

Computer Science 210 s1c
Computer Systems 1
2008 Semester 1
Lecture Notes

Lecture 11, 31Mar08:
The von Neumann Computer

James Goodman



Credits: Slides prepared by Gregory T. Byrd, North Carolina State University

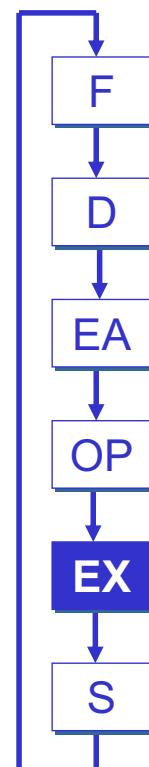
Copyright © The McGraw-Hill Companies, Inc. Permission required for reproduction or display.

Instruction Processing: EXECUTE

Perform the operation,
using the source operands.

Examples:

- send operands to ALU and assert ADD signal
- do nothing (e.g., for loads and stores)

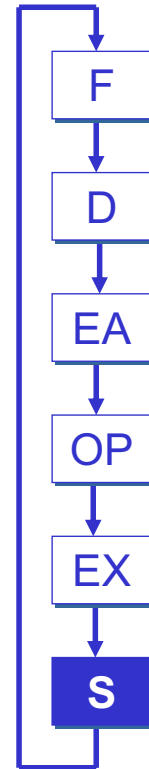


Instruction Processing: STORE RESULT

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 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, ...)

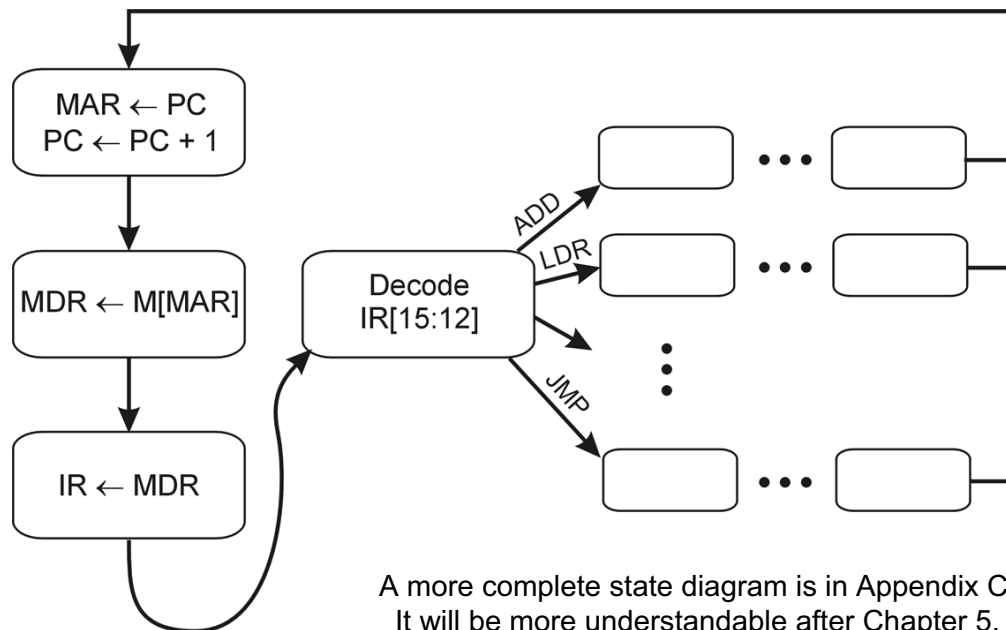
Six basic phases of instruction processing:

F → D → EA → OP → EX → 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:



31-Mar-08

CS210

202

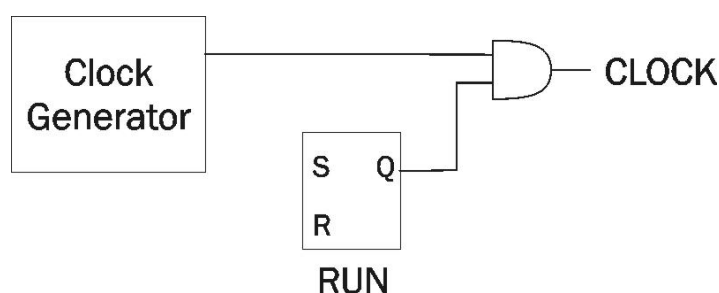
Stopping the Clock

Control unit will repeat instruction processing sequence as long as clock is running.

- If not processing instructions from your application, then it is processing instructions from the Operating System (OS).
- The OS is a special program that manages processor and other resources.

To stop the computer:

- AND the clock generator signal with ZERO
- When control unit stops seeing the CLOCK signal, it stops processing.



31-Mar-08

CS210

203

An Example ISA: The LC-3

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

Instruction Set Architecture

ISA = All of the *programmer-visible* components and operations of the computer

- **memory organization**
 - address space -- how many locations can be addressed?
 - addressability -- how many bits per location?
- **register set**
 - how many? what size? how are they used?
- **instruction set**
 - opcodes
 - data types
 - addressing modes

ISA provides all information needed for someone that wants to write a program in **machine language** (or translate from a high-level language to machine language).

LC-3 Overview: Memory and Registers

Memory

- address space: 2^{16} locations (16-bit addresses)
- addressability: 16 bits

Registers

- temporary storage, accessed in a single machine cycle
 - accessing memory generally takes longer than a single cycle
- eight general-purpose registers: R0 - R7
 - each 16 bits wide
 - how many bits to uniquely identify a register?
- other registers
 - not directly addressable, but used by (and affected by) instructions
 - PC (program counter), condition codes

LC-3 Overview: Instruction Set

Opcodes

- 15 opcodes
- **Operate** instructions: ADD, AND, NOT
- **Data movement** instructions: LD, LDI, LDR, LEA, ST, STR, STI
- **Control** instructions: BR, JSR/JSRR, JMP, RTI, TRAP
- some opcodes set/clear *condition codes*, based on result:
 - N = negative, Z = zero, P = positive (> 0)

Data Types

- 16-bit 2's complement integer

Addressing Modes

- How is the location of an operand specified?
- non-memory addresses: *immediate*, *register*
- memory addresses: *PC-relative*, *indirect*, *base+offset*

Operate Instructions

Only three operations: **ADD, AND, NOT**

Source and destination operands are **registers**

- These instructions **do not** reference memory.
- ADD and AND can use “immediate” mode, where one operand is hard-wired into the instruction.

Will show **dataflow diagram** with each instruction.

- illustrates **when** and **where** data moves to accomplish the desired operation