Previously, we looked at some instruction bits and talked about executing instructions. It’s natural to wonder:

How did those bits get there?

Similarly, when making a peanut butter sandwich:

- Why was the bag closed?
- Where did the bread come from?
- Why was it whole wheat bread?

How Do We Write Instructions?

Put Bits into Memory, Then Execute the Bits

All perfectly valid questions, but be patient!

Our model of programming:

- Place bits into memory locations (you’ll see how in the lab, and later in class).
- Then tell the LC-3 to interpret our bits as instructions.

We can also put data bits in memory.

- But be careful!
- The LC-3 can’t tell the difference between instructions and data. Both are bits.

Put Bits into Memory, Then Execute the Bits

Let’s Illustrate LC-3 Instruction Processing

Let’s execute the LC-3 for a few cycles and see how it works.

We’ll show a few pieces of the datapath:

- memory
- register file
- PC and IR
- MAR and MDR
### What's in the Blank Boxes?

**Bits! (not blanks!)**

<table>
<thead>
<tr>
<th></th>
<th></th>
<th>R0</th>
<th>R1</th>
<th>R2</th>
<th>R3</th>
<th>R4</th>
<th>R5</th>
<th>R6</th>
<th>R7</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>0000</td>
<td>0001</td>
<td>...</td>
<td>123A</td>
<td>0000 1111 0000 1111</td>
<td>CPUD</td>
<td>...</td>
<td>xFFFF</td>
</tr>
</tbody>
</table>

**Register file**

- PC
- IR
- MAR
- MDR

**Memory**

| x3000 | x3001 | ... | x3000 0110 0111 0000 1010 | x670A |

### Why are Some Values in Hex?

For us humans (only). Computers always use bits.

### Fetch #1: MAR ← PC, PC ← PC + 1

<table>
<thead>
<tr>
<th></th>
<th></th>
<th>R0</th>
<th>R1</th>
<th>R2</th>
<th>R3</th>
<th>R4</th>
<th>R5</th>
<th>R6</th>
<th>R7</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>0000</td>
<td>0001</td>
<td>...</td>
<td>123A</td>
<td>0000 1111 0000 1111</td>
<td>CPUD</td>
<td>...</td>
<td>xFFFF</td>
</tr>
</tbody>
</table>

**Register file**

- PC
- IR
- MAR
- MDR

**Memory**

| x3000 | x3001 | ... | x3000 0110 0111 0000 1010 | x670A |

### Fetch #2: MDR ← M[MAR]

<table>
<thead>
<tr>
<th></th>
<th></th>
<th>R0</th>
<th>R1</th>
<th>R2</th>
<th>R3</th>
<th>R4</th>
<th>R5</th>
<th>R6</th>
<th>R7</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>0000</td>
<td>0001</td>
<td>...</td>
<td>123A</td>
<td>0000 1111 0000 1111</td>
<td>CPUD</td>
<td>...</td>
<td>xFFFF</td>
</tr>
</tbody>
</table>

**Register file**

- PC
- IR
- MAR
- MDR

**Memory**

| x3000 | x3001 | ... | x3000 0110 0111 0000 1010 | x670A |
Fetch #3: IR ← MDR

```
<table>
<thead>
<tr>
<th>16 bits</th>
<th>16 bits</th>
</tr>
</thead>
<tbody>
<tr>
<td>R0 bits</td>
<td>x0000</td>
</tr>
<tr>
<td>R1 bits</td>
<td>x0001</td>
</tr>
<tr>
<td>R2 bits</td>
<td>...</td>
</tr>
<tr>
<td>R3 bits</td>
<td>x123A</td>
</tr>
<tr>
<td>R4 bits</td>
<td>0001 0010 0011 0000 x1230</td>
</tr>
<tr>
<td>R5 bits</td>
<td>x3000</td>
</tr>
<tr>
<td>R6 bits</td>
<td>...</td>
</tr>
<tr>
<td>R7 bits</td>
<td>xFFFF</td>
</tr>
</tbody>
</table>
```

Let's Decode the Instruction

The IR has x670A. In bits, that’s

```
0 1 1 1 0 0 1 1 1 0 0 0 1 0 1 0
```

LDR R3, R4, x0A

Which means what? Let’s decode it.

