ECE550
Midterm

Name: NetID:

---

There are 6 questions, with the point values as shown below. You have 75 minutes with a total of 75 points. Pace yourself accordingly.

This exam must be individual work. You may not collaborate with your fellow students. However, you are permitted one page of notes.

**I certify that the work shown on this exam is my own work, and that I have neither given nor received improper assistance of any form in the completion of this work.**

Signature:

---

| # | Question | Points Earned | Points Possible |
|---|---|---|---|
| 1 | Combinatorial Logic | | 15 |
| 2 | Sequential Logic | | 10 |
| 3 | FSMs | | 10 |
| 4 | Asm Programming | | 20 |
| 5 | Datapaths | | 10 |
| 6 | Memory Hierarchy | | 10 |
| | Total | | 75 |
| | Percent | | 100 |

# Question 1 Combinatorial Logic [15 pts]

Given the following truth-table:

| a | b | c | x |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 1 |

1. Write the sum-of-product formula

2. Simplify the formula

3. Write VHDL that implements the formula

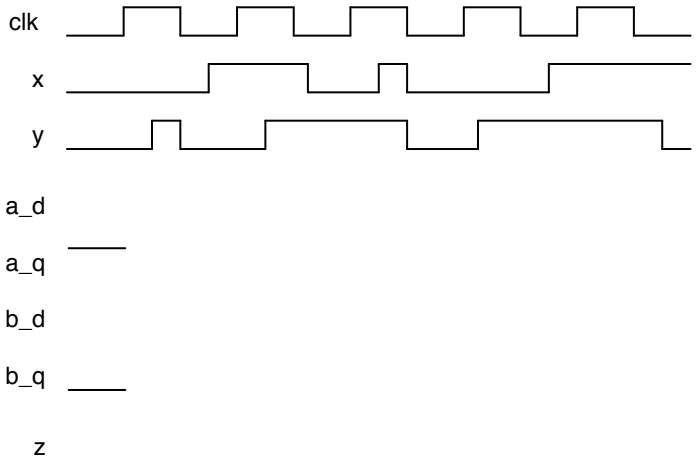
```
entity q1 is
  port (
    a : in  std_logic;
    b : in  std_logic;
    c : in  std_logic;
    x : out std_logic);
end q1;
architecture basic of q1 is
begin



end basic;
```

# Question 2 Sequential Logic [10 pts]

Consider the following VHDL fragment, in which `x`, and `y` are inputs, `z` is an output, and there are two DFFs (`a` and `b`) whose `d` inputs are `a_d` and `b_d` respectively, and whose `q` outputs are `a_q` and `b_q` respectively:

```
b_d <= a_q nand x;
a_d <= (b_q xor a_q) nor y;
z   <= a_q or (not b_q);
```

Complete the waveform below (assume that the DFFs are triggered by the rising edge of `clk`): Note that `a_q` is initially 1, and `b_q` is initially 0:



# Question 3 FSMs [10 pts]

Draw a state machine diagram for a finite state machine which accepts a single bit input (either 0 or 1—you can just label each edge with 0 or 1). This state machine also has a single bit of output, which is initally 0.

- Whenever the FSM receives an input of three *consecutive* 1s, the output goes to 1 (after the third one is read).
- The output remains at 1 until at least three 0s (even if *consecutive*) are received as input by the FSM, at which point the output returns to 0.
- The process then repeats (another three consecutive 1s return the FSM's output to a 1, etc).

Label each state with the bit it outputs. Be sure to indicate your start state (with an arrow to it from nowhere).

# Question 4 Asm Programming [20 pts]

Translate the following C function to MIPS assembly. **Answer on the next 2 pages** where you have each C-code line written out for you with space to write the MIPS assembly for that line directly under it.

```
void randomize(int * ptr, int n) {
  int i = 0 ;
  while (i < n) {
      int idx = randomNumTo(n);
      int temp = ptr[i];
      int temp2 = ptr[idx];
      ptr[i] = temp2;
      ptr[idx] = temp;
      i++;
  }
  return count;
}
```

