

University of Illinois at Urbana-Champaign
Dept. of Electrical and Computer Engineering

ECE 120: Introduction to Computing

Instructions Illustrated

How Do We Write Instructions?

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?

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.

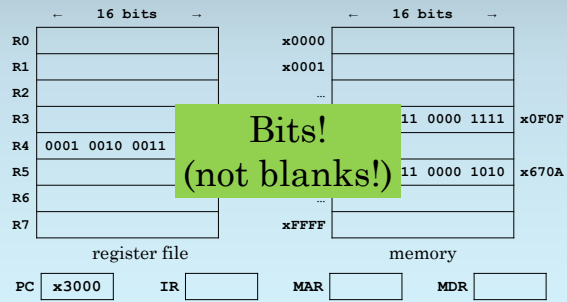
Let's Illustrate LC-3 Instruction Processing

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

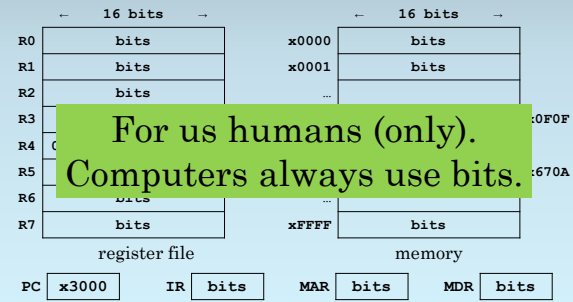
We'll show a few pieces of the datapath:

- memory
- register file
- **PC** and **IR**
- **MAR** and **MDR**

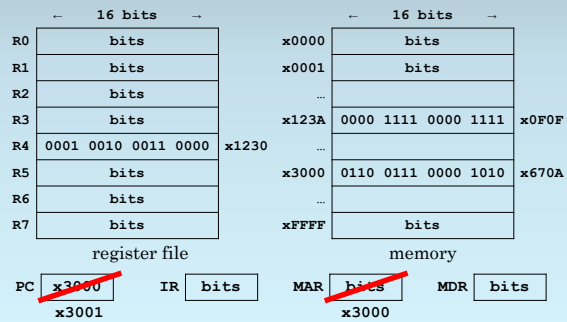
What's in the Blank Boxes?



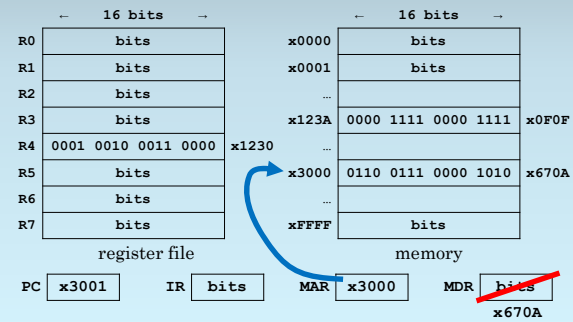
Why are Some Values in Hex?



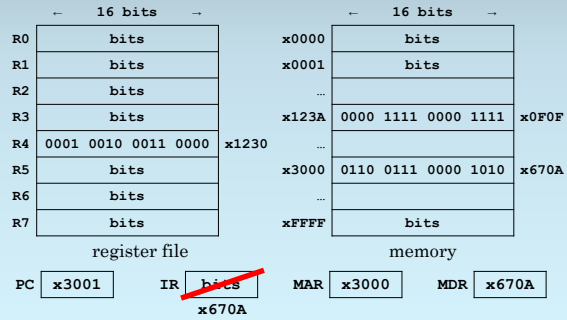
Fetch #1: MAR ← PC, PC ← PC + 1



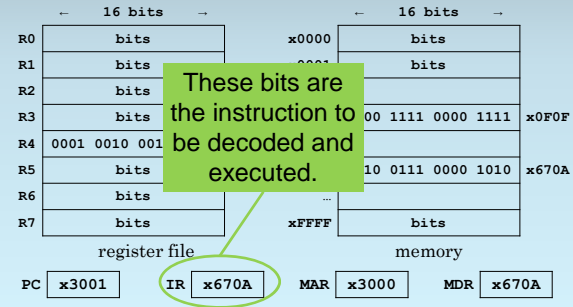
Fetch #2: MDR ← M[MAR]



