CSc 656 Project 1 Part a

(This document available on iLearn.)
due Thursday 3/15/2018 4:15 pm, in class (no grace period)
(4% of your grade)

This is an individual assignment. Work on your own!

For all problems, submissions must be typed. Occasional handwritten annotations are ok, but most text *must* be typed. Points will be deducted for too many messy handwritten annotations.

Problem 1 (30 points):

Suppose we have two versions of a MIPS pipeline.

Version 1: this is the 5-stage pipeline discussed in Chapter 4 up to Slide 73 or so. Taken branches take one stall, non-taken branches do not stall. Cycle time is 400 ps.

Version 2: this is an 8-stage pipeline, similar to the one discussed in Chapter 4 Slides 93-94, but with 3 stages for MEM instead of two:

```
IF1 IF2 ID  EX  ME1 ME2 ME3 WB
```

Cycle time is 250 ps. Refer to the discussion in class for the number of cycles for a load-use stall, and for the number of stalls taken by each type of branch.

We run the program Spectre on the two versions of the MIPS pipeline. Instruction mix:

Loads	35%
Stores	10%
ALU	35%
Branches	20%

55% of the loads incur load-use stalls; the number of cycles for each load-use stall depends on the version of the pipeline only. 65% of all branches are taken.

Let the instruction count of Spectre be IC. What is the total execution time of Spectre on each version of the pipeline? Make sure you show work and specify units clearly.

ANSWER:

Version 1

Each load-use hazard causes 1 stall
Each taken branch causes 1 stall

Execution cycles for loads = .35 * IC + .35 * .55 * IC * 1 = .5425 IC
Execution cycles for stores = .1 * IC
Execution cycles for ALU = .35 * IC
Execution cycles for branches = .2 * IC + .65 * .2 * IC * 1 = .33 IC
Total execution cycles = .5425 IC + .1 IC + .35 IC + .33 IC = 1.3225 IC cycles
Total execution time = 1.3225 IC * 400 ps = 529 ps

Version 2

Each load-use hazard causes 3 stalls
Each taken branch causes 2 stalls

Execution cycles for loads = .35 * IC + .35 * .55 * IC * 3 = .9275 IC
Execution cycles for stores = .1 * IC
Execution cycles for ALU = .35 * IC
Execution cycles for branches = .2 * IC + .65 * .2 * IC * 2 = .46 IC
Total execution cycles = .9275 IC + .1 IC + .35 IC + .46 IC = 1.8375 IC cycles
Total execution time = 1.8375 IC * 250 ps = 459.375 ps

Problem 2a (20 points):

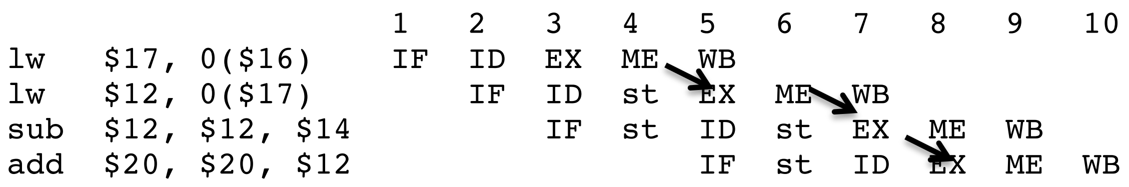
Suppose we have a 5-stage MIPS pipeline with all possible forwarding. Branches work as described up to Chapter 4 Slide 67, i.e., branch condition and target address are calculated in ID. Consider this MIPS loop:

```
loop:lw   $17, 0($16)
     lw   $12, 0($17)
     sub  $12, $12, $14
     add  $20, $20, $12
```

Do not change the order of the instructions. Show timing of this MIPS instruction sequence through the pipeline. Show all stalls clearly, and mark with arrows all cases where forwarding takes place, as in our lectures. (See for example Chapter 4 Slide 50.)

In addition, state clearly how ForwardA and ForwardB are set in the cycles where forwarding takes place (as discussed in class). For example: *Cycle 1 ForwardA = 01*

ANSWER:



```
                        1   2   3   4   5   6   7   8   9   10
lw   $17, 0($16)        IF  ID  EX  ME  WB
lw   $12, 0($17)            IF  ID  st  EX  ME  WB
sub  $12, $12, $14              IF  st  ID  st  EX  ME  WB
add  $20, $20, $12                      IF  st  ID  EX  ME  WB
```

Cycle 5: ForwardA = 01
Cycle 7: ForwardA = 01
Cycle 8: ForwardB = 10

Problem 2b (20 points):

Consider this MIPS code:

```
loop:lw   $4, 0($16)
     lw   $7, 4($16)
     add  $4, $7, $4
     sw   $4, 0($23)
     addi $16, $16, 8
     addi $23, $23, 4
     bne  $16, $2, loop
```

Unroll 3 times the MIPS code (i.e., one unrolled iteration does the work of four original iterations). Make it as efficient as you can. Show timing in the 5-stage MIPS pipeline for one unrolled iteration of the loop. Assume that the number of iterations is always a multiple of four.

ANSWER (many possible orders!):

```
loop:lw   $4, 0($16)
     lw   $7, 4($16)
     lw   $5, 8($16)
     lw   $8, 12($16)
     lw   $6, 16($16)
     lw   $9, 20($16)
     lw   $10, 24($16)
     lw   $11, 28($16)

     add  $4, $7, $4
     add  $5, $8, $5
     add  $6, $9, $6
     add  $10, $11, $10

     sw   $4, 0($23)
     sw   $5, 4($23)
     sw   $6, 8($23)
     sw   $10, 12($23)

     addi $16, $16, 32
     addi $23, $23, 16
     bne  $16, $2, loop
```

Problem 2c (30 points):

Suppose we have a static 2-issue MIPS processor, as in Fig. 4.69. Show timing for one unrolled iteration of the loop from Problem 2b, using a table similar to Fig. 4.70. You should change the order of the instructions to optimize the performance. Use these assumptions for the hardware:

> Unit 1 for ALU/branch instructions, Unit 2 for lw/sw
> If an arithmetic instruction executes in cycle 1,
> > its result can be used in cycle 2 by a lw, sw or arith instruction
> > its result can be used in cycle 3 by a branch
>
> If a lw executes in cycle 1,
> > its result can be used in cycle 3 by lw, sw or arith instruction
> > its result can be used in cycle 4 by a branch
>
> ~~branches are perfectly predicted; if a branch executes in cycle 1,~~
> > ~~its target or fallthrough is executed in cycle 2~~ *[irrelevant to problem]*

ANSWER (many possible orders):

Cycle	Unit 1 (ALU/branch)	Unit 2 (load/store)
1		`lw $4, 0($16)`
2	`addi $16, $16, 32`	`lw $7, 4($16)`
3	`addi $23, $23, 16`	`lw $5, -24($16)`
4	`add $4, $7, $4`	`lw $8, -20($16)`
5		`lw $6, -16($16)`
6	`add $5, $8, $5`	`lw $9, -12($16)`
7		`lw $10, -8($16)`
8	`add $6, $9, $6`	`lw $11, -4($16)`
9		`sw $4, -16($23)`
10	`add $10, $11, $10`	`sw $5, -12($23)`
11		`sw $6, -8($23)`
12	`bne $16, $2, loop`	`sw $10, -4($23)`