CS 240 Pensieve Processing   **Name:**
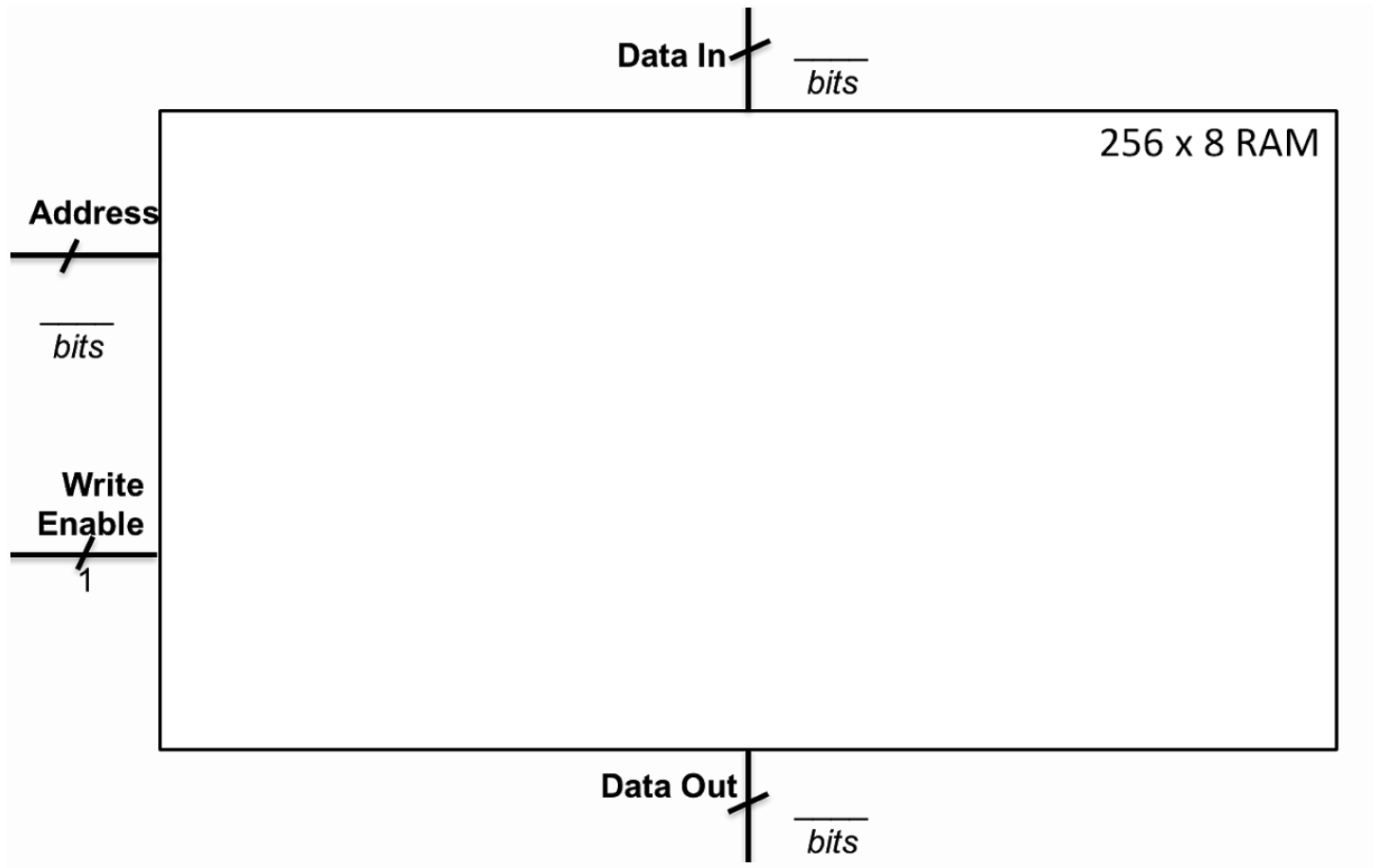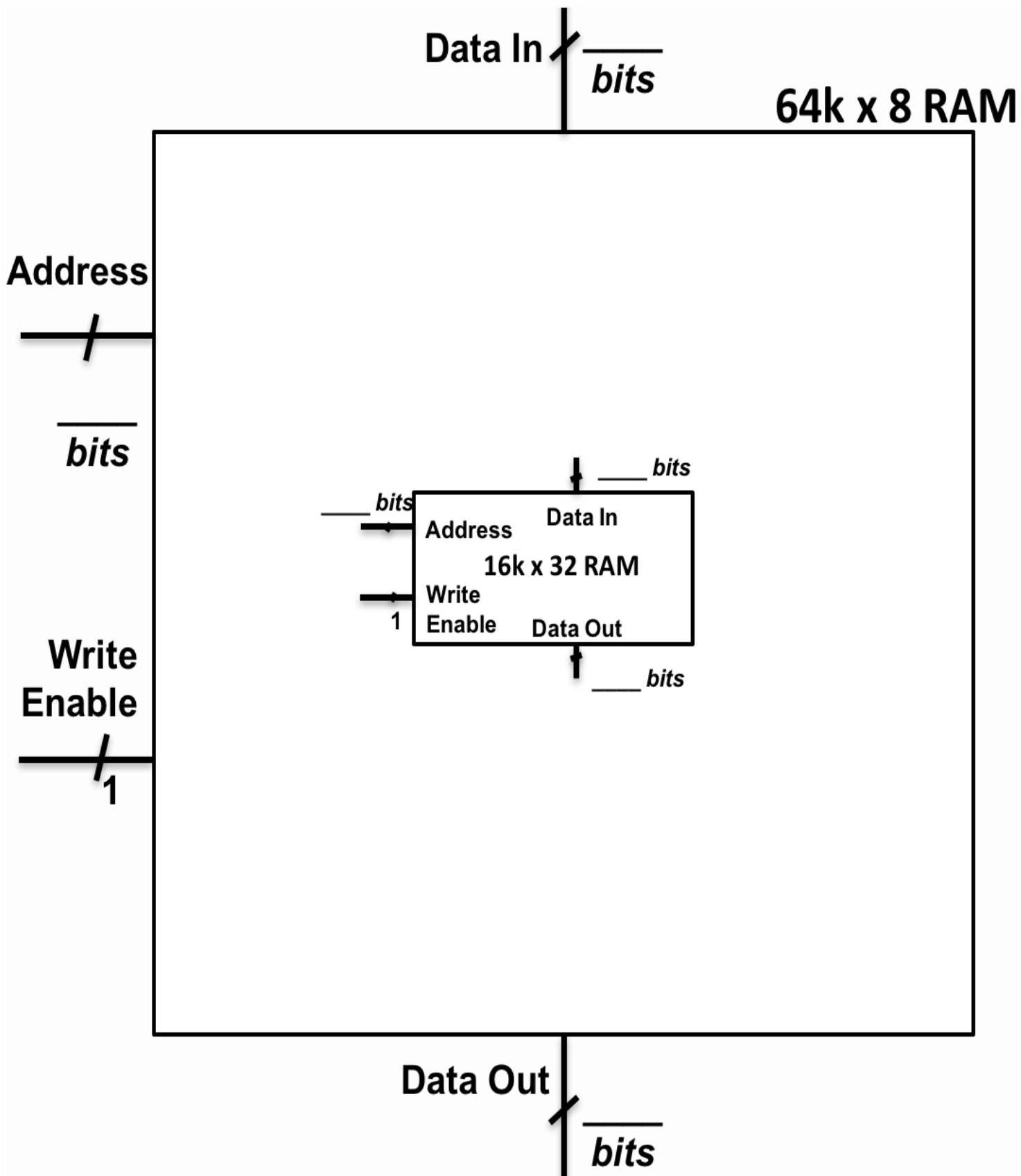
About how many hours did you spend actively working on this assignment? _____

