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

EE 460N Spring 2015
Y. N. Patt, Instructor
Ben Lin, Kishore Punniyamurthy, Will Hoenig TAs
Final Exam
May 15, 2015

Name: Solution

Problem 1 (10 points):_____

Problem 2 (10 points):_____

Problem 3 (10 points):_____

Problem 4 (20 points):_____

Problem 5 (25 points):_____

Problem 6 (30 points):_____

Problem 7 (25 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): An application that is 96% parallelizable is executed on a single processor in 2.5 hours. If the application is allowed to run with an unlimited number of processors, what is the lower bound on its execution time?

6 minutes

Part b (5 points): We wish to use even parity to protect each single-byte value we transmit, by adding a ninth bit. If we wish to transmit 01010101, what nine bits should we transmit?

01010101 0

If we wish to transmit 00110111, what nine bits should we transmit?

00110111 1

Name:_____

Problem 2 (10 points)

The following program fragment operates on 8-bit IEEE-like floating point format. Your job is to figure out how many bits for exponent, how many bits for fraction, and to complete the table below. BIAS (excess) is 4. Rounding is unbiased nearest.

```
float B;
float A = 5/16;

for(int i=0; i < 6; ++i)
{
    B = A/(1<<i);
}
```

Note that (1<<i) is equal to 2^i .

Each row of the table below specifies the **results** of one iteration of the for loop. Note that some iterations cause underflow and/or inexact exceptions, in addtion to producing a value for B.

Hint: Some representations of B are subnormal.

Iteration (i)	Binary Representation of B	Floating point representation of B			Exceptions	
					Underflow	Inexact
0	1.01 x 2^(-2)	0	0010	010	NO	NO
1	1.01 x 2^(-3)	0	0001	010	NO	NO
2	0.101 x 2^(-3)	0	0000	101	NO	NO
3	0.010 x 2^(-3)	0	0000	010	NO	YES
4	0.001 x 2^(-3)	0	0000	001	NO	YES
5	0	0	0000	000	YES	YES

Name:_____

Problem 3 (10 points)

Consider a tightly coupled mutltiprocessor system with two processors (P1 and P2). Each processor has its own private data cache.

The Goodman "write-once" snoopy cache protocol we studied in class is used for maintaining cache coherence. On a cache miss, if another processor has the line in the modified state, its cache supplies the line to the processor having the cache miss.

The table shows the behavior of the system for eight consecutive data accesses, all to location A. Assume the caches are initially empty. Each access is performed by either P1 or P2. Your job: complete the entries in the table.

Note: In the last column, the entry "No one" means there is no need to supply the cache line because the private cache had a cache hit.

Instance	P1 Executes a LOAD/STORE	P2 executes a LOAD/STORE	Bus Activity	Cache line supplied by
1	LOAD A	—	P_1 READS A	MEMORY
2	—	LOAD A	P2 READS A	MEMORY
3	STORE A	—	P1 WRITES A	NO ONE
4	—	STORE A	P_2 WRITES A	P1 CACHE
5	LOAD A	—	P1 READS A	P2 CACHE
6	—	STORE A	P_2 WRITES A	NO ONE
7	—	STORE A	—	NO ONE
8	LOAD A	—	P_1 READS A	P_2 CACHE

Name:_____

Problem 4 (20 points)
Recall the Tomasulo problem on midterm two. The rules are very similar here.

Instructions are of the form ADD Rx,Ry,Rz and MUL Rx,Ry,Rz, as discussed in class. Each instruction requires a fetch cycle, a decode cycle, some number of execution cycles, and a final cycle to store the result into a register and/or a reservation station entry that is waiting for that result. A result is available to subsequent instructions after it is stored in a register or reservation station entry. Functional units not pipelined. Reservation stations are assigned from the top down. The top-most reservation station with both data entries valid is the next to be processed. Each instruction remains in its reservation station until its result is stored.

The only differences in the problem today are the following: There may be more than one adder and more than one multiplier. We have changed the number of execution cycles to 3 cycles for the adder and 4 cycles for the multiplier. All the adder(s) share three reservation stations. All the multiplier(s) likewise share three reservation stations. Finally, each instruction has two unique source registers; that is, for all instructions OP Rx,Ry,Rz, $y \neq z$

A program fragment, consisting of five instructions, is executed on this machine. The first instruction is fetched in cycle 1. Part of your job: Complete the table below, i.e., the complete specification of the five instructions.

