CS251 F16 Assignment 05
Due Monday November 21 4:30pm
3% of Course Mark
50 Total Marks

Branch Data Hazards:
For this assignment a branch data hazard needs to be considered when branch is determined in the ID stage. If an *R-Format* instruction immediately precedes a branch instruction and a data hazard exists, a one clock cycle delay is needed. If a *lw* instruction immediately precedes a branch instruction and a data hazard exists, a two clock cycle delay is needed. The hardware to stall in the case of *branch data hazards* for branching in the ID stage does not exist in the datapath. Stalling exists for load-use hazards only for a one clock cycle stall. Branch data hazards need to be considered wherever applicable in this assignment.

Q1. (8 marks) Given the following code sequence, explain which options (1-3) would be required to handle the branch hazard on the pipelined datapath as described in parts (a-d) below. Choose one of options (1-3) to best solve each situation with minimal costs. You may assume for all options below, data forwarding and load-use stalling exist in the datapath. It is unknown if the branch will be taken. If you selected Option (1) or (3), perform the code rearrangement necessary or use NOPs if needed.

```
32 add $5, $2, $3
36 subi $1, $4, 25
40 and $12, $2, $5
44 beq $1, $2, 25
48 or $13, $6, $2
52 add $14, $2, $2
56 lw $4, 50($7)
```

1. Perform code rearrangement for 3 instructions following branch, or NOPS as needed.
2. The datapath will handle branch hazards with flushing therefore code rearrangement will not have an effect.
3. Perform code rearrangement for 1 instruction following the branch, or NOP as needed.

a) Branches are determined in the MEM stage. No branch flushing exists in the datapath.
Option# 1

```
32 add $5 $2 $3
36 subi $1 $4 25
40 beq $1 $2 25        44 NOP
56 and $12 $2 $5       48 NOP
60 or $13 $6 $2        52 NOP
64 add $14 $2 $2
68 lw $4 50($7)
```

b) Branches are determined in the ID stage. No branch flushing exists in the datapath.
Option #1

```
32 add     the same
36 subi    same
40 beq     $1 $2 25
44 NOP
48 and $12 $2 $5
52 or  $13 $6 $2
56 add $14 $2 $2
60 lw  $4  50($7)
```

c) Branches are determined in the ID stage, and branch flushing exists in the datapath for up to 1 errant instruction. **Option #**2

d) Branches are determined in the MEM stage and branch flushing exists in the datapath for up to 3 errant instructions. **Option #**2

Q2. (11 marks) Given the following code sequence, state the total number of clock cycles required, and the total number of flushed instructions in the pipelined datapath as described in parts (a-c). You need to also consider the startup time for the pipeline and any possible hazards that will lead to extra clock cycles in the total execution time.

```
100 addi $1, $0, 15
104 addi $2, $0, 0
108 addi $4, $0, 0
112 lw $3, 0($6)
116 add $2, $2, $3
120 lw $5, 60($6)
124 add $4, $4, $5
128 addi $1, $1, -1
132 addi $6, $6, 4
136 bne $1, $0, -7
140 slt $5, $4, $0
144 slt $3, $2, $0
148 add $8,$2,$3
```

a) (3 marks) Using a Pipelined datapath with *data forwarding* and *load-use stalling*. Branch flushing exists for instructions that are not needed following a branch. The branch instruction is determined in the MEM stage. Show your work.

i. Given the code sequence above, how many total flushed instructions would there be.

14

ii. What is the Total Number of Clock Cycles required to execute the above code, including all instructions and all possible hazards.

startup time : 4 clock cycles

bne determined in MEM: if \$rs ≠ \$t, branch: flushing instructions in IF/ID/EX: +3 clockcycles

We have 14 Loops, two load-use stalling in 112 to 116 and 120 to 124: 14×(9+3)=168

Total: 4+168+15=187

b) (3 marks) Using a Pipelined datapath with *data forwarding* and *load-use stalling*. Branch flushing exists for instructions that are not needed following a branch. The branch instruction is determined in the ID stage. Show your work.

i. Given the code sequence above, how many total flushed instructions would there be.

14

ii. What is the Total Number of Clock Cycles required to execute the above, including all instructions and all possible hazards.

startup time: 4

bne detemined in ID, two load-use stalling in 112 to 116 and 120 to 124

: 14×(9+1)=140

Total = 4+140+15= 159 .

c) (5 marks) Perform code rearrangement (or insertion of NOPs) in order to avoid any extra clock cycles. The datapath implements *load-use stalling* and *data forwarding*, and branching is determined in the ID stage, but **the datapath does not implement flushing.** The original code sequence has been copied for your convenience.

(A) ORIGINAL	(B) CODE REARRANGEMENT
100 addi $1, $0, 15	100 addi $1 $0 15
104 addi $2, $0, 0	104 addi $2 $0 0
108 addi $4, $0, 0	108 addi $4 $0 $0·
112 lw $3, 0($6)	112 lw $3 0($6)
116 add $2, $2, $3	116 addi $1 $1 -1
120 lw $5, 60($6)	120 add $2 $2 $3
124 add $4, $4, $5	124 lw $5 60($6)
128 addi $1, $1, -1	128 addi $6 $6 4
132 addi $6, $6, 4	132 addi $4 $4 $5
136 bne $1, $0, -7	136 bne $1 $0 -7
140 slt $5, $4, $0	140 NOP
144 slt $3, $2, $0	144 slt $5 $4 $0
148 add $8,$2,$3	148 slt $3 $2 $0
	152 add $8 $2 $3.

i) What is the Total Number of Clock Cycles required to execute your above code in column (B), including all instructions and all possible hazards?

