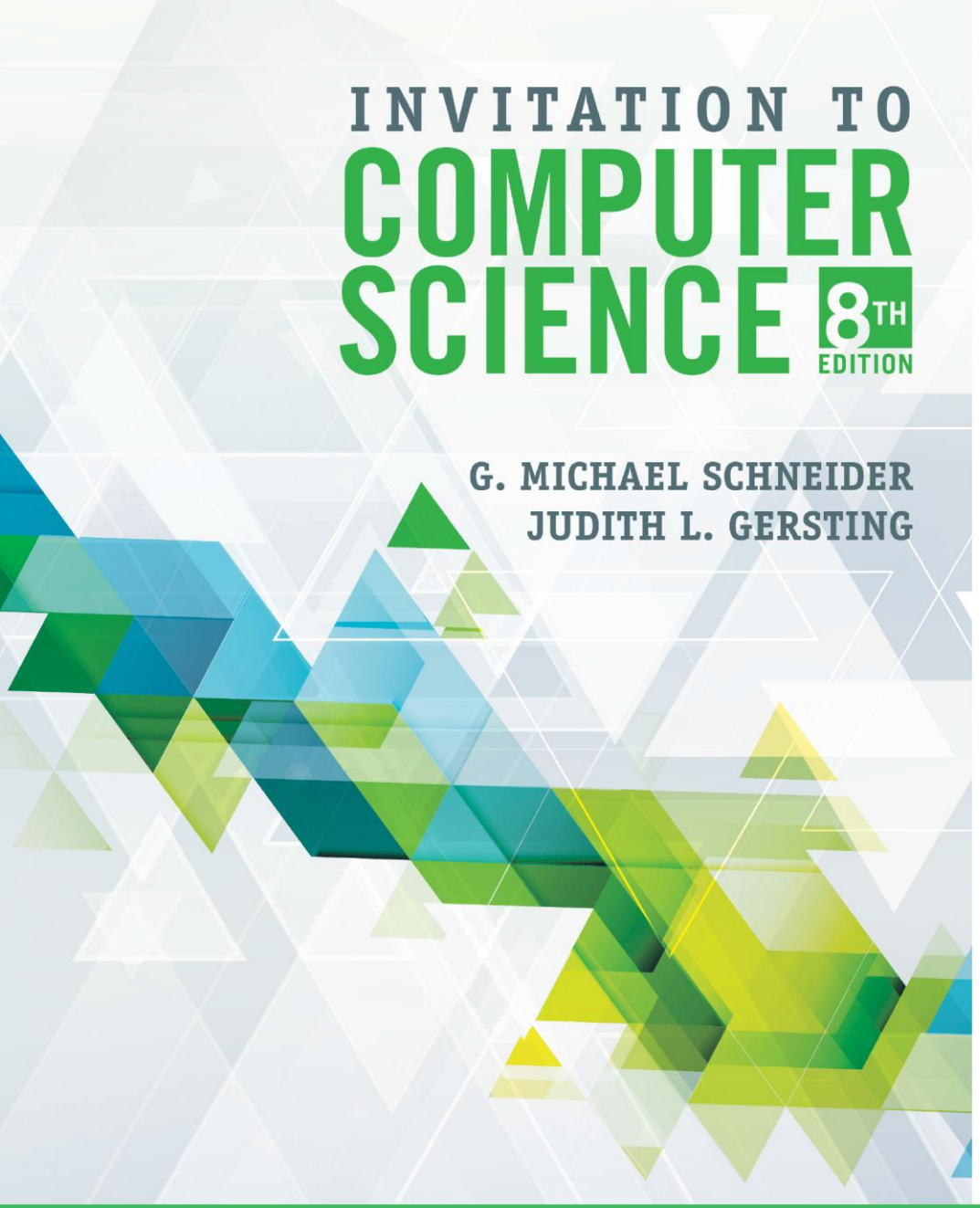


Chapter 5

Computer Systems Organization

CENGAGE

Learning Objectives (1 of 2)

- Enumerate the characteristics of the Von Neumann architecture
- Describe the components of a random access memory system, including how fetch and store operations work, and the use of cache memory to speed up access time
- Diagram the components of a typical arithmetic/logic unit (ALU) and illustrate how the ALU data path operates

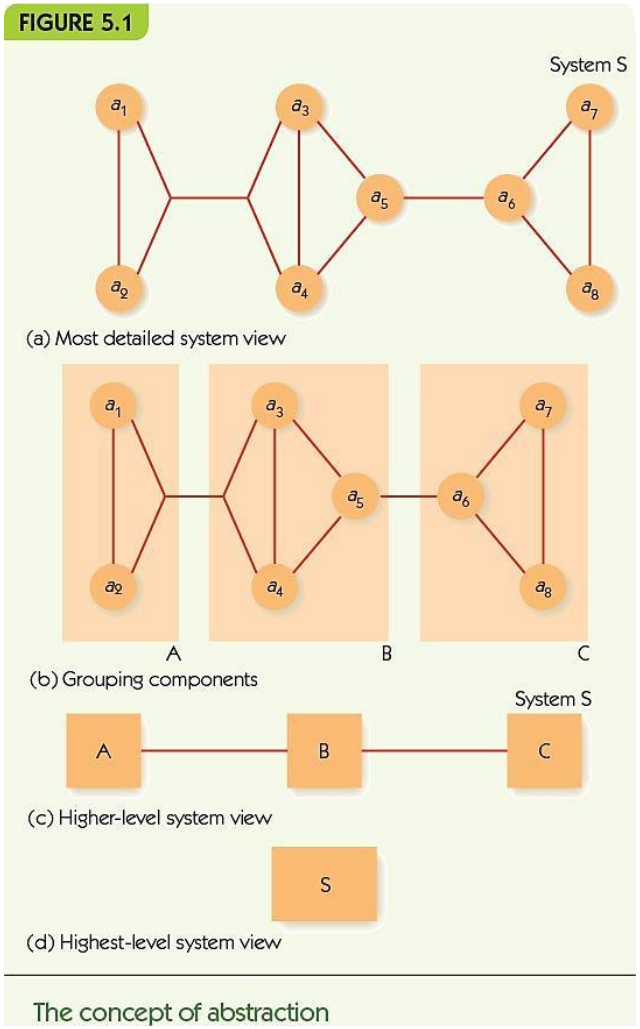
Learning Objectives (2 of 2)

- Describe the operation of the control unit and explain how it implements the stored program characteristic of the Von Neumann architecture
- List and explain the types of instructions in a typical instruction set, and how instructions are commonly encoded
- Diagram the organization of a typical Von Neumann machine
- Show the sequence of steps, using the book's notation, in the fetch, decode, and execute cycle to perform a typical instruction

Introduction (1 of 2)

- This chapter changes the **level of abstraction** and focuses on a higher level of computer system construction
- Focus on **functional units** and **computer organization**
- A **hierarchy of abstractions** hides unnecessary details
- Change focus from transistors to gates and to circuits as the basic unit
- Discusses in detail the Von Neumann architecture

Introduction (2 of 2)

