ICS Seminar Week6 Prep

吴思衡 王效乐 许珈铭

2023.10.23

Rules

remainder <- ordinal number in WeChat Group % 4

for all questions do

    if question number % 4 == remainder then

        you should work on it

    end

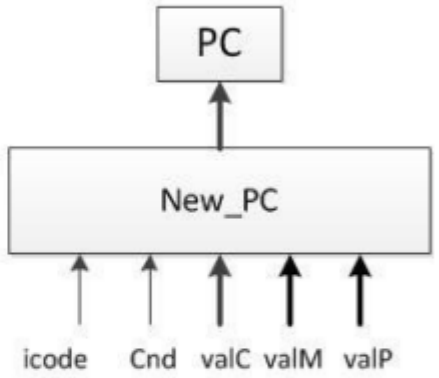
end

Q1

14. 在 Y86 的 SEQ 实现中，PC（Program Counter，程序计数器）更新的逻辑

结构如下图所示，请根据 HCL 描述为①②③④选择正确的数据来源。

```
Int new_pc = [
    # Call.
    Icode = ICALL : ① ;
    # Taken branch.
    Icode = IJXX && Cnd : ② ;
    # Completion of RET instruction.
    Icode = IRET : ③ ;
    # Default.
    1 : ④ ;
```



其中：Icode 为指令类型，Cnd 为条件是否成立，valC 表示指令中的常数值，valM 表示来自返回栈的数据，valP 表示 PC 自增。

A. valC, valM, valP, valP
B. valC, valC, valP, valP
C. valC, valC, valM, valP
D. valM, valC, valC, valP

①ICALL指令更新的PC应该是call指令后面的地址，这个地址在取指阶段存入valC中。
②IJXX和ICALL相同，在Cnd为1的情况下更新PC值为jxx后面的地址，在valC中。
③IRET指令更新PC的值应该是从栈中弹出的值，由于访问了内存，应该是valM。
④其余情况都是在取指阶段计算的valP。

C

Q2

10. 在书中 Y86 的 SEQ 实现下，以下哪一条指令是现有信号通路能完成的：

A. iaddq rA, V：将立即数 V 与 R[rA]相加，其中 rB 域设为 F，结果存入寄存器 rA

B. mmmovq rA, rB：将 R[rA]存的地址开始的 8 字节数据，移动到 R[rB]存的地址

C. leave：相当于先执行 rrmovq %rbp, %rsp，再执行 popq %rbp

D. enter：相当于先执行 pushq %rbp，再执行 rrmovq %rsp, %rbp

A.立即数与寄存器的值相加应该用rB。
B.得到ra指明内存的数据和赋值给rb指明内存都是在访存阶段进行，所以没法直接复制。
C.正确
D.Push指令会改变%rsp的值。

C

Q3

13. 关于流水线技术的描述，错误的是：

A. 流水线技术能够提高执行指令的吞吐率，但也同时增加单条指令的执行时间。

B. 减少流水线的级数，能够减少数据冒险发生的几率。

C. 指令间数据相关引发的数据冒险，都可以通过 data forwarding 来解决。

D. 现代处理器支持一个时钟内取指、执行多条指令，会增加控制冒险的开销。

A：正确，流水线能提高吞吐量，但由于寄存器延迟会增加单条指令的执行时间
B：正确，级数越多，指令间重叠度越高，越易发生冒险
C：加载/使用冒险不可以
D：正确，预测错误增多，时间开销增加

C

Q4

9． 下面对流水线技术的描述，正确的是：

A．流水线技术不仅能够提高执行指令的吞吐率，还能减少单条指令的执行时间。

B．不断加深流水线级数，总能获得性能上的提升。

C．流水级划分应尽量均衡，吞吐率会受到最慢的流水级影响。

D．指令间的数据相关可能会引发流水线停顿，但总是可以通过调度指令来解决。

A：错误，流水线能提高吞吐量，但由于寄存器延迟会增加单条指令的执行时间
B：错误，流水线过深，过高的寄存器延迟会降低效率
C：正确
D：错误，加载/使用冒险不可以

