B38DF
Computer Architecture and Embedded Systems

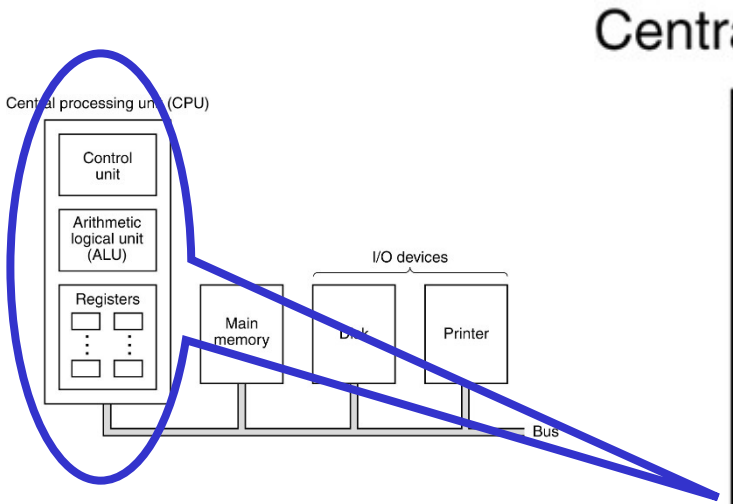
Alexander Belyaev

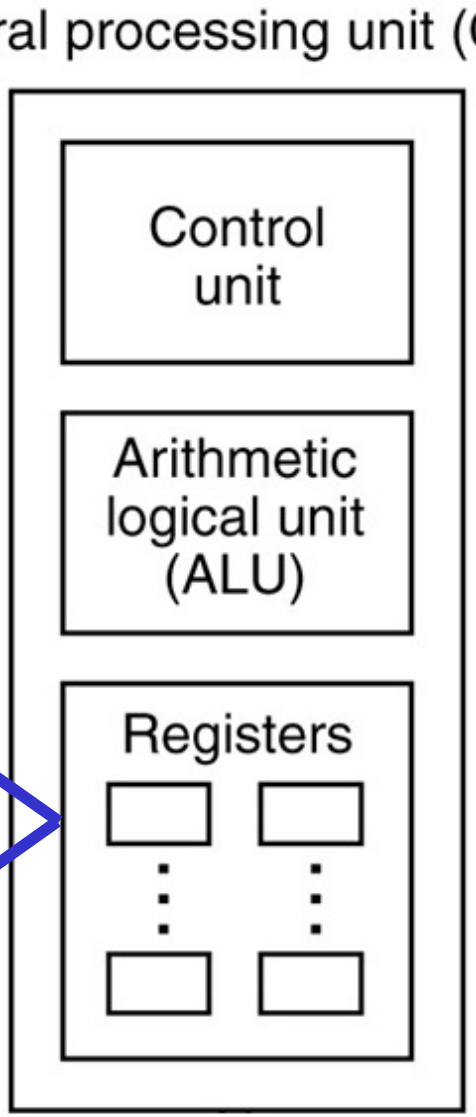
Heriot-Watt University
School of Engineering & Physical Sciences
Electrical, Electronic and Computer Engineering

E-mail: a.belyaev@hw.ac.uk
Office: EM2.29

Based on the slides prepared by Dr. Mustafa Suphi Erden

Processor (CPU) in a Computer





Fetching instructions from the main memory and determining their type

Performs operations such as addition and Boolean AND needed to carry out the instructions

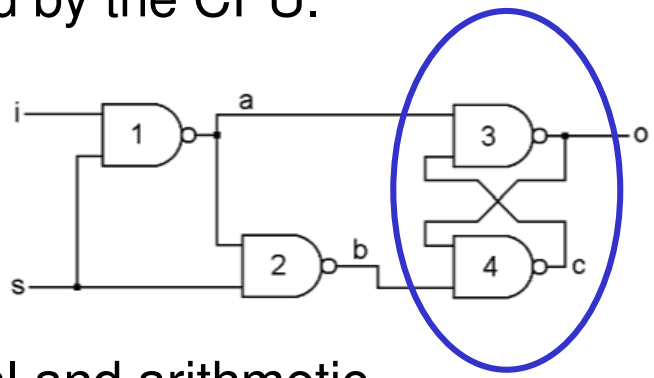
High-speed memory used to store temporary results and certain control information

Program Counter (PC): points to the next instruction to be fetched for execution

Instruction Register (IR): holds the instruction currently being executed

Registers, ALU, Control Unit

- Registers hold data that can be readily accessed by the CPU.
- They can be implemented using D flip-flops.
  - A 32-bit register requires 32 D flip-flops.

