计算机组成原理P5实验报告

彭杰奇15061169

1. 数据通路设计

F级功能部件：

1.IFU模块

（1）基本描述

IFU主要功能是完成取指令功能。IFU内部包括了PC、IM(指令存储器)以及其他相关逻辑。

（2）模块接口

|  |  |
| --- | --- |
| 文件 | 模块接口定义 |
| IFU.v | IFU(nextPC, Clk, Reset, PC\_En, PC4, Instr);  input [31:0] nextPC;  input Clk;  input Reset;  input PC\_En;  output [31:0] Instr;  output [31:0] PC4; |

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| nextPC[31:0] | I | 输入PC的下一条指令地址 |
| Reset | I | 复位信号，1:有效，0:无效 |
| Clk | I | 时钟信号 |
| PC\_En | I | IFU内部PC的使能端，1:有效，0:无效 |
| Instr[31:0] | O | 当前指令输出 |
| PC4[31:0] | O | 当前PC下PC + 4的值 |

（3）功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 复位 | 复位信号有效时，PC设置为0x00003000 |
| 2 | 取指令 | 根据PC当前值从IM中取指令输出 |
| 3 | 输出PC+4 | PC4 为当前PC下 PC + 4 的值 |

2.IM模块

（1）基本描述

IM是指令存储模块，由一个32bit×1024字的存储器组成，其功能是保存指令，并根据输入的PC输出相应指令。

（2）模块接口

|  |  |
| --- | --- |
| 文件 | 模块接口定义 |
| IM.v | IM(Addr,Instr);  input [11:2] Addr; // 输入的指令地址  output [31:0] Instr; // 输出的指令 |

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| Addr[11:2] | I | 输入的指令地址 |
| Instr[31:0] | O | 输出的指令 |

（3）功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 输出指令 | Instr 🡨 im[Addr] |

IF\_ID流水线寄存器：

(1)模块接口

|  |  |
| --- | --- |
| 文件 | 模块接口定义 |
| IF\_ID.v | IF\_ID(IM, ADD4, Clk, Reset, IR\_D\_En, IR\_D, PC4\_D);  input [31:0] IM;  input [31:0] ADD4;  input Clk;  input Reset;  input IR\_D\_En;  output [31:0] IR\_D;  output [31:0] PC4\_D; |

（2）功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 复位 | 清空寄存器 |
| 2 | 存数取数 | 将前一级的值存入寄存器中，将后一级的值输出 |
| 3 | 冻结 | 冻结IR\_D |

D级功能部件：

1.GRF模块

（1）基本描述

GRF模块为通用寄存器堆，主要由32个具有写使能端的32位寄存器组成，能够同时根据由rs和rt输入的地址从其中两个寄存器中读出数据，并根据wr中输入的地址向其中一个寄存器写入数据。

（2）模块接口

|  |  |
| --- | --- |
| 文件 | 模块接口定义 |
| GRF.v | GRF(rs, rt, wr, WData, Clk, Reset, RegWrite, RData1, RData2);  input [4:0] rs;  input [4:0] rt;  input [4:0] wr;  input [31:0] WData;  input Clk;  input Reset;  input RegWrite;  output [31:0] RData1;  output [31:0] RData2; |

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| rs[4:0] | I | rs寄存器的地址 |
| rt[4:0] | I | rt寄存器的地址 |
| wr[4:0] | I | 要写入的寄存器的地址 |
| WData[31:0] | I | 要写入的数据 |
| Clk | I | 时钟信号 |
| Reset | I | 复位信号 |
| RegWrite | I | 一般写使能信号，1:有效，0:无效 |
| RData1[31:0] | O | rs寄存器的值 |
| RData2[31:0] | O | rt寄存器的值 |

（3）功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 读数据 | RData1🡨(GRF[rs])  RData2🡨(GRF[rt]) |
| 2 | 写数据 | RegWrite有效时，(GPR[wr])🡨WData |
| 3 | 清零 | 复位信号有效时，GRF中所有寄存器都清零 |

2.EXT模块

（1）基本描述

EXT模块的作用是将16位立即数扩展为32位。

（2）模块接口

|  |  |
| --- | --- |
| 文件 | 模块接口定义 |
| EXT.v | EXT(Imm\_16, ExtOp, Imm\_32);  input [15:0] Imm\_16;  input [1:0] ExtOp;  output [31:0] Imm\_32; |

