哈爾濱工業大學（深圳）
HARBIN INSTITUTE OF TECHNOLOGY

实验报告

开课学期：2021 夏季

课程名称：计算机设计与实践

实验名称：CPU 设计

实验性质：综合设计型

实验学时：52 地点：T2506

学生班级：19 级 8 班

学生学号：190110824

学生姓名：梁天翼

评阅教师：

报告成绩：

实验与创新实践教育中心制

2021 年 5 月

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）

R 型指令

add

指令名：加法

功能描述：把寄存器 rs1 加上寄存器 rs2，结果写入寄存器 rd。忽略算术溢出。

sub

指令名：减法

功能描述：从寄存器 rs1 减去寄存器 rs2，结果写入寄存器 rd。忽略算术溢出。

and

指令名：逻辑与

功能描述：将寄存器 rs1 和寄存器 rs2 进行按位与，结果写入寄存器 rd。

or

指令名：逻辑或

功能描述：将寄存器 rs1 和寄存器 rs2 进行按位或，结果写入寄存器 rd。

xor

指令名：逻辑异或

功能描述：将寄存器 rs1 和寄存器 rs2 进行按位异或，结果写入寄存器 rd。

sll

指令名: 逻辑左移

功能描述：把寄存器 rs1 左移 rs2 位，低位补 0，结果写入寄存器 rd。rs2[4:0] 表示左移位数，高位忽略。

srl

指令名: 逻辑右移

功能描述：把寄存器 rs1 右移 rs2 位，高位补 0，结果写入寄存器 rd。rs2[4:0]表示右移位数，高位忽略。

sra

指令名: 算术右移

功能描述:把寄存器 rs1 右移 rs2 位,高位补 rs1[31],结果写入寄存器 rd。rs2[4:0]表示右移位数，高位忽略。

I 型指令

addi

指令名：立即数加法

功能描述：把符号扩展的立即数加到寄存器 rs1 上，结果写入寄存器 rd。忽略算术溢出。

andi

指令名：立即数逻辑与

功能描述：把符号扩展的立即数与寄存器 rs1 的值进行按位与，结果写入寄存器 rd。

ori

指令名：立即数逻辑或

功能描述：把符号扩展的立即数与寄存器 rs1 的值进行按位或，结果写入寄存器 rd。

xori

指令名：立即数逻辑异或

功能描述：把符号扩展的立即数与寄存器 rs1 的值进行按位异或，结果写入寄存器 rd。

slli

指令名：立即数逻辑左移

功能描述：把寄存器 rs1 左移 shamt 位，低位补 0，结果写入寄存器 rd。

srli

指令名：立即数逻辑右移

功能描述：把寄存器 rs1 右移 shamt 位，高位补 0，结果写入寄存器 rd。

srai

指令名：立即数算术右移

功能描述：把寄存器 rs1 右移 shamt 位，高位补 rs1[31]，结果写入寄存器 rd。

lw

指令名：字加载

功能描述:从地址(rs1)+sext(offset)读取四个字节,经符号扩展后写入寄存器 rd。

jalr

指令名：跳转并寄存器链接

