

CMSC B240 Computer Organization - Spring 2024

Lab Activity #7 – LC-3 Input/Output

Recall that the only input device available in LC-3 is the keyboard and the output device is the display (Console Window in the LC-3 Simulator). In this lab we will learn the instructions available in LC-3 to do input/output.

The TRAP Instruction

The LC-3 TRAP instruction provides access to some basic **services** often needed in writing programs. These include input, output, as well as the instruction to stop execution of a program (**HALT**).

The structure of the TRAP instruction is shown below:

```
1111 0000 trapvect8
```

The leftmost 4 bits specify the opcode (=1111), the next four bits are always 0000. The rightmost eight bits (**trapvect8**) specify something called a *trap vector*. Trap vectors are designated locations in LC-3's memory where code for providing some services (like input, output, stop, etc.) reside. We will learn more about these later in the course. For now, we can introduce how the TRAP instruction is used to access services. For example, the HALT instruction, that stops execution of a program, is coded as show below:

```
1111 0000 0010 0101
```

Notice that the instruction has the opcode (1111) for the TRAP instruction. The trap vector specified is **x25** (or **0010 0101**). That is, the HALT instruction can also be written as:

```
TRAP x25 ; This is the same as HALT
```

It is the LC-3 Assembly Language that provides the opcode **HALT** and translates it into code for the TRAP instruction as specified above.

In addition to HALT (trapvect8 = x25) the following trap vectors are defined in LC-3:

```
TRAP x23 ; input a character from keyboard into R0
TRAP x21 ; output the char code in R0 to display
TRAP x22 ; output a string (null terminated) to display
          ; Address of first char in string should be in R0
```

Let us write a few short programs to test these:

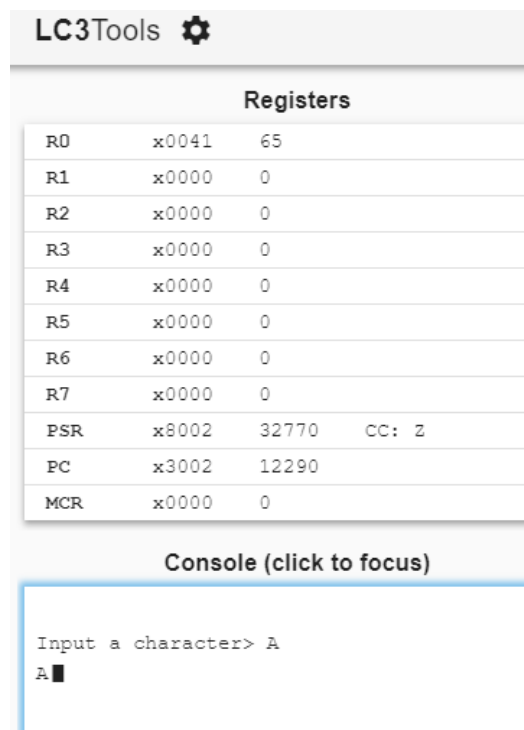
CMSC B240 Computer Organization - Spring 2024

Lab Activity #7 – LC-3 Input/Output

Program 1: Write an LC-3 Assembly program to input a character from keyboard and output it to the display. Here is a short program to do this:

```
.ORIG    x3000
START   TRAP    x23    ; input a character from keyboard
        TRAP    x21
        HALT
        .END
```

Go ahead enter it in the LC-3 simulator and store it in a file called `io.asm`. Assemble it, and then run the program (make sure to put a debugger stop at the HALT instruction). You should see the following in the console window:



The screenshot shows the LC3Tools simulator interface. At the top, there is a header "LC3Tools" with a gear icon. Below this is a "Registers" window containing a table of register values:

Register	Address	Value	Other Info
R0	x0041	65	
R1	x0000	0	
R2	x0000	0	
R3	x0000	0	
R4	x0000	0	
R5	x0000	0	
R6	x0000	0	
R7	x0000	0	
PSR	x8002	32770	CC: Z
PC	x3002	12290	
MCR	x0000	0	

Below the registers is a "Console (click to focus)" window. The console shows the prompt "Input a character>" followed by the user input "A" and a cursor. The character "A" is also echoed back on the next line.

When an input instruction is executed (TRAP x23), a prompt is shown in the Console window (**Input a character>**). Whatever character is entered on the keyboard is then echoed back (by the TRAP x21 instruction). Also, notice that after execution, the register R0 contains the ASCII code for the letter A (= 65).

CMSC B240 Computer Organization - Spring 2024

Lab Activity #7 – LC-3 Input/Output

Program 2: Output a string stored in LC-3 memory to display using the TRAP x24 instruction. Here is the program:

```
        .ORIG    x3000
START   LEA     R0, MMSG      ; load starting address of string (MMSG) in R0
        TRAP    x22          ; output the string
        HALT
MMSG    .STRINGZ "Hello, World!"
        .END
```

This time, you will see the string “Hello, World!” printed in the Console. Examine how LC-3 assembler stores the string (null terminated).

Program 3: Write/Modify one of the above programs to show the following behavior:

```
Input a character> A
Your Entered:A█
```

IN, OUT, PUTS, HALT

Like the HALT instruction, which is equivalent to TRAP x25, the LC3-3 Assembler also defines the following three additional opcodes:

```
IN      ; Input a character from keyboard into R0 (= TRAP x23)
OUT     ; Output a character stored in R0 to Console (TRAP x21)
PUTS    ; Output a null-terminated string whose starting address is
        ; stored in R0 to console.
```

These instructions do not have any operands (just like HALT).

Exercise: Modify Program 3 from above to use the IN, OUT, and PUTS instructions.

CMSC B240 Computer Organization - Spring 2024
Lab Activity #7 – LC-3 Input/Output

LC-3 Assembly Cheat Sheet

Instruction		Action	Addressing Mode
ADD	R2, R2, R3	$R2 = R2 + R3$	Register
ADD	R2, R2, #1	$R2 = R2 + 1$	Register, Immediate
AND	R2, R2, R3	$R2 = R2 \text{ AND } R3$	Register
AND	R2, R2, #1	$R2 = R2 \text{ AND } 1$	Register, Immediate
BR _{[n][z][p]}	LABEL	If [n][z][p] Go to LABEL	CC, PC-Relative
HALT		Stop program execution	
IN		R0 = Input char from keyboard	None
JMP	R1	PC = R1	Register
JSR			
JSRR			
LD	R2, LABEL	$R2 = m[\text{LABEL}]$	Register, PC-Relative
LDI	R2, LABEL	$R2 = m[m[\text{LABEL}]]$	Register, Indirect
LDR	R2, R0, #n ₆	$M[R0 + n]$	Base Register
LEA	R2, LABEL	$R2 = \text{LABEL}$	Register, PC-Relative
NOT	R2, R1	$R2 = \text{NOT}(R1)$	Register
OUT		Output R0 to Console	None
PUTS		Output String at M[R0] to console	None
RET			
RTI			
ST	R2, LABEL	$M[\text{LABEL}] = R2$	Register, PC-Relative
STI	R2, LABEL	$M[m[\text{LABEL}]] = R2$	Register, Indirect
STR	R2, R0, #n ₆	$M[R0 + n] = R2$	Register, Base Register
TRAP	trapvect ₈	Execute Service # trapvect ₈	Immediate