Jason Ou (jaou 1385128)
Partner: Andy Wang
CMPE 12L
9 FEB 2015
TA: Daphne Gorman
Section: 6

Lab Write-Up #3

OVERVIEW/INTRODUCTION

The purpose of this lab was to make a program in LC-3 assembly language that takes in two single digit user inputs in the form of decimal numbers, and using these two numbers, the program will then perform a subtraction, multiplication, and division of the two inputs. The results of these operations will be stormed in four consecutive memory locations in the LC-3 simulator.

METHODOLOGY/PROCEDURES

PART A: SUBTRACTION

To create a subtraction subroutine, I had to take the two's compliment of the second user input and add that with the first user input. This value then gets stored into x3100.

PART B: MULTIPLICATION

To create a multiplication subroutine, I had to continually add user input one with itself for the amount of times the value of user input two was. This value then gets stored into x3101.

PART C: DIVISION

To create a multiplication subroutine, I had to subtract the second user input from the first user input continually until the first user input became either 0 or a negative number. The amount of times I subtracted from the first input until my final condition was met would be the value that is stored into x3102. The remainder of the division would be the leftover of what cannot be subtracted anymore, and that would be stored into x3103.

RESULTS

PART A: SUBTRACTION

Subtraction worked as expected, but only when the first user input is greater than the second user input. The user input also has to be single digit values as well.

PART B: MULTIPLICATION

Multiplication works as expected, but it can only multiply by values that are greater than 0.

PART C: DIVISION

Division works as expected, and it cannot divide by 0.

DISCUSSION

1. **The algorithm you are using in your program is simple, layperson's terms (or in C-like pseudocode).**

- Subtract(int A, int B) {
  int result = 0;
  B = Invert(B);
  B++;
  Result = A+B;
  Store(result, x3100);

Jason Ou (jaou 1385128)
Partner: Andy Wang
CMPE 12L
9 FEB 2015
TA: Daphne Gorman
Section: 6

```
}
```

- ```
  Multiply(int A, int B) {
      int result = 0;
      while(B != 0) {
          result += A;
          B--;
      }
      Store(result, x3101);
  }
  ```
- ```
  Divide(int A, int B) {
      B = Invert(B);
      B++;
      int result = 0;
      While(A != 0 or A != negative) {
          A += B;
          result++;
      }
      If(A == negative) {
          int remainder = A + Invert(B);
          result--;
      } else {
          remainder = 0;
      }
      Store(result, x3102);
      Store(remainder, x3103);
  }
  ```

2. <u>What works and what does not.</u>

- Subtraction: User input one must always be bigger than user input two. It has to also be single digit user inputs.
- Multiplication: Can only multiply single digit user inputs that are greater than 0.
- Division: Cannot divide by 0.

3. <u>Sample output.</u>

- User Inputs → (10, 5)
- Subtract(10, 5) → 5 stored into x3100.
- Multiply(10, 5) → 50 stored into x3101.
- Divide(10, 5) → 2 stored into x3102, 0 stored into x3103

4. <u>How does a branch instruction in LC-3 work?</u>

- A conditional branch instruction is taken and changes the program counter only if a specified condition is true, else it will be ignored. A conditional branch has three conditions, N (negative), Z (zero), and P (positive). Depending on the value of the operation of the line of code above the branch, it will change program counters only if the above operation matches with the condition of the branch.

Jason Ou (jaou 1385128)
Partner: Andy Wang
CMPE 12L
9 FEB 2015
TA: Daphne Gorman
Section: 6

- An unconditional branch (or Jump) instruction will always change the program counter. This is similar to calling a method in more modern programming languages. It will only be able to call the method if there is a method though.

5. **Which store instruction (ST, STI, STR) did you use to store your result to the correct memory location? Why did you use that particular instruction and not the other two?**

    - I used STI to store my results of my program algorithms into the correct memory locations. I used STI because it is an indirect addressing mode, allowing me to read address from a memory location, then load/store my result into that address that I want it to go to.

6. **What is an addressing mode? What are the five LC-3 addressing modes? Give an example of each one.**
    - Addressing modes specify the location of an operand.
    - Immediate: LEA
    - Register: R0
    - PC-relative: LD
    - Indirect: LDI
    - Base + offset: STR

7. **Which register is used as a place to put return values for TRAP instructions?**

    - Whichever register the program specifies and is currently being used for the operation will be where the return values for the TRAP instructions will be at.

8. **By stepping through your program, what does the PUTS trap do? Please explain the entire process of how PUTS is executed.**

    - PUTS trap prints an array of characters or strings onto the console window. The address of the first character is stored into R0 and each register gold two characters and the data is then converted into ASCII before printing to the screen. It continues printing along the path of the string until the program finds consecutive data reading 0x0000;

CONCLUSION

This lab was used to learn how to program using LC-3 and create a program that is able to subtract, multiply, and divide two single digit user inputs. I was able to learn how to create a simple program and learn to basics of LC-3 and assembly. I learned how to use registers, control memory locations, and using logic operations, create a program that did mathematical operations. In the end, I was able to gain a strong foundation on how LC-3 and assembly works.