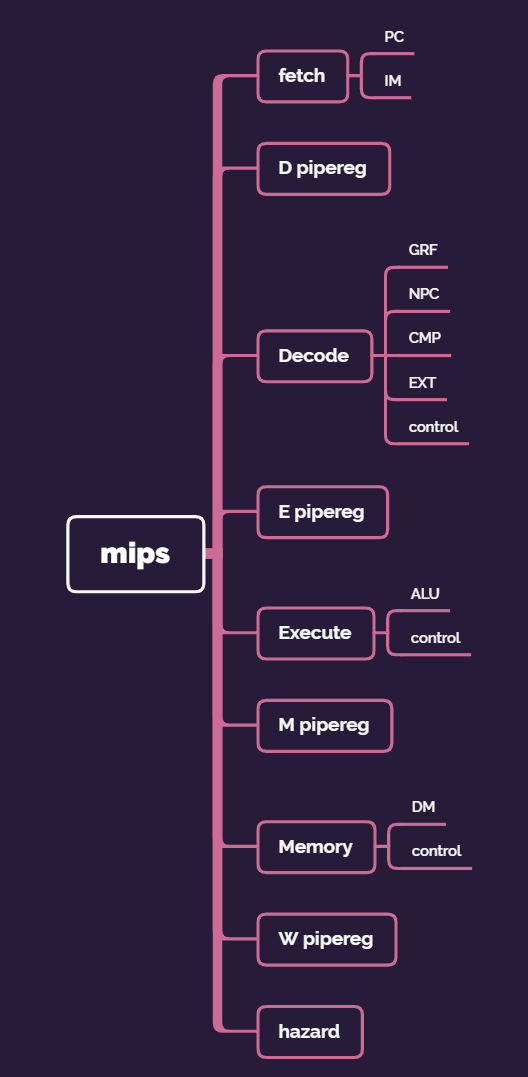
# 计算机组成原理实验报告参考模板

## 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{addu， subu，ori，lw，sw，beq，lui，j，jal，jr，nop}。为了实现这些功能，CPU主要包含了IM、GRF等模块，这些模块按照以下顶层设计逐级展开。



### （二）关键模块定义

#### 1.PC

（1） 端口说明

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 信号名 | 方向 | 描述 |
| 1 | clk | I | 时钟信号 |
| 2 | reset | I | 同步置零 |
| 3 | NPC | I | 下一条指令在IM的地址 |
| 4 | en | I | 阻塞信号 |
| 5 | PC | O | 当前指令在IM的地址 |

（2）功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能 | 描述 |
| 1 | 存储指令地址 | 保存当前执行指令在IM中的地址 |
| 2 | 阻塞 | 阻塞为0时，PC不工作 |

#### 2. IM

（1）端口说明

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 信号名 | 方向 | 描述 |
| 1 | PC[31:0] | I | 当前指令在IM的地址 |
| 2 | Instr[31:0] | O | 当前PC对应的指令 |

（2）功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能 | 描述 |
| 1 | 取指令 | 读取指令 |

#### 3. D\_pipereg

（1）端口说明

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 信号名 | 方向 | 描述 |
| 1 | clk | I | 时钟信号 |
| 2 | reset | I | 同步复位信号 |
| 3 | en | I | 阻塞信号 |
| 4 | F\_PC | I | F级的PC |
| 5 | F\_instr | I | F级的指令 |
| 6 | D\_PC | O | D级的PC |
| 7 | D\_instr | O | D级的指令 |

（2）功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能 | 描述 |
| 1 | 存储要流水的值 | 存储要流水的值 |

#### 4. NPC

（1）端口说明

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 信号名 | 方向 | 描述 |
| 1 | PCSrc | I | 分支信号 |
| 2 | PCj | I | j型跳转信号 |
| 3 | PCjr | I | jr和jalr跳转信号 |
| 4 | D\_PC[31:0] | I | F级PC |
| 5 | F\_PC[31:0] | I | F级PC |
| 6 | Regjr[31:0] | I | jr和jalr跳转的地址 |
| 7 | instr\_index[25:0] | I | D级26位立即数 |
| 8 | NPC[31:0] | O | 下一个PC的值 |

（2）功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能 | 描述 |
| 1 | 计算下一个PC | 计算下一个PC |

#### 5. EXT

（1）端口说明

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 信号名 | 方向 | 描述 |
| 1 | imm | I | 16位立即数 |
| 2 | sign | I | 符号扩展选择信号  0：进行无符号扩展  1：进行符号扩展 |
| 3 | SignImm | O | 扩展后的32位数 |

