目录

图目录

表目录

“系统能力培养大赛”MIPS 指令系统规范

“系统能力培养大赛”MIPS 指令系统是在 MIPS32 指令系统的基础之上进行一定程度地裁剪，在控制系统设计规模的前提下，保证简单系统的可实现性。概要来说，这套指令系统包含所有非浮点 MIPS I 指令和 MIPS32 中的 ERET 指令，少量的 CP0 寄存器以支持中断和系统调用，不实现 TLB MMU 和特权等级。

本文档对竞赛所用指令系统进行具体规定。如果觉得本文档中有定义不精确之处，可以查阅参考文献[1-3]中的相关章节；如两者存在冲突，以参考文献[1-3]中的内容为准。

本文档包含如下章节：

第 1 章，“编程模型”，对支持的数据类型、软件可见寄存器、大小尾端进行定义。

第 2 章，“操作模式”，对处理器需要支持的操作模式进行定义。

第 3 章，“指令定义”，对需实现指令逐条定义。

第 4 章，“存储管理”，定义一套线性虚实地址映射机制。

第 5 章，“中断与例外”，介绍需实现的中断和例外的相关定义。

第 6 章，“系统控制寄存器”，对需实现的系统控制寄存器（俗称 CP0 寄存器）逐个进行定义。

本文档供大家在本学期实验开展过程中按需索引。

每个章节对阅读者有不同的要求：

(1) 阅读第 1 章前，需要读者了解基本的 MIPS 汇编和一些计算机系统的知识。

(2) 阅读第 2 章前，需要读者理解用户态、核心态。

(3) 阅读第 3 章前，需要读者明白机器指令和汇编指令的区别，了解基本 MIPS 汇编。

(4) 阅读第 4 章前，需要读者理解虚实地址翻译的缘由和实现，了解软硬件协同。

(5) 阅读第 5 章前，需要读者了解中断和例外的区别及用途。

(6) 阅读第 6 章前，需要读者了解 CPU 操作模式的控制和切换，以及系统控制器寄存器的作用。

通过本章节的学习，你将获得：

(1) 认识一个简单但比较全面的 MIPS 系统。

(2) 明确 MIPS 基础指令集的定义。

(3) 了解 MIPS 基本存储管理——固定地址映射。

(4) 理解 MIPS 系统部分中断和例外的定义和处理。

(5) 了解 MIPS 基本的系统控制寄存器。

1 编程模型

1.1 数据格式

处理器可处理的数据格式定义如下：

- 比特（bit, b）
- 字节（Byte, 8bits, B）
- 半字（Halfword, 16bits, H）
- 字（Word, 32bits, W）

1.2 寄存器

处理器包含的软件可见的寄存器种类如下：

- 32 个 32 位通用寄存器，r0~r31。其中有两个被赋予了特殊含义：r0，0 号通用寄存器，值永远为 0；r31，31 号通用寄存器，被 JAL，BLTZAL 和 BGEZAL 指令隐式的用作目标寄存器，存放返回地址。
- HI/LO 寄存器。HI 寄存器存放乘法指令结果的高半部分或是除法指令结果的余数，LO 寄存器存放乘法指令结果的低半部分或是除法指令结果的商。
- 程序计数器（PC）。这个寄存器软件无法直接访问。
- 控制寄存器（CP0）。一组用于中断、例外控制的寄存器。

1.3 大小尾端

处理器中的字节顺序和比特顺序均采用小尾端模式，即第 0 字节永远是最低有效字节，第 0 比特永远是最低有效比特。

1.4 存储访问类型

处理器至少支持不可缓存（uncached）这种存储访问类型。属于此类型的访问将直接读、写物理内存（或物理内存地址映射的寄存器），但不会访问或修改各级缓存中的内容。

2 操作模式

处理器永远处于核心态模式（Kernel Mode）。该操作模式下处理器可以执行所有指令，访问 32 位全地址空间。

3 指令定义

3.1 指令格式

所有指令长度均为 32 比特。

除个别指令外，所有指令的格式均为立即数型（I-Type）、跳转型（J-Type）和寄存器型（R-Type）三种类型中的一种。三类指令格式如图 3-1 所示。

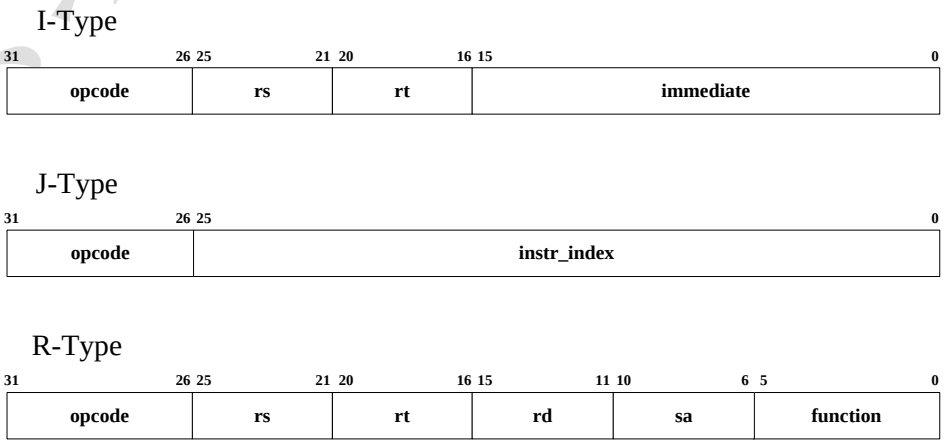


图 3-1 指令格式

3.2 指令功能分类

处理器需要实现的指令包括除 4 条非对齐指令外的所有 MIPS I 指令以及 MIPS32 中的 ERET 指令，有 14 条算术运算指令、8 条逻辑运算指令，6 条移位指令、8 条分支跳转指令、4 条数据移动指令、2 条自陷指令、12 条访存指令、3 条特权指令，共计 57 条。下面分类给出各部分指令的简要功能介绍。

表 3-1 算术运算指令

指令名称格式	指令功能简述
ADD rd, rs, rt	加（可产生溢出例外）
ADDI rt, rs, immediate	加立即数（可产生溢出例外）
ADDU rd, rs, rt	加（不产生溢出例外）
ADDIU rt, rs, immeidate	加立即数（不产生溢出例外）
SUB rd. rs, rt	减（可产生溢出例外）
SUBU rd, rs, rt	减（不产生溢出例外）
SLT rd, rs, rt	有符号小于置 1
SLTI rt, rs, immediate	有符号小于立即数设置 1
SLTU rd, rs, rt	无符号小于设置 1
SLTIU rt, rs, immediate	无符号小于立即数[1]设置 1
DIV rs, rt	有符号字除
DIVU rs,rt	无符号字除
MULT rs, rt	有符号字乘
MULTU rs, rt	无符号字乘

表 3-2 逻辑运算指令

指令名称格式	指令功能简述
AND rd, rs, rt	位与
ANDI rt, rs, immediate	立即数位与
LUI rt,immediate	寄存器高半部分置立即数
NOR rd, rs, rt	位或非
OR rd, rs, rt	位或
ORI rt, rs, immediate	立即数位或

[1] 请注意虽然是无符号比较，但是立即数仍是进行有符号扩展。

指令名称格式	指令功能简述
XOR rd, rs, rt	位异或
XORI rt, rs, immediate	立即数位异或

表 3-3 移位指令

指令名称格式	指令功能简述
SLL rd, rt, sa	立即数逻辑左移
SLLV rd, rs, rt	变量逻辑左移
SRA rd, rt, sa	立即数算术右移
SRAV rd, rs, rt	变量算术右移
SRL rd, rt, sa	立即数逻辑右移
SRLV rd, rs, rt	变量逻辑右移

表 3-4 分支跳转指令

指令名称格式	指令功能简述
BEQ rs, rt, offset	相等转移
BNE rs, rt, offset	不等转移
BGEZ rs, offset	大于等于 0 转移
BGTZ rs, offset	大于 0 转移
BLEZ rs, offset	小于等于 0 转移
BLTZ rs, offset	小于 0 转移
BLTZAL rs, offset	小于 0 调用子程序并保存返回地址
BGEZAL rs, offset	大于等于 0 调用子程序并保存返回地址
J target	无条件直接跳转
JAL target	无条件直接跳转至子程序并保存返回地址
JR rs	无条件寄存器跳转
JALR rd, rs	无条件寄存器跳转至子程序并保存返回地址下

表 3-5 数据移动指令

指令名称格式	指令功能简述
MFHI rd	HI 寄存器至通用寄存器
MFLO rd	LO 寄存器至通用寄存器
MTHI rs	通用寄存器至 HI 寄存器
MTLO rs	通用寄存器至 LO 寄存器

表 3-6 自陷指令

指令名称格式	指令功能简述
BREAK	断点
SYSCALL	系统调用

表 3-7 访存指令

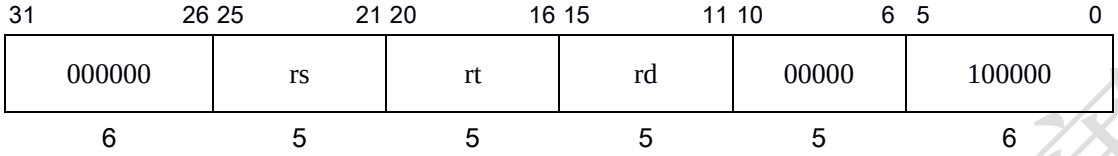
指令名称格式	指令功能简述
LB rt, offset(base)	取字节有符号扩展
LBU rt, offset(base)	取字节无符号扩展
LH rt, offset(base)	取半字有符号扩展
LHU rt, offset(base)	取半字无符号扩展
LW rt, offset(base)	取字
SB rt, offset(base)	存字节
SH rt, offset(base)	存半字
SW rt, offset(base)	存字

