CS/ ECE 252 Introduction to Computer Engineering

Homework 6 – Due at Lecture on Monday , April 6th

Instructions: You should do this homework in a group of **TWO** or **THREE** students from the SAME 252 section. You should hand in **ONE** copy of the homework. Front page of the answer sheet should contain

- **Name** and **UW ID** of the students in that group
- **Section number** (Lec 001 or Lec 002)
- Multiple pages should be **stapled**.

Warning: Most home works will use questions from your textbook, Patt and Patel's *Introduction to Computing Systems*, which we abbreviate *(ItCS)*

First contact for questions is TA Maheswaran Venkatachalam (kvmakes@cs.wisc.edu)

Problems 1, 2, 3 are programming exercises. This is how you need to solve it:

- Open the LC3 editor.
- Write the binary code for each instruction in your program, one instruction per line. The first line should be "0011000000000000", which is the binary representation of x3000 which is the starting address.

- Remember that the last instruction in your program should be HALT.
- Press the button on the menu which says 'B'. This will generate the binary object file (extension 'obj'). Save this file somewhere.
- Open the LC3 Simulator.
- Open the obj file created using 'File'->'Load Program'.
- Double click on the small grey square at the beginning of the line corresponding to the HALT instruction. You should see a red circle there.
- Press the  button. This will execute your program and stop at the breakpoint.
- Take a screenshot. You can use the 'Print Screen' button on your keyboard to do that. You can use an image editor like MS Paint to save it into a file.

Your solution must precisely adhere to the format specified below:

Your solution consists of two parts: a screenshot of the simulator and some written information.

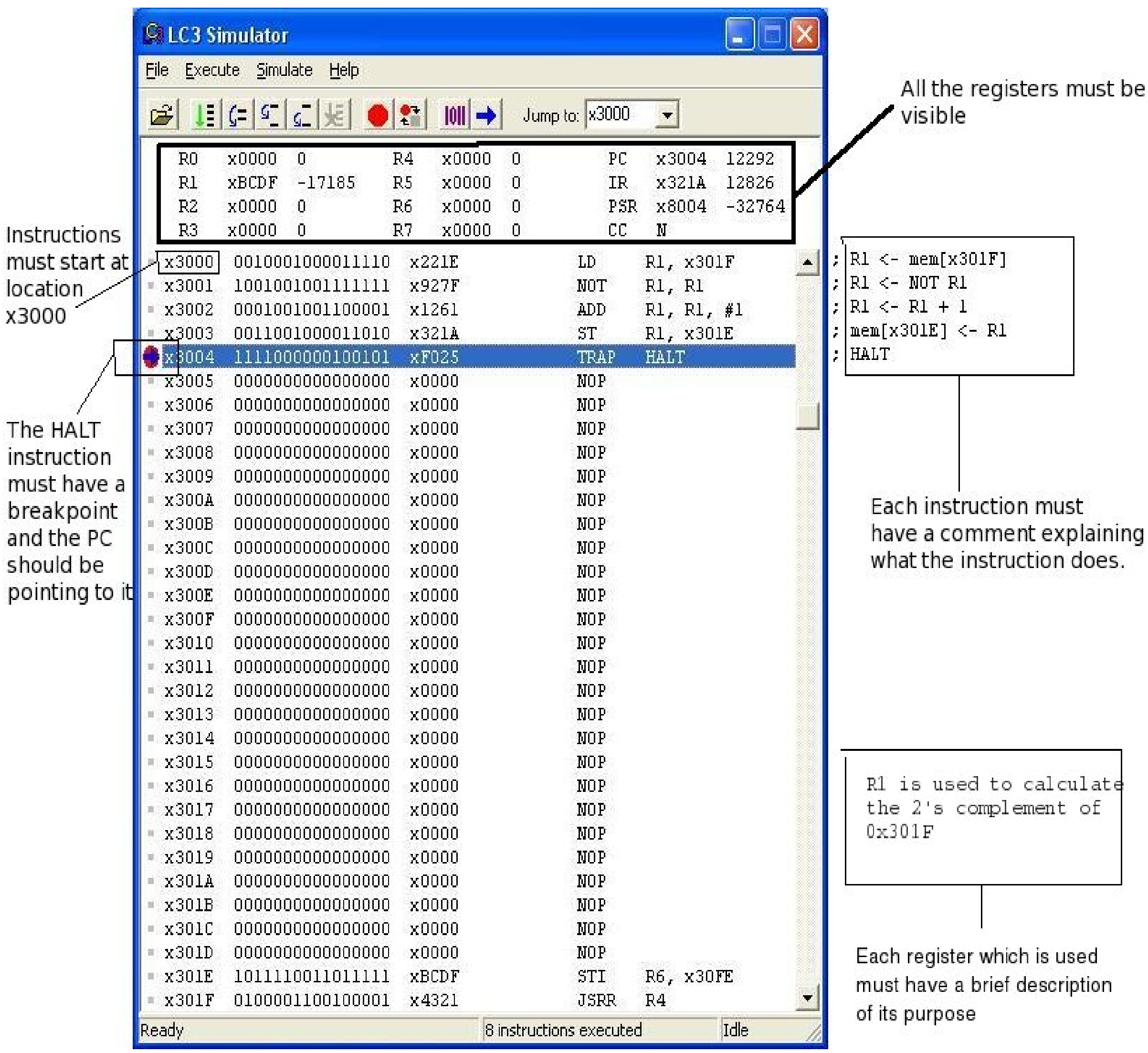
- Your screenshot must have all the following:
  - The instructions should start at x3000.
  - All the registers must be visible.
  - For questions 2,3 the memory locations which are referenced must be visible.
  - You should have a breakpoint (red circle) at your HALT instruction and the PC (blue arrow) should be pointing to that instruction when you take your screenshot.

