Introduction to Computer Organization – Fall 2024

Homework 2

Due: @11:55PM, Monday, October 7th

98

Name: ____Qiulin Fan_____ Uniqname: ____rynnefan_____

1. Submit a pdf of your typed or handwritten homework on Gradescope.

2. Your answers should be neat, clearly marked, and concise. Typed work is recommended, but not required unless otherwise stated. Show all your work where requested, and state any special or non-obvious assumptions you make.

3. You may discuss your solution methods with other students, but the solutions you submit must be your own.

4. **Late Homework Policy:** Submissions turned in by 1:00 am the next day will be accepted but with a 5% penalty. Assignments turned in between 1:00 am and 11:55 pm will get a 30% penalty, and any submissions made after this time will not be accepted.

5. When submitting your answers to Gradescope you need to indicate what page(s) each problem is on to receive credit. The grader may choose not to grade the homework if answer locations are not indicated.

6. After each question (or in some cases question part), we've indicated which lecture number we expect to cover the relevant material. So "**(L7)**" indicates that we expect to cover the material in lecture 7 (but depending on your lecture, it may have been covered later)

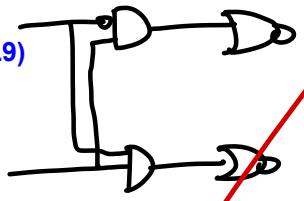
7. **The last question is a group question**.
    - You may do it in a homework group of up to two students including yourself (yes, you can do them by yourself if you wish).
    - If you work in a group of two for these questions, list the name of the student you worked with in your assignment. Further, we suggest that you not split these problems up but rather work on the problem as a group.
    - Turn these group questions in as part of your individual submission.
    - **For these questions (and these questions ONLY) you are allowed to copy/paste solutions from the other student in your homework group.**
    - <u>It is an honor code violation if a student is listed as contributing who did not actually participate in working on that problem.</u>

Problem 1: Short answer questions (20 points)

1. An FSM has 3 input bits, 18 states, and 4 output bits . What total size ROM is needed to implement the FSM? **(8 points)** (L10)

2. Global variables do appear in the symbol table of an object file **(2 points)** (L7)
   Circle One: **True** or **False**

3. A D-latch can be made out of **(select all that apply) (4 points)** (L9)
   a. 3x NOR gates and 1x NOT gate
   b. 2x NOR gates, 1x AND gate, and 1x NOT gate
   c. 3x NOR gates, 2x NOT gate
   d. 2x NOR gates, 2x AND gate, and 1x NOT gate
   e. None of the above



4. A D-Latch only updates its value on the rising edge of the clock signal. **(2 points)** (L9)
   Circle One: **True** or **False**

5. Caller save is always better than callee save because in caller save, leaf functions do not need to save registers even if they use them (Leaf functions do not call other functions) **(2 points)** (L6)
   Circle One: **True** or **False**

6. When building a single cycle processor, the clock period must be at least as long as the execution latency of the longest instruction. **(2 points)** (L10)
   Circle one: **True** or **False**

(0) lw 0 1 3
(1) lw 0 2 4
(2) nor 1 2 5
(3) .fill 5
(4) fill 7

Problem 2: I've had my fill of LC2K (10 points)

Using only .fill commands, write an LC2K program which loads the value 5 into reg 1, loads the value 7 into reg 2 and computes bitwise NOR of those two values, leaving the result in register 5 in as few lines as possible (you need to actually have the computer NOR those two values together!)

(L5)

```
.fill 8454148
.fill 8519685
.fill 4849669
.fill 25165824
.fill 5
.fill 7
```