功能描述：将 PC 设置为(rs1)+sext(offset)，把计算出的地址的最低位设为 0，并将原 PC+4 的值写入寄存器 rd。rd 默认为 1。

S 型指令

sw

指令名：存字

功能描述：将寄存器 rs2 存入内存地址(rs1)+sext(offset)。

B 型指令

beq

指令名：相等时分支

功能描述：若寄存器 rs1 和寄存器 rs2 的值相等，把 PC 的值设置为当前值加上符号扩展的偏移量。

bne

指令名：不相等时分支

功能描述：若寄存器 rs1 和寄存器 rs2 的值不相等，把 PC 的值设置为当前值加上符号扩展的偏移量。

blt

指令名：小于时分支

功能描述：若寄存器 rs1 的值小于寄存器 rs2 的值，把 PC 的值设置为当前值加上符号扩展的偏移量。

bge

指令名：大于等于时分支

功能描述：若寄存器 rs1 的值大于等于寄存器 rs2 的值，把 PC 的值设置为当前值加上符号扩展的偏移量。

U 型指令

Lui

指令名：高位立即数加载

功能描述：将符号扩展的 20 位立即数逻辑左移 12 位，结果写入寄存器 rd。

J 型指令

jal

指令名：跳转并链接

功能描述：把下一条指令的地址保存到寄存器 rd，然后把 PC 设置为当前值加上符号扩展的立即数。rd 默认为 1。

单周期 CPU 频率：25mHz

流水线 CPU 未上板，不涉及频率

设计的主要特色（除基本要求以外的设计）

为 CPU 添加外设拨码开关和 24 个 LED 灯，实现流水灯的功能，循环执行，每隔大约半秒变换一次 LED 灯。可设置如下变换模式：

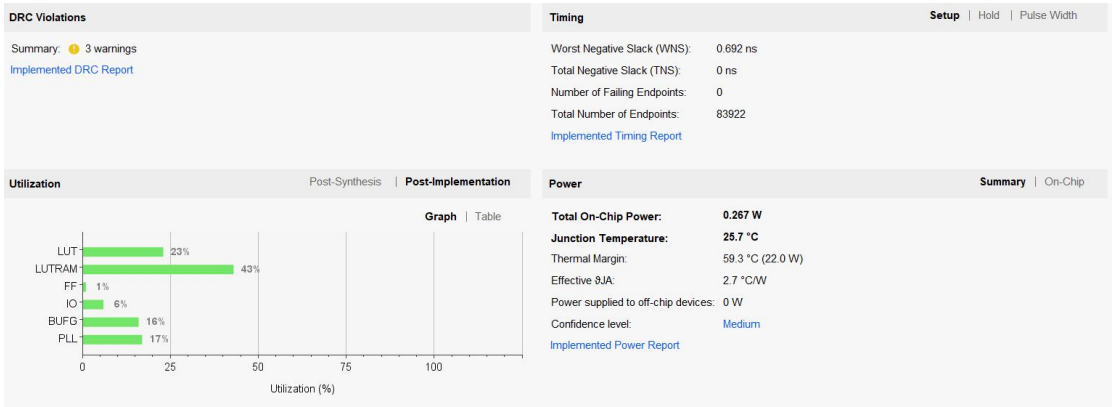
模式 1：LED 灯从两边向中间依次点亮，再从中间向两边依次熄灭。

模式 2：从左向右依次亮 1、2、3...24 盏 LED 灯，至 24 盏 LED 灯全亮后，从左向右灭 1、2、3...24 盏 LED 灯，至所有灯全灭。

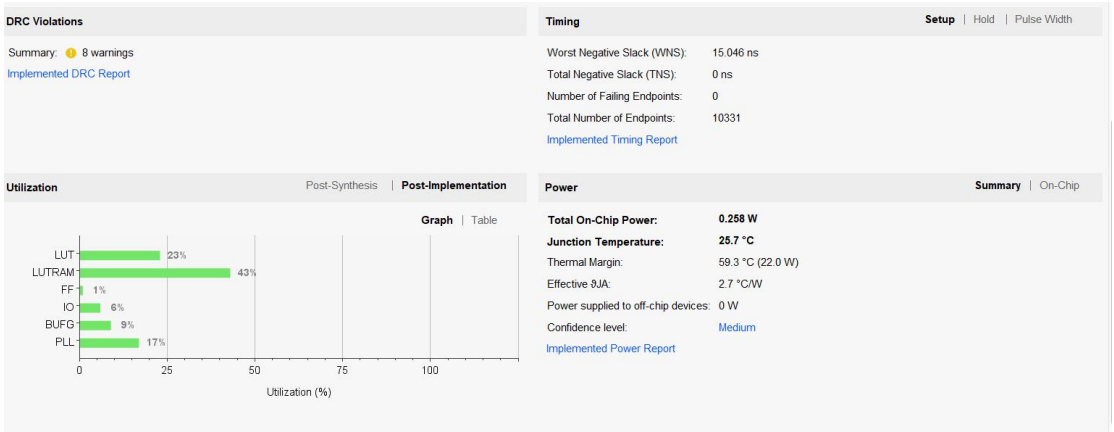
模式 3：由拨码开关控制 LED 灯点亮连续 X 位（X 由拨码开关输入），并循环右移。

资源使用情况、功耗数据截图（实现后）

单周期



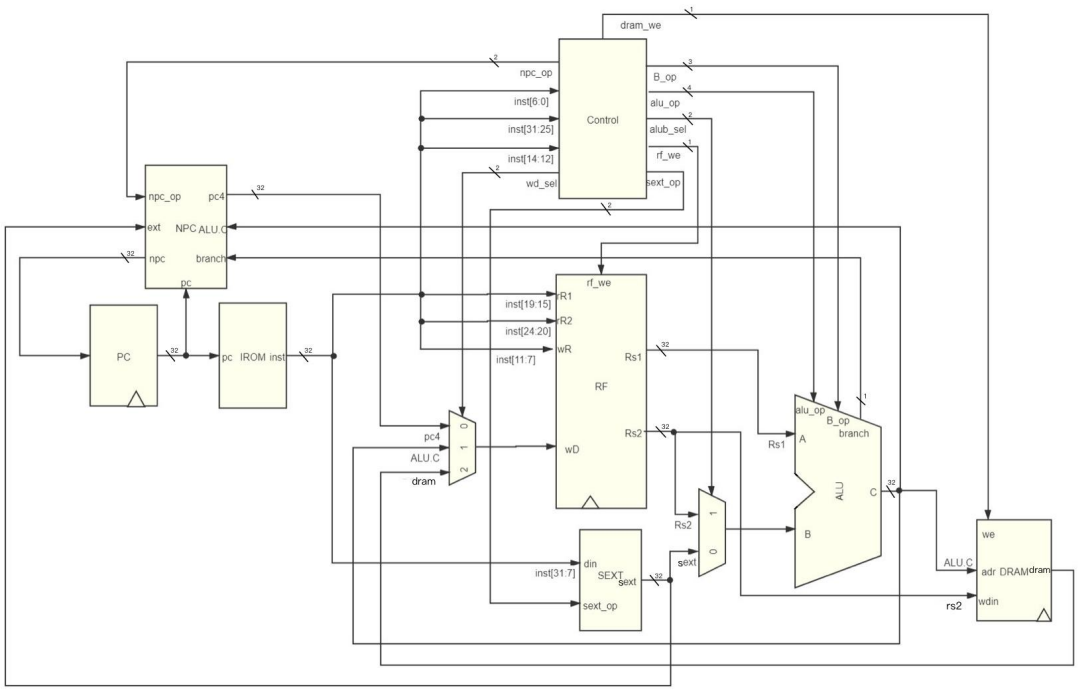
流水线



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）

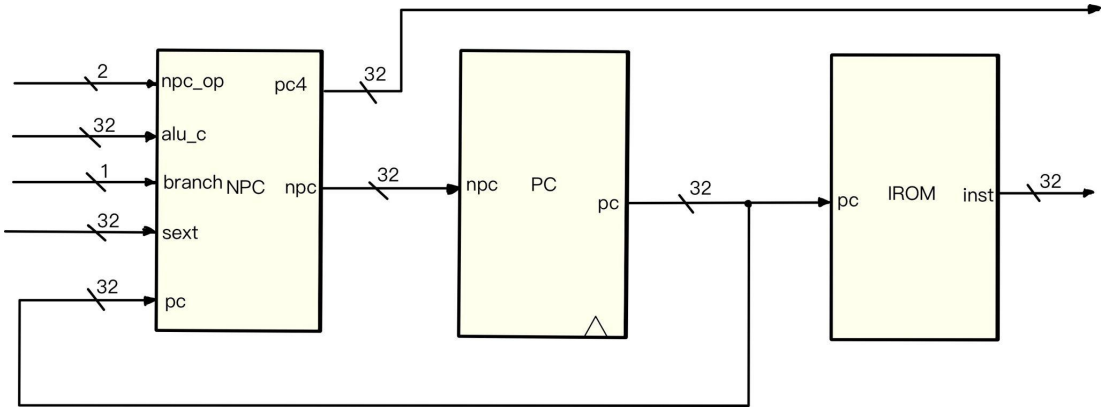


模块功能含义：

PC：地址单元，将地址取出传递给 IROM
NPC:根据指令结果，控制下一条指令地址，将地址传递给 PC 进行相应操作
IROM:指令存储器，参与译码
RF：寄存器堆，包括运算寄存器，目的寄存器等，用来保存数据和数据间的传递
SEXT:立即数生成模块，对立即数进行符号扩展
ALU：运算器，根据控制信号执行相应运算操作
DRAM：数据存储器
Control：控制模块，统筹各模块间协调运作关系

