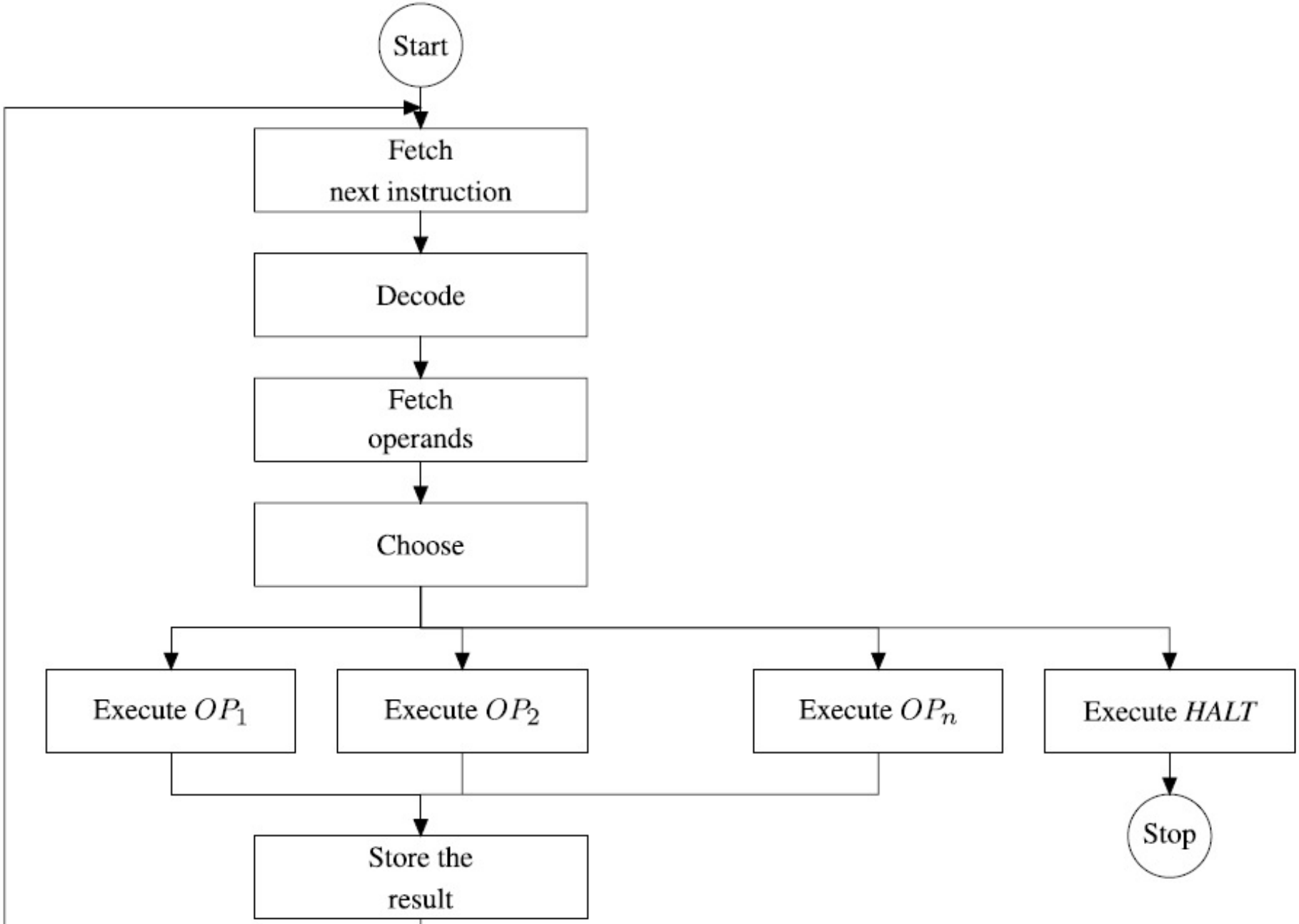
Programming Languages

Bibhas Adhikari

IIT Kharagpur

September 11, 2020

Execution cycle of interpreter



Machine language

Definition

Given a machine $\mathcal{M}_{\mathcal{L}}$, the language $\mathcal{L}$ 'understood" by $\mathcal{M}_{\mathcal{L}}$'s interpreter is called the machine language of $\mathcal{M}_{\mathcal{L}}$.

- Programs are written in the machine language of $\mathcal{M}_{\mathcal{L}}$ will be stored in the machine's storage structures so that they cannot be confused with other primitive data on which the interpreter operates
- Given that the internal representation of the programs executed by the machine $\mathcal{M}_{\mathcal{L}}$ is usually different from its external representation, then we should strictly talk about two different languages
  - Low-level language
  - High-level language

Low-level and High-level languages

Low-level language

A low-level language for a physical machine is its assembly language, which is a symbolic version of the physical machine (that is, which uses symbols such as ADD, MUL, etc., instead of their associated hardware binary codes). Programs in assembly language are translated into machine code using a program called an assembler.

High-level language

A language which supports the use of constructs that use appropriate abstraction mechanisms to ensure that they are independent of the physical characteristics of the computing machine. High-level languages are therefore suited to expressing algorithms in ways that are relatively easy for the human user to understand