（2）功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能 | 描述 |
| 1 | 无符号扩展 | 当sign为0时，进行无符号扩展 |
| 2 | 符号扩展 | 当sign为1时，进行有符号扩展 |

#### 6. GRF

（1）端口说明

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 信号名 | 方向 | 描述 |
| 1 | clk | I | 时钟信号 |
| 2 | reset | I | 异步复位信号，  1：清零  0：保持 |
| 3 | PC[31:0] | I | D级PC的值，用于display |
| 4 | A1[4:0] | I | 5位地址输入，从外界接收[25：21]，代表32个寄存器中的一个，并将其中的值输出到RD1 |
| 5 | A2[4:0] | I | 5位地址输入，从外界接收[20：16]，代表32个寄存器中的一个，并将其中的值输出到RD2 |
| 6 | A3[4:0] | I | 5位地址输入，选择32个寄存器中的一个，将WD输入的数据存储到其中 |
| 7 | WD[31:0] | I | 32位写入数据 |
| 8 | RD1[31:0] | O | 输出A1代表的寄存器中的值 |
| 9 | RD2[31:0] | O | 输出A2代表的寄存器中的值 |

（2）功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能 | 描述 |
| 1 | 异步复位 | Reset为1时，所有寄存器被清零 |
| 2 | 读数据 | 将A1和A2对应的寄存器中的值输出到RD1和RD2 |
| 3 | 写数据 | 如果WE为1，时钟上升沿时将WD中的数据写入到A3对应的寄存器中 |

#### 7. CMP

（1）端口说明

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 信号名 | 方向 | 描述 |
| 1 | A | I | 操作数A |
| 2 | B | I | 操作数B |
| 3 | D\_equal | O | 相等信号 |

（2）功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能 | 描述 |
| 1 | 判断A、B是否相等 | 若A=B，则输出1 |

#### 8. E\_pipereg

（1）端口说明

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 信号名 | 方向 | 描述 |
| 1 | clk | I | 时钟信号 |
| 2 | reset | I | 同步复位信号 |
| 3 | en | I | 阻塞信号 |
| 4 | flush | I | 清洗信号 |
| 5 | D\_PC[31:0] | I | D级PC值 |
| 6 | D\_instr[31:0] | I | D级指令 |
| 7 | D\_signimm[31:0] | I | D级扩展后32位数 |
| 8 | D\_RD1[31:0] | I | D级GRF输出1 |
| 9 | D\_RD2[31:0] | I | D级GRF输出2 |
| 10 | D\_A3[4:0] | I | D级A3地址 |
| 11 | D\_WD[31:0] | I | D级GRF写入值 |
| 12 | E\_PC[31:0] | O | E级PC值 |
| 13 | E\_instr[31:0] | O | E级指令 |
| 14 | E\_signimm[31:0] | O | E级扩展后32位数 |
| 15 | E\_RD1[31:0] | O | E级RD1 |
| 16 | E\_RD2[31:0] | O | E级RD2 |
| 17 | E\_A3[4:0] | O | E级A3地址 |
| 18 | E\_WD[31:0] | O | E级WD |

（2）功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能 | 描述 |
| 1 | 存储要流水的值 | 存储要流水的值 |

#### 9. ALU

（1）端口说明

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 信号名 | 方向 | 描述 |
| 1 | op[2:0] | I | 选择执行哪一个操作  000：加法  001：减法  010：与  011：或  100：左移16位 |
| 2 | inA[31:0] | I | 参与运算的第一个数 |
| 3 | inB[31:0] | I | 参与运算的第二个数 |
| 4 | ALUResult[31:0] | O | 输出inA和inB操作后的结果 |

（2）功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能 | 描述 |
| 1 | 加法 | 将两个输入加起来输出到ALUResult |
| 2 | 减法 | 将两个输入相减输出到ALUResult |
| 3 | 与运算 | 对两个输入进行与操作输出结果到ALUResult |
| 4 | 或运算 | 对两个输入进行或操作输出结果到ALUResult |
| 5 | 左移16位 | 将SrcB左移16位输出到ALUResult |

