
# 第4季

# RISC-V体系结构介绍

# 本节课主要内容

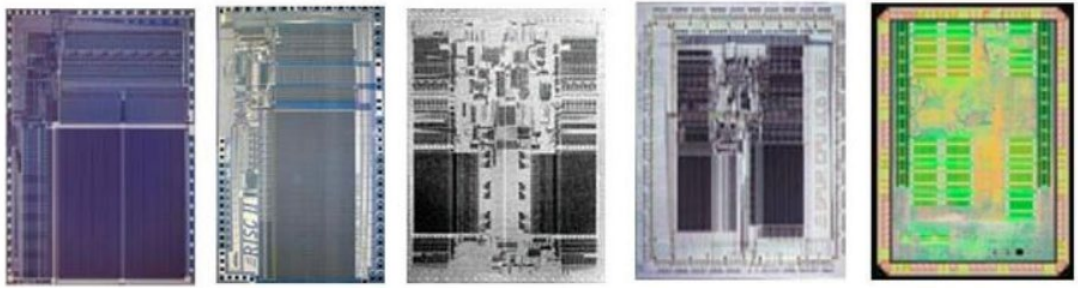
- 本章主要内容
  - RISC-V发展历史
  - RISC-V指令集介绍
  - RISC-V体系结构基础知识

技术手册：
1. The RISC-V Instruction Set Manual Volume I: Unprivileged ISA
2. The RISC-V Instruction Set Manual Volume II: Privileged Architecture


本节课主要讲解书上第1章内容

# RISC-V发展历史

- RISC-V是加州大学伯克利分校第5代RISC指令集
- RSIC：Reduced Instruction Set Computer，精简指令集
- 2010年，商用的复杂和臃肿而且授权费昂贵，因此重新设计一套
- 2015年，RISC-V基金会成立，维护指令集以及架构规范
- RSIC-V采用BSD开源协议
- 成立SiFive商业化公司，推动RISC-V商业化落地



RISC-I 1981　RISC-II 1983　RISC-III (SOAR) 1984　RTSC-TV (SPUR) 1988　RTSC-V 2013

RISC-V®

# RISC-V指令集的特点

- 开源和免费
- 设计简洁，继承了MIPS的优点和风格
- 模块化设计
- 丰富的软件生态

# RISC-V指令集扩展

- 最小指令集合：
  - ✓ RV32I：32位整型最小指令集合
  - ✓ RV64I：64位整型最小指令集合
- M：整型乘法和除法扩展指令
- G：表示IMAFD

实现完这些最基本的指令就可以运行一个小OS

表 1.1 RISC-V 扩展指令集

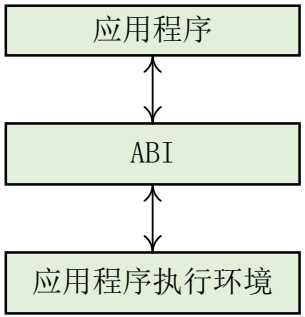
| 指令集扩展 | 说明 |
| --- | --- |
| F | 单精度浮点数扩展指令 |
| D | 双精度浮点数扩展指令 |
| Q | 四倍精度浮点数扩展指令 |
| M | 整型乘法和除法扩展指令 |
| C | 压缩指令 |
| A | 原子操作指令 |
| B | 位操作指令 |
| E | 表示为嵌入式设计的整型指令 |
| H | 虚拟化扩展 |
| K | 密码运算扩展 |
| V | 可伸缩矢量扩展 |
| P | 打包SIMD（Packed-SIMD）扩展 |
| J | 动态翻译语言（Dynamically Translated Languages）扩展 |
| T | 事务内存（ Transactional Memory） |
| N | 用户态中断 |