The hardware machine

- It is physically implemented using logic circuits and electronic components

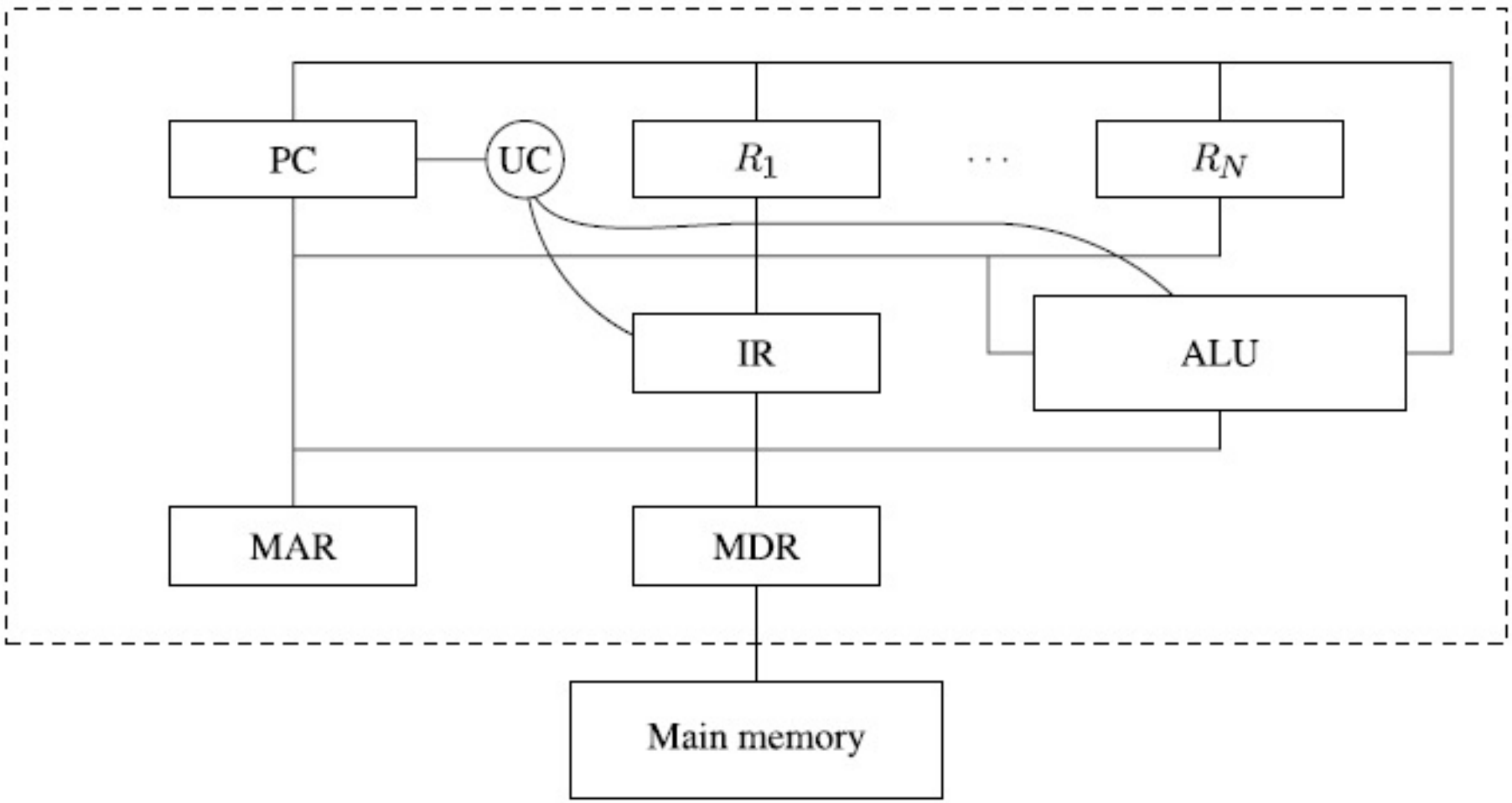


Figure: Structure of a calculator

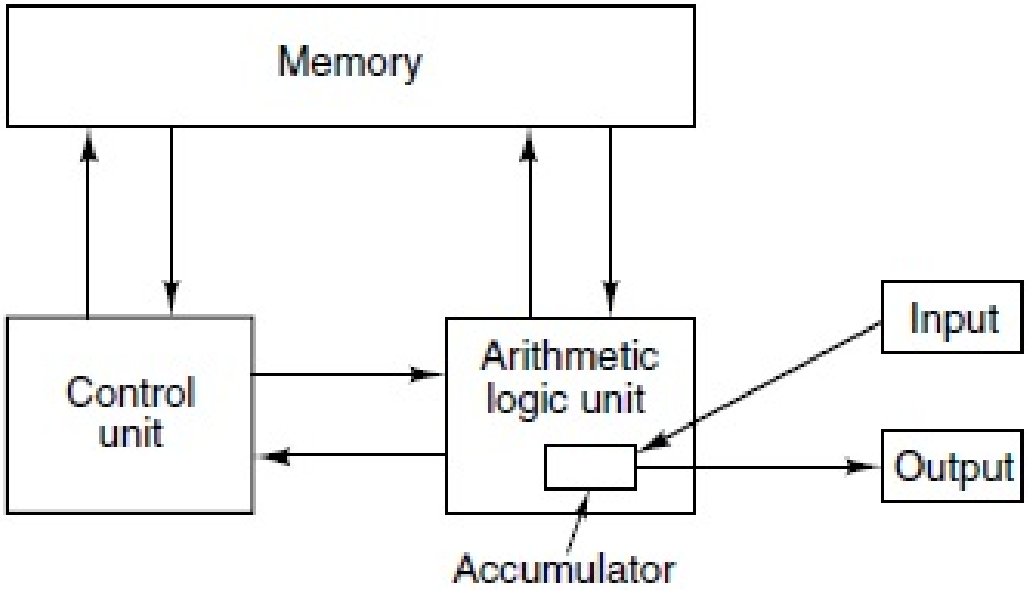
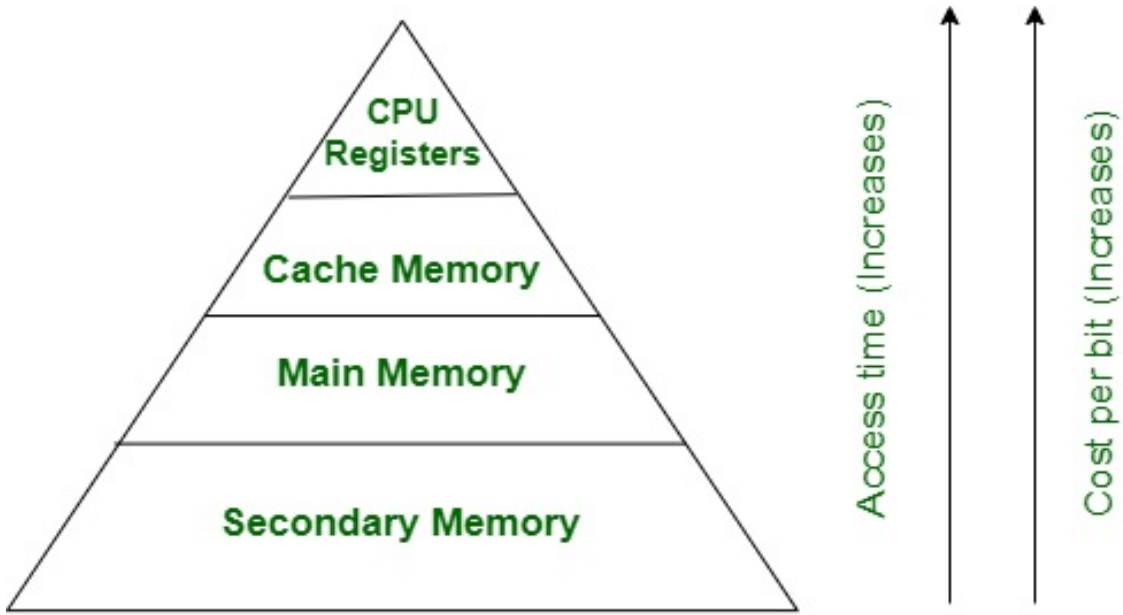