C

Q5

11．关于流水线技术的描述，错误的是：

A．流水线技术能够提高执行指令的吞吐率，但也同时增加单条指令的执行时间

B．增加流水线级数，不一定能获得总体性能的提升

C．指令间数据相关引发的数据冒险，不一定可以通过暂停流水线来解决。

D．流水级划分应尽量均衡，吞吐率会受到最慢的流水级影响，均衡的流水线能提高吞吐量。

A：正确，流水线能提高吞吐量，但由于寄存器延迟会增加单条指令的执行时间
B：正确，流水线过深，寄存器延迟将过高
C：错误，数据冒险总能通过暂停解决
D：正确

C

Q6

12、关于流水线技术的描述，正确的是：

A．指令间数据相关引发的数据冒险，一定可以通过暂停流水线来解决。

B．流水线技术不仅能够提高执行指令的吞吐率，还能减少单条指令的执行时间。

C．增加流水线的级数，一定能获得性能上的提升。

D．流水级划分应尽量均衡，不均衡的流水线会增加控制冒险。

答：（　　）

A：正确

B：错误，流水线能提高吞吐量，但由于寄存器延迟会增加单条指令的执行时间

C：错误，流水线过深，寄存器延迟将过高

D：错误，流水线划分不均衡的话，会使得整体速度取决于最慢的阶段，降低效率

A

Q7

7、下列说法正确的是：

A．在SEQ机器中，我们采用预测跳转总是选择（always taken）的策略比从不选择（never taken）的策略要略好。

B．流水级划分应尽量均衡，不均衡的流水线会增加控制冒险。

C． 如果一台机器的CPI小于1，则它必然不是普通流水线结构。

D． 由于rrmovq %rax, %rax不影响标记位，所以可使用其代替nop指令。

A：错误，SEQ不需要分支预测

B：错误，不均衡划分时，流水线整体速度取决于最慢的阶段，降低效率

C：正确，CPI是指“cycles per instruction”，衡量一条指令花费了多少时间，普通流水线结构每个周期吞吐一条指令，可知该流水线存在并行操作等特殊机制

D：错误，会造成数据相关

C

Q8

12. 一个功能模块包含组合逻辑和寄存器，组合逻辑单元的总延迟是 100ps，单个寄存器的延时是 20ps，该功能模块执行一次并保存执行结果，理论上能达到的最短延时和最大吞吐分别是多少？

A. 20ns, 50GIPS

B. 120ns, 50GIPS

C. 120ns, 10GIPS

D. 20ps,10GIPS

B：增加划分会增加寄存器延迟，故不增加划分时延时最短，此时为100+20=120ps。设划分阶段数目为n，则需用到的寄存器数目为n-1，每个阶段的时间为(100/n+20)，机器每个阶段吞吐一条指令，故每秒吞吐量为1/(100/n+20)*1000/1ns。理论上吞吐量最大时（n趋于无穷），即该表达式的最大值为50GIPS。

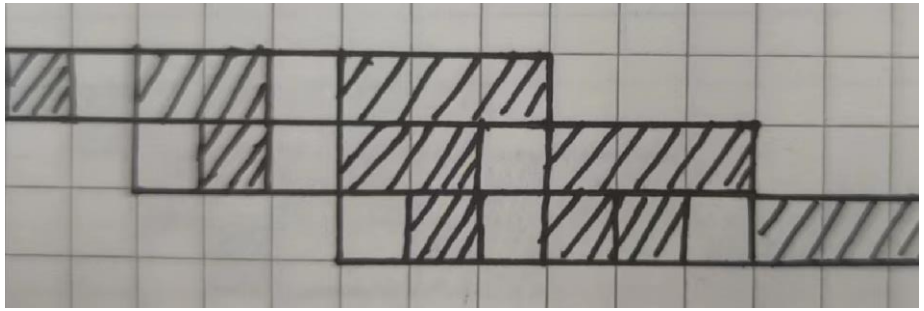
B

Q9

12. 若处理器实现了三级流水线，每一级流水线实际需要的运行时间分别为 1ns、2ns 和 3ns，则此处理器不停顿地执行完毕 10 条指令需要的时间为：

A. 21 ns B. 12 ns C. 24 ns D. 36 ns

1
2
3

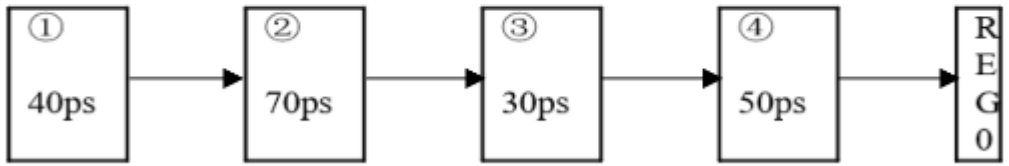


D：如图所示，当划分不均匀时流水线的效率取决于最慢的那一阶段，即第三个阶段，除第一条(用时9ns)外，每三秒执行完毕一条指令。故总时间为9+3*9=36ns

D

Q10

11. 如下图所示，①~④为四个组合逻辑单元，对应的延迟已在图上标出，REG0 为一寄存器，延迟为 20ps。通过插入**额外的 2 个**流水线寄存器 REG1、REG2（延迟均为 20ps），可以对其进行流水化改造。改造后的流水线的吞吐率最大为_____GIPS。