表 3-8 自陷指令

指令名称格式	指令功能简述
ERET	例外处理返回
MFC0	读 CP0 寄存器值至通用寄存器
MTC0	通用寄存器值写入 CP0 寄存器

3.3 算术运算指令

3.3.1 ADD



汇编格式：ADD rd, rs, rt

功能描述：将寄存器 rs 的值与寄存器 rt 的值相加，结果写入寄存器 rd 中。如果产生溢出，则触发整型溢出例外（IntegerOverflow）。

操作定义：tmp ← (GPR[rs]$_{31}$||GPR[rs]$_{31..0}$) + (GPR[rt]$_{31}$||GPR[rt]$_{31..0}$)

if tmp$_{32}$≠tmp$_{31}$ then

SignalException(IntegerOverflow)

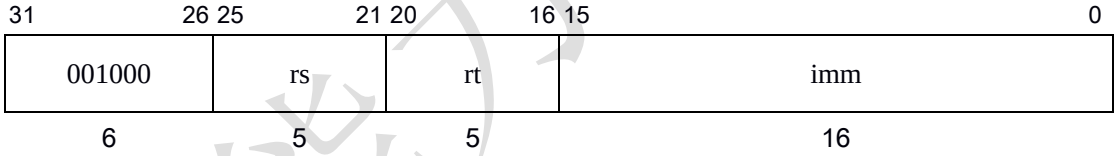
else

GPR[rd]←tmp$_{31..0}$

endif

例　外：如果有溢出，则触发整型溢出例外。

3.3.2 ADDI



汇编格式：ADDI rt, rs, imm

功能描述：将寄存器 rs 的值与有符号扩展至 32 位的立即数 imm 相加，结果写入 rt 寄存器中。如果产生溢出，则触发整型溢出例外（IntegerOverflow）。

操作定义：tmp ← (GPR[rs]$_{31}$||GPR[rs]$_{31..0}$) + sign_extend(imm)

if tmp$_{32}$≠tmp$_{31}$ then

SignalException(IntegerOverflow)

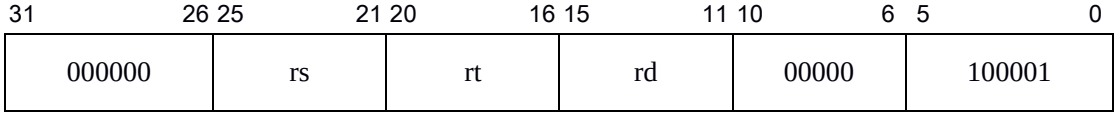
else

GPR[rt]←tmp$_{31..0}$

endif

例　外：如果有溢出，则触发整型溢出例外。

3.3.3 ADDU



6	5	5	5	5	6

汇编格式：ADDU rd, rs, rt

功能描述：将寄存器 rs 的值与寄存器 rt 的值相加，结果写入 rd 寄存器中。

操作定义：GPR[rd] ← GPR[rs] + GPR[rt]

例　外：无

3.3.4 ADDIU

31 26	25 21	20 16	15 0
001001	rs	rt	imm
6	5	5	16

汇编格式：ADDIU rt, rs, imm

功能描述：将寄存器 rs 的值与**有符号扩展**至 32 位的立即数 imm 相加，结果写入 rt 寄存器中。

操作定义：GPR[rt] ← GPR[rs] + sign_extend(imm)

例　外：无

3.3.5 SUB

31 26	25 21	20 16	15 11	10 6	5 0
000000	rs	rt	rd	00000	100010
6	5	5	5	5	6

汇编格式：SUB rd, rs, rt

功能描述：将寄存器 rs 的值与寄存器 rt 的值相减，结果写入 rd 寄存器中。如果产生溢出，则触发整型溢出例外（IntegerOverflow）。

操作定义：tmp ← (GPR[rs]$_{31}$||GPR[rs]$_{31..0}$) - (GPR[rt]$_{31}$||GPR[rt]$_{31..0}$)

```
if tmp32≠tmp31 then
    SignalException(IntegerOverflow)
else
    GPR[rd]←tmp31..0
endif
```

例　外：如果有溢出，则触发整型溢出例外。

3.3.6 SUBU

31 26	25 21	20 16	15 11	10 6	5 0
000000	rs	rt	rd	00000	100011
6	5	5	5	5	6

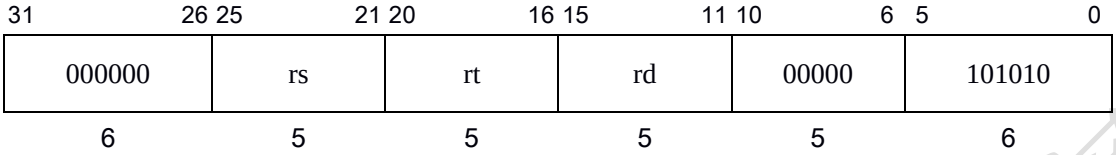
汇编格式：SUBU rd, rs, rt

功能描述：将寄存器 rs 的值与寄存器 rt 的值相减，结果写入 rd 寄存器中。

操作定义：GPR[rd] ← GPR[rs] – GPR[rt]

例　外：无

3.3.7 SLT



汇编格式：SLT rd, rt, rs

功能描述：将寄存器 rs 的值与寄存器 rt 中的值进行有符号数比较，如果寄存器 rs 中的值小，则寄存器 rd 置 1；否则寄存器 rd 置 0。

操作定义：if GPR[rs] < GPR[rt] then

　　GPR[rd] ← 1

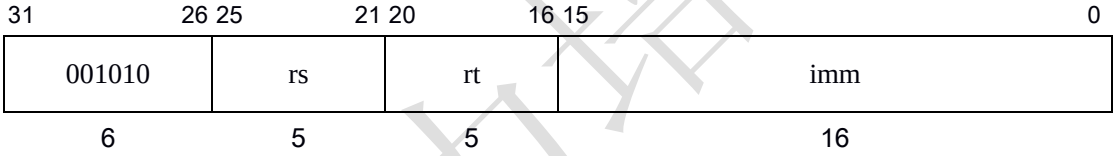
else

　　GPR[rd] ← 0

endif

例　外：无

3.3.8 SLTI



汇编格式：SLTI rt, rs, imm

功能描述：将寄存器 rs 的值与有符号扩展至 32 位的立即数 imm 进行有符号数比较，如果寄存器 rs 中的值小，则寄存器 rt 置 1；否则寄存器 rt 置 0。

操作定义：if GPR[rs] < Sign_extend(imm) then

　　GPR[rt] ← 1

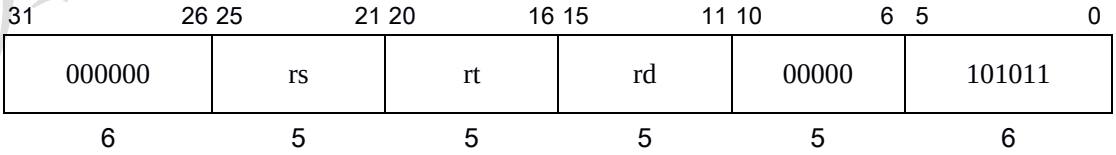
else

　　GPR[rt] ← 0

endif

例　外：无

3.3.9 SLTU



汇编格式：SLTU rd, rs, rt

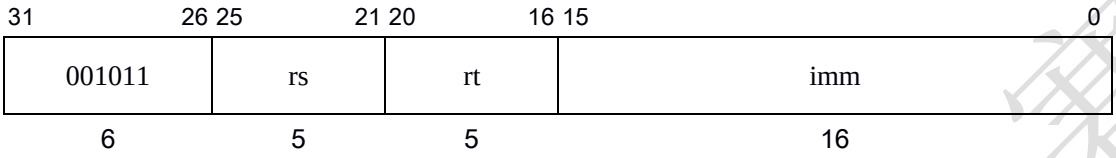
功能描述：将寄存器 rs 的值与寄存器 rt 中的值进行无符号数比较，如果寄存器 rs 中的值小，则寄存器 rd 置 1；否则寄存器 rd 置 0。

操作定义：if $(0||GPR[rs]_{31..0}) < (0||GPR[rt]_{31..0})$ then

GPR[rd] ← 1
else
GPR[rd] ← 0
endif

例　外：无

3.3.10 SLTIU



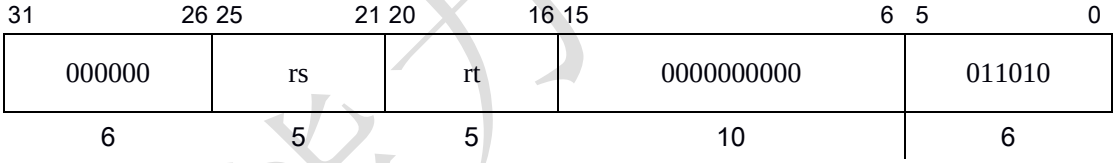
汇编格式：SLTIU rt, rs, imm

功能描述：将寄存器 rs 的值与**有符号扩展**至 32 位的立即数 imm 进行无符号数比较，如果寄存器 rs 中的值小，则寄存器 rt 置 1；否则寄存器 rt 置 0。

操作定义：if (0||GPR[rs]$_{31..0}$) < Sign_extend(imm) then
GPR[rt] ← 1
else
GPR[rt] ← 0
endif

例　外：无

3.3.11 DIV



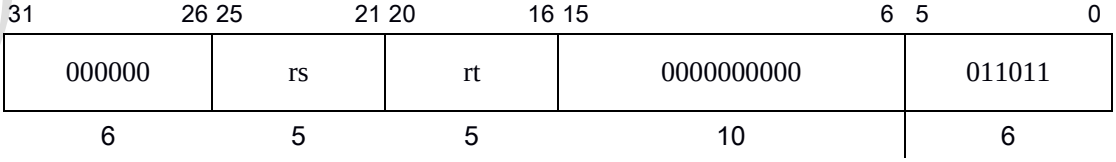
汇编格式：DIV rs, rt

功能描述：有符号除法，寄存器 rs 的值除以寄存器 rt 的值，商写入 LO 寄存器中，余数写入 HI 寄存器中。

操作定义：q ← GPR[rs]$_{31..0}$ div GPR[rt]$_{31..0}$
LO ← q
r ← GPR[rs]$_{31..0}$ mod GPR[rt]$_{31..0}$
HI ← r