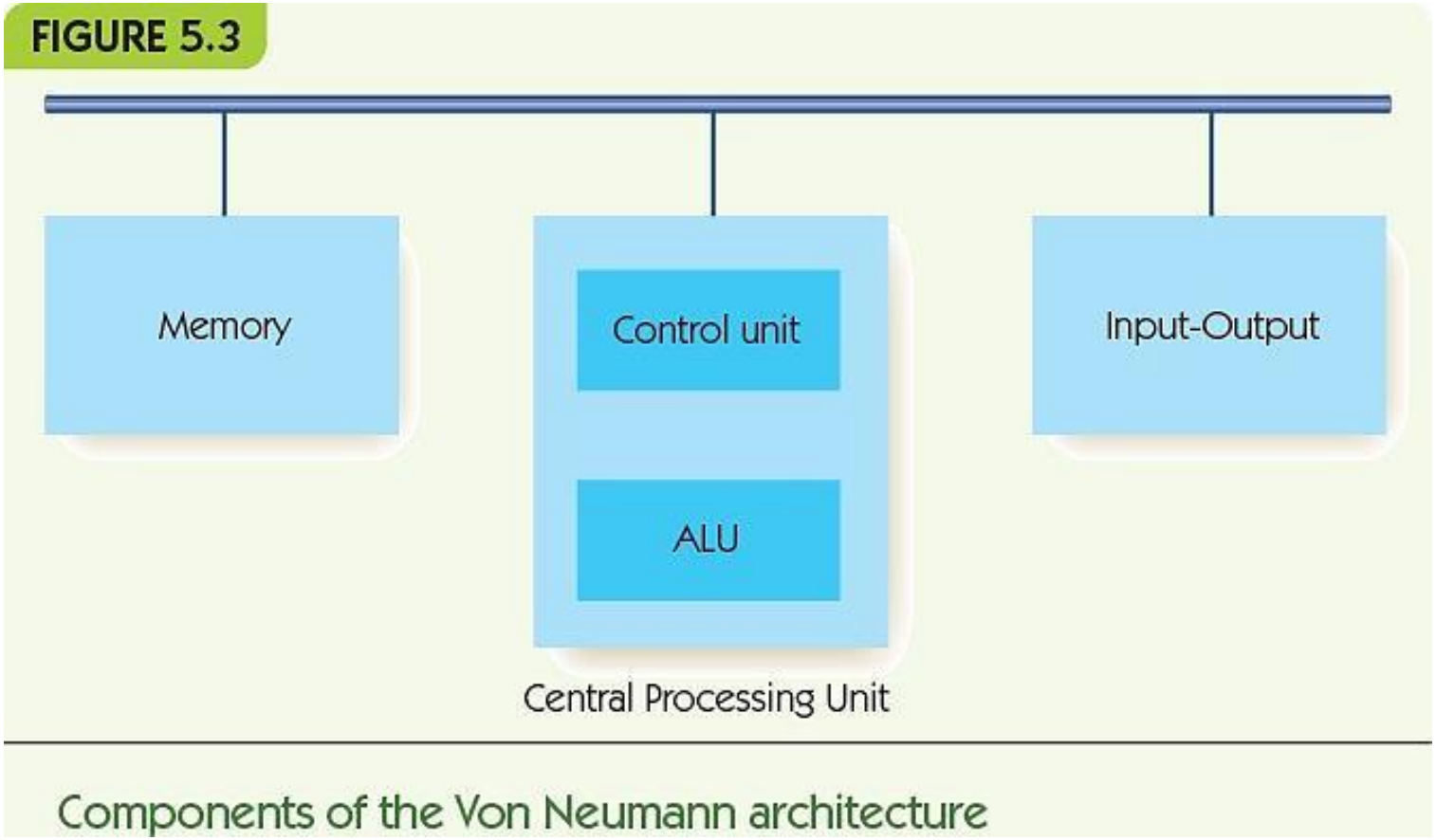


The Components of a Computer System (1 of 2)

Von Neumann architecture is the foundation for nearly all modern computers

- The four major subsystems of the Von Neumann architecture
  - Memory
  - Input/output
  - Arithmetic/logic unit (ALU)
  - Control Unit
    - ALU and control unit are often bundled inside the **central processing unit (CPU)**

The Components of a Computer System (2 of 2)

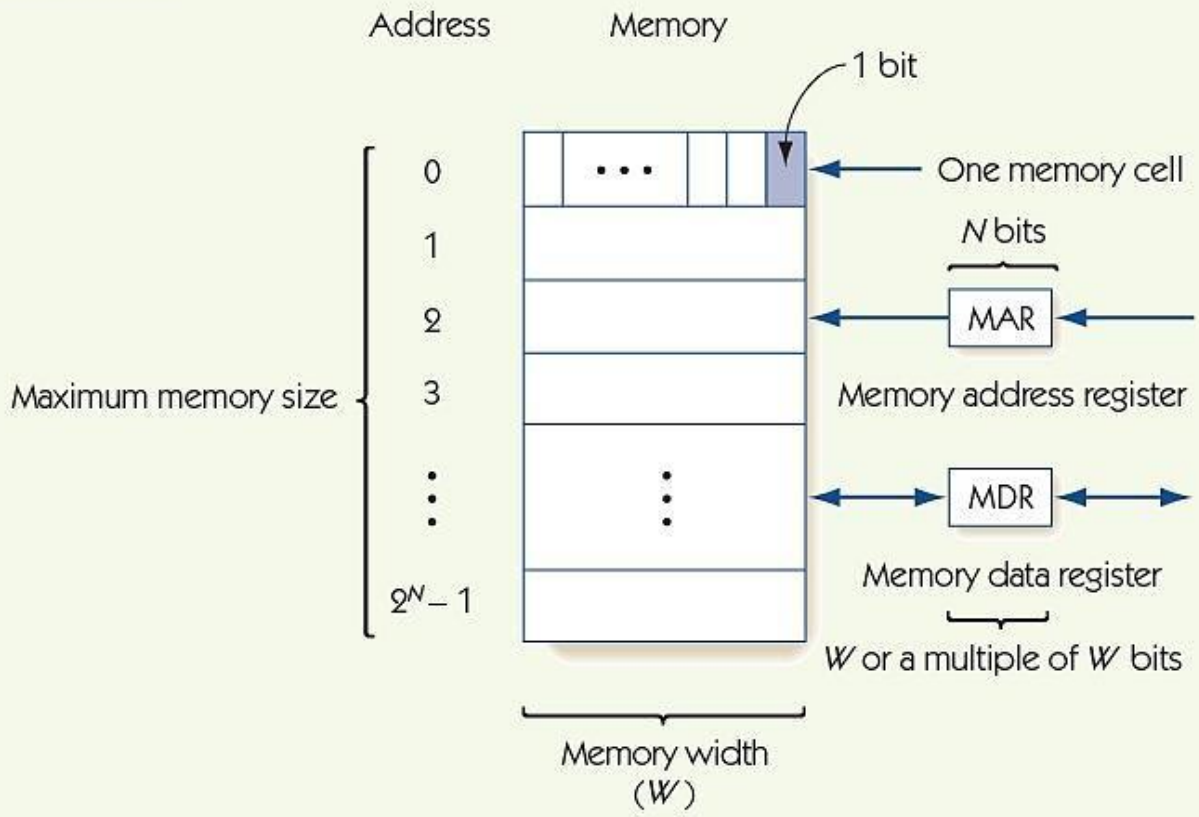


The Components of a Computer System Memory and Cache (1 of 9)

- **Memory:** functional unit where data is stored/retrieved
- **Random access memory (RAM)**
  - Organized into **cells**, each given a unique **address**
  - Equal time to access any cell
  - Cell values may be read and changed
- **Read-only memory (ROM):** A type of RAM with prerecorded information that cannot be modified or changed
- **Cell size/memory width** is typically 8 bits
- **Maximum memory size/address space** is 2^N, where N is length of address

The Components of a Computer System Memory and Cache (2 of 9)

FIGURE 5.4



Structure of random access memory

The Components of a Computer System Memory and Cache (3 of 9)

N	Maximum Memory Size (2^N)
16	65,536
20	1,048,576
22	4,194,304
24	16,777,216
32	4,294,967,296
40	1,099,511,627,776
50	1,125,899,906,842,624

The Components of a Computer System Memory and Cache (4 of 9)

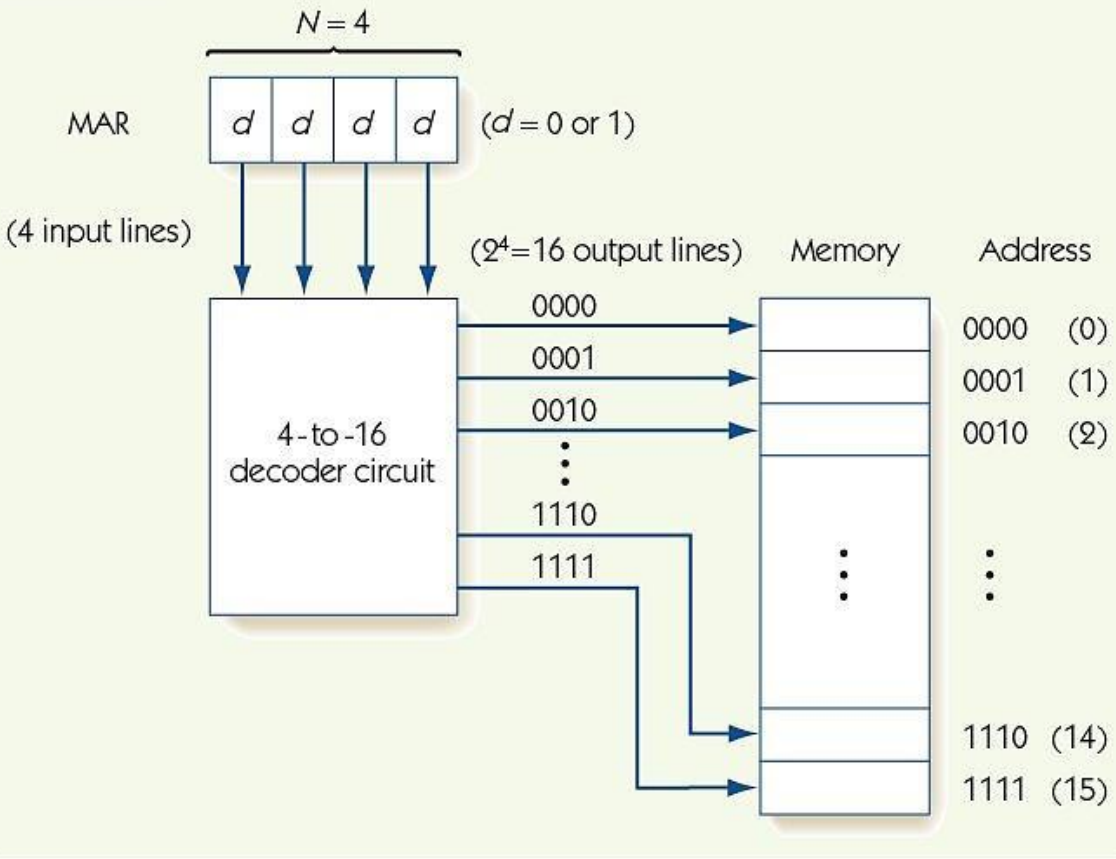
- Fetch: retrieve from memory (**nondestructive fetch**)
- Store: write to memory (**destructive store**)
- **Memory access time**
  - Time required to fetch/store
  - Modern RAM requires 5-10 **nanoseconds**
- **Memory address register (MAR)** holds memory address to access
- **Memory data register (MDR)** receives data from fetch and holds data to be stored

The Components of a Computer System Memory and Cache (5 of 9)

