CS061 – Programming Assignment 08

Objective

The purpose of this final programming assignment is to test the limits of your LC3 skill by giving you a challenging implementation problem straight out of the textbook.

High Level Description

There is a server somewhere, called the Busyness Server. This server tracks whether 16 different machines connected to it are busy (0) or free (1).
You will write a menu-driven system that allows the user to query the availability of the 16 machines in various combinations.

Before You Start Coding

This assignment is based on a variation of Question 9.9 (p. 242 of the textbook (2nd edition)) – which will require that you first work through the following:

- Example 2.11 (p. 37)
- Question 2.36 (p. 46)
- Question 5.6 (p. 146)

Example 2.11

Suppose we have eight machines that we want to monitor with respect to their availability. We can keep track of them with an 8-bit BUSYNESS vector, where a bit is 1 if the unit is free and 0 if the unit is busy. The bits are labeled, from right to left, from 0 to 7.
The BUSYNESS bit vector 11000010 corresponds to the situation where only units 7, 6, and 1 are free, and therefore available for work assignment.

Suppose work is assigned to unit 7. We update our BUSYNESS bit vector by performing the logical AND, where our two sources are the current bit vector 11000010 and the bit mark 01111111. The purpose of the bit mask is to clear bit 7 of the BUSYNESS bit vector. The result is the bit vector 01000010.

Recall that we encountered the concept of bit mask in Example 2.7. Recall that a bit mask enables one to interact some bits of a binary pattern while ignoring the rest. In this case, the bit mask clears bit 7 and leaves unchanged (ignores) bits 6 through 0.

Suppose unit 5 finishes its task and becomes idle. We can update the BUSYNESS bit vector by performing the logical OR of it with the bit mask 00100000. The result is 01100010.

Question 2.36

Refer to Example 2.11 for the following questions.

a. What mask value and what operation would one use to indicate that machine 2 is busy?
b. What mask value and what operation would one use to indicate that machines 2 and 6 are no longer busy? (Note: This can be done with only one operation)
c. What mask value and what operation would one use to indicate that all machines are busy?
d. What mask value and what operation would one use to indicate that all machines are idle (not busy)?
e. Develop a procedure to isolate the status bit of machine 2 as the sign bit. For example, if the BUSYNESS pattern is 01011100, then the output of this procedure is 10000000. If the BUSYNESS pattern is 01110011, then the output is 00000000. In general, if the BUSYNESS pattern is:

b7	b6	b5	b4	b3	b2	b1	b0

the output is

b2	0	0	0	0	0	0	0

Hint: What happens when you ADD a bit pattern to itself?

Question 5.6
Recall the machine busy example from Section 2.7.1. Assuming the BUSYNESS bit vector is stored in R2, we can use the LC3 instruction 0101 001 010 1 0001 (AND R3, R2, #1) to determine whether machine 0 is busy or not. If the result of this instruction is 0, then machine 0 is busy.

a. Write an LC3 instruction that determines whether machine 2 is busy.
b. Write an LC3 instruction that determines whether both machines 2 and 3 are busy.
c. Write an LC3 instruction that indicates whether none of the machines are busy.
d. Can you write an LC3 instruction that determines whether machine 6 is busy? Is there a problem here?

The variation of Question 9.9 that you will implement in this assignment:

a. Check if ***all*** machines are busy; return 1 if all are busy, 0 otherwise.
b. Check if ***all*** machines are free; return 1 if all are free, 0 otherwise.
c. Check ***how many*** machines are busy; return the number of busy machines.
d. Check ***how many*** machines are free; return the number of free machines.
e. Check the ***status*** of a specific machine whose number is passed as an argument in R1; return 1 if that machine is free, 0 if it is busy.
f. Return the ***number*** of the first (lowest numbered) machine that is free. If no machine is free, return -1

Your Tasks The assignment can be broken down into the following tasks:

1. Your main code block should call a MENU subroutine, which prints out a fancy looking menu with numerical options, allows the user to input a choice, and returns in R2 the choice (i.e. 1, 2, 3, 4, 5, 6, 7) that the user made (if the user inputs a non-existent option, the menu must simply repeat until a valid entry is obtained).

   Here's a nice aesthetically pleasing menu. Feel free to steal it :-)

```
**********************
* The Busyness Server *
**********************
1. Check to see whether all machines are busy
2. Check to see whether all machines are free
3. Report the number of busy machines
4. Report the number of free machines
5. Report the status of machine n
6. Report the number of the first available machine
7. Quit
```

   Let's use the BUSYNESS bit vector `0110 1010 0001 0011`
   Outputs should look similar to the following *(but feel free to improve on them!!)*:

```
1 -> Response: "1. All machines are not busy"
2 -> Response: "2. All machines are not free"
3 -> Response: "3. There are 9 busy machines"
4 -> Response: "4. There are 7 free machines"
5 -> Response: "5. Which machine?" 3 -> "Machine 3 is busy"
6 -> Response: "6. The first available machine is number 0"
7 -> Response: "7. Goodbye!"
```

After each response (or after an invalid option), the menu must repeat (except for option 7, of course).
Optionally, output a message of some sort in response to an invalid option before repeating the menu.