Fetch #3: IR ← MDR

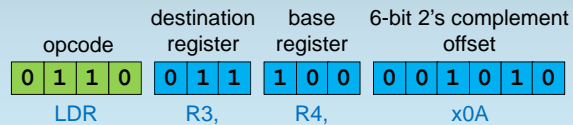


DECODE, then EXECUTE



Let's Decode the Instruction

The IR has **x670A**. In bits, that's

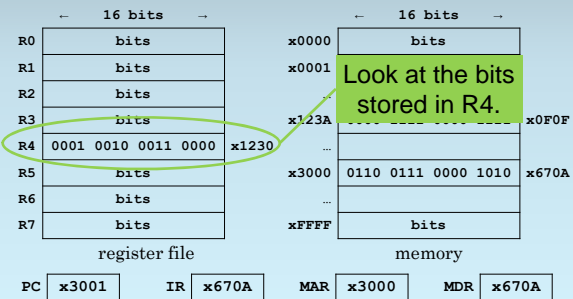


Which means what? Let's decode it.

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

What is the memory address?

The LC-3 Reads the Bits from R4



Let's Calculate the Memory Address

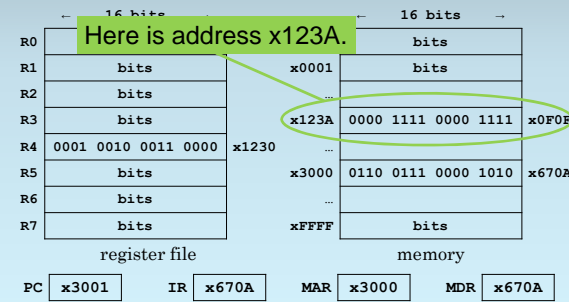
$$R3 \leftarrow M[R4 + x000A]$$

R4 is x1230.

Adding x000A, we obtain ... ? x123A

What is stored at memory address x123A?

The LC-3 Reads Memory at x123A



The LC-3 Stores x0F0F into R3

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

R4 is x1230.

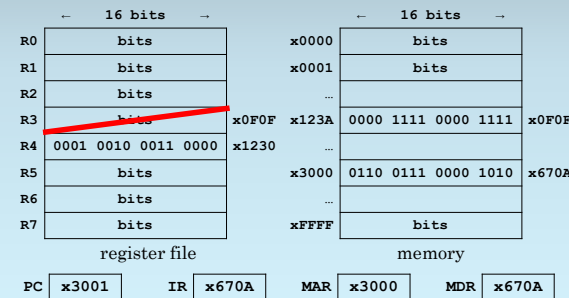
Adding x000A, we obtain ... ? x123A

What is stored at memory address x123A?

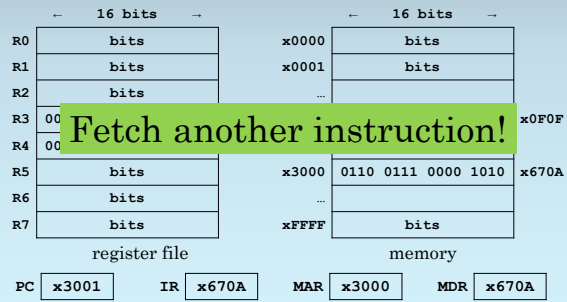
x0F0F

So the LC-3 stores x0F0F into R3.

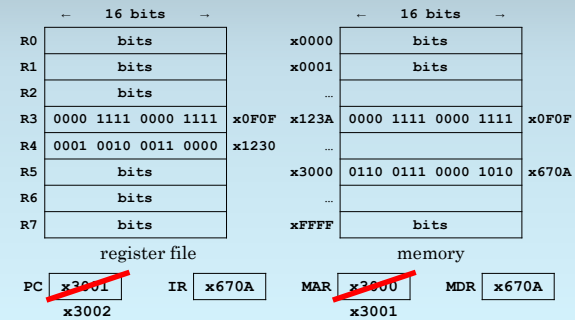
The LC-3 Stores x0F0F into R3



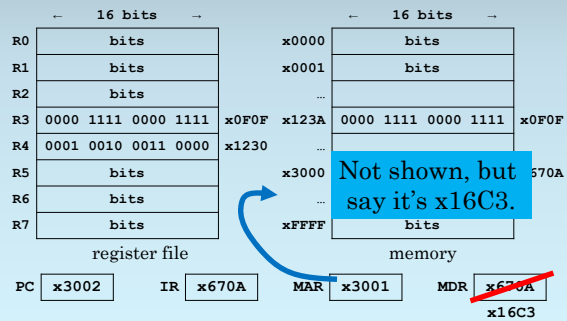
What's Next?



