《计算机组成原理》第七次作业

网络空间安全学院 信息安全 陆皓喆 2211044

4.21

考虑4.6节所述的无法处理数据冒险的流水线（即程序员需要在必要的地方插入NOP指令来处理器数据冒险）。假设（优化之后）一个典型的n条指令的程序需要增加$0.4n$条NOP指令来处理数据冒险。

4.21.1

题目

假设该流水线（没有旁路）的时钟周期是$250ps$，并假设增加旁路硬件可以将NOP指令的数量从$0.4n$减少到$0.05n$，但时钟周期会增加到$300ps$。和原先的流水线相比，新流水线的加速比是多少？

解答

不转发的流水线需要$1.4 \times n \times 250ps$。带转发的流水线需要$1.05 \times n \times 300ps$。因此加速比是

$$\frac{1.4 \times 250}{1.05 \times 300} = 1.11$$

4.21.2

题目

不同的程序需要不同数量的NOP指令。当一个典型程序运行在带旁路的流水线之前，程序中还有多少NOP指令（用指令条数的百分比表示）？

解答

我们的目标是让有转发的流水线比没有转发的流水线更快。我们将y设为剩余$stall$占"$code$"指令的百分比。我们的目标是

$$300 \times (1 + y) \times n < 250 \times 1.4 \times n$$

因此，y必须小于16.7%。

4.21.3

题目

重做练习题$4.21.2$，但使用x代表相对于n的NOP指令的数量（练习题$4.21.2$中，x为0.4），答案用x表示。

解答

我们只需要满足以下条件就可以实现流水线的加速。

$$300 \times (1+y) \times n < 250 \times (1+x) \times n$$

只需要当

$$y < \frac{250x - 50}{300}$$

就可以符合题目的条件了。

4.21.4

题目

一个只用$.075n$条NOP指令的程序能否在带旁路的流水线中执行得更快？请解释原因。

解答

不能。在最好的情况下，转发消除了对每个NOP的需要，程序将花费$300 \times n$的时间在具有转发的管道上运行。这比没有转发的管道上所需的$250 \times 1.075 \times n$要慢。

4.21.5

题目

如果想要程序在有旁路的流水线中执行得更快，最少需要有多少NOP指令（用指令条数的百分比表示）？

解答

当4.21.3的解小于0时，不可能实现加速。

因为，对于

$$\frac{250x - 50}{300} > 0$$

必须满足$x > 0.2$才行，但是很明显不符合。

4.25

考虑下面的循环：

```
LOOP: ld $s0,0($s3)
      ld $s1,8($s3)
      add $s2,$s0,$s1
      addi $s3,$s3,-16
      bnez $s2, LOOP
```

假设使用了完美的分支预测（即没有控制相关引起的阻塞），没有延迟槽，流水线支持完全旁路机制，分支处理在EX级（不是在ID级）进行。

4.25.1

题目

画出该循环前两次迭代的流水线执行图。

解答

以下内容中，“...”表示失速，“！”表示不进行有用的工作阶段。

```
ld x10, 0(x13)     IF ID EX ME | WB
ld x11, 8(x13)        IF ID EX | ME WB
add x12, x10, x11        IF ID | .. EX ME! WB
addi x13, x13, -16          IF | .. ID EX  ME! WB
bnez x12, LOOP                 | .. IF ID  EX  ME! WB!
ld x10, 0(x13)                        IF  ID  EX  ME  WB
ld x11, 8(x13)                            IF  ID  EX  ME WB
add x12, x10, x11                             IF  ID  .. EX | ME! WB
addi x13, x13, -16                                IF  .. ID | EX  ME! WB
bnez x12, LOOP                                        IF | ID  EX  ME! WB!
Completely busy                | N  N  N   N   N   N   N  N  |
```

4.25.2

题目

标记出做无用功的流水级。流水线满负荷工作，某个时钟周期中5个流水级全部都在进行有效工作的频率是多少？（从$addi$指令位于IF级的时钟周期开始，到$bnez$指令位于IF级的时钟周期结束。）

解答

在一个特定的时钟周期中，如果一个流水线的阶段被停止，或者经过该阶段的指令不进行任何有用的工作，那么它就没有做有用的工作。如4.25.1所示，并不存在每个管道阶段都在做有用工作的循环。

