DATA MANIPULATION

KHOA CÔNG NGHỆ THÔNG TIN
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN

Computer Science: An Overview by J. Glenn Brookshear and Dennis Brylow publisher Pearson

Data Manipulation

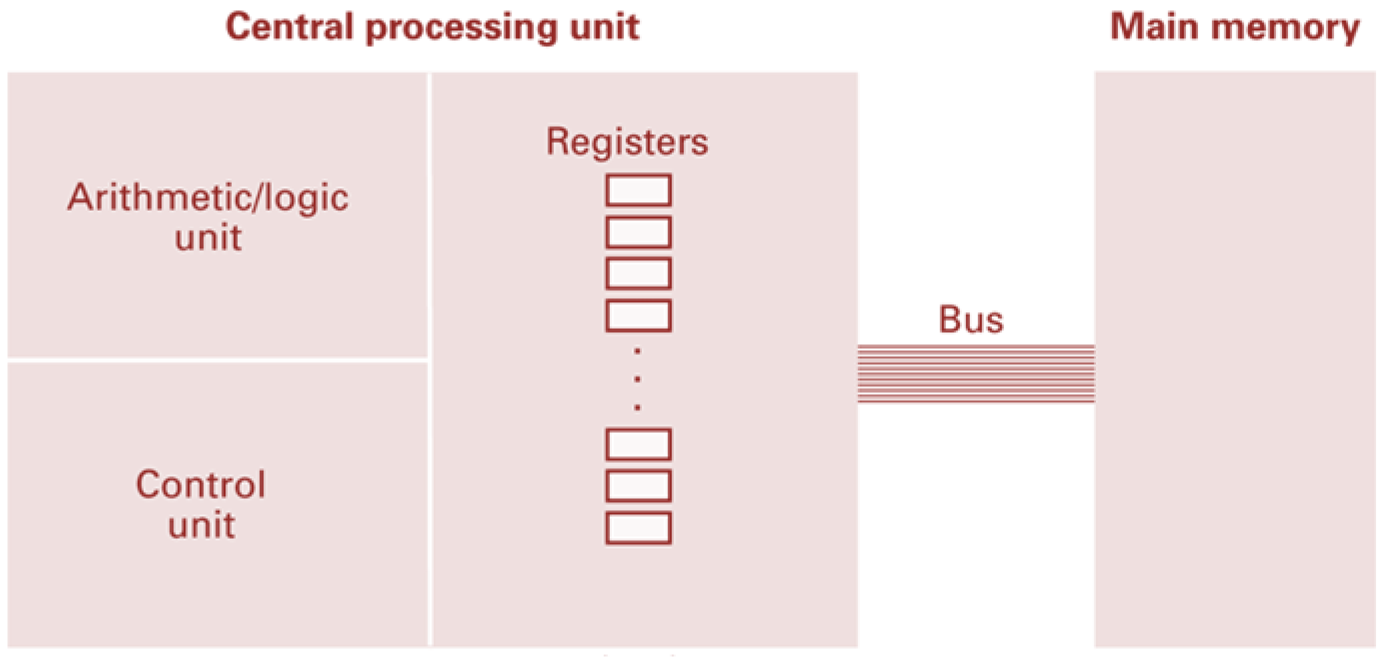
- Computer Architecture
- Machine Language
- Program Execution
- Arithmetic/Logic Instructions
- Communicating with Other Devices
- Other Architectures

COMPUTER ARCHITECTURE

Computer Architecture

- Central Processing Unit (CPU) or processor
  - Arithmetic/Logic unit versus Control unit
  - Registers
    - General purpose
    - Special purpose
- Bus
- Motherboard

CPU and main memory connected via a bus



Motherboards

- The motherboard
  - is the main printed circuit board.
  - contains the buses, or electrical pathways found in a computer. Buses allow data to travel among the various components.
  - accommodates CPU, RAM, expansion slots, heat sink/fan assembly, BIOS chip, chip set, sockets, internal and external connectors, various ports, and the embedded wires that interconnect the motherboard components.

Central Processing Unit

- The Central Processing Unit (CPU) is known as the brain of the computer. It is also referred to as the processor.
- The CPU executes a program, which is a sequence of stored instructions.

Central Processing Unit

- Some CPUs incorporate **hyperthreading** or **hypertransport** to enhance the performance of the CPU.
- The amount of data that a CPU can process at one time depends on the size of the processor data bus.
- Speed of the CPU is measured in **cycles per second** - megahertz (MHz) or gigahertz (GHz).
- **Overclocking** is a technique used to make a processor work at a faster speed than its original specification.

Central Processing Unit

- The latest processor technology has resulted in CPU manufacturers finding ways to incorporate more than one CPU core onto a single chip.
  - Dual Core CPU - Two cores inside a single CPU
  - Triple Core CPU - Three cores inside a single CPU
  - Quad Core CPU - Four cores inside a single CPU
  - Hexa-Core CPU - Six cores inside a single CPU
  - Octa-Core CPU - Eight cores inside a single CPU

The Arithmetic/Logic Unit

- Subsystem that performs addition, subtraction, and comparison for equality
- Components
  - Registers, interconnections between components, and the ALU circuitry
- Register
  - Storage cell that holds the operands of an arithmetic operation and holds its result
- Bus
  - Path for electrical signals

The Arithmetic/Logic Unit

- Registers are similar to RAM with following minor differences
  - They do not have a numeric memory address but are accessed by a special register designator such as A, X or R0
  - They can be accessed much more quickly than regular memory cells
  - They are not used for general purpose storage but for specific purposes such as holding the operands for an upcoming arithmetic computations.
- A typical ALU has 16, 32 or 64 registers.

The Control Unit

- Control unit
  - Tasks: fetch, decode, and execute

MACHINE LANGUAGE

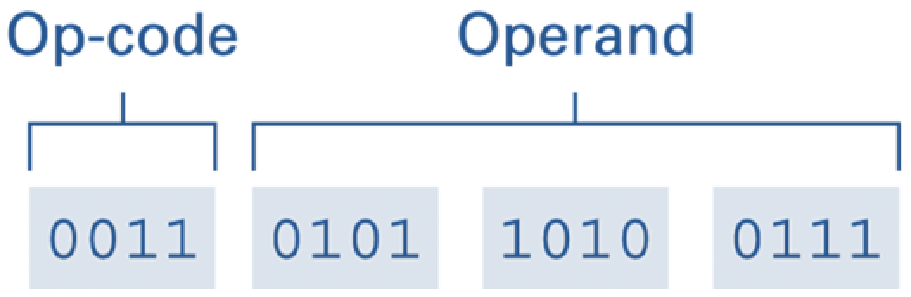
Stored Program Concept

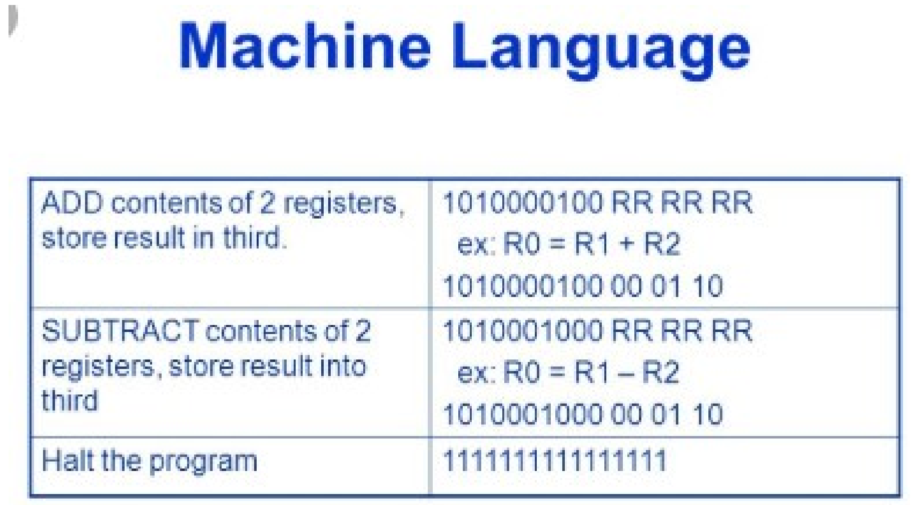
- A program can be encoded as bit patterns and stored in main memory.
- From there, the CPU can then extract the instructions and execute them.
- In turn, the program to be executed can be altered easily.

Terminology

- Machine instruction: An instruction (or command) encoded as a bit pattern recognizable by the CPU

- Machine language: The set of all instructions recognized by a machine





Machine Language

ADD contents of 2 registers, store result in third.	1010000100 RR RR RR ex: R0 = R1 + R2 1010000100 00 01 10
SUBTRACT contents of 2 registers, store result into third	1010001000 RR RR RR ex: R0 = R1 – R2 1010001000 00 01 10
Halt the program	1111111111111111

Machine Language Philosophies

- Reduced Instruction Set Computing (RISC)
  - Few, simple, efficient, and fast instructions
  - Examples: PowerPC from Apple/IBM/Motorola and ARM
- Complex Instruction Set Computing (CISC)
  - Many, convenient, and powerful instructions
  - Example: Intel

Machine Instruction Types

- **Data Transfer:** copy data from one location to another
- **Arithmetic/Logic:** use existing bit patterns to compute a new bit patterns
- **Control:** direct the execution of the program

Adding values stored in memory

Step 1. Get one of the values to be added from memory and place it in a register.