Figure: von Neumann Machine

Components of a machine

Memory

- Primary memory (main/internal memory) : Linear sequence of cells or words of fixed size
- Secondary memory (mass storage) : Optical or magnetic components
- Physical memory : Cache and registers (CPU)



Components of a digital computer

A digital computer is an interconnected system of processors, memories, and input/output devices

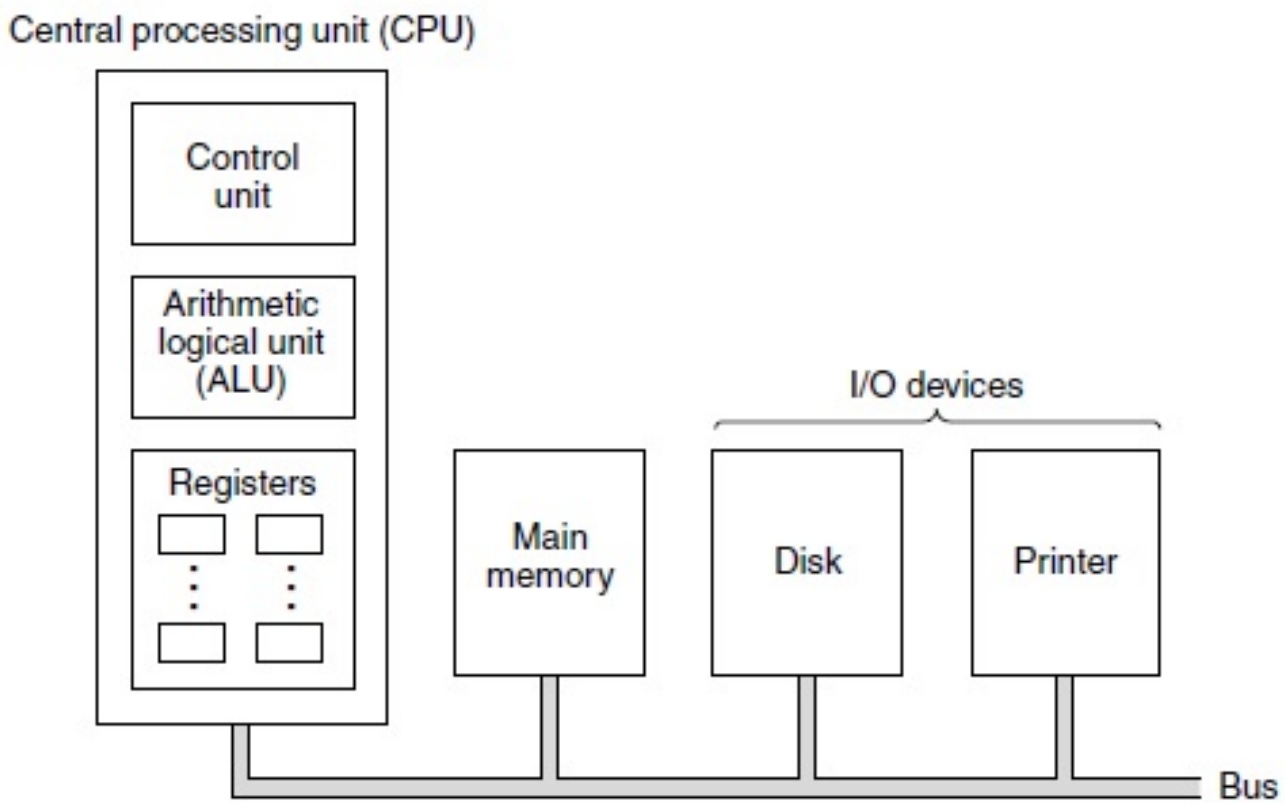


Figure: Organization of a simple computer with one CPU and two I/O devices

Organization of a simple bus-oriented computer

- The CPU (Central Processing Unit) is the "brain" of the computer
- The CPU also contains a small, high-speed memory used to store temporary results and certain control information

