设计者学号：

设计者姓名：

1. **ALU设计（需要设计者完成）**

为了设计ALU的控制信号，首先根据指令功能进行归纳总结，完成下表。

|  |  |  |  |
| --- | --- | --- | --- |
|  | 指令 | 操 作 | ALU功能 |
| 算术运算 | ADD.W | ADD.W将通用寄存器rj中的数据加上通用寄存器rk中的数据，所得结果的[31:0]位写入通用寄存器rd中。 | 加 |
| SUB.W | SUB.W 将通用寄存器rj中的数据减去通用寄存器rk中的数据，所得结果的[31:0]位写入通用寄存器rd中 | 减 |
| MUL.W | MUL.W将通用寄存器rj中的数据与通用寄存器rk中的数据进行相乘，乘积结果的[31:0]位数据写入通用寄存器rd中。 | 乘 |
| ADDI.W | ADDI.W 将通用寄存器rj中的数据加上12比特立即数si12符号扩展后的32位数据，所得结果写入通用寄存器rd中。 | 加 |
| LU12I.W | LU12I.W 将 20 比特立即数si20最低位连接上12比特后写入通用寄存器rd中。 | 左移十二位 |
| AND | AND将通用寄存器rj中的数据与通用寄存器rk中的数据进行按位逻辑与运算，结果写入通用寄存器rd 中。 | 与 |
| OR | OR将通用寄存器rj中的数据与通用寄存器rk中的数据进行按位逻辑或运算，结果写入通用寄存器rd中 | 或 |
| NOR | NOR将通用寄存器rj中的数据与通用寄存器rk中的数据进行按位逻辑或非运算，结果写入通用寄存器  rd 中。 | 或非 |
| XOR | XOR将通用寄存器rj中的数据与通用寄存器rk中的数据进行按位逻辑异或运算，结果写入通用寄存器rd 中。 | 异或 |
| ANDI | ANDI 将通用寄存器rj中的数据与12比特立即数零扩展之后的数据进行按位逻辑与运算，结果写入通用寄存器rd中。 | 与 |
| ORI | ORI 将通用寄存器rj中的数据与12比特立即数零扩展之后的数据进行按位逻辑或运算，结果写入通用  寄存器rd中。 | 或 |
| XORI | XORI 将通用寄存器rj中的数据与12比特立即数零扩展之后的数据进行按位逻辑异或运算，结果写入  通用寄存器rd中。 | 异或 |
| 置位运算 | SLT | SLT 将通用寄存器rj中的数据与通用寄存器rk中的数据视作有符号整数进行大小比较，如果前者小于  后者，则将通用寄存器rd的值置为1，否则置为0。 | 有符号小于置位 |
| SLTU | SLTU 将通用寄存器rj中的数据与通用寄存器rk中的数据视作无符号整数进行大小比较，如果前者小  于后者，则将通用寄存器rd的值置为1，否则置为0。 | 无符号小于置位 |
| SLTI | SLTI 将通用寄存器rj中的数据与12比特立即数si12符号扩展后所得的数据视作有符号整数进行大小  比较，如果前者小于后者，则将通用寄存器rd的值置为1，否则置为0 | 有符号小于置位 |
| SLTUI | SLTUI将通用寄存器rj中的数据与12比特立即数si12符号扩展后所得的数据视作无符号整数进行大小  比较，如果前者小于后者，则将通用寄存器rd的值置为1，否则置为0 | 无符号小于置位 |
| 移位运算 | SLL.W | SLL.W 将通用寄存器rj中的数据逻辑左移，移位结果写入通用寄存器rd中。 | 逻辑左移 |
| SRL.W | SRL.W 将通用寄存器rj中的数据逻辑右移，移位结果写入通用寄存器rd中。 | 逻辑右移 |
| SRA.W | SRA.W 将通用寄存器rj中的数据算术右移，移位结果写入通用寄存器rd中 | 算术右移 |
| SLLI.W | SLLI.W 将通用寄存器rj中的数据逻辑左移，移位结果写入通用寄存器rd中。 | 逻辑左移 |
| SRLI.W | SRLI.W 将通用寄存器rj中的数据逻辑右移，移位结果写入通用寄存器rd中。 | 逻辑右移 |
| SRAL.W | SRAI.W 将通用寄存器rj中的数据算术右移，移位结果写入通用寄存器rd中。 | 算术右移 |
| 转移指令 | BEQ | BEQ将通用寄存器rj和通用寄存器rd的值进行比较，如果两者相等则跳转到目标地址，否则不跳转。 | 减 |
| BNE | BNE 将通用寄存器rj和通用寄存器rd的值进行比较，如果两者不等则跳转到目标地址，否则不跳转。 | 减 |
| BLT | BLT 将通用寄存器rj和通用寄存器rd的值视作有符号数进行比较，如果前者小于后者则跳转到目标地  址，否则不跳转。 | 减 |
| BGE | BGE将通用寄存器rj和通用寄存器rd的值视作有符号数进行比较，如果前者大于等于后者则跳转到目  标地址，否则不跳转。 | 减 |
| BLTU | BLTU 将通用寄存器rj和通用寄存器rd的值视作无符号数进行比较，如果前者小于后者则跳转到目标  地址，否则不跳转。 | 减 |
| BGEU | BGEU将通用寄存器rj和通用寄存器rd的值视作无符号数进行比较，如果前者大于等于后者则跳转到  目标地址，否则不跳转。 | 减 |
| B | B 无条件跳转到目标地址处。其跳转目标地址是将指令码中的26比特立即数offs26逻辑左移2位后再  符号扩展，所得的偏移值加上该分支指令的PC。 | 无 |
| BL | BL 无条件跳转到目标地址处，同时将该指令的PC值加4的结果写入到1号通用寄存器r1中。  该指令的跳转目标地址是将指令码中的26比特立即数offs26逻辑左移2位后再符号扩展，所得的偏移  值加上该分支指令的PC。 | 无 |
| JIRL | JIRL 无条件跳转到目标地址处，同时将该指令的PC值加4的结果写入到通用寄存器rd中。  该指令的跳转目标地址是将指令码中的16比特立即数offs16逻辑左移2位后再符号扩展，所得的偏移  值加上通用寄存器rj中的值。 | 无 |
| 访存指令 | LD.W | LD.W从内存取回一个字的  数据写入通用寄存器rd。 | 加 |
| LD.B | LD.B一个字节的数据符号扩展后写入通用寄存器rd | 加 |
| ST.W | ST.W 将通用寄存器rd中的[31:0]位数据写入到内存中。 | 加 |
| ST.B | ST.B 将通用寄存器rd中的[7:0]位数据写入到内存中。 | 加 |