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）



取指模块：

NPC 根据控制信号 npc_op 确定下一步地址，分为多种情况：

① 直接加 4，传给 npc

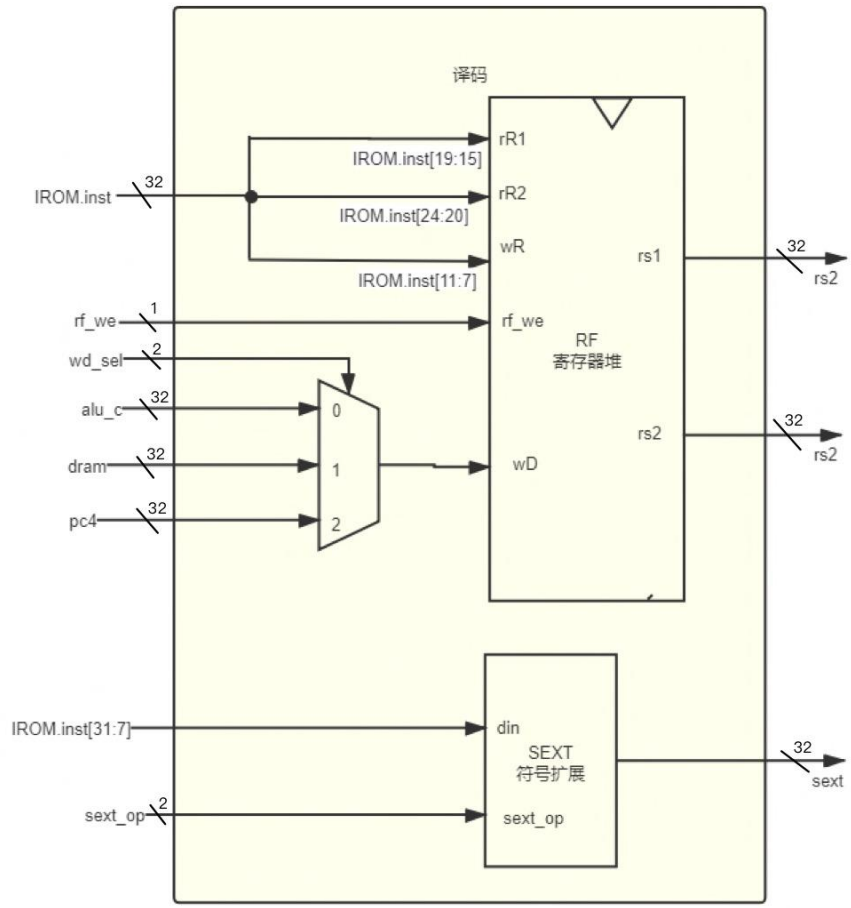
② 根据分支跳转控制信号 branch 确定下一步是否发生分支跳转

③ 直接跳转至当前 pc 加立即数 sext

④ 根据 alu 计算结果执行 jalr 指令跳转地址

同时再将 pc+4 的结果 pc4 传给 wd

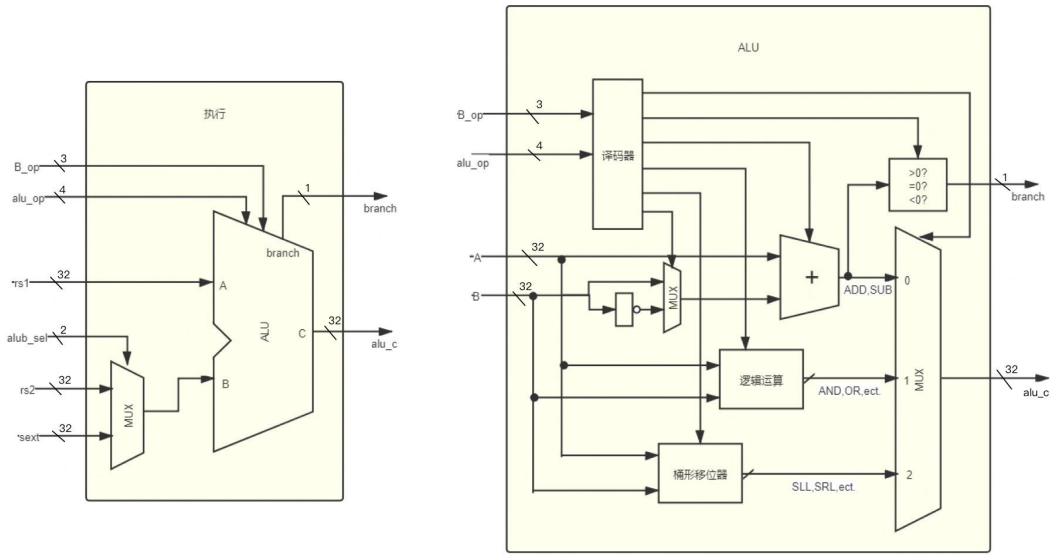
PC 接受到 NPC 给的地址后传给 IROM 取出指令，传给后续进行下一步操作



译码模块与写回

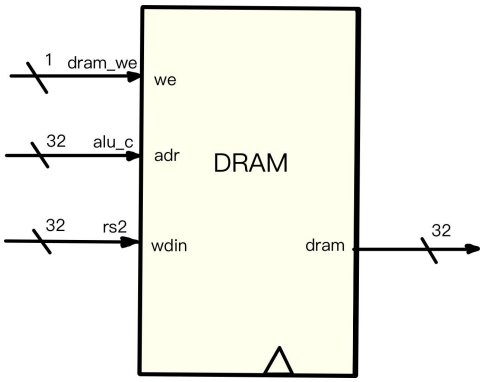
根据取址模块传递的指令写入运算寄存器和目的寄存器，将接受到的 rs1，rs2 传递给下一级。再根据 rf_we 确定是否后续需要写回，由 wd_sel 确定写回寄存器的结果是 pc4/alu_c/dram

SEXT 根据控制信号 sext_op 对指令中的立即数进行符号扩展，并向后传递



执行模块

执行模块接收到来自译码模块传来的寄存器值，根据 alub_sel 选择信号确定传出 alu 中是寄存器值还是立即数，再根据控制单元中 alu_op 操作选择信号进行译码，分为加法操作，逻辑运算操作，位移操作。其中分支操作信号 B_op 决定是哪一种 B 型指令通过加减运算器比较大小，传给 branch 决定是否跳转。最终结果通过 alu_c 和 branch 传给下一级。



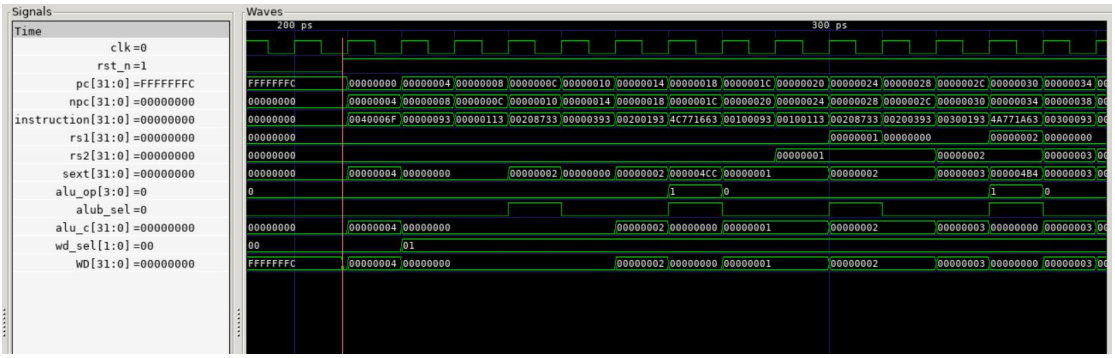
访存模块

数据寄存器接收控制模块的写使能信号 dram_we 判断是否进行写操作，wdin 接收到写入值。adr 接受来自 alu 计算出的地址 alu_c,再将结果通过 dram 信号写回。

