1. 选择题

（13期中）1.下面对流水线技术的描述， 正确的是：( )

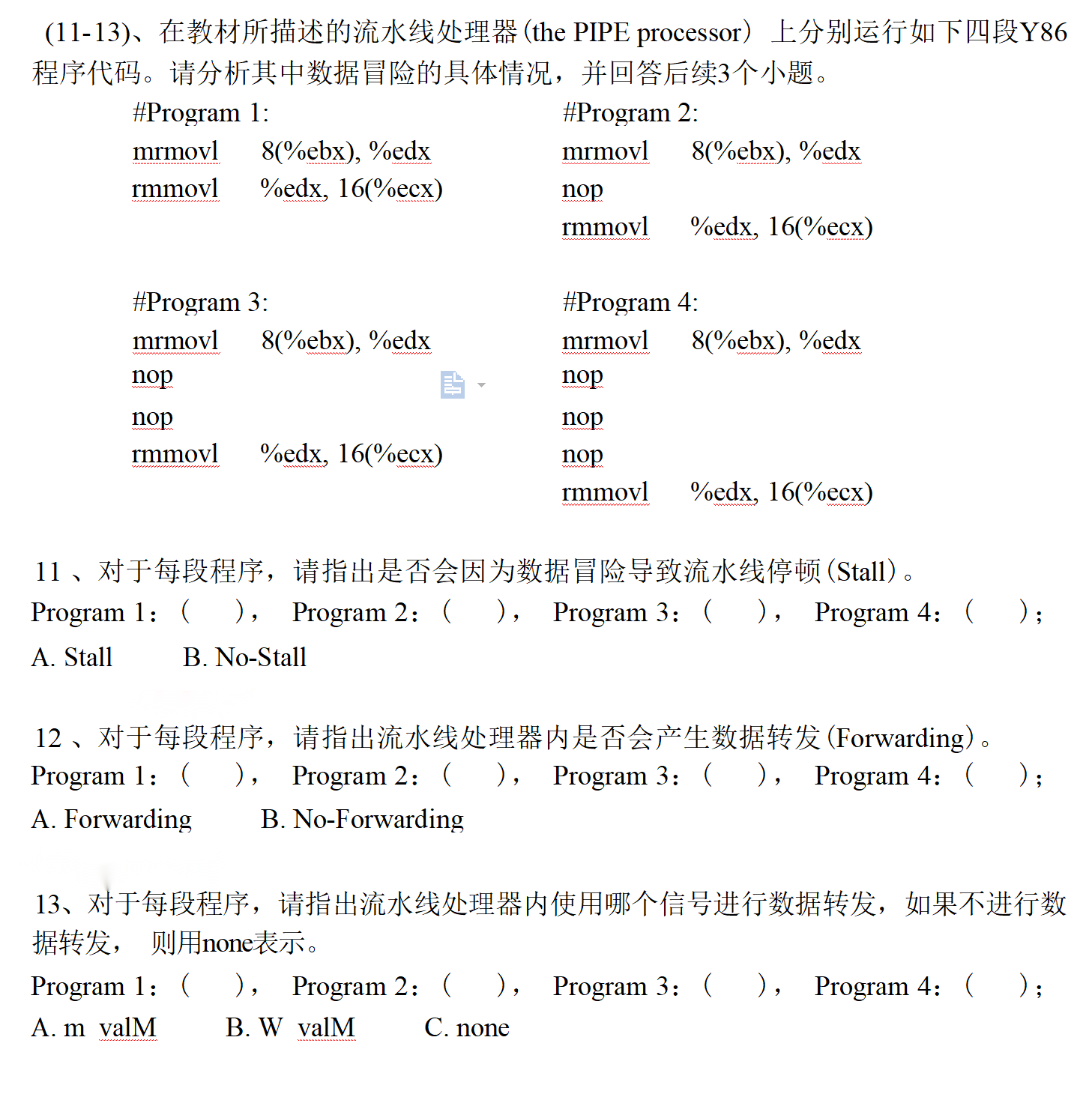
A. 流水线技术不仅能够提高执行指令的吞吐率，还能减少单条指令的执行时间。

B. 不断加深流水线级数，总能获得性能上的提升。

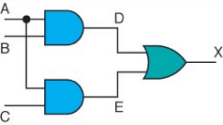
C. 流水级划分应尽量均衡，吞吐率会受到最慢的流水级影响。

D. 指令间的数据相关可能会引发数据冒险，可以通过数据转发或暂停流水线来解决。

（13期中）2.



