《计算机组成原理》第六次作业

网络空间安全学院 信息安全 陆皓喆 2211044

4.16

本练习讨论流水线对处理器时钟周期的影响。假设数据通路中各流水级的延迟如下：

IF	ID	EX	MEM	WB
250ps	350ps	150ps	300ps	200ps

另外，假定处理器中各种指令的使用频度如下：

ALU/Logic	Jump/Branch	Load	Store
45%	20%	20%	15%

4.16.1

题目

流水线处理器与非流水线处理器的时钟周期分别是多少？

解答

- 流水线：350ps
- 非流水线：1250ps

4.16.2

题目

lw指令在流水线处理器和非流水线处理器中的总延迟分别是多少？

解答

- 流水线：1750ps
- 非流水线：1250ps

4.16.3

题目

如果可以将原流水线数据通路的一级拆分为两级，每级的延迟是原级的一半，那么你会选择哪一级进行拆分？拆分后处理器的时钟周期为多少？

解答

选择分拆分ID字段，拆分后处理器的时钟周期是300ps

4.16.4

题目

假设没有阻塞和冒险，数据存储器的利用率是多少（占总周期数的百分比）？

解答

利用率是35%

4.16.5

题目

假设没有阻塞和冒险，寄存器堆的写寄存器端口的利用率是多少？

解答

利用率是65%

4.19

题目

假设寄存器 $ s0的初值为11， $ s1的初值为22。在4.6节所述的不处理数据冒险（即程序员需要在必要的地方插入NOP指令来处理数据冒险）的流水线中执行下面的代码， 寄存器 $ s5的最终结果分别是多少？假设寄存器堆在时钟周期开始时写，在时钟周期结束时读，那么，ID级可以返回WB级在同一个是时钟周期内产生的结果。4.8节和图4-51有详细的说明。

```
addi $s0,$s1,5
add $s2,$s0, $s1
addi $s3,$s0,15
add $s4, $s0,$s0
```

解答

最终结果为54

代码将正确运行，因为第一条指令的结果在第5个周期开始时被写回寄存器堆，而最后一条指令在该周期的后半部分读取x1的更新值。

4.22

考虑下面的MIPS汇编代码段：

```
sd $s5,12($s3)
Id $s5,8($s3)
sub $s4, $s2, $s1
beqz $s4, label
add $s2, $s0, $s1
sub $s2,$s6,$s1
```

假设我们可以修改流水线，只保留一个存储器（指令和数据共用一个存储器）。在这种情况下，如果同一个时钟周期内同时取指令和访问数据，将产生结构冒险。

4.22.1

题目

画出上述代码的流水线图，展示是否产生阻塞。

解答

```
sd  x29, 12(x16)  IF ID EX ME WB
ld  x29, 8(x16)      IF ID EX ME WB
sub x17, x15, x14       IF ID EX ME WB
bez x17, label             ** ** IF ID EX ME WB
add x15, x11, x14                 IF ID EX ME WB
sub x15,x30,x14                      IF ID EX ME WB
```

4.22.2

题目

一般来说，重排代码能否减少阻塞或NOP指令的数量?

解答

重新排序代码是没有用的。每条指令都必须执行。

因此，每次数据访问都会导致停机，重新排序代码，只会改变有冲突的指令对

4.22.3

题目

上述结构冒险必须使用硬件处理吗？数据冒险能够通过添加NOP指令消除，结构冒险能使用相同的方法处理吗？如果能，给出解决方案；如果不能，请解释原因。

解答

是的。

我们使用nop无法解决这种结构冒险，因为即使是nop也必须从指令存储器中获取

4.22.4

题目

该结构冒险将导致一个典型的程序产生多少阻塞（使用练习题4.8中给出的指令组合）？

解答

会产生35%的阻塞，每一次数据的访问都会导致失速。

4.26

本练习讨论流水线处理器中旁路的成本 / 复杂度 / 性能之间的折中。参考图4—43的流水线数据通路，假设指令中有部分存在RAW（read after write，写后读）数据相关。RAW数据相关根据生成结果的流水级（EX或MEM）和使用结果的下一条指令（产生结果的指令后紧跟的第1条或第2条指令）检测。假设在时钟周期的前半部分写寄存器，在后半部分读寄存器，这样“EX到第3条指令”和“MEM到第3条指令”的相关不会产生数据冒险。最后假设无数据冒险时处理器的CPI为1。

EX to 1st only	MEM to 1st only	EX to 2nd only	MEM to 2nd only	EX to 1st and EX to 2nd
5%	20%	5%	10%	10%

假定各级流水线延迟如下。其中EX级给出了不同旁路情况下的延迟。

IF	ID	EX(no FW)	EX(full FW)	EX(FW from EX/MEM only)	EX(FW from MEM/WB only)	MEM	WB
120ps	100ps	110ps	130ps	120ps	120ps	120ps	100ps

