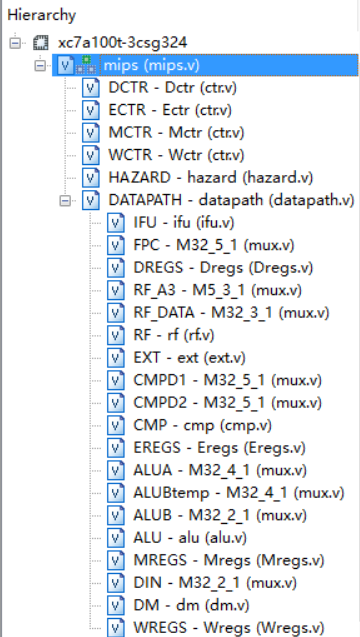
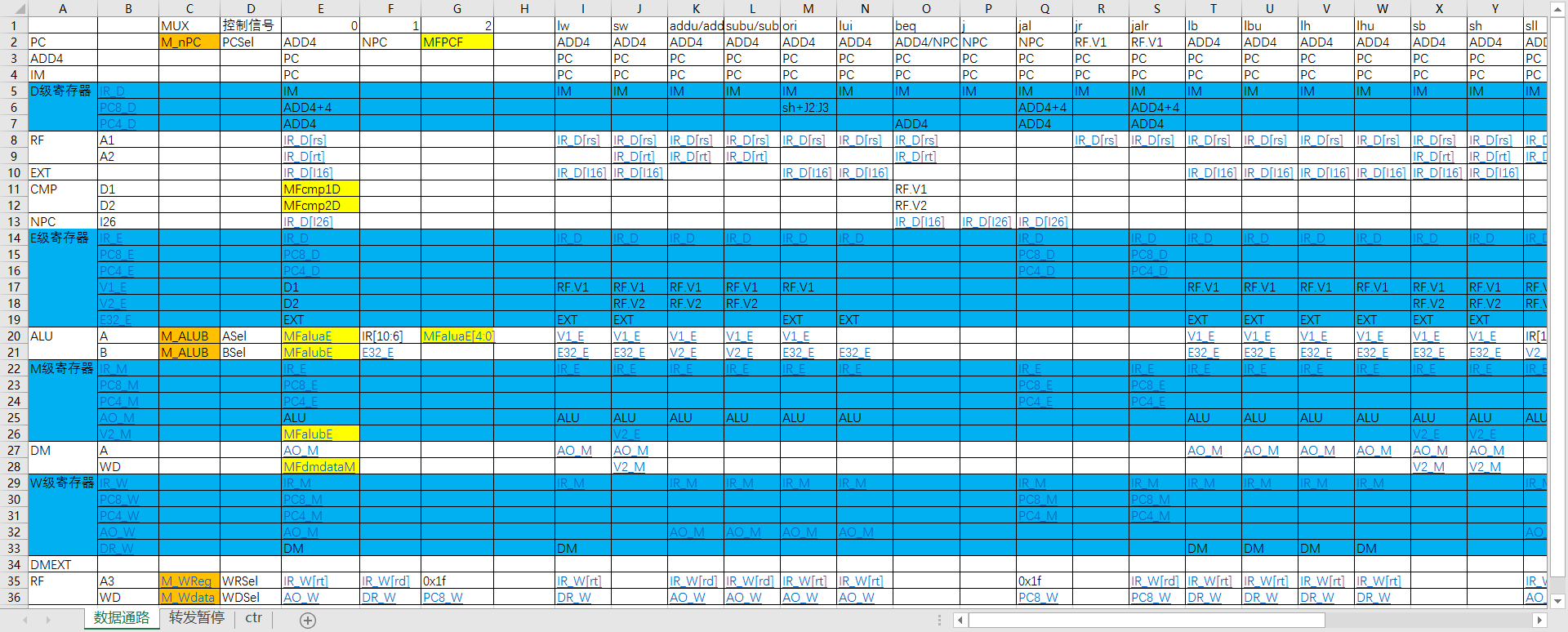
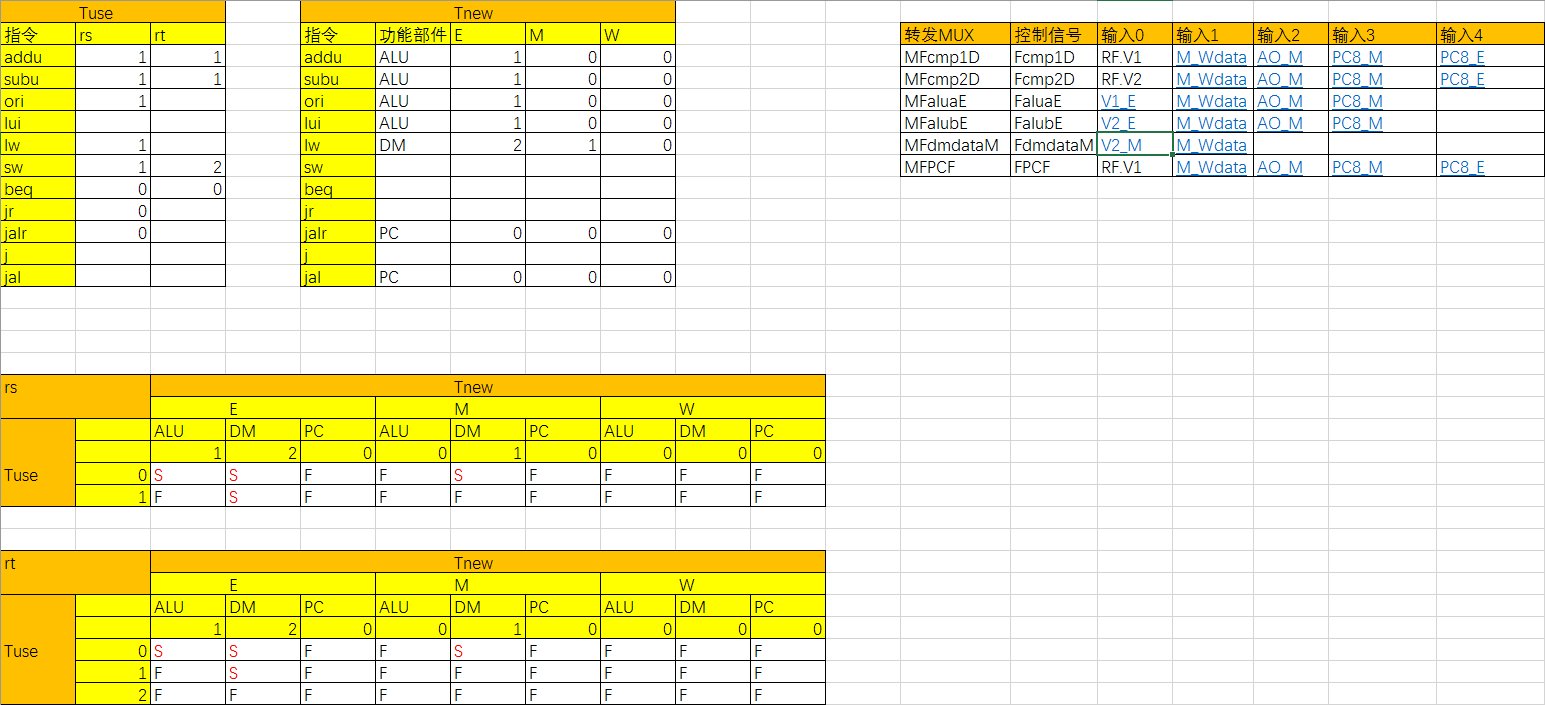
计算机组成原理实验报告

