附录 A 实现的 MIPS 指令集

附 A1 单周期 CPU 实现指令

（1） 无符号加法

ADDU rd, rs, rt　　R 型

31 26	25 21	20 16	15 11	10 6	5 0
000000	rs	rt	rd	00000	100001
6	5	5	5	5	6

GPR[rd] ← GPR[rs] + GPR[rt]

（2） 无符号减法

SUBU rd, rs, rt　　R 型

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

GPR[rd] ← GPR[rs] - GPR[rt]

（3） 有符号比较，小于置位

SLT rd, rs, rt　　R 型

31 26	25 21	20 16	15 11	10 6	5 0
000000	rs	rt	rd	00000	101010
6	5	5	5	5	6

GPR[rd] ← (sign(GPR[rs]) < sign(GPR[rt]))

（4） 按位与

AND rd, rs, rt　　R 型

31 26	25 21	20 16	15 11	10 6	5 0
000000	rs	rt	rd	00000	100100
6	5	5	5	5	6

GPR[rd] ← GPR[rs] & GPR[rt]

（5） 按位或非

NOR rd, rs, rt　　R 型

31 26	25 21	20 16	15 11	10 6	5 0
000000	rs	rt	rd	00000	100111
6	5	5	5	5	6

GPR[rd] ← ~(GPR[rs] | GPR[rt])

（6） 按位或

OR rd, rs, rt　　R 型

31 26	25 21	20 16	15 11	10 6	5 0
000000	rs	rt	rd	00000	100101
6	5	5	5	5	6

GPR[rd] ← GPR[rs] | GPR[rt]

（7） 按位异或

XOR rd, rs, rt　　R 型

31 26	25 21	20 16	15 11	10 6	5 0
000000	rs	rt	rd	00000	100110
6	5	5	5	5	6

GPR[rd] ← GPR[rs] ^ GPR[rt]

（8） 逻辑左移

SLL rd, rt, shf　　R 型

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

GPR[rd] ← zero(GPR[rt]) << shf

（9） 逻辑右移

SRL rd, rt, shf　　R 型

31 26	25 21	20 16	15 11	10 6	5 0
000000	00000	rt	rd	shf	000010
6	5	5	5	5	6

GPR[rd] ← zero(GPR[rt]) >> shf

（10） 立即数、无符号加法

ADDIU rt, rs, imm　　I 型

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

GPR[rt] ← GPR[rs] + sign_ext(imm)

（11） 相等跳转

BEQ rs, rt, offset　　I 型

31 26	25 21	20 16	15 0
000100	rs	rt	offset

6	5	5	16

if GPR[rs] = GPR[rt] then PC ← B_PC + sign_ext(offset)<<2

B_PC：分支跳转参与运算的 PC，在不考虑延迟槽时为分支跳转指令的 PC，考虑延迟槽时为延迟槽指令的 PC，即分支跳转指令的 PC+4。

（12） 不等跳转

BNE rs, rt, offset　　I 型

31 26	25 21	20 16	15 0
000101	rs	rt	offset
6	5	5	16

if GPR[rs] ≠ GPR[rt] then PC ← B_PC + sign_ext(offset)<<2

B_PC：分支跳转参与运算的 PC，在不考虑延迟槽时为分支跳转指令的 PC，考虑延迟槽时为延迟槽指令的 PC，即分支跳转指令的 PC+4。

（13） 装载字

LW rt, offset(base)　　I 型

31 26	25 21	20 16	15 0
100011	base	rt	offset
6	5	5	16

GPR[rt] ← Mem[GPR[base] + sign_ext(offset)]

（14） 存储字

SW rt, offset(base)　　I 型

31 26	25 21	20 16	15 0
101011	base	rt	offset
6	5	5	16

Mem[GPR[base] + sign_ext(offset)] ← GPR[rt]