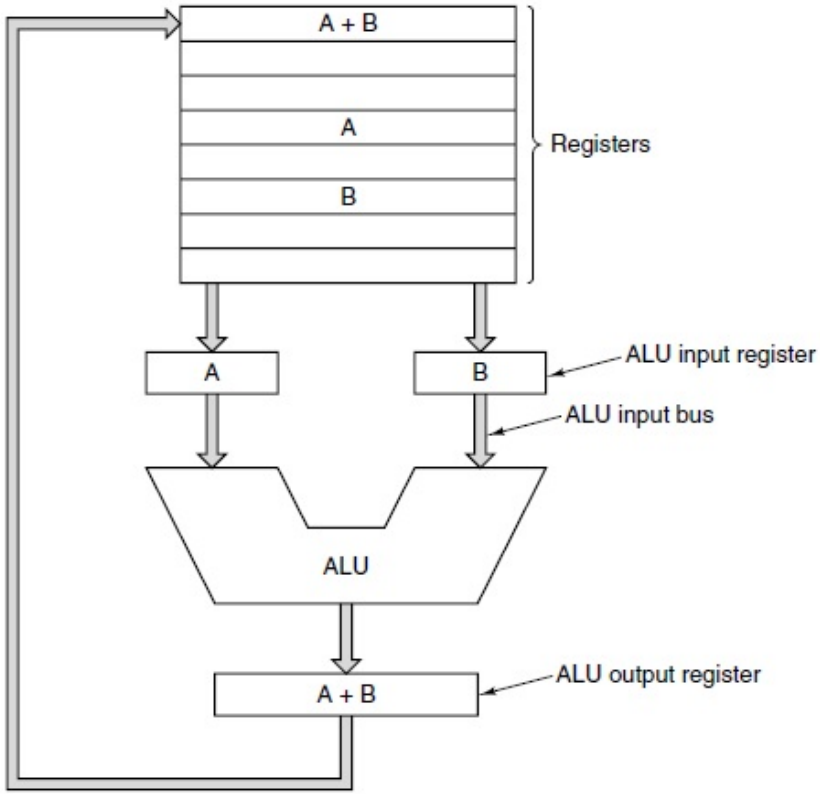


Figure: The data path of a typical von Neumann machine

fetch-decode-execute cycle of CPU

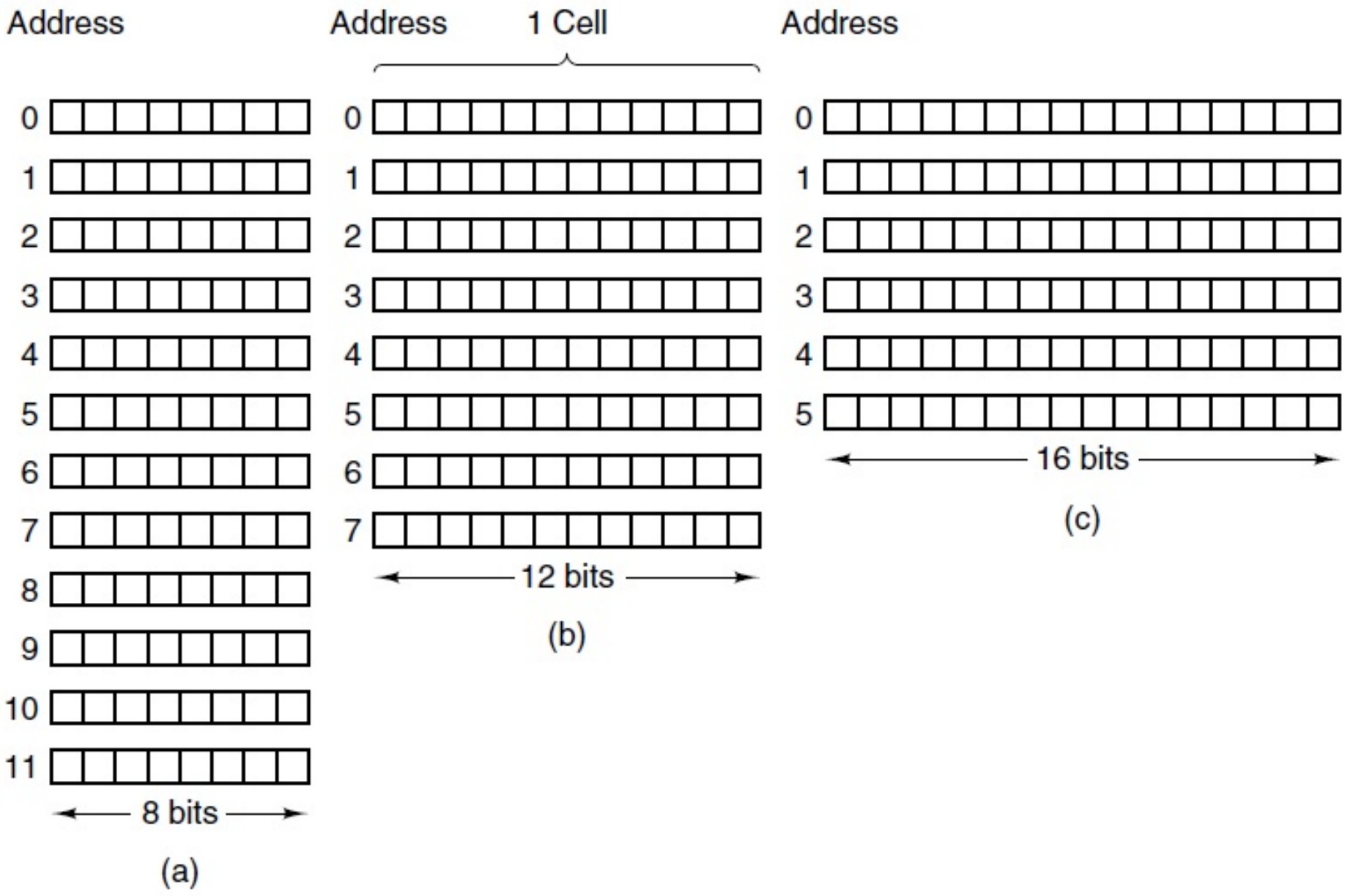
- Fetch the next instruction from memory into the instruction register
- Change the program counter to point to the following instruction.
- Determine the type of instruction just fetched.
- If the instruction uses a word in memory, determine where it is.
- Fetch the word, if needed, into a CPU register.
- Execute the instruction.
- Go to step 1 to begin executing the following instruction.

Primary Memory

- The basic unit of memory is the binary digit, called a bit
- A bit may contain a 0 or a 1
- Each cell has a number, called its address, by which programs can refer to it
- If a memory has n cells, they will have addresses 0 to $n-1$.
- All cells in a memory contain the same number of bits
- If a cell consists of k bits, it can hold any one of 2^k different bit combinations.
  Any sequence of $0, 1$s of length k is nothing but a a function

$$f : \{1, \ldots, k\} \to \{0, 1\}$$

Memory organization



- If an address has m bits, the maximum number of cells addressable is 2^m
- In the preceding picture (a), an address needs at least 4 bits in order to express all the numbers from 0 to 11
- A 3-bit address is sufficient for (b) and (c)
- Homework: A memory with 2^{12} cells of 8 bits each and a memory with 2^{12} cells of 64 bits each need 12-bit addresses

