

单周期 CPU 设计文档（P4）

贾博驿

初稿：2023 年 11 月 5 日

编辑：2024 年 8 月 26 日

设计实现指令

共 41 条
实际指令 39 条

R 型指令：

6	5	5	5	5	6
opcode	rs	rt	rd	shamt	funct

I 型指令：

6	5	5	16
opcode	rs	rt	imm16

J 型指令：

6	26
opcode	instr_index

注：以下的 RTL 除分支与跳转类指令均省略 $PC \leftarrow PC+4$ 。

算数与逻辑类

1. 算数类

a. R 型

名称	funct	RTL
add	100000	$GPR[rd] \leftarrow GPR[rs] + GPR[rt]$
addu	100001	$GPR[rd] \leftarrow GPR[rs] + GPR[rt]$
sub	100010	$GPR[rd] \leftarrow GPR[rs] - GPR[rt]$
subu	100011	$GPR[rd] \leftarrow GPR[rs] - GPR[rt]$

注：本次要求实现的 add 实际上为 addu、sub 实际上为 subu。故本次实现中二者按照相同指令处理。

b. I 型

名称	opcode	RTL
addiu	001001	$GPR[rt] \leftarrow GPR[rs] + \text{sign_extend}(\text{immediate})$

2. 逻辑类

a. R 型

名称	funct	RTL
and	100100	$GPR[rd] \leftarrow GPR[rs] \text{ AND}(\text{bitwise}) GPR[rt]$
or	100101	$GPR[rd] \leftarrow GPR[rs] \text{ OR}(\text{bitwise}) GPR[rt]$
xor	100110	$GPR[rd] \leftarrow GPR[rs] \text{ XOR}(\text{bitwise}) GPR[rt]$
nor	100111	$GPR[rd] \leftarrow GPR[rs] \text{ NOR}(\text{bitwise}) GPR[rt]$
sll	000000	$GPR[rd] \leftarrow GPR[rt]_{(31-\text{shamt})..0} \parallel 0_{\text{shamt}}$
srl	000010	$GPR[rd] \leftarrow 0_{\text{shamt}} \parallel GPR[rt]_{31..\text{shamt}}$
sra	000011	$GPR[rd] \leftarrow (GPR[rt]_{31})_{\text{shamt}} \parallel GPR[rt]_{31..\text{shamt}}$
sllv	000100	$s \leftarrow GPR[rs]_{4..0} \quad GPR[rd] \leftarrow GPR[rt]_{(31-s)..0} \parallel 0_s$
srlv	000110	$s \leftarrow GPR[rs]_{4..0} \quad GPR[rd] \leftarrow 0_s \parallel GPR[rt]_{31..s}$
srav	000111	$s \leftarrow GPR[rs]_{4..0} \quad GPR[rd] \leftarrow (GPR[rt]_{31})_s \parallel GPR[rt]_{31..s}$

注：根据寄存器值移位指令只依照寄存器值的低五位置位，不受高位值影响。

b. I 型

名称	opcode	RTL
andi	001100	$GPR[rt] \leftarrow GPR[rs] \text{ AND}(\text{bitwise } \text{zero_extend}(\text{immediate}))$
ori	001101	$GPR[rt] \leftarrow GPR[rs] \text{ OR}(\text{bitwise } \text{zero_extend}(\text{immediate}))$
xori	001110	$GPR[rt] \leftarrow GPR[rs] \text{ XOR}(\text{bitwise } \text{zero_extend}(\text{immediate}))$

寄存器操作类

a. R 型

名称	funct	RTL
slt	101010	$GPR[rd] \leftarrow 0_{GPRLEN-1} \parallel GPR[rs] < GPR[rt]$
sltu	101011	$GPR[rd] \leftarrow 0_{GPRLEN-1} \parallel (0 \parallel GPR[rs]) < (0 \parallel GPR[rt])$

注：slt 类指令成立时置 1，不成立时置 0。

b. I 型

名称	opcode	RTL
slti	001010	$GPR[rt] \leftarrow 0_{GPRLEN-1} \parallel GPR[rs] < \text{sign_extend}(\text{immediate})$
sltiu	001011	$GPR[rt] \leftarrow 0_{GPRLEN-1} \parallel (0 \parallel GPR[rs]) < (0 \parallel \text{sign_extend}(\text{immediate}))$
lui	001111	$GPR[rt] \leftarrow \text{immediate} \parallel 0_{16}$