（15） 立即数装载高位

LUI rt, imm　　I 型

31 26	25 21	20 16	15 0
001111	00000	rt	imm
6	5	5	16

GPR[rt] ← {imm,16’d0}

（16） 直接跳转

J target　　J 型

31 26	25 0
000010	target
6	26

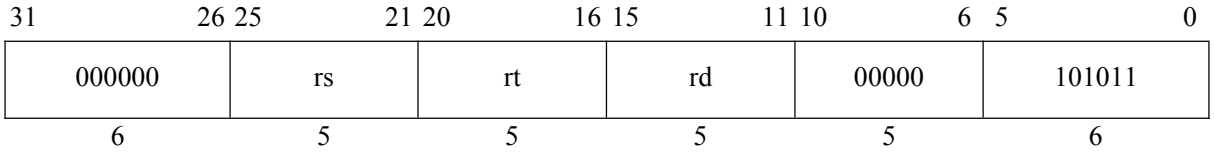
PC ← {B_PC[31:28], target << 2}

B_PC：分支跳转参与运算的 PC，在不考虑延迟槽时为分支跳转指令的 PC，考虑延迟槽时为延迟槽指令的 PC，即分支跳转指令的 PC+4。

附 A2 多周期 CPU 新增实现指令

（17） 无符号小于置位

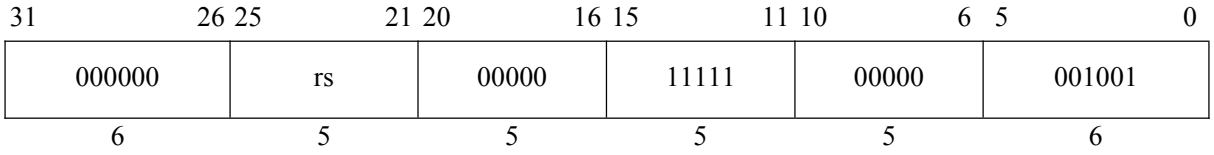
SLTU rd, rs, rt　　R 型



GPR[rd] ← (zero(GPR[rs]) < zero(GPR[rt]))

（18） 跳转寄存器并链接

JALR rs　　R 型

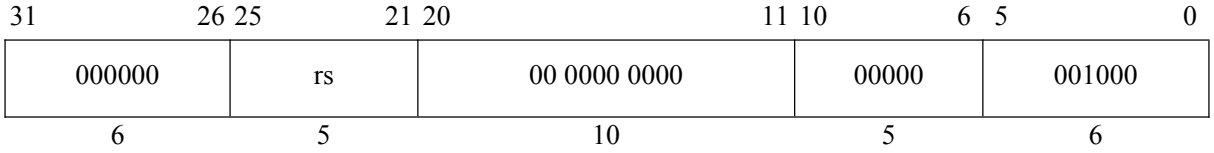


GPR[31] ← B_PC + 4, PC ← GPR[rs]

B_PC：分支跳转参与运算的 PC，在不考虑延迟槽时为分支跳转指令的 PC，考虑延迟槽时为延迟槽指令的 PC，即分支跳转指令的 PC+4。

（19） 跳转寄存器

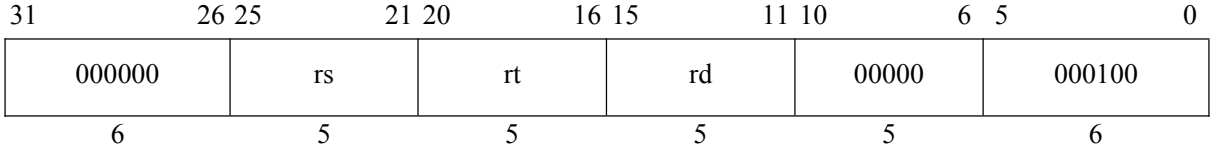
JR rs　　R 型



PC ← GPR[rs]

（20） 变量逻辑左移

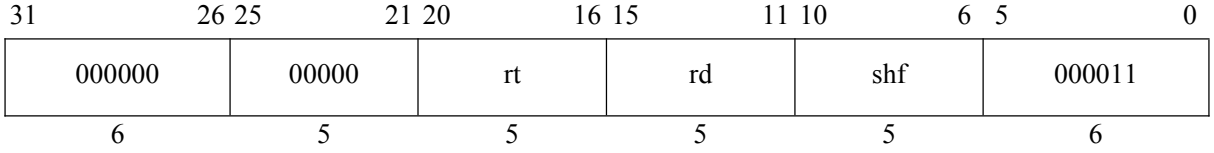
SLLV rd, rt, rs　　R 型



GPR[rd] ← zero(GPR[rt]) << GPR[rs]

（21） 算术右移

SRA rd, rt, shf　　R 型



GPR[rd] ← sign(GPR[rt]) >> shf

（22） 变量算术右移

SRAV rd, rt, rs　　R 型

31 26	25 21	20 16	15 11	10 6	5 0
000000	rs	rt	rd	00000	000111
6	5	5	5	5	6

GPR[rd] ← sign(GPR[rt]) >> GPR[rs]

