

Computer Science 210 s1c
Computer Systems 1
2008 Semester 1
Lecture Notes

Lecture 15, 9Apr08:
The LC-3 ISA; Assembly Language

James Goodman

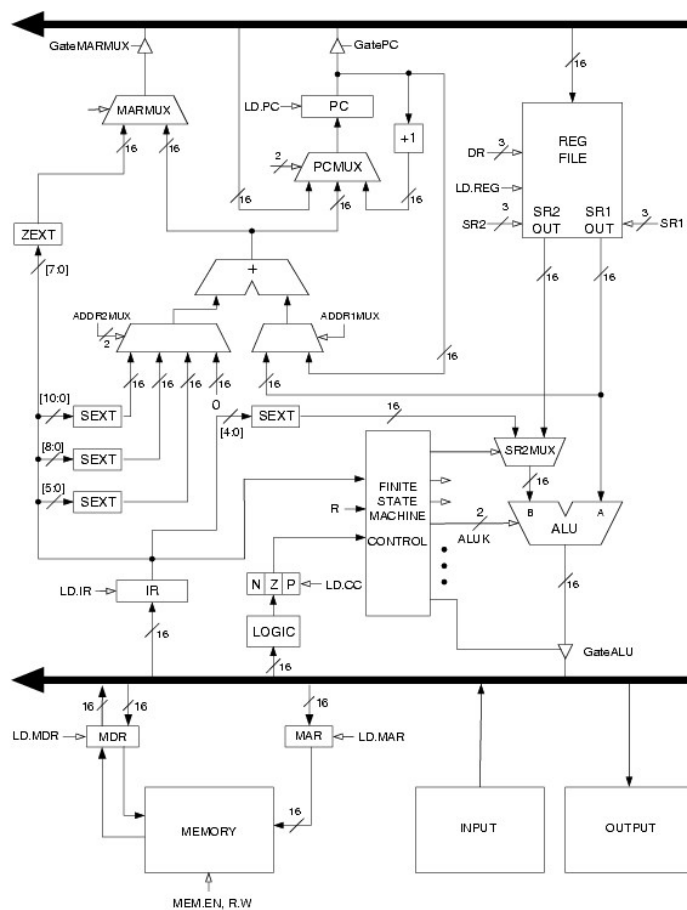


Credits: Slides prepared by Gregory T. Byrd, North Carolina State University

Copyright © The McGraw-Hill Companies, Inc. Permission required for reproduction or display.

**LC-3
Data Path
Revisited**

Filled arrow
= info to be processed.
Unfilled arrow
= control signal.



Chapter 7

Assembly Language

Copyright © The McGraw-Hill Companies, Inc. Permission required for reproduction or display.

Human-Readable Machine Language

Computers like ones and zeros...

0001110010000110

Humans like symbols...

ADD R6,R2,R6 ; *increment index reg.*

Assembler is a program that turns symbols into machine instructions.

- ISA-specific:
 - close correspondence between symbols and instruction set
 - mnemonics for opcodes
 - labels for memory locations
- additional operations for allocating storage and initializing data

An Assembly Language Program

```

;
; Program to multiply a number by the constant 6
;
        .ORIG    x3050
        LD      R1, SIX
        LD      R2, NUMBER
        AND     R3, R3, #0      ; Clear R3. It will
                                ; contain the product.
; The inner loop
;
AGAIN   ADD     R3, R3, R2
        ADD     R1, R1, #-1    ; R1 keeps track of
        BRp    AGAIN          ; the iteration.
;
        HALT
;
NUMBER  .BLKW   1
SIX     .FILL   x0006
;
        .END

```

9-Apr-08

CS210

262

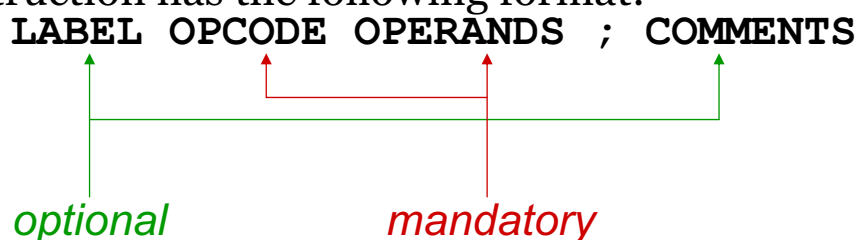
LC-3 Assembly Language Syntax

Each line of a program is one of the following:

- an instruction
- an assembler directive (or pseudo-op)
- a comment

Whitespace (between symbols) and case are ignored.
Comments (beginning with “;”) are also ignored.

An instruction has the following format:



9-Apr-08

CS210

263

Opcodes and Operands

Opcodes

- reserved symbols that correspond to LC-3 instructions
- listed in Appendix A
 - ex: ADD, AND, LD, LDR, ...

Operands

- registers -- specified by Rn, where n is the register number
- numbers -- indicated by # (decimal) or x (hex)
- label -- symbolic name of memory location
- separated by comma
- number, order, and type correspond to instruction format
 - ex:


```
ADD R1, R1, R3
ADD R1, R1, #3
LD  R6, NUMBER
BRz LOOP
```

Labels and Comments

Label

- placed at the beginning of the line
- assigns a symbolic name to the address corresponding to line
 - ex:


```
LOOP  ADD R1, R1, #-1
      BRp LOOP
```

Comment

- anything after a semicolon is a comment
- ignored by assembler
- used by humans to document/understand programs
- tips for useful comments:
 - avoid restating the obvious, as “decrement R1”
 - provide additional insight, as in “accumulate product in R6”
 - use comments to separate pieces of program

Assembler Directives

Pseudo-operations

- do not refer to operations executed by program
- used by assembler
- look like instruction, but “opcode” starts with dot

<i>Opcode</i>	<i>Operand</i>	<i>Meaning</i>
<code>.ORIG</code>	<code>address</code>	starting address of program
<code>.END</code>		end of program
<code>.BLKW</code>	<code>n</code>	allocate n words of storage
<code>.FILL</code>	<code>n</code>	allocate one word, initialize with value n
<code>.STRINGZ</code>	<code>n-character string</code>	allocate n+1 locations, initialize w/characters and null terminator

Trap Codes

LC-3 assembler provides “pseudo-instructions” for each trap code, so you don’t have to remember them.

<i>Code</i>	<i>Equivalent</i>	<i>Description</i>
<code>HALT</code>	TRAP <code>x25</code>	Halt execution and print message to console.
<code>IN</code>	TRAP <code>x23</code>	Print prompt on console, read (and echo) one character from keybd. Character stored in <code>Ro[7:0]</code> .
<code>OUT</code>	TRAP <code>x21</code>	Write one character (in <code>Ro[7:0]</code>) to console.
<code>GETC</code>	TRAP <code>x20</code>	Read one character from keyboard. Character stored in <code>Ro[7:0]</code> .
<code>PUTS</code>	TRAP <code>x22</code>	Write null-terminated string to console. Address of string is in <code>Ro</code> .