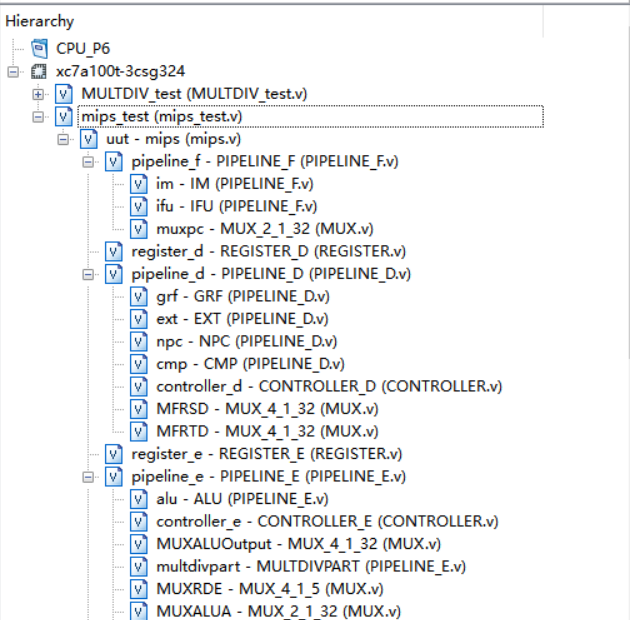
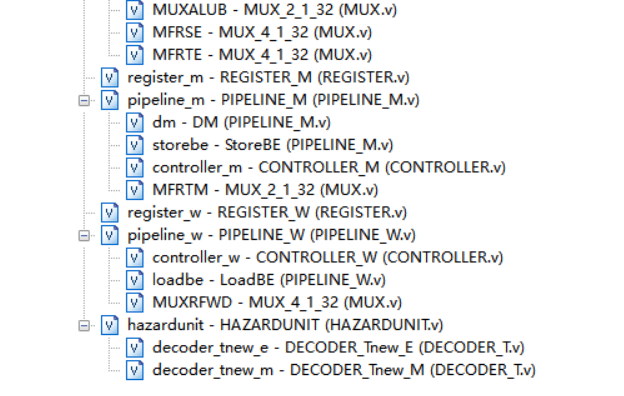
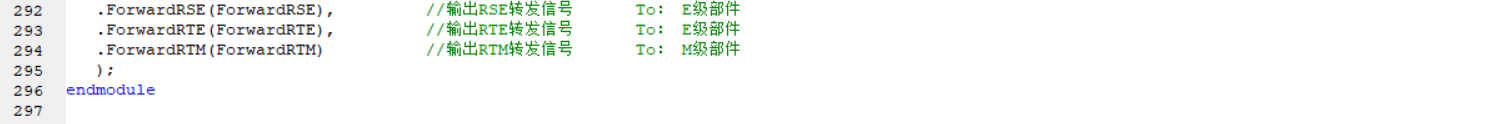
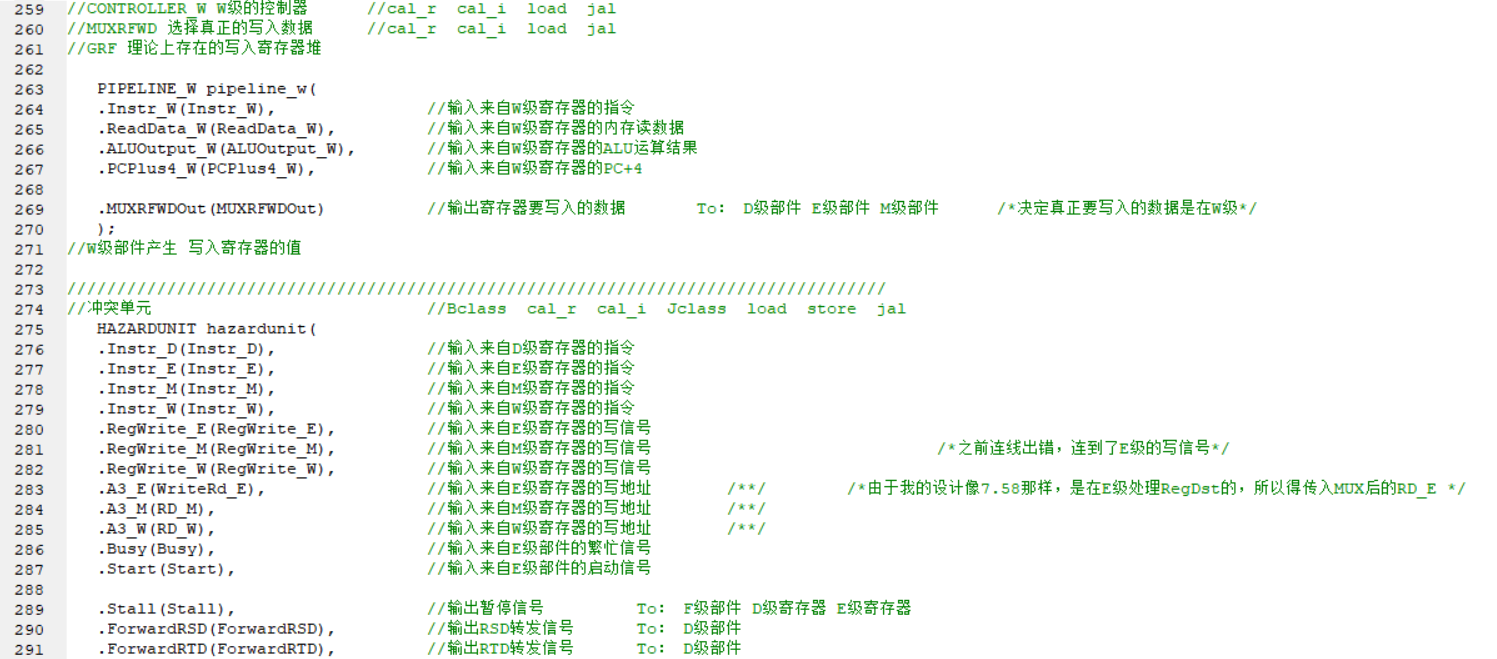
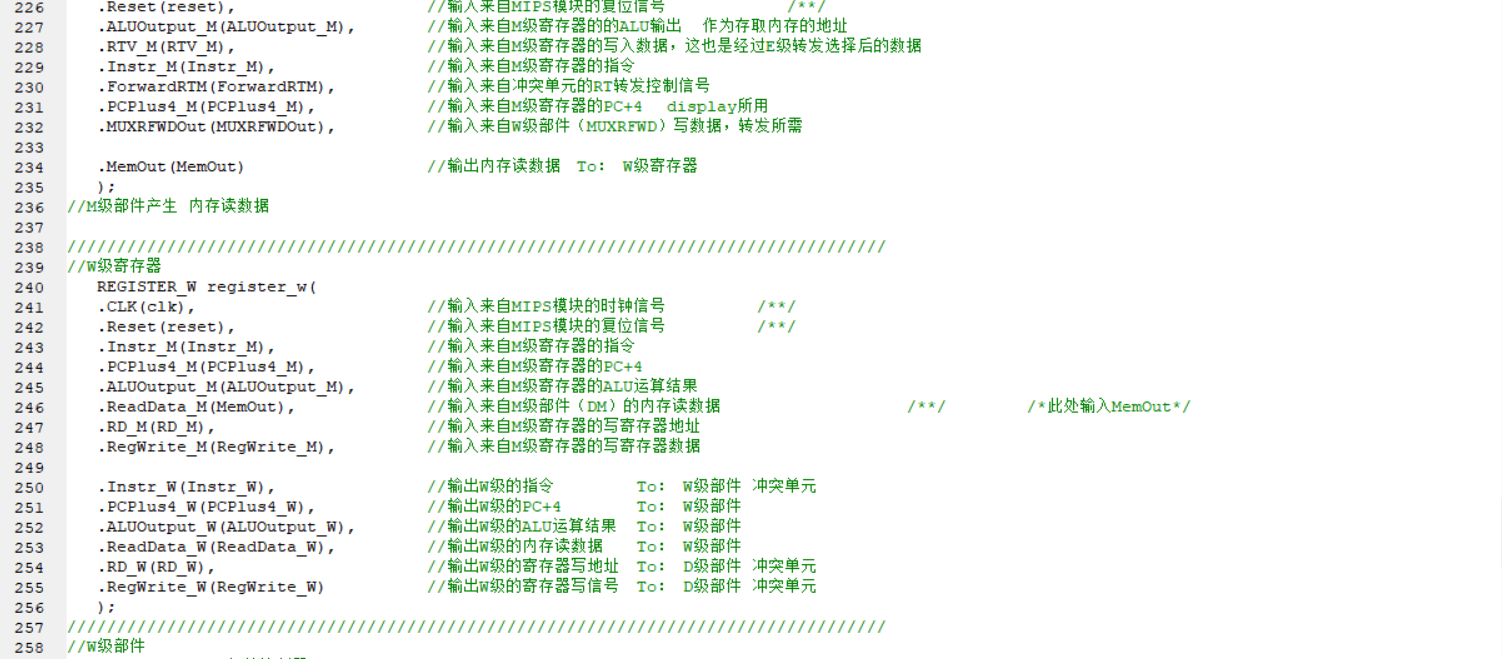
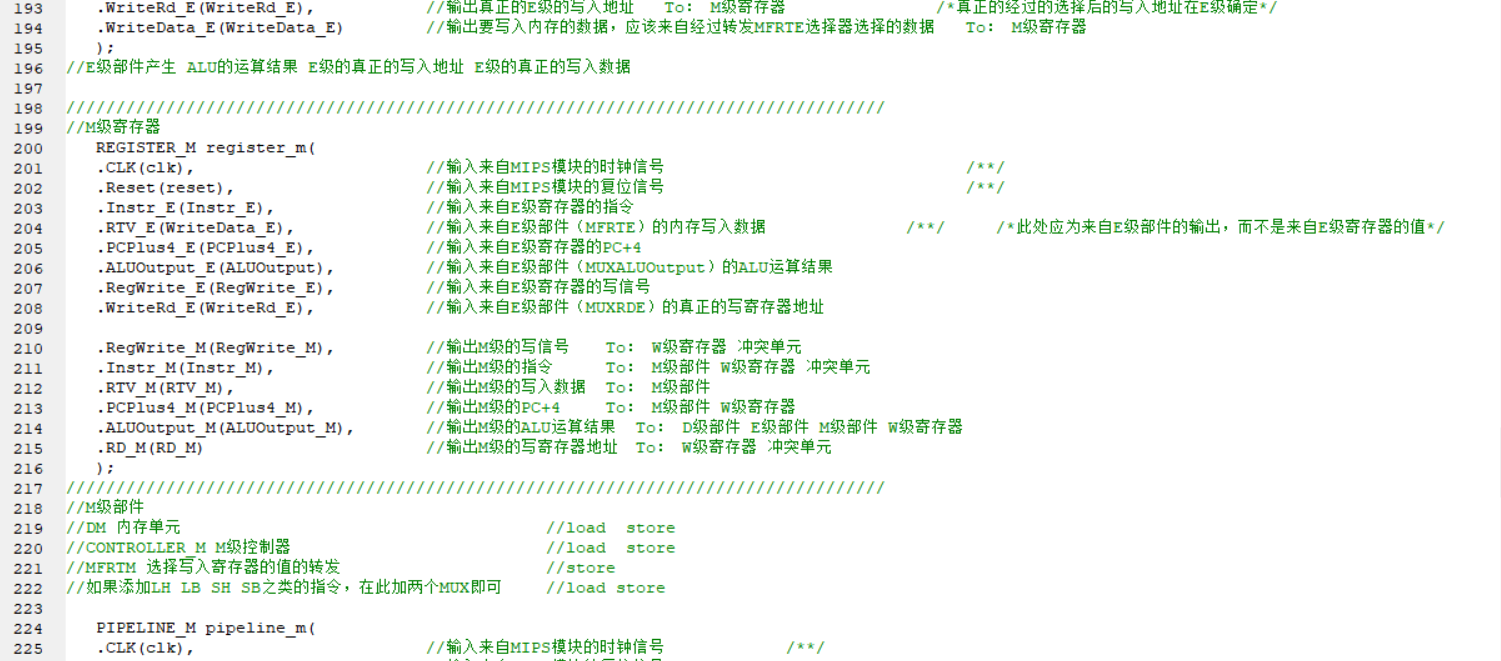
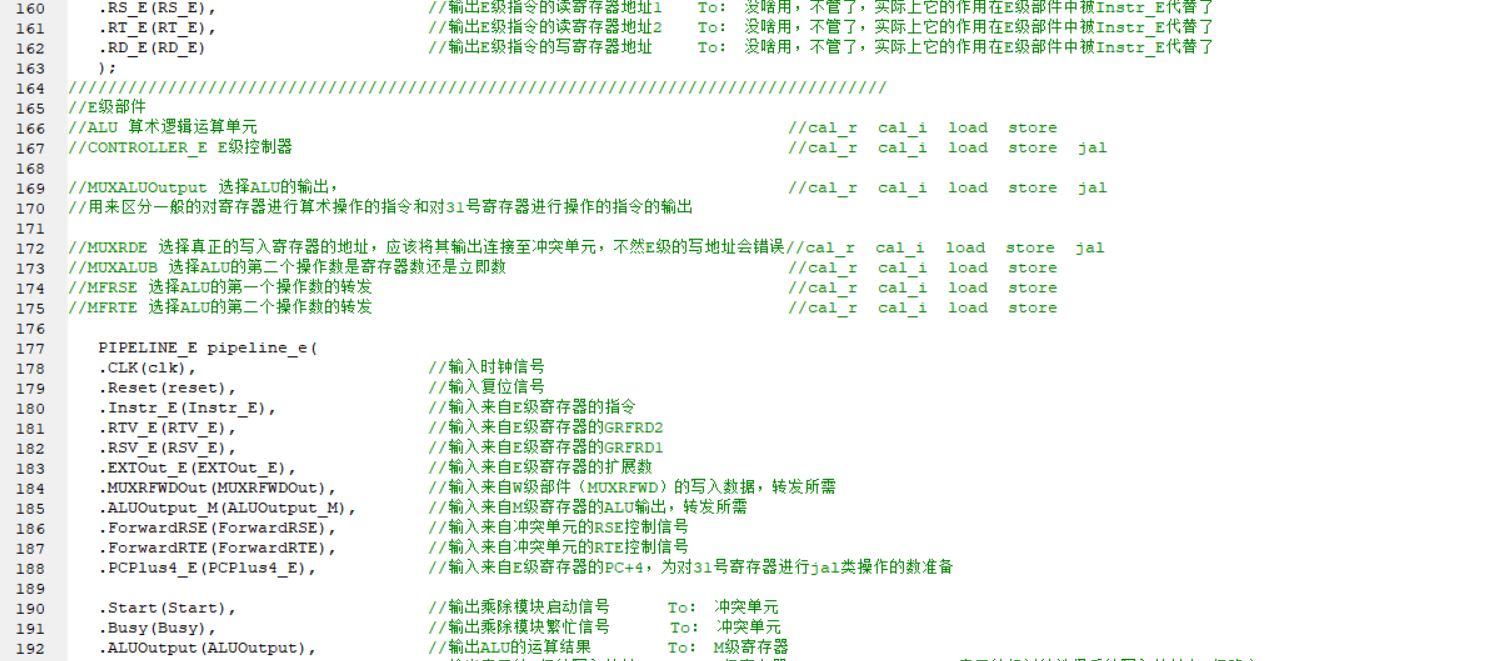
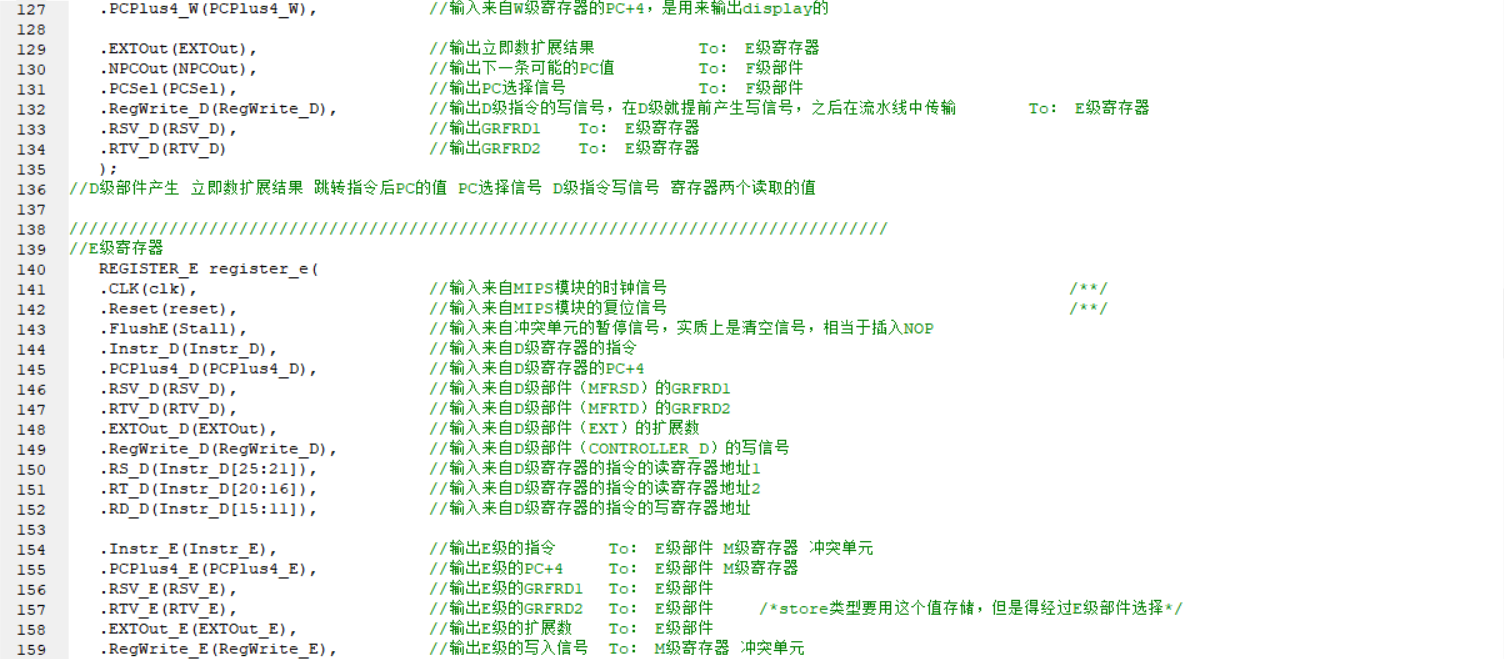
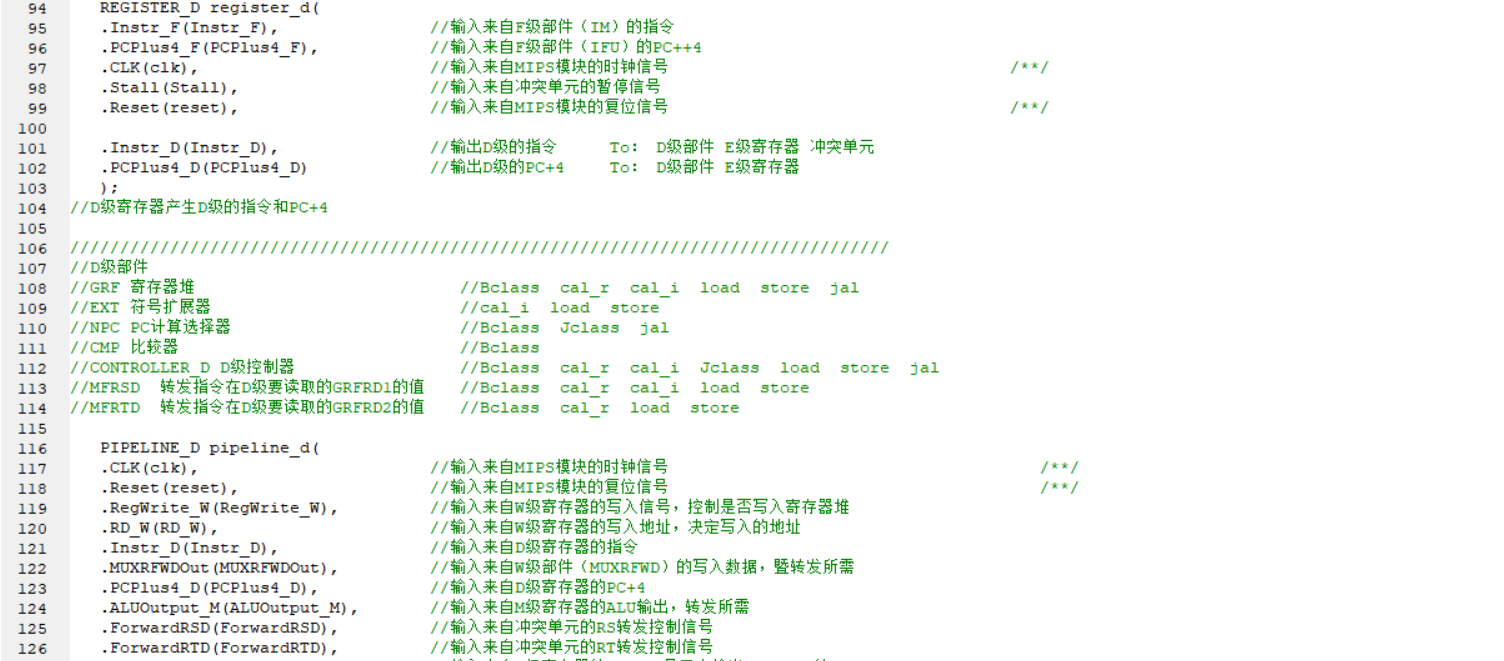
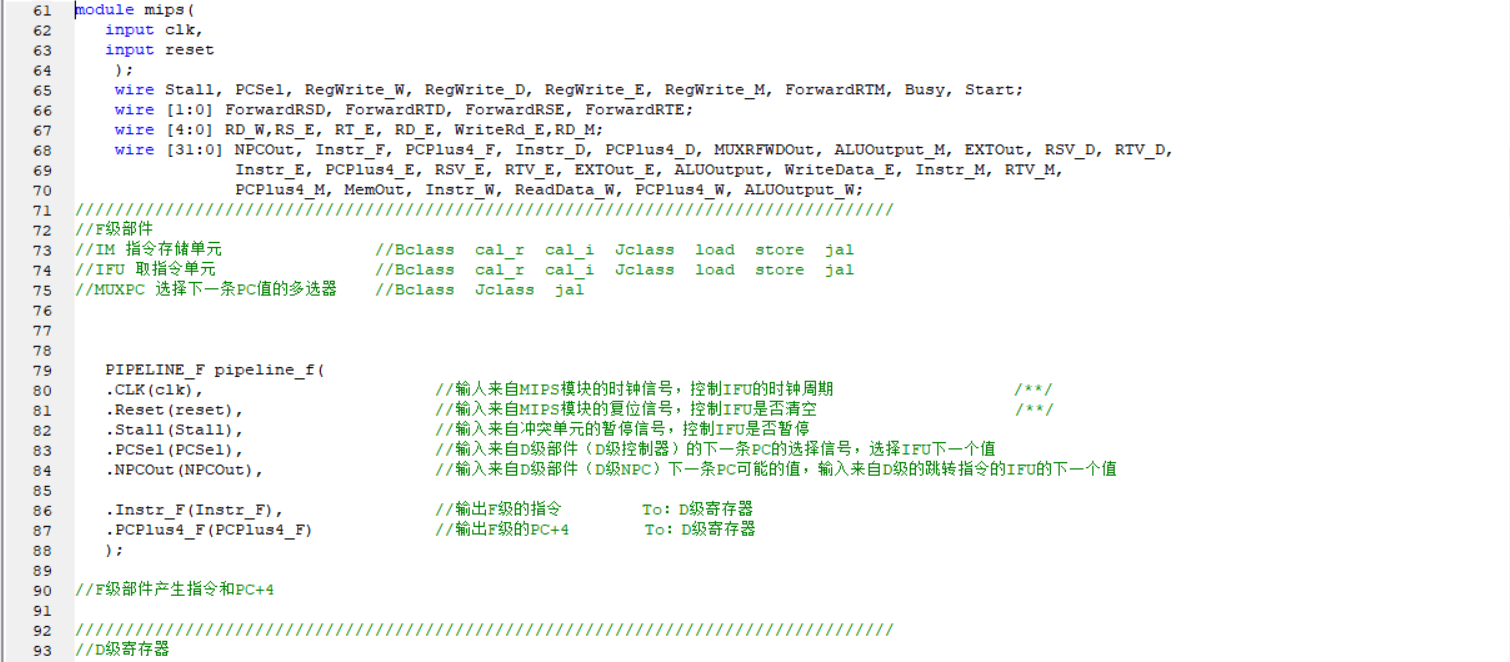
**Project6 Verilog完成流水线CPU开发**