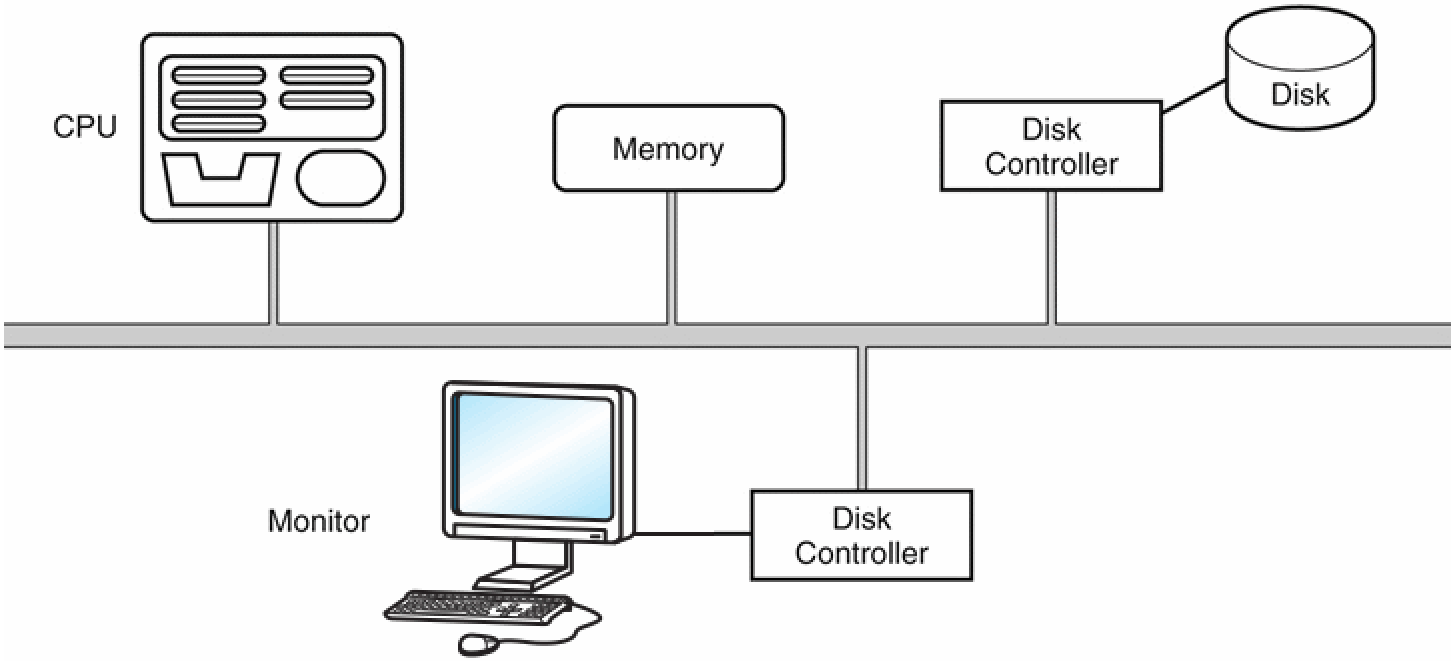


- The arithmetic-logic unit (ALU) carries out logical and arithmetic operations as directed by the control unit.

- The control unit determines which actions to carry out according to the values in a program counter register and a status register.

Buses

- The CPU shares data with other system components by way of a data bus.
  - A bus is a set of wires that simultaneously convey a single bit along each line.
- Two types of buses are commonly found in computer systems: *point-to-point* (e.g., inside CPU), and *multipoint* buses.

