

NAME:

SECTION:

USERNAME:

CS 2011
Exam 2
D-Term 2006

Question 1: ----- (15)
Question 2: ----- (15)
Question 3: ----- (20)
Question 4: ----- (20)
Question 5: ----- (10)
Question 6: ----- (10)
Question 7: ----- (10)
TOTAL: ----- (100)

Fill in your name, section, and username. DO NOT OPEN THIS TEST UNTIL YOU ARE TOLD TO DO SO.

1. (15 points) An LC-3 assembly language programmer is trying to write code to accomplish this selection statement:

```
IF ( (R3 != R4) OR (R7 < 9) )  
  
    R0 = 5  
  
ELSE  
  
    R0 = 6  
  
ENDIF
```

Here is the programmer's code, but three instructions are missing. Fill in the missing instructions.

```
test1      not    r4, r4  
           add    r4, r4, #1  
           add    r3, r3, r4  
                                     ; fill in first missing instruction  
  
test2      add    r7, r7, #-9  
                                     ; fill in second missing instruction  
  
labelIf    and    r0, r0, #0  
           add    r0, r0, #5  
                                     ; fill in third missing instruction  
  
labelElse  and    r0, r0, #0  
           add    r0, r0, #6  
labelEnd   .  
           .  
           .
```

2. (15 points) Circle the *best* answer for the following multiple choice questions:

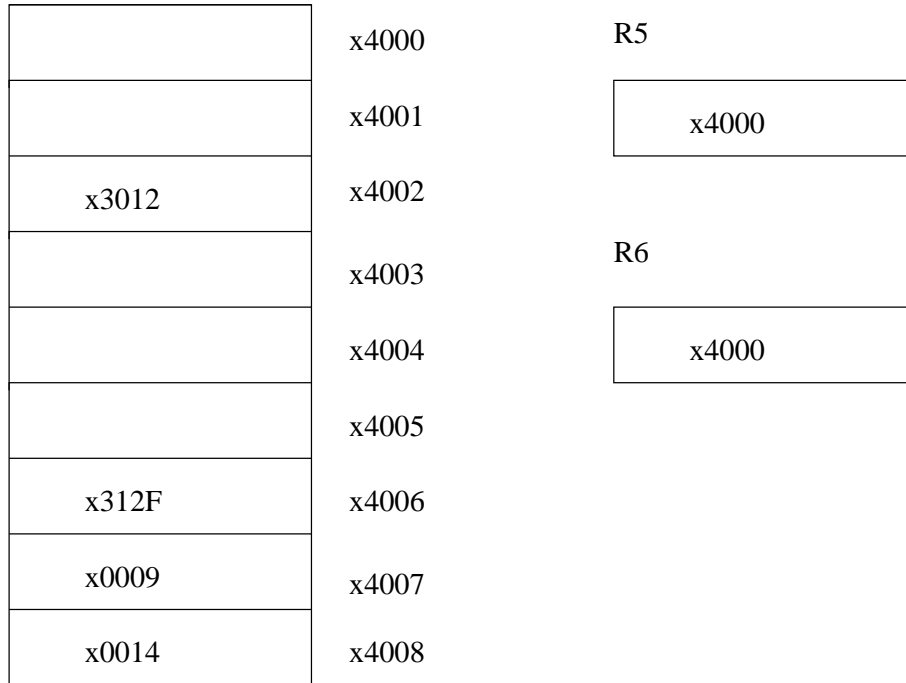
- (a) Interrupt-driven I/O is more efficient than polling because:
- i. Polling consists of interrupt-driven I/O plus additional overhead.
 - ii. Interrupts to the processor take less time than polling loops.
 - iii. The statement is wrong. Polling is more efficient than interrupt-driven I/O.
 - iv. The processor can perform other tasks instead of constantly being in a loop checking to see if a device's status bit has changed.
- (b) If the input service routine reads KBDR without checking the ready bit of KBSR, the following can happen:
- i. Everything will work correctly.
 - ii. The program will drop some characters.
 - iii. No characters will be entered.
 - iv. The program could read the same key multiple times.
- (c) If R3 contains the value x5000, and the instruction **JSRR R3** is stored in location x4000, the value of the PC after the execution of the JSRR instruction is:
- i. x4000
 - ii. x4001
 - iii. x5000
 - iv. x5001

3. (20 points) A C program that calls a function max is compiled into LC-3 machine code. Here is the C program:

```
int main()
{
    int x, y, z;           /* main's local variables */
    scanf ("%d%d", &x, &y); /* reads integer values into x and y */
    z = max (x, y);
    return 0;
}

int max (int a, int b)
{
    int c;                /* max's local variable */
    if (a > b)
        c = a;
    else
        c = b;
    return c;
}
```

`main`'s local variables have been allocated space on the stack beginning at location `x4008`. Here is a picture of the stack and the values of registers `R5` and `R6` just before the instruction `return c` is executed. Some of the memory locations on the stack have been left blank.



- (a) Fill in the contents of each of the blank locations on the stack (specify each value as a 4-digit hexadecimal number).
- (b) Some of the words on the stack were written by the compiled code for the calling function, and some of the words on the stack were written by the compiled code for the called function. Give the address of each location of the stack that was written by the code compiled for the function `max`.

For this question, assume the LC-3 supports the assembly of modules in separate files, as discussed in class.

4. (20 points) Neither the LC-3 assembler nor the 8086 assembler allows a PC-relative reference to an externally-defined label.

(a) Why is this not allowed?

(b) Write a short piece of LC-3 code that shows how to correctly call a subroutine whose entry point is externally defined. Assume the entry point is at label `myFunction`.

