设计的功能描述（含所有实现的指令描述，以及单周期/流水线 CPU 频率）

单周期、流水线 CPU 均实现了必做的 24 条指令，实现的指令及其描述见下表：

指令	描述
add	(rd)=(rs1)+(rs2)
sub	(rd)=(rs1)-(rs2)
and	(rd)=(rs1)&(rs2)
or	(rd)=(rs1)|(rs2)
xor	(rd)=(rs1)^(rs2)
sll	(rd)=(rs1)<<(rs2)[4:0]
srl	(rd)=(rs1)>>(rs2)[4:0]
sra	(rd)=signed(rs1)>>>(rs2)[4:0]
addi	(rd)=(rs1)+sext(imm)
andi	(rd)=(rs1)&sext(imm)
ori	(rd)=(rs1)|sext(imm)
xori	(rd)=(rs1)^sext(imm)
slli	(rd)=(rs1)<<sext(imm)[4:0]
srli	(rd)=(rs1)>>sext(imm)[4:0]
srai	(rd)=signed(rs1)>>>sext(imm)[4:0]
lw	(rd)=Mem[(rs1)+sext(offset)]
jalr	(rd)=(pc)+4 (pc)=((rs1)+sext(offset))&~1
sw	Mem[(rs1)+sext(offset)]=(rs2)
beq	if (rs1)==(rs2) (pc)=(pc)+ sext(offset)
bne	if (rs1)!=(rs2) (pc)=(pc)+ sext(offset)
blt	if signed(rs1)<signed(rs2) (pc)=(pc)+ sext(offset)
bge	if signed(rs1)>=signed(rs2) (pc)=(pc)+ sext(offset)
lui	(rd)=imm[31:12]<<12
jal	(rd)=(pc)+4 (pc)=(pc)+ sext(offset)

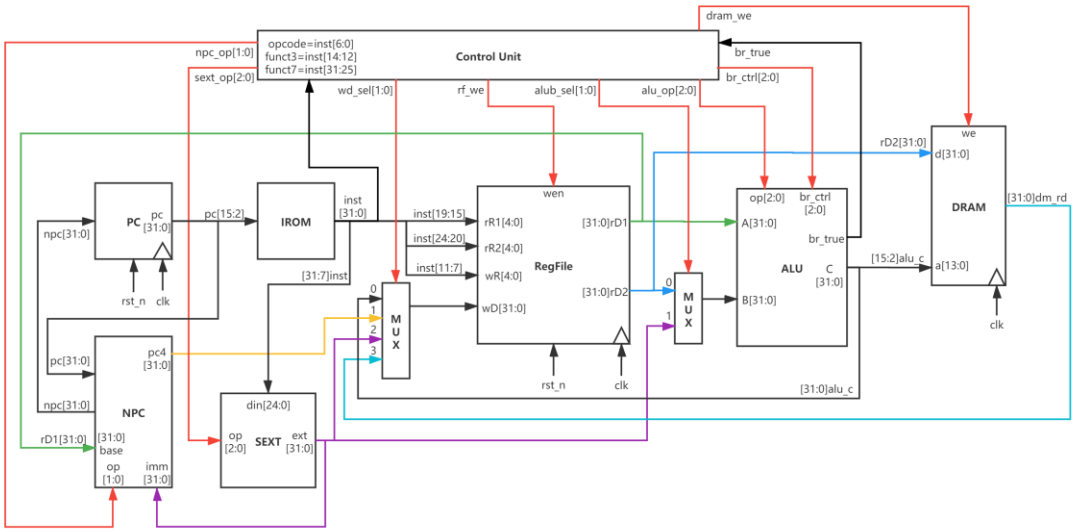
设计的主要特色（除基本要求以外的设计）
资源使用情况、功耗数据截图（实现后）

1 单周期 CPU 设计与实现

1.1 单周期 CPU 整体框图

（要求：无需画出模块内的逻辑，但要标出模块之间信号线的信号名和位宽，以及说明每个模块的功能含义）

整体框图如下图：



模块功能含义：

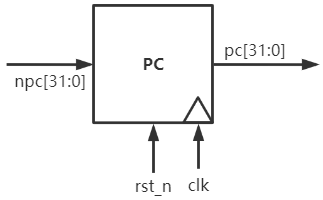
PC：当前指令地址寄存器
NPC：计算下一条指令地址
IROM：指令存储器
DRAM：数据存储器
SEXT：立即数扩展
RegFile：寄存器堆（32 个 32 位寄存器）
ALU：运算单元
MUX：多路选择器
Control Unit：译码控制单元