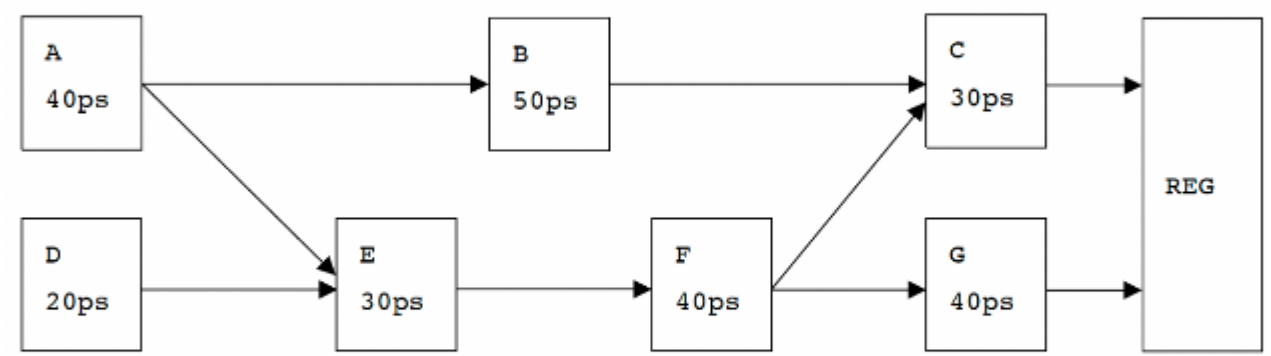
A. 7.69　　B. 8.33　　C. 10.00　　D. 11.11

C：划分不均匀时流水线速度取决于最慢的阶段，在划分中，最均匀的方法是1，2，34（此时最慢阶段为80ps），考虑寄存器延迟为20ps，故每80+20=100ps吞吐一条指令，吞吐率最大为1/100*1000/Ins=10GIPS

C

Q11

8. A-G 为 7 个基本逻辑单元，下图中标出了每个单元的延迟，以及用箭头标出了单元之间所有的依赖关系。寄存器的延迟均为 20ps，在图中以 REG 符号表示。假设流水线寄存器只能添加在有直接依赖关系的基本逻辑单元之间，而不能在 C 或 G 与 REG 之间。以下说法正确的是：



A. 原电路的吞吐量(throughput)舍入后大约是 1000/150=6.667 GIPS。

B. 将该电路改造成 2 级流水线有 8 种方法

C. 如果将该电路改造成 3 级流水线，延迟最小可以到 80 ps。

D. 不论实现该电路时遇到怎样的数据冒险和控制冒险，一定可以对流水线寄存器使用暂停(stalling)解决。

A：错误，原电路处理一条指令用时应为40+30+40+40+20=170ps，即1000/170

B：错误，考虑 D-E-F-G 这条路，要么插 D-E，要么 E-F，要么 F-G。如果是 D-E，那么为了使每条极大路径上都恰有一个新插入的寄存器，考虑 D-E-F-C，那么 F-C 之间也不能插入了。但是 A-E-F-G 上又必须有一个，所以只能插 A-E。这样之后不管是插在 A-B 还是 B-C 所得方案都是合法的。对称地，如果是 F-G，结果也是如此。最后考虑 E-F 的情况，此时 A-E，D-E，F-C，F-G 之间均不能再插入，但 A-B 和B-C 任选一个插入得到的方案都合法。因此一共有 2+2+2=6 种。

C：错误，在A, B, C之间插入寄存器此路延迟70ps；F, G之间必须插入寄存器否则，延迟100ps，因此F, C之间插入寄存器；A, E和D, E间插入寄存器或者E, F之间插入寄存器，可知整个电路最小延迟为90ps

D：正确

D

Q12

10. 在 Y86 的 SEQ 实现中，对仅考虑 IRMMOVQ，ICALL，IPOPQ，IRET 指令，对 mem_addr 的 HCL 描述正确的是：

```
word mem_addr = [
    icode in { (1), (2) } : valE;
    icode in { (3), (4) } : valA;
];
```

