

# Computer Science 1000: Part #5

## Computer Organization

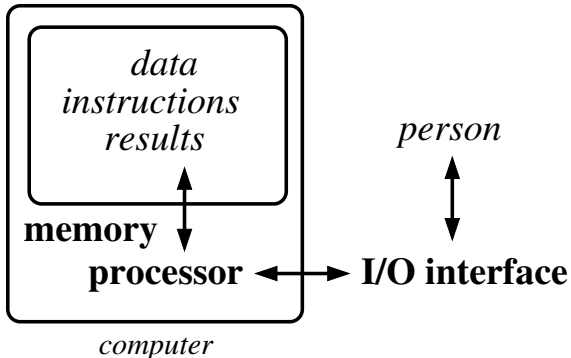
THE VON NEUMANN ARCHITECTURE:  
AN OVERVIEW

COMPUTER MEMORY

COMPUTER PROCESSOR

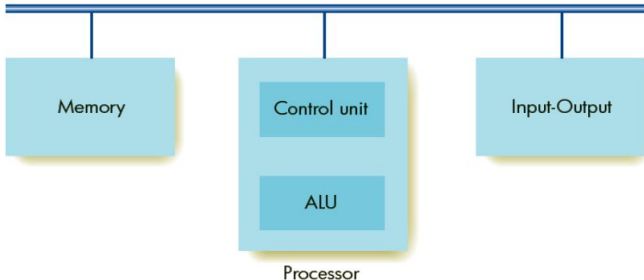
IMPLEMENTING COMPUTERS

# The Von Neumann Architecture: An Abstract View



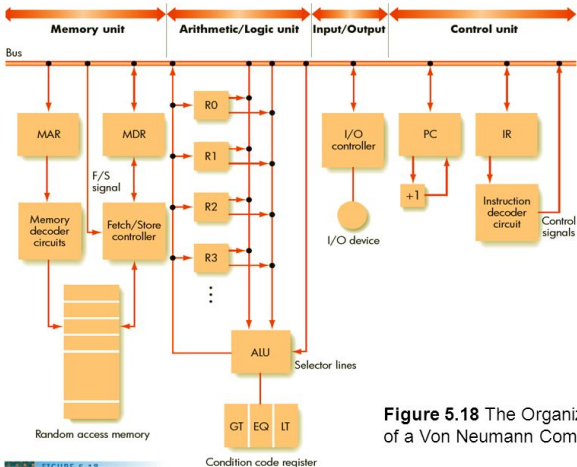
Also known as the stored-program architecture

# The Von Neumann Architecture: A More Detailed View



**Figure 5.2** Components of the Von Neumann Architecture

# The Von Neumann Architecture: A Really Detailed View



**Figure 5.18** The Organization of a Von Neumann Computer

# The Von Neumann Architecture: An Operational View

## The Von Neumann Execution Cycle:

repeat

Fetch next instruction

Decode instruction

Execute instruction

# Computer Memory :

## Overview

- Focus here on (volatile) **Random-Access Memory (RAM)**, cf. (non-volatile) **Read-Only Memory (ROM)**.
- Three characteristics of RAM:
  1. Divided into fixed-width **cells**, each of which has a unique unsigned-integer **address**  $0, 1, 2, \dots, MAX$  (**address space**).
  2. The cell is the minimal unit of fetch / store access.
  3. All cells have the same access time.
- Crucial to distinguish a memory address and the contents of memory at a particular address, e.g.,

address  $\implies$   $5743_{10}$ :  $-29_{10}$   $\longleftarrow$  contents

## Computer Memory : Overview (Cont'd)

- Standard cell-width  $W = 8$  bits (**byte**); standard address = 32 or 64 bits; standard access time  $\approx 5$ -10 nanoseconds.
- Memory size stated in terms of number of bytes:

Kilobyte	(KB)	= $10^3$ (thousand) bytes
Megabyte	(MB)	= $10^6$ (million) bytes
Gigabyte	(GB)	= $10^9$ (billion) bytes
Terabyte	(TB)	= $10^{12}$ (trillion) bytes
Petabyte	(PB)	= $10^{15}$ (quadrillion) bytes
Exabyte	(EB)	= $10^{18}$ (quintillion) bytes
Zettabyte	(ZB)	= $10^{21}$ (sextillion) bytes
		⋮

