Department of Electrical and Computer Engineering
The University of Texas at Austin

EE 460N Fall 2016
Y. N. Patt, Instructor
Siavash Zangeneh, Ali Fakhrzadehgan, Steven Flolid, Matthew Normyle TAs
Final Exam
December 9, 2016

Name: Solution

Problem 1 (10 points):_____

Problem 2 (15 points):_____

Problem 3 (10 points):_____

Problem 4 (15 points):_____

Problem 5 (25 points):_____

Problem 6 (25 points):_____

Problem 7 (30 points):_____

Total (130 points):_____

Note: Please be sure that your answers to all questions (and all supporting work that is required) are contained in the space provided.

Note: Please be sure your name is recorded on each sheet of the exam.

Please sign the following. I have not given nor received any unauthorized help on this exam.

Signature:_____

GOOD LUCK!

Name:_____

Problem 1 (10 points)

Part a (5 points): The LC-3b is expanded to include an IEEE floating point co-processor which is provided with the following code to execute. Assume the co-processor can execute the DIVIDE instruction.

```
        .ORIG x3000
        LEA R3, A
        AND R0, R0, #0
        AND R1, R1, #0
        DIVIDE R2, R0, R1
        STW R2, R3, #0
        HALT
A       .BLKW 1
        .END
```

What happens? Please be specific.

The DIVIDE instruction tries to execute $\frac{0}{0}$
This is a NaN, takes an "invalid" exception

Part b (5 points): Several companies are looking into adding a 10-bit floating point number to their ISAs. One possibility would specify five bits of exponent and four bits of fraction, and a BIAS for computing exponents of 15.

We can represent the number 2 exactly. We can also represent a lot of numbers on the real line as "2" inexactly. That is all numbers greater than X and less than Y are represented by the value 2. Your job: Identify X and Y. Assume rounding mode is unbiased nearest.

X: $1\frac{31}{32}$ Y: $2\frac{1}{16}$

Note wobble!

$X = 2 - \frac{1}{32}$ $Y = 2 + \frac{1}{16}$

Note unbiased nearest rounding mode

Name:_____

Problem 2 (15 points)

Part a (5 points): What does the function Task do? Input parameters are provided in R0, R1, and R2.

```
Task  ADD R3, R1, #0
Loop  MUL R3, R3, R1
      STW R3, R2, #0
      ADD R2, R2, #2
      ADD R0, R0, #-1
      BRp Loop
      RET
```

Produces a power series of $(R1)^2, (R1)^3, (R1)^4, \ldots, (R1)^{n+1}$ where n = R0, starting at memory location in R2

Part b (5 points): Assume the function Task from part a is processed on a one-wide issue machine that executes instructions out-of-order and retires them in-order. Only one instruction can retire each cycle. The machine has five adders and two multipliers.

ADD instructions take two cycles of execution (E) and one cycle to retire (R). MUL instructions take three cycles of execution and one cycle to retire. Memory access instructions take four cycles plus one cycle to retire. Branch instructions take one cycle to execute plus one cycle to retire. Data forwarding is allowed.

The instruction ADD ~~R2~~R3,R1,#0 is fetched (F) in cycle 1, decoded (D) in cycle 2, executed in cycles 3,4, and retired in cycle 5, as shown on the diagram.