A. (1) IRMMOVQ (2) IPOPQ (3) IRET (4) ICALL
B. (1) IRMMOVQ (2) IRET (3) IPOPQ (4) ICALL
C. (1) ICALL (2) IPOPQ (3) IRMMOVQ (4) IRET
D. (1) IRMMOVQ (2) ICALL (3) IPOPQ (4) IRET

（1）（2）这里是读取写入内存需要的地址，而valE是执行阶段计算的结果，IRMMOVEQ在执行阶段是计算地址的偏移，ICALL是计算栈指针的变化。
（3）（4）IPOPQ和IRET指令的valA在之前都是记录变化前%rsp的值，从而在访存阶段读取对应的valM进行出栈操作。

D

Q13

6、在Y86-64的PIPE实现中，仅考虑ICALL、IPOPQ、IPUSHQ、IRET指令，对mem_addr的HCL描述正确的是：

```
word mem_addr = [
    M_icode in { ①, ② } : M_valE;
    M_icode in { ③, ④ } : M_valA;
];
```

A. ①IPUSHQ ②ICALL ③IPOPQ ④IRET
B. ①IPUSHQ ②IRET ③ICALL ④IRET
C. ①IPUSHQ ②IPOPQ ③IRET ④ICALL
D. ①IPUSHQ ②IRET ③IPUSH ④ICALL

（1）（2）这里是读取写入内存需要的地址，而valE是执行阶段计算的结果，ICALL和IPUSHQ是计算栈指针的变化。
（3）（4）IPOPQ和IRET指令的valA在之前都是记录变化前%rsp的值，从而在访存阶段读取对应的valM进行出栈操作。

A

Q14

10. Y86 指令 popl rA 的 SEQ 实现如下图所示，其中❶和❷分别为：

Fetch	icode:ifun ← M_1[PC] ra:rb ← M_1 [PC+1] valP ← ❶
Decode	valA ← R[%esp] valB ← R[%esp]
Execute	valE ← ❷
Memory	valM ← M_4 [valA]
Write Back	R[%esp] ← valE R[ra] ← valM
PC Update	PC ← valP

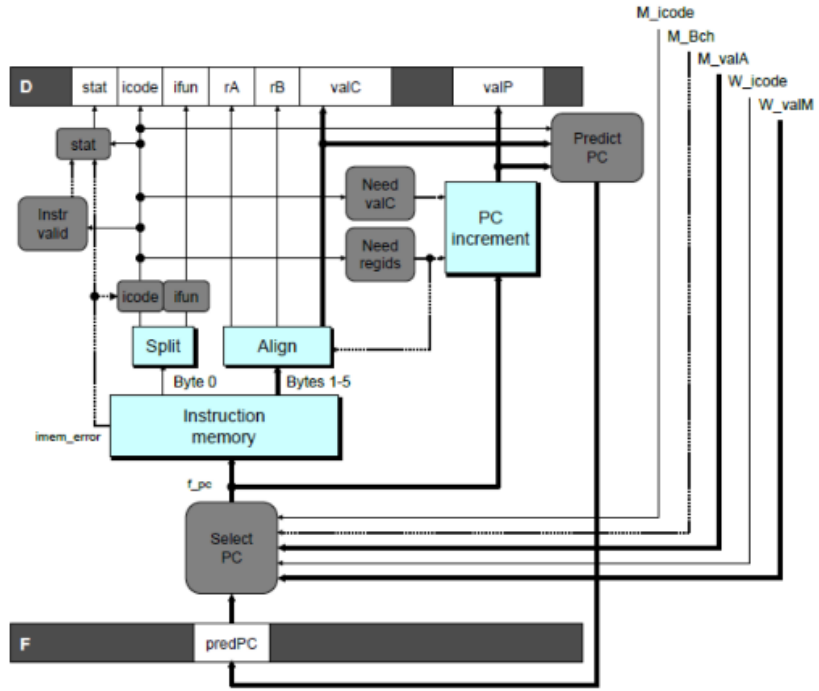
A) PC + 4 valA + 4
B) PC + 4 valA + (- 4)
C) PC + 2 valB + 4
D) PC + 2 valB + (- 4)

1.取出了icode和ifun，rA和rB，分别占用4位，一共是2bytes，所以PC+2。
2.因为是popl，针对4位进行操作，所以pop指令让%rsp+4。

C

Q15

11. 流水线数据通路中的转移预测策略为总是预测跳转。如果转移预测错误，需要恢复流水线，并从正确的目标地址开始取值。其中，用来判断转移预测是否正确的信号是_①_和_②__，用来获得正确的目标地址的信号是_③_。