1.2 单周期 CPU 模块详细设计

（要求：各个模块的详细设计图，要包含内部的子模块，以及关键性逻辑，标出信号名和位宽，并有详细说明）

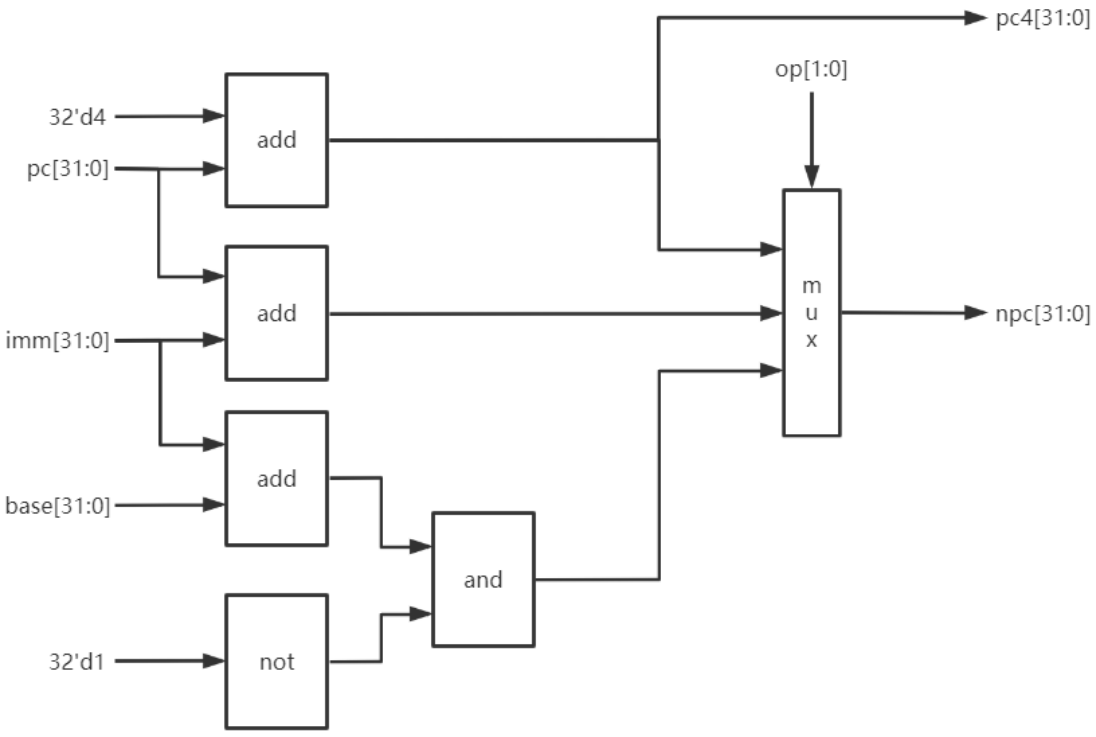
整体：
clk：全局时钟（主频）
rst_n：高电平复位

PC：当前指令地址寄存器（1 个 32 位寄存器）



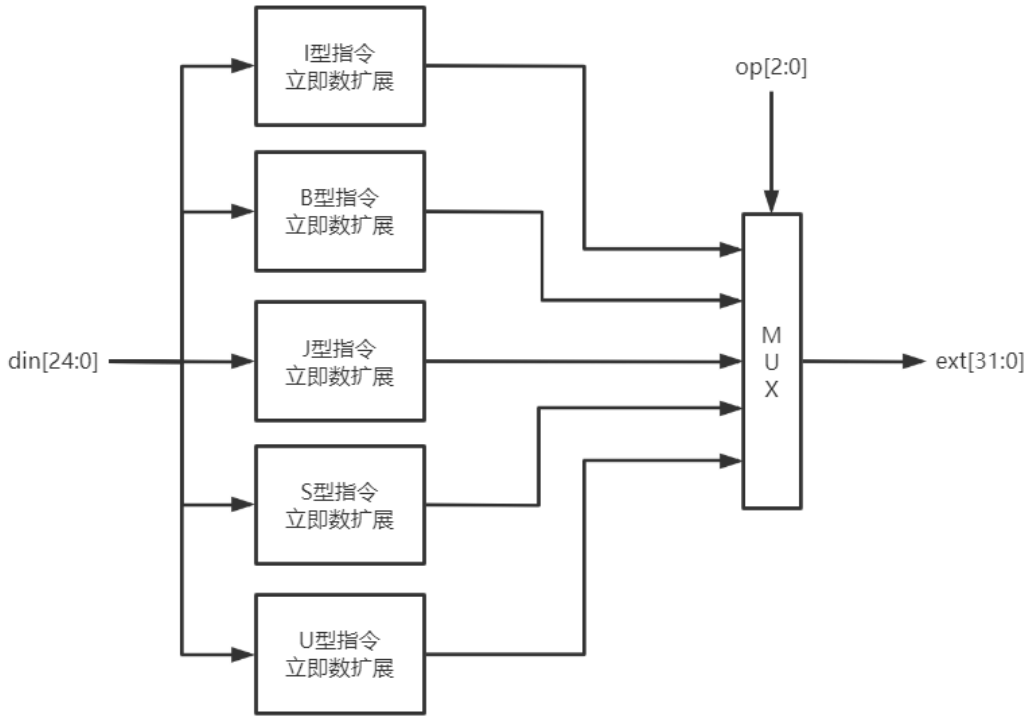
pc[31:0]：当前指令地址
npc[31:0]：下一条指令地址

NPC：计算下一条指令地址



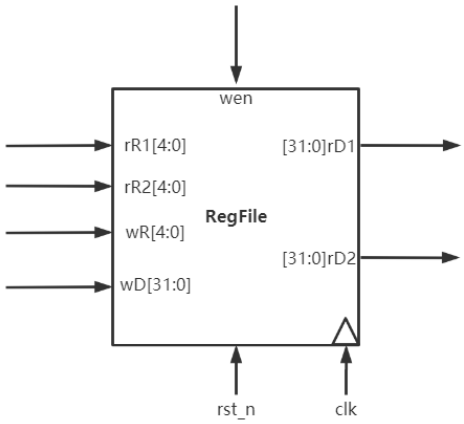
op[1:0]：控制信号
imm[31:0]：扩展后的立即数
base[31:0]：基址
pc4[31:0]：当前指令地址+4

SEXT：立即数扩展



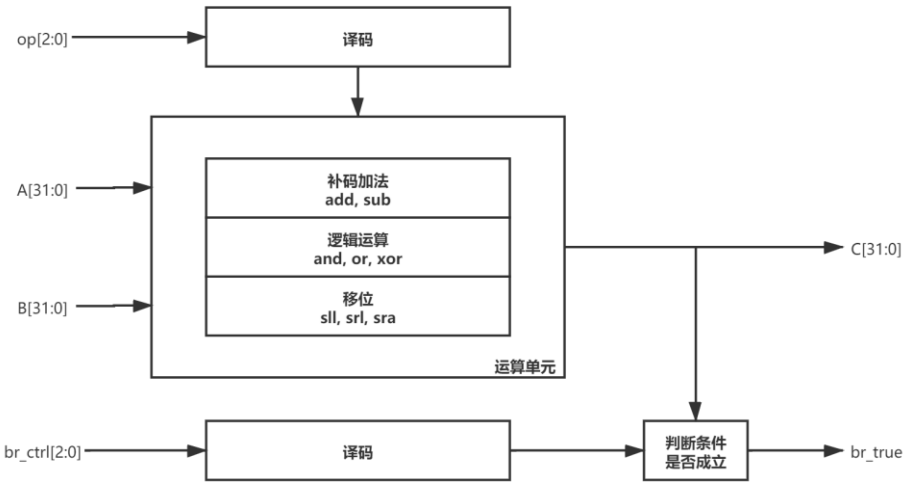
op[2:0]：控制信号
din[24:0]：指令中的立即数段
ext[31:0]：根据指令类型扩展后的立即数