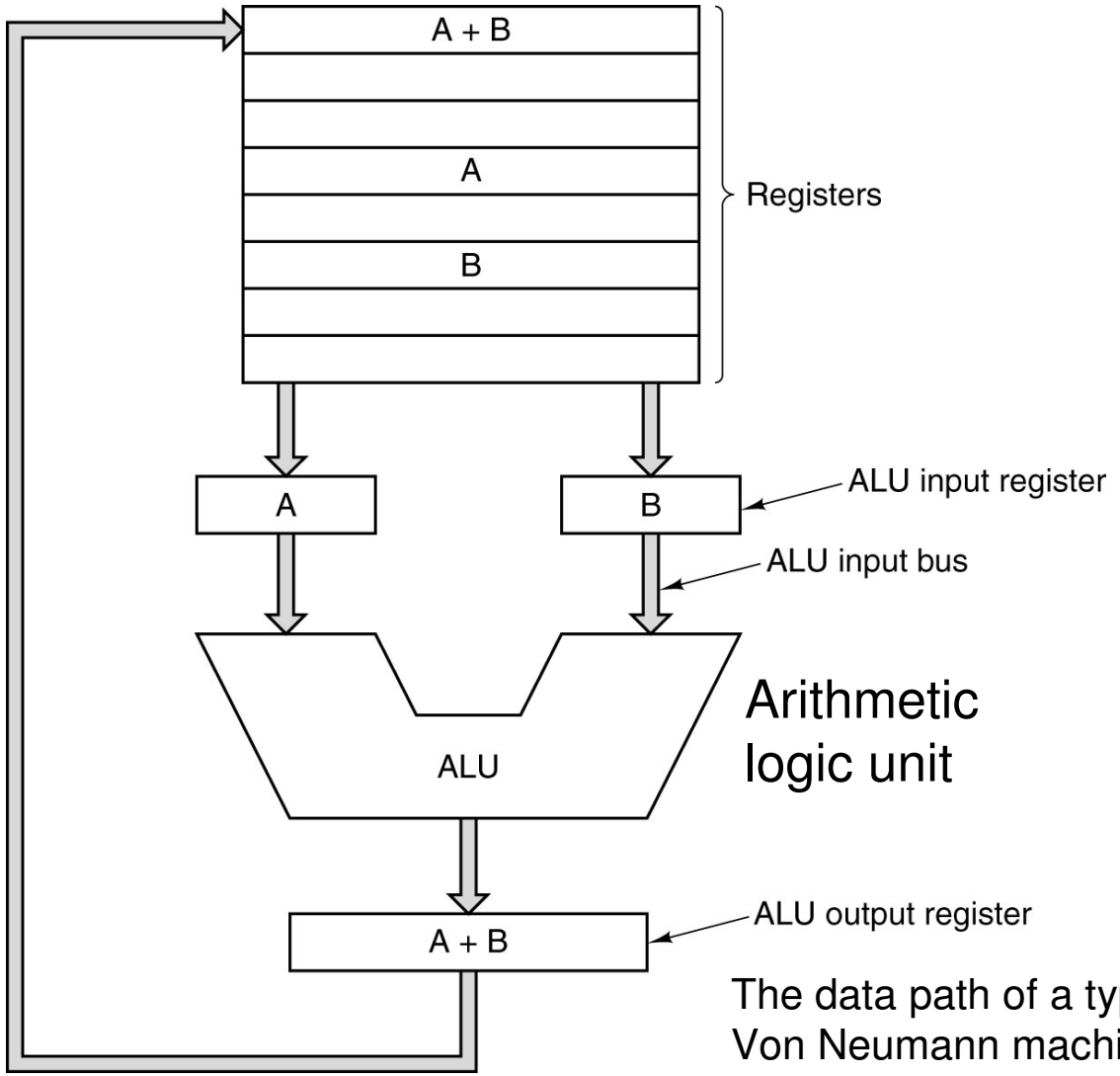


CPU Organization – Data Path

Data Path
=
Registers
+
ALU
+
Connection Buses

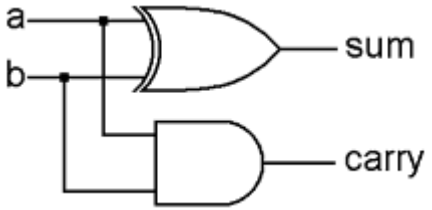
Data Path Cycle:
The process of running two operands through the ALU and storing the result back in the registers

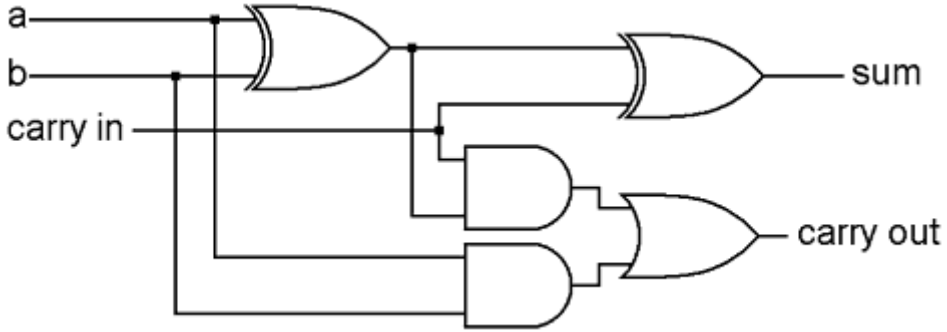
→The **heart** of most CPUs
→The **faster the data path** the faster the machine runs



