、非法指令异常

指令格式:

31	27	26	25	24	20	19	15	14	12	11	7	6	0
01100		imm2		rs2		rs1		101		rd		0001011	

15.5.49 SRH——寄存器移位半字存储指令

语法:

srh rd, rs1, rs2, imm2

操作:

$\text{mem}[(\text{rs1}+\text{rs2}\ll\text{imm2})+1: (\text{rs1}+\text{rs2}\ll\text{imm2})] \leftarrow \text{rd}[15:0]$

执行权限:

M mode/S mode/U mode

异常:

存储指令非对齐访问异常、存储指令访问错误异常、存储指令页面错误异常、非法指令异常

指令格式:

31	27	26	25	24	20	19	15	14	12	11	7	6	0
00100		imm2		rs2		rs1		101		rd		0001011	

15.5.50 SRW——寄存器移位字存储指令

语法:

srw rd, rs1, rs2, imm2

操作:

$\text{mem}[(\text{rs1}+\text{rs2}\ll\text{imm2})+3: (\text{rs1}+\text{rs2}\ll\text{imm2})] \leftarrow \text{rd}[31:0]$

执行权限:

M mode/S mode/U mode

异常:

存储指令非对齐访问异常、存储指令访问错误异常、存储指令页面错误异常、非法指令异常

指令格式:

31	27	26	25	24	20	19	15	14	12	11	7	6	0
01000		imm2		rs2		rs1		101		rd		0001011	

15.5.51 SURB——寄存器低 32 位移位字节存储指令

语法:

surb rd, rs1, rs2, imm2

操作:

$\text{mem}[(rs1+rs2[31:0] \ll imm2)] \leftarrow rd[7:0]$

执行权限:

M mode/S mode/U mode

异常:

存储指令非对齐访问异常、存储指令访问错误异常、存储指令页面错误异常、非法指令异常

说明:

rs2[31:0] 是无符号数, 高位 [63:32] 补 0 进行地址运算

指令格式:

31	27	26	25	24	20	19	15	14	12	11	7	6	0
00010		imm2		rs2		rs1		101		rd		0001011	

15.5.52 SURD——寄存器低 32 位移位双字存储指令

语法:

surd rd, rs1, rs2, imm2

操作:

$\text{mem}[(rs1+rs2[31:0] \ll imm2)+7: (rs1+rs2[31:0] \ll imm2)] \leftarrow rd[63:0]$

执行权限:

M mode/S mode/U mode

异常:

存储指令非对齐访问异常、存储指令访问错误异常、存储指令页面错误异常、非法指令异常

说明:

rs2[31:0] 是无符号数, 高位 [63:32] 补 0 进行地址运算

指令格式:

31	27	26	25	24	20	19	15	14	12	11	7	6	0
01110		imm2		rs2		rs1		101		rd		0001011	

15.5.53 SURH——寄存器低 32 位移位半字存储指令

语法:

surh rd, rs1, rs2, imm2

操作:

$\text{mem}[(\text{rs1}+\text{rs2}[31:0]\ll\text{imm2})+1: (\text{rs1}+\text{rs2}[31:0]\ll\text{imm2})] \leftarrow \text{rd}[15:0]$

执行权限:

M mode/S mode/U mode

异常:

存储指令非对齐访问异常、存储指令访问错误异常、存储指令页面错误异常、非法指令异常

说明:

rs2[31:0] 是无符号数, 高位 [63:32] 补 0 进行地址运算

指令格式:

31	27	26	25	24	20	19	15	14	12	11	7	6	0
00110	imm2			rs2	rs1		101	rd		0001011			

15.5.54 SURW——寄存器低 32 位移位字存储指令

语法:

surw rd, rs1, rs2, imm2

操作:

$\text{mem}[(\text{rs1}+\text{rs2}[31:0]\ll\text{imm2})+3: (\text{rs1}+\text{rs2}[31:0]\ll\text{imm2})] \leftarrow \text{rd}[31:0]$

执行权限:

M mode/S mode/U mode

异常:

存储指令非对齐访问异常、存储指令访问错误异常、存储指令页面错误异常、非法指令异常

说明:

rs2[31:0] 是无符号数, 高位 [63:32] 补 0 进行地址运算

指令格式:

31	27	26	25	24	20	19	15	14	12	11	7	6	0
01010	imm2			rs2	rs1		101	rd		0001011			

15.5.55 SWIA——字存储基地址自增指令

语法:

swia rs2, (rs1), imm5,imm2

操作:

mem[rs1+3:rs1]←rs2[31:0]
rs1←rs1 + sign_extend(imm5 << imm2)

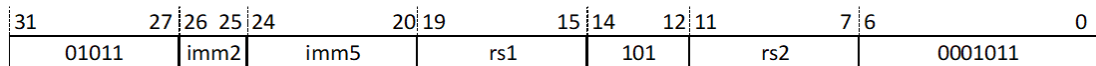
执行权限:

M mode/S mode/U mode

异常:

存储指令非对齐访问异常、存储指令访问错误异常、存储指令页面错误异常、非法指令异常

指令格式:



15.5.56 SWIB——基地址自增字存储指令

语法:

swib rs2, (rs1), imm5,imm2

操作:

rs1←rs1 + sign_extend(imm5 << imm2)
mem[rs1+3:rs1] ←rs2[31:0]

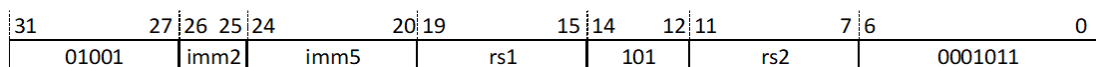
执行权限:

M mode/S mode/U mode

异常:

存储指令非对齐访问异常、存储指令访问错误异常、存储指令页面错误异常、非法指令异常

指令格式:



15.5.57 SWD——双寄存器低 32 位存储指令

语法:

swd rd1,rd2,(rs1),imm2

操作:

address ← rs1 + zero_extend(imm2 << 3)

mem[address+3:address] ← rd1[31:0]

mem[address+7:address+4] ← rd2[31:0]

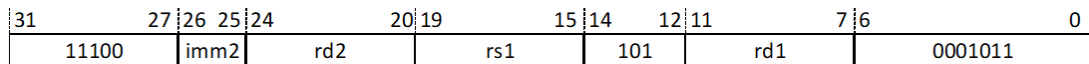
执行权限:

M mode/S mode/U mode

异常:

存储指令非对齐访问异常、存储指令访问错误异常、存储指令页面错误异常、非法指令异常

指令格式:



15.6 附录 B-5 半精度浮点指令术语

半精度浮点指令子集实现了对半精度浮点的支持，每条指令位宽为 32 位，以下指令按英文字母顺序排列。

15.6.1 FADD.H——半精度浮点加法指令

语法:

fadd.h fd, fs1, fs2, rm

操作:

fd ← fs1 + fs2

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

浮点状态位 NV/OF/NX

说明:

rm 决定舍入模式:

- 3' b000: 就近向偶数舍入, 对应的汇编指令 fadd.h fd, fs1,fs2,rne。
- 3' b001: 向零舍入, 对应的汇编指令 fadd.h fd, fs1,fs2,rtz。
- 3' b010: 向负无穷舍入, 对应的汇编指令 fadd.h fd, fs1,fs2,rdn。
- 3' b011: 向正无穷舍入, 对应的汇编指令 fadd.h fd, fs1,fs2,rup。
- 3' b100: 就近向大值舍入, 对应的汇编指令 fadd.h fd, fs1,fs2,rmm。
- 3' b101: 暂未使用, 不会出现该编码。
- 3' b110: 暂未使用, 不会出现该编码。
- 3' b111: 动态舍入, 根据浮点控制寄存器 fcsr 中的 rm 位来决定舍入模式, 对应的汇编指令 fadd.h fd, fs1,fs2。

指令格式:

31	25 24	20 19	15 14	12 11	7 6	0
0000010	fs2	fs1	rm	fd	1010011	

15.6.2 FCLASS.H——半精度浮点分类指令

语法:

fclass.h rd, fs1

操作:

```

if ( fs1 = -inf)
    rd ← 64' h1
if ( fs1 = -norm)
    rd ← 64' h2
if ( fs1 = -subnorm)
    rd ← 64' h4
if ( fs1 = -zero)
    rd ← 64' h8
if ( fs1 = +zero)
    rd ← 64' h10
if ( fs1 = +subnorm)
    rd ← 64' h20
if ( fs1 = +norm)
    rd ← 64' h40
if ( fs1 = +inf)
    rd ← 64' h80
if ( fs1 = sNaN)
    
```

```
rd ← 64' h100
if ( fs1 = qNaN)
    rd ← 64' h200
```

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

无

指令格式:

31	25 24	20 19	15 14	12 11	7 6	0
1110010	00000	fs1	001	rd	1010011	

15.6.3 FCVT.D.H——半精度浮点转换成双精度浮点指令

语法:

```
fcvt.d.h fd, fs1
```

操作:

```
fd ← half_convert_to_double(fs1)
```

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

无

指令格式:

31	25 24	20 19	15 14	12 11	7 6	0
0100001	00010	fs1	000	fd	1010011	

15.6.4 FCVT.H.D——双精度浮点转换成半精度浮点指令

语法:

`fcvt.h.d fd, fs1, rm`

操作:

`fd ← double_convert_to_half(fs1)`

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

浮点状态位 NV/OF/UF/NX

说明:

rm 决定舍入模式:

- 3' b000: 就近向偶数舍入, 对应的汇编指令 `fcvt.h.d fd,fs1,rne`。
- 3' b001: 向零舍入, 对应的汇编指令 `fcvt.h.d fd,fs1,rtz`。
- 3' b010: 向负无穷舍入, 对应的汇编指令 `fcvt.h.d fd,fs1,rdn`。
- 3' b011: 向正无穷舍入, 对应的汇编指令 `fcvt.h.d fd,fs1,rup`。
- 3' b100: 就近向大值舍入, 对应的汇编指令 `fcvt.h.d fd,fs1,rmm`。
- 3' b101: 暂未使用, 不会出现该编码。
- 3' b110: 暂未使用, 不会出现该编码。
- 3' b111: 动态舍入, 根据浮点控制寄存器 `fcsr` 中的 `rm` 位来决定舍入模式, 对应的汇编指令 `fcvt.h.d fd, fs1`。

指令格式:

31	25 24	20 19	15 14	12 11	7 6	0
0100010	00001	fs1	rm	fd	1010011	

15.6.5 FCVT.H.L——有符号长整型转换成半精度浮点数指令

语法:

`fcvt.h.l fd, rs1, rm`

操作:

`fd ← signed_long_convert_to_half(rs1)`

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

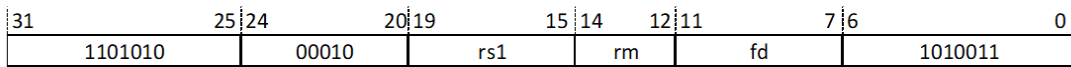
浮点状态位 NX/OF

说明:

rm 决定舍入模式:

- 3' b000: 就近向偶数舍入, 对应的汇编指令 fcvt.h.l fd,rs1,rne。
- 3' b001: 向零舍入, 对应的汇编指令 fcvt.h.l fd,rs1,rtz。
- 3' b010: 向负无穷舍入, 对应的汇编指令 fcvt.h.l fd,rs1,fdn。
- 3' b011: 向正无穷舍入, 对应的汇编指令 fcvt.h.l fd,rs1,rup。
- 3' b100: 就近向大值舍入, 对应的汇编指令 fcvt.h.l fd,rs1,rmm。
- 3' b101: 暂未使用, 不会出现该编码。
- 3' b110: 暂未使用, 不会出现该编码。
- 3' b111: 动态舍入, 根据浮点控制寄存器 fcsr 中的 rm 位来决定舍入模式, 对应的汇编指令 fcvt.h.l fd, rs1。

指令格式:



15.6.6 FCVT.H.LU——无符号长整型转换成半精度浮点数指令

语法:

fcvt.h.lu fd, rs1, rm

操作:

fd ← unsigned_long_convert_to_half(rs1)

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

浮点状态位 NX/OF

说明:

rm 决定舍入模式:

- 3' b000: 就近向偶数舍入, 对应的汇编指令 fcvt.h.lu fd,rs1,rne。
- 3' b001: 向零舍入, 对应的汇编指令 fcvt.h.lu fd, rs1,rtz。
- 3' b010: 向负无穷舍入, 对应的汇编指令 fcvt.h.lu fd, rs1,fdn。
- 3' b011: 向正无穷舍入, 对应的汇编指令 fcvt.h.lu fd, rs1,rup。
- 3' b100: 就近向大值舍入, 对应的汇编指令 fcvt.h.lu fd, rs1,rmm。
- 3' b101: 暂未使用, 不会出现该编码。
- 3' b110: 暂未使用, 不会出现该编码。
- 3' b111: 动态舍入, 根据浮点控制寄存器 fcsr 中的 rm 位来决定舍入模式, 对应的汇编指令 fcvt.h.lu fd, rs1。

指令格式:

31	25 24	20 19	15 14	12 11	7 6	0
1101010	00011	rs1	rm	fd	1010011	

15.6.7 FCVT.H.S——单精度浮点转换成半精度浮点指令

语法:

fcvt.h.s fd, fs1, rm

操作:

fd ← single_convert_to_half(fs1)

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

浮点状态位 NV/OF/UF/NX

说明:

rm 决定舍入模式:

- 3' b000: 就近向偶数舍入, 对应的汇编指令 fcvt.h.s fd,fs1,rne。
- 3' b001: 向零舍入, 对应的汇编指令 fcvt.h.s fd,fs1,rtz。
- 3' b010: 向负无穷舍入, 对应的汇编指令 fcvt.h.s fd,fs1,fdn。

- 3' b011: 向正无穷舍入, 对应的汇编指令 fcvt.h.s fd,fs1,rup。
- 3' b100: 就近向大值舍入, 对应的汇编指令 fcvt.h.s fd,fs1,rmm。
- 3' b101: 暂未使用, 不会出现该编码。
- 3' b110: 暂未使用, 不会出现该编码。
- 3' b111: 动态舍入, 根据浮点控制寄存器 fcsr 中的 rm 位来决定舍入模式, 对应的汇编指令 fcvt.h.s fd, fs1。

指令格式:

31	25 24	20 19	15 14	12 11	7 6	0
0100010	00000	fs1	rm	fd	1010011	

15.6.8 FCVT.H.W——有符号整型转换成半精度浮点数指令

语法:

fcvt.h.w fd, rs1, rm

操作:

fd ← signed_int_convert_to_half(rs1)

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

浮点状态位 NX/OF

说明:

rm 决定舍入模式:

- 3' b000: 就近向偶数舍入, 对应的汇编指令 fcvt.h.w fd,rs1,rne。
- 3' b001: 向零舍入, 对应的汇编指令 fcvt.h.w fd,rs1,rtz。
- 3' b010: 向负无穷舍入, 对应的汇编指令 fcvt.h.w fd,rs1,fdn。
- 3' b011: 向正无穷舍入, 对应的汇编指令 fcvt.h.w fd,rs1,rup。
- 3' b100: 就近向大值舍入, 对应的汇编指令 fcvt.h.w fd,rs1,rmm。
- 3' b101: 暂未使用, 不会出现该编码。
- 3' b110: 暂未使用, 不会出现该编码。
- 3' b111: 动态舍入, 根据浮点控制寄存器 fcsr 中的 rm 位来决定舍入模式, 对应的汇编指令 fcvt.h.w fd, rs1。

