# Verilog流水线处理器

20373864 谭立德

## 一、CPU设计方案综述

### （一）总体设计概述

本CPU为verilog实现的流水线MIPS - CPU，支持的指令集包含MIPS-C3={LB、LBU、LH、LHU、LW、SB、SH、SW、ADD、ADDU、 SUB、 SUBU、 MULT、 MULTU、 DIV、 DIVU、 SLL、 SRL、 SRA、 SLLV、 SRLV、SRAV、AND、OR、XOR、NOR、ADDI、ADDIU、ANDI、ORI、 XORI、LUI、SLT、SLTI、SLTIU、SLTU、BEQ、BNE、BLEZ、BGTZ、 BLTZ、BGEZ、J、JAL、JALR、JR、MFHI、MFLO、MTHI、MTLO}。为了实现这些功能，CPU主要包含了IM、GRF、DM、ALU、PC、CU 等主要模块，这些模块按照自顶向下的顶层设计逐级展开。

### （二）关键模块定义

#### 1. GRF （通用寄存器组，也称为寄存器文件、寄存器堆）

GRF端口定义：

表0 GRF端口表

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| Clk | I | 时钟信号 |
| Reset | I | 复位信号，将32个寄存器中的值全部清零  1：复位  0：无效 |
| We | I | 写使能信号  1：可向GRF中写入数据  0：不可向GEF中写入数据 |
| A1 | I | 5位地址输入信号，指定32个寄存器中的一个，将其中存储的数据读出至RD1 |
| A2 | I | 5位地址输入信号，指定32个寄存器中的一个，将其中存储的数据读出至RD2 |
| A3 | I | 5位地址输入信号，指定32个寄存器中的一个作为写入的目标寄存器 |
| WD3 | I | 32位数据输入信号 |
| RD1 | O | 输出指定的寄存器中的32位数据 |
| RD2 | O | 输出指定的寄存器中的32位数据 |

GRF模块功能定义：

表1 GRF功能表

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 描述 |
| 1 | 复位 | Reset信号有效时，所有寄存器储存的数值清零 |
| 2 | 读数据 | 读出A1，A2地址对应寄存器中所储存的数据到RD1，RD2 |
| 3 | 写数据 | 当WE有效且时钟上升沿来临时，将WD写入A3所对应的寄存器中 |

1. DM （数据存储器）：

DM端口定义：

表2 DM端口表

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| Clk | I | 时钟信号 |
| Reset | I | 复位信号，将32个寄存器中的值全部清零  1：复位  0：无效 |
| We | I | 写使能信号  1：可向DM中写入数据  0：不可向DM中写入数据 |
| A | I | 5位地址输入信号，指定中储存器上的地址，将其中存储的数据读出至RD1 |
| WD | I | 32位数据输入信号 |
| RD | O | 输出储存器指定地址上的32位数据 |

DM模块功能定义：

表3 DM功能表

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 描述 |
| 1 | 复位 | Reset信号有效时，储存器储存的所有数值清零 |
| 2 | 读数据 | 读出A地址对应储存器中所储存的数据到RD |
| 3 | 写数据 | 当WE有效且时钟上升沿来临时，将WD写入A3所对应的寄存器中 |

#### ALU （算术逻辑运算单元）：

ALU端口定义：

表4 ALU端口表

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| SrcA | I | 32位运算数输入信号 |
| SrcB | I | 32位运算数输入信号 |
| ALU Control | I | 3位逻辑运算选择信号，选择进行哪种逻辑运算 |
| Zero | O | 输出比较两运算数比较的1位输出 |
| ALU Result | O | 输出对两运算数进行指定逻辑运算后的32位结果 |

ALU模块功能定义：

表5 ALU功能表

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 描述 |
| 1 | 计算 | 根据控制信号进行对应的逻辑计算并输出 |
| 2 | 比较 | 判断两个输入是否相等 |

#### IM （指令存储器）：

IM端口定义：

表6 IM端口表

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| PC | I | 5位输入地址信号 |
| Instr | 0 | 输出地址所储存32位指令 |

IM模块功能定义：

表7 IM功能表

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 描述 |
| 1 | 读指令 | 根据输入输出对应32位指令 |

