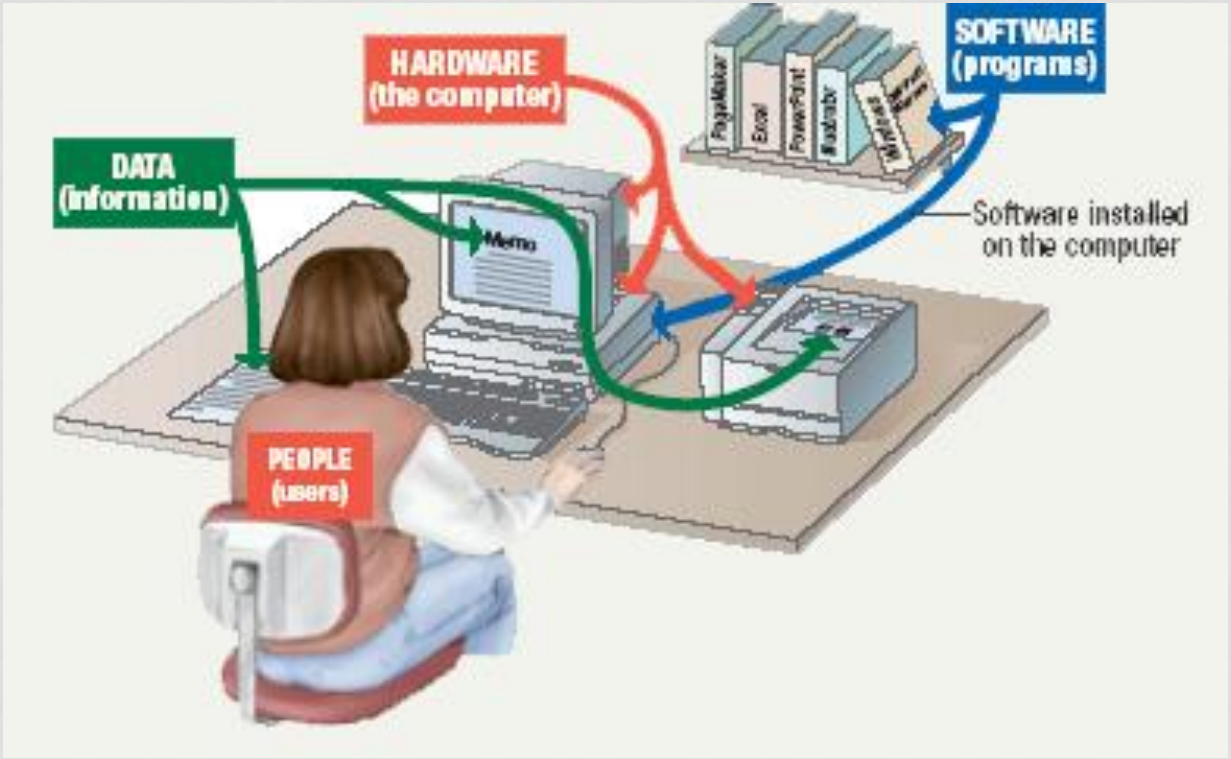
COMPUTER ORGANIZATION

Major Topics for Today

1. Buses
2. I/O Controller
3. Registers
4. Caches
5. Fetch-Decode-Execute Cycle
6. Pipelining

COMPUTER ORGANIZATION

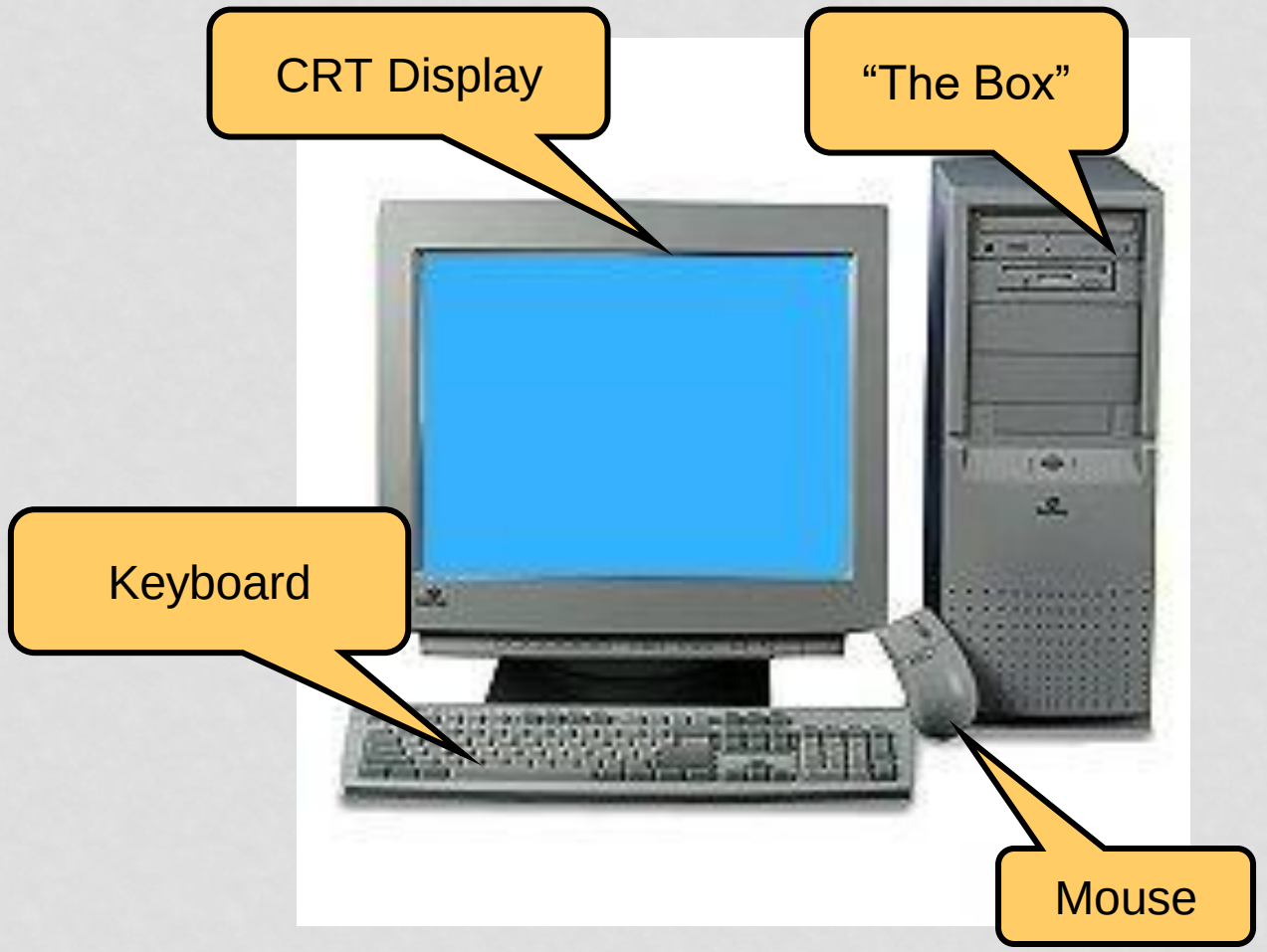
- It deals with the hardware components of a computer system such as the input/output devices, CPU and memory devices



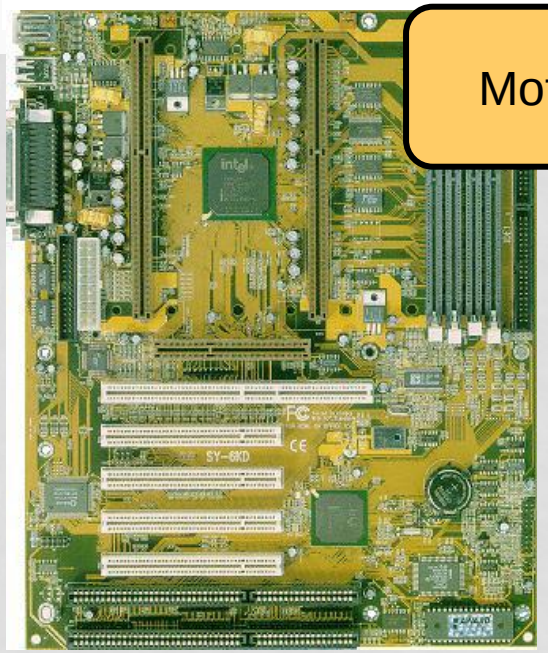
HARDWARE

ESSENTIAL COMPUTER HARDWARE

- Computers use the same basic hardware



INSIDE THE BOX



Motherboard

(RAM)

CPU
(Central Processing Unit)

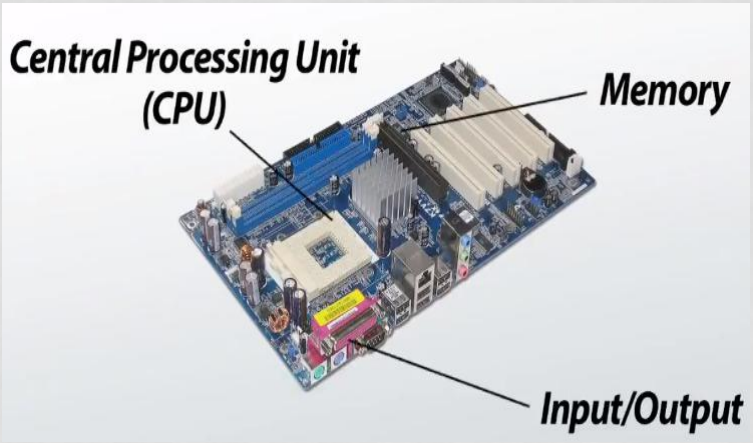
HDD
(Hard Disk Drive)

MOTHER-BOARD

It holds together many of the crucial components of a computer

Parts of a Motherboard

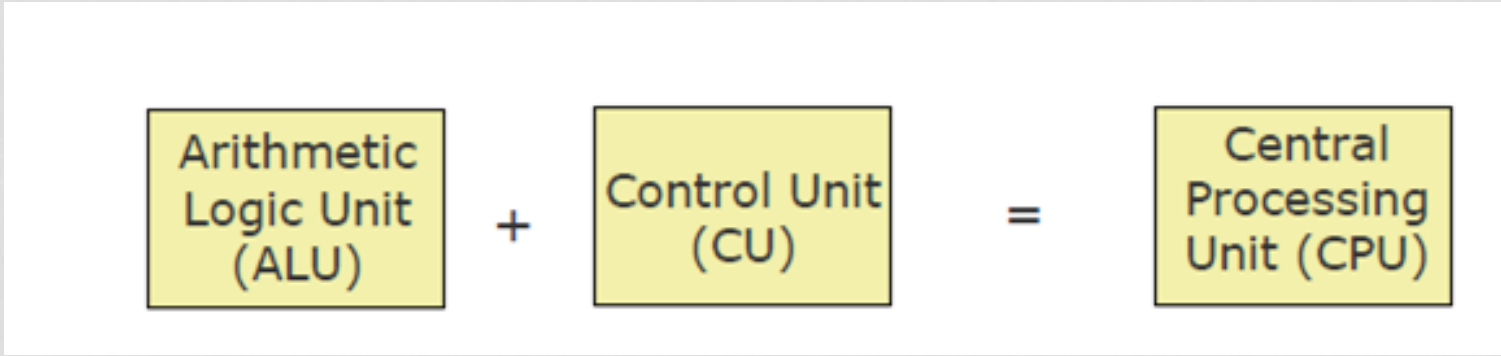
1. A CPU socket
2. A power connector
3. Slots for RAM
4. A second chip that controls the input and output (I/O) functions
5. Slots for one or more hard drives
6. A read-only memory (ROM) chip
7. A slot for a video or graphics card