4.26.1

题目

对于上面列出的每种RAW相关，给出至少3条汇编语句来表示该相关。

解答

```
// EX to 1st only:
add x11, x12, x13
add x14, x11, x15
add x5, x6, x7

// MEM to 1st only:
ld x11, 0(x12)
add x15, x11, x13
add x5, x6, x7

// EX to 2nd only:
add x11, x12, x13
add x5, x6, x7
add x14, x11, x12

// MEM to 2nd only:
ld x11, 0(x12)
add x5, x6, x7
add x14, x11, x13

// EX to 1st and EX to 2nd:
add x11, x12, x13
add x5, x11, x15
add x16, x11, x12
```

4.26.2

题目

对于上面列出的每种RAW相关，需要在练习题4.26.1中你所提供的代码中插入多少NOP指令，才能在没有旁路机制或冒险检测的流水线中正确执行？给出NOP指令插入的位置。

解答

```
// EX to 1st only: 2 nops
add x11, x12, x13
nop
nop
add x14, x11, x15
add x5, x6, x7

// MEM to 1st only: 2 stalls
ld x11, 0(x12)
nop
nop
add x15, x11, x13
add x5, x6, x7

// EX to 2nd only: 1 nop
add x11, x12, x13
add x5, x6, x7
nop
add x14, x11, x12

// MEM to 2nd only: 1 nop
ld x11, 0(x12)
add x5, x6, x7
nop
add x14, x11, x13

// EX to 1st and EX to 2nd: 2 nops
add x11, x12, x13
nop
nop
add x5, x11, x15
add x16, x11, x12
```

4.26.3

题目

如果在没有旁路机制或冒险检测的流水线中执行程序时，对每条指令独立进行分析，计算需要额外执行的NOP指令的数量。写一个3条汇编语句组成的代码序列，当单独分析每条指令时，插入的阻塞数将比该序列为了避免数据相关实际所需要的阻塞数多。

解答

我们使用下面的代码：

```
ld  x11, 0(x5) # MEM to 2nd --- one stall
add x12, x6, x7 # EX to 1st  --- two stalls
add x13, x11, x12
add x28, x29, x30
```

如果我们单独分析每条指令，我们将计算出我们需要增加3个stall(一个用于“MEM到第2位”，两个用于“EX到第1位”)。然而，如下图所示，我们只需要两个stall:

```
ld x11, 0(x5)
add  x12, x6, x7
nop
nop
add  x13, x11, x12
add  x28, x29, x30
```

4.26.4

题目

假设没有其他冒险，上表中的程序在没有旁路的流水线中运行时，CPI是多少？阻塞的时钟周期所占的比例是多少？（为了简化，假设所有必须的条件都在上表中列出，并都能单独进行分析处理。）

解答

对4.26.2的答案进行加权平均，得到CPI为1.85时，每条指令(平均)的stall为`0.05*2 + 0.2*2 + 0.05*1 + 0.1*1 + 0.1*2`。这意味着0.85/1.85个周期，结果是46%，所以是失速。

4.26.5

题目

如果采用全旁路机制（旁路转发所有能够转发的结果），CPI是多少？阻塞的时钟周期所占比例是多少？

解答

唯一不能通过转发处理的依赖关系是从MEM阶段到下一条指令。因此，当CPI为1.2时，20%的指令将产生一个失速。这意味着1.2个周期中有0.2个，即17%是停滞。

4.26.6

题目

假设无法提供全旁路机制所需的三输入多路选择器。现在需要判断只从EX / MEM流水线寄存器进行转发（下一时钟周期转发）的效果好，还是只从MEM / WB流水线寄存器转发（两个周期后转发）的效果好。每种选择的CPI是多少？

解答

如果我们只从EX/MEM登记册转发，我们有以下stall/nop:

- EX to 1st: 0
- MEM to 1st: 2
- EX to 2nd: 1
- MEM to 2nd: 1
- EX to 1st and 2nd: 1

这表示平均为`0.05*0 + 0.2*2 + 0.05*1 + 0.10*1 + 0.10*1 = 0.65`个stall/nop。

因此，CPI为1.65。

如果我们只从MEM/WB转发，我们有以下stalls/nop:

- EX to 1st: 1
- MEM to 1st: 1
- EX to 2nd: 0
- MEM to 2nd: 0
- EX to 1st and 2nd: 1

这表示平均`0.05*1 + 0.2*1 + 0.1*1 = 0.35`个stalls/nop。

因此，CPI为1.35。

4.26.7

题目

对于给出的冒险发生的可能性和各个流水级的延迟，相对于无旁路机制的流水线，每类旁路机制（EX / MEM、MEM / WB或全旁路）获得的加速比分别是多少？

解答

	No forwarding	EX/MEM	MEM/WB	Full Forwarding
CPI	1.85	1.65	1.35	1.2
Period	120	120	120	130
Time	222n	198n	162n	156n
Speedup	-	1.12	1.37	1.42

4.26.8

题目

如果采用“穿越”转发机制来消除所有的数据相关，与练习题4.26.7提供的最快处理器相比，此时获得的额外的加速比是多少？假设这种带实现的“穿越”转发机制使得全转发的EX级的延迟增加100ps。

解答

完全转发时CPI为1.2

“穿越”转发时CPI为1.0

完全转发时CPI为130时钟周期

“穿越”转发时CPI为230时钟周期

$$Speedup = (1.2 \times 130)/(1 \times 230) = 0.68$$

这意味着“穿越”转发实际上使CPU变慢了

4.26.9

题目

冒险类型表区分了“EX to 1st”以及“EX to 1st and EX to 2nd”的入口。为什么没有“MEM to 1st and MEM to 2nd”的入口?

