

Chapter 7 Assembly Language

Based on slides © McGraw-Hill
Additional material © 2004/2005 Lewis/Martin
Modified by Diana Palsetia (2007)

Starting LC-3 simulator

Download **PennSim.jar** to a folder

Open command prompt:

>cd C:\

>cd S:

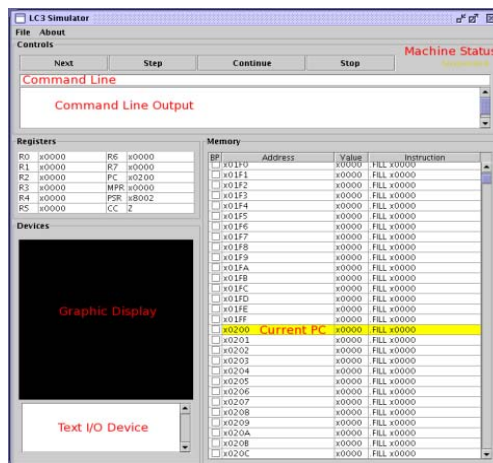
>cd (where you save PennSim.jar)

>java -jar PennSim.jar

CIT 593

2/9

LC3 Simulator



CIT 593

3/9

Memory

Address Range	Usage
x0000 - x00FF	Trap Vector Table
x0100 - x01FF	Interrupt Vector Table
x0200 - x2FFF	Operating System
x3000 - xBFFF	User code & stack
xC000 - xFDFF	Video output
xFE00 - xFFFF	Device register addresses

Note: User program accessing memory other than x3000 – xBFFF will cause your program to abruptly stop (we say an exception occurred)

CIT 593

4/9

Registers

Registers (Values displayed in hex)

- PC (16 bits) indicates the address of the next instruction to be executed
- R0 – R7
 - Do not use R0, R1 & R7 to display any result at the end of your program.
 - Some OS instructions overwrite these registers after the program ends (HALT)
- CC: NZP letters are displayed
- Process Status Register - PSR (16 bits)
 - LC-3 is operating in *supervisor mode (OS)* or *user mode*
 - If supervisor mode is enabled, PSR[15] is 1.
 - Allows OS access to the different devices available to the machine
 - PSR[10:8] specify the priority level of the process being executed (More when we do interrupts)
 - PSR[2:0] contain the bits for the condition codes (CCs)
- Memory Status Registers – MSR (16 bits)
 - Each bit in the MSR controls whether instructions in a given memory range can be executed while in user mode
 - Each bit of the MSR controls 4096 (x1000) memory locations
 - If not authorized – program will abruptly halt

CIT 593

5/9

LC-3 Assembler

Assembler converts Assembly to Machine code

To compile(assemble) your assembly program

- Type in “command line”: `as filename.asm`

Assembler generates two different output files

Symbol file (.sym)

- Includes names of labels (also known as symbols)
- Used by simulator to make code easier to read
- A text file of symbol mappings

Object file (.obj)

- Binary representation of the program

CIT 593

6/9

Object File Format

LC-3 object file contains

- Starting address (location where program must be loaded), followed by...
- Machine instructions
- Real-world object file formats can be more complicated
 - E.g. ELF (Executable and Linking Format) for Unix/Linux

LC-3 Example

```
0011000000000000<— .ORIG x3000
0101010010100000<— AND R2, R2, #0
0010011000010001<— LD R3, PTR
1111000000100011<— TRAP x23
.
.
.
```

CIT 593

7/9

Using Multiple Object Files

An object file is not necessarily a complete program

- System-provided library routines
- Code blocks written by multiple developers

For LC-3 simulator

- Load multiple object files into memory, then start executing at a desired address
 - Command: `load filename.obj`
- Each object file includes a starting address
- User code should be loaded between x3000 and xFDFE
- Be careful not to load overlapping object files

CIT 593

8/9

Aside: Loading & Linking

Loading is the process of copying an executable image into memory

- More sophisticated loaders are able to relocate images to fit into available memory
- Must re-adjust branch targets, load/store addresses

Linking is the process of resolving symbols between independent object files

- Suppose we define a symbol in one module, and want to use it in another
- Some notation, such as `.EXTERNAL`, is used to tell assembler that a symbol is defined in another module
- Linker will search symbol tables of other modules to resolve symbols and complete code generation before loading