# ALU的控制信号设计

从上面表来看，一共 12种操作，使用 二进制，在下表中给出定义。

解释：对于置位运算本质上利用减法输出的标志位判断大小后决定ALU结果是否置位

而对于条件转移指令则利用减法输出的标志位在分支判断模块判断是否分支跳转

|  |  |
| --- | --- |
| 操作 | 编码 |
| 加 | 0001 |
| 减 | 0010 |
| 乘 | 0011 |
| 与 | 0100 |
| 或 | 0101 |
| 或非 | 0110 |
| 异或 | 0111 |
| 逻辑左移 | 1000 |
| 逻辑右移 | 1001 |
| 算术右移 | 1010 |
| 无符号小于置位 | 1011 |
| 有符号小于置位 | 1100 |

1. **控制单元分成两个部分，一个是cu，一个是alucontrol。**
2. 控制信号定义（仿照教材，所以直接给出）

|  |  |  |
| --- | --- | --- |
| 信号名 | “0”时定义 | “1”时定义 |
| RegRead | 读寄存器来自rk | 读寄存器来自rd |
| RegDst | 写入寄存器来自rd | 写入寄存器来自1号寄存器 |
| RegWrite | 无 | 往寄存器里写数据 |
| ALUSrc | 第二个操作数来自寄存器 | 第二个操作数是立即数扩展 |
| MemRead | 无 | 读存储器 |
| MemWrite | 无 | 写存储器 |
| MemByte | 读取或者写入四个字节 | 读取或者写入单个字节 |

