Problem Set 3

Due: October 17th @ 4:59 pm (Before class)

Department of Electrical and Computer Engineering
The University of Texas at Austin
EE 382N.1, Fall 2018
Instructor: Dam Sunwoo
TA: Pritesh Chhajed

Instructions

You are encouraged to work on the problem set in small groups (3 or 4 per group) and turn in one problem set for the entire group on Gradescope. Create a copy of this file to fill in your answers within the spaces provided. If you need more space than given box, make a comment there and use extra work space at end of file. There are extra pages provided for the same.

Before submitting, convert it into a PDF file. Remember to put all your names and eid in the box below. The person submitting must choose everyone in the group.

You will need to refer to the assembly language handout and the LC-3b ISA, microarchitecture, and state diagram documents on the course website.

Student names and EID

Questions

Problem 1
Suppose we have the following loop executing on a pipelined LC-3b machine.

```
DOIT    STW   R1, R6, #0
        ADD   R6, R6, #2
        AND   R3, R1, R2
        BRz   EVEN
        ADD   R1, R1, #3
        ADD   R5, R5, #-1
        BRp   DOIT
EVEN    ADD   R1, R1, #1
        ADD   R7, R7, #-1
        BRp   DOIT
```

Assume that before the loop starts, the registers have the following **decimal** values stored in them:

Register	**Value**
R0	0
R1	0
R2	1
R3	0
R4	0
R5	5
R6	4000
R7	5

The fetch stage takes **one** cycle, the decode stage also takes **one** cycle, the execute stage takes a variable number of cycles depending on the type of instruction (see below), and the store stage takes **one** cycle.

All execution units (including the load/store unit) are fully pipelined and the following instructions that use these units take the indicated number of cycles:

Instruction	**Number of Cycles**
STW	3
ADD	3
AND	2
BR	1

Data forwarding is used wherever possible. Instructions that are dependent on the previous instructions can make use of the results produced right after the previous instruction finishes the execute stage. Multiple Instructions can write the results to the register file concurrently in the same cycle.

The target instruction after a branch can be fetched when the BR instruction is in ST stage. For example, the execution of an ADD instruction followed by a BR would look like:

ADD	F	D	E1	E2	E3	ST		
BR		F	D	-	-	E1	ST	
TARGET							F	D

The pipeline implements “in-order execution.” A scoreboarding scheme is used as discussed in class.

Answer the following questions:

1. How many cycles does the above loop take to execute if no branch prediction is used?

2. How many cycles does the above loop take to execute if all branches are predicted with 100% accuracy?

3. How many cycles does the above loop take to execute if a static BTFN (backward taken-forward not taken) branch prediction scheme is used to predict branch directions?

Clarification A BTFN (backward taken-forward not taken) predictor will predict a branch taken if the branch location is to a PC less than the branch instruction (“backwards” in addresses), and will predict a branch not taken if the branch location is to a PC greater than the branch instruction (“forwards” in addresses).

4. What is the overall branch prediction accuracy? What is the prediction accuracy for each branch?

Problem 2

Consider the following program:

```
        AND R0, R0, #0
        BRz A
        ADD R1, R0, #-1
    C   BRp B
    A   ADD R2, R0, #1
        BRp C
    B   RET
```

A processor implements the GAg (Global History Register, Global Pattern History Table) branch predictor as part of its microarchitecture. Assume the Branch History Register and Pattern History Table are as shown below before the program is run. The direction of the most recent branch is the right-most bit of the BHR; i.e., 1=taken, 0=not taken.

