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

EE 306, Fall 2011
Yale Patt, Instructor
Faruk Guvenilir, Milad Hashemi, Jennifer Davis, Garrett Galow,
Ben Lin, Taylor Morrow, Stephen Pruett, Jee Ho Ryoo TAs
Final Exam, December 9, 2011

Name:___________________________

Part A:

Problem 1 (10 points):_______

Problem 2 (10 points):_______

Problem 3 (10 points):_______

Problem 4 (10 points):_______

Problem 5 (10 points):_______

Part A (50 points):

Part B:

Problem 6 (20 points):_______

Problem 7 (20 points):_______

Problem 8 (20 points):_______

Problem 9 (20 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.

I will not cheat on this exam.

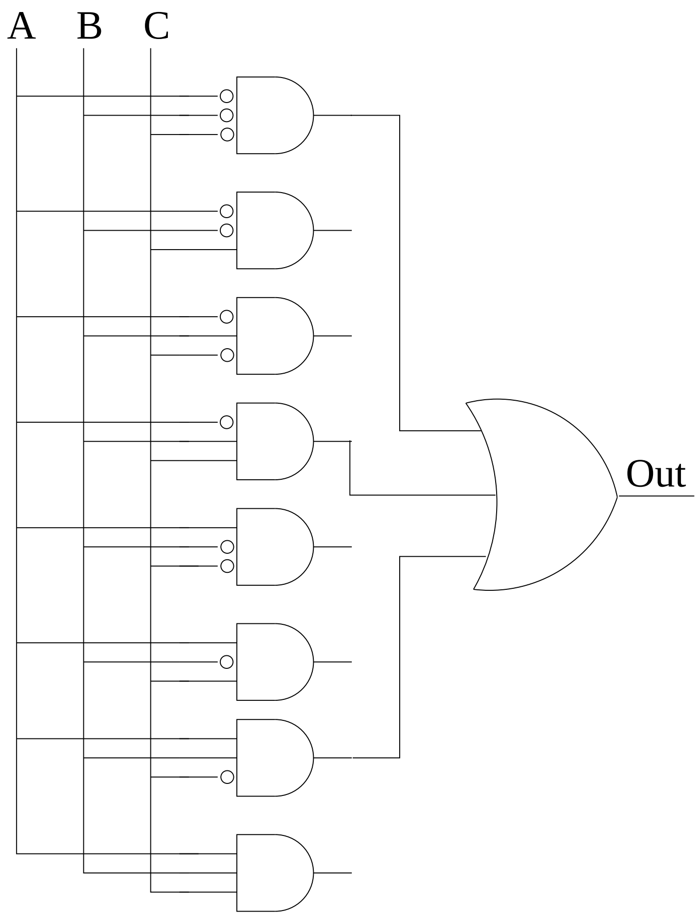
____
Signature

GOOD LUCK!
(HAVE A GREAT SEMESTER BREAK)

Name:____

Problem 1. (10 points):

Part a. (5 points): Construct the output of the truth table for the PLA shown.