例　外：无

3.3.12 DIVU



汇编格式：DIVU rs, rt

功能描述：无符号除法，寄存器 rs 的值除以寄存器 rt 的值，商写入 LO 寄存器中，余数写入 HI 寄存器中。

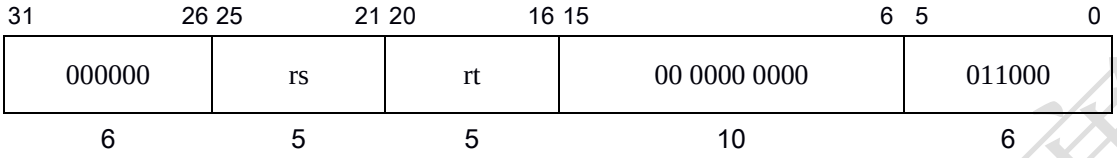
操作定义：q ← (0||GPR[rs]$_{31..0}$) div (0||GPR[rt]$_{31..0}$)
LO ← q

r ← (0||GPR[rs]$_{31..0}$) mod (0||GPR[rt]$_{31..0}$)

HI ← r

例　外：无

3.3.13 MULT



汇编格式：MULT rs, rt

功能描述：有符号乘法，寄存器 rs 的值乘以寄存器 rt 的值，乘积的低半部分和高半部分分别写入 LO 寄存器和 HI 寄存器。

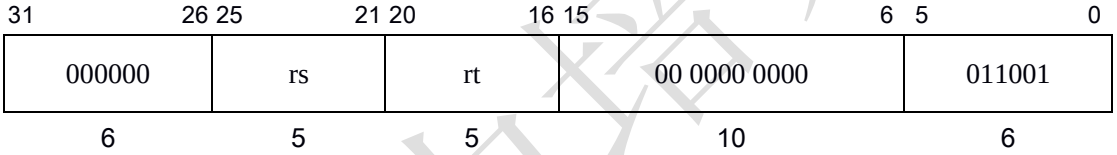
操作定义：prod ← GPR[rs]$_{31..0}$ × GPR[rt]$_{31..0}$

LO ← prod$_{31..0}$

HI ← prod$_{63..32}$

例　外：无

3.3.14 MULTU



汇编格式：MULTU rs, rt

功能描述：无符号乘法，寄存器 rs 的值乘以寄存器 rt 的值，乘积的低半部分和高半部分分别写入 LO 寄存器和 HI 寄存器。

操作定义：prod ← (0||GPR[rs]$_{31..0}$) × (0||GPR[rt]$_{31..0}$)

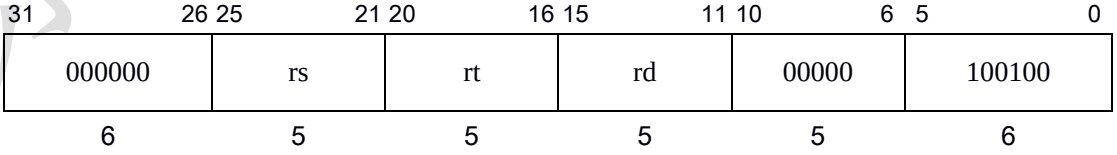
LO ← prod$_{31..0}$

HI ← prod$_{63..32}$

例　外：无

3.4 逻辑运算指令

3.4.1 AND



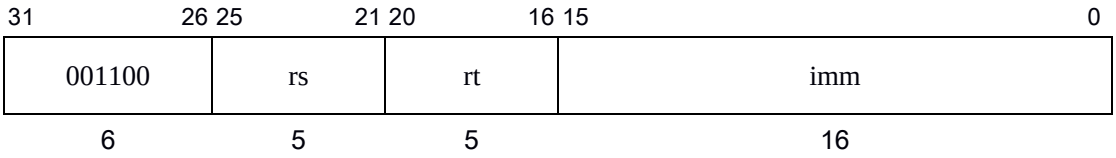
汇编格式：AND rd, rs, rt

功能描述：寄存器 rs 中的值与寄存器 rt 中的值按位逻辑与，结果写入寄存器 rd 中。

操作定义：GPR[rd] ← GPR[rs] & GPR[rt]

例　外：无

3.4.2 ANDI



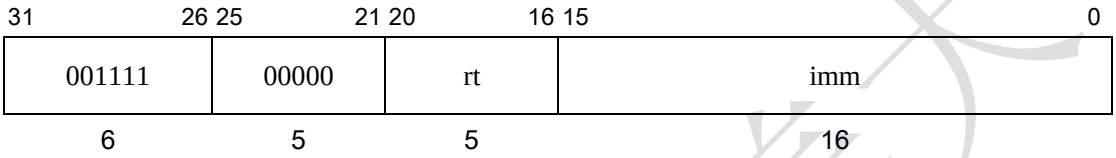
汇编格式：ANDI rt, rs, imm

功能描述：寄存器 rs 中的值与 0 扩展至 32 位的立即数 imm 按位逻辑与，结果写入寄存器 rt 中。

操作定义：GPR[rt] ← GPR[rs] and Zero_extend(imm)

例　外：无

3.4.3 LUI



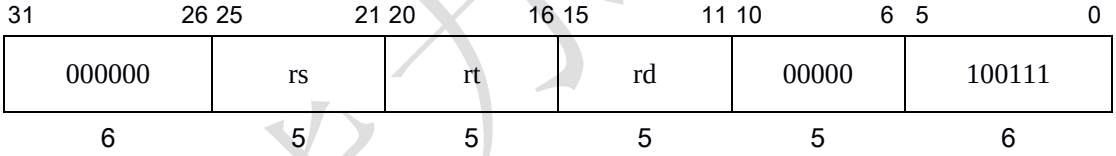
汇编格式：LUI rt, imm

功能描述：将 16 位立即数 imm 写入寄存器 rt 的高 16 位，寄存器 rt 的低 16 位置 0。

操作定义：GPR[rt] ← (imm || 0^{16})

例　外：无

3.4.4 NOR



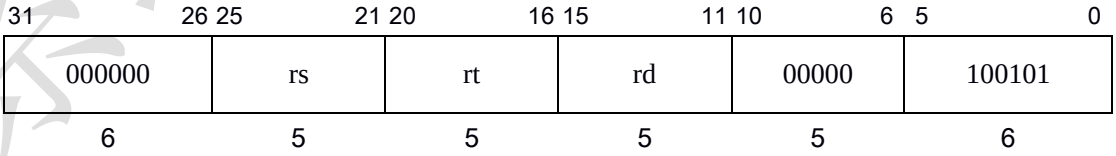
汇编格式：NOR rd, rs, rt

功能描述：寄存器 rs 中的值与寄存器 rt 中的值按位逻辑或非，结果写入寄存器 rd 中。

操作定义：GPR[rd] ← GPR[rs] nor GPR[rt]

例　外：无

3.4.5 OR



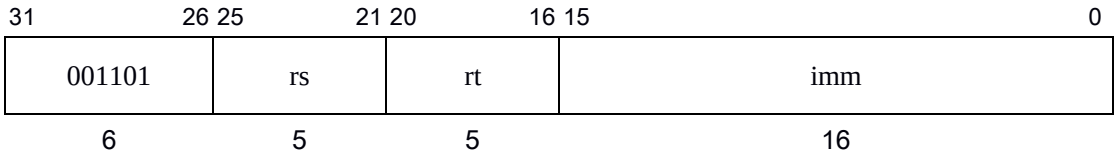
汇编格式：OR rd, rs, rt

功能描述：寄存器 rs 中的值与寄存器 rt 中的值按位逻辑或，结果写入寄存器 rd 中。

操作定义：GPR[rd] ← GPR[rs] or GPR[rt]

例　外：无

3.4.6 ORI



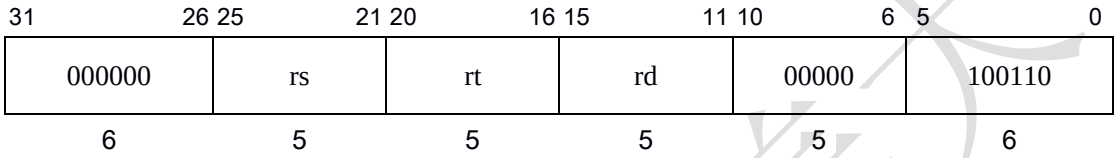
汇编格式：ORI rt, rs, imm

功能描述：寄存器 rs 中的值与 0 扩展至 32 位的立即数 imm 按位逻辑或，结果写入寄存器 rt 中。

操作定义：GPR[rt] ← GPR[rs] or Zero_extend(imm)

例　外：无

3.4.7 XOR



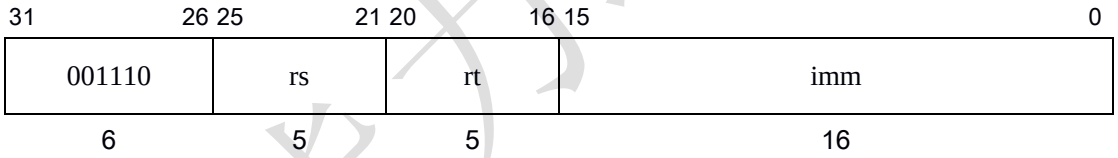
汇编格式：XOR rd, rs, rt

功能描述：寄存器 rs 中的值与寄存器 rt 中的值按位逻辑异或，结果写入寄存器 rd 中。

操作定义：GPR[rd] ← GPR[rs] xor GPR[rt]

例　外：无

3.4.8 XORI



汇编格式：XORI rt, rs, imm

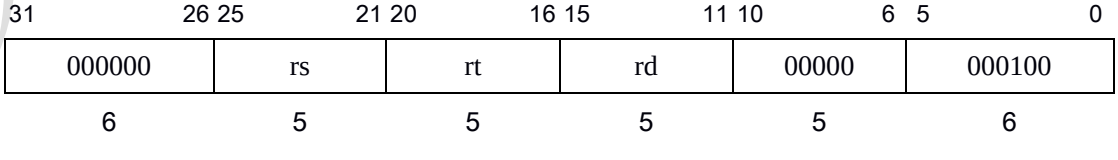
功能描述：寄存器 rs 中的值与 0 扩展至 32 位的立即数 imm 按位逻辑异或，结果写入寄存器 rt 中。

操作定义：GPR[rt] ← GPR[rs] xor Zero_extend(imm)

