ECE/CS 252 Fall 2011 Homework 7 (25 points) // Due in Lecture on Nov. 30, 2011 (Wed)

Instructions:

1. Do this homework in groups. Individual homework will be penalized. Hand in one copy per group.

2. You should do this homework with your assigned groups. Write all the names and WiscIDs of the group members.

3. Staple all the answer sheets together to receive full credit.

4. Problem 1 should be submitted in lecture on Nov 30 and Problem 2 should be submitted in Dropbox by Nov 30 (11.59 pm). Only one person from each group should upload the file.

5. First point of contact for clarification on the questions: Ripudaman Singh (rsingh27@wisc.edu). Contact any TA if you need help on how to get started on a problem.

Problem 1. (2 + 3 + 2 points)

Here is a small program to add numbers starting at address labeled as DATA and the number of numbers to be added is given at address labeled as NUM. The program stores the result at location labeled as RESULT.

Output expected (in memory location RESULT): 6

```
            .ORIG x3000
            AND R0, R0, #0
            LD  R3, NUM
            LEA R1, DATA
            LDR R2, R1, #0
LOOP:       BRz DONE
            ADD R0, R2, R0
            ADD R1, R1, #1
            LDR R2, R1, #0
            ADD R3, R3, #-1
            BR LOOP
RESULT:     .FILL x0000
NUM:        .FILL x0004
DATA:       .FILL x0000
            .FILL x0001
            .FILL x0002
            .FILL x0003
DONE:       ST R0, RESULT
            HALT
            .END
```

(a) Write the symbol table created by the assembler on the first pass of the above program.

(b) Once the symbol table is created, the assembler then creates a binary version (.obj) of the program. Convert the above program into machine code.

(c) The code doesn't function as it should. Your job is to debug and fix it. Please explain why it does not work and what needs to be done in order to make it work as expected. You don't need to turn-in full corrected program.

Problem 2. (5+8+5 points)

Building up on work completed in HW6, we will display a full string in this homework and then also wrap it around when all the characters have gone past the left end.
In this homework your task is to:

1. Write a subroutine to get starting address of bitmap for a given character
2. Use the above subroutine to display a string given to you (only a few characters will show up on screen at a time).
3. Implement circular scrolling i.e. once the characters have gone out from the left side, they should start appearing from the right. Circular Scrolling should be done 5 times.

The details of tasks are:

1. For first task, you have to write a subroutine CALC_BITMAP_ADDRESS that takes in ASCII value of a character (in R1) and the address from where the bitmap starts (in R0). The subroutine should return the starting address of bitmap corresponding to the character given as input in R1. Output register is R0 as well i.e. return value should be contained in R0. You should save and restore the registers before using but not all (why?). This is similar to what was done in Applied Exercise 4.

   Bitmaps of the characters have similar format as in previous homework. This time you are given bitmaps of all alphabets in capital letters i.e. A-Z (ASCII 65-90) in hw7_bitmap.asm. Again, the bitmaps start from address 0x5000 (available in BITMAP_START memory location). You don't need to make any change to this file for this homework. Remember that bitmap of each character takes 49 entries (available in BITMAP_INCR memory location).

2. For second task, you are given a string starting at address 0x4E00. This address is stored in label STRING in hw7.asm. The String is given in hw7_string.asm. This has been given for your ease and testing of your program. This string will be replaced by some random string while grading and hence, you should not hard-code the characters.

   The string will contain characters A-Z (ASCII 65-90) and space (ASCII 32) and you can assume that no characters other than mentioned above will be entered. Further assume that there will be at least one character in the string. The length of string may vary but will be null-terminated. Therefore 0x0 or null character will determine when your string ends.

   You are also provided with DRAW_CHAR subroutine that implements the logic for displaying a character that you implemented in previous homework. This subroutine takes in x-coordinate (in R0), y-coordinate (in R1) and starting address of the bitmap of character (in R2) that is to be displayed. When the subroutine returns, all the registers will have same value as prior to calling of the subroutine (You can confirm this by looking at the part of

code where values in registers are stored to memory locations at the start of subroutine and loaded back to registers before returning). There is no output value of this subroutine.