CENTRAL PROCESSING UNIT

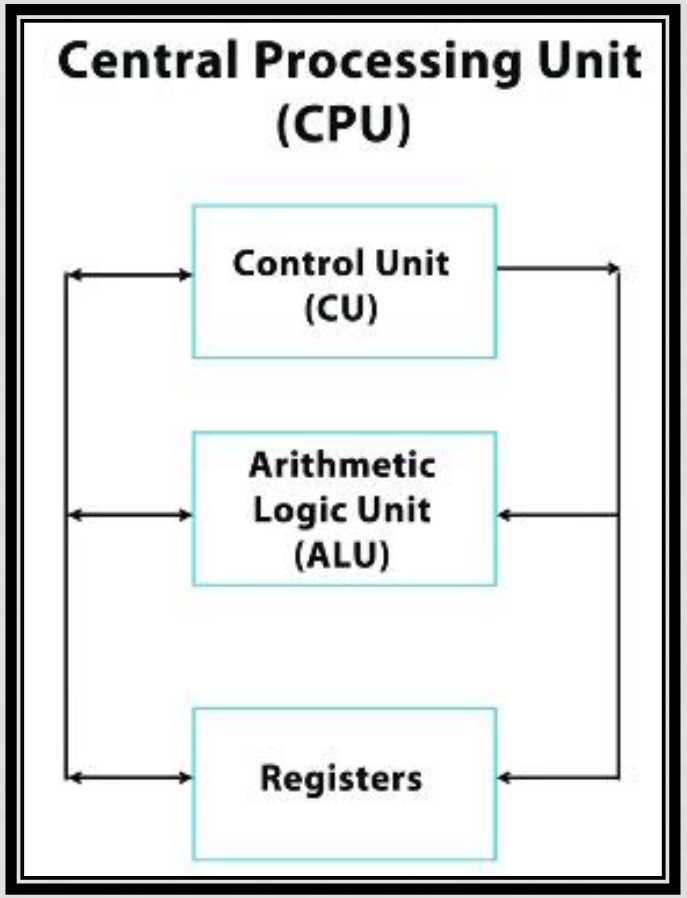
- It is the **brain** of a computer system.
- Carries out instructions from the program.
- Performs the basic arithmetical, logical, and input/output operations.
- !!Remember that this is all about speed.
- **Components**

1. ALU
2. CU
3. Cache



ARITHMETIC LOGIC UNIT- ALU

- An **arithmetic logic unit (ALU)** is a digital circuit used to perform arithmetic and logic operations.
- The **control unit** moves the data between these registers, the ALU, and memory.



HOW AN ALU WORKS

• **Arithmetic operations**	Logic operations
• addition • Subtraction • Multiplication • division	• comparisons of values such as • NOT • AND • OR

- There are also circuits for multiplying, for comparing two numbers, etc.
- **Transistor** switches are used to manipulate binary numbers
  - An open transistor represents a 0.
  - A closed transistor represents a 1.

CONTROL UNIT (CU)

- One of the two basic components of CPU
- Acts as the central nervous system of a computer system
- Selects and interprets program instructions, and coordinates execution
- Fetch □ decode □ execute □ write back
- Has some special purpose registers and a decoder to perform these activities
  - !!The **control unit** moves the data between these registers, the ALU, and memory.

RAM&ROM

- Stores data or programs
- Random Access Memory (RAM)
  - Volatile
  - Stores current data and programs
  - More RAM results in a faster system
  - 2 to 8 GB
- Read Only Memory (ROM)
  - Permanent storage of programs
  - Holds the computer boot directions

STORAGE DEVICES

- Hold data and programs permanently
- Different from RAM
- Magnetic storage
  - Floppy and hard drive
  - Uses a magnet to access data
- Optical storage
  - CD and DVD drives
  - Uses a laser to access data

I/O DEVICES

Allows the user to interact
- Some devices are input and output
  - Touch screens

- **Input Devices**
  - Any peripheral (piece of computer hardware equipment) used to provide data and control signals to a computer.
  - Allows the user to put data into the computer.
- **Output Devices**
- Any peripheral that deliver data
- Computer hardware equipment used to communicate the results of data processing carried out by a computer to the outside world.

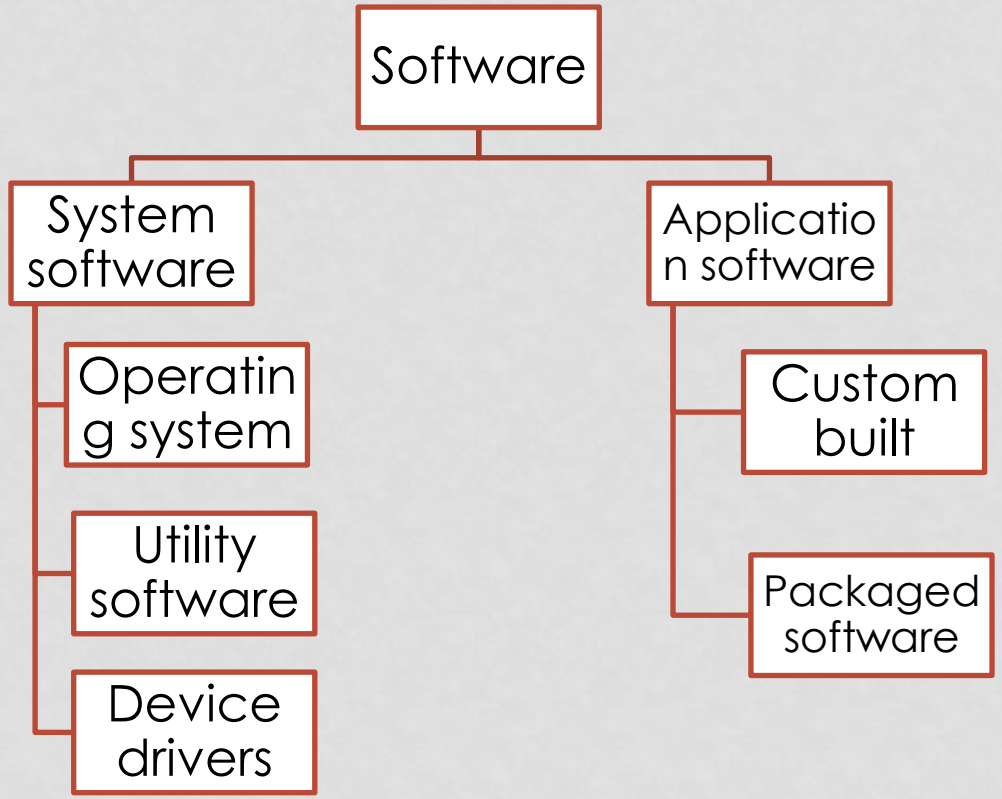
I/O CONTROLLERS

- The main purpose of this system is to help in the interaction of peripheral devices with the control units (CUs).
- Put simply, the I/O controller helps in the connection and control of various peripheral devices, which are input and output devices.
- I/O controllers are also known as channel I/O, DMA controllers, peripheral processors or I/O processors.

SOFTWARE

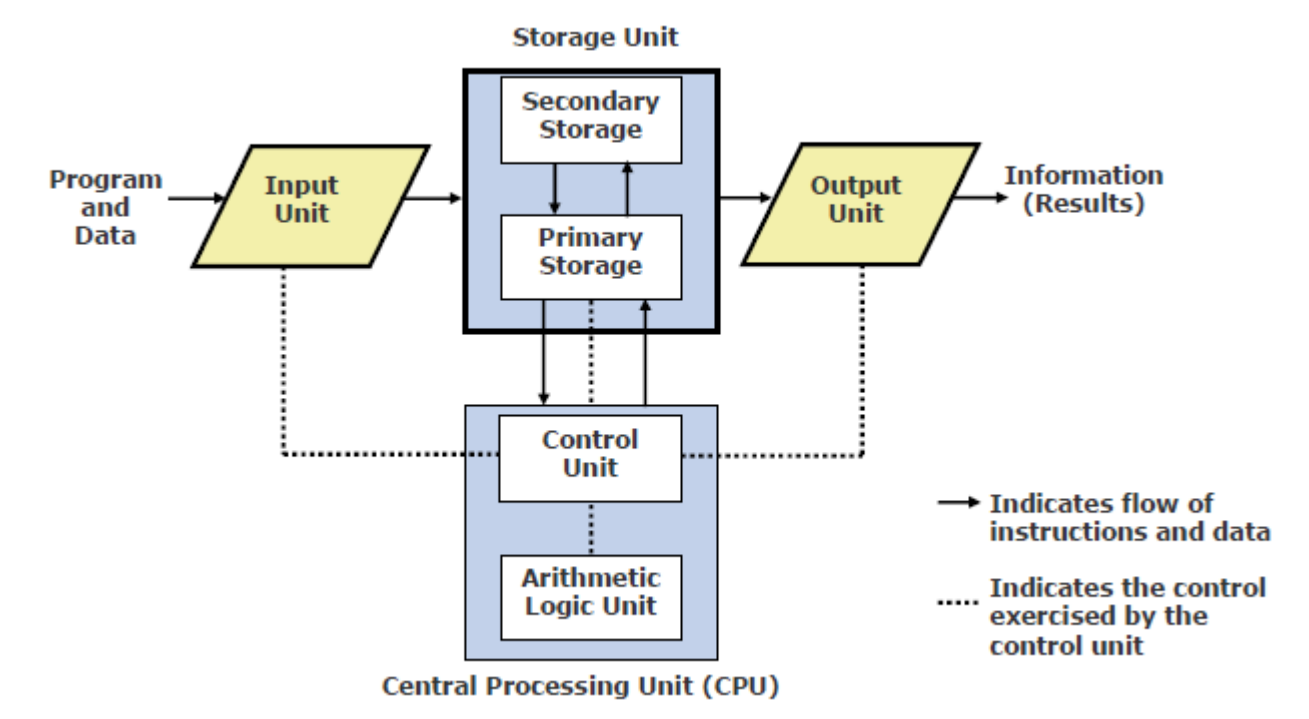
SOFTWARE

- Tells the computer what to do
- Reason people purchase computers
- Two types
  - System software
  - Application software
- System software
  - Most important software
  - Operating system
    - Windows XP
  - Network operating system (OS)
    - Windows Server 2003
  - Utility
    - Symantec Antivirus
- Application software
  - Accomplishes a specific task
  - Most common type of software
    - MS Word
  - Covers most common uses of computers