## Computer Memory : Overview (Cont'd)

- All communication done via the **Memory Address Register (MAR)** and the **Memory Data Register (MDR)**.
- Two basic operations:
  - Fetch(address):
    1. Load address into MAR
    2. Decode address in MAR
    3. Copy cell contents at address into MDR
  - Store(address, value):
    1. Load address into MAR
    2. Load value into MDR
    3. Decode address in MAR
    4. Copy MDR value into addressed cell



# Computer Memory : Internal Structure (Abstract)

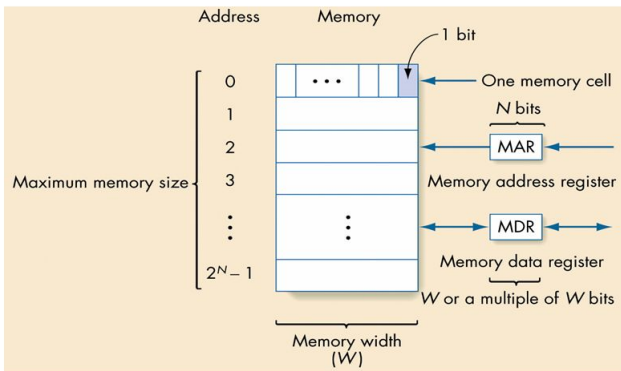


Figure 5.3  
Structure of Random Access Memory

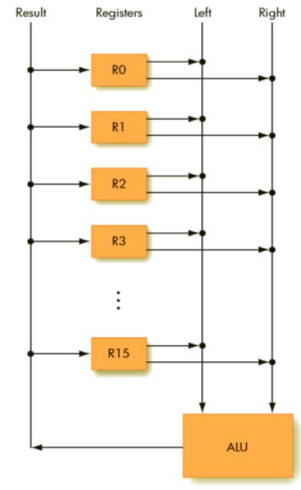
# Computer Processor: Overview

- Two main parts:
  1. **Arithmetic Logic Unit (ALU):** Performs arithmetic and logical operations.
  2. **Control Unit:** Handles interpretation and execution of program instructions. This involves directing the operations of the ALU and memory as well as interacting with the I/O controller.
- Both the ALU and the Control Unit have their own associated groups of special-purpose registers associated with their internal operations.

# Computer Processor: The Arithmetic Logic Unit (ALU)

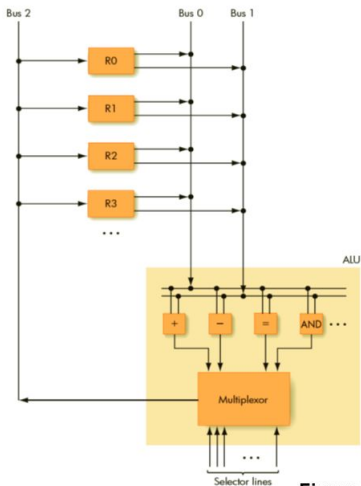
- Two types of ALU registers:
  1. **Value Registers (R0, R1, R2, ...)**: A set of 16–128 registers which contain data for current and upcoming operations as well as intermediate results.
  2. **Condition Code Register (CCR)**: A collection of bits specifying the results (1 if true, 0 if false) of the most recently executed value comparison, e.g., LT (less-than), EQ (equal-to), GT (greater-than).
- Value registers communicate with memory and the ALU and can be specified as either the left or right operand.
- The CCR communicates with the ALU, which passes the value of any condition-bit as requested to the control unit.

# Computer Processor: The Arithmetic Logic Unit (ALU) (Cont'd)



**Figure 5.11** Multiregister  
ALU Organization

# Computer Processor: The Arithmetic Logic Unit (ALU) (Cont'd)



**Figure 5.13** Overall ALU Organization

# Computer Processor: The Control Unit

- Two Control Unit registers:
  1. **Program Counter (PC):** Holds address in memory of next instruction to be executed.
  2. **Instruction Register (IR):** Holds the current instruction being executed. This includes not only the op-code ( $IR_{op}$ ) but the addresses of the instruction operands ( $IR_{add}$ , e.g., memory / ALU value registers).
- Instruction decoder circuitry uses the the  $k$ -bit opcode in the instruction in the IR to specify the appropriate one of the  $2^k$  signals to that instruction's execution circuitry and/or other computer components.