Instruction	Opcode	DR	SR1	SR2
1	MUL	R0	R1	R2
2	ADD	R1	R0	R2
3	MUL	R1	R2	R3
4	ADD	R2	R2	R3
5	ADD	R3	R0	R3

Information on the next page will help you identify the five instructions executed.

Name:_____

The table below shows in what cycles the function units are executing. An **E** in row ADD indicates that in that cycle at least one adder is executing. An **E** in row MUL indicates that at least one multiplier is executing.

	1	2	3	4	5	6	7	8	9	10	11	12
ADD						E	E	E	E	E	E	
MUL			E	E	E	E	E	E	E	E		

Initial values in the Register File are shown below:

R0	1	–	7
R1	1	–	4
R2	1	–	5
R3	1	–	9

The rest of your job is as follows: Provide the missing entries in the Register File and in the reservation stations for the adder(s) and multiplier(s) at the end of cycle X and at the end of cycle 10. Identify which cycle is X.

After Cycle X [4]

Register File:

R0	0	π	–
R1	0	σ	–
R2	1	–	5
R3	1	–	9

Reservation Stations for adder(s):

α	0	π	–	1	–	5
β	–	–	–	–	–	–
γ	–	–	–	–	–	–

Reservation Stations for multiplier(s):

π	1	–	4	1	–	5
σ	1	–	5	1	–	9
τ	–	–	–	–	–	–

After Cycle 10

Register File:

R0	1	–	20
R1	0	σ	–
R2	1	–	14
R3	0	γ	–

Reservation Stations for adder(s):

α	1	–	20	1	–	5
β	–	–	–	–	–	–
γ	1	–	20	1	–	9

Reservation Stations for multiplier(s):

π	–	–	–	–	–	–
σ	1	–	5	1	–	9
τ	–	–	–	–	–	–

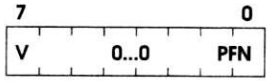
Finally, how many adders and multipliers are there?

Adders [2] Multipliers [1]

Name:_____

Problem 5 (25 points): An LC-3b supporting VAX-style virtual memory has 16-bit virtual addresses, 11-bit physical addresses, and a 128-byte page size. User space occupies virtual memory locations 0x0000 to 0x7FFF; system space occupies locations 0x8000 to 0xFFFF.

Each PTE has the following format (Note: The PFN size is not given):



A user space TLB contains two entries, with perfect LRU replacement.

The computer executes the following three instructions:

```
LDW R0,R1,#0
ADD _________
STW R0,R2,#0
```

You can assume no exceptions occur during their processing.

Before execution:

The TLB contains one valid entry: Page x14, 10000110
PC: x4000
R2: x1278

Your job: Complete the table and fill in the four additional boxes.

Virtual Address	Physical Address	Data	TLB Hit
—	x191	x81	MISS
x8880	x080	x84	MISS
x4000	x200	x6040	MISS
—	x190	x82	MISS
x8860	x160	x88	MISS
x3064	x464	xABCD	MISS
x4002	x202	x1021	HIT
x4004	x204	x7080	HIT
—	x190	x82	MISS
x8824	x124	x87	MISS
x1278	x3F8	xABCE	MISS

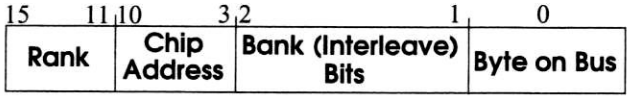
UBR: x8800 SBR: x180 Initial value of R1: x3064

The second instruction: 15 0 0 0 1 0 0 0 0 0 0 1 0 0 0 0 1 0

Name:_____

Problem 6 (30 points)
A 64KB byte-addressable memory is 4-way interleaved. The processor/memory bus is 16 bits wide, and each memory access takes 4 cycles. There is only 1 channel.

The address bits are specified as shown below:



128 64-bit signed integers are stored in a 1024-byte array, starting at address x0000. We wish to know how many of these integers are negative.

The sign of a 64-bit signed integer is specified by its sign bit, i.e., the most significant bit of the most significant byte of the integer. Thus, to determine the signs of all 128 integers, we only need to load and examine 128 bytes from memory, as opposed to all 1024 bytes of the array.