（14期中）3.对应下述组合电路的正确 HCL 表达式为



A．Bool X = (A || B) && (A || C)

B．Bool X = A || (B && C)

C．Bool X = A && (B || C)

D．Bool X = A || B || C

（14期中）4.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 2ns、 2ns 和 1ns，则此处理器不停顿地执行完毕 10 条指令需要的时间为：

A．21ns

B．22ns

C．23ns

D．24ns

（14期中）5.关于 RISC 和 CISC 的描述，正确的是：

A. CISC 指令系统的指令编码可以很短， 例如最短的指令可能只有一个字节， 因此 CISC 的取指部件设计会比 RISC 更为简单。

B. CISC 指令系统中的指令数目较多，因此程序代码通常会比较长；而 RISC 指令系统中通常指令数目较少， 因此程序代码通常会比较短。

C. CISC 指令系统支持的寻址方式较多，RISC 指令系统支持的寻址方式较少， 因此用 CISC 在程序中实现访存的功能更容易。

D. CISC 机器中的寄存器数目较少，函数参数必须通过栈来进行传递； RISC 机器中的寄存器数目较多，只需要通过寄存器来传递参数。

答：( )

（14期中）6.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。

（15期中）7. 下面有关指令系统设计的描述正确的是：

A. 采用 CISC 指令比 RISC 指令代码更长。

B. 采用 CISC 指令比 RISC 指令运行时间更短

C. 采用 CISC 指令比 RISC 指令译码电路更加复杂

D. 采用 CISC 指令比 RISC 指令的流水线吞吐更高

（15期中）8.一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

（15期中）9. 关于流水线技术的描述， 错误的是：

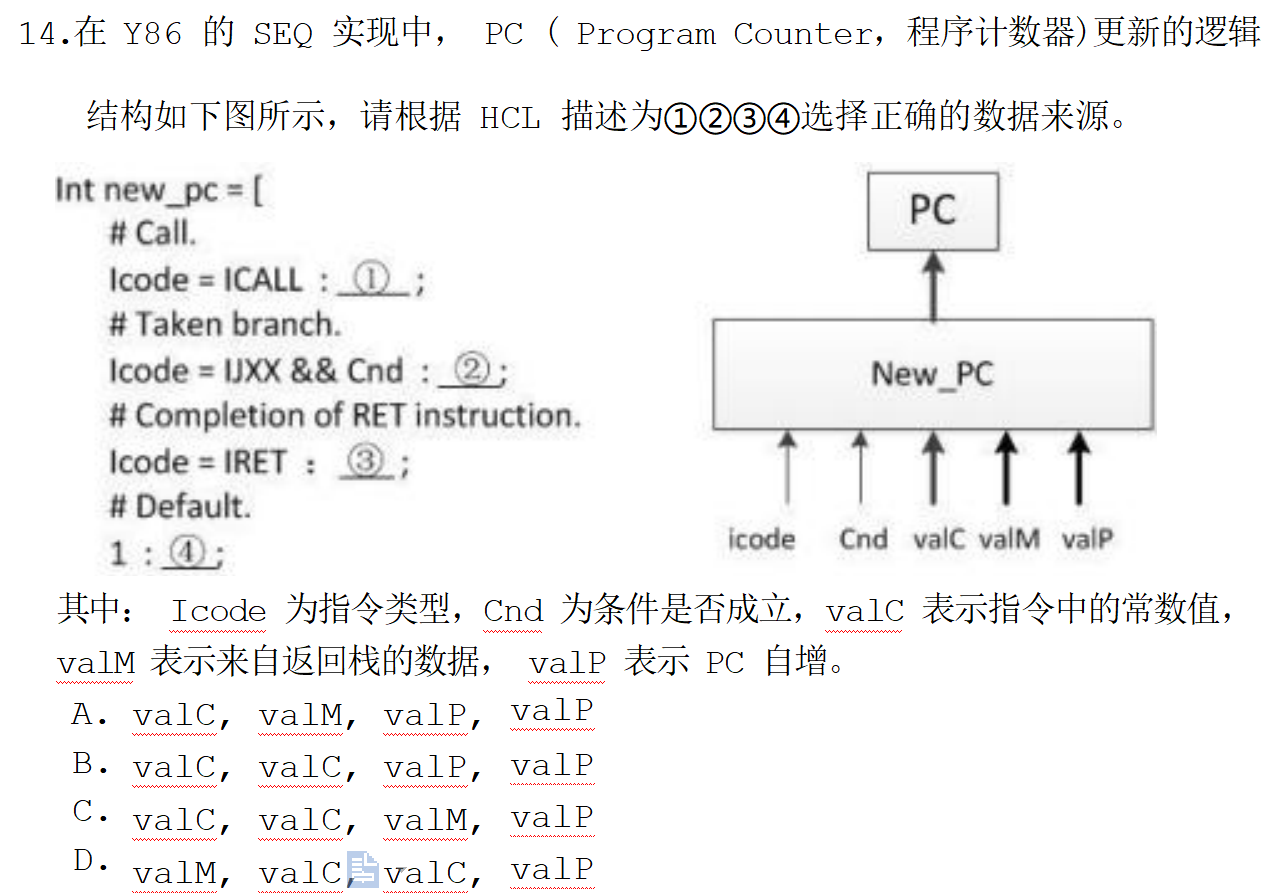
A. 流水线技术能够提高执行指令的吞吐率， 但也同时增加单条指令的执行时 间。

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。

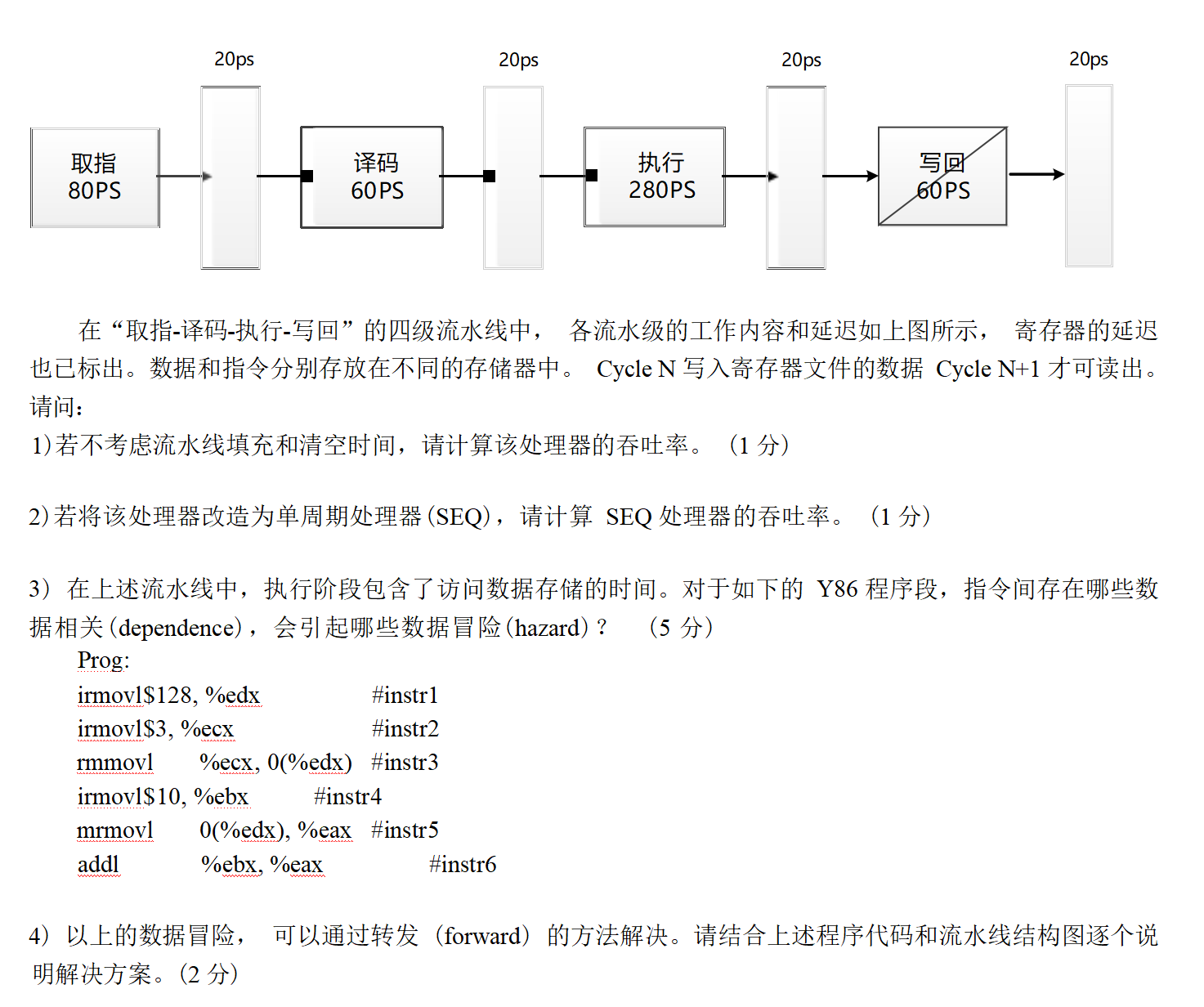
D. 现代处理器支持一个时钟内取指、执行多条指令，会增加控制冒险的开销。

（15期中）10.