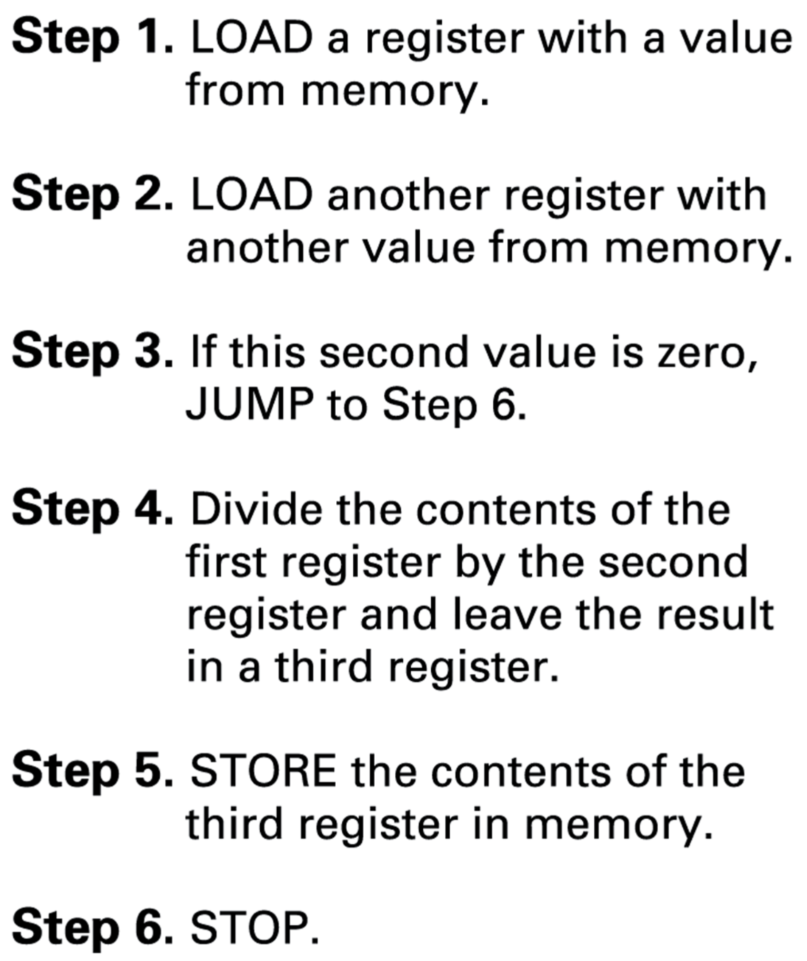
Step 2. Get the other value to be added from memory and place it in another register.

Step 3. Activate the addition circuitry with the registers used in Steps 1 and 2 as inputs and another register designated to hold the result.

Step 4. Store the result in memory.

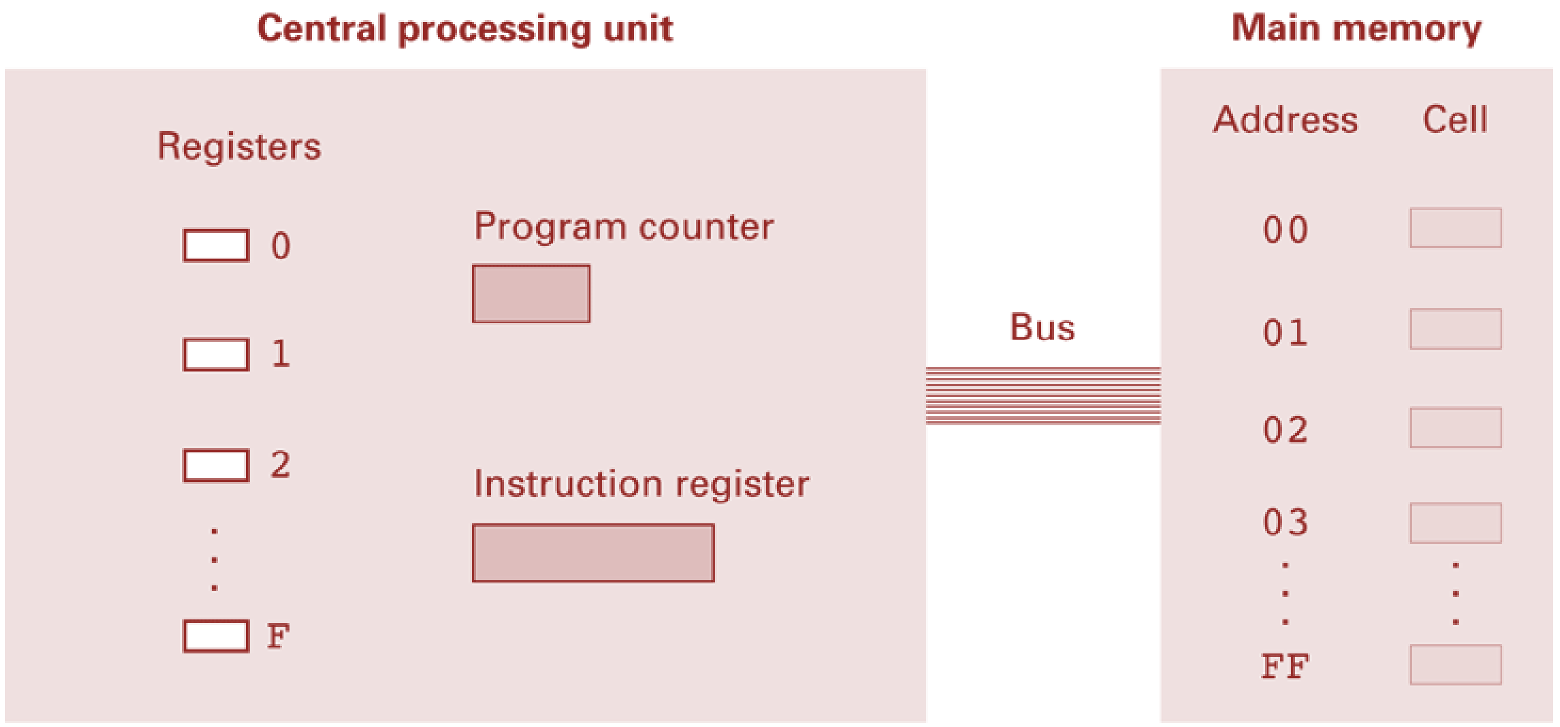
Step 5. Stop.

Dividing values stored in memory



```
Step 1. LOAD a register with a value
        from memory.

Step 2. LOAD another register with
        another value from memory.

Step 3. If this second value is zero,
        JUMP to Step 6.

Step 4. Divide the contents of the
        first register by the second
        register and leave the result
        in a third register.

Step 5. STORE the contents of the
        third register in memory.

Step 6. STOP.
```