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| Imm\_16[15:0] | I | 要扩展的16位立即数 |
| ExtOp[1:0] | I | 扩展方式选择信号  2’b00:符号扩展  2’b01:后接16位0  2’b10:无符号扩展 |
| Imm\_32[31:0] | O | 扩展后的32位立即数 |

（3）功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 位数扩展 | ExtOp为2’b00时，16位立即数正常符号扩展为32位  ExtOp为2’b01时，16为立即数后接16位0扩展为32位  ExtOp为2’b10时，16为立即数无符号扩展为32位 |

3.CMP模块

（1）基本描述

CMP模块用来比较输入的两个数据是否相等，用于beq指令是否跳转的判断

（2）模块接口

|  |  |
| --- | --- |
| 文件 | 模块接口定义 |
| CMP.v | CMP(A1, A2, equal);  input [31:0] A1;  input [31:0] A2;  output equal; |

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| A1[31:0] | I | 第一个数据 |
| A2[31:0] | I | 第二个数据 |
| Equal | O | 输入的数据是否相等  1:相等  0:不相等 |

（3）功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 等于判断 | equal 🡨 (A1==A2)?1:0 |

4.NPC模块

（1）基本描述

NPC模块能够输出当前指令为beq或J类型等跳转指令时下一条指令地址

（2）模块接口

|  |  |
| --- | --- |
| 文件 | 模块接口定义 |
| NPC.v | NPC(PC4, Instr, J\_Sel, Branch, Zero, nPC);  input [31:0] PC4;  input [31:0] Instr;  input [1:0] J\_Sel;  input Branch;  input Zero;  output [31:0] nPC; |

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| PC4[31:0] | I | 来自PC4\_D寄存器 |
| Instr[31:0] | I | 来自IR\_D寄存器 |
| J\_Sel[1:0] | I | 当下J类型指令具体为:  2’b00:不是j类型指令  2’b01:指令为J  2’b10:指令为Jal  2’b11:指令为Jr |
| Branch | I | 当下指令是否为beq  1:是，0:不是 |
| Zero | I | 若为beq指令，比较的两个数据是否相等  1:相等，0:不相等 |
| nPC[31:0] | O | 下一条指令地址  J\_Sel为2’b01或2’b10:  nPC 🡨 PC[31:28]||index||0^2  Branch为1且Zero为1:  nPC 🡨 PC + 4 + Imm\_32||0^2  Branch 为1而Zero为0:  nPC🡨 PC4 + 4  其中:  PC = PC4-4 Index = Instr[25:0] Imm16 = Instr[15:0] |

（3）功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 输出下一条PC地址 | nPC输出当指令为beq或者J类型指令时下一条指令的地址 |

ID\_EX流水线寄存器：

(1)模块接口

|  |  |
| --- | --- |
| 文件 | 模块接口定义 |
| ID\_EX.v | ID\_EX(IR\_D, PC4\_D, RD1, RD2, EXT, Clk, Reset, IR\_E\_Clr, IR\_E, PC4\_E, RS\_E, RT\_E, EXT\_E);  input [31:0] IR\_D;  input [31:0] PC4\_D;  input [31:0] RD1;  input [31:0] RD2;  input [31:0] EXT;  input Clk;  input Reset;  input IR\_E\_Clr;  output [31:0] IR\_E;  output [31:0] PC4\_E;  output [31:0] RS\_E;  output [31:0] RT\_E;  output [31:0] EXT\_E; |

（2）功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 复位 | 清空寄存器 |
| 2 | 存数取数 | 将前一级的值存入寄存器中，将后一级的值输出 |
| 3 | 清除 | 清除IR\_E的值 |

EX级功能部件：

1.ALU模块

（1）基本描述

ALU为算数逻辑单元，可以对输入的两个数据进行加、减、按位与和按位或操作，并能够判断输入数据是否相等。

（2）模块接口

|  |  |
| --- | --- |
| 文件 | 模块接口定义 |
| ALU.v | ALU(A1, A2, ALUCtr, ALUResult);  input [31:0] A1;  input [31:0] A2;  input [2:0] ALUCtr;  output [31:0] ALUResult; |