2. Write the following subroutines:
    a. MENU
    b. ALL_MACHINES_BUSY
    c. ALL_MACHINES_FREE
    d. NUM_BUSY_MACHINES
    e. NUM_FREE_MACHINES
    f. MACHINE_STATUS
    g. FIRST_FREE

Feel free to totally steal the following headers for your code:

```
;-----------------------------------------------------------------------------------------------------------
; Subroutine: MENU
; Inputs: None
; Postcondition: The subroutine has printed out a menu with numerical options, allowed the
;                user to select an option, and returned the selected option.
; Return Value (R1): The option selected:  #1, #2, #3, #4, #5, #6 or #7
; no other return value is possible
;-----------------------------------------------------------------------------------------------------------
;-----------------------------------------------------------------------------------------------------------
; Subroutine: ALL_MACHINES_BUSY
; Inputs: None
; Postcondition: The subroutine has returned a value indicating whether all machines are busy
; Return value (R2): 1 if all machines are busy,    0 otherwise
;-----------------------------------------------------------------------------------------------------------
;-----------------------------------------------------------------------------------------------------------
; Subroutine: ALL_MACHINES_FREE
; Inputs: None
; Postcondition: The subroutine has returned a value indicating whether all machines are free
; Return value (R2): 1 if all machines are free,    0 otherwise
;-----------------------------------------------------------------------------------------------------------
;-----------------------------------------------------------------------------------------------------------
; Subroutine: NUM_BUSY_MACHINES
; Inputs: None
; Postcondition: The subroutine has returned the number of busy machines.
; Return Value (R2): The number of machines that are busy
;-----------------------------------------------------------------------------------------------------------
;-----------------------------------------------------------------------------------------------------------
; Subroutine: NUM_FREE_MACHINES
; Inputs: None
; Postcondition: The subroutine has returned the number of free machines
; Return Value (R2): The number of machines that are free
;-----------------------------------------------------------------------------------------------------------
;-----------------------------------------------------------------------------------------------------------
; Subroutine: MACHINE_STATUS
; Input (R1): Which machine to check
; Postcondition: The subroutine has returned a value indicating whether the machine indicated
;                by (R1) is busy or not.
; Return Value (R2): 0 if machine (R1) is busy, 1 if it is free
;-----------------------------------------------------------------------------------------------------------
;-----------------------------------------------------------------------------------------------------------
; Subroutine: FIRST_FREE
; Inputs: None
; Postcondition:
; The subroutine has returned a value indicating the lowest numbered free machine
; Return Value (R2): the number of the free machine
;------------------------------------------------------------------------------------------------------------
```

Are you INSANE?!?

Nope! This is actually a pretty simple assignment – but we'll give you some hints anyway

Really, Really Important

- 1 means free; 0 means busy
- Store the BUSYNESS vector at x5000 in the following manner**at the very end of assn8.asm**:

  ```
             .orig x5000
  BUSYNESS   .FILL   xF26B    ; or whatever value you like
  ;(the grader will test different values)
  ```

  Make sure to call the label "BUSYNESS". If your busyness vector cannot be **easily** found, you will lose points.

Hint 01
Remember the algorithm for printing a number out in binary. Remember how to examine each individual bit of a 16-bit value (i.e. any number in a register).

Hint 02
Steal the headers we wrote and use them to help guide your step-by-step completion of the assignment (yay - endorsed micro-plagiarism?)

Hint 03 – subs a & b
Notice that if a machine is free, then it is NOT busy. If a machine is busy, then it is NOT free (emphasis on the keyword "NOT").

Hint 04 – subs d & e
If N = the total number of machines (16),
and X = the total number of free machines,
and Y = the total number of busy machines,
then note that X = 16 – Y and Y = 16 – X

Hint 05
Do not forget to follow **all** of the steps of the subroutine construction process.

Hint 06
Did we mention that **1 means free; 0 means busy** ?

Hint 07
To implement your menu-driven system, implement an infinite loop that works like this:
Feel free to use your own messages):

```
while( true )
{
  choice = menu()                  ; call MENU subroutine

  if (choice == 1)
```

```
    R2 = all_machines_busy()      ; call ALL_MACHINES_BUSY subroutine
    if (R2 == 1)
      Print “All machines are busy”
    else
      Print “All machines are not busy”

  else if (choice == 2)
    [similar]

  else if (choice == 3)
    R2 = num_busy_machines()      ; call NUM_BUSY_MACHINES subroutine
    Print “There are “, (R2), “ machines busy”

  else if (choice == 4)
    [similar]

  else if (choice == 5)
    Print “Which machine do you want the status of (0 - 15)?”
    ; NOTE: you may want to build a “helper” subroutine
    ; to obtain validated input here
    R1 = <validated user entry from #0 to #15>
    R2 = machine_status(R1)       ; call MACHINE_STATUS and pass R1
    if (R2 == 1)
      Print “Machine “, (R1), “ is free”
    else
      Print “Machine “, (R1), “ is busy”

  else if (choice == 6)
    R2 = first_machine_free()     ; call FIRST_Free subroutine
    Print “The first available machine is number”, (R2)

  else if (choice == 7)
    Print “Goodbye!”
    HALT
}
```