This task can be broken down in small parts. You should read one character from the string, calculate the bitmap address using CALC_BITMAP_ADDRESS subroutine and then use DRAW_CHAR subroutine for displaying the character if it is not space and repeat the process till you encounter 0x0 or null character. The STRING_INDEX memory location tracks the number of characters already drawn. The flowchart later will explain this better. For simplicity, you have to iterate through all the characters in string and not worry about how many characters are being displayed on screen at any given moment.

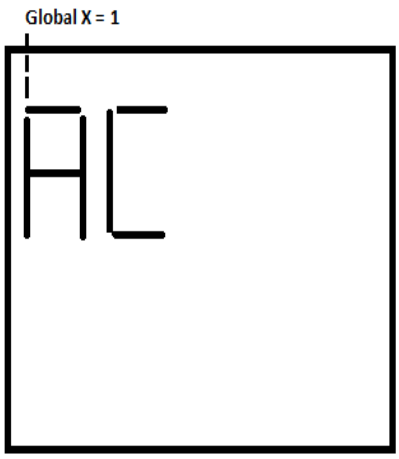
This task requires you to fill in code in STRING_LOOP. The coordinates are tracked in memory locations LOCAL_X and LOCAL_Y which represent the starting (x,y) coordinate of the current character in the string. You can use these values to set up arguments before calling the respective subroutines.

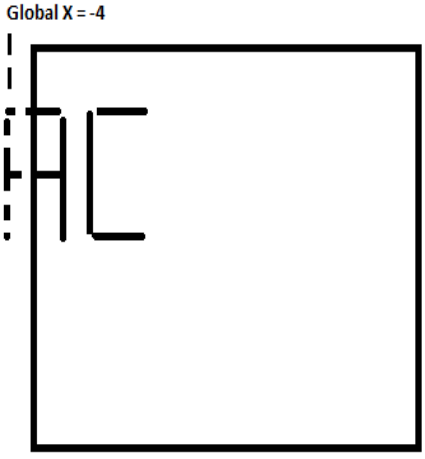
3. Now once you have drawn the string, you need to implement circular scrolling. Since the display is quite small, we can have only 4 characters being displayed at any given time. Hence, scrolling is necessary to read the full string. Once the whole string is displayed, the characters should wrap and should start appearing at the right. Here, you need to come up with the condition for wrap. This task requires you to complete SCROLL_LOOP.

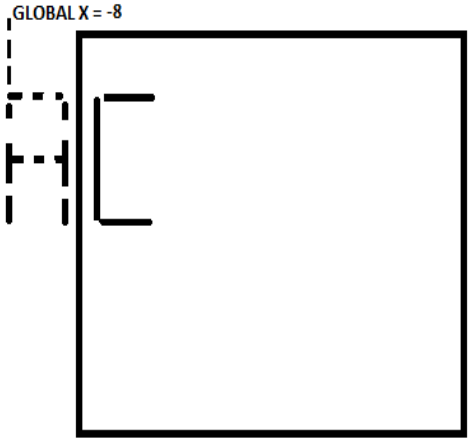
   As in previous homework, you need to make use of global X-coordinate (being tracked in GLOBAL_X memory location) that determines the start of the string for any particular iteration of Scroll loop. This coordinate is decremented by 1 after each iteration of the scroll loop. For displaying the characters from right, global X-coordinate should be reset to maximum X-coordinate (i.e. 31) (available in X_MAX memory location) and then again it should be decremented by 1 till it's the time to wrap again. The condition for wrap essentially is the answer to question: when should global X be set to 31? Following figure shows the use of global X coordinate when 2 characters A and C were being displayed. Dotted character won't be displayed since they are out-of-range of the display region. Note that wrap happens after all the characters in string have moved out of display region.

