# decoder设计文档

译码单元，根据输入的指令生成相应的信号。

# **输入**

* **ifid\_instr[31:0]**

指令内容，32位，来自ifid流水段寄存器

# **输出**

名字以idex开头的输出信号，会送往idex流水段寄存器，其余以id开头的不送。

* **idex\_mem\_w**

指示该指令是否写存储器；

0：不写

1：写

* **idex\_mem\_r**

指示该指令是否读存储器；

0：不读

1：读

* **idex\_reg\_w**

指示该指令是否写通用寄存器；

0: 不写

1：写

* **idex\_branch,**

指示该指令是否是分支指令；

0：不是

1：是

* **idex\_condition [2:0]**

指示该指令是哪种分支指令；

3'b000: 不是分支指令

3'b001: BEQ

3'b010: BNE

3'b011: BGEZ

3'b100: BGTZ

3'b101: BLEZ

3'b110: BLTZ

* **idex\_B\_sel**

选择ALU的B操作数的来源；

0: 取寄存器中的操作数

1: 取立即数扩展单元的输出

* **idex\_ALU\_op[3:0]**

ALU运算单元的操作码；

4'b0000: 加法（不产生溢出）

4'b0001: 减法（不产生溢出）

4'b0010: 前导零

4'b0011: 前导一

4'b0100: 与

4'b0101: slt/slti

4'b0110: 或

4'b0111: sltu/sltiu

4'b1000: 或非

4'b1001: 异或

4'b1010: seb

4'b1011: seh

4'b1110: 加法（可产生溢出）

4'b1111: 减法（可产生溢出）

* **idex\_shamt[4:0]**

提供给移位器的移位位数之一；

来自指令shamt段，即idex\_shamt = ifid\_instr[10:6]

* **idex\_shamt\_sel**

选择真正送往移位器的移位位数；

0: idex\_shamt

1: rs寄存器内容的低5位

* **idex\_shift\_op[1:0]**

移位器的操作码；

2'b00: 逻辑左移

2'b01: 逻辑右移

2'b10: 算术右移

2'b11: 循环右移; 本次实验中不实现循环移位指令

* **idex\_load\_sel[2:0]**

指示该指令是哪种load指令；

3'b000: LB

3'b001: LBU

3'b010: LH

3'b011: LHU

3'b100: LW

3'b101: LWL

3'b110: LWR

* **idex\_store\_sel[2:0]**

指示该指令是哪种store指令；

3'b000: SB

3'b001: SH

3'b010: SW

3'b011: SWL

3'b100: SWR

* **idex\_of\_w\_disen**

指示若运算结果溢出，是否可写入寄存器；

0：可写

1：不可写

* **idex\_cp0\_dest\_addr[4:0]**

MTC0指令索引cp0寄存器组的编号；

来自指令的rd段，即idex\_cp0\_dest\_addr = ifid\_instr[15:11]

* **idex\_cp0\_w\_en**

指示该指令是否写CP0寄存器；

0: 不写

1：写

* **idex\_syscall**

指示该指令是不是syscall指令

0：不是

1：是

* **idex\_eret**

指示该指令是不是eret指令

0: 不是

1: 是

* **id\_imm\_ext[1:0]**

指示对立即数进行何种扩展；

2'b00: 零扩展

2'b01: 符号扩展

2'b10: LUI扩展，即16'bimm|16'b0（拼接）

2'b11: 输出常数4

* **id\_jr**

指示该指令是不是jr指令；

0: 不是

1: 是

* **id\_jump**

指示该指令是不是j或jal指令；

0: 不是

1: 是

* **id\_rd\_addr\_sel[1:0]**

选择寄存器写操作的目标地址；

2'b00: rt

2'b01: rd

2'b10: 31号寄存器(ra)

* **id\_rt\_addr\_sel**

选择送入通用寄存器组的rt地址；

0: 指令的rt段

1: 0, 即选择0号寄存器

* **id\_rt\_data\_sel**

mfc0指令的源操作数来自cp0，而其他指令的源操作数来自寄存器组(rt)，此信号用于在取自cp0的数据和取自GPR中的数据间进行选择；

0: 寄存器组

1: cp0寄存器

* **id\_cp0\_src\_addr[4:0]**

MFC0指令索引cp0寄存器组的编号；

来自指令的rd段，即idex\_cp0\_dest\_addr = ifid\_instr[15:11]