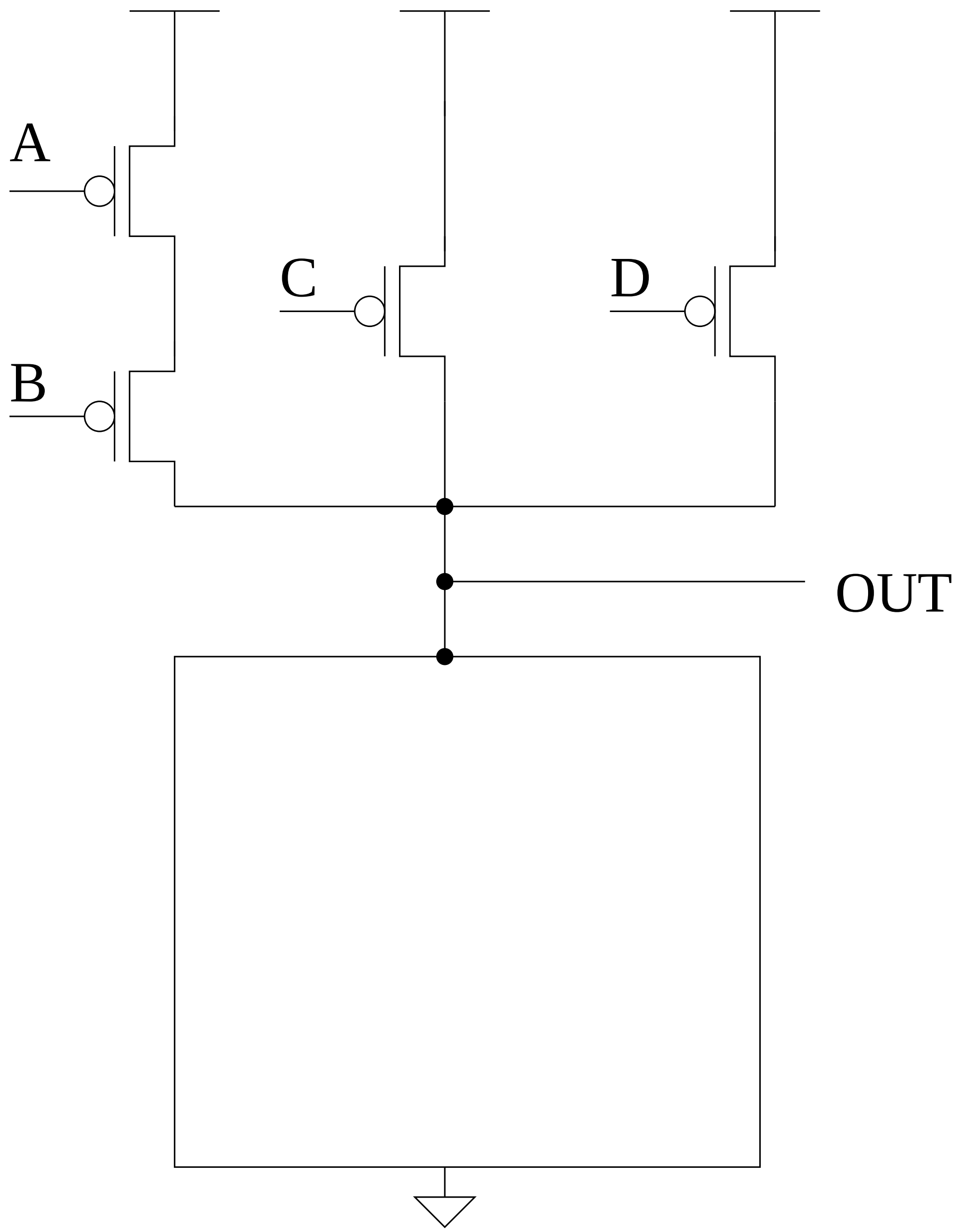


A	B	C	Out
0	0	0	
0	0	1	
0	1	0	
0	1	1	
1	0	0	
1	0	1	
1	1	0	
1	1	1	

Name:____

Part b. (5 points): In the transistor circuit below, all transistors in the path to the power supply are shown. None of the transistors in the path to ground are shown.

Your job:
1. Draw the missing transistor circuit in the box.



Name:____________________________________

Problem 2. (10 points): The following program is assembled and stored in the LC-3's memory. The PC is initially set to x3000. The program is run until the computer halts.

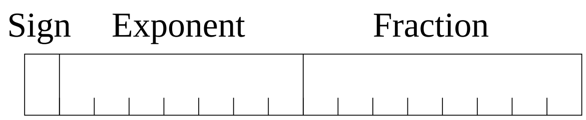
Your job: What is contained in location B after the computer stops?

```
      .ORIG x3000
      AND   R0,R0,#0
      NOT   R1,R0
      ADD   R5,R0,#3
      ADD   R0,R0,#1
      ADD   R0,R0,R0
      ADD   R0,R0,R0
      ADD   R0,R0,R0
      NOT   R3,R0
      AND   R1,R3,R1
A     ADD   R0,R0,R0
      ADD   R0,R0,R0
      ADD   R0,R0,R0
      ADD   R0,R0,R0
      NOT   R3,R0
      AND   R1,R3,R1
      ADD   R5,R5,#-1
      BRp   A
      ST    R1,B
      TRAP  x25
B     .BLKW 1
```

What is the value in location B?

Name:____

Problem 3. (10 points): This problem involves a new 16-bit floating point data type, specified as follows:



To add two floating point values, we first make sure their binary points line up (they have the same exponents).

