CSE-332

Computer Organization

and

Architecture

Topics to be discussed

Topics to be discussed
Introduction, Organization and Architecture
IAS Structure
Computer Evolution and Performance
Pipelining
Superscalar and Parallel Processing
Multicore Processor and system
Bus Interconnection, PCI
Cache Memory
Semiconductor Main Memory
DRAM
ROM Technology

Topics to be discussed

Topics to be discussed
ALU Design: Adder, Multiplier, Divisor
Booth's Algorithm
Floating point Representation and Arithmetic, Hardware
Instruction Sets: Characteristics and Functions
Addressing Modes and Formats
Instruction-Level Parallelism
Control Unit Operation
Hardware Control Design
Micro-programmed control Design
Supercomputer Architecture
Cloud Computing Architecture

Textbooks

- Computer Organization and Architecture (10th Edition) by William Stallings, Pearson Publisher: 10 edition, January 22, 2015.

- Computer Organization and Design MIPS Edition, Fifth Edition: The Hardware/Software Interface, by David A. Patterson and John L. Hennessy, Publisher: Morgan Kaufmann; 5 edition

- Digital Design and Computer Architecture, 2nd Edition, by David Harris Sarah Harris, Morgan Kaufmann, 24th July 2012.

Tests and Evaluation

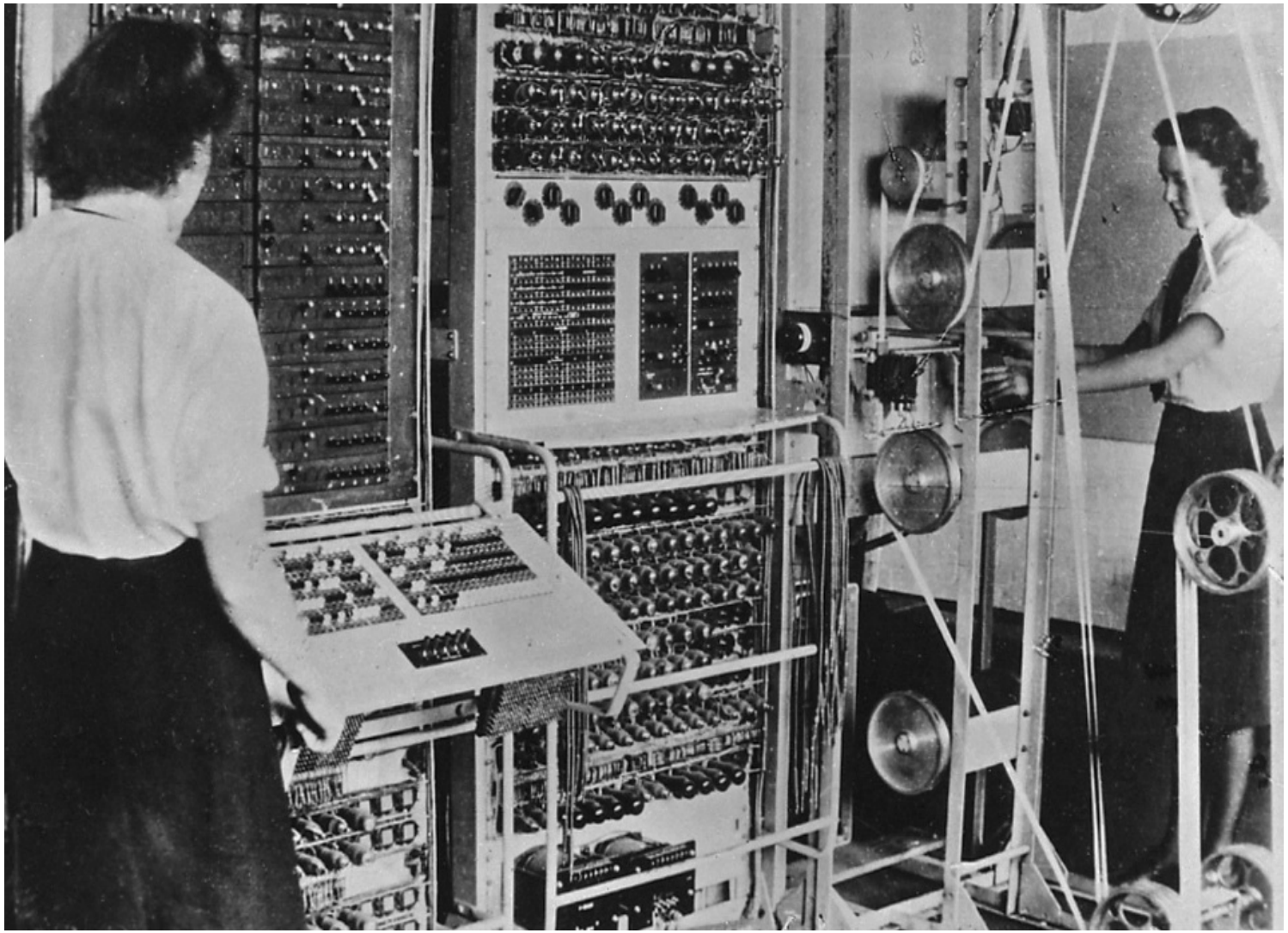
- Assignment: (average) 10%
- Quiz: 20% (your best 4 out of 6 quizzes will be counted)
- Mid Term-1: 15% (after 8th class)
- Mid Term-2: 15% (after 16th class)
- Lab: 10%
- Term Final: 30%
- **Grading:** NSU standard