RegFile：寄存器堆（32 个 32 位寄存器）



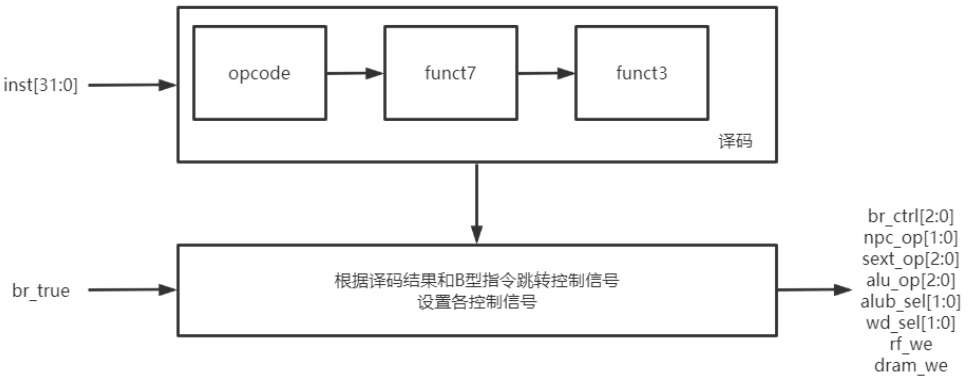
组合逻辑读，时序逻辑写
rR1/rR2[4:0]：源寄存器 1/源寄存器 2 的寄存器号
rD1/rD2[31:0]：源寄存器 1/源寄存器 2 的读出数据
wen：写使能信号
wR[4:0]：目的寄存器的寄存器号
wD[31:0]：目的寄存器的写入数据
第 0 号寄存器的读出数据恒为 0

ALU：运算单元



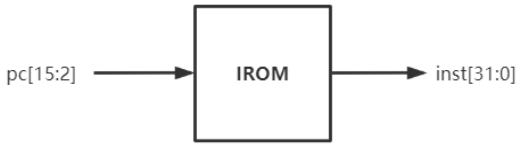
A[31:0]：二元运算符左元
B[31:0]：二元运算符右元
C[31:0]：运算结果
op[2:0]：运算控制信号
br_ctrl[2:0]：比较控制信号（beq、bne、blt、bge）
br_true：B 型指令跳转控制信号

Control Unit：译码控制单元



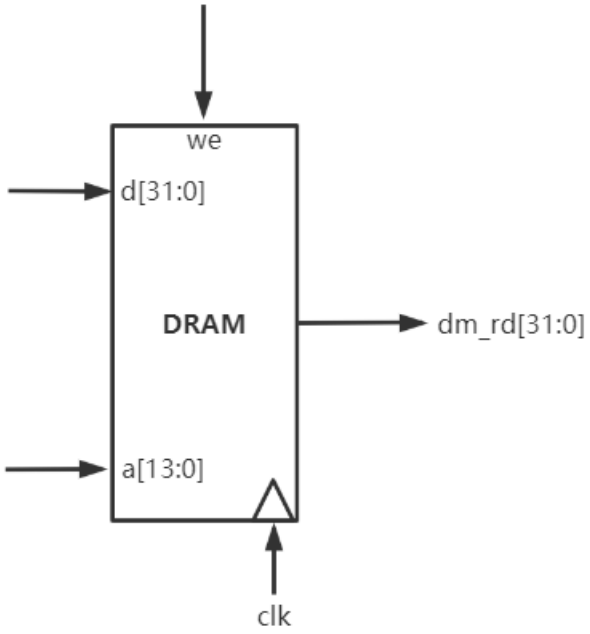
inst[31:0]：指令
br_true：B 型指令跳转控制信号
br_ctrl[2:0]：比较控制信号（beq、bne、blt、bge）
npc_op[1:0]：跳转控制信号（B 型跳转和 jal/jalr/B 型不跳转和其他指令）
sext_op[2:0]：立即数扩展控制信号（I/B/J/S/U 型指令立即数扩展）
alu_op[2:0]：运算控制信号（加减、逻辑、移位运算）
alub_sel[1:0]：ALU 模块输入信号 B 的选择信号
wd_sel[1:0]：目的寄存器写入数据的选择信号
rf_we：寄存器堆写使能信号
dram_we：数据存储器写使能信号

IROM：64KB 指令存储器（IP 核）



pc[15:2]：当前指令地址
inst[31:0]：当前指令

DRAM：64KB 数据存储器（IP 核）

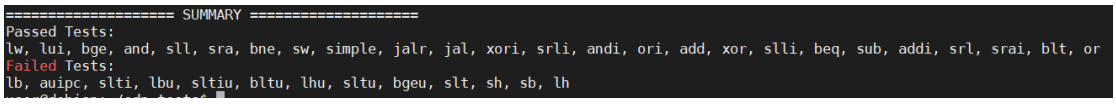


we：写使能信号
a[13:0]：读/写地址
d[31:0]：写数据
dm_rd[31:0]：读数据

1.3 单周期 CPU 仿真及结果分析

（要求：包含逻辑运算指令、访存指令、跳转指令的仿真截图，以及结果分析）

单周期 CPU 的 trace 比对结果如下图：

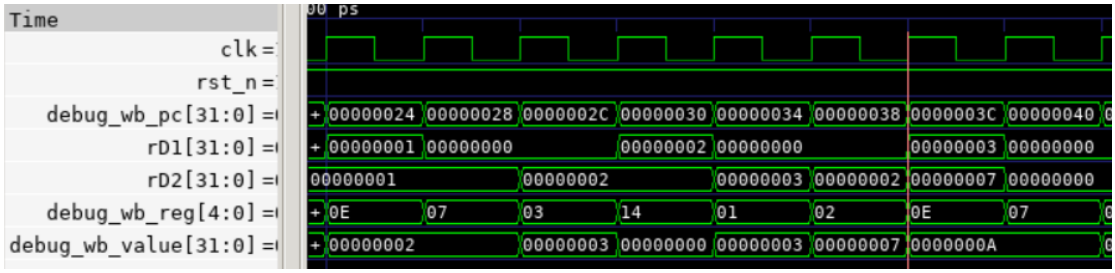


24 条必做指令全部通过

逻辑运算指令反汇编代码及波形截图（仅展示 add、and、sll 波形截图并分析）：

add：

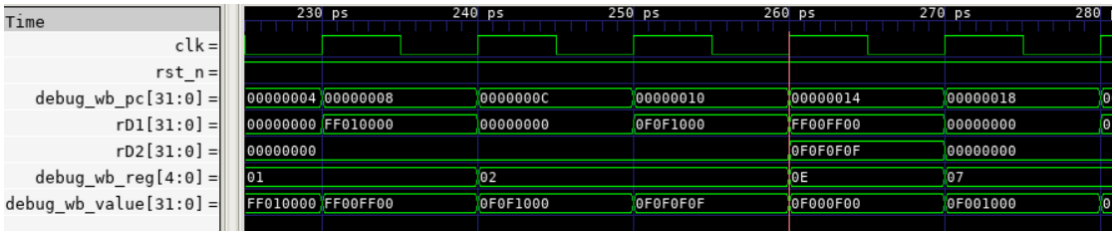
```
34: 00300093            addi  x1,x0,3
38: 00700113            addi  x2,x0,7
3c: 00208733            add x14,x1,x2
```



