به نام خدا

تمرین 7 معماری کامپیوتر

جناب آقای دکتر اسدی

سارا آذرنوش

98170668

(1

```
Addi $s1, $s1, 4
Add $s2, $s2, $s1
Sub $s3, $s2, $s3
Beq $s2, $s3, P2
Add $s4, $s1, $s0
Sub $s5, $s3, $s2
P2: add $s4, $s5, $s2
```

در اینجا سه نوع Data hazard و Data hazard و Structure hazard داریم.

Data hazard :

دستور به نتیجه دستور قبلی نیاز دارد.

Data hazard:

برای برنچ است و مقادیر را مقایسه و تصمیمگیری میکند.

Structure hazard:

دو دستور به یک منبع حافظه نیاز دارند.Mem if باهم همزمان باشند

دستور ۱ و ۲ : در دستور اول مقدار رجیستر s1 با 4 جمع شده و تغییر کرده و در اخرین کلک مربوط به دستور Addi در پایپلاین، مقدار آن به مقدار جدید تغییر میکند. اما در دستور بعدی (Add)، مقدار رجیستر s1 در کلک دوم با s2 جمع میشود اما رجیستر در دستور قبلی هنوز مقداردهی جدید نشده است بنابراین بین دو دستور Addi و Add ، Hazard Data داریم .

دستور ۲ و ۳ : مشابه قبلی است مقدار s2 در انتهای کلک دستور Add آماده میشود اما در کلک دوم دستور Sub به مقدار این رجیستر نیاز است بنابراین Hazard Data داریم.

دستور۳ و ۴ : مقادیر s2 و s3 در کلک سوم با یکدیگر مقایسه میشوند و مقدار جدید برای s3 در آخرین کلک دستور قبلی تولید میشود و branch ممکن است به درستی صورت نگیرد و Hazard Data داریم.

Beq : از دستورات قبل، مقادیر s2 و s3 با یکدیگر برابر و مساوی با ۴ اند بنابراین branchرخ خواهد داد. و Hazard Controlرخ میدهد بنابراین در این دستور نیز نوع دیگری از مخاطره مشاهده میشود.

و در ادامه مخاطره وجود ندارد.

و همچنین از ساختار سخت افزاری اطلاع نداریم ممکن است rf و مموری دچار اختلال شوند و در دستور اول و چهارم و چهارم و آخر مخاطره وجود دارد

(2

Default values: $s0 = 0, $s1 = 4, Mem[0x0000000i] = (convert(i to decimal)) MOD 9

#of bubbles for control hazard = 3

# of bubbles for data hazard = 2

Add $s2, $s1, $s0

Lw $s3, $s2, $s0

Add $s4,$s3, $s0

Beq $s4, $s2, L2

Sub $s3, $s2, $s3

L2:

add $s2, $s1, $s0

دستور۱ و ۲ : مقادیر s1 و s0 با یکدیگر جمع میشوند و حاصل در کلک آخر در رجیستر s2 نوشته میشود. و در دستور بعد باید مقدار s3 توسط s2 آدرس تعیین میشود اما مقدار s2تا آخرین کلک دستور قبل مشخص نشده است بنابراین Hazard Data داریم و نیاز به ۲ بابل خواهیم داشت . s2 = 4 +0 =4

Add => im reg alu dm reg

Lw => im bub bub reg alu dm reg

دستور ۲ و ۳ : مقدار s3 در کلک آخر مقدار دهی شده و ۴ است. s2 = Mem[4] و در دستور ۳ حاصل جمع s3 و s0 در داخل s4 ذخیره میشود و Hazard Data رخ خواهد داد و ۲ بابل ایجاد خواهد شد. S4 = 0 + 4 = 4

Lw => im reg alu dm reg

Add => im bub bub reg alu dm reg

دستور ۳ و ۴ : دستور ۳ در آخرین کلک رجیستر s4 مقدار دهی میشود و در دستور بعد در کلک دوم به مقدار s4 نیاز داریم تا مقدار آن را با s2 مقایسه کنیم. بنابراین باید متوقف شود و Hazard Data داریم و در اینجا نیز ۲ بابل نیاز خواهیم داشت.

Beq: مقادیر رجیسترهای s2 و s4 با یکدیگر مقایسه میشوند. میدانیم مقادیر آنها برابر و ۴ است بنابراین branchاتفاق می افتد و دستور بعدی که باید اجرا شود دستور L2است و Hazard Control داریم. ۳ بابل نیاز داریم تا pc به دستور L2 برود.

مقادیر برابر است و دستور بعد آن اجرا نمیشود و بعد از رفتن به مقدار جدید نیاز به بابل نیست همه انجام شده اند.

بابل ها: 9= 3 + 2 + 2 + 2

(3

5 مرحله

F D X M W

در پایپلاین چون دستورات موازی هستند پیش از چک کردن شرط پرش دستورات بعدی اجرا میشوند و در اینجا در مرحله سوم اماده میشود بنابراین دو دستور بعدی نیز اجرا میشوند و باید جلوگیری کنیم در اینجا تنها delay slot مجاز است بنابراین دو دستور بعدی اجرا نمیشوند و وقفه ایجاد میکنیم.

