# Problem Set 2 Solutions

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

## 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.

In the wrong handout, the instruction JSRR R7 would not execute correctly as the value of R7 would have been overwritten with the value of PC in state 4 itself, and hence, PC would move to the wrong address in state 20

**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?

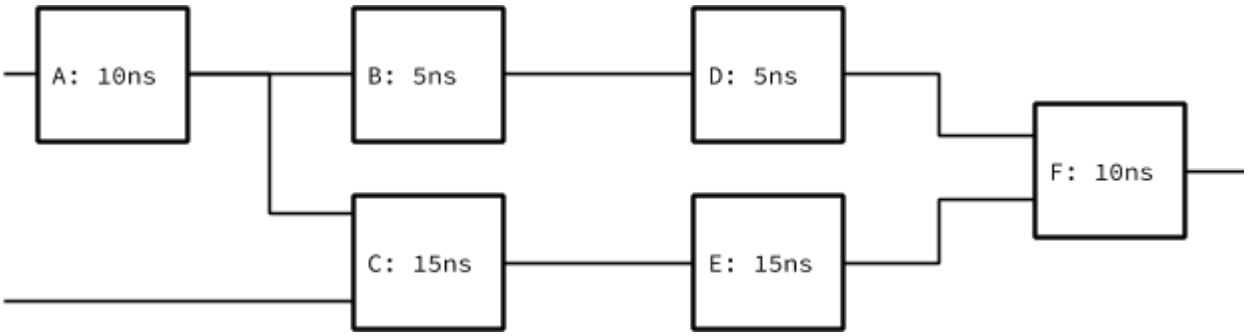
We cannot tell the size of the MAR, since it depends on the number of memory locations and does not depend on only the addressability (the number of bits in each location).
For the MDR, first consider the LC3b. The LC3b is Byte (8 bit) addressable with an MDR size of 16 bits. We cannot tell the size of the MDR based on addressability either.

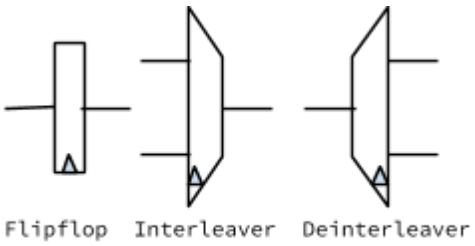
2. We want to increase the number of registers that we can specify in the LC-3b ADD instruction to 32. Do you see any problem with that? Explain.

Each register (DR, SR1, and SR2) would have to be specified with five bits. With the steering bit, the total number of bits used would be 16 – leaving no bits for the opcode

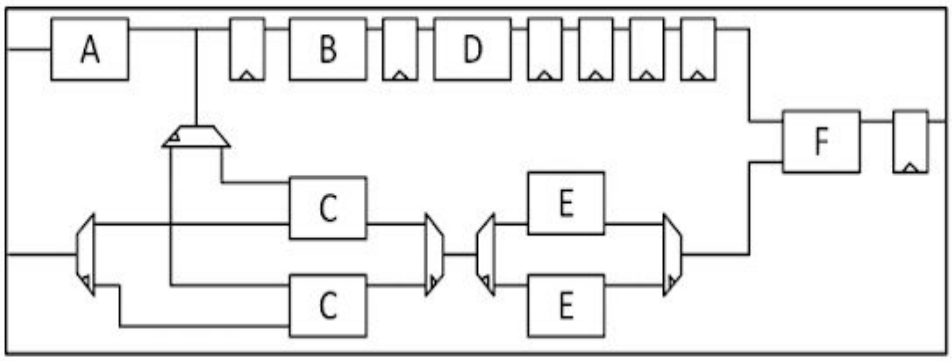
**Problem 3.**

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)



One of the possible solution:




Throughput = $\frac{1}{10\ ns}$

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<br>MUL R2, R1, R4<br>ADD R3, R0, R2<br>ADD R0, R1, R1 | <br>; this is a WAW hazard<br>; this is a RAW hazard<br>; 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.

| 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. ADDs require 7 cycles (fetch, decode, 4 execute, write-back), and MULs require 9 cycles (fetch, decode, 6 execute, write-back). For 3 ADD instructions and 3 MUL instructions, the execution time is 3 × 7 + 3 × 9 = 48 cycles.