For the following two questions, assume the interrupt mechanism for the LC-3 works as described in class and in Chapters 8 and 10.

5. (10 points) Explain *what* the processor is doing in states 43, 47, and 48, and *why* (see Figure C.7).

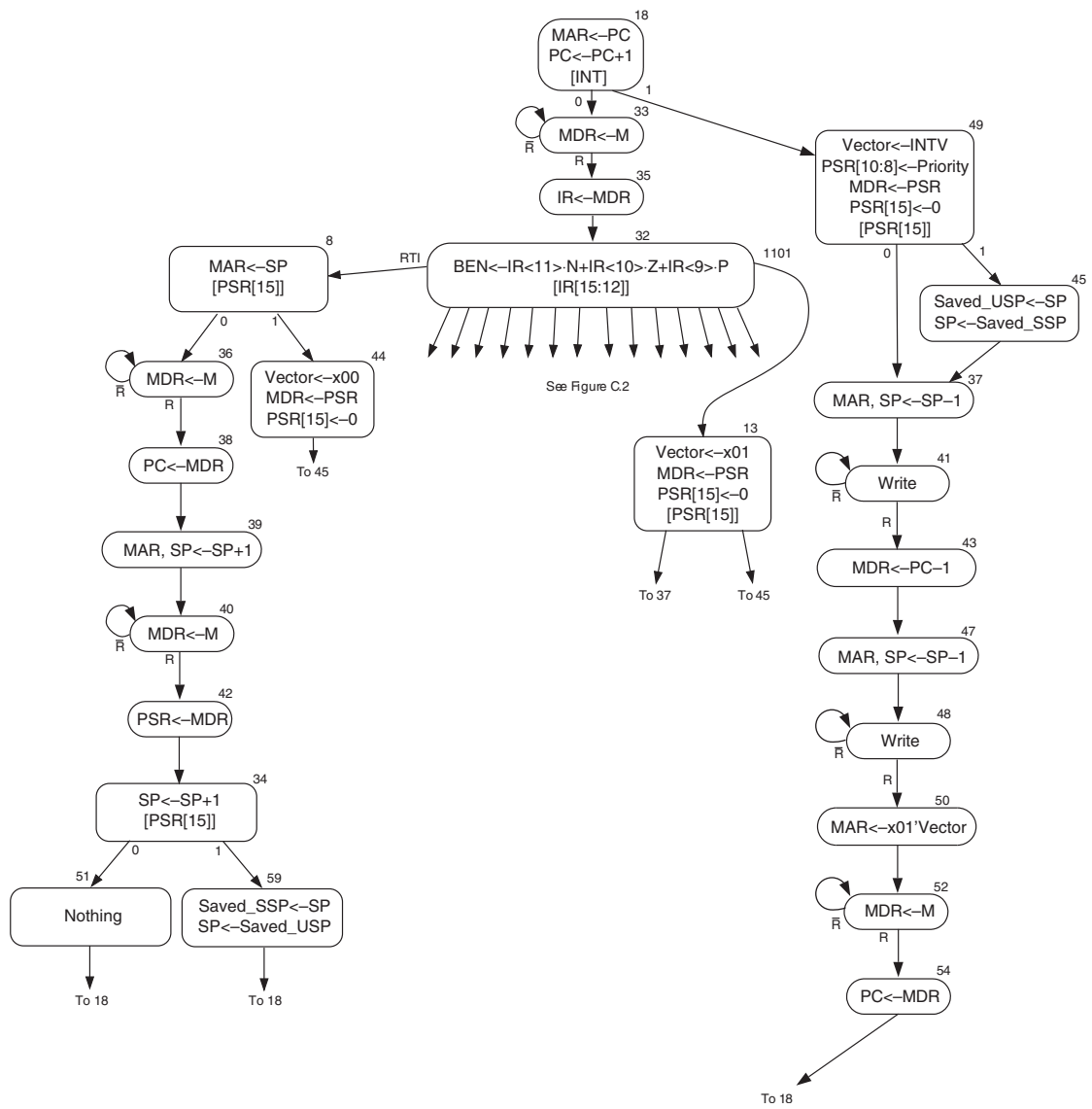
6. (10 points) An LC-3 program running in user space executes the following code:

```
        ...
        JSR    increment
        ...
increment .fill xD000
        LD     R0, increment ; pick up number to increment
        ADD   R0, R0, #1     ; increment it
        ST    R0, increment ; store incremented value
        RET
```

Assume the processor has just finished executing the JSR instruction. List the number of each state in the finite state machine that will execute during the next instruction cycle. (Use Figures C.2 and C.7 to find the numbers of the states.)

7. (10 points) Explain what happens when the 8086 processor executes the instruction instruction

`CMP CX, DX`



	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADD ⁺	0001			DR			SR1			0	00		SR2			
ADD ⁺	0001			DR			SR1			1	imm5					
AND ⁺	0101			DR			SR1			0	00		SR2			
AND ⁺	0101			DR			SR1			1	imm5					
BR	0000			n	z	p	PCoffset9									
JMP	1100			000			BaseR			000000						
JSR	0100			1	PCoffset11											
JSRR	0100			0	00		BaseR			000000						
LD ⁺	0010			DR			PCoffset9									
LDI ⁺	1010			DR			PCoffset9									
LDR ⁺	0110			DR			BaseR			offset6						
LEA ⁺	1110			DR			PCoffset9									
NOT ⁺	1001			DR			SR			111111						
RET	1100			000			111			000000						
RTI	1000			000000000000												
ST	0011			SR			PCoffset9									
STI	1011			SR			PCoffset9									
STR	0111			SR			BaseR			offset6						
TRAP	1111			0000			trapvect8									
reserved	1101															