# Problem Set 2

**Due: October 3rd @ 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**

Jiaqi Gu  jg68999
Junhong Tong jt38826
Yuesen Lu yl33489
Wencan Liu wl8784

## Questions

**Problem 1**
In previous years, the LC-3b state diagram handed out in class contained errors in states 4, 20, and 21. We have posted both versions of the handout: wrong and corrected. Briefly explain the problem we have corrected.

For JSRR instruction, its BaseR could be R7, thus we cannot overwrite R7 with PC before using R7 to calculate the next PC. So, we have to move R7 <- PC to state 20 and 21.

**Problem 2**
Answer the following short questions:

1. A memory's addressability is 64 bits. What does that tell you about the sizes of the MAR and the MDR?

We do not know the memory capacity, so we do not know the size of MAR.
We don’t know the size of MDR either, because the size of each memory location is not necessarily of the same size of MDR.

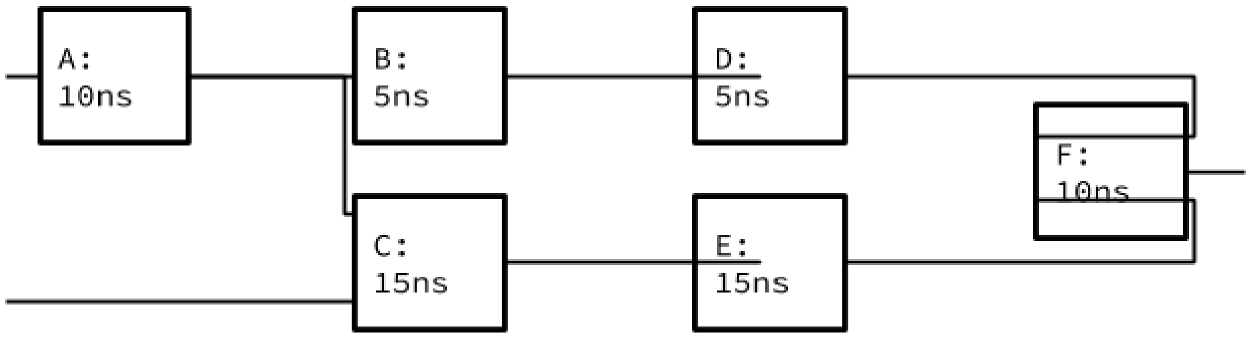
2. We want to increase the number of registers that we can specify in the LC-3b ADD instruction to 32. Do you see any problem with that? Explain.

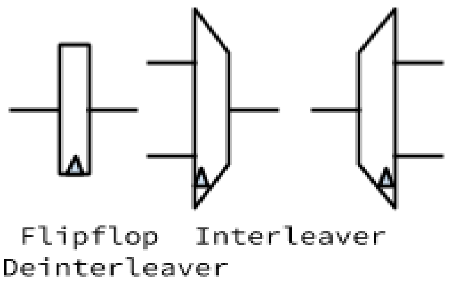
There is a problem. We have to use 5 bits to represent 32 registers. 16 bits instruction are not enough to store 4-bit opcode and three registers.