Part a: What is the minimum number of cycles needed to read these 128 bytes from memory?

The number of cycles needed is the same regardless of endianness. If little-endian, then the sign bits are at bytes 7, 15, 23, .., 1023. If big-endian, the sign bits are ae bytes 0, 8, 16, .., 1016. Either way the difference between different accesses is 8B. Since the bank (interleave) bits are bits [2:1], adding 8 to an address will not change the bank that the address maps to. Furthermore, a 1024-byte array fits entirely in rank 0. Hence all 128 accesses will conflict to the same bank and need to be serialized, requiring $4 \times 128 =$ **512 cycles** total.

Part b: A smart engineer realized the time to read these 128 bytes from memory can be decreased if two bytes of padding were added to each array element (i.e. the entire array now requires 1280 bytes instead of 1024 bytes). What's the minimum number of cycles needed to read the 128 bytes from memory after padding has been added?

Since the bank (interleave) bits are bits [2:1] of the address, adding 2 to the address will always move you to a new bank. It does not matter whether the 2 bytes of padding is added before or after the 8 byte integer, aslong as it's consistent. Thus with the padding, successive accesses will always go to different banks and we get perfect 4-way interleaving. In cycle 1 we start the access for the 1st byte, in cycle 2 we start the access for the 2nd byte, and so forth, and in cycle 128 we start the last (128th) access. The last accesses finishes in cycle **131**, since memory latency is 4 cycles.

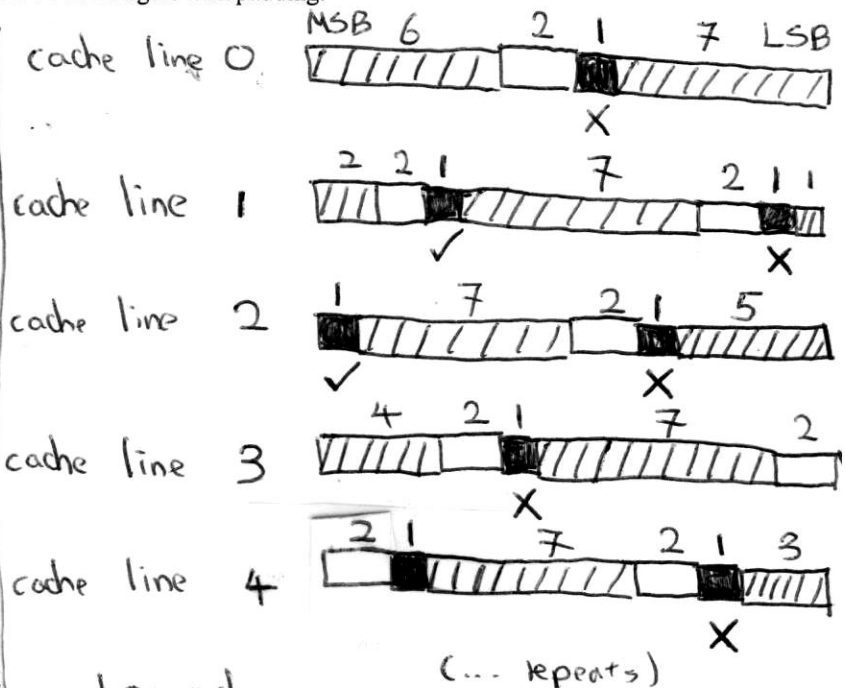
Name:_____

Part c: Assume our algorithm for determining the number of negative integers processes the 128 integers in sequential order. As previously stated, the algorithm only needs to access 1 byte per integer to determine its sign. The processor includes an initially empty 8KB, 4-way set associative data cache having a line size of 16 bytes. Compute the cache hit ratio when the algorithm processes 64-bit integers without padding.

Note the cache is initially empty, and each byte of integer data brought into the cache will only be used once (i.e. there is no reuse). Hence the only hits we'll get are hits to other integers contained within the same cache line. Since the line size is 16B, and each integer is 8B, we have 2 integers per cache line. The access to the first integer in the cache line results in a miss; the next access will be to the other integer in the same cache line, resulting in a hit. The cache hit ratio is thus **50%**

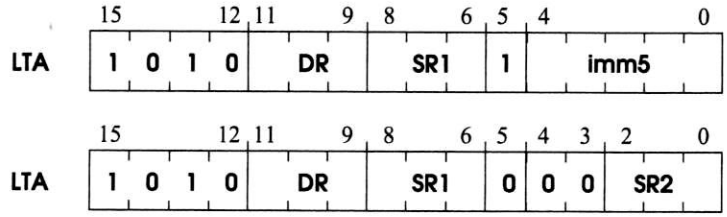
Compute the cache hit ratio when the algorithm processes 64-bit integers with padding.