A. ① M_icode ② M_Bch ③ M_valA
B. ① W_icode ② M_Bch ③ M_valA
C. ① W_icode ② M_Bch ③ W_valM
D. ① M_icode ② M_Bch ③ W_valM

A：M寄存器保存指令执行阶段完成后的一些信号，W保存访存阶段后的一些信号。在执行阶段通过逻辑运算判断是否跳转以及跳转位置，并将信号值保存在M寄存器中，故选择A

A

Q16

9. 在课本 Y86-64 的 PIPE 上执行以下的代码片段，一共使用到了（　　）次数据转发。假设在该段代码执行前和执行后 PIPE 都执行了足够多的 nop 指令。

```
1  mrmovq 0(%rdx), %rax
2  addq   %rbx, %rax
3  mrmovq 8(%rdx), %rcx
4  addq   %rcx, %rax
5  irmovq $10, %rcx
6  addq   %rcx, %rax
7  rmmovq %rax, 16(%rdx)
```

A. 3　　B. 4　　C. 5　　D. 6

加载/使用冒险	E_icode∈{IMRMOVL,IPOPL} & & E_dstM∈{d_srcA,d_srcB}

D：在数据冒险中，除加载/使用需用暂停加转发解决，其余都可以单用转发解决。在图示指令的执行中：

- 12之间为加载/使用冒险，需在暂停后将m_valM转发给valA；
- 24之间需要转发，34之间为加载/使用，需将W_valE和m_valM转发给valA和valB；
- 46需转发，56需转发，分别将M_valE和e_valE转发给valA和valB；
- 67需转发，将e_valE转发给valB

综上所述，共需6次转发

D

Q17

第四题（20 分）

分析32位的Y86 ISA中新加入的条件内存传送指令：crmmovqXX和cmrmovqXX。crmmovqXX和cmrmovqXX指令在条件码满足所需要的约束时，分别执行和rmmovq以及mrmovq同样的语义。其格式如下：

rmmovq	4	0	rA	rB	D（8字节）
crmmovqXX	4	fn	rA	rB	D（8字节）
mrmovq	5	0	rA	rB	D（8字节）
cmrmovqXX	5	fn	rA	rB	D（8字节）

1．请按下表补全每个阶段的操作。需说明的信号可能会包括：icode，ifun，rA，rB，valA，valB，valC，valE，valP，Cnd；寄存器堆R[]，存储器M[]，程序计数器PC，条件码CC。其中对存储器的引用必须标明字节数。

阶段	rmmovq rA,D(rB)	cmrmovqXX D(rB),rA
取指		
译码	valA ← R[rA] valB ← R[rB]	
执行		
访存		
写回	none	
更新PC	PC ← valP	

Q17

F:由图可以得出一共取出了1+1+8共十个字节，所以就和课上讲的完全一致填入三条指令，注意虽然是32位，但D仍然是8字节。最后别忘记计算valP。
E:执行阶段就是计算地址的偏移然后赋值给valE，但因为第二个是条件内存传送指令，所以还要加上Cnd<-cond(CC,ifun)
M:访存阶段右边指令需要判断Cnd是否为1从而判断是否从内存中读取valM，当然这里也可以设计成无论条件码是什么都读取，在写回阶段在判断条件。
W:需要判断是否满足条件。

第四题（20 分）

分析32位的Y86 ISA中新加入的条件内存传送指令：crmmovqXX和cmrmovqXX。crmmovqXX和cmrmovqXX指令在条件码满足所需要的约束时，分别执行和rmmovq以及mrmovq同样的语义。其格式如下：

rmmovq	4	0	rA	rB	D（8字节）
crmmovqXX	4	fn	rA	rB	D（8字节）
mrmovq	5	0	rA	rB	D（8字节）
cmrmovqXX	5	fn	rA	rB	D（8字节）

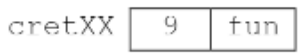
1．请按下表补全每个阶段的操作。需说明的信号可能会包括：icode，ifun，rA，rB，valA，valB，valC，valE，valP，Cnd；寄存器堆R[]，存储器M[]，程序计数器PC，条件码CC。其中对存储器的引用必须标明字节数。