（23） 变量逻辑右移

SRLV rd, rt, rs　　R 型

31 26	25 21	20 16	15 11	10 6	5 0
000000	rs	rt	rd	00000	000110
6	5	5	5	5	6

GPR[rd] ← zero(GPR[rt]) >> GPR[rs]

（24） 立即数有符号比较，小于置位

SLTI rt, rs, imm　　I 型

31 26	25 21	20 16	15 0
001010	rs	rt	imm
6	5	5	16

GPR[rt] ← (sign(GPR[rs]) < sign_ext(imm))

（25） 立即数无符号比较，小于置位

SLTIU rt, rs, imm　　I 型

31 26	25 21	20 16	15 0
001011	rs	rt	imm
6	5	5	16

GPR[rt] ← (zero(GPR[rs]) < sign_ext(imm))

（26） 大于或等于零跳转

BGEZ rs, offset　　I 型

31 26	25 21	20 16	15 0
000001	rs	00001	offset
6	5	5	16

if GPR[rs]≥0 then PC ← B_PC + sign_ext(offset)<<2

B_PC：分支跳转参与运算的 PC，在不考虑延迟槽时为分支跳转指令的 PC，考虑延迟槽时为延迟槽指令的 PC，即分支跳转指令的 PC+4。

（27） 大于零跳转

BGTZ rs, offset　　I 型

31 26	25 21	20 16	15 0

000111	rs	00000	offset
6	5	5	16

if GPR[rs] > 0 then PC ← B_PC + sign_ext(offset)<<2

B_PC：分支跳转参与运算的 PC，在不考虑延迟槽时为分支跳转指令的 PC，考虑延迟槽时为延迟槽指令的 PC，即分支跳转指令的 PC+4。

（28） 小于或等于零跳转

BLEZ rs, offset　　I 型

31 26	25 21	20 16	15 0
000110	rs	00000	offset
6	5	5	16

if GPR[rs] ≤ 0 then PC ← B_PC + sign_ext(offset)<<2

B_PC：分支跳转参与运算的 PC，在不考虑延迟槽时为分支跳转指令的 PC，考虑延迟槽时为延迟槽指令的 PC，即分支跳转指令的 PC+4。

（29） 小于零跳转

BLTZ rs, offset　　I 型

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

if GPR[rs] < 0 then PC ← B_PC + sign_ext(offset)<<2

B_PC：分支跳转参与运算的 PC，在不考虑延迟槽时为分支跳转指令的 PC，考虑延迟槽时为延迟槽指令的 PC，即分支跳转指令的 PC+4。

（30） 装载字节，并作符号扩展

LB rt, offset(base)　　I 型

31 26	25 21	20 16	15 0
100000	base	rt	offset
6	5	5	16

GPR[rt] ← sign(Mem[GPR[base] + sign_ext(offset)])

（31） 装载字节，并作无符号扩展

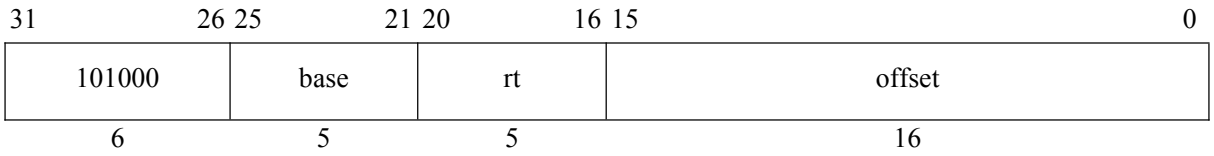
LBU rt, offset(base)　　I 型

31 26	25 21	20 16	15 0
100100	base	rt	offset
6	5	5	16

GPR[rt] ← zero(Mem[GPR[base] + sign_ext(offset)])

（32） 存储字节

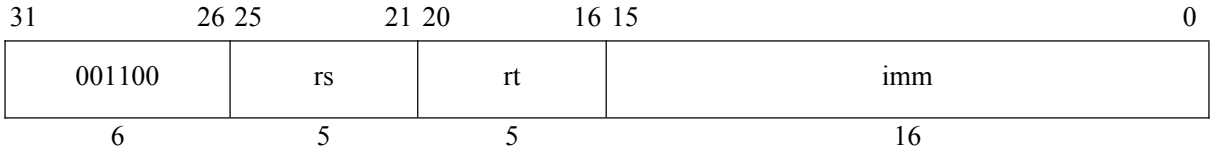
SB rt, offset(base)　　I 型



Mem[GPR[base] + sign_ext(offset)] ← GPR[rt]