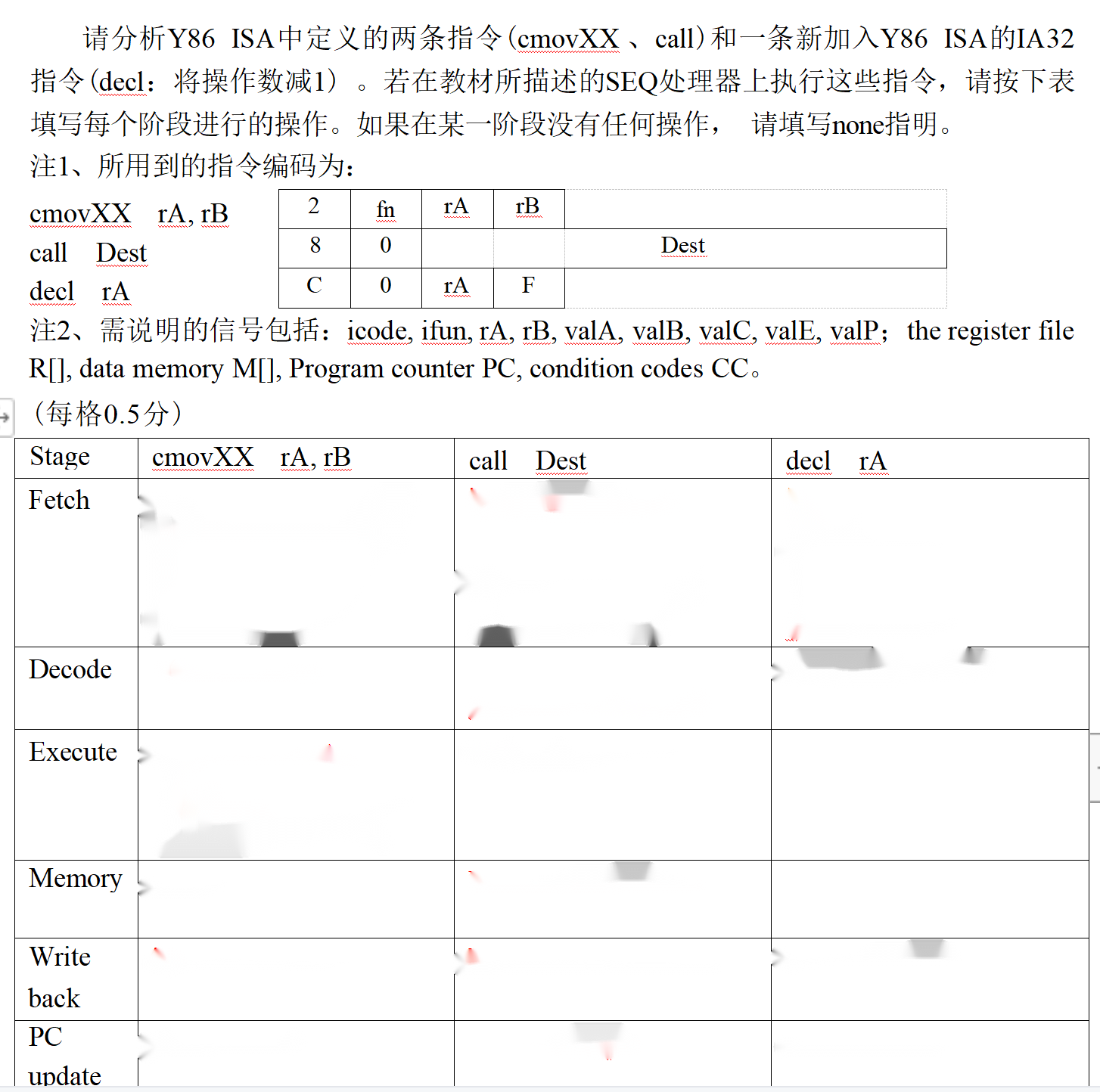


1. 填空与简答题

（13期中）1.



（13期中）2.



（14期中）3.

