CS 240 Arch **ID Number:**

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

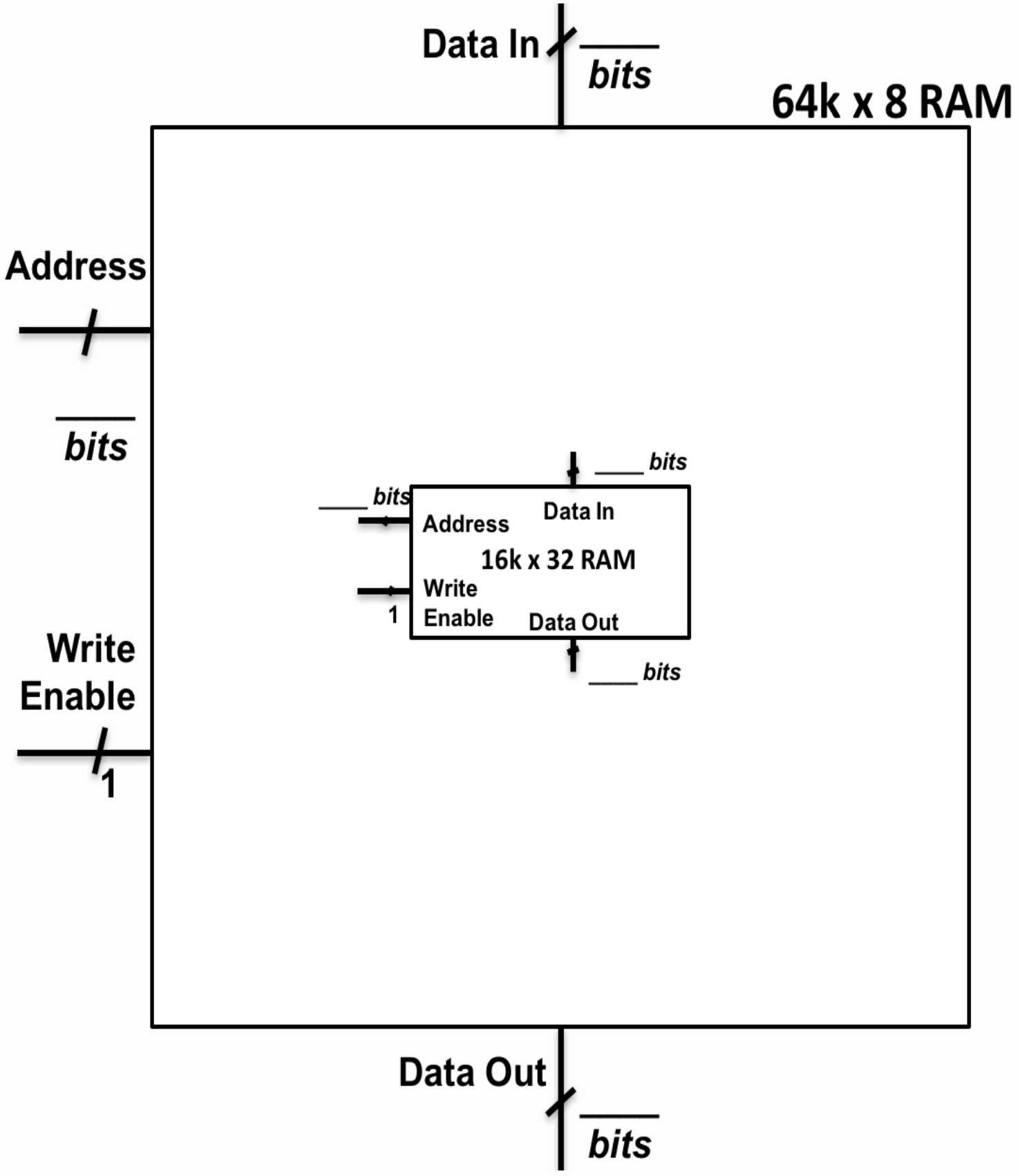
1. Flop-Flip-Flopping

1a. Cycles Completed	Q_2	Q_1	Q_0	**1b. Explanation** (You need not fill this entire space.)
0 (initial)	0	0	0	
1				
2				
3				
4				
5				
6				
7				
8				
9				
10				

2. Reconstructing Memories

2a. Draw a 256×8 RAM based on two 256×4 RAMs. Your logic will go inside the box.

2b. Draw a 64K×8 RAM based on one 16K×32 RAM.