Memory organization

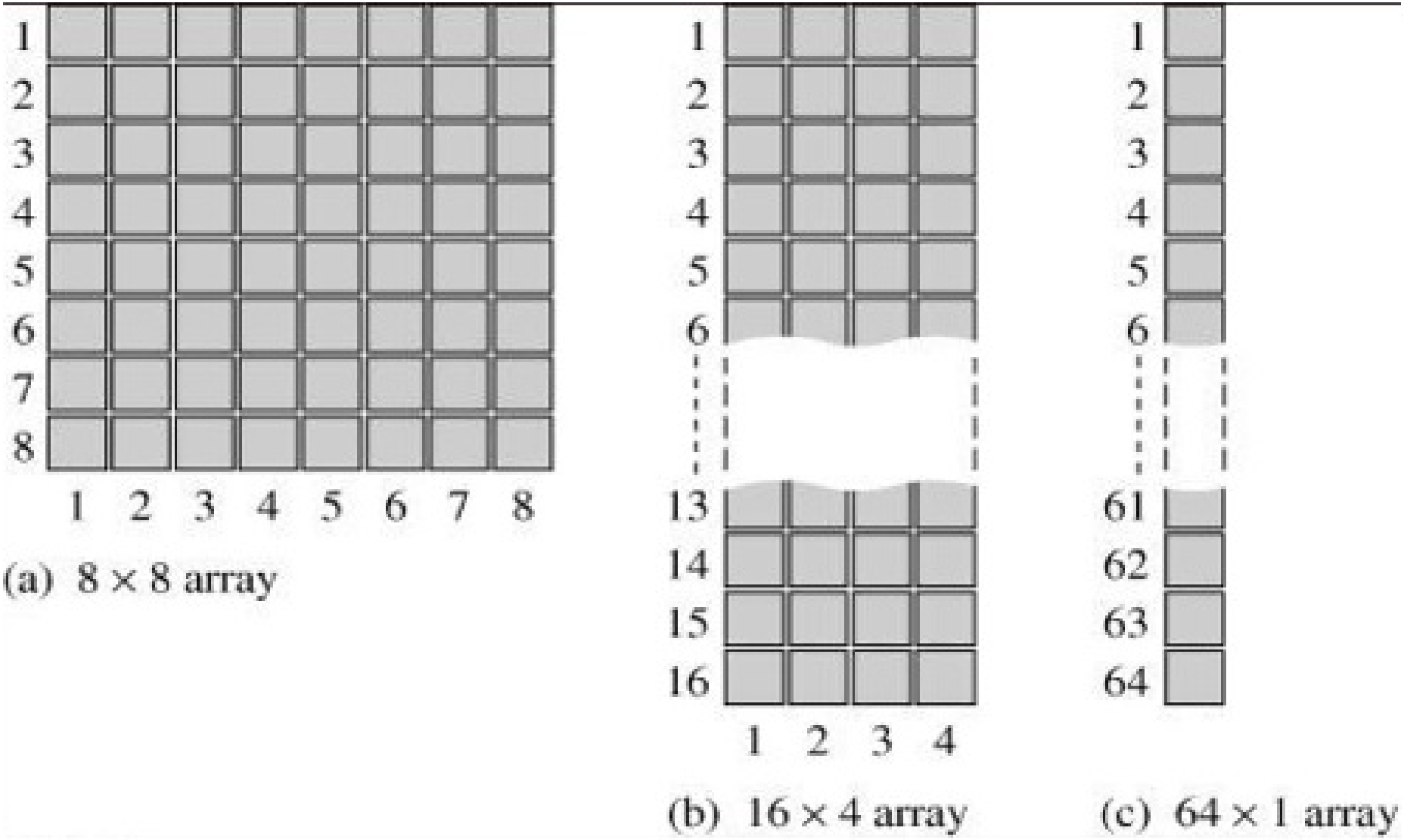


Figure: A 64-cell memory array organized in 3 different ways

Memory organization

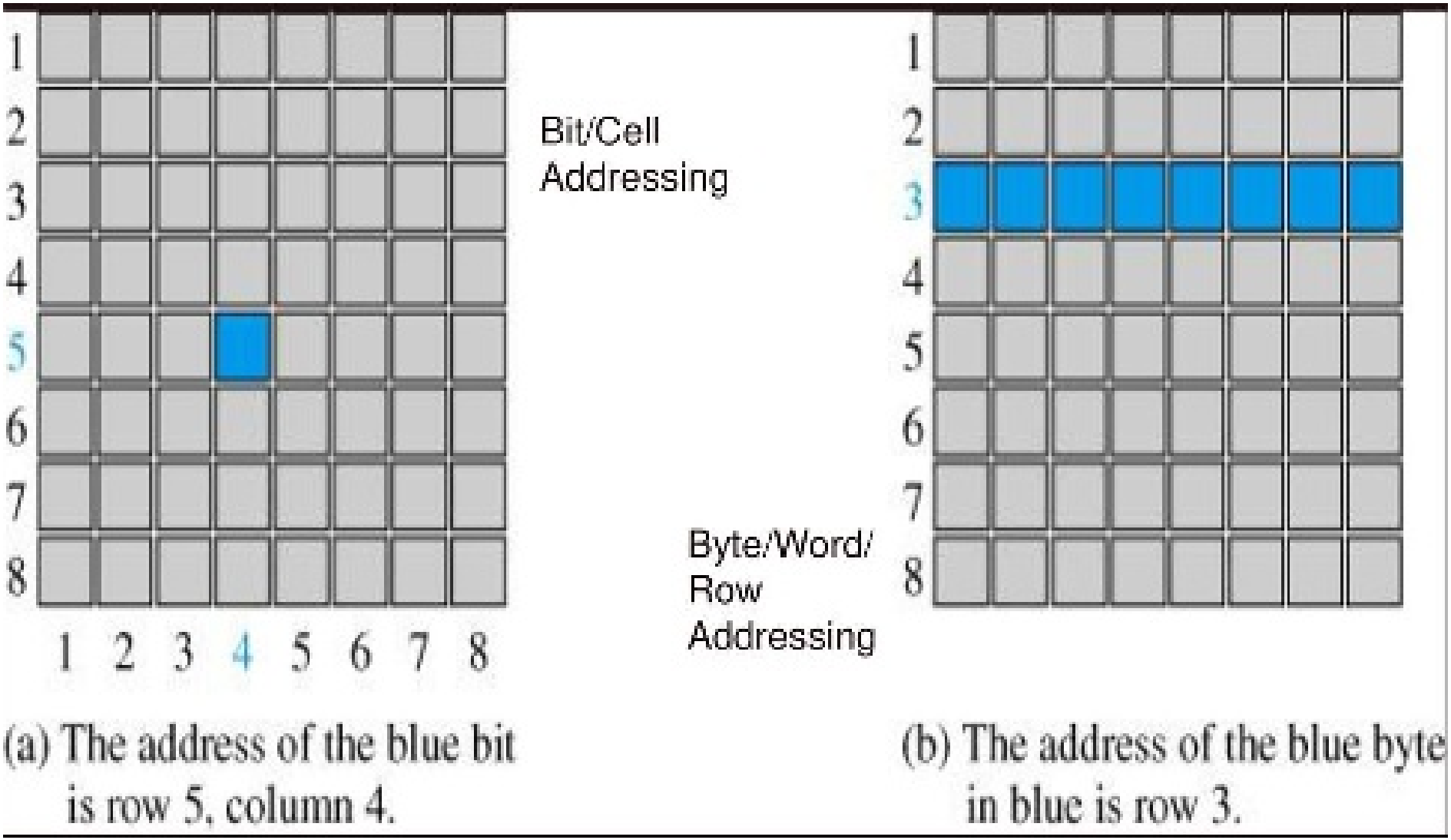


Figure: Example of memory address

Memory organization

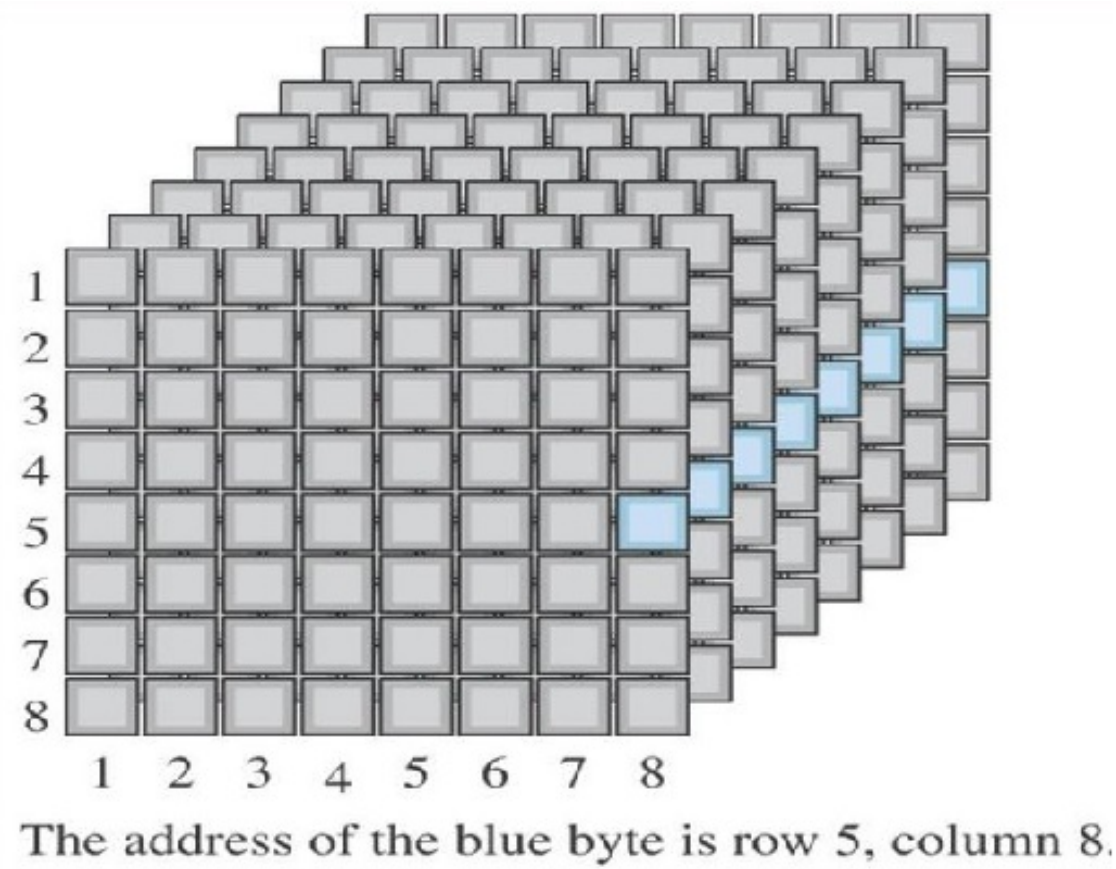


Figure: Example of memory address

Commercially available memory

The number of bits per cell for some computers that have been sold commercially:

Computer	Bits/cell
Burroughs B1700	1
IBM PC	8
DEC PDP-8	12
IBM 1130	16
DEC PDP-15	18
XDS 940	24
Electrologica X8	27
XDS Sigma 9	32
Honeywell 6180	36
CDC 3600	48
CDC Cyber	60

Figure: History of memory which are commercially available

Some more about memory

Memory units

- Bit: A single binary digit
- Nibble: A group of 4 bits accessed together
- Byte: A group of 8 bits accessed together (Nearly all manufacturers have standardised on an 8-bit cell)
- Word: A group of binary bits whose size is typical unit of access for the memory (exp. 1 byte, 2 bytes, 4 bytes, 8 bytes etc.)
  - A computer with a 32-bit word has 4 bytes/word, whereas a computer with a 64-bit words has 8 bytes/word
  - A 32-bit machine will have 32-bit registers and instructions for manipulating 32- bit words – such as adding two words together

Some more about memory

Memory data elements

- Memory data: A bit or group of bits to be stored into or accessed from memory cells
- Memory operations: Operations on memory data supported by the memory unit. Typically, read and write operations over some data element (bit, byte, word etc.)
  - Write operation: Puts data into a specified address in memory
  - Read operation: Takes data out of a specified address in memory
- Primary memory exps.:
  - Random Access Memory (RAM)
  - Read Only Memory (ROM)

Memory size

Number of words (no. of address lines) $\times$ Data width (no. of bits per word)