# RISC-V体系结构特点

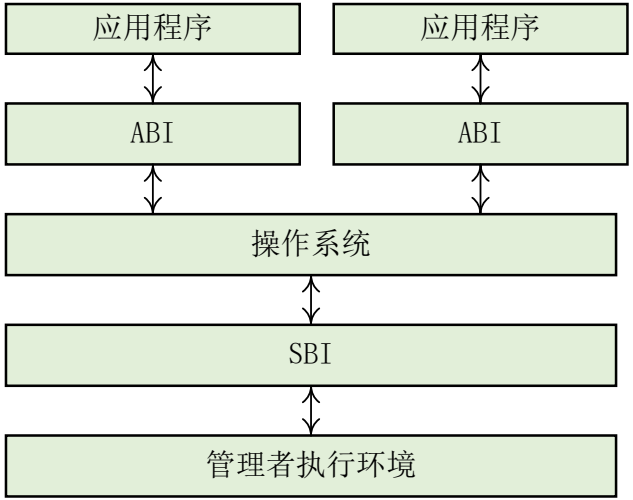
- 一个对学术界和工业界完全开放的指令集架构。
- 真正适合硬件实现的指令集架构，而不是一个模拟或者二进制翻译的指令集架构。
- RISC-V是一个通用的指令集架构，而不是针对某个特定微架构的实现。
- 实现最小集合的整数指令集作为最基础指令集，可以用于教学。在此基础上还实现众多可选扩展指令，以支持通用软件开发。
- 支持IEEE-754浮点标准。
- 支持众多扩展指令集。
- 支持32位及64位地址空间。
- 支持多核及异构架构。
- 支持可选的压缩指令编码，以提高性能、能源效率以及优化静态代码大小。
- 支持虚拟化扩展。
- 支持可伸缩矢量指令扩展（RISC-V V Vector Extension，RVV）。

# RISC-V体系结构中的基本概念

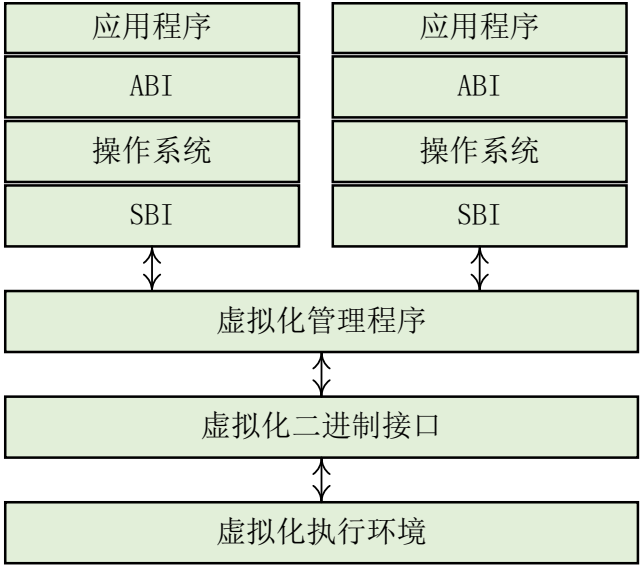
- 执行环境（Execution Environment Interface， EEI）




（a）裸机程序




（b）操作系统




（c）虚拟化

# 哈特（Hart）

- 在RISC-V手册里使用hart来表示一个CPU硬件执行单元。
- 类似x86架构定义的超线程（Simultaneous Multithreading，SMT）概念
- Armv8架构定义的处理机（Processing Element，PE）

# 处理器模式

- RISC-V处理器提供3种处理器模式。
  - 机器模式（M模式）：运行SBI固件，为操作系统提供服务。
  - 特权模式（S模式）：运行OS内核，为应用程序提供服务。
  - 用户模式（U模式）：运行应用程序。

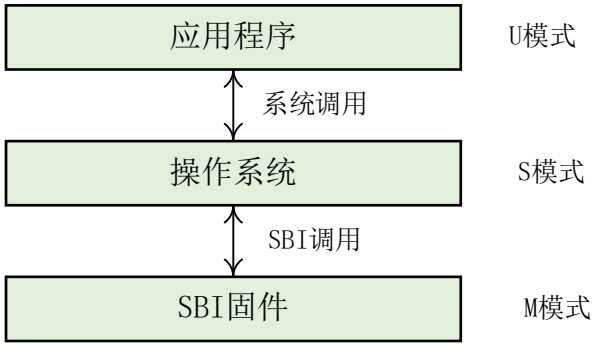
表 1.2 特权级别使用场景

| 特权级别个数 | 支持的特权级别 | 使用场景 |
| --- | --- | --- |
| 1 | M | 嵌入式系统 |
| 2 | M 和 U | 具有安全特性的嵌入式系统 |
| 3 | M、S 和 U | 通用操作系统 |