R3 ← M[R4 + x000A]

What is the memory address?

DECODE, then EXECUTE

These bits are the instruction to be decoded and executed.

The LC-3 Reads the Bits from R4

Look at the bits stored in R4.
Let's Calculate the Memory Address

\[ R3 \leftarrow M[R4 + x000A] \]

R4 is x1230.
Adding x000A, we obtain ... ? x123A

What is stored at memory address x123A?

The LC-3 Reads Memory at x123A

Here is address x123A.

The LC-3 Stores x0F0F into R3

\[ R3 \leftarrow M[R4 + x000A] \]

R4 is x1230.
Adding x000A, we obtain ... ? x123A

What is stored at memory address x123A?

x0F0F
So the LC-3 stores x0F0F into R3.
What’s Next?

- 16 bits - | 16 bits -
---|---
R0 | \( \times 000 \) |
R1 | \( \times 0001 \) |
R2 | \( \times 0F0F \) |
R3 | \( 0000 \times 0001 \times 123A \times 0F0F \) |
R4 | \( 0001 \times 0010 \times 0000 \times 1230 \) |
R5 | \( \times 0000 \times 0110 \times 0111 \times 0100 \times 670A \) |
R6 | \( \times FFFF \) |
R7 | \( \times FFFF \) |

register file | memory
---|---
PC | x3001 |
IR | x670A |
MAR | x3000 |
MDR | x670A |

Fetch another instruction!

Fetch #1: MAR ← PC, PC ← PC + 1

- 16 bits - | 16 bits -
---|---
R0 | \( \times 000 \) |
R1 | \( \times 0001 \) |
R2 | \( \times 0F0F \) |
R3 | \( 0000 \times 0000 \times 0000 \times 0000 \times 0000 \times 0F0F \) |
R4 | \( 0001 \times 0010 \times 0000 \times 0000 \times 1230 \) |
R5 | \( \times 0000 \times 0110 \times 0111 \times 0100 \times 670A \) |
R6 | \( \times FFFF \) |
R7 | \( \times FFFF \) |

register file | memory
---|---
PC | x3002 |
IR | x670A |
MAR | x3001 |
MDR | x670A |

Fetch #2: MDR ← M[MAR]

- 16 bits - | 16 bits -
---|---
R0 | \( \times 000 \) |
R1 | \( \times 0001 \) |
R2 | \( \times 0F0F \) |
R3 | \( 0000 \times 0000 \times 0000 \times 0000 \times 0000 \times 0F0F \) |
R4 | \( 0001 \times 0010 \times 0000 \times 0000 \times 1230 \) |
R5 | \( \times 0000 \times 0110 \times 0111 \times 0100 \times 670A \) |
R6 | \( \times FFFF \) |
R7 | \( \times FFFF \) |

register file | memory
---|---
PC | x3002 |
IR | x670A |
MAR | x3001 |
MDR | x670A |

Fetch #3: IR ← MDR

- 16 bits - | 16 bits -
---|---
R0 | \( \times 000 \) |
R1 | \( \times 0001 \) |
R2 | \( \times 0F0F \) |
R3 | \( 0000 \times 0000 \times 0000 \times 0000 \times 0000 \times 0F0F \) |
R4 | \( 0001 \times 0010 \times 0000 \times 0000 \times 1230 \) |
R5 | \( \times 0000 \times 0110 \times 0111 \times 0100 \times 670A \) |
R6 | \( \times FFFF \) |
R7 | \( \times FFFF \) |

register file | memory
---|---
PC | x3002 |
IR | \( \times 16C3 \) |
MAR | x3001 |
MDR | \( \times 16C3 \)
**DECODE, then EXECUTE**

These bits are the instruction to be decoded and executed.