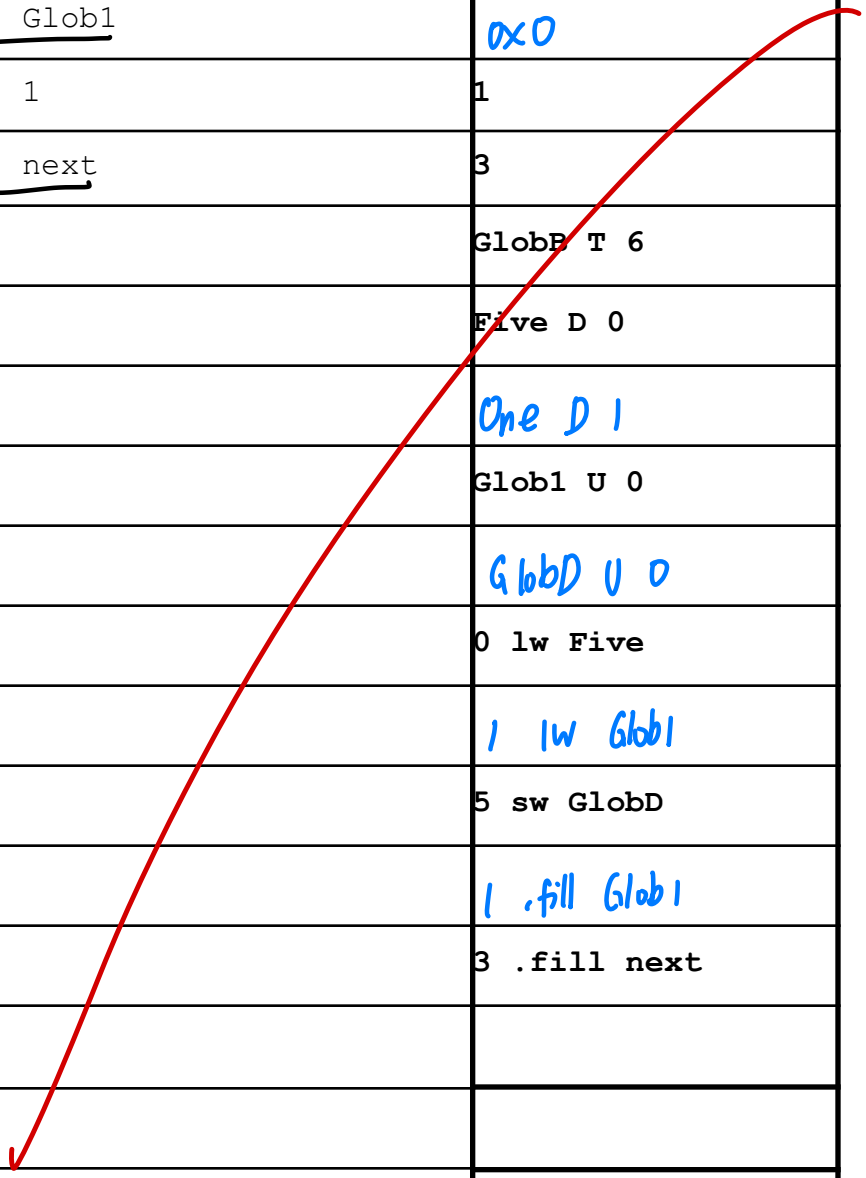
Problem 3: Link Like You've Never Linked Before! (12 Points)

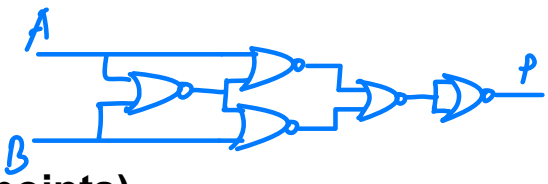
(L11)

Project 2 spec deals with linking in LC2K. Below is an LC2K assembly program and a partially complete object file. **Note: the machine code should be completed in hexadecimal, not decimal.** You are to fill in all the blanks of the object file (don't forget the first line). It is possible that one or more lines should remain blank.

Main.as	Main.obj
	9 4 5 5
start lw 0 2 Five	0x820009
lw 0 1 Glob1	0x810000
beq 0 1 GlobB	0x1010003
next add 2 2 2	0x120002
beq 0 0 start	0x0100FFFB
sw 0 1 GlobD	0x C10000
GlobB nor 1 2 1	0x4A0001
jalr 4 7	0x1870000
halt	0x1800000

Five .fill 5	0x5
One .fill Glob1	0x0
.fill 1	1
.fill next	3
	GlobB T 6
	Five D 0
	One D 1
	Glob1 U 0
	GlobD U 0
	0 lw Five
	1 lw Glob1
	5 sw GlobD
	1 .fill Glob1
	3 .fill next





Problem 4: Just a Bunch of Gates (14 points) (L9)

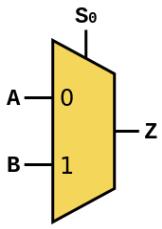
1. Design a circuit that takes in two 1-bit inputs. If the two inputs are different then it outputs 1 otherwise it outputs a 0. You may only use only NOR gates. (You may express your answer either as a circuit diagram or using a Boolean expression) (4 points)

A xor B = ((A nor (A nor B)) nor (B nor (A nor B))) nor ((A nor (A nor B)) nor (B nor (A nor B)))

2. Design an AND gate out of 2-input NAND gates. (2 points) nor A = A nand A

A and B = not (A nand B) = (A nand B) nand (A and B)