The data path of a typical Von Neumann machine

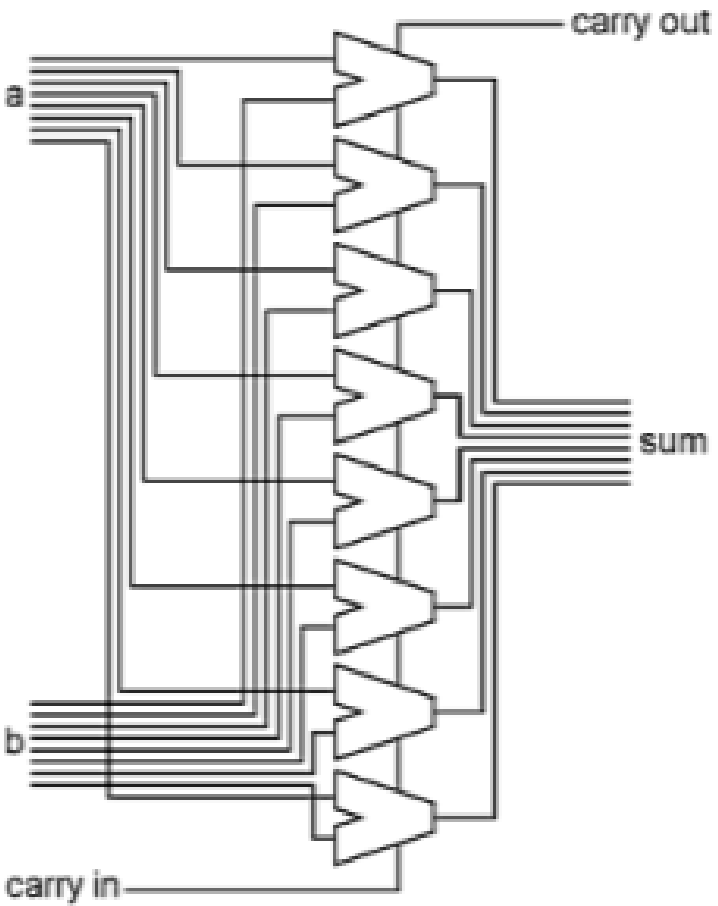
You already know how to build ALU : ADDer, Comparator, …

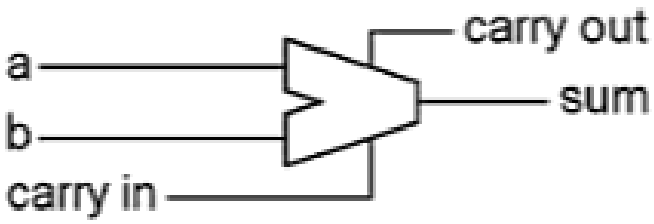


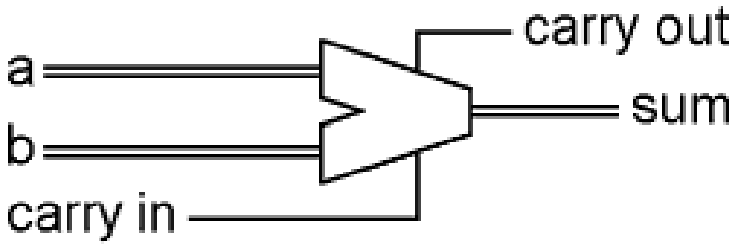


Carry-> 0 1 0 11

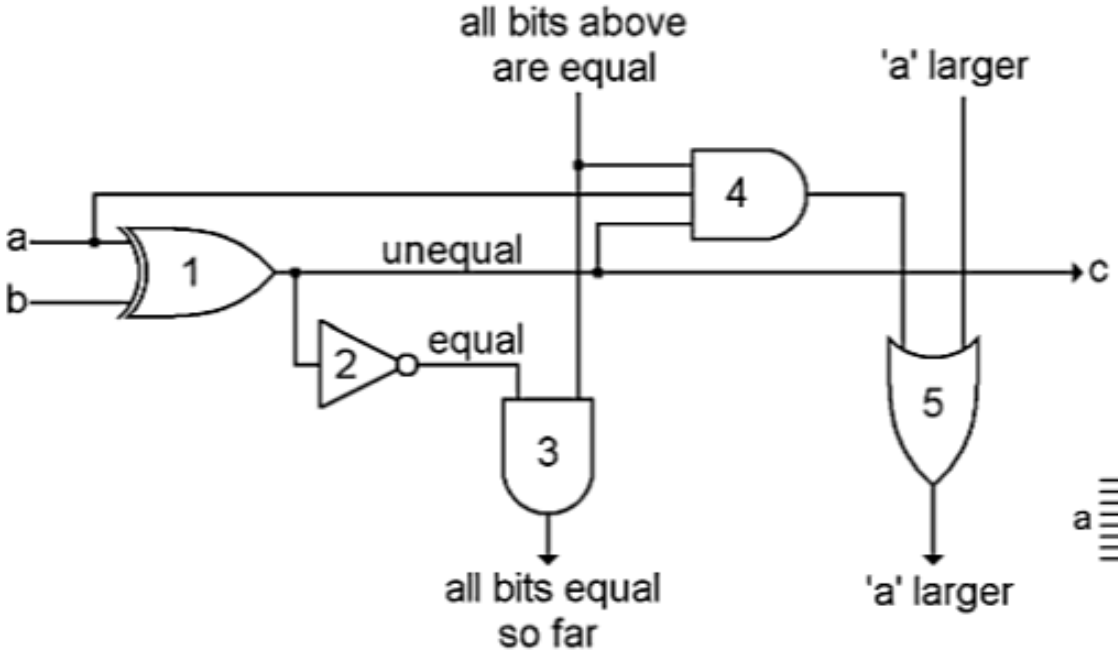
00	01	10	011
+01	+01	+01	+011
01	10	11	110