当 pc=0x3c 时，x1=3，x2=7，写回寄存器 x14=10，结果正确，addi 和 add 指令执行无误

and：

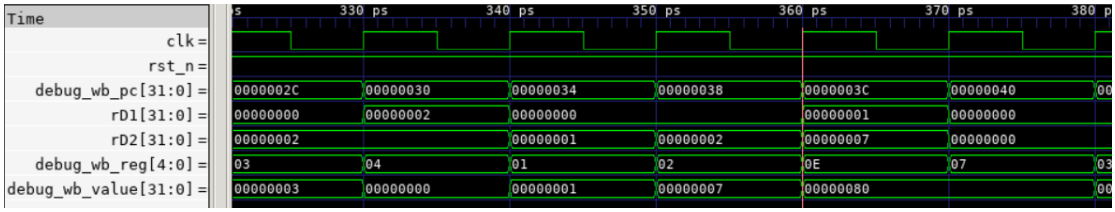
```
  4: ff0100b7           lui x1,0xff010
  8: f0008093           addi  x1,x1,-256 # ff00ff00 <_end+0xff00df00>
  c: 0f0f1137           lui x2,0xf0f1
 10: f0f10113           addi  x2,x2,-241 # f0f0f0f <_end+0xf0eef0f>
 14: 0020f733           and x14,x1,x2
```



当 pc=0x14 时，x1=0xff00ff00，x2=0x0f0f0f0f，写回寄存器 x14=0x0f000f00，结果正确，lui、addi 和 and 指令执行无误

sll：

```
34: 00100093            addi  x1,x0,1
38: 00700113            addi  x2,x0,7
3c: 00209733            sll x14,x1,x2
```

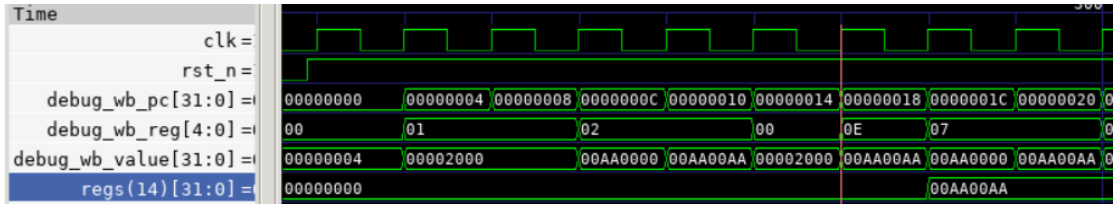


当 pc=0x3c 时，x1=1，x2=7，写回寄存器 x14=0x80，结果正确，addi 和 sll 指令执行无误

访存指令反汇编代码及波形截图（仅展示 sw 波形截图并分析）：

sw：

```
  4: 000020b7           lui x1,0x2
  8: 00008093           addi  x1,x1,0 # 2000 <begin_signature>
  c: 00aa0137           lui x2,0xaa0
 10: 0aa10113           addi  x2,x2,170 # aa00aa <_end+0xa9e07a>
 14: 0020a023           sw  x2,0(x1)
 18: 0000a703           lw  x14,0(x1)
```

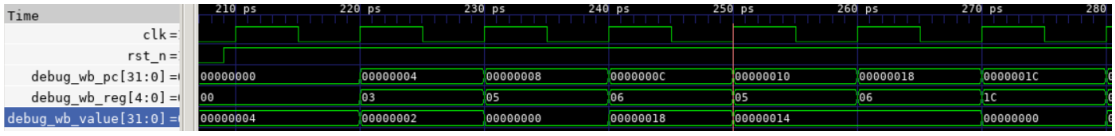


当 pc=0x18 时，写回寄存器 x14=0xaa00aa，结果正确，lui、addi、sw 和 lw 指令执行无误

跳转指令反汇编代码及波形截图（仅展示 jalr、blt 波形截图并分析）：

jalr：

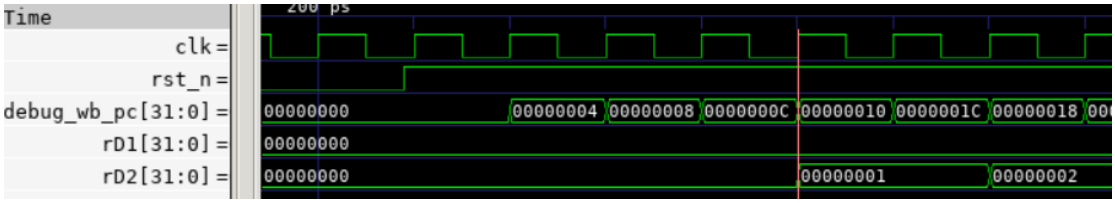
```
00000000 <_start>:
   0: 0040006f            jal x0,4 <reset_vector>

00000004 <reset_vector>:
   4: 00200193            addi  x3,x0,2
   8: 00000293            addi  x5,x0,0
   c: 01800313            addi  x6,x0,24
  10: 000302e7            jalr  x5,0(x6)
```



当 pc=0x10 时，写回寄存器 x5=0x14，pc 跳转到 0x18，结果正确，jal、addi 和 jalr 指令执行无误

blt：

```
  8: 00000093            addi  x1,x0,0
  c: 00100113            addi  x2,x0,1
 10: 0020c663            blt x1,x2,1c <reset_vector+0x18>
```



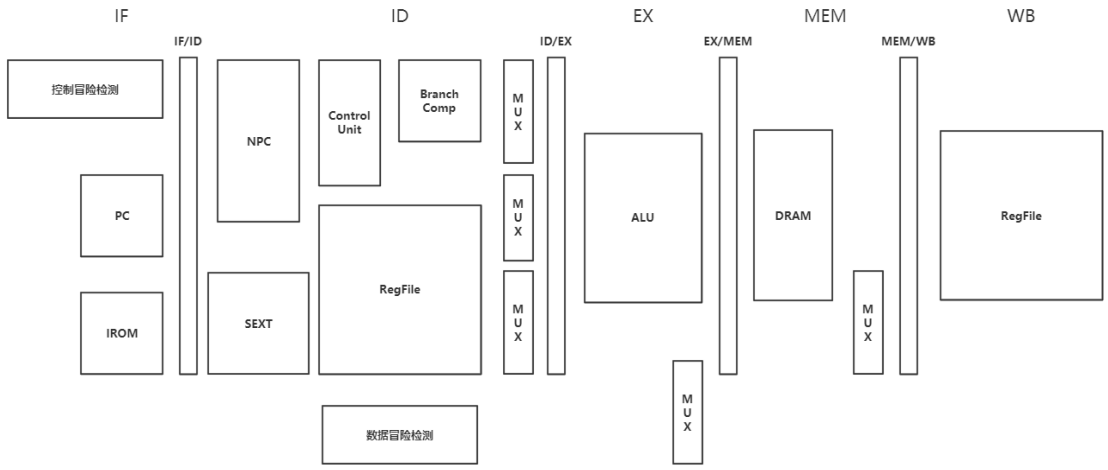
当 pc=0x10 时，x1<x2（0<1），pc 跳转到 0x1c，结果正确，addi 和 blt 指令执行无误

