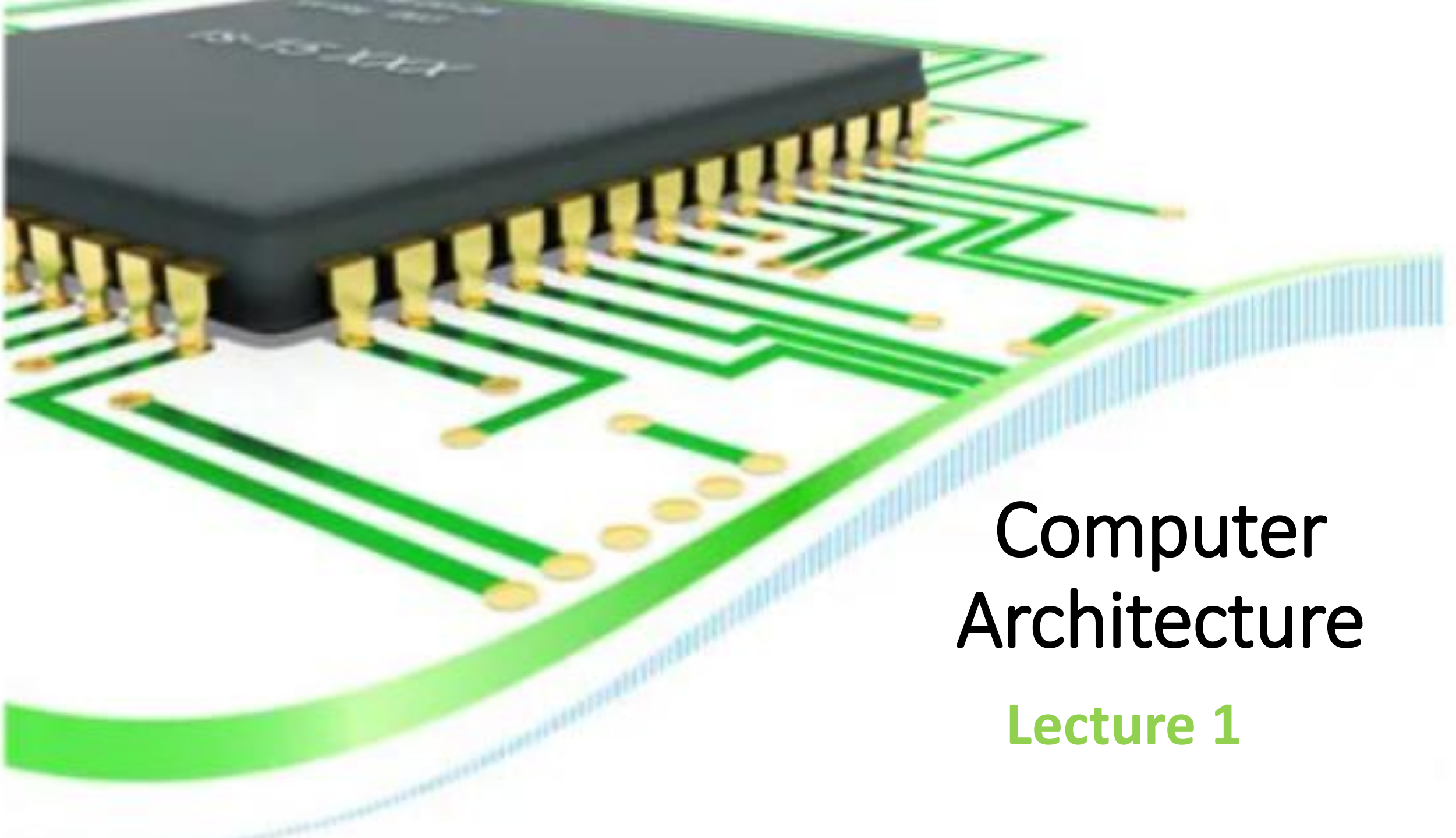


Computer Architecture

Lecture 1

Our Team

Module Leader:

- Dr. Noura El Maghawry

noura.elmaghawry@bue.edu.eg

Teaching Assistants:

- Zeina Swilam

zeinab.swilam@bue.edu.eg

- Ismail Hamdy

Ismail.hamdy@bue.edu.eg

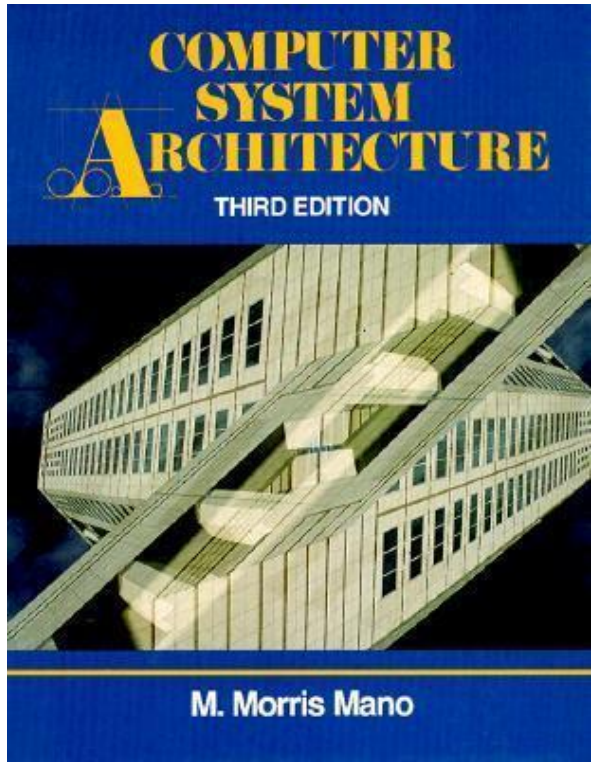
Office hours will be announced soon on the e-learning