#### Control Unit （指令译码器）：

Control Unit端口定义：

表8 Control Unit端口表

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| Opcode[5:0] | I | 指令操作码 |
| Funct[5:0] | I | 指令功能码 |
| Jump | O | 跳转信号 |
| ToHigh16 | O | 高位置位信号 |
| ExtOp | O | 位扩展方式 |
| MemtoReg | O | 读内存信号 |
| MemWrite | O | 内存写使能信号 |
| Branch | O | 分支信号 |
| ALUCtrl[2:0] | O | ALU控制信号 |
| ALUSrc | O | ALU操作数2的来源0：寄存器1：立即数 |
| RegDst | O | 寄存器写地址选择0：Instr[20:16] 1：Instr[15:11] |
| RegWrite | O | 寄存器写使能信号 |
| DMop[1:0] | O | 存储、读取方式控制信号 |

### （三）重要机制实现方法

#### 1. J类型指令

根据输入判断和ALU模块协同工作算出跳转地址后跳转。

#### 2. R类型指令

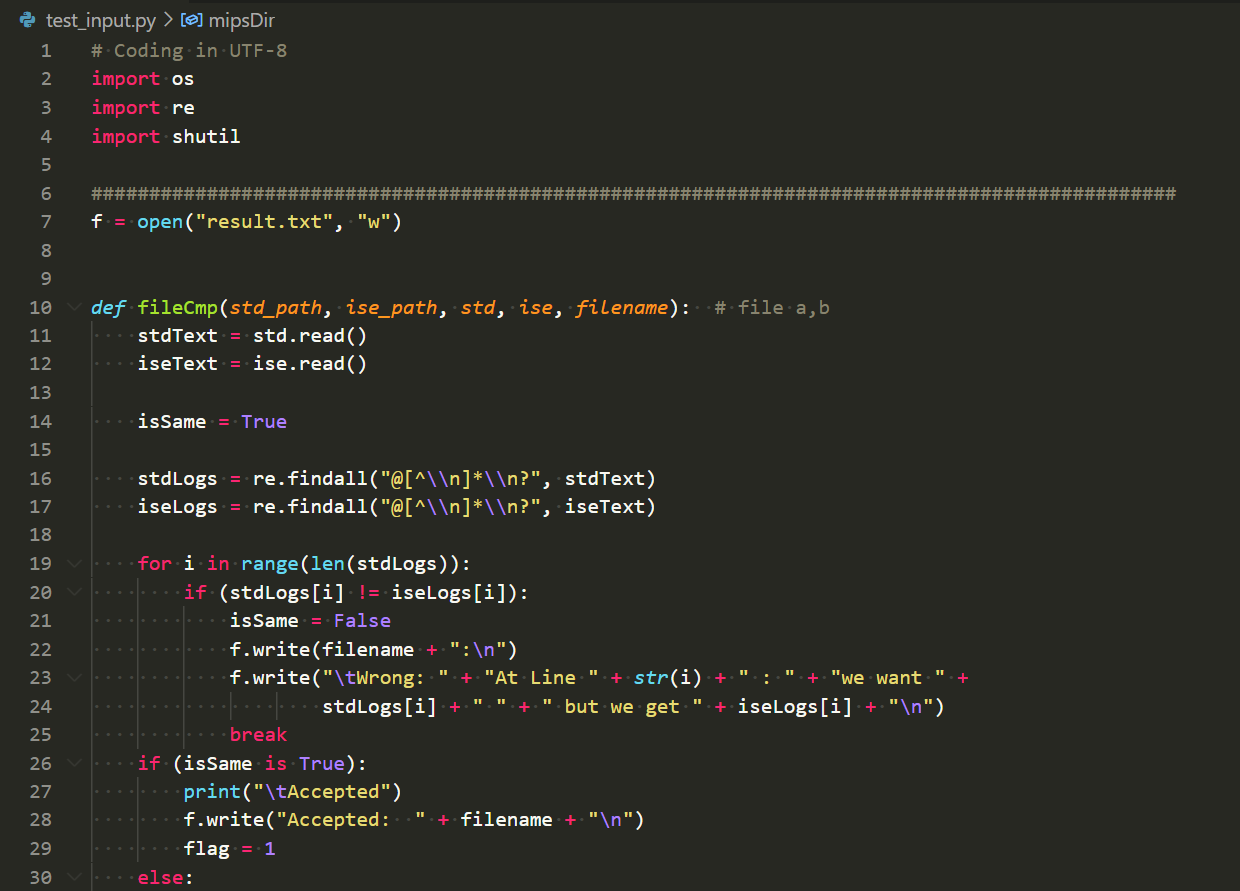
根据输入判断和ALU模块协同工作算出结果后存储回寄存器堆中以实现指令R类型指令。