Evaluation Report as commented in Summer 2020 (please read)

- surprise quizes on that particular day lecture effetced our grade a lot, because we faced network issue ,so couldn't hear lecture but quiz on that same lecture bothered our grade

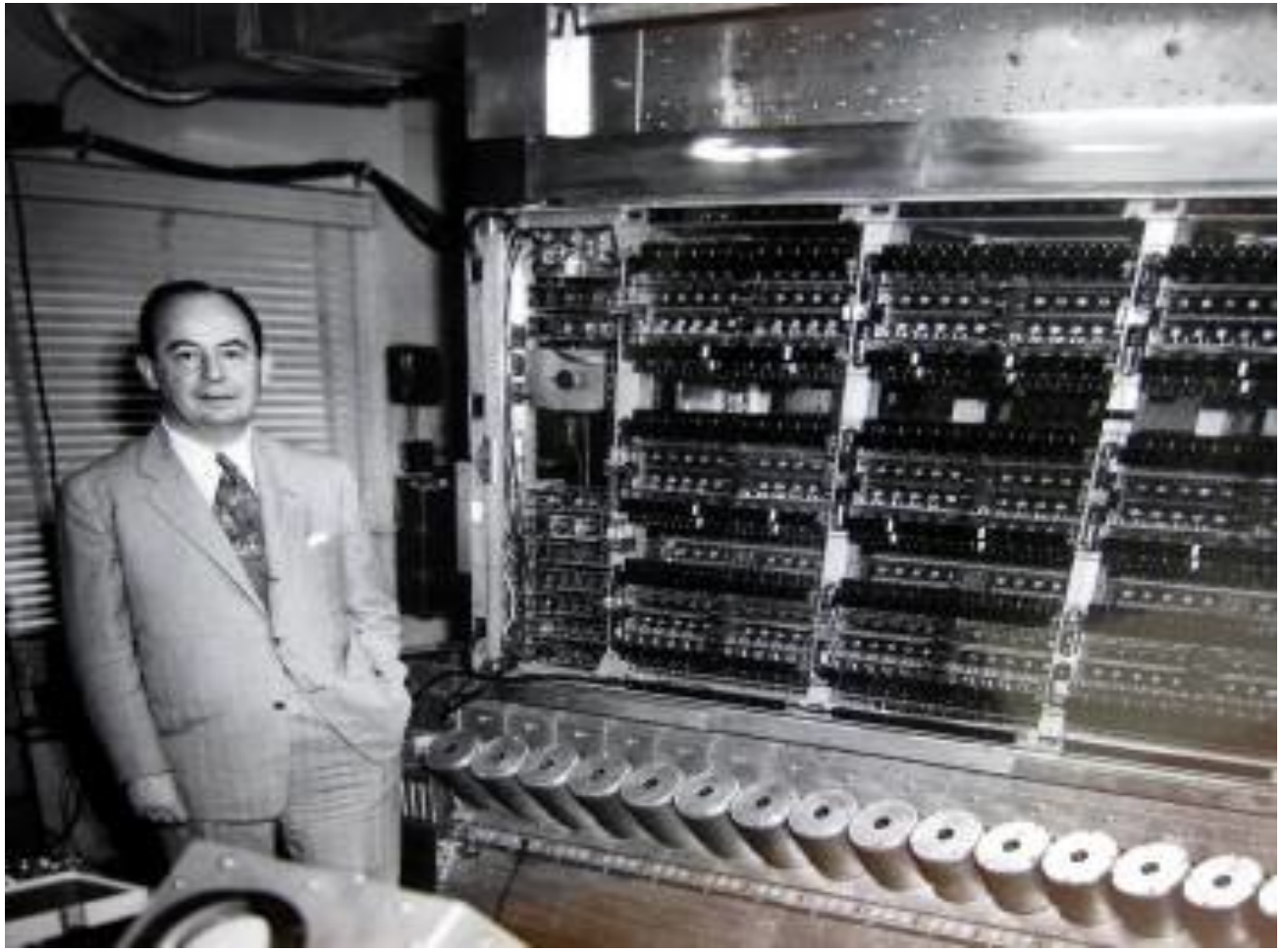
- Who the hell takes MID as surprise? He also took all the quizzes as surprise. Even in the online semester. Totally wrong.

