ECE109 Program Assignment 3 Help Session

Here's a message from Greg Byrd regarding a little problem with the testing of the assignment.

> I mistakenly included code for a Roll routine at the end of the test_dice.asm program, except that I called it Dice. This means that test_dice.asm does not test your code at all -- it simply tests the code that I included.
>
> It also means that you don't have to write the Roll code, although you do need to create a dice.asm file for your code to work. You may simply copy my Dice code from test_dice.asm (at the bottom) into the dice.asm file, rename it, and add the appropriate .ORIG statement and .END. If you do this successfully (or, of course, if you write your own correct version), you will receive the full 30 points of credit for the Roll subroutine.
>
> Consider it an early Christmas present. Ho ho ho.
>
> ...gb
>
> PS. A new test program, called new_test_dice.asm, will be posted on the ECE 109P web site shortly, in case you really do want to test your code.

This same problem occurred with the copies posted for the distance education sections, so that same "present" will also be given here.

What you must do

First, carefully read the program assignment handout. Pay careful attention to the description of the linear feedback shift register (LFSR) and the rules of the Shut the Box game.

Second, copy the `.asm` files from http://www.cs.unca.edu/ece109/programs.html, the course programs web site.

Third, assemble the following downloaded LC/3 assembler files:

```
seed.asm
setup.asm
test_lfsr.asm
```

Fourth, read **once again** the first two pages of the program assignment, paying special attention to the requirements and steps given on the second page.

Fifth, write the LFSR service routine. Because your service routine doesn't read any device registers it will be indistinguishable from a subroutine. Be sure to `.ORIG` your program to `x1000` because that's where `setup.asm` points.

Store your service routine in `lfsr.asm`. My routine is 47 lines long. Seven of those lines are comments, another nine are concerned with saving and restoring registers, and four are `.FILL`s. So there's about thirty hard-working lines of code to be written.

Sixth, assemble `lfsr.asm` and make sure it assembles correctly.

Seventh, start up the LC/3 simulator and load the following object files in the order given below.

```
setup.obj
lfsr.obj
seed.obj
test_lfsr.obj
```

Actually, the only important thing is to load `test_lfsr.obj` last, so that the simulator will naturally start execution with its code.

Eighth, start the simulator at location `x3000` and make sure your program is bug-free.

Ninth, fix your bugs and repeat steps six to eight until are bugs are exterminated.

Tenth, take a 15 minute break now that you have 60 points of the assignment done.

Eleventh, follow the instructions in Greg Byrd's message. Your new `dice.asm` should .ORIG at `x5000`. Be sure to assemble this program.

Twelfth, copy `boxes_ui.asm` to `boxes.asm`. Within `boxes.asm`, change the line

```
Boxes    .BLKW 10    ; one word per box,
```

to

```
WBoxes   .BLKW 10    ; one word per box,
```

[This may be necessary only for the Linux LC/3 assembler.]

Thirteenth, read pages 5 and 6 (starting with "The Program") carefully once again. Play close attention to the six subroutine for managing the boxes and getting the score.

Fourteenth, open `boxes.asm` and find the place where you are suppose to add the six subroutines. This will be around line 45.

Fifteenth, add the subroutines. I used the "separate words in memory" (page 5) approach to represent the state of the game. I added a whopping 106 lines to the file. Of these 12 lines were either blank or contained only comments, 39 were concerned with saving and restoring registers, 6 were instructions for returning from the routines, and there were `.FILL`s for saving the ASCII characters for `'0'` and `'X'`. That still leaves about 50 lines of hard-working code. Some routines were harder than others. Here is my hard-working code count for each routine. (Remember, this does not include lines for saving and restoring registers or returning to the caller.) This should give you some idea of which to write first.

`CheckBox`	3
`GetScore`	13
`OpenBox`	12
`OpenBoxes`	14
`PrintBoxes`	6
`ShutBox`	12

Sixteenth, assemble `boxes.asm` and make sure it assembles correctly.

Seventeenth, start up the LC/3 simulator and load the following object files in the order given below.

```
setup.obj
lfsr.obj
seed.obj
roll.obj
boxes.obj
```

Again, the only important thing is to load `boxes.obj` last, so that the simulator will start there.

Eighteenth and on, simulate and fix until all bugs are gone.