2 流水线 CPU 设计与实现

2.1 流水线的划分

（要求：画出流水线的划分，并标明每个阶段 CPU 完成的功能）

划分为 IF-ID-EX-MEM-WB 五个流水级，如下图所示：

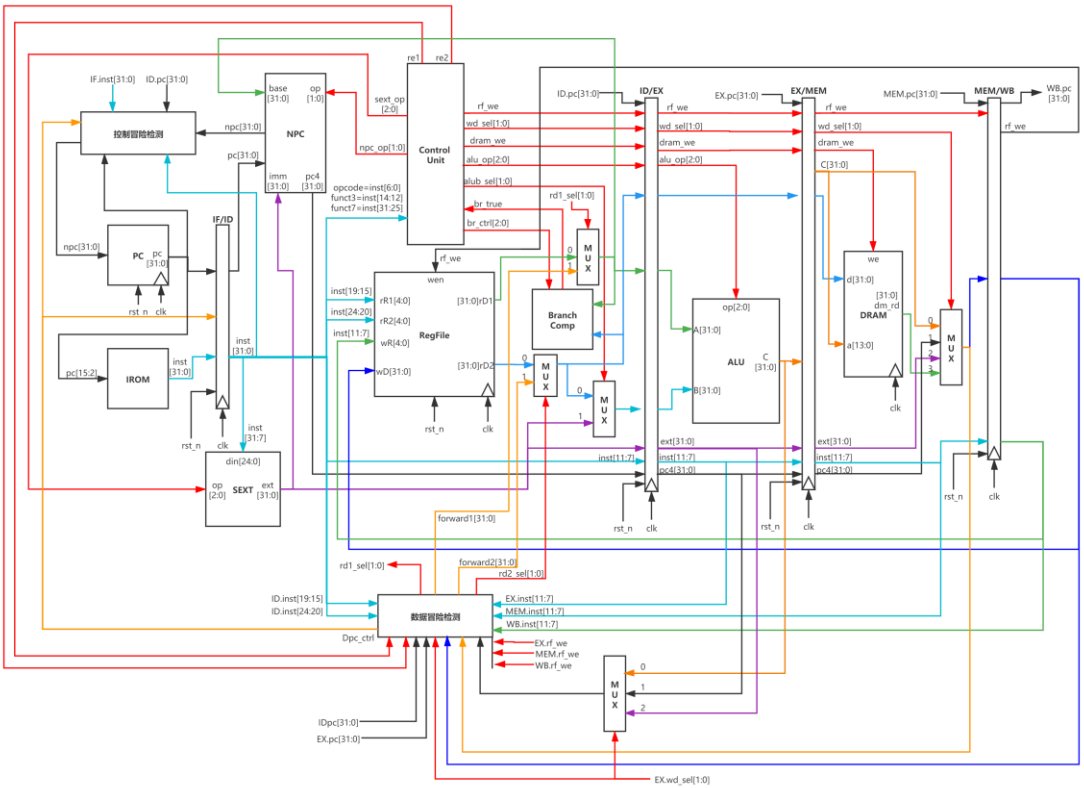


取指（IF）阶段：根据当前指令地址访问指令存储器，取出指令
译码（ID）阶段：译码控制、立即数扩展、读寄存器堆、比较、计算跳转地址
执行（EX）阶段：加减、逻辑、移位运算
访存（MEM）阶段：读/写数据存储器
写回（WB）阶段：写寄存器堆

2.2 流水线 CPU 整体框图

（要求：无需画出模块内的逻辑，但要标出模块之间信号线的信号名和位宽，以及说明每个模块的功能含义）

整体框图：



模块功能含义：

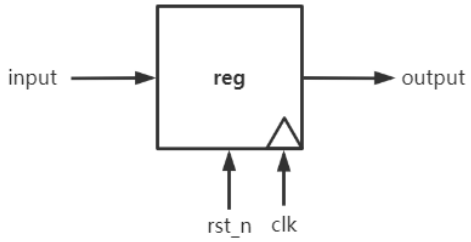
IF/ID、ID/EX、EX/MEM、MEM/WB：流水寄存器
PC：当前指令地址寄存器
IROM：指令存储器
控制冒险检测：检测控制冒险，控制 PC 寄存器更新（停顿、+4、跳转）
Control Unit：译码控制单元
NPC：计算跳转地址
Branch Comp：比较，B 型指令跳转判断
SEXT：立即数扩展
RegFile：寄存器堆（32 个 32 位寄存器）
数据冒险检测：检测数据冒险，发送停顿信号或进行数据前递
ALU：运算单元
DRAM：数据存储器
MUX：多路选择器

