CS 104 PRACTICE Midterm Exam 2

This is a full length practice midterm exam. If you want to take it at exam pace, give yourself 75 minutes to take the entire test. Just like the real exam, each question has a point value. There are 75 points in the exam, so that you can pace yourself to average 1 point per minute (some parts will be faster, some slower).

Questions:

1. Vocabulary [5 points]
2. Coding [10 points]
3. Boolean Algebra + Gates [10 points]
4. Flipflops [10 points]
5. Performance [10 points]
6. Datapaths [10 points]
7. Pipeline diagrams [10 points]
8. Short-answer [10 points]

Question 1: Vocabulary [5 pts]

Match each of the following definitions with the appropriate vocab word:

1. A set of states and the transitions between them.
2. Key part of the interface between hardware and software
3. This type of gate is true if one of two inputs is true (but not both)
4. A circuit capable of adding, subtracting, comparing, and performing bit-wise operations.
5. The pipeline stage in which the PC is used to read instruction memory.

A Abstraction

B ADD

C AND

D Arithmetic Logic Unit (ALU)

E Combinational

F Control signal

G CPI

H Datapath

I Decode

J Execute

K Fetch

L Finite State Machine (FSM)

M Instruction Set Architecture (ISA)

N NOR

O OR

P XOR

Question 2: Coding [10 pts]

Write the MIPS assembly for the `strfiddle` function below. It calls `test` and `replace`, which you know nothing about other than that they obey the MIPS calling convention:

```
int test (char c);
int replace (char c);

void strfiddle(char * str) {
  while ( *str != '\0' ) {
     int x = test (*str);
     if (x == 3) {
        *str = replace (*str);
     }
     str++;
  }
}
```

Question 3: Boolean Algebra + Gates [10 pts]

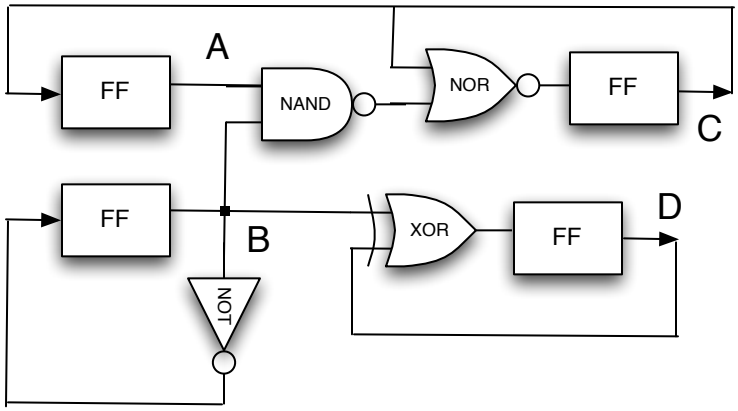
For the following truth table

- Write the formula for the truth table in sum-of-products form.
- Simplify the formula as much as possible.
- Draw the logic gates which correspond to your simplified formula.

A	B	C	Out
0	0	0	1
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	0

Question 4: Flipflops [10 pts]

Consider the following circuit (assume all flip-flops are positive edge triggered):



Show the values of the indicated signals at each time step. The clock is shown on the top line of the table, and the initial values are shown in the first column.

Clk	0	1	0	1	0	1	0	1
A	1							
B	0							
C	0							
D	1							

Question 5: Performance [10 pts]

After you have completed your special purpose application from Homework 4 you come up with another brilliant idea. Alex, Geoffrey, and Lindsay again are building custom hardware and compilers for your project, because they apparently have a lot of free time and like writing hardware and compilers. The following information is known.

