CS 2110: Lecitation 08

Name:

Wednesday, June 13th, 2018

Introduction

Today we will be writing simple LC-3 assembly programs! The lecitation will be divided into three parts.

Make sure to ask the TA's questions!

Part 1: Calculating the Absolute Value (`absolute.asm`)

Assignment: Write an assembly program that returns the absolute value of `U`.

Pseudocode:

```
if (U < 0) {
  ANSWER = -U
} else {
  ANSWER = U
}
```

Explanation: `U` is a label for a `.fill` in the template file. It will be any integer in the range $-32,767$ to $32,767$, inclusive. You will store your answer in the label `ANSWER`.

Testing: Run the command `lc3test testing/absolute_test.xml absolute.asm`.

Part 2: Searching an Array (`search.asm`)

Assignment: Write a program to search for a certain value in an array.

Pseudocode:

```
var counter = 0;
RESULT = 0;

while (counter < LENGTH) {

  if (ARRAY[counter] == NUMBER) {
    RESULT = 1;
    break;
  }

  counter++;
}
```

Explanation: The elements of an array are always contiguous in memory. The length of the array is given in `LENGTH`, and the starting address of the array is in `ARRAY`. The number you are searching for is in `NUMBER`. If the number exists in the array, store the number `1` in `RESULT`. Otherwise, store the number `0` in `RESULT`. **Note:** The array can have zero or more elements.

Testing: Run the command `lc3test testing/search_test.xml search.asm`.

Part 3: Writing a TRAP Routine (trap40.asm)

BE SURE TO ENABLE TRUE TRAPS IN COMPLX BEFORE YOU RUN YOUR CODE.

Introduction: Traps are subroutines that you can call using the TRAP instruction. PUTS, GETC, OUT, and HALT are all examples of pseudo-operations that assemble to trap instructions. They work using the Trap Vector Table.

The Trap Vector Table is an area in memory (x0000 - x00FF) that contains the addresses of where the data of the trap subroutines are located in memory. For example, if the code for TRAP x60 was located at address x7000 in memory, the corresponding entry in the trap vector table (x60) will contain the value x7000. When the instruction TRAP x60 is executed, the current PC is saved to register R7, the datapath looks up where TRAP x60 is located (x7000), the PC is changed to that value (x7000), the trap executes, and finally, after the trap is done executing, the PC is changed back to the original value (the one initially stored in R7).

Assignment: Implement a trap that makes a character string pointed to by R0 all lowercase. For instance, if R0 points to the string "Hello World!", the trap should change its contents to "hello world!". This trap should be callable by writing TRAP x40 in your code. No registers (except R7) should be modified. You do not need to conform to any calling conventions.

Pseudocode:

```
var c = str[0];

while (c != 0) {
  if (c >= 'A' && c <= 'Z') {
    c += 32;
    str[0] = c;
  }

  string++;
  c = str[0];
}
```

Note: If you see "Warning at 0x3007: Unsupported Trap x40" then enable true traps from State > True Traps (or just ctrl + T). Yes, it's normal that your program is halting at address 0x054A. By enabling true traps, the actual trap code gets processed by complx, and this is the location where the processor stops in the HALT trap.

Strings in C and LC-3 are null-terminated, meaning that there is a value of zero at the end of the string. Printing functions like printf and PUTS know to stop printing characters when they reach this value.

Testing: Run the command lc3test testing/trap40_test.xml trap40.asm.