



## LC-3 Details and Examples

ECE/CS 252, Fall 2010  
Prof. Mikko Lipasti  
Department of Electrical and Computer Engineering  
University of Wisconsin – Madison

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

### Review: Instruction Set Architecture

**ISA** = All of the *programmer-visible* components and operations of the computer

- **memory organization**
  - address space -- how many locations can be addressed?
  - addressability -- how many bits per location?
- **register set**
  - how many? what size? how are they used?
- **instruction set**
  - opcodes
  - data types
  - addressing modes

ISA provides all information needed for someone that wants to write a program in *machine language* (or translate from a high-level language to machine language).

5-2

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

### Using Operate Instructions

With only ADD, AND, NOT...

How do we subtract?

Subtract:  $R3 = R1 - R2$   
Take 2'sC of R2, then add  
(1)  $R2 = \text{NOT}(R2)$   
(2)  $R2 = R2 + 1$   
(3)  $R3 = R1 + R2$

How do we OR?

OR:  $R3 = R1 \text{ OR } R2$   
Use DeMorgan's Law  
(1)  $R1 = \text{NOT}(R1)$   
(2)  $R2 = \text{NOT}(R2)$   
(3)  $R3 = R1 \text{ AND } R2$   
(4)  $R3 = \text{NOT}(R3)$

How do we copy from one register to another?

Register-to-register copy:  $R3 = R2$   
 $R3 = R2 + 0$  (Add-immediate)

How do we initialize a register to zero?

Initialize to zero:  $R1 = 0$   
 $R1 = R1 \text{ AND } 0$  (And-immediate)

5-3

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

### Review: Data Movement Instructions

**Load** -- read data *from memory to register*

- **LD**: PC-relative mode
- **LDR**: base+offset mode
- **LDI**: indirect mode **NEW**

**Store** -- write data *from register to memory*

- **ST**: PC-relative mode
- **STR**: base+offset mode
- **STI**: indirect mode **NEW**

**Load effective address** -- compute address, save in register

- **LEA**: PC-relative mode
- *does not access memory*

5-4

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

### Indirect Addressing Mode

With PC-relative mode, can only address data within 256 words of the instruction.

- What about the rest of memory?

**Solution #1:**

- Read address from memory location, then load/store to that address.

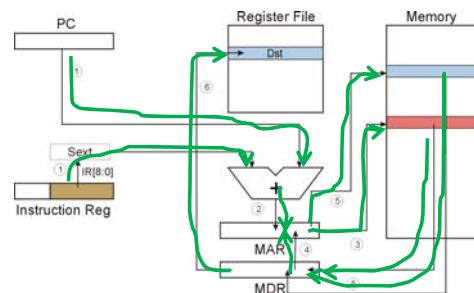
First address is generated from PC and IR (just like PC-relative addressing), then content of that address is used as target for load/store.

5-5

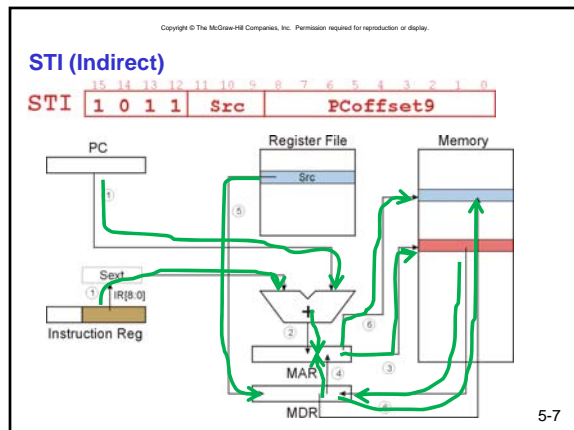
Copyright © The McGraw-Hill Companies, Inc. Permission required for reproduction or display.

### LDI (Indirect)

LDI 1 0 1 0 Dst PCoffset9



5-6



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

### Example

Address	Instruction	Comments
x30F6	1 1 1 0 0 0 1 1 1 1 1 1 1 0 1	$R1 \leftarrow PC - 3 = x30F4$
x30F7	0 0 0 1 0 1 0 0 1 1 0 1 1 1 0	$R2 \leftarrow R1 + 14 = x3102$
x30F8	0 0 1 1 0 1 0 1 1 1 1 1 1 0 1	$M[PC - 5] \leftarrow R2$ $M[x30F4] \leftarrow x3102$
x30F9	0 1 0 1 0 1 0 1 0 1 0 1 0 0 0	$R2 \leftarrow 0$
x30FA	0 0 0 1 0 1 0 1 0 1 0 1 0 1 1	$R2 \leftarrow R2 + 5 = 5$
x30FB	0 1 1 1 0 1 0 0 1 0 0 1 1 1 0	$M[R1 + 14] \leftarrow R2$ $M[x3102] \leftarrow 5$
x30FC	1 0 1 0 0 1 1 1 1 1 1 0 1 1 1	$R3 \leftarrow M[M[x30F4]]$ $R3 \leftarrow 5$

opcode

5-8

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

### Control Instructions

Used to alter the sequence of instructions (by changing the Program Counter)

#### Conditional Branch

- branch is *taken* if a specified condition is true
  - > signed offset is added to PC to yield new PC
- else, the branch is *not taken*
  - > PC is not changed, points to the next sequential instruction