Colossus was an **electronic digital computer**, built **during WWII** (1943–1945) from over 1700 valves (tubes). It was **used to break** the **codes** of the **German** Lorenz SZ-40 **cipher machine** that was **used** by the **German** High Command. **Colossus** is sometimes referred to as the **world's first fixed program, digital, electronic, computer.**



von Neumann (1913 – 1957)

a Hungarian-American mathematician, physicist, computer scientist, and polymath.



proposed STORED-PROGRAM CONCEPT: that a **program** be electronically **stored** in binary-number format in a memory device so that instructions could be modified by the computer as determined by intermediate computational results.

*The **Institute for Advanced Study** (**IAS**) in Princeton, New Jersey, in the United States, is an independent, postdoctoral research center for theoretical research and intellectual inquiry founded in 1930.*

Among its present and past Faculty and Members are 34 Nobel Laureates, 42 of the 60 Fields Medalists (mathematician's Nobel Prize)

The IAS machine was built from 1942 to 1951 under von Neumann's direction introduced the basic architecture of all modern digital computers.

Stored program computer

The **Manchester Baby**, also called the **Small-Scale Experimental Machine** (SSEM), was the first electronic stored-program computer, was built at the **University of Manchester** by Frederic C. Williams, Tom Kilburn, and Geoff Tootill, and ran its first program on 21 June 1948.