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。

|  |  |  |  |
| --- | --- | --- | --- |
| C | fn | rA | rB |

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 | icode:ifun ← M1 [PC]  rA:rB ← M1 [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 |

（15期中）4.

请分析Y86 ISA中新加入的一条指令：NewJE，其格式如下。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 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。

|  |  |
| --- | --- |
| Stage | NewJE rA, rB, Dest |
| Fetch | icode:ifun  M1 [PC]  rA:rB  M1 [PC+1]  valC  M4 [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 |

2.若在教材所描述的PIPE处理器上执行NewJE指令，如果跳转条件不满足， 一共会错误执行\_\_2\_\_条指令。

为了减小错误预测的代价，现将教材所描述的PIPE处理器做如下改进：在 Decode阶段增加一个比较器，用于判断(R[rA] = R[rB])条件，比较器的输 出信号为d\_equal。如果相等， 则d\_equal = 1，反之 d\_equal = 0。

此时，如果执行NewJE指令时跳转条件不满足，一共会错误执行\_\_1\_\_条指令。 (

3.在教材所描述的PIPE处理器上执行JXX指令时，发生转移预测错误的判断条件 和各级流水线寄存器的控制信号如下所示：

|  |  |
| --- | --- |
| Condition | Trigger |
| Mispredicted Branch | E\_icode = IJXX & !e\_Cnd |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Condition | F | D | E | M | W |
| Mispredicted Branch | normal | bubble | bubble | normal | normal |

在第 (2) 小题所述的改进后的处理器上执行 NewJE 指令，发生转移预测错误的判 断条件和各级流水线寄存器的控制信号应如何设置？

|  |  |
| --- | --- |
| Condition | Trigger |
| Mispredicted Branch | D\_icode = INewJE & !d\_equal |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Condition | F | D | E | M | W |
| Mispredicted Branch | normal | bubble | normal | normal | normal |

4.在第 (2) 小题所述的改进后的处理器上执行如下代码，

0x000: mrmovl 0 (%eax), %edx

0x006: 0x00c: 0x012:

0x013: 0x014: 0x015: 0x016: 0x01c: 0x022:

NewJE %edx, %eax, t

irmovl $1, %eax # Fall through

nop

nop

nop

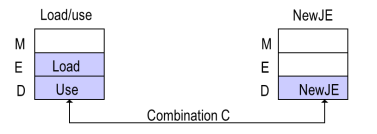
halt

t:irmovl $3, %edx # Target (Should not execute)

irmovl $4, %ecx # Should not execute

irmovl $5, %edx # Should not execute

会发生 load-use 和 misprediction 组合的 hazard 情况, 如下图所示



请问此时，各级流水线寄存器的控制信号应如何设置？

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Condition | F | D | E | M | W |
| Combination C | stall | stall | bubble | normal | normal |

5.在教材 PIPE 处理器设计中，data memory 实际是高速缓存 (cache)。假设 在执行上述 (4)中代码时， 0x000 指令中的 0(%eax)地址中的数据不在 data memory 中， 则 data memory 会将输出信号 m\_datamiss 置为 1，直到数据从 内存中取回到 data memory，再将 m\_datamiss 置为 0。(m\_datamiss 的默 认值为 0)

这种情况的判断条件如下，请问各级流水线寄存器的控制信号应如何设置？

|  |  |
| --- | --- |
| Condition | Trigger |
| Data Miss | M\_icode in { IMRMOVL, IPOPL } && m\_datamiss |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Condition | F | D | E | M | W |
| data miss | stall | stall | stall | stall | bubble |

1. 答案与解析：

选择：

1.CD

2.(1)答案： A, B, B, B

(2)答案: A,A,A,B

(3)答案: A,A,B,C

1. **C**

4. **D**

**2+2+2\*10** **=** **24**

5 C

考查对 CISC 和 RISC 基本特点的描述， A 和 B 都是描述反了， D 则是太绝对， RISC

也有可能用栈来传递参数。

6. **A**

说明： **B** 会增加单条指令的执行时间， **C** 可能会降低性能， **D** 和控制冒险没有关系

1. C ，CISC 的比 RISC 代码复杂(如不定长)，因此译码也更复杂，优点是 代码更短。运行时间和吞吐都谁更高要依据实际应用

8.B，主要考察延时和吞吐的区别，延时最小是 logic + 1 个 register， 吞吐最高时是把 logic 划分成无穷多份， 每一级需要 20ps, 1000/20ps =