1. CPU设计文档
2. 总体设计



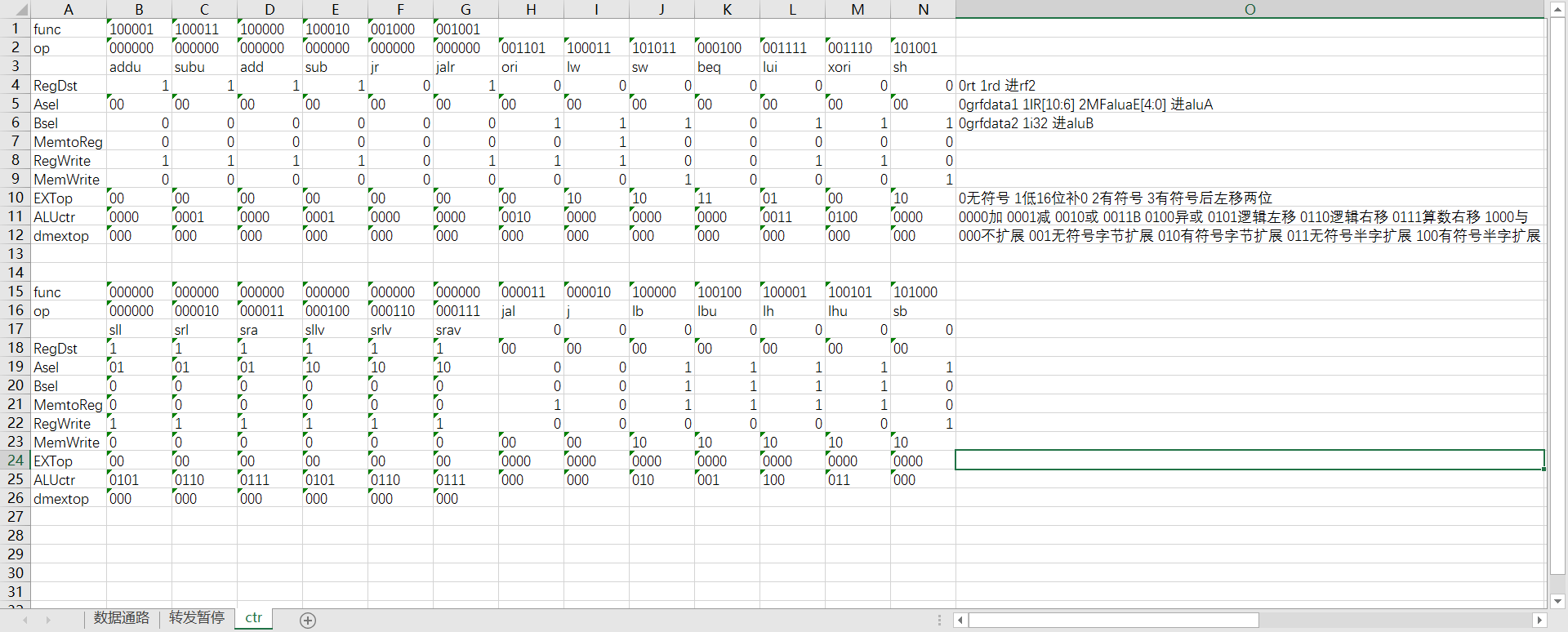
图表 1模块设计



图表 2数据通路设计

图表 3 转发暂停设计

图表 4控制器设计

1. 数据通路设计
2. datapath（数据通路）
3. 端口说明

表格 1datapath端口说明

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| clk | I | 时钟信号 |
| reset | I | 复位信号1：复位0：无效 |
| PCsel[1:0] | I | D控制器发来选择nPC信号10：转发器MFPCF结果01：NPC 00：ADD4 |
| i16[15:0] | I | D控制器发来16位立即数 |
| i26[25:0] | I | D控制器发来26位立即数 |
| RegWrite | I | W控制器发来写寄存器信号1：写寄存器0：无效 |
| MemWrite | I | M控制器发来写内存信号1：写内存0：无效 |
| EXTop[2:0] | I | D控制器发来扩展器信号000：无符号扩展 001：低16位补0 010：有符号扩展 011：有符号扩展后逻辑左移两位 |
| ALUctr[2:0] | I | E控制器发来ALU控制信号000：加运算001：减运算010：或运算011：输出写入数据2 100:异或运算 |
| WRsel[1:0] | I | D控制器发来选择寄存器A3端口信号10:31号寄存器01：IR\_W[rd] 00：IR\_W[rt] |
| WDsel[1:0] | I | W控制器发来选择寄存器输入信号00：AO\_W01：DR\_W10：PC8\_W |
| Fcmp1D[2:0] | I | 冒险单元发来选择MFcmp1D信号000：RF.V1 001：M\_Wdata 010：AO\_M 011：PC8\_M 100: PC8\_E |
| Fcmp2D[2:0] | I | 冒险单元发来选择MFcmp2D信号 000：RF.V2 001：M\_Wdata 010：AO\_M 011：PC8\_M 100: PC8\_E |
| FdmdataM | I | 冒险单元发来选择MFdmdataM信号 0：V2\_M 1：M\_Wdata |
| FaluaE[1:0] | I | 冒险单元发来选择MFaluaE信号 00：V1\_E 01：M\_Wdata 10：AO\_M 11：PC8\_M |
| FalubE[1:0] | I | 冒险单元发来选择MFalubE信号 00：V2\_E 01：M\_Wdata 10：AO\_M 11：PC8\_M |
| FPCF[2:0] | I | 冒险单元发来选择MFPCF信号 000：RF.V1 001：M\_Wdata 010：AO\_M 011：PC8\_M 100: PC8\_E |
| Bsel | I | E控制器发来ALUb选择信号 0：MFalubE选择结果 1：E32\_E |
| stall | I | 冒险单元发来选择暂停信号 1：暂停 0：无效 |
| beq | I | D控制器发来beq识别信号 1：beq 0：无效 |
| IRF[31:0] | O | 输出到D控制器的F级指令 |
| IRD[31:0] | O | 输出到E控制器的D级指令 |
| IRE[31:0] | O | 输出到M控制器的E级指令 |
| IRM[31:0] | O | 输出到W控制器的M级指令 |
| RESE[1:0] | O | 输出到冒险单元的E级Tnew状态 00：NW不写 01：写ALU 10：写DM 11：写PC |
| RESM[1:0] | O | 输出到冒险单元的M级Tnew状态 00：NW不写 01：写ALU 10：写DM 11：写PC |
| RESW[1:0] | O | 输出到冒险单元的W级Tnew状态 00：NW不写 01：写ALU 10：写DM 11：写PC |
| RFA3E[4:0] | O | 输出到冒险单元的E级指令A3寄存器 |
| RFA3M[4:0] | O | 输出到冒险单元的M级指令A3寄存器 |
| RFA3W[4:0] | O | 输出到冒险单元的W级指令A3寄存器 |

1. 功能定义

表格 2datapath功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 连接基本模块 | 通过datapath，以声明中间变量和实例化引用的方式连接各基础模块 |

1. ifu（取指令单元）
2. 端口说明

表格 3ifu端口说明

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| reset | I | 复位信号 1：复位 0：无效 |
| clk | I | 时钟信号 |
| PCsel[1:0] | I | D控制器发来选择nPC信号 10：转发器MFPCF结果 01：NPC 00：ADD4 |
| CO | I | cmp发来比较信号 1：alu两输入相等 0：alu两输入不等 |
| i16 [15:0] | I | D控制器发来16位立即数 |
| i26[25:0] | I | D控制器发来26位立即数 |
| PCtempD[31:0] | I | MFPCF转发多选器的结果 |
| beq | I | D控制器发来beq识别信号 1：beq 0：无效 |
| stall | I | 冒险单元发来选择暂停信号 1：暂停 0：无效 |
| instruction[31:0] | O | 输出的指令 |
| PC8[31:0] | O | 输出的当前PC+8 |

1. 功能定义

表格 4 ifu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被置为0x00000000 |
| 2 | 取指令 | 根据PC从IM中取出指令 |
| 3 | 计算下条指令地址 | PCPC+4 || PCreg1data || PCPC + 4 + immed32 || PC{PC[31:28], immed26, 2'b0} |
| 4 | 暂停 | Stall信号有效时，冻结PC寄存器 |

1. rf（寄存器堆）
2. 端口说明

表格 5grf端口说明

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| reset | I | 复位信号 1：复位 0：无效 |
| clk | I | 时钟信号 |
| reg1 [4:0] | I | 读寄存器号1编号 |
| Reg2 [4:0] | I | 读寄存器号2编号 |
| writereg[4:0] | I | 写寄存器编号 |
| regwrite | I | 写控制信号 1：写入 0：无效 |
| writedata[31:0] | I | 写入的32位数据 |
| data1[31:0] | O | 32位寄存器1输出 |
| data2[31:0] | O | 32位寄存器2输出 |

1. 功能定义

表格 6grf功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 复位 | 复位信号有效时，32个寄存器被置为0x00000000 |
| 2 | 写寄存器 | 写寄存器控制信号有效时，把32位数据写入寄存器 |
| 3 | 读寄存器 | 根据输入的地址读出两个寄存器中的值 |

1. alu（算术逻辑单元）
2. 端口说明

表格 7alu端口说明

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| A[31:0] | I | 32位写入数据1 |
| B[31:0] | I | 32位写入数据2 |
| ALUOp[2:0] | I | 控制信号 000：加运算 001：减运算 010：或运算 011：输出写入数据2 100:异或运算 |
| AO[31:0] | O | 32位输出数据 |

1. 功能定义

表格 8alu功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 加运算 | A+B |
| 2 | 减运算 | A-B |
| 3 | 或运算 | A|B |
| 4 | 输出写入数据2 | B |
| 5 | 异或运算 | A^B |

1. dm（数据存储器）
2. 端口说明

表格 9dm端口说明

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| D1[31:0] | I | 32位输入数据1 |
| D2[31:0] | I | 32位输入数据2 |
| CO | O | 比较结果 0：不相等 1：相等 |

1. 功能定义

表格 10dm功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 比较 | 比较两输入数据大小，相等输出1，否则输出0 |

1. cmp（比较器）
2. 端口说明

表格 11dm端口说明

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| clk | I | 时钟信号 |
| reset | I | 复位信号 1：复位 0：无效 |
| ADDR [31:0] | I | 32位写入内存地址 |
| din[31:0] | I | 32位写入数据 |
| PC8[31:0] | I | 当前PC+8 |
| MemWrite | I | 写内存控制信号 1：写入 0：无效 |
| dout[31:0] | O | 32位输出数据 |

1. 功能定义

表格 12dm功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 复位 | 复位信号有效时，内存和读出内存的寄存器被置为0x00000000 |
| 2 | 写内存 | 写内存控制信号有效时，根据输入的地址写入32位数据 |
| 3 | 读内存 | 根据输入的地址读出内存数据 |

1. ext（位扩展器）
2. 端口说明

表格 13ext端口说明

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| immed16[15:0] | I | 16位写入立即数 |
| EXTop[2:0] | I | 扩展控制信号 000：无符号扩展 001：低16位补0 010：有符号扩展 011：有符号扩展后逻辑左移两位 |
| EO[31:0] | O | 32位输出立即数 |

1. 功能定义

表格 14ext功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 无符号扩展 | 高16位补0 |
| 2 | 低16位补0 | 低16位补0 |
| 3 | 有符号扩展 | Immed[15]为1时高16位补1，为0时高16位补0 |
| 4 | 有符号扩展后逻辑左移两位 | Immed[15]为1时高16位补1，为0时高16位补0，再左移两位，溢出舍去，低2位补0 |

1. mux（多路选择器）
2. 端口说明

表格 15mux端口说明

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| A[4:0] | I | 5位输入A |
| B[4:0] | I | 5位输入B |
| C[4:0] | I | 5位输入C |
| Op[1:0] | I | 选择控制信号 10：输出C 01：输出B 00：输出A |
| O [4:0] | O | 5位输出O |
| A[31:0] | I | 32位输入A |
| B[31:0] | I | 32位输入B |
| C[31:0] | I | 32位输入C |
| D[31:0] | I | 32位输入D |
| E[31:0] | I | 32位输入E |
| op[2:0] | I | 选择控制信号 100：输出E 011：输出D 010：输出C 001：输出B 000：输出A |
| O[31:0] | O | 32位输出O |

1. 功能定义

表格 16mux功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 5位输入3选1 | option为10输出C，为01输出B，为00输出A |
| 2 | 32位输入2选1 | option为1输出B，为0输出A |
| 3 | 32位输入3选1 | option为10输出C，为01输出B，为00输出A |
| 4 | 32位输入4选1 | option为11输出D，为10输出C，为01输出B，为00输出A |
| 5 | 32位输入5选1 | option100输出E，011输出D，010输出C，001输出B，000输出A |