1. Reconstructing Memories

1a. Draw a 256×8 RAM based on two 256×4 RAMs. Your logic will go inside the box.

Data In — _bits_

256 x 8 RAM

Address — _bits_

Write Enable — 1

Data Out — _bits_

1b. Draw a 64K×8 RAM based on one 16K×32 RAM.

Data In / ‾‾‾ *bits*

64k x 8 RAM

Address

/

‾‾‾ *bits*

___ *bits*

___ *bits*

Address   **Data In**

16k x 32 RAM

Write
Enable   **Data Out**

1

___ *bits*

Write
Enable

/
1

Data Out / ‾‾‾ *bits*
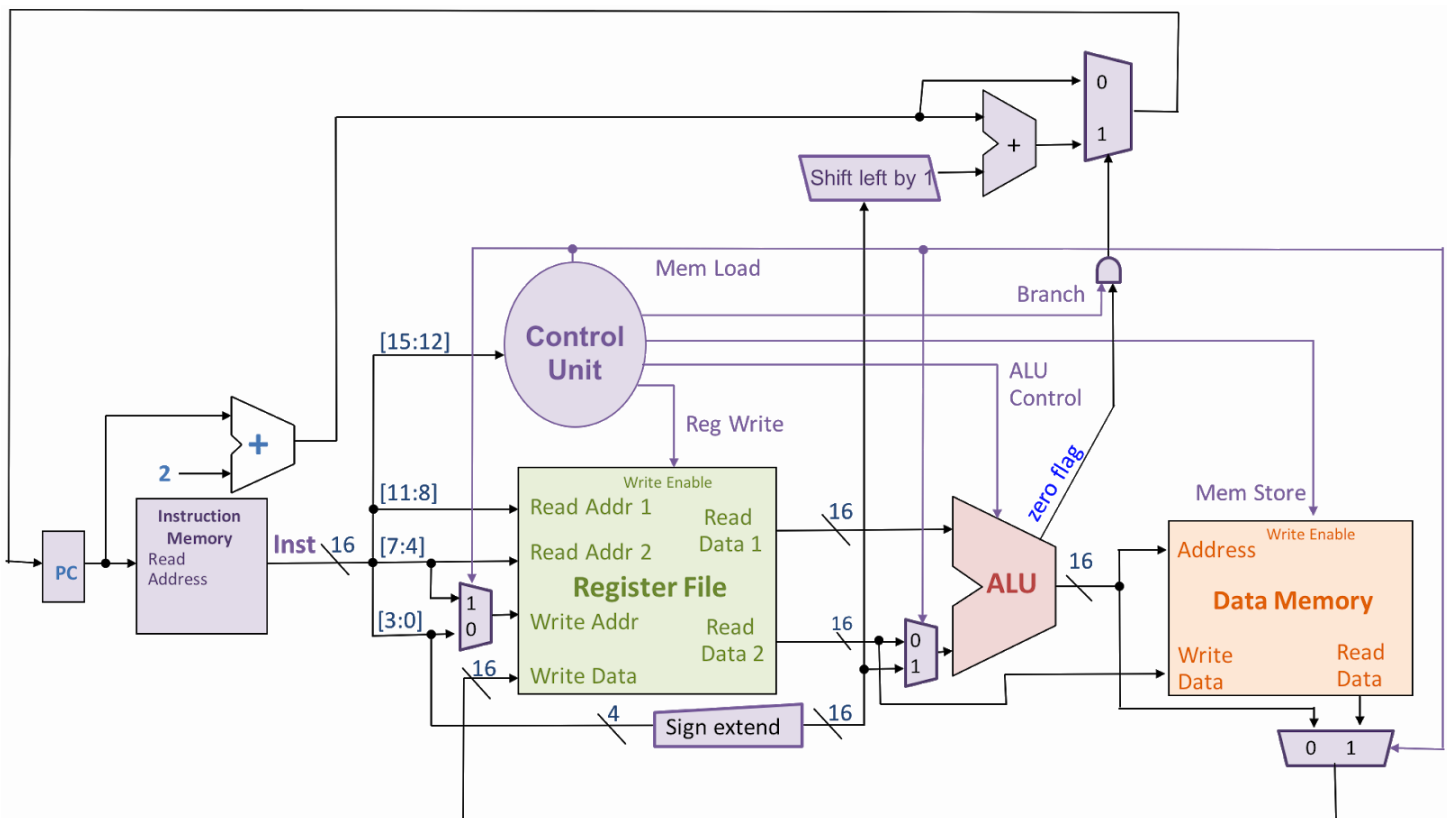
2. Taking Control

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 Load (1 bit)	Branch (1 bit)	Jump **(4a)** (1 bit)
LW							
SW							
ADD							
SUB							
AND							
OR							
BEQ							
JMP **(4a)**							
NAND **(5b)**							

3. Jumping into the Unknown

3a. Draw JMP logic and fill the JMP row in the control unit truth table above.

3b 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: _____

3b ii. Single line of C code equivalent to this code.

R4 = _____ ;

_____

4. Instruction Not Missing
4a. The instruction `NOT Rs,Rd` can be emulated by running the following instructions instead:

4b-c. NAND/NOT encoding and definition

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

Assembly	Meaning	Opcode [15:12]	Rs [11:8]	Rt [7:4]	Rd [3:0]
(5b) `NAND Rs,Rt,Rd`	`R[d] ← ~(Rs & Rt)`				
(5c) `NOT   Rs,Rd`	`R[d] ← ~Rs`				

5. Points Affixed and Afloat in a C of Numbers

5a. 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`			

5b. Floating point conversion.

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