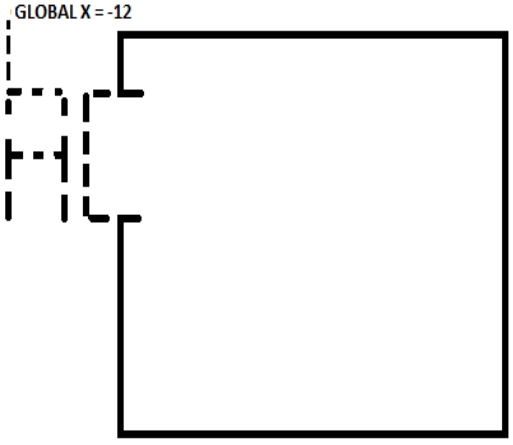
   The scroll loop should end after 5 such wraps. Memory location labeled WRAP_COUNTER tracks the number of wraps being done. For ease of understanding the intent and flow of program, flow chart is given on next page.

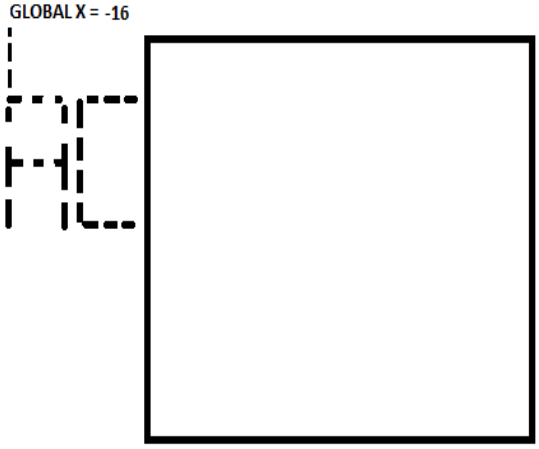
   Remember to load lc3os_mod, hw7_bitmap, hw7_string programs into LC3 before you test your program.

