**MIPS32中通用寄存器的约定用法：**

**MIPS32架构中定义的特殊寄存器有三个: PC (Program Counter程序计数器)、HI (乘除结果高位寄存器)、LO (乘除结果低位寄存器)。进行乘法运算时，HI和LO保存乘法运算的结果，其中HI存储高32位，LO存储低32位；进行除法运算时，HI和LO保存除法运算的结果，其中HI存储余数，LO存储商。**

**数据在存储器（不是寄存器）中是按照字节存放的，处理器也是按照字节访问存储器中的指令或数据，但是如果需要读出一个字，比如读出的是mem[n]、mem[n+1]、 mem[n+2]、mem[n+3]这四个字节，采用大端模式(Big-Endian)交给处理器的结果：**

**{mem[n], mem[n+1], mem[n+2], mem[n+3]}**

**在大端模式下，数据的高位保存在存储器的低地址中。**

**非阻塞赋值 a<=b**

**阻塞赋值 a=b**

**阻塞赋值在该语句结束时就立即完成赋值操作。在多条阻塞赋值语句中，前面的赋值语句没有完成之前，后面的语句就肯定没有执行，仿佛被阻塞了一样，因此称为阻塞赋值方式。**

**在此数据通路中，Regfile在ID读，WB写；Hilo在EX读，MEM写**

控制器部分(controller)与单周期相同，仍然由 main decoder 和 alu decoder 构成，但由于改为五级流水线后，每一个阶段所需要的控制信号仅为一部分，控制器产生信号的阶段为译码阶段，产生控制信号后，依次通过触发器传到下一阶段，若为当前阶段需要的信号，则不需要继续传递到下一阶段：

控制信号说明如下：

|  |  |
| --- | --- |
| **sign\_ext** | 是否做符号拓展（1为符号拓展，0为0拓展） |
| **regwrite** | 是否要写寄存器（为0时memtoreg无效） |
| **regdst[1:0]** | 目的寄存器地址（00为rt，01为rd，10为31号寄存器） |
| **alusrc** | ALU操作数来源（0为rt，1为立即数） |
| **branch** | 是否为分支指令 |
| **memwrite** | 是否向数据cache中写数据 |
| **memwrite\_con[1:0]** | 决定是哪条写指令（字节，半字，字） |
| **memread\_con[2:0]** | 决定是哪条读指令（字节，半字，字；有无符号拓展） |
| **memtoreg[1:0]** | 是否将数据cache中数据**写入寄存器**  （00：ALU结果，01：内存中数据，10：hi寄存器，11：lo寄存器） |
| **jump** | 是否为无条件跳转指令 |
| **jr** | 是否为jumpR型指令 |
| **link** | 分支指令是否要链接到寄存器 |
| **hilowrite** | 是否要写hilo寄存器 |
| **aluop[4:0]** | ALU控制码，为指明ALU要做的运算 |