- Memory system circuits: **decoder** and **fetch/store controller**
- Decoder converts MAR into signal to a specific memory cell
  - One-dimensional versus two-dimensional memory organization
- Fetch/Store controller ▸ traffic cop for MDR
  - Takes in a signal that indicates fetch or store
  - Routes data flow to/from memory cells and MDR

The Components of a Computer System Memory and Cache (6 of 9)

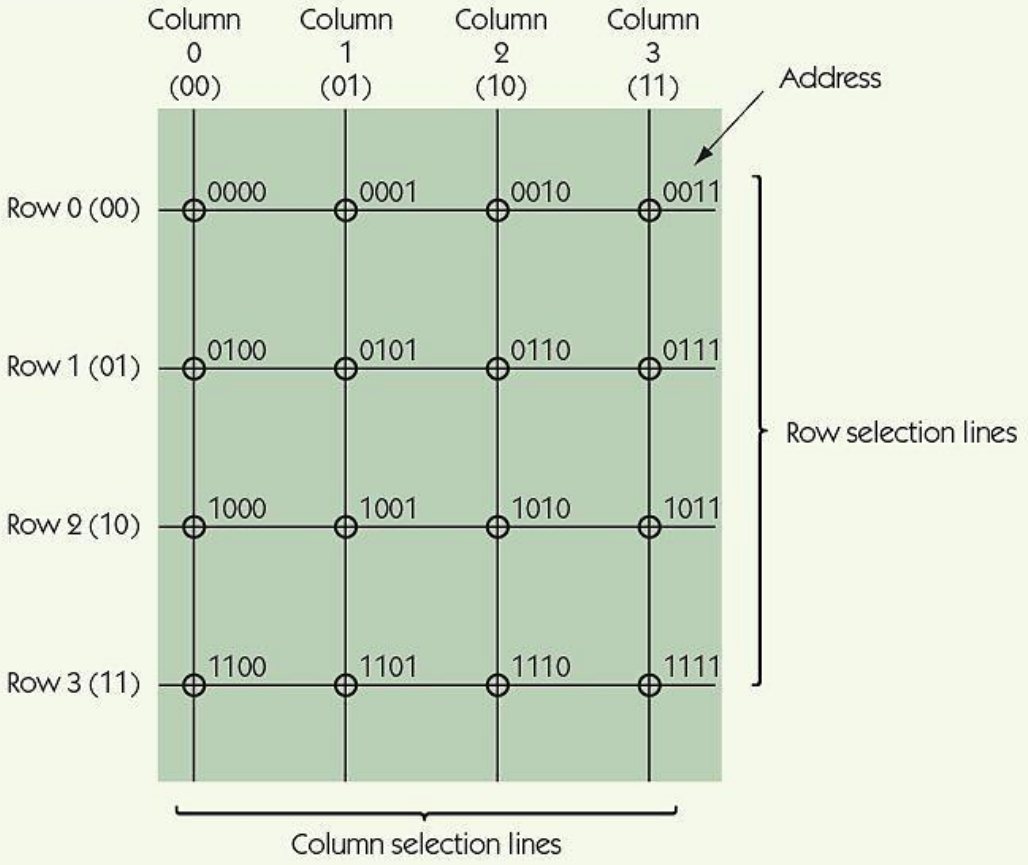
FIGURE 5.6



Organization of memory and the decoding logic

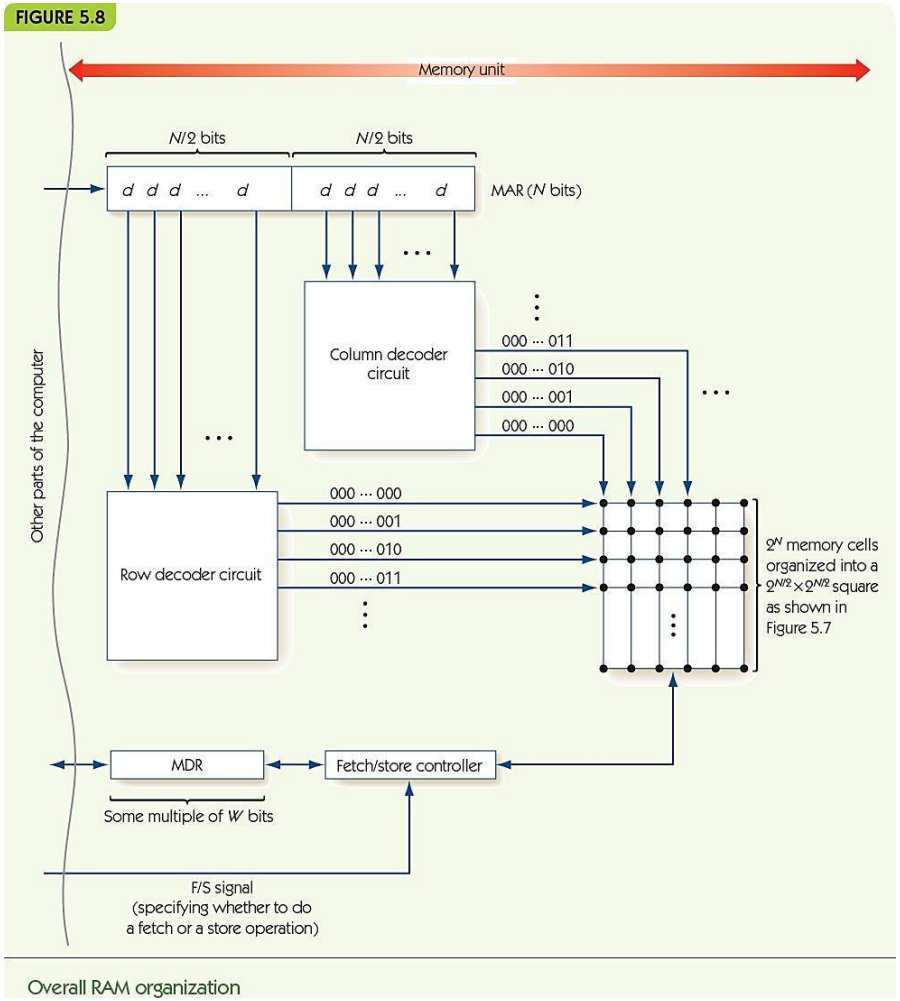
The Components of a Computer System Memory and Cache (7 of 9)

FIGURE 5.7



Two-dimensional memory organization

The Components of a Computer System Memory and Cache (8 of 9)



The Components of a Computer System Memory and Cache (9 of 9)

- RAM speeds increased more slowly than CPU speeds
- **Cache memory** is fast but expensive
  - Built into the CPU for fast access times
- **Principle of locality**
  - Values close to recently accessed memory are more likely to be accessed
  - Load neighbors into cache and keep recent values there
- **Cache hit rate:** percentage of times values are found in cache

The Components of a Computer System Input/Output and Mass Storage

- **Input/output (I/O)** connects the processor to the outside world
  - Humans: keyboard, monitor, etc.
  - Data storage: hard drive, DVD, flash drive
  - Other computers: network
- RAM = **volatile memory** (gone without power)
- **Mass storage systems** = **nonvolatile memory**
  - **Direct access storage devices (DASDs)**
  - **Sequential access storage devices (SASDs)**

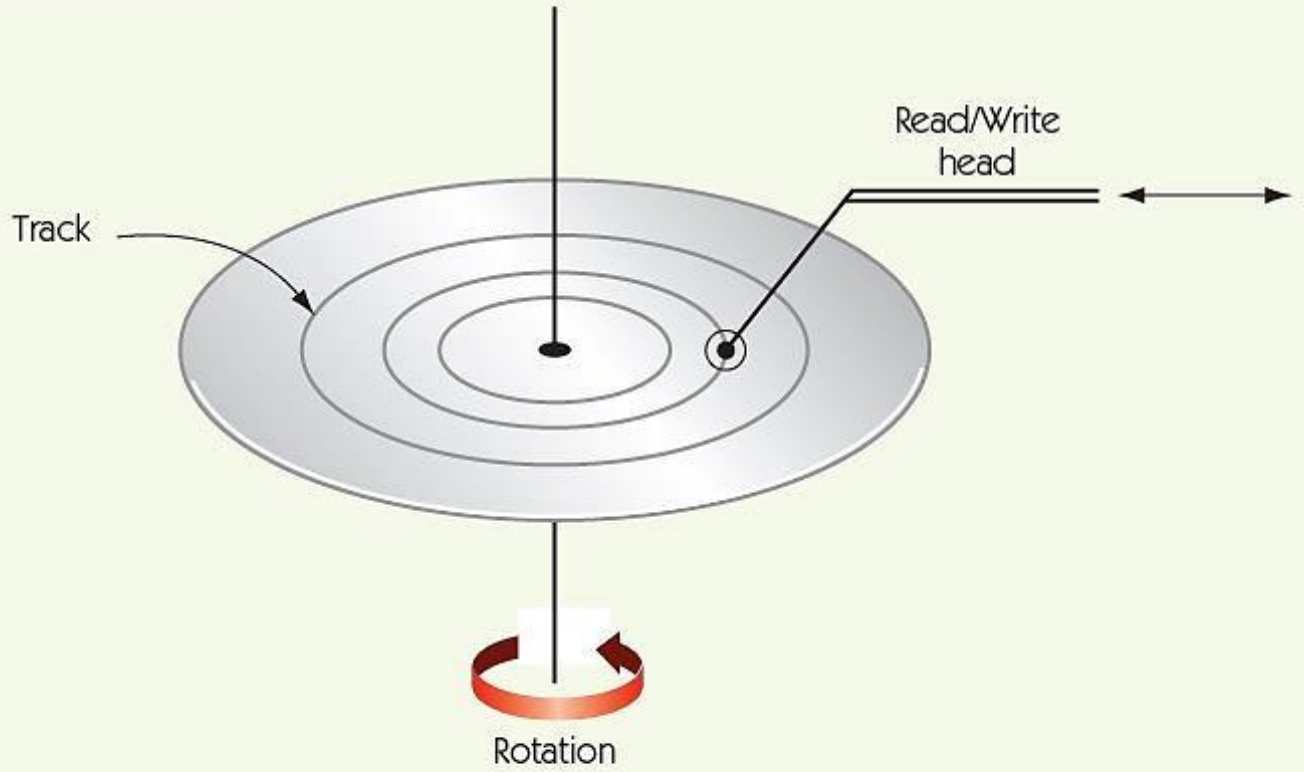
The Components of a Computer System I/O and Mass Storage (1 of 6)