例　外：无

3.5 移位指令

3.5.1 SLLV



汇编格式：SLLV rd, rt, rs

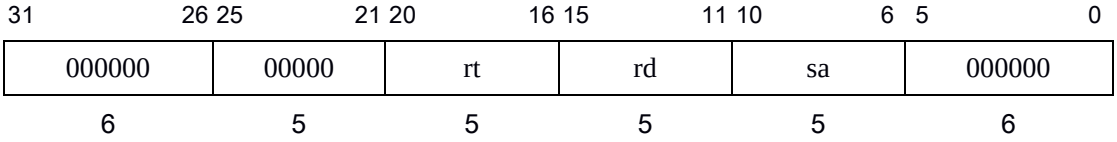
功能描述：由寄存器 rs 中的值指定移位量，对寄存器 rt 的值进行逻辑左移，结果写入寄存器 rd 中。

操作定义：$s \leftarrow GPR[rs]_{4..0}$

$GPR[rd] \leftarrow GPR[rt]_{(31-s)..0} || 0^s$

例　外：无

3.5.2 SLL



汇编格式：SLL rd, rt, sa

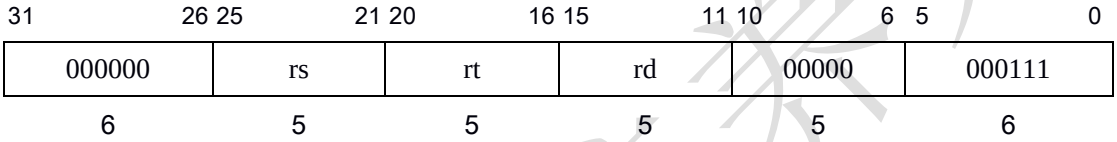
功能描述：由立即数 sa 指定移位量，对寄存器 rt 的值进行逻辑左移，结果写入寄存器 rd 中。

操作定义：s ← sa

$GPR[rd] \leftarrow GPR[rt]_{(31-s)..0} || 0^s$

例　外：无

3.5.3 SRAV



汇编格式：SRAV rd, rt, rs

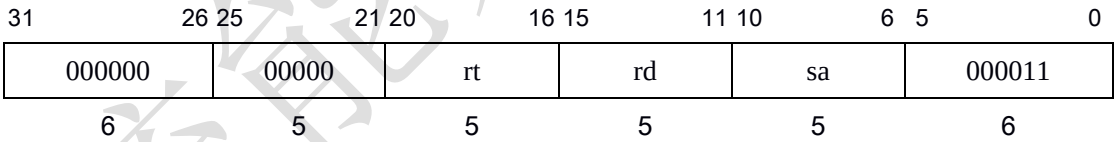
功能描述：由寄存器 rs 中的值指定移位量，对寄存器 rt 的值进行算术右移，结果写入寄存器 rd 中。

操作定义：$s \leftarrow GPR[rs]_{4..0}$

$GPR[rd] \leftarrow (GPR[rt]_{31})^s \,||\, GPR[rt]_{31..s}$

例　外：无

3.5.4 SRA



汇编格式：SRA rd, rt, sa

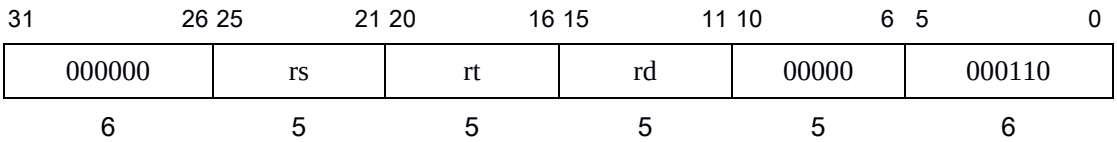
功能描述：由立即数 sa 指定移位量，对寄存器 rt 的值进行算术右移，结果写入寄存器 rd 中。

操作定义：s ← sa

$GPR[rd] \leftarrow (GPR[rt]_{31})^s \,||\, GPR[rt]_{31..s}$

例　外：无

3.5.5 SRLV



汇编格式：SRLV rd, rt, rs

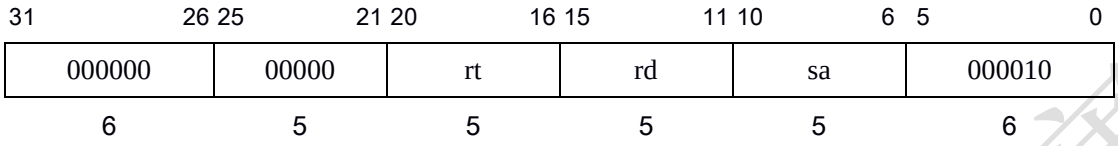
功能描述：由寄存器 rs 中的值指定移位量，对寄存器 rt 的值进行逻辑右移，结果写入寄存器 rd 中。

操作定义： $s \leftarrow GPR[rs]_{4..0}$

$GPR[rd] \leftarrow 0^s \| GPR[rt]_{31..s}$

例　外：无

3.5.6 SRL



汇编格式：SRL rd, rt, sa

功能描述：由立即数 sa 指定移位量，对寄存器 rt 的值进行逻辑右移，结果写入寄存器 rd 中。

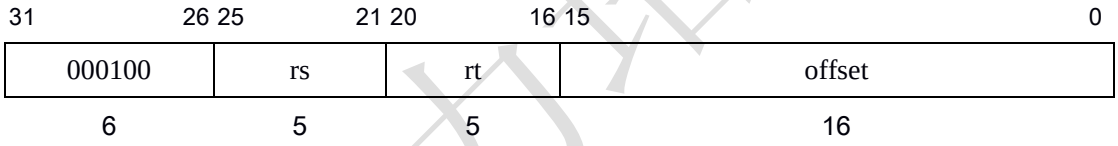
操作定义： $s \leftarrow sa$

$GPR[rd] \leftarrow 0^s \| GPR[rt]_{31..s}$

例　外：无

3.6 分支跳转指令

3.6.1 BEQ



汇编格式：BEQ rs, rt, offset

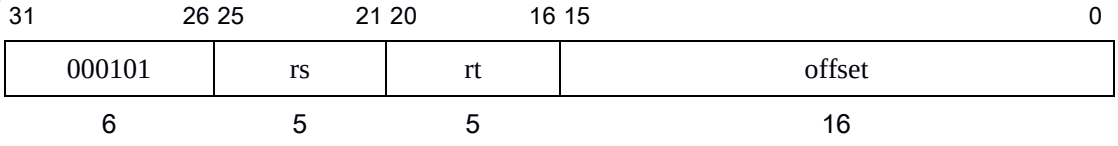
功能描述：如果寄存器 rs 的值等于寄存器 rt 的值则转移，否则顺序执行。转移目标由立即数 offset 左移 2 位并进行有符号扩展的值加上该分支指令对应的延迟槽指令的 PC 计算得到。

操作定义：

```
I:      condition ← GPR[rs] = GPR[rt]
        target_offset ← Sign_extend(offset||0^2)
I+1:    if condition then
            PC ← PC+ target_offset
        endif
```

例　外：无

3.6.2 BNE



汇编格式：BNE rs, offset

功能描述：如果寄存器 rs 的值不等于寄存器 rt 的值则转移，否则顺序执行。转移目标由立即数 offset 左移 2 位并进行有符号扩展的值加上该分支指令对应的延迟槽指令的 PC 计算得到。

操作定义：I: condition ← GPR[rs] ≠ GPR[rt]

target_offset ← Sign_extend(offset||0^2)

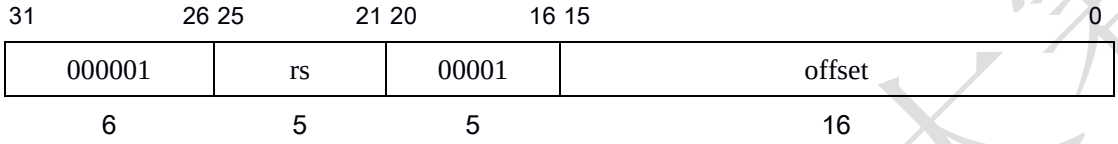
I+1: if condition then

PC ← PC+ target_offset

endif

例　外：无

3.6.3 BGEZ



汇编格式：BGEZ rs, offset

功能描述：如果寄存器 rs 的值大于等于 0 则转移，否则顺序执行。转移目标由立即数 offset 左移 2 位并进行有符号扩展的值加上该分支指令对应的延迟槽指令的 PC 计算得到。

操作定义：I: condition ← GPR[rs] ≥ 0

target_offset ← Sign_extend(offset||0^2)

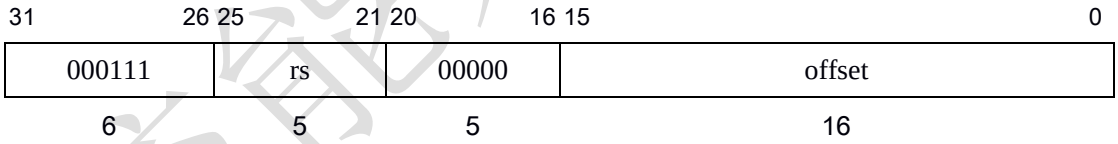
I+1: if condition then

PC ← PC+ target_offset

endif

例　外：无

3.6.4 BGTZ



汇编格式：BGTZ rs, offset

功能描述：如果寄存器 rs 的值大于 0 则转移，否则顺序执行。转移目标由立即数 offset 左移 2 位并进行有符号扩展的值加上该分支指令对应的延迟槽指令的 PC 计算得到。

操作定义：I: condition ← GPR[rs] > 0

target_offset ← Sign_extend(offset||0^2)

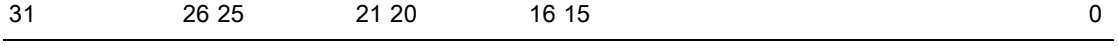
I+1: if condition then

PC ← PC+ target_offset

endif

例　外：无

3.6.5 BLEZ



000110	rs	00000	offset
6	5	5	16

汇编格式：BLEZ rs, offset

功能描述：如果寄存器 rs 的值小于等于 0 则转移，否则顺序执行。转移目标由立即数 offset 左移 2 位并进行有符号扩展的值加上该分支指令对应的延迟槽指令的 PC 计算得到。