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）

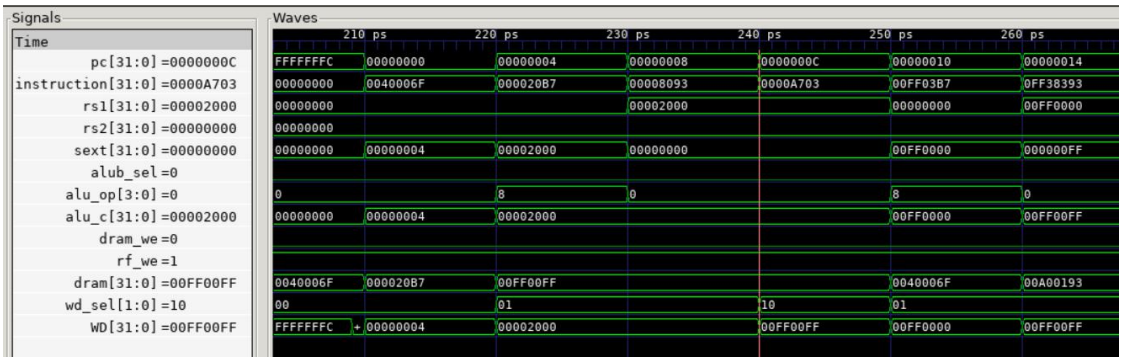
逻辑运算




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

00000004 <reset_vector>:
   4: 00000093            addi  x1,x0,0
   8: 00000113            addi  x2,x0,0
   c: 00208733            add x14,x1,x2
  10: 00000393            addi  x7,x0,0
  14: 00200193            addi  x3,x0,2
  18: 4c771663            bne x14,x7,4e4 <fail>

0000001c <test_3>:
  1c: 00100093            addi  x1,x0,1
  20: 00100113            addi  x2,x0,1
  24: 00208733            add x14,x1,x2
  28: 00200393            addi  x7,x0,2
  2c: 00300193            addi  x3,x0,3
  30: 4a771a63            bne x14,x7,4e4 <fail>
```

地址 0000_0004 执行 addi x1,x0,0 指令，此时 rs1 为 x0 的值 0，立即数 sext 为 0，alub_sel 选择立即数 sext 进如 alu，根据操作控制信号 alu_op=1 执行加法操作，wd_sel=1 选择写回 wd 值为 alu_c=0，结束 addi 指令

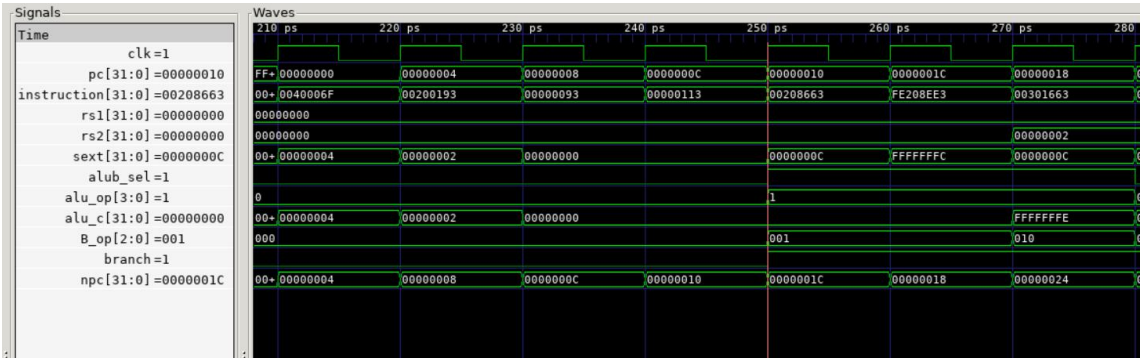
访存指令




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

00000004 <reset_vector>:
   4: 000020b7            lui x1,0x2
   8: 00008093            addi  x1,x1,0 # 2000 <begin_signature>
   c: 0000a703            lw  x14,0(x1)
  10: 00ff03b7            lui x7,0xff0
  14: 0ff38393            addi  x7,x7,255 # ff00ff <_end+0xfee0ef>
  18: 00200193            addi  x3,x0,2
  1c: 26771a63            bne x14,x7,290 <fail>
```

地址 0000_000c 执行 lw x14,0(x1)指令，rs1 接收 x1 地址 00002000，立即数接收

到 0，通过 alu_op 选择加法指令，计算出地址 alu_c=0000_2000，在 dram 取出 00ff00ff 写入 dram 信号，wd_sel 选择输入为 dram，写回进 wd=00ff00ff,结束跳转指令



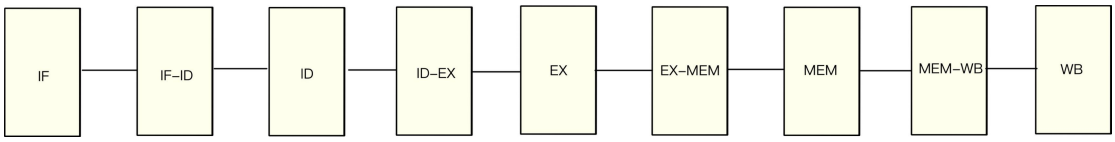

```
00000000 <_start>:
   0: 0040006f            jal x0,4 <reset_vector>

00000004 <reset_vector>:
   4: 00200193            addi  x3,x0,2
   8: 00000093            addi  x1,x0,0
   c: 00000113            addi  x2,x0,0
  10: 00208663            beq x1,x2,1c <reset_vector+0x18>
  14: 2a301863            bne x0,x3,2c4 <fail>
  18: 00301663            bne x0,x3,24 <test_3>
  1c: fe208ee3            beq x1,x2,18 <reset_vector+0x14>
  20: 2a301263            bne x0,x3,2c4 <fail>
```

地址 0000_0010 执行 beq x1,x2,1c 指令，x1,x2 分别存入 rs1,rs2，传入 alu 中，根据 alu_op=1 执行减法操作，B_op 判断为 beq 指令，根据结果得到 branch 为 1，发生跳转，npc 计算出跳转地址 0000_001c，执行下一步操作

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）



五级流水线划分：

IF 取指：是指将指令从存储器中读取出来的过程。

ID 译码：是指将存储器中取出的指令进行翻译的过程。经过译码之后得到指令需要的操作数寄存器索引，可以使用此索引从通用寄存器组中将操作数读出。

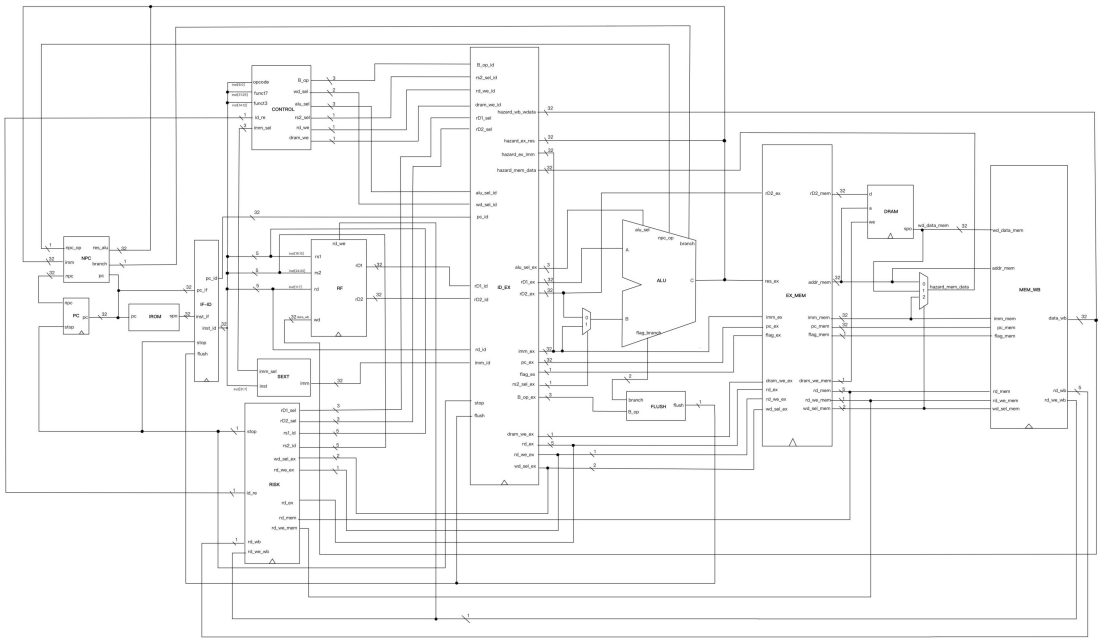
EX 执行：是指对指令进行真正运算的过程。譬如，如果指令是一条加法运算指令，则对操作数进行加法操作；如果是减法运算指令，则进行减法操作。在“执行”阶段的最常见部件为算术逻辑部件运算器，作为实施具体运算的硬件功能单元。