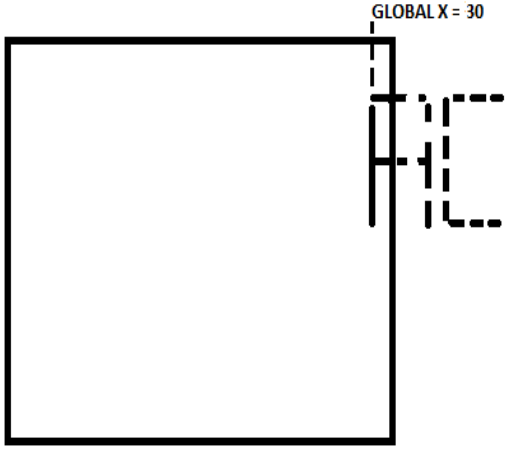

Global X = 1

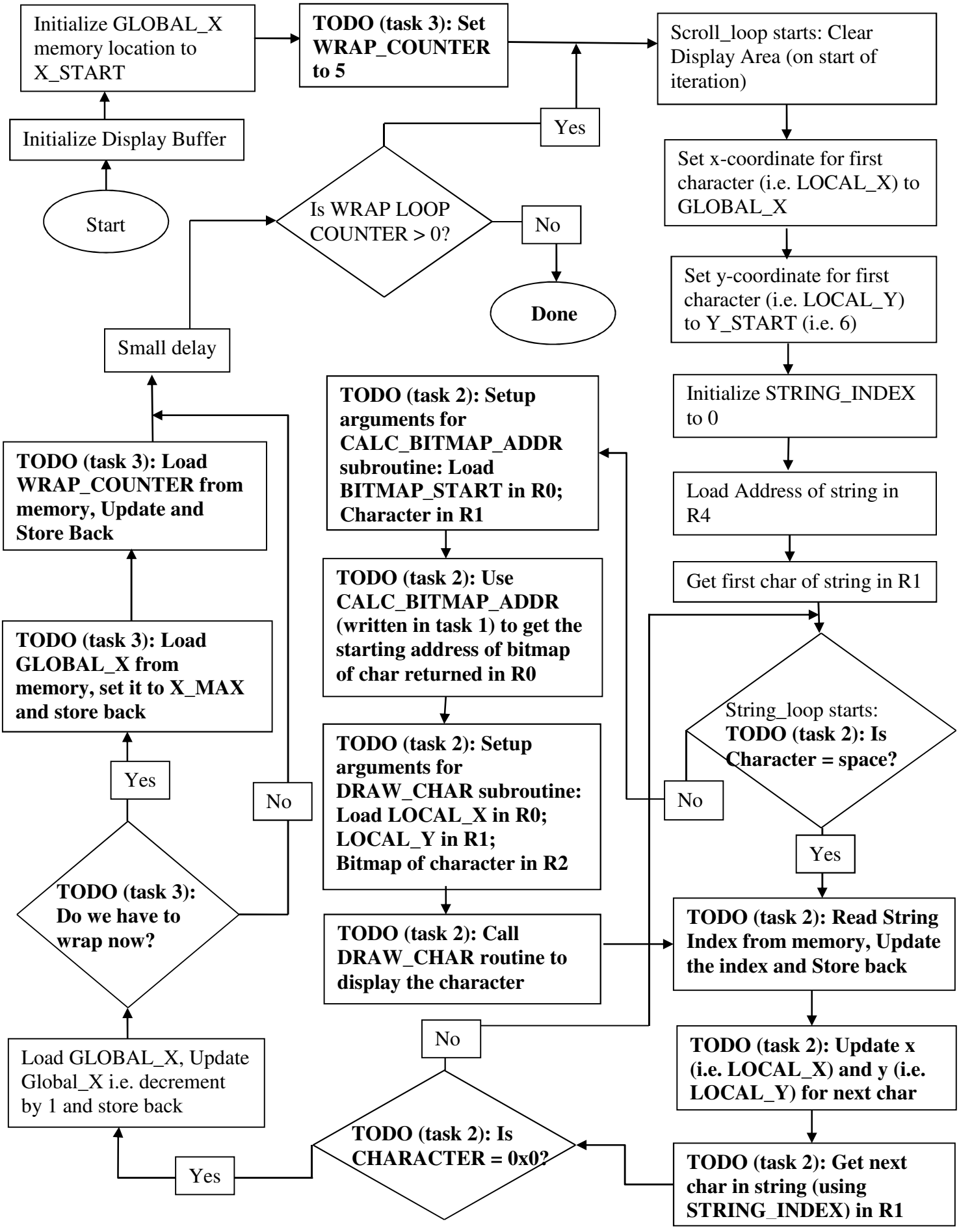

Global X = -4


GLOBAL X = -8


GLOBAL X = -12


GLOBAL X = -16


GLOBAL X = 30


Initialize GLOBAL_X memory location to X_START
TODO (task 3): Set WRAP_COUNTER to 5
Scroll_loop starts: Clear Display Area (on start of iteration)
Initialize Display Buffer
Yes
Set x-coordinate for first character (i.e. LOCAL_X) to GLOBAL_X
Start
Is WRAP LOOP COUNTER > 0?
No
Set y-coordinate for first character (i.e. LOCAL_Y) to Y_START (i.e. 6)
Done
Small delay
Initialize STRING_INDEX to 0
TODO (task 2): Setup arguments for CALC_BITMAP_ADDR subroutine: Load BITMAP_START in R0; Character in R1
TODO (task 3): Load WRAP_COUNTER from memory, Update and Store Back
Load Address of string in R4
TODO (task 2): Use CALC_BITMAP_ADDR (written in task 1) to get the starting address of bitmap of char returned in R0
Get first char of string in R1
TODO (task 3): Load GLOBAL_X from memory, set it to X_MAX and store back
String_loop starts: TODO (task 2): Is Character = space?
Yes
TODO (task 2): Setup arguments for DRAW_CHAR subroutine: Load LOCAL_X in R0; LOCAL_Y in R1; Bitmap of character in R2
No
No
Yes
TODO (task 3): Do we have to wrap now?
TODO (task 2): Call DRAW_CHAR routine to display the character
TODO (task 2): Read String Index from memory, Update the index and Store back
No
TODO (task 2): Update x (i.e. LOCAL_X) and y (i.e. LOCAL_Y) for next char
Load GLOBAL_X, Update Global_X i.e. decrement by 1 and store back
TODO (task 2): Is CHARACTER = 0x0?
Yes
TODO (task 2): Get next char in string (using STRING_INDEX) in R1