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| A1[31:0] | I | 第一个运算数 |
| A2[31:0] | I | 第二个运算数 |
| ALUCtr[2:0] | I | ALU控制信号  2’b000:加法运算  2’b001:减法运算  2’b010:按位与运算  2’b011:按位或运算 |
| ALUResult[31:0] | O | ALU运算结果 |

（3）功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 加法运算 | ALUResult 🡨 A1+A2 |
| 2 | 减法运算 | ALUResult 🡨 A1-A2 |
| 3 | 按位与运算 | ALUResult 🡨 A1&A2 |
| 4 | 按位或运算 | ALUResult 🡨 A1|A2 |

EX\_MEM流水线寄存器：

(1)模块接口

|  |  |
| --- | --- |
| 文件 | 模块接口定义 |
| EX\_MEM.v | EX\_MEM(IR\_E, PC4\_E, AO, RT\_E, Clk, Reset, IR\_M, PC4\_M, AO\_M, RT\_M);  input [31:0] IR\_E;  input [31:0] PC4\_E;  input [31:0] AO;  input [31:0] RT\_E;  input Clk;  input Reset;  output [31:0] IR\_M;  output [31:0] PC4\_M;  output [31:0] AO\_M;  output [31:0] RT\_M; |

（2）功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 复位 | 清空寄存器 |
| 2 | 存数取数 | 将前一级的值存入寄存器中，将后一级的值输出 |

MEM级功能部件：

1.DM模块

（1）基本描述

DM模块为数据存储器，由一个32bit \* 32字的存储器构成，起始地址为0x00000000用于存储数据。

（2）模块接口

|  |  |
| --- | --- |
| 文件 | 模块接口定义 |
| DM.v | DM(Addr,Din,MemWrite,MemRead,Clk,Reset,Dout);  input [31:0] Addr;  input [31:0] Din;  input MemWrite;  input MemRead;  input Clk;  input Reset;  output [31:0] Dout; |

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| Addr[31:0] | I | 读/写DM的地址 |
| Din[31:0] | I | 要写入DM的数据 |
| MemWrite | I | 写DM的控制信号 |
| MemRead | I | 读DM的控制信号 |
| Clk | I | 时钟信号 |
| Reset | I | 复位信号 |
| Dout[31:0] | O | 从DM读出的数据 |

（3）功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 读数据 | 当MemRead为1时，ReadData 🡨 RAM(Addr[11:2]) |
| 2 | 写数据 | 当MemWrite为1时，RAM(Addr) 🡨 WriteData |
| 3 | 清零 | 复位信号有效时，存储器清零 |

MEM\_WB流水线寄存器：

(1)模块接口

|  |  |
| --- | --- |
| 文件 | 模块接口定义 |
| MEM\_WB.v | MEM\_WB(IR\_M, PC4\_M, AO\_M, DR\_M, Clk, Reset, IR\_W, PC4\_W, AO\_W, DR\_W);  input [31:0] IR\_M;  input [31:0] PC4\_M;  input [31:0] AO\_M;  input [31:0] DR\_M;  input Clk;  input Reset;  output [31:0] IR\_W;  output [31:0] PC4\_W;  output [31:0] AO\_W;  output [31:0] DR\_W; |

（2）功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 复位 | 清空寄存器 |
| 2 | 存数取数 | 将前一级的值存入寄存器中，将后一级的值输出 |

1. 控制器设计
2. Controller模块定义

（1）基本描述

Controller模块为CPU控制器，可以根据输入指令的opcode和funct值输出各种控制信号。

（2）模块接口

|  |  |
| --- | --- |
| 文件 | 模块接口定义 |
| Controller.v | Controller(Instr, RegDst, ALUSrc, MemtoReg, RegWrite, MemWrite, MemRead, ExtOp, Branch, J\_Sel, ALUCtr, PC\_Sel);  input [31:0] Instr;  output [1:0] RegDst;  output ALUSrc;  output [1:0] MemtoReg;  output RegWrite;  output MemWrite;  output MemRead;  output [1:0] ExtOp;  output Branch;  output [1:0] J\_Sel;  output [1:0] PC\_Sel;  output [2:0] ALUCtr; |