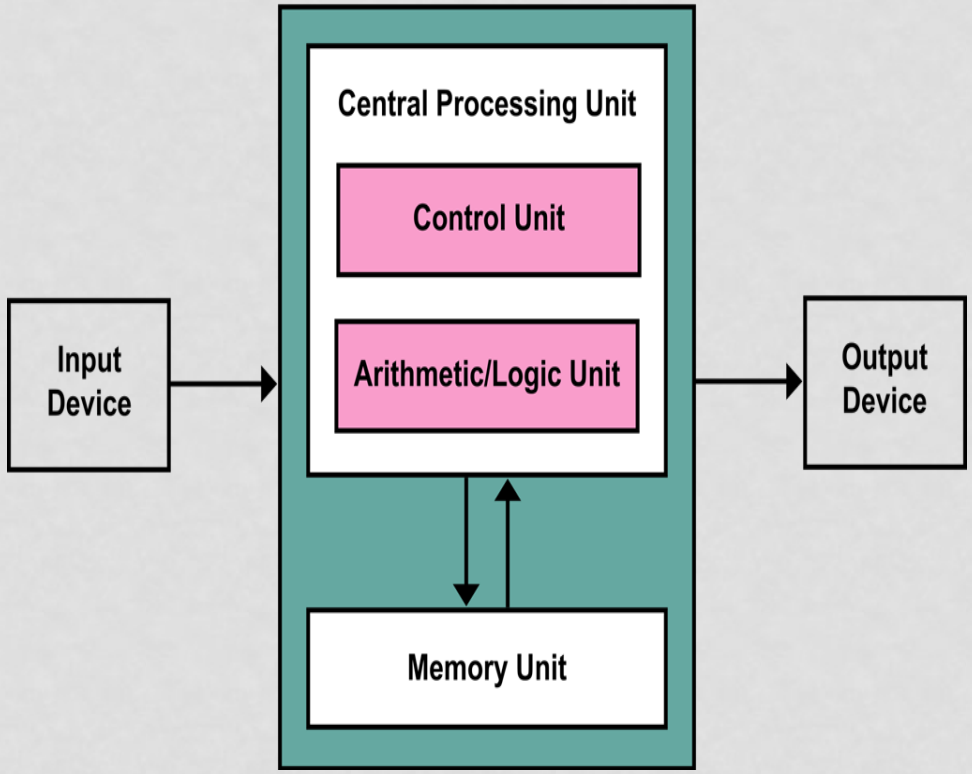
BASIC ORGANIZATION OF COMPUTER SYSTEM

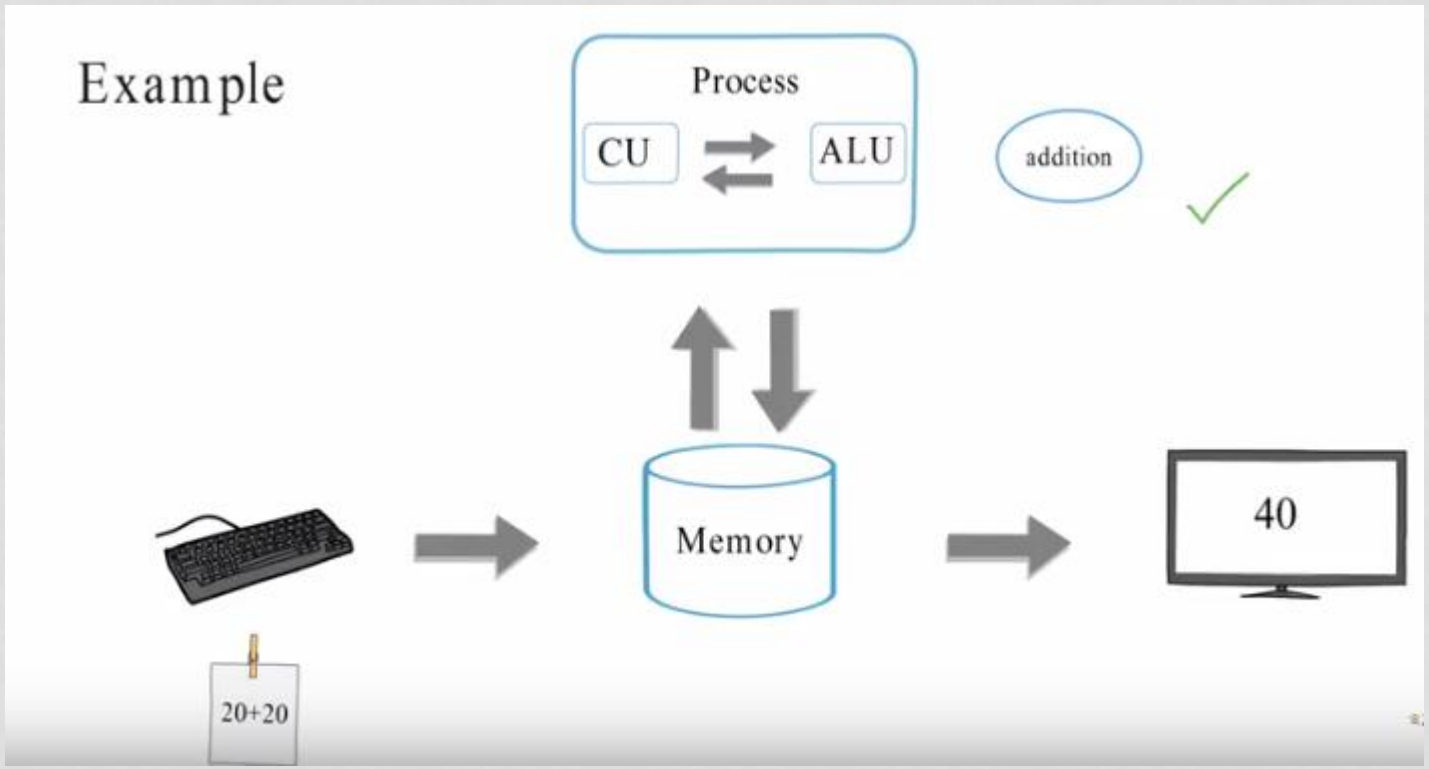
BASIC ORGANIZATION OF COMPUTER SYSTEM



VON NEUMANN BASIC STRUCTURE

- Let's consider them in detail.
- **Arithmetic and Logic Unit (ALU)**
  - The arithmetic logic unit is that part of the CPU that handles all the calculations the CPU may need, e.g. Addition, Subtraction, Comparisons.
  - It performs Logical Operations, Bit Shifting Operations, and Arithmetic Operation.
- **Control Unit**
  - A control unit (CU) handles all processor control signals. It directs all input and output flow, fetches code for instructions and controlling how data moves around the system.




Example
Process
CU
ALU
addition
Memory
40
20+20

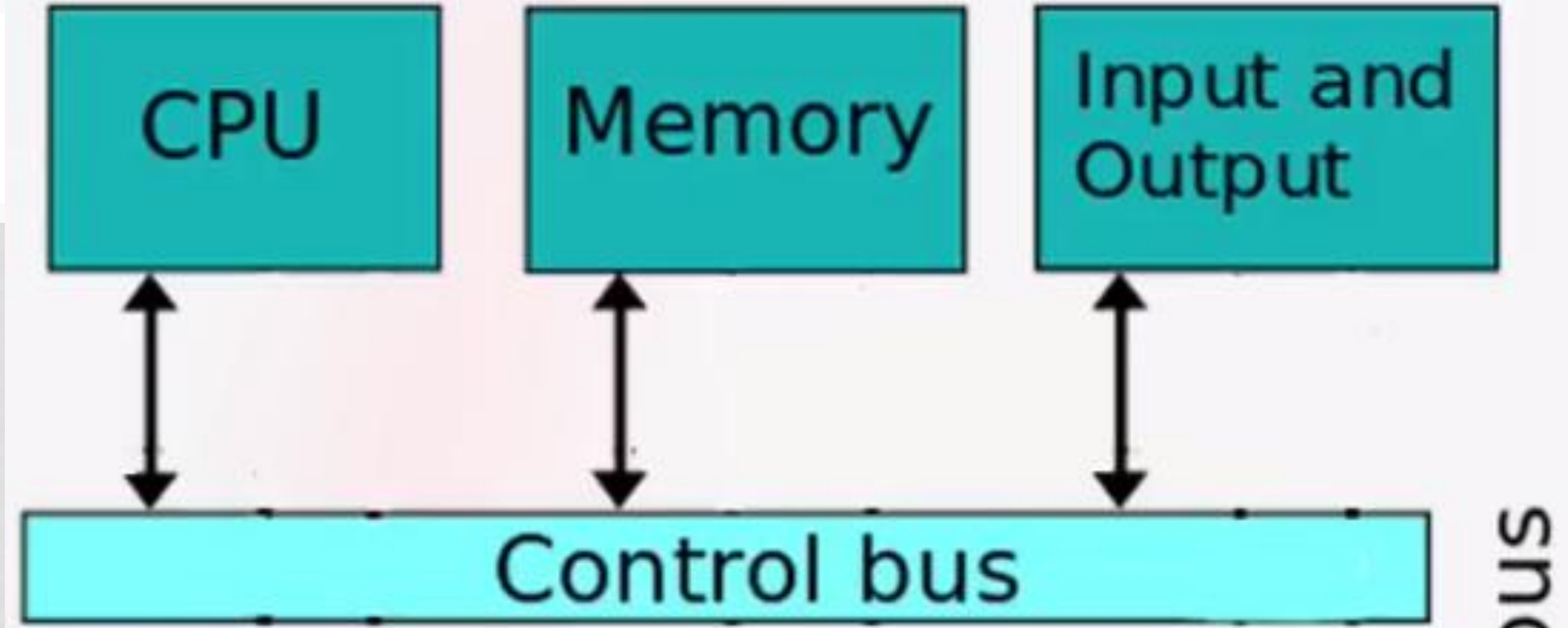
COMPUTER BUS

Bus:

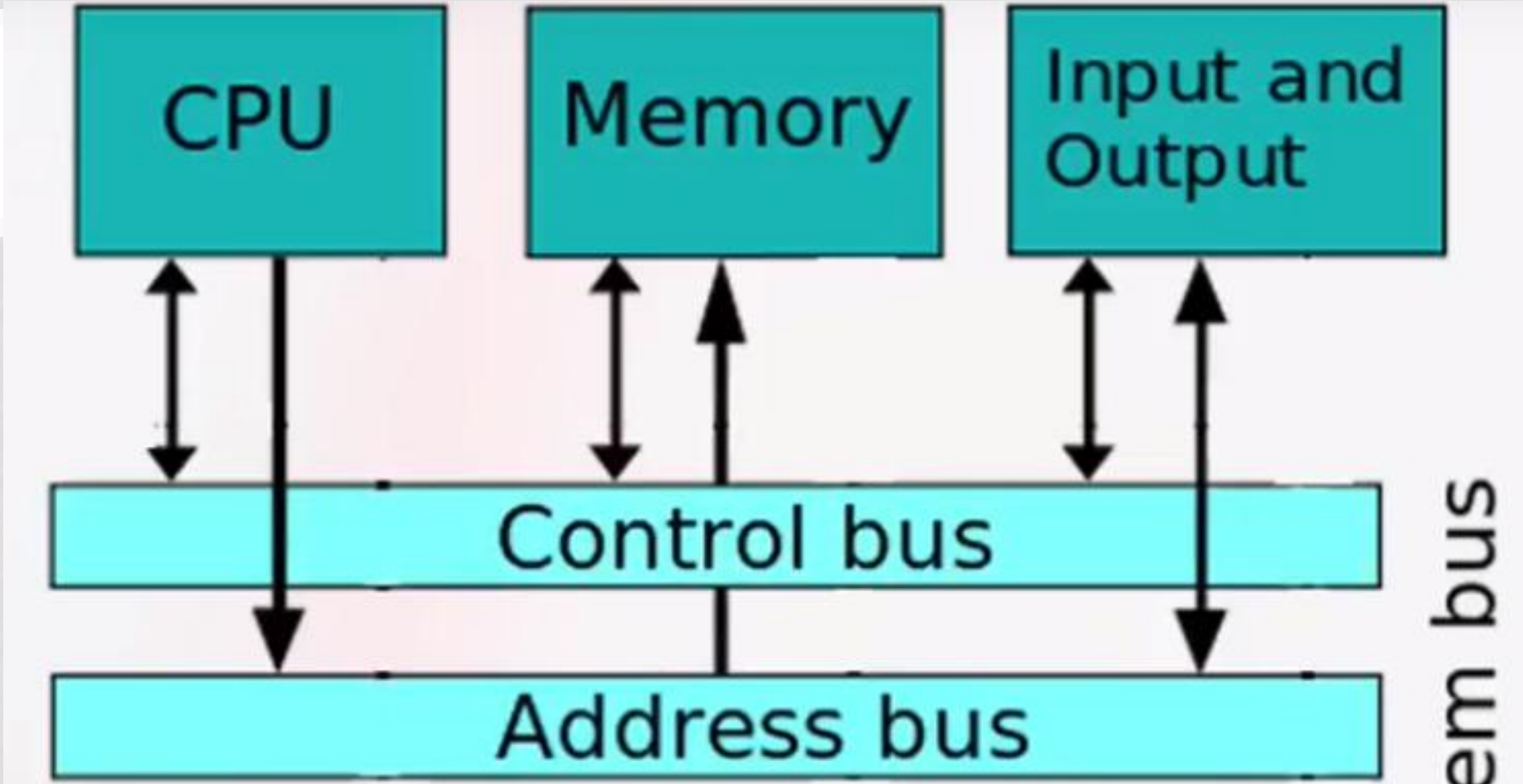
- Inside computers, there are many internal components. For these components to communicate with each other they make use of wires that are known as a 'bus' .
- A bus is a common pathway through which information flows from one computer component to another. This pathway is used for communication purpose and it is established between two or more computer components.