# Computer Processor: The Control Unit (Cont'd)

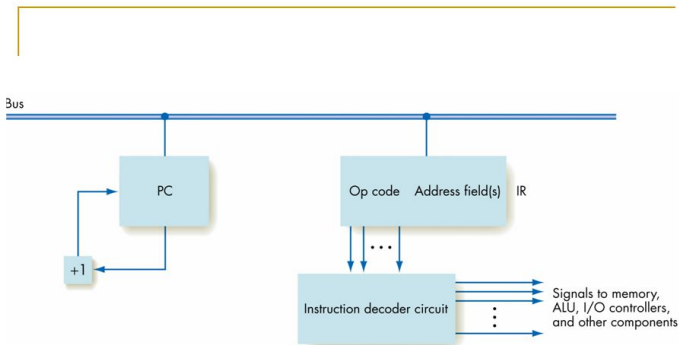


Figure 5.16  
Organization of the Control Unit Registers and Circuits

## Computer Processor: Machine Language

- An instruction = op-code + 0–3 address fields, e.g.,

op-code	address-1	address-2
000101	000000110011	000010001100
COMPARE	Addr1	Addr2

- Is part of either **Reduced or Complex Instruction Set Computer (RISC / CISC)** machine language; differ in tradeoff of required hardware vs. resulting program size.



# Computer Processor: Machine Language (Cont'd)

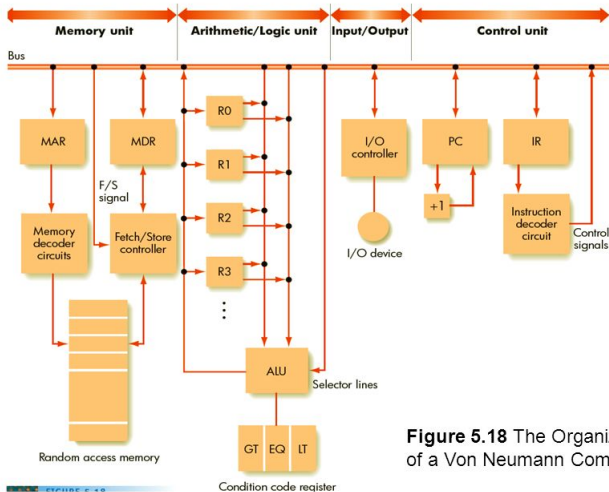
Four types of machine language instructions:

1. **Data Transfer:** Move values between memory cells and/or ALU registers, e.g., `LOAD Addr1, LOAD Addr2, MOVE Addr1 Addr2.`
2. **Arithmetic:** Perform arithmetic / logical operations on values in memory cells and/or ALU registers, e.g., `ADD Addr1 Addr2 Addr3, ADD Addr1 Addr2.`
3. **Comparison:** Compare two values and set CCR bits, e.g., `COMPARE Addr1 Addr2.`
4. **Branch:** Alter next instruction to be executed (often on basis of preceding comparison), e.g., `JUMP Addr1, JUMPGT Addr1, HALT.`

## Computer Processor: Machine Language (Cont'd)

	100	value of $a$
	101	value of $b$
	102	value of $c$
	...	
set $a$ to value of $b + c$	50	LOAD 101
	51	ADD 102
	52	STORE 100
	...	
if $a > b$ then	60	COMPARE 100 101
set $c$ to value of $a$	61	JUMPGT 64
else	62	MOVE 101 102
set $c$ to value of $b$	63	JUMP 65
	64	MOVE 100 102
	65	...