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| Instr[31:0] | I | 指令 |
| RegDst[1:0] | O | 寄存器写入端地址控制  2’b00:选择rt字段  2’b01:选择rd字段  2’b10:选择31号寄存器 |
| ALUSrc | O | ALU输入端A2选择  0:选择MSRTE  1:选择EXT\_E |
| MemtoReg[1:0] | O | 寄存器堆写入端WD选择  2’b00:来自ALU输出  2’b01:来自DM输出  2’b10:来自PC4\_W+4 |
| RegWrite | O | 写寄存器控制信号 |
| MemWrite | O | 写DM控制信号 |
| MemRead | O | 读DM控制信号 |
| ExtOp[1:0] | O | EXT扩展方式控制信号 |
| Branch | O | 判断是否为beq指令 |
| J\_Sel[1:0] | O | 2’b00:其他指令  2’b01:J指令  2’b10:Jal指令  2’b11:Jr指令 |
| PC\_Sel[1:0] | O | 2’b00:IFU的nextPC选择PC+4  2’b01:IFU的nextPC选择nPC  2’b10:IFU的nextPC选择RData1 |
| ALUCtr[2:0] | O | ALU控制信号 |

1. Controller真值表

|  |  |  |  |
| --- | --- | --- | --- |
| Instr | Subu | addu | Jr |
| opcode | 000000 | 000000 | 000000 |
| funct | 100011 | 100001 | 001000 |
| RegDst[1:0] | 2’b01 | 2’b01 | 2’b01 |
| ALUSrc | 0 | 0 | 0 |
| MemtoReg[1:0] | 2’b00 | 2’b00 | 2’b00 |
| RegWrite | 1 | 1 | 0 |
| Branch | 0 | 0 | 0 |
| J\_Sel[1:0] | 2’b00 | 2’b00 | 2’b11 |
| ExtOp[1:0] | X | X | X |
| MemRead | 0 | 0 | 0 |
| MemWrite | 0 | 0 | 0 |
| PC\_Sel[1:0] | 2’b00 | 2’b00 | 2’b10 |
| ALUCtr[2:0] | 3’b000 | 3’b001 | 3’b000 |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Instr | J | ori | sw | lw | lui | beq | jal |
| opcode | 000010 | 001101 | 100011 | 101011 | 000100 | 001111 | 000011 |
| funct | N/A | | | | | | |
| RegDst[1:0] | X | 2’b00 | 2’b00 | X | X | 2’b00 | 2’b10 |
| ALUSrc | 0 | 1 | 1 | 1 | 0 | 1 | 0 |
| MemtoReg[1:0] | X | 0 | 1 | X | X | 0 | 2’b10 |
| RegWrite | 0 | 1 | 1 | 0 | 0 | 1 | 1 |
| Branch | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| J\_Sel[1:0] | 2’b01 | 2’b00 | 2’b00 | 2’b00 | 2’b00 | 2’b00 | 2’b10 |
| ExtOp[1:0] | 2’b00 | 2’b10 | 2’b00 | 2’b00 | 2’b00 | 2’b01 | 2’b00 |
| MemRead | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| MemWrite | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| PC\_Sel[1:0] | 2’b01 | 2’b00 | 2’b00 | 2’b00 | 2’b00 | 2’b01 | 2’b01 |
| ALUOp[2:0] | 3’b000 | 3’b010 | 3’b000 | 3’b000 | 3’b000 | 3’b000 | 3’b000 |

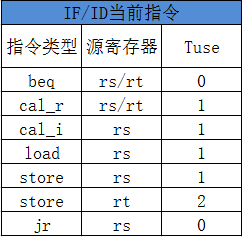
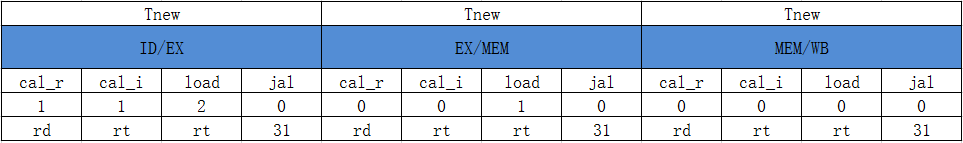
1. 冲突控制器