（33） 立即数按位与

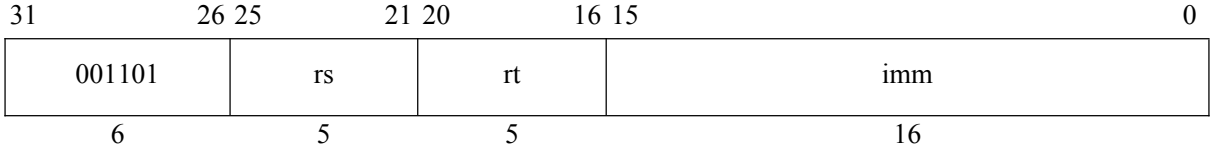
ANDI rt, rs, imm　　I 型



GPR[rt] ← GPR[rs] & zero_ext(imm)

（34） 立即数按位或

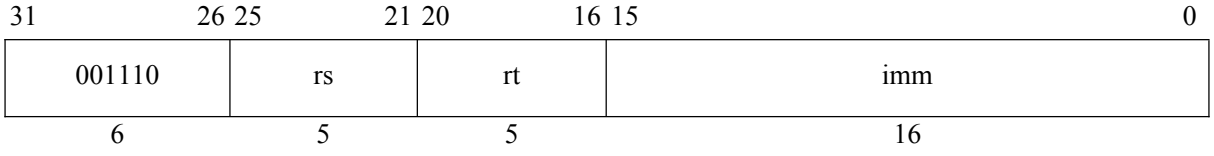
ORI rt, rs, imm　　I 型



GPR[rt] ← GPR[rs] | zero_ext(imm)

（35） 立即数按位异或

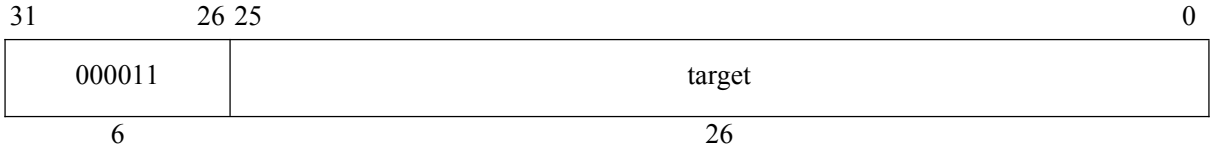
XORI rt, rs, imm　　I 型



GPR[rt] ← GPR[rs] ^ zero_ext(imm)

（36） 跳转和链接

JAL target　　J 型



GPR[31] ← B_PC + 4, PC←{B_PC[31:28], target << 2}

B_PC：分支跳转参与运算的 PC，在不考虑延迟槽时为分支跳转指令的 PC，考虑延迟槽时为延迟槽指令的 PC，即分支跳转指令的 PC+4。

附 A3 静态 5 级流水 CPU 新增实现指令

（37） 有符号字乘法

MULT rs, rt　　R 型

31–26	25–21	20–16	15–6	5–0
000000	rs	rt	00 0000 0000	011000
6	5	5	10	6

(HI, LO) ← sign(GPR[rs]) * sign(GPR[rt])

（38） 从 LO 寄存器取值

MFLO rd　　R 型

31–26	25–21	20–16	15–11	10–6	5–0
000000	00 0000 0000	rd	00000	010010	
6	10	5	5	6	

GPR[rd] ← [LO]

（39） 从 HI 寄存器取值

MFHI rd　　R 型

31–26	25–21	20–16	15–11	10–6	5–0
000000	00 0000 0000	rd	00000	010000	
6	10	5	5	6	

GPR[rd] ← [HI]

（40） 向 LO 寄存器存值

MTLO rs　　R 型

31–26	25–21	20–16	15–6	5–0
000000	rs	000 0000 0000 0000	010011	
6	5	15	6	

[LO] ← GPR[rs]

（41） 向 HI 寄存器存值

MTHI rs　　R 型

31–26	25–21	20–16	15–6	5–0
000000	rs	000 0000 0000 0000	010001	
6	5	15	6	

[HI] ← GPR[rs]