- **Types of Bus:**

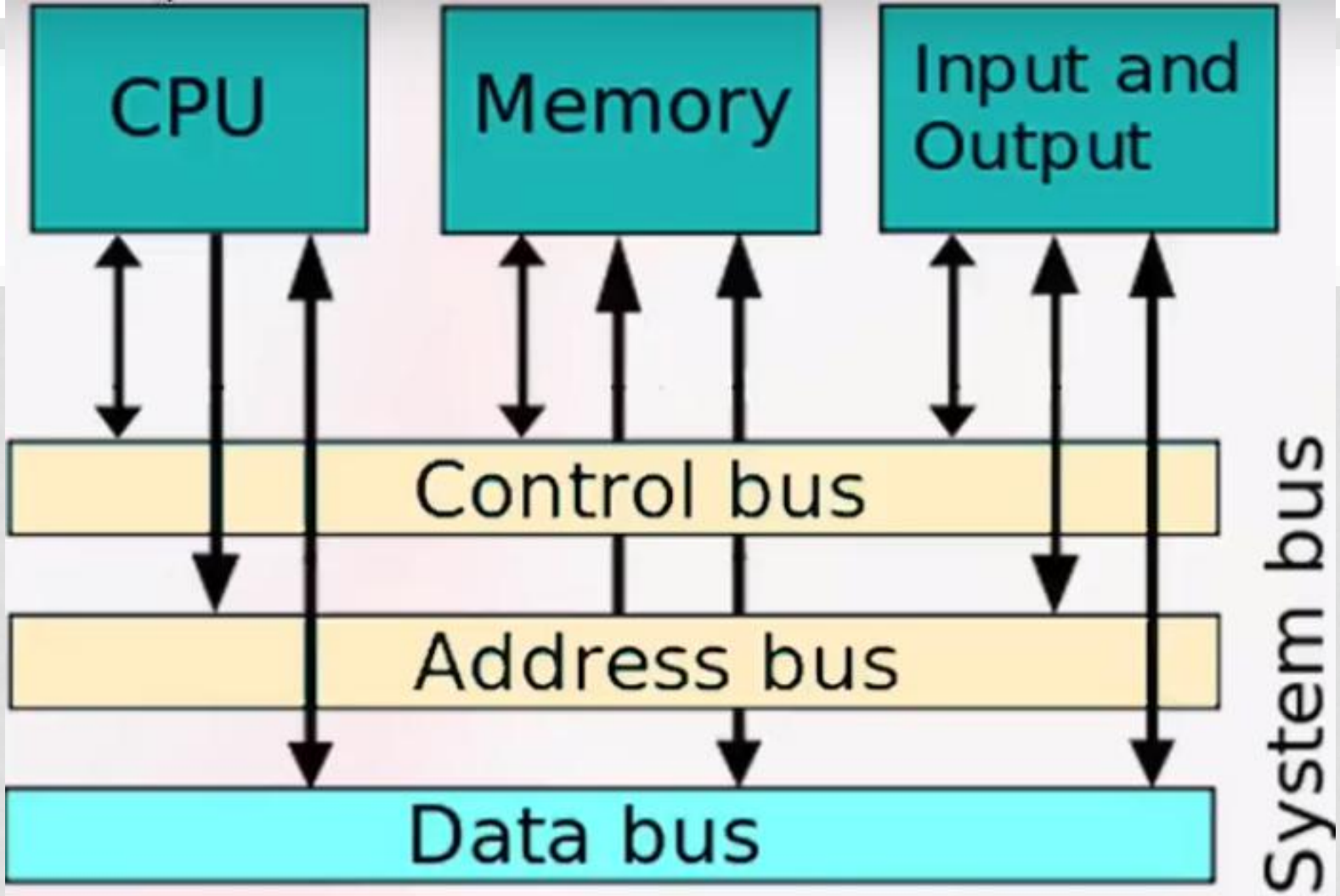
- **Data bus** is used for transfer of data between subunits
- **Address bus** is used to transmit location information between units such as where the data is going or coming from.
- **Control bus** control working of other components. It is used to provide information as to how data is being sent.



The **control bus** carries signals that control the actions of the computer eg reading and writing data to and from memory



Address Bus: Carries addresses from the Processor to main memory or other I/O devices
It is one direction (Uni-Directional)
The processor generates an address
All data/Instructions are returned on the Data Bus



Data Bus: Carries Data/Instructions from Main Memory to the Processor
(or from other secondary storage devices) to the processor.
Bi-Directional (two way)
Data can be read/written

REGISTERS

- Special **memory units**, called registers, are used to hold information on a temporary basis as the instructions are interpreted and executed by the CPU
- Registers are part of the CPU (not main memory) of a computer and is super fast memory.
- The length of a register, sometimes called its *word size*, equals the number of bits it can store
- With all other parameters being the same, a CPU with 32-bit registers can process data twice larger than one with 16-bit registers

CPU REGISTERS

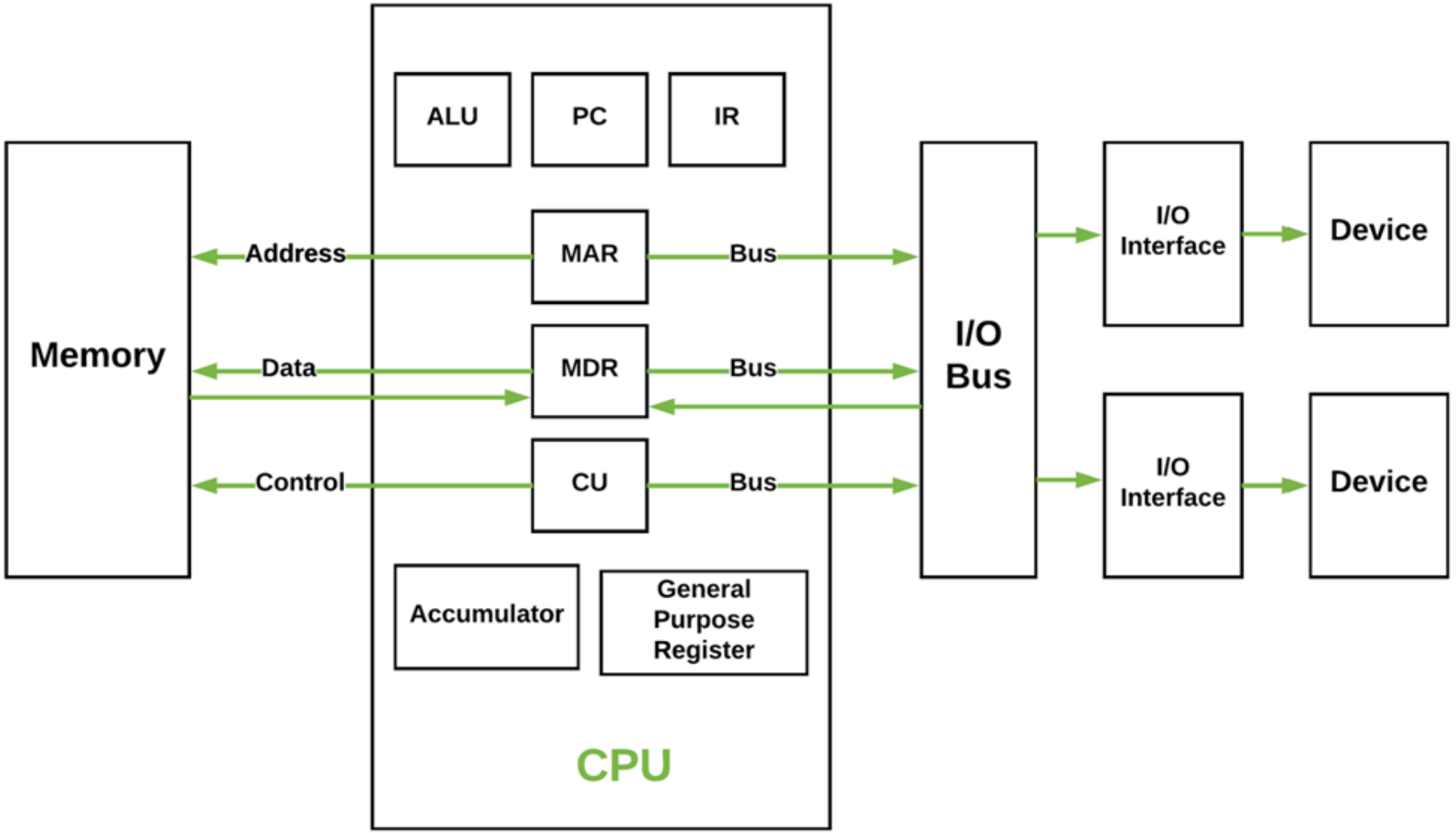
- **Accumulator:** Stores the results of calculations made by ALU.
- **Program Counter (PC):** Keeps track of the memory location of the next instructions to be dealt with. The PC then passes this next address to Memory Address Register (MAR).
- **Memory Address Register (MAR):** Holds the address of the location in memory, which contains data, that is required by the current instruction being executed. simply MAR points to the memory location that contains data required.
- **Memory Data Register (MDR):** It stores instructions fetched from memory or any data that is to be transferred to, and stored in, memory.
- **Current Instruction Register (CIR):** Holds the **instruction** currently being executed or decoded.
- **Instruction Buffer Register (IBR):** The instruction that is not to be executed immediately is placed in the instruction buffer register IBR.

CPU REGISTERS

- General Purpose Registers:

Some registers can be used for more than one purpose. Such registers are called general purpose registers.