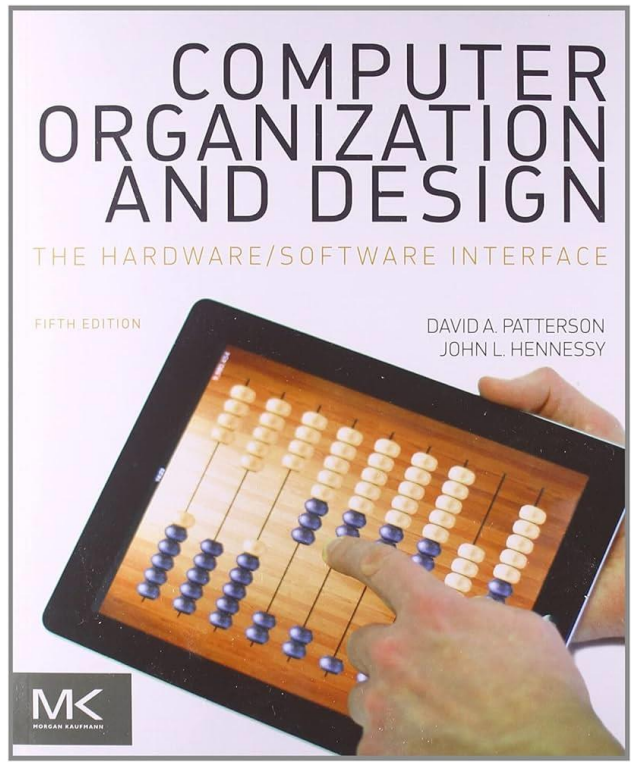
Assessments

- Coursework:
  - ❑ In-class Assessment 1 –> Week 8 20%
  - ❑ In-class Assessment 2 –> Week 11 20%

- Unseen Written Exam 60%

Reading List





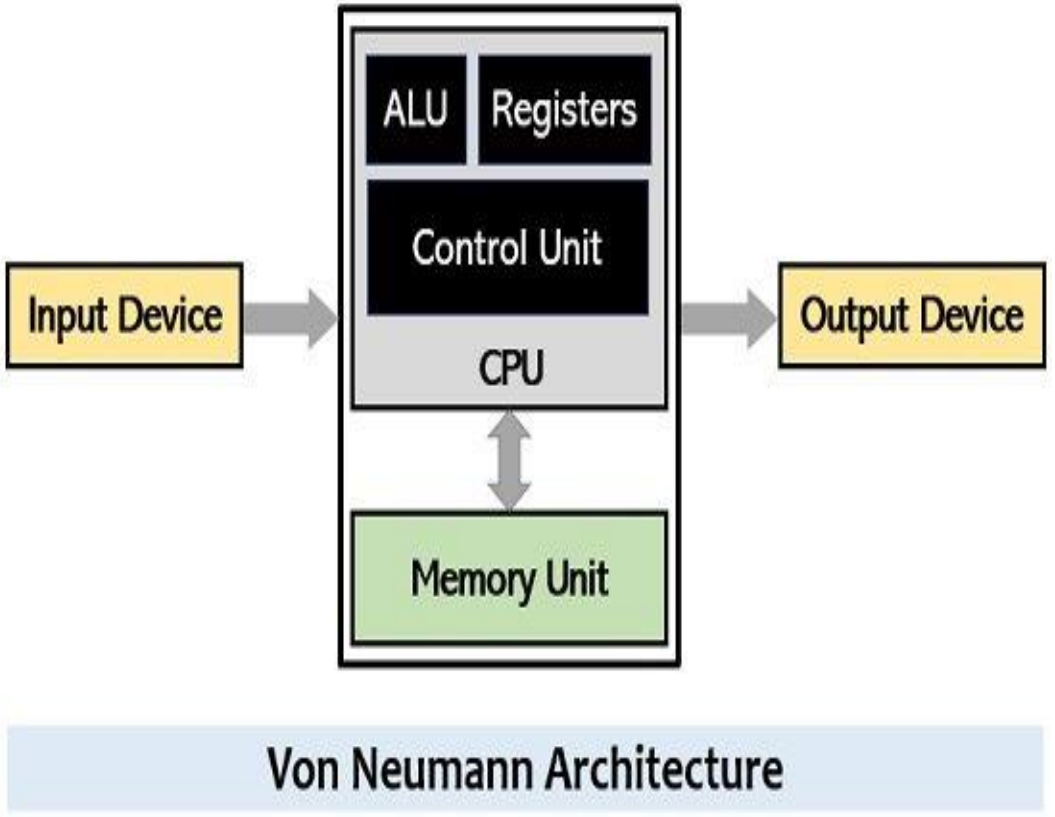
Computer Architecture

- Computer architecture is a specification detailing how a set of software and hardware technology standards interact to form a computer system.

- In short, computer architecture refers to how a computer system is designed and what technologies it is compatible with.