（42） 从协处理器 0 寄存器取值

MFC0 rt, cs.sel　　R 型

31–26	25–21	20–16	15–11	10–3	2–0

010000	00000	rt	cs	00000000	sel
6	5	5	5	8	3

GPR[rt] ← CPR[cs.sel]

（43） 向协处理器 0 寄存器存值

MTC0 rt, cd.sel　　R 型

31 26	25 21	20 16	15 11	10 3	2 0
010000	00100	rt	cd	00000000	sel
6	5	5	5	8	3

CPR[cd.sel] ← GPR[rt]

（44） 系统调用

SYSCALL

31 26	25 6	5 0
000000	code	001100
6	20	6

CPR[14.0] ← PC, CPR[13.0][6:2] ← 01000, CPR[12.0][1] ← 1, PC ←EXC_ENTER_ADDR

EXC_ENTER_ADDR 为例外入口地址，原本应为 CPR[15.1] + 0x180，但在课程设计中为方便编写测试程序，将 EXC_ENTER_ADDR 设置为 0。

（45） 异常返回

ERET

31 26	25	21 20 16 15 11 10 6	5 0
010000	1	000 0000 0000 0000 0000	011000
6	1	19	6

CPR[12.0][1] ← 0, PC ← CPR[14.0]

附录 B 实现的 cp0 寄存器

附 B1 Status 寄存器（CP0 寄存器 12，选择 0）

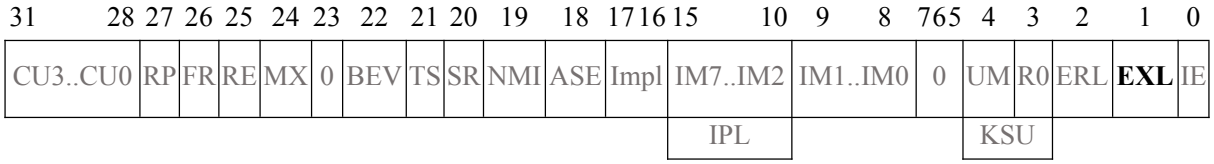


表 B-1 EXL（Exception Level）域

域		描述	读/写	复位状态	规则
名称	位				
EXL	1	Exception 级别；当出现任何除了复位、软复位、NMI 或缓存错误的例外时由处理器置位。	可读/写	Undefined	Required

表 B-2 EXL 编码表

编码	含义
0	正常级别
1	Exception 级别

附 B2 Cause 寄存器（CP0 寄存器 13，选择 0）

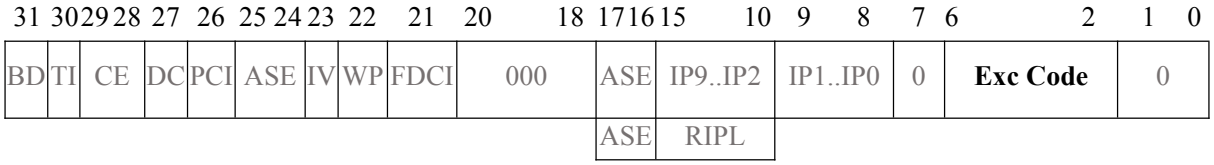


表 B-3 例外编码表

例外编码值		助记符	描述
十进制	十六进制		
0	0x00	Int	中断
1	0x01	Mod	TLB 修正例外
2	0x02	TLBL	TLB 例外（装载或取指）
3	0x03	TLBS	TLB 例外（存储）
4	0x04	AdEL	地址错误例外（装载或取指）
5	0x05	AdES	地址错误例外（存储）
6	0x06	IBE	总线错误例外（取指）
7	0x07	DBE	总线错误例外（数据相关：装载或存储）
8	0x08	Sys	系统调用例外
9	0x09	Bp	断点例外
10	0x0a	RI	保留指令例外

表 B-4 Exc Code（Exception Code）域

域		描述	读/写	复位状态	规则
名称	位				
ExcCode	6..2	例外编码	只读	Undefined	Required

附 B3 例外程序计数器 EPC（CP0 寄存器 14，选择 0）

31 0

EPC

表 B-5 EPC（Exception Program Counter）域

域		描述	读/写	复位状态	规则
Name	**Bits**				
EPC	31..0	例外程序计数器	可读/写	Undefined	Required