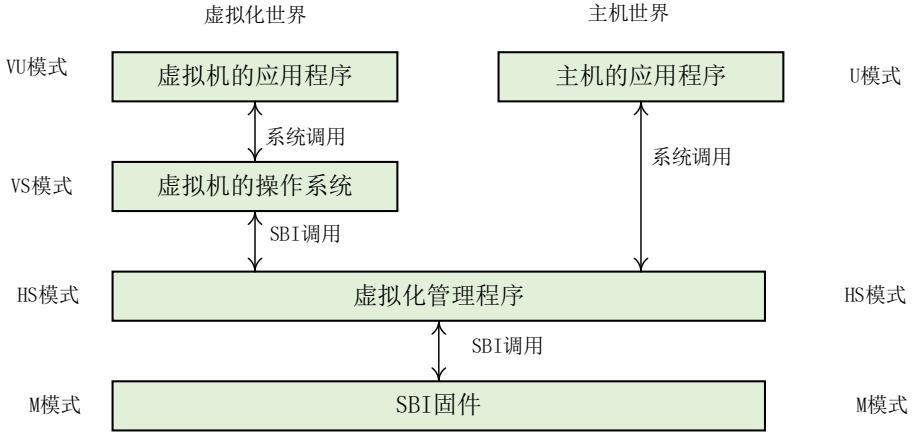
- 虚拟化扩展
  - 新增了HS模式，VS，VU模式

# SBI接口

- Supervisor Binary Interface：对所有RISC-V硬件平台中共性的功能做了抽象，为运行在S模式的操作系统或者HS模式的虚拟化管理软件提供统一的服务接口。
- SBI类似于操作系统中的系统调用层或者X86中的BIOS
- RISC-V有一个通用的SBI实现：OpenSBI
  - ✓ 为运行在低级别的处理器模式提供访问M模式硬件资源的抽象接口
  - ✓ 保证系统稳定和安全
  - ✓ 可移植性