3. Fill in the truth table below for the 2-to-1 Mux below (4 points)

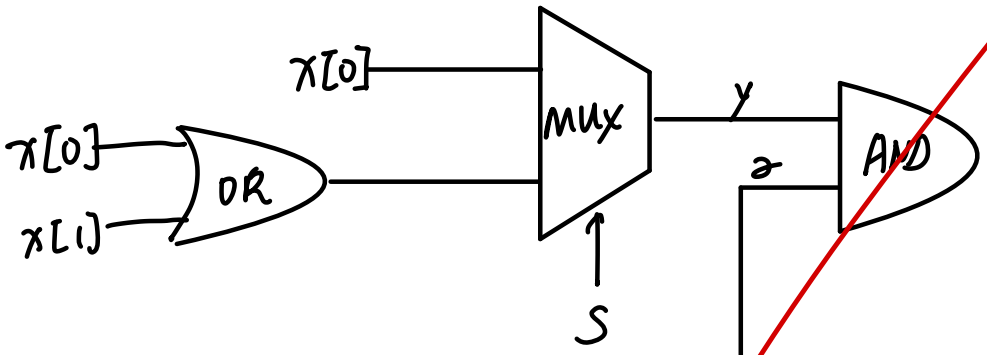


S	A	B	Z
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

4. Given the following code, design a circuit which has 34 bits of input (X[31:0] and the select bit S and the single bit Z), and has 1 bit of output (Y). For this problem you should consider the input X to be an unsigned number.

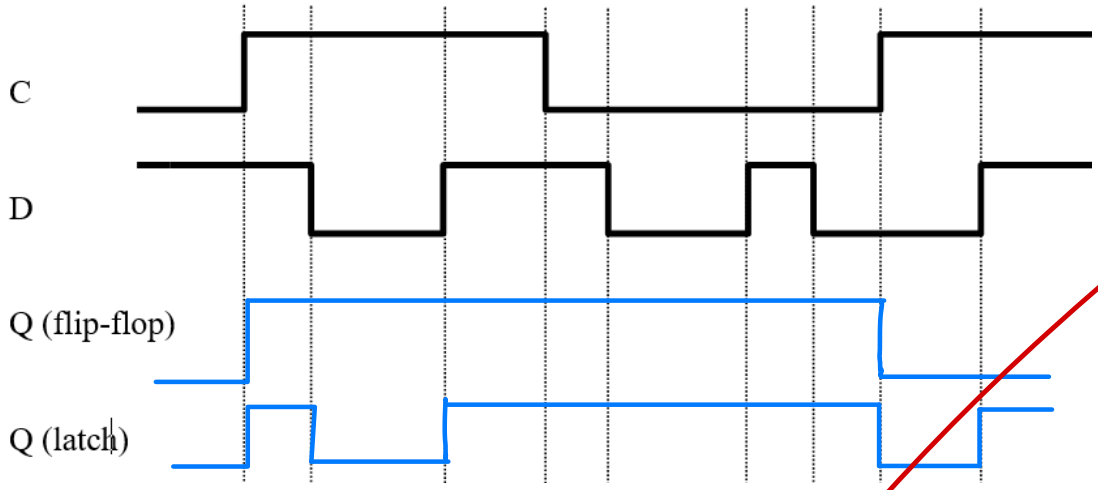
   You can use a single bit 2x1 mux, one AND gate and one OR gate. (4 points)

```
bool HW3Func(uint_t32 x, bool S, bool Z){
    bool y;
    if(S){
      y = (x % 4 != 0);
    }
    else{
      y = (x % 2 != 0);
    }
    return y & Z;
}
```



Problem 5: Latches and flip-flops (8 points)

Complete the timing diagrams for a D latch and D flip-flop given the provided inputs. The flip-flop is a positive-edge triggered. For the latch we are using “C” for the “G” input (which is fairly common). Assume both the latch and the flip-flop have an initial value of zero. **(L9)**