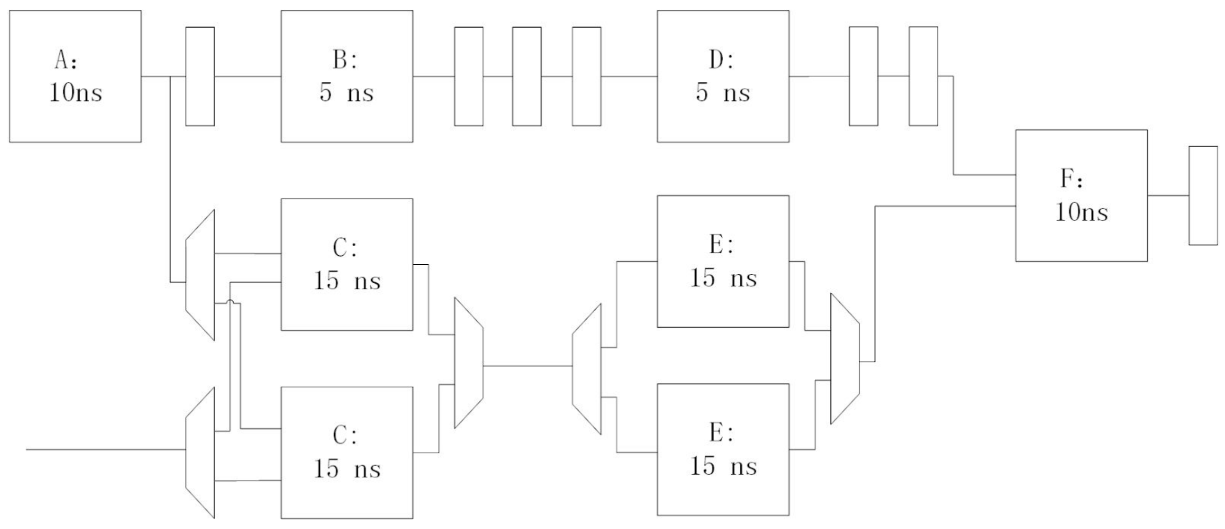
**Problem 3.**

Pipeline the circuit below. Optimize throughput (inputs processed per ns) and cost ($). You may add any number of the blocks in the circuit (A-F), edge-triggered flip-flops, edge-triggered interleavers, and edge-triggered de-interleavers (see figure below). No other modifications are permitted. The latency of flip-flops, interleavers, and de-interleavers is 0ns. The cost of any item (whether already there or one you add), regardless of which item is $1. You have a total budget of $25 and the components already drawn below cost $6. Draw your solution in the space provided below. What is the throughput, latency and cost of your solution?

Prior Test Question (Fall 2017)







We would like to make it a 7-stage pipeline. We notice the bottleneck is C and E, whose latency is 15 ns. We would like to use interleaving to improve throughput. Given cost limitation, we only use 2-way interleaving.

Throughput = 1/10ns

Latency = 70 ns

Cost = $ 20

**Problem 4**

Given below is an instruction stream with a few missing register names. Fill in the proper registers such that the appropriate hazard noted in the comments is caused or vice versa. Assume MULs are much longer than ADDs, and there are no structural hazards on the functional units. Instructions are in the format: *opcode dest, operand1, operand2*.

**MUL R2, R0, R0**
**MUL __, R1, R4 ; this is a WAW hazard**
**ADD R3, R0, __ ; this is a RAW hazard**
**ADD R0, R1, R1 ; this is a ___ hazard**

**MUL R2, R0, R0**
**MUL _R2_, R1, R4 ; this is a WAW hazard**
**ADD R3, R0, _R2_ ; this is a RAW hazard**
**ADD R0, R1, R1 ; this is a _WAR__ hazard**

**Problem 5**

Consider the following snapshots of the register file for an in-order-execution pipelined machine:

| Cycle 0 | | |
|---|---|---|
| | Valid | Value |
| R0 | 1 | 0 |
| R1 | 1 | 5 |
| R2 | 1 | 7 |

| End of Cycle 8 | | |
|---|---|---|
| | Valid | Value |
| R0 | 1 | 10 |
| R1 | 0 | 5 |
| R2 | 0 | 7 |

| After Cycle 11 | | |
|---|---|---|
| | Valid | Value |
| R0 | 1 | 10 |
| R1 | 1 | 17 |
| R2 | 1 | 20 |

The program consists of three ADD instructions and finishes execution after the 11th cycle.

- The ADD instruction takes **three** cycles to execute.
- Fetch, Decode and Store all take **one** cycle.
- The machine has **two** adders
- There is NO data forwarding

1. What was the program that was executed? *Only R0, R1, and R2 were used*.

| | | | |
|---|---|---|---|
| ADD | R0 | R1 | R1 |
| ADD | R1 | R0 | R2 |
| ADD | R2 | R0 | R0 |

2. Complete the following timeline of the program execution. The first instruction has been filled already.

**We assume that register read is at "execute" stage. So instruction 2 can execute right after "store" stage of instruction 1.

| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |
|---|---|---|---|---|---|---|---|---|---|---|
| ***F*** | ***D*** | ***E*** | ***E*** | ***E*** | ***S*** | | | | | |
| | F | D | D | D | D | E | E | E | S | |
| | | F | F | F | F | D | E | E | E | S |

**Problem 6**

Given the following code:

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

*Note: Each instruction is specified with the destination register first.*

Calculate the number of cycles it takes to execute the given code on the following models:

1. A non-pipelined machine

2. A pipelined machine with scoreboarding and five adders and five multipliers.
3. A pipelined machine with scoreboarding and one adder and one multiplier.

*Note: For all machine models, use the basic instruction cycle as follows:*

- Fetch (one clock cycle)
- Decode (one clock cycle)
- Execute (MUL takes 6, ADD takes 4 clock cycles). The multiplier and the adder are not pipelined.
- Write-back (one clock cycle)

Do not forget to list any assumptions you make about the pipeline structure (e.g., data forwarding between pipeline stages). In fact, we encourage you to solve the above mentioned questions with data forwarding as well, but, you are not required to do so.

1. For non-pipelined structure: 3 * 9 + 3 * 7 = 48 cycles
2. For pipelined structure:
   Assumption
   (1)no data forwarding
   (2)single write ports for register file

   It takes 21 cycles

3. Assumption
   (1)no data forwarding
   (2)single write ports for register file

   It takes 27 cycles