注：sltiu 对立即数的处理是先有符号扩展再进行无符号比较。

分支与跳转类

1. 分支类（I 型）

名称	opcode	rt	RTL
(all)			if condition then $PC \leftarrow PC + 4 + \text{sign_extend}(\text{offset} \parallel 0_2)$ endif
bltz	000001	00000	condition $\leftarrow GPR[rs] < 0_{32}$
bgez	000001	00001	condition $\leftarrow GPR[rs] \geq 0_{32}$
beq	000100	rt	condition $\leftarrow (GPR[rs] = GPR[rt])$
bne	000101	rt	condition $\leftarrow (GPR[rs] \neq GPR[rt])$
blez	000110	00000	condition $\leftarrow GPR[rs] \leq 0_{32}$
bgtz	000111	00000	condition $\leftarrow GPR[rs] > 0_{32}$

2. 跳转类

a. R 型

名称	funct	RTL
jr	001000	$PC \leftarrow GPR[rs]$
jalr	001001	$I: GPR[rd] \leftarrow PC + 4 \quad PC \leftarrow GPR[rs]$

b. J 型

名称	opcode	RTL
j	000010	$PC \leftarrow PC_{31..28} \parallel \text{instr_index} \parallel 0_2$
jal	000011	$GPR[31] \leftarrow PC + 4 \quad PC \leftarrow PC_{31..28} \parallel \text{instr_index} \parallel 0_2$

内存操作类（I 型）

名称	opcode	RTL
(all)		$\text{addr} \leftarrow \text{sign_extend}(\text{offset}) + GPR[\text{base}]$

lb	100000	byte \leftarrow addr1..0 GPR[rt] \leftarrow sign_extend(memory[addr] _{7+8*byte..8*byte})
lh	100001	half \leftarrow addr1 GPR[rt] \leftarrow sign_extend(memory[addr] _{15+8*half..8*half})
lw	100011	GPR[rt] \leftarrow memory[addr]
lbu	100100	byte \leftarrow addr1..0 GPR[rt] \leftarrow zero_extend(memory[addr] _{7+8*byte..8*byte})
lhu	100101	half \leftarrow addr1 GPR[rt] \leftarrow zero_extend(memory[addr] _{15+8*half..8*half})
sb	101000	memory[addr] _{7..0} \leftarrow GPR[rt] _{7..0}
sh	101001	memory[addr] _{15..0} \leftarrow GPR[rt] _{15..0}
sw	101011	memory[addr] _{31..0} \leftarrow GPR[rt]

注：

- 1.针对半字操作需要 **addr** 为 2 的非负整数倍，针对字节操作需要 **addr** 为 4 的非负整数倍，本次实现时实际只涉及针对字节操作，且保证地址是合法的。
- 2.针对字节和半字操作时只修改对应字节和半字的值，同一字的其他部分不变。

模块设计

IFU

1. 描述

取指令模块。内有 PC 子模块、NPC 子模块和 ROM。接收分支和跳转指令的控制信号和数据并实现跳转。输出指令和 PC 寄存器取值。

2. 接口

名称	方向	描述
RESET	I	同步复位信号
clk	I	时钟信号
NPCSel[1:0]	I	NPC 子模块控制信号
BranchComp	I	NPC 子模块分支指令控制信号
offset[15:0]	I	分支指令跳转偏移
instr_index[25:0]	I	j、jal 指令跳转地址
instr_register[31:0]	I	jr、jalr 指令跳转寄存器值
Instr[31:0]	O	取出的当前指令
PC[31:0]	O	当前 PC 寄存器的值

3. 控制信号

BranchComp: 接 ALU 的 Comp。

NPCSel: 接 CTRL 的 NPCSel。

指令	取值	描述
其他	00	$PC \leftarrow PC + 4$
分支类	01	使用 offset, 结合 BranchComp 判断
j、jal	10	使用 instr_index
jr、jalr	11	使用 instr_register

