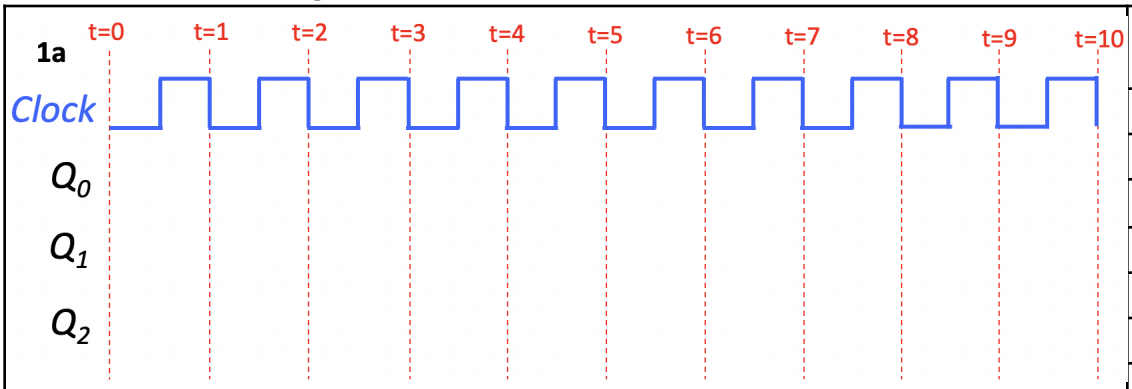
CS 240 Arch Assignment [72 points] **ID Number:**