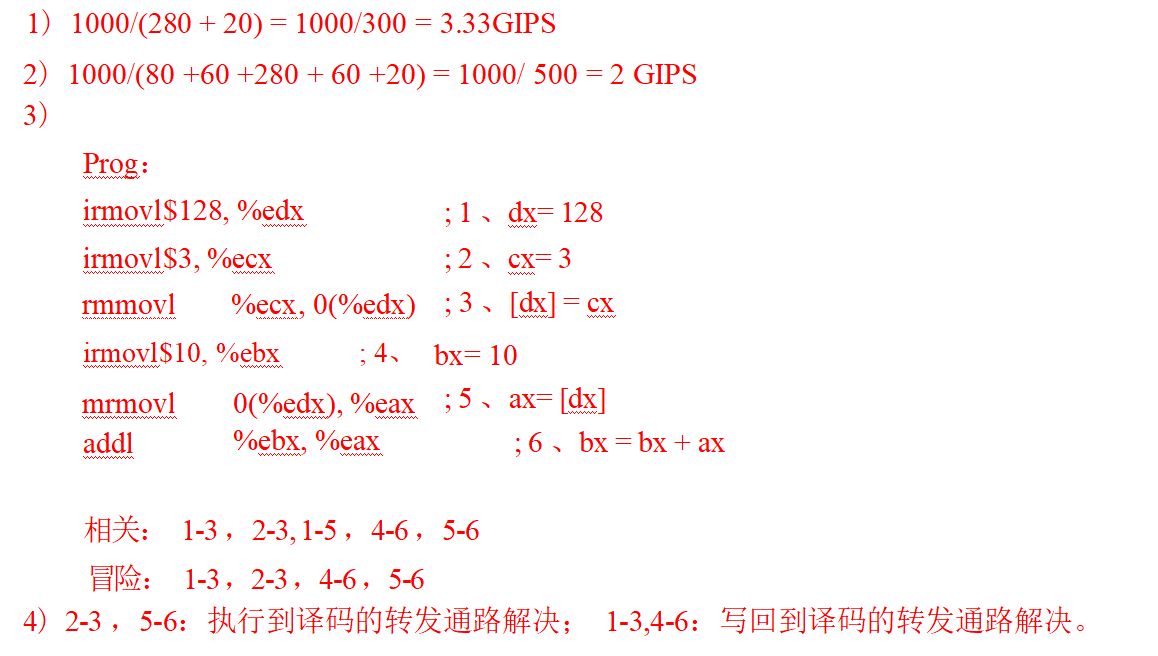
50GIPS

9.C，load-use 冒险 不能通过 data forwarding 来解决