With extra padding, each integer now takes up 10B, and may not align to cache line boundaries. However, every 8th integer will still align to cache line boundaries, as each integer is 10B, each cache line 16B, and the Least Common Multiple of 10B and 16B is 80B, which is 8 integers. Hence we only need to compute the cache hit ratio for the first 8 integers, as the same memory layout is repeated. The diagram on the right assumes little-endianness and assumes the padding comes after the integer, but the cache hit ratio is the same regardless of endianness or placement of padding - 3 hits for every 8 accesses, or **3/8**

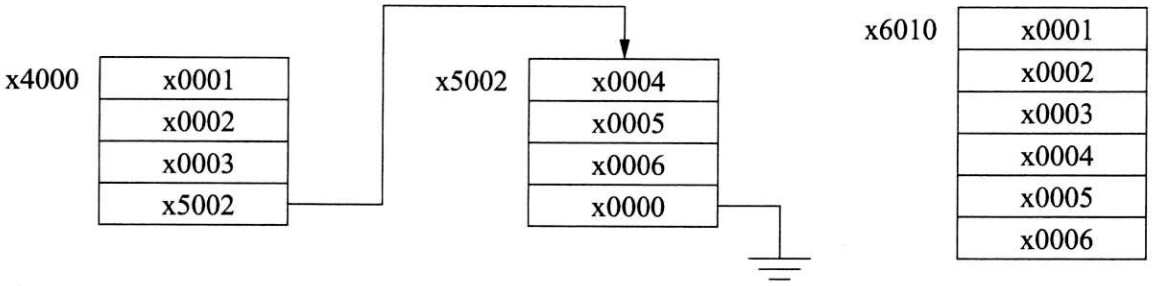


Name:_____

Problem 7 (25 points): We wish to augment the LC-3b with a new instruction LTA, which copies a linked list into a sequential array. (LTA: Linked To Array). The LTA instruction can be of either of the following two formats:



An example may help explain what is going on:



Linked list to be copied

Resulting sequential array

SR1 contains the memory address of the head of the linked list (x4000, in the above example). Each node in the linked list consists of n consecutive 16-bit words, followed by the pointer to the next node. SR2 or the immediate field contains the value for n (in the above example, n=3).

LTA copies the nodes into sequential locations of memory, starting with the location specified by DR (in the above example, x6010). Note: there is no need to copy the pointers, since the nodes are now stored in sequential memory locations.

Assume that DR, SR1, and SR2 (if SR2 is being used) all refer to different registers. Assume that all the linked list nodes and the destination array are aligned in memory and do not overlap.

