CS 240 Arch Assignment [60 points] **ID Number:**

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

1 A Loopy Program [14 points]

1a [8 points] Execution Table for P1

PC	*Instruction*	*State Changes*

1b [3 points] Final contents	**R2:**	**R3:**	**R4:**

1c [3 points] C statements equivalent to P1:

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

2 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 **(5a(ii))** (1 bit)
LW	`0000`						
SW	`0001`						
ADD	`0010`						
SUB	`0011`						
AND	`0100`						
OR	`0101`						
BEQ	`0111`						
NAND **(3b(ii)) [1 pointt]**							
JMP **(4a(iii)) [1 point]**	`1000`						

3 Instruction Not Missing [10 points]

3a [4 points] The instruction **`NOT Rs,Rd`** can be emulated by running the following instructions instead. Briefly justify why these instructions work.

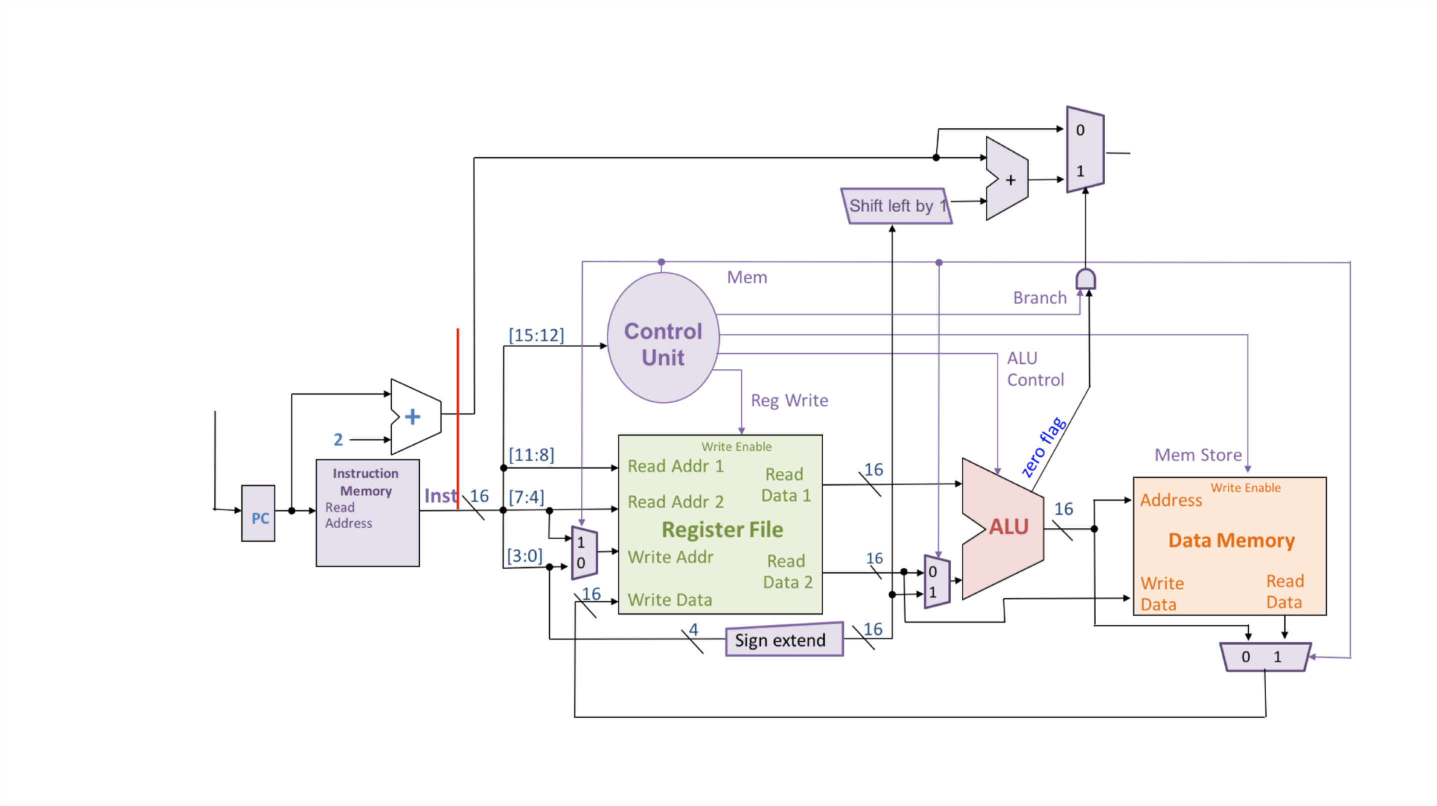
3b-c NAND/NOT encoding and definition ------------------- 16-bit encoding ------------------

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

4 Jumping into the Unknown [15 points]

4a(i) [10 points]. Below, add a `Jump` output wire from the Control Unit and modify logic to use it to implement JMP instruction. You may need new *Zero Extend* and *Shift left by 1* units.