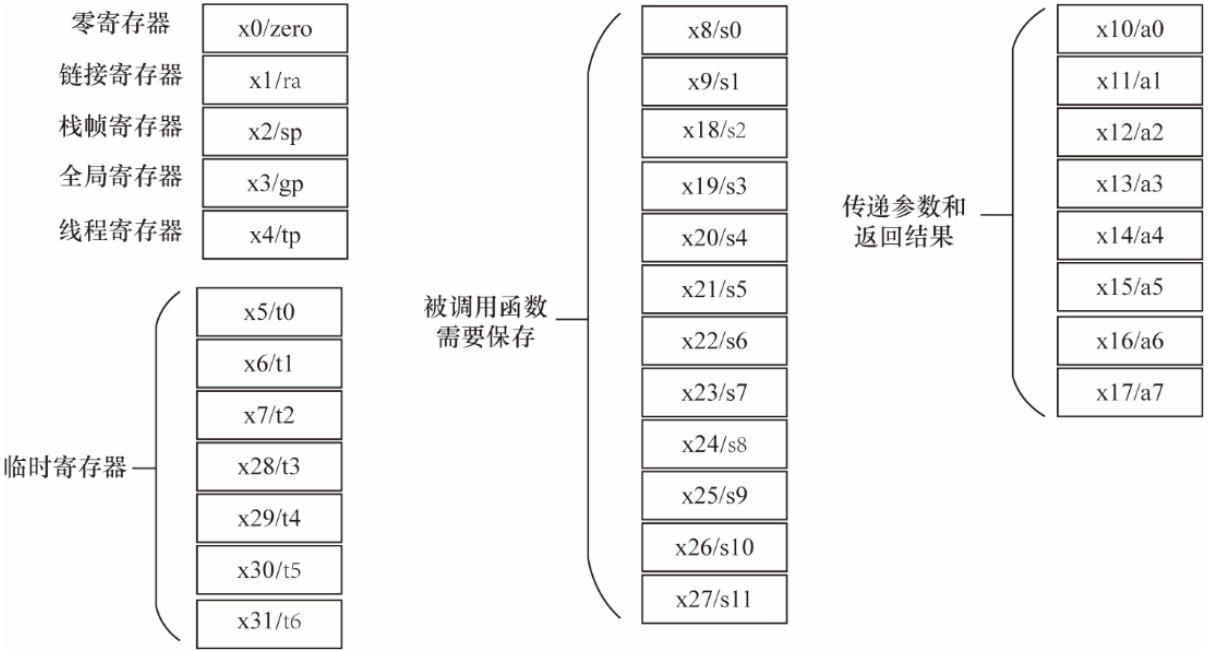

没使能虚拟化场景




使能虚拟化场景

# RISC-V通用寄存器

- 64位的RISC-V架构提供32个64位的整型通用寄存器：x0 ~ x31寄存器
- 对于浮点数运算，也提供32个浮点数通用寄存器：f0 ~ f31寄存器



# 通用寄存器别名

- RISC-V为通用寄存器具有别名（花名）和特殊用途

  - x0花名为zero。内容全是0，可以用作源寄存器，也可以用作目标寄存器。
  - x1花名为ra，链接寄存器，保存函数返回地址。
  - x2花名为sp，栈帧寄存器，指向栈的地址。
  - x3花名为gp，全局寄存器，用于松弛链接优化，见第6章。
  - x4花名为tp，线程寄存器，在OS里保存指向进程控块task_struct数据结构的指针。
  - x5~x7以及x28~x31用于临时寄存器，花名分别是t0 ~ t6。
  - x8~x9以及x19~x27的花名s0~s11。如果在函数调用过程中使用这些寄存器需要保存到栈里。
  - x10~x17的花名为a0~a7，在函数调用时传递参数和返回值。

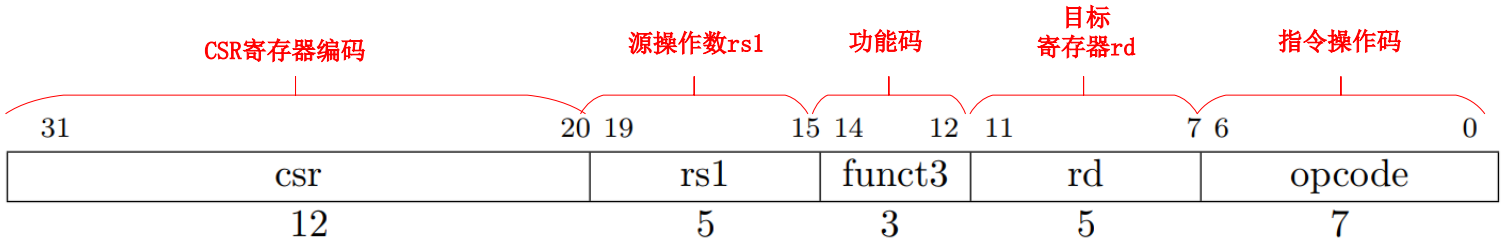
除用于数据运算和存储之外，通用寄存器还可以在函数调用过程中起到特殊作用

# RISC-V系统寄存器

- RISC-V体系结构定义系统控制和状态寄存器（Control and Status Registers，CSRs）

- RISC-V体系结构支持如下3类系统寄存器（暂不考虑虚拟化）：
  - ✓ M模式系统寄存器
  - ✓ S模式系统寄存器
  - ✓ U模式系统寄存器

- 软件通过CSR指令访问系统寄存器，例如CSRRW指令
- 在CSR指令编码中预留了12位编码空间（csr[11:0]）用来索引系统寄存器



- 12位CSR编码空间和分组
  - ✓ Bit[11:10]用来表示系统寄存器读写属性，0b11表示只读，其余表示可读可写。
  - ✓ Bit[9:8]表示允许访问该系统寄存器的处理器模式，0b00表示U模式，0b01表示S模式，0b11表示M模式，0b10表示HS/VS模式
  - ✓ 详见《Privileged Architecture》