1. **顶层设计**







1. **模块设计**
2. **PIPELINE\_F**

**（1）端口定义**

|  |  |  |
| --- | --- | --- |
| 信号 | 方向 | 描述 |
| CLK | I | 时钟信号 |
| Reset | I | 复位信号 |
| Stall | I | 输入来自冲突单元的暂停信号 |
| PCSel | I | 来自D级部件的PC选择信号 |
| NPCOut | I | 来自D级部件的NPC输出 |
| Instr\_F | O | 输出F级的指令 |
| PCPlus4\_F | O | 输出F级的PC+4 |

1. **功能定义**

|  |  |  |
| --- | --- | --- |
| 序号 | 内部部件 | 功能描述 |
| 1 | IM | 存储和输出当前指令 |
| 2 | IFU | 存储和输出当前指令地址PC |
| 3 | MUXPC | 选择下一条指令地址 |

1. **内部部件**
2. **IM**

**①端口定义**

|  |  |  |
| --- | --- | --- |
| 信号 | 方向 | 描述 |
| InstrAddr | I | 输入指令地址 |
| Instr\_F | O | 输出当前指令 |

**②功能定义**

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 取指令 | 根据InstrAddr指定的地址从IM中取出指令 |

1. **IFU**

**①端口定义**

|  |  |  |
| --- | --- | --- |
| 信号 | 方向 | 描述 |
| CLK | I | 时钟信号 |
| Reset | I | 复位信号 |
| Stall | I | 来自冲突单元的暂停信号 |
| PCIn | I | 来自MUXPC的下一个PC值 |
| PCOut | O | 输出PC值 |
| PCPlus4\_F | O | 输出当前PC+4 |

**②功能定义**

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 复位 | 当Reset信号有效时，PC被置为0x00003000 |

1. **MUX\_PC（MUX\_2\_1\_32）**

**①端口定义**

|  |  |  |
| --- | --- | --- |
| 信号 | 方向 | 描述 |
| PCPlus4\_F | I | 输入来自IFU的PC+4 |
| NPCOut | I | 输入来自D级部件的B或J类指令对应的PC |
| PCSel | I | 输入来自D级部件的PC选择信号 |
| PCIn | O | 输出下一条要更新的PC值 |

**②功能定义**

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 选择下一条指令地址 | 当PCSel有效时选择NPCOut,即跳转指令的地址 |

1. **Regsiter\_D**
2. **模块接口**

|  |  |  |
| --- | --- | --- |
| 信号 | 方向 | 描述 |
| Instr\_F | I | 来自F级部件的指令 |
| PCPlus4\_F | I | 来自F级部件的PC+4 |
| CLK | I | 时钟信号 |
| Stall | I | 来自冲突单元的暂停信号 |
| Reset | I | 复位信号 |
| Instr\_D | O | 输出D级寄存器的指令 |
| PCPlus4\_D | O | 输出D级寄存器的PC+4 |

1. **PIPELINE\_D**
2. **端口定义**

|  |  |  |
| --- | --- | --- |
| 信号 | 方向 | 描述 |
| CLK | I | 时钟信号 |
| Reset | I | 复位信号 |
| RegWrite\_W | I | 输入来自W级寄存器的写信号 |
| RD\_W | I | 输入来自W级寄存器的写地址 |
| Instr\_D | I | 输入来自D级寄存器的指令 |
| MUXRFWDOut | I | 输入来自W级部件的写数据，也是转发所需要的数据 |
| PCPlus4\_D | I | 输入来自D级寄存器的PC+4 |
| ALUOutput\_M | I | 输入来自M级寄存器的ALU运算结果，也是转发所需要的数据 |
| ForwardRSD | I | 输入来自冲突单元的RSD转发控制信号 |
| ForwardRTD | I | 输入来自冲突单元的RTD转发控制信号 |
| PCPlus4\_W | I | 输入来自W级寄存器的PC+4 |
| EXTOut | O | 输出D级部件产生的立即数扩展结果 |
| NPCOut | O | 输出D级部件的跳转指令产生的PC值 |
| PCSel | O | 输出D级部件产生的PC选择信号 |
| RegWrite\_D | O | 输出D级部件的指令的写信号 |
| RSV\_D | O | 输出寄存器堆读数据1 |
| RTV\_D | O | 输出寄存器堆读数据2 |

1. **功能定义**

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 读数据 | 输出指令要读的rs、rt的寄存器对应的值 |
| 2 | 扩展立即数 | 扩展指令中的立即数作为E级的ALU操作数 |
| 3 | 计算跳转地址 | 计算跳转指令对应的跳转地址 |
| 4 | 控制跳转 | 判断当前指令是否为跳转指令 |
| 5 | 写数据 | 将指令要写的数据写入rd寄存器中 |

**（3）内部部件**

**1）GRF**

**①模块接口**

|  |  |  |
| --- | --- | --- |
| 信号 | 方向 | 描述 |
| GRF\_Rs | I | 读寄存器地址1 |
| GRF\_Rt | I | 读寄存器地址2 |
| GRF\_Rd | I | 写寄存器地址 |
| GRF\_WD | I | 写寄存器数据 |
| CLK | I | 时钟信号 |
| Reset | I | 复位信号 |
| RegWrite | I | 寄存器写信号 |
| PCPlus4\_W | I | 写指令的地址 |
| GRF\_RD1 | O | 读寄存器数据1 |
| GRF\_RD2 | O | 读寄存器数据2 |

**②功能定义**

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 读数据 | 读出GRF\_Rs和GRF\_Rt地址对应寄存器中的数据到GRF\_RD1、GRF\_RD2 |
| 2 | 写数据 | 当RegWrite信号有效且时钟上升沿到来时，将GRF\_WD写入GRF\_Rd对应的寄存器中 |

1. **EXT**

**①模块接口**

|  |  |  |
| --- | --- | --- |
| 信号 | 方向 | 描述 |
| Imm16 | I | 进行扩展的立即数 |
| EXTSel | I | 扩展选择信号 |
| EXTOut | O | 扩展结果 |

**②功能定义**

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 零扩展 | 对立即数进行高位补0扩展 |
| 2 | 符号扩展 | 对立即数进行符号扩展至32位 |
| 3 | 加载到高位 | 将立即数加载到高位，低位补0 |

1. **CMP**

**①模块接口**

|  |  |  |
| --- | --- | --- |
| 信号 | 方向 | 描述 |
| CMPD1 | I | 比较的第一个数 |
| CMPD2 | I | 比较的第二个数 |
| CMPOut | O | 比较的结果 |
| CMPZeroOut | O | 第一个数与0的比较结果 |

**②功能定义**

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 比较两个操作数 | CMPOut:  00：CMPD1 == CMPD2  01：CMPD1>CMPD2  10:CMPD1<CMPD2 |
| 2 | 比较第一个数与0 | CMPZeroOut[0]:  0:CMPD1 ≤ 0  1:CMPD1 > 0  CMPZeroOut[1]:  0:CMPD1 == 0  1:CMPD1 != 0 |

1. **NPC**

**①模块接口**

|  |  |  |
| --- | --- | --- |
| 信号 | 方向 | 描述 |
| GRF\_RD1(MFRSDOut) | I | 输入Ra中存储的地址 |
| PCPlus4\_D | I | 输入来自D级寄存器的PC+4 |
| Instr\_D | I | 输入D级寄存器的指令，实质上是传输指令中的偏移量 |
| NPCSel | I | 输入NPC选择信号 |
| NPCOut | O | 输出B J JR指令对应的PC |

**②功能定义**

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 输出跳转地址 | NPCSel00：B型指令地址  NPCSel01：J型指令地址  NPCSel10：JR指令地址 |

1. **MFRSD（MUX\_4\_1\_32）**

**①模块接口**

|  |  |  |
| --- | --- | --- |
| 信号 | 方向 | 描述 |
| GRF\_RD1 | I | 输入GRF的读数据1 |
| MUXRFWDOut | I | 输入来自W级部件的GRF要写入的数据 |
| ALUOutput\_M | I | 输入来自M级寄存器的ALU运算结果 |
| ForwardRSD | I | 输入来自冲突单元的RSD转发信号 |
| MFRSDOut | O | 输出选择后的指令要用的GRF操作数1 |