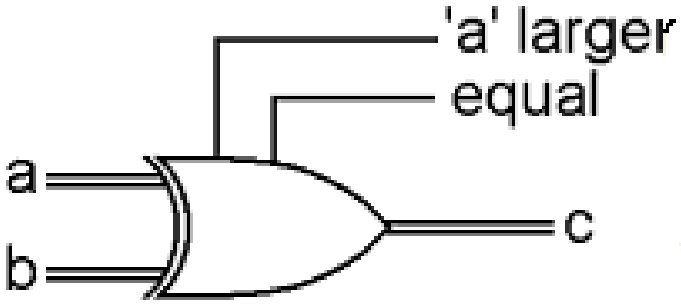


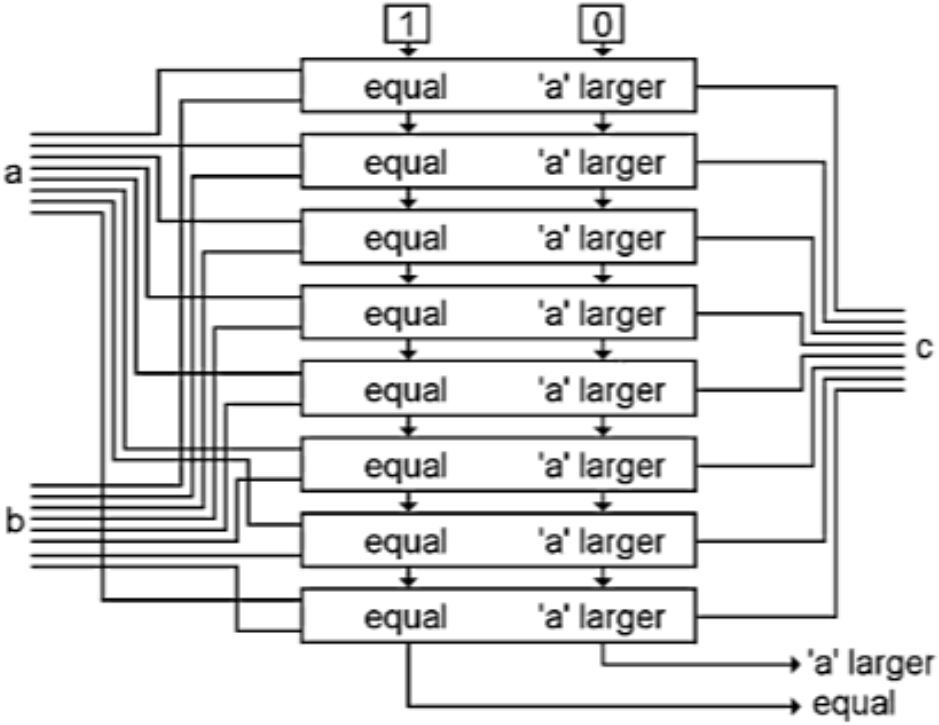




You already know how to build ALU : ADDer, Comparator, …

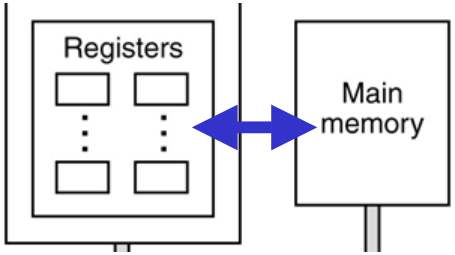




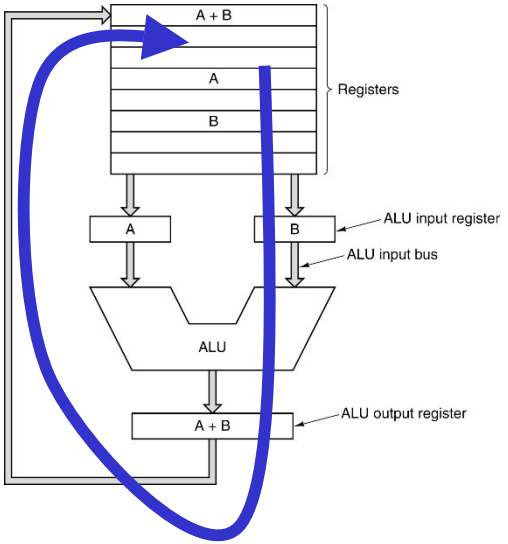


Instruction Types

- **Register-memory instructions**
  - Allow memory words to be fetched into registers
  - Allow registers to be stored back into memory