表 1.3 CSR 地址空间映射

| 地址范围 | CSR 编码 | | | 访问模式 | 访问权限 |
|---|---|---|---|---|---|
| | Bit[11:10] | Bit[9:8] | Bit[7:4] | | |
| 0x000-0x0FF | 00 | 00 | XXXX | U | 读写 |
| 0x400-0x4FF | 01 | 00 | XXXX | U | 读写 |
| 0x800-0x8FF | 10 | 00 | XXXX | U | 读写（用户自定义系统寄存器） |
| 0xC00-0xC7F | 11 | 00 | 0XXX | U | 读写 |
| 0xC80-0xCBF | 11 | 00 | 10XX | U | 读写 |
| 0xCC0-0xCFF | 11 | 00 | 11XX | U | 读写 |
| 0x100-0x1FF | 00 | 01 | XXXX | S | 读写 |
| 0x500-0x57F | 01 | 01 | 0XXX | S | 读写 |
| 0x580-0x5BF | 01 | 01 | 10XX | S | 读写 |
| 0x5C0-0x5FF | 01 | 01 | 11XX | S | 读写（用户自定义系统寄存器） |
| 0x900-0x97F | 10 | 01 | 0XXX | S | 读写 |
| 0x980-0x9BF | 10 | 01 | 10XX | S | 读写 |
| 0x9C0-0x9FF | 10 | 01 | 11XX | S | 读写（用户自定义系统寄存器） |
| 0xD00-0xD7F | 11 | 01 | 0XXX | S | 只读 |
| 0xD80-0xDBF | 11 | 01 | 10XX | S | 只读 |
| 0xDC0-0xDFF | 11 | 01 | 11XX | S | 只读（用户自定义系统寄存器） |
| 0x300-0x3FF | 00 | 11 | XXXX | M | 读写 |
| 0x700-0x77F | 01 | 11 | 0XXX | M | 读写 |
| 0x780-0x79F | 01 | 11 | 100X | M | 读写 |
| 0x7A0-0x7AF | 01 | 11 | 1010 | M | 读写，用于调试寄存器 |
| 0x7B0-0x7BF | 01 | 11 | 1011 | M | 只能用于调试寄存器 |
| 0x7C0-0x7FF | 01 | 11 | 11XX | M | 读写（用户自定义系统寄存器） |
| 0xB00-0xB7F | 10 | 11 | 0XXX | M | 读写 |
| 0xB80-0xBBF | 10 | 11 | 10XX | M | 读写 |
| 0xBC0-0xBFF | 10 | 11 | 11XX | M | 读写（用户自定义系统寄存器） |
| 0xF00-0xF7F | 11 | 11 | 0XXX | M | 只读 |

触发非法指令异常。

- ✓ 访问不存在或者没有实现的系统寄存器。
- ✓ 尝试写如只具有只读属性的系统寄存器。
- ✓ 在低级别的处理器模式下访问高级别的处理器模式的系统寄存器

# U模式下的CSRs

表 1.4 U 模式系统寄存器

| 地址 | CSR 名称 | 属性 | 说明 |
|---|---|---|---|
| 0x001 | fflags | URW | 浮点数累积异常（Accrued Exceptions） |
| 0x002 | frm | URW | 浮点数动态舍入模式(Dynamic Rounding Mode) |
| 0x003 | fcsr | URW | 浮点数控制和状态寄存器 |
| 0xC00 | cycle | URO | 读取时钟周期（Cycle counter），映射到 RDCYCLE 伪指令 |
| 0xC01 | time | URO | 读取 time 计数，映射到 RDTIMER 伪指令 |
| 0xC02 | instret | URO | 执行指令数目，映射到 RDINSTRET 伪指令 |
| 0xC03 ~ 0xC1F | hpmcounter3 ~ hpmcounter31 | URO | 性能监测寄存器 |