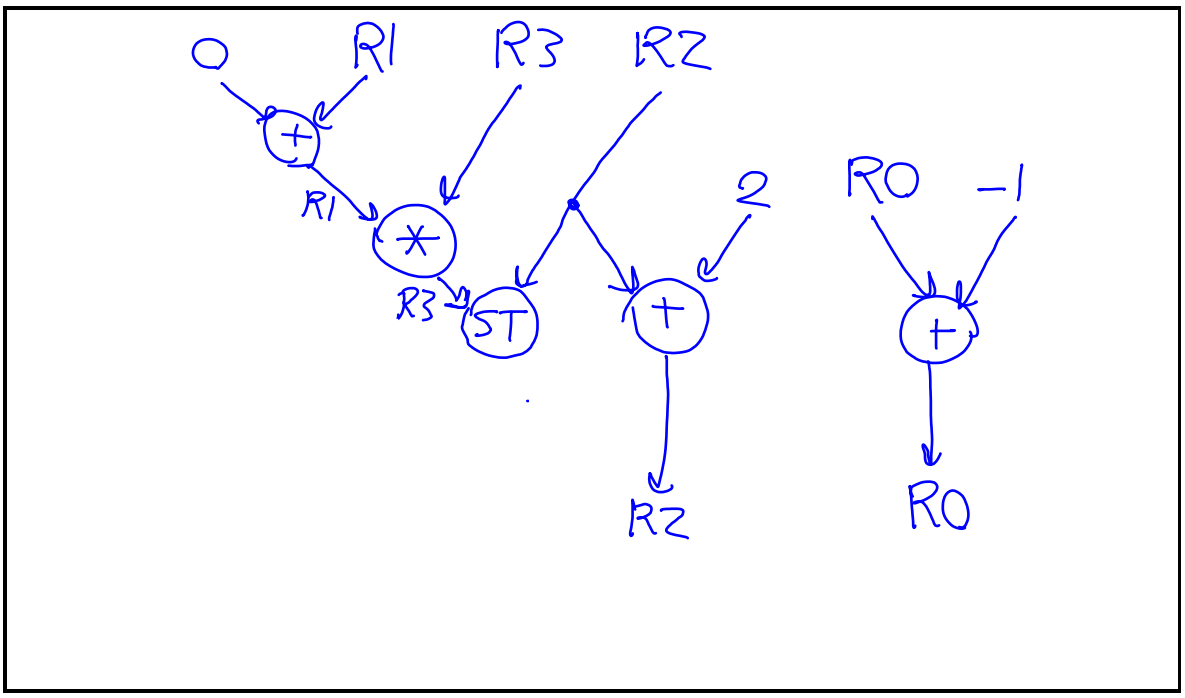
Your job: fill in the appropriate symbols F,D,E,R for each cycle up to the point where BRp Loop retires.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
(A) ADD R3, R1, #0	F	D	E	E	R													
(B) Loop MUL R3, R3, R1		F	D		E	E	E	R										
(C) STW R3, R2, #0			F	D				E	E	E	E	R						
(D) ADD R2, R2, #2				F	D	E	E						R					
(E) ADD R0, R0, #-1					F	D	E	E						R				
BRp Loop						F	D		E						R			

Name:_____

Part c (5 points): Five instructions in part b have been identifed with a label from A to E. For example, ADD R2,R1,#0 is labeled A.

Your job: Construct the data flow graph for code represented by the five instructions labeled A,B,C,D,E.