4.31

本练习比较单发射和双发射处理器的性能，并考虑对双发射处理器进行程序优化。本题使用下面的循环（C语言编写）：

```
for(i=0;i!=j;i+=2)
    b[i]=a[i]-a[i+1];
```

进行少量优化或不进行优化的编译器可能产生下面这段MIPS汇编代码：

```
    li $s0,0
    jal ENT
TOP:sll $t0,$s0,3
    add $t1,$s2,$t0
    lw $t2,0($t1)
    lw $t3,8($t1)
    sub $t4,$t2,$t3
    add $t5,$s3,$t0
    sw $t4,0($t5)
    addi $s0,$s0,2
ENT:bne $s0,$s1,TOP
```

上面的代码用到了以下这些寄存器：

i	j	a	b	temporary values
`$s0`	`$s1`	`$s2`	`$s3`	`$t0-$t5`

假设本题中的双发射、静态调度处理器具有如下特性：

1．一条指令必须是存储器操作，另一条指令必须是算术 / 逻辑运算指令或一条分支指令。

2．处理器流水线的各级之间有所有可能的旁路路径（包括到ID级的用于解决分支问题的路径）。

3．处理器有完美的分支预测。

4．如果一条指令与另外一条指令相关，两条指令不会在同一个包中一起发射（见4.11.2节）。

5．如果需要阻塞，发射包中的两条指令都需要阻塞（见4.11.2节）。

做完这些练习你会发现，要获得达到近似最优加速比的代码需要做多少工作。

4.31.1

题目

画出一个流水线执行图，展示上述$MIPS$代码在双发射处理器上如何执行。假设两个迭代后循环退出。

解答

```
                1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28
li x12, 0       IF ID EX ME WB
jal ENT         IF ID .. EX ME WB

bne x12, x13, TOP  IF .. ID EX ME WB
slli x5, x12, 3    IF .. ID .. EX ME WB

add x6, x10, x5          IF .. ID EX ME WB
ld x7, 0(x6)             IF .. ID .. EX ME WB

ld x29, 8(x6)                  IF .. ID EX ME WB
sub x30, x7, x29               IF .. ID .. .. EX ME WB

add x31, x11, x5                     IF .. .. ID EX ME WB
sd x30, 0(x31)                       IF .. .. ID .. EX ME WB

addi x12, x12, 2                           IF .. ID EX ME WB
bne x12, x13, TOP                          IF .. ID .. EX ME WB

slli x5, x12, 3                                  IF .. ID EX ME WB
add x6, x10, x5                                  IF .. ID .. EX ME WB

ld x7, 0(x6)                                           IF .. ID EX ME WB
ld x29, 8(x6)                                             IF .. ID .. EX ME WB

sub x30, x7, x29                                             IF .. ID .. EX ME WB
add x31, x11, x5                                             IF .. ID .. .. EX ME WB

sd x30, 0(x31)                                                     IF .. .. ID EX ME WB
addi x12, x12, 2                                                   IF .. .. ID .. EX ME WB

bne x12, x13, TOP                                                            IF .. ID EX ME WB
slli x5, x12, 3                                                              IF .. ID .. EX ME WB
```

4.31.2

题目

从单发射处理器到双发射处理器的加速比是多少？（假设循环迭代数千次。）

解答

原始代码在1期机器上需要10个周期/循环，在2期机器上需要10个周期/循环。这不会带来净加速。但是我们为此增加了一倍的硬件。我们知道代码在有两个问题的机器上需要10个周期/迭代，因为循环1中的第一条指令($slli$)在周期6中开始执行，而迭代3中的第一条指令在周期26中开始执行，因此
$(26-6)/2=10$

```
    li x12,0
    jal ENT
TOP:
    slli x5, x12, 3
    add x6, x10, x5
    ld x7, 0(x6)
    ld x29, 8(x6)
    <stall>
    sub x30, x7, x29
    add x31, x11, x5
    sd x30, 0(x31)
```

```
    addi x12, x12, 2
ENT:
    bne x12, x13, TOP
```