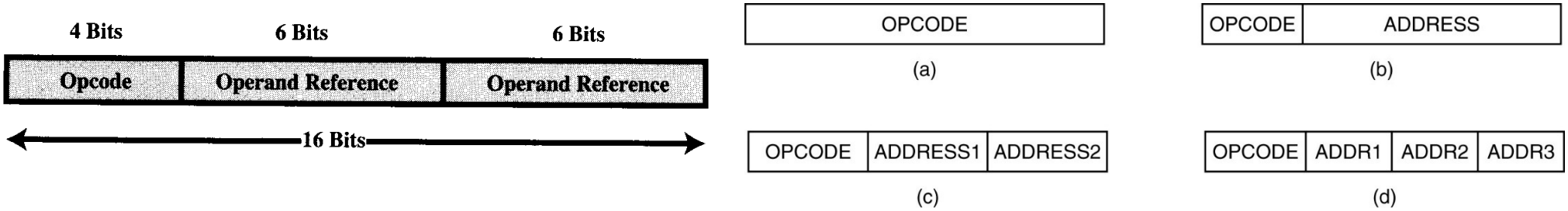


- **Register-register instructions**
  1. **Fetch** two operands from the register,
  2. **bring** them to the ALU input registers,
  3. **perform** some operation on them,
  4. **store** the result back in one of the registers.



Instruction Format

- Instructions are coded with bits
- Machine Instruction = the **opcode** + the **operands**



- Operation Code (*Opcode*)

  The instruction set assigns each operation to a unique code.

  integer addition → opcode five

  integer subtraction → opcode twelve

- Mnemonics are representations of Opcodes

ADD	Add
SUB	Subtract
MPY	Multiply
DIV	Divide
LOAD	Load data from memory
STOR	Store data to memory