* **idex\_exres\_sel[2:0]**

选择运算结果传到exmem寄存器；

3'b000: ALU运算结果

3'b001: 移位器运算结果

3'b010: pc+8

3'b011: rs寄存器的内容

3'b100: 乘除法单元的运算结果

* **idex\_movn**

指示该指令是不是movn指令

0: 不是

1: 是

* **idex\_movz**

指示该指令是不是movz指令

0: 不是

1: 是

* **idex\_div\_mul[3:0]**

指示该指令是乘除法相关的哪种指令

4'b0000: 不是

4'b0001: DIV

4'b0010: DIVU

4'b0011: MFHI

4'b0100: MFLO

4'b0101: MTHI

4'b0110: MTLO

4'b0111: MUL

4'b1000: MULT

4'b1001: MULTU

# **相关代码**

1. 每条指令都一样的信号(idex\_shamt, idex\_rs\_addr, idex\_rt\_addr, idex\_cp0\_dest\_addr, id\_cp0\_src\_addr), 使用assign语句进行连线赋值。

assign idex\_shamt = ifid\_instr[10:6];

assign idex\_rs\_addr = ifid\_instr[25:21];

assign idex\_rt\_addr = ifid\_instr[20:16];

assign idex\_cp0\_dest\_addr = ifid\_instr[15:11];

assign id\_cp0\_src\_addr = ifid\_instr[15:11];

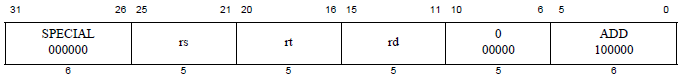
2. 其他由指令所决定的信号，在一个always语句块中依指令内容生成, 每种情况都囊括在内(完整的if-else，完整的case）使得可以利用该always块实现复杂的译码器组合逻辑。

3. decoder模块是纯组合逻辑电路。

# **处理的指令**

这里列出实验中实现的指令及其相关信号，但只指出一些需注意的重要信号而不是指出该条指令对应的所有信号内容。

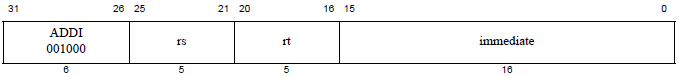
* **ADD**

****

描述：GPR[rd] ← GPR[rs] + GPR[rt]

相关信号：运算结果溢出不可写，故输出的idex\_w\_disen为1.

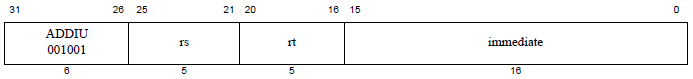
* **ADDI**

****

描述：GPR[rt] ← GPR[rs] + immediate

相关信号：运算结果溢出不可写，故输出的idex\_w\_disen为1；立即数符号扩展，则id\_imm\_ext = 2'b01；立即数参与运算，则idex\_B\_sel = 1；写往rt寄存器，则id\_addr\_sel = 2'b00.

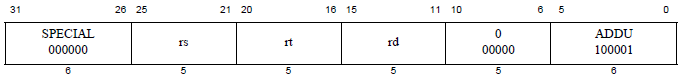
* **ADDIU**

****

描述：GPR[rt] ← GPR[rs] + immediate

相关信号：运算结果溢出可写，则idex\_w\_disen = 0.

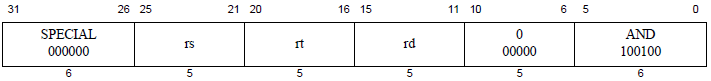
* **ADDU**

****

描述：GPR[rd] ← GPR[rs] + GPR[rt]

相关信号：溢出可写，则idex\_w\_disen = 0.

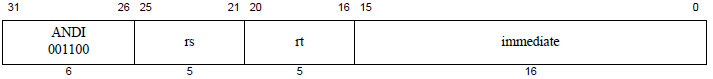
* **AND**

****

描述：GPR[rd] ← GPR[rs] AND GPR[rt]

相关信号：与操作，idex\_ALU\_op = 4'b0100.

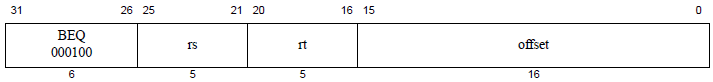
* **ANDI**

****

描述：GPR[rt] ← GPR[rs] AND immediate

相关信号：立即数无符号扩展，则id\_imm\_ext = 2'b00

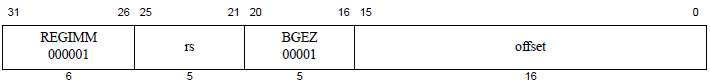
* **BEQ**

****

描述：if GPR[rs] = GPR[rt] then branch

相关信号：这是分支指令(==)，则idex\_branch = 1；idex\_condition = 3'b001；根据减法结果做出判断，则idex\_ALU\_op = 4'b0001(减)；跳转地址是pc+4+立即数(符号扩展)，则 id\_imm\_ext = 2'b01。

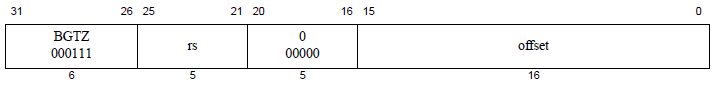
* **BGEZ**

****

描述：if GPR[rs] ≥ 0 then branch

相关信号：这是分支指令(>=0)，则idex\_branch = 1；idex\_condition = 3'b011；与0做比较，则rt选择零号寄存器进行运算, id\_rt\_addr\_sel = 1, id\_rt\_data\_sel = 0.