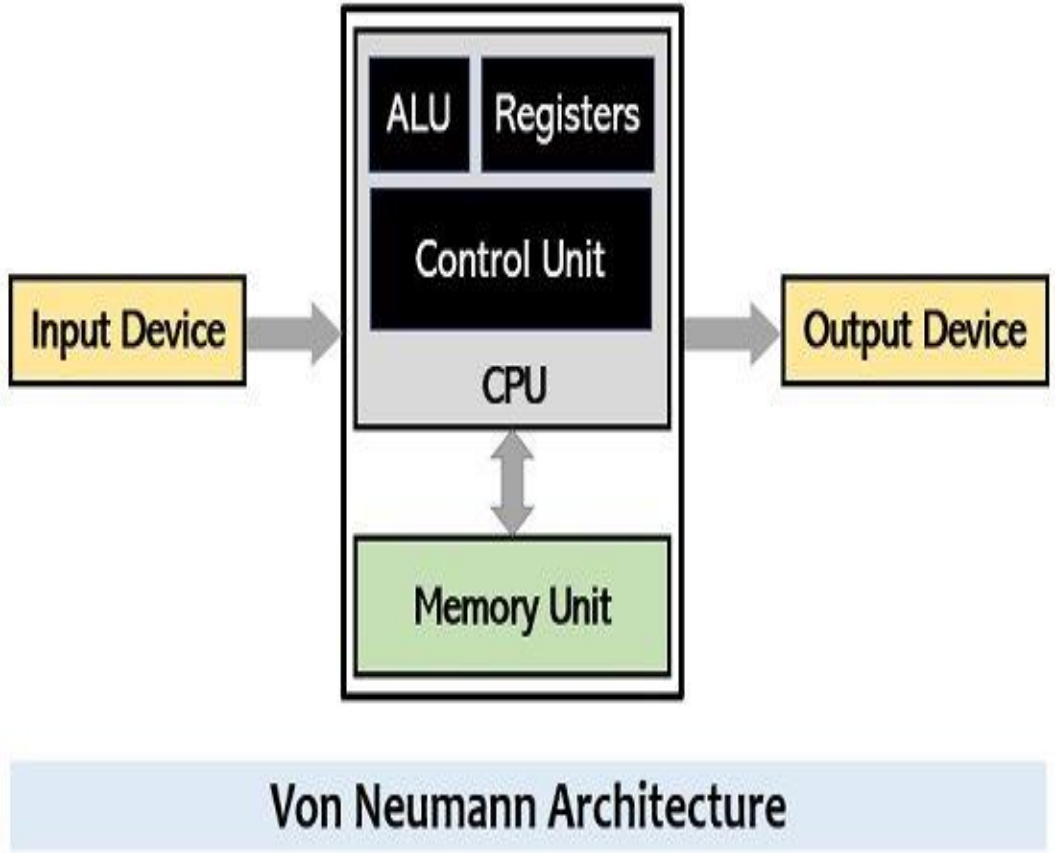
Von Neumann Architecture

- Von Neumann envisioned the structure of a computer system as being composed of three Subsystems:
- Central Processing unit (CPU) composed of:
  - Control Unit (CU)
  - Arithmetic Logic Unit (ALU)
  - Registers
- Memory
- Input and Output Subsystems



Subunits of CPU

- **Control Unit (CU):** The brain within the brain.
- **Arithmetic Logic Unit (ALU) :** responsible for performing arithmetic operations (add, sub,…).
- **Registers:** Temporarily storage area within the CPU.



General Scenario for an Instruction cycle

Control Unit

Control lines

=1

Control Signals

R1

1

R2

4

R3

5

ALU

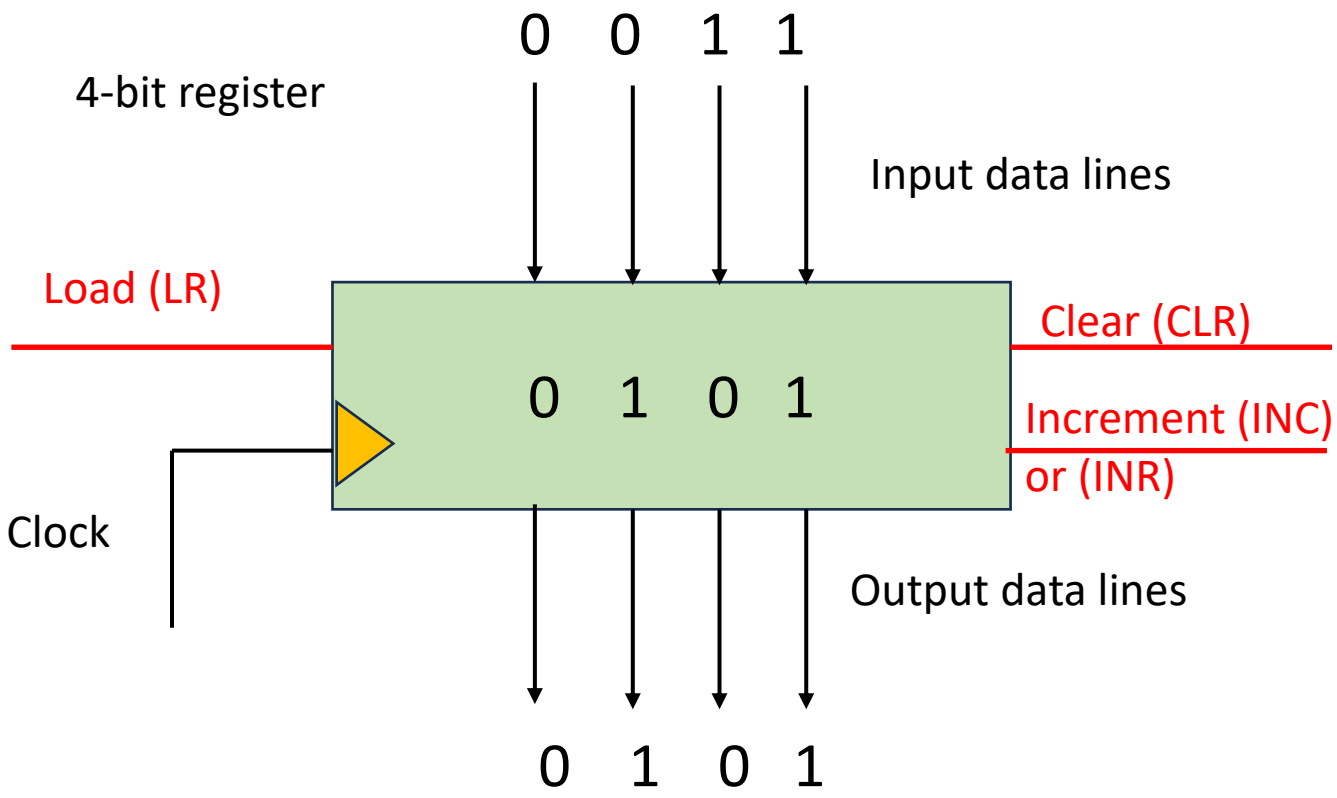
Memory

1+4

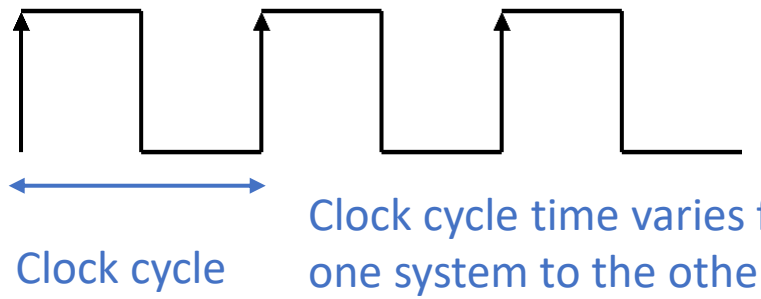
5

Register

- A special, high-speed temporary storage area within the CPU.
- There is a limited number of registers in each architecture.