指令格式:

31	25 24	20 19	15 14	12 11	7 6	0
1101010	00000	rs1	rm	fd	1010011	

15.6.9 FCVT.H.WU——无符号整型转换成半精度浮点数指令

语法:

fcvt.h.wu fd, rs1, rm

操作:

fd ← unsigned_int_convert_to_half_fp(rs1)

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

浮点状态位 NX/OF

说明:

rm 决定舍入模式:

- 3' b000: 就近向偶数舍入, 对应的汇编指令 fcvt.h.wu fd,rs1,rne。
- 3' b001: 向零舍入, 对应的汇编指令 fcvt.h.wu fd,rs1,rtz。
- 3' b010: 向负无穷舍入, 对应的汇编指令 fcvt.h.wu fd,rs1,fdn。
- 3' b011: 向正无穷舍入, 对应的汇编指令 fcvt.h.wu fd,rs1,rup。
- 3' b100: 就近向大值舍入, 对应的汇编指令 fcvt.h.wu fd,rs1,rmm。
- 3' b101: 暂未使用, 不会出现该编码。
- 3' b110: 暂未使用, 不会出现该编码。
- 3' b111: 动态舍入, 根据浮点控制寄存器 fcsr 中的 rm 位来决定舍入模式, 对应的汇编指令 fcvt.h.wu fd, rs1。

指令格式:

31	25 24	20 19	15 14	12 11	7 6	0
1101010	00001	rs1	rm	fd	1010011	

15.6.10 FCVT.L.H——半精度浮点转换成有符号长整型指令

语法:

`fcvt.l.h rd, fs1, rm`

操作:

$rd \leftarrow \text{half_convert_to_signed_long}(fs1)$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

浮点状态位 NV/NX

说明:

rm 决定舍入模式:

- 3' b000: 就近向偶数舍入, 对应的汇编指令 `fcvt.l.h rd,fs1,rne`。
- 3' b001: 向零舍入, 对应的汇编指令 `fcvt.l.h rd,fs1,rtz`。
- 3' b010: 向负无穷舍入, 对应的汇编指令 `fcvt.l.h rd,fs1,rdn`。
- 3' b011: 向正无穷舍入, 对应的汇编指令 `fcvt.l.h rd,fs1,rup`。
- 3' b100: 就近向大值舍入, 对应的汇编指令 `fcvt.l.h rd,fs1,rmm`。
- 3' b101: 暂未使用, 不会出现该编码。
- 3' b110: 暂未使用, 不会出现该编码。
- 3' b111: 动态舍入, 根据浮点控制寄存器 `fcsr` 中的 `rm` 位来决定舍入模式, 对应的汇编指令 `fcvt.l.h rd, fs1`。

指令格式:

31	25 24	20 19	15 14	12 11	7 6	0
1100010	00010	fs1	rm	rd	1010011	

15.6.11 FCVT.LU.H——半精度浮点转换成无符号长整型指令

语法:

`fcvt.lu.h rd, fs1, rm`

操作:

$rd \leftarrow \text{half_convert_to_unsigned_long}(fs1)$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

浮点状态位 NV/NX

说明:

rm 决定舍入模式:

- 3' b000: 就近向偶数舍入, 对应的汇编指令 fcvt.lu.h rd,fs1,rne。
- 3' b001: 向零舍入, 对应的汇编指令 fcvt.lu.h rd,fs1,rtz。
- 3' b010: 向负无穷舍入, 对应的汇编指令 fcvt.lu.h rd,fs1,rdn。
- 3' b011: 向正无穷舍入, 对应的汇编指令 fcvt.lu.h rd,fs1,rup。
- 3' b100: 就近向大值舍入, 对应的汇编指令 fcvt.lu.h rd,fs1,rmm。
- 3' b101: 暂未使用, 不会出现该编码。
- 3' b110: 暂未使用, 不会出现该编码。
- 3' b111: 动态舍入, 根据浮点控制寄存器 fcsr 中的 rm 位来决定舍入模式, 对应的汇编指令 fcvt.lu.h rd, fs1。

指令格式:

31	25 24	20 19	15 14	12 11	7 6	0
	1100010	00011	fs1	rm	rd	1010011

15.6.12 FCVT.S.H——半精度浮点转换成单精度浮点指令

语法:

fcvt.s.h fd, fs1

操作:

fd ← half_convert_to_single(fs1)

执行权限:

M mode/S mode/U mode

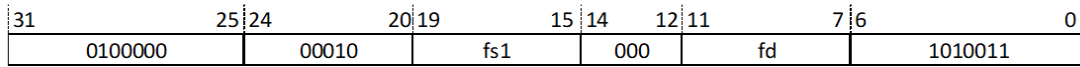
异常:

非法指令异常

影响标志位:

无

指令格式:



15.6.13 FCVT.W.H——半精度浮点转换成有符号整型指令

语法:

fcvt.w.h rd, fs1, rm

操作:

tmp[31:0] ← half_convert_to_signed_int(fs1)
rd ← sign_extend(tmp[31:0])

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

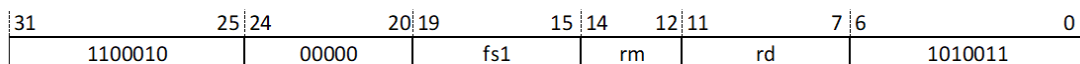
浮点状态位 NV/NX

说明:

rm 决定舍入模式:

- 3' b000: 就近向偶数舍入, 对应的汇编指令 fcvt.w.h rd,fs1,rne。
- 3' b001: 向零舍入, 对应的汇编指令 fcvt.w.h rd,fs1,rtz。
- 3' b010: 向负无穷舍入, 对应的汇编指令 fcvt.w.h rd,fs1,rdn。
- 3' b011: 向正无穷舍入, 对应的汇编指令 fcvt.w.h rd,fs1,rup。
- 3' b100: 就近向大值舍入, 对应的汇编指令 fcvt.w.h rd,fs1,rmm。
- 3' b101: 暂未使用, 不会出现该编码。
- 3' b110: 暂未使用, 不会出现该编码。
- 3' b111: 动态舍入, 根据浮点控制寄存器 fcsr 中的 rm 位来决定舍入模式, 对应的汇编指令 fcvt.w.h rd, fs1。

指令格式:



15.6.14 FCVT.WU.H——半精度浮点转换成无符号整型指令

语法:

fcvt.wu.h rd, fs1, rm

操作:

tmp[31:0] ← half_convert_to_unsigned_int(fs1)
rd ← sign_extend(tmp[31:0])

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

浮点状态位 NV/NX

说明:

rm 决定舍入模式:

- 3' b000: 就近向偶数舍入, 对应的汇编指令 fcvt.wu.h rd,fs1,rne。
- 3' b001: 向零舍入, 对应的汇编指令 fcvt.wu.h rd,fs1,rtz。
- 3' b010: 向负无穷舍入, 对应的汇编指令 fcvt.wu.h rd,fs1,rdn。
- 3' b011: 向正无穷舍入, 对应的汇编指令 fcvt.wu.h rd,fs1,rup。
- 3' b100: 就近向大值舍入, 对应的汇编指令 fcvt.wu.h rd,fs1,rmm。
- 3' b101: 暂未使用, 不会出现该编码。
- 3' b110: 暂未使用, 不会出现该编码。
- 3' b111: 动态舍入, 根据浮点控制寄存器 fcsr 中的 rm 位来决定舍入模式, 对应的汇编指令 fcvt.wu.h rd, fs1。

指令格式:

31	25 24	20 19	15 14	12 11	7 6	0
1100010	00001	fs1	rm	rd	1010011	

15.6.15 FDIV.H——半精度浮点除法指令

语法:

fdiv.h fd, fs1, fs2, rm

操作:

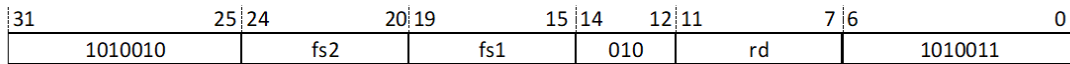
异常:

非法指令异常

影响标志位:

浮点状态位 NV

指令格式:



15.6.17 FLE.H——半精度浮点比较小于等于指令

语法:

fle.h rd, fs1, fs2

操作:

```
if(fs1 <= fs2)
    rd ← 1
else
    rd ← 0
```

执行权限:

M mode/S mode/U mode

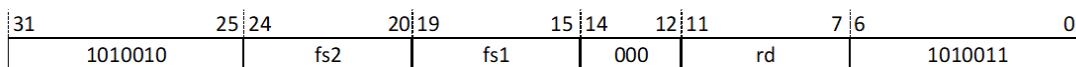
异常:

非法指令异常

影响标志位:

浮点状态位 NV

指令格式:



15.6.18 FLH——半精度浮点加载指令

语法:

flh fd, imm12(rs1)

操作:

address ← rs1 + sign_extend(imm12)
 fd[15:0] ← mem[(address+1):address]
 fd[63:16] ← 48' hfffffff

执行权限:

M mode/S mode/U mode

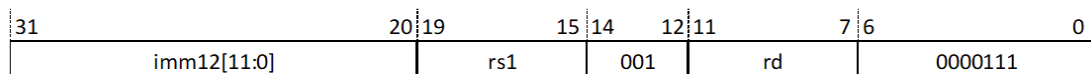
异常:

加载指令非对齐访问异常、加载指令访问错误异常、加载指令页面错误异常、非法指令异常

影响标志位:

无

指令格式:



15.6.19 FLT.H——半精度浮点比较小于指令

语法:

flt.h rd, fs1, fs2

操作:

if(fs1 < fs2)
 rd ← 1
 else
 rd ← 0

执行权限:

M mode/S mode/U mode

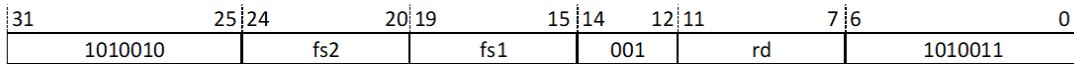
异常:

非法指令异常

影响标志位:

浮点状态位 NV

指令格式:



15.6.20 FMADD.H——半精度浮点乘累加指令

语法:

fmadd.h fd, fs1, fs2, fs3, rm

操作:

$fd \leftarrow fs1 * fs2 + fs3$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

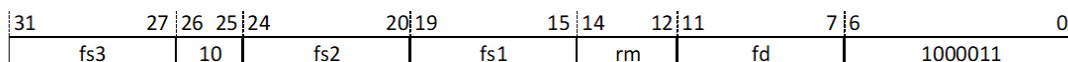
浮点状态位 NV/OF/UF/IX

说明:

rm 决定舍入模式:

- 3' b000: 就近向偶数舍入, 对应的汇编指令 fmadd.h fd,fs1, fs2, fs3, rne。
- 3' b001: 向零舍入, 对应的汇编指令 fmadd.h fd,fs1, fs2, fs3, rtz。
- 3' b010: 向负无穷舍入, 对应的汇编指令 fmadd.h fd,fs1, fs2, fs3, rdn。
- 3' b011: 向正无穷舍入, 对应的汇编指令 fmadd.h fd,fs1, fs2, fs3, rup。
- 3' b100: 就近向大值舍入, 对应的汇编指令 fmadd.h fd,fs1, fs2, fs3, rmm。
- 3' b101: 暂未使用, 不会出现该编码。
- 3' b110: 暂未使用, 不会出现该编码。
- 3' b111: 动态舍入, 根据浮点控制寄存器 fcsr 中的 rm 位来决定舍入模式, 对应的汇编指令 fmadd.h fd,fs1, fs2, fs3。

指令格式:



15.6.21 FMAX.H——半精度浮点取最大值指令

语法:

fmax.h fd, fs1, fs2

操作:

```
if(fs1 >= fs2)
    fd ← fs1
else
    fd ← fs2
```

执行权限:

M mode/S mode/U mode

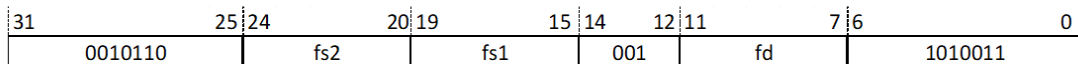
异常:

非法指令异常

影响标志位:

浮点状态位 NV

指令格式:



15.6.22 FMIN.H——半精度浮点取最小值指令

语法:

fmin.h fd, fs1, fs2

操作:

```
if(fs1 < fs2)
    fd ← fs1
else
    fd ← fs2
```

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

浮点状态位 NV

指令格式:

31	25 24	20 19	15 14	12 11	7 6	0
0010110	fs2	fs1	000	fd	1010011	

15.6.23 FMSUB.H——半精度浮点乘累减指令

语法:

fmsub.h fd, fs1, fs2, fs3, rm

操作:

$fd \leftarrow fs1 * fs2 - fs3$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

浮点状态位 NV/OF/UF/NX

说明:

rm 决定舍入模式:

- 3' b000: 就近向偶数舍入, 对应的汇编指令 fmsub.h fd,fs1, fs2, fs3, rne。
- 3' b001: 向零舍入, 对应的汇编指令 fmsub.h fd,fs1, fs2, fs3, rtz。
- 3' b010: 向负无穷舍入, 对应的汇编指令 fmsub.h fd,fs1, fs2, fs3, rdn。
- 3' b011: 向正无穷舍入, 对应的汇编指令 fmsub.h fd,fs1, fs2, fs3, rup。
- 3' b100: 就近向大值舍入, 对应的汇编指令 fmsub.h fd, fs1, fs2, fs3,rmm。
- 3' b101: 暂未使用, 不会出现该编码。
- 3' b110: 暂未使用, 不会出现该编码。
- 3' b111: 动态舍入, 根据浮点控制寄存器 fcsr 中的 rm 位来决定舍入模式, 对应的汇编指令 fmsub.h fd,fs1, fs2, fs3。

指令格式:

31	27 26	25 24	20 19	15 14	12 11	7 6	0
fs3	10	fs2	fs1	rm	fd	1000111	

15.6.24 FMUL.H——半精度浮点乘法指令

语法:

`fmul.h fd, fs1, fs2, rm`

操作:

$fd \leftarrow fs1 * fs2$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

浮点状态位 NV/OF/UF/NX

说明:

rm 决定舍入模式:

- 3' b000: 就近向偶数舍入, 对应的汇编指令 `fmul.h fd, fs1, fs2, rne`。
- 3' b001: 向零舍入, 对应的汇编指令 `fmul.h fd, fs1, fs2, rtz`。
- 3' b010: 向负无穷舍入, 对应的汇编指令 `fmul.h fd, fs1, fs2, rdn`。
- 3' b011: 向正无穷舍入, 对应的汇编指令 `fmul.h fd, fs1, fs2, rup`。
- 3' b100: 就近向大值舍入, 对应的汇编指令 `fmul.h fd, fs1, fs2, rmm`。
- 3' b101: 暂未使用, 不会出现该编码。
- 3' b110: 暂未使用, 不会出现该编码。
- 3' b111: 动态舍入, 根据浮点控制寄存器 `fcsr` 中的 `rm` 位来决定舍入模式, 对应的汇编指令 `fmul.h fs1, fs2`。