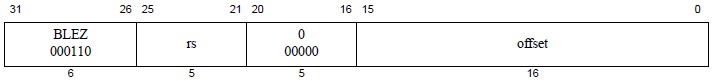
* **BGTZ**

****

描述：if GPR[rs] > 0 then branch

相关信号：这是分支指令(>0)，则idex\_branch = 1；idex\_condition = 3'b100；与0做比较，则rt选择零号寄存器进行运算, id\_rt\_addr\_sel = 1, id\_rt\_data\_sel = 0.

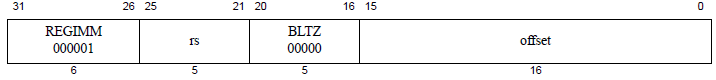
* **BLEZ**

****

描述：if GPR[rs] ≤ 0 then branch

相关信号：这是分支指令(<=0)，则idex\_branch = 1；idex\_condition = 3'b101；与0做比较，则rt选择零号寄存器进行运算, id\_rt\_addr\_sel = 1, id\_rt\_data\_sel = 0.

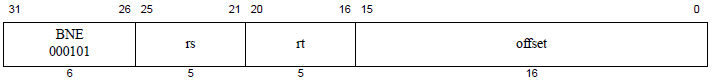
* **BLTZ**

****

描述：if GPR[rs] < 0 then branch

相关信号：这是分支指令(<0)，则idex\_branch = 1；idex\_condition = 3'b110；与0做比较，则rt选择零号寄存器进行运算, id\_rt\_addr\_sel = 1, id\_rt\_data\_sel = 0.

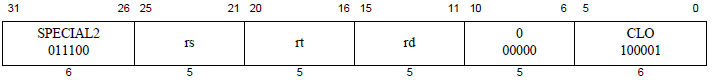
* **BNE**

****

描述：if GPR[rs] ≠ GPR[rt] then branch

相关信号：这是分支指令(!=)，则idex\_branch = 1；idex\_condition = 3'b010；与0做比较，则rt选择零号寄存器进行运算, id\_rt\_addr\_sel = 1, id\_rt\_data\_sel = 0.

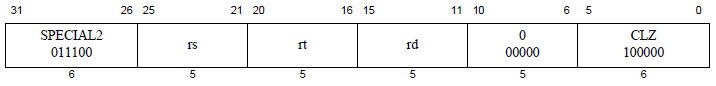
* **CLO**

****

描述：GPR[rd] ← count\_leading\_ones GPR[rs]

相关信号：前导1指令，idex\_ALU\_op = 4'b0011

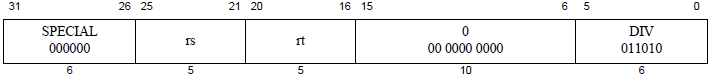
* **CLZ**

****

描述：GPR[rd] ← count\_leading\_zeros GPR[rs]

相关信号：前导0指令，Idex\_ALU\_op = 4'b0000

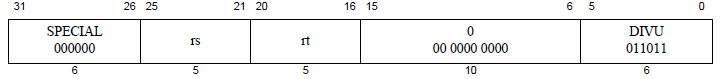
* **DIV**

****

描述：(HI, LO) ← GPR[rs] / GPR[rt], 商写往LO, 余数写往HI

相关信号：除法指令，结果写往hi, lo而不是通用寄存器组，则idex\_reg\_w = 0, idex\_div\_mul = 4'b0001; 溢出可写，idex\_w\_disen = 0

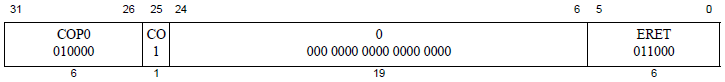
* **DIVU**

****

描述：(HI, LO) ← GPR[rs] / GPR[rt], 商符号扩展后写往LO, 余数符号扩展后写往HI

相关信号：除法指令，结果写往hi, lo而不是通用寄存器组，则idex\_reg\_w = 0, idex\_div\_mul = 4'b0001; 溢出可写，idex\_w\_disen = 0