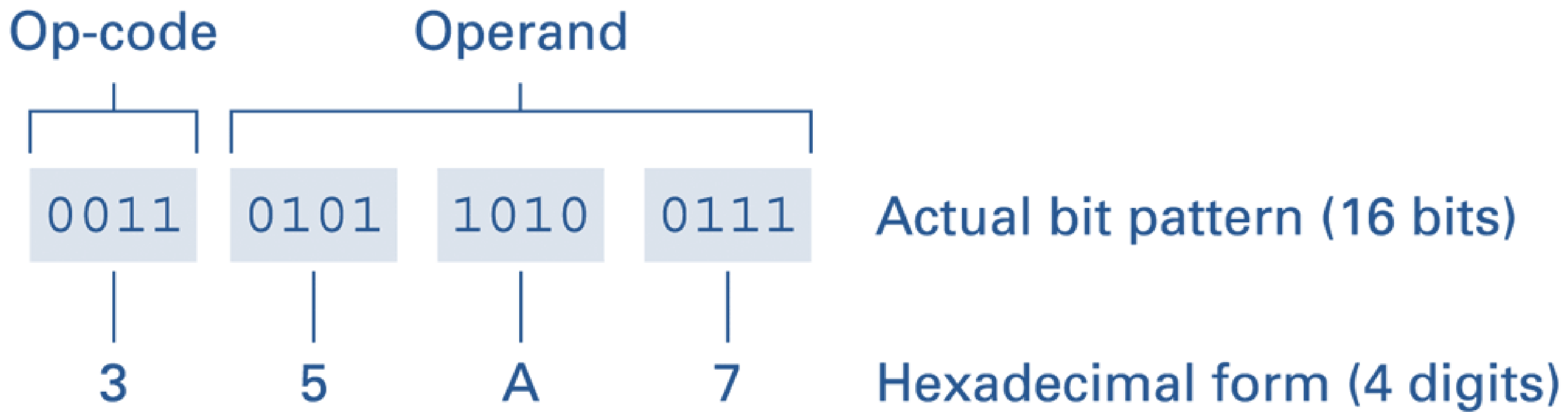
An architecture of the machine



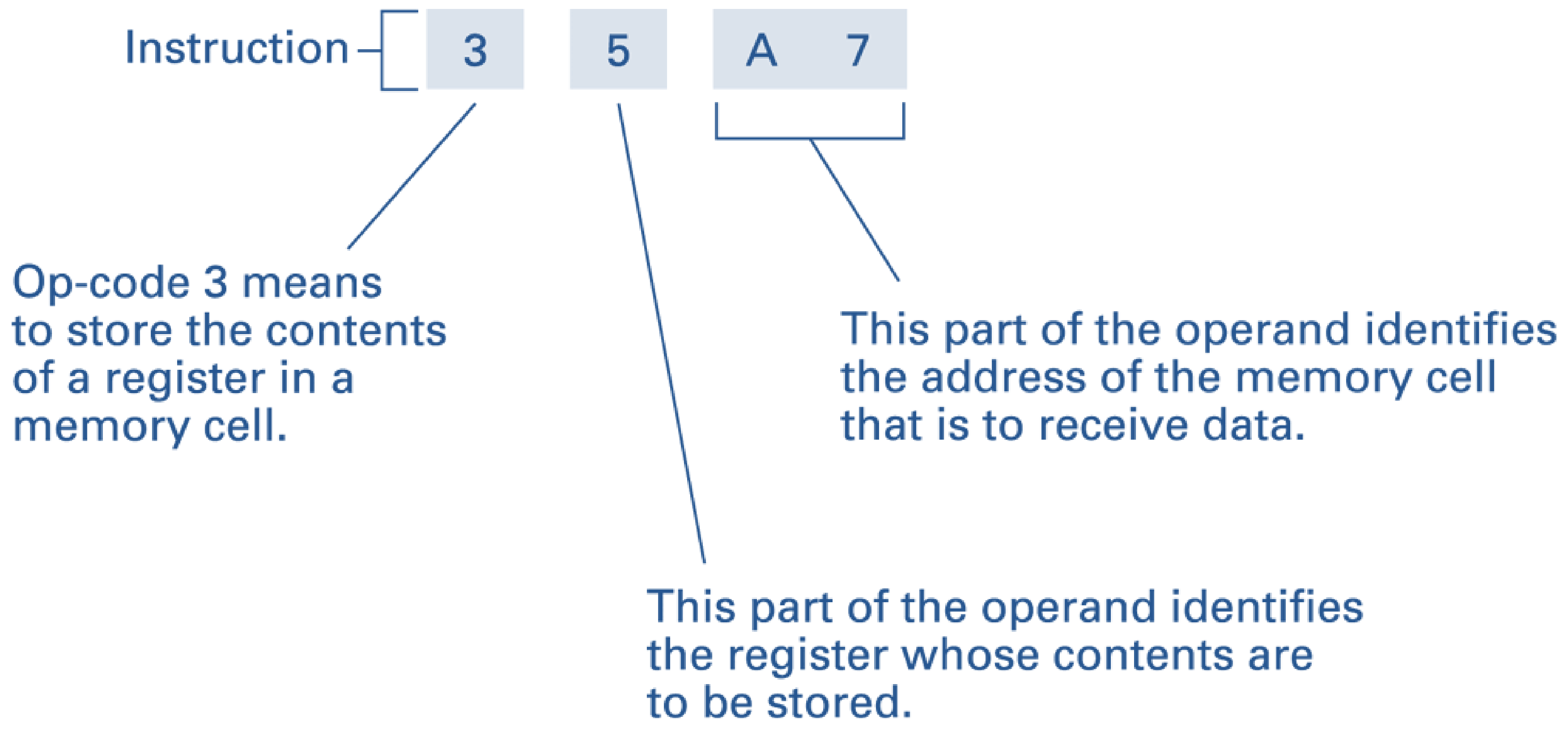
Parts of a Machine Instruction

- **Op-code**: Specifies which operation to execute
- **Operand**: Gives more detailed information about the operation
  - Interpretation of operand varies depending on op-code

The composition of an instruction



Decoding the instruction 35A7


Instruction
3
5
A 7
Op-code 3 means to store the contents of a register in a memory cell.
This part of the operand identifies the address of the memory cell that is to receive data.
This part of the operand identifies the register whose contents are to be stored.

A Simple Machine Language

Op-code	Operand	Description
1	RXY	LOAD reg. R from cell XY.
2	RXY	LOAD reg. R with XY.
3	RXY	STORE reg. R at XY.
4	0RS	MOVE R to S.
5	RST	ADD S and T into R. (2’s comp.)
6	RST	ADD S and T into R. (floating pt.)

A Simple Machine Language

Op-code	Operand	Description
7	RST	OR S and T into R.
8	RST	AND S and T into R.
9	RST	XOR S and T into R.
A	R0X	ROTATE reg. R X times.
B	RXY	JUMP to XY if R = reg. 0.
C	0	HALT.

A Simple Machine Language

- 14A3: Load
  - The contents of the **memory cell** located at address **A3** to be placed in register 4.
- 20A3: Load
  - The value **A3** to be placed in register 0.
- 35B1: Store
  - The contents of register 5 to be placed in the memory cell whose address is **B1**.
- 40A4: Move
  - The contents of register **A** to be copied into register **4**.

A Simple Machine Language

- 5726: Add
  - The binary values in registers 2 and 6 to be added and the sum placed in register 7.
- 634E: Add
  - The values in registers 4 and E to be added as **floating-point** values and the result to be placed in register 3.

A Simple Machine Language

- 7CB4: Or
  - The result of ORing the contents of registers B and 4 to be placed in register C.
- 8045: And
  - The result of ANDing the contents of registers 4 and 5 to be placed in register 0.
- 95F3: Xor
  - The result of XORing the contents of registers F and 3 to be placed in register 5.
- A403: Rotate
  - The contents of registers 4 to be rotated 3 bits to the right in a circular fashion.

A Simple Machine Language

- B43C:
  - Compare the contents of register **4** with the contents of register 0.
  - If equal, the pattern **3C** would be placed in the program counter so that the next execution executed would be the one located at that memory address.
  - Otherwise, nothing would be done.
- C000:
  - Stop program execution.

An encoded version of the instructions

Encoded instructions	Translation
156C	Load register 5 with the bit pattern found in the memory cell at address 6C.
166D	Load register 6 with the bit pattern found in the memory cell at address 6D.
5056	Add the contents of register 5 and 6 as though they were two's complement representation and leave the result in register 0.
306E	Store the contents of register 0 in the memory cell at address 6E.
C000	Halt.

QUIZ

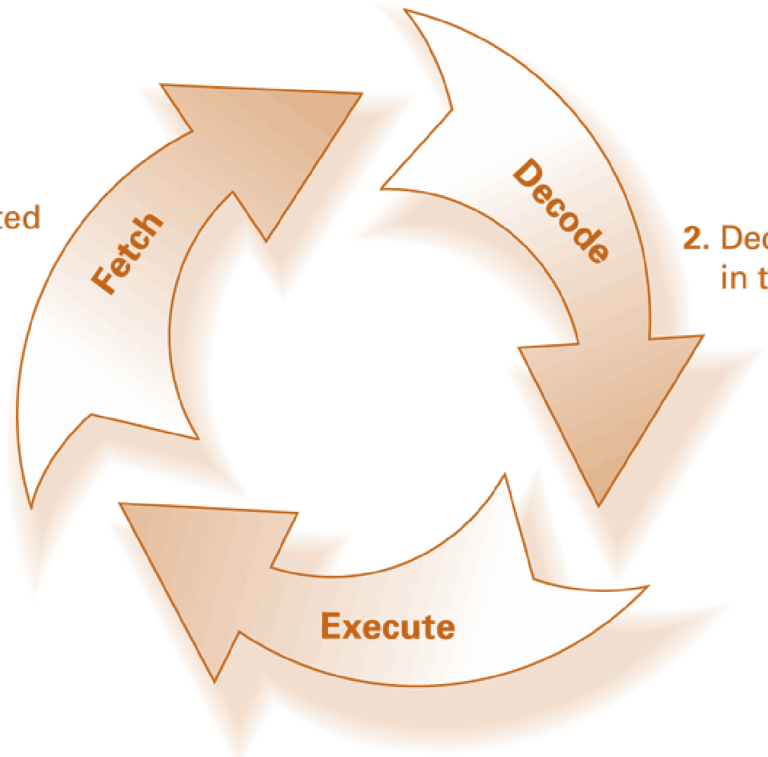
PROGRAM EXECUTION

Program Execution

- Controlled by two special-purpose registers
  - Program counter: address of next instruction
  - Instruction register: current instruction
- Machine Cycle
  - Fetch
  - Decode
  - Execute

The machine cycle

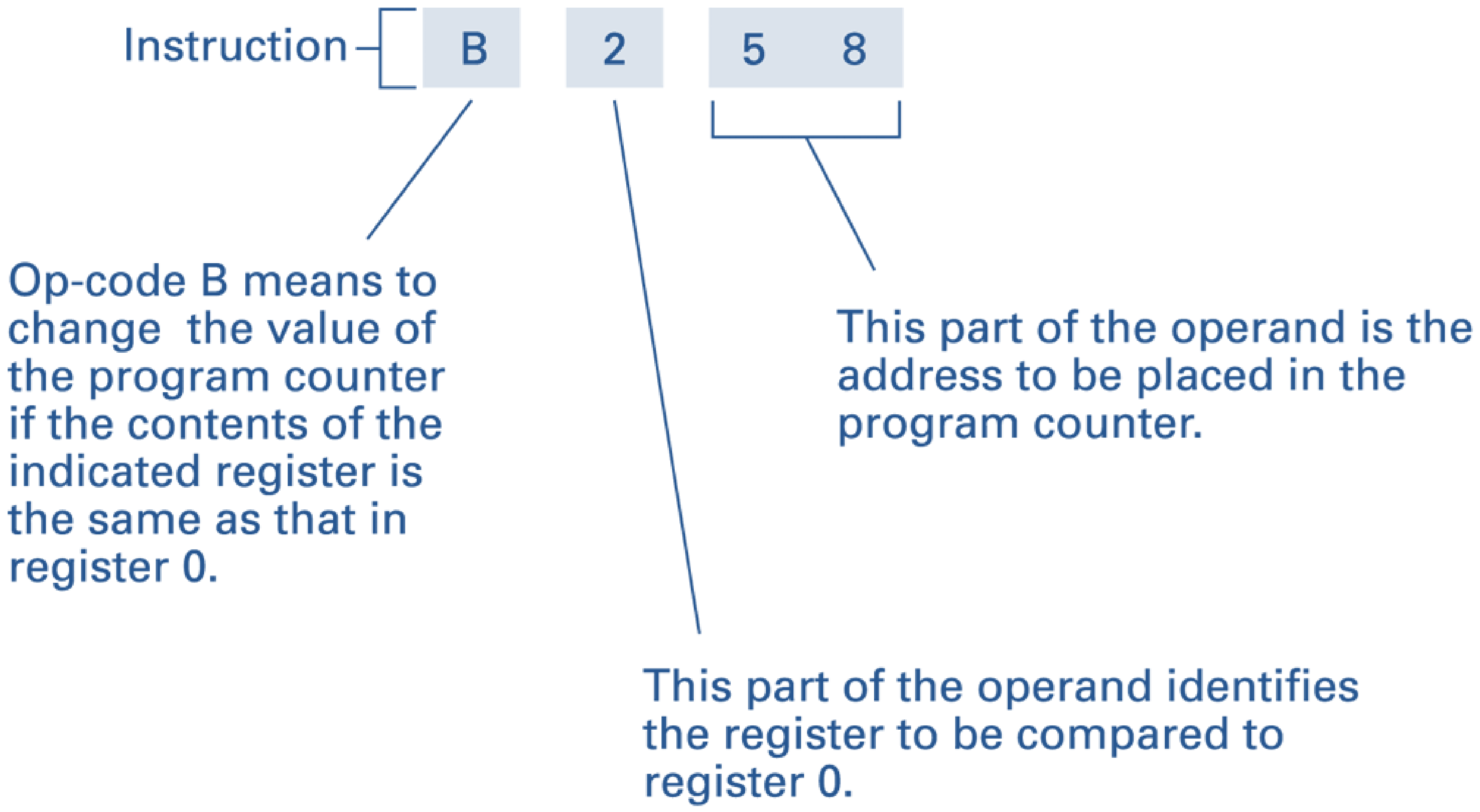
1. Retrieve the next instruction from memory (as indicated by the program counter) and then increment the program counter.