# S模式下的CSRs

表 1.5 S 模式系统寄存器

| 地址 | CSR 名称 | 属性 | 说明 |
|---|---|---|---|
| 0x100 | sstatus | SRW | S 模式的处理器状态寄存器 |
| 0x104 | sie | SRW | S 模式的中断使能寄存器 |
| 0x105 | stvec | SRW | S 模式的异常向量入口地址寄存器 |
| 0x106 | scounteren | SRW | S 模式的计数使能寄存器 |
| 0x10A | senvcfg | SRW | S 模式的环境配置寄存器 |
| 0x140 | sscratch | SRW | 用于异常处理的临时寄存器 |
| 0x141 | sepc | SRW | S 模式异常模式程序计数器（PC）寄存器 |
| 0x142 | scause | SRW | S 模式的异常原因寄存器 |
| 0x143 | stval | SRW | S 模式的异常向量寄存器 |
| 0x144 | sip | SRW | S 模式中断待定寄存器 |
| 0x180 | satp | SRW | S 模式的地址转换与保护寄存器 |
| 0x5A8 | scontext | SRW | S 模式的上下文寄存器（用于调试） |

- S模式下的处理器状态
- 异常和中断相关的寄存器
- MMU处理相关寄存器
- 性能相关的寄存器。
- 其他寄存器

# sstatus寄存器：表示处理器状态

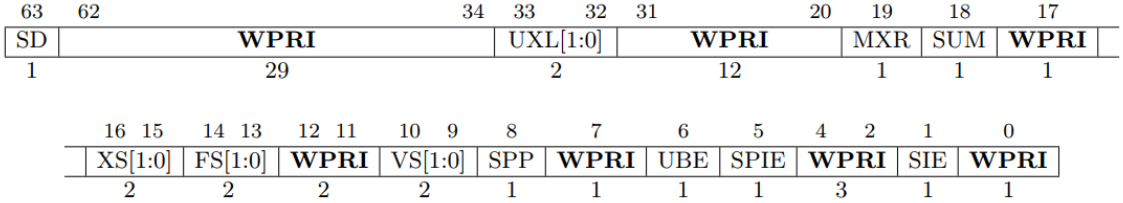



表 1.6 sstatus 寄存器

| 字段 | 位段 | 说明 |
|---|---|---|
| SIE | Bit[1] | 中断使能位，用来使能和关闭 S 模式中所有的中断。 |
| SPIE | Bit[5] | 中断使能保存位。当一个异常陷入到 S 模式，SIE 的值保存到 SPIE 中， SIE 设置为 0。当调用 SRET 指令返回时，从 SPIE 中恢复 SIE，然后 SPIE 设置为 1。 |
| UBE | Bit[6] | 用来控制 U 模式加载和存储内存访问的大小端模式。<br>0：小端模式<br>1：大端模式 |
| SPP | Bit[8] | 陷入到 S 模式之前 CPU 的处理模式。<br>0：表示从 U 模式陷入到 S 模式<br>1：表示在 S 模式触发的异常 |
| VS | Bit[10:9] | 用来使能可伸缩矢量扩展（RVV） |
| FS | Bit[14:13] | 用来使能浮点数单元 |
| XS | Bit[16:15] | 用来使能其他 U 模式扩展的状态 |
| SUM | Bit[18] | S 模式下能否允许访问 U 模式的内存。<br>0：S 模式访问 U 模式的内存时会触发异常。<br>1：S 模式可以访问 U 模式的内存 |
| MXR | Bit[19] | 访问内存的权限<br>0：可以加载只读页面<br>1：可以加载可读和可执行的页面 |
| UXL | Bit[33:32] | 用来表示 U 模式的寄存器长度，通常是一个只读字段，并且 U 模式的寄存器长度等于 S 模式下的寄存器长度。 |
| SD | Bit[63] | 用来表示 VS、FS 以及 XS 中任意一个字段已经设置。 |

