

Notes: The LC-3 Computer (Data Movement and Addressing Modes)

A. Why?

Low-level programs rely on moving data between storage and registers. Addressing modes are different ways to specify a location in memory.

B. Outcomes

By the end of the class you should

- Know how the LC-3 load and store instructions work.
- Know the different addressing modes of the LC-3.

C. Data Movement Instructions

- Basic data movement instructions
 - Load: Read data from memory to register
 - Store: Write data from register to memory
 - Different commands for different addressing modes
 - PC-relative, Base+Offset, Indirect, and Immediate

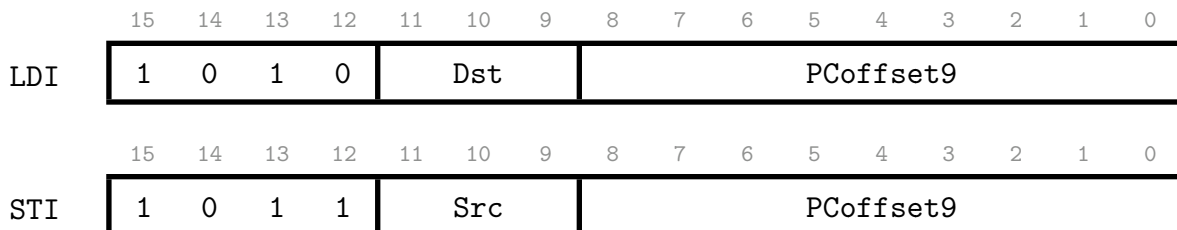
D. PC-relative addressing mode

- Instruction contains 9-bit signed integer (“offset”: $\geq -256, \leq +255$)
- Effective address of operand = PC + sign-extended 9-bit offset
 - PC is incremented as part of the FETCH phase.
 - So at EVALUATE ADDRESS state, the PC *already points to the next instruction*.
- LD instruction: Destination register, 9-bit offset
 - Load register with value at memory location (PC-relative addressing)
 - Dst reg $\leftarrow M[PC+offset]$
- ST instruction: Source register, 9-bit offset
 - Store register’s value to memory location (PC-relative addressing).
 - $M[PC+offset] \leftarrow \text{Src reg}$
- LEA instruction (Load Effective Address): Destination register, 9-bit offset
 - Calculate effective address using PC-relative addressing (like LD).
 - Load register *with the effective address* (not the value at the effective address) — doesn’t access memory.
 - Dst reg $\leftarrow PC+offset$ (not $M[PC+offset]$)



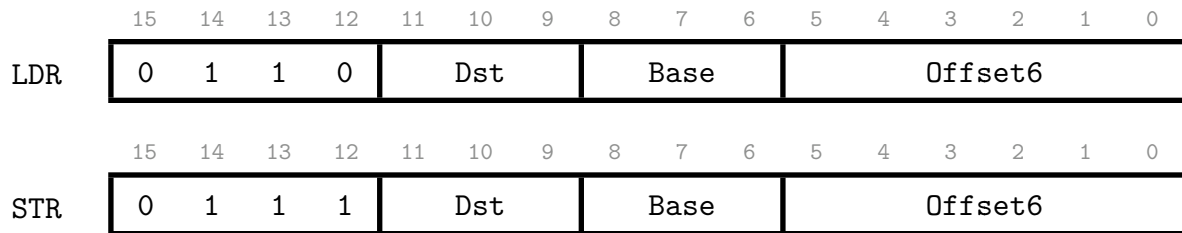
E. Indirect Addressing Mode

- PC-relative addressing only accesses “close” locations.
- One way to access “far” locations is to use pointers.
- LC-3 indirect addressing mode: Calculate effective address = $M[PC+offset]$
- (Compare with PC-relative addressing: effective address = $PC+offset$.)
- LDI (Load indirect)
 - $\text{Dst reg} \leftarrow M[M[PC+offset]]$
- STI (Store indirect)
 - $M[M[PC+offset]] \leftarrow \text{Src reg}$



F. Base+Offset Addressing Mode

- Another way to get to far-away locations
- Use a “base” register to generate a full 16-bit address:
 - Take contents of base register, add 6 bits of sign-extended offset.
 - Why 6 bits of offset?
 - $6 = 16 - 4$ (opcode) - 3 (name of src/dest register) - 3 (name of base register)
 - Effective address = Base+offset
- LDR (Load using base Register)
 - $\text{Dst reg} = M[\text{Base reg} + \text{offset}]$
- STR (Store using base Register)
 - $M[\text{Base reg} + \text{offset}] = \text{Src reg}$



G. Summary of Instructions So Far

Data Operation

- Unary
 - NOT 1001 Dst Src1 111111
- Binary Immediate
 - ADD 0001 Dst Src1 1 Immed5
 - AND 0101 Dst Src1 1 Immed5
- Binary Register
 - ADD 0001 Dst Src1 0 00 Src2
 - AND 0101 Dst Src1 0 00 Src2

Data Movement

- PC Offset
 - LD 0010 Dst PCoffset9
 - ST 0011 Src PCoffset9
- Immediate
 - LEA 1110 Dst PCoffset9
- Indirect
 - LDI 1010 Dst PCoffset9
 - STI 1011 Src PCoffset9
- Base Register Offset
 - LDR 0110 Dst Base Offset6
 - STR 0111 Src Base Offset6

H. Examples

Addr	OpC	Instruction	Comments
x30F6	LEA	1110 001 111111101	$R1 \leftarrow PC-3 = x30F7-3 = x30F4$
x30F7	ADD	0001 010 001 1 01110	$R2 \leftarrow R1+14 = x30F4+14 = x3102$
x30F8	ST	0011 010 111111011	$M[PC-5] \leftarrow R2 \ // \ M[x30F4] \leftarrow x3102$
x30F9	AND	0101 010 010 1 00000	$R2 \leftarrow R2 \text{ AND } 0 (= 0)$
x30FA	ADD	0001 010 010 1 00101	$R2 \leftarrow R2+5 = 0+5 = 5$
x30FB	STR	0111 010 001 001110	$M[R1+14] \leftarrow R2 \ // \ M[x3102] \leftarrow 5$
x30FC	LDI	1010 011 111110111	$R3 \leftarrow M[M[PC-9]] = M[M[x30F4]] = M[x3102] = 5$