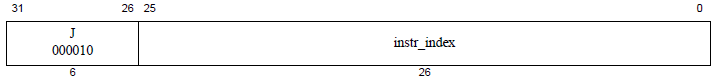
* **ERET**

****

描述：用于回归原来被中断的程序继续执行;eret指令会原子性地把中断响应打开，状态由kernel转到user级，并返回原地址执行。

相关信号：这是eret指令，则idex\_eret = 1

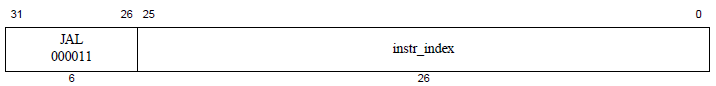
* **J target**

****

描述：PC ← PCGPRLEN-1..28 || instr\_index || 02, 跳转到目标地址

相关信号：这是J指令，则idex\_jump = 1

* **JAL**

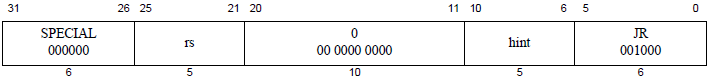
****

描述：GPR[31] ← PC + 8

PC ← PCGPRLEN-1..28 || instr\_index || 02

相关信号：将pc+8送往31号寄存器, 则idex\_reg\_w = 1, id\_rd\_addr\_sel = 2'b10； 实验中pc+8由已有的pc+4再加上立即数扩展单元输出的4得到，则id\_imm\_ext = 2'b11；这是jal指令，则id\_jump = 1

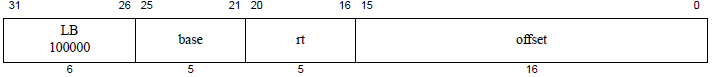
* **JR**

****

描述：PC ← GPR[rs]

相关信号：这是jr指令，id\_jr = 1

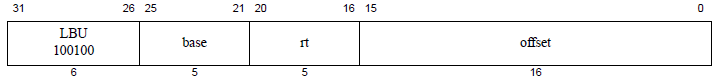
* **LB**

****

描述：GPR[rt] ← memory[GPR[base] + offset]

相关信号：读内存写rt寄存器，idex\_mem\_r = 1, idex\_mem\_w = 0, idex\_reg\_w = 1, id\_rd\_addr\_sel = 2'b00; 地址由即地址与偏移量（立即数扩展）相加而得，则idex\_ALU\_op = 4'b0000(加), id\_imm\_ext = 2'b01；这是LB指令，idex\_load\_sel = 3'b000.

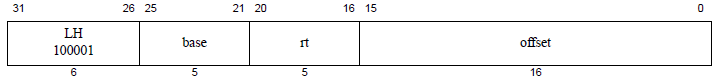
* **LBU**

****

描述：GPR[rt] ← memory[GPR[base] + offset]

相关信号：读内存写rt寄存器，idex\_mem\_r = 1, idex\_mem\_w = 0, idex\_reg\_w = 1, id\_rd\_addr\_sel = 2'b00; 地址由即地址与偏移量（立即数扩展）相加而得，则idex\_ALU\_op = 4'b0000(加), id\_imm\_ext = 2'b01；这是LBU指令，idex\_load\_sel = 3'b001.

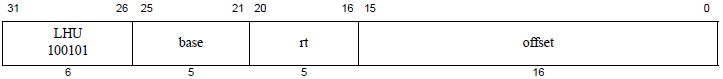
* **LH**

****

描述：GPR[rt] ← memory[GPR[base] + offset]

相关信号：读内存写rt寄存器，idex\_mem\_r = 1, idex\_mem\_w = 0, idex\_reg\_w = 1, id\_rd\_addr\_sel = 2'b00; 地址由即地址与偏移量（立即数扩展）相加而得，则idex\_ALU\_op = 4'b0000(加), id\_imm\_ext = 2'b01；这是LH指令，idex\_load\_sel = 3'b010.

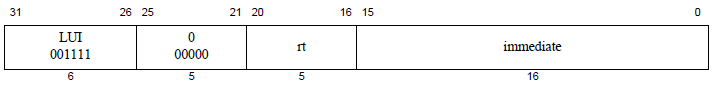
* **LHU**

****

描述：GPR[rt] ← memory[GPR[base] + offset]

相关信号：读内存写rt寄存器，idex\_mem\_r = 1, idex\_mem\_w = 0, idex\_reg\_w = 1, id\_rd\_addr\_sel = 2'b00; 地址由即地址与偏移量（立即数扩展）相加而得，则idex\_ALU\_op = 4'b0000(加), id\_imm\_ext = 2'b01；这是LHU指令，idex\_load\_sel = 3'b011.