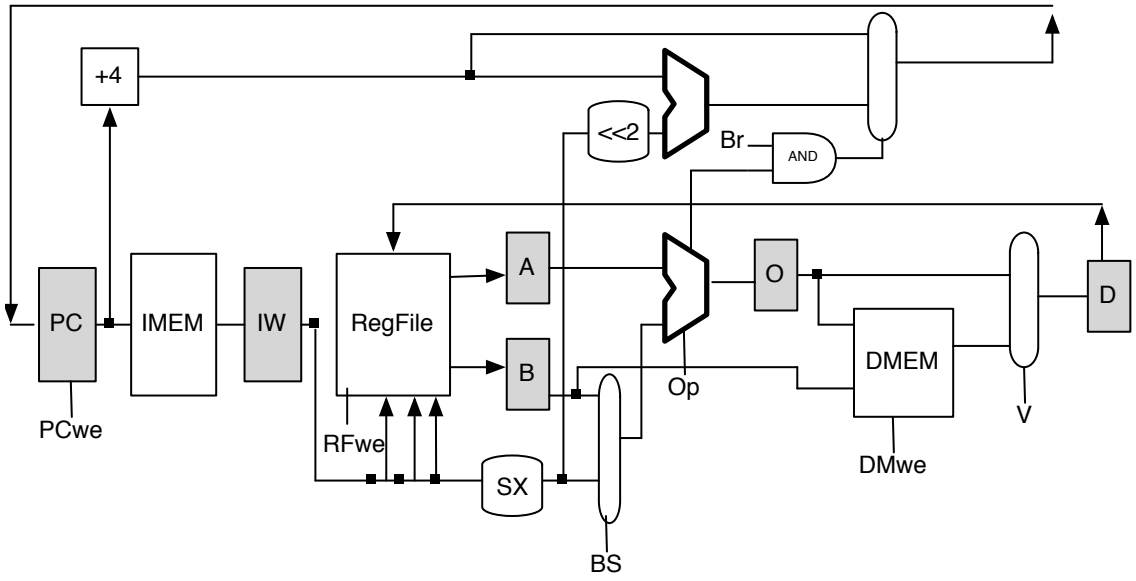
## Answer on next 2 pages

```
void randomize(int * ptr, int n) {









  int i = 0 ;



  while (i < n) {



      int idx = randomNumTo(n);



      int temp = ptr[i];



      int temp2 = ptr[idx];



      ptr[i] = temp2;
```

```
        ptr[idx] = temp;



        i++;



    }



    return count;



}
```

# Question 5 Datapaths [10 pts]

Consider the following multi-cycle data path:




1. Modify the above datapath to support the `lb $rt, imm($rs)` instruction. This instruction loads a *byte* from memory and sign-extends it. In modifying your datapath, you must ensure that you do not remove its capability to perform any instructions it already supports (*e.g.*, `lw`). Clearly label any control signal(s) that you introduce.

2. Fill in the table below to show the control signals required in each cycle to execute the `lb` instruction. If the control signal is "dont care" in a given cycle, write an X. If you added any control signals, put them in the empty columns at the right side of the table.

   - Mux selectors have 0 for the top input, 1 for the bottom.
   - Write enables (we) are 0 for disabled, 1 for enabled
   - For Op, you can write down the symbol for the mathematical operation you want (`+`, `-`, `*`, `<<`,etc).

| Cyc | PCwe | RFwe | BS | Op | DMwe | Br | V | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | | | | | | | | | | |
| 2 | | | | | | | | | | |
| 3 | | | | | | | | | | |
| 4 | | | | | | | | | | |
| 5 | | | | | | | | | | |

# Question 6 Memory Hierarchy [10 pts]

Suppose that you have a memory hierarchy with the following caches:

**L1 Data Cache** 1 cycle hit, 10% miss rate

**L2 Cache** 10 cycle hit, 10% miss rate

**Main Memory** 200 cycle latency

- What is the average access latency of the L1 Data Cache (in cycles)?

- Suppose that the processor's clock frequency were doubled, what would the new average access time of the L1 Data Cache be (in cycles)?

- Instead of doubling the clock frequency, suppose that an L3 cache were added with a hit latency of 50 cycles. What hit rate is required to make the average access time of the L1 data cache 3 cycles?