操作定义：I: condition ← GPR[rs] ≤ 0

target_offset ← Sign_extend(offset||0^2)

I+1: if condition then

PC ← PC+ target_offset

endif

例　外：无

3.6.6 BLTZ

31 26	25 21	20 16	15 0
000001	rs	00000	offset
6	5	5	16

汇编格式：BLTZ rs, offset

功能描述：如果寄存器 rs 的值小于 0 则转移，否则顺序执行。转移目标由立即数 offset 左移 2 位并进行有符号扩展的值加上该分支指令对应的延迟槽指令的 PC 计算得到。

操作定义：I: condition ← GPR[rs] < 0

target_offset ← Sign_extend(offset||0^2)

I+1: if condition then

PC ← PC+ target_offset

endif

例　外：无

3.6.7 BGEZAL

31 26	25 21	20 16	15 0
000001	rs	10001	offset
6	5	5	16

汇编格式：BGEZAL rs, offset

功能描述：如果寄存器 rs 的值大于等于 0 则转移，否则顺序执行。转移目标由立即数 offset 左移 2 位并进行有符号扩展的值加上该分支指令对应的延迟槽指令的 PC 计算得到。无论转移与否，将该分支对应延迟槽指令之后的指令的 PC 值保存至第 31 号通用寄存器中。

操作定义：I: condition ← GPR[rs] ≥ 0

target_offset ← Sign_extend(offset||0^2)

GPR[31] ← PC + 8

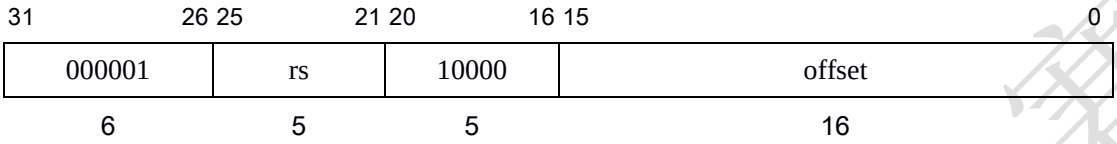
I+1: if condition then

PC ← PC+ target_offset

endif

例　外：无

3.6.8 BLTZAL



汇编格式：BLTZAL rs, offset

功能描述：如果寄存器 rs 的值小于 0 则转移，否则顺序执行。转移目标由立即数 offset 左移 2 位并进行有符号扩展的值加上该分支指令对应的延迟槽指令的 PC 计算得到。无论转移与否，将该分支对应延迟槽指令之后的指令的 PC 值保存至第 31 号通用寄存器中。

操作定义：I: condition ← GPR[rs] < 0

target_offset ← Sign_extend(offset||0^2)

GPR[31] ← PC + 8

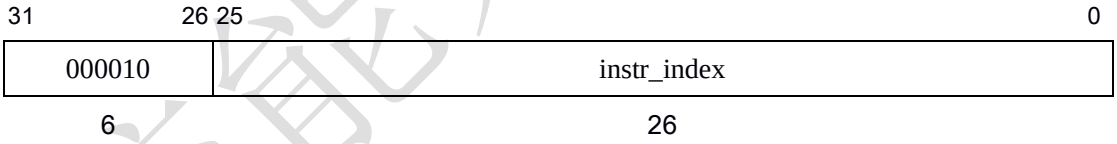
I+1: if condition then

PC ← PC+ target_offset

endif

例　外：无

3.6.9 J



汇编格式：J target

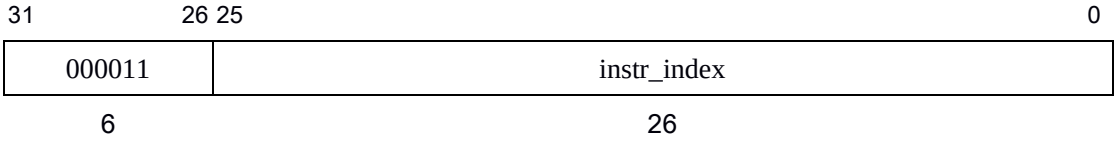
功能描述：无条件跳转。跳转目标由该分支指令对应的延迟槽指令的 PC 的最高 4 位与立即数 instr_index 左移 2 位后的值拼接得到。

操作定义：I:

I+1: PC ← $PC_{31..28}$ || instr_index || 0^2

例　外：无

3.6.10 JAL



汇编格式：JAL target

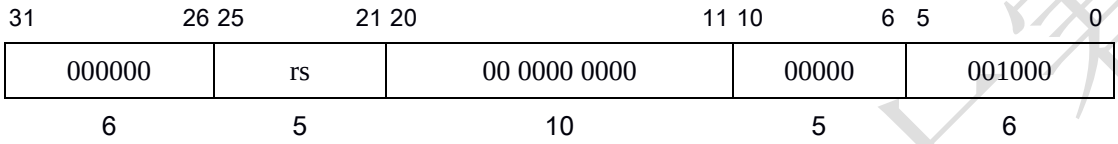
功能描述：无条件跳转。跳转目标由该分支指令对应的延迟槽指令的 PC 的最高 4 位与立即数 instr_index 左移 2 位后的值拼接得到。同时将该分支对应延迟槽指令之后的指令的 PC 值保存至第 31 号通用寄存器中。

操作定义：I: GPR[31] ← PC + 8

I+1: PC ← $PC_{31..28}$ || instr_index || 0^2

例　外：无

3.6.11 JR



汇编格式：JR rs

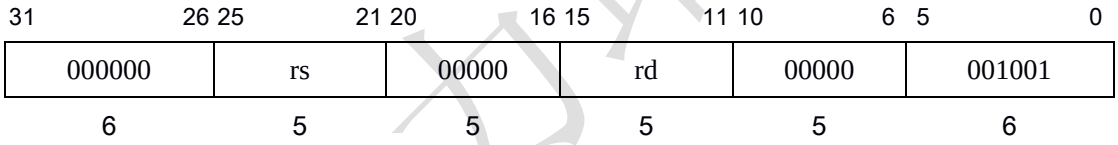
功能描述：无条件跳转。跳转目标为寄存器 rs 中的值。

操作定义：I: temp ← GPR[rs]

I+1: PC ← temp

例　外：无

3.6.12 JALR



汇编格式：JALR rd, rs

JALR rs (rd=31 implied)

功能描述：无条件跳转。跳转目标为寄存器 rs 中的值。同时将该分支对应延迟槽指令之后的指令的 PC 值保存至寄存器 rd 中。

操作定义：I: temp← GPR[rs]

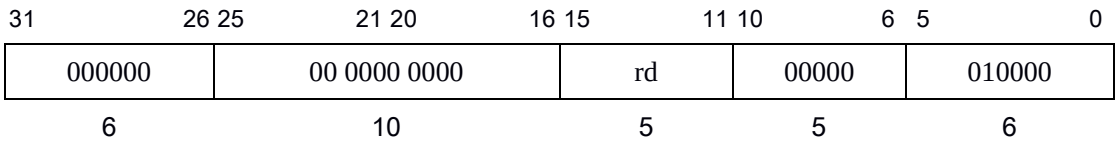
GPR[rd] ← PC + 8

I+1: PC ← temp

例　外：无

3.7 数据移动指令

3.7.1 MFHI



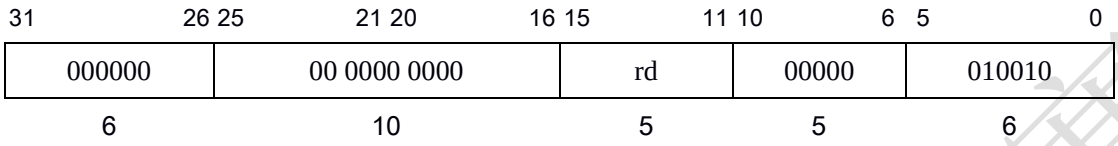
汇编格式：MFHI rd

功能描述：将 HI 寄存器的值写入到寄存器 rd 中。

操作定义：GPR[rd] ← HI

例　外：无

3.7.2 MFLO



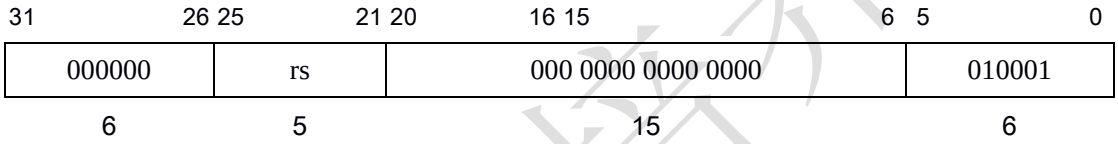
汇编格式：MFLO rd

功能描述：将 LO 寄存器的值写入到寄存器 rd 中。

操作定义：GPR[rd] ← LO

例　外：无

3.7.3 MTHI



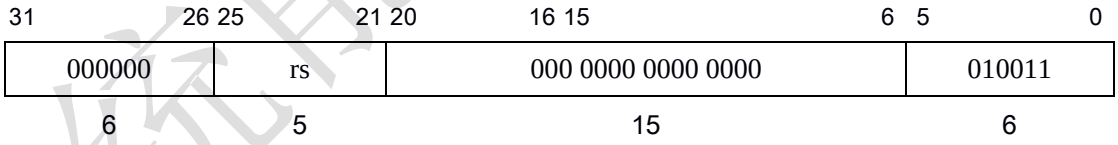
汇编格式：MTHI rs

功能描述：将寄存器 rs 的值写入到 HI 寄存器中。

操作定义：HI ← GPR[rs]

例　外：无

3.7.4 MTLO



汇编格式：MTLO rs

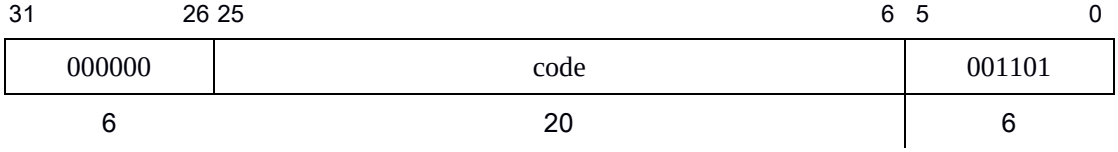
功能描述：将寄存器 rs 的值写入到 LO 寄存器中。

操作定义：LO ← GPR[rs]

