《计算机组成原理》第七次作业

信息安全 胡博浩 2212998

4.21

1)

没有旁路的流水线需要1.4*n*250ps，有旁路的流水线需要1.05*n*300ps。因此，加速比是（1.4*250）/（1.05*300）=1.11

2)

目标是使具有旁路的流水线比没有旁路的流水线更快。设 y 为剩余的stall数占“code”指令的百分比。目标是 300*（1+y）*n < 250*1.4*n。因此，y 必须小于 16.7%。

3)

目标是 300（1 + y）*n < 250（1 + x）*n。当 y < （250x — 50）/300 时，就会发生这种情况。

4)

不能。在最好的情况下，旁路消除了对每个 NOP 的需求，程序将需要 300*n 的时间才能在具有旁路的流水线上运行。这比没有旁路的流水线上所需的 250*1.075*n 要慢。

5)

由0<（250x－50）/300解得x必须至少为0.2即20%

4.25

1)

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!
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  |
```

2)

如图所示，！表示不做有用工作的阶段。

在特定的时钟周期中，如果流水线级停滞不前，或者通过该级的指令没有执行任何有用的工作，则流水线级不会执行有用的工作。如上图所示，没有任何周期，在此期间，每个管道阶段都在做有用的工作。

4.31

1)

```
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
```

```
21
22 ld x7, 0(x6)                                                    IF .. ID EX ME
   WB
23 ld x29, 8(x6)                                                  IF .. ID .. EX
   ME WB
24
25 sub x30, x7, x29                                                  IF .. ID
   .. EX ME WB
26 add x31, x11, x5                                                  IF .. ID
   .. .. EX ME WB
27
28 sd x30, 0(x31)                                                       IF
   .. .. ID EX ME WB
29 addi x12, x12, 2                                                     IF
   .. .. ID .. EX ME WB
30
31 bne x12, x13, TOP
       IF .. ID EX ME WB
32 slli x5, x12, 3
       IF .. ID .. EX ME WB
```

2)

如下图所示，在单发射处理器上一个循环10个周期。

```
 1     li x12,0
 2     jal ENT
 3 TOP:
 4     slli x5, x12, 3
 5     add x6, x10, x5
 6     ld x7, 0(x6)
 7     ld x29, 8(x6)
 8 <stall>
 9     sub x30, x7, x29
10     add x31, x11, x5
11     sd x30, 0(x31)
12     addi x12, x12, 2
13 ENT:
14     bne x12, x13, TOP
```

循环1中的第一条指令（slli）在第6个周期开始执行，迭代3中的第一条指令在第26个周期开始执行，因此（26–6）/2=10。所以代码在双发射处理器上需要10个周期/迭代。

因此，这不会带来净加速。

3)

```
    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:
```

4)

```
    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:
```

5)

```
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
```

```
6
7  ld x7, 0(x6)                  IF .. ID EX ME WB
8  add x31, x11, x5              IF .. ID EX ME WB
9
10 ld x29, 8(x6)                    IF ID EX ME WB
11 addi x12, x12, 2                 IF ID EX ME WB
12
13 sub x30, x7, x29                    IF ID .. EX ME WB
14 sd x30, 0(x31)                      IF ID .. .. EX ME WB
15
16 bne x12, x13, TOP                      IF .. .. ID EX ME WB
17 slli x5, x12, 3                        IF .. .. ID .. EX ME WB
18
19 add x6, x10, x5                           IF .. ID EX ME WB
20 ld x7, 0(x6)                              IF .. ID .. EX ME WB
21
22 add x31, x11, x5                             IF .. ID EX ME WB
23 ld x29, 8(x6)                                IF .. ID EX ME WB
24
25 addi x12, x12, 2                                IF ID EX ME WB
26 sub x30, x7, x29                                IF ID .. EX ME WB
27
28 sd x30, 0(x31)                                     IF .. ID EX
   ME WB
29 bne x12, x13, TOP                                  IF .. ID EX
   ME WB
30
31 slli x5, x12, 3                                       IF ID
   EX ME WB
32 add x6, x10, x5                                       IF ID
   .. EX ME WB
```

6)

4.31.3中的代码每次迭代需要9个周期，4.31.4中的代码每次迭代需要7.5个周期。因此，加速比为9/7.5=1.2。

7)

```
1      beqz x13, DONE
2      li x12, 0
3   TOP:
4      slli x5, x12, 3
5      add x6, x10, x5
6      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:
```

8)