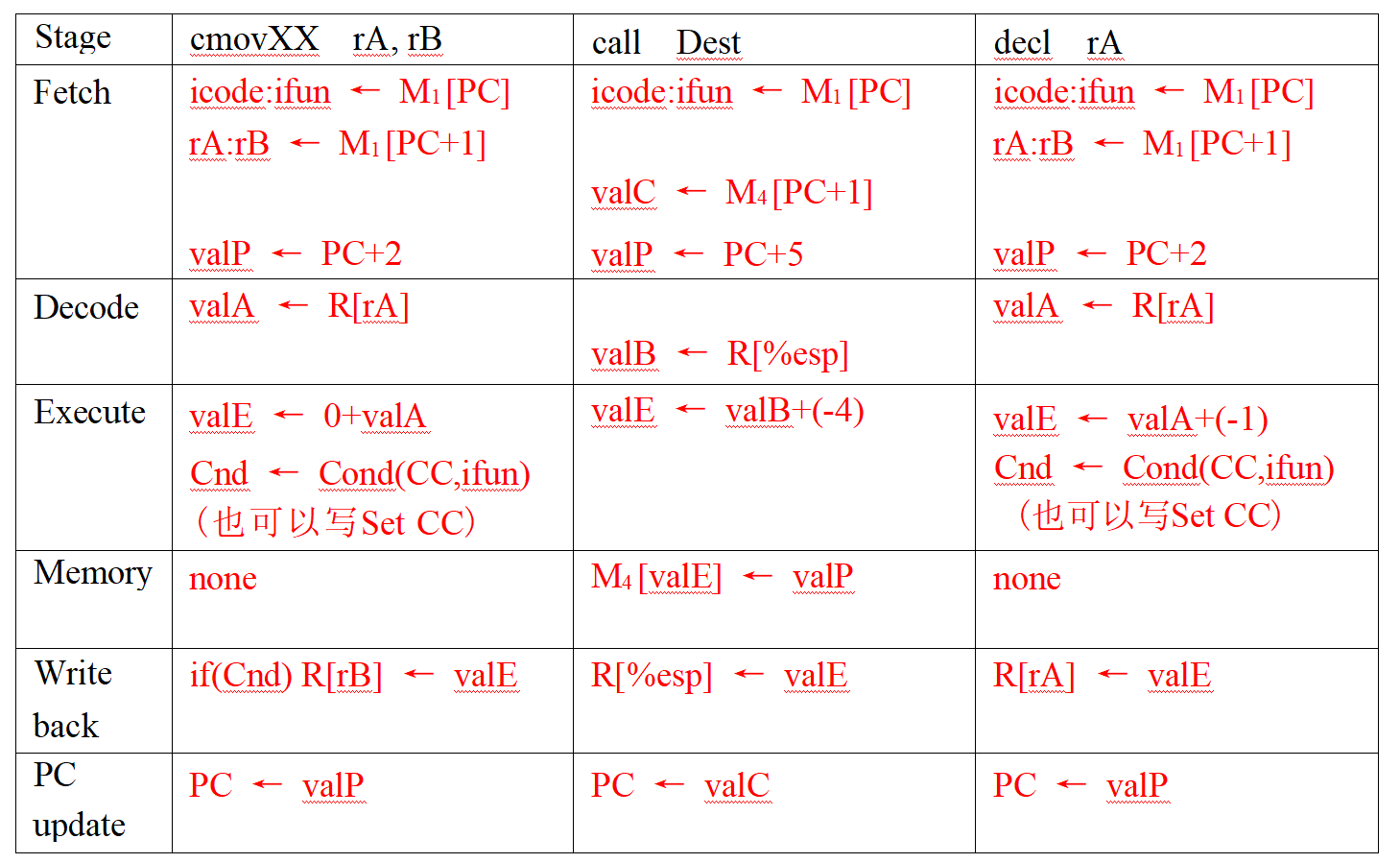
10.C

填空与解答：

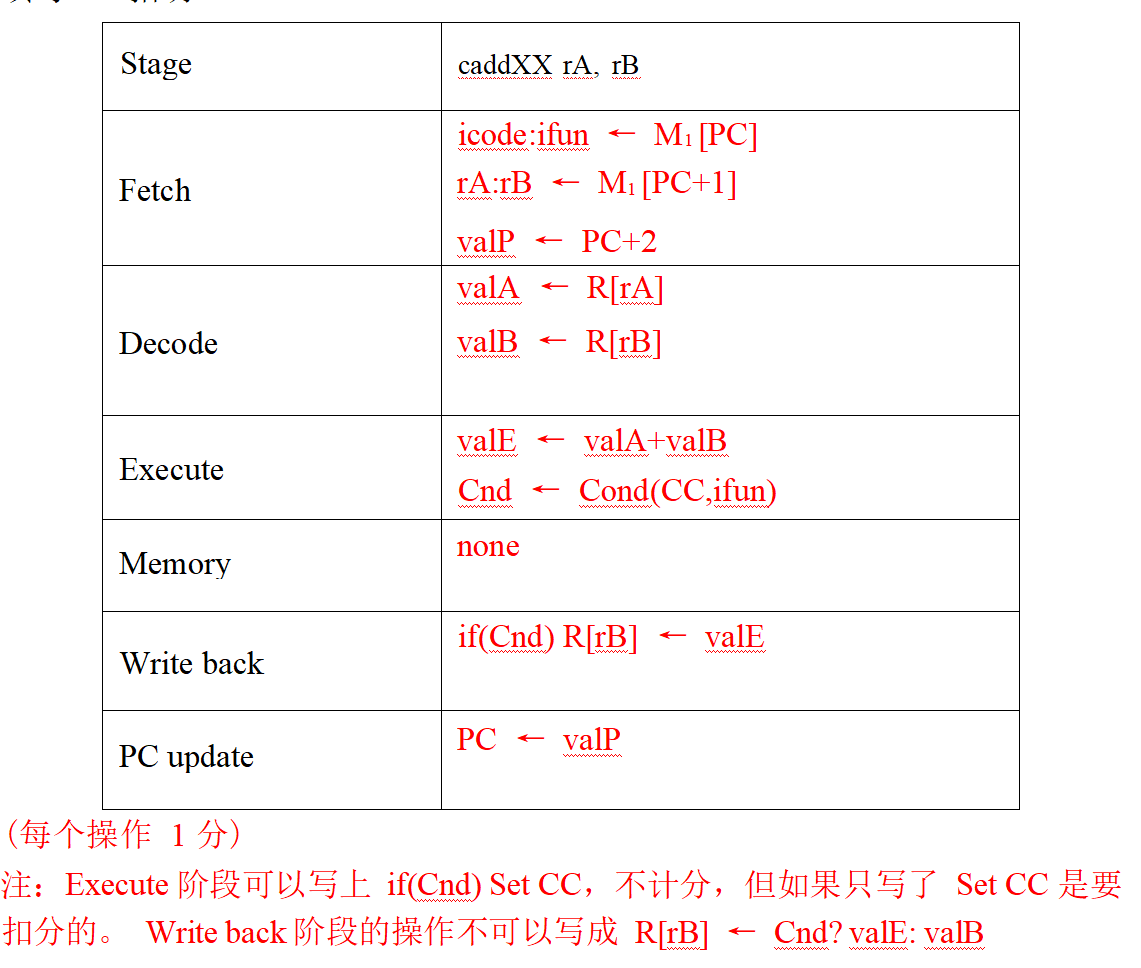
1.



