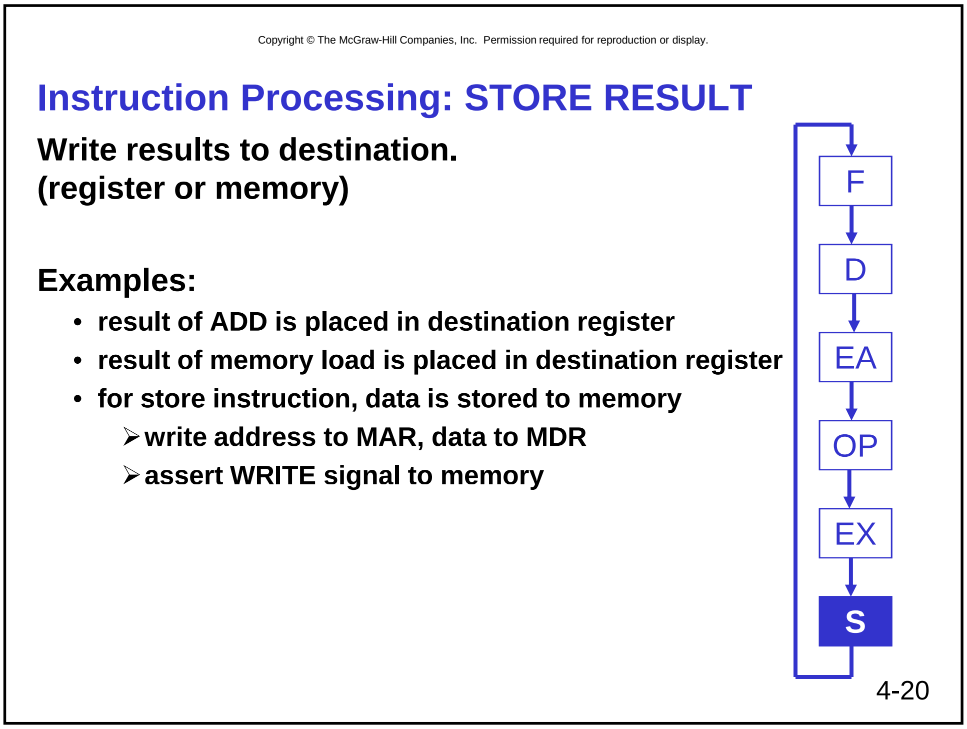
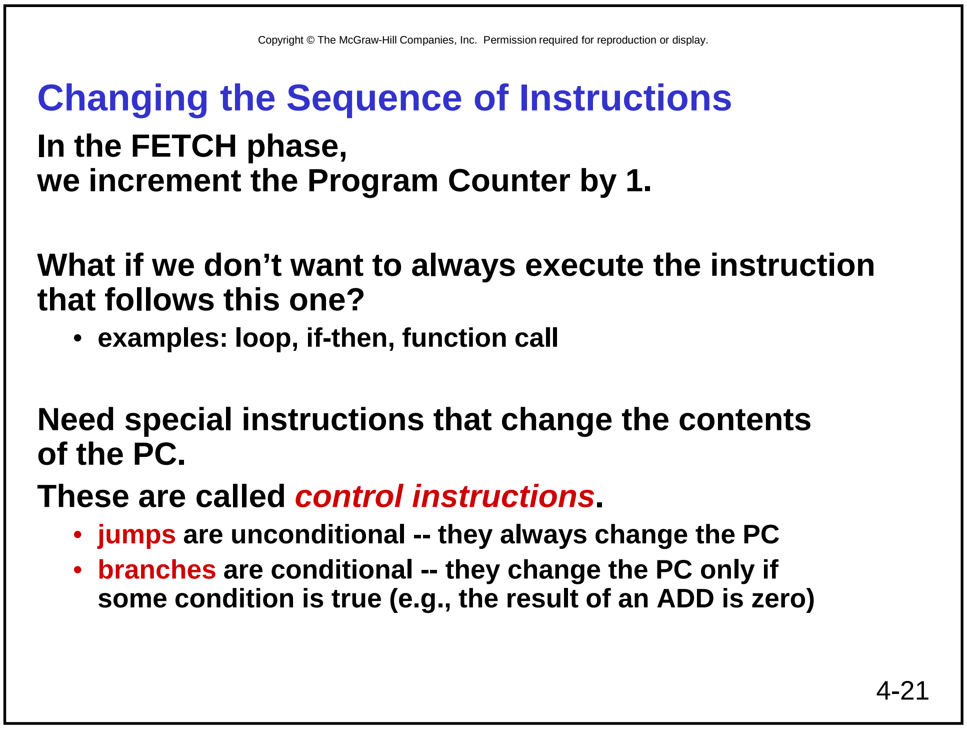


