Gavin Mack – gmmack

Lab section: 4 TTh TA: Ehsan

Due: 2/9/14

Lab Partner: None

Title:

Lab 3: Introduction to LC-3

Purpose:

The purpose of this lab is to familiarize ourselves with LC-3 assembly language, and to become familiar with keeping track of allocated memory addresses.

Procedure:

I first looked through the provided echo.asm file to get used to the syntax of LC-3. I then Googled various LC-3 examples and read Chapter 5 of our textbook to get a better idea of what I needed to do. I then began writing code, first figuring out how to display a greeting, then how to display a prompt, get user input with getc, echo it back with putc, and store it using ST. I quickly realized that before you store inputs from getc you must subtract 48 to accommodate ascii values, and before you display anything you have to add 48. To make this easier, I set R5 to 48 and R6 to -48. This is definitely not the most efficient way to use the registers, but these registers were extraneous for this assignment so it does not really matter. I checked to make sure everything was working so far, and then began formulating the algorithm for subtraction, then multiplication, then division. Once I knew how I was going to implement these functions, I began writing the code for them. Subtraction was fairly easy to implement, and although I struggled through writing the division and multiplication functions, my algorithms for them were accurate and I got it working in the end.

Algorithms and Other Data:

The subtraction algorithm was by far the easiest to implement. In order to perform subtraction, we first invert our second input and add one to form the 2's complement. We then simply add this number to input one and print out the result. I had a fair amount of difficulty implementing the multiplication algorithm; mostly because I got confused on the registers I was loading into. Multiplication is done with a loop. For each iteration of the loop I would add input one to R2 and subtract one from input two. I would then check if input two was greater than zero I would keep looping, else I would stop looping and my product would be in R2. If the product was <10 I would print it, but if it was >=10 I needed to call a different print function which I named PRINT2. PRINT2 works with a loop also. It works by subtracting ten from the result until value<10. I would count the number of times through the loop, and the count would become the ten's place value. The remaining value when the loop exits is the one's place value.

Division was implemented with a loop as well. For each iteration of the loop I would subtract input two from input one and increment R2. I would then check if result < INPUT2 stop looping, and the value in R2 is the quotient and the value in R0 is my remainder. If result >=INPUT2 keep looping. I would then store and print my results. Once each of the individual functions were complete, I put in a branch to START directly before my HALT call, so that the program can continually take input.

What Went Wrong or What Were The Challenges?

Where do I start? One of the most frustrating things for me was that I would I would write some segment of code, it wouldn't work, I'd spend a bunch of time figuring out what was wrong and fixing it, then I would go to test it but forget to hit assemble on the edit browser. I would then spend another twenty minutes or so trying to figure out why my solution did not fix the problem. Granted this is a problem that was arising out of my own stupidity. One of the most difficult parts of the lab for me was keeping track of the information in each register. The algorithms for each arithmetic operation took some time to figure out, but they were not too complicated. Getting used to LC-3 syntax took some time as well but it is a fairly straight-forward language. It just takes a lot of code to do anything. I would however find myself continually overwriting some register which I needed, and then it would take me a while to go back through my code to discover where I made the mistake.

Other Information:

Sample output:

Please enter a one digit number: 9

Please enter a one digit number: 4

Subtraction Result: 5

Multiplication Result: 36

Division Result: 2

Division Remainder: 1

Please enter a one digit number:

A branch instruction in LC-3 works by sending program execution back to wherever is specified by the operand of the branch. For each, BR START sends execution back to wherever START: is written in the program, in this case it is the start of the program.

I used ST to store to the correct memory locations, unless I wanted to store to a specific register I used STI.

An addressing mode is how operands are specified, or how the next instruction to execute is specified. The five LC-3 addressing modes are Immediate, Register, PC-Relative, Base+Offset, and Memory-Indirect. Immediate is when a numeric value embedded in the instruction is the actual operand. Register

is when a source or destination operand is specified as content of one of the registers R0-R7. PC-Relative is when a data or instruction memory location is specified as an offset relative to the incremented PC. Base+Offset is when a data or instruction memory location is specified as a signed offset from a base register. Memory-Indirect is when a data memory location is a pointer to a data memory location.

R7 is used to put return value for TRAP instructions.

The PUTS TRAP is executed by first looking up the starting address. It then transfers to service routine, then Returns (JMP R7). PUTS writes a string to the console.

Conclusion:

Everything definitely did not go as expected. I thought I had everything working at first, and then when I went to test it, the program gave me 8 errors. I googled the error and it turns out I was having an issue with string declarations being more than 256 memory addresses away from where they were being called. I thought about trying to move around some of my string declarations, but that seemed like it might create more problems. In the end I fixed this by condensing some of my strings so that I used fewer memory addresses. I learned how to create an algorithm to perform multiplication and division, and this lab was very helpful in reinforcing how to do 2's complement subtraction. This lab was a good exercise in keeping track of memory addresses and allocating memory. I now have a much better understanding of registers, and know what all of the various system calls in LC-3 do.

Extra:

I don't know if anyone else felt this way, but I thought we were vastly unprepared for this lab. I'm not sure if this is the intention or if anyone else felt this way, but I felt like I was for the most part teaching myself LC-3.