Clear	Load	Increment	Operation
0	0	0	No change
0	0	1	Increment value by 1
0	1	X	Load new input
1	X	X	Clear value to Zero



Control Signals to Registers

- **Increment (INC or INR):**

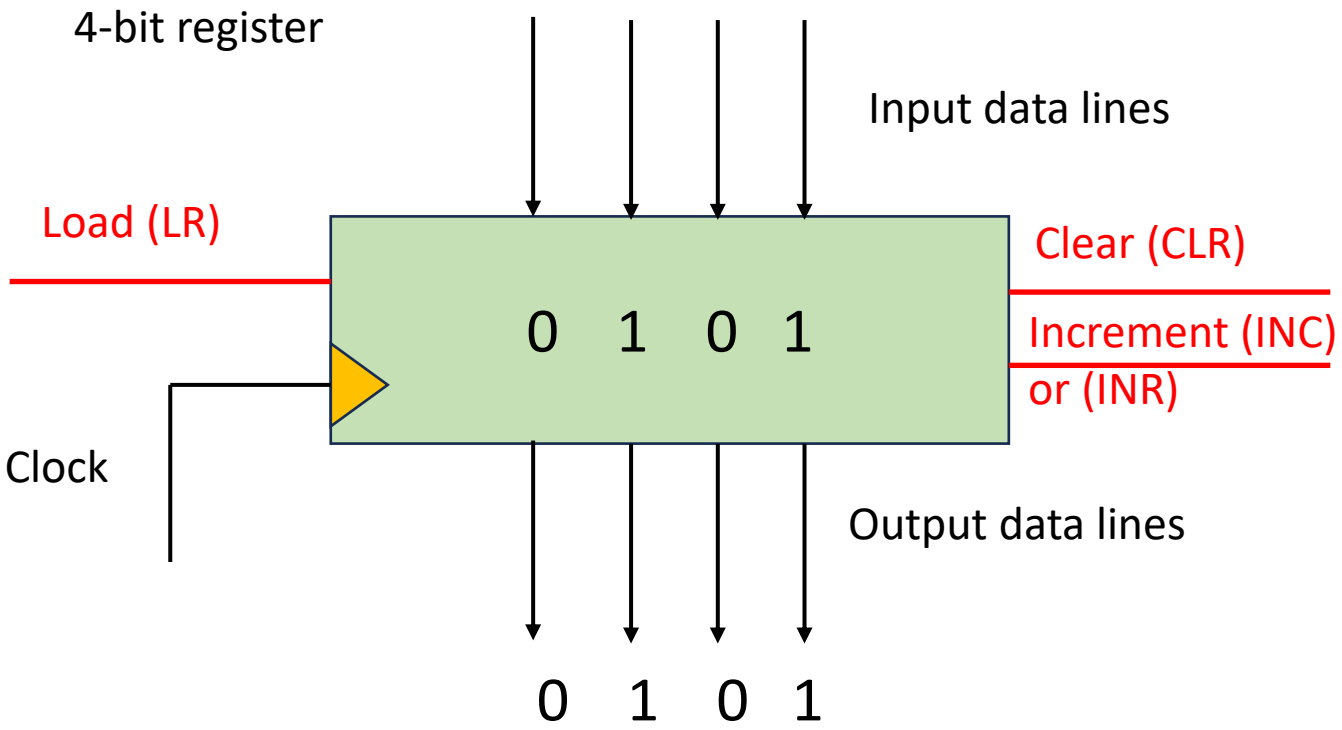
Increments the value inside the register.

- **Load (LD):**

Loads a new value inside the register.

- **Clear (CLR):**

Clears the value inside the register to be zero.



Types of Registers

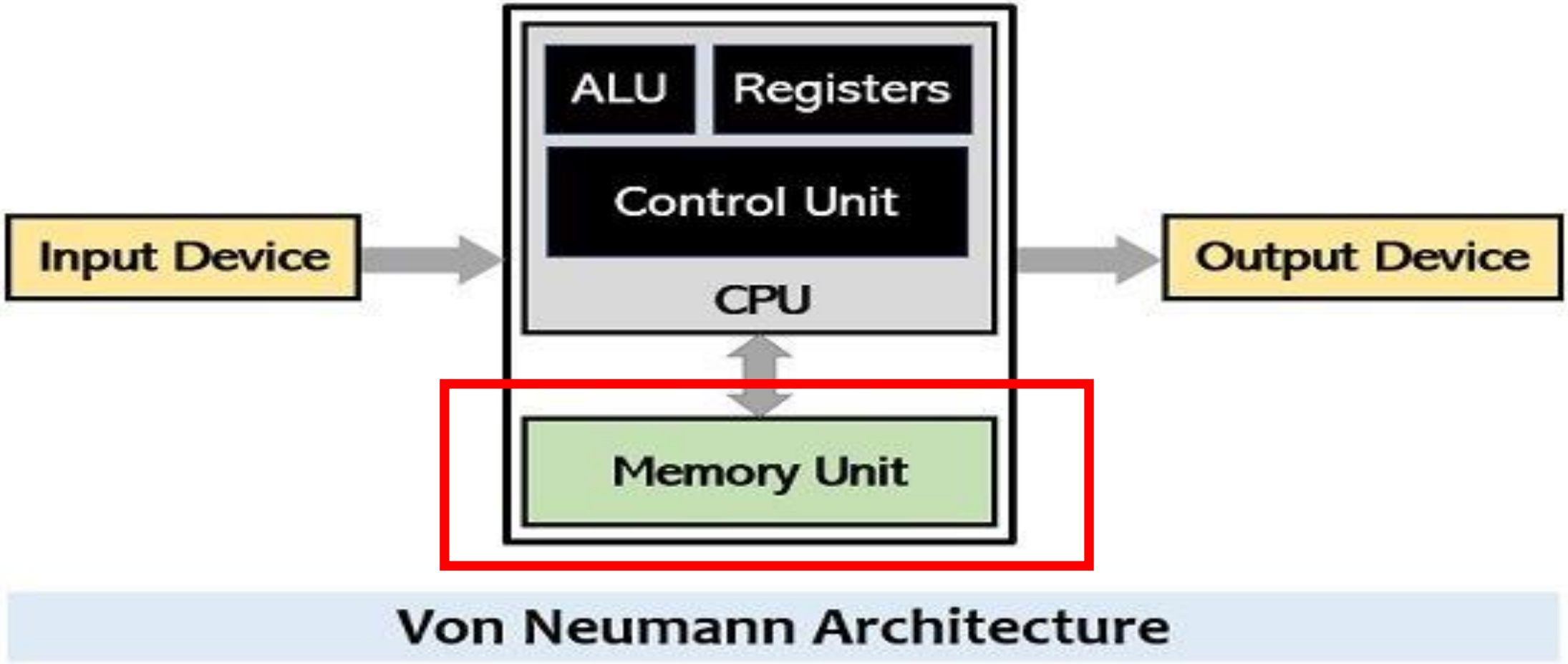
- ***General purpose registers***

These are mainly used for operands for logical and arithmetic operations.