MEM 访存：是指存储器访问指令将数据从存储器中读出，或者写入存储器的过程。

WB 写回：是指将指令执行的结果写回通用寄存器组的过程。如果是普通运算指令，该结果值来自于“执行”阶段计算的结果；如果是存储器读指令，该结果来自于“访存”阶段从存储器中读取出来的数据。

2.2 流水线 CPU 整体框图

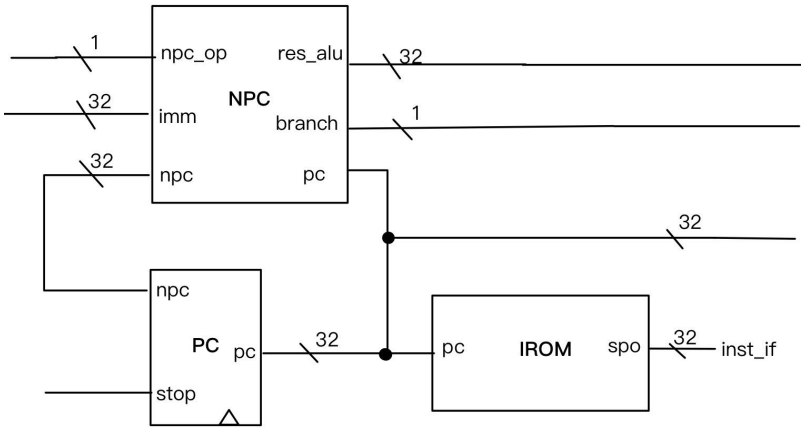
（要求：无需画出模块内的逻辑，但要标出模块之间信号线的信号名和位宽，以及说明每个模块的功能含义）



IF:取指模块，包括 NPC,PC,IROM,实现地址的控制和指令的取出
IF_ID:IF/ID 流水寄存器，与停顿和分支预测相关
ID：译码模块，包括 RF，SEXT,将地址转换为指令，对立即数进行符号扩展，传递给下一级
ID_EX：ID/EX 流水寄存器，与停顿，前递，分支预测相关
EX：执行模块，包括 ALU 和分支预测，完成各项运算过程，参与前递过程
EX_MEM:EX/MEM 流水寄存器，与执行和访存相关
MEM：访存模块，包括 DRAM 实现访存相关指令，参与前递过程
MEM_WB：MEM/WB 流水寄存器，控制写回内容

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）

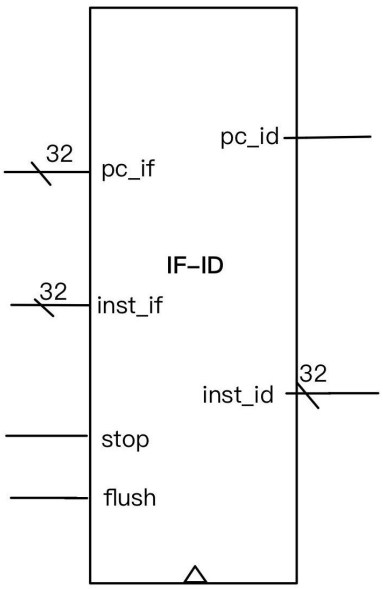


IF 模块

NPC 接收当前 pc 地址根据传来的控制信号 npc_op 与 branch 分支跳转信号决定下一步地址取值为 pc+4 或者是 alu 计算出的跳转地址，传给 pc。

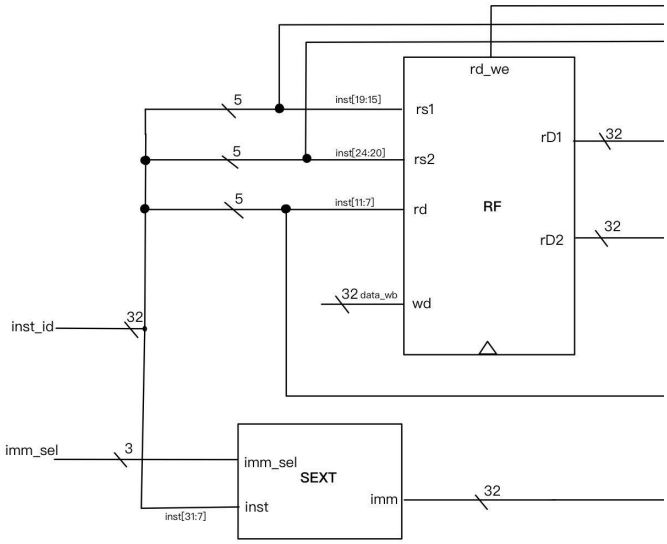
PC 根据接收到下一步地址进行转换，接收 stop 信号通过停顿解决 lw 指令的数据冒险，将当前地址 pc 传给 IROM 和 NPC。