**②功能定义**

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 输出转发选择后的GRF读数据1 | ForwardRSD：  00 GRF\_RD1  01 MUXRFWDOut  10 ALUOutput\_M |

1. **MFRTD（MUX\_4\_1\_32）**

**①模块接口**

|  |  |  |
| --- | --- | --- |
| 信号 | 方向 | 描述 |
| GRF\_RD2 | I | 输入GRF的读数据2 |
| MUXRFWDOut | I | 输入来自W级部件的GRF要写入的数据 |
| ALUOutput\_M | I | 输入来自M级寄存器的ALU运算结果 |
| ForwardRTD | I | 输入来自冲突单元的RTD转发信号 |
| MFRTDOut | O | 输出选择后的指令要用的GRF操作数2 |

**②功能定义**

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 输出转发选择后的GRF读数据2 | ForwardRTD：  00 GRF\_RD2  01 MUXRFWDOut  10 ALUOutput\_M |

1. **Controller\_D**

**①模块接口**

|  |  |  |
| --- | --- | --- |
| 信号 | 方向 | 描述 |
| Instr\_D | I | 输入来自D级寄存器的指令 |
| CMPOut | I | 输入来自D级部件CMP的比较结果 |
| CMPZeroOut | I | 输入来自D级部件CMP的与零比较结果 |
| NPCSel | O | 输出给NPC的选择信号 |
| EXTSel | O | 输出给EXT的选择信号 |
| RegWrite\_D | O | 输出给E级寄存器的GRF写信号 |
| PCSel | O | 输出给F级部件MUXPC的选择信号 |

**②功能定义**

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 选择指令跳转类型 | 通过NPCSel选择指令跳转地址 |
| 2 | 选择是否跳转 | 通过PCSel选择是否跳转 |
| 3 | 选择立即数扩展类型 | 通过EXTSel选择扩展类型 |
| 4 | 判断指令是否进行写寄存器操作 | 通过RegWrite\_D进行判断 |

1. **Register\_E**
2. **模块接口**

|  |  |  |
| --- | --- | --- |
| 信号 | 方向 | 描述 |
| CLK | I | 输入时钟信号 |
| Reset | I | 输入复位信号 |
| FlushE | I | 来自冲突单元的清空信号，实质上是Stall |
| Instr\_D | I | 输入来自D级寄存器的指令 |
| PCPlus4\_D | I | 输入来自D级寄存器的PC+4 |
| RSV\_D | I | 输入来自D级部件的RS的数据 |
| RTV\_D | I | 输入来自D级部件的RT的数据 |
| EXTOut\_D | I | 输入来自D级部件的EXT结果 |
| RegWrite\_D | I | 输入来自D级部件的GRF写信号 |
| RS\_D | I | 输入来自D级寄存器的RS地址 |
| RT\_D | I | 输入来自D级寄存器的RT |
| RD\_D | I | 输入来自D级寄存器的RD |
| Instr\_E | O | 输出E级的指令 |
| PCPlus4E | O | 输出E级的PC+4 |
| RSV\_E | O | 输出E级的RS对应的GRF读数据1 |
| RTV\_E | O | 输出E级的RD对应的GRF读数据2 |
| EXTOut\_E | O | 输出E级的EXT结果 |
| RegWrite\_E | O | 输出E的GRF写信号 |
| RS\_E | O | 输出E级的RS地址 |
| RT\_E | O | 输出E的RT地址 |
| RD\_E | O | 输出E的RD地址 |

1. **PIPELINE\_E**
2. **模块接口**

|  |  |  |
| --- | --- | --- |
| 信号 | 方向 | 描述 |
| Instr\_E | I | 输入来自E级寄存器的指令 |
| RTV\_E | I | 输入来自E级寄存器的GRF读数据2 |
| RSV\_E | I | 输入来自E级寄存器的GRF读数据1 |
| EXTOut\_E | I | 输入来自E级寄存器的扩展数 |
| MUXRFWDOut | I | 输入来自W级部件的GRF写数据，是转发所需数据 |
| ALUOutput\_M | I | 输入来自M级寄存器的ALU运算结果，是转发所需数据 |
| ForwardRSE | I | 输入来自冲突单元的RSE转发选择信号 |
| ForwardRTE | I | 输入来自冲突单元的RTE转发选择信号 |
| PCPlus4\_E | I | 输入来自E级寄存器的PC+4，也是jal类指令要写入31寄存器的数 |
| ALUOutput | O | 输出ALU运算结果 |
| WriteRd\_E | O | 输出寄存器写地址 |
| WriteData\_E | O | 输出内存写数据 |
| Start | O | 输出乘除模块开始运行信号 |
| Busy | O | 输出乘除模块正在运行信号 |

1. **功能定义**

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | ALU运算 | 通过ALU进行指令要求的运算 |
| 2 | 选择ALU输出，ALU输出可作为load store类指令的地址，可作为cal运算的结果，可作为jal指令的写入值 | 如果是JAL类对31号寄存器进行地址写入的指令，置为PCPlus4\_E+4，否则如果是正常的ALU运算，就置为ALU运算结果 |
| 3 | 选择写入寄存器地址 | 如果是三寄存器操作指令，置为rd，如果是二寄存器操作指令，置为rt，如果是jal类指令，置为31 |
| 4 | 选择写入内存数据，仅仅针对store类指令 | store类指令要写入内存的数据可在E级部件通过转发获取 |

1. **内部部件**
2. **ALU**

**①模块接口**

|  |  |  |
| --- | --- | --- |
| 信号 | 方向 | 描述 |
| ALUOprand\_A | I | ALU操作数A |
| ALUOprand\_B | I | ALU操作数B |
| ALUSel | I | ALU选择信号 |
| ALUOutput | O | ALU运算结果 |

**②功能定义**

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 加运算 | ALUOutput=ALUOprand\_A+ALUOprand\_B |
| 2 | 减运算 | ALUOutput=ALUOprand\_A-ALUOprand\_B |
| 3 | 或运算 | ALUOutput=ALUOprand\_A|ALUOprand\_B |
| 4 | 逻辑左移运算 | ALUOutput=ALUOprand\_B << ALUOprand\_A[4:0] |
| 5 | 逻辑右移运算 | ALUOutput=ALUOprand\_B >> ALUOprand\_A[4:0] |
| 6 | 算术右移运算 | ALUOutput=$signed(ALUOprand\_B) >>> ALUOprand\_A[4:0] |
| 7 | 与运算 | ALUOutput=ALUOprand\_A & ALUOprand\_B |
| 8 | 小于则置位 | ALUOutput=$signed(ALUOprand\_A) < $signed(ALUOprand\_B) |
| 9 | 异或运算 | ALUOutput=ALUOprand\_A ^ ALUOprand\_B |
| 10 | 或非运算 | ALUOutput=~(ALUOprand\_A | ALUOprand\_B) |
| 11 | 无符号小于则置位 | ALUOutput=ALUOprand\_A < ALUOprand\_B |

1. **Controller\_E**

**①模块接口**

|  |  |  |
| --- | --- | --- |
| 信号 | 方向 | 描述 |
| Instr\_E | I | 输入来自E级寄存器的指令 |
| ALUSel | O | 输出ALU操作选择信号 |
| MUXALUBSel | O | 输出ALUB功能选择信号，选择ALU操作数B是寄存器数还是立即数 |
| RegDst | O | 输出当前指令的写寄存器选择信号 |
| ALUOutputSel | O | 输出ALU的结果选择信号，是jal型对应的PC+8还是正常的ALU运算结果，还是HI或LO寄存器的值 |
| Start | O | 输出乘除模块开始工作的信号 |
| MDSel | O | 输出乘除模块功能选择信号 |
| MUXALUASel | O | 输出ALUA功能选择信号，选择ALU操作数A是寄存器RS，还是立即数 |

**②功能定义**

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 选择ALU运算类型 | 通过ALUSel选择当前E级指令对应的ALU运算 |
| 2 | 选择ALU运算数 | 通过MUXALUBSel选择当前E级指令对应的ALU运算数是寄存器数还是立即数 |
| 3 | 选择写寄存器地址 | 通过RegDst选择当前E级指令写入寄存器的地址是rt 还是rd还是31 |
| 4 | 选择ALU输出 | 通过ALUOutputSel选择当前E级指令要用的ALU输出是PC+8（JAL）还是正常的运算结果（正常运算指令），还是HI或LO寄存器的值 |

1. **MUXALUB**

**①模块接口**

|  |  |  |
| --- | --- | --- |
| 信号 | 方向 | 描述 |
| MFRTEOut | I | 输入ALUB转发多选器的选择结果，为寄存器操作数 |
| EXTOut\_E | I | 输入来自E级寄存器的扩展数 |
| MUXALUBSel | I | 输入来自E级控制器的MUXALUB的选择信号 |
| MUXALUBOut | O | 输出ALUB |

**②功能定义**

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 选择ALU的第二个操作数 | 当MUXALUBSel为1时选择立即数，为0时选择寄存器操作数 |

1. **MFRSE**

**①模块接口**

|  |  |  |
| --- | --- | --- |
| 信号 | 方向 | 描述 |
| RSV\_E | I | 来自E级寄存器的的GEF读数据1 |
| MUXRFWDOut | I | 来自W级部件的GRF写回数据 |
| ALUOutput\_M | I | 来自M级寄存器的ALU的输出 |
| ForwardRSE | I | 来自E级控制器的MFRSE的选择信号 |
| MFRSEOut | O | 输出MFRSE选择后的数据 |

**②功能定义**

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 选择ALU的第一个操作数 | ForwardRSE:  00 RSV\_E  01 MUXRFWDOut  10 ALUOutput\_M |

1. **MFRTE**

**①模块接口**

|  |  |  |
| --- | --- | --- |
| 信号 | 方向 | 描述 |
| RTV\_E | I | 来自E级寄存器的GRF的读数据2 |
| MUXRFWDOut | I | 来自W级寄存器的GRF写回数据 |
| ALUOutput\_M | I | 来自M级寄存器的ALU的输出 |
| ForwardRTE | I | 来自E级控制器的MFRTE的选择信号 |
| MFRTEOut | O | 输出MFRTE选择后的数据 |

**②功能定义**

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 选择ALU的第二个操作数中的寄存器数 | ForwardRTE:  00 RTV\_E  01 MUXRFWDOut  10 ALUOutput\_M |

**6）MUXRDE**

**①模块接口**

|  |  |  |
| --- | --- | --- |
| 信号 | 方向 | 描述 |
| RT\_E | I | 来自E级寄存器的GRF写地址1 |
| RD\_E | I | 来自E级寄存器的GRF写地址2 |
| 5’b11111 | I | 31号寄存器 |
| RegDst | I | 来自E级控制器的写寄存器选择信号 |
| WriteRd\_E | O | 输出E级当前指令要写的寄存器地址 |

**②功能定义**

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 选择当前指令要写入的寄存器地址 | RegDst:  00 RT\_E  01 RD\_E  10 31 |

**7）MUXALUOutput**

**①模块接口**

|  |  |  |
| --- | --- | --- |
| 信号 | 方向 | 描述 |
| ALUOutput\_ALU | I | 输入来自ALU的运算结果 |
| PCPlus4\_E+4 | I | 输入来自E级寄存器的PCPlus4\_E+4，作为JAL指令对应要输入的值 |
| HI | I | 输入来自E级乘除模块的HI寄存器的值 |
| LO | I | 输入来自E级乘除模块的LO寄存器的值 |
| ALUOutputSel | I | 输入来自E级寄存器E级的ALU输出选择信号 |
| ALUOutput | O | 输出E级的ALU运算结果，其可能是正常指令的运算结果，也可能是对应JAL指令的PC+8 |

**②功能定义**

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 选择E级部件的ALU运算输出是正常ALU运算指令的运算结果还是JAL指令对应的PC+8 | ALUOutputSel  0 ALUOutput\_ALU  1 PCPlus4\_E+4 |

1. **MUXALUA**

**①模块接口**

|  |  |  |
| --- | --- | --- |
| 信号 | 方向 | 描述 |
| MFRSEOut | I | 输入ALUA转发多选器的选择结果，为寄存器操作数 |
| EXTOut\_E | I | 输入来自E级寄存器的扩展数 |
| MUXALUASel | I | 输入来自E级控制器的MUXALUA的选择信号 |
| MUXALUAOut | O | 输出ALUA |

**②功能定义**

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 选择ALU的第一个操作数 | 当MUXALUASel为1时选择来自EXT的立即数，为0时选择来自RS的寄存器数，为满足移位指令特设 |

1. **MULTDIVPART**

**①模块接口**

|  |  |  |
| --- | --- | --- |
| 信号 | 方向 | 描述 |
| CLK | I | 输入时钟信号 |
| Reset | I | 输入复位信号 |
| Start | I | 输入乘除模块开始工作信号 |
| ALUOprand\_A | I | 输入乘除模块操作数1 |
| ALUOprand\_B | I | 输入乘除模块操作数2 |
| MDSel | I | 输入乘除模块控制信号 |
| HI | O | 输出HI寄存器的值 |
| LO | O | 输出LO寄存器的值 |
| Busy | O | 输出乘除模块正在工作的信号 |

**②功能定义**

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 进行乘除相关运算 | MDSel：  1： MULT  2: MULTU  3: DIV  4: DIVU  5: MTHI  6: MTLO |
| 2 | 输出正在工作标志 | Busy = Counter > 0; |

1. **Register\_M**

**（1）模块接口**

|  |  |  |
| --- | --- | --- |
| 信号 | 方向 | 描述 |
| CLK | I | 输入时钟信号 |
| Reset | I | 输入复位信号 |
| Instr\_E | I | 输入来自E级寄存器的指令 |
| RTV\_E | I | 输入来自E级部件的要写入内存的数据 |
| PCPlus4\_E | I | 输入来自E级寄存器的的PC+4 |
| ALUOutput\_E | I | 输入来自E级部件的ALU运算结果 |
| RegWrite\_E | I | 输入来自E级寄存器的GRF写信号 |
| WriteRd\_E | I | 输入来自E级部件的当前指令对应的写寄存器地址 |
| RegWrite\_M | O | 输出M级寄存器的GRF写信号 |
| Instr\_M | O | 输出M级寄存器的指令 |
| RTV\_M | O | 输出M级寄存器的要写入内存的数据 |
| PCPlus4\_M | O | 输出M级寄存器的PC+4 |
| ALUOutput\_M | O | 输出M级寄存器的ALU运算结果 |
| RD\_M | O | 输出M级寄存器的写寄存器地址 |

1. **PIPELINE\_M**
2. **模块接口**

|  |  |  |
| --- | --- | --- |
| 信号 | 方向 | 描述 |
| CLK | I | 输入时钟信号 |
| Reset | I | 输入复位信号 |
| ALUOutput\_M | I | 输入M级寄存器的ALU运算结果，作为存取内存的地址 |
| RTV\_M | I | 输入来自M级寄存器的写入内存数据 |
| Instr\_M | I | 输入来自M级寄存器的指令 |
| ForwardRTM | I | 输入来自冲突单元的RTM转发控制信号 |
| PCPlus4\_M | I | 输入来自M级寄存器的PC+4，用来display |
| MUXRFWDOut | I | 输入来自W级部件的GRF写入数据，是转发所需数据 |
| MemOut | O | 输出M级部件读出的内存数据 |

1. **功能定义**

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 读内存 | 根据输入的ALUOutput\_M作为内存地址读出内存中的数据 |
| 2 | 写内存 | 当写内存信号有效且在时钟上升沿将数据写入对应的内存地址 |
| 3 | 复位 | 当Reset信号有效且时钟上升沿到来时将所有内存数据置零 |

1. **内部部件**
2. **DM**

**①模块接口**

|  |  |  |
| --- | --- | --- |
| 信号 | 方向 | 描述 |
| CLK | I | 输入时钟信号 |
| Reset | I | 输入复位信号 |
| MemWrite | I | 输入来自M级控制器的内存写使能信号 |
| MemAddr | I | 输入来自M级寄存器额内存写地址，即ALU运算结果 |
| MemWD | I | 输入来自MFRTM的内存写数据 |
| PCPlus4\_M | I | 输入来自M级寄存器的PC+4，用来display |
| MemOut | O | 输出内存读数据 |

**②功能定义**

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 读内存 | 根据输入的ALUOutput\_M作为内存地址读出内存中的数据 |
| 2 | 写内存 | 当写内存信号有效且在时钟上升沿将数据写入对应的内存地址 |
| 3 | 复位 | 当Reset信号有效且时钟上升沿到来时将所有内存数据置零 |