DASDs

- Disks: Hard drives and optical media (CDs/DVDs)
  - **Tracks:** concentric rings around the disk surface
  - **Sectors:** fixed size segments of tracks, unit of retrieval
  - Time to retrieve data based on
    - **Seek time**
    - **Latency**
    - **Transfer time**
- Other nondisk DASDs: flash memory and solid-state drives (random access mass storage)

The Components of a Computer System I/O and Mass Storage (2 of 6)

FIGURE 5.13

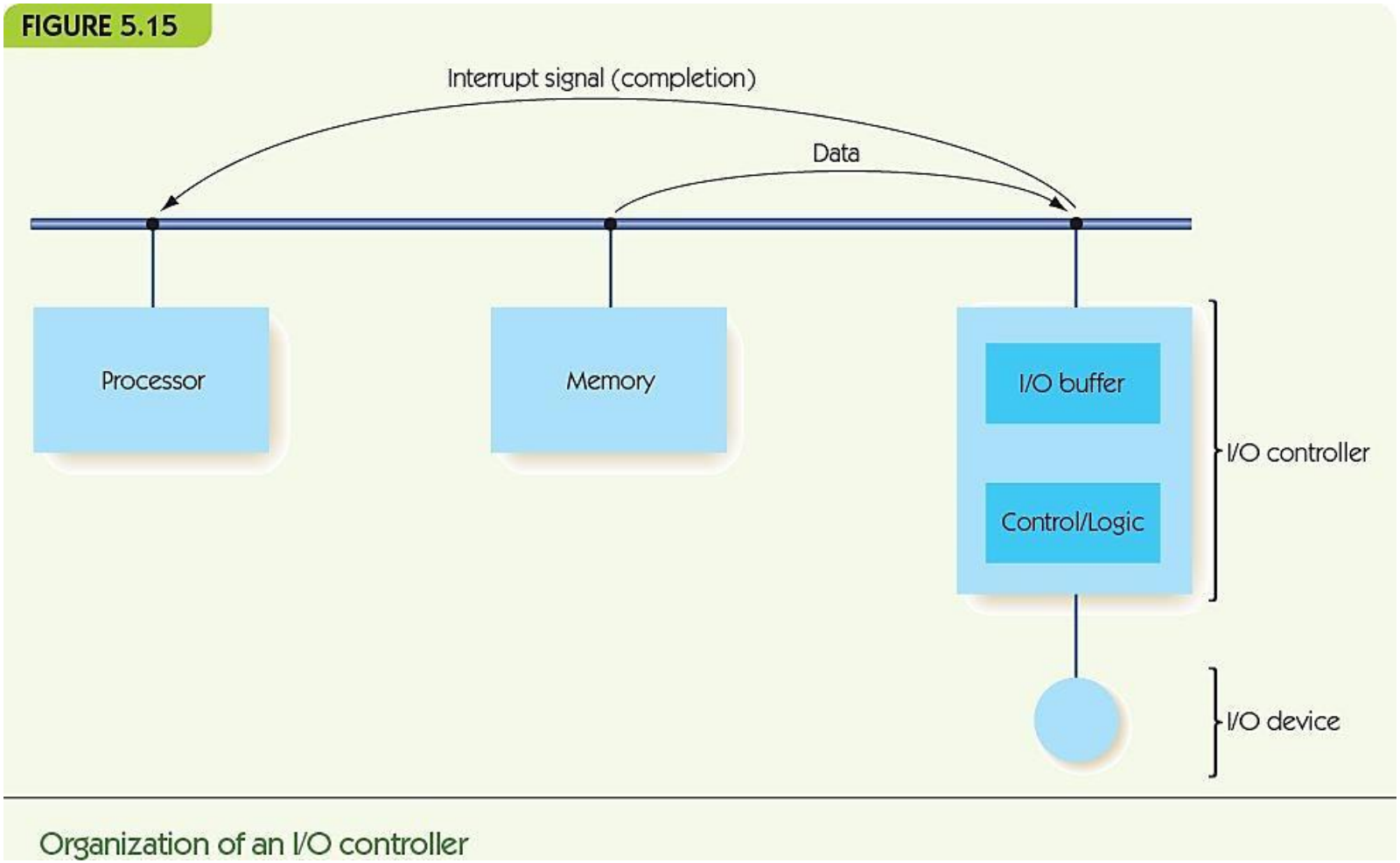


Overall organization of a typical rotating disk

The Components of a Computer System I/O and Mass Storage (3 of 6)

- DASDs and SASDs are orders of magnitude slower than RAM: (microseconds or milliseconds).
- **I/O Controller** manages data transfer with slow I/O devices, freeing processor to do other work.
- Controller sends an **interrupt signal** to processor when I/O task is done.

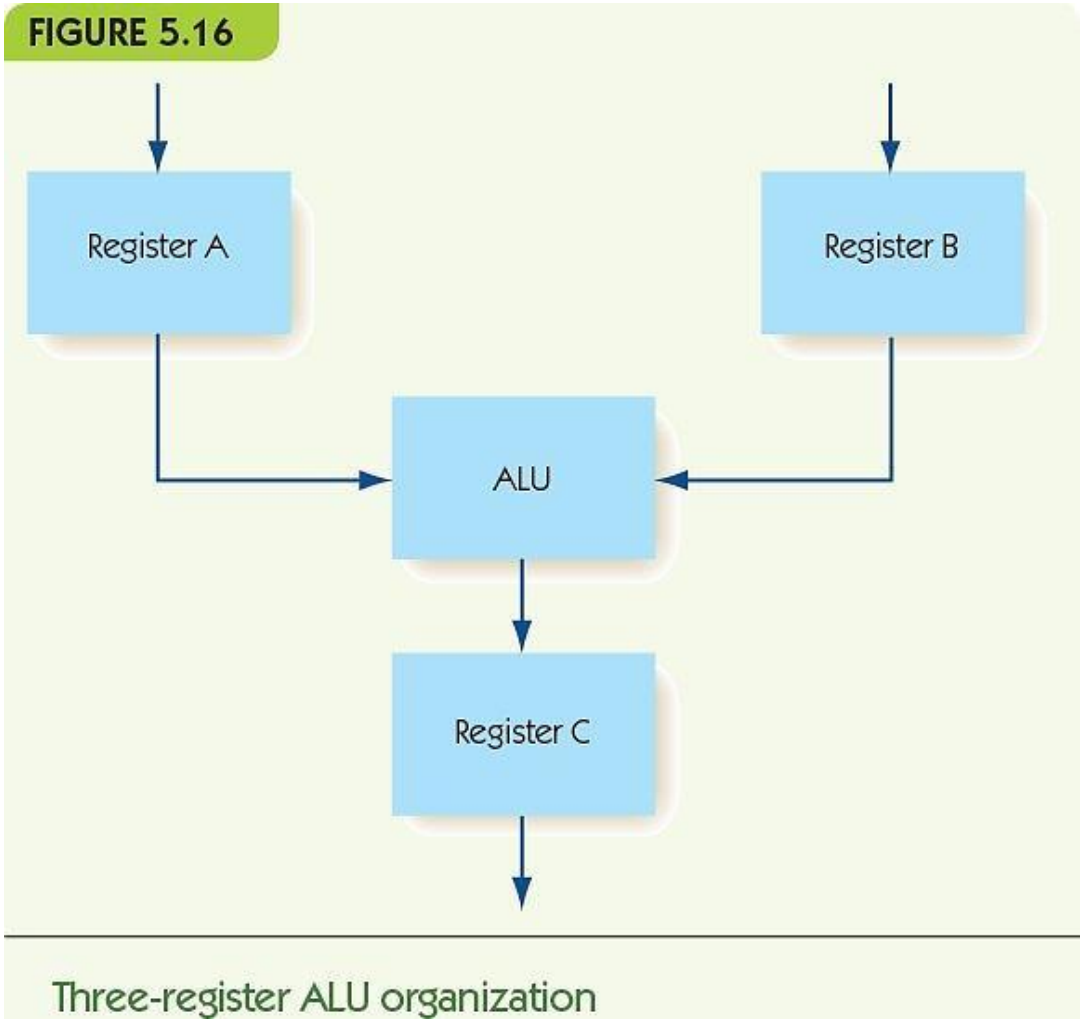
The Components of a Computer System I/O and Mass Storage (4 of 6)