- ***Special purpose registers***

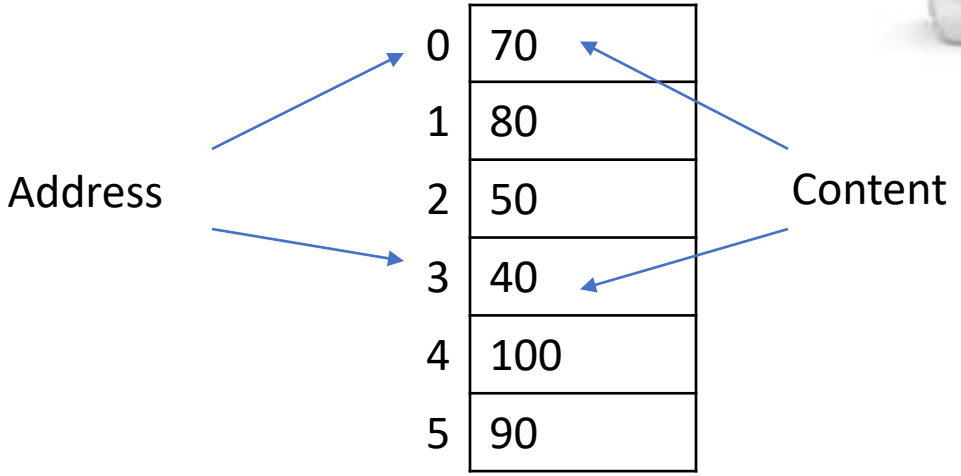
A special purpose register is one that has a specific task to carry out.

Von Neumann Architecture

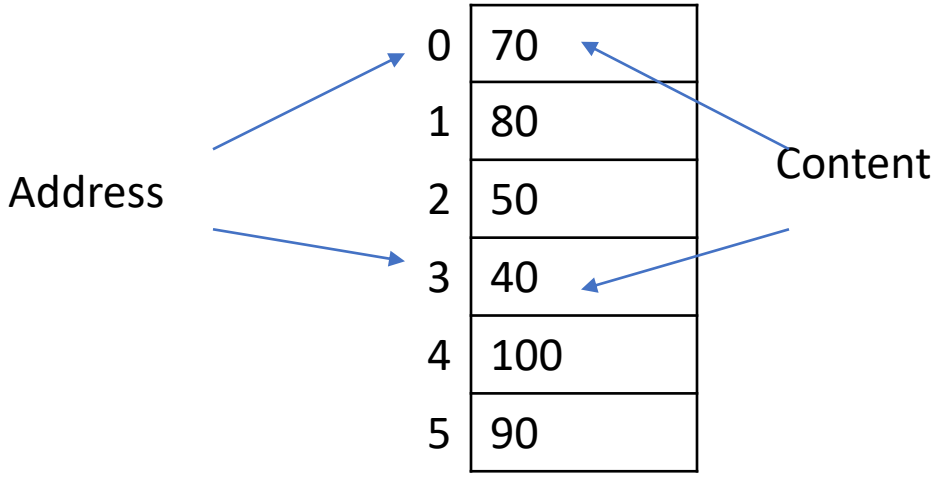


Memory Unit

- Memory Unit is an array of cells
  - All cells have the same size
  - Each cell has its own *address*



Memory



1 byte=8 bits

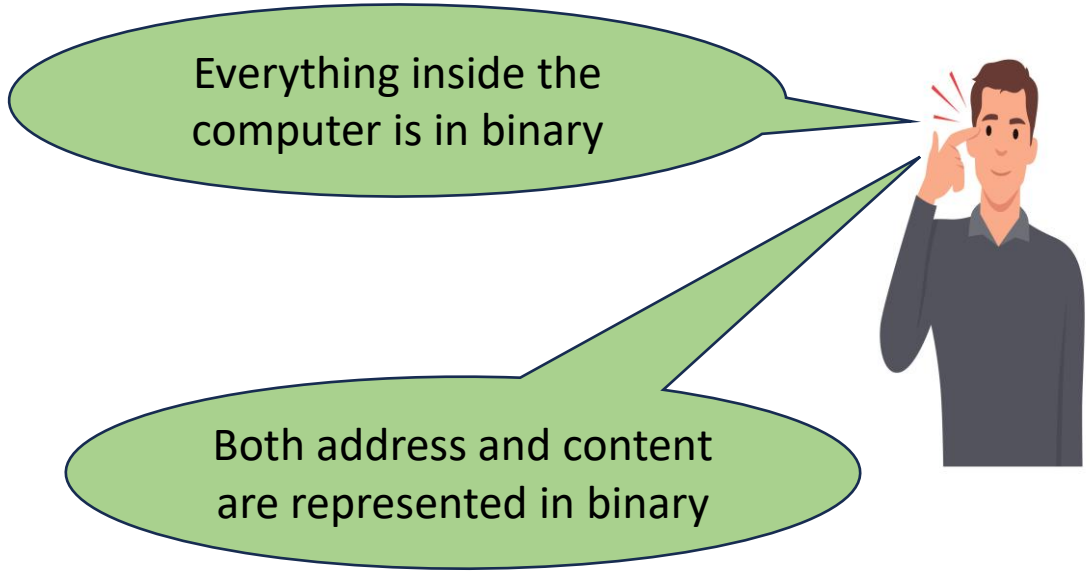
1 Kbyte=1024 bytes

= 2^{10} bytes

1 MB = 2^{10} Kbytes

= 2^{20} bytes

1 Gb = 2^{30} bytes



Assume 16 cells memory.