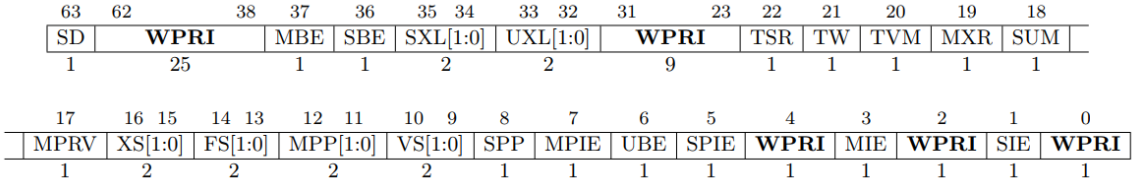
- sie寄存器用来使能和关闭S模式下的中断
- stvec寄存器用来在S模式下配置异常向量表入口地址和异常访问模式
- scounteren寄存器是一个32位寄存器，用来使能U模式下的硬件性能监测和和计数寄存器
- sscratch寄存器是一个专门给S模式使用的临时寄存器，它主要用途是当处理器运行在U模式时，它用来保存S模式下进程控制块的指针
- sepc寄存器：当处理器陷入到S模式时，被中断或遇到异常的指令的虚拟地址会写入到sepc寄存器中
- Scause寄存器：S模式下的异常原因。
- Stval寄存器：当处理器陷入到S模式时，stval寄存器记录了发生异常的虚拟地址
- sip寄存器用来表示哪些中断处于待定（pending）状态
- satp寄存器用于地址转换和保护

# M模式下的CSRs

表 1.7 M 模式系统寄存器

| 地址 | CSR 名称 | 属性 | 说明 |
|---|---|---|---|
| 0xF11 | mvendorid | MRO | 机器厂商 ID 寄存器 |
| 0xF12 | marchid | MRO | 架构编号寄存器 |
| 0xF13 | mimpid | MRO | 实现编号寄存器 |
| 0xF14 | mhartid | MRO | 处理器硬件线程编号寄存器 |
| 0xF15 | mconfigptr | MRO | 配置数据结构寄存器 |
| 0x300 | mstatus | MRW | M 模式处理器状态寄存器 |
| 0x301 | misa | MRW | 指令集架构和扩展寄存器 |
| 0x302 | medeleg | MRW | M 模式异常委托寄存器 |
| 0x303 | mideleg | MRW | M 模式中断委托寄存器 |
| 0x304 | mie | MRW | M 模式中断使能寄存器 |
| 0x305 | mtvec | MRW | M 模式的异常向量入口地址寄存器 |
| 0x306 | mcounteren | MRW | M 模式的计数使能寄存器 |
| 0x340 | mscratch | MRW | 用于异常处理的临时寄存器 |
| 0x341 | mepc | MRW | M 模式异常模式程序计数器（PC）寄存器 |
| 0x342 | mcause | MRW | M 模式的异常原因寄存器 |
| 0x343 | mtval | MRW | M 模式的异常向量寄存器 |
| 0x344 | mip | MRW | M 模式中断待定寄存器 |
| 0x34A | mtinst | MRW | M 模式陷入指令（用于虚拟化） |
| 0x34B | mtval2 | MRW | M 模式的异常向量寄存器（用于虚拟化） |

# mstatus寄存器：表示处理器状态



| 63 | 62–38 | 37 | 36 | 35–34 | 33–32 | 31–23 | 22 | 21 | 20 | 19 | 18 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| SD | WPRI | MBE | SBE | SXL[1:0] | UXL[1:0] | WPRI | TSR | TW | TVM | MXR | SUM |
| 1 | 25 | 1 | 1 | 2 | 2 | 9 | 1 | 1 | 1 | 1 | 1 |

| 17 | 16–15 | 14–13 | 12–11 | 10–9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| MPRV | XS[1:0] | FS[1:0] | MPP[1:0] | VS[1:0] | SPP | MPIE | UBE | SPIE | WPRI | MIE | WPRI | SIE | WPRI |
| 1 | 2 | 2 | 2 | 2 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