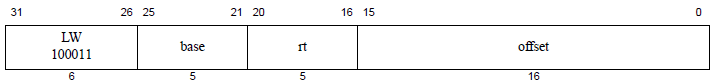
* **LUI**

****

描述：GPR[rt] ← immediate || 016

相关信号：立即数lui扩展，id\_imm\_ext = 2'b10；结果写往rt寄存器，则id\_rd\_addr\_sel = 2'b00

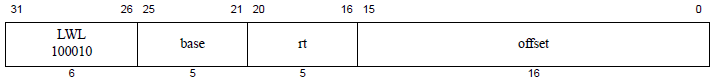
* **LW**

****

描述：GPR[rt] ← memory[GPR[base] + offset]

相关信号：读内存写rt寄存器，idex\_mem\_r = 1, idex\_mem\_w = 0, idex\_reg\_w = 1, id\_rd\_addr\_sel = 2'b00; 地址由即地址与偏移量（立即数扩展）相加而得，则idex\_ALU\_op = 4'b0000(加), id\_imm\_ext = 2'b01；这是LW指令，idex\_load\_sel = 3'b100.

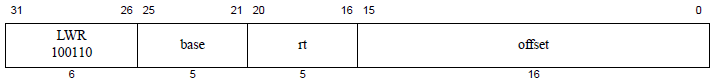
* **LWL**

****

描述：GPR[rt] ← GPR[rt] MERGE memory[GPR[base] + offset]

相关信号：读内存写rt寄存器，idex\_mem\_r = 1, idex\_mem\_w = 0, idex\_reg\_w = 1, id\_rd\_addr\_sel = 2'b00; 地址由即地址与偏移量（立即数扩展）相加而得，则idex\_ALU\_op = 4'b0000(加), id\_imm\_ext = 2'b01；这是LWL指令，idex\_load\_sel = 3'b101.

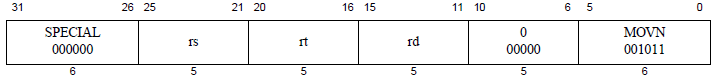
* **LWR**

****

描述：GPR[rt] ← GPR[rt] MERGE memory[GPR[base] + offset]

相关信号：读内存写rt寄存器，idex\_mem\_r = 1, idex\_mem\_w = 0, idex\_reg\_w = 1, id\_rd\_addr\_sel = 2'b00; 地址由即地址与偏移量（立即数扩展）相加而得，则idex\_ALU\_op = 4'b0000(加), id\_imm\_ext = 2'b01；这是LWR指令，idex\_load\_sel = 3'b110.

* **MOVN**

****

描述：if GPR[rt] ≠ 0 then GPR[rd] ← GPR[rs]

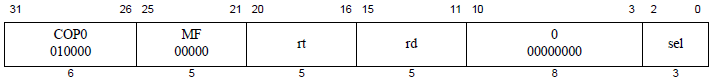
相关信号：这是movn指令，idex\_movn = 1

* **MOVZ**

描述：if GPR[rt] = 0 then GPR[rd] ← GPR[rs]

相关信号：这是movz指令，idex\_movz = 1

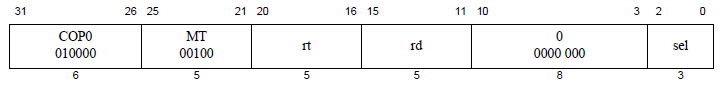
* **MFC0**

****

描述：GPR[rt] ← CPR[0,rd,sel]

相关信号：CP0的内容送往rt，idex\_reg\_w = 1, id\_rd\_addr\_sel = 2'b00, id\_rt\_data\_sel = 0(cp0)

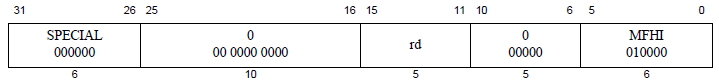
* **MTC0**

****

描述：CPR[0, rd, sel] ← GPR[rt]

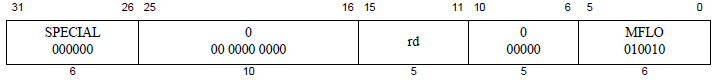
相关信号：rt的内容送往CP0，写CP0信号有效idex\_cp0\_w\_en = 1, idex\_reg\_w = 0, id\_rt\_data\_sel = 1(rt);

* **MFHI**

****描述：GPR[rd] ← HI

相关信号：写rd寄存器，idex\_reg\_w = 1, id\_rd\_addr\_sel = 2'b01; 这是MFHI指令，idex\_div\_mul = 4'b0011

