CS061 - Lab 05

More Arrays

1 High Level Description

In today's lab you will play around with arrays and loops a bit more, get some more practice with LDR, do some doubling down, and further explore the mysteries of console output.

2 Our Targets for This Week

1. Lab 04 review – Exercise 01
2. What a character! Exercise 02
3. Bits and pieces – Exercises 03 - 04
4. I rock!

3 Ok, let's hit these targets!

3.1 Lab 04 review

You will be building on the skills you learned last week: prompts, sentinel-controlled loops, and arrays, so let's start with a variation on last week's array – this time storing the results of a calculation, rather than user input.

Remember to always open the Text Window of your simpl LC-3 emulator!

Exercise 01

Use .BLKW to set up an array of 10 values, starting at memory location x4000, as in lab 4.
Now *programmatically* populate the array with the numbers 0 through 9 – i.e. hard-code just the first value (0), then calculate the rest one at a time, storing them in the array as you go.
After you've stored them all, grab the *seventh* value (which would be 6) and store it in R2.
Clearly, you can't access this location via a label, so you'll need it's actual address.
How will you obtain that? And how will you use it to get the value stored there?

As always, step through your program and examine the values as they are stored in the array, and examine the final value stored in R2, to make sure your program works as expected.

3.2 What a character!

Exercise 02

You'll notice that Exercise 01 didn't ask you to output to console the array you built.
Why not?
Because as you know by now, numbers are not digits!

So now go and add an output loop, making it output the *characters* corresponding to the *numbers* stored in your array, just as you learned to do in assignments 2 & 3.

3.3 Bits and pieces ...

Exercise 03

Let's try another modification of our well-used array program from Exercise 01:

This time, instead of calculating and storing the numbers from 0 to 9 in the array, calculate and store the first ten *powers of 2*, starting with $2^0 = 1$
Finally, grab the seventh value (2^6) from the array, and store it in R2.

In order to do this, you will have to figure out how to calculate powers of 2.
Some hints:

- Mathematically speaking: How do I obtain 2^{n+1} if I have 2^n?
- What LC3 operation could I use to multiply a number by 2?

As always, place a breakpoint in your program, and step through it, examining the values as they are being stored in the array, and examine the final value stored in R2, to make sure your program works as expected.

You already understand that you can't simply output the values in the array to the console "as is", so we have to manipulate them somehow to turn them into characters.
But this time all but the first four are *multi-digit numbers* when represented as decimal values, so our trick from the last exercise won't work – it can only convert the numbers from 0 to 9 into the single-digit ascii characters '0' through '9'.

This is going to take some more effort to solve, so we'll spread it out over a couple of labs & assignments.

Exercise 04
The first step will be to output the 1's and 0's of a stored value as *16 ascii characters*:

But wait! … You already did this in your last assignment!

I. Make a copy of your assn 3 code and modify it so that it prints out the contents of register 2.
II. Now copy your ex3 code into lab05_ex4.asm, and add in your new output code: drop it in after you load the 7th array value to R2.
At this point your program should print out 2^6 in binary format:

```
b0000 0000 0100 0000
```

So now we know how to print any number in binary format *(put the value in R2 and paste the binary print code)*; and we also know how to write code that can build and traverse an array. It is time to combine our knowledge to print out all elements in the array!!

III. Modify your program so that it prints all the values in the array of powers of 2.

i.e.

```
b0000 0000 0000 0001
b0000 0000 0000 0010
b0000 0000 0000 0100
        ….
        ….
b0000 0001 0000 0000
b0000 0010 0000 0000
```

Keep your code clean and well organized, for example in pseudo-code it should look something like this:

```
ex. 03: Prepare the binary array with successive powers of 2

ex. 04: Load starting address of array into a base register (say R6)
for each element in array
        load the value to R2
        print the value in R2 (use 'the code')
end of program (HALT)
// program data
```

3.4 Submission

Add, commit, and push your lab05_ex1.asm through lab05_ex4.asm files to your lab 5 GitHub repo.

4 So what do I know now?

- You should be able to do counter and sentinel controlled loops in your sleep
- Ditto for multi-way branches (the AL version of if statements)
- The difference between a *number* (an abstract concept) and its various *representations* – in different bases (2, 10, 16), and as a character(s) (specifically, decimal numeric digits) with an ascii code
- Output values from 0 to 9 as their corresponding ascii characters
- Use the above techniques to inspect each bit of a 16-bit number and output it as the character '1' or '0'