1. **Controller\_M**

**①模块接口**

|  |  |  |
| --- | --- | --- |
| 信号 | 方向 | 描述 |
| Instr\_M | I | 来自M级的指令 |
| MemWrite | O | 输出内存写使能信号 |
| StoreSel | O | 输出写入内存的数据选择信号 |

**②功能定义**

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 判断是否写内存 | 根据当前的M级指令输出MemWrite信号控制是否写内存 |
| 2 | 选择写入内存的数据 | 根据当前M级指令输出StoreSel信号选择写入内存的数据 |

1. **MFRTM**

**①模块接口**

|  |  |  |
| --- | --- | --- |
| 信号 | 方向 | 描述 |
| RTV\_M | I | 输入来自M级寄存器的GRF读数据2 |
| MUXRFWDOut | I | 输入来自W级部件的GRF写回数据，是转发所需数据 |
| ForwardRTM | I | 输入来自冲突单元的MFRTM选择信号 |
| MFRTMOut | O | 输出MFRTM的转发选择结果，作为写入内存的数据 |

**②功能定义**

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 选择写入内存的数据 | ForwardRTM:  0 RTV\_M  1 MUXRFWDOut |

1. **StoreBE**

**①模块接口**

|  |  |  |
| --- | --- | --- |
| 信号 | 方向 | 描述 |
| MFRTMOut | I | 输入经过转发选择后的要存储的数据 |
| StoreSel | I | 输入来自M级控制器的数据存储方式选择信号 |
| BSel | I | 输入存储数据的字段选择信号 |
| MemRD | I | 输入存储地址当前对应的数据 |
| MemWD | O | 输出要存入存储地址的数据 |

**②功能定义**

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 选择写入内存的数据的字段 | BSel:  00: [7:0]  01: [15:8]  10: [23:16]  11: [31:24] |
| 2 | 选择写入内存的数据的方式 | StoreSel:  00: sw  01: sb  10: sh |

**8、Register\_W**

|  |  |  |
| --- | --- | --- |
| 信号 | 方向 | 描述 |
| CLK | I | 输入时钟信号 |
| Reset | I | 输入复位信号 |
| Instr\_M | I | 输入来自M级寄存器的指令 |
| PCPlus4\_M | I | 输入来自M级寄存器的PC+4 |
| ALUOutput\_M | I | 来自M级寄存器的ALU输出 |
| ReadData\_M | I | 输入来自M级部件的内存读数据 |
| RD\_M | I | 输入来自M级寄存器的GRF写地址 |
| RegWrite\_M | I | 输入来自M级寄存器的GRF写信号 |
| Instr\_W | O | 输出W级寄存器的指令 |
| PCPlus4\_W | O | 输出W级寄存器的PC+4 |
| ALUOutput\_W | O | 输出W级寄存器的ALU输出 |
| ReadData\_W | O | 输出W级寄存器的内存读数据 |
| RD\_W | O | 输出W级寄存器的GRF写地址地址 |
| RegWrite\_W | O | 输出W级寄存器的GRF写信号 |

**9、PIPELINE\_W**

1. **模块接口**

|  |  |  |
| --- | --- | --- |
| 信号 | 方向 | 描述 |
| Instr\_W | I | 输入来自W级寄存器的指令 |
| ReadData\_W | I | 输入来自W级寄存器的内存读数据 |
| ALUOutput\_W | I | 输入来自W级寄存器的ALU运算结果 |
| PCPlus4\_W | I | 输入来自W级寄存器的PC+4 |
| MFRDWDOut | O | 输出经过选择后的GRF要写入的数据 |

1. **功能定义**

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 选择写入GRF的数据 | MemtoReg\_W:  0 ALUOutput\_W  1 ReadData\_W |

1. **内部部件**
2. **Controller\_W**

**①模块接口**

|  |  |  |
| --- | --- | --- |
| 信号 | 方向 | 描述 |
| Instr\_W | I | 来自W级的指令 |
| MemtoReg\_W | O | MUXRFWD的选择信号 |
| LoadSel | O | 输出内存取出的数据的选择信号 |

**②功能定义**

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 输出选择写入GRF的数据的控制信号 | Memtoreg:  load指令1，其他指令0 |
| 2 | 输出选择写入GRF的内存读数据的方式的选择信号 | LoadSel:  000 lw  001 lb  010 lub  011 lh  100 lhu |

1. **MUXRFWD**

**①模块接口**

|  |  |  |
| --- | --- | --- |
| 信号 | 方向 | 描述 |
| ReadData\_W | I | 来自W级的内存数据 |
| ALUOutput\_W | I | 来自W级的ALU输出 |
| MemtoReg\_W | I | MUXRFWD选择信号 |
| MUXRFWDOut | O | 输出寄存器写入数据 |

**②功能定义**

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 选择写入GRF的数据 | MemtoReg\_W:  0 ALUOutput\_W  1 ReadData\_W |

1. **LoadBE**

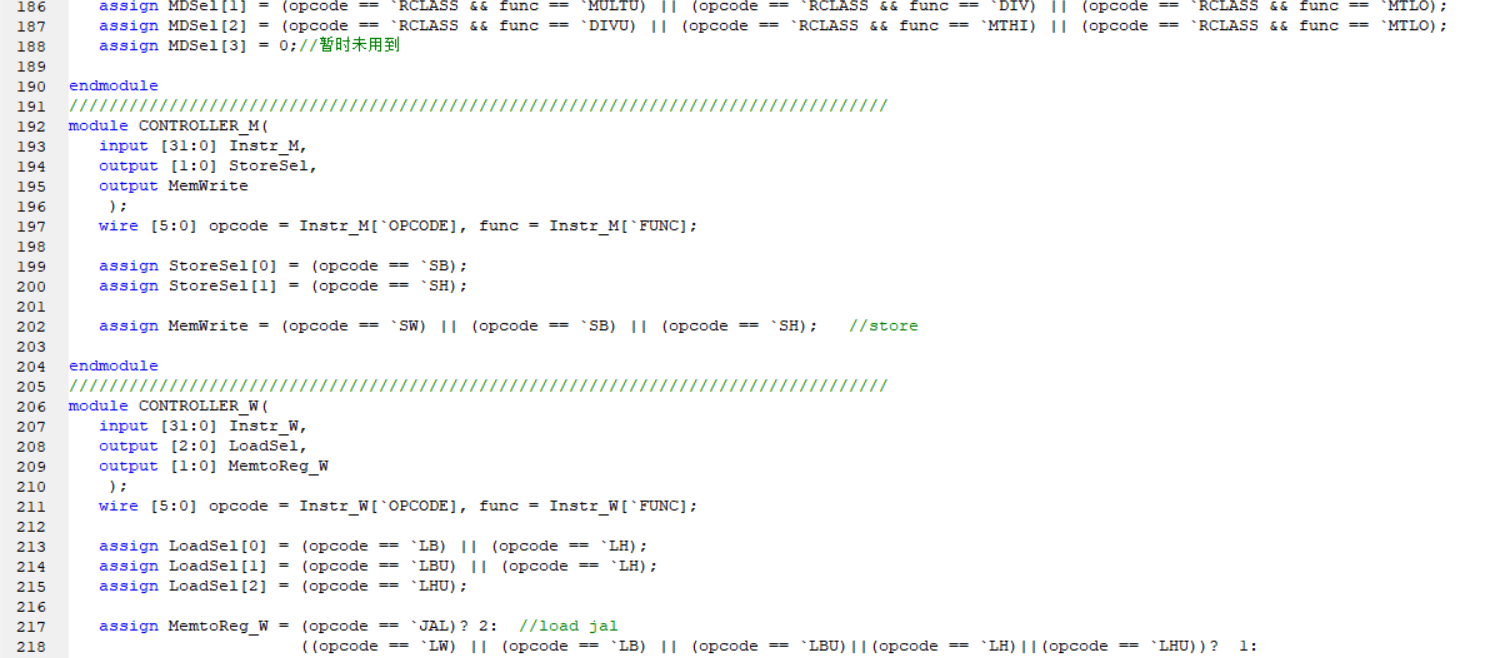
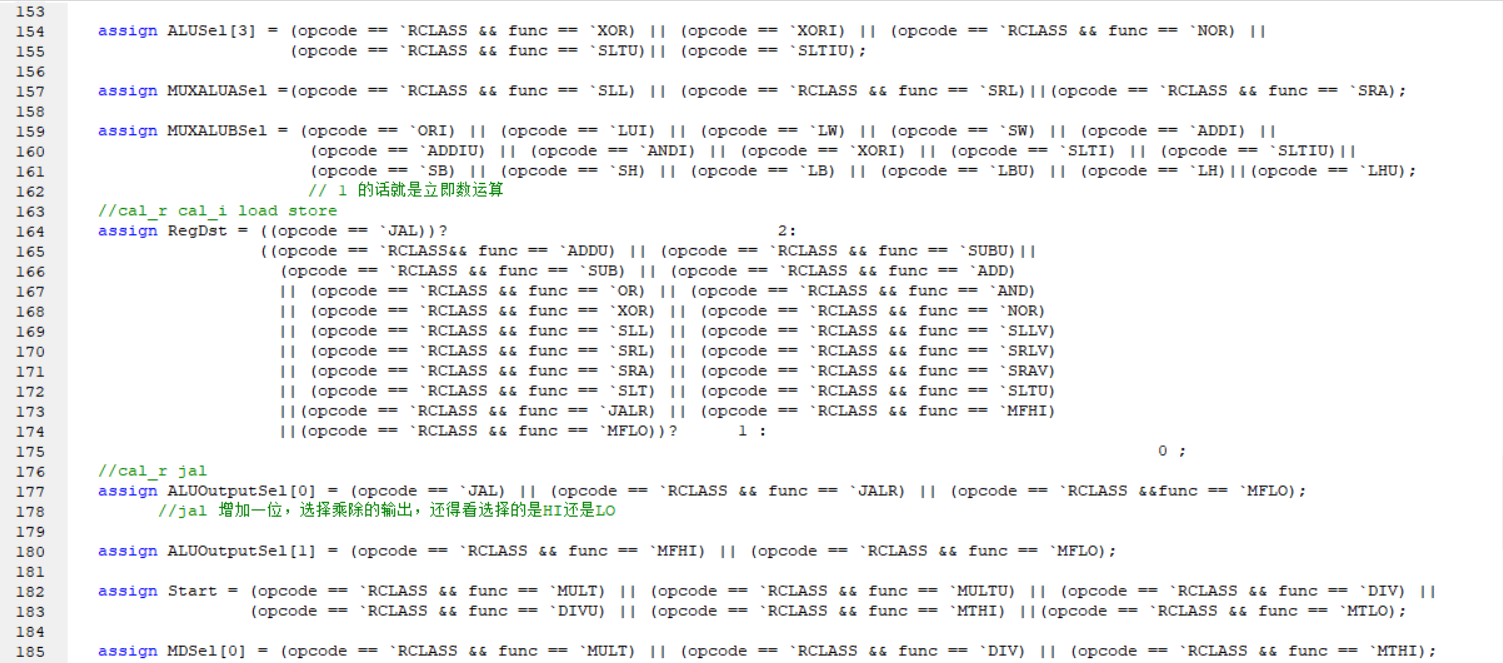
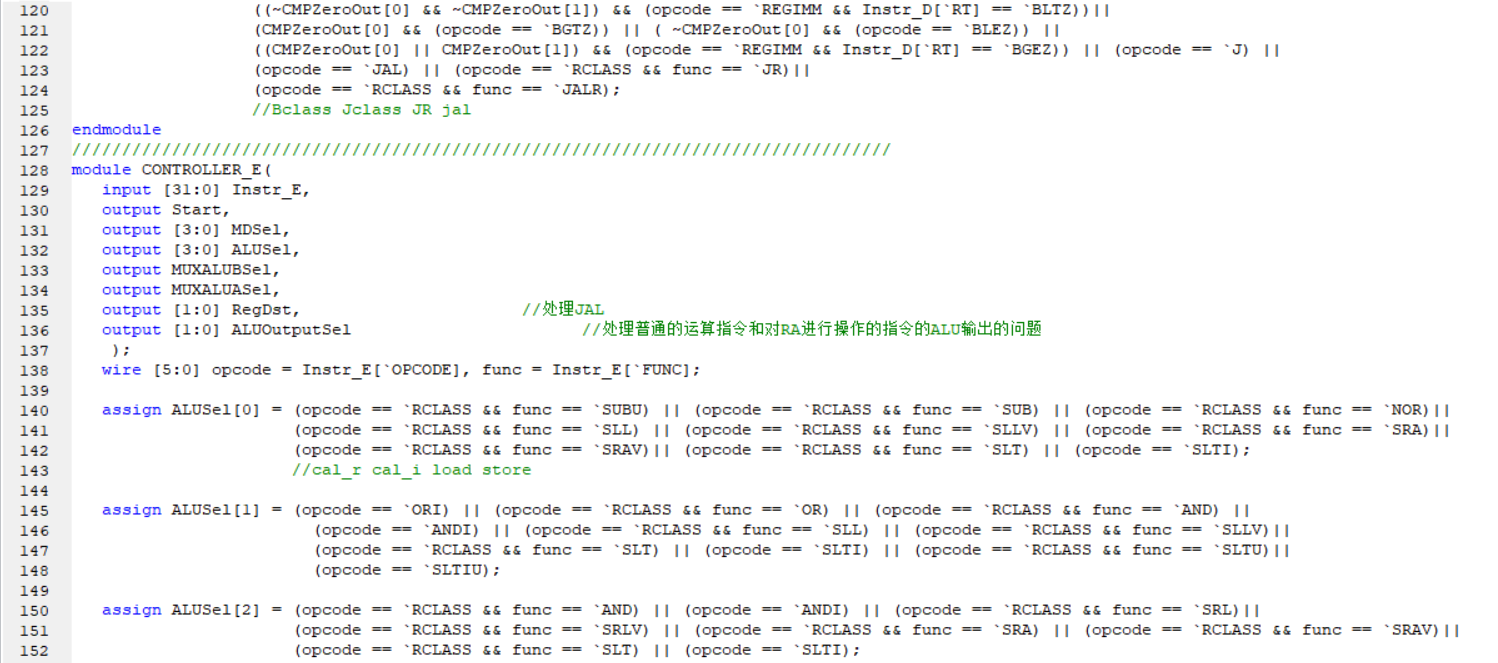
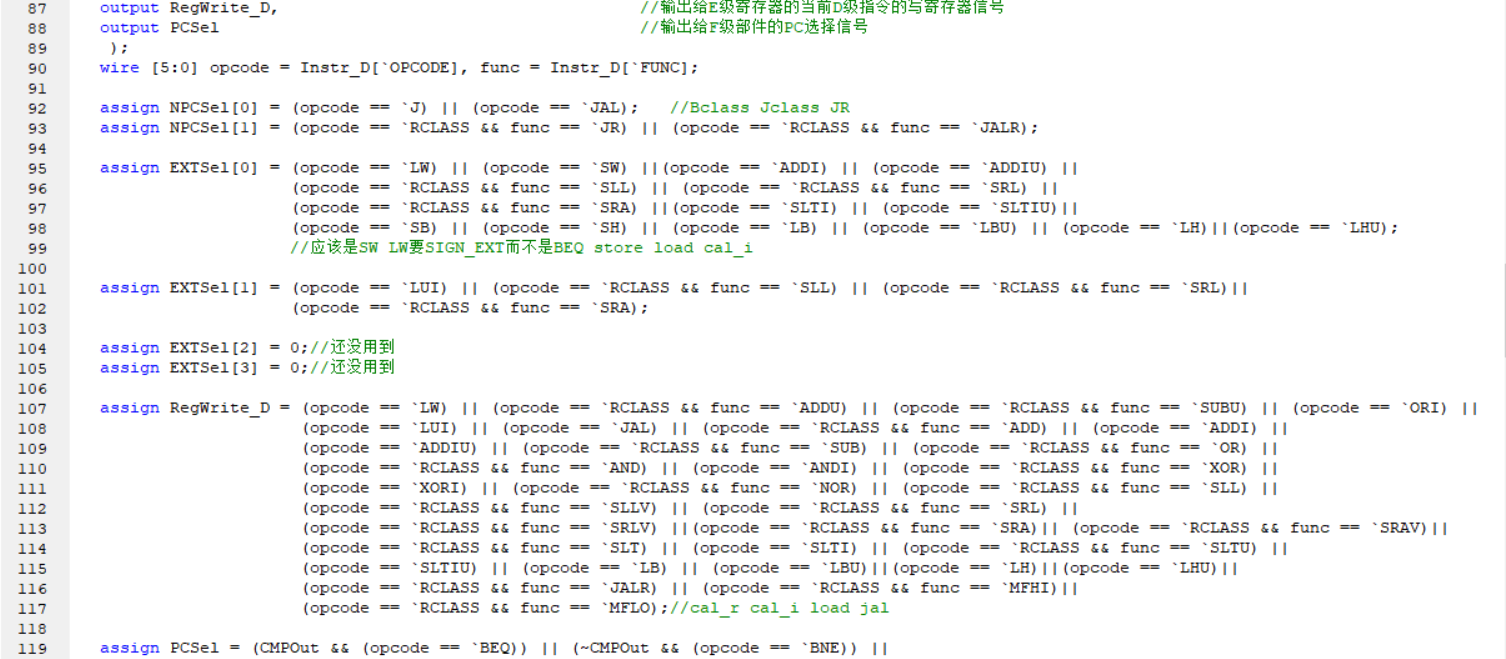
**①模块接口**

|  |  |  |
| --- | --- | --- |
| 信号 | 方向 | 描述 |
| ReadData\_W | I | 输入来自W级寄存器的内存数据 |
| BSel | I | 输入要读取的内存读数据的字段选择信号 |
| LoadSel | I | 输入要读取的内存读数据的数据形式的选择信号 |
| ReadData | O | 输出指令要读取的数据 |