2.3 流水线 CPU 模块详细设计

（要求：各个模块的详细设计图，要包含内部的子模块，以及关键性逻辑，标出信号名和位宽，并有详细说明；数据冒险与控制冒险的解决方法必须要详细说明）

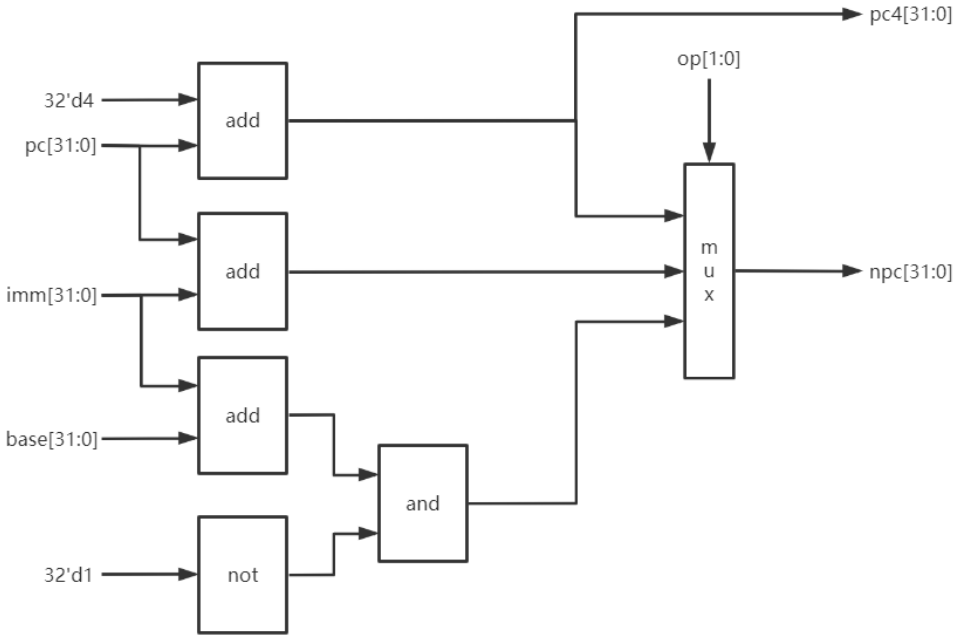
整体：
clk：全局时钟（主频）
rst_n：高电平复位

流水寄存器、PC 寄存器：



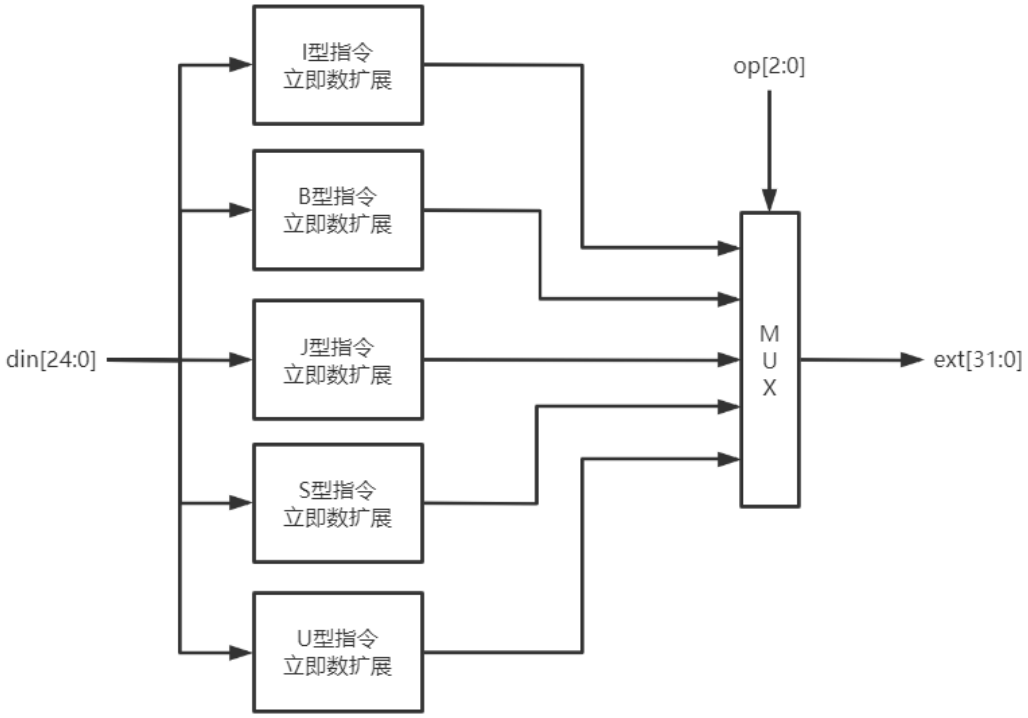
其中，不同流水级间的流水寄存器其输入输出取决于后面的流水级需要用到前一流水级中的哪些信号，图中的输入输出信号名和位宽略（见整体框图）

NPC：计算跳转地址



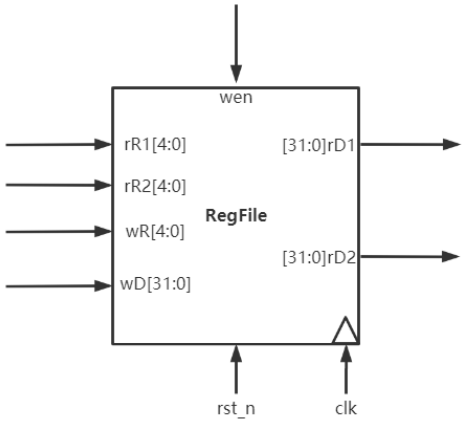
op[1:0]：控制信号
imm[31:0]：扩展后的立即数
base[31:0]：基址
pc4[31:0]：当前指令地址+4
pc[31:0]：当前指令地址
npc[31:0]：跳转地址

SEXT：立即数扩展



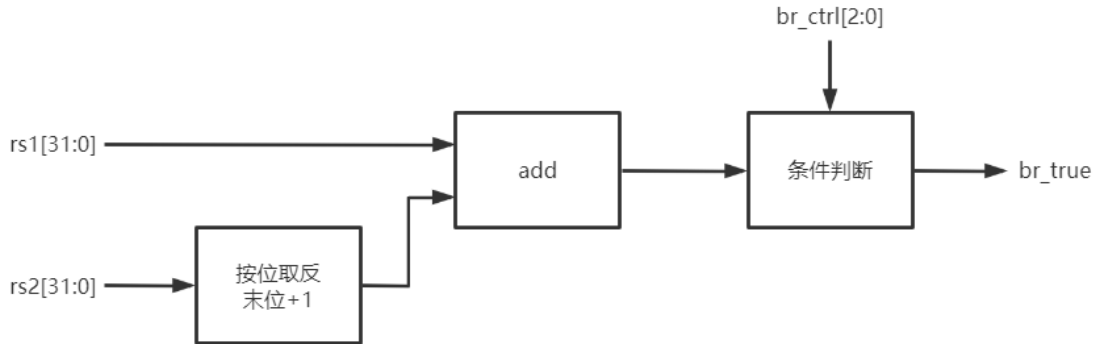
op[2:0]：控制信号
din[24:0]：指令中的立即数段
ext[31:0]：根据指令类型扩展后的立即数