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

Q1. Flop-Flip-Flopping [10 points] *Time spent on Q1:* _____

1a



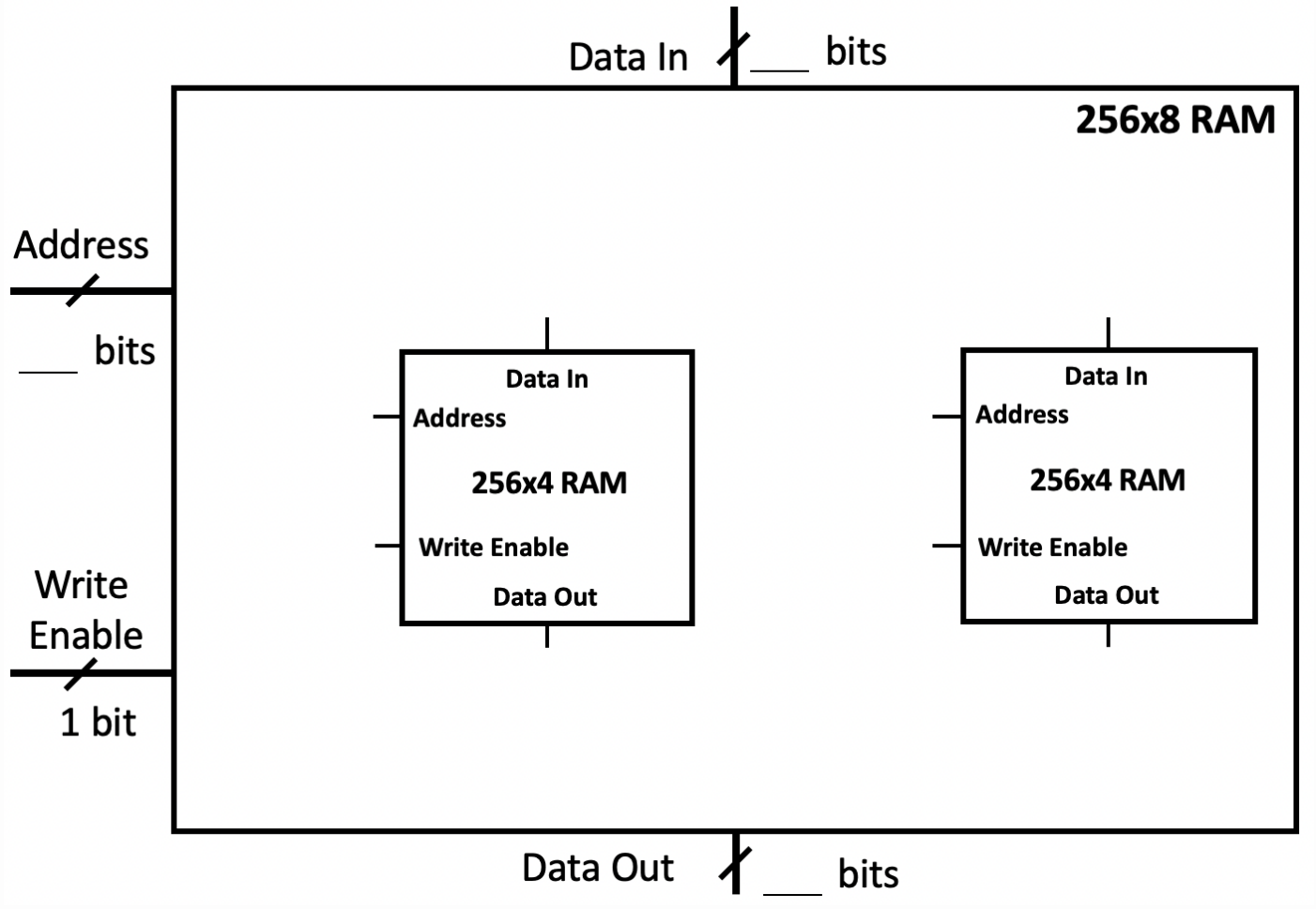
1c. Explanation