Instruction Format – Operand References

- **Source Operand** Reference

  → the place where the inputs are for the operation

- **Result Operand** Reference

  → the place where the result is to be recorded

- **Next Instruction** Reference

  → the place of the next instruction

Program Execution - *CPU Cycle*

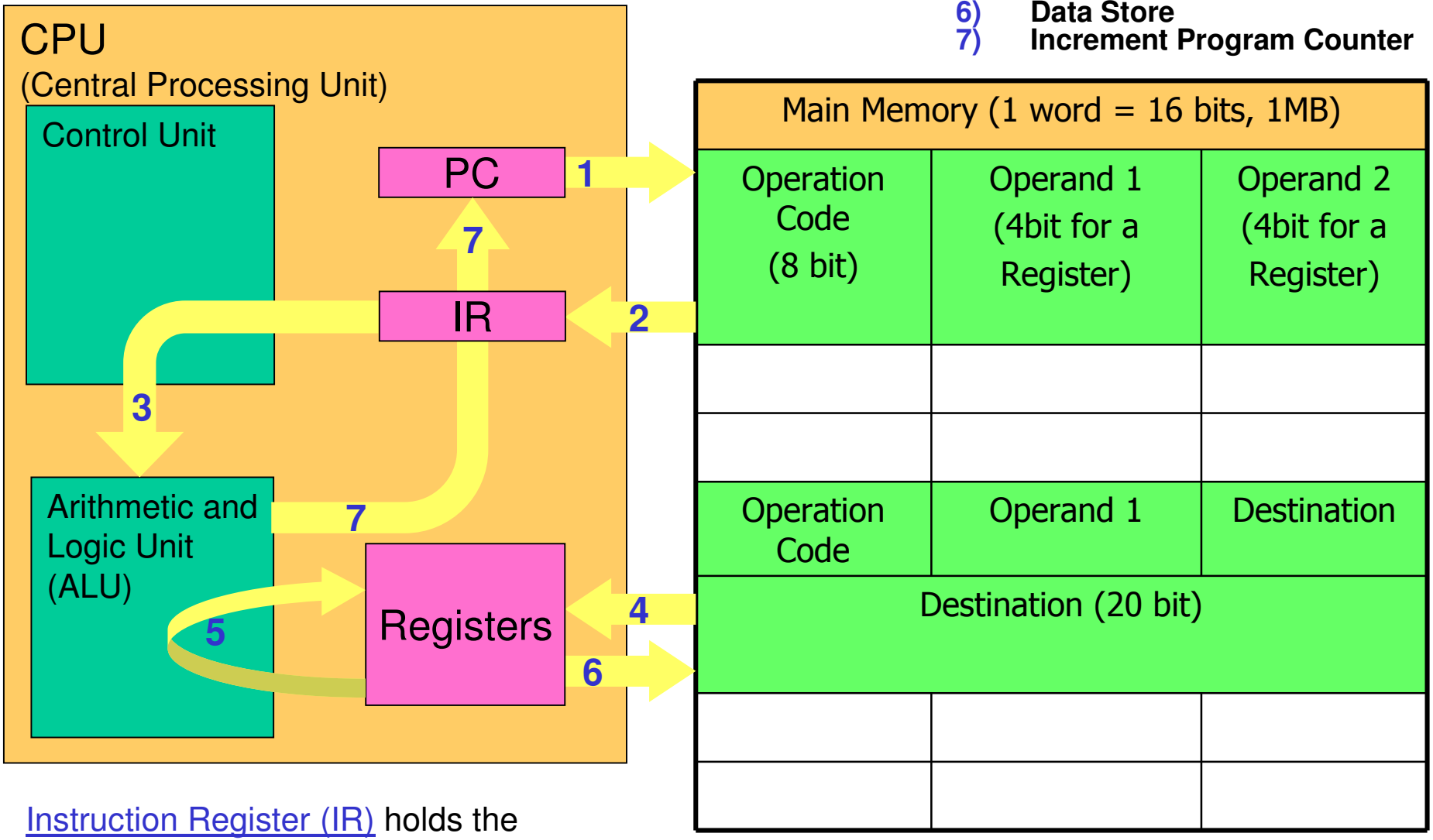
Fetch-decode-execute

1) Instruction Read
2) Instruction Fetch
3) Decode and Specify Instruction
4) Data Read
5) Data Process
6) Data Store
7) Increment Program Counter

Program Execution
CPU Cycle

Program Counter (PC) register points to the next instruction to be fetched for execution

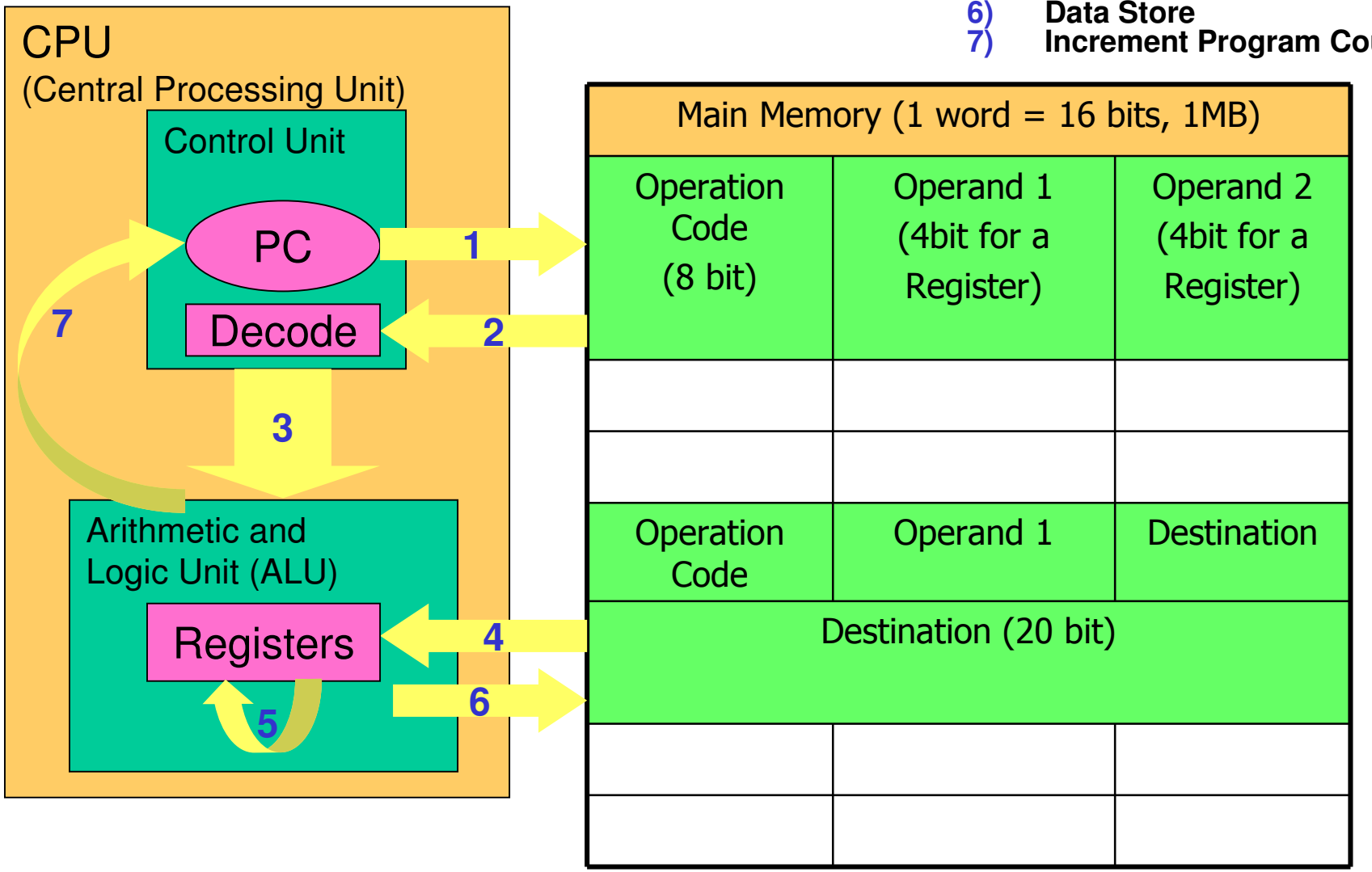
1) **Instruction Read**
2) **Instruction Fetch**
3) **Decode and Specify Instruction**
4) **Data Read**
5) **Data Process**
6) **Data Store**
7) **Increment Program Counter**