4.31.3

题目

重排或重写上述$MIPS$代码，以便在双发射处理器上获得更好的性能。提示：如果$j=0$，使用指令$beqz\ \$s1$，$DONE$直接跳出循环。

解答

我们给出一个方案。

```
    beqz x13, DONE
    li x12, 0
    jal ENT
TOP:
    slli x5, x12, 3
    add x6, x10, x5
    ld x7, 0(x6)
    ld x29, 8(x6)
    addi x12, x12, 2
    sub x30, x7, x29
    add x31, x11, x5
    sd x30, 0(x31)
ENT:
    bne x12, x13, TOP
DONE:
```

如果我们切换到“基于指针”的方法，我们可以节省一个循环。

```
for (i = 0; i ! = j; i+ = 2) {
    *b = *a - *(a+1);
    b+=2;
    a+=2;
}
    bez x13, DONE
    li x12, 0
    jal ENT
TOP:
    ld x7, 0(x10)
    ld x29, 8(x10)
    addi x12, x12, 2
    sub x30, x7, x29
    sd x30, 0(x11)
    addi x10, x10, 16
    addi x11, x11, 16
ENT:
    bne x12,x13,TOP
DONE:
```

4.31.4

题目

重排或重写上述$MIPS$代码，以便在双发射处理器上获得更好的性能，但不要展开循环。

解答

我们不展开循环进行编写，结果如下所示。

```
    beqz x13, DONE
    li x12, 0
TOP:
    slli x5, x12, 3
    add x6, x10, x5
    ld x7, 0(x6)
    add x31, x11, x5
    ld x29, 8(x6)
    addi x12, x12, 2
    sub x30, x7, x29
    sd x30, 0(x31)
    bne x12, x13, TOP
DONE:
```

我们切换到基于指针来进行编写，结果如下所示。

```
for (i = 0; i ! = j; i+ = 2) {
    *b = *a - *(a+1);
    b+=2;
    a+=2;
}
    beqz x13, DONE
    li x12, 0
TOP:
    ld x7, 0(x6)
    addi x12, x12, 2
    ld x29, 8(x6)
    addi x6, x6, 16
    sub x30, x7, x29
    sd x30, 0(x31)
    bne x12, x13, TOP
DONE:
```

4.31.5

题目

重复练习题4.31.1，此次使用练习题4.31.4中的优化代码。

解答

```
                  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
beqz x13, DONE    IF ID EX ME WB
li x12, 0         IF ID .. EX ME WB

slli x5, x12, 3      IF .. ID EX ME WB
add x6, x10, x5      IF .. ID .. EX ME WB

ld x7, 0(x6)               IF .. ID EX ME WB
add x31, x11, x5           IF .. ID EX ME WB

ld x29, 8(x6)                    IF ID EX ME WB
addi x12, x12, 2                 IF ID EX ME WB

sub x30, x7, x29                    IF ID .. EX ME WB
sd x30, 0(x31)                      IF ID .. .. EX ME WB

bne x12, x13, TOP                      IF .. .. ID EX ME WB
slli x5, x12, 3                        IF .. .. ID .. EX ME WB

add x6, x10, x5                              IF .. ID EX ME WB
ld x7, 0(x6)                                 IF .. ID .. EX ME WB

add x31, x11, x5                                   IF .. ID EX ME WB
ld x29, 8(x6)                                      IF .. ID EX ME WB

addi x12, x12, 2                                         IF ID EX ME WB
sub x30, x7, x29                                         IF ID .. EX ME WB

sd x30, 0(x31)                                              IF .. ID EX ME WB
bne x12, x13, TOP                                           IF .. ID EX ME WB

slli x5, x12, 3                                                   IF ID EX ME WB
add x6, x10, x5                                                   IF ID .. EX ME WB
```

4.31.6

题目

执行练习题4.31.3和练习题4.31.4中的优化代码，从单发射处理器到双发射处理器的加速比是多少?