指令格式:

31	25 24	20 19	15 14	12 11	7 6	0
0001010	fs2	fs1	rm	fd	1010011	

15.6.25 FMV.H.X——半精度浮点写传送指令

语法:

`fmv.h.x fd, rs1`

操作:

fd[15:0] ← rs1[15:0]
 fd[63:16] ← 48' hfffffffffff

执行权限:

M mode/S mode/U mode

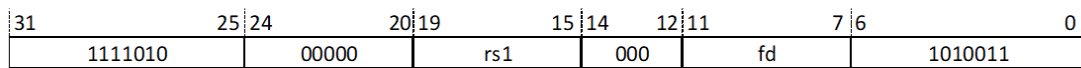
异常:

非法指令异常

影响标志位:

无

指令格式:



15.6.26 FMV.X.H——半精度浮点寄存器读传送指令

语法:

fmv.x.h rd, fs1

操作:

tmp[15:0] ← fs1[15:0]
 rd ← sign_extend(tmp[15:0])

执行权限:

M mode/S mode/U mode

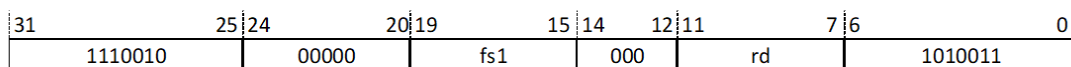
异常:

非法指令异常

影响标志位:

无

指令格式:



15.6.27 FNMADD.H——半精度浮点乘累加取负指令

语法:

fnmadd.h fd, fs1, fs2, fs3, rm

操作:

$fd \leftarrow -(fs1 * fs2 + fs3)$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

浮点状态位 NV/OF/UF/IX

说明:

rm 决定舍入模式:

- 3' b000: 就近向偶数舍入, 对应的汇编指令 fnmadd.h fd,fs1, fs2, fs3, rne。
- 3' b001: 向零舍入, 对应的汇编指令 fnmadd.h fd,fs1, fs2, fs3, rtz。
- 3' b010: 向负无穷舍入, 对应的汇编指令 fnmadd.h fd,fs1, fs2, fs3, rdn。
- 3' b011: 向正无穷舍入, 对应的汇编指令 fnmadd.h fd,fs1, fs2, fs3, rup。
- 3' b100: 就近向大值舍入, 对应的汇编指令 fnmadd.h fd,fs1, fs2, fs3, rmm。
- 3' b101: 暂未使用, 不会出现该编码。
- 3' b110: 暂未使用, 不会出现该编码。
- 3' b111: 动态舍入, 根据浮点控制寄存器 fcsr 中的 rm 位来决定舍入模式, 对应的汇编指令 fnmadd.h fd,fs1, fs2, fs3。

指令格式:

31	27	26	25	24	20	19	15	14	12	11	7	6	0
fs3			10		fs2		fs1		rm		fd		1001111

15.6.28 FNMSUB.H——半精度浮点乘累减取负指令

语法:

fnmsub.h fd, fs1, fs2, fs3, rm

操作:

$fd \leftarrow -(fs1 * fs2 - fs3)$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

浮点状态位 NV/OF/UF/IX

说明:

rm 决定舍入模式:

- 3' b000: 就近向偶数舍入, 对应的汇编指令 fnmsub.h fd,fs1, fs2, fs3, rne。
- 3' b001: 向零舍入, 对应的汇编指令 fnmsub.h fd,fs1, fs2, fs3, rtz。
- 3' b010: 向负无穷舍入, 对应的汇编指令 fnmsub.h fd,fs1, fs2, fs3, rdn。
- 3' b011: 向正无穷舍入, 对应的汇编指令 fnmsub.h fd,fs1, fs2, fs3, rup。
- 3' b100: 就近向大值舍入, 对应的汇编指令 fnmsub.h fd,fs1, fs2, fs3, rmm。
- 3' b101: 暂未使用, 不会出现该编码。
- 3' b110: 暂未使用, 不会出现该编码。
- 3' b111: 动态舍入, 根据浮点控制寄存器 fcsr 中的 rm 位来决定舍入模式, 对应的汇编指令 fnmsub.h fd,fs1, fs2, fs3。

指令格式:

31	27	26	25	24	20	19	15	14	12	11	7	6	0
fs3			10		fs2		fs1		rm		fd		1001011

15.6.29 FSGNJ.H——半精度浮点符号注入指令

语法:

fsgnj.h fd, fs1, fs2

操作:

fd[14:0] ← fs1[14:0]
 fd[15] ← fs2[15]
 fd[63:16] ← 48' hfffffffffff

执行权限:

M mode/S mode/U mode

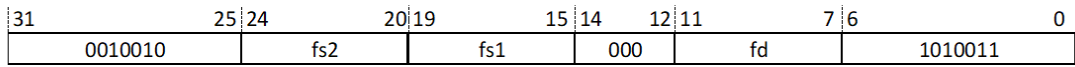
异常:

非法指令异常

影响标志位:

无

指令格式:



15.6.30 FSGNJN.H——半精度浮点符号取反注入指令

语法:

fsgnfn.h fd, fs1, fs2

操作:

fd[14:0] ← fs1[14:0]

fd[15] ← ! fs2[15]

fd[63:16] ← 48' hfffffffffff

执行权限:

M mode/S mode/U mode

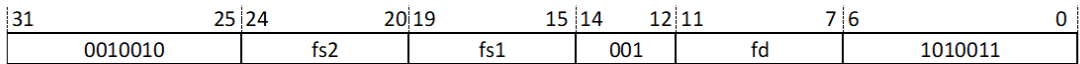
异常:

非法指令异常

影响标志位:

无

指令格式:



15.6.31 FSGNJX.H——半精度浮点符号异或注入指令

语法:

fsgnjx.h fd, fs1, fs2

操作:

fd[14:0] ← fs1[14:0]

fd[15] ← fs1[15] ^ fs2[15]

fd[63:16] ← 48' hfffffffffff

执行权限:

M mode/S mode/U mode

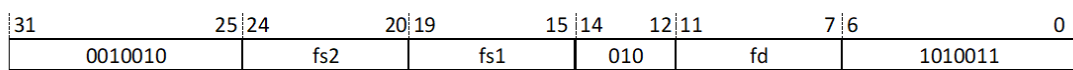
异常:

非法指令异常

影响标志位:

无

指令格式:



15.6.32 FSH——半精度浮点存储指令

语法:

fsh fs2, imm12(fs1)

操作:

address ← fs1 + sign_extend(imm12)
 mem[(address+1):address] ← fs2[15:0]

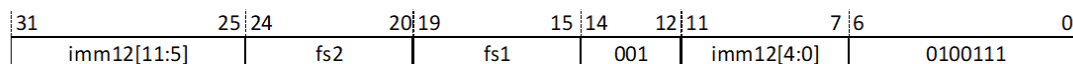
执行权限:

M mode/S mode/U mode

异常:

存储指令非对齐访问异常、存储指令访问错误异常、存储指令页面错误异常、非法指令异常

指令格式:



15.6.33 FSQRT.H——半精度浮点开方指令

语法:

fsqrt.h fd, fs1, rm

操作:

fd ← sqrt(fs1)

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

浮点状态位 NV/NX

说明:

rm 决定舍入模式:

- 3' b000: 就近向偶数舍入, 对应的汇编指令 fsqrt.h fd, fs1,rne
- 3' b001: 向零舍入, 对应的汇编指令 fsqrt.h fd, fs1,rtz
- 3' b010: 向负无穷舍入, 对应的汇编指令 fsqrt.h fd, fs1,rdn
- 3' b011: 向正无穷舍入, 对应的汇编指令 fsqrt.h fd, fs1,rup
- 3' b100: 就近向大值舍入, 对应的汇编指令 fsqrt.h fd, fs1,rmm
- 3' b101: 暂未使用, 不会出现该编码。
- 3' b110: 暂未使用, 不会出现该编码。
- 3' b111: 动态舍入, 根据浮点控制寄存器 fcsr 中的 rm 位来决定舍入模式, 对应的汇编指令 fsqrt.h fd, fs1。

指令格式:

31	25 24	20 19	15 14	12 11	7 6	0
0101110	00000	fs1	rm	fd	1010011	

15.6.34 FSUB.H——半精度浮点减法指令

语法:

fsub.h fd, fs1, fs2, rm

操作:

$fd \leftarrow fs1 - fs2$

执行权限:

M mode/S mode/U mode

异常:

非法指令异常

影响标志位:

浮点状态位 NV/OF/NX

说明:

rm 决定舍入模式:

- 3' b000: 就近向偶数舍入, 对应的汇编指令 fsub.h fd, fs1,fs2,rne
- 3' b001: 向零舍入, 对应的汇编指令 fsub.h fd, fs1,fs2,rtz
- 3' b010: 向负无穷舍入, 对应的汇编指令 fsub.h fd, fs1,fs2,rdn
- 3' b011: 向正无穷舍入, 对应的汇编指令 fsub.h fd, fs1,fs2,rup
- 3' b100: 就近向大值舍入, 对应的汇编指令 fsub.h fd, fs1,fs2,rmm
- 3' b101: 暂未使用, 不会出现该编码。
- 3' b110: 暂未使用, 不会出现该编码。
- 3' b111: 动态舍入, 根据浮点控制寄存器 fcsr 中的 rm 位来决定舍入模式, 对应的汇编指令 fsub.h fd, fs1,fs2。

指令格式:

31	25 24	20 19	15 14	12 11	7 6	0
	0000110	fs2	fs1	rm	fd	1010011

第十六章 附录 C 控制寄存器

本附录对机器模式控制寄存器、超级用户模式控制寄存器和用户模式控制寄存器进行详细说明。

16.1 附录 C-1 机器模式控制寄存器

机器模式控制寄存器按照功能分为：机器模式信息寄存器组、机器模式异常配置寄存器组、机器模式异常处理寄存器组、机器模式内存保护寄存器组、机器模式计数器寄存器组、机器模式计数器配置寄存器组、机器模式处理器控制和状态扩展寄存器组、机器模式 Cache 访问扩展寄存器组和机器模式处理器型号寄存器组。

16.1.1 机器模式信息寄存器组

16.1.1.1 机器模式供应商编号寄存器 (MVENDORID)

机器模式供应商编号寄存器 (MVENDORID) 存储了 JEDEC 分配给平头哥半导体有限公司的供应商编号信息，该寄存器值固定为 64' h5B7。

该寄存器的位长是 64 位，寄存器的读写权限是机器模式只读，即非机器模式访问和机器模式写操作访问都会导致非法指令异常。

16.1.1.2 机器模式架构编号寄存器 (MARCHID)

机器模式架构编号寄存器 (MARCHID) 存储了处理器核的架构编号，用于存放平头哥半导体有限公司产品的内部编号，C906 目前未定义该寄存器，值固定为 64' h0。

该寄存器的位长是 64 位，寄存器的读写权限是机器模式只读，即非机器模式访问和机器模式写操作访问都会导致非法指令异常。

16.1.1.3 机器模式硬件实现编号寄存器 (MIMPID)

机器模式硬件实现编号寄存器 (MIMPID) 存储了处理器核的硬件实现编号。C906 目前未定义该寄存器，值固定为 64' h0。

该寄存器的位长是 64 位，寄存器的读写权限是机器模式只读，即非机器模式访问和机器模式写操作访问都会导致非法指令异常。

16.1.1.4 机器模式逻辑内核编号寄存器 (MHARTID)

机器模式逻辑内核编号寄存器 (MHARTID) 存储了处理器核的硬件逻辑内核编号, 具体是 MARATID 的最低三位指示了多核处理器的核心编号。C906 中目前该寄存器值固定为 64'h0。

该寄存器的位长是 64 位, 寄存器的读写权限是机器模式只读, 即非机器模式访问和机器模式写操作访问都会导致非法指令异常。

16.1.2 机器模式异常配置寄存器组

16.1.2.1 机器模式处理器状态寄存器 (MSTATUS)

机器模式处理器状态寄存器 (MSTATUS) 存储了处理器在机器模式下的状态和控制信息, 包括全局中断有效位、异常保留中断有效位、异常保留特权模式位等。

该寄存器的位长是 64 位, 寄存器的读写权限是机器模式可读写, 即非机器模式访问都会导致非法指令异常。

	63	62																															36	35	34	33	32
SD	0																														SXL	UXL					
Reset	0																														2	2					

	31				25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	0			VS	TSR	TW							XS	FS	MPP	0	SPP	0	0	MIE	0	SIE	0							
						TVM			SUM								MPIE		SPIE											
						MXR			MPRV																					
Reset	0			0	0	0	0	0	0	0	0	0	0	0	0	3	0	1	0	0	0	0	0	0	0	0	0	0	0	0

图 16.1: 机器模式处理器状态寄存器 (MSTATUS)

SIE-全局超级用户模式中断使能位:

- 当 SIE 为 0 时, 超级用户中断无效;
- 当 SIE 为 1 时, 超级用户中断有效;

该位会被 reset 清零, 处理器被降级到超级用户模式响应中断时被清零; 在处理器退出中断服务程序时被置为 SPIE 的值。

MIE-全局机器模式中断使能位:

- 当 MIE 为 0 时, 机器模式中断无效;
- 当 MIE 为 1 时, 机器模式中断有效;

该位会被 reset 清零, 处理器在机器模式响应中断时被清零; 在处理器退出中断服务程序时被置为 MPIE 的值。

SPIE-超级用户模式保留中断使能位:

该位用于保存处理器在降级到超级用户模式响应中断前 SIE 位的值。

该位会被 reset 清零，在处理器退出中断服务程序时被置 1。

MPIE-机器模式保留中断使能位:

该位用于保存处理器在机器模式响应中断前 MIE 位的值。

该位会被 reset 清零，在处理器退出中断异常服务程序时被置 1。

SPP-超级用户模式保留特权状态位:

该位用于保存处理器在降级到超级用户模式进入异常服务程序前的特权状态。

- 当 SPP 为 2' b00 时，表示处理器进入异常服务程序前处于用户模式；
- 当 SPP 为 2' b01 时，表示处理器进入异常服务程序前处于超级用户模式；

该位会被 reset 置 2' b01。

MPP-机器模式保留特权状态位:

该位用于保存处理器在机器模式进入异常服务程序前的特权状态。

- 当 MPP 为 2' b00 时，表示处理器进入异常服务程序前处于用户模式；
- 当 MPP 为 2' b01 时，表示处理器进入异常服务程序前处于超级用户模式；
- 当 MPP 为 2' b11 时，表示处理器进入异常服务程序前处于机器模式；

该位会被 reset 置 2' b11。

FS-浮点单元状态位:

根据浮点状态位，可以判断上下文切换的时候，是否需要保存浮点相关寄存器。

- 当 FS 为 2' b00 时，浮点单元处于关闭状态，此时访问浮点相关寄存器会触发非法指令异常。
- 当 FS 为 2' b01 时，浮点单元处于初始化状态。
- 当 FS 为 2' b10 时，浮点单元处于 clean 态。
- 当 FS 为 2' b11 时，浮点单元处于 dirty 态，表明浮点寄存器和控制寄存器被修改过。