2. Decode the bit pattern in the instruction register.

3. Perform the action required by the instruction in the instruction register.

Decoding the instruction B258

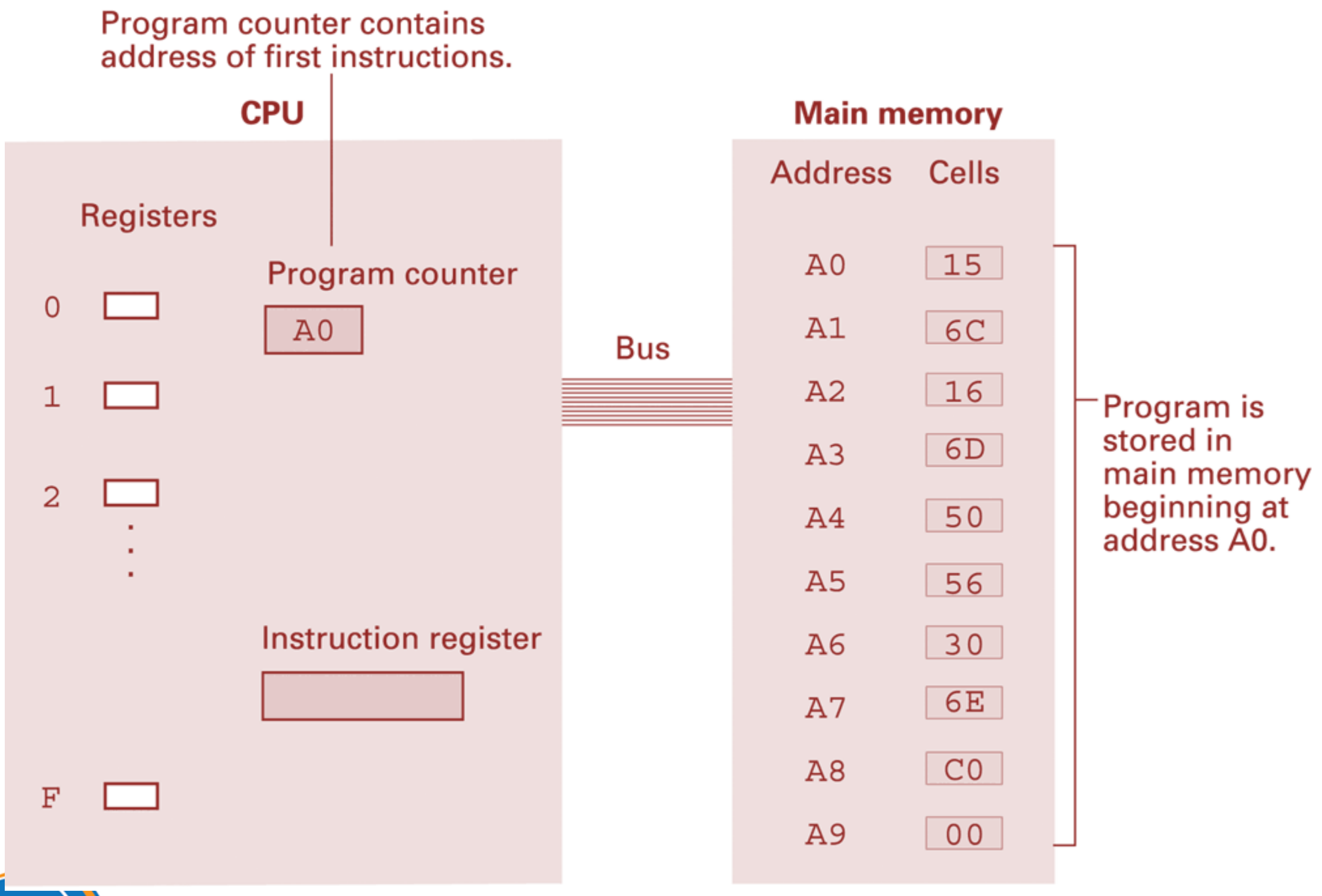


Op-code B means to change the value of the program counter if the contents of the indicated register is the same as that in register 0.

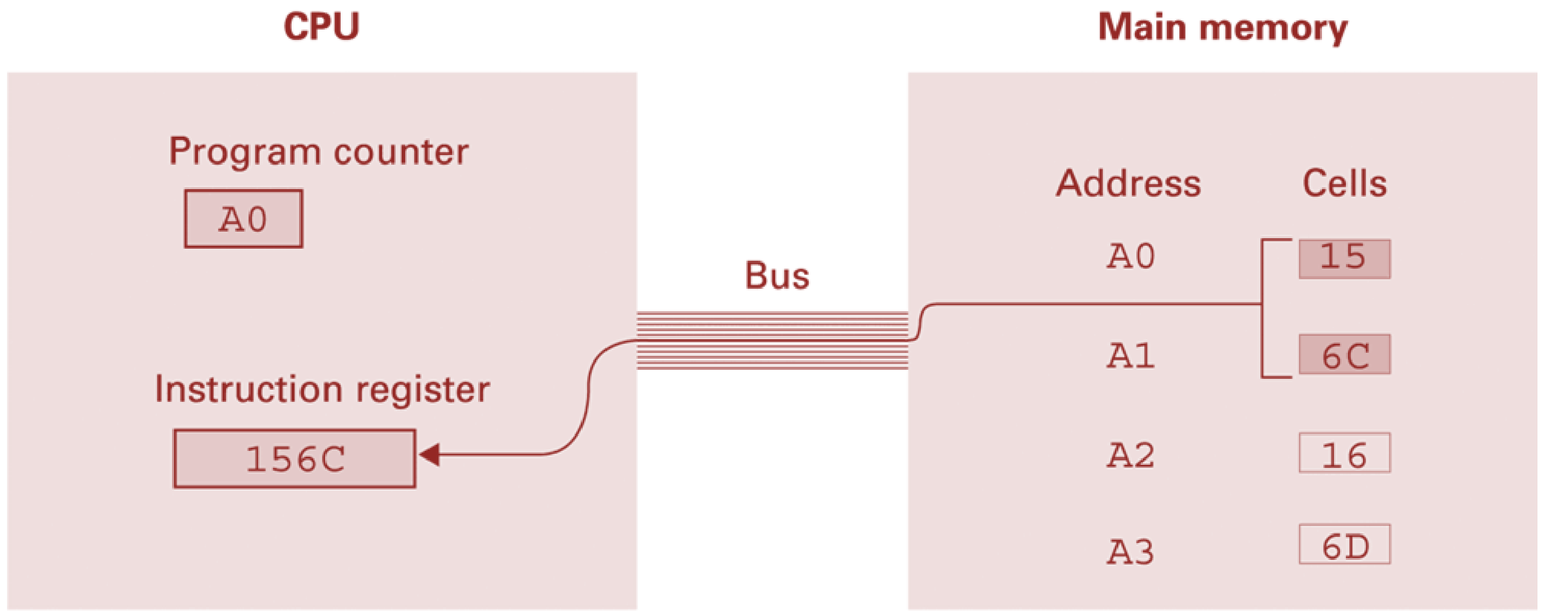
This part of the operand is the address to be placed in the program counter.

This part of the operand identifies the register to be compared to register 0.