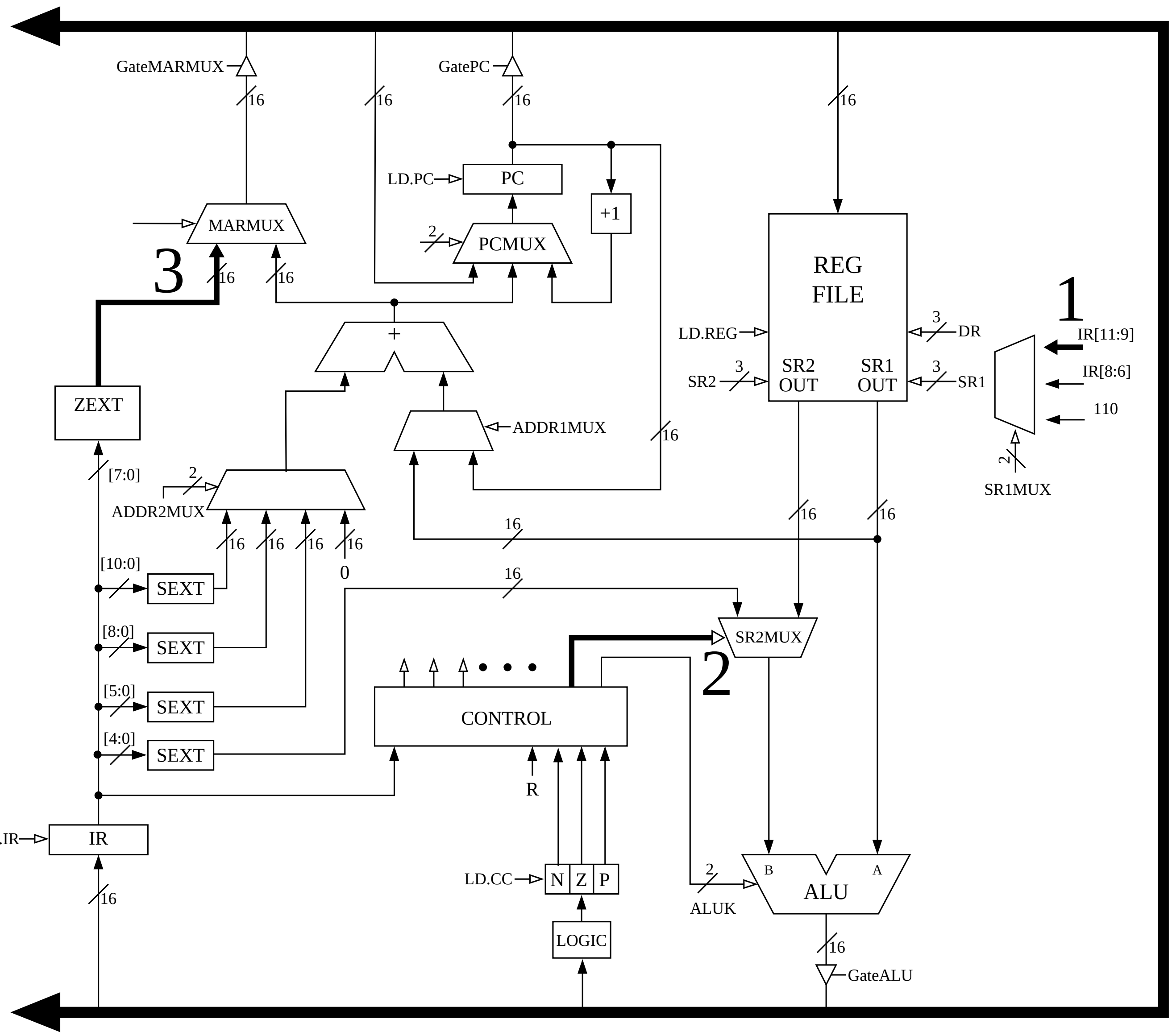
The assembly program shown below, after the missing instructions have been filled in, compares the exponents of two floating point numbers that have been previously loaded into locations A and B. If the exponents are the same, R5 is set to 0 before the RET is taken. If the exponents are different, R5 is set to 1 before the RET is taken.

Your job: Fill in the missing instructions.

```
        .ORIG  x3000
        ST R0,SaveR0
        ST R1,SaveR0
      [                    ]
        LD R2, MASK
        AND R5, R5, #0
        LD R0,A
        LD R1,B
      [                    ]

      [                    ]
        NOT R1, R1
        ADD R1,R1,#1
      [                    ]
        BRz DONE
        ADD R5,R5,#1
DONE    LD R0,SaveR0
        LD R1,SaveR1
      [                    ]
        RET
MASK  [                    ]

A      .BLKW  #1
B      .BLKW  #1
SaveR0 .BLKW  #1
SaveR1 .BLKW  #1
      [                    ]
C
       .END
```

Name:____

Problem 4. (10 points):



1. What opcodes use IR[11:9] as inputs to SR1?

2. Where does the control signal of this mux come from? Be specific!

3. What opcodes use this input to the MARMUX?

Name:______________________________________

Problem 5. (10 points): The modulo operator (A mod B) is the remainder one gets when dividing A by B. For example, 10 mod 5 is 0, 12 mod 7 is 5.
The program below is supposed to perform A mod B, where A is in x3100 and B is in x3101. The result should be stored at location x3200. However, the programmer made a serious mistake, so the program does not work. You can assume that A and B are both positive integers.

```
         .ORIG x3000        ; Line 1
         LD R3, L2          ; 2
         LDR R0, R3, #0     ; 3
         LDR R1, R3, #1     ; 4
         NOT R2, R1         ; 5
         ADD R2, R2, #1     ; 6
L1       ADD R0, R0, R2     ; 7
         BRzp L1            ; 8
         ADD R0, R0, R1     ; 9
         ST R0, L3          ; 10
         HALT               ; 11
L2       .FILL x3100        ; 12
L3       .FILL x3200        ; 13
         .END               ; 14
```

Part A. After the instruction at line 6 has executed, what are the contents of R0,R1,and R2? NOTE: the correct answer in each case is one of the following: A, -A, B, -B, 0, 1, -1.