该位会被 reset 置为 2' b00。

XS-扩展单元状态位:

C906 没有扩展单元，固定为 0。

MPRV-修改特权模式:

- 当 MPRV=1 时，加载和存储请求执行时根据 MPP 中的特权态进行执行。
- 当 MPRV=0 时，加载和存储请求执行时根据当前处理器所处特权模式进行执行。

该位会被 reset 置为 1' b0。

SUM-允许超级用户模式下访问 U 态虚拟内存空间

- 当 SUM=1 时，超级用户模式下，加载、存储和取指令请求可以访问标记为用户态的虚拟内存空间。

- 当 SUM=0 时，超级用户模式下，加载、存储和取指令请求不可以访问标记为用户态的虚拟内存空间。该位会被 reset 置为 1' b0。

MXR-允许加载请求访问标记为可执行的内存空间

- 当 MXR=1 时，允许加载请求访问标记为可执行和可读的虚拟内存空间。
 - 当 MXR=0 时，允许加载请求只能访问标记为可读的虚拟内存空间。
- 该位会被 reset 置为 1' b0。

TVM-陷阱虚拟内存

- 当 TVM=1 时，超级用户模式读写 STAP 控制寄存器以及执行 SFENCE 指令，触发非法指令异常。
 - 当 TVM=0 时，超级用户模式可以读写 STAP 控制寄存器以及执行 SFENCE 指令。
- 该位会被 reset 置为 1' b0。

TW-超时等待

- 当 TW=1 时，超级用户模式执行低功耗指令 WFI，触发非法指令异常。
- 当 TW=0 时，超级用户模式执行低功耗指令 WFI。

TSR-陷阱 SRET

- 当 TSR=1 时，超级用户模式执行 SRET 指令，产生非法指令异常。
 - 当 TSR=0 时，允许超级用户模式执行 SRET 指令。
- 该位会被 reset 置为 1' b0。

VS-矢量单元状态位

根据矢量状态位，可以判断上下文切换的时候，是否需要保存矢量相关寄存器。

- 当 VS 为 2' b00 时，矢量单元处于关闭状态，此时访问矢量相关寄存器会触发非法指令异常。
- 当 VS 为 2' b01 时，矢量单元处于初始化状态。
- 当 VS 为 2' b10 时，矢量单元处于 clean 态。
- 当 VS 为 2' b11 时，矢量单元处于 dirty 态，表明矢量寄存器和矢量控制寄存器被修改过。

VS 位仅当配置矢量执行单元时有效，不配置时恒为 0，该位会被 reset 置为 2' b00。

UXL-U 态寄存器位宽

只读，固定值是 2，表示在 U 态下，寄存器的位宽是 64bit。

SXL-S 态寄存器位宽

只读，固定值是 2，表示在 S 态下，寄存器的位宽是 64bit。

SD-浮点、矢量和扩展单元 dirty 状态总和位

- 当 SD=1 时，表明浮点或矢量或扩展单元处在 dirty 状态。
 - 当 SD=0 时，表明浮点、矢量和扩展单元处都不处在 dirty 状态。
- 该位会被 reset 置为 1' b0。

16.1.2.2 机器模式处理器指令集架构寄存器 (MISA)

机器模式处理器指令集架构寄存器 (MISA) 存储了处理器所支持的指令集架构特性。

该寄存器的位长是 64 位，寄存器的读写权限是机器模式可读写，即非机器模式访问都会导致非法指令异常。

C906 支持的指令集架构为 RV64GCV，对应的 MISA 寄存器复位值为 64'h8000_0000_00b4_112d。具体的赋值规则请参考 RISC-V 官方文档《riscv-privileged》。

C906 不支持动态配置 MISA 寄存器，对该寄存器进行写操作不产生任何效果。

16.1.2.3 机器模式异常降级控制寄存器 (MEDELEG)

机器模式异常降级控制寄存器 (MEDELEG) 可以将超级用户和用户模式发生的异常降级到超级用户模式响应。MEDELEG 低 16 比特和表 3.9 异常和中断向量分配中异常向量号一一对应，可以选择将异常降级到超级用户模式响应。

该寄存器的位长是 64 位，复位值为 64'h0。寄存器的读写权限是机器模式可读写，即非机器模式访问都会导致非法指令异常。

16.1.2.4 机器模式中断降级控制寄存器 (MIDELEG)

机器模式中断降级控制寄存器 (MIDELEG) 可以将超级用户模式中断降级到超级用户模式响应。

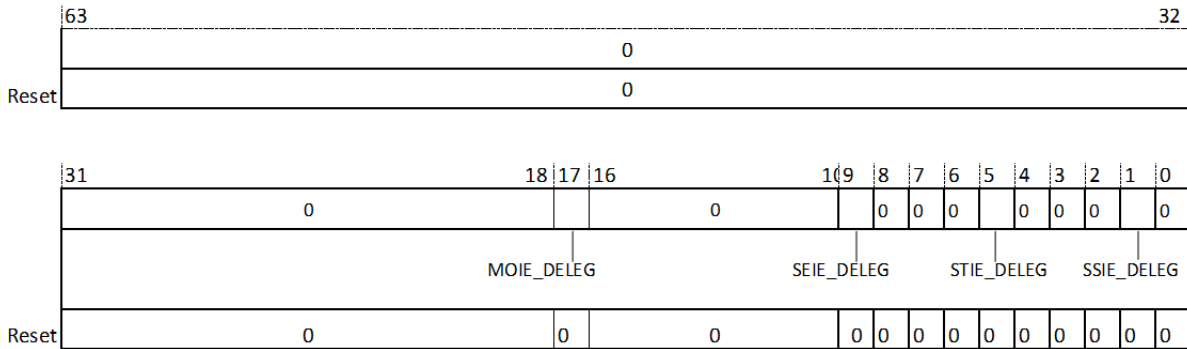


图 16.2: 机器模式中断降级控制寄存器 (MIDELEG)

该寄存器的位长是 64 位，寄存器的读写权限是机器模式可读写，即非机器模式访问都会导致非法指令异常。

SSIE_DELEG-超级用户模式软件中断

- 当 SSIE_DELEG=1 时，表明超级用户模式软件中断可以在处理器被降级到超级用户模式下处理。
- 当 SSIE_DELEG=0 时，表明超级用户模式软件中断只能在处理器切换到机器模式下被处理。

该位会被 reset 置为 1'b0。

STIE_DELEG-超级用户模式计时器中断

- 当 STIE_DELEG=1 时，表明超级用户模式计时器中断可以在处理器被降级到超级用户模式下处理。
- 当 STIE_DELEG=0 时，表明超级用户模式计时器中断只能在处理器切换到机器模式下被处理。

该位会被 reset 置为 1' b0。

SEIE_DELEG-超级用户模式外部中断

- 当 SEIE_DELEG=1 时，表明超级用户模式外部中断可以在处理器被降级到超级用户模式下处理。
- 当 SEIE_DELEG=0 时，表明超级用户模式外部中断只能在处理器切换到机器模式下被处理。

该位会被 reset 置为 1' b0。

MOIE_DELEG-性能监测单元事件计数器溢出中断

- 当 MOIE_DELEG=1 时，表明性能监测单元事件计数器溢出中断可以在处理器被降级到超级用户模式下处理。
- 当 MOIE_DELEG=0 时，表明性能监测单元事件计数器溢出中断只能在处理器切换到机器模式下被处理。

该位会被 reset 置为 1' b0。

16.1.2.5 机器模式中断使能控制寄存器 (MIE)

机器模式中断使能控制寄存器 (MIE) 用于控制不同中断类型的使能和屏蔽。该寄存器的位长是 64 位，寄存器的读写权限是机器模式可读写，即非机器模式访问都会导致非法指令异常。

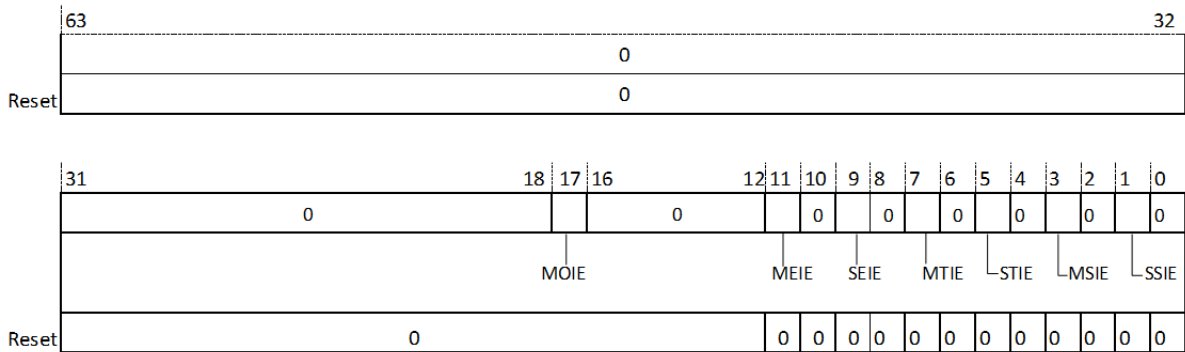


图 16.3: 机器模式中断使能控制寄存器 (MIE)

SSIE-超级用户模式软件中断使能位:

- 当 SEIE 为 0 时，超级用户模式软件外部中断未始能。
- 当 SEIE 为 1 时，超级用户模式软件外部中断被始能。

该位会被 reset 置为 1' b0。

MSIE-机器模式软件中断使能位:

- 当 MSIE 为 0 时，机器模式软件中断未始能。

- 当 MSIE 为 1 时，机器模式软件中断被始能。
该位会被 reset 置为 1' b0。

STIE-超级用户模式计时器中断使能位:

- 当 STIE 为 0 时，超级用户模式计时器中断未始能。
- 当 STIE 为 1 时，超级用户模式计时器中断被始能。
该位会被 reset 置为 1' b0。

MTIE-机器模式计时器中断使能位:

- 当 MTIE 为 0 时，机器模式计时器中断未始能。
- 当 MTIE 为 1 时，机器模式计时器中断被始能。
该位会被 reset 置为 1' b0。

SEIE-超级用户模式外部中断使能位:

- 当 SEIE 为 0 时，超级用户模式外部中断未始能。
- 当 SEIE 为 1 时，超级用户模式外部中断被始能。
该位会被 reset 置为 1' b0。

MEIE-机器模式外部中断使能位:

- 当 MEIE 为 0 时，机器模式外部中断未始能。
- 当 MEIE 为 1 时，机器模式外部中断被始能。
该位会被 reset 置为 1' b0。

MOIE-性能监测单元机器模式事件计数器溢出中断使能位:

- 当 MOIE 为 0 时，机器模式计数器溢出中断未始能。
- 当 MOIE 为 1 时，机器模式计数器溢出中断被始能。
该位会被 reset 置为 1' b0。

16.1.2.6 机器模式向量基址寄存器 (MTVEC)

机器模式向量基址寄存器 (MTVEC) 用于配置异常服务程序的入口地址。

该寄存器的位长是 64 位，寄存器的读写权限是机器模式可读写，即非机器模式访问都会导致非法指令异常。

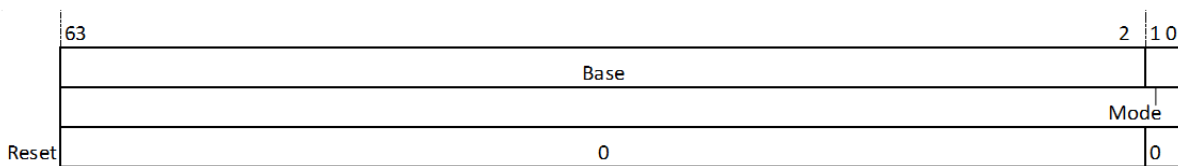


图 16.4: 机器模式向量基址寄存器 (MTVEC)

BASE-向量基址位:

向量基址位指示了异常服务程序入口地址的高 37 位，将此基址拼接 2' b00 即可得到异常服务程序入口地址。

该位会被 reset 置为 62' h0。

MODE-向量入口模式位:

- 当 MODE[1:0] 为 2' b00 时，异常和中断都统一使用 BASE 地址作为异常入口地址。
- 当 MODE[1:0] 为 2' b01 时，异常使用 BASE 地址作为入口地址，中断使用 BASE + 4*Exception Code。

该位会被 reset 置为 2' b00。

16.1.2.7 机器模式计数器访问授权寄存器 (MCOUNTEREN)

机器模式计数器访问授权寄存器 (MCOUNTEREN)，用于授权超级用户模式是否可以访问用户模式计数器。

具体请参考性能监测单元。

16.1.3 机器模式异常处理寄存器组**16.1.3.1 机器模式异常临时数据备份寄存器 (MSCRATCH)**

机器模式异常临时数据备份寄存器 (MSCRATCH) 用于处理器在异常服务程序中备份临时数据。一般用来存储机器模式本地上下文空间的入口指针值。

该寄存器的位长是 64 位，寄存器的读写权限是机器模式可读写，即非机器模式访问都会导致非法指令异常。

16.1.3.2 机器模式异常保留程序计数器寄存器 (MEPC)

机器模式异常保留程序计数器 (MEPC) 用于存储程序从异常服务程序退出时要返回的程序计数器值 (即 PC 值)。C906 支持 RVC 指令集，MEPC 的值以 16 位宽对齐，最低位为零。

该寄存器的位长是 64 位，寄存器的读写权限是机器模式可读写，即非机器模式访问都会导致非法指令异常。

16.1.3.3 机器模式异常事件向量寄存器 (MCAUSE)

机器模式异常事件向量寄存器 (MCAUSE) 用于保存触发异常的异常事件向量号，用于在异常服务程序中处理对应事件。

该寄存器的位长是 64 位，寄存器的读写权限是机器模式可读写，即非机器模式访问都会导致非法指令异常。

Interrupt-中断标记位:

- 当 Interrupt 位为 0 时，表示触发异常的来源不是中断，Exception Code 按照异常解析。

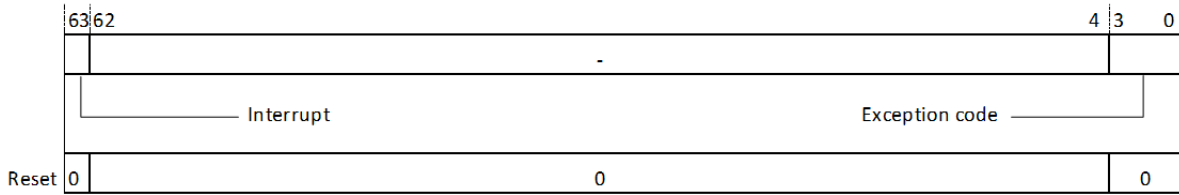


图 16.5: 机器模式异常事件向量寄存器 (MCAUSE)

- 当 Interrupt 位为 1 时，表示触发异常的来源是中断，Exception Code 按照中断解析。该位会被 reset 置为 1' b0。

Exception Code-异常向量号位:

在处理器响应异常或中断时，该域会被更新为对应异常号，具体请参考表 3.9 异常和中断向量分配。该位会被 reset 置为 5' b0。

16.1.3.4 机器模式中断等待状态寄存器 (MIP)