Each cell size is 8-bits

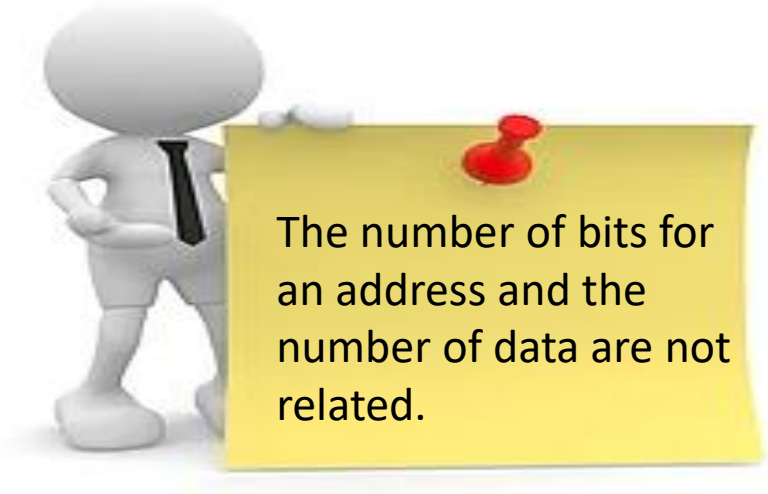
0	5
15	7

Byte-addressable memory

0 = 0000	00000101
15 = 1111	00000111

If I know how many cells are in the memory, could I know how many bits I need for an address?

Number of bits for an address = $\log_2$(number of cells)



The number of bits for an address and the number of data are not related.

Memory Unit

- **Memory Address:**
- A number that is assigned to each cell in a computer memory.

- **Address lines:**
- The number of bits(lines) to represent memory address.

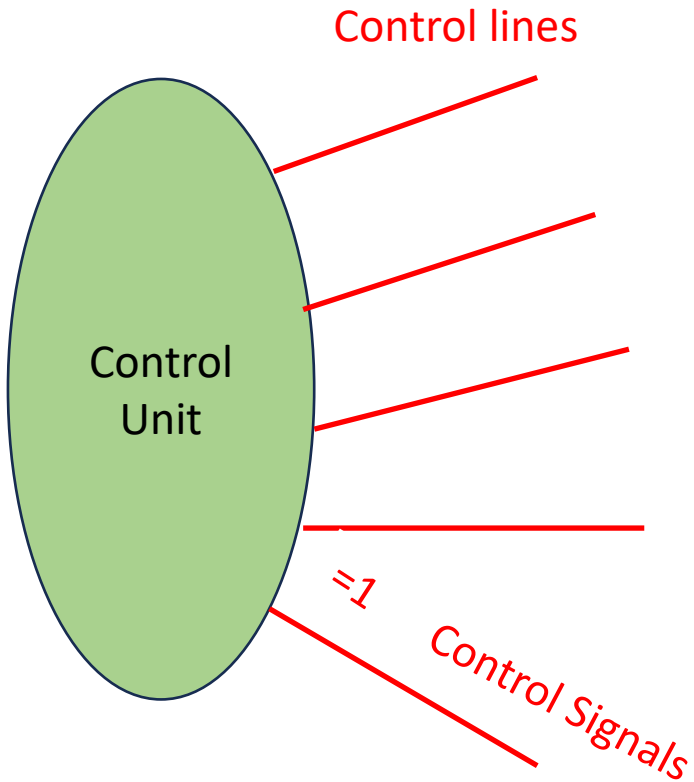
- **Data lines:**
- The number of bits(lines) to represent the data in each memory address.

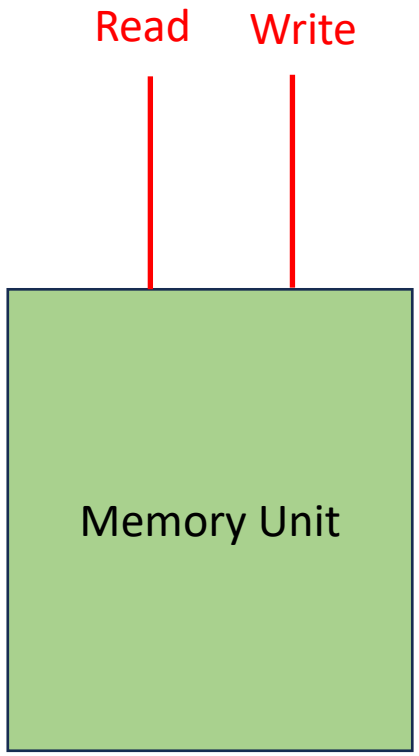
Check your understanding

- A byte-addressable memory unit having 128 cells
- How many data lines and address lines are needed for this memory unit?

Data lines = 8-bits (lines)
Address lines = 7-bits (lines)

Control Signals to Memory





Read	Write	Memory Operation
0	0	None
1	0	Reading from memory is enabled
0	1	Writing to memory is enabled

Memory Unit

- The Memory is described by the number of words that the memory could store and the word size.

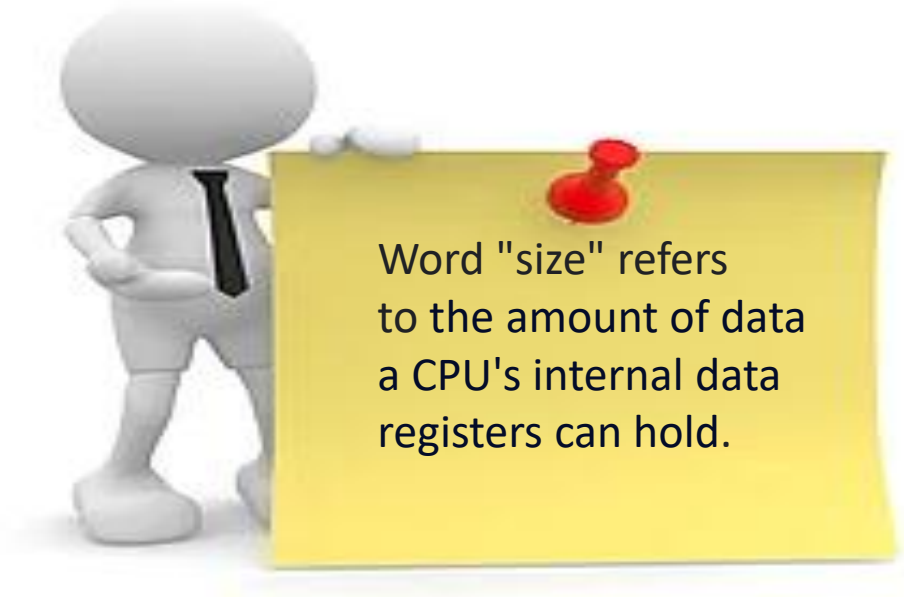
- How many address lines and data lines are there in the following memory unit, given that the cell holds 1 word.