R0: [ ] R1: [ ] R2: [ ]

Part B. There is a bug in the program. The instruction at line [ ] should be [ ]

Name:____

Problem 6. (20 points): A free list is a collection of blocks of consecutive memory locations of various sizes that are not being used by currently executing programs. A free list is normally organized as a linked list, where each element in the linked list is associated with a single block of memory. Each element consists of three words: the address of the next element in the linked list, the number of consecutive memory locations in this block, and the starting address of the block. R1 contains the address of a memory location that points to the first node in the free list.

```
R1: xC000    xC000: x8000      x8000: xA000     xA000: x0000
                               x8001: x0100     xA001: x0010
                               x8002: x6000     xA002: x7050
```

The free list above consists of two nodes, one of size x100 comprising M[x6000] to M[x60FF] and one of size x10 comprising locations M[x7050] to M[x705F].

A procedure MALLOC is used to provide blocks of storage to programs that request them.

If Program A needs n words of memory, it loads n into R2 and does a JSR to MALLOC. MALLOC finds the first block in the free list that can satisfy the request, loads the starting address of the block into R0, updates the free list to reflect the fact that those n words are no longer available, and does a JMP R7. If MALLOC can't find a block that can satisfy the request, x0000 is returned in R0. If the block that supplied the n-words consisted of exactly n-words (a perfect fit), then no words from that block are still available and so the node is removed from the free list.

On the next page is the procedure MALLOC. Your job: Add the missing instructions.

Name:____

```
MALLOC      ST  R1, SAVE_R1
            ST  R3, SAVE_R3
            ST  R4, SAVE_R4
            ST  R5, SAVE_R5

            AND R0, R0, #0
            NOT R3, R2
            ADD R3, R3, #1

NEXT_NODE   LDR R4, R1, #0
            BRz RETURN
            LDR R5, R4, #1
            ADD R5, R3, R5
            BRz PERFECT_FIT
            BRp FRAGMENT

      [                                    ]

            BRnzp NEXT_NODE
PERFECT_FIT LDR R0, R4, #2

      [                                    ]

            STR R4, R1, #0
            BRnzp   RETURN
FRAGMENT    LDR R0, R4, #2
            STR R5, R4, #1

      [                                    ]

            STR R1, R4, #2

RETURN      LD  R5, SAVE_R5
            LD  R4, SAVE_R4
            LD  R3, SAVE_R3
            LD  R1, SAVE_R1
            RET
SAVE_R1     .BLKW 1
SAVE_R3     .BLKW 1
SAVE_R4     .BLKW 1
SAVE_R5     .BLKW 1
```

Name:____

Problem 7. (20 points): During the processing of an LC-3 program by the data path we have been using in class, the computer stops due to a breakpoint set at x3000. The contents of certain registers and memory locations at that time are as follows:

```
R2 through R7: x0000
     M[x3000]: x1263
     M[x3003]: x0000
```

The LC-3 is restarted and executes exactly four instructions. To accomplish this, a number of clock cycles are required. In 15 of those clock cycles, the bus must be utilized. The table below lists those 15 clock cycles in sequential order, along with the values that are gated onto the LC-3 bus in each.

	BUS
1st:	x3000
2nd:	x1263
3rd:	x009A
4th:	x3001
5th:	xA000
6th:	
7th:	
8th:	
9th:	
10th:	
11th:	
12th:	
13th:	x3003
14th:	x1263
15th:	x009D

Part a: Fill in the missing entries above.

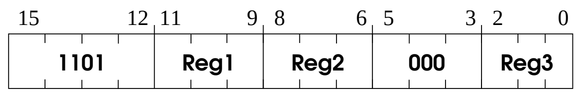
Part b: What are the four instructions that were executed?

Part c: What are the contents of R0 and R1 after the four instructions execute?