注解：对于扩展的原始位数不同所以截取最长的字段IM[25:5]放入扩展模块

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 信号名 | 00 | 01 | 10 | 11 |
| MemtoReg | ALU输出作为结果寄存器输入 | 存储器输出作为结果寄存器输入 | Pc+4作为寄存器输入 | 无 |
| Extension | IM[21:10]立即数零扩展 | IM[21:10]立即数符号扩展 | IM[25:10]立即数offset左移2位并进行有符号扩展 | IM[24:5]零扩展 |
| ui5From | 来自GR[rk][4:0] | 来自ui5字段 | 5’d12 |  |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 信号名 | 000 | 001 | 010 | 011 | 100 | 101 | 110 | 111 |
| PCSrc | PC <-- PC+4 | PC <-- 相等条件跳转分支地址 | PC <-- 不相等条件跳转分支地址 | PC <--小于条件跳转分支地址 | PC <-- 大于等于条件跳转分支地址 | PC <--无符号小于条件跳转分支地址 | PC <-- 无符号大于等于条件跳转分支地址 | PC <-- 无条件间接跳转分支地址（JIRL） |

|  |  |  |
| --- | --- | --- |
| 多选器控制 | 源 | 解释 |
| ForwardA=00 | ID/EX | 第一个ALU操作数来自寄存器堆 |
| ForwardA=10/11 | EX/MEM | 第一个ALU操作数由上一个ALU运算结果转发获得 |
| ForwardA=01 | MEM/WB | 第一个ALU操作数从数据存储器或者前面的ALU结果中转发获得 |
| ForwardB=00 | ID/EX | 第二个ALU操作数来自寄存器堆 |
| ForwardB=10/11 | EX/MEM | 第二个ALU操作数由上一个ALU运算结果转发获得 |
| ForwardB=01 | MEM/WB | 第二个ALU操作数从数据存储器或者前面的ALU结果中转发获得 |

1. 针对指令的cu输入/输出真值表

将IM[[31:15]作为操作码

对于3R-type的指令（IM[31:20]==12’b 0 0 0 0 0 0 0 0 0 0 0 1）将function字段定义为IM[19:15]进行二级译码

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 指令 | RegRead | RegDst | RegWrite | ALUSrc | PCSrc | MemRead | MemWrite | MemtoReg | Ui5From | MemByte | Extension | ALUop |
| ADD.W | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 00 | 00 | 0 | 0 | 0 |
| SUB.W | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 00 | 00 | 0 | 0 | 0 |
| MUL.W | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 00 | 00 | 0 | 0 | 0 |
| AND | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 00 | 00 | 0 | 0 | 0 |
| OR | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 00 | 00 | 0 | 0 | 0 |
| NOR | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 00 | 00 | 0 | 0 | 0 |
| XOR | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 00 | 00 | 0 | 0 | 0 |
| SLL.W | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 00 | 00 | 0 | 0 | 0 |
| SRL.W | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 00 | 00 | 0 | 0 | 0 |
| SRA.W | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 00 | 00 | 0 | 0 | 0 |
| LU12I.W | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 00 | 10 | 0 | 11 | 1000 |
| ADDI.W | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 00 | 00 | 0 | 01 | 0001 |
| ANDI | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 00 | 00 | 0 | 0 | 0100 |
| ORI | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 00 | 00 | 0 | 0 | 0101 |
| XORI | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 00 | 00 | 0 | 0 | 0111 |
| SLLI.W | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 00 | 01 | 0 | 0 | 1000 |
| SRLI.W | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 00 | 01 | 0 | 0 | 1001 |
| SRAI.W | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 00 | 01 | 0 | 0 | 1010 |
| SLT | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 00 | 00 | 0 | 0 | 1100 |
| SLTU | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 00 | 00 | 0 | 0 | 1011 |
| SLTI | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 00 | 00 | 0 | 01 | 1100 |
| SLTUI | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 00 | 00 | 0 | 0 | 1011 |
| BEQ | 1 | 0 | 0 | 0 | 001 | 0 | 0 | 00 | 00 | 0 | 10 | 0010 |
| BNE | 1 | 0 | 0 | 0 | 010 | 0 | 0 | 00 | 00 | 0 | 10 | 0010 |
| BLT | 1 | 0 | 0 | 0 | 011 | 0 | 0 | 00 | 00 | 0 | 10 | 0010 |
| BGE | 1 | 0 | 0 | 0 | 100 | 0 | 0 | 00 | 00 | 0 | 10 | 0010 |
| BLTU | 1 | 0 | 0 | 0 | 101 | 0 | 0 | 00 | 00 | 0 | 10 | 0010 |
| BGEU | 1 | 0 | 0 | 0 | 110 | 0 | 0 | 00 | 00 | 0 | 10 | 0010 |
| B | 0 | 0 | 0 | 0 | 000 | 0 | 0 | 00 | 00 | 0 | 0 | 0 |
| BL | 0 | 1 | 1 | 0 | 000 | 0 | 0 | 10 | 00 | 0 | 0 | 0 |
| JIRL | 0 | 0 | 1 | 1 | 111 | 0 | 0 | 10 | 00 | 10 | 0 | 0001 |
| LD.W | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 01 | 00 | 0 | 0 | 0001 |
| LD.B | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 01 | 00 | 1 | 0 | 0001 |
| ST.W | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 00 | 00 | 0 | 0 | 0001 |
| ST.B | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 00 | 00 | 1 | 0 | 0001 |