机器模式中断等待状态寄存器 (MIP) 用于保存处理器的中断等待状态。当处理器出现中断无法立即响应的情况时，MIP 寄存器中的对应位会被置位。

该寄存器的位长是 64 位，非机器模式访问都会导致非法指令异常。

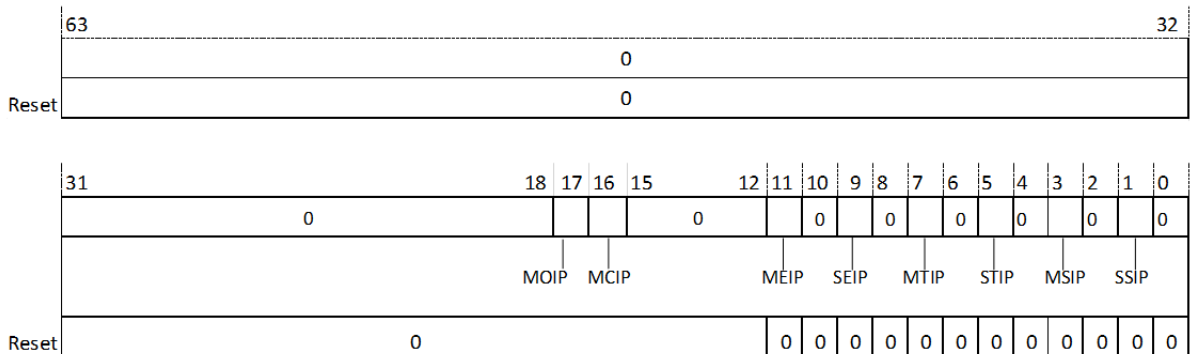


图 16.6: 机器模式中断等待状态寄存器 (MIP)

SSIP-超级用户模式软件中断等待位:

- 当 SSIP 为 0 时，处理器当前没有处于等待状态的超级用户模式软件中断。
- 当 SSIP 为 1 时，处理器当前有处于等待状态的超级用户模式软件中断。

该位会被 reset 置为 1' b0，访问权限为机器模式可读写。

MSIP-机器模式软件中断等待位:

- 当 MSIP 为 0 时，处理器当前没有处于等待状态的机器模式软件中断。
- 当 MSIP 为 1 时，处理器当前有处于等待状态的机器模式软件中断。

该位会被 reset 置为 1'b0，访问权限为机器模式只读。

STIP-超级用户模式计时器中断等待位：

- 当 STIP 为 0 时，处理器当前没有处于等待状态的超级用户模式计时器中断。
- 当 STIP 为 1 时，处理器当前有处于等待状态的超级用户模式计时器中断。

该位会被 reset 置为 1'b0，访问权限为机器模式可读写。

MTIP-机器模式计时器中断等待位：

- 当 MTIP 为 0 时，处理器当前没有处于等待状态的机器模式计时器中断。
- 当 MTIP 为 1 时，处理器当前有处于等待状态的机器模式计时器中断。

该位会被 reset 置为 1'b0，访问权限为机器模式只读。

SEIP-超级用户模式外部中断等待位：

- 当 SEIP 为 0 时，处理器当前没有处于等待状态的超级用户模式外部中断。
- 当 SEIP 为 1 时，处理器当前有处于等待状态的超级用户模式外部中断。

该位会被 reset 置为 1'b0，访问权限为机器模式可读写。

MEIP-外部中断等待位：

- 当 MEIP 为 0 时，处理器当前没有处于等待状态的机器模式外部中断。
- 当 MEIP 为 1 时，处理器当前有处于等待状态的机器模式外部中断。

该位会被 reset 置为 1'b0，访问权限为机器模式只读。

MOIP-机器模式事件计数器溢出中断等待位：

- 当 MOIP 为 0 时，处理器当前没有处于等待状态的机器模式计数器溢出中断。
- 当 MOIP 为 1 时，处理器当前有处于等待状态的机器模式计数器溢出中断。

该位会被 reset 置为 1'b0，访问权限为机器模式只读。

16.1.4 机器模式内存保护寄存器组

机器模式内存保护寄存器组是和内存保护单元设置相关的控制寄存器。

16.1.4.1 机器模式物理内存保护配置寄存器 (PMPCFG)

机器模式物理内存保护配置寄存器 (PMPCFG) 用于配置物理内存的访问权限、地址匹配模式。

该寄存器的位长是 64 位，寄存器的读写权限是机器模式可读写，即非机器模式访问都会导致非法指令异常。

具体信息请参考物理内存保护设置寄存器 (PMPCFG)。

16.1.4.2 机器模式物理内存地址寄存器 (PMPADDR)

机器模式物理内存地址寄存器 (PMPADDR) 用于配置物理内存的每个表项的地址区间。

该寄存器的位长是 64 位，寄存器的读写权限是机器模式可读写，即非机器模式访问都会导致非法指令异常。

具体信息请参考物理内存保护地址寄存器 (PMPADDR)。

16.1.5 机器模式计数器寄存器组

机器模式计数器寄存器组属于性能监测单元，用于统计程序运行中的软件信息和部分硬件信息，供软件开发人员进行程序优化。

16.1.5.1 机器模式周期计数器 (MCYCLE)

机器模式周期计数器 (MCYCLE) 用于存储处理器已经执行的时钟周期数，当处理器处于执行状态（即非低功耗状态）下，MCYCLE 寄存器就会在每个执行周期自增计数，调试模式下该计数器停止计数。

机器模式周期计数器为 64 位，周期计数器会被 reset 清零。

具体信息请参考事件计数器。

16.1.5.2 机器模式退休指令计数器 (MINSTRET)

机器模式退休指令计数器 (MINSTRET) 用于存储处理器已经退休的指令数，MINSTRET 寄存器会在每条指令退休时自增计数，调试模式下执行的指令不会触发该计数器计数。

退休指令计数器为 64 位，退休计数器会被 reset 清零。

具体信息请参考事件计数器。

16.1.5.3 机器模式事件计数器 (MHPMCOUNTERn)

机器模式事件计数器 (MHPMCOUNTERn) 用于对事件进行计数。

事件计数器为 64 位，事件计数器会被 reset 清零。

具体信息请参考事件计数器。

16.1.6 机器模式计数器配置寄存器组

机器模式计数器配置寄存器用于给机器模式事件计数器选择计数的事件。

16.1.6.1 机器模式事件选择器 (MHPMEVENTn)

机器模式计数器配置寄存器 (MHPMEVENTn) 用于给机器模式事件计数器选择计数事件，MHPMEVENT3-17 和 MHPMCOUNTER3-17 一一对应。在 C906 中，各事件计数器只能对固定事件进行计数，因此，MHPMEVENT3-17 只能写入指定数值。

事件选择器均为 64 位，事件选择器会被 reset 清零。

具体信息请参考性能监测事件选择寄存器。

16.1.7 机器模式处理器控制和状态扩展寄存器组

C906 对处理器和状态扩展了部分寄存器，包含：机器模式扩展状态寄存器 (MXSTATUS)、机器模式硬件控制寄存器 (MHCR)、机器模式硬件操作寄存器 (MCOR)、机器模式隐式操作寄存器 (MHINT)、机器模式复位向量基址寄存器 (MRVBR)、机器模式计数器写使能授权寄存器 (MCOUNTERWEN)、机器模式事件计数器溢出中断使能寄存器 (MCOUNTERINTEN)、机器模式事件计数器上溢出标注寄存器 (MCOUNTEROF)。

16.1.7.1 机器模式扩展状态寄存器 (MXSTATUS)

机器模式扩展状态寄存器 (MXSTATUS) 存储了处理器当前所处特权模式和 C906 扩展功能开关位。

该寄存器的位长是 64 位，寄存器的读写权限是机器模式可读写，即非机器模式访问都会导致非法指令异常。

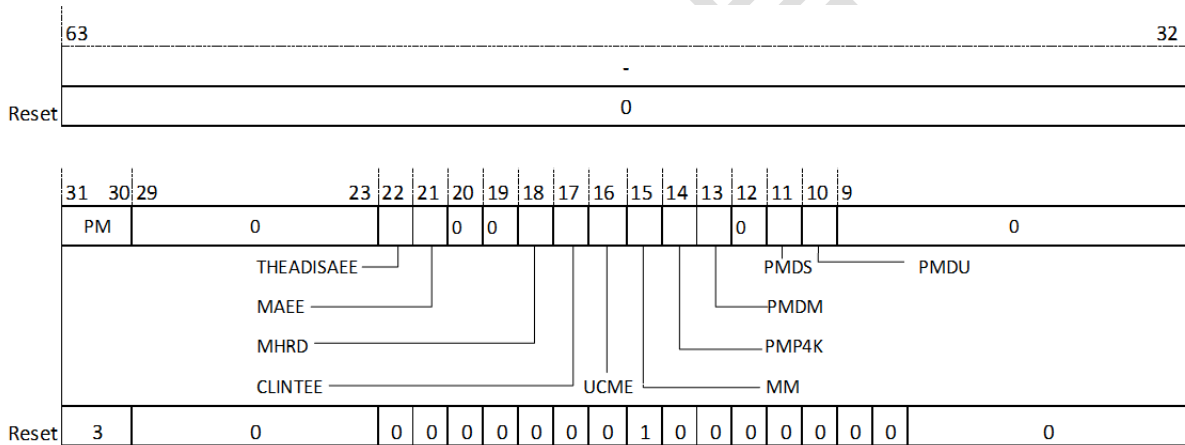


图 16.7: 机器模式扩展状态寄存器 (MXSTATUS)

PMDU-用户模式性能监测计数使能位：

- 当 PMDU 为 0 时，用户模式下允许性能计数器计数。
- 当 PMDU 为 1 时，用户模式下禁止性能计数器计数。

该位会被 reset 置为 1' b0。

PMDS-超级用户模式性能监测计数使能位：

- 当 PMDS 为 0 时，超级用户模式下允许性能计数器计数。

- 当 PMDS 为 1 时，超级用户模式下禁止性能计数器计数。

PMDM-机器模式性能监测计数使能位：

- 当 PMDM 为 0 时，机器模式下允许性能计数器计数。
- 当 PMDM 为 1 时，机器模式下禁止性能计数器计数。

该位会被 reset 置为 1' b0。

PMP4K-PMP 最小粒度控制位：

C906 当前只支持 PMP 最小粒度为 4K，不受该位影响。

该位会被 reset 置为 1' b0。

MM-非对齐访问使能位：

- 当 MM 为 0 时，不支持非对齐访问，非对齐访问将产生非对齐异常。
- 当 MM 为 1 时，支持非对齐访问，硬件处理非对齐访问。

该位会被 reset 置为 1' b1。

UCME-U 态执行扩展 cache 指令：

- 当 UCME 为 0 时，用户模式不能执行扩展的 cache 操作指令，产生非法指令异常。
- 当 UCME 为 1 时，用户模式可以执行扩展的 DCACHE.CIVA/DCACHE.CVA/ICACHE.IVA 指令。

该位会被 reset 置为 1' b0。

CLINTEE-CLINT 计时器/软件中断超级用户扩展使能位：

- 当 CLINTEE 为 0 时，CLINT 发起的超级用户软件中断和计时器中断不会被响应。
- 当 CLINTEE 为 1 时，CLINT 发起的超级用户软件中断和计时器中断可以被响应。

该位会被 reset 置为 1' b0。

MHRD-关闭硬件回填：

- 当 MHRD 为 0 时，TLB 缺失后，硬件会进行硬件回填。
- 当 MHRD 为 1 时，TLB 缺失后，硬件不会进行硬件回填。

该位会被 reset 置为 1' b0。

MAEE-扩展 MMU 地址属性：

- 当 MAEE 为 0 时，不扩展 MMU 地址属性。
- 当 MAEE 为 1 时，MMU 的 pte 中扩展地址属性位，用户可以配置页面的地址属性。

该位会被 reset 置为 1' b0。

THEADISAEE-使能扩展指令集：

- 当 THEADISAEE 为 0 时，执行 C906 扩展指令时将触发非法指令异常。
- 当 THEADISAEE 为 1 时，可以正常执行 C906 扩展指令。

该位会被 reset 置为 1' b0。

PM-处理器所处特权模式：

- 当 PM=2' b00 时，表征当前处理器运行在用户模式。
- 当 PM=2' b01 时，表征当前处理器运行在超级用户模式。
- 当 PM=2' b11 时，表征当前处理器运行在机器模式。

该位会被 reset 置为 2' b11。

16.1.7.2 机器模式硬件配置寄存器 (MHCR)

机器模式硬件配置寄存器 (MHCR) 用于对处理器进行配置，包括性能和功能。

该寄存器的位长是 64 位，寄存器的读写权限是机器模式可读写，即非机器模式访问都会导致非法指令异常。

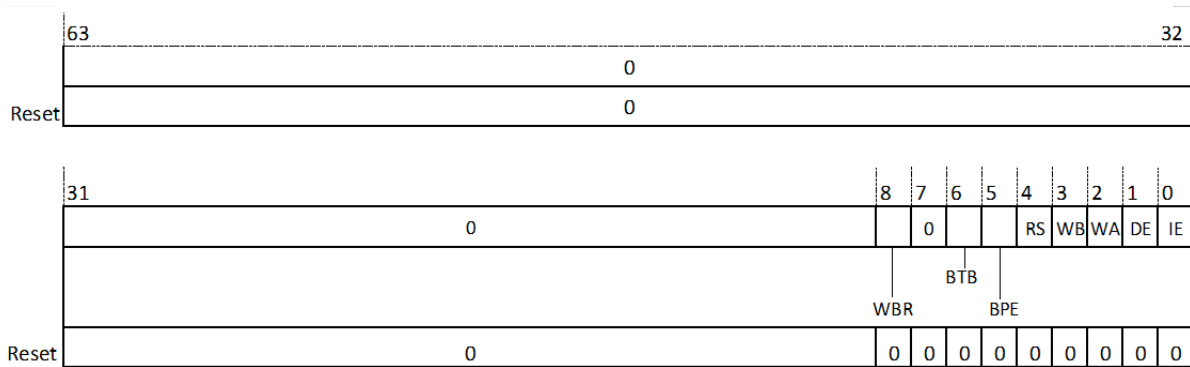


图 16.8: 机器模式硬件配置寄存器 (MHCR)

IE-Icache 使能位：

- 当 IE=0 时，Icache 关闭；
- 当 IE=1 时，Icache 打开。

该位会被 reset 置为 1' b0。

DE-Dcache 使能位：

- 当 DE=0 时，Dcache 关闭；
- 当 DE=1 时，Dcache 打开。

该位会被 reset 置为 1' b0。

WA-高速缓存写分配设置位：

- 当 WA=0 时，数据高速缓存为 write non-allocate 模式；
- 当 WA=1 时，数据高速缓存为 write allocate 模式。

该位会被 reset 置为 1' b0。

WB-高速缓存写回设置位：

- 当 WB=0 时，数据高速缓存为 write through 模式。

- 当 WB=1 时，数据高速缓存为 write back 模式。

C906 只支持 write back 模式，WB 固定为 1。

RS-地址返回栈设置位：

- 当 RS=0 时，返回栈关闭；
- 当 RS=1 时，返回栈开启。

该位会被 reset 置为 1' b0。

BPE-允许预测跳转设置位：

- 当 BPE=0 时，预测跳转关闭；
- 当 BPE=1 时，预测跳转开启。

该位会被 reset 置为 1' b0。

BTB-分支目标预测使能位：

- 当 BTB=0 时，分支目标预测关闭。
- 当 BTB=1 时，分支目标预测开启。

该位会被 reset 置为 1' b0。

WBR-写突发传输使能位：

- 当 WBR=0 时，不支持写突发传输。
- 当 WBR=1 时，支持写突发传输。

C906 中 WBR 固定为 1。

16.1.7.3 机器模式硬件操作寄存器 (MCOR)