#### 10. M\_pipereg

（1）端口说明

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 信号名 | 方向 | 描述 |
| 1 | clk | I | 时钟信号 |
| 2 | reset | I | 同步复位信号 |
| 3 | flush | I | 清洗信号 |
| 4 | E\_PC[31:0] | I | E级PC |
| 5 | E\_WD[31:0] | I | E级WD |
| 6 | E\_A3[4:0] | I | E级A3 |
| 7 | E\_instr[31:0] | I | E级instr |
| 8 | E\_ALUResult[31:0] | I | E级ALUResult |
| 9 | E\_RD2[31:0] | I | E级RD2 |
| 10 | M\_PC[31:0] | O | M级pc |
| 11 | M\_WD[31:0] | O | M级WD |
| 12 | M\_A3[4:0] | O | M级A3 |
| 13 | M\_instr[31:0] | O | M级instr |
| 14 | M\_ALUResult[31:0] | O | M级ALUResult |
| 15 | M\_RD2[31:0] | O | M级RD2 |

（2）功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能 | 描述 |
| 1 | 存储要流水的值 | 存储要流水的值 |

#### 11. DM

（1）端口说明

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 信号名 | 方向 | 描述 |
| 1 | clk | I | 时钟信号 |
| 2 | reset | I | 同步复位信号 |
| 3 | DMaddr[31:0] | I | 减去0x3000，择取其[13:2]位，作为读取或写入信号的地址 |
| 4 | WD[31:0] | I | 32位要写入的数据 |
| 5 | WE | I | 写使能信号，外界对应MemWrite |
| 6 | RD[31:0] | O | 32位读出的数据 |
| 7 | PC[31:0] | I | 用于display，满足课下需求 |

（2）功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能 | 描述 |
| 1 | 同步复位 | Reset为1且为时钟上升沿时，所有寄存器被清零 |
| 2 | 读数据 | 将A地址对应的数据输出到RD（这个功能始终存在，不管WE是否为1） |
| 3 | 写数据 | WE为1且时钟上升沿到来时，将WD的数据写入到A对应的地址 |

#### 12. W\_pipereg

（1）端口说明

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 信号名 | 方向 | 描述 |
| 1 | clk | I | 时钟信号 |
| 2 | reset | I | 同步复位信号 |
| 3 | M\_PC[31:0] | I | M级PC |
| 4 | M\_instr[31:0] | I | M级instr |
| 5 | M\_A3[4:0] | I | M级A3 |
| 6 | M\_WD[31:0] | I | M级WD |
| 7 | W\_PC[31:0] | O | W级PC |
| 8 | W\_instr[31:0] | O | W级instr |
| 9 | W\_A3[4:0] | O | W级A3 |
| 10 | W\_WD[31:0] | O | W级WD |

（2）功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能 | 描述 |
| 1 | 存储要流水的值 | 存储要流水的值 |

#### 13. Hazard

（1）端口说明

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 信号名 | 方向 | 描述 |
| 1 | clk | I | 时钟信号 |
| 2 | reset | I | 同步复位信号 |
| 3 | D\_A1[4:0] | I | D级读取rs的序号 |
| 4 | D\_A2[4:0] | I | D级读取rt的序号 |
| 5 | D\_RD1[31:0] | I | D级读取rs的值 |
| 6 | D\_RD2[31:0] | I | D级读取rt的值 |
| 7 | D\_rs\_Tuse | I | D级是否在使用rs |
| 8 | D\_rt\_Tuse | I | D级是否在使用rt |
| 9 | E\_A1[4:0] | I | E级读取rs的序号 |
| 10 | E\_A2[4:0] | I | E级读取rt的序号 |
| 11 | E\_RD1[31:0] | I | E级读取rs的值 |
| 12 | E\_RD2[31:0] | I | E级读取rt的值 |
| 13 | E\_A3[4:0] | I | E级写入寄存器的序号 |
| 14 | E\_WD[31:0] | I | E级写入寄存器的值 |
| 15 | E\_rs\_Tuse | I | E级是否使用rs |
| 16 | E\_rt\_Tuse | I | E级是否使用rt |
| 17 | M\_A2[4:0] | I | M级读取rt的序号 |
| 18 | M\_RD2[31:0] | I | M级读取到寄存器的值 |
| 19 | M\_A3[4:0] | I | M级写入寄存器的序号 |
| 20 | M\_WD[31:0] | I | M级写入寄存器的值 |
| 21 | W\_A3[4:0] | I | W级写入寄存器的序号 |
| 22 | W\_WD[31:0] | I | W级写入寄存器的值 |
| 23 | D\_Forward1[31:0] | O | 转发给D级的值1 |
| 24 | D\_Forward2[31:0] | O | 转发给D级的值2 |
| 25 | E\_Forward1[31:0] | O | 转发给E级的值1 |
| 26 | E\_Forward2[31:0] | O | 转发给E级的值2 |
| 27 | M\_Forward2[31:0] | O | 转发给M级的值2 |
| 28 | F\_en | O | PC的使能信号 |
| 29 | D\_pipereg\_en | O | D级流水线寄存器的使能信号 |
| 30 | E\_pipereg\_en | O | E级流水线寄存器的使能信号 |
| 31 | E\_pipereg\_flush | O | E级流水线寄存器的清洗信号 |
| 32 | M\_pipereg\_flush | O | M级流水线寄存器的清洗信号 |