#### 3. I类型指令

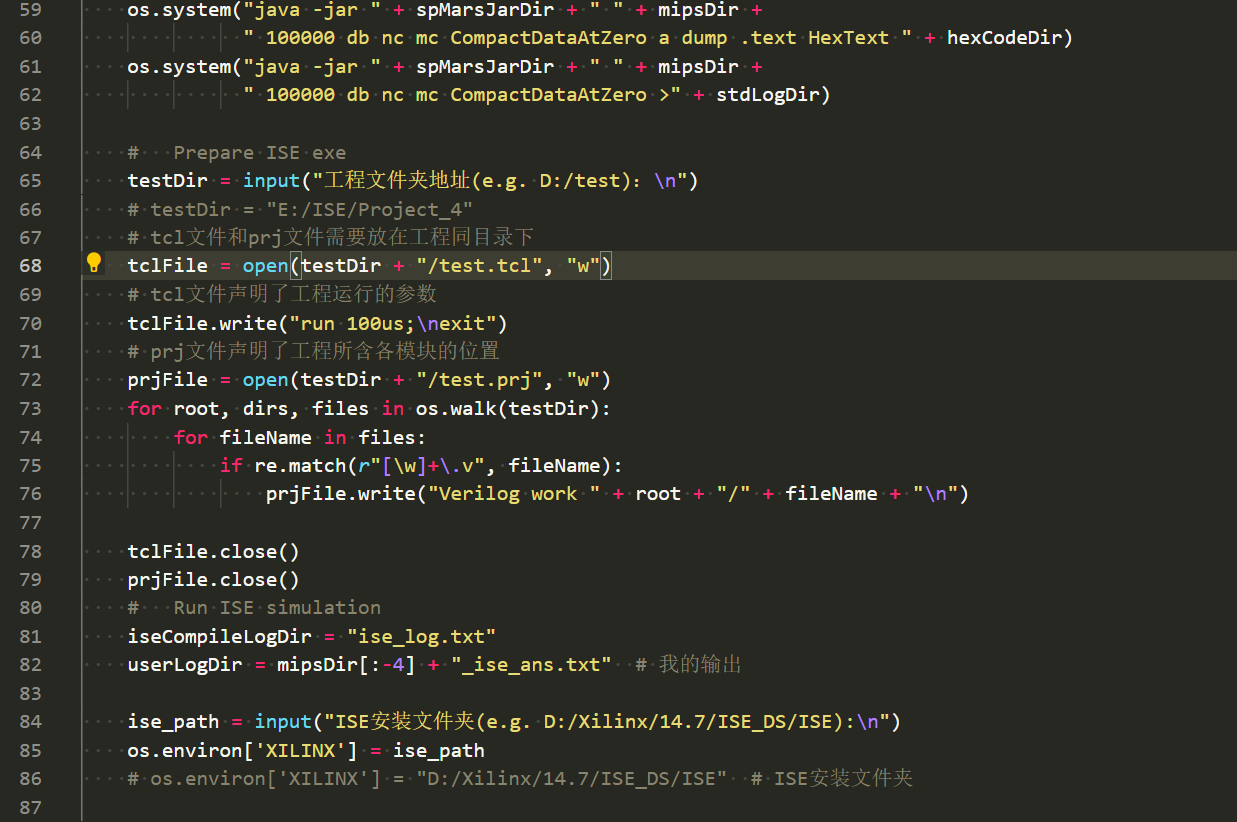
根据输入判断和ALU模块和DM模块协同工作支持I类型指令。

## 测试方案

（同P5）









## 三、思考题

### 1.为什么需要有单独的乘除法部件而不是整合进 ALU？为何需要有独立的 HI、LO 寄存器？

解：

乘除指令运算非常慢，所以我们的 CPU 要模拟这种时延（两种乘法延时 5 个周期，两种除法延时 10 个周期）。为了不让乘除法拖慢速度，我们遇到相关指令时让他们进入一个特殊的乘除部件进行运算，并且让结果存储在HI，LO中，只在特殊指令进行调用，其他指令继续执行。如果乘除法正在计算，而我们遇到了新的要用到乘除部件的指令，那么就 stall。