- Along with this you must also write the following:
    - For each instruction, you need to write a small comment on what it does. You can just annotate the screenshot.
    - You need to specify what registers are used by the program and what it is used for.

- A sample programming exercise and solution is given below.

- **Problem 0**

Write an LC-3 program that finds the two's complement of a number stored in memory and stores the result back in memory. The number is stored in memory location x301F. Store the result in x301E. For the screenshot, make the value in memory location x301F be x4321.



Problem 1

Write a LC3 program which checks for the equality of two numbers in R3 and R4. If the numbers are equal, register R1 must contain the value 1. Else, it must contain the value 0. For the screenshot, make R3 = x0045 and R4 = x0045

Problem 2

Write a LC3 program which compares two numbers in memory locations 0x301D and 0x301E and puts the smaller number in memory location 0x301F.For the screenshot, make the value in memory location x301D to be x0027 and the value in memory location x301E to be x0036. Your screenshot should show memory location x301F.

Problem 3

Write a LC3 program which multiplies two numbers in memory locations 0x301C and 0x301D and puts the result in memory location 0x301E.For the screenshot, make the value in memory location x301C to be x0012 and the value in memory location x301D to be x0020. Your screenshot should show memory location x301E.

Problem 4

Assume R1 has the value 0 initially. R4 has the value 2. Describe how register R1 changes after the execution of each of the following programs and what is its value in it when the program terminates.