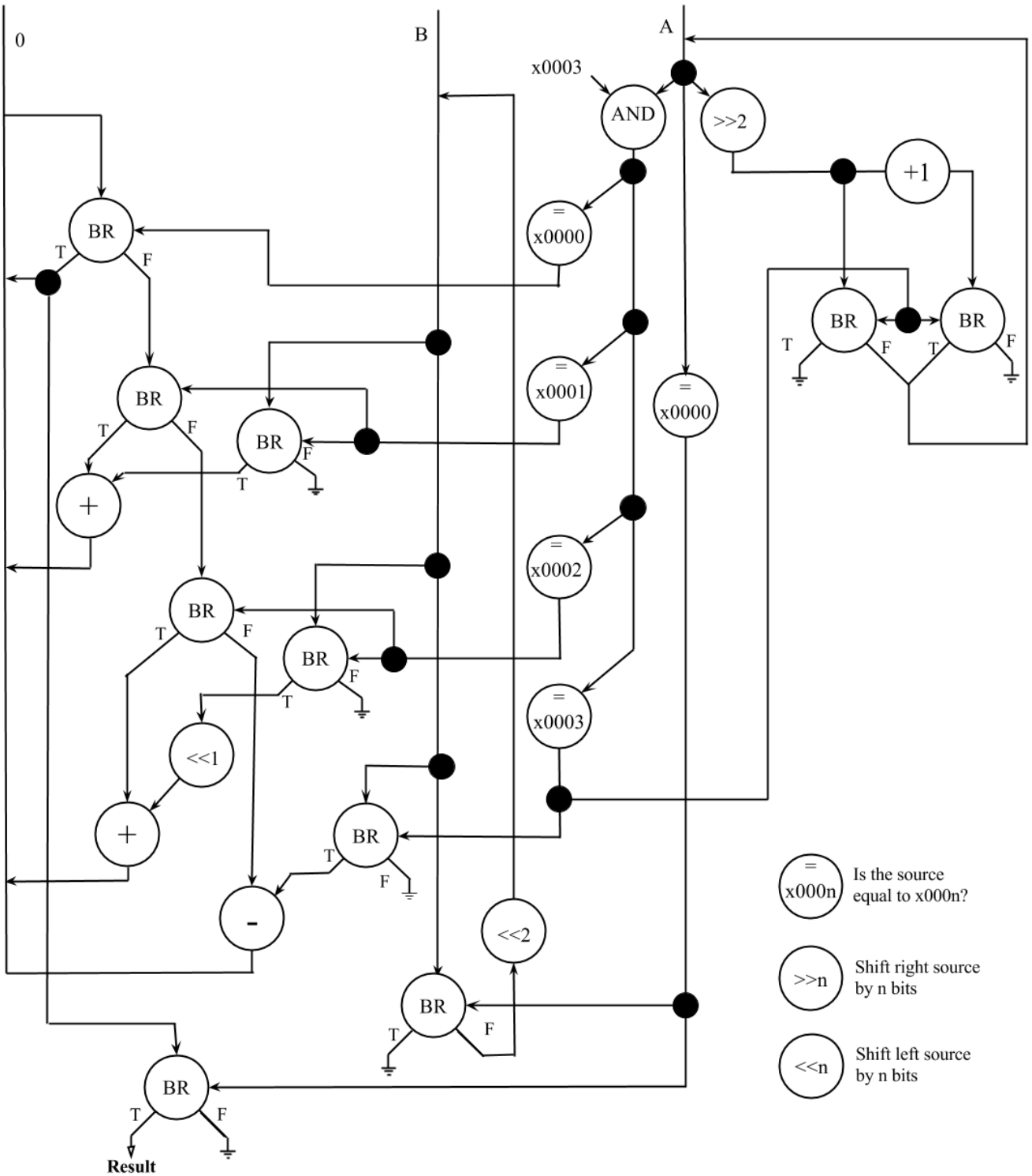


Name:_____

Problem 3 (10 points): What result is produced by execution of the procedure implemented by the data flow graph shown below. A and B are 16bit, positive 2's-complement integers. What is the common name for this procedure? Copy nodes are represented by filled-in 1/4 inch circles. Note: Explanations for the relational data flow node = and the operate node shift are provided in the lower right hand corner of this page.

Result: A * B

Procedure: Booth's algorithm



Name:_____

Problem 4 (15 points)
A vector processor is asked to multiply two matrices A, B, producing the resulting matrix C. A is a 5 by 10 matrix. B is a 10 by 4 matrix. As you recall from linear algebra, each element c_{ij} is computed as the inner product (also called dot product) of row i of A and column j of B.

The code below will produce in vector register V2 each of the component products a_{i0}*b_{0j}, a_{i1}*b_{1j}, ... a_{i9}*b_{9j}. Note: The MUL instruction follows Cray's convention: MUL src, src, dest.

We have not included the final step of adding the components of V2 to produce the final result c_{ij}.

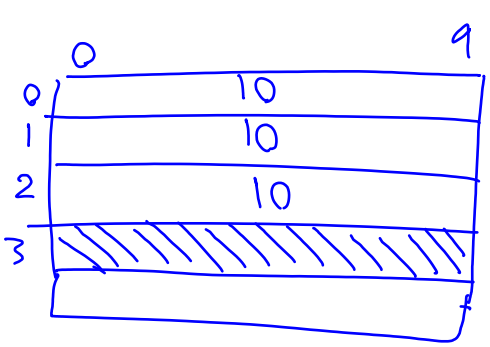
Assume matrices A and B are stored in row major order; a_{00} starting at location x2000, b_{00} starting at location x3000. Memory is word addressable.

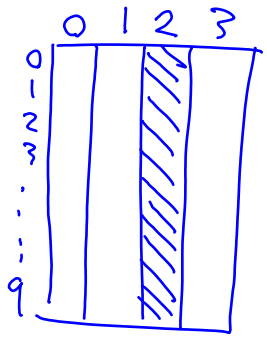
```
MOVI [ 10 ], VLN
MOVI [ 1  ], VST
VLD V0, X ;V0 contains row i of A
MOVI [ 4  ], VST
VLD V1, Y ;V1 contains column j of B
MUL V0, V1, V2
```

Your job: To compute $\mathbf{c_{32}}$, fill in the three boxes and compute X and Y.