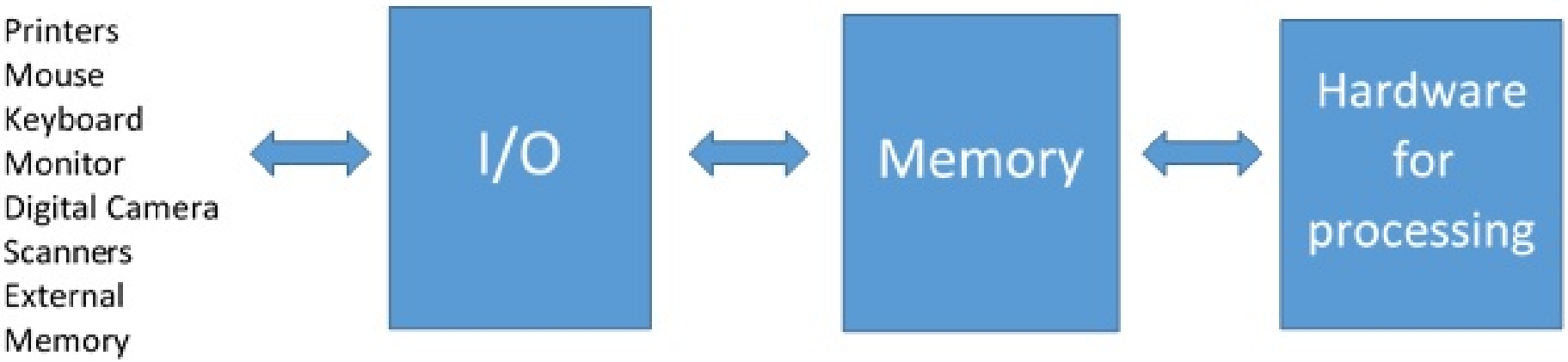
- Units: K (kilo): 2^{10}, M (mega): 2^{20}, G(giga): 2^{30}

Errorcorrecting codes

- Computer memories occasionally make errors due to voltage spikes on the power line, cosmic rays, or other causes
- To safeguard from such errors, some memories use error-detecting or error-correcting codes
- Suppose that a memory word consists of m data bits to which we will add r redundant, or check, bits. Let the total length be n (i.e., $n = m + r$). An n-bit unit containing m data and r check bits is often referred to as an n-bit codeword.
- Hamming distance: The number of bit positions in which two codewords differ. Example: the Hamming distance between 10001001 and 10110001 is 3
- If a memory read turns up an invalid codeword, the computer knows that a memory error has occurred

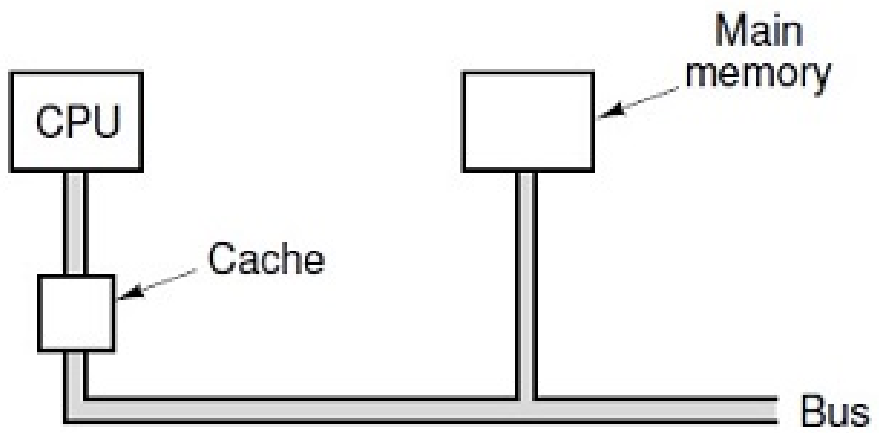
Memory basic process (computer systems)

1. Content from memory is sent to hardware (usually consists of registers) to be processed
2. The processed information is then returned to the same or different memory address
3. Input and output devices may also interact with memory



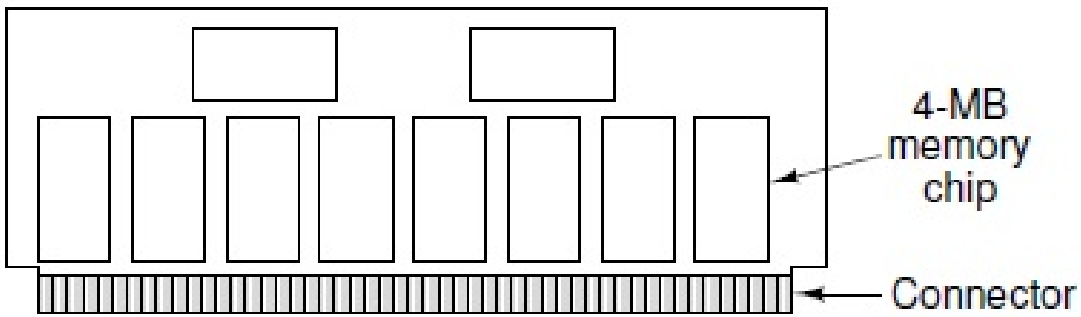
Cache memory (computer systems)

- The most heavily used memory words are kept in the cache
- When the CPU needs a word, it first looks in the cache
- Only if the word is not there, it goes to main memory.
- If a substantial fraction of the words are in the cache, the average access time can be greatly reduced.
- The cache is logically between the CPU and main memory



Memory packaging (computer systems)

- Until the early 1990s, memory was manufactured, bought, and installed as single chips
- Chip densities went from 1k bits to 1M bits and beyond, but each chip was a separate unit
- Today, a different arrangement is often used
- A group of chips, typically 8 or 16, is mounted on a tiny printed circuit board and sold as a unit
- This unit is called a SIMM (Single Inline Memory Module) or a DIMM (Dual Inline Memory Module), depending on whether it has a row of connectors on one side or both sides of the board.



Predefined data types

Data Type

- Integers and reals : integers can be represented by 1s or 2s complement numbers, and reals as floating point numbers
- Characters: Alphanumeric characters are also implemented as sequences of binary numbers (ASCII)

Language of the machine

Let us denote such a machine as $\mathcal{MH}_{\mathcal{LH}}$, where the $\mathcal{LH}$ denotes the corresponding machine language ($\mathcal{H}$ stands for hardware)