例　外：无

3.8 自陷指令

3.8.1 BREAK



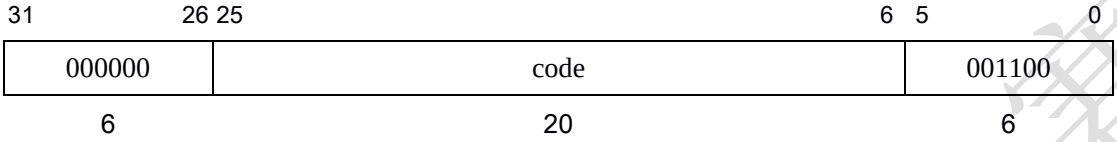
汇编格式：BREAK

功能描述：触发断点例外。

操作定义：SignalException(Breakpoint)

例　外：断点例外

3.8.2 SYSCALL



汇编格式：SYSCALL

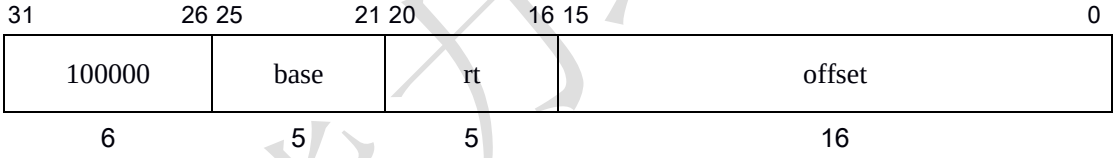
功能描述：触发系统调用例外。

操作定义：SignalException(SystemCall)

例　外：系统调用例外

3.9 访存指令

3.9.1 LB



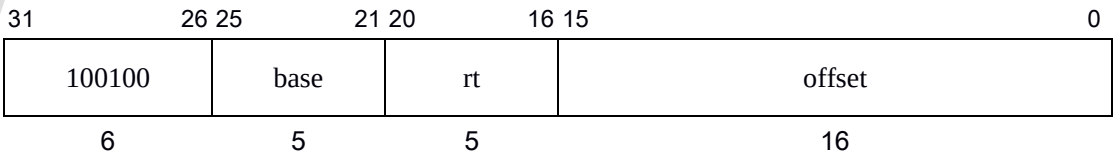
汇编格式：LB rt, offset(base)

功能描述：将 base 寄存器的值加上符号扩展后的立即数 offset 得到访存的虚地址，据此虚地址从存储器中读取 1 个字节的值并进行符号扩展，写入到 rt 寄存器中。

操作定义：vAddr ← GPR[base] + sign_extend(offset)
(pAddr, CCA) ← AddressTranslation(vAddr, DATA, LOAD)
membyte ← LoadMemory(CCA, BYTE, pAddr, vAddr, DATA)
GPR[rt] ← sign_extend($membyte_{7..0}$)

例　外：无

3.9.2 LBU



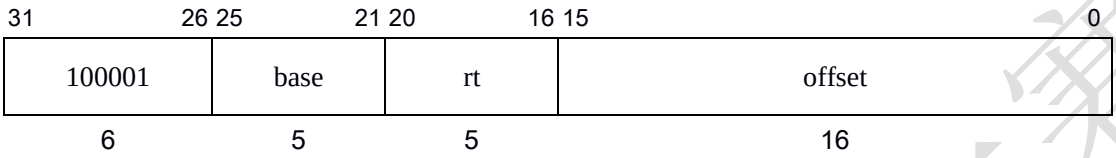
汇编格式：LBU rt, offset(base)

功能描述：将 base 寄存器的值加上符号扩展后的立即数 offset 得到访存的虚地址，据此虚地址从存储器中读取 1 个字节的值并进行 0 扩展，写入到 rt 寄存器中。

操作定义：vAddr ← GPR[base] + sign_extend(offset)
(pAddr, CCA) ← AddressTranslation(vAddr, DATA, LOAD)
membyte ← LoadMemory(CCA, BYTE, pAddr, vAddr, DATA)
GPR[rt] ← zero_extend($membyte_{7..0}$)

例　外：无

3.9.3 LH



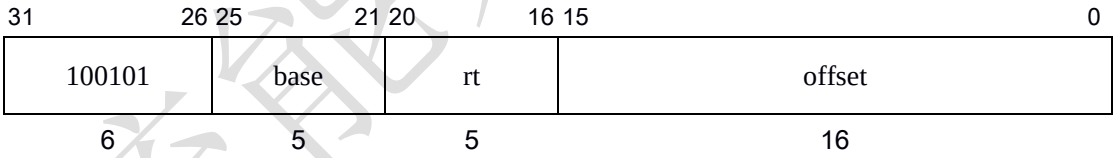
汇编格式：LH rt, offset(base)

功能描述：将 base 寄存器的值加上符号扩展后的立即数 offset 得到访存的虚地址，如果地址不是 2 的整数倍则触发地址错例外，否则据此虚地址从存储器中读取连续 2 个字节的值并进行符号扩展，写入到 rt 寄存器中。

操作定义：vAddr ← GPR[base] + sign_extend(offset)
if $vAddr_0 \neq 0$ then
　　SignalException(AddressError)
endif
(pAddr, CCA) ← AddressTranslation(vAddr, DATA, LOAD)
memhalf ← LoadMemory(CCA, HALFWORD, pAddr, vAddr, DATA)
GPR[rt] ← sign_extend($memhalf_{15..0}$)

例　外：地址最低 1 位不为 0，触发地址错例外

3.9.4 LHU



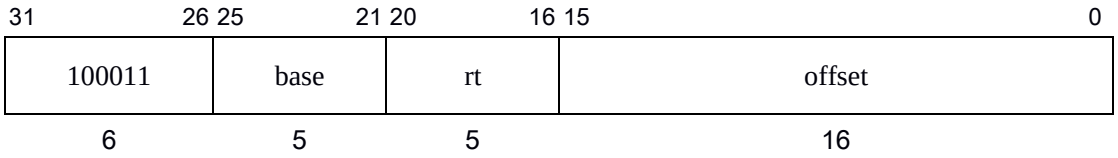
汇编格式：LHU rt, offset(base)

功能描述：将 base 寄存器的值加上符号扩展后的立即数 offset 得到访存的虚地址，如果地址不是 2 的整数倍则触发地址错例外，否则据此虚地址从存储器中读取连续 2 个字节的值并进行 0 扩展，写入到 rt 寄存器中。

操作定义：vAddr ← GPR[base] + sign_extend(offset)
if $vAddr_0 \neq 0$ then
　　SignalException(AddressError)
endif
(pAddr, CCA) ← AddressTranslation(vAddr, DATA, LOAD)
memhalf ← LoadMemory(CCA, HALFWORD, pAddr, vAddr, DATA)
GPR[rt] ← zero_extend($memhalf_{15..0}$)

例　外：地址最低 1 位不为 0，触发地址错例外

3.9.5 LW



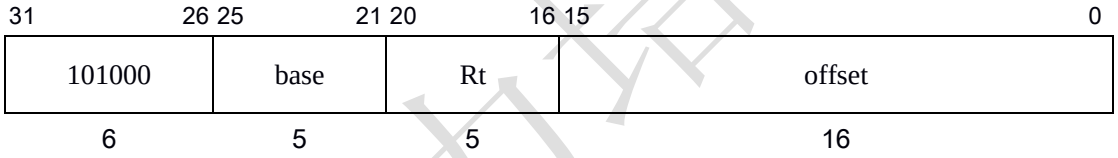
汇编格式：LW rt, offset(base)

功能描述：将 base 寄存器的值加上符号扩展后的立即数 offset 得到访存的虚地址，如果地址不是 4 的整数倍则触发地址错例外，否则据此虚地址从存储器中读取连续 4 个字节的值并进行符号扩展，写入到 rt 寄存器中。

操作定义：vAddr ← GPR[base] + sign_extend(offset)
if $vAddr_{1..0} \neq 0$ then
SignalException(AddressError)
endif
(pAddr, CCA) ← AddressTranslation(vAddr, DATA, LOAD)
memword ← LoadMemory(CCA, WORD, pAddr, vAddr, DATA)
GPR[rt] ← sign_extend($memword_{31..0}$)

例　外：地址最低 2 位不为 0，触发地址错例外

3.9.6 SB



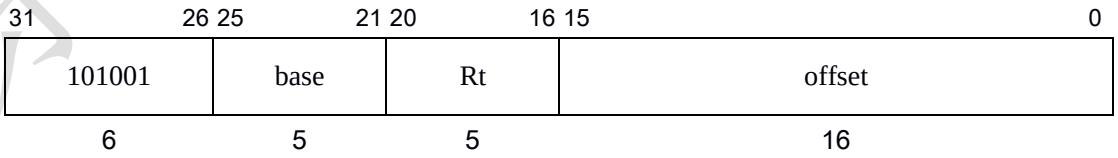
汇编格式：SB rt, offset(base)

功能描述：将 base 寄存器的值加上符号扩展后的立即数 offset 得到访存的虚地址，据此虚地址将 rt 寄存器的最低字节存入存储器中。

操作定义：vAddr ← GPR[base] + sign_extend(offset)
(pAddr, CCA) ← AddressTranslation(vAddr, DATA, STORE)
databyte ← $GPR[rt]_{7..0}$
StoreMemory(CCA, BYTE, databyte, pAddr, vAddr, DATA)

例　外：无

3.9.7 SH



汇编格式：SH rt, offset(base)

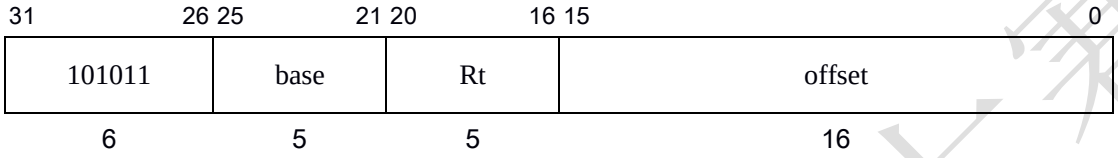
功能描述：将 base 寄存器的值加上符号扩展后的立即数 offset 得到访存的虚地址，如果地址不是 2 的整数倍则触发地址错例外，否则据此虚地址将 rt 寄存器的低半字存入存储器中。