### 2.参照你对延迟槽的理解，试解释 “乘除槽”。

解：

当乘除法进行或即将开始时，下一条乘除有关指令会被阻塞在 IF/ID 流水线寄存器，即相当于处于“乘除槽”，而其他指令正常流水执行。

### 3.举例说明并分析何时按字节访问内存相对于按字访问内存性能上更有优势。（Hint： 考虑 C 语言中字符串的情况）

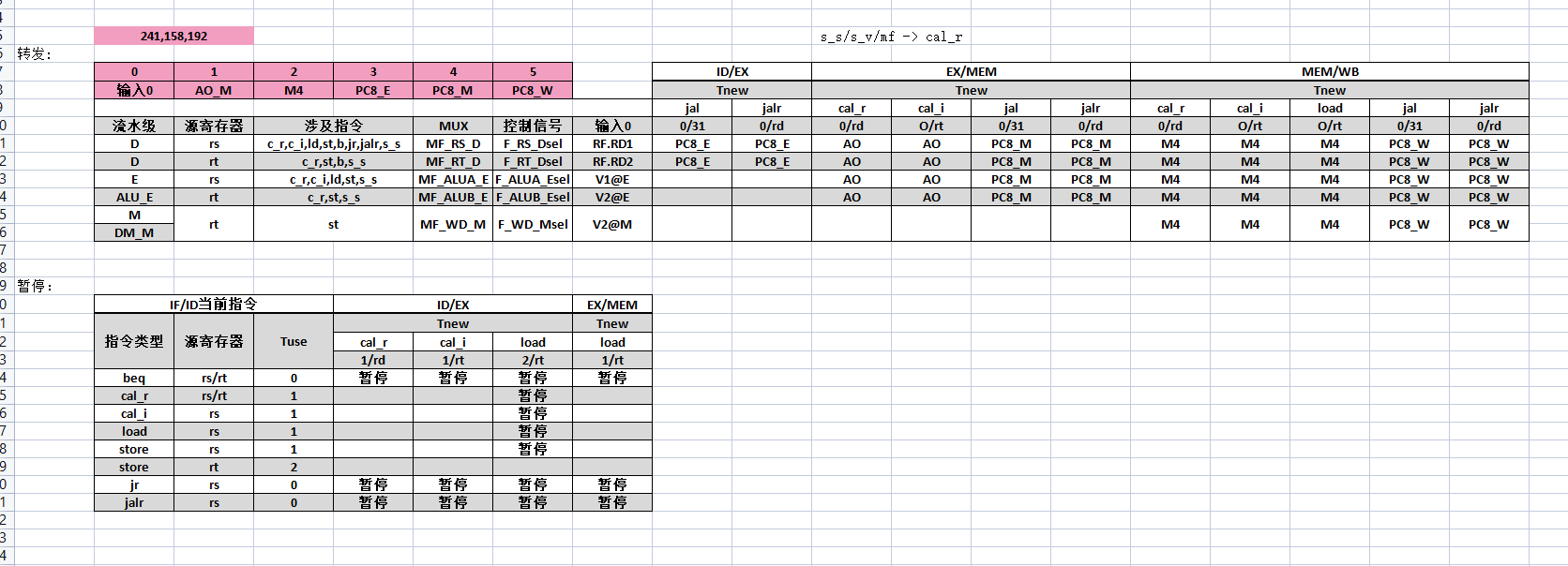
解：

当访问类型只占一个字节时，比如char时。

### 4.在本实验中你遇到了哪些不同指令类型组合产生的冲突？你又是如何解决的？相应的测试样例是什么样的？（如果你是手动构造的样例，请说明构造策略，说明你的测试程序如何保证覆盖了所有需要测试的情况；如果你是完全随机生成的测试样例，请思考完全随机的测试程序有何不足之处；如果你在生成测试样例时采用了特殊的策略，比如构造连续数据冒险序列，请你描述一下你使用的策略如何结合了随机性达到强测的效果。（此思考题请同学们结合自己测试 CPU 使用的具体手段，按照自己的实际情况进行回答））