IFU_PC

1. 描述

程序计数器子模块。在复位信号有效时复位为 32'h00003000。

2. 接口

名称	方向	描述
RESET	I	同步复位信号
clk	I	时钟信号
D[31:0]	O	下一指令地址
Q[31:0]	O	当前指令地址

IFU_NPC

1. 描述

下一条指令地址计算子模块。输入 IFU 接收分支跳转指令控制信号和相关数据，计算出下一条指令的地址。

2. 接口

名称	方向	描述
PC[31:0]	1	当前指令地址

NPCSel[1:0]	I	同 IFU
BranchComp	I	同 IFU
offset[15:0]	I	同 IFU
instr_index[25:0]	I	同 IFU
instr_register[31:0]	I	同 IFU
NPC[31:0]	O	下一指令地址

SPL

1. 描述

指令分线器模块。按照三种指令的格式将一条指令分为不同的部分，供其他模块使用。

2. 接口

名称	方向	描述
Instr[31:0]	I	当前指令
opcode[5:0]	O	指令的操作数
rs[4:0]	O	R、I 型指令的 rs
rt[4:0]	O	R、I 型指令的 rt
rd[4:0]	O	R 型指令的 rd
shamt[4:0]	O	R 型指令的 shamt
funct[5:0]	O	R 型指令的 funct
imm16[15:0]	O	I 型指令的立即数
instr_index[25:0]	O	J 型指令的跳转地址

GRF

1. 描述

寄存器堆模块，共有 31 个寄存器（0 号寄存器直接接地）。通过 A1、A2 输入地址，可以取出指定寄存器存储的数据。当 WE 使能时，通过 A3 输入地址，WD 输入数据，可以修改指定寄存器存储的数据。

2. 接口

名称	方向	描述
RESET	I	同步复位信号
clk	I	时钟信号
WE	I	写入使能信号
A1[4:0]	I	读取的第一个寄存器的编号
A2[4:0]	I	读取的第二个寄存器的编号
A3[4:0]	I	写入的寄存器的编号
WD[31:0]	I	写入寄存器的数据
RD1[31:0]	O	读取的第一个寄存器的值
RD2[31:0]	O	读取的第二个寄存器的值

3. 控制信号

WE：接 CTRL 的 GRFWE。

指令	取值	描述
其他	0	不涉及寄存器写入的指令
R 型：除 jr	1	

I 型：算数与逻辑、寄存器操作类	1	
I 型：内存操作类读取类	1	
jal、jalr	1	

4. 关联 MUX 控制信号

GRFA3MUX：接 CTRL 的 GRFA3MUX。

指令	取值	描述
其他	00	固定 0(0x00)，保护
R 型：忽略不使能的 jr	01	接入 rd
I 型：忽略不使能的指令	10	接入 rt
jal	11	固定 31(0x1f)

GRFWDMUX：接 CTRL 的 GRFWDMUX。

指令	取值	描述
其他	00	接入 ALU
lui	01	接入 EXT
I 型：内存操作读取类	10	接入 DM
jalr、jal	11	接入 PC + 4

EXT

1. 描述

位扩展模块，将 16 位立即数无符号扩展、有符号扩展、右补 16 位 0 为 32 位。

2. 接口

名称	方向	描述
EXTSel[1:0]	I	扩展方式控制信号
imm16[15:0]	I	16 位立即数
EXT[31:0]	O	32 位扩展结果

3. 控制信号

EXTSel：接 CTRL 的 EXTSel。

指令	取值	描述
其他	00	有符号拓展
andi、ori、xori	01	无符号拓展
lui	10	右补 16 位 0

ALU

1. 描述

算数逻辑模块，完成算数运算（不考虑算数溢出的加、减）、逻辑运算（与、或、异或、或非）、移位运算（左移、逻辑右移、算数右移）、比较运算（两数相等、不等、有符号小于、无符号小于，一数有符号小于、小于等于、大于、大于等于零）。计算结果用于保存到寄存器、判断分支判断分支指令是否执行。

2. 接口

名称	方向	描述
OPSel[1:0]	I	OP 输出控制信号
FuncSel[2:0]	I	计算方式控制信号
CompSel[2:0]	I	比较部分控制信号

rs[31:0]	I	第一个操作数
rt[31:0]	I	第二个操作数
shamt[4:0]	I	移位立即数
OP[31:0]	O	计算结果
Comp	O	比较结果