1. Dregs（D级流水线寄存器）
2. 端口说明

表格 17 Dregs端口说明

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| clk | I | 时钟信号 |
| reset | I | 复位信号 1：复位0：无效 |
| IR[31:0] | I | D级部件将使用的指令 |
| PC8[31:0] | I | D级部件对应指令的PC+8 |
| Stall | I | 冒险单元输入的暂停信号 |
| PCsel[1:0] | I | 将传回F级部件的PC选择信号 |
| I16[15:0] | I | 将传回F级部件的16位立即数 |
| I26[25:0] | I | 将传回F级部件的26位立即数 |
| Beq | I | 将传回F级部件的beq识别信号 |
| EXTop[2:0] | I | 扩展控制信号 000：无符号扩展 001：低16位补0 010：有符号扩展 011：有符号扩展后逻辑左移两位 |
| WRsel[1:0] | I | 选择寄存器A3端口信号 10:31号寄存器 01：IR\_W[rd] 00：IR\_W[rt] |
| IR\_D [31:0] | O | 输出到E级寄存器的指令 |
| PC8\_D[31:0] | O | 输出到E级寄存器的PC+8 |
| PCsel\_D[1:0] | O | 输出到F级寄存器的PC选择信号 |
| i16\_D | O | 将传回F级部件的16位立即数 |
| I26\_D | O | 将传回F级部件的26位立即数 |
| Beq\_D | O | 将传回F级部件的beq识别信号 |
| EXTop\_D[2:0] | O | 扩展控制信号 000：无符号扩展 001：低16位补0 010：有符号扩展 011：有符号扩展后逻辑左移两位 |
| WRsel\_D[1:0] | O | 选择寄存器A3端口信号 10:31号寄存器 01：IR\_W[rd] 00：IR\_W[rt] |

1. 功能定义

表格 18 Dregs功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 存储结果 | 存储F级部件结果，发送到D级部件或F级部件 |

1. Eregs（E级流水线寄存器）
2. 端口说明

表格 19 Eregs端口说明

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| clk | I | 时钟信号 |
| reset | I | 复位信号 1：复位 0：无效 |
| IR[31:0] | I | E级部件将使用的指令 |
| PC8[31:0] | I | E级部件对应指令的PC+8 |
| Stall | I | 冒险单元输入的暂停信号 |
| V1[31:0] | I | MFcmp1D转发而来的结果 |
| V2[31:0] | I | MFcmp2D转发而来的结果 |
| E32[31:0] | I | EXT结果 |
| RFA3[4:0] | I | E级指令要写入的寄存器编号 |
| Bsel | I | Alub多选器的选择信号 |
| ALUctr [2:0] | I | 控制信号 000：加运算 001：减运算 010：或运算 011：输出写入数据2 100:异或运算 |
| IR\_E[31:0] | O | 输出到E级寄存器的指令 |
| PC8\_E[31:0] | O | 输出到E级寄存器的PC+8 |
| RFA3E[4:0] | O | 输出到E级的要写入的寄存器编号 |
| V1[31:0] | O | MFcmp1D转发而来的结果 |
| V2[31:0] | O | MFcmp2D转发而来的结果 |
| Bsel\_E | O | Alub多选器的选择信号 |
| ALUctr\_E[2:0] | O | alu控制信号 000：加运算 001：减运算 010：或运算 011：输出写入数据2 100:异或运算 |
| Res\_E [1:0] | O | E级指令对部件的产生结果位置 ALU：在alu产生结果 DM：在dm产生结果 PC：产生PC结果 NW：nowrite，不产生结果 |

1. 功能定义

表格 20Eregs功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 存储结果 | 存储D级部件结果，发送到E级部件 |
| 2 | 产生控制转发信号 | 计算E级指令对部件的产生结果位置 |

1. Mregs（M级流水线寄存器）
2. 端口说明

表格 21Mregs端口说明

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| clk | I | 时钟信号 |
| reset | I | 复位信号 1：复位 0：无效 |
| IR[31:0] | I | M级部件将使用的指令 |
| PC8[31:0] | I | M级部件对应指令的PC+8 |
| AO[31:0] | I | ALU的结果 |
| V2[31:0] | I | MFcmp2D转发而来的结果 |
| RFA3[4:0] | I | M级指令要写入的寄存器编号 |
| MemWrite | I | Dm写入控制信号 1：写入 0：无效 |
| IR\_M[31:0] | O | 输出到W级寄存器的指令 |
| PC8\_M[31:0] | O | 输出到W级寄存器的PC+8 |
| RFA3M[4:0] | O | 输出到M级的要写入的寄存器编号 |
| AO\_M[31:0] | O | 输出到W级的alu结果 |
| V2[31:0] | O | 输出到W级的MFcmp2D转发而来的结果 |
| MemWrite | O | 输出到M级部件的Dm写入控制信号 1：写入 0：无效 |
| Res\_M[1:0] | O | M级指令对部件的产生结果位置 ALU：在alu产生结果 DM：在dm产生结果 PC：产生PC结果 NW：nowrite，不产生结果 |

1. 功能定义

表格 22Mregs功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 存储结果 | 存储E级部件结果，发送到M级部件 |
| 2 | 产生控制转发信号 | 计算M级指令对部件的产生结果位置 |

1. Wregs（W级流水线寄存器）
2. 端口说明

表格 23Wregs端口说明

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| clk | I | 时钟信号 |
| reset | I | 复位信号 1：复位 0：无效 |
| IR[31:0] | I | W级部件将使用的指令 |
| PC8[31:0] | I | W级部件对应指令的PC+8 |
| AO[31:0] | I | ALU的结果 |
| DR[31:0] | I | DM的结果 |
| RFA3[4:0] | I | W级指令要写入的寄存器编号 |
| RegWrite | I | 寄存器堆写入控制信号 1：写入 0：无效 |
| WDsel[1:0] | I | 寄存器堆写入数据选择信号 10: PC8\_W 01：DR\_W 00：AO\_W |
| PC8\_W[31:0] | O | 输出到W级部件的PC+8 |
| RFA3W[4:0] | O | 输出到W级的要写入的寄存器编号 |
| AO\_W[31:0] | O | 输出到W级部件的alu结果 |
| DR\_W[31:0] | O | 输出到W级部件DM的结果 |
| RegWrite | O | 输出到W级部件的寄存器写入控制信号 1：写入 0：无效 |
| WDsel\_W[1:0] | O | 寄存器堆写入数据选择信号 10: PC8\_W 01：DR\_W 00：AO\_W |
| Res\_W[1:0] | O | W级指令对部件的产生结果位置 ALU：在alu产生结果 DM：在dm产生结果 PC：产生PC结果 NW：nowrite，不产生结果 |

1. 功能定义

表格 24Wregs功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 存储结果 | 存储M级部件结果，发送到W级部件 |
| 2 | 产生控制转发信号 | 计算W级指令对部件的产生结果位置 |

1. Muldiv（乘除单元）
2. 端口说明

表格 25Muldiv端口说明

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| clk | I | 时钟信号 |
| reset | I | 复位信号 1：复位 0：无效 |
| A[31:0] | I | 乘除单元输入数据1 |
| B[31:0] | I | 乘除单元输入数据2 |
| op[2:0] | I | 乘除单元控制信号 000：无效 001：有符号乘 010：无符号乘 011：有符号除 100：无符号除 101：mthi 110：mtlo |
| HI[31:0] | O | HI寄存器 |
| LO[31:0] | O | LO寄存器 |
| busy | O | 乘除运算进行信号 1：正在运算 0：无效 |
| start | O | Busy信号产生信号 1：下一个时钟上升沿busy有效 0：无效 |

1. 功能定义

表格 26Muldiv功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 复位 | 复位信号有效时，HI和LO寄存器和信号被置为0 |
| 2 | 有符号乘 | 有符号乘法，其中结果高32位保存在HI，低32位保存在LO，时长5周期 |
| 3 | 无符号乘 | 无符号乘法，其中结果高32位保存在HI，低32位保存在LO，时长5周期 |
| 4 | 有符号除 | 有符号除法，其中结果余数保存在HI，商保存在LO，时长10周期 |
| 5 | 无符号除 | 无符号除法，其中结果余数保存在HI，商保存在LO，时长10周期 |
| 6 | 修改HI | 将rs的值存入HI寄存器 |
| 7 | 修改LO | 将rs的值存入LO寄存器 |
| 8 | 读取HI | 读取HI寄存器的值，写入rd |
| 9 | 读取LO | 读取LO寄存器的值，写入rd |

1. 冒险单元
2. 端口说明

表格 27hazard端口说明

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| IR[31:0] | I | D级部件将使用的指令 |
| Res\_E[1:0] | I | E级传来的控制信号 |
| Res\_M[1:0] | I | M级传来的控制信号 |
| Res\_W[1:0] | I | W级传来的控制信号 |
| A3\_E[4:0] | I | E级指令要写入的寄存器编号 |
| A3\_M[4:0] | I | M级指令要写入的寄存器编号 |
| A3\_W[4:0] | I | W级指令要写入的寄存器编号 |
| A1\_D[4:0] | I | D级指令要读入的寄存器编号1 |
| A2\_D[4:0] | I | D级指令要读入的寄存器编号2 |
| A1\_E[4:0] | I | E级指令要读入的寄存器编号1 |
| A2\_E[4:0] | I | E级指令要读入的寄存器编号2 |
| A2\_M[4:0] | I | M级指令要读入的寄存器编号 |
| Busy | I | 乘除运算进行信号 1：正在运算 0：无效 |
| start | I | Busy信号产生信号 1：下一个时钟上升沿busy有效 0：无效 |
| stall | O | 暂停信号 1：暂停 0：无效 |
| Fcmp1D[2:0] | O | 输出到D级的cmp编号1转发信号 000：RF.V1 001：M\_Wdata 010：AO\_M 011：PC8\_M 100: PC8\_E |
| Fcmp2D[2:0] | O | 输出到D级的cmp编号2转发信号 000：RF.V2 001：M\_Wdata 010：AO\_M 011：PC8\_M 100: PC8\_E |
| FaluaE[1:0] | O | 输出到E级ALUa的转发信号 00：V1\_E 01：M\_Wdata 10：AO\_M 11：PC8\_M |
| FalubE[1:0] | O | 输出到E级ALUb的转发信号 00：V2\_E 01：M\_Wdata 10：AO\_M 11：PC8\_M |
| FPCF [2:0] | O | 输出到F级PC的转发信号 000：RF.V1 001：M\_Wdata 010：AO\_M 011：PC8\_M 100: PC8\_E |
| FdmdataM | O | 输出到M级DMin的转发信号 0：V2\_M 1：M\_Wdata |