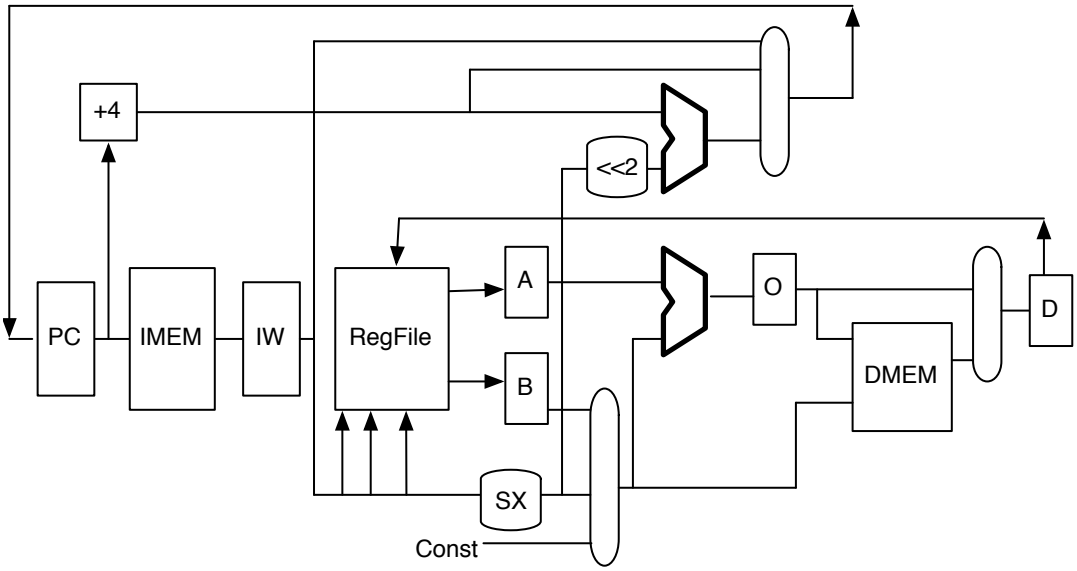
- Geoffrey has already gone ahead and built his design, and it has a known execution length for your new program of 600 seconds.
- Alex proposes a singe cycle data path, and plans to use a compiler that will generate 300 Billion dynamic instructions per execution.
- Lindsay proposes a pipelined data path, with a clock period of 1.2 ns. Her compiler will generate 400 Billion instructions. Of these 400 Billion instructions, 25% are loads, and 20% are branches. 40% of all loads cause a 2-cycle load-use stall. The penalty for a mis-predicted branch is 5 cycles. The processor has full bypassing, and does not suffer from any other stalls.

1. How fast does Alex need to make his clock cycle in order to make the program run faster than Geoffrey's?

2. How accurate does Lindsay's branch predictor need to be in order to make the program run faster than Geoffrey's?

Question 6: Datapaths [10 pts]

Consider the typical multi-cycle datapath we have seen in lecture (note that its modified slightly—the ALU input B select mux has an input to select a constant value sent in by the datapath controller):



The x86 ISA has an instruction, CALL target, which has the following effect:

```
MEM[r7] = PC + 4
r7 = r7 - 8
PC = target
```

1. Show the changes you need to make to the datapath design (add muxes, flip-flops, wires, etc) to support the CALL instruction. Note, you can assume that there are muxes on the register file read/write inputs that allow the datapath control logic to specify a constant register number from the control logic (not pictured).

2. Describe what happens cycle by cycle in your modified datapath to execute the CALL instruction.

Question 7: Pipeline diagrams [10 pts]

Assuming the following assembly instructions are executed in the order that they are found in the table below, fill out the chart indicating the stage of the standard 5-stage pipeline that the instruction will be in during the clock cycles. If an instruction is not in any stage during a cycle, simply leave that box blank. Indicate the stages of the pipeline using: F, D, X, M, and W. If there is gap in stages due to a data hazard, make sure to indicate the data hazard using d*.

First, show the pipeline execution if the pipeline has no bypassing:

Instructions	1	2	3	4	5	6	7	8	9	0	1	2
sub \$2, \$3, \$1												
lw \$5, 0(\$2)												
addi \$4, \$5, 1												
add \$5, \$3, \$1												

Now show what would happen if the pipeline had full bypassing:

Instructions	1	2	3	4	5	6	7	8	9	0	1	2
sub \$2, \$3, \$1												
lw \$5, 0(\$2)												
addi \$4, \$5, 1												
add \$5, \$3, \$1												

Finally, how much (expressed as a percentage) did the bypassing improve performance on this code fragment?

Question 8: Short-answer [10 pts]

1. Increasing the number of pipeline stages decreases the clock period. Give two reason why processors do not have hundreds or thousands of pipeline stages.

2. In a pipeline datapath, an individual instruction does not take any less time than in a multi-cycle datapath (some even take more because they go through all stages in a pipeline), however, a pipeline typically has higher performance. Explain why.