The Components of a Computer System The Arithmetic/Logic Unit

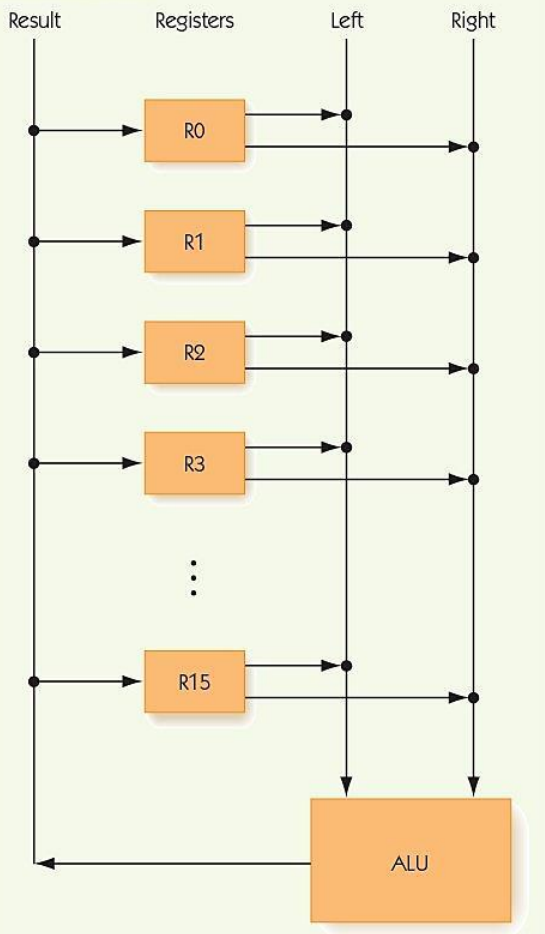
- ALU is part of the **processor**
- Contains circuits for arithmetic
  - Addition, subtraction, multiplication, and division
- Contains circuits for comparison and logic
  - Equality, and, or, not
- Contains **registers**: high-speed, dedicated memory connected to circuits
- **Data path:** how information flows in the ALU
  - From registers to circuits
  - From circuits back to registers

The Components of a Computer System I/O and Mass Storage (5 of 6)



The Components of a Computer System I/O and Mass Storage (6 of 6)

FIGURE 5.17



Multiregister ALU organization

The Components of a Computer System The ALU (1 of 4)

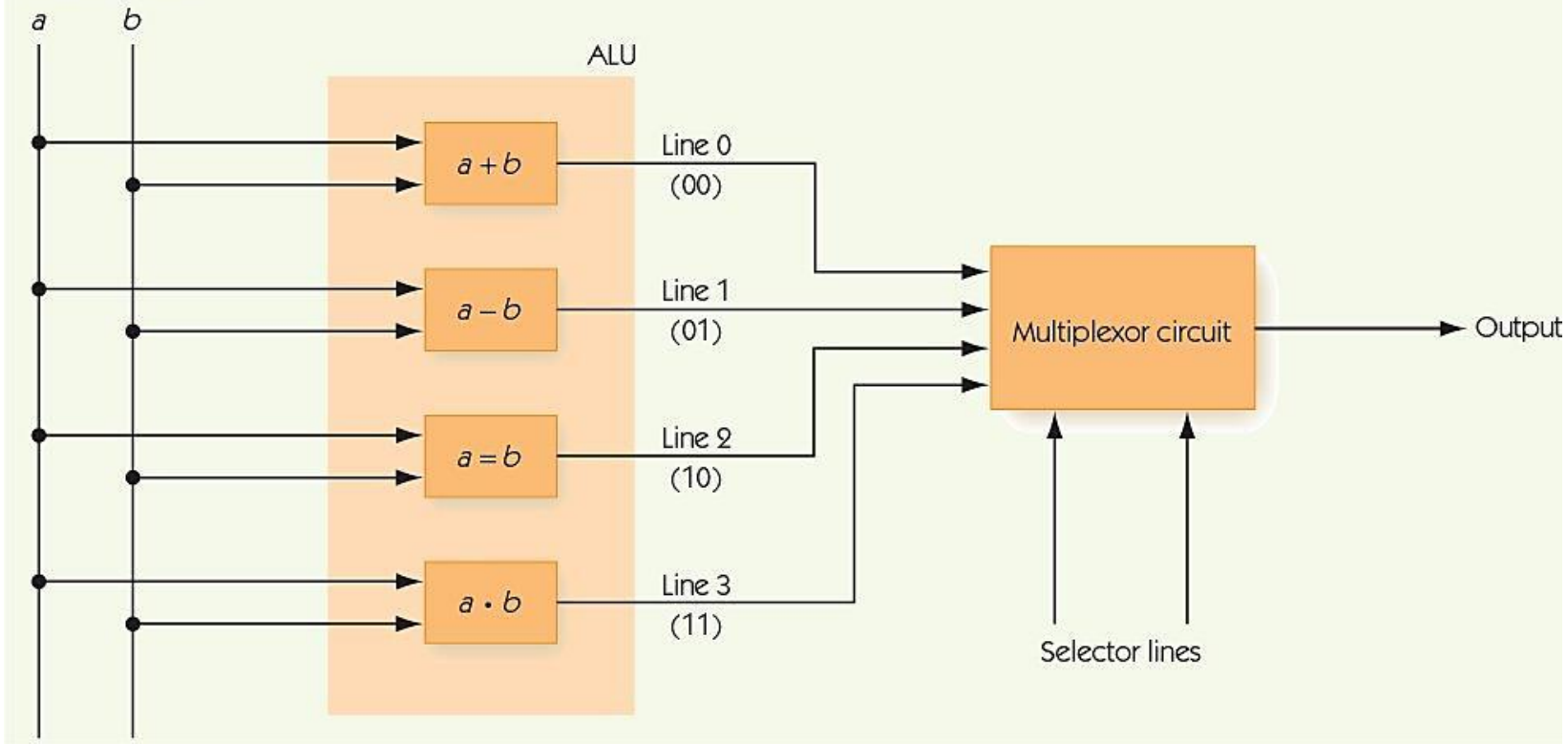
- How is the operation to perform chosen?
  - Option 1: decoder signals one circuit to run.
  - Option 2: run all circuits, multiplexer selects one output from all circuits.
- In practice, option 2 is usually chosen.

The Components of a Computer System The ALU (2 of 4)

- Information flow
  - Data comes in from outside to registers
  - Signal comes from registers to ALU
  - Signal moves from ALU to multiplexer
  - Multiplexer selects the value to keep and discards the rest
  - Result from the multiplexer goes back to the register and then to outside

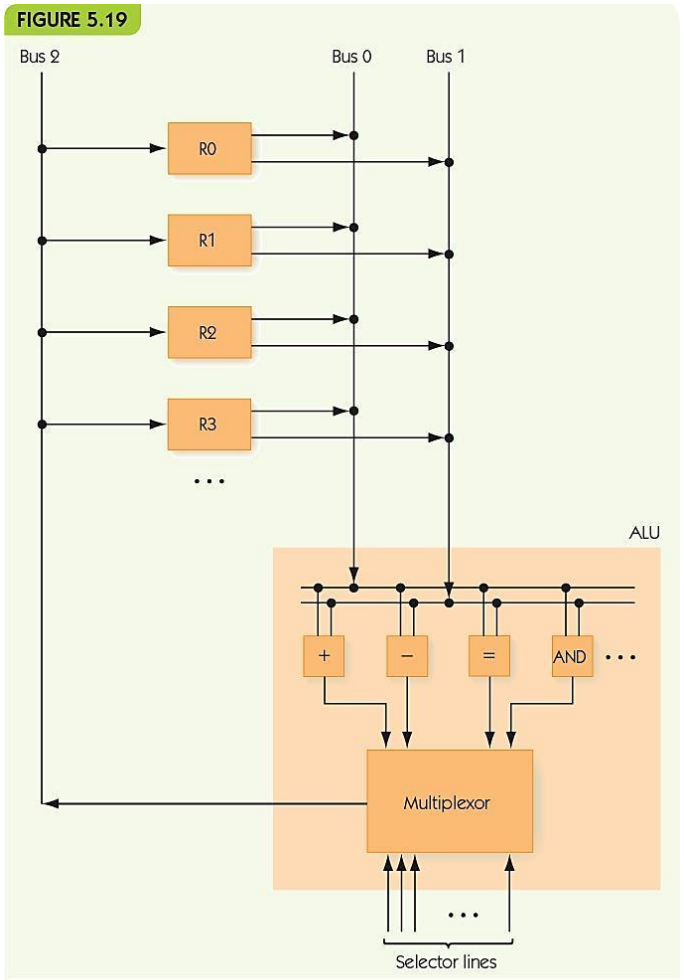
The Components of a Computer System
The ALU (3 of 4)

FIGURE 5.18



Using a multiplexor circuit to select the proper ALU result

The Components of a Computer System
The ALU (4 of 4)

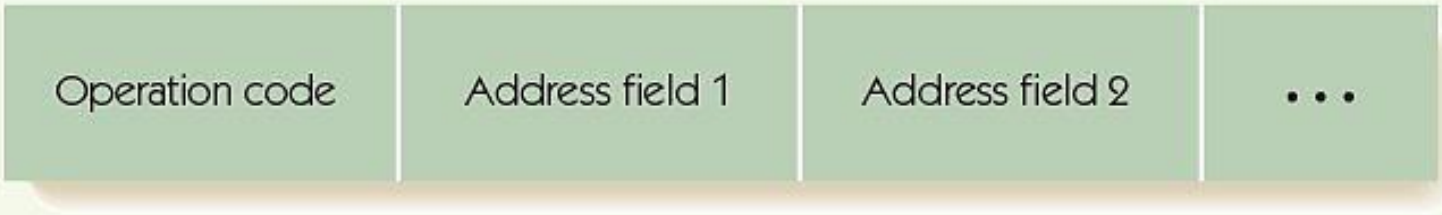


The Components of a Computer System The Control Unit (1 of 9)