阶段	rmmovq rA,D(rB)	cmrmovqXX D(rB),rA
取指	icode:ifun ← M_1[PC] rA:rB ← M_1[PC+1] valC ← M_8[PC+2] valP ← PC + 10	
译码	valA ← R[rA] valB ← R[rB]	
执行	valE ← valB + valC	valE ← valB + valC Cnd ← Cond(CC, ifun)
访存	M_8[valE] ← valA	valM ← M_8[valE] 或 if(Cnd) valM ← M_8[valE]
写回	none	if(Cnd) R[rA] ← valM
更新PC	PC ← valP	

Q18

第四题（20 分）

请分析32位的Y86 ISA中新加入的一组条件返回指令：cretXX，其格式如下。



类似cmovXX，该组指令只有当条件码(Cnd)满足时，才执行函数返回；如果条件不满足，则顺序执行。

1．若在教材所描述的SEQ处理器上执行这条指令，请按下表补全每个阶段的操作。需说明的信号可能会包括：icode, ifun, rA, rB, valA, valB, valC, valE, valP, Cnd; the register file R[], data memory M[], Program counter PC, condition codes CC。其中对存储器的引用必须标明字节数。如果在某一阶段没有任何操作，请填写nonc指明。

Stage	cretXX Offset
Fetch	
Decode	
Execute	
Memory	
Write back	
PC update	

Q18

第四题（20 分）

请分析32位的Y86 ISA中新加入的一组条件返回指令：cretXX，其格式如下。

cretXX	9	fun

类似cmovXX，该组指令只有当条件码(Cnd)满足时，才执行函数返回；如果条件不满足，则顺序执行。

1．若在教材所描述的SEQ处理器上执行这条指令，请按下表补全每个阶段的操作。需说明的信号可能会包括：icode, ifun, rA, rB, valA, valB, valC, valE, valP, Cnd; the register file R[], data memory M[], Program counter PC, condition codes CC。其中对存储器的引用必须标明字节数。如果在某一阶段没有任何操作，请填写none指明。

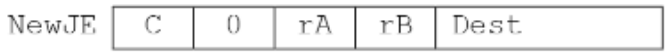
Stage	cretXX Offset
Fetch	icode:ifun ← M_1[PC] valP ← PC+1
Decode	valB ← R[%esp] valA ← R[%esp]
Execute	valE ← valB + 4 Cnd ← Cond(CC, ifun)
Memory	valM ← M_4[valA]
Write back	if (Cnd) R[%esp] ← valE
PC update	PC ← Cnd ? valM : valP

F:由图只需要取出1个byte，然后计算valP
D:与ret指令相同，将%esp赋给valA和valB。注意，因为是32位，不是rsp是esp。
E:这个阶段计算%esp的偏移，同样注意此处是+4而不是+8。同时设置条件码。
M:需要从valA指向的内存中读取4个字节。
W:题设中说只有符合条件才会返回，那么如果不符合条件，即使执行阶段计算了esp的偏移，也不真正赋值给esp，并没有真正执行。
PC:与跳转指令相似，满足条件返回，否则顺序执行。

Q19

第四题（20 分）

请分析Y86 ISA中新加入的一条指令：NewJE，其格式如下。



其功能为：如果R[rA]= R[rB]，则跳转到Dest继续执行，否则顺序执行。

1．若在教材所描述的SEQ处理器上执行这条指令，请按下表补全每个阶段的操作。需说明的信号可能会包括：icode, ifun, rA, rB, valA, valB, valC, valE, valP, Cnd; the register file R[], data memory M[], Program counter PC, condition codes CC。其中对存储器的引用必须标明字节数。如果在某一阶段没有任何操作，请填写none指明。

Stage	NewJE rA, rB, Dest
Fetch	icode:ifun ← M_1[PC] rA:rB ← M_1[PC+1] valC ← valP ←
Decode	valA ← R[rA] valB ← R[rB]
Execute	valE ←
Memory	
Write back	none
PC update	PC ← valE==0 ? ____ : _____

Q19

第四题（20 分）

请分析Y86 ISA中新加入的一条指令：NewJE，其格式如下。

NewJE	C	0	rA	rB	Dest

其功能为：如果R[rA]= R[rB]，则跳转到Dest继续执行，否则顺序执行。

1．若在教材所描述的SEQ处理器上执行这条指令，请按下表补全每个阶段的操作。需说明的信号可能会包括：icode, ifun, rA, rB, valA, valB, valC, valE, valP, Cnd; the register file R[], data memory M[], Program counter PC, condition codes CC。其中对存储器的引用必须标明字节数。如果在某一阶段没有任何操作，请填写none指明。