解答

4.31.3中的代码每次迭代需要9个循环。4.31.5中的代码每次迭代需要7.5个循环。

$$\frac{9}{7.5} = 1.2$$

因此，加速比为1.2。

4.31.7

题目

将练习题4.31.3中的$MIPS$代码进行循环展开，展开的循环中每个迭代可以处理原代码中的两个迭代。然后，重排或重写展开的代码，以便在单发射处理器上获得更好的性能。假设j是4的倍数。

解答

```
    beqz x13, DONE
    li x12, 0
TOP:
    slli x5, x12, 3
    add x6, x10, x5
    add x31, x11, x5
    ld x7, 0(x6)
    ld x29, 8(x6)
    ld x5, 16(x6)
    ld x15, 24(x6)
    addi x12, x12, 4
    sub x30, x7, x29
    sub x14, x5, x15
    sd x30, 0(x31)
    sd x14, 16(x31)
    bne x12, x13, TOP
DONE:
```

4.31.8

题目

将练习题4.31.4中的$MIPS$代码进行循环展开，展开的循环中每个迭代可以处理原代码中的两个迭代。然后，重排或重写展开的代码，以便在双发射处理器上获得更好的性能。假设j是4的倍数。（提示：重新组织循环，使得一些计算出现在循环体的外面和循环结束处。可以假设临时寄存器中的值在循环之后不再需要。）

解答

```
    beqz x13, DONE
    li x12, 0
    addi x6, x10, 0
TOP:
    ld x7, 0(x6)
    add x31, x11, x5
    ld x29, 8(x6)
    addi x12, x12, 4
    ld x16, 16(x6)
    slli x5, x12, 3
    ld x15, 24(x6)
    sub x30, x7, x29
    sd x30, 0(x31)
```

```
    sub x14, x16, x15
    sd x14, 16(x31)
    add x6, x10, x5
    bne x12,x13,TOP
DONE:
```

4.31.9

题目

执行练习题4.31.7和练习题4.31.8中循环展开以及优化后的代码，从单发射处理器到双发射处理器的加速比是多少?

解答

4.31.7中的代码每次展开迭代需要13个循环。这相当于每个原始迭代6.5个周期。4.30.4中的代码每次展开迭代需要7.5个循环。这相当于每次原始迭代3.75个周期。

$$\frac{6.5}{3.75} = 1.73$$

因此，加速比是1.73

```
                      1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24
beqz x13, DONE        IF ID EX ME WB
li x12, 0             IF ID .. EX ME WB
addi x6, x10, 0          IF .. ID EX ME WB
ld x7, 0(x6)             IF .. ID .. EX ME WB
add x31, x11, x5                IF .. ID EX ME WB
ld x29, 8(x6)                   IF .. ID EX ME WB
addi x12, x12, 4                      IF ID EX ME WB
ld x16, 16(x6)                        IF ID EX ME WB
slli x5, x12, 3                          IF ID EX ME WB
ld x15, 24(x6)                           IF ID EX ME WB
sub x30, x7, x29                            IF ID EX ME WB
sd x30, 0(x31)                              IF ID .. EX ME WB
sub x14, x16, x15                              IF .. ID EX ME WB
sd x14, 16(x31)                                IF .. ID EX ME WB
add x6, x10, x5                                      IF ID EX ME WB
bne x12,x13,TOP                                      IF ID EX ME WB
ld x7, 0(x6)                                            IF ID EX ME WB
add x31, x11, x5                                        IF ID EX ME WB
ld x29, 8(x6)                                              IF ID EX ME WB
addi x12, x12, 4                                           IF ID EX ME WB
ld x16, 16(x6)                                                IF ID EX ME WB
slli x5, x12, 3                                               IF ID EX ME WB
ld x15, 24(x6)                                                   IF ID EX ME WB
sub x30, x7, x29                                                 IF ID EX ME WB
sd x30, 0(x31)                                                      IF ID EX ME WB
sub x14, x16, x15                                                   IF ID .. EX ME WB
sd x14, 16(x31)                                                        IF .. ID EX ME WB
add x6, x10, x5                                                        IF .. ID EX ME WB
bne x12,x13,TOP                                                              IF ID EX ME WB
ld x7, 0(x6)                                                                 IF ID EX ME WB
```

4.31.10

题目

重复练习题$4.31.8$和练习题$4.31.9$，但此次假设双发射处理器能同时执行两条算术 / 逻辑指令。（也就是说，发射包中的第一条指令可以是任意类型的指令，但第二条指令必须是算术或逻辑指令。两条存储器操作指令不能同时被调度。）