- **Stored program** characteristic
  - Programs are encoded in binary and stored in computer's memory
- **Control unit** fetches instructions from memory, decodes them, and executes them
- Instructions encoded
  - Operation code (op code) tells which operation
  - Addresses tell which memory addresses/registers to operate on

The Components of a Computer System
The Control Unit (2 of 9)

FIGURE 5.20



Typical machine language instruction format

The Components of a Computer System The Control Unit (3 of 9)

- **Machine language**
  - Binary strings encode instructions
  - Instructions can be carried out by hardware
  - Sequences of instructions encode algorithms
- **Instruction set**
  - Instructions implemented by a particular chip
  - Each kind of processor has a different instruction set (speaks a different language)

The Components of a Computer System The Control Unit (4 of 9)

- **Reduced instruction set computer (RISC)**
  - Small instruction sets
  - Each instruction highly optimized
  - Easy to design hardware
- **Complex instruction set computer (CISC)**
  - Large instruction set
  - Single instruction can do a lot of work
  - Complex to design hardware
- Modern hardware is a compromise between RISC and CISC

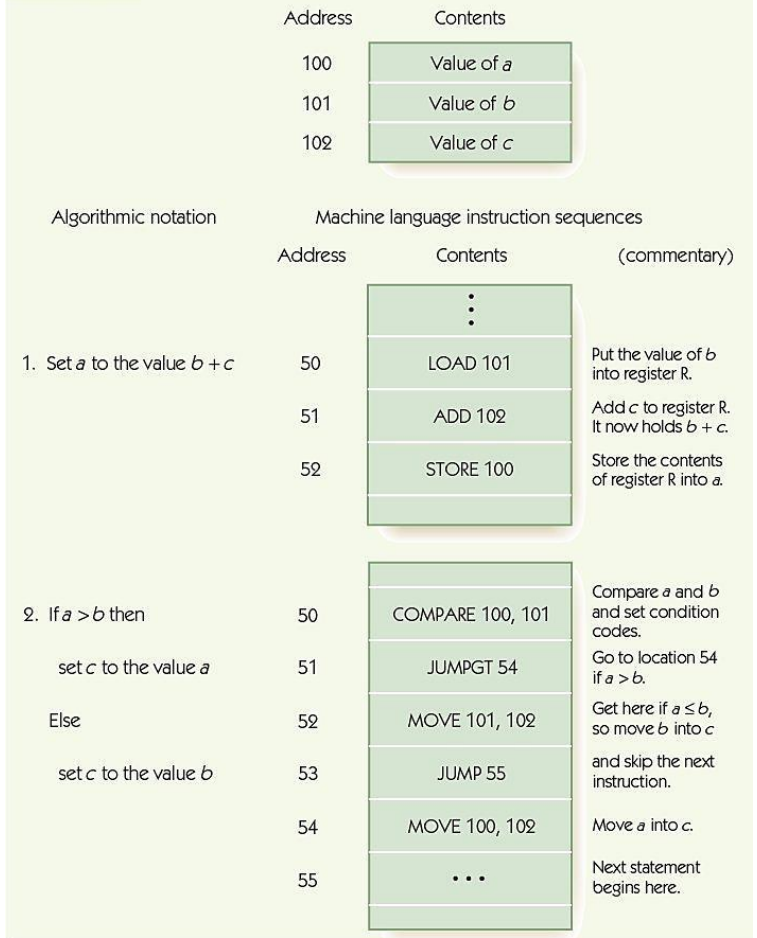
The Components of a Computer System The Control Unit (5 of 9)

Instruction set examples:

- Data transfer, e.g., move data from memory to register
- Arithmetic, e.g., add, but also AND
- Comparison: compare two values
- Branch: change to a nonsequential instruction
  - Branching allows for conditional and loop forms
  - E.g., JUMPLT *a* = If previous comparison of A and B found A < B, then jump to instruction at address *a*

The Components of a Computer System
The Control Unit (6 of 9)

FIGURE 5.21



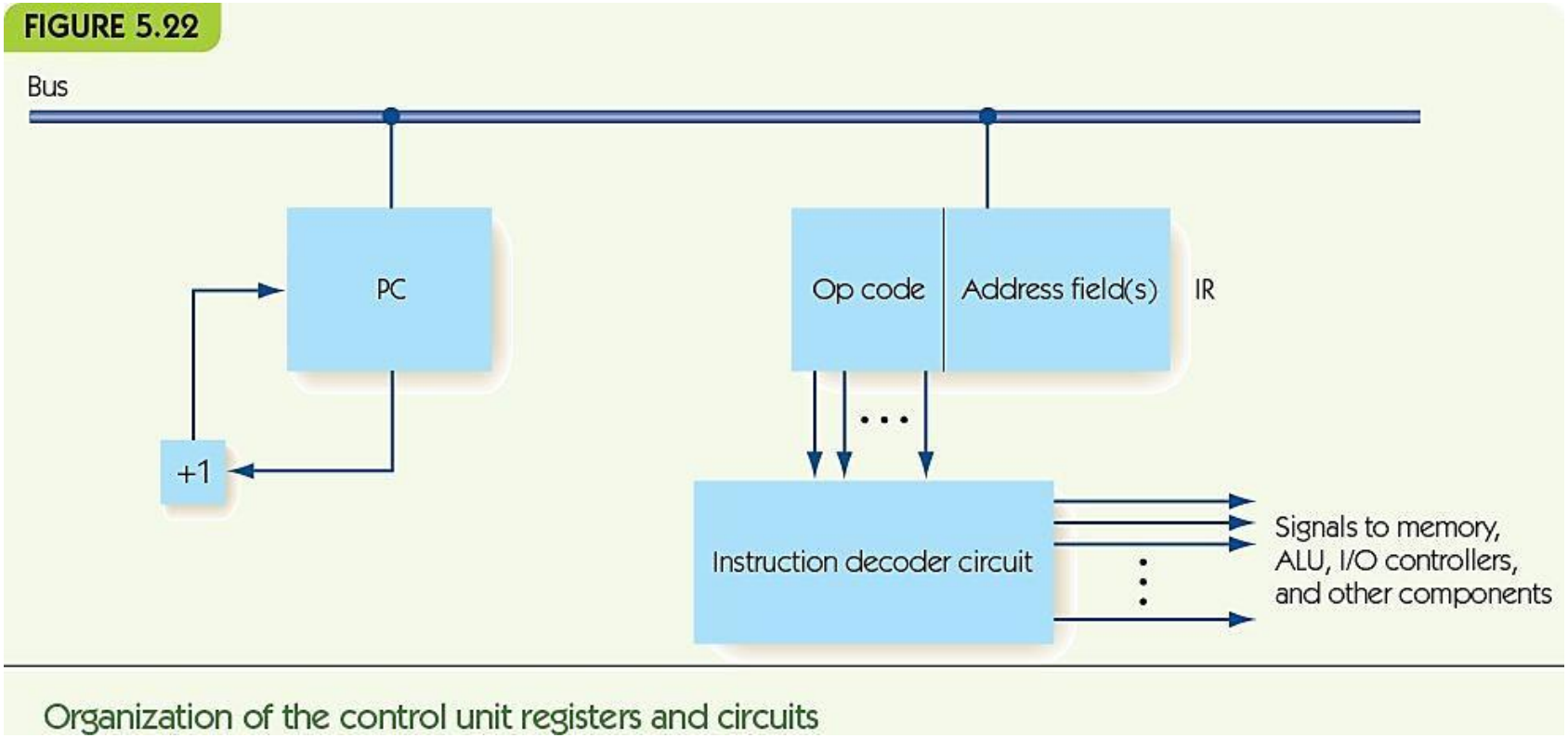
Examples of simple machine language instruction sequences

The Components of a Computer System The Control Unit (7 of 9)

Control unit contains:

- **Program counter (PC)** register: holds address of next instruction
- **Instruction register (IR)**: holds encoding of current instruction
- Instruction decoder circuit
  - Decodes op code of instruction and signals helper circuits, one per instruction
    - Helpers send addresses to proper circuits
    - Helpers signal ALU, I/O controller, and memory