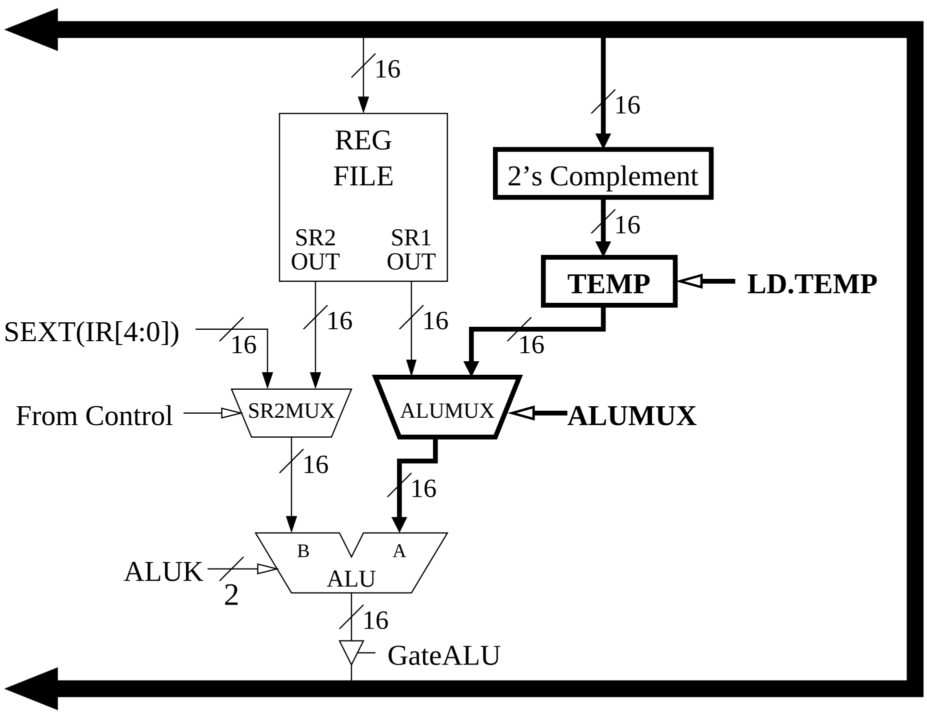
R0: ____________ R1: ____________

Name:____

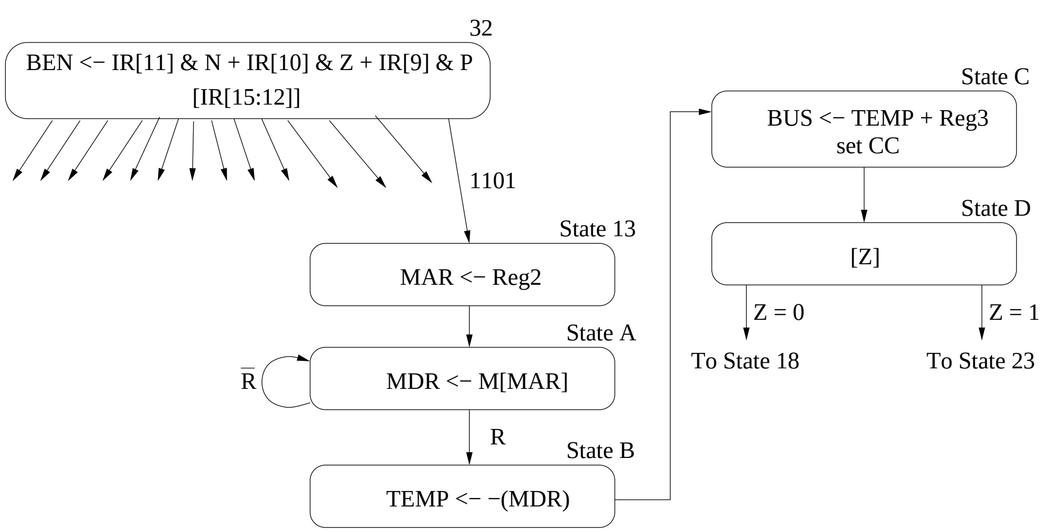
Problem 8. (20 points): Let’s use the unused opcode to implement a new instruction, as shown below:



To accomplish this, we will need a small addition to the data path, shown below in boldface:



The following five additional states are needed to control the data path to carry out the work of this instruction.



Note: State B loads the negative of the contents of MDR into TEMP.

Name:____

Part a: Complete the table below by identifying the values of the control signals needed to carry out the work of each state.

Note: For a particular state, if the value of a control signal does not matter, fill it with an X.

	LD.PC	LD.MAR	LD.MDR	LD.CC	LD.TEMP	GatePC	GateMDR	GateALU	SR1MUX[1:0]		ALUMUX	ALUK[1:0]		MIO.EN	R.W
State 13															
State A															
State B															
State C															
State D															

```
LD.PC       0: load not enabled
            1: load enabled

LD.MAR      0: load not enabled
            1: load enabled

LD.MDR      0: load not enabled
            1: load enabled

LD.CC       0: load not enabled
            1: load enabled

LD.TEMP     0: load not enabled
            1: load enabled

GatePC      0: do not pass signal
            1: pass signal

GateMDR     0: do not pass signal
            1: pass signal

GateALU     0: do not pass signal
            1: pass signal

SR1MUX      00: Source IR[11:9]
            01: Source IR[8:6]
            10: Source R6

ALUMUX      0: Choose SR1
            1: Choose TEMP

ALUK        00: ADD
            01: AND
            10: NOT
            11: Pass input A

MIO.EN      0: MIO not enabled
            1: MIO enabled

R.W         0: Read
            1: Write
```

Part b: What does the new instruction do?

Name:____

Problem 9. (20 points): Consider a two player game where the players must think quickly each time it is their turn to make a move. Each player has a total allotted amount of time to make all his/her moves. Two clocks display the remaining time for each player. While a player is thinking of his/her move, his clock counts down. If time runs out, the other player wins. As soon as a player makes his/her move, he hits a button, which serves to stop counting down his clock and start counting down the other player's clock.