Instruction Register (IR) holds the instruction currently being executed

Program Execution
CPU Cycle

1) Instruction Read
2) Instruction Fetch
3) Decode and Specify Instruction
4) Data Read
5) Data Process
6) Data Store
7) Increment Program Counter



Clocks

- Every computer contains at least one clock that synchronizes the activities of its components.
- A fixed number of clock cycles are required to carry out each data movement or computational operation.
- The clock frequency, measured in megahertz or gigahertz, determines the speed with which all operations are carried out.
- Clock cycle time is the reciprocal of clock frequency.
  - An 800 MHz clock has a cycle time of 1.25 ns.

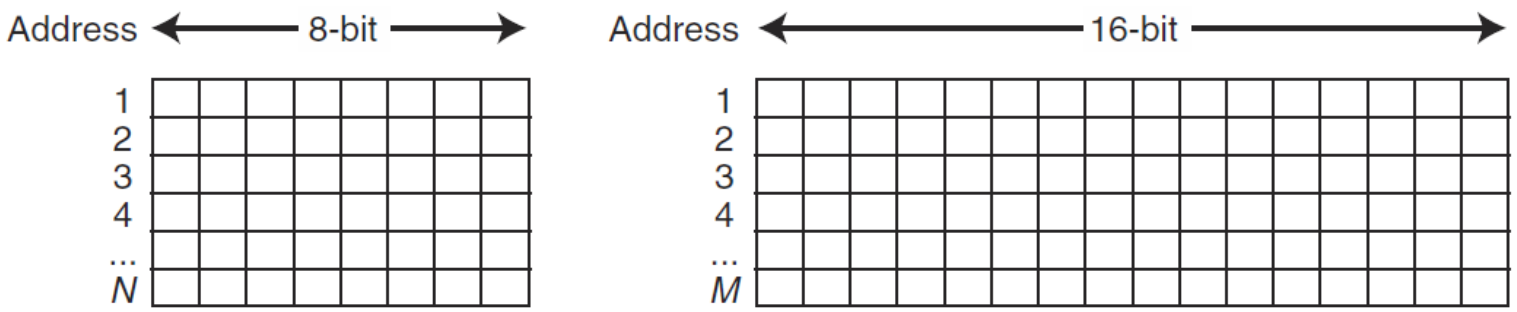
- Clock speed should not be confused with CPU performance.
- The CPU time required to run a program is given by the general performance equation:

$$\text{CPU Time} = \frac{\text{seconds}}{\text{program}} = \frac{\text{instructions}}{\text{program}} \times \frac{\text{avg. cycles}}{\text{instruction}} \times \frac{\text{seconds}}{\text{cycle}}$$

We can improve CPU throughput when we reduce the number of instructions in a program, reduce the number of cycles per instruction, or reduce the number of nanoseconds per clock cycle.

Memory Organization

- Computer memory consists of a linear array of addressable storage cells that are similar to registers.
- Memory can be byte-addressable, or word-addressable, where a word typically consists of two or more bytes.
- Memory is constructed of RAM chips, often referred to in terms of length × width.
- If the memory word size of the machine is 16 bits (2 bytes), then a 4M × 16 means the memory is 4M long (it has $4M = 2^2 x 2^{20} = 2^{22}$ words)



Measures of Capacity and Speed

- Kilo- (K) = 1 thousand = 10^3 and 2^{10}
- Mega- (M) = 1 million = 10^6 and 2^{20}
- Giga- (G) = 1 billion = 10^9 and 2^{30}
- Tera- (T) = 1 trillion = 10^{12} and 2^{40}
- Peta- (P) = 1 quadrillion = 10^{15} and 2^{50}
- Exa- (E) = 1 quintillion = 10^{18} and 2^{60}
- Zetta- (Z) = 1 sextillion = 10^{21} and 2^{70}
- Yotta- (Y) = 1 septillion = 10^{24} and 2^{80}

Memory Organization

- Physical memory usually consists of more than one RAM chip.
- Example: Suppose we have a memory consisting of 16 2K x 8 bit chips.
  - Memory is 32K = $2^5 \times 2^{10} = 2^{15}$
  - 15 bits are needed for each address.
  - We need 4 bits to select the chip, and 11 bits for the offset into the chip that selects the byte.