**②功能定义**

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 选择指令要读取的内存的数据的字段 | BSel:  00: [7:0]  01: [15:8]  10: [23:16]  11: [31:24] |
| 2 | 选择指令要读取的内存的数据的形式 | LoadSel:  000 lw  001 lb  010 lub  011 lh  100 lhu |

**10、CONTROLLER**



|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **op**  **code** | **func** | **NPC**  **Sel** | **EXT**  **Sel** | **PC**  **Sel** | **Reg**  **Write** | **Reg**  **Dst** | **ALU**  **Out**  **put**  **Sel** | **ALU**  **Sel** | **MUXALUB**  **Sel** | **Mem**  **Write** | **Mem**  **toReg** |
| **addu** | **000000** | **100001** | **X** | **X** | **0** | **1** | **1** | **0** | **0** | **0** | **0** | **0** |
| **add** | **000000** | **100000** | **X** | **X** | **0** | **1** | **1** | **0** | **0** | **0** | **0** | **0** |
| **addi** | **001000** | **N/A** | **X** | **1** | **0** | **1** | **0** | **0** | **0** | **1** | **0** | **0** |
| **addiu** | **001001** | **N/A** | **X** | **1** | **0** | **1** | **0** | **0** | **0** | **1** | **0** | **0** |
| **subu** | **000000** | **100011** | **X** | **X** | **0** | **1** | **1** | **0** | **1** | **0** | **0** | **0** |
| **sub** | **000000** | **100010** | **X** | **X** | **0** | **1** | **1** | **0** | **1** | **0** | **0** | **0** |
| **ori** | **001101** | **N/A** | **X** | **0** | **0** | **1** | **0** | **0** | **2** | **1** | **0** | **0** |
| **or** | **000000** | **100101** | **X** | **X** | **0** | **1** | **1** | **0** | **2** | **0** | **0** | **0** |
| **sll** | **000000** | **000000** | **X** | **3** | **0** | **1** | **1** | **0** | **3** | **0** | **0** | **0** |
|  | **op**  **code** | **func** | **NPC**  **Sel** | **EXT**  **Sel** | **PC**  **Sel** | **Reg**  **Write** | **Reg**  **Dst** | **ALU**  **Out**  **put**  **Sel** | **ALU**  **Sel** | **MUXALUB**  **Sel** | **Mem**  **Write** | **Mem**  **toReg** |
| **sllv** | **000000** | **000100** | **X** | **X** | **0** | **1** | **1** | **0** | **3** | **0** | **0** | **0** |
| **srl** | **000000** | **000010** | **X** | **3** | **0** | **1** | **1** | **0** | **4** | **0** | **0** | **0** |
| **srlv** | **000000** | **000110** | **X** | **X** | **0** | **1** | **1** | **0** | **4** | **0** | **0** | **0** |
| **sra** | **000000** | **000011** | **X** | **3** | **0** | **1** | **1** | **0** | **5** | **0** | **0** | **0** |
| **srav** | **000000** | **000111** | **X** | **X** | **0** | **1** | **1** | **0** | **5** | **0** | **0** | **0** |
| **and** | **000000** | **100100** | **X** | **X** | **0** | **1** | **1** | **0** | **6** | **0** | **0** | **0** |
| **andi** | **001100** | **N/A** | **X** | **0** | **0** | **1** | **0** | **0** | **6** | **1** | **0** | **0** |
| **slt** | **000000** | **101010** | **X** | **X** | **0** | **1** | **1** | **0** | **7** | **0** | **0** | **0** |
| **slti** | **001010** | **N/A** | **X** | **1** | **0** | **1** | **0** | **0** | **7** | **1** | **0** | **0** |
| **sltu** | **000000** | **101011** | **X** | **X** | **0** | **1** | **1** | **0** | **10** | **0** | **0** | **0** |
| **sltiu** | **001011** | **N/A** | **X** | **1** | **0** | **1** | **0** | **0** | **10** | **1** | **0** | **0** |
| **xor** | **000000** | **100110** | **X** | **X** | **0** | **1** | **1** | **0** | **8** | **0** | **0** | **0** |
| **xori** | **001110** | **N/A** | **X** | **0** | **0** | **1** | **0** | **0** | **8** | **1** | **0** | **0** |
| **nor** | **000000** | **100111** | **X** | **X** | **0** | **1** | **1** | **0** | **9** | **0** | **0** | **0** |
| **lui** | **001111** | **N/A** | **X** | **2** | **0** | **1** | **0** | **0** | **0** | **1** | **0** | **0** |
| **lw** | **100011** | **N/A** | **X** | **1** | **0** | **1** | **0** | **0** | **0** | **1** | **0** | **1** |
| **lb** | **100000** | **N/A** | **X** | **1** | **0** | **1** | **0** | **0** | **0** | **1** | **0** | **1** |
| **lbu** | **100100** | **N/A** | **X** | **1** | **0** | **1** | **0** | **0** | **0** | **1** | **0** | **1** |
| **lh** | **100001** | **N/A** | **X** | **1** | **0** | **1** | **0** | **0** | **0** | **1** | **0** | **1** |
|  | **op**  **code** | **func** | **NPC**  **Sel** | **EXT**  **Sel** | **PC**  **Sel** | **Reg**  **Write** | **Reg**  **Dst** | **ALU**  **Out**  **put**  **Sel** | **ALU**  **Sel** | **MUXALUB**  **Sel** | **Mem**  **Write** | **Mem**  **toReg** |
| **lhu** | **100101** | **N/A** | **X** | **1** | **0** | **1** | **0** | **0** | **0** | **1** | **0** | **1** |
| **sw** | **101011** | **N/A** | **X** | **1** | **0** | **0** | **X** | **0** | **0** | **1** | **1** | **0** |
| **sb** | **101000** | **N/A** | **X** | **1** | **0** | **0** | **X** | **0** | **0** | **1** | **1** | **0** |
| **sh** | **101001** | **N/A** | **X** | **1** | **0** | **0** | **X** | **0** | **0** | **1** | **1** | **0** |
| **beq** | **000100** | **N/A** | **0** | **X** | **1** | **0** | **X** | **X** | **0** | **0** | **0** | **0** |
| **bne** | **000101** | **N/A** | **0** | **X** | **1** | **0** | **X** | **X** | **0** | **0** | **0** | **0** |
| **blez** | **000110** | **N/A** | **0** | **X** | **1** | **0** | **X** | **X** | **0** | **0** | **0** | **0** |
| **bgtz** | **000111** | **N/A** | **0** | **X** | **1** | **0** | **X** | **X** | **0** | **0** | **0** | **0** |
| **bltz** | **000001** | **00000** | **0** | **X** | **1** | **0** | **X** | **X** | **0** | **0** | **0** | **0** |
| **bgez** | **000001** | **00001** | **0** | **X** | **1** | **0** | **X** | **X** | **0** | **0** | **0** | **0** |
| **j** | **000010** | **N/A** | **1** | **X** | **1** | **0** | **X** | **X** | **0** | **X** | **0** | **0** |
| **jr** | **000000** | **001000** | **2** | **X** | **1** | **0** | **X** | **X** | **0** | **X** | **0** | **0** |
| **jal** | **000011** | **N/A** | **1** | **X** | **1** | **1** | **2** | **1** | **0** | **X** | **0** | **0** |
| **jalr** | **000000** | **001001** | **2** | **X** | **1** | **1** | **1** | **1** | **0** | **X** | **0** | **0** |
| **mult** | **000000** | **011000** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** |
| **multu** | **000000** | **011001** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** |
| **div** | **000000** | **011010** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** |
| **divu** | **000000** | **011011** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** |
|  | **op**  **code** | **func** | **NPC**  **Sel** | **EXT**  **Sel** | **PC**  **Sel** | **Reg**  **Write** | **Reg**  **Dst** | **ALU**  **Out**  **put**  **Sel** | **ALU**  **Sel** | **MUXALUB**  **Sel** | **Mem**  **Write** | **Mem**  **toReg** |
| **mfhi** | **000000** | **010000** | **0** | **0** | **0** | **1** | **1** | **2** | **X** | **X** | **0** | **0** |
| **mflo** | **000000** | **010010** | **0** | **0** | **0** | **1** | **1** | **3** | **X** | **X** | **0** | **0** |
| **mthi** | **000000** | **010001** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** |
| **mtlo** | **000000** | **010011** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** |

**特殊指令的控制信号：**

**（1）移位类指令**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **op**  **code** | **func** | **NPC**  **Sel** | **EXT**  **Sel** | **PC**  **Sel** | **Reg**  **Write** | **Reg**  **Dst** | **ALU**  **Out**  **put**  **Sel** | **ALU**  **Sel** | **MUXALUASel** | **MUXALUB**  **Sel** | **Mem**  **Write** | **Mem**  **toReg** |
| **sll** | **000000** | **000000** | **X** | **3** | **0** | **1** | **1** | **0** | **3** | **1** | **0** | **0** | **0** |
| **sllv** | **000000** | **000100** | **X** | **X** | **0** | **1** | **1** | **0** | **3** | **0** | **0** | **0** | **0** |
| **srl** | **000000** | **000010** | **X** | **3** | **0** | **1** | **1** | **0** | **4** | **1** | **0** | **0** | **0** |
| **srlv** | **000000** | **000110** | **X** | **X** | **0** | **1** | **1** | **0** | **4** | **0** | **0** | **0** | **0** |
| **sra** | **000000** | **000011** | **X** | **3** | **0** | **1** | **1** | **0** | **5** | **1** | **0** | **0** | **0** |
| **srav** | **000000** | **000111** | **X** | **X** | **0** | **1** | **1** | **0** | **5** | **0** | **0** | **0** | **0** |

**（2）跳转类指令**

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **op**  **code** | **func** | **NPC**  **Sel** | **EXT**  **Sel** | **PC**  **Sel** | **Reg**  **Write** | **Reg**  **Dst** | **ALU**  **Out**  **put**  **Sel** | **ALU**  **Sel** | **MUXALUB**  **Sel** | **Mem**  **Write** | **Mem**  **toReg** |
| **beq** | **000100** | **N/A** | **0** | **X** | **1** | **0** | **X** | **X** | **0** | **0** | **0** | **0** |
| **bne** | **000101** | **N/A** | **0** | **X** | **1** | **0** | **X** | **X** | **0** | **0** | **0** | **0** |
| **blez** | **000110** | **N/A** | **0** | **X** | **1** | **0** | **X** | **X** | **0** | **0** | **0** | **0** |
| **bgtz** | **000111** | **N/A** | **0** | **X** | **1** | **0** | **X** | **X** | **0** | **0** | **0** | **0** |
| **bltz** | **000001** | **00000** | **0** | **X** | **1** | **0** | **X** | **X** | **0** | **0** | **0** | **0** |
| **bgez** | **000001** | **00001** | **0** | **X** | **1** | **0** | **X** | **X** | **0** | **0** | **0** | **0** |

**（3）小于则置位类指令**

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **op**  **code** | **func** | **NPC**  **Sel** | **EXT**  **Sel** | **PC**  **Sel** | **Reg**  **Write** | **Reg**  **Dst** | **ALU**  **Out**  **put**  **Sel** | **ALU**  **Sel** | **MUXALUB**  **Sel** | **Mem**  **Write** | **Mem**  **toReg** |
| **slt** | **000000** | **101010** | **X** | **X** | **0** | **1** | **1** | **0** | **7** | **0** | **0** | **0** |
| **slti** | **001010** | **N/A** | **X** | **1** | **0** | **1** | **0** | **0** | **7** | **1** | **0** | **0** |
| **sltu** | **000000** | **101011** | **X** | **X** | **0** | **1** | **1** | **0** | **7** | **10** | **0** | **0** |
| **sltiu** | **001011** | **N/A** | **X** | **1** | **0** | **1** | **0** | **0** | **7** | **10** | **0** | **0** |

**（4）Load和Store类指令**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **op**  **code** | **func** | **NPC**  **Sel** | **EXT**  **Sel** | **PC**  **Sel** | **Reg**  **Write** | **Reg**  **Dst** | **ALU**  **Out**  **put**  **Sel** | **ALU**  **Sel** | **MUXALUB**  **Sel** | **Mem**  **Write** | **StoreSel** | **Mem**  **toReg** | **LoadSel** |
| **lw** | **100011** | **N/A** | **X** | **1** | **0** | **1** | **0** | **0** | **0** | **1** | **0** | **X** | **1** | **000** |
| **lb** | **100000** | **N/A** | **X** | **1** | **0** | **1** | **0** | **0** | **0** | **1** | **0** | **X** | **1** | **001** |
| **lbu** | **100100** | **N/A** | **X** | **1** | **0** | **1** | **0** | **0** | **0** | **1** | **0** | **X** | **1** | **010** |
| **lh** | **100001** | **N/A** | **X** | **1** | **0** | **1** | **0** | **0** | **0** | **1** | **0** | **X** | **1** | **011** |
| **lhu** | **100101** | **N/A** | **X** | **1** | **0** | **1** | **0** | **0** | **0** | **1** | **0** | **X** | **1** | **100** |
| **sw** | **101011** | **N/A** | **X** | **1** | **0** | **0** | **X** | **0** | **0** | **1** | **1** | **00** | **0** | **X** |
| **sb** | **101000** | **N/A** | **X** | **1** | **0** | **0** | **X** | **0** | **0** | **1** | **1** | **01** | **0** | **X** |
| **sh** | **101001** | **N/A** | **X** | **1** | **0** | **0** | **X** | **0** | **0** | **1** | **1** | **10** | **0** | **X** |

**（5）乘除类指令**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **op**  **code** | **func** | **NPC**  **Sel** | **EXT**  **Sel** | **PC**  **Sel** | **Reg**  **Write** | **Reg**  **Dst** | **ALU**  **Out**  **put**  **Sel** | **ALU**  **Sel** | **MUXALUB**  **Sel** | **start** | **MDSel** | **Mem**  **Write** | **Mem**  **toReg** |
| **mult** | **000000** | **011000** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **1** | **1** | **0** | **0** |
| **multu** | **000000** | **011001** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **1** | **2** | **0** | **0** |
| **div** | **000000** | **011010** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **1** | **3** | **0** | **0** |
| **divu** | **000000** | **011011** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **1** | **4** | **0** | **0** |
| **mfhi** | **000000** | **010000** | **0** | **0** | **0** | **1** | **1** | **2** | **X** | **X** | **0** | **0** | **0** | **0** |
| **mflo** | **000000** | **010010** | **0** | **0** | **0** | **1** | **1** | **3** | **X** | **X** | **0** | **0** | **0** | **0** |
| **mthi** | **000000** | **010001** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **1** | **5** | **0** | **0** |
| **mtlo** | **000000** | **010011** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **1** | **6** | **0** | **0** |

**（1）Controller\_D**

**①模块接口**

|  |  |  |
| --- | --- | --- |
| 信号 | 方向 | 描述 |
| Instr\_D | I | 输入来自D级寄存器的指令 |
| CMPOut | I | 输入来自D级部件CMP的比较结果 |
| NPCSel | O | 输出给NPC的选择信号 |
| EXTSel | O | 输出给EXT的选择信号 |
| RegWrite\_D | O | 输出给E级寄存器的GRF写信号 |
| PCSel | O | 输出给F级部件MUXPC的选择信号 |