机器模式硬件操作寄存器 (MCOR) 用于对高速缓存和分支预测部件进行相关操作。

该寄存器的位长是 64 位，寄存器的读写权限是机器模式可读写，即非机器模式访问都会导致非法指令异常。

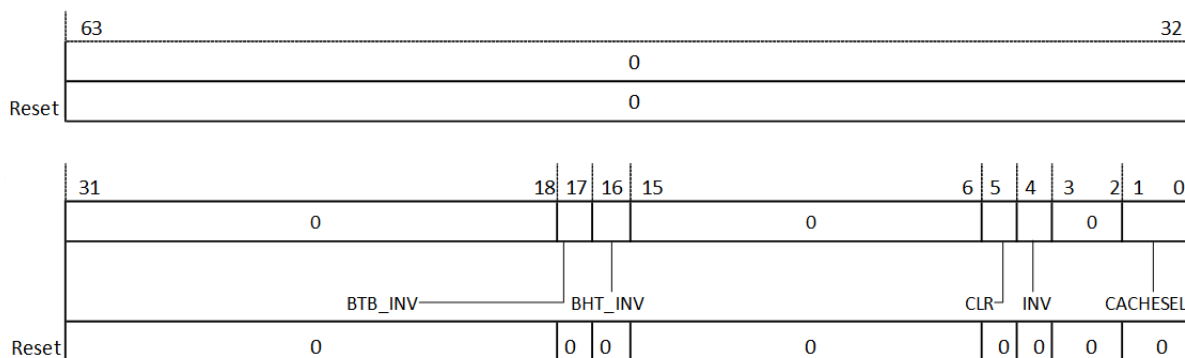


图 16.9: 机器模式硬件操作寄存器 (MCOR)

CACHESEL-高速缓存选择位:

- 当 CACHE_SEL=2' b01 时, 选中指令高速缓存;
- 当 CACHE_SEL=2' b10 时, 选中数据高速缓存;
- 当 CACHE_SEL=2' b11 时, 选中指令和数据高速缓存。

该位会被 reset 置为 2' b00。

INV-高速缓存无效化设置位:

- 当 INV=0 时, 高速缓存不在进行无效化操作;
- 当 INV=1 时, 高速缓存正在进行无效化操作。

该位会被 reset 置为 1' b0。

CLR-高速缓存脏表项清除设置位:

- 当 CLR=0 时, 高速缓存被标记为脏的表项不会被写到片外;
- 当 CLR=1 时, 高速缓存被标记为脏的表项会被写到片外。

该位会被 reset 置为 1' b0。

BHT_INV-BHT 无效设置位:

- 当 BHT_INV=0 时, 分支历史表内的数据不进行无效化;
- 当 BHT_INV=1 时, 分支历史表内的数据进行无效化。

该位会被 reset 置为 1' b0。

BTB_INV-BTB 无效设置位:

- 当 BTB_INV=0 时, 分支目标缓冲器内的数据不进行无效化;
- 当 BTB_INV=1 时, 分支目标缓冲器内的数据进行无效化。

该位会被 reset 置为 1' b0。

以上所有无效化操作和清脏表项操作, 在写的时候置高, 在操作完成时, 清 0。

16.1.7.4 机器模式隐式操作寄存器 (MHINT)

机器模式隐式操作寄存器 (MHINT) 用于高速缓存多种功能开关控制。

该寄存器的位长是 64 位, 寄存器的读写权限是机器模式可读写, 即非机器模式访问都会导致非法指令异常。

DPLD-DCACHE 预取使能位:

- 当 DPLD 为 0 时, DCACHE 预取关闭;
- 当 DPLD 为 1 时, DCACHE 预取开启。

该位会被 reset 置为 1' b0。

AMR-L1 DCache 写分配策略自动调整使能位:

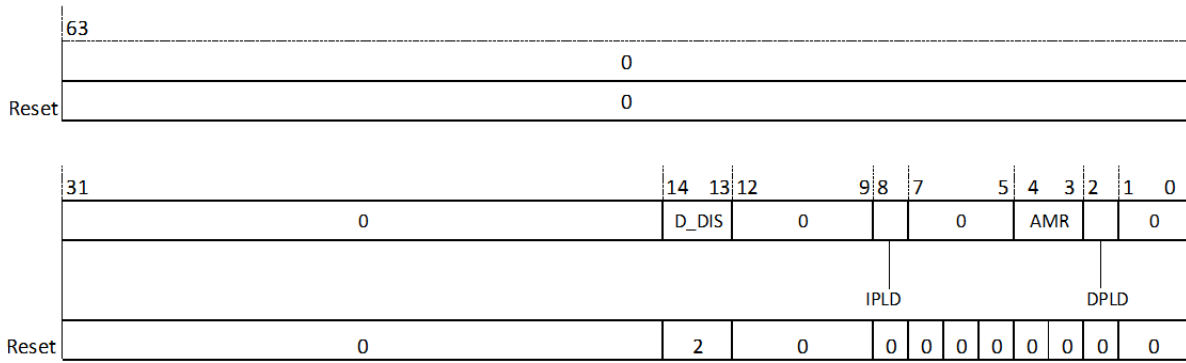


图 16.10: 机器模式隐式操作寄存器 (MHINT)

- 当 AMR 为 0 时, 写分配策略由访问地址的页面属性 WA 决定;
 - 当 AMR 为 1 时, 在出现连续 3 条缓存行的存储操作时后续连续地址的存储操作不再写入 L1 Cache;
 - 当 AMR 为 2 时, 在出现连续 64 条缓存行的存储操作时后续连续地址的存储操作不再写入 L1 Cache;
 - 当 AMR 为 3 时, 在出现连续 128 条缓存行的存储操作时后续连续地址的存储操作不再写入 L1 Cache。
- 该位会被 reset 置为 2' b0。

IPLD-ICACHE 预取使能位:

- 当 IPLD 为 0 时, ICACHE 预取关闭;
 - 当 IPLD 为 1 时, ICACHE 预取开启。
- 该位会被 reset 置为 1' b0。

IWPE-ICACHE 路预测使能位:

- 当 IWPE 为 0 时, ICACHE 路预测关闭;
 - 当 IWPE 为 1 时, ICACHE 路预测开启。
- 该位会被 reset 置为 1' b0。

D_DIS-DCACHE 预取缓存行数量:

- 当 DPLD 为 0 时, 预取 2 条缓存行;
 - 当 DPLD 为 1 时, 预取 4 条缓存行;
 - 当 DPLD 为 2 时, 预取 8 条缓存行;
 - 当 DPLD 为 3 时, 预取 16 条缓存行。
- 该位会被 reset 置为 2' b10。

16.1.7.5 机器模式复位向量基址寄存器 (MRVBR)

机器模式复位寄存器 (MRVBR) 用来保存复位异常向量的基址。

该寄存器的位长是 64 位, 寄存器的读写权限是机器模式可读写, 即非机器模式访问都会导致非法指令异常。

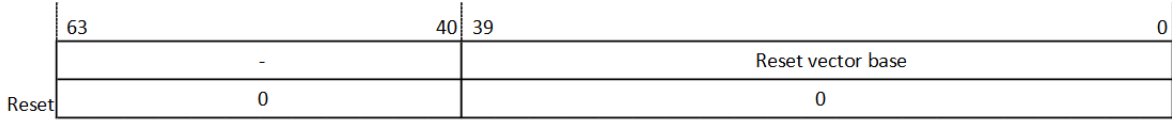


图 16.11: 机器模式复位向量基址寄存器 (MRVBR)

Reset vector base-复位基址:

该寄存器的值由 SoC 系统集成时传入 C906 的复位启动地址，机器模式写无效。

16.1.7.6 机器模式计数器写使能授权寄存器 (MCOUNTERWEN)

机器模式计数器写使能授权寄存器 (MCOUNTERWEN)，用于授权超级用户模式是否可以写机器模式计数器的镜像寄存器 SHPMCOUNTERn 等。

该寄存器的位长是 64 位，寄存器的读写权限是机器模式可读写，即非机器模式访问都会导致非法指令异常。

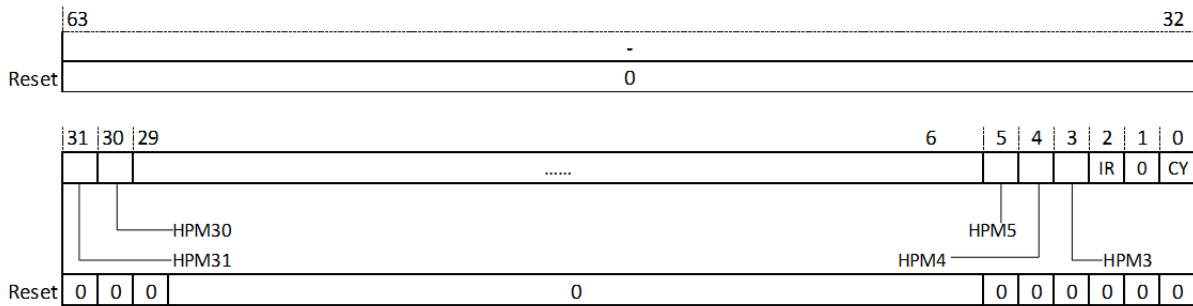


图 16.12: 超级用户态计数器写使能寄存器 (MCOUNTERWEN)

- 当 MCOUNTERWEN.bit[n] 为 1 时，允许超级用户模式下写对应 SHPMCOUNTERn、SINSTRET、SCYCLE 寄存器；
- 当 MCOUNTERWEN.bit[n] 为 0 时，不允许超级用户模式下写对应 SHPMCOUNTERn、SINSTRET、SCYCLE 寄存器，否则产生非法指令异常。

16.1.7.7 机器模式事件中断使能寄存器 (MCOUNTERINTEN)

机器模式事件中断使能寄存器 (MCOUNTERINTEN)，用于使能各事件计数上溢出时产生中断；

该寄存器的位长是 64 位，寄存器的读写权限是机器模式可读写，即非机器模式访问都会导致非法指令异常。详情请见机器模式计数器写使能授权寄存器 (MCOUNTERWEN)。

16.1.7.8 机器模式事件计数器上溢出标注寄存器 (MCOUNTEROF)

机器模式事件计数器上溢出标注寄存器 (MCOUNTEROF)，用于表示各事件计数器是否发生了上溢出。

该寄存器的位长是 64 位，寄存器的读写权限是机器模式可读写，即非机器模式访问都会导致非法指令异常。详情请见性能检测单元事件溢出中断。

16.1.7.9 机器模式外设地址高位寄存器 (MAPBADDR)

该寄存器用于标识 PLIC 等核内挂载在 APB 总线上的寄存器基址的高 13 位，该地址由 SoC 集成使用 C906 时指定，该寄存器位长为 64 位，机器模式只读。

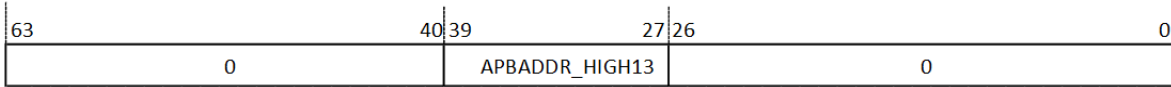


图 16.13: 机器模式外设地址高位寄存器 (MAPBADDR)

16.1.8 机器模式 Cache 访问扩展寄存器组

机器模式 Cache 访问扩展寄存器用于直接读取 L1 高速缓存中内容，便于对高速缓存进行调试。

16.1.8.1 机器模式 Cache 指令寄存器 (MCINS)

机器模式 Cache 指令寄存器 (MCINS) 用于向 L1 高速缓存发起读请求。

该寄存器的位长是 64 位，寄存器的读写权限是机器模式可读写，即非机器模式访问都会导致非法指令异常。

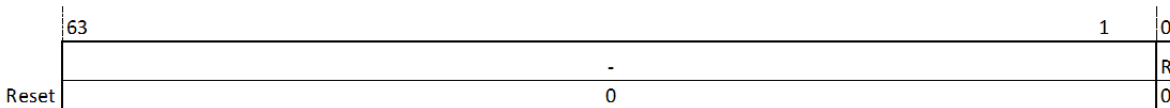


图 16.14: 机器模式 Cache 指令寄存器 (MCINS)

R-Cache 读访问:

- 当 R 为 0 时，不发起 Cache 读请求；
- 当 R 为 1 时，发起 Cache 读请求。

该位会被 reset 置为 1' b0。

16.1.8.2 机器模式 Cache 访问索引寄存器 (MCINDEX)

机器模式 Cache 访问索引寄存器 (MCINDEX) 用于配置读请求访问的 Cache 位置信息。

该寄存器的位长是 64 位，寄存器的读写权限是机器模式可读写，即非机器模式访问都会导致非法指令异常。

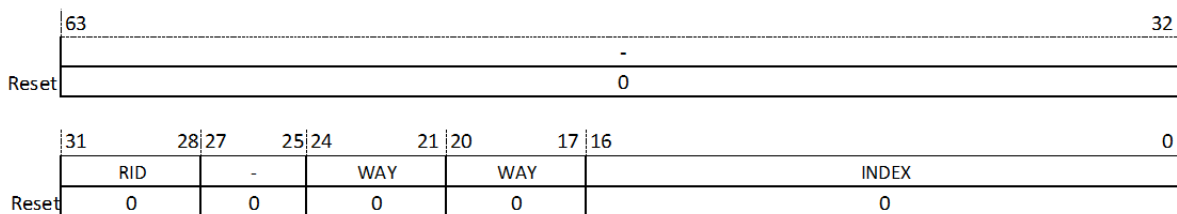


图 16.15: 机器模式 Cache 访问索引寄存器 (MCINDEX)

RID-RAM 标志位:

指示访问的 RAM 信息。

- 当 RID 为 0 时, 表示访问的是 ICACHE TAG RAM。
- 当 RID 为 1 时, 表示访问的是 ICACHE DATA RAM。
- 当 RID 为 2 时, 表示访问的是 DCACHE TAG RAM。
- 当 RID 为 3 时, 表示访问的是 DCACHE DATA RAM。

该位会被 reset 置为 4' b0。

WAY-Cache 路信息:

指示 RAM 访问的路位置信息。

该位会被 reset 置为 4' b0。

INDEX-Cache 索引:

指示 RAM 访问的索引位置信息。该域按照地址的格式写入:

对于 data array, 每次读取 16 字节, index 的使用会忽略低 4 位。

对于 tag array, 因 C906 cacheline 大小为 64 字节, index 的使用会忽略低 6 位。

该位会被 reset 置为 21' b0。

16.1.8.3 机器模式 Cache 数据寄存器 (MCDATA0/1)