A program stored in main memory

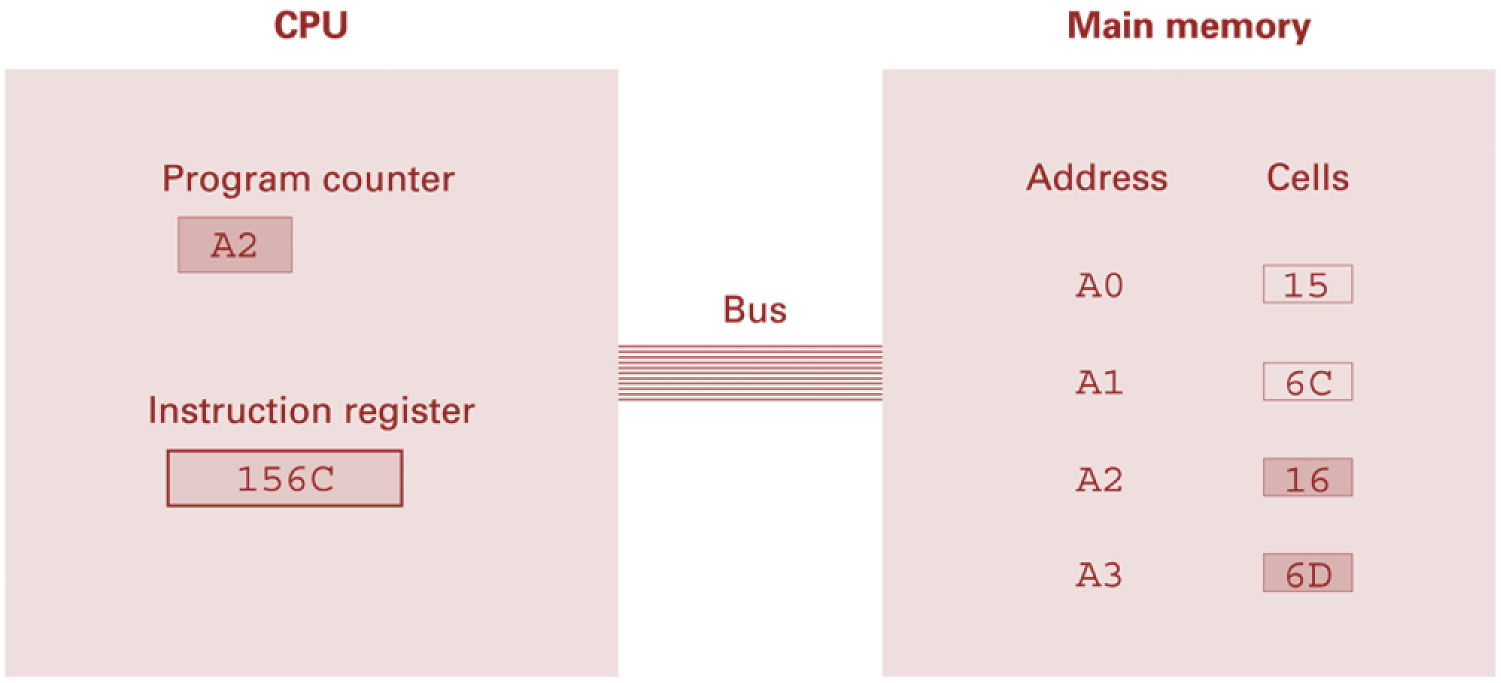


The fetch step of the machine cycle



a. At the beginning of the fetch step the instruction starting at address A0 is retrieved from memory and placed in the instruction register.

The fetch step of the machine cycle



b. Then the program counter is incremented so that it points to the next instruction.

ARITHMETIC/LOGIC INSTRUCTIONS

Arithmetic/Logic Operations

- Logic: AND, OR, XOR
  - Masking
- Rotate and Shift:
  - circular shift (rotation)
  - logical shift
  - arithmetic shift
- Arithmetic: add, subtract, multiply, divide
  - Precise action depends on how the values are encoded (two’s complement versus floating-point).

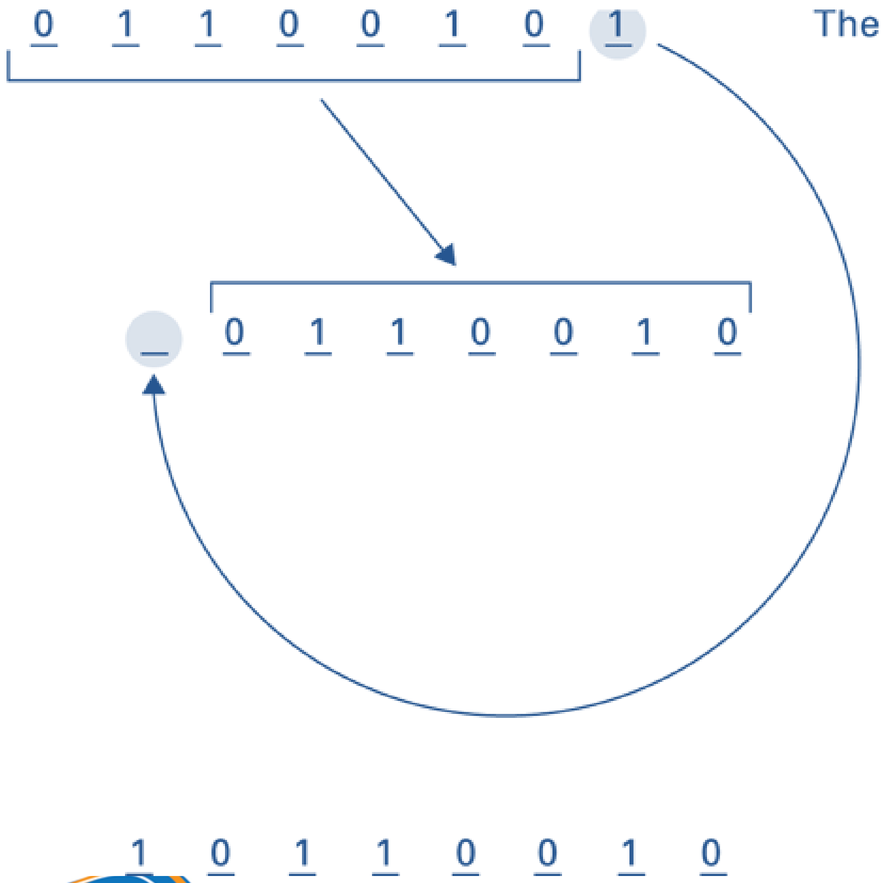
Masking

$$\begin{array}{r} \boxed{00001111} \\ \underline{\text{AND}\ \ 10101010} \\ 00001010 \end{array}$$

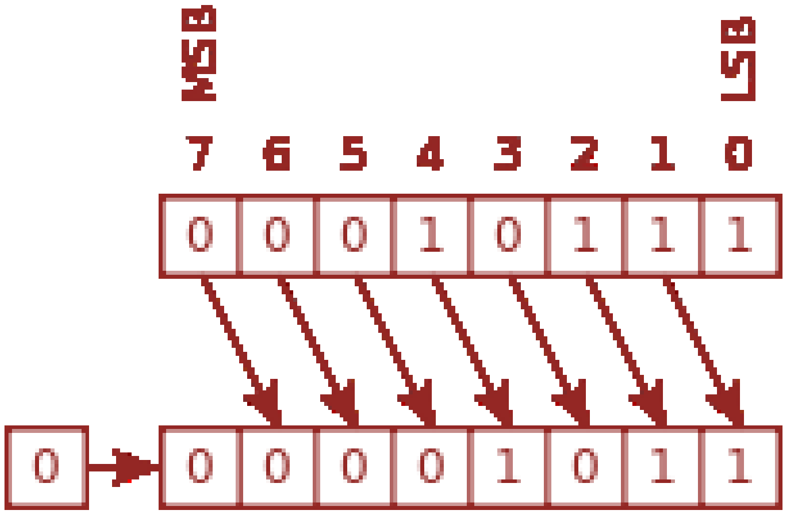
$$\begin{array}{r} 11110000 \\ \underline{\text{OR}\ \ 10101010} \\ 11111010 \end{array}$$

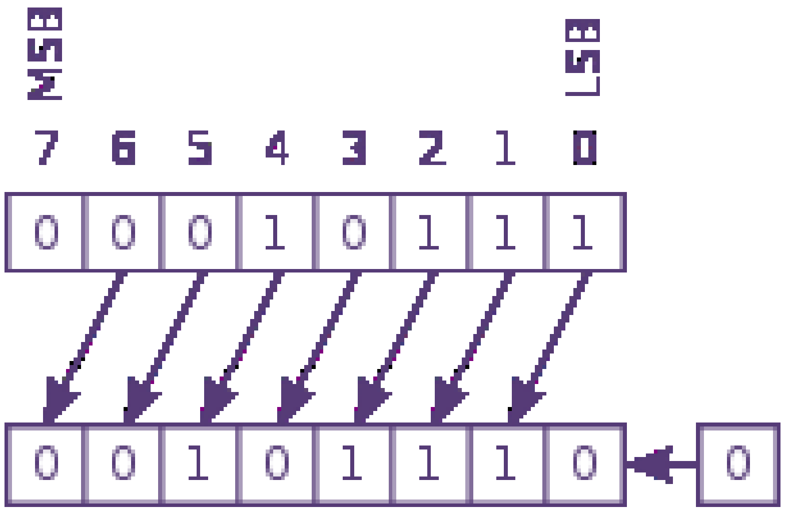
$$\begin{array}{r} 11111111 \\ \underline{\text{XOR}\ \ 10101010} \\ 01010101 \end{array}$$

Circular shift


0 1 1 0 0 1 0 1
The original bit pattern
_ 0 1 1 0 0 1 0
The bits move one position to the right. The rightmost bit "falls off" the end and is placed in the hole at the other end.
1 0 1 1 0 0 1 0
The final bit pattern