The program on the next page implements this mechanism. The main program keeps track of the time remaining for each player by decrementing the proper counter once per second while the player is thinking. When a player's counter reaches zero, a message is printed on the screen declaring the winner. When a player hits the button, an interrupt is taken. The interrupt service routine takes such action as to enable the main program (after returning from the interrupt) to start decrementing the other counter.

The interrupt vector for the button is x35. The priority level of the button is #2. Assume that the operating system has set the Interrupt Enable bit of the button to enable it to interrupt. Assume the main program runs at priority #1 and executes in user mode.

Part a: In order for the interrupt service routine to be executed when the button is pushed, what memory location must contain what value?

Address: ☐ Value: ☐

Part b: Assume a player hits the button while the instruction at line 16 is being executed. What two values (in hex) will be pushed on the stack?

☐ ☐

Part c: Fill in the missing instructions in the user program.

Part d: This program has a bug that will only occur if an interrupt is taken at an inappropriate time. Write down the line number of an instruction such that if the button is pressed while that instruction is executing, unintended behavior will result.

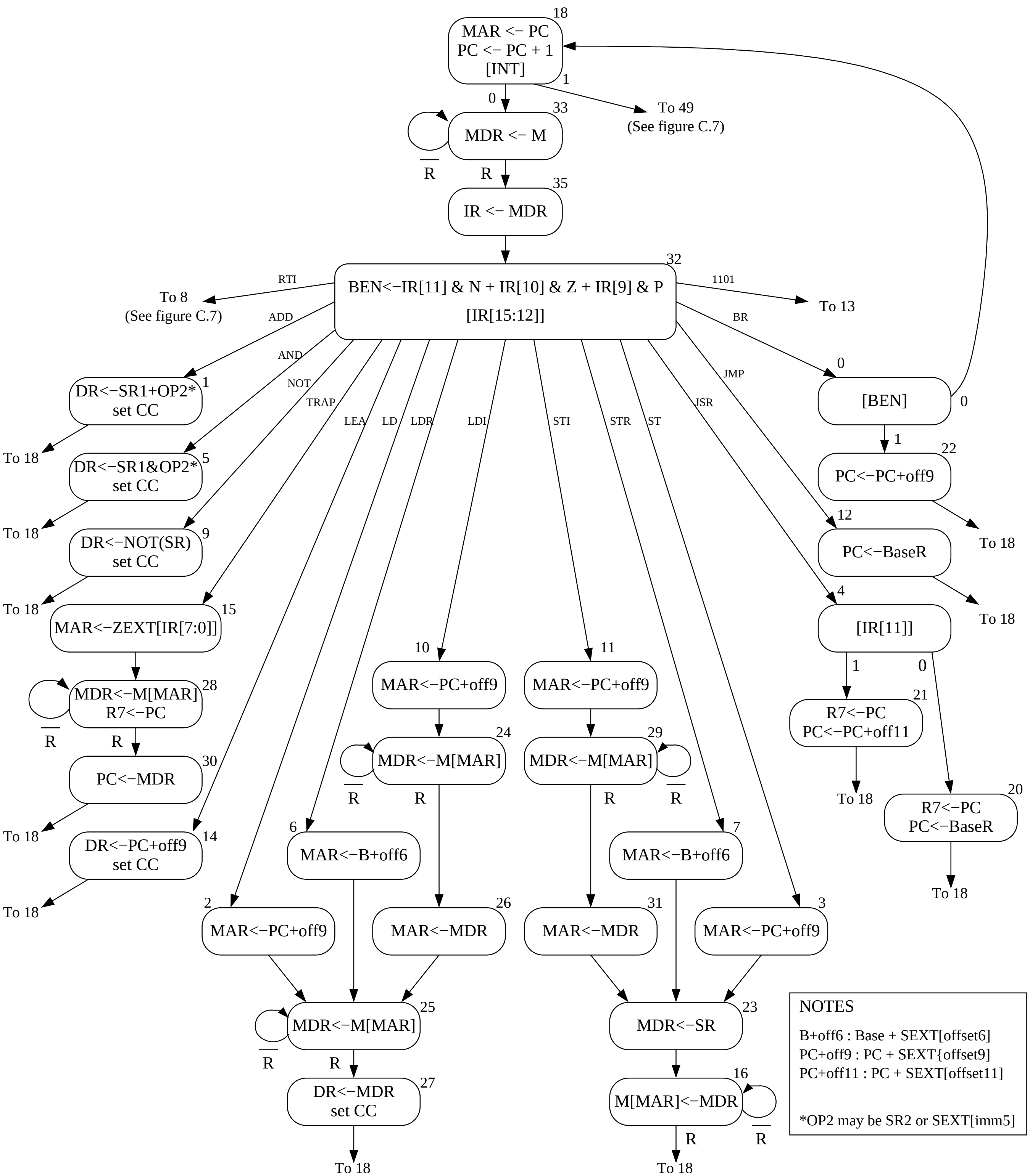
Line Number: ☐

How could we fix this bug?