```
    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:
```

9)

如下图所示，4.31.7中的代码每次展开迭代13个周期。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7.5个周期。这相当于每个原始迭代3.75个周期。因此，加速比为 6.5/3.75=1.73。

```
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
```

```
 6
 7 add x31, x11, x5                IF .. ID EX ME WB
 8 ld x29, 8(x6)                   IF .. ID EX ME WB
 9
10 addi x12, x12, 4                      IF ID EX ME WB
11 ld x16, 16(x6)                        IF ID EX ME WB
12
13 slli x5, x12, 3                          IF ID EX ME WB
14 ld x15, 24(x6)                           IF ID EX ME WB
15
16 sub x30, x7, x29                            IF ID EX ME WB
17 sd x30, 0(x31)                              IF ID .. EX ME WB
18
19 sub x14, x16, x15                              IF .. ID EX ME WB
20 sd x14, 16(x31)                                IF .. ID EX ME WB
21
22 add x6, x10, x5                                      IF ID EX ME WB
23 bne x12,x13,TOP                                      IF ID EX ME WB
24
25 ld x7, 0(x6)                                            IF ID EX ME WB
26 add x31, x11, x5                                        IF ID EX ME WB
27
28 ld x29, 8(x6)                                              IF ID EX ME WB
29 addi x12, x12, 4                                           IF ID EX ME WB
30
31 ld x16, 16(x6)                                                IF ID EX ME WB
32 slli x5, x12, 3                                               IF ID EX ME WB
33
34 ld x15, 24(x6)                                                   IF ID EX ME
   WB
35 sub x30, x7, x29                                                 IF ID EX ME
   WB
36
37 sd x30, 0(x31)                                                      IF ID EX
   ME WB
38 sub x14, x16, x15                                                   IF ID ..
   EX ME WB
39
40 sd x14, 16(x31)                                                        IF ..
   ID EX ME WB
41 add x6, x10, x5                                                        IF ..
   ID EX ME WB
42
43 bne x12,x13,TOP
   IF ID EX ME WB
44 ld x7, 0(x6)
   IF ID EX ME WB
```

10）

使用与4.31.8中相同的代码，因为没有由于结构冒险而导致的stall，所以没有提供净改进，加速比仍为1.73。

4.32

1）

两种设计的能量是相同的：读取 I-Mem，读取两个寄存器，并写入一个寄存器。故有：140pJ + 2*70pJ + 60j = 340pJ。

2）

指令存储器读取所有的指令。每条指令还会产生两次寄存器读取。load指令导致存储器读取和寄存器写入;store指令导致内存写入;所有其他指令最多导致单个寄存器写入。由于内存读取和寄存器写入的能量的总和大于存储器写入的能量，因此消耗能量最大的指令是load指令。而load指令消耗的能量为140pJ + 2*70pJ + 60pJ + 140pJ = 480pJ。

3）

可以避免读取不会使用其值的寄存器。因此，将RegRead1和RegRead2控制输入添加到寄存器单元，以启用或禁用每个寄存器读取。为避免延长时钟周期时间，必须快速生成这些控制信号。使用这些新的控制信号，load指令只读取一个寄存器，所以更改后每个load指令节省一个寄存器读取的能量即70pJ，能耗降低的比例为70/480 = 14.6%。

4）

jal指令将受益，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beqz（即beq x0，...）和li（addi xn， x0， ...）等指令也会受益。

5）

更改前，控制单元在进行寄存器读取时对指令进行解码。

更改后，Control和Register Read的延迟不能重叠，这会增加ID的延迟。如果ID成为最长的延迟级，则可能会影响处理器的时钟周期时间。但是，寄存器读取（90ps）和控制单元（150ps）的延迟总和小于当前的250ps周期时间。

6）

如果在每个周期中都读取数据存储器，则该值要么对于load指令是必需的，要么对于写入寄存器的非load指令它没有通过WB Mux，或者对于所有其他包括包括分支和停顿的指令它不会写入任何寄存器。此更改不会影响时钟周期时间，因为时钟周期时间已经必须允许足够的时间在MEM阶段读取数据存储器。如果未使用的数据存储器读取导致缓存未命中，则可能会影响整体性能。

这种变化也会影响能耗，因为每个周期都会发生一次数据存储器读取。在为期250ps的时钟周期的75%的时间，能耗增加了140pJ。这相当于大约0.46瓦的功耗（不包括因缓存未命中而消耗的任何能量）。