表 1.9 mstatus 寄存器

| 字段 | 位段 | 说明 |
| --- | --- | --- |
| SIE | Bit[1] | 中断使能位，用来使能和关闭 S 模式中所有的中断 |
| MIE | Bit[3] | 中断使能位，用来使能和关闭 M 模式中所有的中断 |
| SPIE | Bit[5] | 中断使能保存位。当一个异常陷入到 S 模式，SIE 的值保存到 SPIE 中， SIE 设置为 0。当调用 SRET 指令返回时，从 SPIE 中恢复 SIE，然后 SPIE 设置为 1。 |
| UBE | Bit[6] | 用来控制 U 模式加载和存储内存访问的大小端模式。<br>0：小端模式<br>1：大端模式 |
| MPIE | Bit[7] | 中断使能保存位。当一个异常陷入到 M 模式，MIE 的值保存到 MPIE 中，MIE 设置为 0。当调用 MRET 指令返回时，从 MPIE 中恢复 MIE，然后 MPIE 设置为 1。 |
| SPP | Bit[8] | 陷入到 S 模式之前 CPU 的处理模式。<br>0：表示从 U 模式陷入到 S 模式<br>1：表示在 S 模式触发的异常 |
| VS | Bit[10:9] | 用来使能可伸缩矢量扩展（RVV） |
| MPP | Bit[12:11] | 陷入到 M 模式之前 CPU 的处理模式。<br>0：表示从 U 模式陷入到 M 模式<br>1：表示从 S 模式陷入到 M 模式<br>2：表示在 M 模式触发的异常 |
| FS | Bit[14:13] | 用来使能浮点数单元 |
| XS | Bit[16:15] | 用来使能其他 U 模式扩展的状态 |

| | | |
|---|---|---|
| MPRV | Bit[17] | 用来修改有效特权模式。<br>0：加载和存储指令按照当前的处理器模式进行地址转换与内存保护。<br>1：加载和存储指令将按照 MPP 字段中存储的处理器模式的权限进行内存保护检查。 |
| SUM | Bit[18] | S 模式下能否允许访问 U 模式的内存。<br>0：S 模式访问 U 模式的内存时会触发异常。<br>1：S 模式可以访问 U 模式的内存 |
| MXR | Bit[19] | 访问内存的权限<br>0：可以加载只读页面<br>1：可以加载可读和可执行的页面 |
| TVM | Bit[20] | 支持拦截 S 模式的虚拟内存管理操作。<br>0：在 S 模式可以正常访问 satp 系统寄存器或者执行 SFENCE.VMA/ SINVAL.VMA 指令。<br>1：在 S 模式访问 satp 系统寄存器或者执行 SFENCE.VMA/ SINVAL.VMA 指令会触发一个非法指令异常。 |
| TW | Bit[21] | 支持拦截 WFI 指令。<br>0：WFI 指令可以在低权限模式下执行<br>1：如果 WFI 以任何低特权模式执行，并且它没有在特定实现中约定的有限时间内完成，WFI 指令会触发一个非法指令异常。 |
| TSR | Bit[22] | 支持拦截 SRET 指令。<br>0：在 S 模式正常执行 SRET 指令<br>1：在 S 模式执行 SRET 指令会触发一个非法指令异常 |
| UXL | Bit[33:32] | 用来表示 U 模式的寄存器长度 |
| SXL | Bit[35:34] | 用来表示 S 模式的寄存器长度 |
| SBE | Bit[36] | 用来控制 S 模式加载和存储内存访问的大小端模式。<br>0：小端模式<br>1：大端模式 |
| MBE | Bit[37] | 用来控制 M 模式加载和存储内存访问的大小端模式。<br>0：小端模式<br>1：大端模式 |
| SD | Bit[63] | 用来表示 VS、FS 以及 XS 中任意一个字段已经设置 |

# Thanks