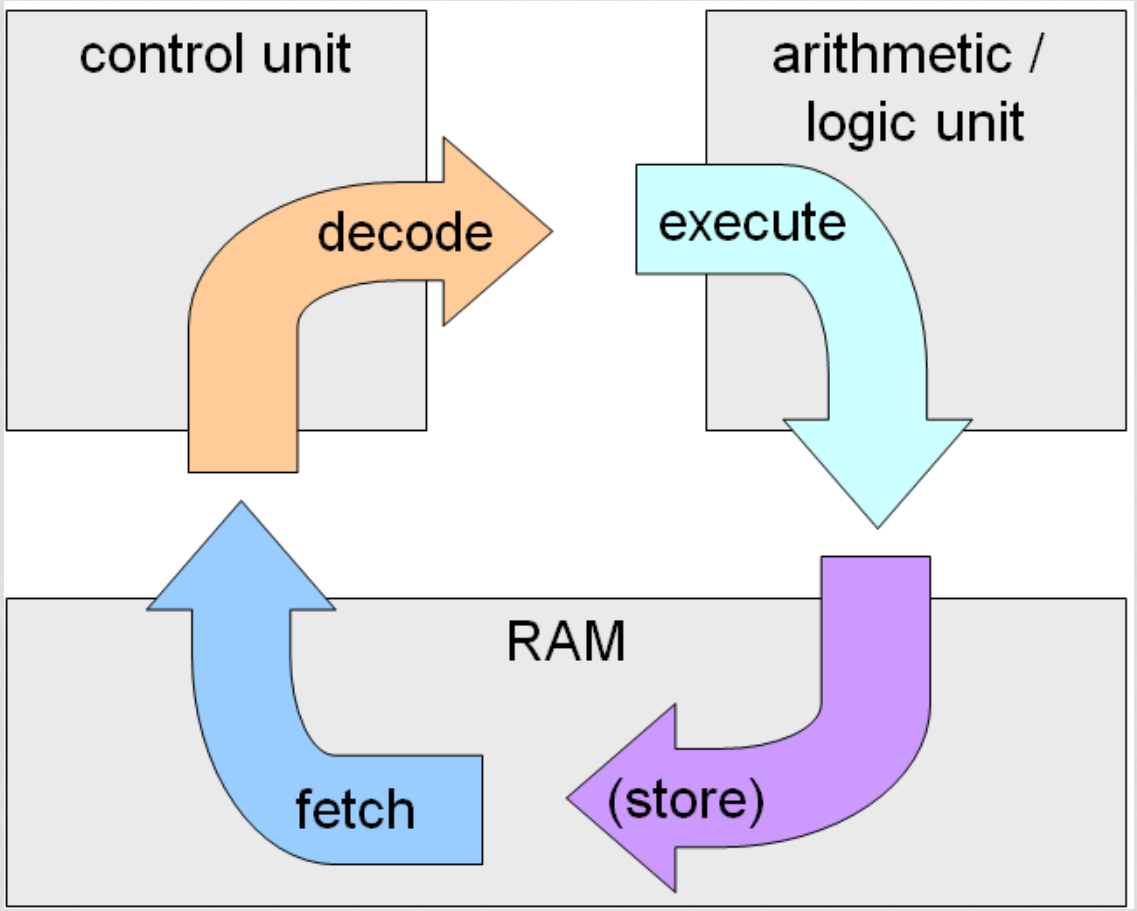
CPU REGISTERS



FETCH DECODE EXECUTE CYCLE

- **Fetch :**
  - Get an instruction from main memory
- **Decode:**
  - Translate it into computer command
- **Execute:**
  - Actual processing of command
- **Mem:**
  - Write result to memory.

FETCH DECODE EXECUTE CYCLE



A FIVE-STEP CYCLE

- **These operations are repeated in a never-ending sequence**
- **A five-step cycle:**
  1. Instruction Fetch (IF)
  2. Instruction Decode (ID)
  3. Data Fetch (DF) / Operand Fetch (OF)
  4. Instruction Execution (EX)
  5. Result Return (RR) / Store (ST)

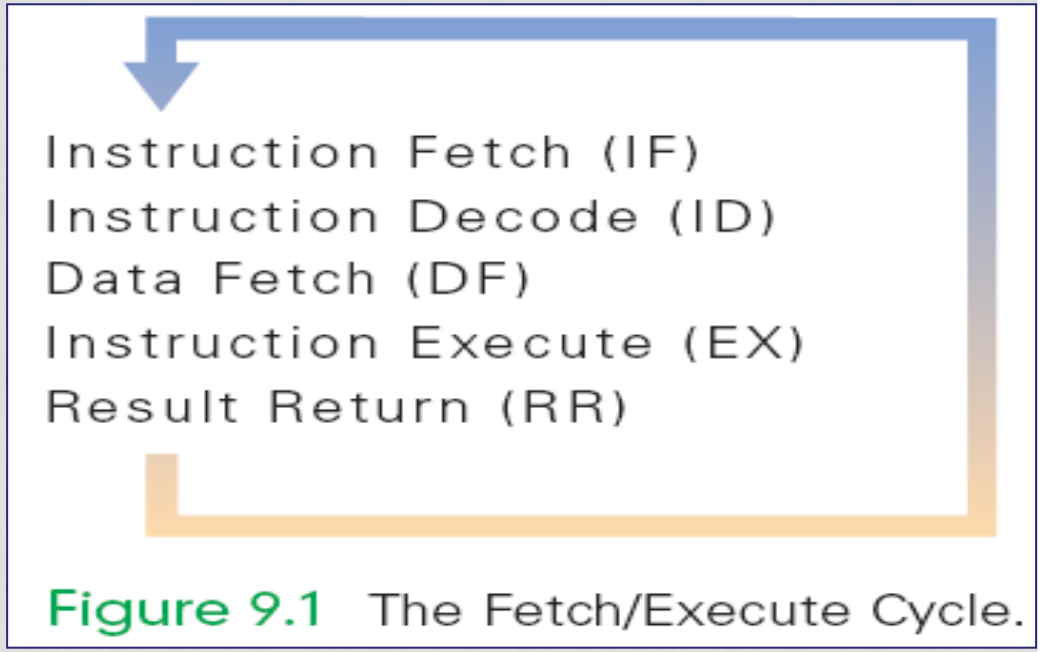
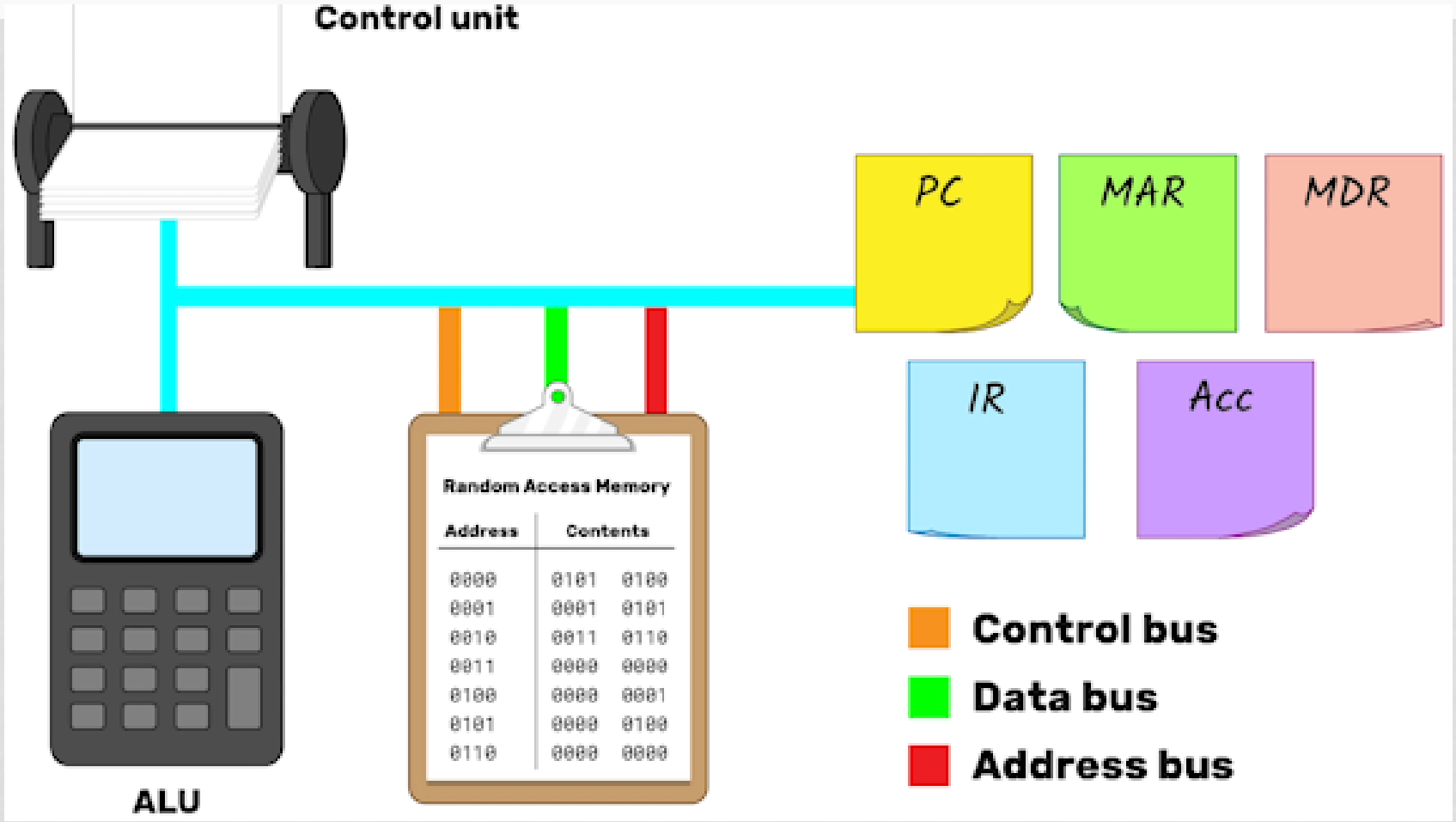


Figure 9.1 The Fetch/Execute Cycle.

A FIVE-STEP CYCLE



ADD 800, 428, 884

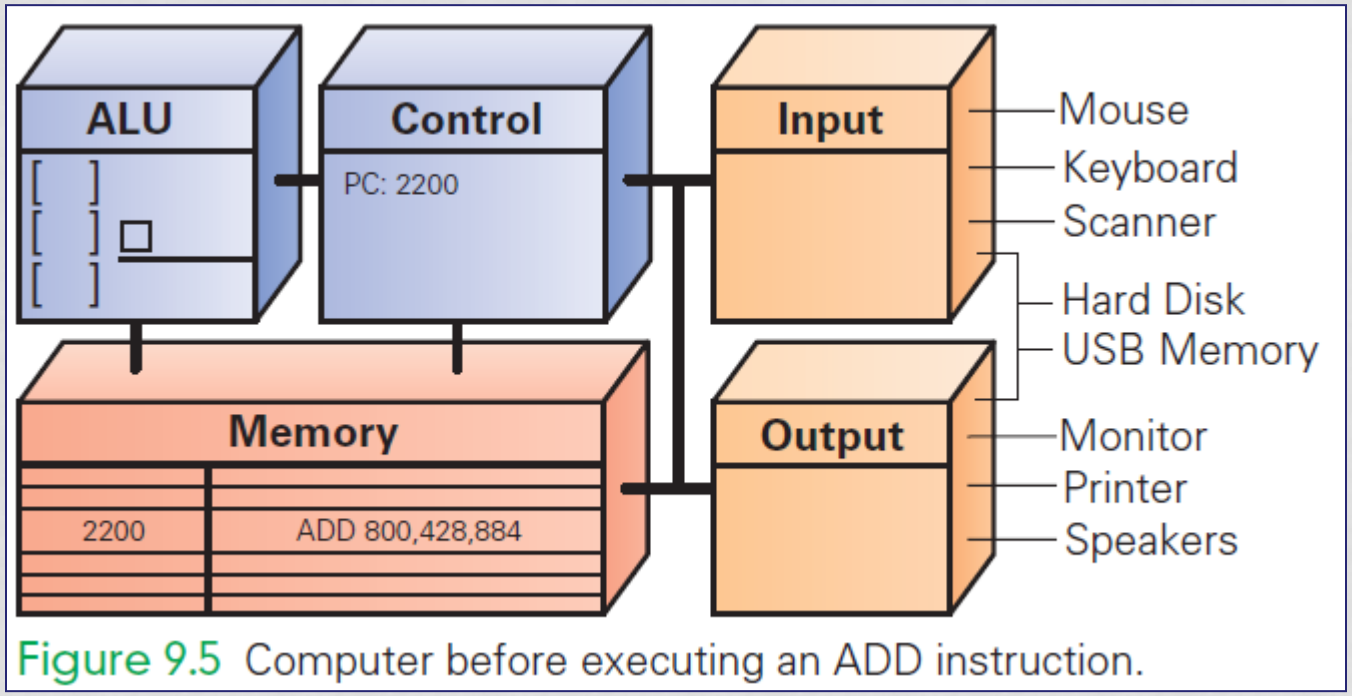


Figure 9.5 Computer before executing an ADD instruction.