IROM 接收到 pc 地址，取出相应指令传给下一级



IF/ID 流水寄存器

接收指令和地址，并传给下一级，当停顿信号 stop 为 1 时，传递上一次地址和指令实现停顿，当分支预测信号为 1 时，将传递信号清零完成静态分支预测。

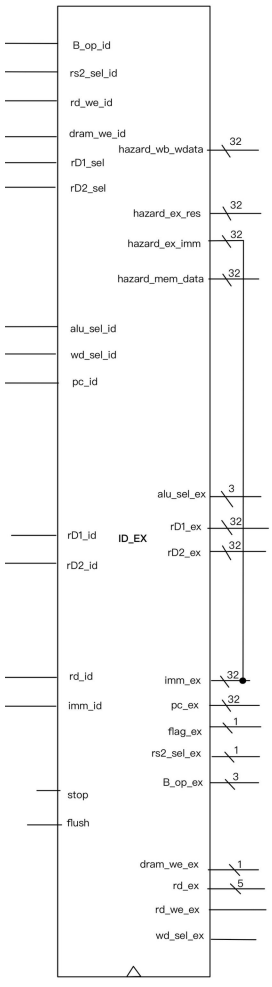


ID 模块

指令 inst 传入 ID，RF 寄存器堆根据指令接收目的寄存器和运算寄存器，将寄存器数据通过 rD1 和 rD2 传给下一级。

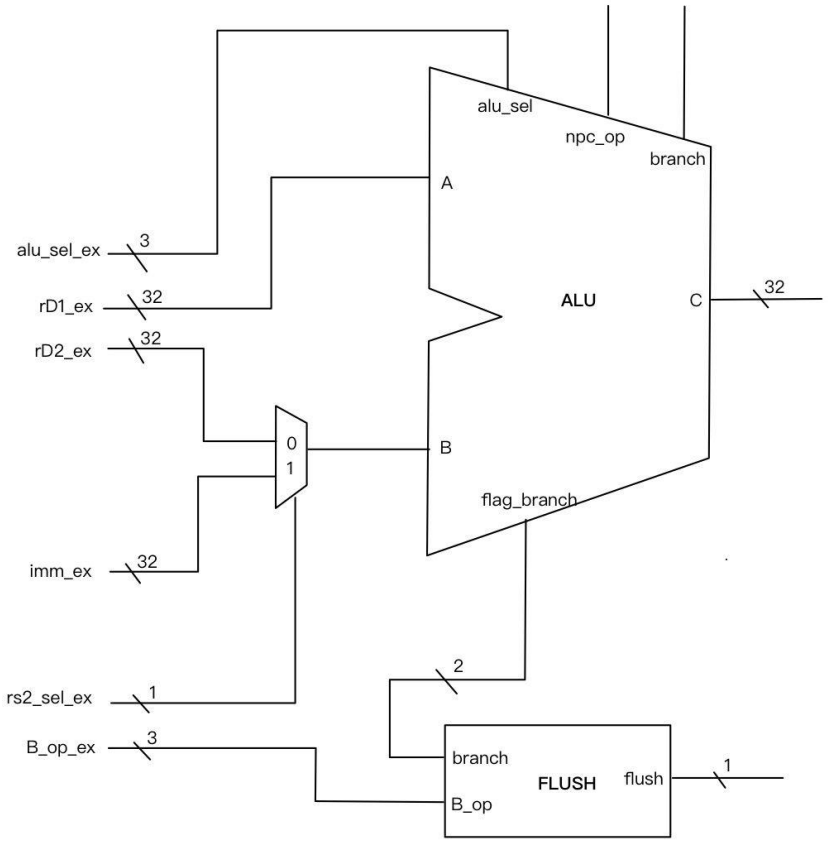
SEXT 根据指令格式进行立即数扩展，传给下一级。

RF 接收写回使能信号，将 data_wd 写回寄存器堆。



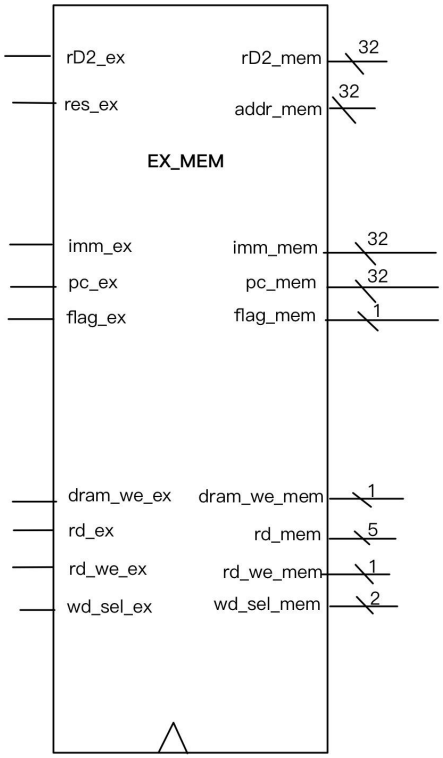
ID/EX 流水寄存器

接收 ID 阶段的信号传递给下一级，stop 信号用停顿解决 lw 指令的数据冒险，接收 flush 分支预测信号进行分支预测。
当发生数据冒险时，根据 risk 模块传来的 rD1_sel 和 rD2_sel 决定接收的前递信号为 ID/EX/MEM/WB 阶段的信号。

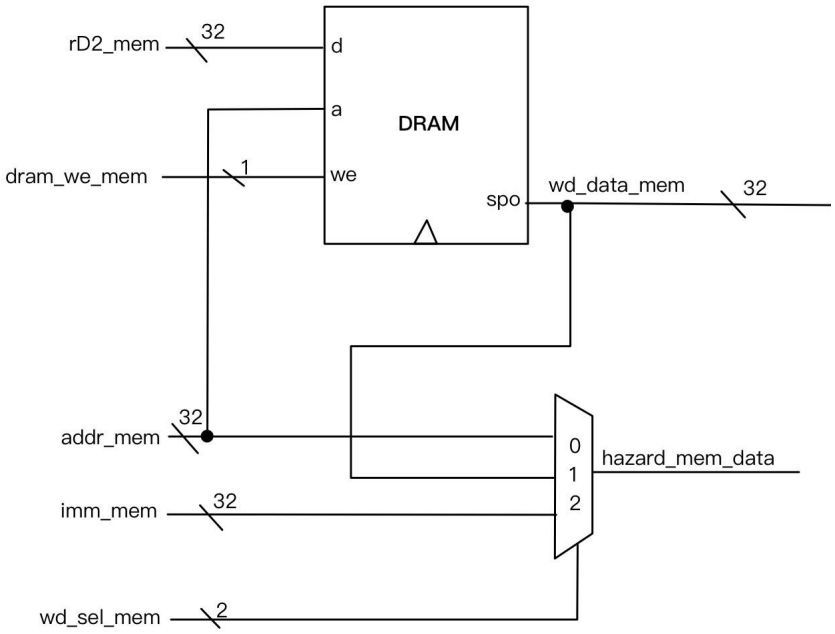


EX 模块
rs2_sel_ex 控制传入 alu_B 的值，ALU 接收到 A,B 之后根据 alu_sel 执行指令对应的运算操作通过 C 输出传递给相应模块。如果存在跳转，则将跳转指令类型 npc_op 和分支跳转信号 branch 传回 npc，进行跳转。

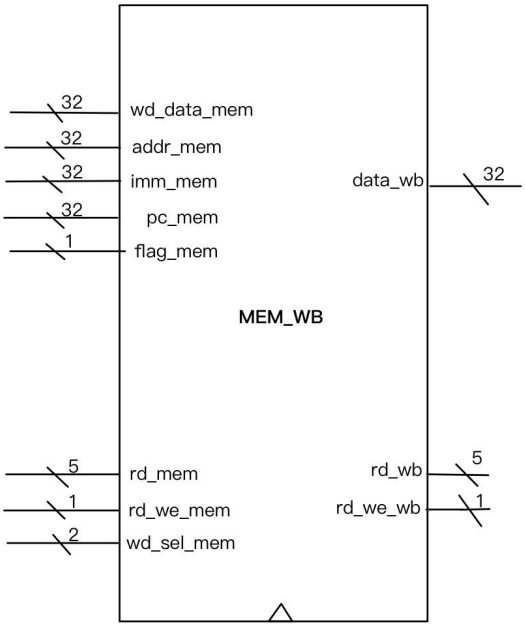
FLUSH 进行默认不跳转的静态预测，如果结果发生跳转，则将分支预测信号 flush 置 1 传回，对 IF_ID,ID_EX 两个流水寄存器清零，完成分支预测。