# The Von Neumann Architecture: A Detailed View Redux



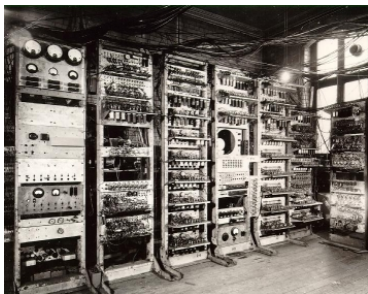
**Figure 5.18** The Organization of a Von Neumann Computer

# The Von Neumann Architecture: An Operational View Redux

The Von Neumann Execution Cycle:

```
while no HALT or fatal error do
  Fetch next instruction
  Decode instruction
  Execute instruction
```

## Implementing Computers: Beginnings



SSEM ("Baby")  
(1948, U. Manchester)



EDSAC  
(1949, U. Cambridge)

Though the Zuse Z3 was the first operational stored-program computer, SSEM and EDSAC were world's first operational electronic stored-program computers.

# Implementing Computers: Mainframes

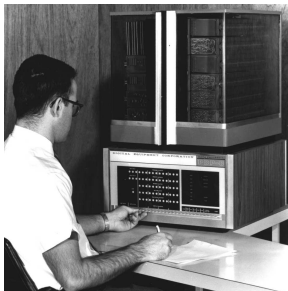


IBM System/360 (1967)

# Implementing Computers: Minicomputers



PDP I (1960)



PDP 8 (1965)

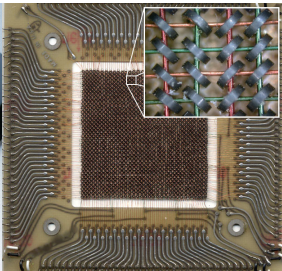
## Implementing Computers: Memory



Magnetic  
tape  
(1951)



Magnetic disk  
(1956)



Magnetic core  
(1953)

Massive cheap transistor-based storage possible in 1990s.



# Implementing Computers: I/O Interfaces



Punch card / tape  
(1940s)



Teletype  
(1940s)



CRT Display  
(1940s)

# Implementing Computers: Microprocessors

*Instead of being a little mainframe, the PC is, in fact, more like an incredibly big chip. – Robert X. Cringely*

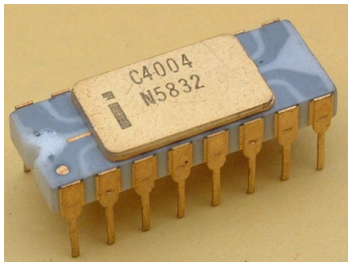


Image courtesy of CPU-Zone.com. Used with permission.



The microprocessor was invented by Ted Hoff in 1971.

# Implementing Computers: Microcomputers



IBM PC (1981) [\$2880]



Apple Macintosh (1984)  
[\$2500]

## Implementing Computers: Microcomputers (Cont'd)



First Portable Personal Computer: Osborne I (1981) [\$1795]

## Implementing Computers: Microcomputers (Cont'd)

www.oid-computers.com



- First true “laptop” PCs (GRiD Compass 1101 (1982) [\$8K]; see above left) appear in 1980s, *cf.*, portable “desktop” PCs like the Osborne I; expense of display and memory technologies limits market severely.
- Laptops finally surpass desktops in sales in 2008.

## Implementing Computers: Microcomputers (Cont'd)



- Hand-held personal computing appears first as Personal Digital Assistants (PDAs) in early 1990s.
- Early PDAs (Palm Pilot, Newton) were typically too expensive and based on technologies of limited user interest, e.g., handwriting recognition.
- Second-generation PDAs achieve success among business and government users when combined with basic secure messaging abilities, e.g., Blackberry (1999).

## Implementing Computers: Microcomputers (Cont'd)



- The convergence of hand-held multimedia-enabled computing and communication technology has resulted in tablet computers and smartphones; the former is preferable for screen size and the latter for device size.

# Implementing Computers: Non-Von Neumann Architectures



CM-2 (1987)



DWAVE 2000Q (2017)

Based on massively parallel instruction execution by multiple processors (CM-2) or quantum entanglement (DWAVE 2000Q).



## ... And If You Liked This ...

- MUN Computer Science courses on this area:
  - COMP 2003: Computer Architecture
  - COMP 4723: Introduction to Microprocessors
- MUN Computer Science professors teaching courses / doing research in in this area:
  - Rod Byrne