**②功能定义**

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 选择指令跳转类型 | 通过NPCSel选择指令跳转地址 |
| 2 | 选择是否跳转 | 通过PCSel选择是否跳转 |
| 3 | 选择立即数扩展类型 | 通过EXTSel选择扩展类型 |
| 4 | 判断指令是否进行写寄存器操作 | 通过RegWrite\_D进行判断 |

**（2）Controller\_E**

**①模块接口**

|  |  |  |
| --- | --- | --- |
| 信号 | 方向 | 描述 |
| Instr\_E | I | 输入来自E级寄存器的指令 |
| ALUSel | O | 输出ALU操作选择信号 |
| MUXALUBSel | O | 输出ALUB功能选择信号，选择ALU操作数B是寄存器数还是立即数 |
| MUXALUASel | O | 输出ALUA功能选择信号，选择ALU操作数A是来自寄存器RS还是立即数的位移字段 |
| RegDst | O | 输出当前指令的写寄存器选择信号 |
| ALUOutputSel | O | 输出ALU的结果选择信号，是jal型对应的PC+8还是正常的ALU运算结果 |
| Start | O | 输出乘除模块的运行开始信号 |
| MDSel | O | 输出乘除模块的操作选择信号 |

**②功能定义**

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 选择ALU运算类型 | 通过ALUSel选择当前E级指令对应的ALU运算 |
| 2 | 选择ALU运算数 | 通过MUXALUBSel选择当前E级指令对应的ALU运算数是寄存器数还是立即数 |
| 3 | 选择写寄存器地址 | 通过RegDst选择当前E级指令写入寄存器的地址是rt 还是rd还是31 |
| 4 | 选择ALU输出 | 通过ALUOutputSel选择当前E级指令要用的ALU输出是PC+8（JAL）还是正常的运算结果（正常运算指令） |
| 6 | 选择乘除模块是否开始工作 | 通过Start信号控制乘除模块是否开始工作 |
| 7 | 选择乘除模块运算类型 | 通过MDSel选择乘除模块的运算类型 |

**（3）Controller\_M**

**①模块接口**

|  |  |  |
| --- | --- | --- |
| 信号 | 方向 | 描述 |
| Instr\_M | I | 来自M级的指令 |
| MemWrite | O | 输出内存写使能信号 |
| StoreSel | O | 输出内存写数据的形式选择信号 |

**②功能定义**

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 判断是否写内存 | 根据当前的M级指令输出MemWrite信号控制是否写内存 |
| 2 | 选择内存写入数据的形式 | 根据当前的M级指令输出StoreSel信号控制内存写入数据的形式 |

**（4）Controller\_W**

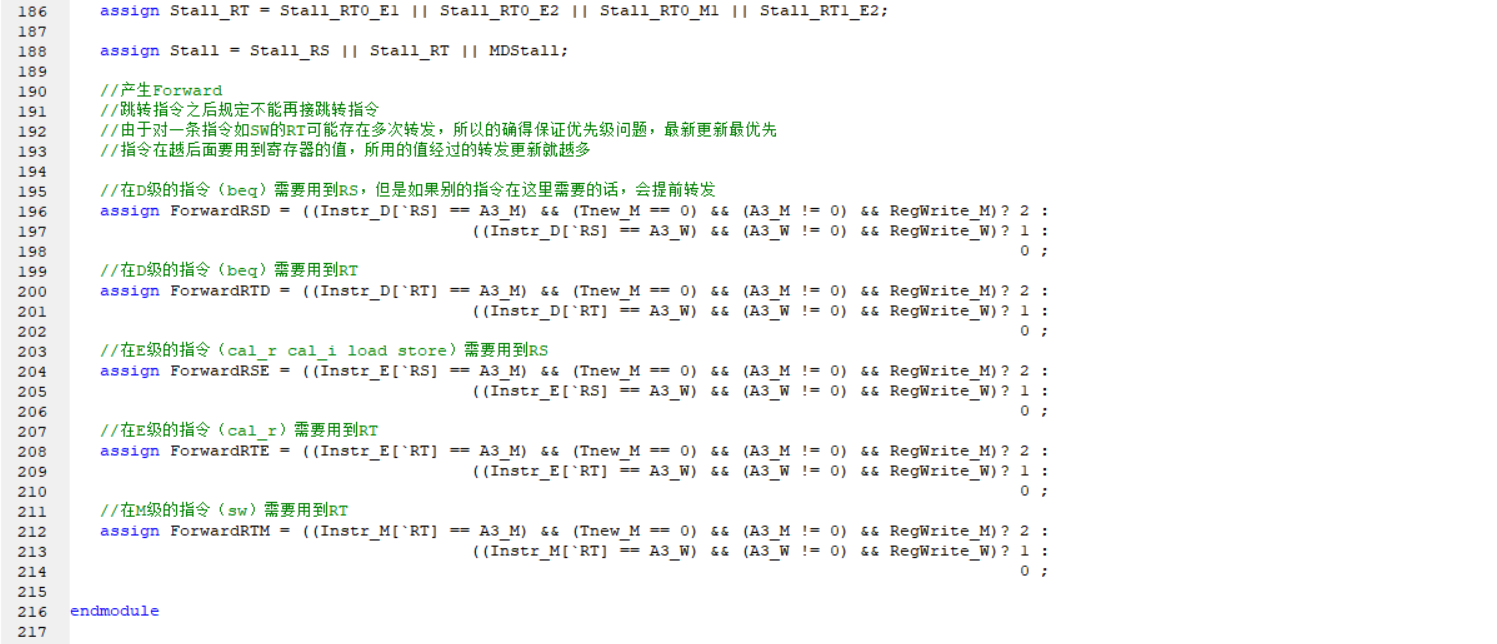
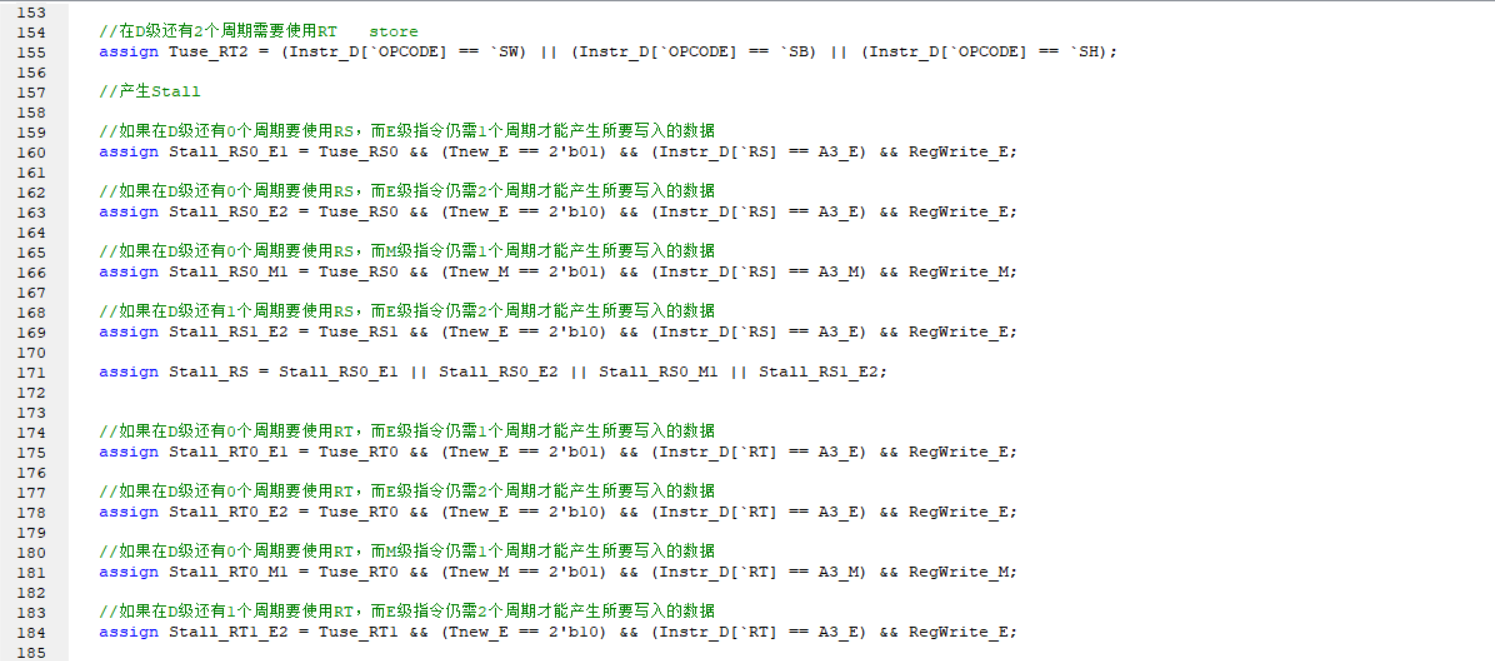
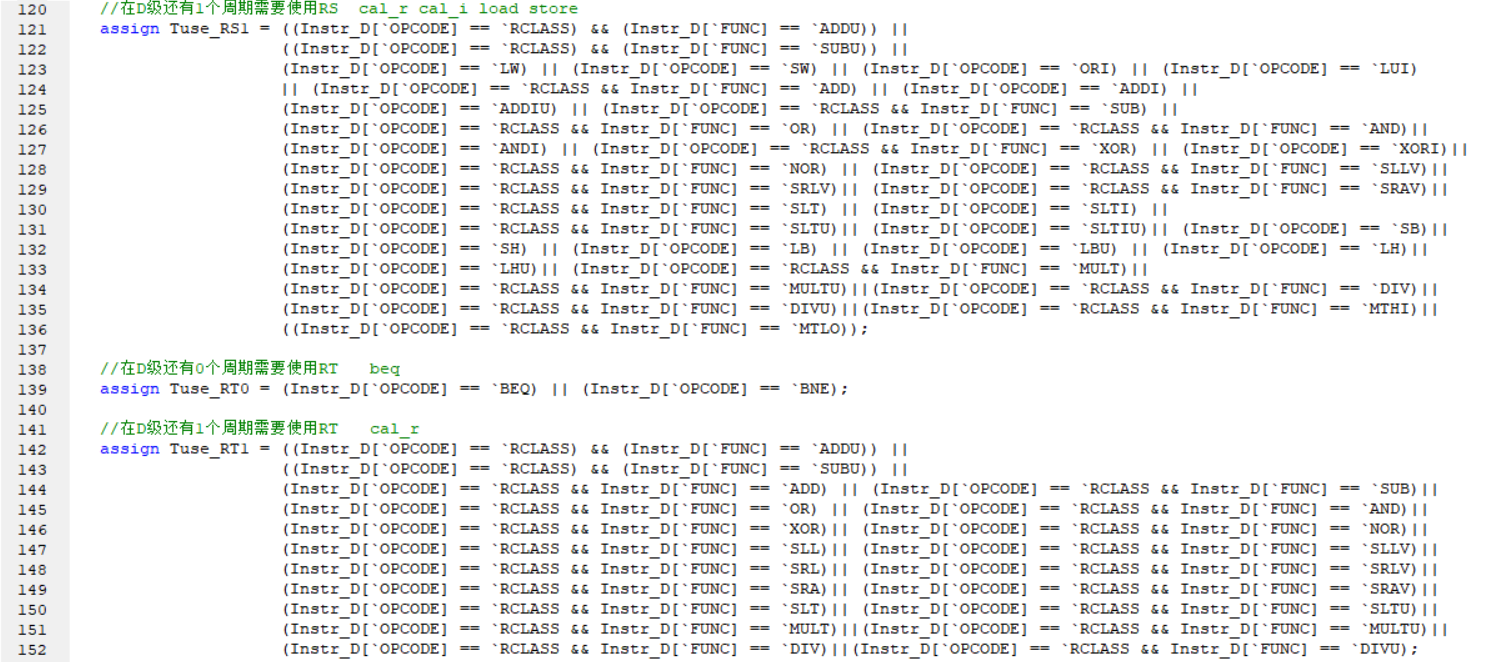
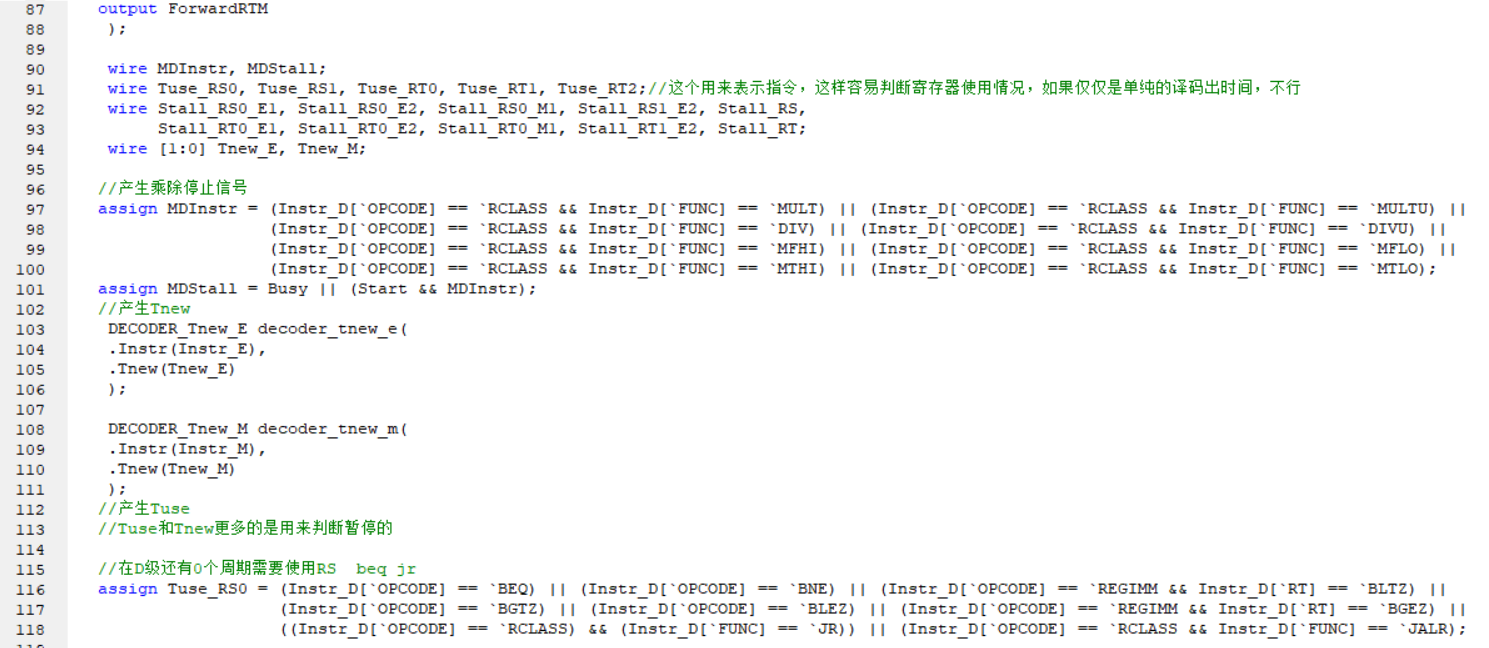
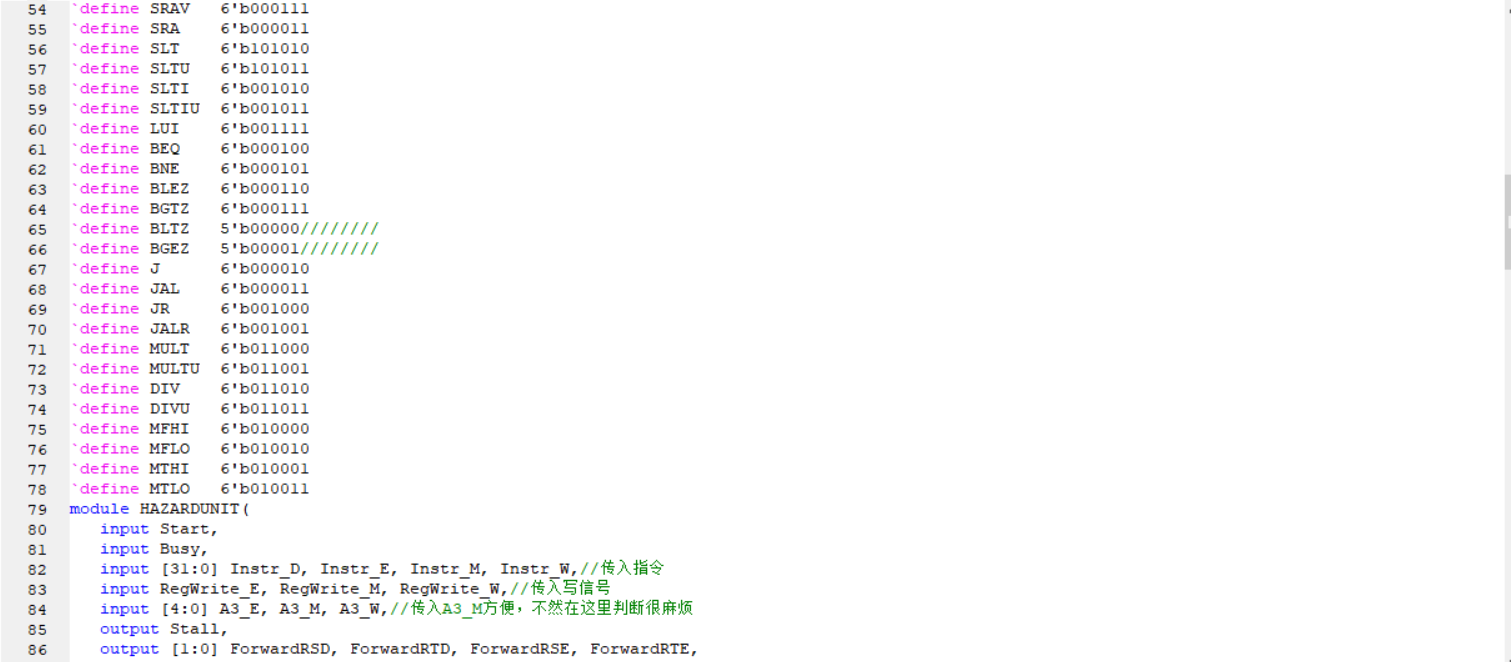
**①模块接口**

|  |  |  |
| --- | --- | --- |
| 信号 | 方向 | 描述 |
| Instr\_W | I | 来自W级的指令 |
| MemtoReg\_W | O | MUXRFWD的选择信号 |
| LoadSel | O | 输出读内存指令要读的数据的形式控制信号 |