(i) Program 1

0001001001101010

0001100100111111

0000001111111101

1111000000100101

(ii) Program 2

0001001001101010

0001100100111111

0000101111111101

1111000000100101

(iii) Program 3

0001001001101010

0001100100111111

0000011111111101

1111000000100101

(iv) Program 4

0001001001101010

0001100100111111

0000111111111101

1111000000100101

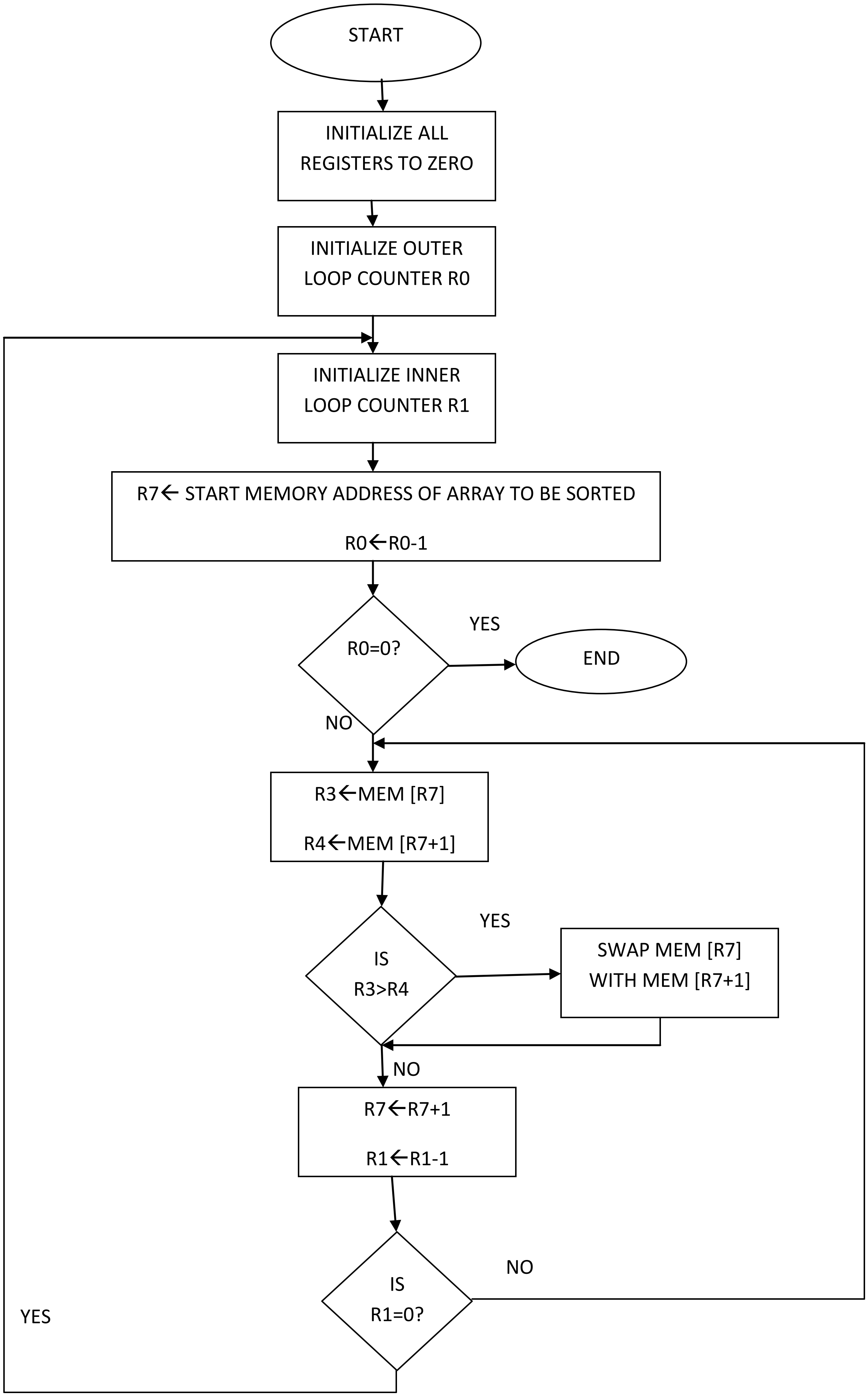
Problem 5

i. Do all operate instructions require two source operands? If not, identify which instruction(s) require(s) only one source operand.

ii. Consider the instructions AND, LD, STI. Identify whether the instructions are operate instructions, data movement instructions or control instructions. For each instruction list the addressing mode(s) that they use

Problem 6

Given below is the flowchart for bubble sort algorithm. (Ascending order)


START
INITIALIZE ALL REGISTERS TO ZERO
INITIALIZE OUTER LOOP COUNTER R0
INITIALIZE INNER LOOP COUNTER R1
R7← START MEMORY ADDRESS OF ARRAY TO BE SORTED
R0←R0-1
R0=0?
YES
END
NO
R3←MEM [R7]
R4←MEM [R7+1]
IS R3>R4
YES
SWAP MEM [R7] WITH MEM [R7+1]
NO
R7←R7+1
R1←R1-1
IS R1=0?
NO
YES

The assembly code given below contains 4 bugs.

Debug the following program so that it sorts N elements in ascending order. The debugged code should follow the above flowchart. For the screenshot, load the debugged code and make the memory locations from x3000 to x301F visible.

Code (with 4 bugs):

```
        .orig x3000
        AND R0,R0,#0
        AND R1,R1,#0
        AND R2,R2,#0
        AND R3,R3,#0
        AND R4,R4,#0
        AND R5,R5,#0
        AND R6,R6,#0
        AND R7,R7,#0

        ADD R0,R0,#9

O_loop  AND R1,R1,#0
        ADD R1,R1,#9
        LD R7,val
        ADD R0,R0,#-1
        BRZ end

I_loop  LDR R3,R7,#0
        LDR R4,R7,#1
        ADD R6,R4,#0
        NOT R6,R6
        ADD R6,R6,#1
        ADD R5,R3,R6
```

```
            BRN swap
            ADD R7,R7,#1
            ADD R1,R1,#-1
            BRZ I_loop
            BRP O_loop


swap        STR R3,R7,#1
            STR R4,R7,#0
            ADD R7,R7,#1
            ADD R1,R1,#-1
            BRZ O_loop
            BRP I_loop

end         HALT
val         .fill x3030
            .end
```

The starting address for the array is set as x3030. This program sorts 9 elements. (Determined by the value of R0 and R1)

Given below is the assembly code for storing the elements in the memory location starting from x3030. There are 10 elements. The debugged code should sort only the first 9 elements.

```
            .orig x3030
            .fill x0023
            .fill x0011
            .fill x0043
            .fill x0054
            .fill x0012
            .fill x0001
```

```
.fill x0087
.fill x0096
.fill x0043
.fill x9999
.end
```

Create object files for both programs and load them into the LC3 simulator.