Branch History Register

101

Branch History Table

0	10
1	00
2	10
3	01
4	10
5	11
6	00
7	01

1. How many times does the branch predictor predict correctly?

2. What does the Branch History Register look like after the program finishes? The Branch History Table?

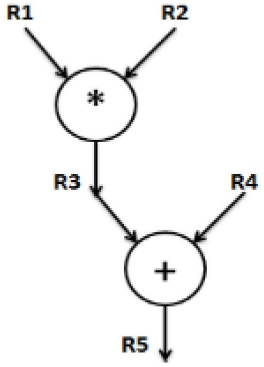
Problem 3

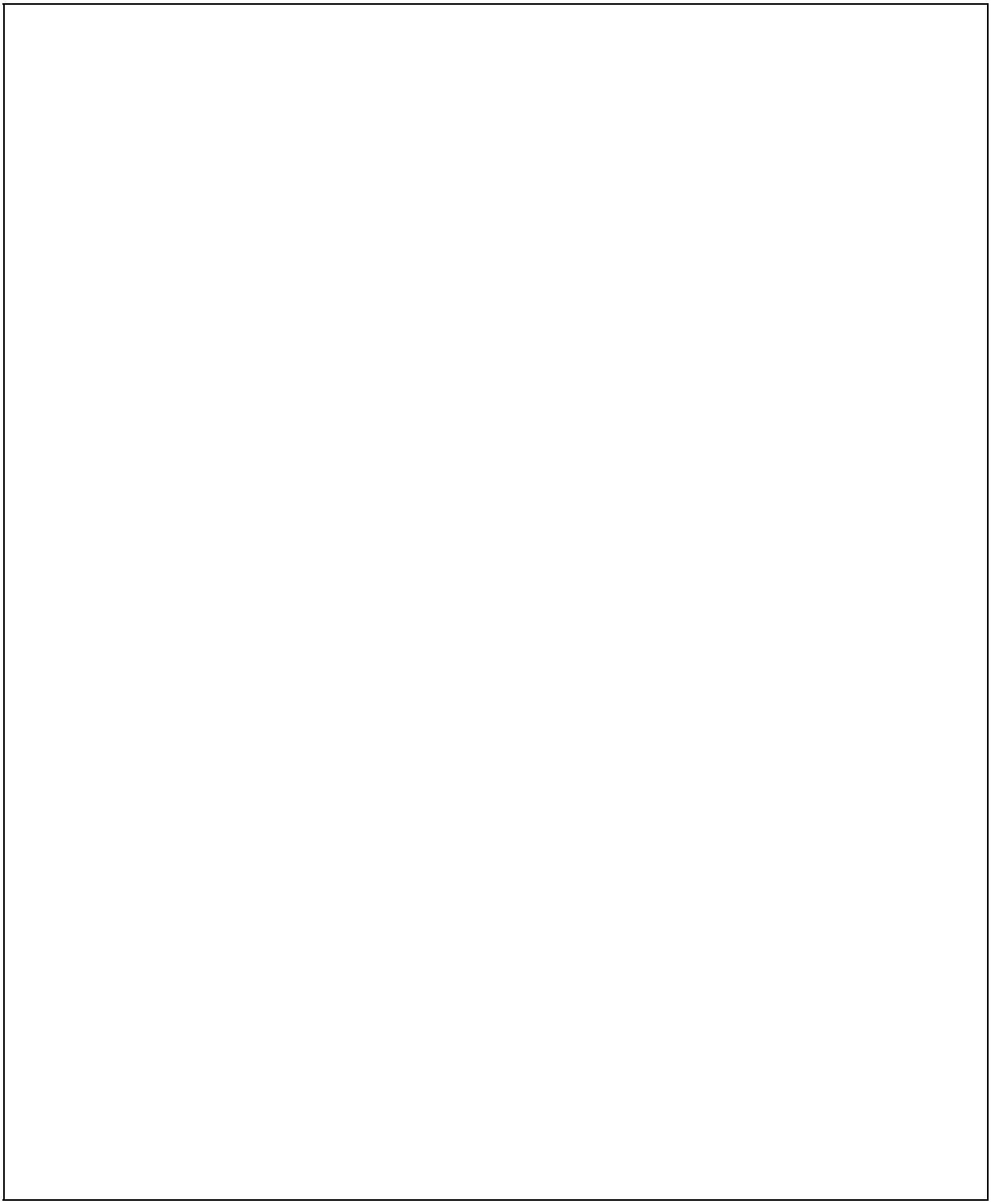
Consider the following example used to explain Tomasulo's Algorithm:

```
MUL R3,  R1, R2
ADD R5,  R3, R4
ADD R7,  R2, R6
ADD R10, R8, R9
MUL R11, R7, R10
ADD R5,  R5, R11
MUL R10, R4, R10
```

Construct the Data Flow Graph for this program.

Hint We've done the first two instructions for you, seen below.





Problem 4

For the given assembly program, reverse-engineer the microarchitecture of the Tomasulo pipeline that would execute it as the execution timeline shown. Find the minimum cost solution (minimum number of reservation station entries, pipeline registers and etc).

Things that are known about the microarchitecture:

- There are one ADD unit, one MUL unit, and one DIV unit.
- The pipeline stages are IF, ID, (EXE/MEM), WB, where the latency of EXE/MEM stage varies among functional units.

Instr.	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
R1=R2*R3	IF	ID	M	M	M	M	M	M	M	WB													
R2=R1+R1		IF	ID	RS	RS	RS	RS	RS	RS	A	A	A	WB										
R4=R1+R2			IF	ID	RS	RS	RS	RS	RS	RS	RS	RS	A	A	A	WB							
R0=R0+R3				IF	ID	A	A	A	WB														
R3=R2+R3					IF	ID	ID	ID	ID	RS	RS	RS	RS	A	A	A	WB						
R5=R2*R5						IF	IF	IF	IF	ID	RS	RS	M	M	M	M	M	M	M	WB			
R1=R1/R0										IF	ID	D	D	D	D	D	D	D	D	D	D	WB	
R7=R5+R7											IF	ID	RS	RS	RS	RS	RS	RS	RS	A	A	A	WB
R6=R0/R2												IF	ID	RS	RS	RS	RS	RS	RS	RS	RS	D	D
R0=R2*R2													IF	ID	M	M	M	M	M	M	M	WB	

*A represents the ADD stage, M represents the MUL stage, and D represents the DIV stage.