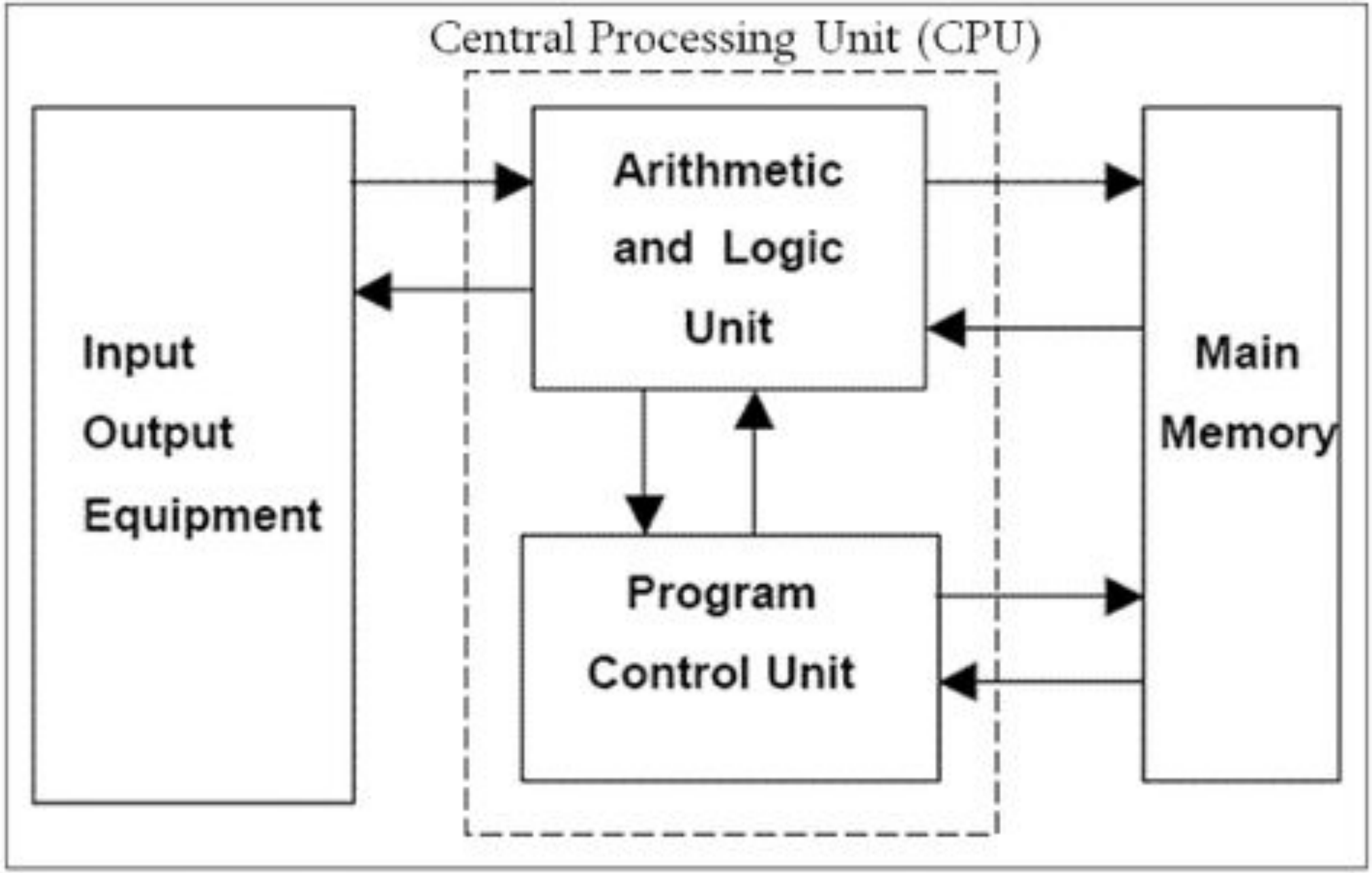
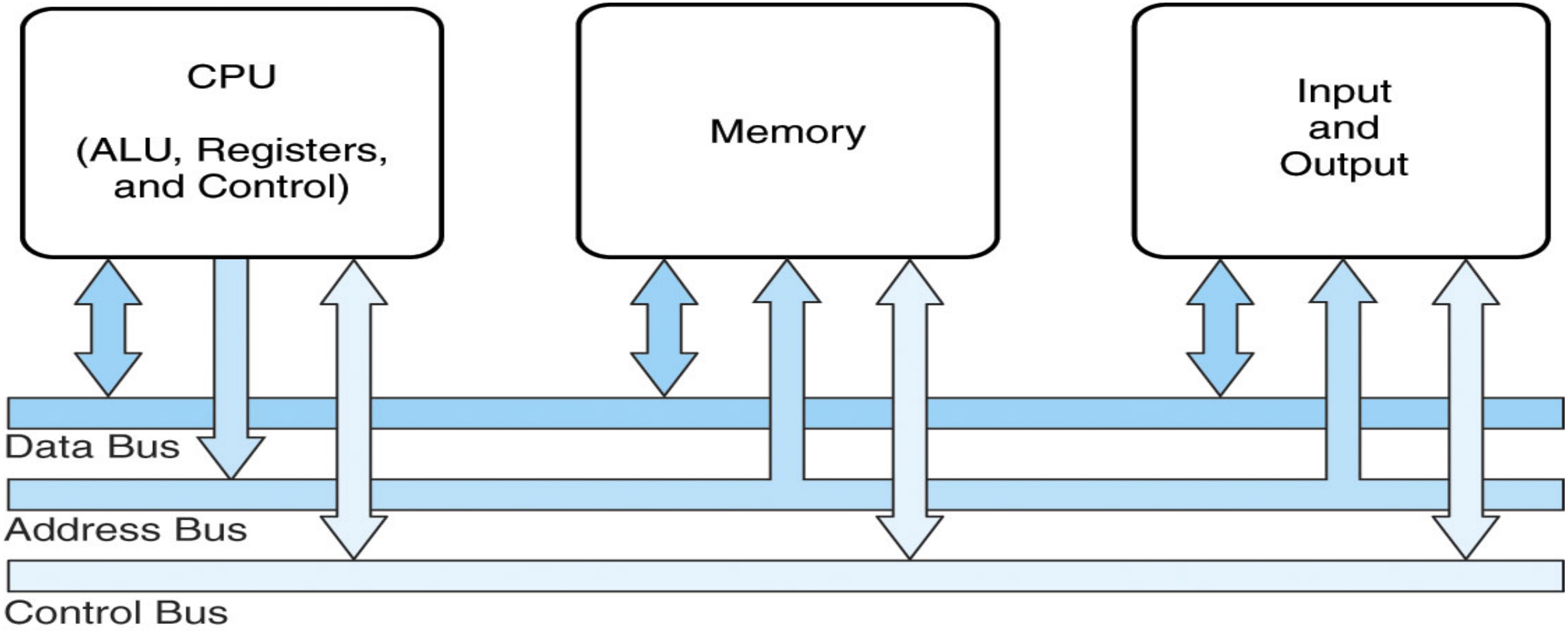