X: x201E

Y: x3002





X = x2000 + 30 = x201E

Y = x3000 + 2 = x3002

Name:_____

Problem 5 (25 points)

① Program accesses (256)(4) integers in consecutive memory location

$256 \times 4 \times 4 = 4096$ KB

② It results in 8 row buffer misses

Each row buffer = 4KB/8 = 512B

The following piece of code

```
int M[4][n];

int main() {
  int sum = 0;
  for(int i = 0; i < 4; ++i) {
    for(int j = 0; j < 256; ++j ) {
      sum += M[i][j]
    }
  }
}
```

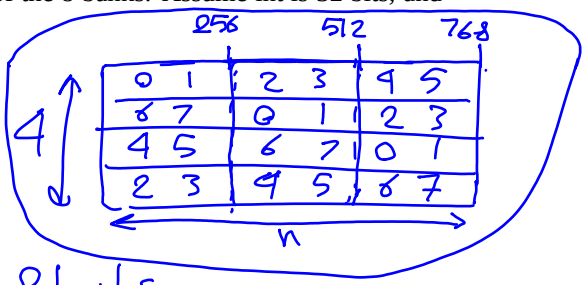
is executed on a processor connected via a single channel 32bit bus to a 64KB byte-addressable memory addressed as follows:

a row buffer

Row	Bank	Column	Byte on Bus

Memory is organized as a single rank, with 8 banks.

Example for part b



Suppose n=256. If the 4 by n (4 by 256) matrix M is stored in row major order and the row buffers are initially empty, execution accesses all 8 banks, and results in one row buffer miss in each of the 8 banks. Assume int is 32 bits, and variables *sum*, *i*, and *j* are allocated in registers.

Part a: What is the row buffer size? 512 B

Row bits: 4 [15:12] remaining bits

Bank bits: 3 [11:9] Because there are 8 banks

Column bits: 7 [8:2] ① row buffer = 512 B ② 4 chips in lockstep } row buffer per chip = 128 B

BoB bits: 2 [1:0] Because bus is 32 bits

Part b: Assuming the 4 by n matrix M is stored in row major order and assuming the row buffers are initially empty, For what values of n greater than 256 will execution access all 8 banks and result in one row buffer miss in each of the 8 banks. Explain.

The loop only accesses the first 256 elements of each row, so if the size of the row is $256 + 512n$, the first 256 integers of each row will fall on 8 different banks.

Look at the example provided (n=768), numbers written in the matrix are bank numbers.

Name:_____

Problem 6 (25 points)

A multiprocessor system with three processors, each with its own cache, maintains cache coherence using the Goodman "write once snoopy cache protocol" we studied in class. Recall, if processor A takes a cache miss and processor B has the line in the modified (dirty) state, processor B supplies the line. Memory is byte-addressable.

The table below shows ~~8~~ 7 successive memory accesses made by the processors.

Access Number	Memory Location	Processor Number	Read/Write	Hit/Miss
1	0101110101000	3	R	Miss
2	0101110100000	1	R	Miss
✓ 3	0101110110000	2	W	Miss
4	0101110100001	1	R	Hit
✗ 5	0101110101000	3	W	Hit
✗ 6	0101110101000	3	W	Hit
7	0101110100001	1	R	Miss

Part a (18 points): What is the line size of the cache? Explain.

16B. Explanation below

Part b (7 points): Which of the three stores are write throughs to memory? Explain.

access ③ → write-through : first write to cache line in processor 2
access ⑤ → write-through : first write to cache line in processor 3
access ⑥ → write-back : second write to cache line in processor 3

Explanation of part a:

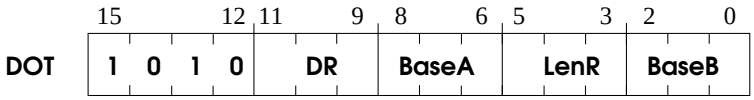
1- If line size is larger than 16B, access ③ kicks out the cach line brought in processor 1, which makes access ④ to be a miss

2 - If line size is smaller than 16B, accesses ⑤ & ⑥ write to a different cach line from ④, which makes access ⑦ to be a hit