（2）功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能 | 描述 |
| 1 | 产生各转发值 | 产生各转发值 |
| 2 | 产生各阻塞信号 | 产生各阻塞信号 |
| 3 | 产生各清洗信号 | 产生各清洗信号 |

#### 14. Control（分布式译码）

（1）端口说明

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 信号名 | 方向 | 描述 |
| 1 | instr[31:0] | I | 每一级的指令 |
| 2 | D\_equal | I | D级RD1和RD2是否相等 |
| 3 | sign | O | ext是否进行有符号扩展 |
| 4 | PCSrc | O | 是否进行分支 |
| 5 | PCj | O | 是否为J型指令 |
| 6 | PCjr | O | 是否为jr或jalr |
| 7 | D\_WDsel | O | D级中写入GRF值的选择信号 |
| 8 | D\_A3[4:0] | O | D级中写入GRF地址的选择信号 |
| 9 | ALUControl[3:0] | O | ALU选择哪种计算方式 |
| 10 | ALUASel | O | ALU里面inA选择哪个作为输入 |
| 11 | ALUBSel | O | ALU里面inB选择哪个作为输入 |
| 12 | E\_WDSel[1:0] | O | E级中写入GRF值的选择信号 |
| 13 | MemWrite | O | E级中写入GRF地址的选择信号 |
| 14 | width[1:0] | O | 写入DM的位宽 |
| 15 | sign\_l | O | 写入DM的值是否进行有符号扩展 |
| 16 | M\_WDSel | O | M级中写入GRF值的选择信号 |
| 17 | D\_rs\_Tuse | O | D级是否读取rs |
| 18 | D\_rt\_Tuse | O | D级是否读取rt |
| 19 | E\_rs\_Tuse | O | E级是否读取rs |
| 20 | E\_rt\_Tuse | O | E级是否读取rt |

（2）真值表

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 端口 | addu | subu | ori | lw | sw | beq | lui |
| func | 100001 | 100011 | n/a | n/a | n/a | n/a | n/a |
| op | 000000 | 000000 | 001101 | 100011 | 101011 | 000100 | 001111 |
| sign | x | x | 0 | 1 | 1 | x | x |
| Branch | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| MemWrite | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| RegWrite | 1 | 1 | 1 | 1 | 0 | 0 | 1 |
| MemtoReg | 0 | 0 | 0 | 1 | x | x | 0 |
| ALUSrc | 0 | 0 | 1 | 1 | 1 | 0 | 1 |
| RegDst | 1 | 1 | 0 | 0 | x | x | 0 |
| ALUControl | 000 | 001 | 011 | 000 | 000 | 001 | 100 |

### （三）重要机制实现方法

#### 1. 跳转

CMP模块在D级生成跳转信号

#### 2. 流水线延迟槽

保证在b指令和j指令来临时，一定会执行后一条指令，根据判断结果，再进行跳转

#### 3. 转发

在hazard统一生成，再传到各个流水线级中

#### 4. 进入中断处理程序

用en和flush进行终端和清洗

## 二、测试方案

### （1）测试代码

ori $a0,$0,0x100

ori $a1,$a0,0x123

lui $a2,456

lui $a3,0xffff

ori $a3,$a3,0xffff

addu $s0,$a0,$a2

addu $s1,$a0,$a3

addu $s4,$a3,$a3

subu $s2,$a0,$a2

subu $s3,$a0,$a3

sw $a0,0($0)

sw $a1,4($0)

sw $a2,8($0)

sw $a3,12($0)

sw $s0,16($0)

sw $s1,20($0)

sw $s2,24($0)

sw $s3,44($0)

sw $s4,48($0)

lw $a0,0($0)

lw $a1,12($0)

sw $a0,28($0)

sw $a1,32($0)

ori $a0,$0,1

ori $a1,$0,2

ori $a2,$0,1

beq $a0,$a1,loop1

beq $a0,$a2,loop2

loop1: sw $a0,36($t0)

loop2: sw $a1,40($t0)

jal loop3

jal loop3

sw $s5,64($t0)

ori $a1,$a1,4

jal loop4

loop3:sw $a1,56($t0)

sw $ra,60($t0)