Logical shift





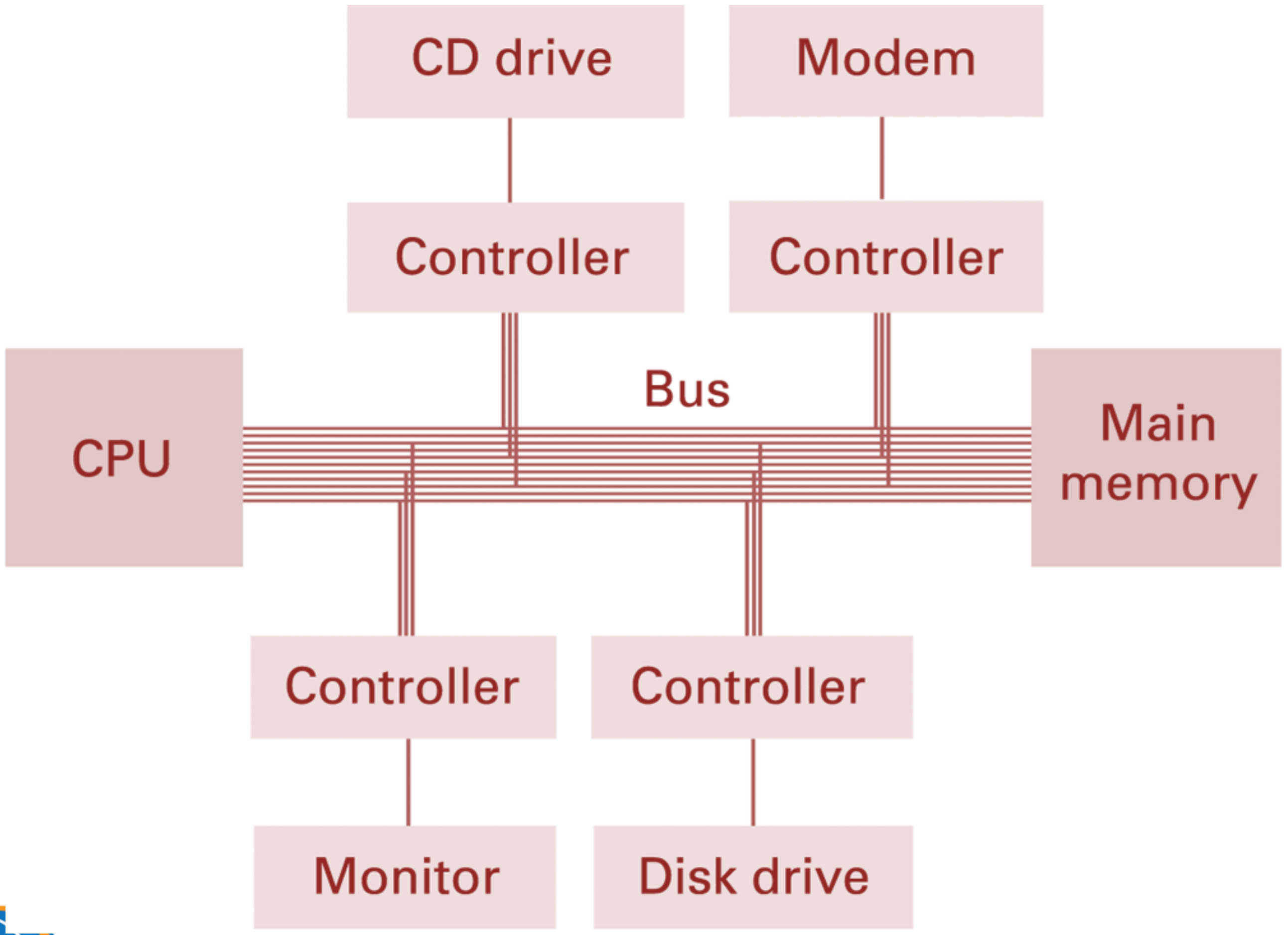
QUIZ

COMMUNICATING WITH OTHER DEVICES

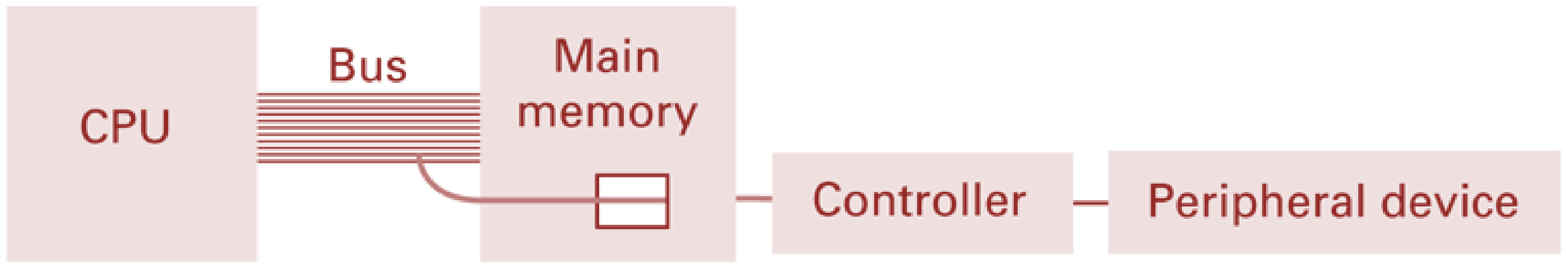
Communicating with Other Devices

- Controller: An intermediary apparatus that handles communication between the computer and a device
  - Specialized controllers for each type of device
  - General purpose controllers (USB and FireWire)
- Port: The point at which a device connects to a computer
- Memory-mapped I/O: CPU communicates with peripheral devices as though they were memory cells

Controllers attached to a machine’s bus



A conceptual representation of memory-mapped I/O

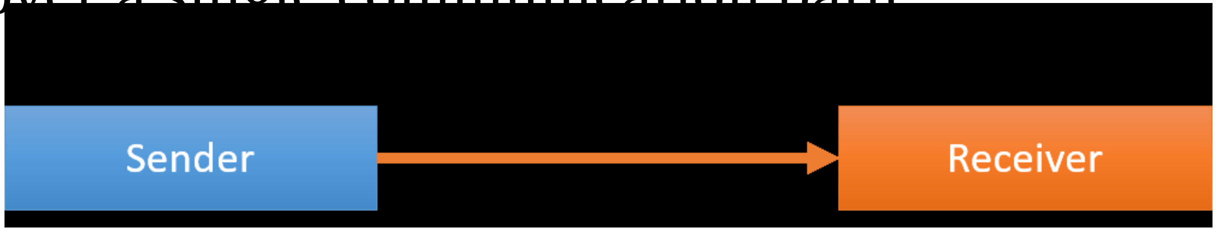


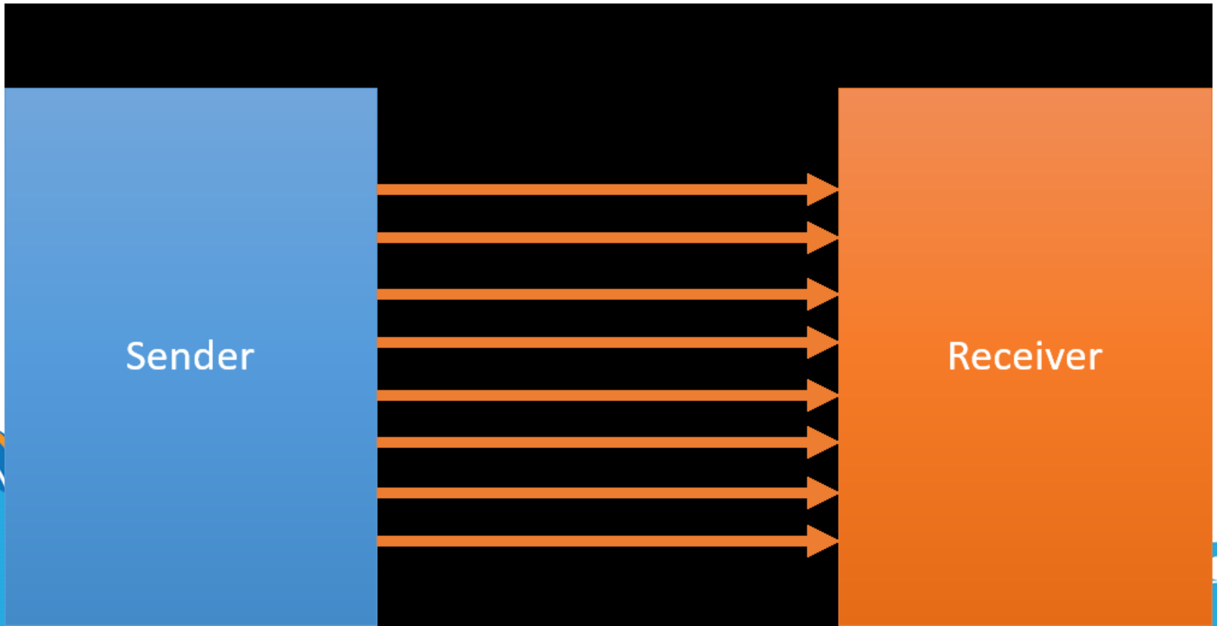
Communicating with Other Devices

- Direct memory access (DMA): Main memory access by a controller over the bus
- Von Neumann Bottleneck: Insufficient bus speed impedes performance
- Handshaking: The process of coordinating the transfer of data between components

Communicating with Other Devices

- Parallel Communication: Several communication paths transfer bits simultaneously.
- Serial Communication: Bits are transferred one after the other over a single communication path.





Data Communication Rates

- Measurement units
  - Bps: Bits per second
  - Kbps: Kilo-bps (1,000 bps)
  - Mbps: Mega-bps (1,000,000 bps)
  - Gbps: Giga-bps (1,000,000,000 bps)
- Bandwidth: Maximum available rate

Quiz

- Assume that the machine (using the simple machine language as described) uses memory-mapped I/O and that the address B5 is the location within the printer port to which data to be printed should be sent.

a. If register 7 contains the ASCII code for the letter A, what machine language instruction should be used to cause that letter to be printed at the printer?

b. If the machine executes a million instructions per second, how many times can this character be sent to the printer in one second?