Since scenario 1 & 2 cannot be true, line size is 16B

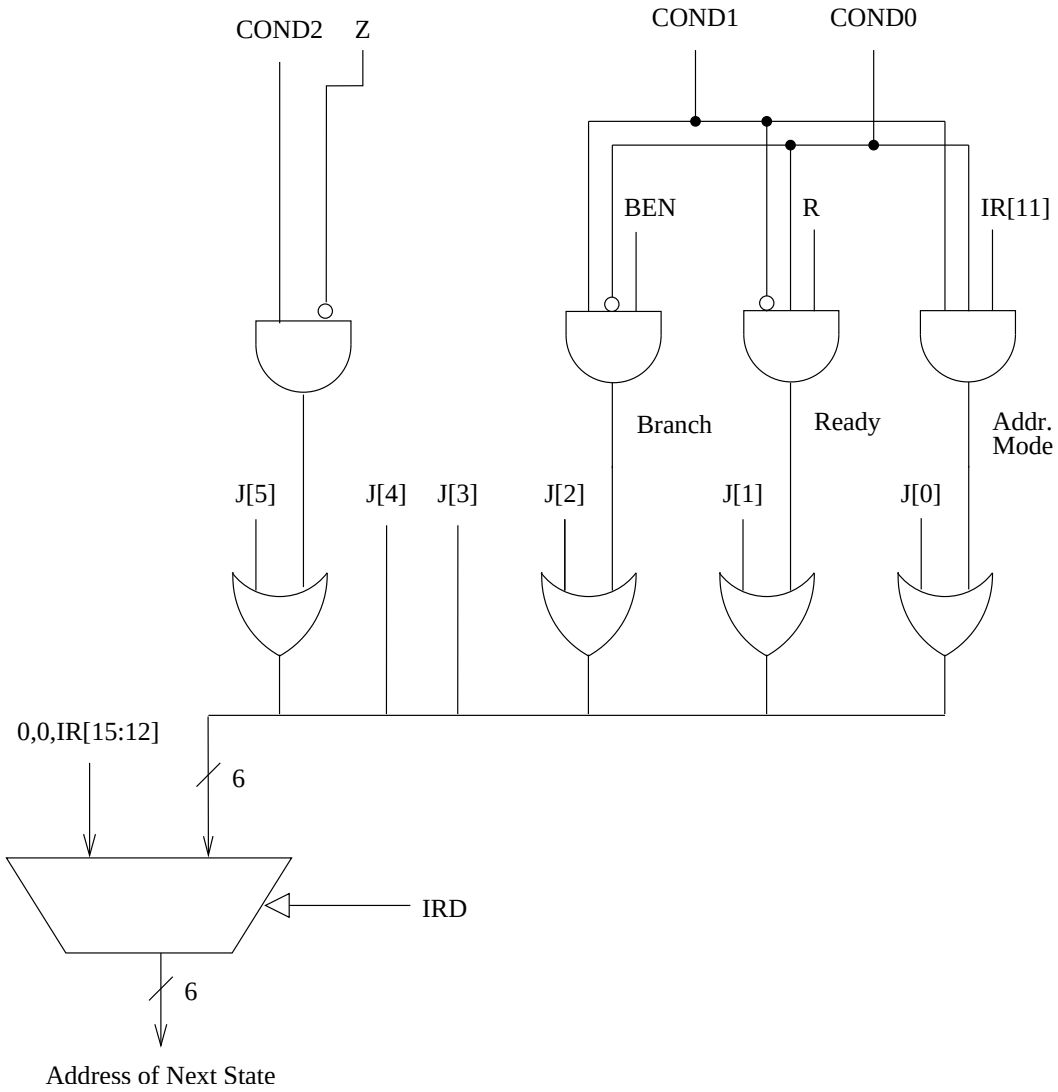
Name:_____

Problem 7 (30 points): We wish to use one of the unused opcodes of the LC-3b to define a new instruction, which we will call DOT. DOT performs the dot product (also called inner product) of two vectors. The dot product of two vectors A and B is equal to the sum of A[0] * B[0] + A[1] * B[1] + A[2] * B[2] + ... + A[N-1] * B[N-1]. The instruction format for DOT is:



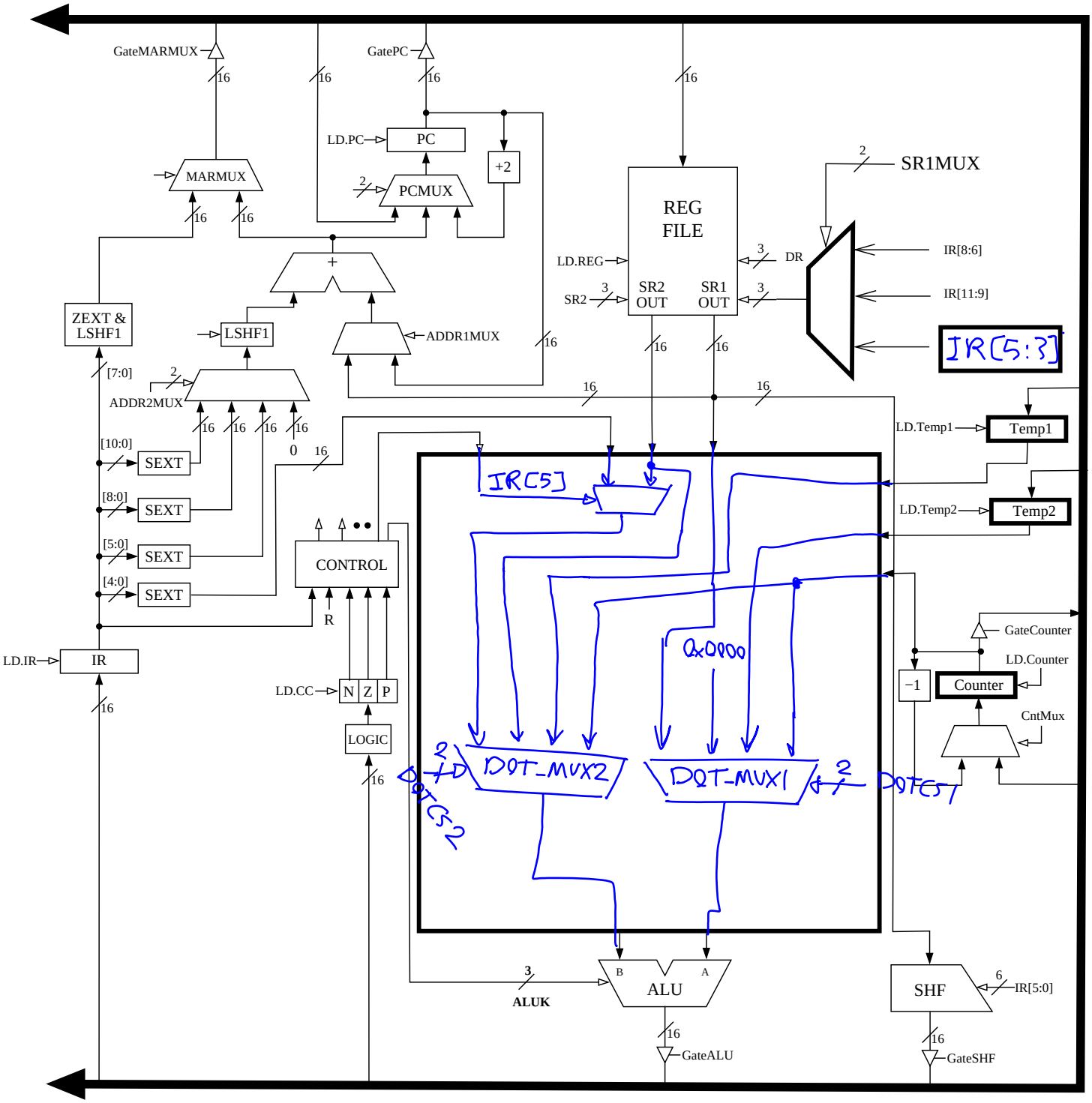
BaseA is a register that contains the starting address of vector A. BaseB is a register that contains the starting address of vector B. LenR is a register that contains the length of the vectors. Both vectors are simply byte arrays. DR must be different from BaseA and BaseB.

Your job: Implement DOT on the LC-3b by making the required changes to both the state machine and data path. We have filled in some of the states, and made some of the changes to the datapath. We have made all required changes to the microsequencer, which is shown below. Please use it as is; i.e., do not make any changes to it.