3. 控制信号

OPSel: 接 CTRL 的 OPSel。

指令	取值	描述
add、addu、sub、subu、addiu	00	算数运算
I 型: 内存操作类	00	算数运算
and、or、xor、nor、andi、ori、xori	01	逻辑运算
sll、srl、sra、sllv、srlv、srav	10	移位运算
slt、sltu、slti、sltiu	11	比较运算(无符号拓展 Comp 为 32 位)

FuncSel: 接 CTRL 的 FuncSel, 根据 OPSel 选择。

OPSel == 00 时。

指令	取值	描述
add、addu、addiu	000	不考虑算数溢出加法
I 型: 内存操作类读取类	000	不考虑算数溢出加法
sub、subu	001	不考虑算数溢出减法

OPSel == 01 时, 根据指令是否为 I 型选择。R 型接 funct[2:0]、I 型接 opcode[2:0]。

指令	取值	描述
and(100100)、andi(001100)	100	与
or(100101)、ori(001101)	101	或
xor(100110)、xori(001110)	110	异或
nor(100111)	111	或非

OPSel == 10 时, 接 funct[2:0]。

指令	取值	描述
sll(000000)	000	逻辑左移立即数
srl(000010)	010	逻辑右移立即数
sra(000011)	011	算数右移立即数
sllv(000100)	100	逻辑左移寄存器
srlv(000110)	110	逻辑右移寄存器
srav(000111)	111	算数右移寄存器

OPSel == 11 时, FuncSel 无效, 保持默认 000 即可。

CompSel: 接 CTRL 的 CompSel。

指令	取值	描述
bltz(00000)	000	有符号小于 0
bgez(00001)	001	有符号大于等于 0
slt(101010)、slti(001010)	010	有符号两数小于
sltu(101011)、sltiu(001011)	011	无符号两数小于
beq(000100)	100	两数相等
bne(000101)	101	两数不等
blez(000110)	110	有符号小于等于 0

bgtz(000 111)	111	有符号大于 0
-----------------------	------------	---------

4. 关联 MUX 控制信号

ALUrtMUX: 接 CTRL 的 ALUrtMUX。

指令	取值	描述
其他	0	接入 GRF
I 型: 除分支与跳转类	1	接入 EXT

DM

1. 描述

内存模块。包括 RAM 和位处理两个部分。位处理部分用于处理针对半字和字节的读取和写入。

2. 接口

名称	方向	描述
RESET	I	同步复位信号
clk	I	时钟信号
WE	I	写入使能信号
DMSel[2:0]	I	读取写入模式控制信号
A[31:0]	I	读取或写入的地址
D_write[31:0]	I	写入的数据
D_read[31:0]	O	读取的数据

3. 控制信号

WE: 接 CTRL 的 GRFWE。

指令	取值	描述
其他	0	
sb、sh、sw	1	

DMSel: 接 CTRL 的 DMSel, 接 opcode[2:0]。

指令	取值	描述
lb(100 000)、sb(101 000)	000	针对 byte 操作, 读取时有符号拓展
lh(100 001)、sh(101 001)	001	针对 halfword 操作, 读取时有符号拓展
lw(100 011)、sw(101 011)	011	针对 word 操作
lbu(100 100)	100	针对 byte 操作, 读取时无符号拓展
lhu(100 101)	101	针对 halfword 操作, 读取时无符号拓展

CTRL

1. 描述

控制模块。通过 opcode、funct 和 rt 判断指令的类型, 输出各个模块的控制信号。

2. 接口

名称	方向	描述
opcode[5:0]	I	指令的操作数
funct[5:0]	I	R 型指令 funct
rt[4:0]	I	分支指令 rt
OPSel[1:0]	O	接入 ALU
EXTSel[1:0]	O	接入 EXT

DMSel[2:0]	O	接入 DM
NPCSel[1:0]	O	接入 IFU
FuncSel[2:0]	O	接入 ALU
CompSel[2:0]	O	接入 ALU
GRFWE	O	接入 GRF
DMWE	O	接入 DM
GRFA3MUX[1:0]	O	接入 GRF 的 GRFA3MUX
GRFWDMUX[1:0]	O	接入 GRF 的 GRFWDMUX
ALUrtMUX	O	接入 ALU 的 ALUrtMUX