1. 功能定义

表格 28hazard功能定义

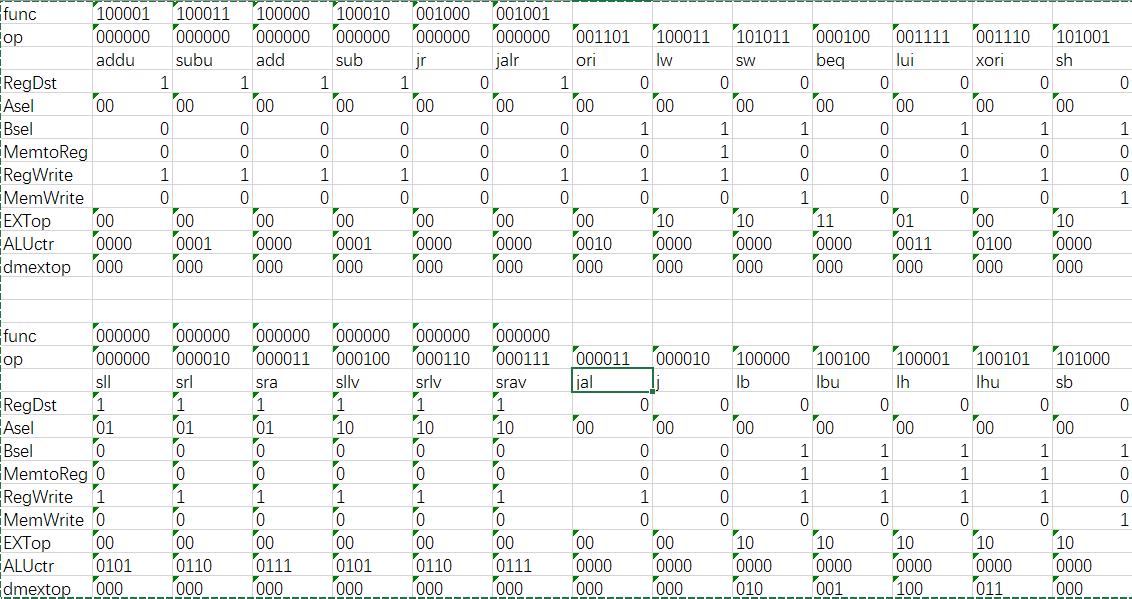
|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 产生暂停信号 | 根据Tuse和Tnew产生暂停信号 |
| 2 | 产生转发信号 | 根据Tuse和Tnew产生转发信号 |

1. 控制器设计
2. 端口说明

表格 29ctr端口说明

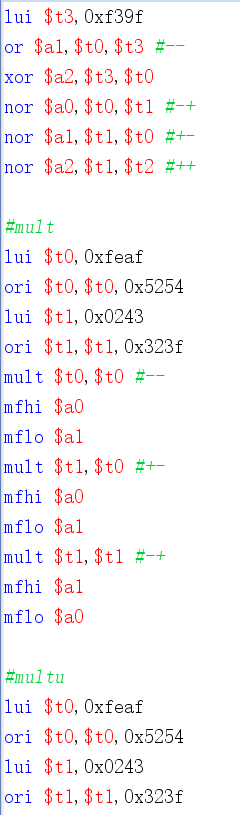
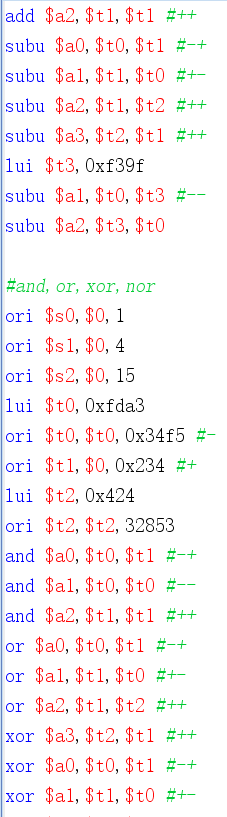
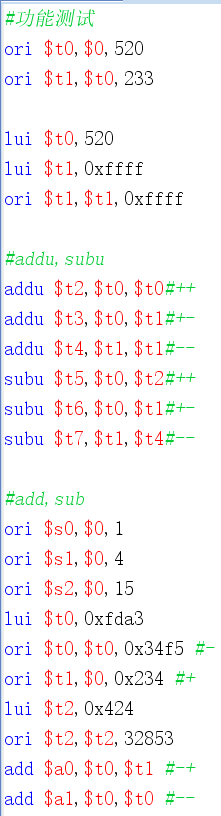
|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| instruction[31:0] | I | 32位指令 |
| WR\_sel | O | grf写寄存器决定信号 0：rd 1：rt 2：31号寄存器 |
| A\_sel | O | alu输入数据A决定信号 0：MFaluaE 1：IR[10:6] 2：MFaluaE[4:0] |
| B\_sel | O | alu输入数据B决定信号 1：32位立即数 0：GRF寄存器2输出值 |
| RegWrite | O | grf写寄存器信号 1：写寄存器 0：无效 |
| MemWrite | O | 写内存dm信号 1：写入内存 0：无效 |
| PC\_sel | O | PC跳转信号`ADD4：跳转PC+4 `NPC：跳转npc `RFV1：跳转寄存器堆1 |
| EXTop[1:0] | O | 扩展控制信号 00：无符号扩展 01：低16位补0 10：有符号扩展 11：有符号扩展后逻辑左移两位 |
| ALUctr[3:0] | O | alu控制信号0000加 0001减 0010或 0011B 0100异或 0101逻辑左移 0110逻辑右移 0111算数右移 1000与 1001或非 1010有符号小于置1 1011无符号小于置1 |
| beq | O | Beq指示信号 0：无效 1：beq |
| bne | O | bne指示信号 0：无效 1：bne |
| Bgez | O | Bgez指示信号 0：无效 1：Bgez |
| Blez | O | Blez指示信号 0：无效 1：Blez |
| Bgtz | O | Bgtz指示信号 0：无效 1：Bgtz |
| blez | O | blez指示信号 0：无效 1：blez |
| Muldivop[2:0] | O | 乘除模块控制信号 0：无效 1：mult 2：multu 3：div 4：divu 5：mthi 6：mtlo |
| WDsel[1:0] | O | 写入寄存器堆数据选择信号 0：DR\_W 1：PC8\_W 2：AO\_W |
| Dmexto[2:0] | O | Dm扩展器控制信号 0：无效 1：lbu 2：lb 3：lhu 4：lh |

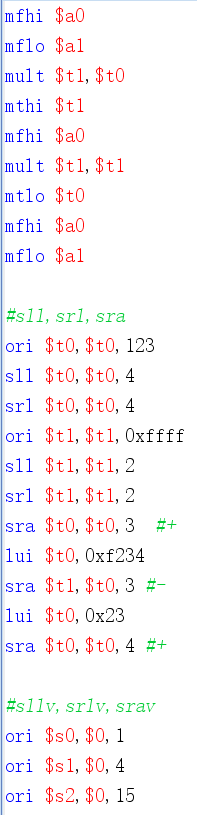
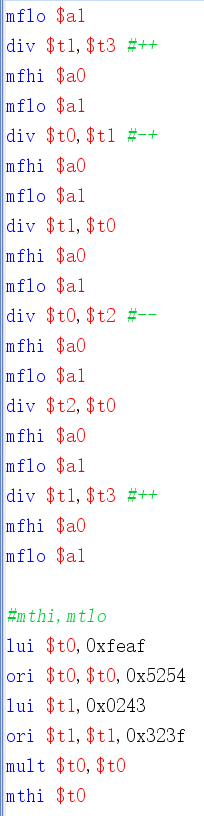
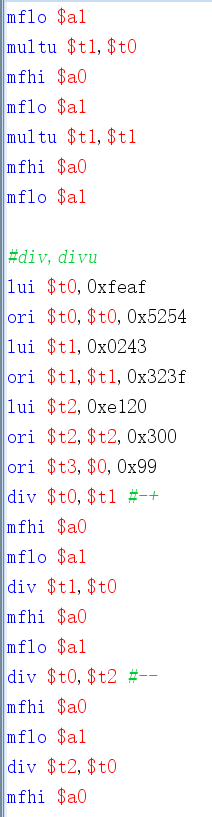
1. 真值表

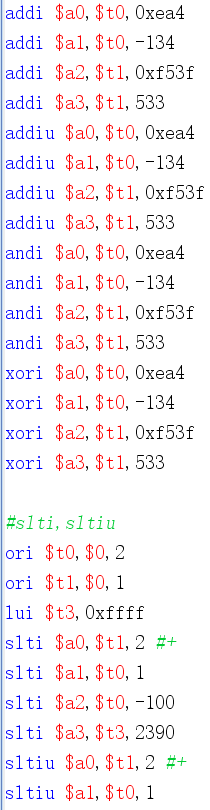
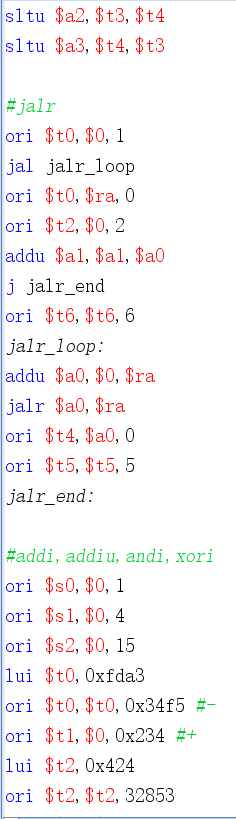
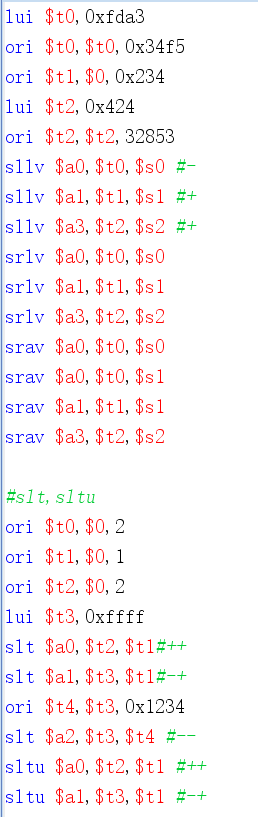
表格 30ctr真值表