机器模式 Cache 数据寄存器 (MCDATA0/1) 用于记录读取 L1 高速缓存的数据。

该寄存器的位长是 64 位, 寄存器的读写权限是机器模式可读写, 即非机器模式访问都会导致非法指令异常。

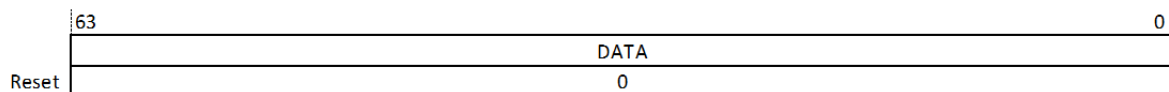


图 16.16: 机器模式 Cache 访问数据寄存器 (MCDATA)

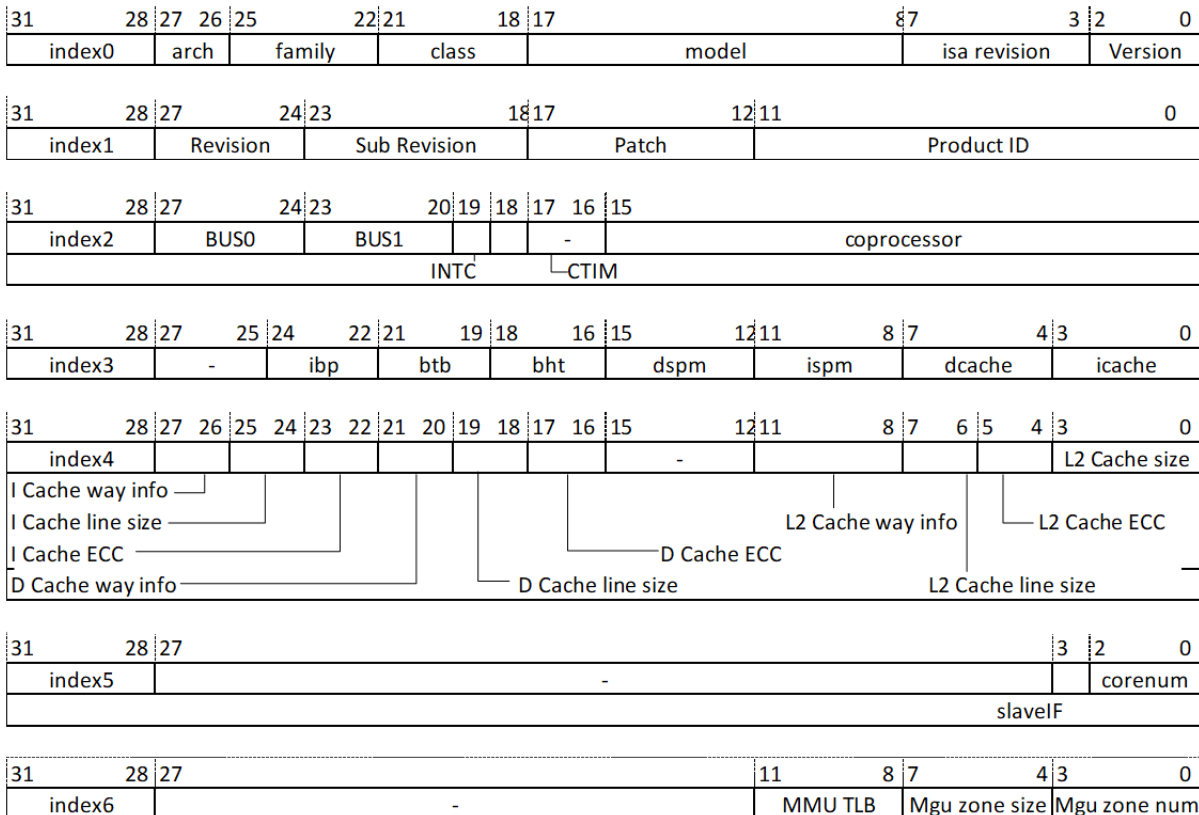
表 16.1: 机器模式 Cache 访问数据寄存器与 RAM 类型对应关系

RAM 类型	CDATA 内容
ICACHE TAG	CDATA0[39:12]: TAG CDATA0[0]: VALID
ICACHE DATA	CDATA1: DATA[127:64] CDATA0: DATA[63:0]
DCACHE TAG	CDATA0[39:12]: TAG CDATA0[2]: DIRTY CDATA0[0]: VALID
DCACHE DATA	CDATA1: DATA[127:64] CDATA0: DATA[63:0]

16.1.9 机器模式处理器型号寄存器组

16.1.9.1 机器模式处理器型号寄存器 (MCPUID)

机器模式处理器型号寄存器 (MCPUID) 存储了处理器型号信息，其复位值由产品本身决定并遵守平头哥产品定义规范，便于软件识别。通过连续读取 MCPUID 寄存器可以获取至多 7 个不同的返回值用于表示 C906 的产品信息，如图 ?? 所示。



玄铁 CPU 的调试代理程序 DebugServer 在连接包含玄铁 CPU 的芯片时会自动将上述信息进行解析并打印出来供用户识别。

16.2 附录 C-2 超级用户模式控制寄存器

超级用户模式控制寄存器按照功能分为：超级用户模式异常配置寄存器组、超级用户模式异常处理寄存器组、超级用户模式地址转换寄存器组、超级用户模式处理器控制和状态扩展寄存器组、超级用户模式 MMU 扩展寄存器组。

16.2.1 超级用户模式异常配置寄存器组

当异常和中断被降级到超级用户模式响应时，跟机器模式一样，需要通过超级用户模式异常配置寄存器组进行异常的配置。

16.2.1.1 超级用户模式处理器状态寄存器 (SSTATUS)

超级用户模式处理器状态寄存器 (SSTATUS) 存储了处理器在超级用户模式下的状态和控制信息，包括全局中断有效位、异常保留中断有效位、异常保留特权模式位等，是 MSTATUS 的部分映射。

该寄存器的位长是 64 位，寄存器的读写权限是机器模式和超级用户模式可读写，用户模式访问会导致非法指令异常。

	63	62																															36	35	34	33	32
	SD	0																														0	UXL				
Reset	0	0																														2	2				

	31																								25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	0																							VS	0	0	0				0	XS	FS	0	0	SPP	0	0			0	0	0	0	SIE	0				
																								MXR			SUM			SPIE																				
Reset	0	0																							0	0	0	0	0	0	0	0	0	0	0	0	3	0	1	0	0	0	0	0	0	0	0	0		

图 16.17: 超级用户模式处理器状态寄存器 (SSTATUS)

具体信息请参考机器模式处理器状态寄存器 (MSTATUS)。

16.2.1.2 超级用户模式中断使能控制寄存器 (SIE)

超级用户模式中断使能控制寄存器 (SIE) 用于控制不同中断类型的使能和屏蔽，是 MIE 的部分映射。该寄存器的位长是 64 位，寄存器的读写权限是机器模式可读写，超级用户模式可读，超级用户模式下的写权限由 MIDELEG 对应位决定，用户模式访问会导致非法指令异常。

具体信息请参考机器模式中断使能控制寄存器 (MIE)。

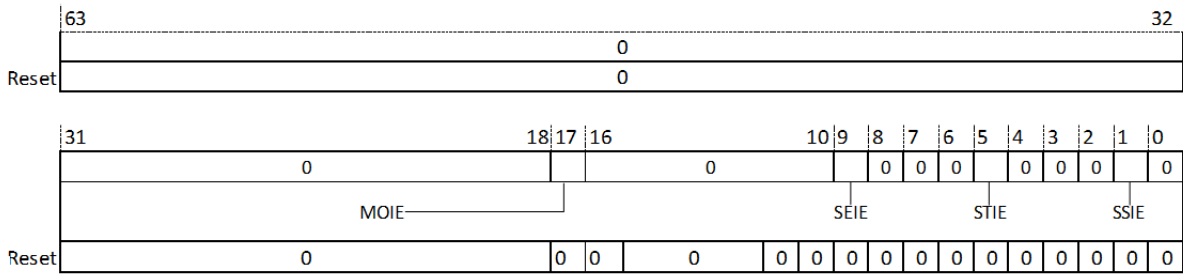


图 16.18: 超级用户模式中断使能控制寄存器 (SIE)

16.2.1.3 超级用户模式向量基址寄存器 (STVEC)

超级用户模式向量基址寄存器 (STVEC) 用于配置异常服务程序的入口地址，用于在异常和中断被降级到超级用户模式下处理时使用。

该寄存器的位长是 64 位，寄存器的读写权限是机器模式和超级用户模式可读写，用户模式访问会导致非法指令异常。

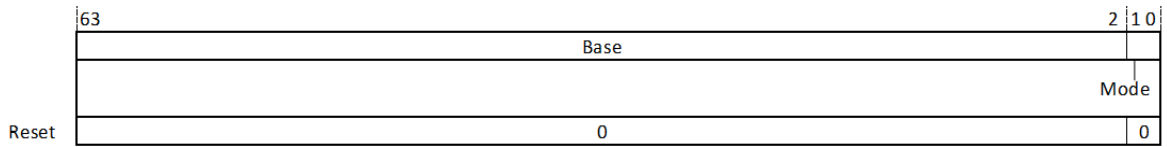


图 16.19: 超级用户模式向量基址寄存器 (STVEC)

具体信息请参考机器模式向量基址寄存器 (MTVEC)。每个域的含义相同。

16.2.1.4 超级用户模式计数器访问授权寄存器 (SCOUNTEREN)

超级模式计数器访问授权寄存器 (SCOUNTEREN)，用于授权用户模式是否可以访问用户模式计数器。

具体信息，请参考超级用户模式计数器访问授权寄存器 (SCOUNTEREN)。

16.2.2 超级用户模式异常处理寄存器组

16.2.2.1 超级用户模式异常临时数据备份寄存器 (SSCRATCH)

超级用户模式异常临时数据备份寄存器 (SSCRATCH) 用于处理器在异常服务程序中备份临时数据。一般用来存储超级用户模式本地上下文空间的入口指针值。

该寄存器的位长是 64 位，寄存器的读写权限是机器模式超级用户模式可读写，用户模式访问会导致非法指令异常。

16.2.2.2 超级用户模式异常保留程序计数器寄存器 (SEPC)

超级用户模式异常保留程序计数器 (SEPC) 用于存储程序从异常服务程序退出时的程序计数器值 (即 PC 值)。C906 支持 RVC 指令集, SEPC 的值以 16 位宽对齐, 最低位为零。

该寄存器的位长是 64 位, 寄存器的读写权限是机器模式和超级用户模式可读写, 用户模式访问会导致非法指令异常。

16.2.2.3 超级用户模式异常事件向量寄存器 (SCAUSE)

超级用户模式异常事件向量寄存器 (SCAUSE) 用于保存触发异常的异常事件向量号, 用于在异常服务程序中处理对应事件。

该寄存器的位长是 64 位, 寄存器的读写权限是机器模式和超级用户模式可读写, 用户模式访问都会导致非法指令异常。

16.2.2.4 超级用户模式中断等待状态寄存器 (SIP)

超级用户模式中断等待状态寄存器 (SIP) 用于保存处理器的中断等待状态。当处理器出现中断无法立即响应的情况时, SIP 寄存器中的对应位会被置位。

该寄存器的位长是 64 位, 寄存器的读写权限是机器模式可读写, 超级用户模式可读, 写权限由 MIDELEG 对应位决定, 用户模式访问会导致非法指令异常。

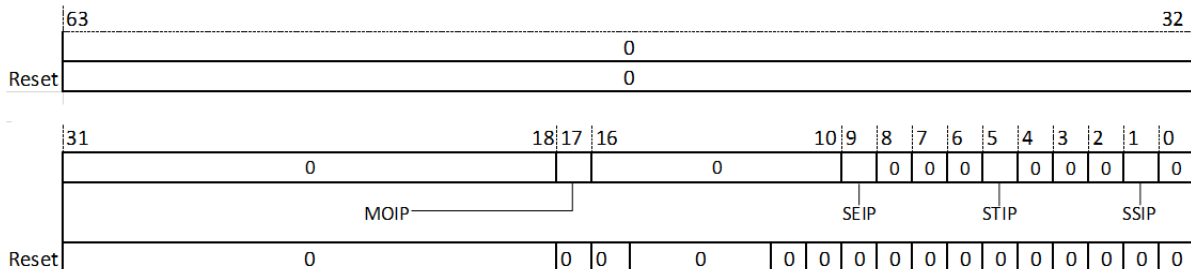


图 16.20: 超级用户模式中断等待状态寄存器 (SIP)

16.2.2.5 超级用户模式异常事件原因寄存器 (STVAL)

超级用户模式异常事件原因寄存器 (STVAL) 用于保存触发异常的异常事件原因, 用于在异常服务程序中处理对应事件。

该寄存器的位长是 64 位, 寄存器的读写权限是机器模式和超级用户模式可读写, 用户模式访问会导致非法指令异常。

16.2.3 超级用户模式地址转换寄存器组

超级用户模式下，需要访问虚拟内存空间。超级用户模式地址转换寄存器 (SATP) 用于控制 MMU 单元的模式切换、硬件回填基地址和进程号。

16.2.3.1 超级用户模式地址转换寄存器 (SATP)

超级用户模式地址转换寄存器 (SATP) 用于控制 MMU 单元的模式切换、硬件回填基地址和进程号。

该寄存器的位长是 64 位，寄存器的读写权限是机器模式和超级用户模式可读写，用户模式访问会导致非法指令异常。

具体信息请参考 *MMU 地址转换寄存器 (SATP)*。

16.2.4 超级用户模式处理器控制和状态扩展寄存器组

16.2.4.1 超级用户模式扩展状态寄存器 (SXSTATUS)

超级用户模式扩展状态寄存器 (SXSTATUS) 是机器模式扩展状态寄存器 (MXSTATUS) 的映射。

该寄存器的位长是 64 位，寄存器的读写权限是机器模式读写，超级用户模式可读，MM/PMDS/PMDM 位可写，用户模式访问会导致非法指令异常。

具体信息请参考 *机器模式扩展状态寄存器 (MXSTATUS)*。

16.2.4.2 超级用户模式硬件控制寄存器 (SHCR)

超级用户模式硬件控制寄存器 (SHCR) 是机器模式硬件控制寄存器 (MHCR) 的映射。

该寄存器的位长是 64 位，寄存器的读写权限是机器模式读写，超级用户模式可读，用户模式访问会导致非法指令异常。

具体信息请参考 *机器模式硬件配置寄存器 (MHCR)*。

16.2.4.3 超级用户模式事件溢出中断使能寄存器 (SCOUNTERINTEN)

超级用户模式事件计数器溢出中断使能寄存器 (SCOUNTERINTEN) 是机器模式事件计数器溢出中断使能寄存器 (MCOUNTERINTEN) 的映射。具体信息请参考 *事件计数器*。

该寄存器的位长是 64 位，寄存器的读写权限是机器模式读写，超级用户模式可读写，用户模式访问会导致非法指令异常。

当 MCOUNTERWEN.bit[n] 为 0 时，SCOUNTERINTEN.bit[n] 读为 0，写无效；当 MCOUNTERWEN.bit[n] 为 1 时，SCOUNTERINTEN.bit[n] 写有效。SCOUNTERINTEN.bit[n] 为 1 时，对应 SHPMCOUNTERn、SCYCLE、SINSTRET 计数上溢出中断被始能。