操作定义：vAddr ← GPR[base] + sign_extend(offset)
if $vAddr_0 \neq 0$ then

SignalException(AddressError)
endif
(pAddr, CCA) ← AddressTranslation(vAddr, DATA, STORE)
datahalf ← $GPR[rt]_{15..0}$
StoreMemory(CCA, HALFWORD, datahalf, pAddr, vAddr, DATA)

例　外：地址最低 1 位不为 0，触发地址错例外

3.9.8 SW



汇编格式：SW rt, offset(base)

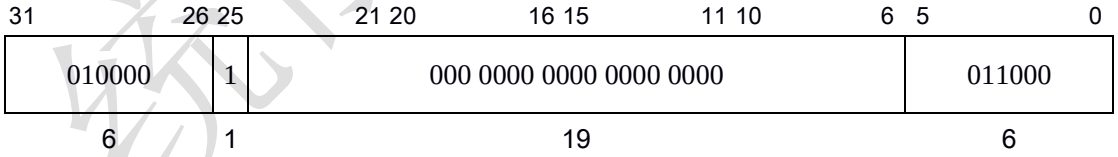
功能描述：将 base 寄存器的值加上符号扩展后的立即数 offset 得到访存的虚地址，如果地址不是 4 的整数倍则触发地址错例外，否则据此虚地址将 rt 寄存器存入存储器中。

操作定义：vAddr ← GPR[base] + sign_extend(offset)
if $vAddr_{1..0} \neq 0$ then
SignalException(AddressError)
endif
(pAddr, CCA) ← AddressTranslation(vAddr, DATA, STORE)
dataword ← $GPR[rt]_{31..0}$
StoreMemory(CCA, WORD, dataword, pAddr, vAddr, DATA)

例　外：地址最低 2 位不为 0，触发地址错例外

3.10特权指令

3.10.1 ERET



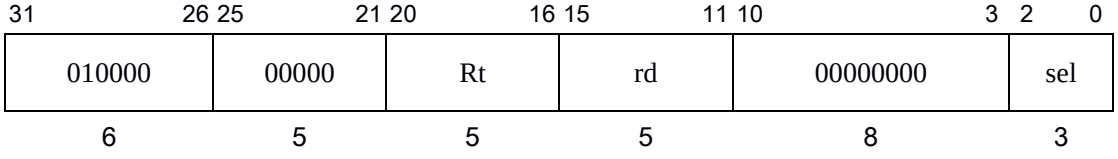
汇编格式：ERET

功能描述：从中断、例外处理返回。

操作定义：PC ← EPC，Status.EXL ← 0，刷新流水线。

例　外：无

3.10.2 MFC0



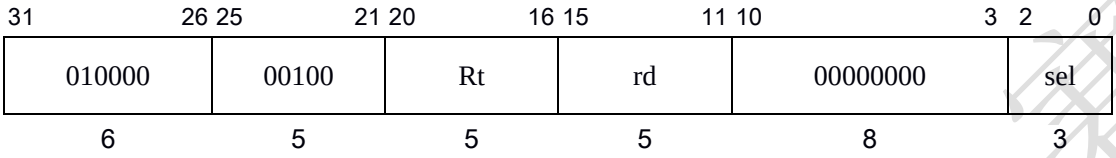
汇编格式：MFC0 rt, rd, sel

功能描述：从协处理器 0 的寄存器取值

操作定义：GPR[rt] ← CP0[rd, sel]

例　外：无

3.10.3 MTC0



汇编格式：MTC0 rt, rd, sel

功能描述：向协处理器 0 的寄存器存值

操作定义：CP0[rd, sel] ← GPR[rt]

例　外：无

4 存储管理

本指令系统对操作模式和存储访问类型进行了严格的限定，所以存储管理部分仅对虚实地址映射方式进行定义。

处理器可访问 32 位虚地址空间。整个 4GB 大小的虚地址空间被划分成 5 个段。处理器采用固定地址映射（FMT）机制，在进行虚实地址映射时，不同段的虚地址以线性方式固定地映射到一段连续的物理地址上。具体映射方式如图 4-1 所示。

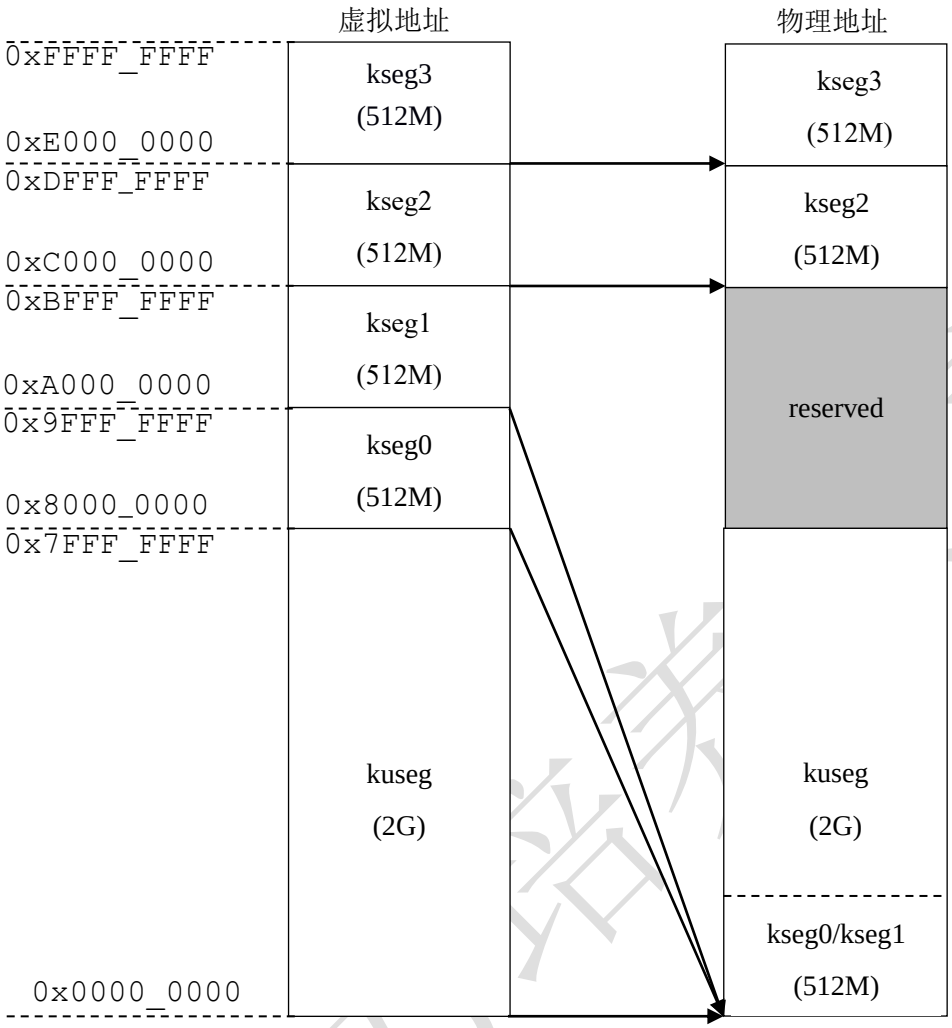


图 4-1 虚实地址映射关系图

5 中断与例外

5.1 处理器例外

5.1.1 例外优先级

当一条指令同时满足多个例外触发条件时，处理器核将按照表 5-1 所示例外优先级，优先触发优先级高的例外。

表 5-1 例外优先级

例外	类型
中断	异步
地址错例外—取指	同步
保留指令例外	同步
整型溢出例外、陷阱例外、系统调用例外	同步
地址错例外—数据访问	同步

5.1.2 例外入口向量位置

所有例外（含中断）的例外入口地址统一为 0xBFC0.0380

5.1.3 处理器硬件响应例外的一般性过程

当开始响应例外时，处理器硬件会作以下事情

(1) 当 CP0.Status.EXL 为 0 时，更新 CP0.EPC：将例外处理返回后重新开始执行的指令 PC 填入到 CP0.EPC 寄存器寄存器中。如果发生例外的指令不在分支延迟槽中，则重新开始执行的指令 PC 就等于发生例外的指令的 PC；否则重新开始执行的指令 PC 等于发生例外的指令的 PC-4（即该延迟槽对应的分支指令的 PC），

(2) 当 EXL 为 0 时，更新 CP0.Cause 寄存器的 BD 位：如果发生例外的指令在分支延迟槽中，则将 CP0.Cause.BD 置为 1。

(3) 更新 CP0.Status.EXL 位：将其置为 1。

(4) 进入约定的例外入口重新取指，软件开始执行例外处理程序。

当软件完成例外处理程序的执行后，通常会使用 ERET 指令从例外处理程序返回，ERET 指令的作用是：

(1) 将 CP0.Status.EXL 清零。

从 CP0.EPC 寄存器取出重新执行的 PC 值，然后从该 PC 值处继续取指执行。

5.1.4 中断例外

当未屏蔽的中断到来时，触发中断例外。关于中断的详细描述，请参考。

控制寄存器 Cause 的 ExcCode 域：

0x00 (Int)

响应例外时的额外硬件状态更新：

寄存器	状态更新描述
Cause	IP 域记录待处理的中断。

5.1.5 地址错例外

当发生下列条件时触发地址错例外：

- 字 load/store 指令，其访问地址不对齐于字边界。
- 半字 load/store 指令，其访问地址不对齐与半字边界。
- 取指 PC 不对齐于字边界。

控制寄存器 Cause 的 ExcCode 域：

0x04 (AdEL)：取指或读数据

0x05 (AdES)：写数据

响应例外时的额外硬件状态更新：

寄存器	状态更新描述
BadVAddr	记录触发例外的虚地址。

5.1.6 整型溢出例外

当一条 ADD、ADDI 或 SUB 指令执行结果溢出时，触发整型溢出例外。

控制寄存器 Cause 的 ExcCode 域：

0x0c (Ov)