A typical instruction with two operands

opCode Operand1 Operand2

- *ADD R5, R0,*
- *ADD (R5), (R0)*

Components of a physical/hardware machine

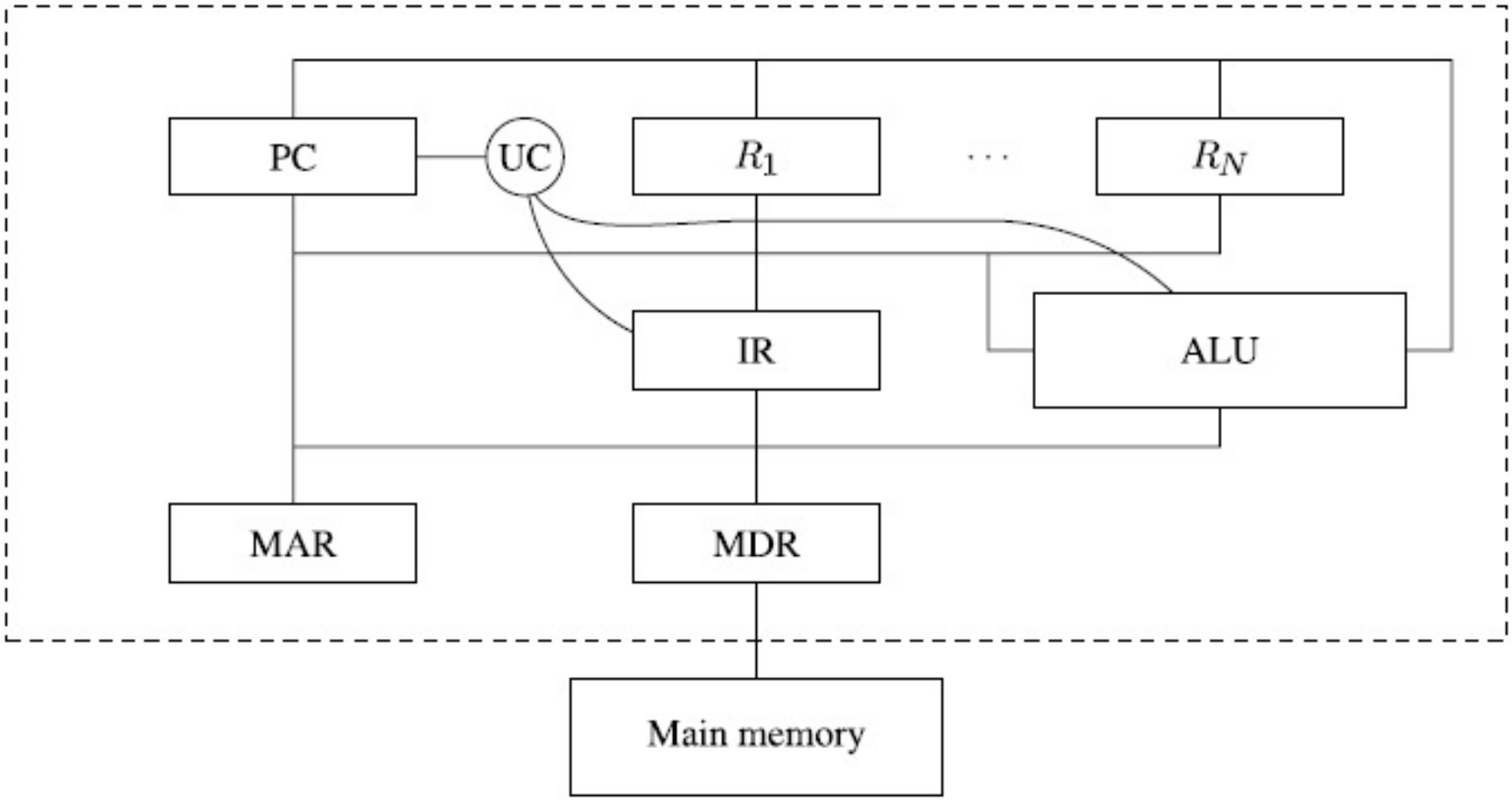


Figure: Structure of a caculater

Components of a physical/hardware machine

Conventional

- ALU (Arithmetic and Logic Unit) : for processing primitive data are the usual arithmetic and logical operations
- Program Counter (PC) register: for the control of instruction sequence execution
- Memory Address Register (MAR) and Memory Data Register (MDR) : to handle data transfer, the CPU registers interfacing with main memory
- Memory processing

Modern

Hardware stack instead of registers, there is stack data structure together with the push and pop operations

Implementation of interpreter

Control Unit (CU, fetch-decode-execute cycle)

- fetch phase : In the fetch phase, the next instruction to be executed is retrieved from memory. This is the instruction whose address is held in the PC register. The instruction, which, it should be recalled, is formed of an operation code and perhaps some operands, is then stored in a special register, called the Instruction Register (IR)
- decode phase : instruction stored in the instruction register is decoded using special logic circuits
- execute phase : primitive hardware operation is actually executed, for example, using the circuits of the ALU if the operation is an arithmetic or logical one

Implementation of language and abstract machine

- Implementing a programming language $\mathcal{L}$ means implementing an abstract machine which has $\mathcal{L}$ as its machine language
- Hardware implementation of $\mathcal{M}_{\mathcal{L}}$'s constructs:
  - implementation in hardware
  - simulation using software

Implementation in an ideal scenario

Assumptions

- Let $\mathcal{L}$ be a language
- Let $\mathcal{M}_{\mathcal{L}}$ be the abstract machine on which $\mathcal{L}$ will be implemented
- Let $\mathcal{M}_{o_{\mathcal{L}_o}}$ be an abstract machine, called a *host machine* in which a construct (machine, interpreter, program) of a language is already implemented

Goal

Implementing $\mathcal{L}$ on $\mathcal{M}_{o_{\mathcal{L}_o}}$ takes place using a "translation" from $\mathcal{L}$ to $\mathcal{L}_o$.

- purely interpreted implementation
- purely compiled implementation.

Some notations

- Let $\mathcal{P}rog^{\mathcal{L}}$ denote the set of all possible programs that can be written in language $\mathcal{L}$
- Let $\mathcal{D}$ denote the set of input and output data
- Partial function: Let A and B be two sets. Then a partial function $f : A \rightarrow B$ is a function expect the fact that f can be undefined for some elements of A. For example,

$$f(x) = \begin{cases} 1 \ if \ x = 1 \\ undefined, \ otherwise \end{cases}$$

represents the following program:

```
read(x);
if (x == 1) then print(x);
else while (true) do skip
```

Definition

A program written in $\mathcal{L}$ can be described as a partial function

$$\mathcal{P}^{\mathcal{L}} : \mathcal{D} \rightarrow \mathcal{D}$$

such that

$$\mathcal{P}^{\mathcal{L}}(\text{Input}) = \mathit{Output}$$

if the execution of $\mathcal{P}^{\mathcal{L}}$ on input data terminates and produces output