2.



3.



4.请分析Y86 ISA中新加入的一条指令：NewJE，其格式如下。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 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。

|  |  |
| --- | --- |
| Stage | NewJE rA, rB, Dest |
| Fetch | icode:ifun  M1 [PC]  rA:rB  M1 [PC+1]  valC  M4 [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 |

(红字处每行 1 分，共 5 分)

2.若在教材所描述的PIPE处理器上执行NewJE指令，如果跳转条件不满足， 一共会错误执行\_\_2\_\_条指令。

为了减小错误预测的代价，现将教材所描述的PIPE处理器做如下改进：在 Decode阶段增加一个比较器，用于判断(R[rA] = R[rB])条件，比较器的输 出信号为d\_equal。如果相等， 则d\_equal = 1，反之 d\_equal = 0。

此时，如果执行NewJE指令时跳转条件不满足，一共会错误执行\_\_1\_\_条指令。 (每空 1 分， 共 2 分)

3.在教材所描述的PIPE处理器上执行JXX指令时，发生转移预测错误的判断条件 和各级流水线寄存器的控制信号如下所示：

|  |  |
| --- | --- |
| Condition | Trigger |
| Mispredicted Branch | E\_icode = IJXX & !e\_Cnd |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Condition | F | D | E | M | W |
| Mispredicted Branch | normal | bubble | bubble | normal | normal |

在第 (2) 小题所述的改进后的处理器上执行 NewJE 指令，发生转移预测错误的判 断条件和各级流水线寄存器的控制信号应如何设置？

|  |  |
| --- | --- |
| Condition | Trigger |
| Mispredicted Branch | D\_icode = INewJE & !d\_equal |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Condition | F | D | E | M | W |
| Mispredicted Branch | normal | bubble | normal | normal | normal |

(每空 1 分， 共 5 分)

4.在第 (2) 小题所述的改进后的处理器上执行如下代码，

0x000: mrmovl 0 (%eax), %edx

0x006: 0x00c: 0x012:

0x013: 0x014: 0x015: 0x016: 0x01c: 0x022:

NewJE %edx, %eax, t

irmovl $1, %eax # Fall through

nop

nop

nop

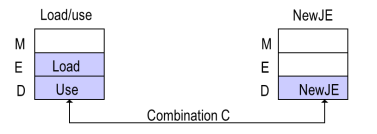
halt

t:irmovl $3, %edx # Target (Should not execute)

irmovl $4, %ecx # Should not execute

irmovl $5, %edx # Should not execute

会发生 load-use 和 misprediction 组合的 hazard 情况, 如下图所示



请问此时，各级流水线寄存器的控制信号应如何设置？

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Condition | F | D | E | M | W |
| Combination C | stall | stall | bubble | normal | normal |

(每空 1 分， 共 3 分)

5.在教材 PIPE 处理器设计中，data memory 实际是高速缓存 (cache)。假设 在执行上述 (4)中代码时， 0x000 指令中的 0(%eax)地址中的数据不在 data memory 中， 则 data memory 会将输出信号 m\_datamiss 置为 1，直到数据从 内存中取回到 data memory，再将 m\_datamiss 置为 0。(m\_datamiss 的默 认值为 0)

这种情况的判断条件如下，请问各级流水线寄存器的控制信号应如何设置？

|  |  |
| --- | --- |
| Condition | Trigger |
| Data Miss | M\_icode in { IMRMOVL, IPOPL } && m\_datamiss |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Condition | F | D | E | M | W |
| data miss | stall | stall | stall | stall | bubble |

(每空 1 分， 共 5 分）