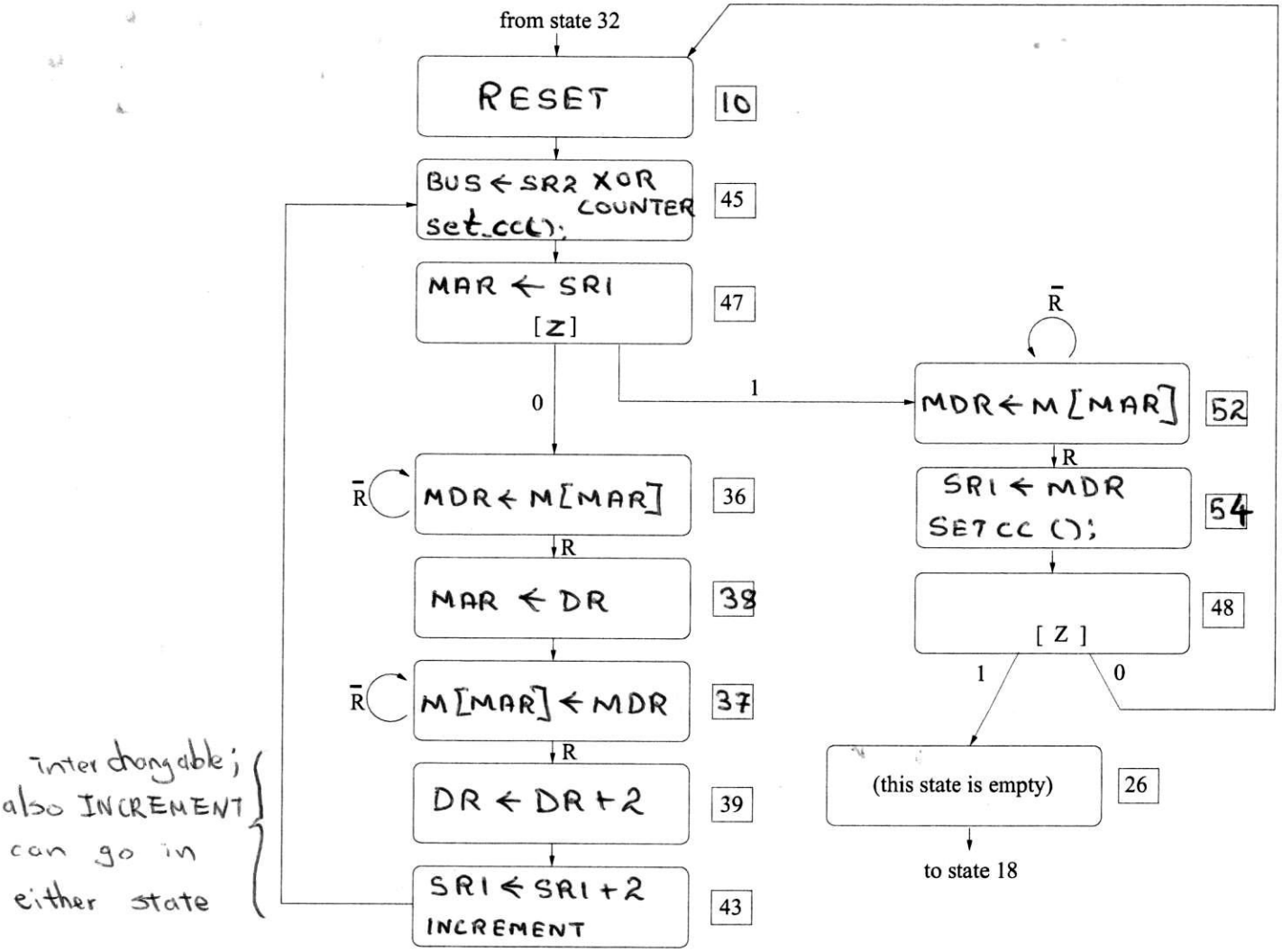
Note: After processing the LTA instruction, SR1 contains the null pointer, since the linked list no longer exists; DR contains the address of the next location following the sequential array (in the above example x601C). The condition code will be set to Z.

Part A. Implement the state machine.