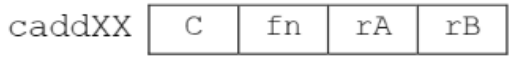
F:32位的话就取出四个字节的目标地址，PC+6；64位则取出八个字节目标地址，PC+10
E:判断valA与valB是否相等可以用他们相减判断是否为0，这里如果选择设置条件码那么更新PC时可以用Cnd判断，否则可以用valE是否为0判断。
M:不用进行操作
PC:需要判断是否进行跳转

Stage	NewJE rA, rB, Dest
Fetch	icode:ifun ← M_1[PC] rA:rB ← M_1[PC+1] valC ← M_4[PC+2] valP ← PC+6
Decode	valA ← R[rA] valB ← R[rB]
Execute	valE ← valA – valB （注：也可以是valB – valA）
Memory	none
Write back	none
PC update	PC ← valE==0 ? valC : valP

Q20

第六题（10 分）

请分析Y86 ISA中新加入的一条指令：caddXX，条件加法。其功能可以参考add和cmovXX两条指令。



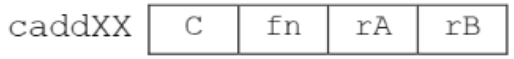
若在教材所描述的SEQ处理器上执行这条指令，请按下表填写每个阶段进行的操作。需说明的信号包括：icode, ifun, rA, rB, valA, valB, valC, valE, valP, Cnd; the register file R[], data memory M[], Program counter PC, condition codes CC。其中对存储器的引用必须标明字节数。如果在某一阶段没有任何操作，请填写none指明。

Stage	caddXX rA, rB
Fetch	
Decode	
Execute	
Memory	
Write back	
PC update	

Q20

第六题（10 分）

请分析Y86 ISA中新加入的一条指令：caddXX，条件加法。其功能可以参考add和cmovXX两条指令。



若在教材所描述的SEQ处理器上执行这条指令，请按下表填写每个阶段进行的操作。需说明的信号包括：icode, ifun, rA, rB, valA, valB, valC, valE, valP, Cnd; the register file R[], data memory M[], Program counter PC, condition codes CC。其中对存储器的引用必须标明字节数。如果在某一阶段没有任何操作，请填写none指明。

F:由图可知取出两个字节，别忘记计算valP
D:取出两个寄存器的值
E:执行加法操作，将结果赋值给valE，同时设置条件码
M:不用进行操作
W:如果条件符合那么将valE赋值给rB寄存器，否则不进行任何操作
PC:正常更新PC为valP

Stage	caddXX rA, rB
Fetch	icode:ifun ← M_1[PC] rA:rB ← M_1[PC+1] valP ← PC+2
Decode	valA ← R[rA] valB ← R[rB]
Execute	valE ← valA+valB Cnd ← Cond(CC,ifun)
Memory	none
Write back	if(Cnd) R[rB] ← valE
PC update	PC ← valP

Q21

(11-13)、在教材所描述的流水线处理器（the PIPE processor）上分别运行如下四段Y86程序代码。请分析其中数据冒险的具体情况，并回答后续3个小题。

#Program 1: mrmovl 8(%cbx), %cdx rmmovl %cdx, 16(%ccx)	#Program 2: mrmovl 8(%cbx), %cdx nop rmmovl %edx, 16(%ecx)
#Program 3: mrmovl 8(%ebx), %edx nop nop rmmovl %cdx, 16(%ccx)	#Program 4: mrmovl 8(%ebx), %edx nop nop nop rmmovl %edx, 16(%ecx)

11、对于每段程序，请指出是否会因为数据冒险导致流水线停顿（Stall）。

Program 1：（ ），Program 2：（ ），Program 3：（ ），Program 4：（ ）；

A. Stall B. No-Stall

12、对于每段程序，请指出流水线处理器内是否会产生数据转发（Forwarding）。

Program 1：（ ），Program 2：（ ），Program 3：（ ），Program 4：（ ）；

A. Forwarding B. No-Forwarding

13、对于每段程序，请指出流水线处理器内使用哪个信号进行数据转发，如果不进行数据转发，则用none表示。

Program 1：（ ），Program 2：（ ），Program 3：（ ），Program 4：（ ）；

A. m_valM B. W_valM C. none

A, B, B, B
A, A, A, B
A, A, B, C

Q22

(11-13)、在教材所描述的流水线处理器（the PIPE processor）上分别运行如下四段Y86程序代码。请分析其中数据冒险的具体情况，并回答后续3个小题。

