CS 240 Arch Assignment [75 points] **ID Number:**

About how many hours did you spend actively working on this assignment? _____

Q1 vALUe Judgement [22 points] Draw circuits on next page, text answers here.

1.1 Condition Flags [5 points]
(draw circuits on next page)

1.2. (a) [3 points] A, B with correct result

A	B	A - B	sign(A-B)	Is A < B?
positive	positive			no
negative	negative			yes
different signs	different signs			

1.2. (b) [2 points] A, B with incorrect result

A	B	A - B	sign(A-B)	Is A < B?
positive				no
negative				yes

1.2. (c) [1 point] Key effect

1.2. (d) [5 points] Draw your circuit for the `Less-Than` on the next page.

1.2. (e) [1 points] Control lines for `Less-Than`

Invert A = *Negate B =* *Operation =*

1.3. (a) [4 points] Draw your `Equals` Flag design on the next page.

1.3. (b) [1 points] Control lines for `Equals`

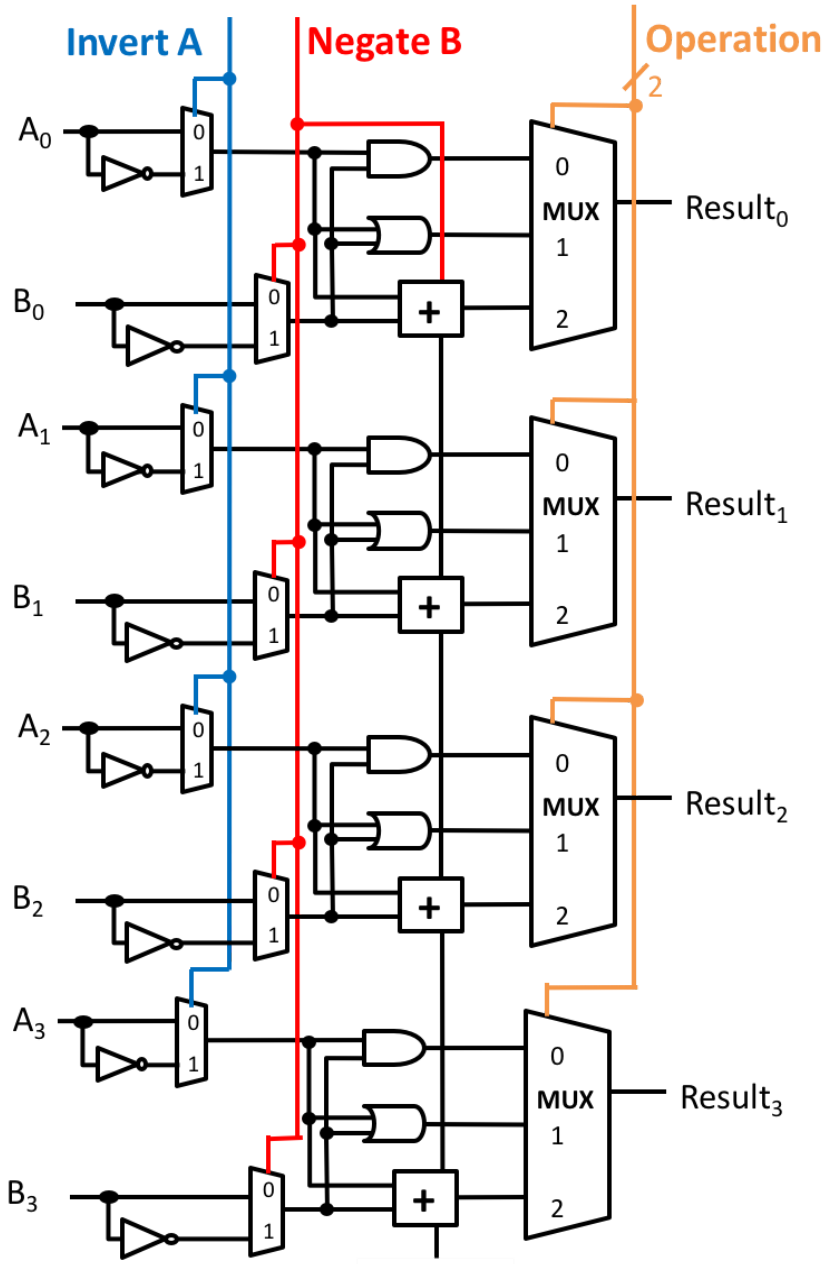
Invert A = *Negate B =* *Operation =*

Q2. Flop-flip-flopping [10 points]

Cycles Completed	Q_2	Q_1	Q_0
0 (initial)	0	0	0
1			
2			
3			
4			
5			
6			
7			
8			
9			
10			

Q2.2 Explanation (You need not fill this entire space.)

1.1. (a-d) Condition Flags, **1.2. (d)** `Less-Than` Flag, **1.3. (a)** `Equals` Flag. **Label outputs clearly.**



Q3 Some Loopy Programs [14 points]

3.1 [8 points] Execution Table for P1

PC	*Instruction*	*State Changes*

3.2 [3 points] Final contents, P1	**R2:**	**R3:**	**R4:**

3.3 [3 points] C statements equivalent to P1:

```
int R0 = 0;
int R1 = 1;
int R2 = R0+R1;
```

3.4 (a) [3 points] Result of P2

Execute this code, assuming **R2** holds 2 and **R3** holds 3. Indicate the final register values when the code reaches **HALT**.

```
0x0:  AND R2, R2, R4
0x2:  AND R3, R3, R5
0x4:  BEQ R5, R0, 3
0x6:  SUB R5, R1, R5
0x8:  ADD R4, R4, R4
0xA:  JMP 2
0xC:  HALT # Stops execution.
```

R2: R3: R4: R5:

3.4 (b) [2 points] C line for P2

Single line of C code equivalent to P2.
Use only basic C operations (no function calls).

```
R4 =
```

Q4 Taking Control [8 points]

Control Unit Truth Table

Instruction Name	$Opcode_{[3:0]}$ (4 bits)	Reg Write (1 bit)	$ALU\ Op_{[3:0]}$ (4 bits)	Mem Store (1 bit)	Mem (1 bit)	Branch (1 bit)	Jump **(Q6.2 [1pt])** (1 bit)
LW	0000	1	0010	0	1	0	0
SW							
ADD							
SUB							
AND							
OR							
BEQ							
NAND **(Q5.2 [3 pts])**							
JMP **(Q6.3 [1 pt])**							

Q5 Instruction Not Missing [8 points]

Fill in, following the format of slide 14 of the A Simple Processor lecture notes.

---------- 16-bit encoding ----------

Assembly	Meaning	Opcode [15:12]	Rs [11:8]	Rt [7:4]	Rd [3:0]
5.1 [3 points] `NAND Rs,Rt,Rd`	`R[d] ← ~(Rs & Rt)`				
5.3 [2 points] `NOT Rs,Rd`	`R[d] ← ~Rs`				

Q6 Jumping into the Unknown [8 points]

6.1 [6 points]. Below, add a `Jump` output wire from the Control Unit and modify logic to use it to implement JMP instruction. Note: if you use the new red write split off from `Inst`, be sure to label which range ([?, ?]) of bits you use.