Instruction Processing: STORE RESULT

Write results to destination.
(register or memory)

Examples:

- **result of ADD is placed in destination register**
- **result of memory load is placed in destination register**
- **for store instruction, data is stored to memory**
  - ➢ **write address to MAR, data to MDR**
  - ➢ **assert WRITE signal to memory**



Changing the Sequence of Instructions

In the FETCH phase,
we increment the Program Counter by 1.

What if we don’t want to always execute the instruction that follows this one?

- **examples: loop, if-then, function call**

Need special instructions that change the contents of the PC.

These are called *control instructions*.

- **jumps are unconditional -- they always change the PC**
- **branches are conditional -- they change the PC only if some condition is true (e.g., the result of an ADD is zero)**

Example: LC-3 JMP Instruction

Set the PC to the value contained in a register. This becomes the address of the next instruction to fetch.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
JMP				0	0	0	Base			0	0	0	0	0	0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	0	0	0	0	0	0	1	1	0	0	0	0	0	0

“Load the contents of R3 into the PC.”

Instruction Cycle – start over

- **Start over …**
  - **The control unit just keeps repeating this whole process: so it now Fetches a new instruction from the address currently stored in the PC.**
    - **Recall that the PC was incremented in the first step (FETCH), so the instruction retrieved will be the next in the program as stored in memory - unless the instruction just executed changed the contents of the PC.**

Instruction Processing Summary

Instructions look just like data -- it's all interpretation.

Three basic kinds of instructions:

- **computational instructions (ADD, AND, …)**
- **data movement instructions (LD, ST, …)**
- **control instructions (JMP, BRnz, …)**

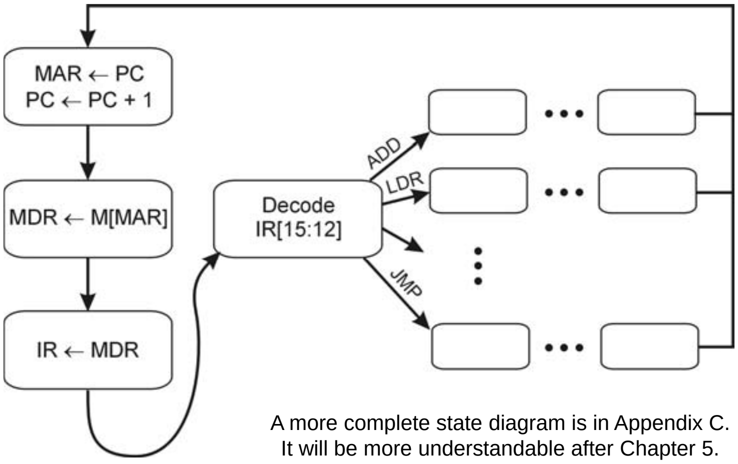
Six basic phases of instruction processing:

$$F \rightarrow D \rightarrow EA \rightarrow OP \rightarrow EX \rightarrow S$$

- **not all phases are needed by every instruction**
- **phases may take variable number of machine cycles**

Control Unit State Diagram

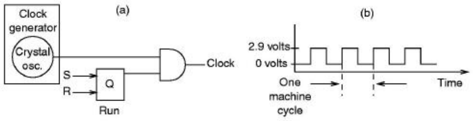
The control unit is a state machine. Here is part of a simplified state diagram for the LC-3:



A more complete state diagram is in Appendix C.
It will be more understandable after Chapter 5.

Stopping the clock

- "User" programs terminate simply by handing control back to the Operating System (OS):
  - The OS then enters a "waiting" loop until a new program is run.
  - The Control Unit is still actively stepping through the instruction cycle.
- Left to itself, the control unit would just keep fetching instructions from memory … until we pull the plug!
  - We can cease processing completely by stopping the machine cycle - i.e. by stopping the clock.
    - ➢ AND the clock generator signal with ZERO
    - ➢ When control unit stops seeing the CLOCK signal, it stops processing.


(a)
Clock generator
Crystal osc.
S
R
Q
Run
Clock
(b)
2.9 volts
0 volts
One machine cycle
Time