number of words → 64Kw ×8 ← number of bits per word

Data Lines = 8 lines

64Kw = 2^{16} x 2^{10} = 2^{16} words

Thus, Address lines = 16 lines



Check your understanding

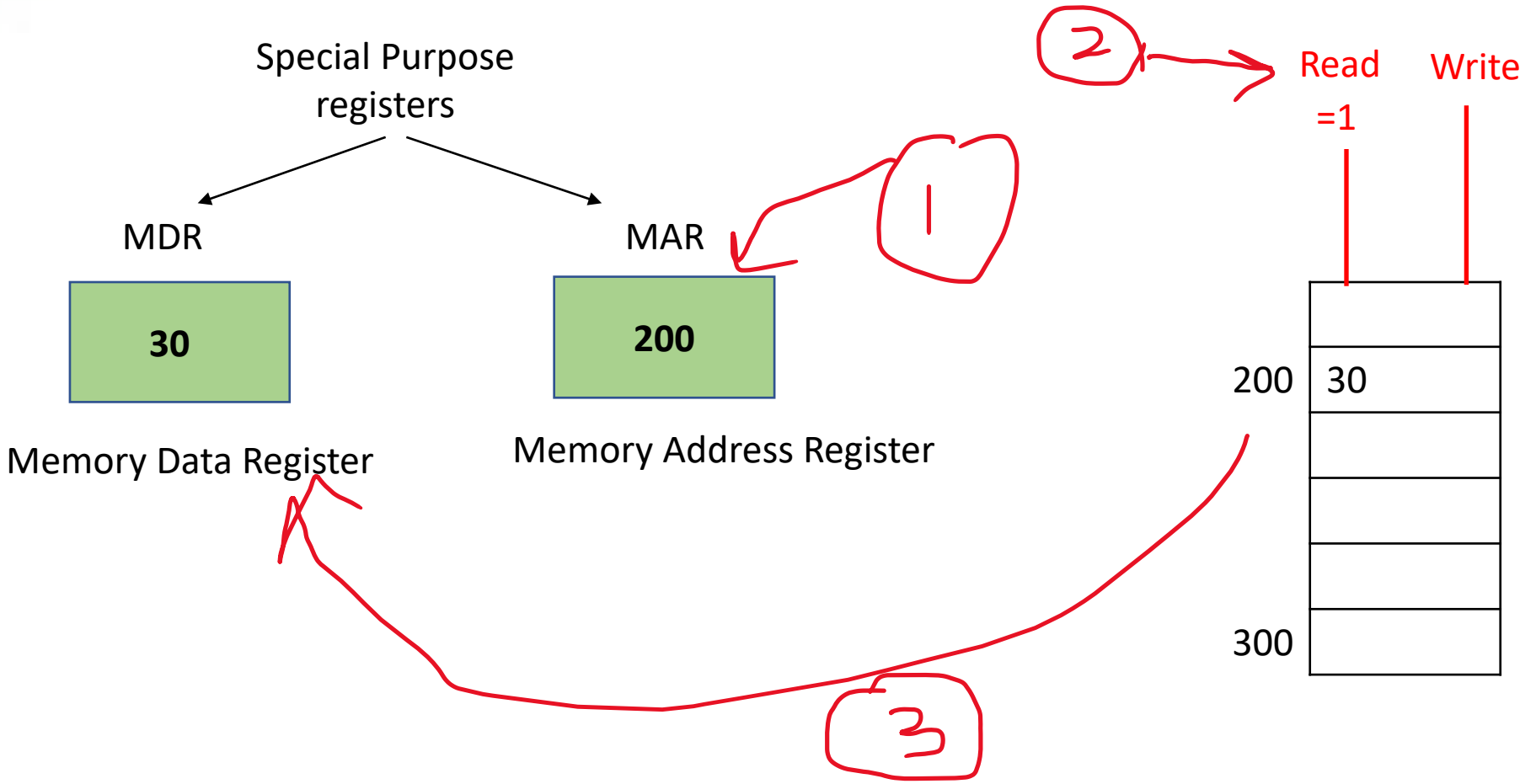
- A byte-addressable memory unit of 128Kw x 8. How many data lines and address lines are needed for this memory unit?

number of words → 128Kw x 8 ← number of bits per word

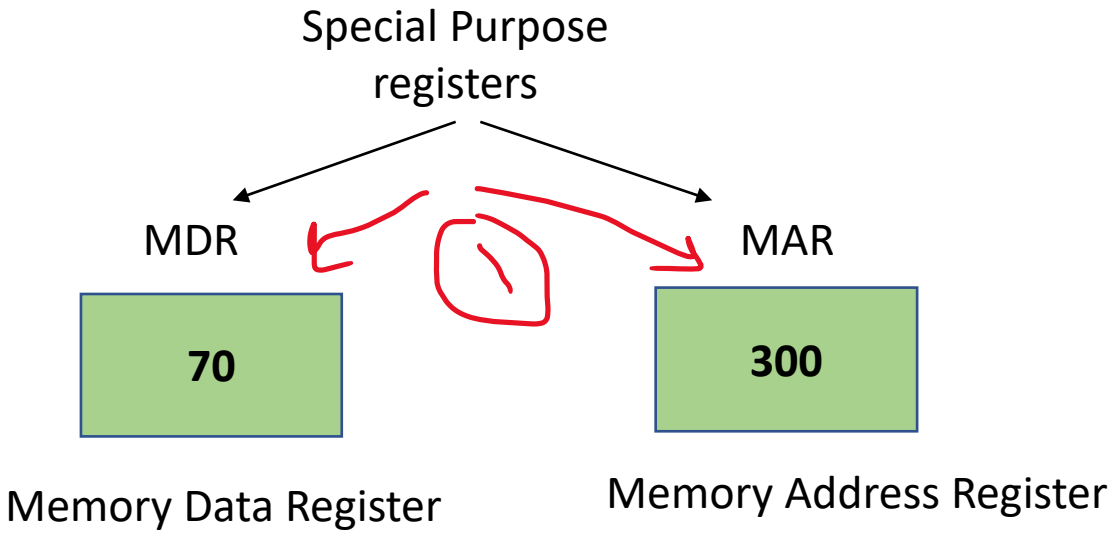
Data lines = 8-bits (lines)
128 Kw = 2^7 x 2^{10} =2^{17}
Thus Address lines = 17 bits (lines)

