Christofer P Paes
CSC 230 GCC
Kurt Marley
12/16/21

1. Dependence detection

This question covers your understanding of dependences between instructions. Using the code below, list all of the dependence types (read after write-RAW, write after read-WAR, write after write-WAW). List the dependences in the respective table. Look ahead to the next TWO instructions. (example INST-X to INST-Y) by writing in the instruction numbers involved with the dependence. Include the letter of the dependency. You need only “look ahead” to the next two instructions.

```
I0: A = C * D;
I1: B = F / A;
I2: B = B + C;
I3: A = A - B;
I4: D = A + B;
1                   2         3            4         5
InstructionMemory---REREG------ALU/Res----DataMem----WriREG
                     C  *D
A

-----------------InstructionMemory---REREG------ALU/Res----
                                                F/A
 5           6
DataMem----WriREG
             B
                          3                    4          5
--------------------------InstructionMemory---REREG------AL
U/Res----                                      B  +  C

6            7
DataMem----WriREG
             B

                                               4
---------------------------------------InstructionMemory---

 5         6       7            8
REREG------AL----  DataMem----WriREG
  A-b
                               A

                                               5
------------------------------------------InstructionMemory
---
 6        7       8            9
```

REREG------AL---- DataMem----WriREG

A+b

RAW Dependence		WAR Dependence		WAW Dependence	
From Instr	To Instr	From Instr	To Instr	From Instr	To Instr
I0 A	I1			I1B	I3
I1 B	I3				
I1 B	I2				
I2 B	I4				
I2B	I3				

2. What double precision floating point number is represented by the following 64-bit binary? Show your work.

```
0x403E60000000000
```

0100000000111110011000000000000000000000000000000000000000

30.375

3. What instruction is represented by the binary? Show your work:

```
0xA648FFF8
op     rt    offset          rs
101001 10010 0100011111111111 11000
```

shw $s2, 18431 ($t8)

4. Consider the following assembly language code:

```
I0: lw   $t7,($s0)
I1: bnez $t7, loop
I2: add  $t1,$t1,$s0
I3: beq  $t1, exit
I4: add  $t1,$t1,$t5
I5: lw   $s0, value
I6: add  $s1,$s0,$s1
I7: add  $t1,$t1,$s0
```

Consider a pipeline with forwarding, hazard detection, and 1 delay slot for branches. The pipeline is the typical 5-stage IF, ID, EX, MEM, WB MIPS design. For the above code, complete the pipeline diagram below (instructions on the left, cycles on top) for the code. Insert the characters IF, ID, EX, MEM, WB for each instruction in the boxes. Assume that there two levels of forwarding/bypassing, that the second half of the decode stage performs a read of source registers, and that the first half of the write-back stage writes to the register file.

Label all data stalls (Draw an X in the box). Label all data forwards that the forwarding unit detects (arrow between the stages handing off the data and the stages receiving the data). What is the final execution time of the code?

Side note:: this was difficult to get the arrows to align with the table columns properly, so I left some notes to be a little more explicit. My apologies.

	T0	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	T11	T12	T13	T14	T15	T16
I0 `lw $t7,($s0)`	IF	ID	EX	MEM ↘ mem to execute	WB												
I1 `bnez $t7, loop`		IF	ID	EX forward	MEM	WB											
I2 `add $t1,$t1,$s0`			IF	ID	EX ↘	MEM ↘	WB										
I3 `beq $t1, exit`				IF	ID	X	EX forward	MEM	WB								
I4 `add $t1,$t1,$t5`					IF	x	ID	EX	MEM	WB							

I5 `lw $s0, value`							IF	ID	EX	MEM	WB						
I6 `add $s1,$s0,$s1`								IF	ID	X	EX fowr ad from me m	M E M	WB				
I7 `add $t1,$t1,$s0`									IF	ID	EX forw ard from ex	M E M	WB				

```
Total Cycles used: 12 cycles used
```

5. Branch Prediction. Consider the following sequence of actual outcomes for a single static branch. T means the branch is taken. N means the branch is not taken. For this question, assume that this is the only branch in the program.

N T N T N T N T N T N T T N N T N N

Assume that we try to predict this sequence with a Branch History Table (BHT) using one-bit counters. The counters in the BHT are initialized to the N state. Which of the branches in this sequence would be mis-predicted? Answer yes or no under **Misprediction**.

You may use this table for your answer:

Predictor state before prediction	**Branch outcome**	**Misprediction?**
N	N	no
N	T	yes
T	N	yes
N	T	yes
N	N	no
N	T	yes
T	N	yes
N	T	yes
T	N	yes
T	T	no
T	N	yes
N	T	yes
T	T	no
T	N	yes
N	N	no
N	T	yes
T	N	yes
N	N	no

6. Given a **REP MOVBS** 80386 assembly instruction, list the FIVE steps that happen each time this instruction is executed, along with the register used in each step. Assume an STD instruction has been executed first. (I will accept SIX steps too.)
I believe there is a typo and you mean REP MovSB , which means repeat move string, bytes.

1. First the instruction is fetched from the ds:si register
2. Then stores the instruction at es:di .
3. The third stage is to increment or decrement the si and di registers, which is done one by one.
4. The fourth stage is to check ex if it is zero, if not
5. move the values from ds:si register to es: di register
6. decrement cx register and repeat

7. Identify the stalls.

CODE SEQUENCE	ANSWER “STALL” or “NO”

`add $1,$1,$2`
`add $1,$1,$3`
`add $1,$1,$4`
	stall

`add $1, $1, $2`
`lw $1, ($7)`
`beq $1, $0, label`
	stall

`sub $2, $1,$3`
`and $12,$2,$5`
`or $13,$6,$2`
`add $14,$2,$2`
`sw $15,100($2)`
	stall

`lw $7, 4($14)`
`ori $6, $6, 3`
	no

`add $12, $1, $1`
`sub $2, $12, $3`
	stall

`add $1, $1, $2`
`lw $1, ($7)`
`add $2, $3, $4`
`beq $1, $0, label`
	no

`Add $1, $2, $3`
`sub $2, $4, $5`
	no

`lw $1, ($2)`
`addi $2, $1, 4`
	stall