**②功能定义**

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 输出选择写入GRF的数据的控制信号 | load指令1，其他指令0 |
| 2 | 输出读内存指令要读的形式控制信号 | LoadSel:  000 lw  001 lb  010 lub  011 lh  100 lhu |

**11、HAZARDUNIT**



1. **模块接口**

|  |  |  |
| --- | --- | --- |
| 信号 | 方向 | 描述 |
| Start | I | 输入来自E级部件的E级控制器的乘除模块运行开始标志 |
| Busy | I | 输入来自E级部件的E级乘除模块的运行中的信号 |
| Instr\_D | I | 输入来自D级寄存器的指令 |
| Instr\_E | I | 输入来自E级寄存器的指令 |
| Instr\_M | I | 输入来自M级寄存器的指令 |
| Instr\_W | I | 输入来自W级寄存器的指令 |
| RegWrite\_E | I | 输入来自E级的写使能信号 |
| RegWrite\_M | I | 输入来自M级的写使能信号 |
| RegWrite\_W | I | 输入来自W级的写使能信号 |
| A3\_E | I | 输入来自E级的写地址 |
| A3\_M | I | 输入来自M级的写地址 |
| A3\_W | I | 输入来自W级的写地址 |
| Stall | O | 输出暂停信号 |
| ForwardRSD | O | 输出RSD转发控制信号 |
| ForwardRTD | O | 输出RTD转发控制信号 |
| ForwardRSE | O | 输出RSE转发控制信号 |
| ForwardRTE | O | 输出RTE转发控制信号 |
| ForwardRTM | O | 输出RTM转发控制信号 |

1. **功能定义**

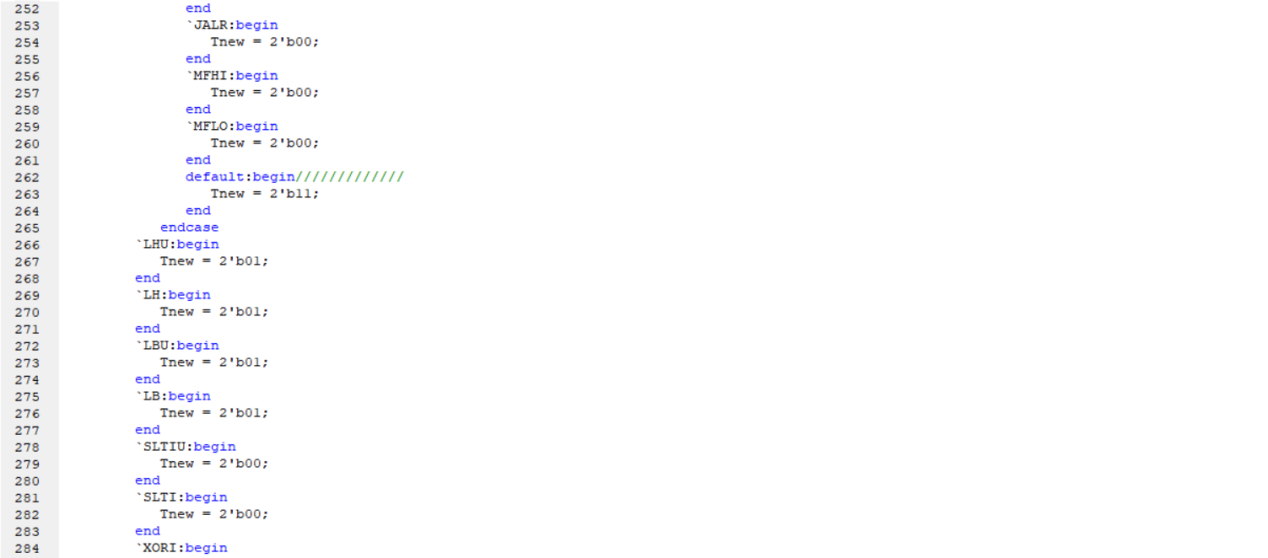
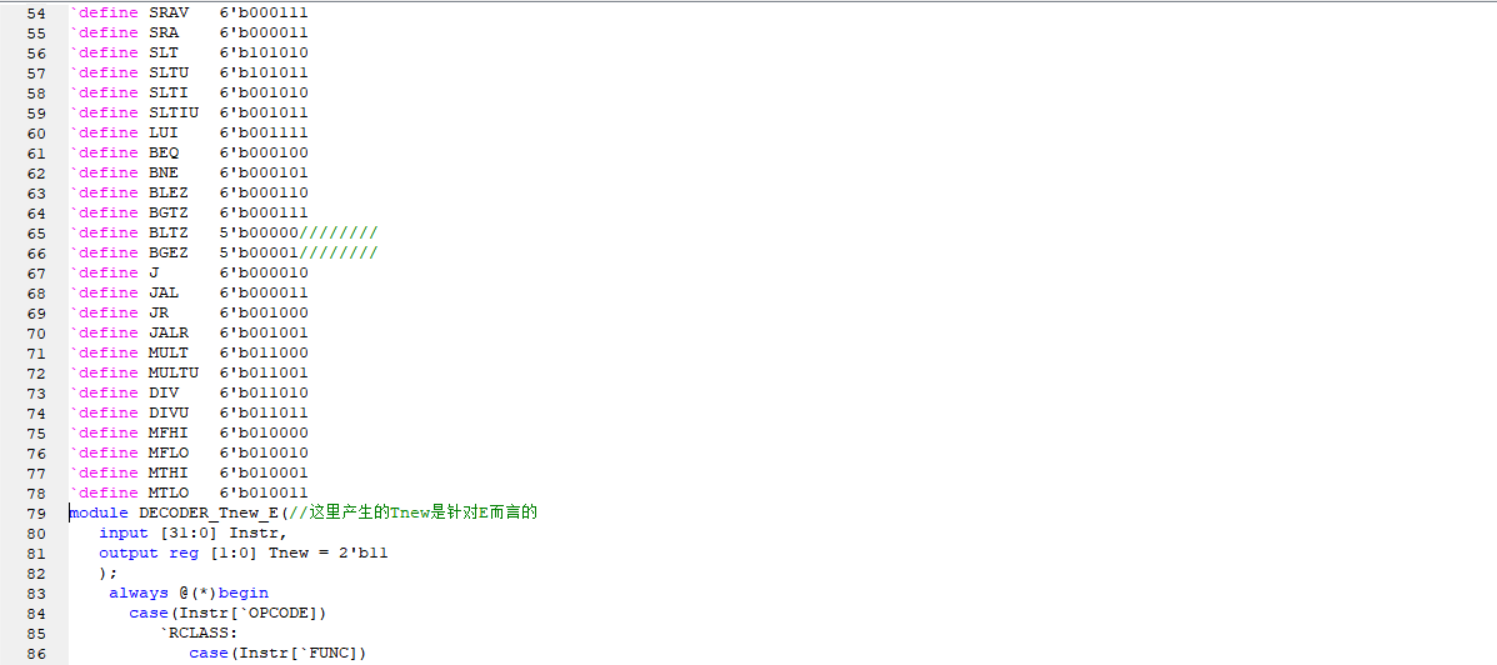
|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 判断暂停 | 当前D级指令所需使用的寄存器的时间周期大于E、M、W级相同的寄存器写入指令的产生写入数据时间周期时，需要暂停。有两类寄存器和八种情况  RS0\_E1 RS0\_E2  RS0\_M1 RS1\_E2  RT0\_E1 RT0\_E2  RT0\_M1 RT1\_E2 |
| 2 | 判断转发 | 当前D级指令所需使用的寄存器的时间周期小于或等于E、M、W级相同的寄存器写入指令的产生写入数据时间周期时，可以转发。有两类寄存器和五种情况  FRSD FRTD  FRSE FRTE  FRTM |

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | cal\_r | | cal\_i | | store | load |  |  |  |  |  |
| addu | subu | ori | lui | sw | lw | beq | jr | jal | nop | j |
| Tuse\_RS0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | X | X | X |
| Tuse\_RS1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | X | X | X |
| Tuse\_RT0 | 0 | 0 | X | X | 0 | X | 1 | X | X | X | X |
| Tuse\_RT1 | 1 | 1 | X | X | 0 | X | 0 | X | X | X | X |
| Tuse\_RT2 | 0 | 0 | X | X | 1 | X | 0 | X | X | X | X |
| Tnew | 1 | 1 | 1 | 1 | 3 | 2 | 3 | 3 | 3 | 3 | 3 |

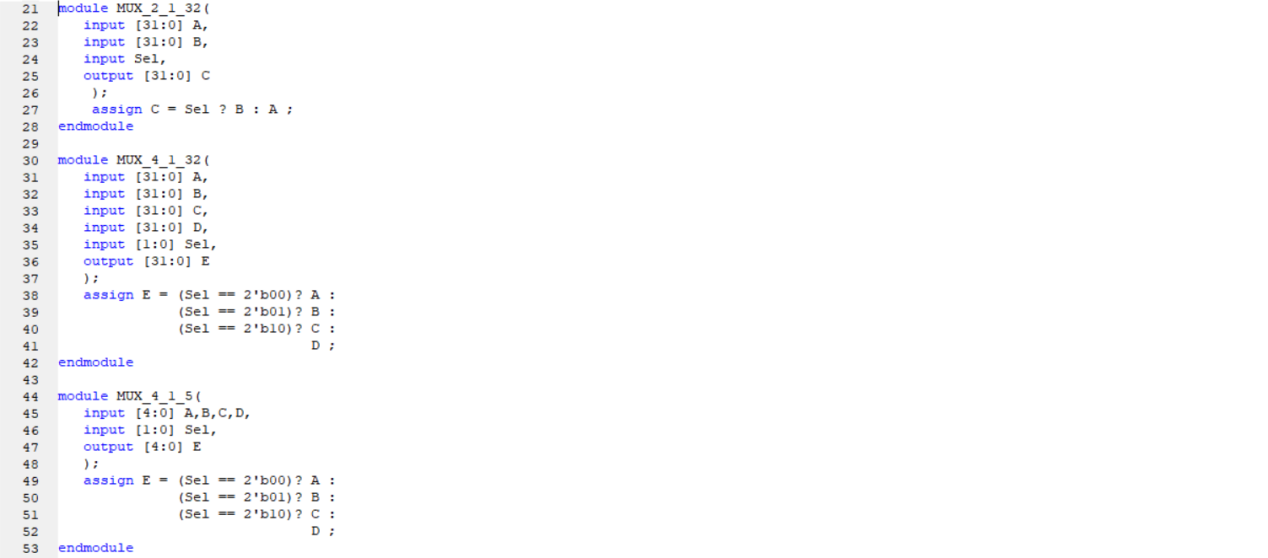
注：Tnew=3 代表不产生写入寄存器数据



1. **内部部件DECODER**



1. **MUX**



1. **测试程序**

# basic compute

ori $t0 $0 1

ori $t1 $0 2

addu $t2 $t0 $t1

add $t3 $t2 $t1

lui $t4 0xffff

ori $t4 $t4 0xfff0

addu $t5 $t3 $t4

add $t6 $t5 $t4

lui $t0 0x7fff

ori $t0 $t0 0x1234

addiu $t0 $t0 0xffff

addi $t0 $t0 0xffff

add $t7 $t0 $t6

lui $t8 0xaaaa

addu $t9 $t7 $t8

addiu $t9 $t9 0xffffaaaa

addiu $t9 $t9 -1

addi $t9 $t9 -1

subu $s0 $t9 $t7

subu $s1 $t7 $t9

subu $s2 $s1 $s0

lui $t0 0xffff

ori $t0 $t0 0xfff0

ori $t1 $0 10

sub $s3 $t1 $t0

sub $s4 $t0 $t1

sub $s3 $s4 $s3

# load store

addi $s7 $0 8

addi $t4 $t4 -1

sw $t4 -8($s7)

lb $t0 -8($s7)

addi $t5 $t5 2

sb $t5 -4($s7)

lbu $t1 -7($s7)

lh $t2 -4($s7)

subu $t6 $t6 $s7

sb $t6 -3($s7)

lhu $t3 -4($s7)

sh $t7 -2($s7)

lw $t0 -4($s7)

ori $s0 $s0 1

sw $s0 0($s7)

ori $s1 $s1 2

sh $s1 4($s7)

lw $t1 0($s7)

addi $s2 $s2 -2

sb $s2 6($s7)

lh $t3 6($s7)

sub $s3 $s3 $s7

sb $s3 7($s7)

lbu $t0 7($s7)

# logic

lui $t0 0xabcd

lui $t1 0xcdef

ori $t0 $t0 0x1234

ori $t1 $t1 0x5678

and $t2 $t0 $t1

or $t3 $t2 $t1

and $t4 $t3 $t2

or $t5 $t4 $t3

andi $t6 $t5 0xa8b6

ori $t7 $t6 0x12b4

xor $t8 $t7 $t6

xor $t9 $t8 $t7

xori $t0 $t9 0x7823

andi $t1 $t0 0xffff

nor $t2 $t1 $t0

nor $t3 $t2 $t1

xori $t4 $t3 0xabcd

nor $t5 $t4 $t3

#branch

ori $t0 $0 1

ori $t1 $0 2

lui $t2 0xffff

ori $t2 $t2 0xffff

sub $t2 $t2 $t0

sub $t2 $t2 $t0

bne $t0 $t1 mark1

lui $t4 0xffff

mark1\_back:

ori $s0 $0 1

beq $t3 $t4 mark2

addi $t3 $t3 1

mark2\_back:

bltz $t2 mark3

addi $t2 $t2 1

mark3\_back:

addi $t2 $t2 1

blez $t2 mark4

subu $t2 $t2 $t1

mark4\_back:

bgtz $t2 mark5

add $t2 $t2 $t1

mark5\_back:

bgez $t2 mark6

addi $t2 $t2 1

mark6\_back:

j end

nop

mark1:

ori $s0 $0 2

bne $t4 $0 mark1\_back

lui $t3 0xffff

mark2:

addi $t4 $t4 1

beq $t3 $t4 mark2\_back

nop

mark3:

sub $t2 $t2 $t0

bltz $t2 mark3\_back

nop

mark4:

blez $t2 mark4\_back

addi $t2 $t2 112

mark5:

bgtz $t2 mark5\_back

sub $t2 $t2 $t1

mark6:

bgez $0 mark6\_back

end:

#shift

lui $t0 0xfff0

ori $t0 $t0 0xabcd

ori $t1 $0 3

sll $t2 $t0 2

sllv $t2 $t2 $t1

srl $t2 $t2 3

addi $t1 $t1 -1

srlv $t2 $t2 $t1

addi $t1 $t1 3

srlv $t2 $t2 $t1

lui $t3 0x8000

or $t2 $t3 $t2

sra $t2 $t2 4

addi $t1 $t1 -3

srav $t2 $t2 $t1

lui $t3 0x7fff

ori $t3 $t3 0xffff

and $t2 $t2 $t3

sra $t2 $t2 2

srav $t2 $t2 $t1

#jump

j mark7

ori $s0 $0 0xffff

mark7\_back:

jal mark8

nop

ori $s0 $0 0xaaaa

addiu $s3 $s3 8

jr $s3

nop

ori $s0 $0 0xbbbb

j mark9

nop

mark7:

j mark7\_back

addu $s1 $s0 $s0

mark8:

addiu $s2 $ra 16

jalr $s3 $s2

lui $s7 0xffff

mark9:

jr $ra

nop

jal mark10

lui $s6 0xffff

mark10:

addi $s5 $ra 16

jalr $s4 $s5

lui $t7 0xbbbb

lui $t9 0xcccc

j end\_2

ori $s7 0xffff

lui $t9 0xffff

lui $t8 0xaaaa

jr $s4

end\_2:

#mult\_div

ori $t0 $0 3

ori $t1 $0 4

lui $t2 0xffff

ori $t2 $t2 0xfff0

lui $t3 0x7000

ori $t3 $t3 0xffff

lui $t4 0xabcd

ori $t4 $t4 0xefef

mult $t0 $t1

mflo $s0

mthi $t4

mfhi $t4

mult $t2 $t3

mfhi $s1

mflo $s2

multu $t3 $t2

mfhi $s1

mflo $s2

or $s3 $s1 $0

mthi $s2

mtlo $s3

multu $t2 $t4

mfhi $s3

mflo $s4

div $t0 $t1

mfhi $s0

mflo $s1

div $t2 $t0

mfhi $s2

mflo $s3

divu $t2 $t0

mfhi $s2

mflo $s3

div $t3 $t4

mfhi $s2

mflo $s3

mthi $0

mtlo $0

mflo $s3

mfhi $s2

divu $t4 $t3

mfhi $s0

mflo $s1

#setbit

ori $t0 $0 2

ori $t1 $0 5

lui $t2 0x7000

lui $t3 0xffff

ori $t3 $t3 0xffff

slt $s0 $t0 $t3

slt $s0 $t3 $t0

slt $s0 $t3 $t3

slti $s1 $t2 0x7fff

slti $s1 $t3 0

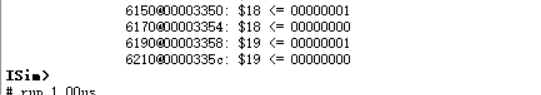
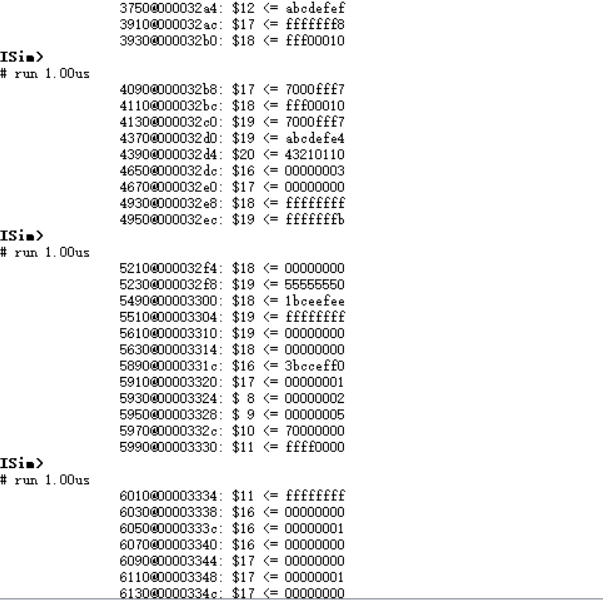
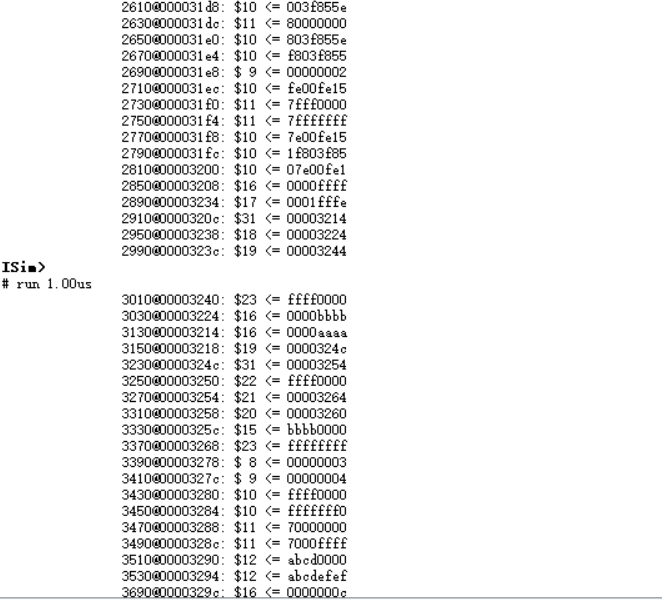
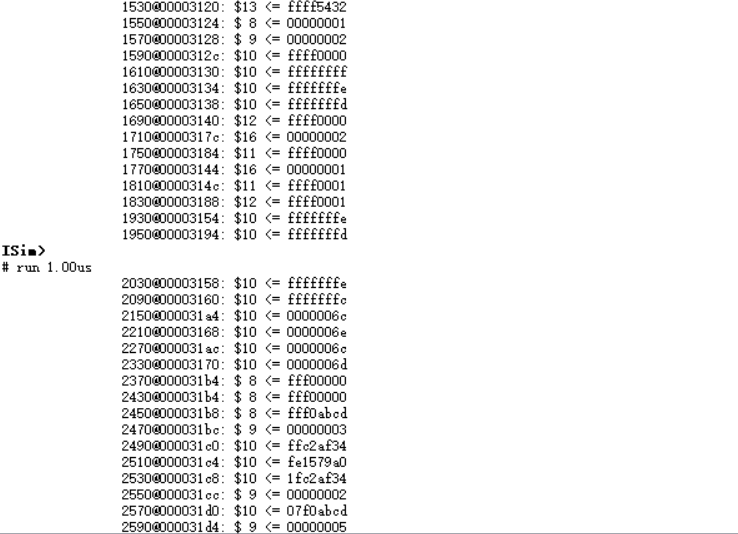
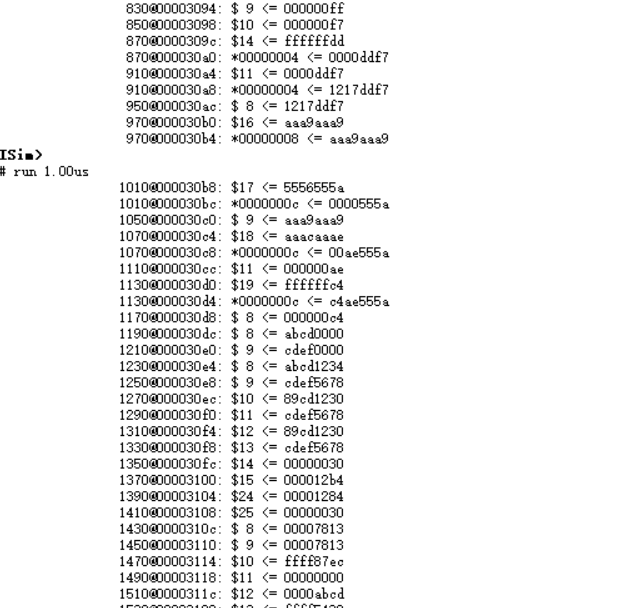
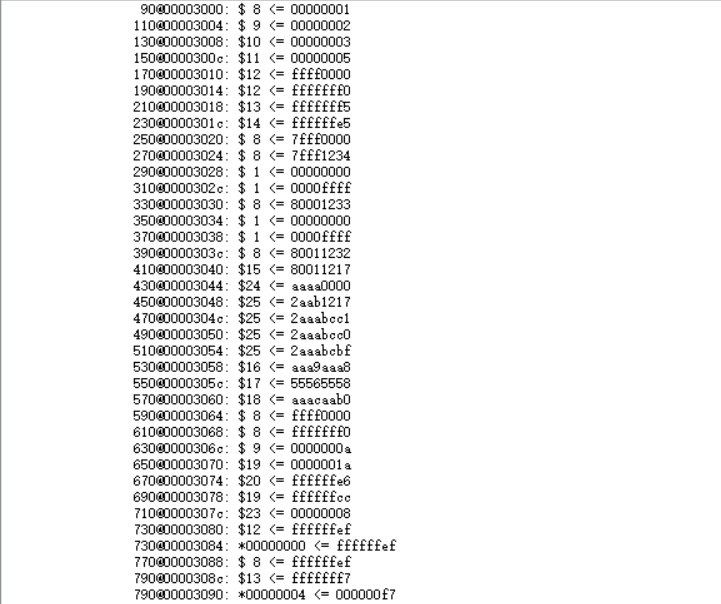
slti $s1 $t3 -1

sltu $s2 $t0 $t3

sltu $s2 $t3 $t2

sltiu $s3 $t1 0x0fff

sltiu $s3 $t3 1



**四、思考题**

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

实际中乘除法运算有延时，并且在我们的CPU设计中也模拟了该延迟，但是正常的ALU运算不模拟延迟，所以如果乘除部件整合进ALU的话会增加CPU的工作延迟，如果设置单独的乘法部件的话，在进行乘除运算的同时不会影响其他运算指令的运行，能提高CPU工作效率，还有就是乘除部件整合进ALU的话，还会增加ALU的控制信号和复杂度，非常不便。

设置独立的HI、LO寄存器是因为，通用寄存器堆中的每一个寄存器都可以被读写，如果HI、LO混进通用寄存器堆的话，很容易产生乘除数据被修改的意外情况。其次，如果HI、LO寄存器混进通用寄存器堆的话，就会造成乘除指令要进行写入寄存器的操作，从而会影响到其他写回寄存器指令的运行，还得设置相应的暂停机制。

1. **参照你对延迟槽的理解，试解释“乘除槽”。**

MIPS的延迟槽是伴随着流水线的结构而产生的。实验中考虑更多的是分支延迟槽，只有分支指令进入ID级执行后，下一条指令的地址才能被获知，假若无延迟槽，在等待下一条指令地址的时候必须让流水线暂停一个周期，而有了延迟槽之后，在等待下一条指令地址的过程中可以执行分支指令的下一条指令，这样做的好处是提前假设分支不发生，执行了下一条指令，提高效率，还有就是即使分支发生了，分支指令的下一条指令也会被执行，提高了指令运行的效率。

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

在实际情况中，DM的写入一般是先写入到cache中，然后再向主存储器写入，即只要写入cache，向主存储器写入的过程与CPU时钟周期，但是DM的读取，最坏情况下是从主存储器中读取数据，这就意味着，平均情况下DM的写入要比读取快。所以，假若把数据扩展模块放在DM之后，读取数据在M级会经过DM和扩展部件，会使读取的时钟周期变长，进而会增加流水线的CPU时钟周期。但是如果放在MEM/WB之后的话，LOAD类指令在M级的执行时间不变，在W级写入寄存器时不影响后续存取指令在M级的执行，所以大概率不会增加CPU时钟周期。

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）**

C语言中每个字符占一字节的存储空间，如果按字访问的话，在运行C程序时为避免字符串的读取错误，势必会将一个字符存储在一个字内，造成巨大的空间浪费。所以相比于按字存储，按字节存储则方便和节省空间得多。

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？**

我觉得我的设计风格更偏向于第一种“侦测者”型。从产生数据冲突的原因入手，展示了数据冲突产生的逻辑所在，代码短小，很多时候加指令时不用改动。

对抗复杂性：一、在总体设计中多抽象了一层不同的流水级部件F、D、E、M、W级，这样做的好处是将不同的流水级部件封装到一个模块中，只保留其与其他模块的接口，从而其内部的连线不会对其他模块产生影响，使顶层连线更为简单和清晰。二、在设计流水线信号时会在信号后面加上下划线和流水级的缩写，表明这个信号所处的流水线级。三、使用宏定义避免大量输入01识别码而产生的输入错误。

1. **你对流水线CPU设计风格有何见解？**

在设计暂停和转发时，按逻辑进行设计会更加抓住CPU的本质，但是不直观；按指令类型进行设计，直观但是又略显冗杂。

**7.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。(**非常重要**)**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 序号 | 类型 | Tuse\_RS | Tuse\_RT | Tnew\_E | 冲突寄存器 | 解决方案 | 测试程序 |
| 1 | R-J | 0 |  | 1 | RS | 暂停1个时钟周期后从M到D转发 | ori $ra $0,0x0000300c  jr $ra |
| 2 | L-J | 0 |  | 2 | RS | 暂停2个时钟周期后从W到D转发 | lw $ra,0($0)  jr $ra |
| 3 | R-B | 0 | 0 | 1 | RS或RT | 暂停1个时钟周期后从M到D转发 | addu $t1,$t1,$t2  beq $t1,$t2,loop |
| 4 | L-B | 0 | 0 | 2 | RS或RT | 暂停两个时钟周期后从W到D转发 | lw $t1,0($0)  beq $t1,$t2,loop |
| 5 | R-R | 1 | 1 | 1 | RS或RT | 从M到E转发 | addu $t1,$t1,$t2  subu $t1,$t1,$t2 |
| 6 | L-R | 1 | 1 | 2 | RS或RT | 暂停一个时钟周期后从W到E转发 | lw $t1,0($0)  addu $t1,$t1,$t2 |
| 7 | R-R | 1 |  | 1 | RS | 从M到E级转发 | ori $t1,$0,1  lui $t1,1 |
| 8 | L-R | 1 |  | 2 | RS | 暂停一个时钟周期后从W到E转发 | lw $t1,0($0)  lui $t1,1 |
| 9 | R-S | 1 | 2 | 1 | RS | 从M到E级转发 | lui $t1,1  sw $t1,0($0) |
| 10 | L-S | 1 | 2 | 2 | RS | 暂停1个周期后从W到E级转发 | lw $t1,0($0)  sw $t2,0($t1) |
| 11 | L-S | 1 | 2 | 2 | RT | 从W到M级转发 | lw $t1,0($0)  sw $t1,0($0) |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 序号 | 类型 | Tuse\_RS | Tuse\_RT | Tnew\_M | 冲突寄存器 | 解决方案 | 测试程序 |
| 1 | L-R-J | 0 |  | 1 | RS | 暂停一个周期后从W到D级转发 | lw $ra,0($0)  lui $t2,1  jr $ra |
| 2 | L-R-B | 0 | 0 | 1 | RT或RT | 暂停一个周期后从W到D级转发 | lw $t1,0($0)  lui $t2,1  beq $t1,$t2,loop |
| 3 | L-R-R | 1 | 1 | 1 | RS或RT | 从W到E级转发 | lw $t1,0($0)  lui $t2,1  addu $t1,$t3,$t1 |
| 4 | L-R-R | 1 |  | 1 | RS | 从W到E级转发 | lw $t1,0($0)  lui $t2,1  ori $t1,$t1,1 |
| 5 | L-R-S | 1 | 2 | 1 | RS | 从W到E级转发 | lw $t1,0($0)  lui $t2,1  sw $t1,0($0) |
| 6 | R-R-J | 0 |  | 0 | RS | 从M级到D级转发 | addu $ra,$t1,$t2  lui $t2,1  jr $ra |
| 7 | R-R-B | 0 | 0 | 0 | RS或RT | 从M级到D级转发 | addu $t1,$t1,$t2  lui $t3,1  beq $t1,$t2,loop |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 序号 | 类型 | Tuse\_RS | Tuse\_RT | Tnew\_W | 冲突寄存器 | 解决方案 | 测试程序 |
| 1 | R-N-N-J | 0 |  | 0 | RS | 从W到D级转发 | addu $ra,$t1,$t2  nop  nop  jr $ra |
| 2 | R-N-N-B | 0 | 0 | 0 | RS或RT | 从W到D级转发 | addu $t1,$t1,$t2  nop  nop  beq $t1,$t2,loop |