RegFile：寄存器堆（32 个 32 位寄存器）



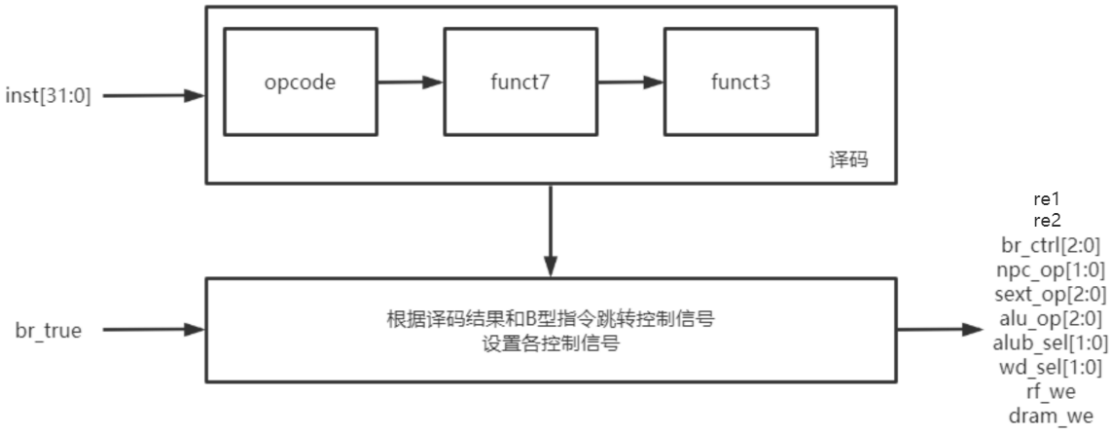
组合逻辑读，时序逻辑写
rR1/rR2[4:0]：源寄存器 1/源寄存器 2 的寄存器号
rD1/rD2[31:0]：源寄存器 1/源寄存器 2 的读出数据
wen：写使能信号
wR[4:0]：目的寄存器的寄存器号
wD[31:0]：目的寄存器的写入数据
第 0 号寄存器的读出数据恒为 0

Branch Comp：比较，B 型指令跳转判断



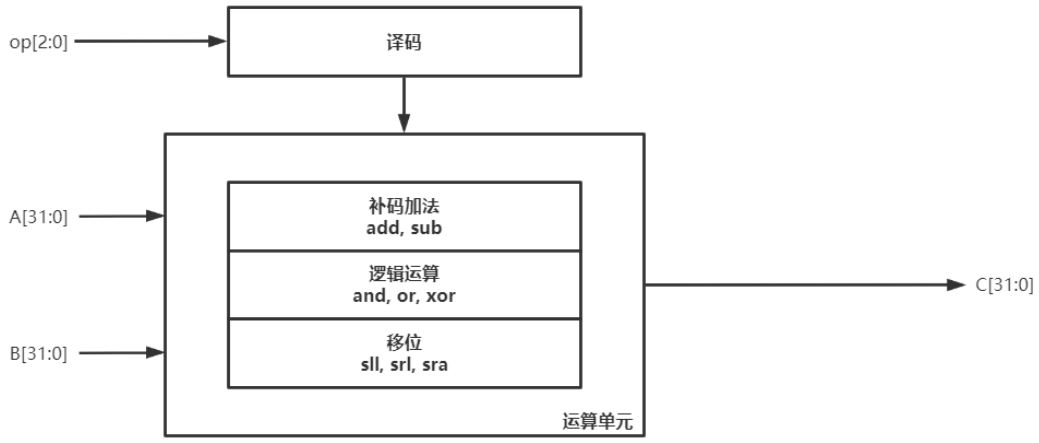
rs1/rs2[31:0]：源寄存器 1/源寄存器 2 数据（数据前递后）
br_ctrl[2:0]：比较控制信号（beq、bne、blt、bge）
br_true：B 型指令跳转控制信号

Control Unit：译码控制单元



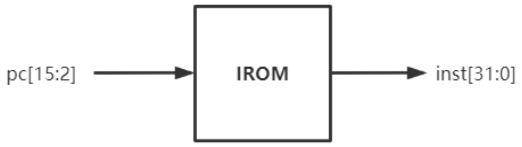
inst[31:0]：指令
br_true：B 型指令跳转控制信号
br_ctrl[2:0]：比较控制信号（beq、bne、blt、bge）
npc_op[1:0]：跳转控制信号（B 型跳转和 jal/jalr/B 型不跳转和其他指令）
sext_op[2:0]：立即数扩展控制信号（I/B/J/S/U 型指令立即数扩展）
alu_op[2:0]：运算控制信号（加减、逻辑、移位运算）
alub_sel[1:0]：ALU 模块输入信号 B 的选择信号
wd_sel[1:0]：目的寄存器写入数据的选择信号
rf_we：寄存器堆写使能信号
dram_we：数据存储器写使能信号
re1：指令读寄存器堆源寄存器 1 标志信号
re2：指令读寄存器堆源寄存器 2 标志信号

ALU：运算单元



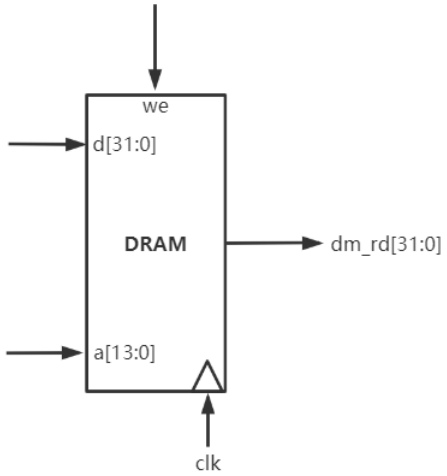
A[31:0]：二元运算符左元
B[31:0]：二元运算符右元
C[31:0]：运算结果
op[2:0]：运算控制信号

IROM：64KB 指令存储器（IP 核）



pc[15:2]：当前指令地址
inst[31:0]：当前指令

DRAM：64KB 数据存储器（IP 核）



we：写使能信号
a[13:0]：读/写地址
d[31:0]：写数据
dm_rd[31:0]：读数据

数据冒险检测和控制冒险检测输入输出信号名与位宽见整体框图，以下是两种冒险检测和解决方法的说明：

数据冒险检测：检测数据冒险，发送停顿信号或进行数据前递

数据前递：当 ID 阶段的指令需要用到 EX/MEM/WB 阶段的待写回数据时，将待写回数据前递到 ID 阶段使用。

停顿（load-use 冒险）：当 ID 阶段的指令需要用到 EX 阶段的待写回数据，但 EX 阶段是 lw 指令，需要其执行到 MEM 阶段才能准备好待写回数据，因此停顿一个周期。停顿方法是向控制冒险检测模块和 IF/ID 流水寄存器发送停顿信号，使 IF/ID 流水寄存器不更新，而控制冒险检测模块使 PC 寄存器不更新（此时 IF 阶段的指令 PC 是 ID 阶段的 PC+4）。停顿前后流水级指令执行情况如下表（假设发生冒险的 lw 指令 PC 为 PC）：