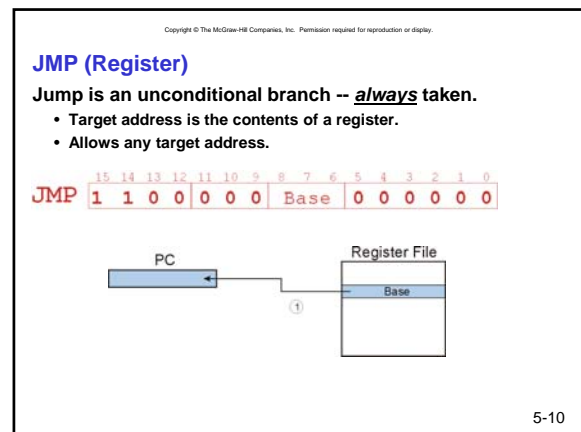
#### Unconditional Branch (or Jump)

- always changes the PC

#### TRAP

- changes PC to the address of an OS "service routine"
- routine will return control to the next instruction (after TRAP)

5-9



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

### TRAP

TRAP 1 1 1 1 0 0 0 0 trapvect8

Calls a **service routine**, identified by 8-bit "trap vector."

vector	routine
x23	input a character from the keyboard
x21	output a character to the monitor
x25	halt the program

When routine is done,  
PC is set to the instruction following TRAP.  
(We'll talk about how this works later.)

5-11

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

### Another Example

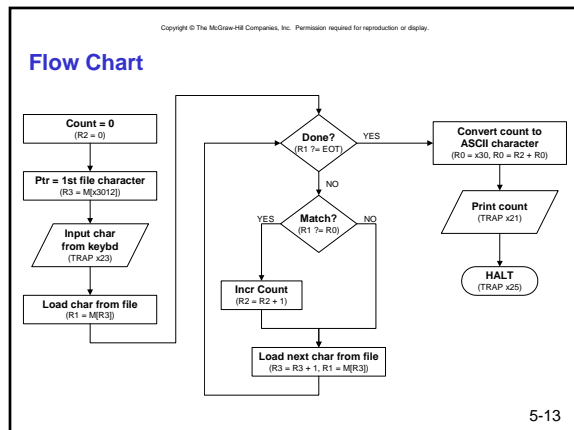
#### Count the occurrences of a character in a file

- Program begins at location x3000
- Read character from keyboard
- Load each character from a "file"
  - File is a sequence of memory locations
  - Starting address of file is stored in the memory location immediately after the program
- If file character equals input character, increment counter
- End of file is indicated by a special ASCII value: **EOT (x04)**
- At the end, print the number of characters and halt (assume there will be less than 10 occurrences of the character)

A special character used to indicate the end of a sequence is often called a **sentinel**.

- Useful when you don't know ahead of time how many times to execute a loop.

5-12



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

### Program (1 of 2)

Address	Instruction	Comments
x3000	0 1 0 1 0 1 0 0 1 0 0 0 0 0	$R2 \leftarrow 0$ (counter)
x3001	0 0 1 0 0 1 1 0 0 0 0 1 0 0 0 0	$R3 \leftarrow M[x3102]$ (ptr)
x3002	1 1 1 1 0 0 0 0 0 0 1 0 0 0 1 1	Input to R0 (TRAP x23)
x3003	0 1 1 0 0 0 1 0 1 1 0 0 0 0 0 0	$R1 \leftarrow M[R3]$
x3004	0 0 0 1 1 0 0 0 1 1 1 1 1 1 0 0	$R4 \leftarrow R1 - 4$ (EOT)
x3005	0 0 0 0 0 1 0 0 0 0 0 0 1 0 0 0	If Z, goto x300E
x3006	1 0 0 1 0 0 1 0 0 1 1 1 1 1 1 1	$R1 \leftarrow \text{NOT } R1$
x3007	0 0 0 1 0 0 1 0 0 1 1 0 0 0 0 1	$R1 \leftarrow R1 + 1$
x3008	0 0 0 1 0 0 1 0 0 1 0 0 0 0 0 0	$R1 \leftarrow R1 + R0$
x3009	0 0 0 0 1 0 1 0 0 0 0 0 0 0 0 1	If N or P, goto x300B

5-14

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

### Program (2 of 2)

Address	Instruction	Comments
x300A	0 0 0 1 0 1 0 0 1 0 1 0 0 0 0 1	$R2 \leftarrow R2 + 1$
x300B	0 0 0 1 0 1 1 0 1 1 1 0 0 0 0 1	$R3 \leftarrow R3 + 1$
x300C	0 1 1 0 0 0 1 0 1 1 0 0 0 0 0 0	$R1 \leftarrow M[R3]$
x300D	0 0 0 0 1 1 1 1 1 1 1 1 0 1 1 0	Goto x3004
x300E	0 0 1 0 0 0 0 0 0 0 0 0 1 0 0	$R0 \leftarrow M[x3013]$
x300F	0 0 0 1 0 0 0 0 0 0 0 0 0 1 0	$R0 \leftarrow R0 + R2$
x3010	1 1 1 1 0 0 0 0 0 0 1 0 0 0 0 1	Print R0 (TRAP x21)
x3011	1 1 1 1 0 0 0 0 0 0 1 0 0 1 0 1	HALT (TRAP x25)
X3012	Starting Address of File	
x3013	0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0	ASCII x30 ('0')

5-15

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

### Summary

Common sequences for ALU ops

Indirect addressing mode: LDI/STI

Control: Jump/Trap

Detailed example

5-16