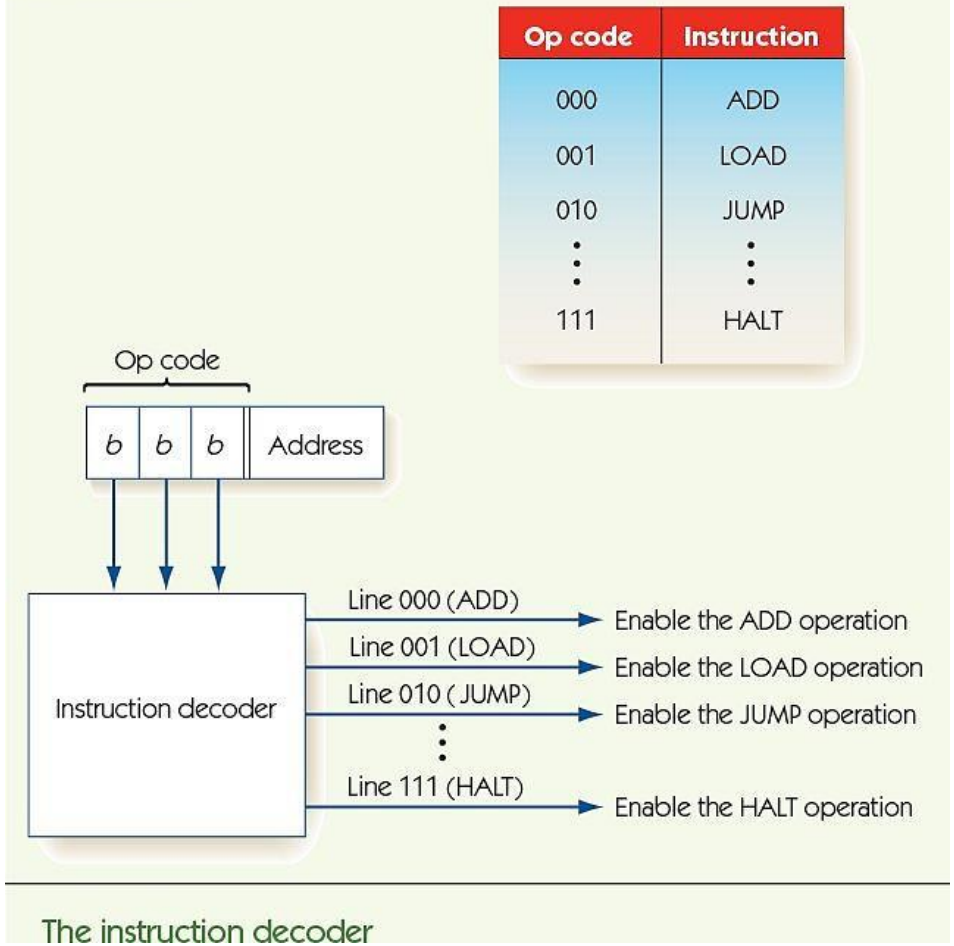
The Components of a Computer System
The Control Unit (8 of 9)



Organization of the control unit registers and circuits

The Components of a Computer System
The Control Unit (9 of 9)

FIGURE 5.23

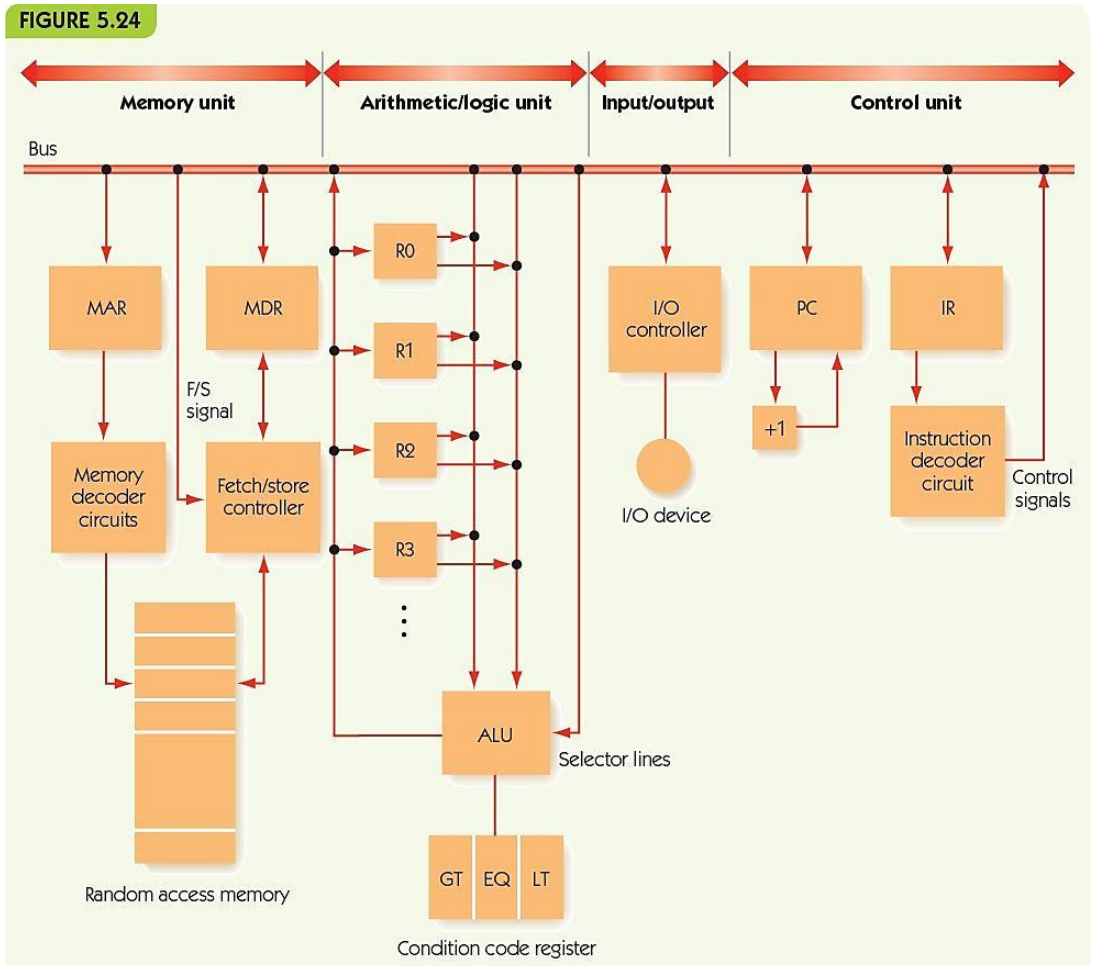


The instruction decoder

Putting the Pieces Together—the Von Neumann Architecture (1 of 9)

- Combine previous pieces: Von Neumann machine
- Fetch/decode/execute phase
  - Machine repeats until HALT instruction or error
  - Also called Von Neumann cycle
- Fetch phase: get next instruction into memory
- Decode phase: instruction decoder gets op code
- Execute phase: different for each instruction

Putting the Pieces Together—the Von Neumann Architecture (2 of 9)



Putting the Pieces Together—the Von Neumann Architecture (3 of 9)

Notation for computer's behavior

CON(A)	Contents of memory cell A
A → B	Send value in register A to register B (special registers: PC, MAR, MDR, IR, ALU, R, GT, EQ, LT, +1)
FETCH	Initiate a memory fetch operation
STORE	Initiate a memory store operation
ADD	Instruct the ALU to select the output of the adder circuit
SUBTRACT	Instruct the ALU to select the output of the subtract circuit

Putting the Pieces Together—the Von Neumann Architecture (4 of 9)

Fetch phase

1. PC → MAR Send address in PC to MAR
2. FETCH Initiate fetch, data to MDR
3. MDR → IR Move instruction in MDR to IR
4. PC + 1 → PC Add one to PC

Decode phase

1. IR_{op} → instruction decoder

Putting the Pieces Together—the Von Neumann Architecture (5 of 9)

FIGURE 5.25

Binary Op Code	Operation	Meaning
0000	LOAD X	CON(X) → R
0001	STORE X	R → CON(X)
0010	CLEAR X	0 → CON(X)
0011	ADD X	R + CON(X) → R
0100	INCREMENT X	CON(X) + 1 → CON(X)
0101	SUBTRACT X	R – CON(X) → R
0110	DECREMENT X	CON(X) – 1 → CON(X)
0111	COMPARE X	if CON(X) > R then GT = 1 else 0 if CON(X) = R then EQ = 1 else 0 if CON(X) < R then LT = 1 else 0
1000	JUMP X	Get the next instruction from memory location X.
1001	JUMPGT X	Get the next instruction from memory location X if GT = 1.
1010	JUMPEQ X	Get the next instruction from memory location X if EQ = 1.
1011	JUMPLT X	Get the next instruction from memory location X if LT = 1.
1100	JUMPNEQ X	Get the next instruction from memory location X if EQ = 0.
1101	IN X	Input an integer value from the standard input device and store into memory cell X.
1110	OUT X	Output, in decimal notation, the value stored in memory cell X.
1111	HALT	Stop program execution.