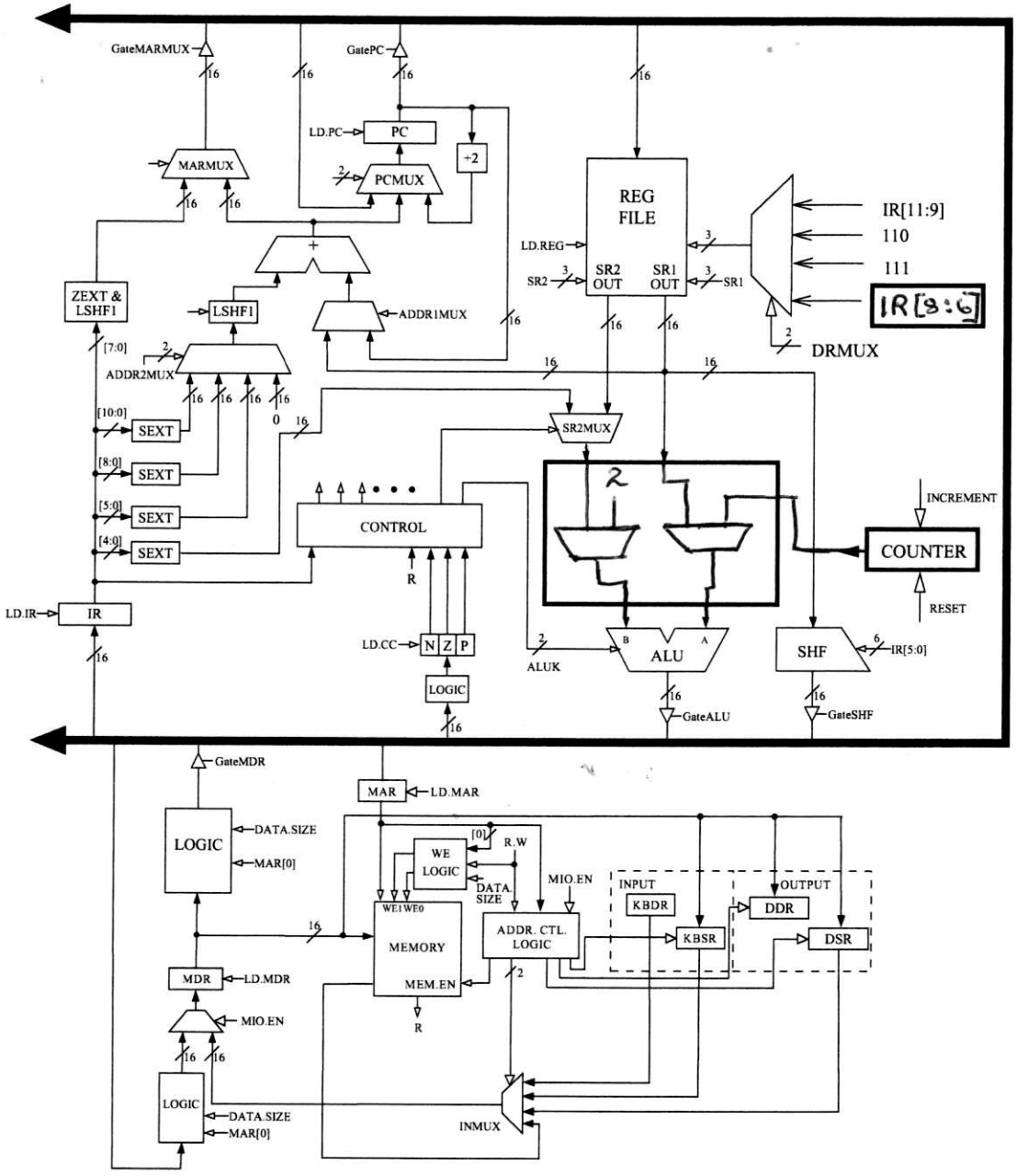
Part B. Complete the data path diagram by augmenting the DRMUX and adding any other necessary structures and control signals inside the provided box. Note: we have given you a counter which can be incremented or reset to 0. **Hint: A xor A = 0**

Part C. Complete the microsequencer. Hint: you will need to add one AND gate and one OR gate.