INSTRUCTION FETCH (IF)

- Execution begins by moving the instruction at the address given by the (*PC 2200*) from memory to the control unit
- Once instruction is fetched, the PC can be changed for fetching the next instruction

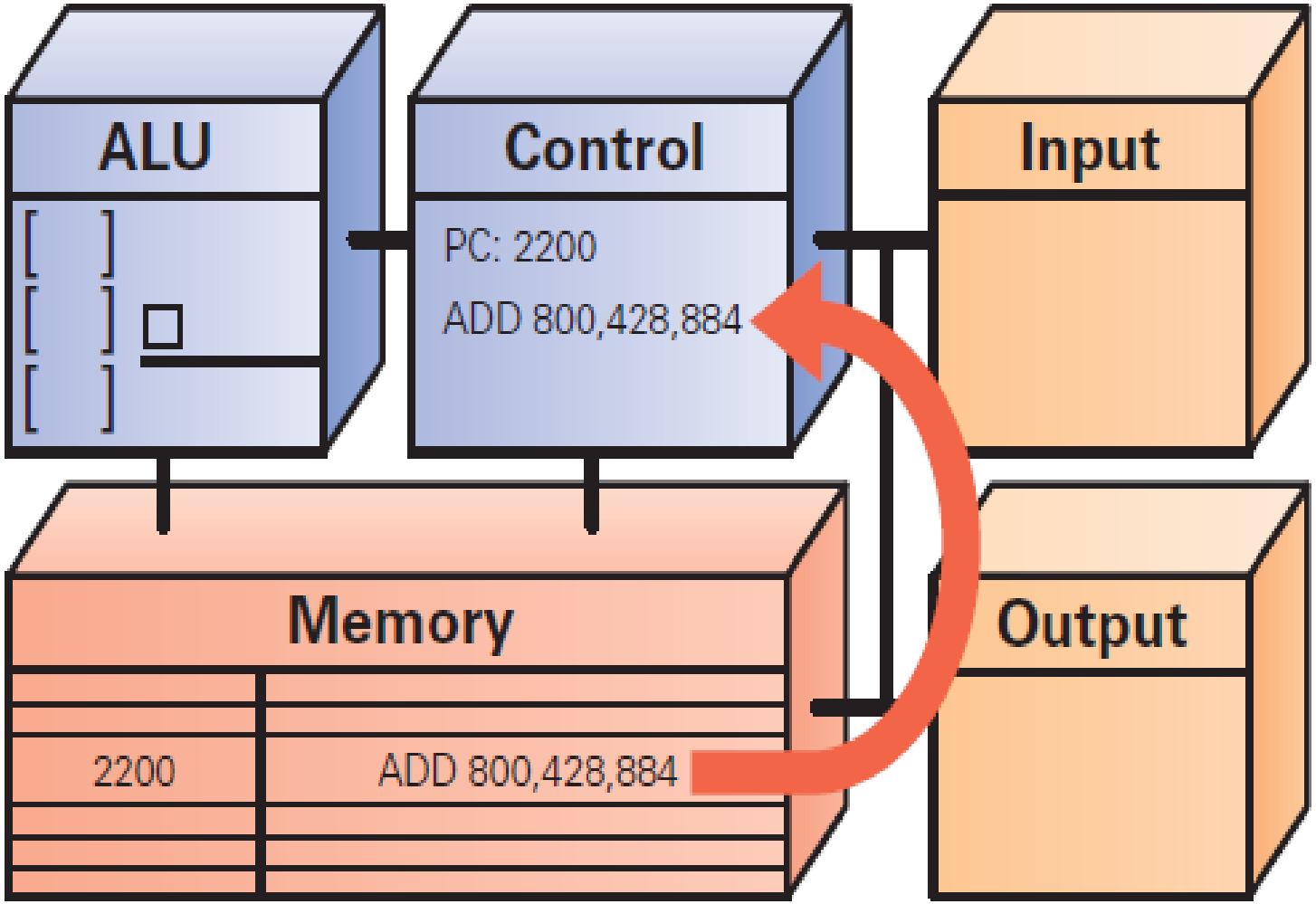


Figure 9.6 Instruction Fetch: Move instruction from memory to the control unit.

INSTRUCTION DECODE (ID)

- **ALU is set up for the operation**
- **Decoder finds the memory address of the instruction's data (*source operands*)**
  - Most instructions operate on two data values stored in memory (like ADD), so most instructions have addresses for two source operands
  - These addresses are passed to the circuit that fetches them from memory during the next step
  - Decoder finds the *destination address* for the Result Return step and places the address in the RR circuit
  - Decoder determines what operation the ALU will perform (ADD), and sets up the ALU

ID

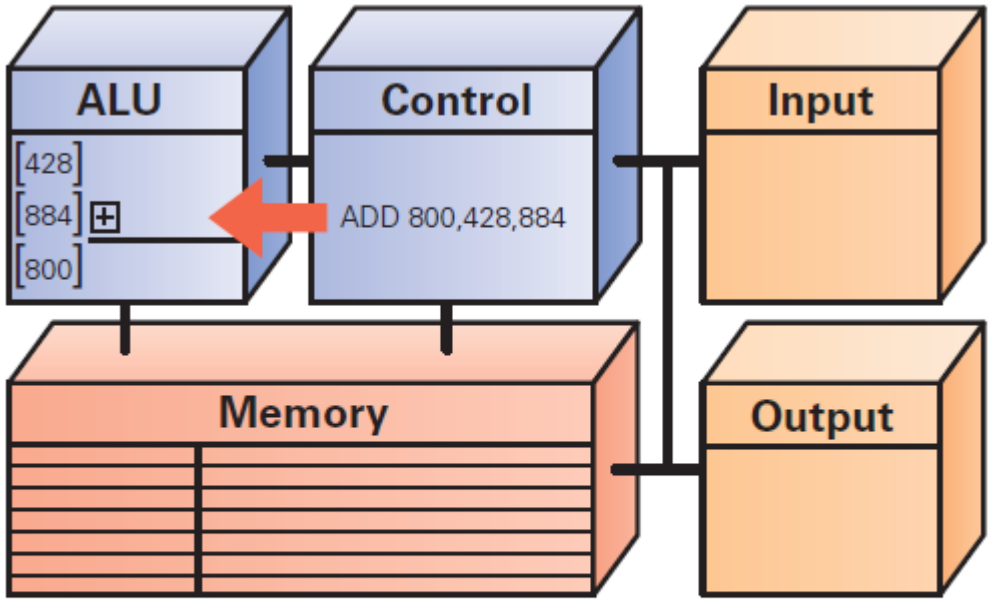


Figure 9.7 Instruction Decode: Pull apart the instruction, set up the operation in the ALU, and compute the source and destination operand addresses.

DATA FETCH (DF)

- The data values to be operated on are retrieved from memory
- Bits at specified memory locations are copied into locations in the ALU circuitry
- Data values remain in memory (they are not destroyed)

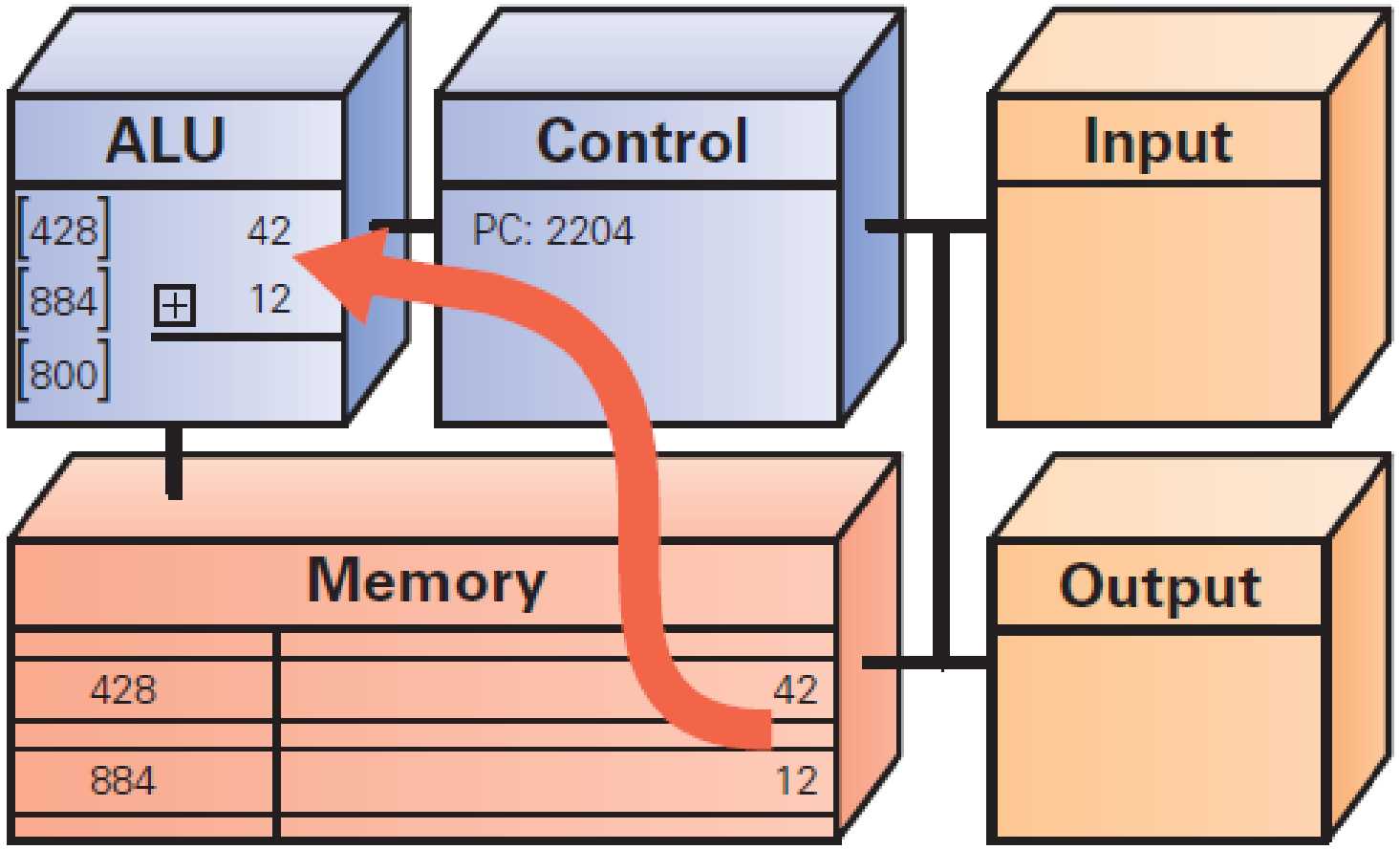


Figure 9.8 Data Fetch: Move the operands from memory to the ALU.