Quiz

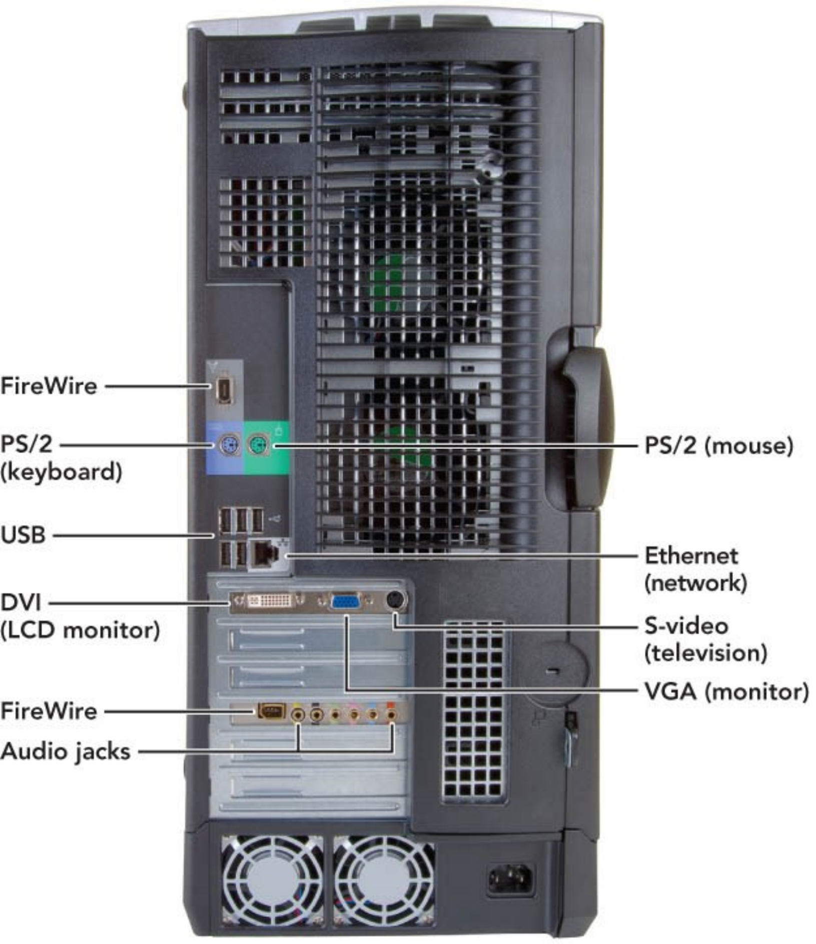
- Assume that the machine (using the simple machine language as described) uses memory-mapped I/O and that the address B5 is the location within the printer port to which data to be printed should be sent.

c. If the printer is capable of printing five traditional pages of text per minute, will it be able to keep up with the characters being sent to it in (b)?

Quiz

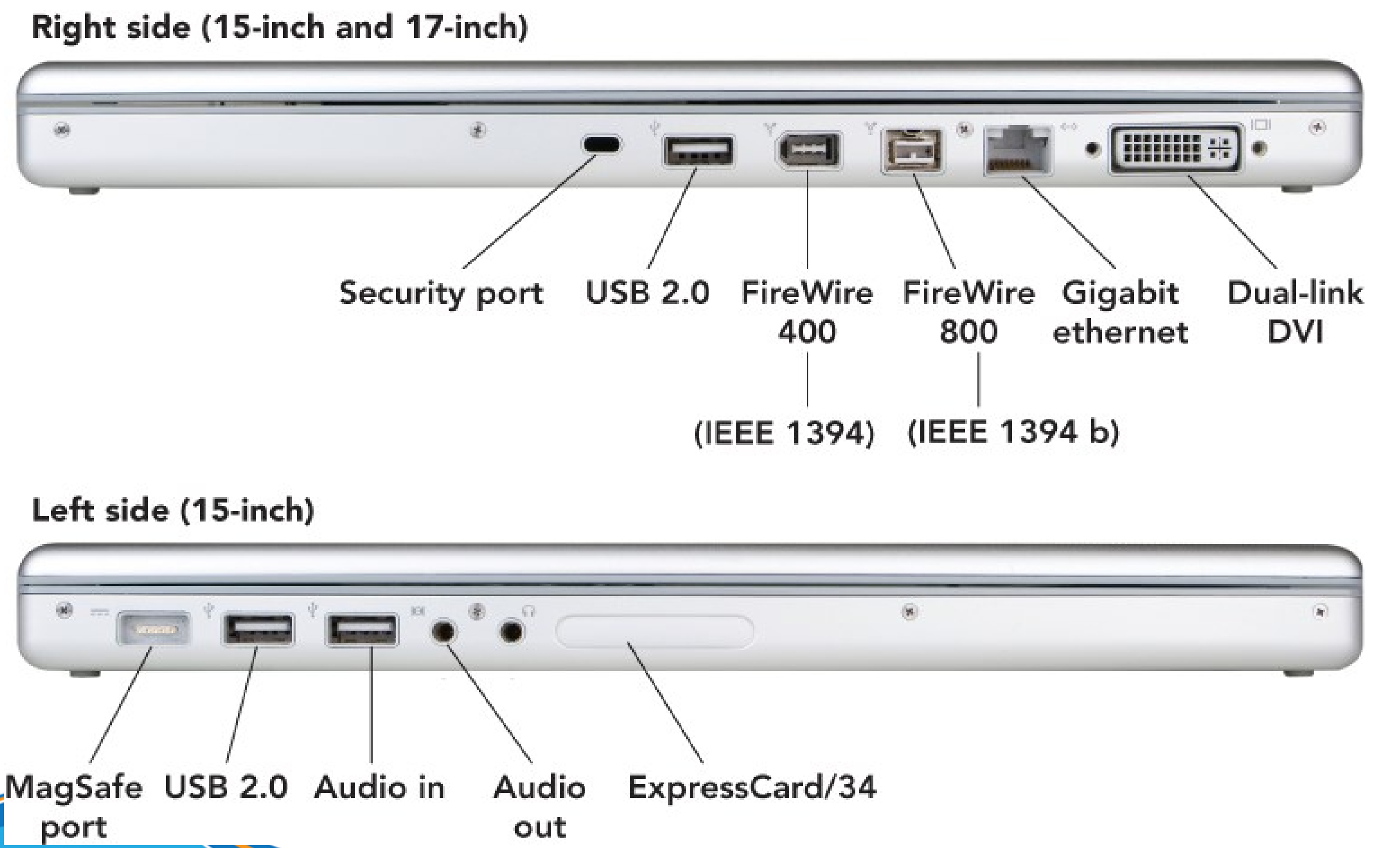
- Estimate how long it would take to transfer a 250-page novel (about 500,000 characters) encoded in 16-bit Unicode characters at a transfer rate of 54Mbps.

What’s on the Outside of the Box?



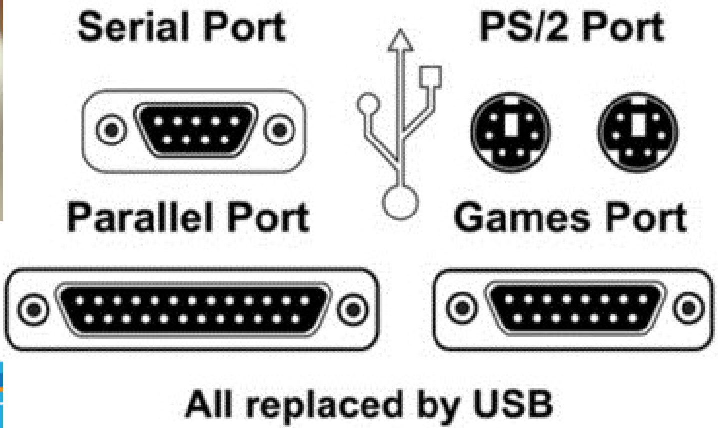
What's on the Outside of the Box?

- Connectors on a notebook may vary



Universal serial bus

- USB (universal serial bus) ports
  - Connects up to 127 peripheral devices
  - USB 2.0 (high-speed USB)—fully compatible with USB 1.1 products, cables, and connectors
  - Designed to replace older parallel and serial ports
  - Connects a variety of devices to the computer, including:
    - Keyboards
    - Mice
    - Printers
    - Digital cameras



Universal serial bus

- USB 2.0
  - Uses an external bus
  - Supports data transfer rates of 480 Mbps between the computer and the peripheral device
  - Supports hot swapping—ability to connect and disconnect devices without shutting down the computer
  - Plug-and-play (PnP)—allows computers to automatically detect the device when you plug it in
- USB 3.0 (2008): 5Gbps
- USB 3.1 (2013): 10Gbps
- USB hub
  - Device that plugs into existing USB port
  - Contains four or more additional ports

FireWire

- FireWire (1395 ports)
  - Created by Apple in 1995
  - IEEE 1394 Higher Performance Serial Bus, also known as Sony i.Link
  - Offers high-speed connections for dozens of peripheral devices (up to 63)
  - Enables hot swapping and PnP
  - Data transfer rates of FireWire
    - FireWire 400—400 Mbps
    - FireWire 800—800 Mbps
    - Declared “dead” by Steve Jobs.