解答

使用与4.31.8中相同的代码，新的数据路径没有提供任何实际改进，因为没有由于结构冒险而导致的停顿。

4.32

本题讨论能效与性能的关系。假设数据通路各部件（指令存储器、寄存器、数据存储器）的活动功耗如下表所示，其他部件的功耗可以忽略（寄存器读和寄存器写只针对寄存器堆）。

I-Mem	1 Register Read	Register Write	D-Mem Read	D-Mem Write
140pJ	70pJ	60pJ	140pJ	120pJ

假定数据通路上的部件延迟如下表所示，其他功能部件的延迟可以忽略。

I-Mem	Control	Register Read or Write	ALU	D-Mem Read or Write
200ps	150ps	90ps	90ps	250ps

4.32.1

题目

在单周期处理器和5级流水线处理器中执行一条加法指令，分别消耗多少能量?

解答

这两种设计的能量是相同的:$I-Mem$被读取，两个寄存器被读取，一个寄存器被写入。

我们有:

$$140pJ + 2 \times 70pJ + 60pJ = 340pJ$$

4.32.2

题目

消耗能量最大的$MIPS$指令是哪一条？执行这条指令的能耗是多少？

解答

读取所有指令的指令存储器。每条指令还会导致两次寄存器读取(即使实际上只使用了其中一个值)。加载指令导致存储器读取和寄存器写入;存储指令导致内存写入;所有其他指令最多只能写一次寄存器。由于存储器读和寄存器写能量的总和大于存储器写能量，所以最坏情况下的指令是负载指令。

对于负载消耗的能量，我们有:

$$140pJ + 2 \times 70pJ + 60pJ + 140pJ = 480pJ$$

4.32.3

题目

如果降低能耗是最重要的，应该怎样修改流水线设计？改进之后，执行一条lw指令时的能耗降低的比例是多少？

解答

每条指令都必须读取指令存储器。然而，我们可以避免读取那些值不被使用的寄存器。要做到这一点，我们必须向寄存器单元添加$regregad1$和$regregad2$控制输入，以启用或禁用每个寄存器读取。我们必须快速生成这些控制信号，以避免延长时钟周期时间。使用这些新的控制信号，一条加载指令只会读取一个寄存器(我们仍然必须读取用于生成地址的寄存器)，因此我们的更改每次加载节省了$70pJ$(一个寄存器读取)。

这是$70/480 = 14.6\%$的节省。

4.32.4

题目

还有哪些指令能从练习题4.32.3的改进中潜在收益？

解答

`jal`将从中受益，因为它根本不需要读取任何寄存器。

i型指令也将受益，因为它们只需要读取一个寄存器。如果我们添加逻辑来检测`x0`作为源寄存器，那么像beqz(`beq x0, …`)和li(`addi xn, x0, …`)这样的指令也可以受益。

4.32.5

题目

练习题4.32.3的改进如何影响CPU的性能?

解答

在改变之前，控制单元在读取寄存器的同时对指令进行解码。更改后，控制和读寄存器的延迟不能重叠。这增加了ID阶段的延迟，如果ID阶段成为延迟时间最长的阶段，可能会影响处理器的时钟周期时间。然而，寄存器读取($90ps$)和控制单元($150ps$)的延迟总和小于当前$250ps$的周期时间。

4.32.6

题目

我们可以去掉$MemRead$控制信号，即每个周期都读数据存储器（$MemRead$恒为1）。解释为什么去掉该控制信号后处理器依然能正常工作。它对时钟频率和能耗又有什么影响?

解答

如果在每个周期中都读取内存，则该值要么是需要的(对于加载指令)，要么没有通过`WB Mux`(对于向寄存器写入的非加载指令)，要么没有写入任何寄存器(所有其他指令，包括分支和暂停)。此更改不会影响时钟周期时间，因为时钟周期时间必须已经允许在MEM阶段读取内存的足够时间。如果未使用的内存读取导致缓存丢失，则会影响整体性能。

这种变化也会影响能量:在每个周期中都会读取内存，而不是只在加载指令处于MEM阶段时才读取内存。在$250ps$时钟周期的75%时间内，这增加了$140pJ$的能量消耗。这相当于大约0.46瓦的消耗(不包括由于缓存丢失而消耗的任何能量)。