HW3. 2018/662 이건영

1. (a) <기존>

R, I-type : 30 + 250 + 150 + 200 + 25 + 20 + 25 = 700 PS

LW : 30 + 250 + 150 + 25 + 200 + 250 + 25 + 20 = 950 PS

SW : 30 + 250 + 150 + 200 + 25 + 250 = 905 PS

beq : 30 + 250 + 150 + 25 + 200 + 5 + 25 + 20 = 705 PS

<변경 후>

R, I-type : 30 + 250 + 160 + 25 + 200 + 25 + 20 = 710 PS.

LW : 30 + 250 + 160 + 25 + 200 + 250 + 25 + 20 = 960 PS.

SW : 30 + 250 + 160 + 200 + 25 + 250 = 915 PS

beq : 30 + 250 + 160 + 25 + 200 + 5 + 25 + 20 = 715 PS.

기존과 변경 후의 clock period는 각각 950, 960 PS 이다.

변경 후에는 lw와 sw의 수가 12% 감소하므로,

12% × (25% + 11%) = 4.3%, 따라서 전체 instruction의 100 − 4.3 = 95.7% 가 된다.

즉, 960 × 0.957 = 918.7 PS.

따라서 speedup = 950 / 918.7 = 1.03, "3% speed up."

(b) <기존>

Sum의 cost = 1000 + 200 + 10×4 + 100 + 30×2 + 2000 + 5 + 100 + 1×2 + 500×2 = 4507

<변경 후>

" => beq와 file의 cost가 200 → 400이 되었으므로 4707.

Cost 증가량 = 4707 / 4507 = 1.04 . "4% 증가."

(c) makes sense : 성능이 우수시되는 경우 (4%의 비용 증가를 감안하더라도)

doesn't : 성능 증가율보다 비용 증가율이 더 크므로 대부분의 경우에서 부적합

2 (a).

sd	IF	ID	EX	MEM	WB.						
ld		IF	ID	EX	MEM	WB.					
sub			IF	ID	EX	MEM	WB				
beqz				*	*	IF	ID	EX	MEM	WB.	
add							IF	ID	EX	MEM	WB
sub								IF	ID	EX	MEM	WB.

(b). 코드의 순서를 재배열하더라도 hazard가 일어나는 코드 상만 바꾸고 stall은 여전히 발생하기 때문에 도움이 되지 않는다.

(C) Nop를 사용하더라도 결국 Instruction Fetch가 일어나야 하기 때문에 해결되지 못한다.

(d). 36%.

3. (a)

ld	IF	ID	EX	MEM	WB.						
ld		IF	ID	EX	MEM	WB					
add			IF	ID	*	EX	MEM	WB			
add i				IF	*	ID	EX	MEM	WB.		
bnez					*	IF	ID	EX	MEM	WB	
ld							IF	ID	EX	MEM	WB.
ld								IF	ID	EX	MEM	WB
add									IF	ID	*	EX	MEM	WB
add i										IF	*	ID	EX	MEM	WB
bnez.											IF	ID	EX	MEM	WB

(b) 0으로 표시하였다. Pipeline이 full일 경우 모든 stage에서 useful work를 하는 cycle은 없다.

4. (a)

```
add $S3, $S1, $S0
nop
nop
lw  $S2, 4($S3)
lw  $S1, 0($S4)
nop
or  $S2, $S3, $S2
nop
nop
sw  $S2, 0($S3)
```

(b)

```
add $S3, $S1, $S0
lw  $S1, 0($S4)   ← 순서 변경
nop
lw  $S2, 4($S3)
nop
nop
or  $S2, $S3, $S2
nop
nop
sw  $S2, 0($S3)
```

=> lw instruction의 순서를 바꿨으나 다른 instruction끼리 연관되어있어 NOP의 수를 줄이지는 못했다

(c) load instruction의 결과가 MEM stage 이후에 register에 배정되는데, hazard detection unit이 없다면 이를 감지하지 못해 그후 다른 instruction에서 잘못된 값(load 전 값)을 이용하게 될 수 있다.

(d)

	CC 1	2	3	4	5	6	7	PCWrite	Signals. ALUin1	ALUin2
add $S3, $S1, $S0	IF	ID	EX	MEM	WB			1	X	X
lw $S2, 4($S3)		IF	ID	EX	MEM	WB		1	X	X
lw $S1, 0($S4)			IF	ID	EX	MEM	WB	1	0	0
or $S2, $S3, $S2				IF	ID	EX	MEM	1	1	0
sw $S2, 0($S3)					IF	ID	EX	1	0	0

(e). ID stage에 있는 instruction은 EX나 MEM stage의 결과물에 따라 stall 될 수 있다. 즉 각 instruction의 rd register를 확인해야 한다.

① EX stage : R-type, load instruction에서 rd를 확인해야 한다.

② MEM stage : 내가 이미 선택되었으므로 register number를 확인해야 한다.

new input : ID/EX pipeline 레지스터의 rd, EX/MEM pipeline 레지스터의 output register number.

new output : 필요 X. (기존의 output signal로 출력)

example : lw $s2, 4($s3) 는 직전 instruction인 add $s3, $s1, $s0 의 rd를 사용하므로 stall 된다.

(f)

	cc 1	2	3	4	5.	Signal PCWrite
add $s3, $s1, $s0	IF	ID	EX	MEM	WB	1
lw $s2, 4($s3)		IF	ID	*	*	1
lw $s1, 0($s4)			IF	*	*	1
or $s2, $s3, $s2					*	0
sw $s2, 0($s3)						0.