Move information between registers & memory

- Register ← Memory (Read) LOAD
- Memory ← Register (Write) STORE

* Source information remains unchanged, destination overwritten
Data movement instructions

Format:

O₃ O₂ O₁ O₀ R₂ R₁ R₀ A₈ A₇ A₆ A₅ A₄ A₃ A₂ A₁ A₀

Opcodes: Source / Address generation bits
           Destination

4 ways to interpret address gen. bits
1. Suppose data is some known distance (offset) from current PC:

* LD (load, PC-relative mode): \( DR \leftarrow M[PC + SEXT(\text{PC offset} + 9)] \)

Example: \( R2 \)

0010 010 11010111

Opcode (LD)
you are PC, go offset distance to get data
equivalent ST
2. Data is now far away from current PC and address is known offset:

* `LDR (load, base register + offset)`:

```
DR ← M[BaseR + SEXT(Offset6)]
```

Setcc

Example:
```
0110 001 010 011101
```

Opcode
(LDR)

Corresponding STR

Get data from friend's house
from offset distance from use, whose address I know
Corresponding STR
3. Data is located far from PC but address of data is located in offset distance from PC

* LDI (load, indirect mode):
* LDI (load, indirect node):

   \[ \text{DR} \leftarrow M[\text{M}[PC + \text{SEXT}(PC_{offset} 9)]] \]

   \[ \text{Setcc} \]

   Example: R3
   1010 011 111001100

   Opcode (LDI)

---

A friend ↔

address of the distance from MAR changes 2 times

① MAR \leftarrow PC + \text{SEXT}(PC_{offset} 9)

② MDR \leftarrow M[MAR]  \quad * \text{Note: MDR now has an address}

③ MAR \leftarrow MDR

④ MDR \leftarrow M[MAR]
knows address of data
and
my friend is offset
in where PC is.
Corresponding

STI
```plaintext
* LEA (load effective address):

\[ \text{DR} \leftarrow \text{PC} + \text{SEXT} (\text{PC offset}) \]

Example:

\[
\begin{array}{cccccccccccc}
0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 0 & 1
\end{array}
\]

Opcode

(LEA)

---

No memory access.

No Store

No memory access.
```
No memory access.
LC-3 control instructions

LC-3 INSTRUCTION SET - PART II
Control instructions

They change the sequence of execution of instructions by manipulating the value of the PC register.

* JMP (unconditional jump): load PC with content of register

\[
PC \leftarrow \text{BaseR}
\]

Example:

```
1100 000 011 000000
```

Oprcode (JMP)  R3

If R3 has value \( x_{5400} \) and PC is \( x_{3500} \) then this will result in instruction at \( x_{5400} \) being executed next.
* BR (conditional branch): if any of the relevant condition
BR (conditional branch); if any of the relevant condition codes occur then increment PC with PCoffset9

Notation:
- $n, z, p$: bits in instruction (lower case)
- $N, Z, P$: bits in condition codes (upper case)

If $(n \text{ AND } N) \text{ OR }(z \text{ AND } Z) \text{ OR }(p \text{ AND } P)$
then $PC \leftarrow PC + \text{SEXT} (\text{PCoffset9})$

Example:

```
x0D9
0000 111 011011001
```

...
Question: what happens if in the instruction \( n = z = p = 1 \)?

unconditional jump.
Introduction to programming in machine code

* Notation:

```
1111 0000 0010 0101 ; Comment sentence
Machine code, use
spaces for clarity
Comments are preceded by ;
```
Starting/Stopping program:

First line of program is initial address, not an instruction.

In LC-3, programs must be stored after x2FFF address.

0011 0000 0000 0000 ; Load program starting in x3000

Last line of program must stop computer

1111 0000 0010 0101 ; HALT instruction (TRAP x25)