Instruction set for our Von Neumann machine

Putting the Pieces Together—the Von Neumann Architecture (6 of 9)

Execution phase

LOAD X meaning CON(X) → R

1. IR_{addr} → MAR Send address X to MAR
2. FETCH Initiate fetch, data to MDR
3. MDR → R Copy data in MDR into R

STORE X meaning R → CON(X)

1. IR_{addr} → MAR Send address X to MAR
2. R → MDR Send data in R to MDR
3. STORE Initiate store of MDR to X

Putting the Pieces Together—the Von Neumann Architecture (7 of 9)

ADD X meaning R + CON(X) → R

1. IR_{addr} → MAR Send address X to MAR
2. FETCH Initiate fetch, data to MDR
3. MDR → ALU Send data in MDR to ALU
4. R → ALU Send data in R to ALU
5. ADD Select ADD circuit as result
6. ALU → R Copy selected result to R

JUMP X meaning get next instruction from X

1. IR_{addr} → PC Send address X to PC

Putting the Pieces Together—the Von Neumann Architecture (8 of 9)

COMPARE X meaning:

if CON(X) > R, then GT = 1, else 0

if CON(X) = R, then EQ = 1, else 0

if CON(X) < R, then LT = 1, else 0

1. $IR_{addr} \rightarrow MAR$ Send address X to MAR
2. FETCH Initiate fetch, data to MDR
3. MDR → ALU Send data in MDR to ALU
4. R → ALU Send data in R to ALU
5. SUBTRACT Evaluate CON(X) – R

Sets EQ, GT, and LT

Putting the Pieces Together—the Von Neumann Architecture (9 of 9)

JUMPGT X meaning:

if GT = 1, then jump to X,

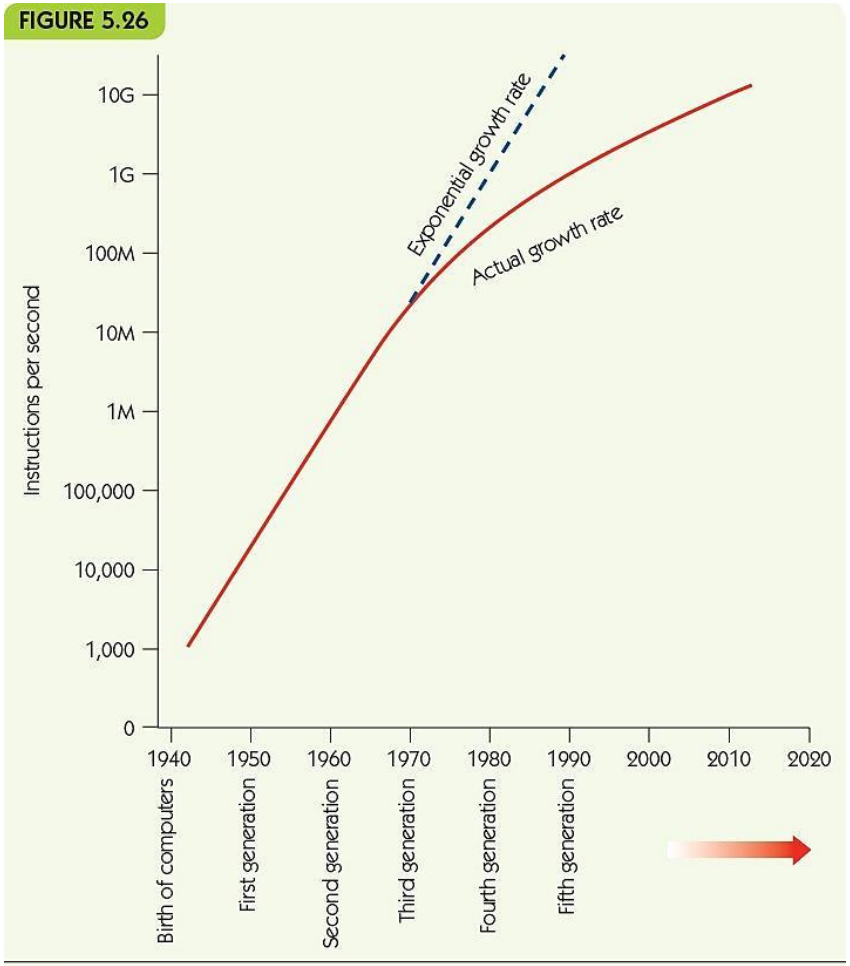
else continue to next instruction

1. IF GT = 1 THEN $IR_{addr} \rightarrow PC$

Non–Von Neumann Architectures (1 of 6)

- Problems to solve are always larger
- Computer chip speeds no longer increase exponentially
- Reducing size puts gates closer together, faster
  - Speed of light pertains to signals through wire
  - Cannot put gates much closer together
  - Heat production increases too fast
- **Von Neumann bottleneck**: inability of sequential machines to handle larger problems

Non–Von Neumann Architectures (2 of 6)



Non–Von Neumann Architectures (3 of 6)

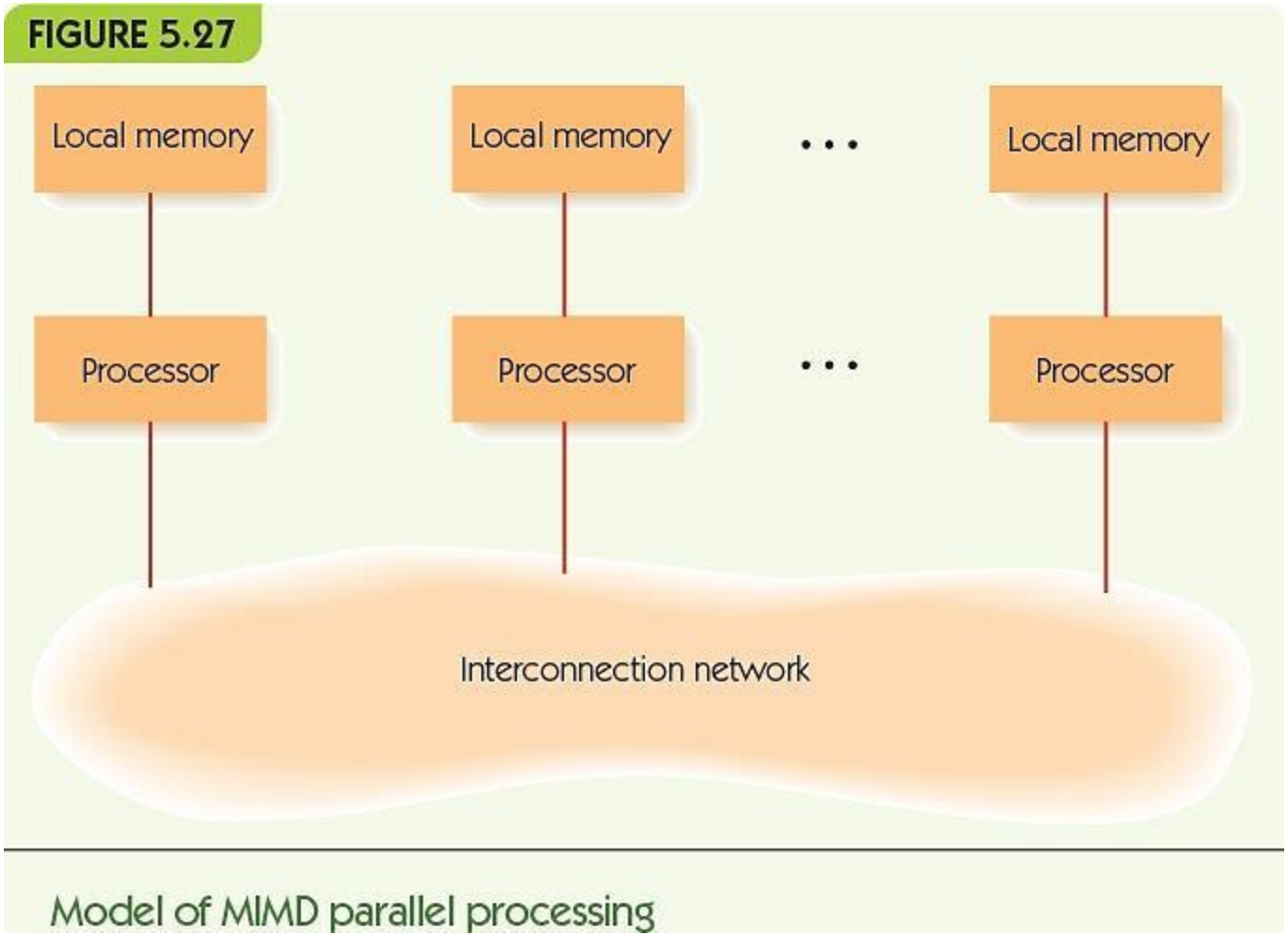
- **Non–Von Neumann architectures**
  - Other ways to organize computers
  - Most are experimental/theoretical, EXCEPT parallel processing
- **Parallel processing**
  - Many processing units operating at the same time
  - Supercomputers (in the past)
  - Desktop multicore machines and “the cloud” (in the present)
  - **Quantum computing** (in the future)

Non–Von Neumann Architectures (4 of 6)

MIMD parallel processing

- Multiple instruction stream/Multiple data streams
  - Cluster computing
- Multiple, independent processors
- Each ALU operates on its own data
- Each processor can operate independently
  - On its own data
  - On its own program
  - At its own rate

Non–Von Neumann Architectures (5 of 6)



Non–Von Neumann Architectures (6 of 6)

Varieties of MIMD systems

- Special-purpose systems: newer supercomputers
- **Cluster computing**: standard machines communicating over LAN or WAN
- **Grid computing**: machines of varying power, over large distances/Internet
  - Examples
    - SETI project
    - BOINC at Berkley
- Hot research area ▸ **parallel algorithms**
  - Need to take advantage of all this processing power

Summary (1 of 2)

- We must abstract in order to manage system complexity—no more writing instructions in machine language.
- Von Neumann architecture is standard for modern computing.
- Von Neumann machines have memory, I/O, ALU, and control unit; programs are stored in memory; execution is sequential unless program says otherwise.
- Memory is organized into addressable cells; data is fetched and stored based on MAR and MDR; uses decoder and fetch/store controller.

Summary (2 of 2)

- Mass data storage is nonvolatile; disks store and fetch sectors of data stored in tracks.
- I/O is slow, needs dedicated controller to free CPU.
- ALU performs computations, moving data to/from dedicated registers.
- Control unit fetches, decodes, and executes instructions; instructions are written in machine language.
- Parallel processing architectures can perform multiple instructions at one time.