Note: if you use the new red write split off from `Inst`, be sure to label which range ([?, ?]) of bits you use.



4b(i) [3 points] 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:

4b(ii) [2 points]

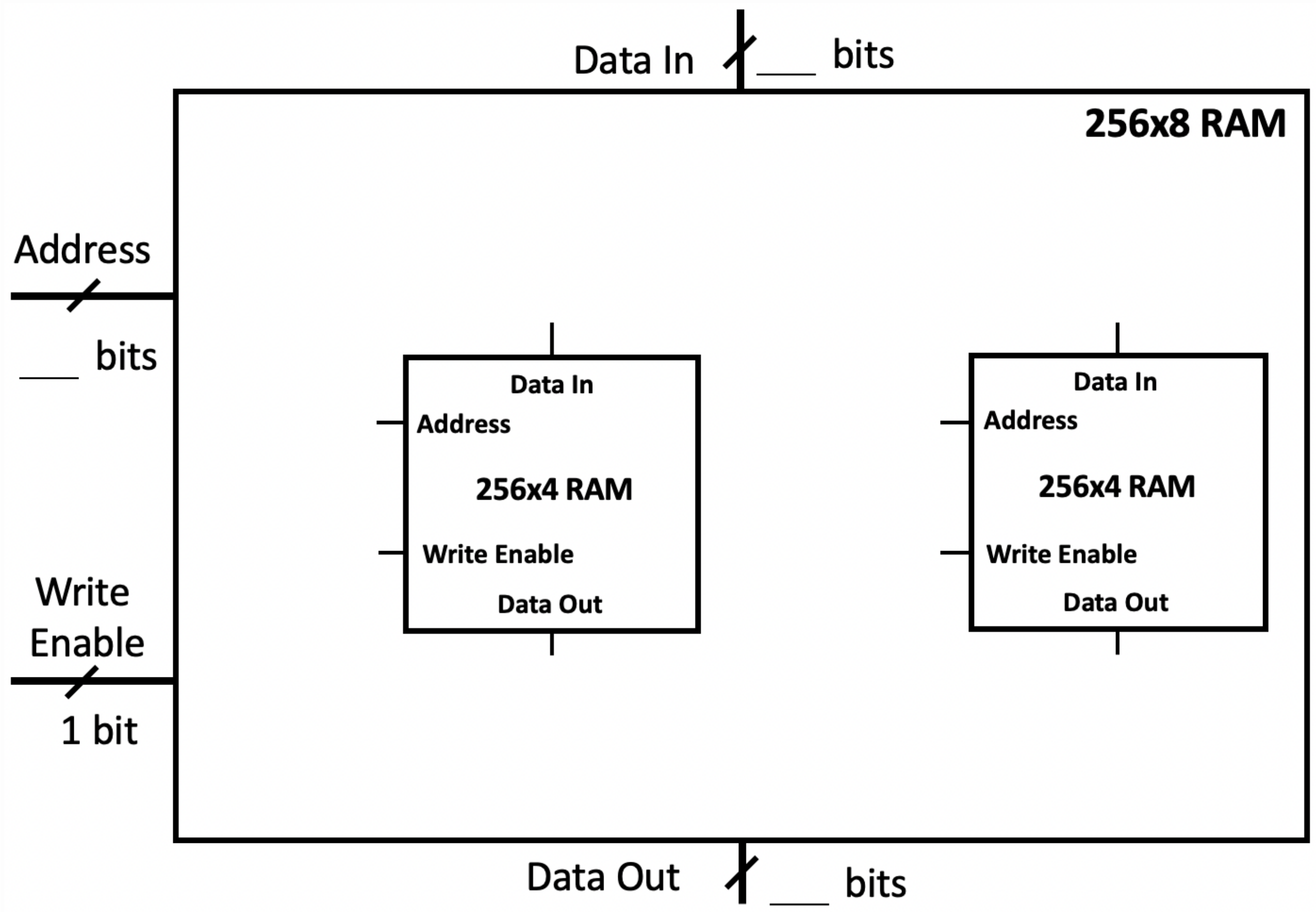
Single line of C code equivalent to the HW ISA code.