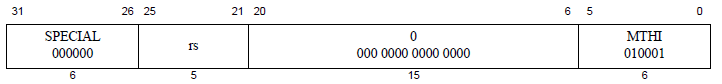
* **MFLO**

****

描述：GPR[rd] ← LO

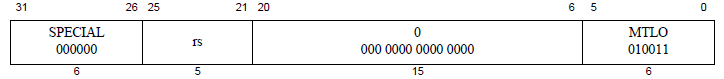
相关信号：写rd寄存器，idex\_reg\_w = 1, id\_rd\_addr\_sel = 2'b01; 这是MFHI指令，idex\_div\_mul = 4'b0100

* **MTHI**

****

描述：HI ← GPR[rs]

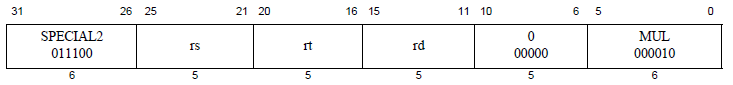
相关信号：写hi寄存器而不是通用寄存器组，idex\_reg\_w = 0; 这是MTHI指令，idex\_div\_mul = 4'b0101

* **MTLO**

描述：LO ← GPR[rs]

相关信号：写lo寄存器而不是通用寄存器组，idex\_reg\_w = 0; 结果取自乘除法单元，idex\_exres\_sel = 3'b10; 这是MTHI指令，idex\_div\_mul = 4'b0110

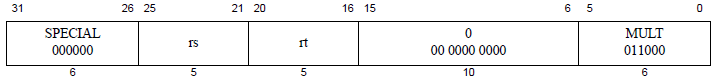
* **MUL**

****

描述：GPR[rd] ← GPR[rs] × GPR[rt]

相关信号：写rd寄存器，idex\_reg\_w = 1, 结果取自乘除法单元，idex\_exres\_sel = 3'b10; 这是MUL指令，idex\_div\_mul = 4'b0111

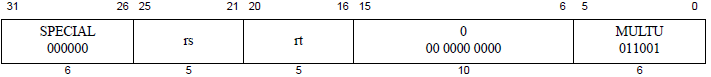
* **MULT**

****

描述：(HI, LO) ← GPR[rs] × GPR[rt]

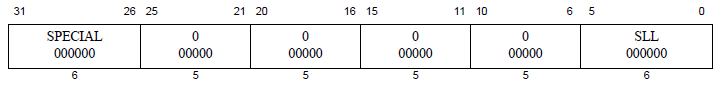
相关信号：写hi,lo而不是通用寄存器，idex\_reg\_w=0; 这是MULT指令, idex\_div\_mul = 4'b1000

* **MULTU**

****

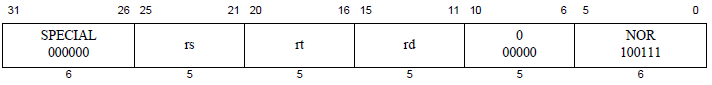
描述：(HI, LO) ← GPR[rs] × GPR[rt]

* 相关信号：写hi,lo而不是通用寄存器，idex\_reg\_w=0; 这是MULT指令, idex\_div\_mul = 4'b1001
* **NOP**

****

描述：空指令，无任何实际操作

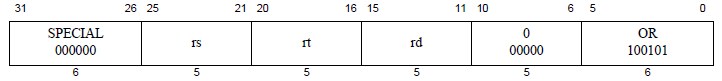
* **NOR**

****

描述：GPR[rd] ← GPR[rs] NOR GPR[rt]

相关信号：idex\_ALU\_op = 4'b1001

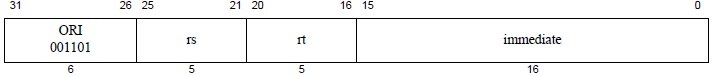
* **OR**

****

描述：GPR[rd] ← GPR[rs] or GPR[rt]

相关信号：idex\_ALU\_op = 4'b0110

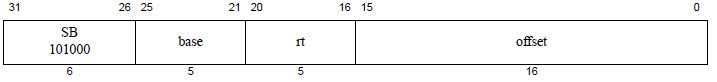
* **ORI**

****

描述: GPR[rt] ← GPR[rs] or immediate

相关信号：idex\_ALU\_op = 4'b0110; 立即数0扩展，id\_imm\_ext = 0; 结果送rt, id\_rd\_addr\_sel = 0

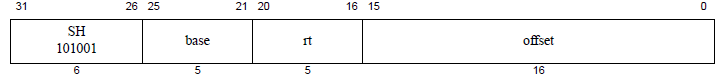
* **SB**

****

描述：memory[GPR[base] + offset] ← GPR[rt]

相关信号：将rt寄存器内容写内存，idex\_mem\_w = 1, idex\_mem\_r = 0, idex\_reg\_w = 0,; 地址由即地址与偏移量（立即数扩展）相加而得，则idex\_ALU\_op = 4'b0000(加), id\_imm\_ext = 2'b01；这是SB指令，idex\_store\_sel = 3'b000.