2. A pipelined machine using scoreboarding with 1 adder and 1 multiplier. (Use classic CDC6600 scoreboarding. The one shown in class)
No bypass: 27 cycles
Bypass: 26 cycles

Notation for the following timing diagrams:
*s indicates a stall
bypass : forward data from the W stage to a stage before, so an instruction does not need to stall until until after the W stage completes

No bypass: 27 cycles

| 1 ADD, 1 MUL | no bypass | | | | | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 |
| MUL R3,R1,R2 | F | D | M | M | M | M | M | M | W | | | | | | | | | | | | | | | | | | |
| ADD R5,R4,R3 | | F | D | As | As | As | As | As | As | A | A | A | A | W | | | | | | | | | | | | | |
| ADD R6,R4,R1 | | | F | D | Ds | Ds | Ds | Ds | Ds | Ds | Ds | Ds | Ds | A | A | A | A | W | | | | | | | | | |
| MUL R7,R8,R9 | | | | F | Fs | Fs | Fs | Fs | Fs | Fs | Fs | Fs | Fs | D | M | M | M | M | M | M | W | | | | | | |
| ADD R4,R3,R7 | | | | | | | | | | | | | | F | D | Ds | Ds | As | As | As | As | A | A | A | A | W | |
| MUL R0,R5,R6 | | | | | | | | | | | | | | | F | Fs | Fs | D | Ds | Ds | M | M | M | M | M | M | W |

Bypass: 26 cycles

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 ADD, 1 MUL | bypass | | | | | | | | | | | | | | | | | | | | | | | | | |
| MUL R3,R1,R2 | F | D | M | M | M | M | M | M | W | | | | | | | | | | | | | | | | | |
| ADD R5,R4,R3 | | F | D | As | As | As | As | As | A | A | A | A | W | | | | | | | | | | | | | |
| ADD R6,R4,R1 | | | F | D | Ds | Ds | Ds | Ds | Ds | Ds | Ds | Ds | A | A | A | A | W | | | | | | | | | |
| MUL R7,R8,R9 | | | | F | Fs | Fs | Fs | Fs | Fs | Fs | Fs | Fs | D | M | M | M | M | M | M | W | | | | | | |
| ADD R4,R3,R7 | | | | | | | | | | | | | F | D | Ds | Ds | As | As | As | A | A | A | A | W | | |
| MUL R0,R5,R6 | | | | | | | | | | | | | | F | Fs | Fs | D | Ds | Ds | M | M | M | M | M | M | W |

3. A pipelined machine using scoreboarding with 5 adders and 5 multipliers. (Use classic CDC6600 scoreboarding.)
No bypass: 21 cycles
Bypass: 19 cycles

No bypass: 21 cycles

| 5 ADD, 5 MUL | no bypass | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 |
| MUL R3,R1,R2 | F | D | M | M | M | M | M | M | W | | | | | | | | | | | | |
| ADD R5,R4,R3 | | F | D | As | As | As | As | As | As | A | A | A | A | W | | | | | | | |
| ADD R6,R4,R1 | | | F | D | A | A | A | A | As | W | | | | | | | | | | | |
| MUL R7,R8,R9 | | | | F | D | M | M | M | M | M | M | W | | | | | | | | | |
| ADD R4,R3,R7 | | | | | F | D | As | As | As | As | As | As | A | A | A | A | W | | | | |
| MUL R0,R5,R6 | | | | | | F | D | Ms | Ms | Ms | Ms | Ms | Ms | Ms | M | M | M | M | M | M | W |

Bypass: 19 cycles

| 5 ADD, 5 MUL | bypass | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 |
| MUL R3,R1,R2 | F | D | M | M | M | M | M | M | W | | | | | | | | | | |
| ADD R5,R4,R3 | | F | D | As | As | As | As | As | A | A | A | A | W | | | | | | |
| ADD R6,R4,R1 | | | F | D | A | A | A | A | As | W | | | | | | | | | |
| MUL R7,R8,R9 | | | | F | D | M | M | M | M | M | M | W | | | | | | | |
| ADD R4,R3,R7 | | | | | F | D | As | As | As | As | As | A | A | A | A | W | | | |
| MUL R0,R5,R6 | | | | | | F | D | Ms | Ms | Ms | Ms | Ms | M | M | M | M | M | M | W |

**Other possible solution:**

Assumption:
Assuming we have a in-order execution pipeline. Scoreboarding doesn't make a difference, just a mechanism to detect dependencies.

Assuming the standard pipeline model we have been using:

1. 48 cycles

For 5 adders and multipliers
Without data-forwarding-

| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| F | D | E | E | E | E | E | E | S | | | | | | |
| | F | D | D | D | D | D | D | D | E | E | E | E | S | |
| | | F | F | F | F | F | F | F | D | E | E | E | E | S |
| | | | | | | | | | F | D | E | E | E | E |
| | | | | | | | | | | F | D | D | D | D |
| | | | | | | | | | | | F | F | F | F |

| 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | |
| E | E | S | | | | | | | | | | | | |
| D | D | D | E | E | E | E | S | | | | | | | |
| F | F | F | D | E | E | E | E | E | E | S | | | | |

2. 26 cycles.
3. 29 cycles (Instruction 3 is delayed by 3 cycles as it needs to wait for the previous addition to complete.

With data-forwarding-
2. 24 cycles (1 cycles saved for the dependencies between Instructions 1 & 2, and Instructions 4 & 5.
3. 27 cycles.