R4 =

4a(iii) Add to the control table above.

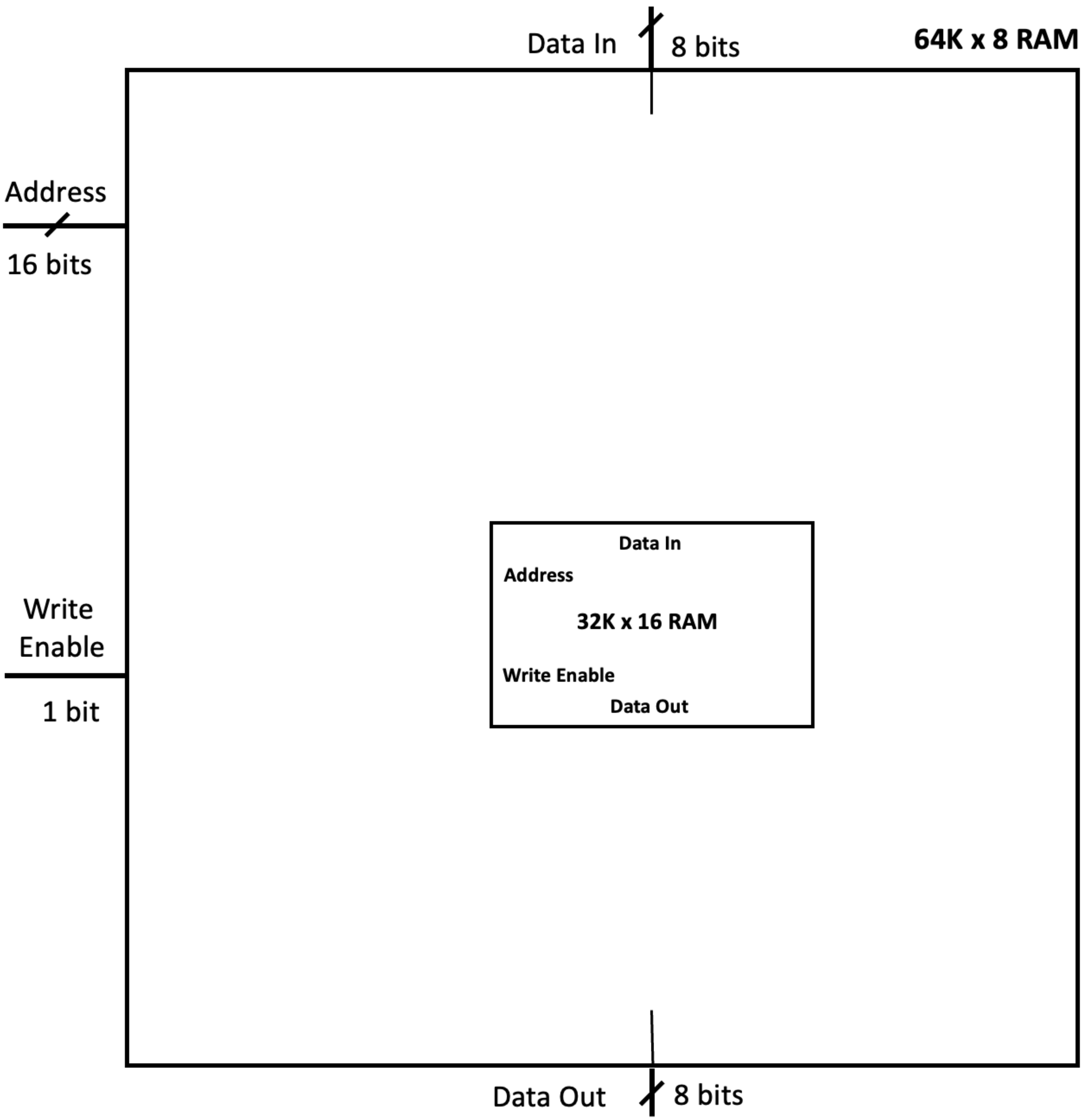
5 [13 points] Reconstructing Memories

5a [5 points] Draw a 256×8 RAM that's implemented by two 256×4 RAMs. Your logic will go inside the box.



5b [8 points]

Draw a 64K×8 RAM that's implemented by one 32K×16 RAM. Your logic will go inside the box.



6 [OPTIONAL PROBLEM] Points Affixed and Afloat in a C of Numbers

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`			

Floating point conversion.

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