解：

不同类型组合产生的冲突：



解决方法：构造转发和暂停机制，在遇到冲突冒险时进行对应转发和冒险操作。

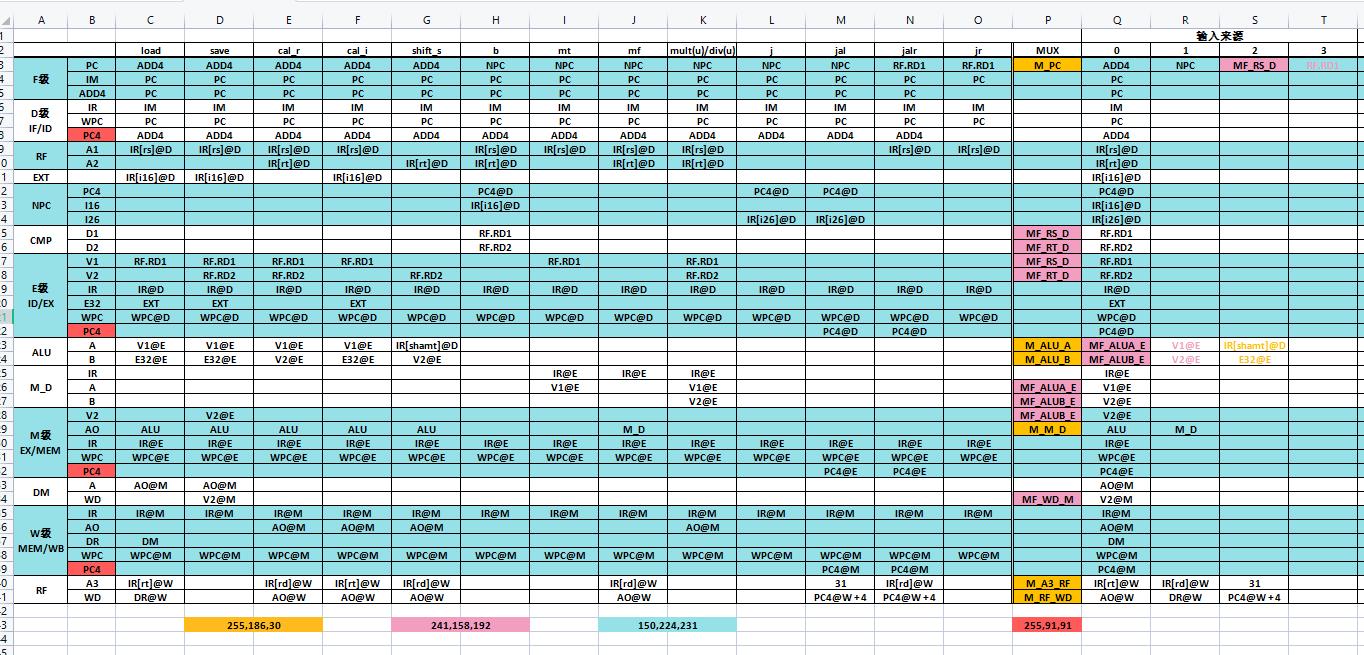
### 5.为了对抗复杂性你采取了哪些抽象和规范手段？这些手段在译码和处理数据冲突的时候有什么样的特点与帮助？

解：

主要采用了指令分类的方法，P6相对于P5所新增的指令对应的特点都可以归纳为几类，因此对于每条指令而言，只需译码后将其加入对应的分类，构造数据通路，分析转发、暂停条件，并添加一个乘除模块即可。

