Data Movement Instructions

Load -- read data from memory to register
  • LD: PC-relative mode
  • LDR: base+offset mode
  • LDI: indirect mode

Store -- write data from register to memory
  • ST: PC-relative mode
  • STR: base+offset mode
  • STI: indirect mode

Load effective address -- compute address, save in register
  • LEA: immediate mode
  • does not access memory
PC-Relative Addressing Mode

Want to specify address directly in the instruction
  • But an address is 16 bits, and so is an instruction!
  • After subtracting 4 bits for opcode and 3 bits for register, we have 9 bits available for address.

Solution:
  • Use the 9 bits as a signed offset from the current PC.

9 bits: \(-256 \leq \text{offset} \leq +255\)

Can form any address \(X\), such that: \(\text{PC} - 256 \leq X \leq \text{PC} + 255\)

Remember that PC is incremented as part of the FETCH phase;
This is done before the EVALUATE ADDRESS stage.
LD (PC-Relative)

LD 0 0 1 0  Dst  PCoffset9

1. IR[8:0]
2. Sext
3. MAR
4. Register File

Copyright © The McGraw-Hill Companies, Inc. Permission required for reproduction or display.
ST (PC-Relative)

ST 0 0 1 1  Src  PCoffset9
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.
LDI (Indirect)

LDI 1010 Dst PCoffset9
STI (Indirect)

STI [1 0 1 1] Src PC offset 9
Base + Offset Addressing Mode

With PC-relative mode, can only address data within 256 words of the instruction.
  • What about the rest of memory?

Solution #2:
  • Use a register to generate a full 16-bit address.

4 bits for opcode, 3 for src/dest register, 3 bits for base register -- remaining 6 bits are used as a signed offset.
  • Offset is sign-extended before adding to base register.
LDR (Base + Offset)

LDR | 0 1 1 0 | Dst | Base | offset6

Diagram:

- Instruction Reg
- Sext
- IR[5:0]
- Register File
- Base
- MAR
- MDR
- Memory
**STR (Base+Offset)**

```
+-------+-------+-------+-------+-------+-------+-------+-------+
|       |       |       |       |       |       |       |       |
|       |       |       |       |       |       |       |       |
| 0     | 1     | 1     | 1     | Src   | Base  | offset6 |
+-------+-------+-------+-------+-------+-------+-------+
```

Diagram:
- Instruction Reg
- IR[5:0]
- Sext
- +/-
- MAR
- MDR
- Memory

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

Computes address like PC-relative (PC plus signed offset) and stores the result into a register.

Note: The address is stored in the register, not the contents of the memory location.
LEA (Immediate)

LEA 1 1 1 0 Dst PCoffset9

Instruction Reg

Register File

PC

IR[8:0]

Sext

+
## Example

<table>
<thead>
<tr>
<th>Address</th>
<th>Instruction</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>x30F6</td>
<td>1 1 1 0 0 0 1 1 1 1 1 1 1 1 0 1</td>
<td>R1 ← PC – 3 = x30F4</td>
</tr>
<tr>
<td>x30F7</td>
<td>0 0 0 1 0 1 0 0 0 1 1 0 1 1 1 0</td>
<td>R2 ← R1 + 14 = x3102</td>
</tr>
</tbody>
</table>
| x30F8   | 0 0 1 1 0 1 0 1 1 1 1 1 1 0 1 1 | M[PC - 5] ← R2  
M[x30F4] ← x3102 |
| x30F9   | 0 1 0 1 0 1 0 0 1 0 1 0 0 0 0 0 | R2 ← 0 |
| x30FA   | 0 0 0 1 0 1 0 0 1 0 1 0 0 1 0 1 | R2 ← R2 + 5 = 5 |
| x30FB   | 0 1 1 1 0 1 0 0 0 1 0 0 1 1 1 0 | M[R1+14] ← R2  
M[x3102] ← 5 |
| x30FC   | 1 0 1 0 0 1 1 1 1 1 1 1 1 0 1 1 1 | R3 ← M[M[x30F4]]  
R3 ← M[x3102]  
R3 ← 5 |
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)
Condition Codes

LC-3 has three condition code registers:

N -- negative
Z -- zero
P -- positive (greater than zero)

Set by any instruction that writes a value to a register (ADD, AND, NOT, LD, LDR, LDI, LEA)

Exactly one will be set at all times
  • Based on the last instruction that altered a register
Branch Instruction

Branch specifies one or more condition codes. If the set bit is specified, the branch is taken.

- PC-relative addressing:
  target address is made by adding signed offset (IR[8:0]) to current PC.

- Note: PC has already been incremented by FETCH stage.
- Note: Target must be within 256 words of BR instruction.

If the branch is not taken, the next sequential instruction is executed.