16.2.4.4 超级用户模式事件上溢出标注寄存器 (SCOUNTEROF)

超级用户模式事件计数器上溢出标注寄存器(SCOUNTEROF)是机器模式事件计数器上溢出标注寄存器(MCOUNTEROF)的映射,具体信息请参考事件计数器。

该寄存器的位长是 64 位,寄存器的读写权限是机器模式和超级用户模式可读写,用户模式访问会导致非法指令异常。

当 MCOUNTERWEN.bit[n] 为 1 时,SCOUNTEROF.bit[n] 表示对应 SHMPCOUNTERn、SCYCLE、SINSTRET 计数器是否产生溢出中断。当 MCOUNTERWEN.bit[n] 为 0 时,超级用户模式读 SCOUNTEROF.bit[n] 为 0,写无效。

16.2.4.5 超级用户模式周期计数器 (SCYCLE)

超级用户模式周期计数器 (SCYCLE) 用于存储处理器已经执行的时钟周期数,当处理器处于执行状态(即非低功耗状态)下,SCYCLE 寄存器就会在每个执行周期自增计数。

周期计数器为 64 位,周期计数器会被 reset 清零。

具体信息请参考事件计数器。

16.2.4.6 超级用户模式退休指令计数器 (SINSTRET)

超级用户模式退休指令计数器 (SINSTRET) 用于存储处理器已经退休的指令数,SINSTRET 寄存器会在每条指令退休时自增计数。

退休指令计数器为 64 位,退休指令计数器会被 reset 清零。

具体信息请参考事件计数器。

16.2.4.7 超级用户模式事件计数器 (SHMPCOUNTERn)

超级用户模式事件计数器 (SHMPCOUNTERn) 是机器模式事件计数器 (MHPMPCOUNTERn) 的映射。

具体信息请参考事件计数器。

16.2.5 超级用户模式 MMU 扩展寄存器组

C906 中 MMU 单元扩展了 MMU 相关寄存器,实现软件回填功能,软件可以直接对 TLB 进行读写等操作。

16.2.5.1 超级用户模式 MMU 控制寄存器 (SMCIR)

该寄存器的位长是 64 位,寄存器的读写权限是机器模式读写,超级用户模式可读,用户模式访问会导致非法指令异常。

具体信息请参考 MMU 控制寄存器 (SMCIR)。

16.2.5.2 超级用户模式 MMU 控制寄存器 (SMIR)

该寄存器的位长是 64 位，寄存器的读写权限是机器模式读写，超级用户模式可读，用户模式访问会导致非法指令异常。

具体信息请参考 *MMU Index 寄存器 (SMIR)*。

16.2.5.3 超级用户模式 MMU 控制寄存器 (SMEH)

该寄存器的位长是 64 位，寄存器的读写权限是机器模式读写，超级用户模式可读，用户模式访问会导致非法指令异常。

具体信息请参考 *MMU EntryHi 寄存器 (SMEH)*。

16.2.5.4 超级用户模式 MMU 控制寄存器 (SMEL)

该寄存器的位长是 64 位，寄存器的读写权限是机器模式读写，超级用户模式可读，用户模式访问会导致非法指令异常。

具体信息请参考 *MMU EntryLo 寄存器 (SMEL)*。

16.3 附录 C-3 用户模式控制寄存器

用户模式控制寄存器按照功能主要分为用户模式浮点控制寄存器组、用户模式事件计数器寄存器组、用户模式扩展浮点控制寄存器组和矢量扩展寄存器组。

16.3.1 用户模式浮点控制寄存器组

16.3.1.1 浮点异常累积状态寄存器 (FFLAGS)

浮点异常累积状态寄存器 (FFLAGS) 是浮点控制状态寄存器 (FCSR) 的异常累积域映射，具体信息请参考浮点控制状态寄存器 (*FCSR*)。

16.3.1.2 浮点动态舍入模式寄存器 (FRM)

浮点动态舍入模式寄存器 (FRM) 是浮点控制状态寄存器 (FCSR) 的舍入模式域映射，具体信息请参考浮点控制状态寄存器 (*FCSR*)。

16.3.1.3 浮点控制状态寄存器 (FCSR)

浮点控制状态寄存器 (FCSR) 用于记录浮点的异常累积和舍入模式控制。

该寄存器的位长是 64 位，该寄存器任何模式都可以读写。

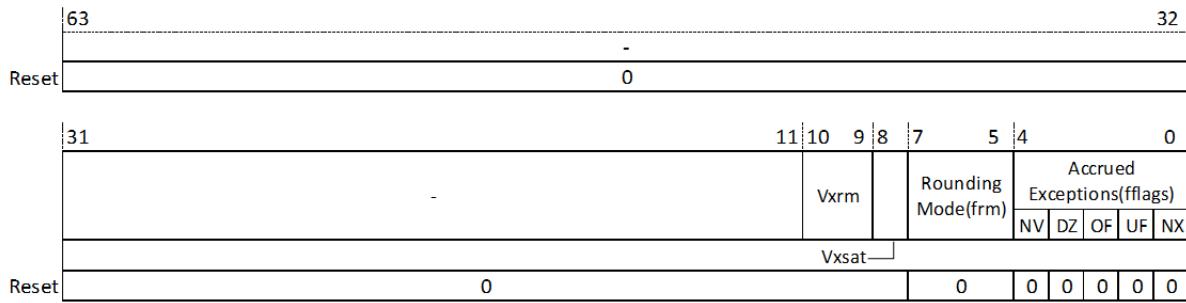


图 16.21: 浮点控制状态寄存器 (FCSR)

NX-非精确异常:

- 当 NX=0 时, 没有产生非精确异常;
- 当 NX=1 时, 产生过非精确异常。

该位会被 reset 置为 1' b0。

UF-下溢异常:

- 当 UF=0 时, 没有产生下溢异常;
- 当 UF=1 时, 产生过下溢异常。

该位会被 reset 置为 1' b0。

OF-上溢异常:

- 当 OF=0 时, 没有产生上溢异常;
- 当 OF=1 时, 产生过上溢异常。

该位会被 reset 置为 1' b0。

DZ-除 0 异常:

- 当 DZ=0 时, 没有产生除 0 异常;
- 当 DZ=1 时, 产生过除 0 异常。

该位会被 reset 置为 1' b0。

NV-无效操作数异常:

- 当 NV=0 时, 没有产生无效操作数异常;
- 当 NV=1 时, 产生过无效操作数异常。

该位会被 reset 置为 1' b0。

RM-舍入模式:

- 当 RM=0 时, RNE 舍入模式, 向最近偶数舍入;
- 当 RM=1 时, RTZ 舍入模式, 向 0 舍入;
- 当 RM=2 时, RDN 舍入模式, 向负无穷舍入;

- 当 RM=3 时, RUP 舍入模式, 向正无穷舍入;
- 当 RM=4 时, RMM 舍入模式, 向最近舍入。

该位会被 reset 置为 3' b0。

VXSAT-矢量溢出标志位:

VXSAT 最低位比特的映射。

该位会被 reset 置为 1' b0。

VXRM-矢量舍入模式位:

VXRM 最低两位比特的映射。

该位会被 reset 置为 2' b0。

16.3.2 用户模式事件计数器寄存器组

16.3.2.1 用户模式周期计数器 (CYCLE)

用户模式周期计数器 (CYCLE) 用于存储处理器已经执行的周期数, 当处理器处于执行状态 (即非低功耗状态) 下, CYCLE 寄存器就会在每个执行周期自增计数。

周期计数器为 64 位, 机器模式和超级用户模式可读写, 用户模式只读。周期计数器会被 reset 清零。

具体信息请参考事件计数器。

16.3.2.2 用户模式时间计数器 (TIME)

用户模式时间计数器 (TIME) 是 MTIME 的只读映射, 机器模式和超级用户模式可读写。

具体信息请参考事件计数器。

16.3.2.3 用户模式退休指令计数器 (INSTRET)

用户模式退休指令计数器 (MINSTRET) 用于存储处理器已经退休的指令数, INSTRET 寄存器会在每条指令退休时自增计数。

退休指令计数器为 64 位, 机器模式和超级用户模式可读写, 用户模式只读。退休指令计数器会被 reset 清零。

具体信息请参考事件计数器。

16.3.2.4 用户模式事件计数器 (HPMCOUNTERn)

用户模式事件计数器 (HPMCOUNTERn) 是机器模式事件计数器 (MHPMCOUNTERn) 的只读映射, 机器模式和超级用户模式可读写。

具体信息请参考事件计数器。

16.3.3 用户模式扩展浮点控制寄存器组

16.3.3.1 用户模式浮点扩展控制寄存器 (FXCR)

用户模式浮点扩展控制寄存器 (FXCR) 用于浮点扩展功能开关和浮点异常累积位。

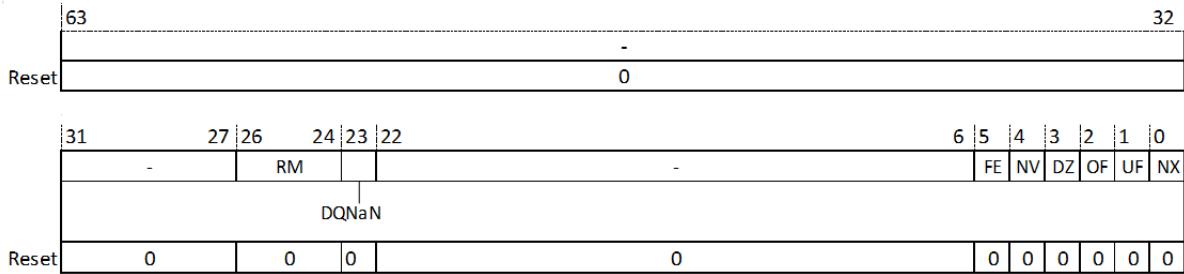


图 16.22: 浮点扩展控制寄存器 (FXCR)

NX-非精确异常:

FCSR 对应位的映射。

UF-下溢异常:

FCSR 对应位的映射。

OF-上溢异常:

FCSR 对应位的映射。

DZ-除 0 异常:

FCSR 对应位的映射。

NV-无效操作数异常:

FCSR 对应位的映射。

FE-浮点异常累积位:

当有任何一个浮点异常发生时, 该位将被置为 1。

该位会被 reset 置为 1' b0。

DQNaN-输出 QNaN 模式位:

当 DQNaN 为 0 时, 计算输出的 QNaN 值为默认的固定值, RISC-V 规定的固定值, 即符号位为 0, 指数全 1, 尾数最高位为 1, 其余位为零。

当 DQNaN 为 1 时, 计算输出的 QNaN 值根 IEEE754 标准一致。

该位会被 reset 置为 1' b0。

RM-舍入模式:

FCSR 对应位的映射。

16.3.4 矢量扩展寄存器组

16.3.4.1 矢量起始位置寄存器 (VSTART)

矢量起始位置寄存器指定了执行矢量指令时起始元素位置，每条矢量指令执行后 VSTART 会被清零。

16.3.4.2 定点溢出标志位寄存器 (VXSAT)

定点溢出标志位寄存器表示是否有定点指令产生溢出结果。

16.3.4.3 定点舍入模式寄存器 (VXRM)

定点舍入模式寄存器指定了定点指令采用的舍入模式。

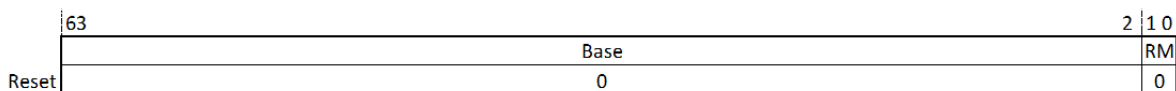


图 16.23: 定点舍入模式寄存器 (VXRM)

RM-定点舍入模式:

- 当 RM=0 时，RNU 舍入模式，向大数舍入；
- 当 RM=1 时，RNE 舍入模式，向偶数舍入；
- 当 RM=2 时，RDN 舍入模式，向零舍入；
- 当 RM=3 时，ROD 舍入模式，向奇数舍入。

16.3.4.4 矢量长度寄存器 (VL)

矢量长度寄存器指定了矢量指令更新目的寄存器的范围，矢量指令更新目的寄存器中元素序号小于 VL 的元素，清零目的寄存器中元素序号大于等于 VL 的元素。特别的，当 VSTART ≥ VL 或 VL 为 0 时，目的寄存器的所有元素不被更新。

该寄存器是任意模式下的只读寄存器，但是 vsetvli、vsetvl 以及 fault-only-first 指令能够更新该寄存器的值。

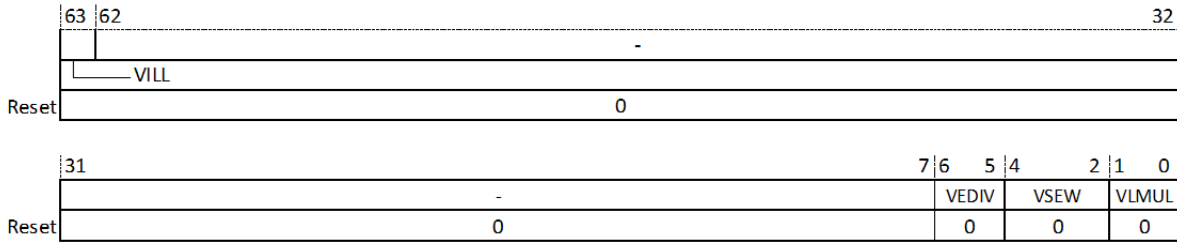


图 16.24: 定点类型寄存器 (VTYPE)

16.3.4.5 矢量数据类型寄存器 (VTYPE)

VTYPE 寄存器指定了矢量寄存器组的数据类型以及矢量寄存器的元素组成。

该寄存器是任意模式下的只读寄存器，但是通过 `vsetvli` 和 `vsetvl` 指令能够更新该寄存器的值。

VILL-非法操作标志位:

当 `vsetvli/vsetvl` 指令以 C906 不支持的值更新 VTYPE 寄存器时该位置起，否则为 0。当该位置起时，执行矢量指令产生非法指令异常。

VEDIV-EDIV 扩展使能位:

C906 不支持 EDIV 扩展，EDIV 位为 0。

VSEW-矢量元素位宽设置位:

VSEW 决定了矢量元素位宽 (SEW)，C906 支持的矢量元素位宽如表 16.2 所示。

表 16.2: 矢量元素位宽

VSEW[2:0]			元素位宽
0	0	0	8
0	0	1	16
0	1	0	32

当 VSEW 为其他值时，C906 执行矢量指令产生非法指令异常。

VLMUL-矢量寄存器分组设置位:

多个矢量寄存器可以构成矢量寄存器组，矢量指令作用于寄存器组中的所有矢量寄存器。VLMUL 决定了矢量寄存器组中矢量寄存器的数量 (LMUL)，如表 16.3 所示。

表 16.3: 矢量寄存器组中寄存器数量

VLMUL[1:0]		LMUL
0	0	1
0	1	2
1	0	4
1	1	8

16.3.4.6 矢量寄存器字节数寄存器 (VLENB)

该寄存器用于表示矢量寄存器的数据位宽，以实际位宽除以 8 得到的字节数体现。C906 矢量寄存器为 128 位，因此 VLENB 值固定为 16。

该寄存器位长是 64 位，用户模式只读。