附1：

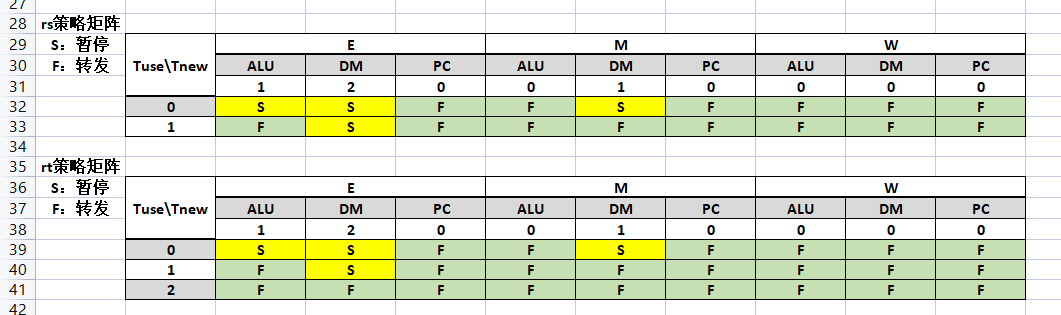
数据通路

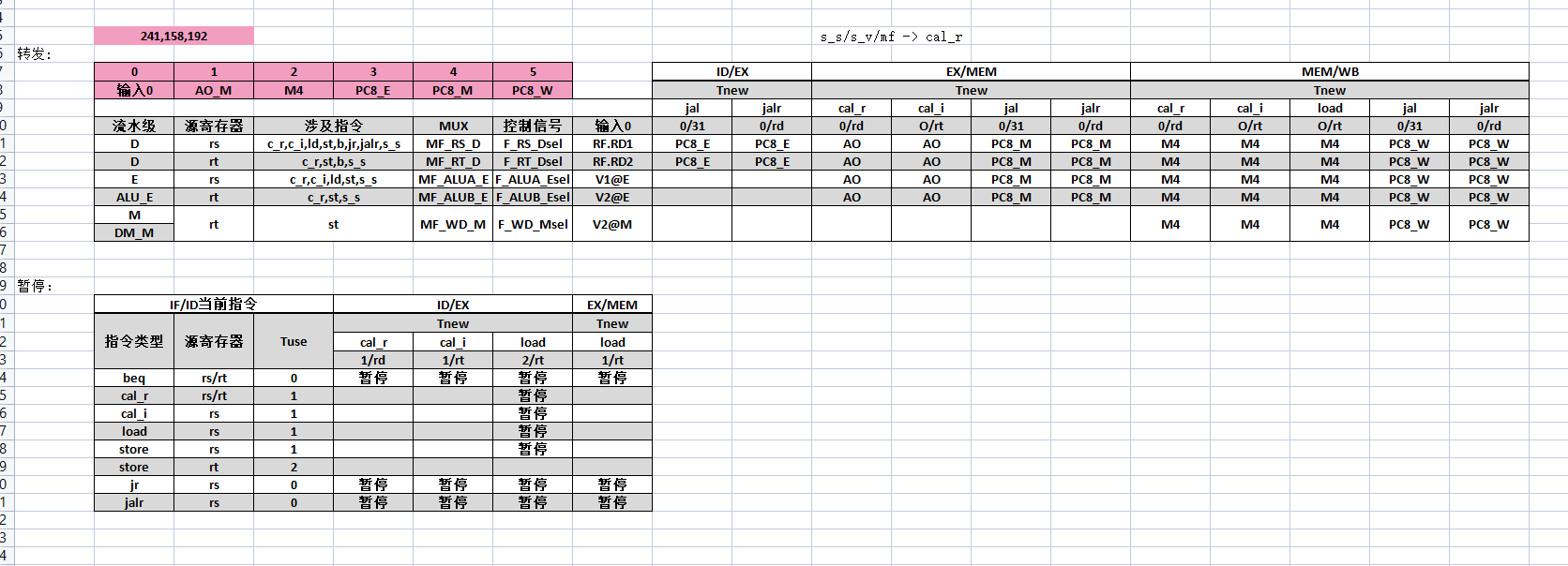


附2：

冲突策略矩阵

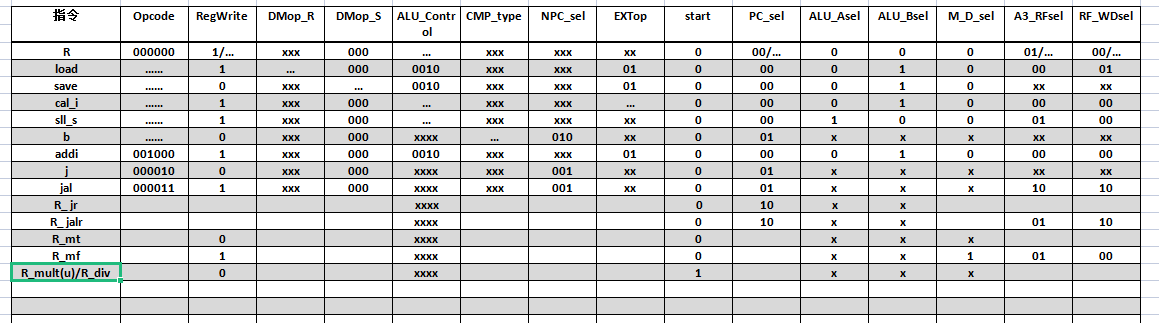


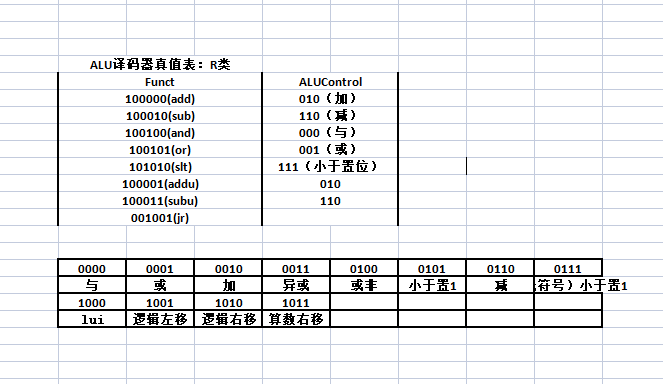




附3：

控制器





附4：

多路选择器控制信号