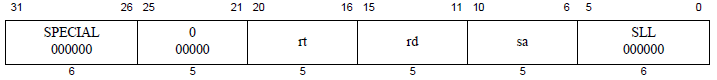
* **SH**

****

描述：memory[GPR[base] + offset] ← GPR[rt]

相关信号：将rt寄存器内容写内存，idex\_mem\_w = 1, idex\_mem\_r = 0, idex\_reg\_w = 0,; 地址由即地址与偏移量（立即数扩展）相加而得，则idex\_ALU\_op = 4'b0000(加), id\_imm\_ext = 2'b01；这是SH指令，idex\_store\_sel = 3'b001.

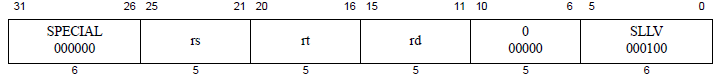
* **SLL**

****

描述：SLL rd, rt, sa

相关信号：逻辑左移，idex\_shift\_op = 2'b00; 移位位数取自指令shamt段idex\_shamt\_sel = 0

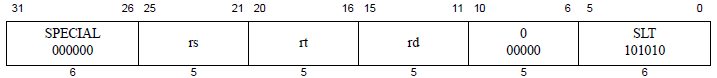
* **SLLV**

****

描述：GPR[rd] ← GPR[rt] << rs

相关信号：逻辑左移，idex\_shift\_op = 2'b00; 移位位数取自rs寄存器低5位idex\_shamt\_sel = 1

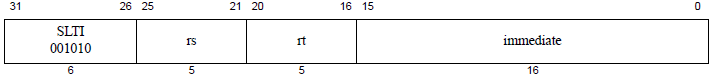
* **SLT**

****

描述：GPR[rd] ← (GPR[rs] < GPR[rt])

相关信号：相关信号：idex\_ALU\_op = 4'b0101; 立即数符号扩展，id\_imm\_ext = 1, idex\_B\_sel = 1; 运算结果送rt, id\_rd\_addr\_sel = 0

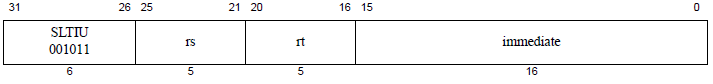
* **SLTI**

****

描述：GPR[rt] ← (GPR[rs] < immediate)

相关信号：idex\_ALU\_op = 4'b0101; 立即数符号扩展，id\_imm\_ext = 1, idex\_B\_sel = 1; 运算结果送rt, id\_rd\_addr\_sel = 0

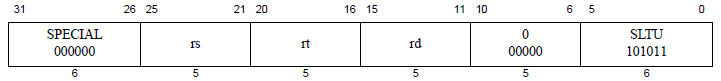
* **SLTIU**

****

描述：GPR[rt] ← (GPR[rs] < immediate)

相关信号：idex\_ALU\_op = 4'b0111; 立即数符号扩展，id\_imm\_ext = 1, idex\_B\_sel = 1; 运算结果送rt, id\_rd\_addr\_sel = 0

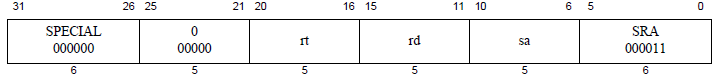
* **SLTU**

****

描述：GPR[rd] ← (GPR[rs] < GPR[rt])

相关信号：idex\_ALU\_op = 4'b0111

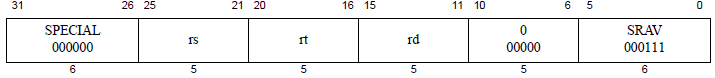
* **SRA**

****

描述: GPR[rd] ← GPR[rt] >> sa (arithmetic)

相关信号：逻辑右移，idex\_shift\_op = 2'b10, 移位位数取自指令shamt段，idex\_shamt\_sel = 0

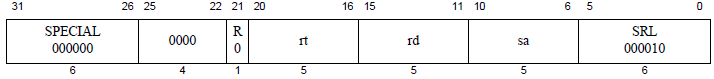
* **SRAV**

****

描述：GPR[rd] ← GPR[rt] >> rs (arithmetic)