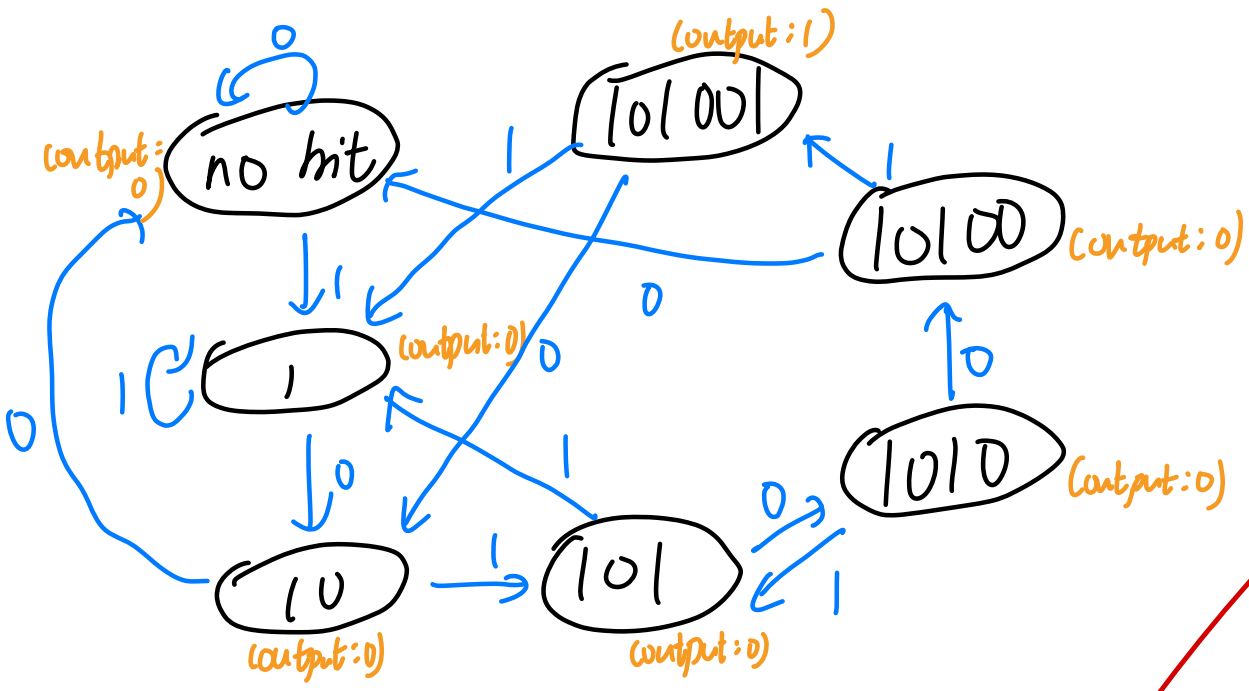
Problem 6: Finite State Machine (16 Points) (L10)

Design a Moore machine where the output Y goes high (=1) when the last six bits of the input X were 101001:

6th to last bit seen = 1
5th to last bit seen = 0
4th to last bit seen = 1
3rd to last bit seen = 0
2nd to last bit seen = 0
Last bit seen = 1

Your machine must be designed to use the least number of states possible. Your reset state should be a starting state in which no bits of X have been read. Your FSM should then proceed to read an infinite number of bits producing an output of 1 whenever the 6 most recent bits match the pattern 101001 and otherwise to produce an output of 0.

1. Use this page to draw your FSM (6 points)



2. Complete the following state transition table based on your FSM. (10 points)

State	Next State (Input X = 0)	Next State (Input X = 1)	Output
no bit	not bit	1	0
1	10	1	0
10	not bit	101	0
101	1010	1	0
1010	10100	101	0
10100	not bit	101001	0
101001	10	1	1

Collaborator: Zifei Bai

lw R 1 W 2 R 5 10

Problem 7: SC Datapath Performance (20 Points) [Group] (L10)

Consider the single-cycle datapath from lecture with the following delays:

Read memory	10 ns
Write memory	16.5 ns
Read register file	4 ns
Write register file	7 ns
ALU	5 ns
All other operations	0 ns

lw: 10+4+10+5+7 = 36
sw: 10+4+5+16.5 = 35.5
add: 10+4+5+7 = 26
nor: 26

1. What is the clock period for the single-cycle processor if we only had to support add, nor, beq and sw? [5]

   35.5 ns (sw highest)

2. What is the clock period for the single-cycle processor if we support all LC2K instructions except jalr? [5]

   10+4+10+5+7 = 36 ns (now lw highest)

3. If we can decrease the delay for one of the operations by 10%, which operation should we apply it to in order to improve performance the most? What is the clock period after improvement? [5]

lw: 9+4+5+9 = 34 ns
sw: 9+4+5+16.5 = 34.5

Read from memory. Since reading from memory takes highest cost in the process of the higher cost instruction (and every inst has one), the reduction on delay by applying the 10% decrease on reading memory will be optimal, better than applying that to other operations

The clock period will be 36-1-1= 34ns after improvement

Consider a benchmark of 1000 instructions. The instructions consist of:

instruction	percentage
add / nor	35%
sw	20%
lw	25%
beq	15%
noop / halt	5%

With the original latency, what is the execution time of this benchmark on the single-cycle processor? [5]

36 x 1000 = 36000 ns