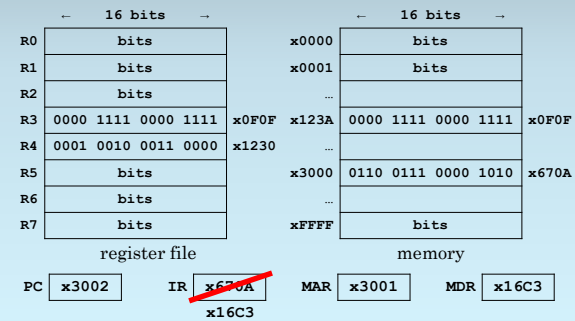
Fetch #1: MAR ← PC, PC ← PC + 1



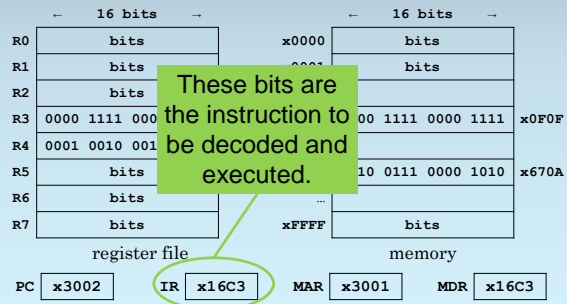
Fetch #2: MDR ← M[MAR]



Fetch #3: IR ← MDR

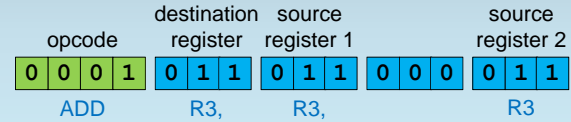


DECODE, then EXECUTE



Let's Decode the Instruction

The IR has **x16C3**. In bits, that's

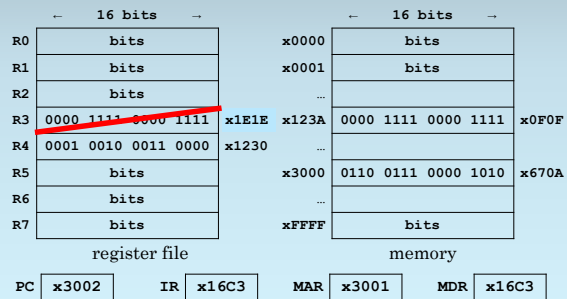


Which means what? Let's decode it.

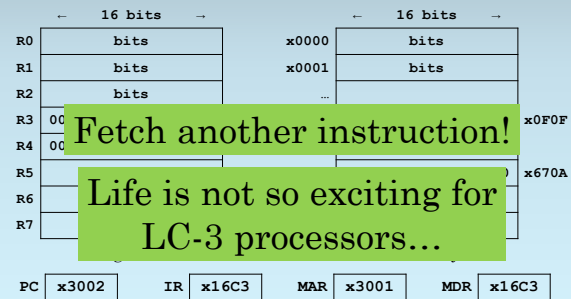
$$R3 \leftarrow R3 + R3$$

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

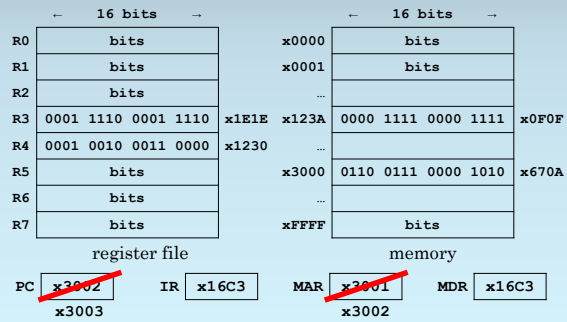
The LC-3 Stores x1E1E into R3



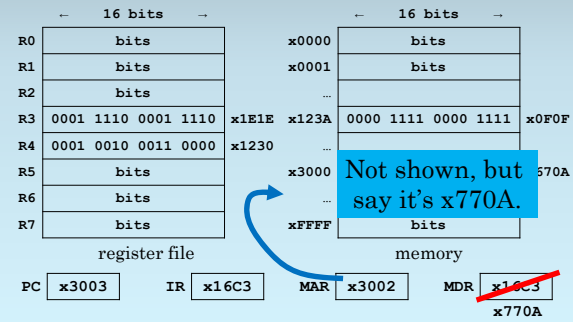
What's Next?