Rubric

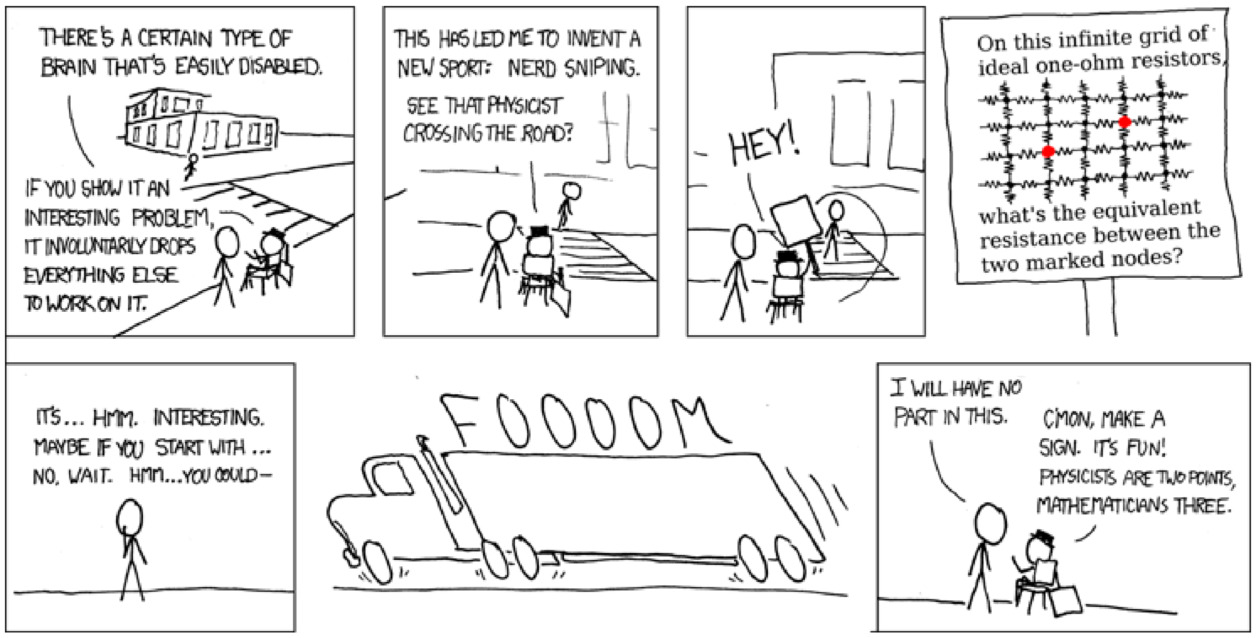
Does anyone even read this part? It’s rather important

- Code does not assemble: (-10 points)
- No header: (-10 points)
- Lacking complete headers on subroutines (-2 points)
- Using 1==BUSY, 0==FREE (-2 points)

- Subroutine MENU works as per specs (+1 point)
- Subroutine ALL_MACHINES_BUSY works as per specs (+2 point)
- Subroutine: ALL_MACHINES_FREE works as per specs (+1 point)
- Subroutine: NUM_BUSY_MACHINES works as per specs (+2 point)
- Subroutine: NUM_ FREE _MACHINES works as per specs (+1 point)
- Subroutine: MACHINE_STATUS works as per specs (+2 point)
- Subroutine: FIRST_FREE works as per specs (+1 point)

Make sure you use appropriate whitespace when inputting/outputting text (e.g. don’t start printing the menu again on the same line as some user input) and the menu is not confusing or worded poorly. You are welcome to steal the example menu given in the specs.

One last XKCD comic for the quarter!


THERE'S A CERTAIN TYPE OF BRAIN THAT'S EASILY DISABLED.
IF YOU SHOW IT AN INTERESTING PROBLEM, IT INVOLUNTARILY DROPS EVERYTHING ELSE TO WORK ON IT.
THIS HAS LED ME TO INVENT A NEW SPORT: NERD SNIPING.
SEE THAT PHYSICIST CROSSING THE ROAD?
HEY!
On this infinite grid of ideal one-ohm resistors, what's the equivalent resistance between the two marked nodes?
IT'S... HMM. INTERESTING. MAYBE IF YOU START WITH ... NO, WAIT. HMM...YOU COULD—
FOOOOM
I WILL HAVE NO PART IN THIS.
C'MON, MAKE A SIGN. IT'S FUN! PHYSICISTS ARE TWO POINTS, MATHEMATICIANS THREE.

Source: http://xkcd.com/356/