Video connectors

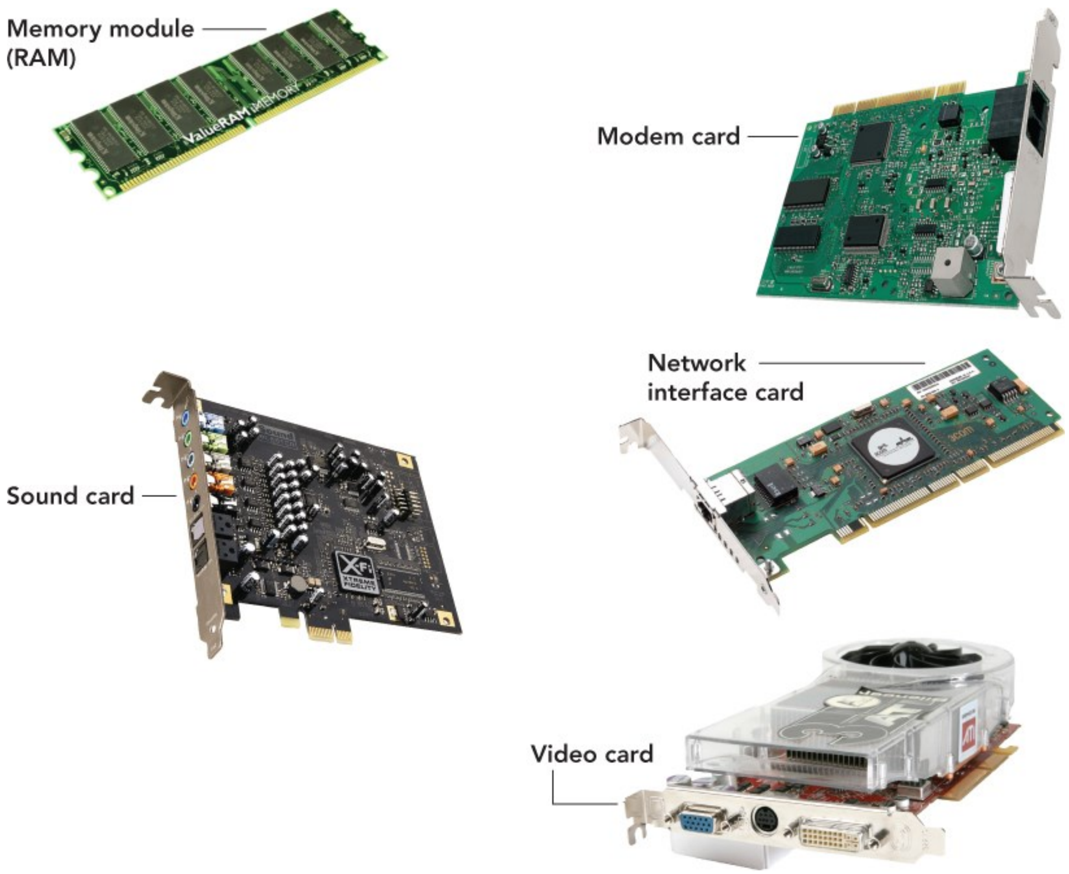
- VGA (video graphics array)
  - 15-pin male connector—works with standard monitor cables
  - Transmits analog video signals
  - Used for legacy technology cathode ray (CRT) monitors
- DVI (Digital visual Interface) port—lets LCD monitors use digital signals
- Onboard video—video circuitry built into the motherboard where the video connector is on the back of the system unit case

HDMI

- High-Definition Multimedia Interface
  - A video/audio interface for transmitting uncompressed video data, and compressed/uncompressed audio data.
  - Digital replacement for analog video standards.
  - Designed: 2002 (7 companies).
  - HDMI 2.1: 48Gbps.
  - Newer version: 3D, Ethernet data connection, Consumer Electronic Control (CEC) extensions.

Additional connectors

- Telephone
- Network
- PC card slot
  - PC card
  - ExpressCard
- Sound card
- Game card
- TV/sound capture board

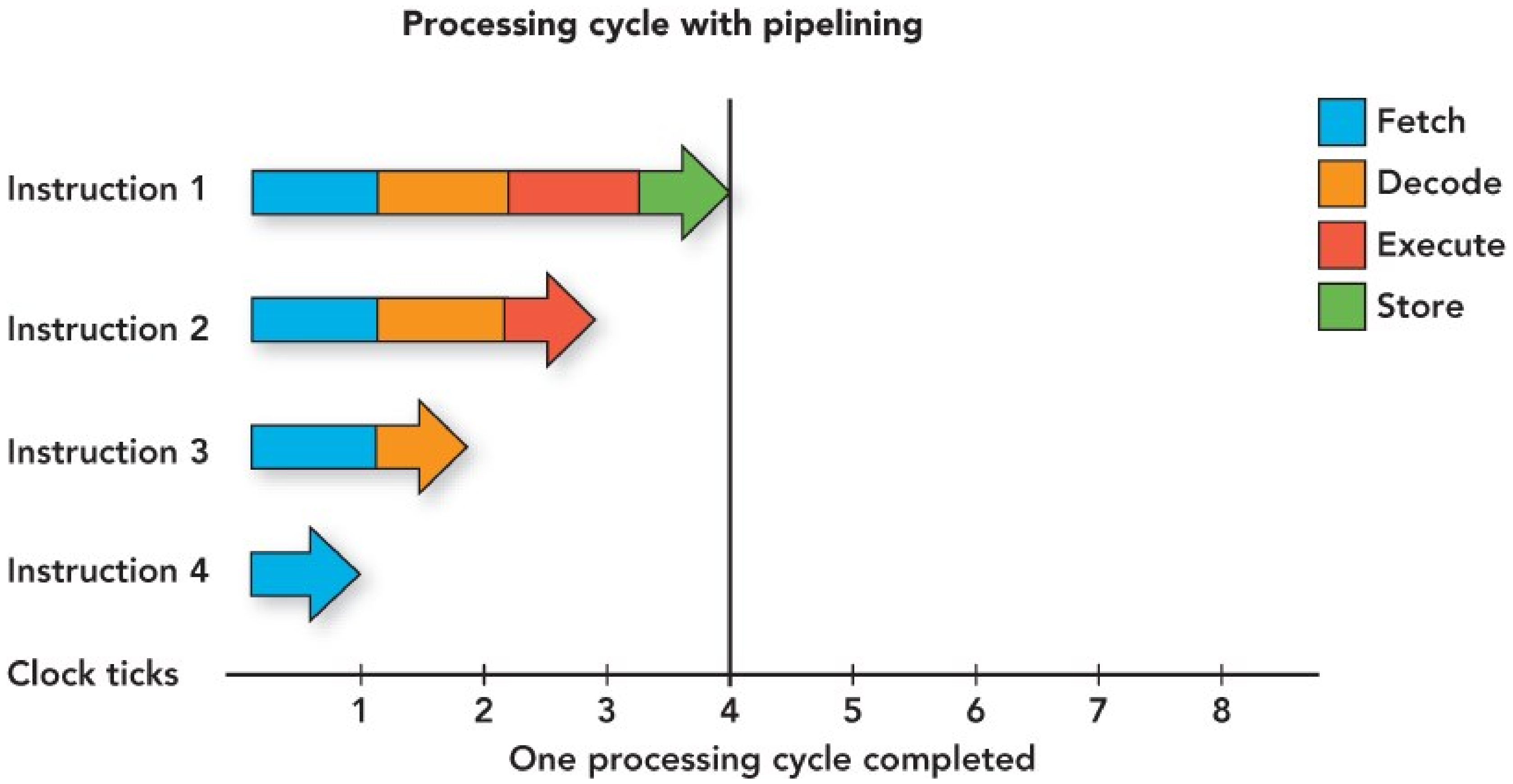


OTHER ARCHITECHTURES

Other Architectures

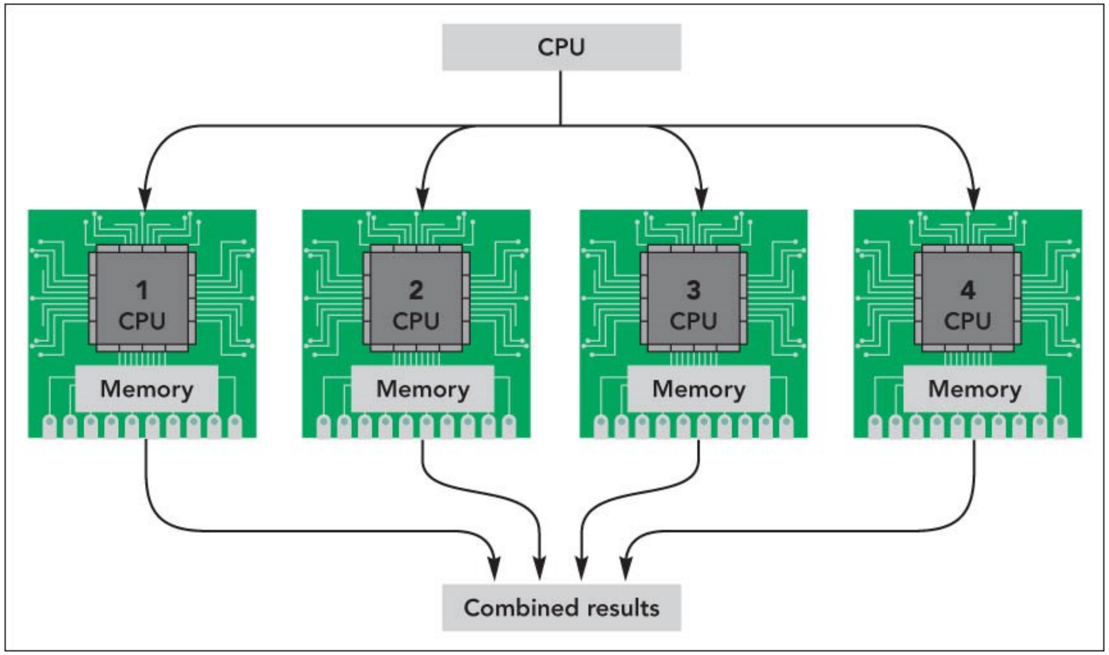
- Technologies to increase throughput:
  - Pipelining: Overlap steps of the machine cycle
  - Parallel Processing: Use multiple processors simultaneously
    - SISD: No parallel processing
    - MIMD: Different programs, different data
    - SIMD: Same program, different data

Pipelining


Processing cycle with pipelining
Instruction 1
Instruction 2
Instruction 3
Instruction 4
Clock ticks
1
2
3
4
5
6
7
8
One processing cycle completed
Fetch
Decode
Execute
Store

Parallel processing

- Method where more than one processor performs at the same time—faster processing


CPU
1 CPU
Memory
2 CPU
Memory
3 CPU
Memory
4 CPU
Memory
Combined results