**Aludec模块信号说明：**

|  |  |  |
| --- | --- | --- |
| **ALUOp[4:0]** | **Funct** | **ALUControl[4:0]** |
| **00000** | **X(for lw, sw, addi)** | **00010 (Add)** |
| **00101** | **X(for addiu)** | **10010(Add\_unsign)** |
| **00110** | **X(for slti)** | **00111(slt)** |
| **00111** | **X(for sltiu)** | **10111(slt\_u)** |
| **01000** | **X(for beq)** | **00110 (Subtract)** |
|  |  |  |
| **00001** | **X(for andi)** | **00000(And)** |
| **00010** | **X(for xori)** | **00011(Xor)** |
| **00011** | **X(for ori)** | **00001(Or)** |
| **00100** | **X(for lui)** | **00101(LUI)** |
|  |  |  |
| **10000** | **100000 (add)** | **00010 (Add)** |
| **10000** | **100001(addu)** | **10010(Add\_unsign)** |
| **10000** | **100010 (sub)** | **00110 (Subtract)** |
| **10000** | **100011(subu)** | **10110(Subtract\_u)** |
| **10000** | **101010(slt)** | **00111(slt)** |
| **10000** | **101011(sltu)** | **10111(slt\_u)** |
|  |  |  |
| **10000** | **100100 (and)** | **00000 (And)** |
| **10000** | **100101 (or)** | **00001 (Or)** |
| **10000** | **100110(Xor)** | **00011(Xor)** |
| **10000** | **100111(nor)** | **00100(nor)** |
|  |  |  |
| **10000** | **000000(sll)** | **01000(sll，逻辑左移)** |
| **10000** | **000010(srl)** | **01001(srl，逻辑右移)** |
| **10000** | **000011(sra)** | **01010(sra，算术右移)** |
| **10000** | **000100(sllv)** | **11000(sllv，逻辑左移)** |
| **10000** | **000110(srlv)** | **11001(srlv，逻辑右移)** |
| **10000** | **000111(srav)** | **11010(srav，算术右移)** |
|  |  |  |
| **10000** | **010001(mthi)** | **10000(rs赋值)** |
| **10000** | **010011(mtlo)** | **10000(rs赋值)** |

这里需要注意，不按照表中的信号也可，只要保证不同运算的 alucontrol 信号不同即可。

**Memwrite\_con信号用于选择写入内存的数据（当memwrite为1时有效）**

|  |  |
| --- | --- |
| **Memwrite[1:0]** | **类型** |
| **01** | **SW（写字）** |
| **10** | **SH（写半字）** |
| **11** | **SB（写字节）** |

**Memread\_con信号用于选择从readdata中获得的数据（当memtoreg为01时有效）**

|  |  |
| --- | --- |
| **Memread\_con[2:0]** | **类型** |
| **000** | **LW（读字）** |
| **001** | **LH（读半字，符号拓展）** |
| **101** | **LHU（读半字，无符号拓展）** |
| **010** | **LB（读字节，符号拓展）** |
| **110** | **LBU（读字节，无符号拓展）** |

**B型（有条件跳转）指令说明：**

|  |  |  |
| --- | --- | --- |
| **类型** | **说明** | **目的地址** |
| **BEQ** | **rs==rt跳转** | **offset左移2位再进行有符号拓展（32bit）+(pc+4)** |
| **BNE** | **rs!=rt跳转** | **offset左移2位再进行有符号拓展（32bit）+(pc+4)** |
| **BGEZ** | **rs>=0跳转** | **offset左移2位再进行有符号拓展（32bit）+(pc+4)** |
| **BGTZ** | **rs>0跳转** | **offset左移2位再进行有符号拓展（32bit）+(pc+4)** |
| **BLEZ** | **rs<=0跳转** | **offset左移2位再进行有符号拓展（32bit）+(pc+4)** |
| **BLTZ** | **rs<0跳转** | **offset左移2位再进行有符号拓展（32bit）+(pc+4)** |
| **BGEZAL** | **rs>=0跳转** | **offset左移2位再进行有符号拓展（32bit）+(pc+4)**  无论转移与否，将（PC+8）保存至第31号寄存器中 |
| **BLTZAL** | **rs<0跳转** | **offset左移2位再进行有符号拓展（32bit）+(pc+4)**  无论转移与否，将（PC+8）保存至第31号寄存器中 |

**J型（无条件跳转）指令说明：**

|  |  |
| --- | --- |
| **类型** | **目的地址** |
| **J** | 由(PC+4)的高4位与立即数 instr\_index 左移 2 位后的值（28bit）拼接得到 |
| **JAL** | 由(PC+4)的高4位与立即数 instr\_index 左移 2 位后的值（28bit）拼接得到  将（PC+8）保存至第31号寄存器中 |
| **JR** | 为寄存器rs中的值 |
| **JALR** | 为寄存器rs中的值  将（PC+8）保存至rd寄存器中 |