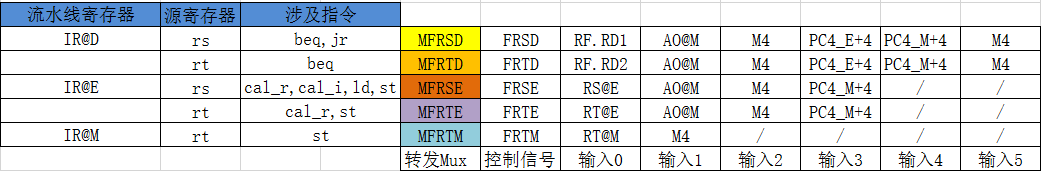
(1)模块接口

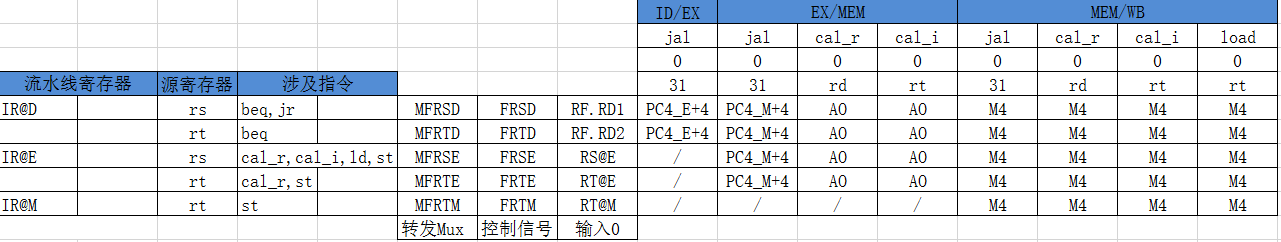
|  |  |
| --- | --- |
| 文件 | 模块接口定义 |
| Conflict\_manager.v | Conflict\_manager(IR\_D, IR\_E, IR\_M, IR\_W, FRSD, FRTD, FRSE, FRTE, FRTM, stall);  input [31:0] IR\_D;  input [31:0] IR\_E;  input [31:0] IR\_M;  input [31:0] IR\_W;  output [1:0] FRSD;  output [1:0] FRTD;  output [1:0] FRSE;  output [1:0] FRTE;  output [1:0] FRTM;  output stall; |

（2）功能定义

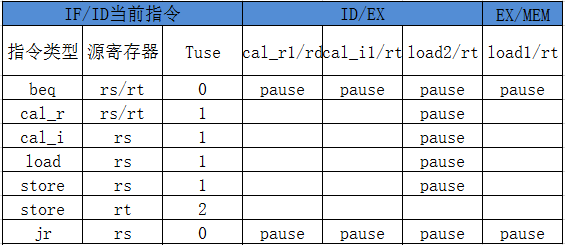
|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 产生暂停信号 | 暂停信号stall |
| 2 | 产生转发信号 | 5个转发信号 |



转发机制



暂停机制



1. 测试程序

lui $3,1

ori $4,$3,2

start:

addu $5,$3,$4 #R=addu

beq $5,$3,label #R\_E\_RS

subu $6,$5,$3 #R=subu

ori $7,$6,3 #R\_M\_RS

ori $8,$6,1 #R\_W\_RS

lui $9,2

j label #test j

subu $13,$7,$8 #延迟槽

addu $10,$8,$9

label:

subu $10,$7,$8 #R=subu

addu $11,$8,$10 #R\_M\_RT

subu $12,$11,$10 #R\_W\_RT

beq $10,$13,label2 #R\_W\_RS

nop

label3:

sw $3,4($0)

sw $4,8($0)

ori $5,$4,0xf1

lui $6,0xff

sw $6,12($0)

lw $7,12($0) #Load

beq $7,$5,label4 #L\_E\_RS

nop

lw $8,4($0) #Load

addu $9,$8,$7 #L\_M\_RS

ori $10,$8,5 #L\_W\_RS

beq $3,$8,label4 #L\_W\_RT

lui $11,2

nop

nop

label5:

jal label6

addu $10,$31,$0 #Jal\_M\_RS

label6:

beq $10,$31,label7 #Jal\_M\_RT

nop

label7:

jal label8

addu $11,$31,$31 #Jal\_M\_RS\_RT

subu $12,$11,$31

beq $12,$12,label9

nop

label8:

addu $12,$31,$31 #Jal\_W\_RS\_RT

jr $31 #Jal\_W\_RS

subu $11,$11,$11

label9: #othertest

lui $17,0xfe

beq $17,$17,label10

ori $18,0x30b8

lui $20,18

jal end

nop

label2:

ori $5,$3,1 #I=ori

beq $5,$3,label #I\_E\_RS

addu $6,$5,$4 #I\_W\_RS

addu $7,$5,$3 #内部转发

ori $8,$6,3 #I=ori

subu $9,$8,$7 #I\_M\_RS

addu $10,$9,$8 #I\_W\_RT

beq $8,$7,label3 #I\_W\_RS

nop

ori $12,$5,2

nop

lui $13,3 #I=lui

addu $14,$12,$13 #I\_M\_RT

ori $13,$13,0xfe

ori $11,$5,2 #I=ori

beq $12,$11,label3 #I\_E\_RT

subu $12,$11,$5

label4:

lw $12,8($0) #Load

subu $13,$5,$12 #L\_M\_RT

addu $14,$12,$13 #L\_W\_RS

addu $16,$5,$0

sw $5,16($0)

lw $15,16($0) #Load

beq $16,$15,label5 #L\_E\_RT

addu $16,$16,$15

label10:

jr $18

nop

end:

nop

预期结果

$ 3 <= 00010000

$ 4 <= 00010002

$ 5 <= 00020002

$ 6 <= 00010002

$ 7 <= 00010003

$ 8 <= 00010003

$ 9 <= 00020000

$13 <= 00000000

$10 <= 00000000

$11 <= 00010003

$12 <= 00010003

$ 5 <= 00010001

$ 6 <= 00020003

$ 7 <= 00020001

$ 8 <= 00020003

$ 9 <= 00000002

$10 <= 00020005

$12 <= 00010003

$13 <= 00030000

$14 <= 00040003

$13 <= 000300fe

$11 <= 00010003

$12 <= 00000002

\*00000004 <= 00010000

\*00000008 <= 00010002

$ 5 <= 000100f3

$ 6 <= 00ff0000

\*0000000c <= 00ff0000

$ 7 <= 00ff0000

$ 8 <= 00010000

$ 9 <= 01000000

$10 <= 00010005

$11 <= 00020000

$12 <= 00010002

$13 <= 000000f1

$14 <= 000100f3

$16 <= 000100f3

\*00000010 <= 000100f3

$15 <= 000100f3

$16 <= 000201e6

$31 <= 00003084

$10 <= 00003084

$31 <= 00003094

$11 <= 00006128

$12 <= 00006128

$11 <= 00000000

$12 <= ffffcf6c

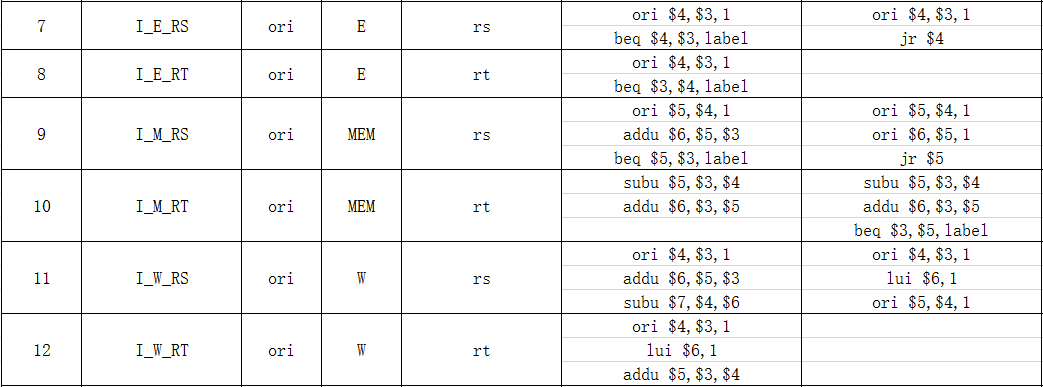
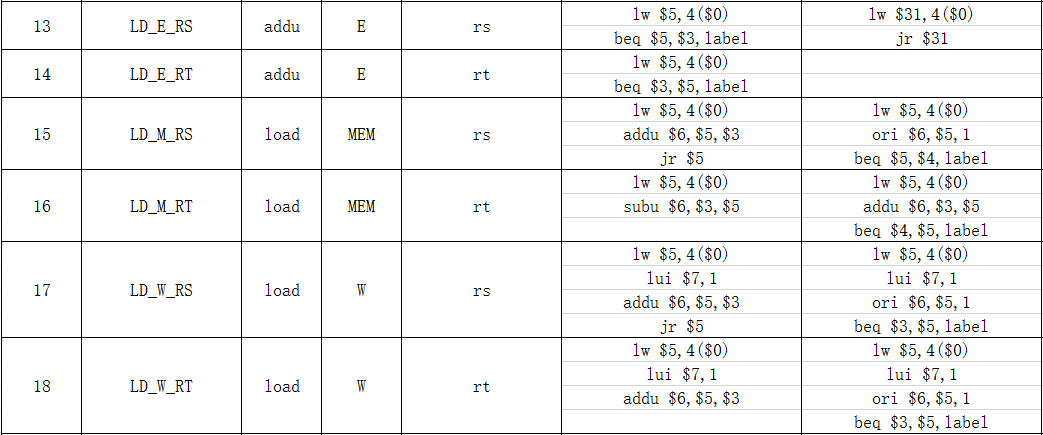
$17 <= 00fe0000