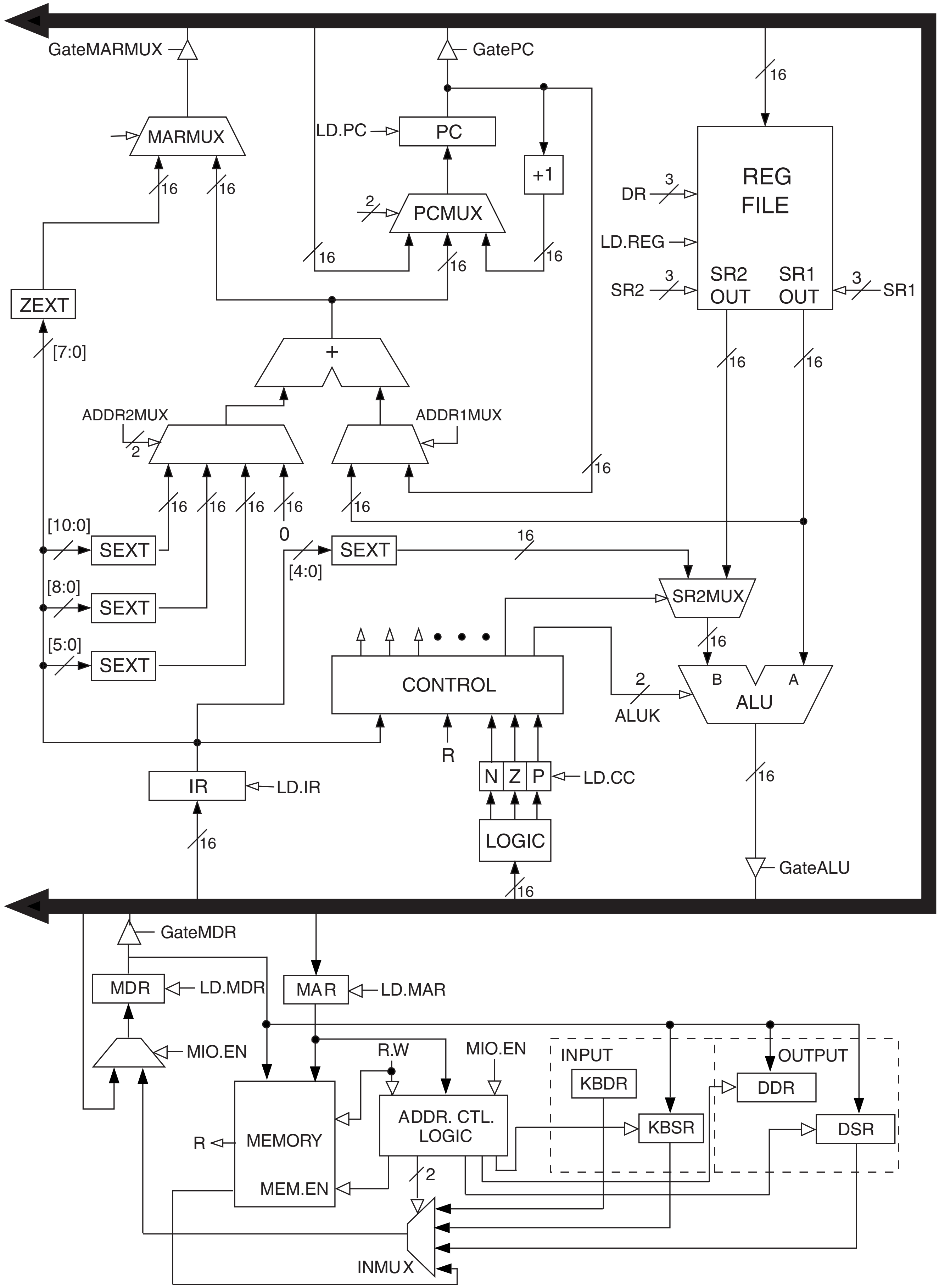
☐

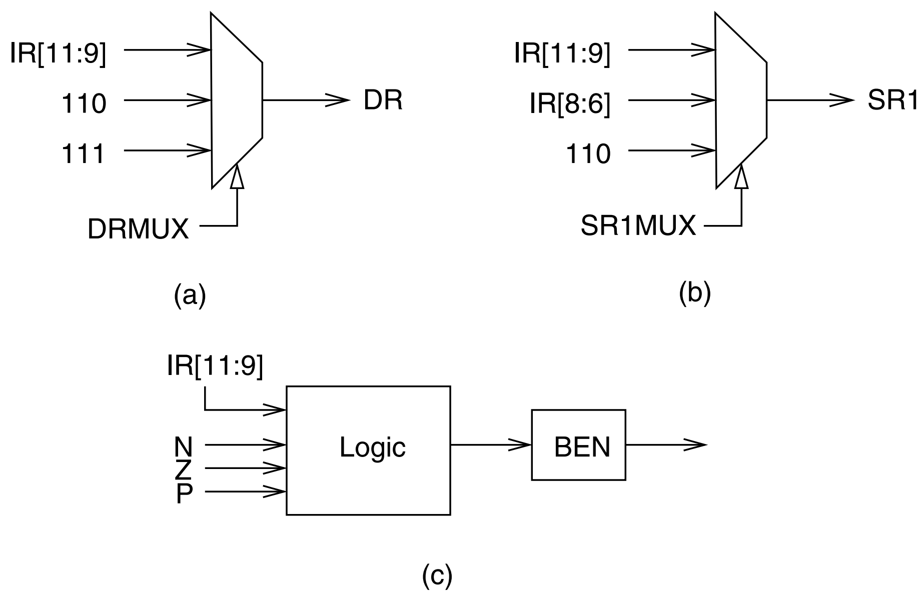
Name:____

```
; Interrupt Service Routine
       .ORIG x1550
       NOT   R0, R0
       RTI
       .END

; User Program
       .ORIG x3000
       AND   R0, R0, #0       ; Line 1
       LD    R1, TIME         ; Line 2
       LD    R2, TIME         ; Line 3

NEXT   


       BRn   P2_DEC           ; Line 6

       ADD   R1, R1, #-1      ; Line 7


       LEA   R0, P2WINS       ; Line 9
       BRnzp END              ; Line 10

P2_DEC ADD   R2, R2, #-1      ; Line 11


       LEA   R0, P1WINS       ; Line 13

END    PUTS                   ; Line 14
       HALT                   ; Line 15

COUNT  LD    R3, SECOND       ; Line 16
LOOP   ADD   R3, R3, #-1      ; Line 17
       BRp   LOOP             ; Line 18


TIME   .FILL    #300
SECOND .FILL    #25000     ; 1 second
P1WINS .STRINGZ "Player 1 Wins."
P2WINS .STRINGZ "Player 2 Wins."
       .END
```



18
MAR <− PC
PC <− PC + 1
[INT]
1
To 49
(See figure C.7)
0
33
MDR <− M
R
R
35
IR <− MDR
32
BEN<−IR[11] & N + IR[10] & Z + IR[9] & P
[IR[15:12]]
RTI
To 8
(See figure C.7)
1101
To 13
ADD