#Program 1: mrmovl 8(%cbx), %cdx rmmovl %cdx, 16(%ccx)	#Program 2: mrmovl 8(%cbx), %cdx nop rmmovl %edx, 16(%ecx)
#Program 3: mrmovl 8(%ebx), %edx nop nop rmmovl %cdx, 16(%ccx)	#Program 4: mrmovl 8(%ebx), %edx nop nop nop rmmovl %edx, 16(%ecx)

11、对于每段程序，请指出是否会因为数据冒险导致流水线停顿（Stall）。

Program 1：（ ），Program 2：（ ），Program 3：（ ），Program 4：（ ）；

A. Stall B. No-Stall

12、对于每段程序，请指出流水线处理器内是否会产生数据转发（Forwarding）。

Program 1：（ ），Program 2：（ ），Program 3：（ ），Program 4：（ ）；

A. Forwarding B. No-Forwarding

13、对于每段程序，请指出流水线处理器内使用哪个信号进行数据转发，如果不进行数据转发，则用none表示。

Program 1：（ ），Program 2：（ ），Program 3：（ ），Program 4：（ ）；

A. m_valM B. W_valM C. none

A, B, B, B
A, A, A, B
A, A, B, C

Q23

(11-13)、在教材所描述的流水线处理器（the PIPE processor）上分别运行如下四段Y86程序代码。请分析其中数据冒险的具体情况，并回答后续3个小题。

#Program 1: mrmovl 8(%cbx), %cdx rmmovl %cdx, 16(%ccx)	#Program 2: mrmovl 8(%cbx), %cdx nop rmmovl %edx, 16(%ecx)
#Program 3: mrmovl 8(%ebx), %edx nop nop rmmovl %cdx, 16(%ccx)	#Program 4: mrmovl 8(%ebx), %edx nop nop nop rmmovl %edx, 16(%ecx)

11、对于每段程序，请指出是否会因为数据冒险导致流水线停顿（Stall）。

Program 1：（ ），Program 2：（ ），Program 3：（ ），Program 4：（ ）；

A. Stall B. No-Stall

12、对于每段程序，请指出流水线处理器内是否会产生数据转发（Forwarding）。

Program 1：（ ），Program 2：（ ），Program 3：（ ），Program 4：（ ）；

A. Forwarding B. No-Forwarding

13、对于每段程序，请指出流水线处理器内使用哪个信号进行数据转发，如果不进行数据转发，则用none表示。

Program 1：（ ），Program 2：（ ），Program 3：（ ），Program 4：（ ）；

A. m_valM B. W_valM C. none

A, B, B, B
A, A, A, B
A, A, B, C

Q24

(11-13)、在教材所描述的流水线处理器（the PIPE processor）上分别运行如下四段Y86程序代码。请分析其中数据冒险的具体情况，并回答后续3个小题。

	#Program 1:	#Program 2:
1	mrmovl 8(%ebx), %edx	mrmovl 8(%ebx), %edx
2	rmmovl %edx, 16(%ecx)	nop
3		rmmovl %edx, 16(%ecx)
	#Program 3:	**#Program 4:**
1	mrmovl 8(%ebx), %edx	mrmovl 8(%ebx), %edx
2	nop	nop
3	nop	nop
4	rmmovl %edx, 16(%ecx)	nop
5		rmmovl %edx, 16(%ecx)

11、对于每段程序，请指出是否会因为数据冒险导致流水线停顿（Stall）。

Program 1：（ ），Program 2：（ ），Program 3：（ ），Program 4：（ ）；

A. Stall B. No-Stall

12、对于每段程序，请指出流水线处理器内是否会产生数据转发（Forwarding）。

Program 1：（ ），Program 2：（ ），Program 3：（ ），Program 4：（ ）；

A. Forwarding B. No-Forwarding

13、对于每段程序，请指出流水线处理器内使用哪个信号进行数据转发，如果不进行数据转发，则用none表示。

Program 1：（ ），Program 2：（ ），Program 3：（ ），Program 4：（ ）；

A. m_valM B. W_valM C. none

加载/使用冒险	E_icode∈{IMRMOVL,IPOPL} && E_dstM∈{d_srcA,d_srcB}

11、

- 由上述条件可知，p1停顿，p2/p3/p4正常运行

12、

- p1在2正常解码时，1位于访存阶段（执行中暂停了一次），需将m_valM转发给valB
- p2在3正常解码时，1位于访存阶段，需将m_valM转发给valB
- p3在4正常解码时，1位于写回阶段，需将W_valM转发给valB
- p4不需要转发

13、

见上题

A, B, B, B
A, A, A, B
A, A, B, C