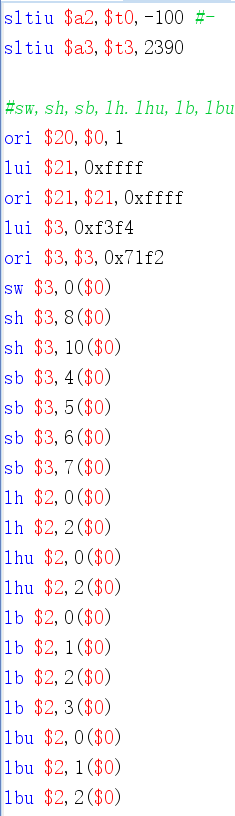
1. 测试
2. 测试程序

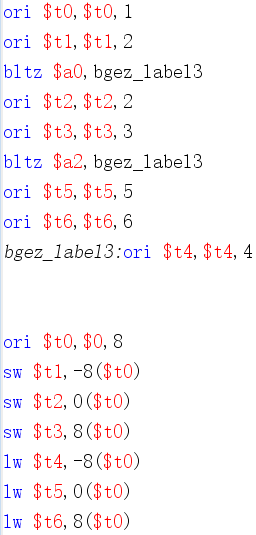
功能测试：



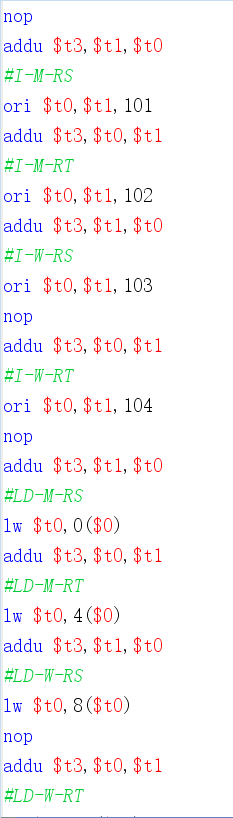
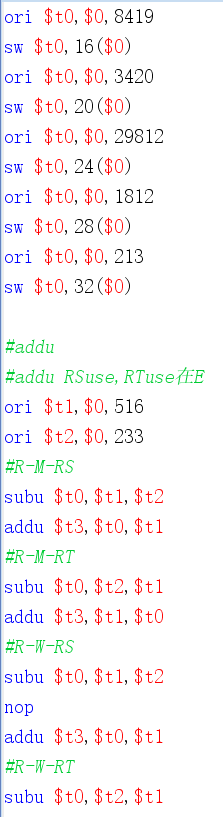
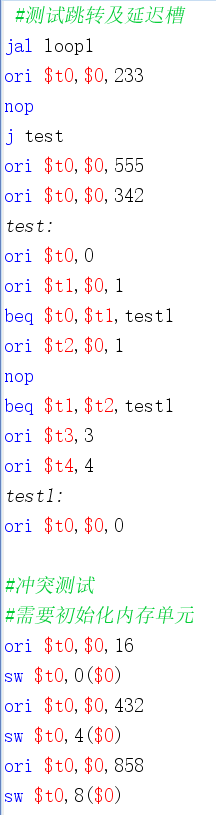


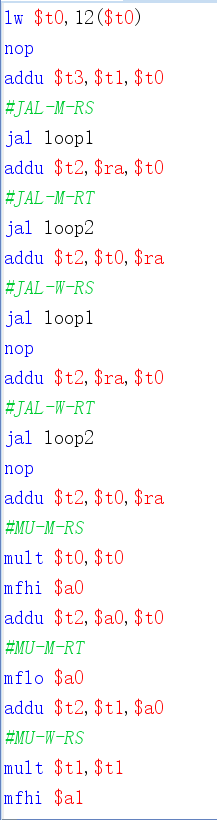


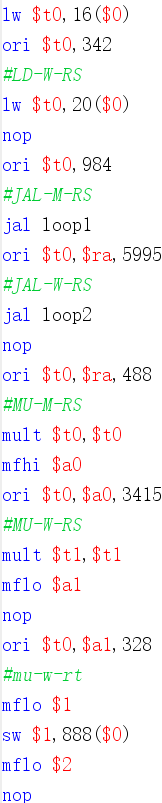
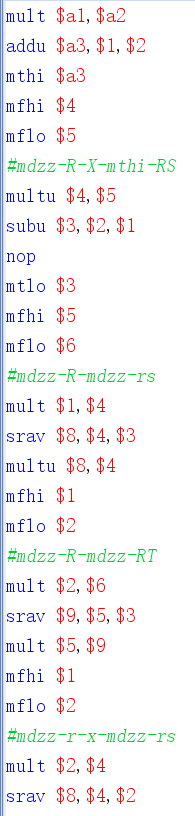


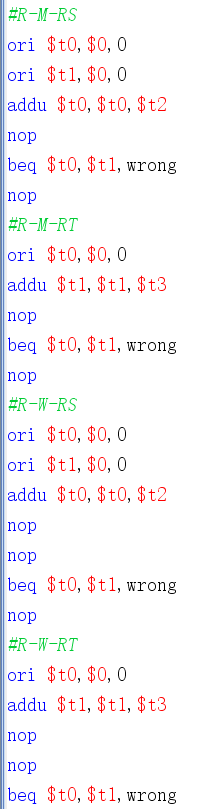
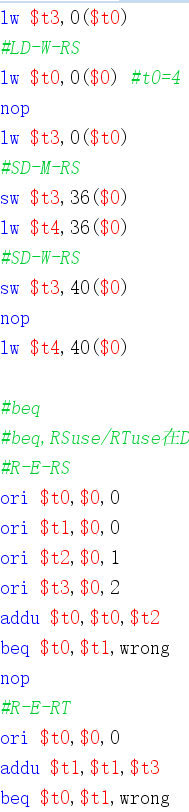
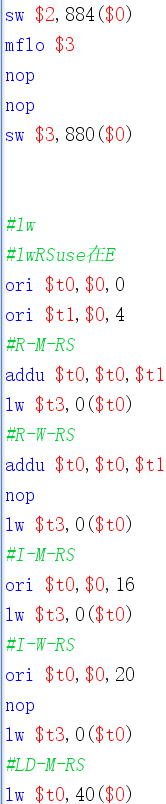


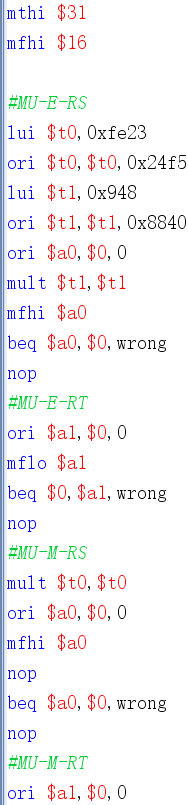
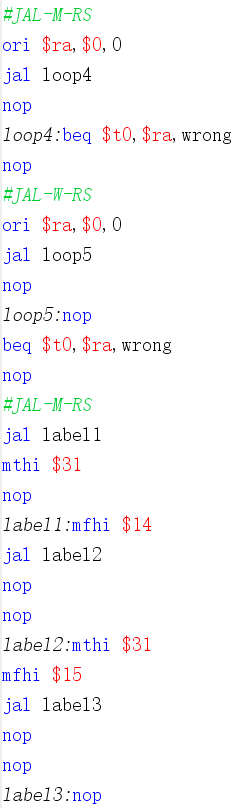
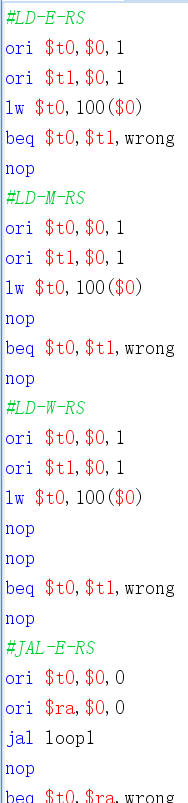
转发暂停测试：

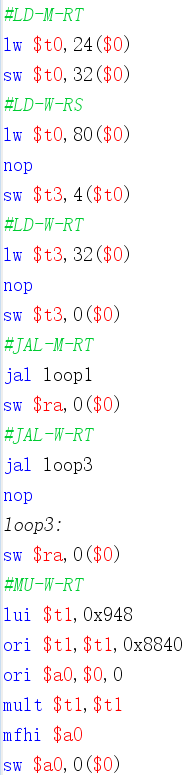
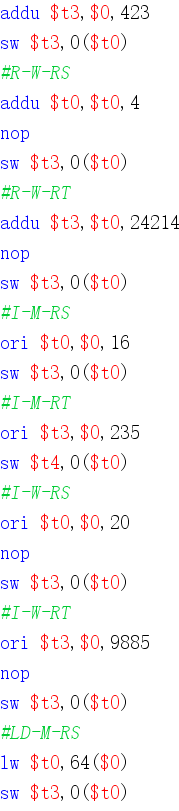
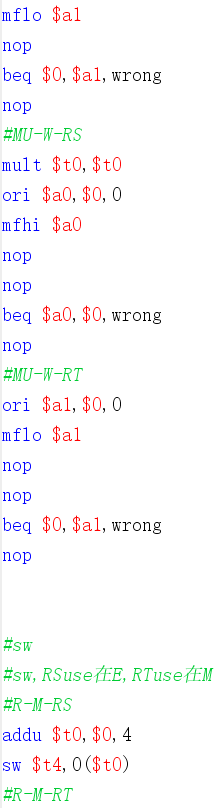


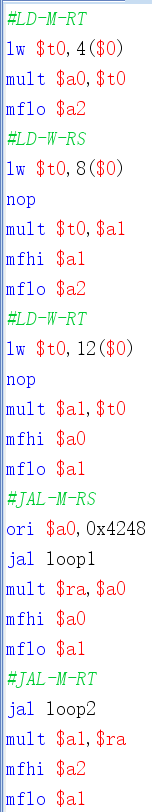
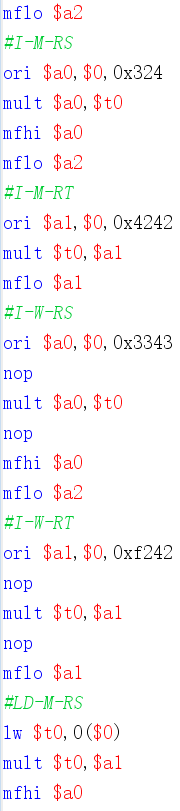