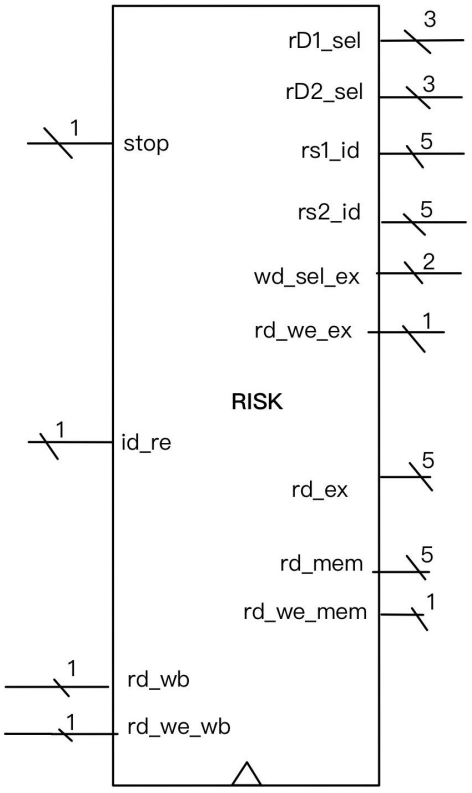
EX_MEM 模块
接收 EX 模块信号传递给下一级 MEM



MEM 模块
DRAM 根据地址 a，写使能信号控制读写，d 为写入内容。通过 spo 从 dram 中取出写回数据。
wd_sel_mem 信号选择前递的信号，传回给 ID/EX 流水寄存器



MEM/WB 流水寄存器
接收来自 MEM 的信号，根据 wd_sel_mem 决定写回的结果 data_wb 为 dram 中取出结果 wd_data_mem/addr/imm，写回到 RF 寄存器堆。



数据冒险检测模块
根据上一条指令的目的寄存器与下一条指令的运算寄存器比较进行数据冒险检测：

如果两个寄存器相同，则发生数据冒险，进行进一步的判断，分为四种情况

①上一条指令为 R 型且相隔一个周期。在发生冒险时，对此种情况进行判断：
上一条指令执行到 EX 阶段
上一条指令写回的寄存器不为固定值 x0
上一条指令需写回（非 S 和 B 型指令）
上一条指令写回的结果是 ALU 的计算值（非 J 和 U 型和 jalr 指令）
下一条指令不是 U 或者 J 型指令（不会发生冒险）

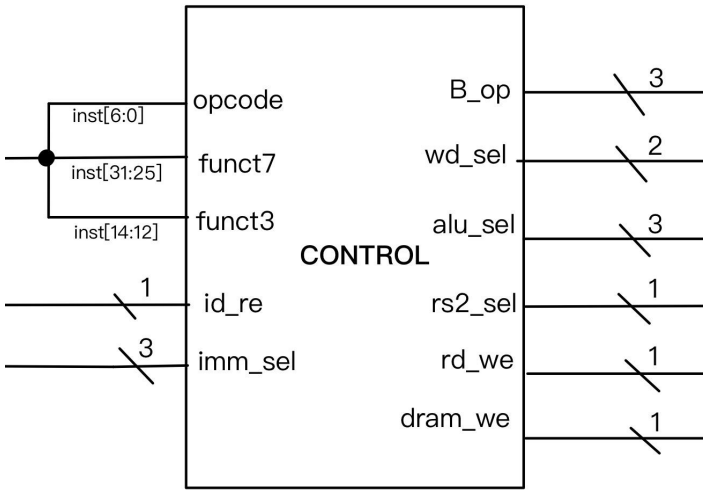
②上一条指令为 U 型、J 型或 jalr 指令，且相隔一个周期。在发生冒险时，从 MEM 阶段将数据前递。对此种情况进行判断：
上一条指令执行到 EX 阶段
上一条指令写回的寄存器不为固定值 x0
上一条指令需写回（非 S 和 B 型指令）
上一条指令写回的结果不是 ALU 计算值（为 S 或 J 或 U 或 jalr 指令）
下一条指令不是 U 或者 J 型指令（不会发生冒险）

③上一条指令为 R 型且相隔两个周期。在发生冒险时，对此种情况进行判断：
上一条指令执行到 MEM 阶段(J、U、jalr 指令不经过 MEM 阶段)
上一条指令写回的寄存器不为固定值 x0
上一条指令需写回（非 S 和 B 型指令）
下一条指令不是 U 或者 J 型指令（不会发生冒险）

③上一条指令为 R 型且相隔三个周期。在发生冒险时，对此种情况进行判断：
上一条指令执行到 WB 阶段
上一条指令写回的寄存器不为固定值 x0
上一条指令需写回（非 S 和 B 型指令）
下一条指令不是 U 或者 J 型指令（不会发生冒险）