3. A Loopy Program

3a. Execution Table for P1 (should have 18 rows)

PC	*Instruction*	*State Changes*

3b. Final Register Contents	**R2:**	**R3:**	**R4:**

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

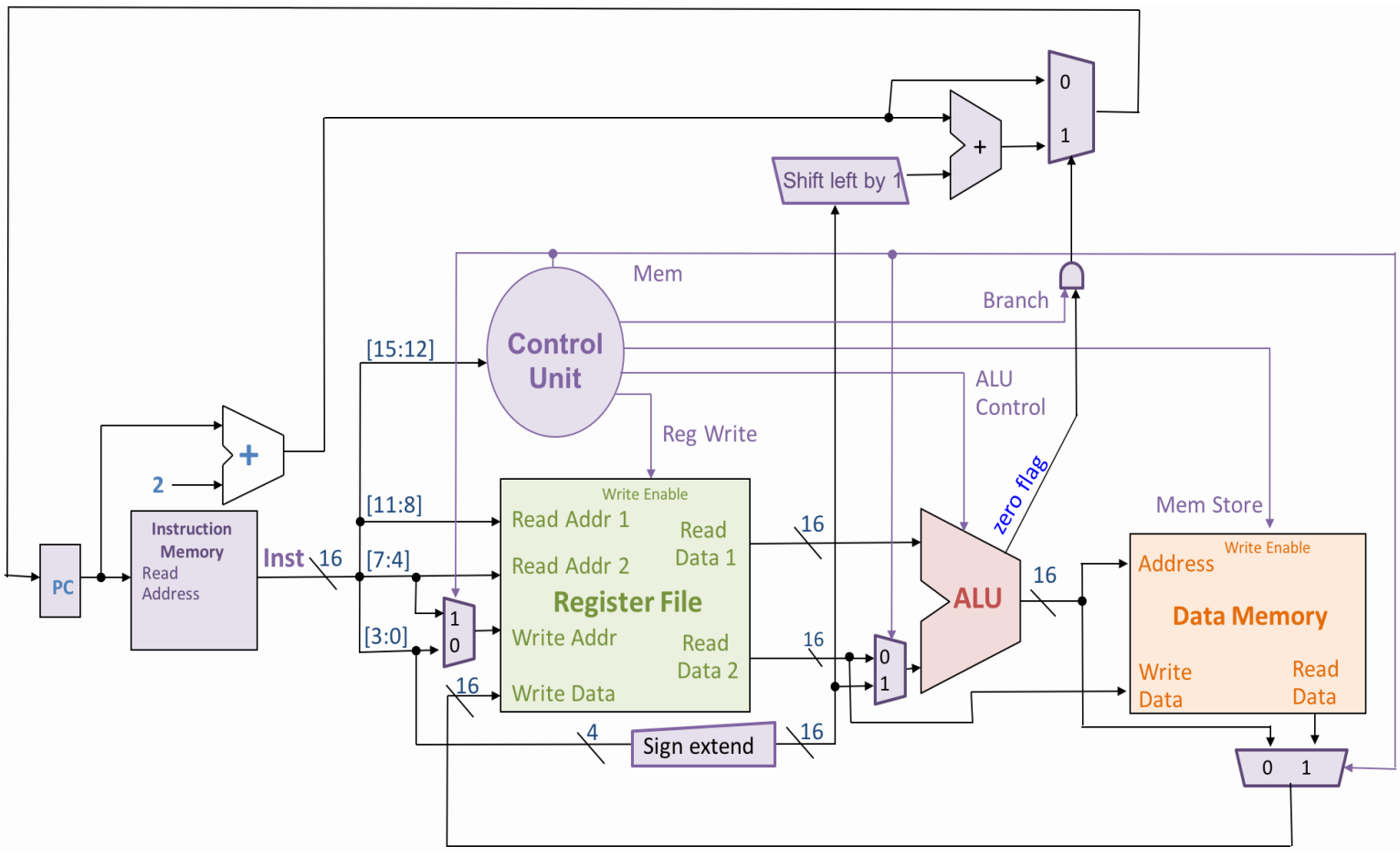
4. Taking Control

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

5. Jumping into the Unknown

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



5b(i). Execute this code, assuming **R2** holds 5 and **R3** holds 2. 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). Single line of C code equivalent to this code.

R4 = _____ **;**

6. Instruction Not Missing

6a. The instruction **NOT Rs,Rd** can be emulated by running the following instructions instead:

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). **NAND Rs,Rt,Rd**	***R[d] ← ~(Rs & Rt)***				
6c. **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						