解答

考虑4.26.6中的“EX/MEM”转发时，“EX to 1st”不产生stall，而“EX to 1st and EX to 2nd”产生一个stall。

但是，“MEM to 1st"和“MEM to 1st and MEM to 2nd”将始终生成相同数量的stall。(无论转发的类型如何，所有“MEM to 1st”依赖都会导致停机。这导致第二个指令的ID阶段与基本指令的WB阶段重叠，在这种情况下不需要转发。)

4.27

本题用到下面的指令序列。假设该指令序列在一个5级流水线的数据通路中执行：

```
add $s3, $s1, $s0
lw $s2,4($s3)
lw $s1,0($s4)
or $s2,$s3,$s2
sw $s2,0($s3)
```

4.27.1

题目

如果没有旁路或者冒险检测，插入NOP指令以保证该序列能够正确执行。

解答

```
add x15, x12, x11
nop
nop
ld x13, 4(x15)
ld x12, 0(x2)
nop
or x13, x15, x13
nop
nop
sd x13, 0(x15)
```

4.27.2

题目

现在，修改或重排该指令序列，来将需要的NOP指令减到最少。假设在修改后的代码中，寄存器 $ t0用于存储临时变量。

解答

我们做不到减少nop指令的数量

4.27.3

题目

假设处理器有旁路机制，但忘记了实现冒险检测单元，那么题目提供的原始代码在执行时会发生什么情况?

解答

代码正确执行。只有当加载后的指令使用加载的结果时，我们才需要危险检测来插入一个失速，但是这种情况不会发生。

4.27.4

题目

假设有旁路机制，在代码执行的前7个时钟周期，图4—59中的旁路单元和冒险检测单元在每个时钟周期将发出哪些信号?

解答

Cycle	1	2	3	4	5	6	7	8	
add	IF	ID	EX	MEM	WB				
ld		IF	ID	EX	MEM	WB			
ld			IF	ID	EX	MEM	WB		
or				IF	ID	EX	MEM	WB	
sd					IF	ID	EX	MEM	WB

因为在这段代码中没有停顿，PCWrite和IF/IDWrite总是1,ID/EX之前的mux总是设置为传递控制值。

- (1) ForwardA = X;
- (2) ForwardA = X;
- (3) ForwardA = 0;ForwardB = 0(无转发)
- (4) ForwardA = 2;ForwardB = 0(基寄存器取自上一条指令的结果)
- (5)ForwardB = 1(基寄存器取自前两条指令的结果)
- (6)ForwardA = 0;ForwardB= 2 (rs1 = x15从寄存器堆中获取;rs2 = x13，取自第一条ld指令的结果)
- (7)ForwardA = 0;ForwardB = 2(从寄存器堆中获取的基本寄存器，要写入的指令取自先前的指令)

4.27.5

题目

如果没有旁路机制，图4—59中的冒险检测单元需要哪些新的输入和输出信号？用本题提供的指令作为例子，解释每个信号增加的原因。

解答

危害检测单元还需要从MEM/WB寄存器中输出的rd值。如果当前处于ID阶段的指令依赖于由EX中的指令或MEM阶段中的指令产生(或从该指令转发)的值，则需要停止该指令。所以我们需要检查这两条指令的目的寄存器。Hazard单元已经有来自EX/MEM寄存器的rd值作为输入，所以我们只需要添加来自MEM/WB寄存器的值。

不需要额外的输出。我们可以使用我们已经拥有的三个输出信号来停止流水线。

EX/MEM中的rd值用于检测add和下一条指令之间的数据冒险，MEM/WB中的rd值用于检测第一条ld指令和or指令之间的数据冒险。

4.27.6

题目

对于练习题4.26.5提供的新的冒险检测单元，说明指令序列执行的前5个时钟周期都发出了哪些输出信号。

解答

Cycle	1	2	3	4	5	6
add	IF	ID	EX	MEM	WB	
ld		IF	ID	-	-	EX
ld			IF	-	-	ID

- (1) PCWrite = 1; IF/IDWrite = 1; control mux = 0
- (2) PCWrite = 1; IF/IDWrite = 1; control mux = 0
- (3) PCWrite = 1; IF/IDWrite = 1; control mux = 0
- (4) PCWrite = 0; IF/IDWrite = 0; control mux = 1
- (5) PCWrite = 0; IF/IDWrite = 0; control mux = 1