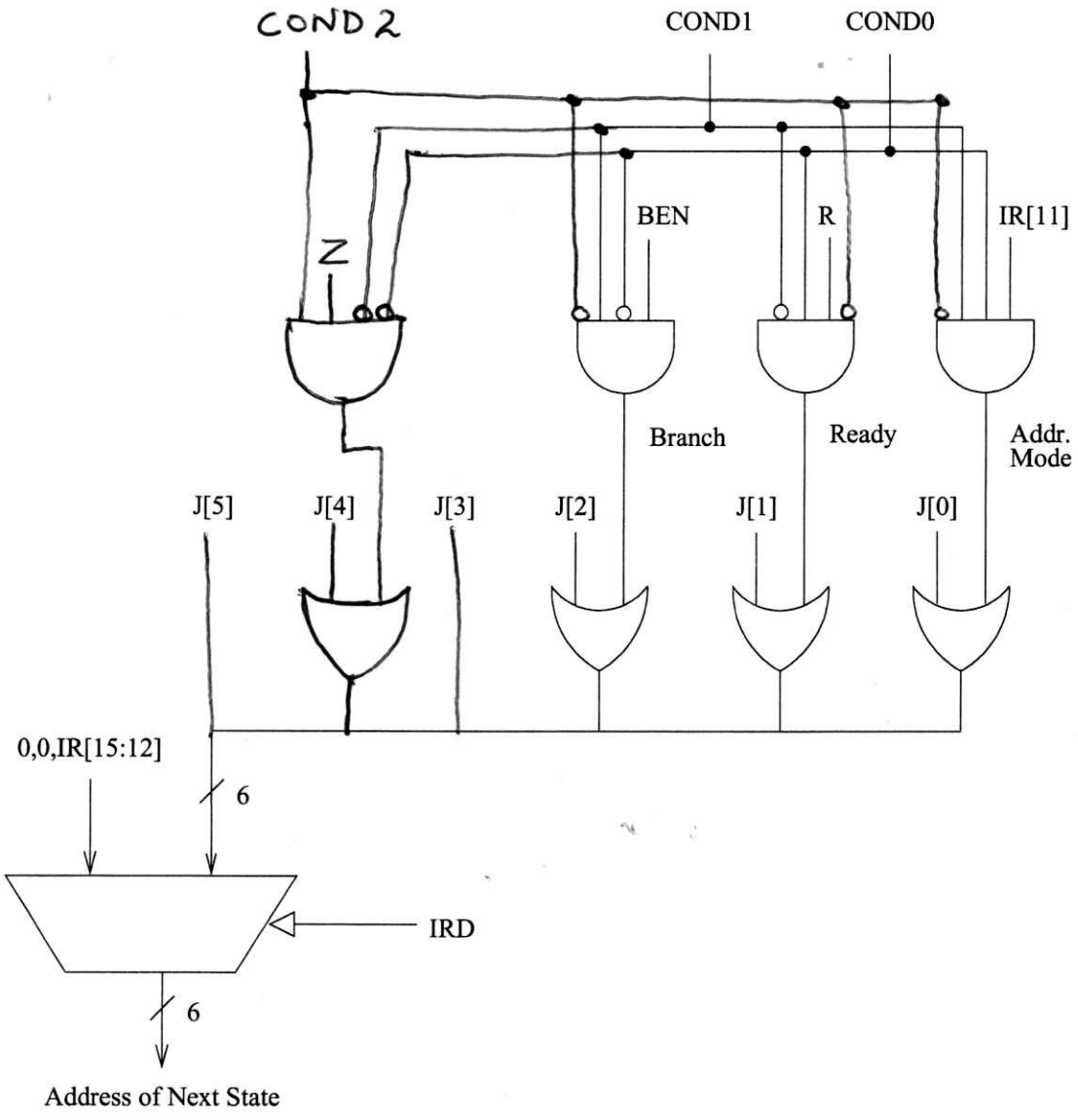
Name:_____


from state 32
RESET
10
BUS ← SR2 XOR COUNTER
set_cc();
45
MAR ← SR1
[Z]
47
0
1
R̄
MDR ← M[MAR]
52
R
SR1 ← MDR
SET CC ();
54
[Z]
48
1
0
(this state is empty)
26
to state 18
R̄
MDR ← M[MAR]
36
R
MAR ← DR
38
R̄
M[MAR] ← MDR
37
R
DR ← DR + 2
39
SR1 ← SR1 + 2
INCREMENT
43
interchangable; also INCREMENT can go in either state

Name:_____


GateMARMUX
GatePC
LD.PC
PC
+2
PCMUX
MARMUX
REG FILE
LD.REG
SR2
SR2 OUT
SR1 OUT
SR1
IR[11:9]
110
111
IR[8:6]
DRMUX
ZEXT & LSHF1
LSHF1
ADDR1MUX
ADDR2MUX
[7:0]
[10:0]
[8:0]
[5:0]
[4:0]
SEXT
SR2MUX
CONTROL
INCREMENT
COUNTER
RESET
LD.IR
IR
LD.CC
N Z P
ALUK
ALU
SHF
IR[5:0]
LOGIC
GateALU
GateSHF
GateMDR
MAR
LD.MAR
DATA.SIZE
MAR[0]
WE LOGIC
R.W
MIO.EN
INPUT
OUTPUT
KBDR
DDR
KBSR
DSR
WE1 WE0
MEMORY
MEM.EN
ADDR. CTL. LOGIC
MDR
LD.MDR
INMUX

Name:_____


COND 2
COND1
COND0
BEN
R
IR[11]
Z
Branch
Ready
Addr.
Mode
J[5]
J[4]
J[3]
J[2]
J[1]
J[0]
0,0,IR[15:12]
6
IRD
6
Address of Next State