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

Q2 [15 points] Reconstructing Memories *Time spent on Q2:* _____

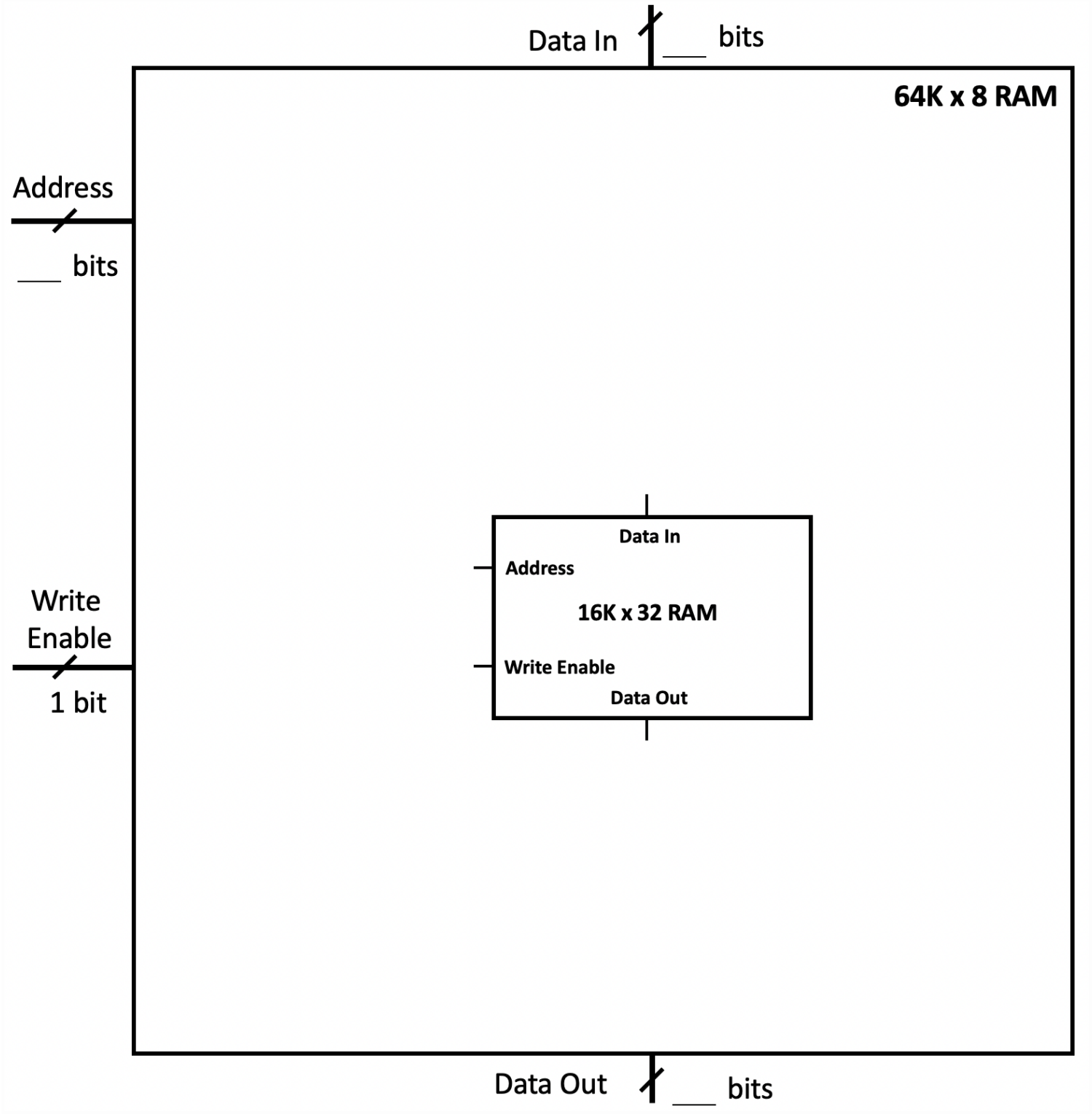
2a [5 points] *Time spent on Q2a:* _____