Start up : 4 .

14 loops : 14×7

Total: 14(7+1) +4 +14 = 130

Q3. (15 marks)
Here is a series of 4 bit address references given as word addresses in both decimal and binary.

Dec Addr	0	1	2	3	0	1	6	8	0	10	2	3	1	0
Binary	0000	0001	0010	0011	0000	0001	0110	1000	0000	1010	0010	0011	0001	0000

Below are four different 8-word caches. For each cache type, assuming the cache is initially empty, show the final contents of the cache. In the table at the bottom, show how many cache hits and misses there are for each type of cache. Write your solution in the tables below, assuming the above word address are 4-bit binary numbers. You should write the binary form of the tag in the tables indicating how many bits are in the tag field and index field. Also, the data column would be M[3], for data at memory address 3, M[8] for data at memory address 8.
You must use a LRU (least recently used) replacement scheme for bumping out entries in the cache when necessary. When inserting an element into the cache, if there are multiple empty slots for that index, you should put the new element in the left-most empty slot.

Direct-Mapped Cache

Index	Tag	Data
000	~~0~~10	
001	0	
010	~~0~~~~1~~0	
011	0	
100		
101		
110	0	
111		

Two-Way Set Associative

Index	Tag	Data	Tag	Data
00	00		10	
01	00			
10	~~00~~ 10		~~01~~ 00	
11	00			

Four -Way Set Associative

Index	Tag	Data	Tag	Data	Tag	Data	Tag	Data
0	000		~~001~~ 101		~~011~~ 001		100	
1	100		001					

Fully Associative

Tag	Data	Tag	Data	Tag	Data	Tag	Data	Tag	Data	Tag	Data	Tag	Data	Tag	Data
0000		0001		0010		0011		0110		1000		1010			

Set	Hits	Misses
One-way (Direct-mapped)	5	9
Two-way	6	8
Four-way	6	8
Fully Associative	7	7

Q4. (12 marks)
Consider the following MIPS code for summing the values of an array stored in Memory.

```
096 addi $4, $4, 320
100 addi $1, $0, 200
104 addi $2, $0, 0
108 lw $3, 0($4)
112 add $2, $2, $3
116 addi $1, $1, -1
120 addi $4, $4, 4
124 bne $1, $0, -5
128 sll $0,$0, $0
```

Assume that this code is to run on a pipelined datapath where branching is determined in the ID stage (Figure 4.65 of the text). Branch flushing, data forwarding and load-use stalling all exist in the datapath. Assume that each instruction takes 1 clock cycle, memory is read into cache in blocks, and that reading a block of **n** words of memory takes 20 + n clock cycles. (i.e., if the block size is 1, then it takes 21 clock cycles to read a word from memory; if the block size is 2, then it takes 22 clock cycles to read two consecutive words of memory etc.).
How long will this segment of code take to execute, assuming cache block sizes of 2, 4, and 8?
Fill in the following table to determine your answer. Assume that the pipeline is already full (ignore first 4 startup instructions for the pipeline). Assume initially that *instruction cache* and *data cache* are empty. Once data or instructions are brought into cache from memory they will remain there for the remaining execution of the program (i.e., items are not bumped out of cache).
You need to also consider any possible hazards that will lead to extra clock cycles in the total execution time for Instructions in the pipeline. Show your computations in the table.

Cache Block Size	Instructions in Pipeline clock cycles	Instruction Cache Misses total clock cycles	Data Cache Misses total clock cycles	Total Clock Cycles
Size 02	1403	13244	4400	19047
Size 04	1403	9624	4800	15827
Size 08	1403	56	5600	7059

Q5. (4 marks) The datapath on the next page shows the hardware needed to execute branch in the ID stage. The zero bit ANDed with the Branch control bit is missing from this diagram; however you may assume it exists and all the necessary hardware to take a branch in ID exists in the datapath. As noted at the top of the assignment, when branching is determined in the ID stage, data hazards may now exist between the branch instruction and an instruction that immediately precedes it. A copy of a condition to detect a data hazard between two instructions has been copied from the course notes and is given below:

(if (MEM/WB.RegWrite)
and (MEM/WB.RegisterRd != 0)
and (EX/MEM.RegisterRd != ID/EX.RegisterRs)
and (MEM/WB.RegisterRd = ID/EX.RegisterRs))
This condition detects a data hazard between an instruction in the WB stage and an instruction in the EX stage.

State the conditions necessary to detect a data hazard between a branch instruction in the ID stage and an R-format instruction in the MEM stage. You need to only state the necessary conditions to detect a data hazard for the *$rs* register in the ID stage. There are no forwarding control bits that need to be set.

(if (MEM/WB.RegWrite)
and (MEM/WB.Register Rd ! =0)
and (ID/EX .RegisterRs = MEM/WB .RegisterRd)).