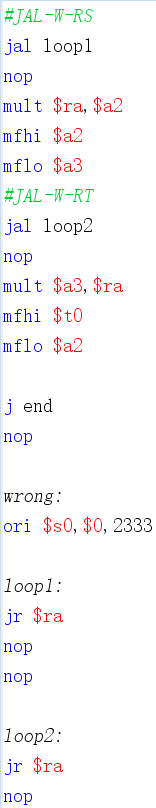




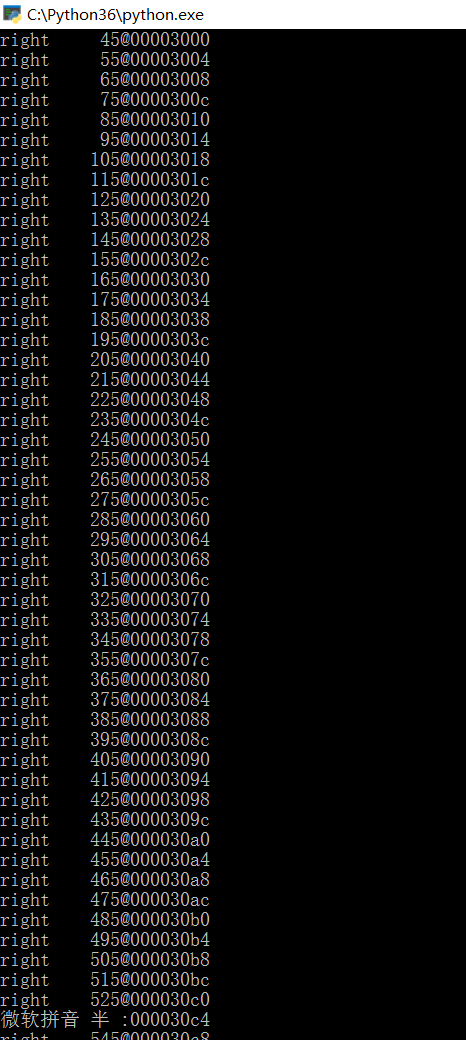
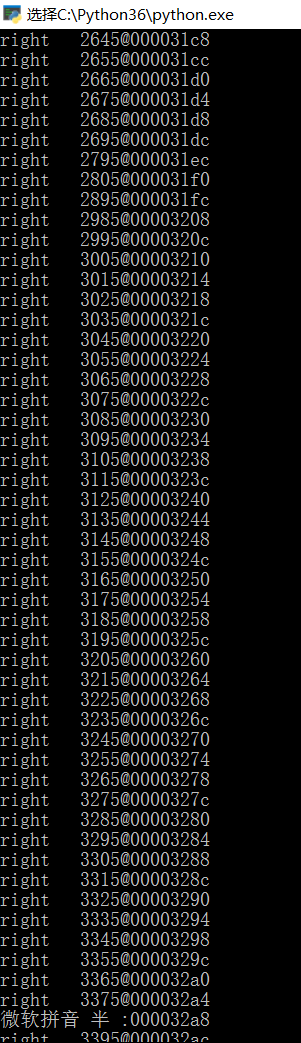
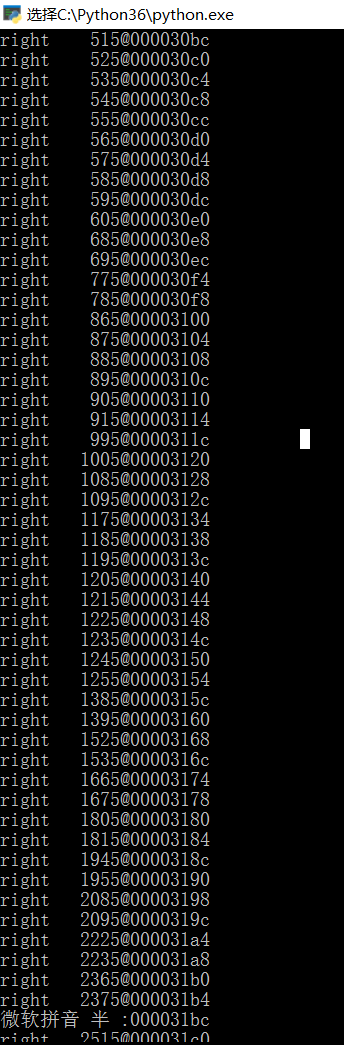


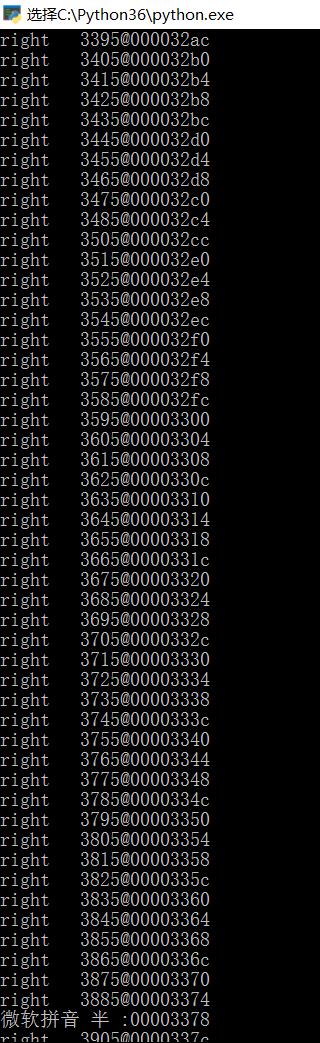


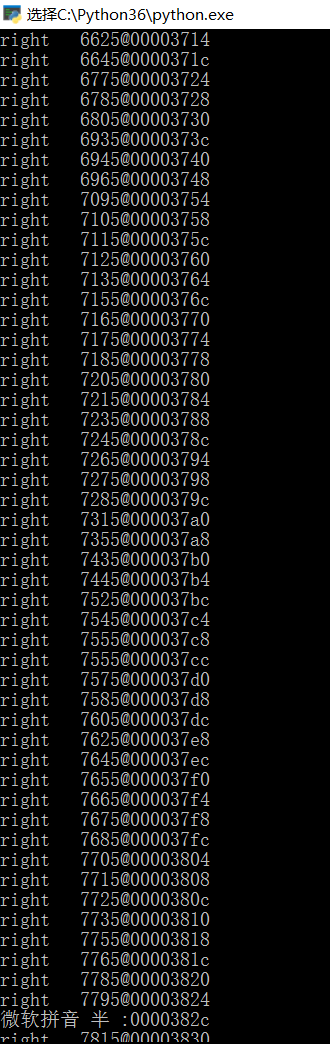
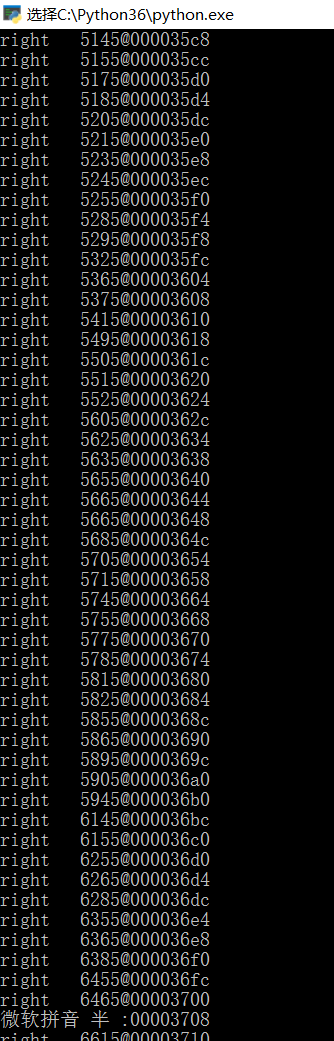
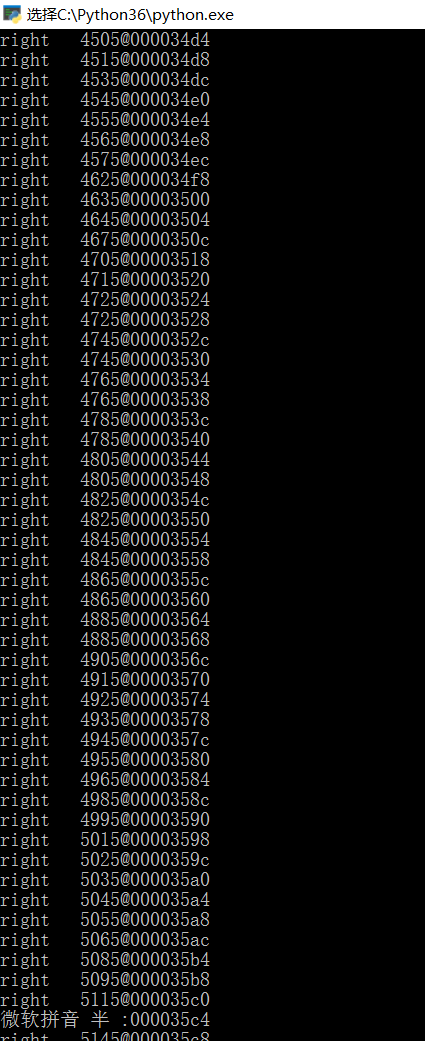


