Homework – Get 1 Number

This is a programming assignment that will be graded and scored as a home work since it is so simple. This one is less simple than previous assignments

Write a subroutine for the LC3 that will stop and wait for the user to enter a single digit number from 0 to 9. The subroutine should function in a way like GETC but should not return until a digit is pressed. If anything other than a numeric character is pressed the program should simply ignore the character.

A further detailed description of the program operation is given below.

This program should be named ***hwGet1Num.asm.*** The file should be a proper assembly file for the LC3 processor.

USE THE ASSEMBLER for this assignment. Do not manually convert to hex. Write your assembly file and convert it to hex for testing. **UPLOAD YOUR ASM FILE after you have it working in the simulator.**

The "MAIN" part of your program (the top lines) should look like this.

```
        .orig x3000
MAIN
        JSR  GETNUM
        JSR  GETNUM  ;Yes, there are TWO calls to GETNUM
        HALT
```

After the halt should come your subroutine which should be named ***GETNUM*** and does the following

> Get a single character from the user. Use GETC. Note that GETC is a system subroutine and will affect R7. That is, if you are in a subroutine and you call GETC, it will save its return address in R7 and wipe out your subroutine's return address.
>
> If the character typed is a digit, place the value of the character in R0 and return. That is if, the character is 0 – 9 then R0 should contain the value 0-9 and NOT the ASCII code.
>
> If any other character is typed, the routine should allow the user to enter a new character (i.e. loop back to GETC.)
>
> The call to the subroutine should not affect any Registers other than R0.
>
> If the user typed ***a*** then ***b*** then ***X*** then ***8*** then R0 should contain 8.

Submit your ***hwGet1Num.asm*** file to Web-CAT when done. Make sure to test your program with characters OTHER than 0-9. Make especially sure to test with '/' (ASCII 47) and ':' (ASCII 58).

NOTE: You will lose 1 point for every submission over 5. You will lose 10 points for each day late.

Submitting more than 2 days late will result in a zero.

Note that late days start at 9:00 am. So, submitting after 9:00 am on the due date will result in -10 points. Submitting after 9:00 am on the day after that will result in an additional -10 points. You will not be able to submit after 9:00 am on the day after that.

Explanation of the operation of the program.

GETNUM waits for a digit to be pressed then returns the value of that digit in R0. So, if a 5 is pressed, the number 5 should be stored in R0 after GETNUM returns. NOTE that this is the number 5 and not the ASCII code for 5. You will have to convert since GETC returns ASCII codes.

Since you are running GETNUM twice from main, it means you will have to type TWO digits for the program to finish. Only the second digit will be stored in R0 after the program ends since the second call to GETNUM will overwrite the value of the first call.

ALL OTHER INPUTS SHOULD BE TREATED AS IF THEY DIDN'T HAPPEN.

Some examples:

If you run the program and type wxyz56, the first call to GETNUM should ignore w, x, y, and z and return with 5 in R0. Then GETNUM will be called again and will return 6 in R0. So after your program runs, if you type wxyz56, the number 6 should be in R0 and NOT decimal 54 or the hex value 36.

If you run the program again and type abce!!:*7ww}}{{8, the first call to GETNUM will ignore everything up to the 7 and return with 7 in R0. Then GETNUM will be called again and will ignore the ww}}{{ and return with the number 8 in R0.

How to save and restore registers. This example shows how to save and restore R1, R2, and R7. The exact registers YOU need to save will depend on your program.

```
GETNUM
        ; Save R1, R2, and R7.
        ST R1, GN_SR1
        ST R2, GN_SR2
        ST R7, GN_SR7


        …       do some code
        …       do some code


        ;Restore R1 and R2
        LD R1, GN_SR1
        LD R2, GN_SR2
        LD R7, GN_SR7

        RET

; Create storage locations GN_SR1, GN_SR2, and GN_SR3
; Do this immediately after the RET for all subroutines
GN_SR1          .fill 0
GN_SR2          .fill 0
GN_SR7          .fill 0
```

Notes about saving and restoring.

1. Save registers that you use in your subroutine.
2. Save register R7 if you call another subroutine or a trap inside your subroutine.
3. DO NOT save registers used for returning results (R0 in GETNUM).
4. Labels must be unique. SR1 means Save R1. You can't reuse SR1 in a different subroutine. So, I use GN_SR1 to mean GETNUM's SR1. In MULT, I would use M_SR1.
5. Always put the storage for registers IMMEDIATELY following the RET for the subroutine they are used in. Don't put ALL the save locations at the end of the program. The PCOFFSET will be too large if you have many subroutines.