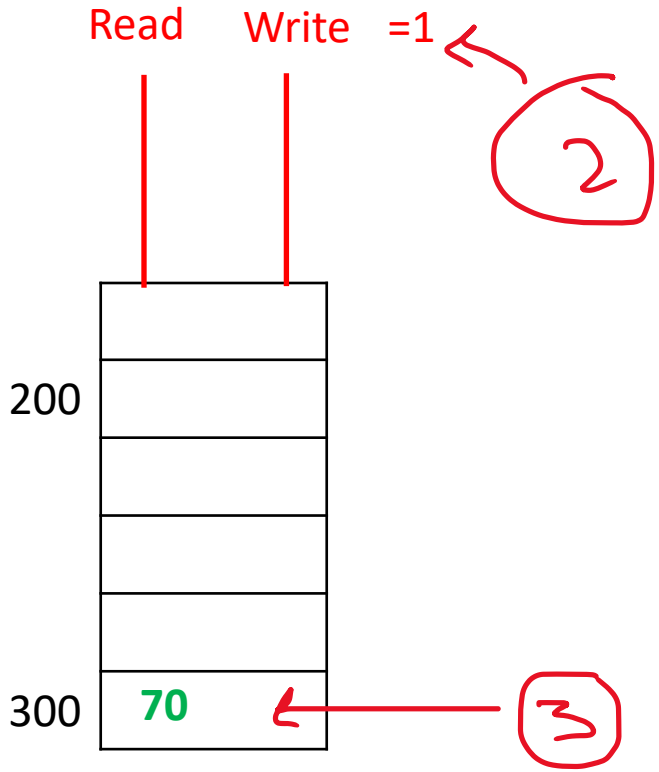
What about a memory unit of 128Kw x 16?

General Scenario for reading from memory Unit



General Scenario for writing to memory Unit





Two Special purpose registers

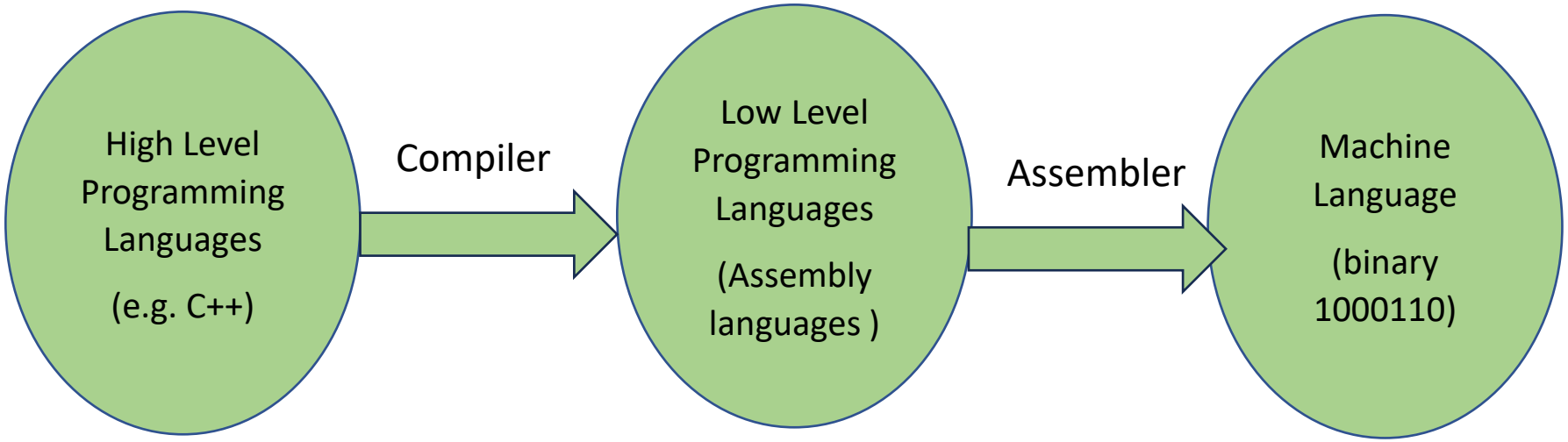
- **Memory data register (MDR)** A register used for holding data transferred from the memory to the central processor, or vice versa. It is also called **Memory Buffer Register(MBR).**

- Memory address register (MAR) is the CPU register that either stores the memory address from which data will be fetched(read) to the CPU registers, or the address to which data will be sent and stored in the memory unit.

More Special purpose registers will be introduced later

Coding in the module

- Coding in the module will be in Assembly Language which is low level programming language.
- Java and C++ are high level programming languages.
- The computer understands only machine language which is “binary”



High-level language program (in C)

```
swap(int v[], int k)
{
   int temp;
   temp = v[k];
   v[k] = v[k+1];
   v[k+1] = temp;
}
```

C compiler

Assembly language program (for MIPS)

```
swap:
     add  $v0, $a1, $a1;     # $v0 = 2 * $a1
     add  $v0, $v0, $v0;     # $v0 = 4 * $a1
     add  $v0, $a0, $v0;     # $v0 = 4 * $a1 + $a0
     lw   $t7, 0($v0);       # $t7 = mem[$v0]
     lw   $t8, 4($v0);       # $t8 = mem[$v0 + 4]
     sw   $t8, 0($v0);       # mem[$v0] = mem[$v0 + 4]
     sw   $t7, 4($v0);       # mem[$v0 + 4] = $t7
     jr   $ra;               # return to caller
```

Assembler

Binary machine language program (for MIPS)

```
00000000101001010001000000100000
00000000010000100001000000100000
00000000100000100001000000100000
10001100010011110000000000000000
10001100010110000000000000000100
10101100010110000000000000000000
10101100010011110000000000000100
00000011111100000000000000001000
```

Next Week's Lab Topics

- Revision on:
  - ❑ Conversion from hexadecimal to binary and vice versa.
  - ❑ Positive and negative values in binary.
  - ❑ Odd and even numbers in binary.
  - ❑ Addition of binary numbers
  - ❑ Subtraction of binary numbers using 2's complement.
  - ❑ Digital logic gates and simple circuits.
  - ❑ Multiplexer.

Thank You