根据判断结果，将前递选择信号传给 ID/EX 解决数据冒险。

lw 指令使用停顿的方法。当 rs1/rs2 使用 lw 写回的寄存器时发生冒险，使用停顿的方法，等待 lw 指令执行完毕写回寄存器，后续指令再执行，解决冒险。

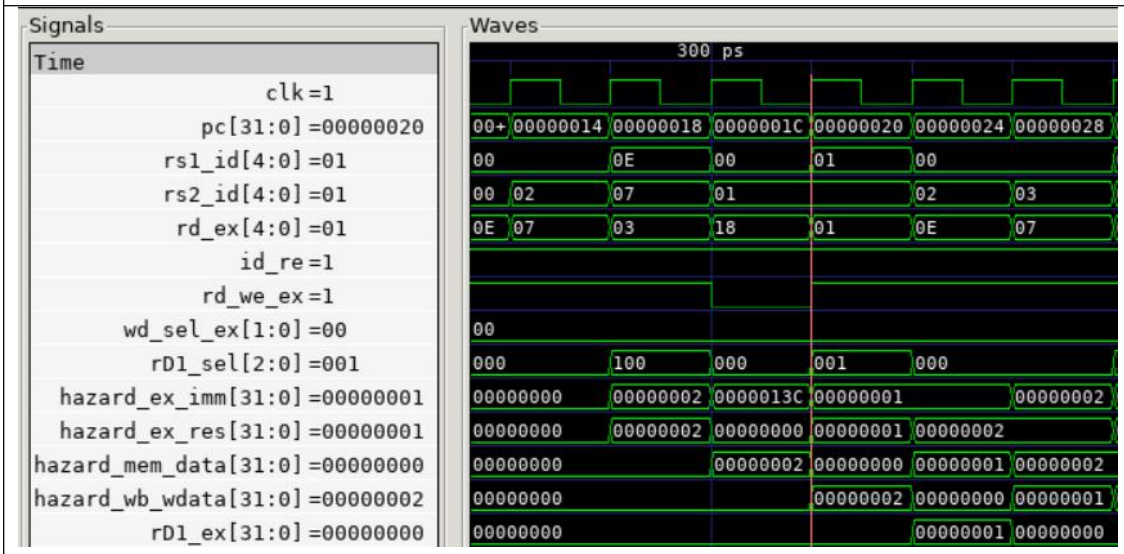


控制模块

根据指令对应输出结果，给出选择信号确定数据通路，统筹输出结果。

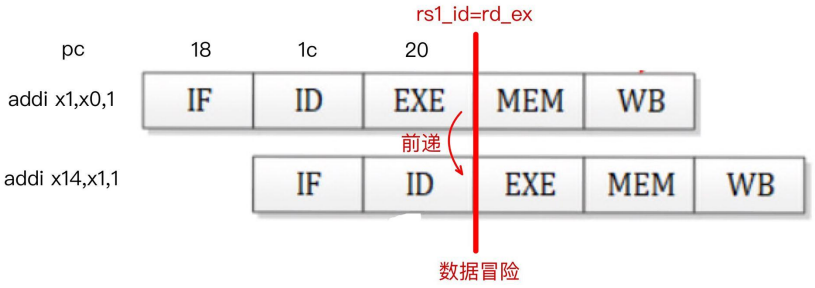
2.4 流水线 CPU 仿真及结果分析

（要求：包含数据冒险、控制冒险的仿真截图，以及结果分析）

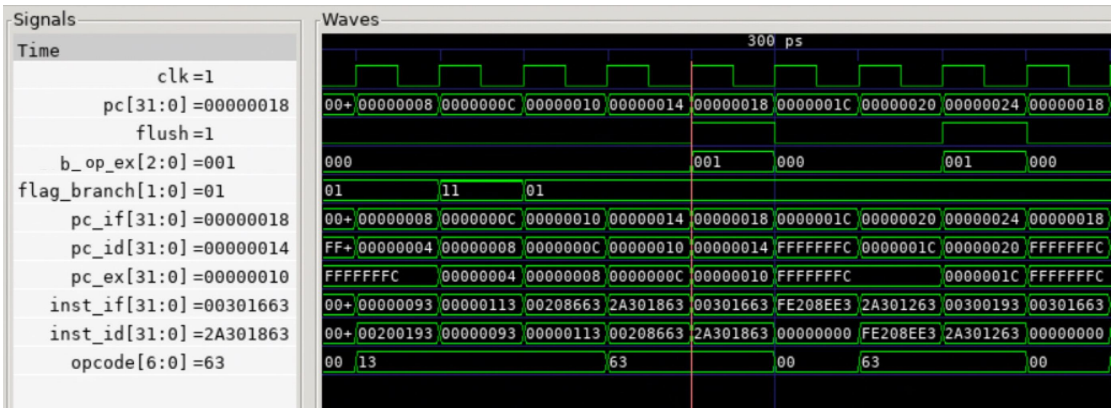



```
17 00000018 <test_3>:
18   18: 00100093          addi  x1,x0,1
19   1c: 00108713          addi  x14,x1,1
20   20: 00200393          addi  x7,x0,2
21   24: 00300193          addi  x3,x0,3
22   28: 26771263          bne x14,x7,28c <fail>
```

数据冒险

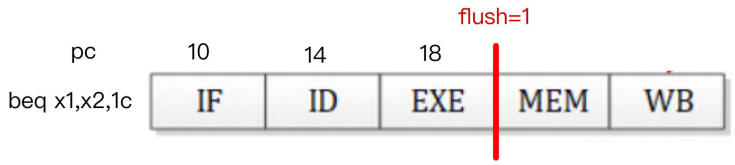


当上一条指令 addi x1,x0,1 执行到 EX 阶段时，下一条指令 addi x14,x1,1 执行到 ID 阶段，此时 risk 模块检测到上一条指令的目的寄存器与下一条指令的运算寄存器 rs1_id=rd_ex，发生数据冒险。
rd_ex≠0，即上一条指令写回的寄存器不为固定值 x0
rd_we_ex=1，即上一条指令需写回（非 S 和 B 型指令）
wd_sel_ex=0 上一条指令写回的结果是 ALU 的计算值（非 J 和 U 型和 jalr 指令）
id_re≠0 下一条指令不是 U 或者 J 型指令（不会发生冒险）
判断为第一种数据冒险 rD1_sel=1，将 rd_ex 的结果前递至 rD1_ex 进行后续运算。




```
10 00000004 <reset_vector>:
11    4: 00200193            addi  x3,x0,2
12    8: 00000093            addi  x1,x0,0
13    c: 00000113            addi  x2,x0,0
14   10: 00208663            beq x1,x2,1c <reset_vector+0x18>
15   14: 2a301863            bne x0,x3,2c4 <fail>
16   18: 00301663            bne x0,x3,24 <test_3>
17   1c: fe208ee3            beq x1,x2,18 <reset_vector+0x14>
18   20: 2a301263            bne x0,x3,2c4 <fail>
```

控制冒险：



当 beq x1,x2,1c 执行到 EX 阶段时，b_op_ex 检测为 beq 指令，结果相等 flag_branch=1，预测不跳转而实际要发生跳转，分支预测信号 flush 置为 1，对 IF/ID,ID/EX 寄存器中的信号清零，从新开始从跳转地址开始执行。

3 设计过程中遇到的问题及解决方法

设计过程中遇到了诸多的困难。

首先,遇到最大的困难就是如何下手,从哪一方面开始进行整个程序的设计。由于理论和实践的差距,对于具体到每一部分应该如何设计,以及总体框架之间的联系并不是完全清楚。通过查阅课件,上网查阅相关资料和同学讨论,逐渐定下总体框架,并去一步一步细化,经过不断的试错去慢慢解决这些问题。

其次,就是设计逻辑上的漏洞,很多时候因为信号之间传递的逻辑存在问题,或者缺失或冗余了一些判断条件,导致最后执行的结果异常。通过看反汇编代码以及测试波形去找到错漏之处,反复查看修正,不断解决问题。

最后,就是上板遇到的困难。上板考虑的因素很多电脑配置的环境,cpu 设计的频率,有些 trace 验证没有问题但是上板过程中会遇到的逻辑不严谨的漏洞等等都会导致失败。通过不断调试,以及向同学请教会遇到的常见问题,相互讨论最终才得以把问题解决。

4 总结

（要求：个人收获以及对课程的建议）

《计算机设计与实践》课程作为《计算机组成原理》课程的后续课程和实践环节,将计算机组成原理课程的教学内容深化到应用实践,教学过程中不仅仅传授有关硬件设计的课本知识,更重要的是将重视理论知识与实践过程结合,实践教学内容不仅要将组成原理知识应用到实践中,还需将知识综合灵活运用,重视学生综合能力和创新能力的训练和培养。

实验不仅仅是对理论的验证,重要的是技术训练和能力培养,包括动手能力、分析问题和解决问题的能力、书写能力和表达能力、团队协作能力等的培养。实践可以启发我们深入思考、敢于创新,增强我们对未来的适应性和针对性,达到良好的理论联系实际的效果。