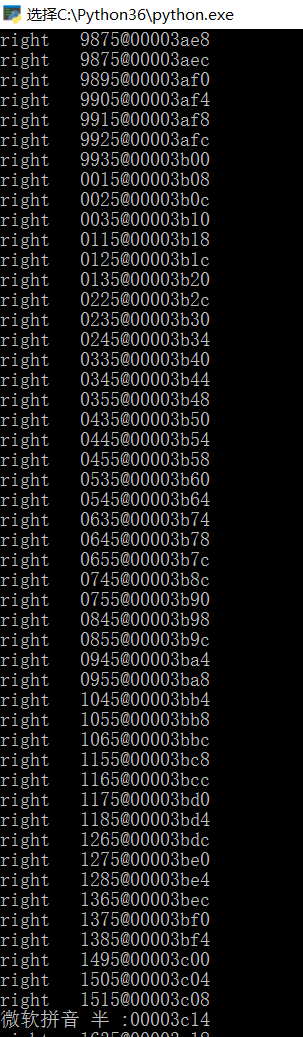
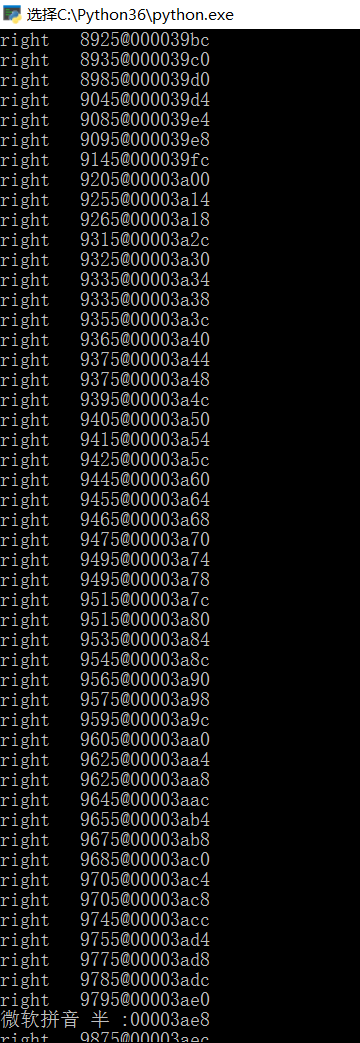
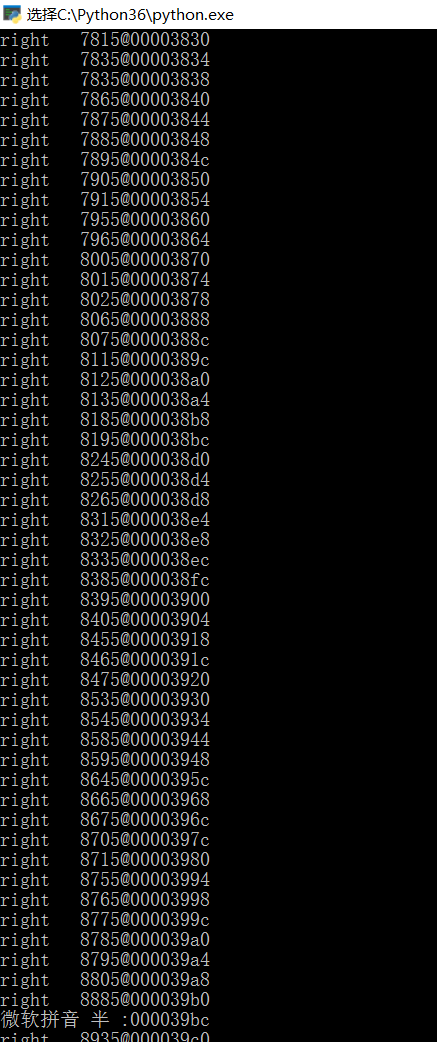


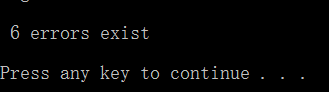
1. 期望输出与实际输出比对









1. 结论

期望输出与实际输出相同。

1. 思考题
2. 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？

乘除指令中我们需要将其计算结果分别存入HI寄存器和LO寄存器，所以需要独立的特殊寄存器HI寄存器和LO寄存器。乘除法在实际实现中是比其他的运算需要更多的时间，倘若整合到ALU，势必会大大增加流水线的周期，对于cpu的执行效率来说是不利的，所以需要有单独的乘除法部件。

独立的LO和HI我认为一是为了配合乘除相关指令，二是由于在汇编指令中乘除相关指令一半会配合取HI寄存器和LO寄存器的操作，为了转发方便节省周期数，需要有独立的HI和LO寄存器。

1. 参照你对延迟槽的理解，试解释“乘除槽”。

跳转指令需要经过一个周期的判断和地址输送才能改变下一个地址来读取下一条指令，所以势必需要一个周期进行暂停，所以在一个周期去处理一条不相关指令时可行的，可以提高cpu的执行效率。我们称这个跳转的空档为延迟槽，原因来源于跳转指令的延迟。

而对于乘除指令，其在进入E级部件后也会产生5-10个周期的延迟，其相关指令势必要进行暂停，但是在这个空档中是可以处理一些不相关指令的，也可以提高cpu的执行效率。所以相比较延迟槽，这个进行乘除法操作的延迟，可以被称为乘除槽。

1. 为何上文文末提到的lb等指令使用的数据扩展模块应在 MEM/WB 之后，而不能在 DM 之后?

MEM部件的延迟最高，而流水线的执行周期是根据整条流水线中延迟最高的部件决定的，倘若把数据扩展模块放在DM之后的话，其势必会增加MEM部件的延迟，同时增加了整条流水线的执行周期，对于cpu的运行效率有很大影响。

但是一定会转发（无论转发过去是否被选择），数据的经过通路是相同的，所以对于时钟周期实际上没有影响，放在DM之后也可以。

如果考虑了异常和中断，比如还要几个周期才能产生数据，那么是让load指令停在E/M还是M/W，可能会有影响？（猜测）

1. 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）

当进行sb指令时，若DM按字来访问，需要提出该字地址的整个字后，修改相应字节的内容，再存回去。而若把DM分成四个字节，在存储字节操作上我们可以直接将相应字节存入相应地址，而不需要把数据读出再操作。虽然在我们实现中看不出来时间的差异，但是在实际中按字节访问在效率和时间利用上是比按字访问内存更有优势的。

1. 如何概括你所设计的CPU的设计风格？为了对抗复杂性你采取了哪些抽象和规范手段？

我的CPU设计风格是规划者型，对于每个控制信号以不同指令来判断信号。

以转发、暂停为例，分析新指令是否需要rs和rt和是否写寄存器来分析冲突，然后构建相应的Tuse和Tnew信号，就可以完成一条新增指令的冲突分析。在最初的冲突构建中已经考虑到了所有的冲突情况，把所有条件抽象出来。所以在新增指令到来时，只需要将指令添加到相对应的抽象信号中即可。

1. 你对流水线CPU设计风格有何见解？
2. 设计中善于使用宏定义是很有优势的，用简洁明了的名称来代替二进制编码虽然在代码开头需要更多处理，但是在后面的编写过程，尤其是条件判断和信号生成上会更加的直观明显。
3. 在刚开始设计的时候，最好能够考虑到它的可扩展性，体现在控制信号的位数上，或者是否可能整合某几个信号，比如cmp单元我用多个信号分别表示大于0、小等0、大等0、小于0、等于0、不等于0、大于、小等、大等、小于、等于、不等，其实完全可以用两个多位信号，分别表示和0比较结果和两个数比较结果。
4. 在本实验中你遇到了哪些不同指令组合产生的冲突？你又是如何解决的？相应的测试样例是什么样的？请有条理的罗列出来。

cal\_r型(add,addu,sub,subu,sllv,srlv,srav,and,or,xor,nor,slt,sltu)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 编号 | 类型 | 前序指令 | 冲突时钱前序指令位置 | 冲突寄存器 | 测试样例 | 解决方法 |
| 1 | R-M-RS | cal\_r | MEM | rs | Addu $1,$2,$3  Addu $4,$1,$5 | 从M级将AO\_M转发到E级部件alua |
| 2 | R-M-RT | cal\_r | MEM | rt | Addu $1,$2,$3  Addu $4,$5,$1 | 从M级将AO\_M转发到E级部件alub |
| 3 | R-W-RS | cal\_r | WB | rs | Addu $1,$2,$3  nop  Addu $4,$1,$5 | 从W级将AO\_W转发到E级部件alua |
| 4 | R-W-RT | cal\_r | WB | rt | Addu $1,$2,$3  nop  Addu $4,$5,$1 | 从W级将AO\_W转发到E级部件alub |
| 5 | I-M-RS | cal\_i | MEM | rs | Ori $1,$2,1  Addu $3,$1,$4 | 从M级将AO\_M转发到E级部件alua |
| 6 | I-M-RT | cal\_i | MEM | rt | Ori $1,$2,1  Addu $3,$4,$1 | 从M级将AO\_M转发到E级部件alub |
| 7 | I-W-RS | cal\_i | WB | rs | Ori $1,$2,1  nop  Addu $3,$1,$4 | 从W级将AO\_W转发到E级部件alua |
| 8 | I-W-RT | cal\_i | WB | rt | Ori $1,$2,1  nop  Addu $3,$4,$1 | 从W级将AO\_W转发到E级部件alub |
| 9 | LD-M-RS | load | MEM | rs | Lw $1,0($2)  Addu $3,$1,$4 | 暂停  从W级将DR\_W转发到E级部件alua |
| 10 | LD-M-RT | load | MEM | rt | Lw $1,0($2)  Addu $3,$4,$1 | 暂停  从W级将DR\_W转发到E级部件alub |
| 11 | LD-W-RS | load | WB | rs | Lw $1,0($2)  nop  Addu $3,$1,$4 | 从W级将DR\_W转发到E级部件alua |
| 12 | LD-W-RT | load | WB | rt | Lw $1,0($2)  nop  Addu $3,$4,$1 | 从W级将DR\_W转发到E级部件alub |
| 13 | JAL-M-RS | jal | MEM | rs | Jal loop  Addu $1,$2,$31 | 从M级将PC8\_M转发到E级部件alua |
| 14 | JAL-M-RT | jal | MEM | rt | Jal loop  Addu $1,$31,$2 | 从M级将PC\_M转发到E级部件alub |
| 15 | JAL-W-RS | jal | WB | rs | Jal loop  nop  Addu $1,$2,$31 | 从W级将PC8\_W转发到E级部件alua |
| 16 | JAL-W-RT | jal | WB | rt | Jal loop  nop  Addu $1,$31,$2 | 从W级将PC8\_W转发到E级部件alub |

cal\_i型(addi,addiu,andi,ori,xori,lui,slti,sltiu)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 编号 | 类型 | 前序指令 | 冲突时钱前序指令位置 | 冲突寄存器 | 测试样例 | 解决方法 |
| 1 | R-M-RS | cal\_r | MEM | rs | Addu $1,$2,$3  Ori $4,$1,1 | 从M级将AO\_M转发到E级部件alua |
| 2 | R-W-RS | cal\_r | WB | rs | Addu $1,$2,$3  nop  Ori $4,$1,1 | 从W级将AO\_W转发到E级部件alua |
| 3 | I-M-RS | cal\_i | MEM | rs | Ori $1,$2,1  Ori $3,$1,1 | 从M级将AO\_M转发到E级部件alua |
| 4 | I-W-RS | cal\_i | WB | rs | Ori $1,$2,1  nop  Ori $3,$1,1 | 从W级将AO\_W转发到E级部件alua |
| 5 | LD-M-RS | load | MEM | rs | Lw $1,0($2)  Ori $3,$1,1 | 暂停  从W级将DR\_W转发到E级部件alua |
| 6 | LD-W-RS | load | WB | rs | Lw $1,0($2)  nop  Ori $3,$1,1 | 从W级将DR\_W转发到E级部件alua |
| 7 | JAL-M-RS | jal | MEM | rs | Jal loop  Ori $2,$31,1 | 从M级将PC8\_M转发到E级部件alua |
| 8 | JAL-W-RS | jal | WB | rs | Jal loop  nop  Ori $2,$31,1 | 从W级将PC8\_W转发到E级部件alua |