BRANCH F D X M W

INSTR1 F D X M W

INSTR2 F D X M W

INSTR3 F D X M W

```
BRANCH: branch-instruction
INSTR1: branch-delay-slot-1
INSTR2: branch-delay-slot-2
```

1)

ADD R5 <- R4, R3

OR R3 <- R1, R2

SUB R7 <- R5, R6

J X

DELAY SLOTS

LW R10 <- (R7)

ADD R6 <- R1, R2

X:

2(

ADD R5 <- R4, R3

OR R3 <- R1, R2

SUB R7 <- R5, R6

BEQ R5 <- R7, X

DELAY SLOTS

LW R10 <- (R7)

ADD R6 <- R1, R2

X:

بعد از هر دستور برنچ یا جامپ به delay slot 2 نیاز است.

ب)

مقصد جامپ و برنچ و هدف آن‌ها را به مرحله دیکود میبریم. و یک سایکل زودتر انجام میشن و باعث میشه به یک delay slot نیاز بشه.

1)

ADD R5 <- R4, R3

J X

OR R3 <- R1, R2

SUB R7 <- R5, R6

LW R10 <- (R7)

ADD R6 <- R1, R2

X:

2(

ADD R5 <- R4, R3

SUB R7 <- R5, R6

BEQ R5 <- R7, X

OR R3 <- R1, R2

nop

LW R10 <- (R7)

ADD R6 <- R1, R2

X:

4)

PipelineSpeedup=

(ClockCycleTimeWithoutPipelining ⁎CPIWithoutPipelining)/

ClockCycleTimeWithPipelining ⁎ CPIWithPipelining)

Without pipeline:(multi sycle)

1.4*10*n = 14n

حال باید حالت pipeline مورد بررسی قرار گیرد. برای بدست آوردن بیشترین تسریع، باید فرض کنیم که بهبود زمانی پایپ لین بسیار زیاد است بنابراین باید فرض کنیم در هیچکدام از مراحل pipeline ، bubbleوارد نمیشود. از سوی دیگر برای فرمول time executionبرای pipeline ،با در نظر گرفت pipeline به شیوه ایده آل مقدار CPI برابر ۱ خواهد بود. از سوی دیگر CTنیز برای پایپ لین، برابر زمان طولنیترین بند همراه با تاخیر latch است

Pipeline:

(1*n+4)*10= 40 + 4n

Lim 14n/(40 + 4n) = 14/4 = 3.5

Without pipeline:(single sycle)

1.4*10*n = 14n

Pipeline:

(1*n+4)*(10/5)= 8 + 2n

Lim 14n/(8 + 2n) = 14/2 = 7

ب)

Max(1,1.5,4,3,0.5) = 4

Pipeline:

(5+n-1)*4 = 4n + 16

No pipeline:

(1 + 1.5 + 3 + 4+ .5)*n = 10n

Lim 10n /(4n +16) = 10/4 =2.5

Lim 14n /(4n +16) = 14/4 =3.5

ج)

Cpi:

.2*2 + 3* .05 + .75*1 = .4 + .75 + .15 = 1.3

Pipeline:

10 x (1.3 CPI x n inst + 4 cycle drain) = 13n + 40

Multi sycle باشد: (1*n+4)*10= 40 + 4n

Lim (40 + 4n)/(13n + 40) = 4/13

Single sycle باشد : (1*n+4)*(10/5)= 8 + 2n

Lim (8 + 2n) / (13n + 40) = 2/13

5)

تک = 4

پایپلاین = 1.5

تاخیر = .25

Control hazard = 1

Other hazards = 2

```
li $t0, 58
li $t1, 1
li $t2, 0
L1: beq $t2, $t0, L2
 addi $t2, $t2, 1
 subi $t3, $t2, 3
 add $t3, $t3, $t1 (t3++)
 j L1
```

L2:

im reg alu dm reg

الف)

مقدار T2 = T0 شود برنامه به قسمت بعد میرود که در آن58 = t0 است و t2 در هر حلقه یک مقدار افزایش میابد.

3 دستور قبل از حلقه و یک دستور بعد از آن است در هر حلقه 5 دستور است و 58 بار حلقه اجرا میشود.

ci = 3 + 1 + 5 *58 = 294

time = ci * cpi * ct = 1 * 4 * 294 = 1176

ب)

مخاطره‌ها را برسی میکنیم پیش از خط beq هنوز رجیسترهایی که مقایسه میکنیم مقدار دهی نشده‌اند و بنابراین data hazard داریم که 2 بابل گفته شده است.

در آخرین برسی زمانی که هر دو با هم برابر میشوند control hazard داریم و به l2 میرود و برنامه تمام میشود و برابر 1 بابل است.

در طول حلقه نیز برای برای addi به subi و subi به add control hazard داریم. بابل ها =><(2*58*2 = 116*2 = 232)

بابل‌ها:

1 + 2 + 232 = 235

Ct = clock + latch = 1.5 +.25 = 1.75

Instr + bubble + ic = 235 + 294 + 4 = 533

533 * 1.75 = 932.75

ج)

PipelineSpeedup=

(ClockCycleTimeWithoutPipelining ⁎CPIWithoutPipelining)/

ClockCycleTimeWithPipelining ⁎ CPIWithPipelining)

1176 / 932.75 = 1.26

6)

Always taken:

$3*(1-0.45)*0.25 = 0.41.$

Always not taken:

$3*(1-0.55)*0.25 = 0.34.$

2bit:

$3*(1-0.85)*0.25 = 0.113.$

ب)

1 + 3 * (1-.8) * .15 = 1.09

1 + 1 * (1 - .8) * .15 * .5 = 10.45

1.0.9 / 1.045 = 1.043

ج)

1 + (1 + 3 *(1-.8)) * .15 * .5 = 1.12

1.09 / 1.120 = .97

د)

پیشبینی درست:

N*0.8

پیشبینی درست بدون لوپ:

N*0

0/n.8=0