1. 针对指令的aluctrl输入/输出真值表

除了3R-type指令其他指令的ALUop就是ALU控制码

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 指令 | 类型 | ALUop | Function字段 | ALU操作 | ALU控制码 |
| ADD.W |  | 0 | 0 0 0 0 0 | 加 | 0001 |
| SUB.W |  | 0 | 0 0 0 1 0 | 减 | 0010 |
| MUL.W |  | 0 | 1 1 0 0 0 | 乘 | 0011 |
| AND |  | 0 | 0 1 0 0 1 | 与 | 0100 |
| OR |  | 0 | 0 1 0 1 0 | 或 | 0101 |
| NOR |  | 0 | 0 1 0 0 0 | 或非 | 0110 |
| XOR |  | 0 | 0 1 0 1 1 | 异或 | 0111 |
| SLL.W |  | 0 | 0 1 1 1 0 | 逻辑左移 | 1000 |
| SRL.W |  | 0 | 0 1 1 1 1 | 逻辑右移 | 1001 |
| SRA.W |  | 0 | 1 0 0 0 0 | 算术右移 | 1010 |
| LU12I.W |  | 1000 | \*\*\*\*\* | 左移12位 | 1000 |
| ADDI.W |  | 0001 | \*\*\*\*\* | 加 | 0001 |
| ANDI |  | 0100 | \*\*\*\*\* | 与 | 0100 |
| ORI |  | 0101 | \*\*\*\*\* | 或 | 0101 |
| XORI |  | 0111 | \*\*\*\*\* | 异或 | 0111 |
| SLLI.W |  | 1000 | \*\*\*\*\* | 逻辑左移 | 1000 |
| SRLI.W |  | 1001 | \*\*\*\*\* | 逻辑右移 | 1001 |
| SRAI.W |  | 1010 | \*\*\*\*\* | 算术右移 | 1010 |
| SLT |  | 1100 | \*\*\*\*\* | S小于置位 | 1100 |
| SLTU |  | 1011 | \*\*\*\*\* | U小于置位 | 1011 |
| SLTI |  | 1100 | \*\*\*\*\* | S小于置位 | 1100 |
| SLTUI |  | 1011 | \*\*\*\*\* | U小于置位 | 1011 |
| BEQ |  | 0010 | \*\*\*\*\* | 减 | 0010 |
| BNE |  | 0010 | \*\*\*\*\* | 减 | 0010 |
| BLT |  | 0010 | \*\*\*\*\* | 减 | 0010 |
| BGE |  | 0010 | \*\*\*\*\* | 减 | 0010 |
| BLTU |  | 0010 | \*\*\*\*\* | 减 | 0010 |
| BGEU |  | 0010 | \*\*\*\*\* | 减 | 0010 |
| B |  | 0 | \*\*\*\*\* | 无 | 0 |
| BL |  | 0 | \*\*\*\*\* | 无 | 0 |
| JIRL |  | 0 | \*\*\*\*\* | 无 | 0 |
| LD.W |  | 0001 | \*\*\*\*\* | 加 | 0001 |
| LD.B |  | 0001 | \*\*\*\*\* | 加 | 0001 |
| ST.W |  | 0001 | \*\*\*\*\* | 加 | 0001 |
| ST.B |  | 0001 | \*\*\*\*\* | 加 | 0001 |