响应例外时的额外硬件状态更新：

无

5.1.7 系统调用例外

当执行 SYSCALL 指令时，触发系统调用例外。

控制寄存器 Cause 的 ExcCode 域：

0x08 (Sys)

响应例外时的额外硬件状态更新：

无

5.1.8 断点例外

当执行一条 BREAK 指令时，触发断点例外。

控制寄存器 Cause 的 ExcCode 域：

0x09 (Bp)

响应例外时的额外硬件状态更新：

无

5.1.9 保留指令例外

当执行一条未实现的指令时，触发保留指令例外。

控制寄存器 Cause 的 ExcCode 域：

0x0a (RI)

响应例外时的额外硬件状态更新：

无

5.2 中断

本节描述的中断所涉及的范围包括硬件中断、软件中断和计时器中断。

5.2.1 中断响应的必要条件

处理器响应中断的必要条件是：

- Status.IE=1，表示全局中断使能开启。
- Status.EXL=0，表示没有例外正在处理。
- 某个中断源产生中断且该中断源未被屏蔽外，

5.2.2 中断模式

处理器支持 2 个软件中断 (SW0~SW1)、6 个硬件中断 (HW0~HW5)和 1 个计时器中断。其中计时器中断复用 HW5 硬件中断。

软件中断的中断源即为 Cause.IP[1:0]两位，仅可以通过软件对 Cause.IP[1:0]位写 1 进行触发，软件写 0 进行清除。

计时器中断的中断源记录在 Cause.TI 位中，在 Count[31:0]等于 Compare[31:0]的情况下由硬件置 1，软件可通过写 Compare 寄存器间接清除 Cause.TI 位记录的中断。

硬件中断的中断源来自于处理器外部，由硬件逐拍采样处理器接口上的 6 个中断输入引脚，软件需要反向遍历系统的中断路由路径，清除终端设备或路由路径上的中断状态，以此来清除处理器的硬件中断。

除全局中断使能外，每个中断源各自含有一个中断屏蔽位。各中断请求生成关系如表 5-2 所示。

表 5-2 各中断请求生成条件

中断类型	中断源	中断请求生成
硬件中断 5 号、计时器中断	HW5	Cause.IP7 & Status.IM7
硬件中断 4~0 号	HW4	Cause.IP6 & Status.IM6
	HW3	Cause.IP5 & Status.IM5
	HW2	Cause.IP4 & Status.IM4
	HW1	Cause.IP3 & Status.IM3
	HW0	Cause.IP2 & Status.IM2
软件中断	SW1	Cause.IP1 & Status.IM1
	SW0	Cause.IP0 & Status.IM0

所有中断采用相同的例外入口地址，中断例外处理程序需查询 Cause.IP 以及 Status.IM 以确定具体中断源。对于同时出现多个有效中断源的情况，软件可通过对查询次序调控来实现中断处理的优先级。

6 系统控制寄存器

6.1 系统控制寄存器概览

系统控制寄存器一览表

Reg	Sel.	寄存器名称	功能定义
8	0	BadVAddr	记录最新地址相关例外的出错地址
9	0	Status	处理器内部计数器
12	0	Status	处理器状态与控制寄存器
13	0	Cause	存放上一次例外原因
14	0	EPC	存放上一次发生例外指令的 PC

6.2 BadVAddr 寄存器 (CP0 Register 8, Select 0)

BadVAddr 寄存器是一个只读寄存器，用于记录最近一次导致发生地址错例外的虚地址：

图 6-1 说明了 BadVAddr 寄存器的格式；表 6-1 对 BadVAddr 寄存器各域进行了描述。

图 6-1 BadVAddr 寄存器格式

表 6-1 BadVAddr 寄存器域描述

域名称	位	功能描述	读/写	复位值
BadVAddr	31..0	出错的虚地址。	R	无

6.3 Count 寄存器 (CP0 Register 9, Select 0)

Count 寄存器与 Compare 寄存器配合在一起，用于实现一个处理器内部的高精度定时器及定时中断。该计时器自增 1 的频率是处理器核流水线时钟的频率的 1/2。在执行过程中，处理器核流水线时钟频率可能被动态调整，因此 Count 的自增频率也随之变化。

为完成某些功能或诊断目的，软件可以配置 Count 寄存器，例如计时器的复位、同步等操作。

图 6-2 说明了 Count 寄存器的格式；表 6-2 对 Count 寄存器各域进行了描述。

图 6-2 Count 寄存器格式

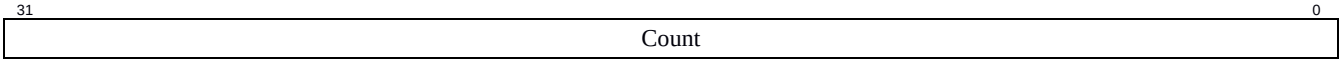


表 6-2 Count 寄存器域描述

域名称	位	功能描述	读/写	复位值

域名称	位	功能描述	读/写	复位值
Count	31..0	内部计数器。每隔一个 CPU Clock（也就是每两个 CPU Clock）累加 1。	R/W	无

6.4 Status 寄存器 (CP0 Register 12, Select 0)

Status 寄存器是一个可读写寄存器，包含有处理器操作模式、中断使能以及处理器状态诊断信息。

图 6-3 说明了 Status 寄存器的格式；表 6-3 对 Status 寄存器各域进行了描述。

图 6-3 Status 寄存器格式

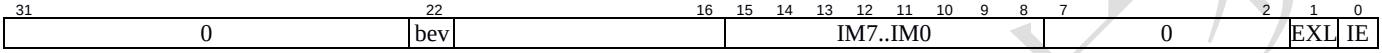


表 6-3 Status 寄存器域描述

域名称	位	功能描述	读/写	复位值
0	31..23	只读恒为 0。	0	0
Bev	22	针对大赛，恒设为 1	R	1
0	21..16	只读恒为 0。	0	0
IM7..IM0	15..8	中断屏蔽位。每一位分别控制一个外部中断、内部中断或软件中断的使能。 1：使能；0：屏蔽。	R/W	无
0	7..2	只读恒为 0。	0	0
EXL	1	例外级。当发生例外时该位被置 1。0：正常级；1：例外级。 当 EXL 位置为 1 时： ◆ 处理器自动处于核心态 ◆ 所有硬件与软件中断被屏蔽 ◆ EPC、$Cause_{BD}$ 在发生新的例外时不做更新。	R/W	0x0
IE	0	全局中断使能位。 0：屏蔽所有硬件和软件中断； 1：使能所有硬件和软件中断。	R/W	0x0

6.5 Cause 寄存器 (CP0 Register 13, Select 0)

Cause 寄存器主要用于描述最近一次例外的原因。除此之外还对软件中断进行了控制。除了 IP1..IP0 域外，Cause 寄存器的其它域对于软件均只读。

图 6-4 说明了 Cause 寄存器的格式；表 6-4 对 Cause 寄存器各域进行了描述。

图 6-4 Cause 寄存器格式

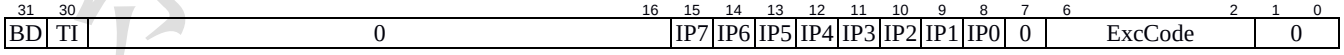


表 6-4 Cause 寄存器域描述

域名称	位	功能描述	读/写	复位值
BD	31	标识最近发生例外的指令是否处于分支延迟槽。1：在延迟槽中；0：不在延迟槽中	R	0x0
TI	30	计时器中断指示。1：有待处理的计时器中断；0：没有计时器中断。	R	0x0
0	30..16	只读恒为 0。	0	0

域名称	位	功能描述	读/写	复位值
IP7..IP2	15..10	待处理硬件中断标识。每一位对应一个中断线，IP7~IP2 依次对应硬件中断 5~0。 1：该中断线上由待处理的中断；0：该中断线上无中断。	R	0x0
IP1..IP0	9..8	待处理软件中断标识。每一位对应一个软件中断，IP1~IP0 依次对应软件中断 1~0。 软件中断标识位可由软件设置和清除。	R/W	0x0
0	7	只读恒为 0。	0	0
ExcCode	6..2	例外编码。详细描述请见表 6-5。		
0	1..0	只读恒为 0。	0	0

表 6-5 ExcCode 编码及其对应例外类型

ExcCode	助记符	描述
0x00	Int	中断
0x04	AdEL	地址错例外（读数据或取指令）
0x05	AdES	地址错例外（写数据）
0x08	Sys	系统调用例外。
0x09	Bp	断点例外。
0x0a	RI	保留指令例外。
0x0c	Ov	算出溢出例外。

6.6 EPC 寄存器 (CP0 Register 14, Select 0)

EPC 寄存器是一个 32 位可读写寄存器，其包含例外处理完成后继续开始执行的指令的 PC。

在响应同步（精确）例外时，处理器向 EPC 寄存器中写入：

直接触发例外的指令的 PC。

当直接触发例外的指令位于分支延迟槽时，记录该指令前一条分支或跳转指令的 PC，同时 Cause.BD 置为 1。

在响应异步（非精确）例外时，处理器向 EPC 寄存器中写入例外处理完成后继续执行的指令的 PC。

当 Status 寄存器的 EXL 位为 1 时，发生例外时不更新 EPC 寄存器。

图 6-5 说明了 EPC 寄存器的格式；表 6-6 对 EPC 寄存器各域进行了描述。

图 6-5 EPC 寄存器格式

表 6-6 EPC 寄存器域描述

域名称	位	功能描述	读/写	复位值
EPC	31..0	例外程序计数器。	R/W	无

参考文献

[1] MIPS Architecture For Programmers Volume I-A: Introduction to the MIPS32 Architecture. Revision 3.02.

[2] MIPS Architecture For Programmers Volume II-A: The MIPS32 Instruction Set. Revision 3.02.

[3] MIPS Architecture For Programmers Vol. III: MIPS32/microMIPS32 Privileged Resource Architecture. Revision 3.02.

[4] See MIPS Run Linux, 2nd Edition, 2006, Morgan Kaufmann, Dominic Sweetman