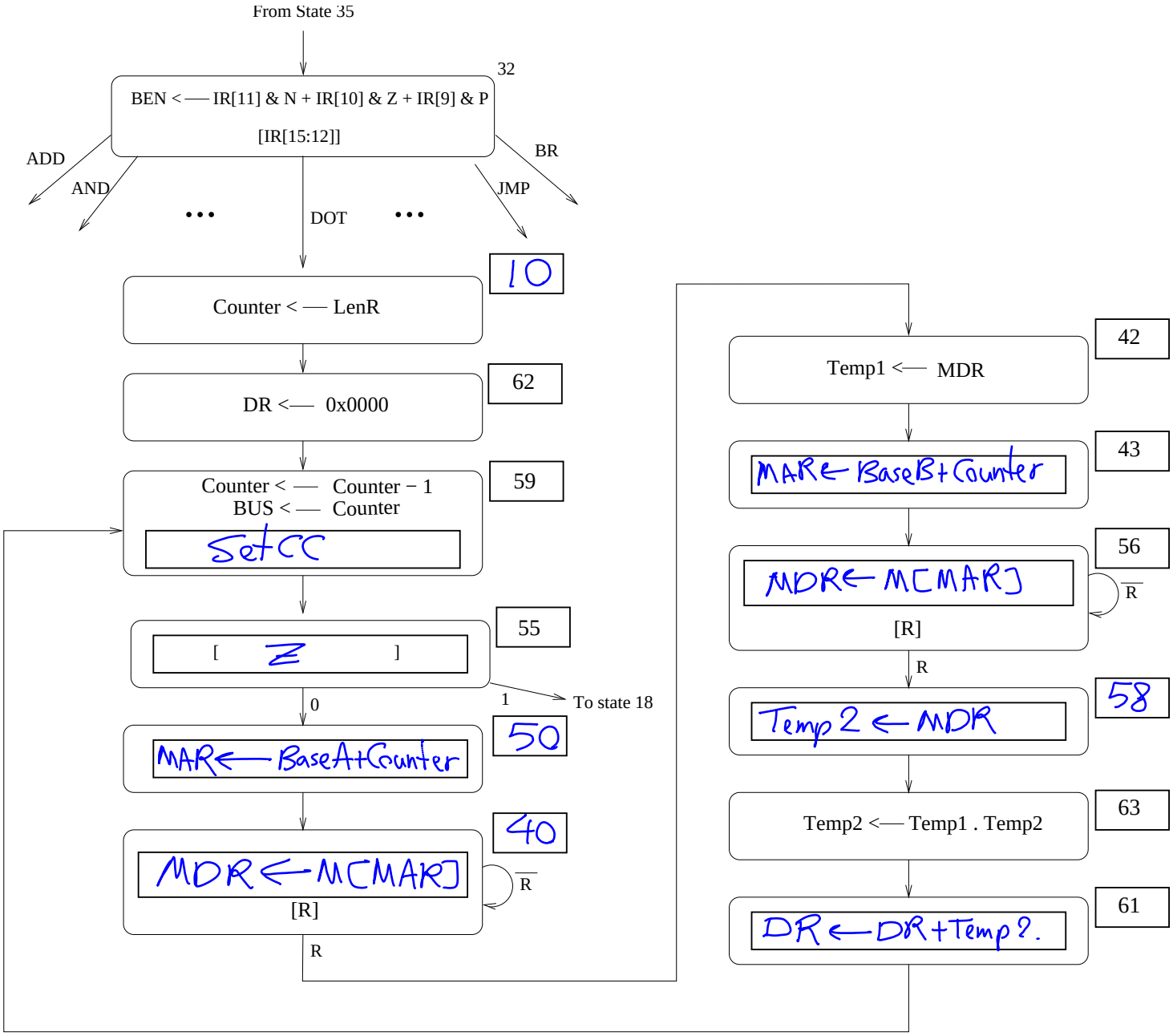
Name:_____

Part a (12 points): Complete the data path diagram by augmenting the SR1MUX and adding any other necessary structures and control signals inside the provided box. Note: we have given you the logic for a counter and two temporary registers. Note also that the ALU has been modfied to perform multiplication (ALUK is a 3-bit field).



Name:_____

Part b (16 points): Implement the state machine.



Part c (2 points): What change(s) would be required to relax the constraint that DR must not be the same register as BaseA or BaseB.

Use another Temp register to accumulate the sum