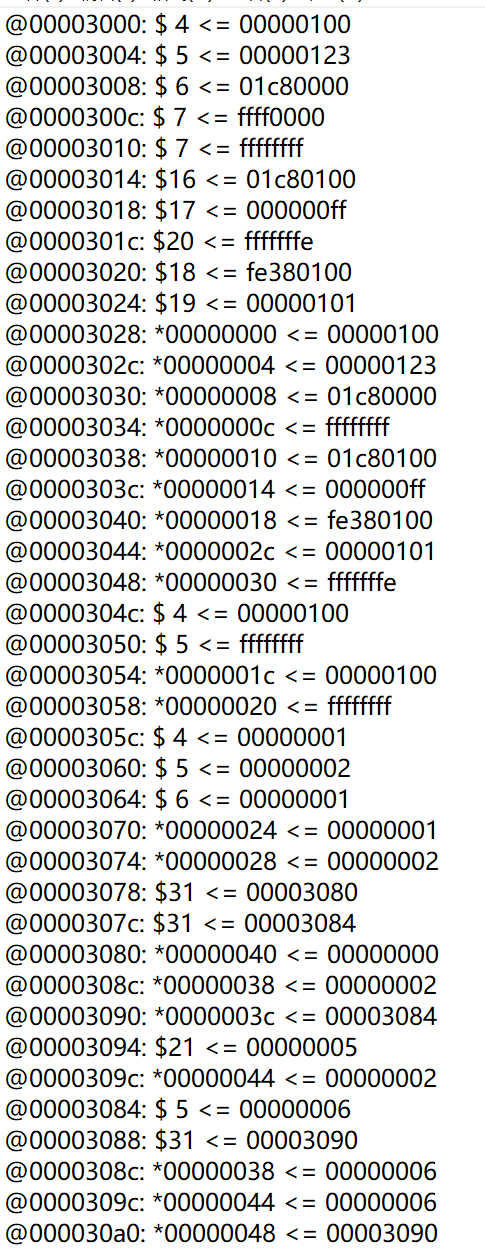
ori $s5,$s5,5

jr $ra

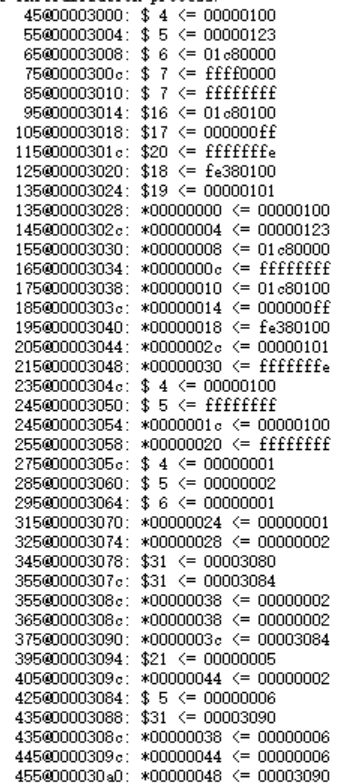
loop4: sw $a1,68($t0)

sw $ra,72($t0)