AND
NOT
TRAP
LEA
LD
LDR
LDI
STI
STR
ST
JSR
JMP
BR
0
[BEN]
0
1
22
PC<−PC+off9
To 18
12
PC<−BaseR
To 18
4
[IR[11]]
1
0
21
R7<−PC
PC<−PC+off11
To 18
20
R7<−PC
PC<−BaseR
To 18
1
DR<−SR1+OP2*
set CC
To 18
5
DR<−SR1&OP2*
set CC
To 18
9
DR<−NOT(SR)
set CC
To 18
15
MAR<−ZEXT[IR[7:0]]
28
MDR<−M[MAR]
R7<−PC
R
R
30
PC<−MDR
To 18
14
DR<−PC+off9
set CC
To 18
10
MAR<−PC+off9
24
MDR<−M[MAR]
R
R
11
MAR<−PC+off9
29
MDR<−M[MAR]
R
R
6
MAR<−B+off6
7
MAR<−B+off6
2
MAR<−PC+off9
26
MAR<−MDR
31
MAR<−MDR
3
MAR<−PC+off9
25
MDR<−M[MAR]
R
R
27
DR<−MDR
set CC
To 18
23
MDR<−SR
16
M[MAR]<−MDR
R
R
To 18
NOTES
B+off6 : Base + SEXT[offset6]
PC+off9 : PC + SEXT{offset9]
PC+off11 : PC + SEXT[offset11]
*OP2 may be SR2 or SEXT[imm5]


GateMARMUX
GatePC
MARMUX
LD.PC
PC
+1
16
REG
FILE
DR
3
LD.REG
SR2
SR1
SR2
OUT
SR1
OUT
PCMUX
2
ZEXT
[7:0]
ADDR2MUX
ADDR1MUX
[10:0]
SEXT
[8:0]
[5:0]
0
[4:0]
SR2MUX
CONTROL
R
ALUK
B
A
ALU
IR
LD.IR
N
Z
P
LD.CC
LOGIC
GateALU
GateMDR
MDR
LD.MDR
MAR
LD.MAR
MIO.EN
R.W
INPUT
OUTPUT
KBDR
DDR
KBSR
DSR
ADDR. CTL.
LOGIC
MEMORY
MEM.EN
INMUX


IR[11:9]
110
111
DR
DRMUX
(a)
IR[11:9]
IR[8:6]
110
SR1
SR1MUX
(b)
IR[11:9]
N
Z
P
Logic
BEN
(c)