1. What are the latencies of each functional unit?

2. What stages can be bypassed?

3. What would be the first instruction that would execute differently if your answer to the above was not the case?

4. Are reservation stations centralized (shared among all functional units) or distributed (assigned to specific functional units)? Explain why the other can't be the case.

5. What would be the first instruction that would execute differently if your answer to the above was not the case? Assume the same number of RS entries total.

6. How many reservation station entries are there? For centralized RS, show total number of entries. For distributed RS, show how many RS entries each FU has. Find the minimum cost solution.

Problem 5

Consider a fully-bypassed classic Tomasulo machine. There is an adder, a multiplier and a divider, each with its own single entry reservation station. All functional units are unpipelined. The register file has infinite read ports but only one write port and there is only one CDB. There are Fetch, Decode, and Writeback stages. In the case of a conflict, the oldest instruction is always scheduled first.
Note: "fully-bypassed classic Tomasulo machine" means that an instruction can bypass the reservation station too and be scheduled on the functional unit if it is free.

Do the following:
1. Determine the latency of each of the functional units
2. Complete the timing diagram, indicating what stage/functional unit each instruction is in for each cycle (you must fill in every box between F and W in each row)

Functional Unit	**Latency**
Adder	
Multiplier	
Divider	

	1	2	3	4	5	6	7	8	9	10	11	12	13	14
R0 = R1 + R2	F						W							
R0 = R0 * R0		F								W				
R1 = R1 / R2			F					W						
R2 = R3 * R2				F					W					
R3 = R1 / R3					F						W			

R0 = R0 / R2						**F**							**W**	
R0 = R2 + R2							**F**							**W**

Problem 6

A five instruction sequence executes according to Tomasulo's algorithm. Each instruction is of the form ADD DR,SR1,SR2 or MUL DR,SR1,SR2. ADDs are pipelined and take 9 cycles (F-D-E1-E2-E3-E4-E5-E6-WB). MULs are also pipelined and take 11 cycles (two extra execute stages). The microengine must wait until a result is in a register before it sources it (reads it as a source operand)

The register file before and after the sequence are shown below (tags for "After" are ignored).

Before

	V	tag	value
R0	1	z	4
R1	1	z	5
R2	1	z	6
R3	1	z	7
R4	1	z	8
R5	1	z	9
R6	1	z	10
R7	1	z	11

After

	V	tag	value
R0	1		310
R1	1		5
R2	1		410
R3	1		31
R4	1		8
R5	1		9
R6	1		10
R7	1		21

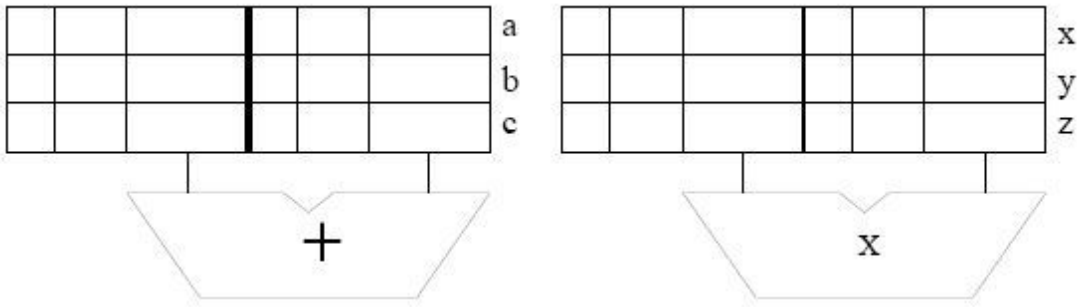
1. Complete the five instruction sequence in program order in the space below. Note that we have helped you by giving you the opcode and two source operand addresses for instruction 4. (The program sequence is unique.)

1	
2	
3	
4	MUL ☐, R6, R6
5	

2. In cycle 1 instruction 1 is fetched. In cycle 2, instruction 1 is decoded and instruction 2 is fetched. In cycle 3, instruction 1 starts execution, instruction 2 is decoded, and instruction 3 is fetched.

Assume the reservation stations are all initially empty. Put each instruction into the next available reservation station. For example, the first ADD goes into “a”. The first MUL goes into “x”. Instructions remain in the reservation stations until they are completed. Show the state of the reservation stations at the end of cycle 8.

Note: to make it easier for the grader, when allocating source registers to reservation stations, please always have the higher numbered register be assigned to SR2.



3. Show the state of the Register Alias Table (V, tag, Value) at the end of cycle 8.

	V	tag	value
R0			
R1			
R2			
R3			
R4			
R5			
R6			
R7