流水级	停顿前指令 PC	停顿后指令 PC
IF	PC+8	PC+8
ID	PC+4	PC+4
EX	PC	PC+4
MEM	PC-4	PC
WB	PC-8	PC-4

停顿一个周期后，MEM 阶段 lw 指令准备好待写回数据，将数据前递到 ID 阶段解决数据冒险，PC 寄存器和流水寄存器恢复更新。

综上，在数据冒险检测模块中，考虑了 EX/MEM/WB 冒险判断优先级、三个阶段指令的目的寄存器号和 ID 阶段指令的源寄存器号、读源寄存器 1/2 的标志信号、三个阶段的寄存器堆写使能信号、EX 阶段是否为 lw 指令、ID 和 EX 阶段的指令 PC，并首先排除 ID 阶段指令源寄存器号为 0 的情况。

控制冒险检测：检测控制冒险，控制 PC 寄存器更新（停顿、+4、跳转）

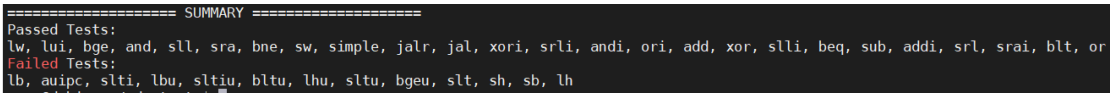
检测到 IF 阶段指令为 jal/jalr/B 型指令，停顿一个周期，但该停顿优先级低于数据冒险发送的停顿信号，而且该停顿仅使 PC 寄存器不更新。
停顿一个周期后，ID 阶段 NPC 模块计算出应该跳转往的指令地址，并送回该模块，再由该模块控制 PC 寄存器更新。
在信号判断上解除这种停顿的方法是考虑 IF 和 ID 阶段的指令 PC、ID 阶段是否为跳转指令。
无停顿和跳转时控制 PC 寄存器加 4。控制冒险解决情况如下表：

流水级	停顿前指令 PC	停顿后指令 PC	跳转后指令 PC
IF	PC	PC	NPC
ID	PC-4	PC	PC
EX	PC-8	PC-4	PC
MEM	PC-12	PC-8	PC-4
WB	PC-16	PC-12	PC-8

2.4 流水线 CPU 仿真及结果分析

（要求：包含数据冒险、控制冒险的仿真截图，以及结果分析）

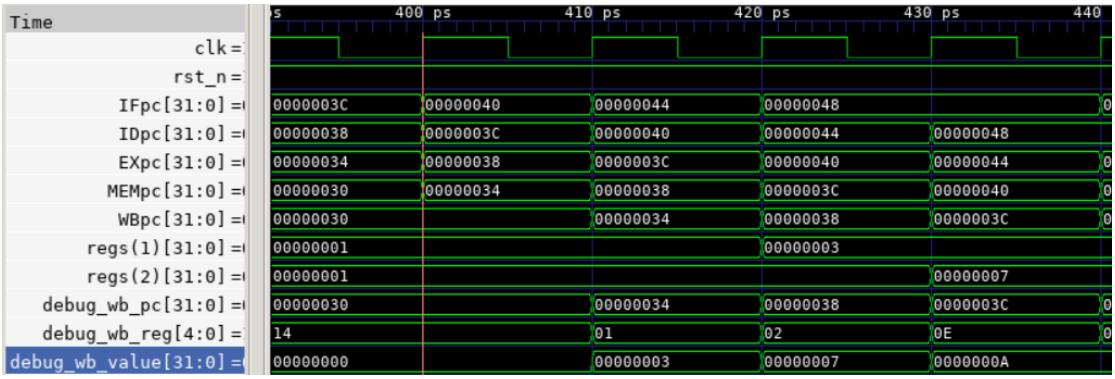
流水线 CPU 的 trace 比对结果如下图：



必做 24 条指令全部通过

包含数据冒险的反汇编代码和波形截图（仅展示 EX、MEM 冒险的数据前递并进行结果分析）：

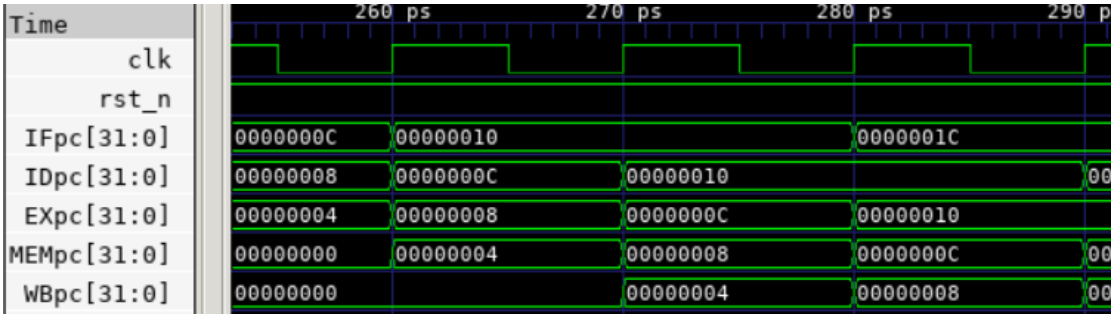
```
34: 00300093            addi  x1,x0,3
38: 00700113            addi  x2,x0,7
3c: 00208733            add x14,x1,x2
40: 00a00393            addi  x7,x0,10
44: 00400193            addi  x3,x0,4
48: 48771e63            bne x14,x7,4e4 <fail>
```



当 IDpc=0x3c 时，x1=1，x2=1，如果没有进行数据前递，最后写回 x14=2，但实际上当 WBpc=0x3c 时，写回 x14=10，即检测到数据冒险且通过数据前递解决（图中的 pc=0x48 持续了两个周期，是因为这是 bne 指令，属于控制冒险，停顿一个周期，pc=0x30 同理）

包含控制冒险的反汇编代码和波形截图（仅展示 blt 指令并分析）：

```
 4: 00200193            addi  x3,x0,2
 8: 00000093            addi  x1,x0,0
 c: 00100113            addi  x2,x0,1
10: 0020c663            blt x1,x2,1c <reset_vector+0x18>
```



当 IFpc=0x10 时，为 blt 指令，停顿一个周期，在 ID 阶段完成比较和跳转地址计算，0<1 成立，跳转到 0x1c，即下一个 IFpc=0x1c，波形符合，故检测到控制冒险且通过停顿一个周期解决

3 设计过程中遇到的问题及解决方法

（包括设计过程中的错误及测试过程中遇到的问题）

4 总结

（要求：个人收获以及对课程的建议）