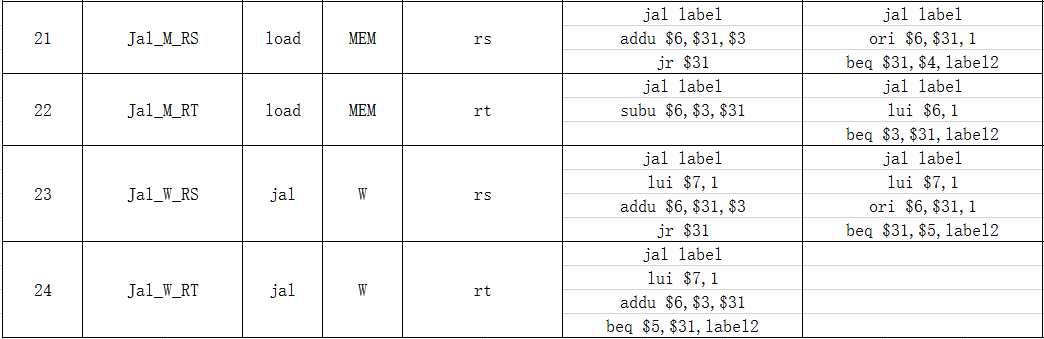
$18 <= 000030b8

$20 <= 00120000

$31 <= 000030c4

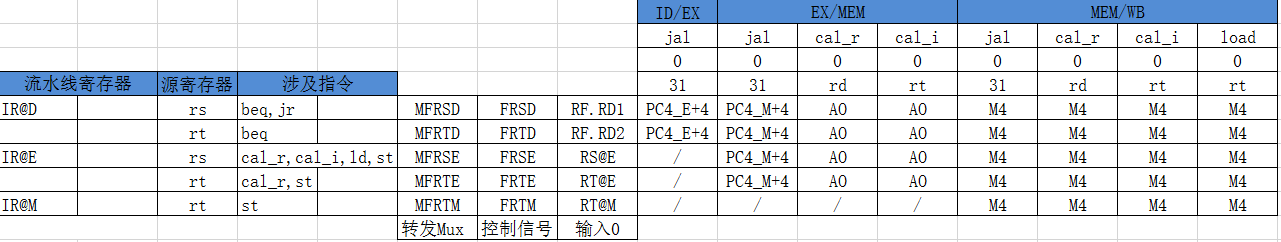
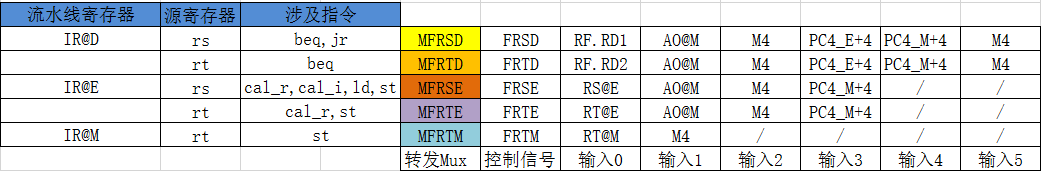
1. 思考题





冲突处理

转发



暂停