INSTRUCTION EXECUTION (EX)

- For this ADD instruction, the addition circuit adds the two source operands together to produce their sum
- Sum is held in the ALU circuitry
- This is the actual computation

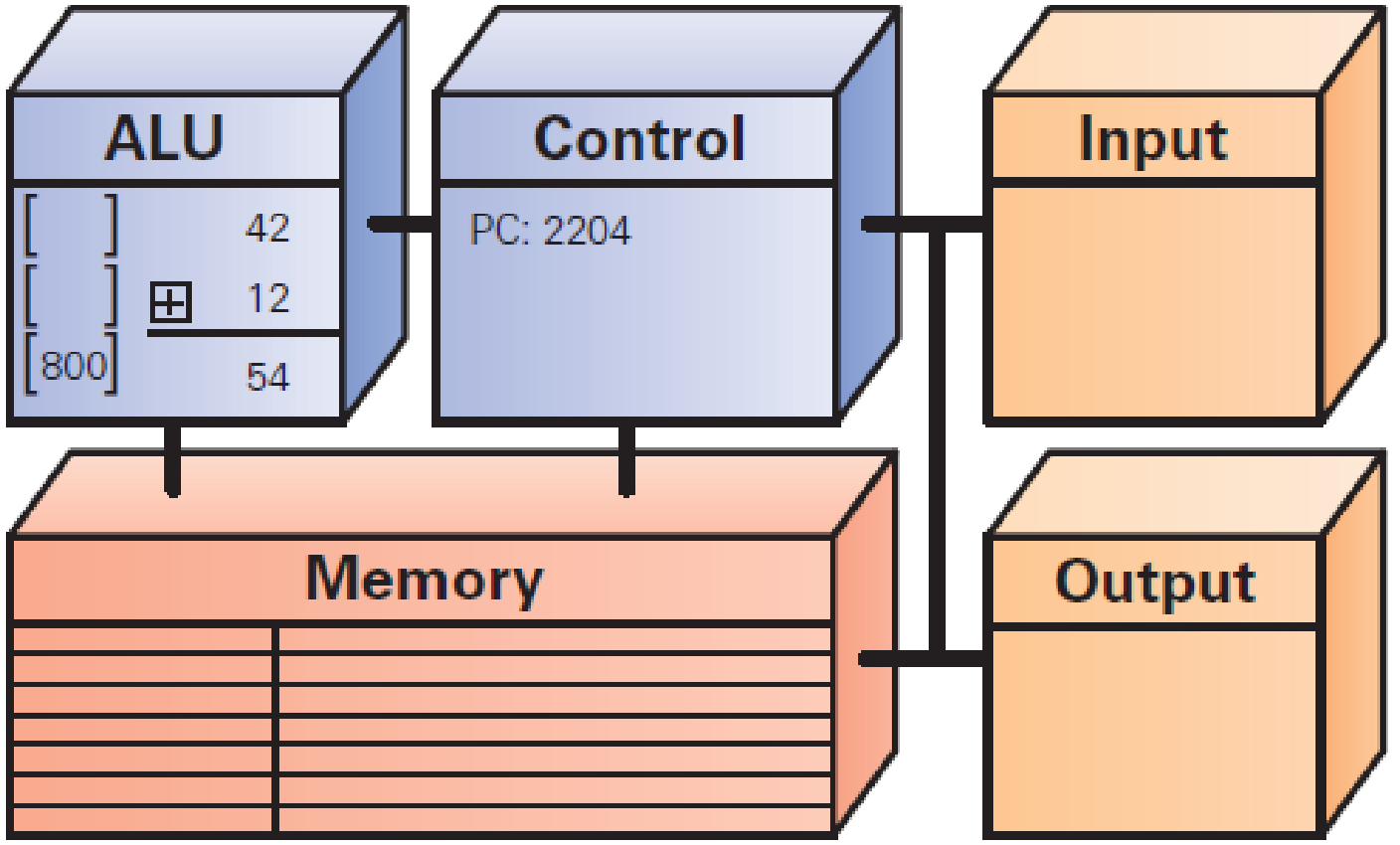


Figure 9.9 Instruction Execute: Compute the result of the operation in the ALU.

RETURN RESULT (RR)

- RR returns the result of EX to the memory location specified by the destination address.
- Once the result is stored, the cycle begins again

RR

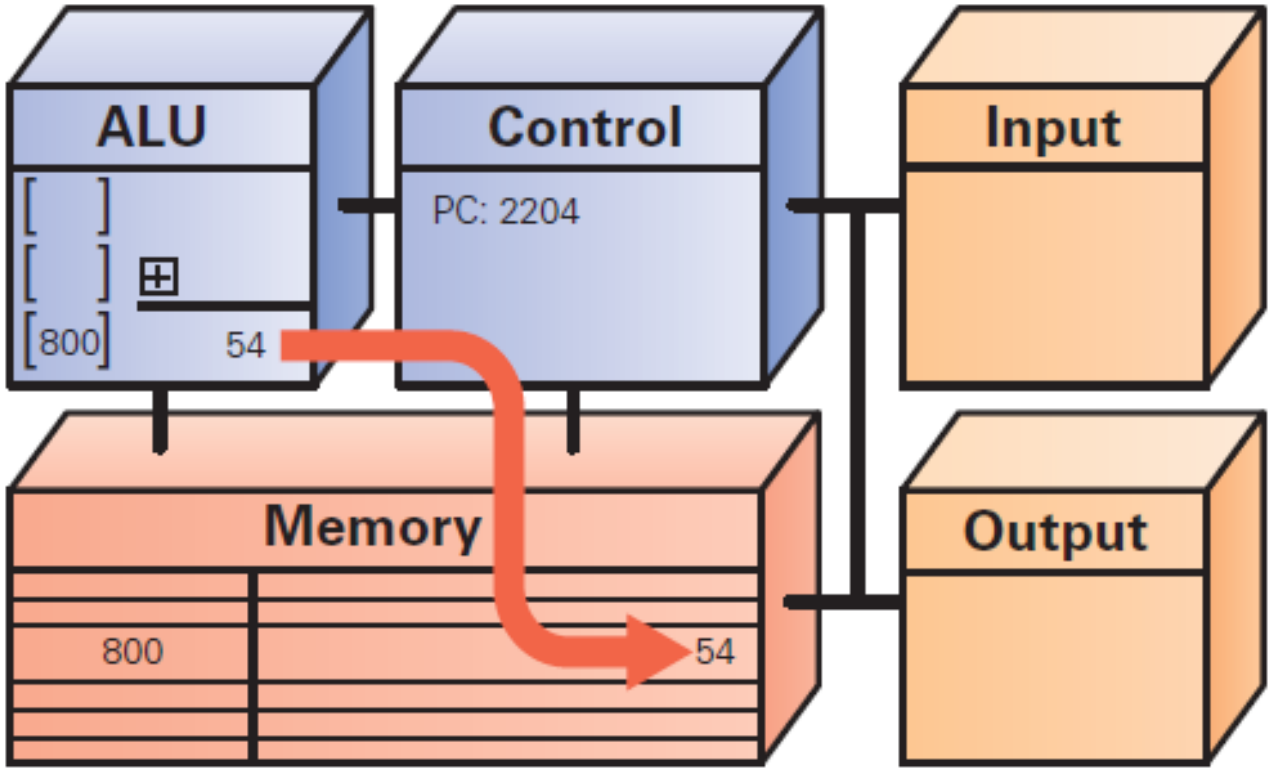


Figure 9.10 Result Return: Store the result from the ALU into the memory at the destination address.

CYCLING THE FETCH/EXECUTE CYCLE

- ADD is representative of the complexity of computer instructions…some are slightly simpler, some slightly more complex
- Computers achieve success at what they can do with speed.
- They show their impressive capabilities by executing many simple instructions per second

PROCESSOR AND MEMORY

MEMORY

- **Memory stores both the program while it is running and the data on which the program operates**
- **Properties of memory:**
  - **Discrete locations**
    - Memory is organized as a sequence of discrete locations
    - In modern memory, each location is composed of 1 byte (8 bits)

MEMORY

- **Addresses**
  - Every memory location has an address, whole numbers starting at 0
- **Values**
  - Memory locations record or store values
- **Finite capacity**
  - Memory locations have a finite capacity (limited size),
  - Data may not “fit” in the memory location

BYTE-SIZE MEMORY LOCATION

- Common visualization of computer memory
- Discrete locations are shown as boxes holding 1-byte each

Figure 9.3 Diagram of computer memory illustrating its key properties.

- Address of location is displayed above the box and the contents of location is shown in the box

BYTE-SIZE MEMORY LOCATION

- That **1-byte memory** location can store one ASCII character or a number less than 256
- Blocks of **four bytes are** used as a unit so often that they are called memory words.

RAM

- **Computer memory is called random access memory (RAM)**
  - "Random access" is out-of-date and simply means that the computer can refer to the memory locations in any order
- **RAM is measured in megabytes (MB) or gigabytes (GB)**
- **Lots of memory is need to handle the space required of programs and data**

MEMORY CAPACITY

- Memory capacity of a computer is equal to the number of bytes that can be stored in its primary storage
- Its units are:
  - Kilobytes (KB) : 1024 (2^{10}) bytes
  - Megabytes (MB) : 1,048,576 (2^{20}) bytes
  - Gigabytes (GB) : 1,073,741824 (2^{30}) bytes

CONTROL UNIT-CU

- The **control unit** of a computer is where the Fetch/Execute Cycle occurs
- Its circuitry *fetches* an instruction from memory and performs the other operations of the Fetch/Execute Cycle on it
- A typical machine instruction has the form
  ADD 4000, 2000, 2080

TASK

- **ADD 4000, 2000, 2080**
  - Looks like those three numbers should be added together
  - What it really means is that whatever numbers are stored in memory locations 2000 and 2080 be added together, and the result be stored in location 4000