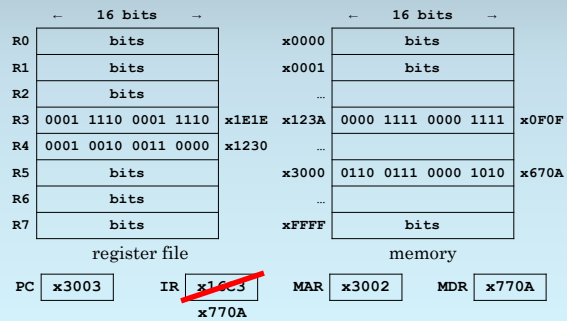
Fetch #1: MAR ← PC, PC ← PC + 1



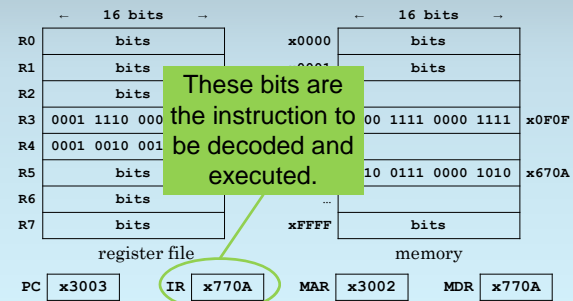
Fetch #2: MDR ← M[MAR]



Fetch #3: IR ← MDR

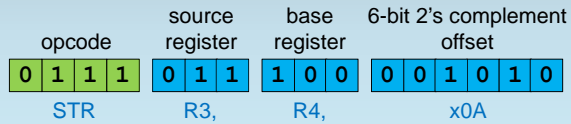


DECODE, then EXECUTE



Let's Decode the Instruction

The **IR** has **x770A**. In bits, that's

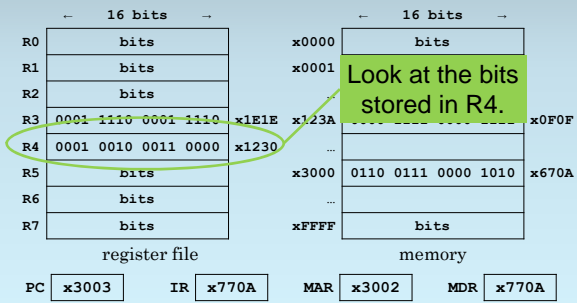


Which means what? Let's decode it.

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

What is the memory address?

The LC-3 Reads the Bits from R4



Let's Calculate the Memory Address

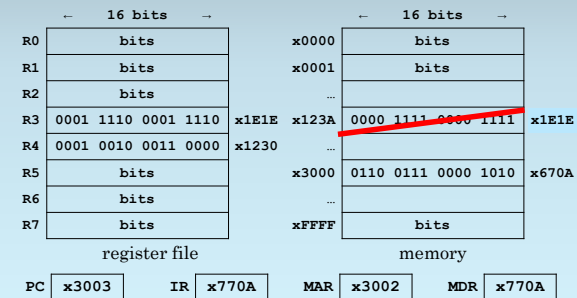
$$M[R4 + x000A] \leftarrow R3$$

R4 is **x1230**.

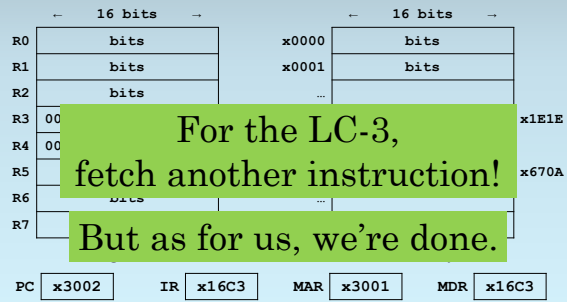
Adding **x000A**, we obtain ... ? **x123A**

So the LC-3 stores the bits in R3 to memory address **x123A**.

The LC-3 Writes x1E1E to Memory Address x123A



What's Next?



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.**