Draw a 256×8 RAM that's implemented by two 256×4 RAMs. Your logic will go inside the box.



2b [10 points] *Time spent on Q2b:* _____

Draw a 64K×8 RAM that's implemented by one 16K×32 RAM.



Q3. A Loopy Program [14 points] *Time spent on Q3:* _____

3a [10 points] Execution Table for P1 (should have 18 rows)

PC	*Instruction*	*State Changes*

3b [1.5] Final Register Contents	**R2:**	**R3:**	**R4:**

3c [2.5] Python, Java, or Javascript statements equivalent to P1:

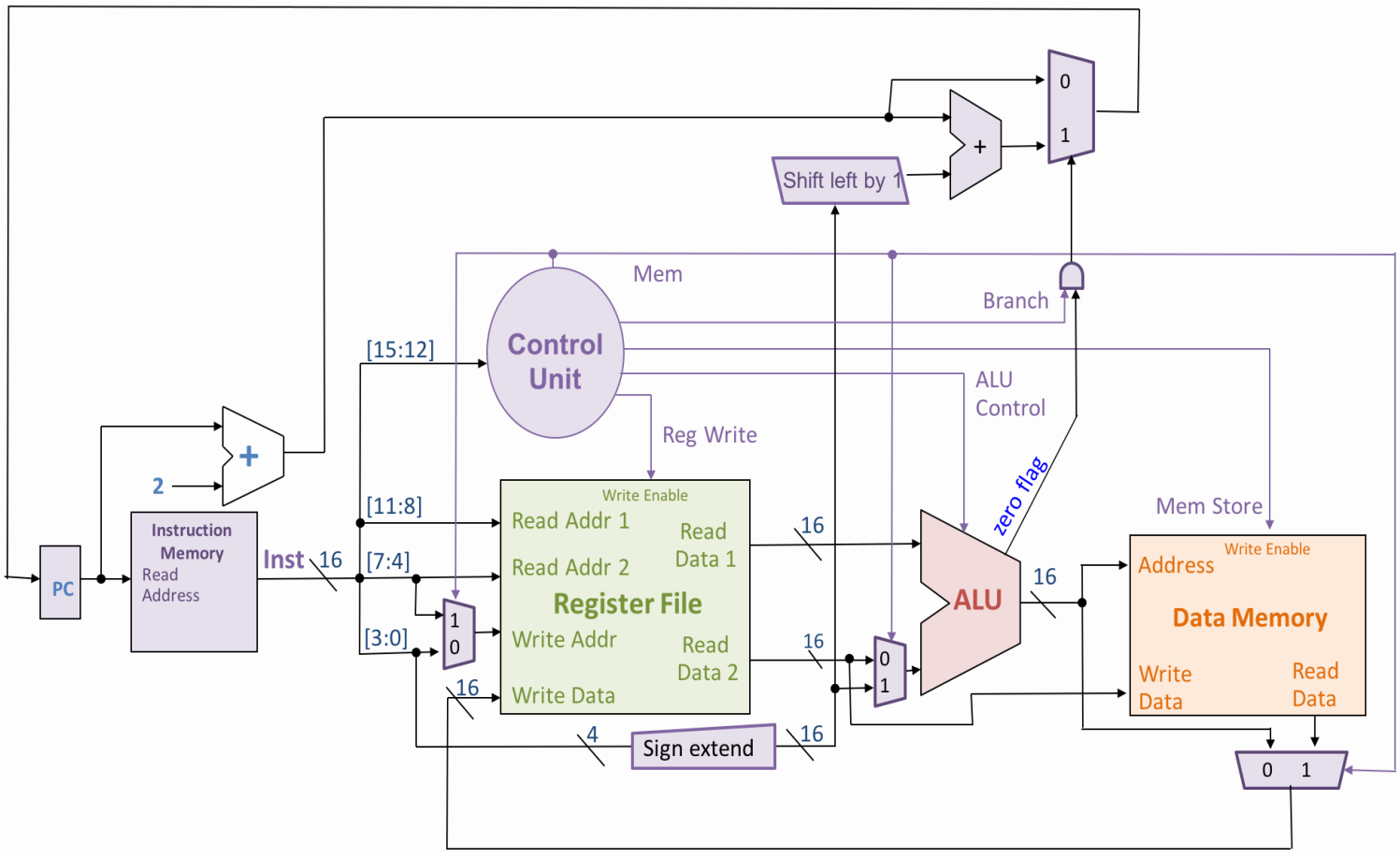
Q4. Taking Control [8 points] *Time spent on Q4:* _____

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 **(5a(ii) [1])** (1 bit)
LW							
SW							
ADD							
SUB							
AND							
OR							
BEQ							
JMP **(5a(iii) [2])**							
NAND **(6b(ii)[1])**							

Q5. Jumping into the Unknown [15 points] *Time spent on Q5:* _____

5a(i) [10]. Below, add Jump output from Control Unit and modify logic to use it to implement JMP instruction.



5b(i) [3] Execute this code, assuming **R2** holds 5 and **R3** holds 3. Indicate the final register values when the code reaches **HALT**.

```
0:  AND R2, R2, R4
2:  AND R3, R3, R5
4:  BEQ R5, R0, 3
6:  SUB R5, R1, R5
8:  ADD R4, R4, R4
A:  JMP 2
C:  HALT # Stops execution.
```

R2: R3: R4: R5:

5b(ii) [2] Single line of C code equivalent to this code.

R4 =

Q6. Instruction Not Missing [10 points]

Time on Q6: _____

6a [4] The instruction **NOT Rs,Rd** can be emulated by running the following instructions instead. (Also briefly justify why these instructions work.)

6b-c. NAND/NOT encoding and definition ---------- 16-bit encoding ----------

Assembly	Meaning	Opcode [15:12]	Rs [11:8]	Rt [7:4]	Rd [3:0]
6b(i) [3] **NAND Rs,Rt,Rd**	***R[d] ← ~(Rs & Rt)***				
6c [2] **NOT Rs,Rd**	***R[d] ← ~Rs***				

7. Points Affixed and Afloat in a C of Numbers (OPTIONAL PROBLEM!)

7a. Fixed point numbers **Sea Type**	Minimum (base ten)	Maximum (base ten)	**iii.** Adder (It fits! Reuse provided parts.)
I. `signed fixed8ths char`			
Ii. `signed fixed32nds char`			

7b. Floating point conversion.

6-bit floating-point encoding	**110101**	**100001**	**011100**	**000011**	**010010**	**111101**
Decimal number represented						