相关信号：算术右移，idex\_shift\_op = 2'b01; 移位位数取自rs寄存器低5位idex\_shamt\_sel = 1

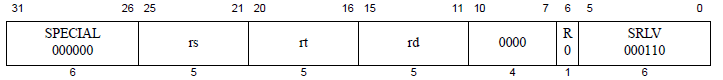
* **SRL**

****

描述：GPR[rd] ← GPR[rt] >> sa (logical)

相关信号：逻辑右移，idex\_shift\_op = 2'b01, 移位位数取自指令shamt段，idex\_shamt\_sel = 0

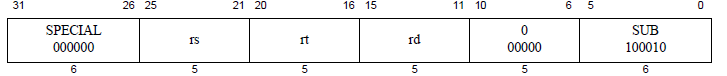
* **SRLV**

****

描述：GPR[rd] ← GPR[rt] >> GPR[rs] (logical)

相关信号：逻辑右移，idex\_shift\_op = 2'b01; 移位位数取自rs寄存器低5位idex\_shamt\_sel = 1

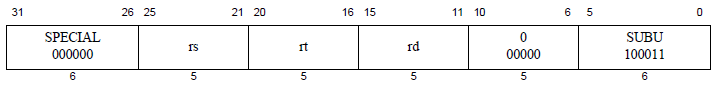
* **SUB**

****

描述：GPR[rd] ← GPR[rs] − GPR[rt]

相关信号：减法，idex\_ALU\_op = 4'b0001, 溢出不可写, idex\_w\_disen = 1

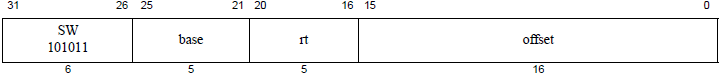
* **SUBU**

****

描述：GPR[rd] ← GPR[rs] − GPR[rt]

相关信号：溢出可写，idex\_w\_disen = 0

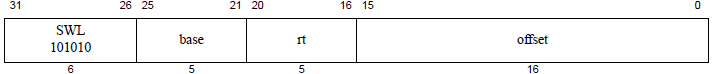
* **SW**

****

描述：memory[GPR[base] + offset] ← GPR[rt]

相关信号：将rt寄存器内容写内存，idex\_mem\_w = 1, idex\_mem\_r = 0, idex\_reg\_w = 0,; 地址由即地址与偏移量（立即数扩展）相加而得，则idex\_ALU\_op = 4'b0000(加), id\_imm\_ext = 2'b01；这是SWL指令，idex\_store\_sel = 3'b011.

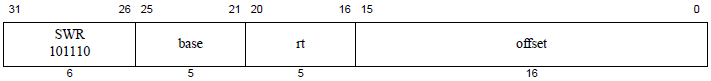
* **SWL**

****

描述：memory[GPR[base] + offset] ← GPR[rt]

相关信号：将rt寄存器内容写内存，idex\_mem\_w = 1, idex\_mem\_r = 0, idex\_reg\_w = 0,; 地址由即地址与偏移量（立即数扩展）相加而得，则idex\_ALU\_op = 4'b0000(加), id\_imm\_ext = 2'b01；这是SWL指令，idex\_store\_sel = 3'b011.

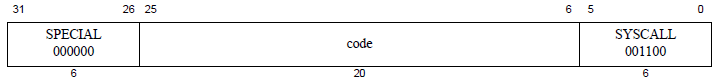
* **SWR**

****

描述：memory[GPR[base] + offset] ← GPR[rt]

相关信号：将rt寄存器内容写内存，idex\_mem\_w = 1, idex\_mem\_r = 0, idex\_reg\_w = 0,; 地址由即地址与偏移量（立即数扩展）相加而得，则idex\_ALU\_op = 4'b0000(加), id\_imm\_ext = 2'b01；这是SWR指令，idex\_store\_sel = 3'b100.

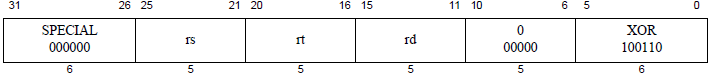
* **SYSCALL**

****

描述：系统调用

相关信号：idex\_syscall = 1

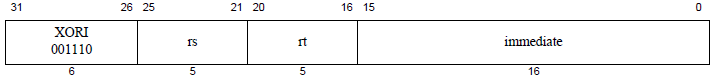
* **XOR**

****

描述：GPR[rd] ← GPR[rs] XOR GPR[rt]

相关信号：异或操作，idex\_ALU\_op = 4'b1001

* **XORI**

****

描述：GPR[rt] ← GPR[rs] xor zero\_extend(immediate)

相关信号：立即数０扩展，id\_imm\_ext = 0；写rt寄存器，idex\_reg\_w = 1, id\_rd\_addr\_sel = 0