**Let's Decode the Instruction**

The IR has \textbf{x16C3}. In bits, that's:

<table>
<thead>
<tr>
<th>bits</th>
<th>bits</th>
<th>bits</th>
<th>bits</th>
<th>bits</th>
<th>bits</th>
</tr>
</thead>
<tbody>
<tr>
<td>R0</td>
<td>R1</td>
<td>R2</td>
<td>R3</td>
<td>R4</td>
<td>R5</td>
</tr>
<tr>
<td>0001 1001 1000 1001 1001 1001</td>
<td>10 1111 0000 1010</td>
<td>\textbf{x0F0F}</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Which means what? Let's decode it.

R3 ← R3 + R3

Add R3 to R3, storing the sum back into R3.

**The LC-3 Stores x1E1E into R3**

The LC-3 Stores \textbf{x1E1E} into R3.

**What’s Next?**

What’s Next?

Fetch another instruction!

Life is not so exciting for LC-3 processors...
Fetch #1: MAR ← PC, PC ← PC + 1

- 16 bits -

| 0001 | 0010 0011 0000 | x1230 |
|-----|-----------------|
| 0001 | 1110 0001 1110 | x123A |
|     |                | x0000 |
|     |                | x0001 |
|     |                | x0002 |
|     |                | x0003 |

register file memory

| PC | x3003 | IR | x16C3 | MAR | x3002 | MDR | x16C3 |

Fetch #2: MDR ← M[MAR]

- 16 bits -

| 0001 | 0010 0011 0000 | x1230 |
|-----|-----------------|
| 0001 | 1110 0001 1110 | x123A |
|     |                | x0000 |
|     |                | x0001 |
|     |                | x0002 |
|     |                | x0003 |

register file memory

| PC | x3003 | IR | x16C3 | MAR | x3002 | MDR | x16C3 |

Fetch #3: IR ← MDR

- 16 bits -

| 0001 | 0010 0011 0000 | x1230 |
|-----|-----------------|
| 0001 | 1110 0001 1110 | x123A |
|     |                | x0000 |
|     |                | x0001 |
|     |                | x0002 |
|     |                | x0003 |

register file memory

| PC | x3003 | IR | x16C3 | MAR | x3002 | MDR | x16C3 |

DECODE, then EXECUTE

- 16 bits -

| 0001 | 0010 0011 0000 | x1230 |
|-----|-----------------|
| 0001 | 1110 0001 1110 | x123A |
|     |                | x0000 |
|     |                | x0001 |
|     |                | x0002 |
|     |                | x0003 |

register file memory

| PC | x3003 | IR | x16C3 | MAR | x3002 | MDR | x16C3 | x770A |

These bits are the instruction to be decoded and executed.
Let’s Decode the Instruction

The IR has $x^{770A}$. In bits, that’s

<table>
<thead>
<tr>
<th>opcode</th>
<th>source register</th>
<th>base register</th>
<th>6-bit 2’s complement offset</th>
</tr>
</thead>
<tbody>
<tr>
<td>0111</td>
<td>011</td>
<td>100</td>
<td>001010</td>
</tr>
</tbody>
</table>

STR, R3, R4, x0A

Which means what? Let’s decode it.

$M[R4 + x^{000A}] \leftarrow R3$

What is the memory address?

Let’s Calculate the Memory Address

$M[R4 + x^{000A}] \leftarrow R3$

R4 is $x^{1230}$.

Adding $x^{000A}$, we obtain ... ? $x^{123A}$

So the LC-3 stores the bits in R3 to memory address $x^{123A}$.

The LC-3 Reads the Bits from R4

The LC-3 Writes $x^{1E1E}$ to Memory Address $x^{123A}$
What’s Next?

For the LC-3, fetch another instruction!

But as for us, we’re done.

Computers Just Execute Instructions

What does that instruction sequence do?
Multiplies the value at address x123A by 2 (shifts it left by 1 bit).

What if R3 held something important before we executed those instructions?
Too bad. Those bits are gone.
The programmer controls the computer.
The computer just does what it’s told.