Load型(lw,lh,lhu,lb,lbu)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 编号 | 类型 | 前序指令 | 冲突时钱前序指令位置 | 冲突寄存器 | 测试样例 | 解决方法 |
| 1 | R-M-RS | cal\_r | MEM | rs | Addu $1,$2,$3  Lw $4,0($1) | 从M级将AO\_M转发到E级部件alua |
| 2 | R-W-RS | cal\_r | WB | rs | Addu $1,$2,$3  nop  Lw $4,0($1) | 从W级将AO\_W转发到E级部件alua |
| 3 | I-M-RS | cal\_i | MEM | rs | Ori $1,$2,1  Lw $3,0($1) | 从M级将AO\_M转发到E级部件alua |
| 4 | I-W-RS | cal\_i | WB | rs | Ori $1,$2,1  nop  Lw $3,0($1) | 从W级将AO\_W转发到E级部件alua |
| 5 | LD-M-RS | load | MEM | rs | Lw $1,0($2)  Lw $3,0($1) | 暂停  从W级将DR\_W转发到E级部件alua |
| 6 | LD-W-RS | load | WB | rs | Lw $1,0($2)  nop  Lw $3,0($1) | 从W级将DR\_W转发到E级部件alua |
| 7 | JAL-M-RS | jal | MEM | rs | Jal loop  Lw $2,0($31) | 从M级将PC8\_M转发到E级部件alua |
| 8 | JAL-W-RS | jal | WB | rs | Jal loop  nop  Lw $2,0($31) | 从W级将PC8\_W转发到E级部件alua |

Store型(sw,sh,sb)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 编号 | 类型 | 前序指令 | 冲突时钱前序指令位置 | 冲突寄存器 | 测试样例 | 解决方法 |
| 1 | R-M-RS | cal\_r | MEM | rs | Addu $1,$2,$3  Sw $4,0($1) | 从M级将AO\_M转发到E级部件alua |
| 2 | R-W-RT | cal\_r | WB | rt | Addu $1,$2,$3  Sw $1,0($4) | 从W级将AO\_W转发到M级部件dmin |
| 3 | R-W-RS | cal\_r | WB | rs | Addu $1,$2,$3  nop  Sw $4,0($1) | 从W级将AO\_W转发到E级部件alua |
| 4 | I-M-RS | cal\_i | MEM | rs | Ori $1,$2,1  Sw $3,0($1) | 从M级将AO\_M转发到E级部件alua |
| 5 | I-W-RT | cal\_i | WB | rt | Ori $1,$2,1  Sw $1,0($3) | 从W级将AO\_W转发到M级部件dmin |
| 6 | I-W-RS | cal\_i | WB | rs | Ori $1,$2,1  nop  Sw $3,0($1) | 从W级将AO\_W转发到E级部件alua |
| 7 | LD-M-RS | load | MEM | rs | Lw $1,0($2)  Sw $3,0($1) | 暂停  从W级将DR\_W转发到E级部件alua |
| 8 | LD-W-RT | load | WB | rt | Lw $1,0($2)  Sw $1,0($3) | 从W级将DR\_W转发到M级部件dmin |
| 9 | LD-W-RS | load | WB | rs | Lw $1,0($2)  nop  Sw $3,0($1) | 从W级将DR\_W转发到E级部件alua |
| 10 | JAL-M-RS | jal | MEM | rs | Jal loop  Sw $2,0($31) | 从M级将PC8\_M转发到E级部件alua |
| 11 | JAL-W-RT | jal | WB | rt | Jal loop  Sw $31,0($2) | 从W级将PC8\_W转发到M级部件dmin |
| 12 | JAL-W-RS | jal | WB | rs | Jal loop  nop  Sw $2,0($31) | 从W级将PC8\_W转发到E级部件alua |

B(Beq,bne,bgez,bgtz,blez,bltz)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 编号 | 类型 | 前序指令 | 冲突时钱前序指令位置 | 冲突寄存器 | 测试样例 | 解决方法 |
| 1 | R-M-RS | cal\_r | MEM | rs | Addu $1,$2,$3  Beq $1,$4,loop | 暂停  从M级将AO\_M转发到D级部件cmpD1 |
| 2 | R-M-RT | cal\_r | MEM | rt | Addu $1,$2,$3  Beq $4,$1,loop | 暂停  从M级将AO\_M转发到D级部件cmpD2 |
| 3 | R-M-RS | cal\_r | MEM | rs | Addu $1,$2,$3  nop  Beq $1,$4,loop | 从M级将AO\_M转发到D级部件cmpD1 |
| 4 | R-M-RT | cal\_r | MEM | rt | Addu $1,$2,$3  nop  Beq $1,$4,loop | 从M级将AO\_M转发到D级部件cmpD2 |
| 5 | I-M-RS | cal\_i | MEM | rs | Ori $1,$2,1  Beq $1,$3,loop | 暂停  从M级将AO\_M转发到D级部件cmpD1 |
| 6 | I-M-RT | cal\_i | MEM | rt | Ori $1,$2,1  Beq $3,$1,loop | 暂停  从M级将AO\_M转发到D级部件cmpD2 |
| 7 | I-M-RS | cal\_i | MEM | rs | Ori $1,$2,1  nop  Beq $1,$3,loop | 从M级将AO\_M转发到D级部件cmpD1 |
| 8 | I-M-RT | cal\_i | MEM | rt | Ori $1,$2,1  nop  Beq $3,$1,loop | 从M级将AO\_M转发到D级部件cmpD2 |
| 9 | LD-M-RS | load | MEM | rs | Lw $1,0($2)  Beq $1,$3,loop | 暂停  暂停  从W级将DR\_W转发到D级部件cmpD1 |
| 10 | LD-M-RT | load | MEM | rt | Lw $1,0($2)  Beq $3,$1,loop | 暂停  暂停  从W级将DR\_W转发到D级部件cmpD2 |
| 11 | LD-W-RS | load | WB | rs | Lw $1,0($2)  nop  Beq $1,$3,loop | 暂停  从D级将DR\_W转发到E级部件cmpD1 |
| 12 | LD-W-RT | load | WB | rt | Lw $1,0($2)  nop  Beq $3,$1,loop | 暂停  从W级将DR\_W转发到D级部件cmpD2 |
| 13 | JAL-E-RS | jal | EX | rs | Jal loop  Beq $31,$2,loop | 从E级将PC8\_E转发到D级部件cmpD1 |
| 14 | JAL-E-RT | jal | EX | rt | Jal loop  Beq $2,$31,loop | 从E级将PC8\_E转发到D级部件cmpD2 |
| 15 | JAL-M-RS | jal | MEM | rs | Jal loop  nop  Beq $31,$2,loop | 从M级将PC8\_M转发到D级部件cmpD1 |
| 16 | JAL-M-RT | jal | MEM | rt | Jal loop  nop  Beq $2,$31,loop | 从M级将PC8\_M转发到D级部件cmpD2 |

J(Jr,jalr)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 编号 | 类型 | 前序指令 | 冲突时钱前序指令位置 | 冲突寄存器 | 测试样例 | 解决方法 |
| 1 | R-M-RS | cal\_r | MEM | rs | Addu $1,$2,$3  Jr $1 | 暂停  从M级将AO\_M转发到D级部件cmpD1 |
| 2 | R-M-RS | cal\_r | MEM | rs | Addu $1,$2,$3  nop  Beq $1,$4,loop | 从M级将AO\_M转发到D级部件cmpD1 |
| 3 | I-M-RS | cal\_i | MEM | rs | Ori $1,$2,1  Beq $1,$3,loop | 暂停  从M级将AO\_M转发到D级部件cmpD1 |
| 4 | I-M-RS | cal\_i | MEM | rs | Ori $1,$2,1  nop  Beq $1,$3,loop | 从M级将AO\_M转发到D级部件cmpD1 |
| 5 | LD-M-RS | load | MEM | rs | Lw $1,0($2)  Beq $1,$3,loop | 暂停  暂停  从W级将DR\_W转发到D级部件cmpD1 |
| 6 | LD-W-RS | load | WB | rs | Lw $1,0($2)  nop  Beq $1,$3,loop | 暂停  从W级将DR\_W转发到D级部件cmpD1 |
| 7 | JAL-E-RS | jal | EX | rs | Jal loop  Beq $31,$2,loop | 从E级将PC8\_E转发到D级部件cmpD1 |
| 8 | JAL-M-RS | jal | MEM | rs | Jal loop  nop  Beq $31,$2,loop | 从M级将PC8\_M转发到D级部件cmpD1 |

s(sll,srl,sra)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 编号 | 类型 | 前序指令 | 冲突时钱前序指令位置 | 冲突寄存器 | 测试样例 | 解决方法 |
| 1 | R-M-RS | cal\_r | MEM | rt | Addu $1,$2,$3  sll $4,$1,1 | 从M级将AO\_M转发到E级部件alub |
| 2 | R-W-RS | cal\_r | WB | rt | Addu $1,$2,$3  nop  sll $4,$1,1 | 从W级将AO\_W转发到E级部件alub |
| 3 | I-M-RS | cal\_i | MEM | rt | Ori $1,$2,1  sll $3,$1,1 | 从M级将AO\_M转发到E级部件alub |
| 4 | I-W-RS | cal\_i | WB | rt | Ori $1,$2,1  nop  sll $3,$1,1 | 从W级将AO\_W转发到E级部件alub |
| 5 | LD-M-RS | load | MEM | rt | Lw $1,0($2)  sll $3,$1,1 | 暂停  从W级将DR\_W转发到E级部件alub |
| 6 | LD-W-RS | load | WB | rt | Lw $1,0($2)  nop  sll $3,$1,1 | 从W级将DR\_W转发到E级部件alub |
| 7 | JAL-M-RS | jal | MEM | rt | Jal loop  sll $2,$31,1 | 从M级将PC8\_M转发到E级部件alub |
| 8 | JAL-W-RS | jal | WB | rt | Jal loop  nop  sll $2,$31,1 | 从W级将PC8\_W转发到E级部件alub |