### （2）MARS中结果



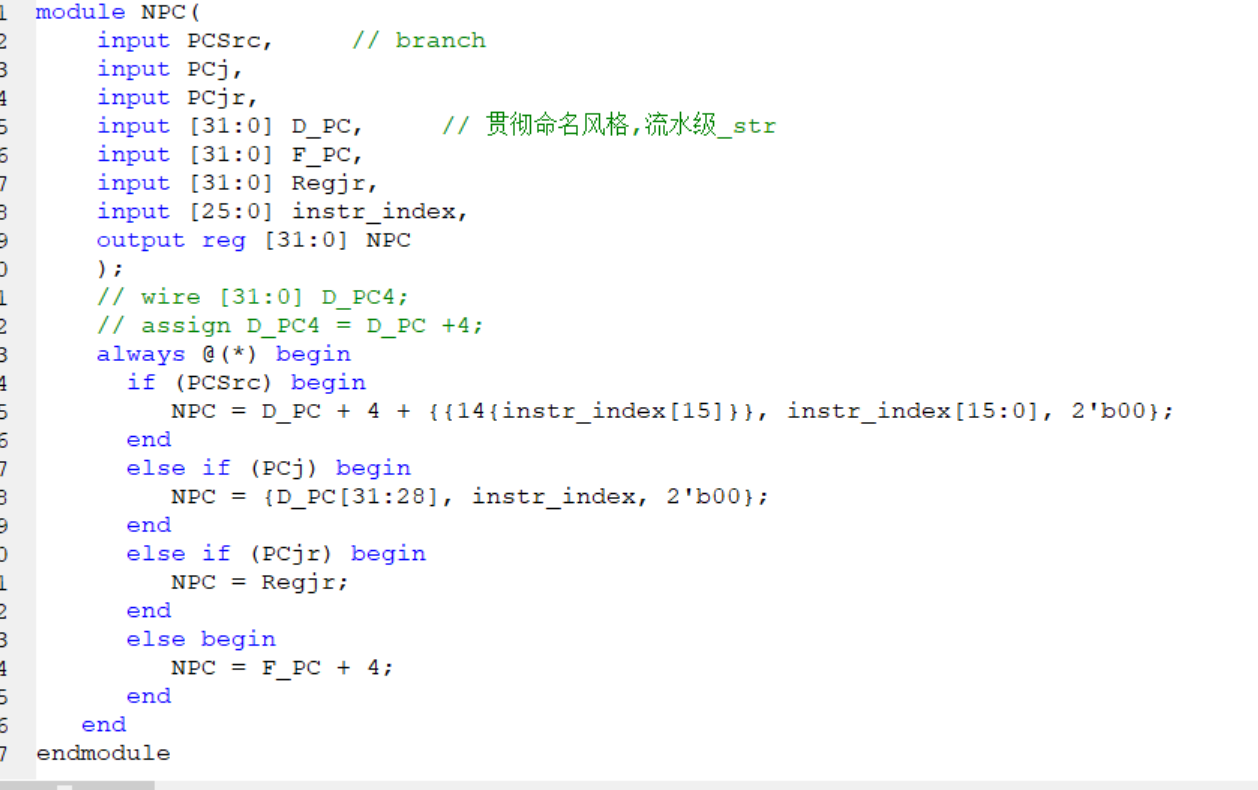
### （3）该CPU中的结果



## 三、思考题

## （一）**流水线冒险**

### 在采用本节所述的控制冒险处理方式下，PC 的值应当如何被更新？请从数据通路和控制信号两方面进行说明。



数据通路：通过NPC产生新的PC，再传入到PC中

控制信号：Control解码出对应的PC

1. 对于 jal 等需要将指令地址写入寄存器的指令，为什么需要回写 PC+8 ？

因为后面有延迟槽。

## （二）**数据冒险的分析**

## 为什么所有的供给者都是存储了上一级传来的各种数据的**流水级寄存器**，而不是由 ALU 或者 DM 等部件来提供数据？

暂停-转发就是要算出来，在流水线寄存器中停一下，然后再转发。这样设计简单，而且是课程要求。

## **AT 法处理流水线数据冒险**

**1.“转发（旁路）机制的构造”中的 Thinking 1-4；**

1. 就是转发的优先级。以jr为例，在D级就要产生写入的NPC数据，此时若E级和M级要求写入同一个寄存器，导致数据冲突，则需要选择E级流水线的数据。
2. 简化流水转发，不使用内部转发就要把每周期GRF的使用分成前后两周期，前周期用于写入，后周期用于读取。
3. 因为不可能向0号寄存器写入东西，0号寄存器保持为0
4. 级数低的流水层产生的优先。

**2.在 AT 方法讨论转发条件的时候，只提到了“供给者需求者的A相同，且不为 0”，但在 CPU 写入 GRF 的时候，是有一个 we 信号来控制是否要写入的。为何在 AT 方法中不需要特判 we 呢？为了用且仅用 A 和 T 完成转发，在翻译出 A 的时候，要结合 we 做什么操作呢？**

如果不需要写寄存器，直接将A译码为为0，这样甚至可以直接省略we。

## （四）**在线测试相关说明**

## 1. 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？

有sw和r型指令的冲突；定义相应的阻塞信号，以解决冲突。

2.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。

不是手动构造的

3.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。

* 画电路图，统一线的命名规范；
* 画暂停转发表，构造暂停条件
* 在电路图中找到要转发的位置，并根据命令情况将要转发的线连到需要转发的位置，以此构造转发命令。
* 使用讨论区的测试数据