TASK

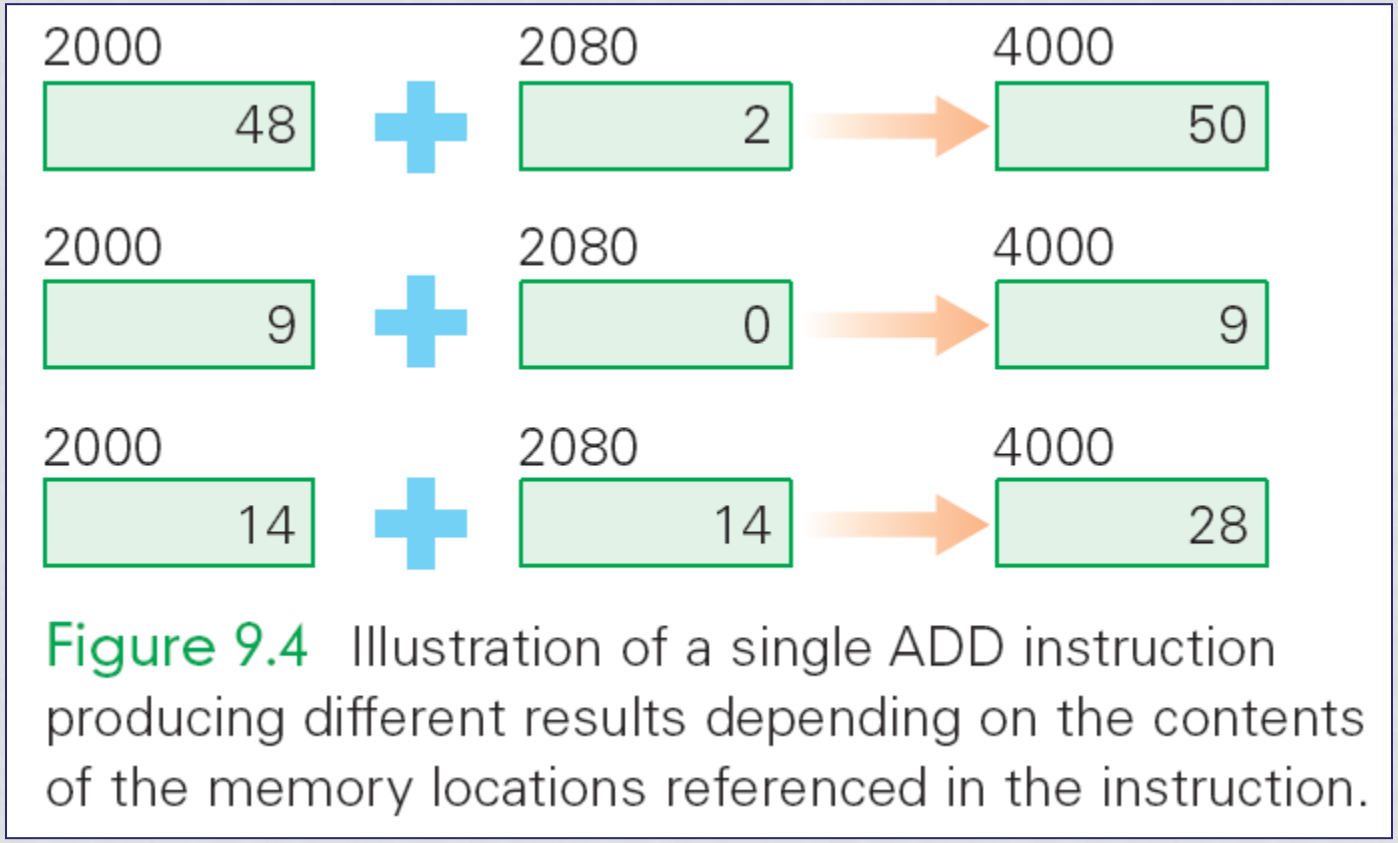


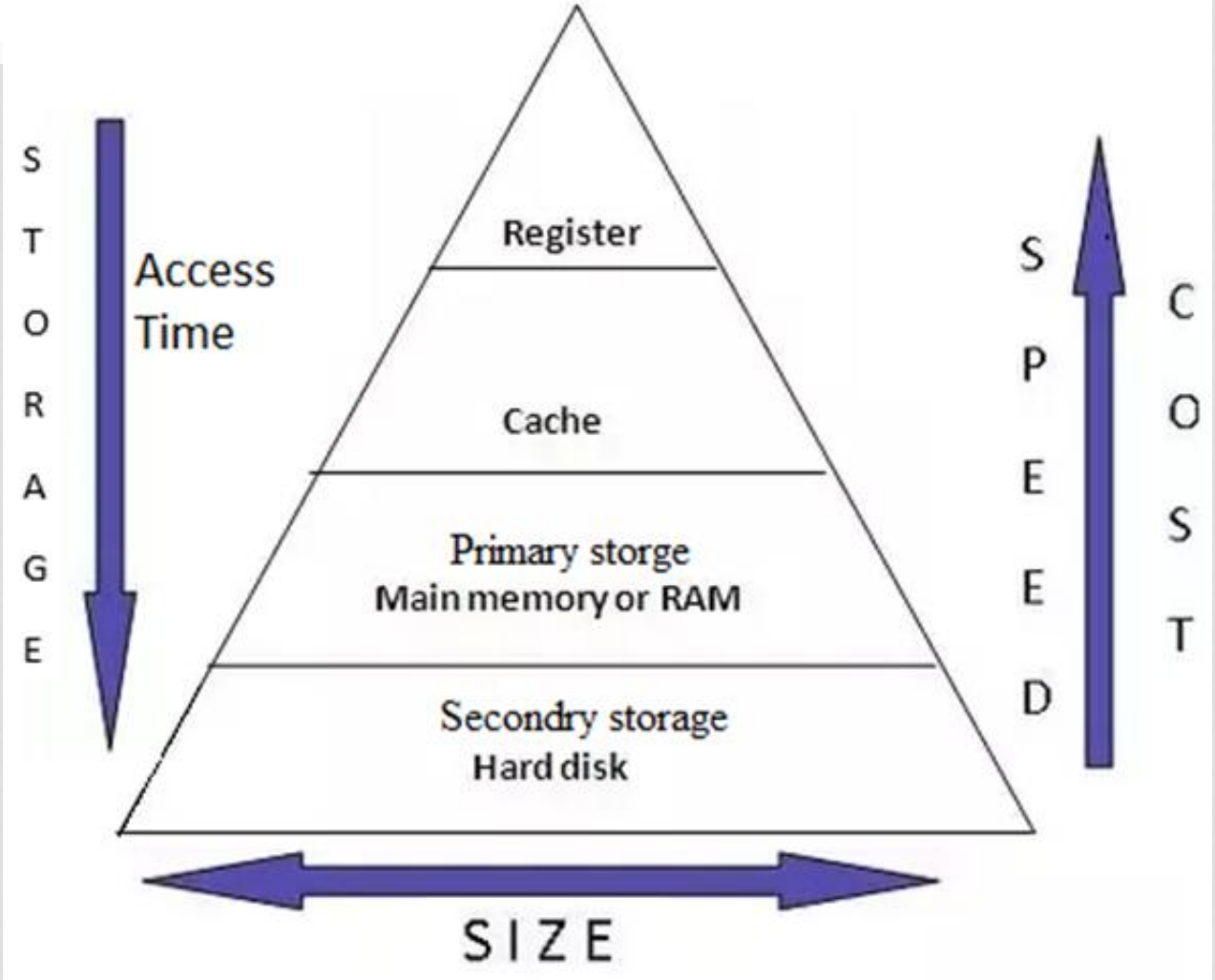
Figure 9.4 Illustration of a single ADD instruction producing different results depending on the contents of the memory locations referenced in the instruction.

CACHES

- The data/instructions stored in RAM are retrieved faster than hard disks. That's why the data/instructions required by CPU is kept in RAM.
- There is another type of memory which is much faster than RAM. It's called cache. Some part of data/instructions currently required by CPU is copied into cache from RAM.
- The CPU then gets instructions and data from caches.
- Caches are much smaller in capacity than RAMs.
- They are built-in on the CPU

STORAGE HIERARCHY

- Capacity and access are inversely proportional to each other



THE PROGRAM COUNTER (PC)

- **How does the computer determine which instruction it should execute next?**
- **Address of the Next Instruction**
  - The instruction is stored in memory and the computer has its address
  - Computers use the address (known as the *program counter* or *PC)* to keep track of the next instruction

THE PROGRAM COUNTER

- The computer gets ready to process the next instruction
- It assumes that the next instruction is the next instruction in sequence
- Because instructions use 4 bytes of memory, the next instruction must be at the memory address PC + 4 or 4 bytes further along the sequence

THE COMPUTER CLOCK

- Computers are instruction execution engines.
- Since the computer does one instruction per cycle in principle, the speed of a computer depends on the number of Fetch/Execute Cycles it completes per second.

THE COMPUTER CLOCK

- The rate of the Fetch/Execute Cycle is determined by the computer's clock, and it is measured in megahertz, or millions (mega) of cycles per second (hertz).
- A 1,000 MHz clock ticks a billion (in American English) times per second, which is one gigahertz (1 GHz)

STANDARD PREFIXES

1000^1	kilo-	$1024^1 = 2^{10} = 1{,}024$	milli-	1000^{-1}
1000^2	mega-	$1024^2 = 2^{20} = 1{,}048{,}576$	micro-	1000^{-2}
1000^3	giga-	$1024^3 = 2^{30} = 1{,}073{,}741{,}824$	nano-	1000^{-3}
1000^4	tera-	$1024^4 = 2^{40} = 1{,}099{,}511{,}627{,}776$	pico-	1000^{-4}
1000^5	peta-	$1024^5 = 2^{50} = 1{,}125{,}899{,}906{,}842{,}624$	femto-	1000^{-5}
1000^6	exa-	$1024^6 = 2^{60} = 1{,}152{,}921{,}504{,}606{,}876{,}976$	atto-	1000^{-6}
1000^7	zetta-	$1024^7 = 2^{70} = 1{,}180{,}591{,}620{,}717{,}411{,}303{,}424$	zepto-	1000^{-7}
1000^8	yotta-	$1024^8 = 2^{80} = 1{,}208{,}925{,}819{,}614{,}629{,}174{,}706{,}176$	yocto-	1000^{-8}

Figure 9.11 Standard prefixes from the Système International (SI) convention on scientific measurements. Generally a prefix refers to a power of 1000, except when the quantity (for example, memory) is counted in binary; for binary quantities the prefix refers to a power of 1024, which is 2^{10}.

COMPUTER'S VIEW OF SOFTWARE

- **A program "sees" software as a long sequence of 4-byte groups of bits (0's and 1's)**

```
. . . 10001111 10010100 00000011 01110100
      10001111 10011000 00000001 10101100
      00000010 10011000 10100000 00100000  ↔   ADD 20, 20, 24
      10101111 10010100 00000001 10010000 . . .
```

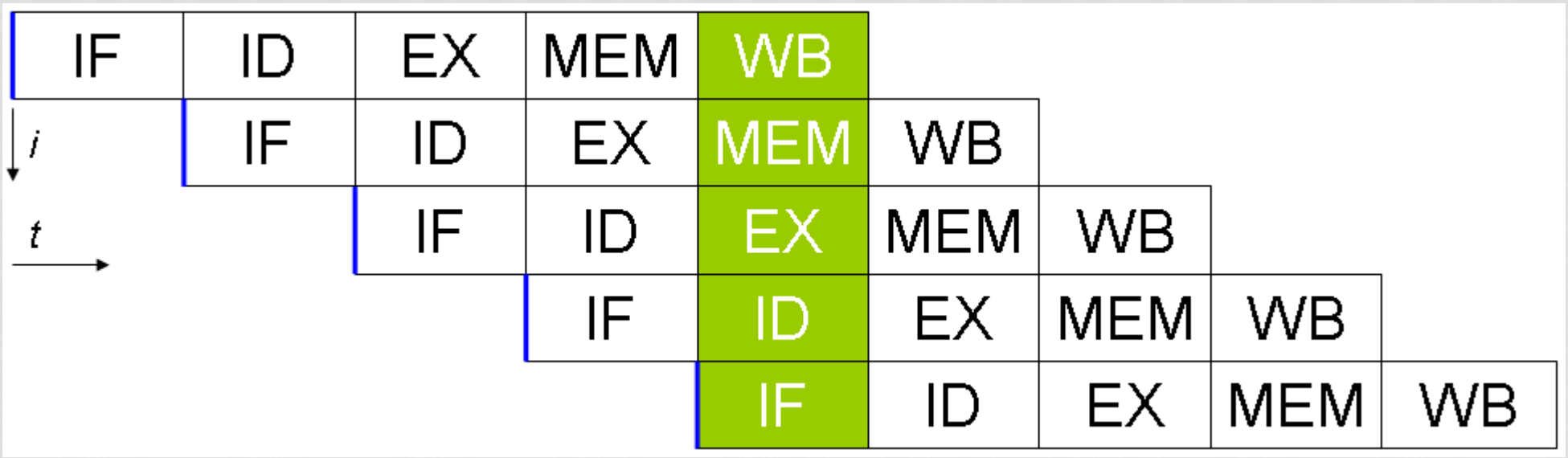
- This binary object file can be hundreds of thousands to millions of words long

COMPUTER'S VIEW OF SOFTWARE

- **Once installed, the computer runs the software by:**
  - copying the binary instructions into the RAM
  - interpreting them using the Fetch/Execute Cycle.
- **It does whatever the instructions tell it to do**

PIPELINING

Ignore the WB stage.



RECOMMENDED

- https://www.futurelearn.com/courses/how-computers-work/0/steps/49284