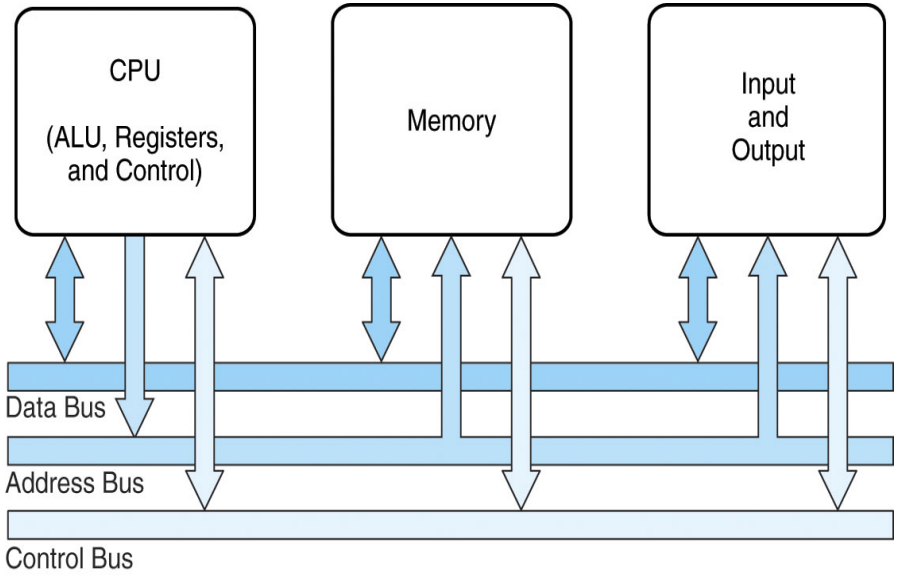


Figure : General structure of Von Neumann Architecture

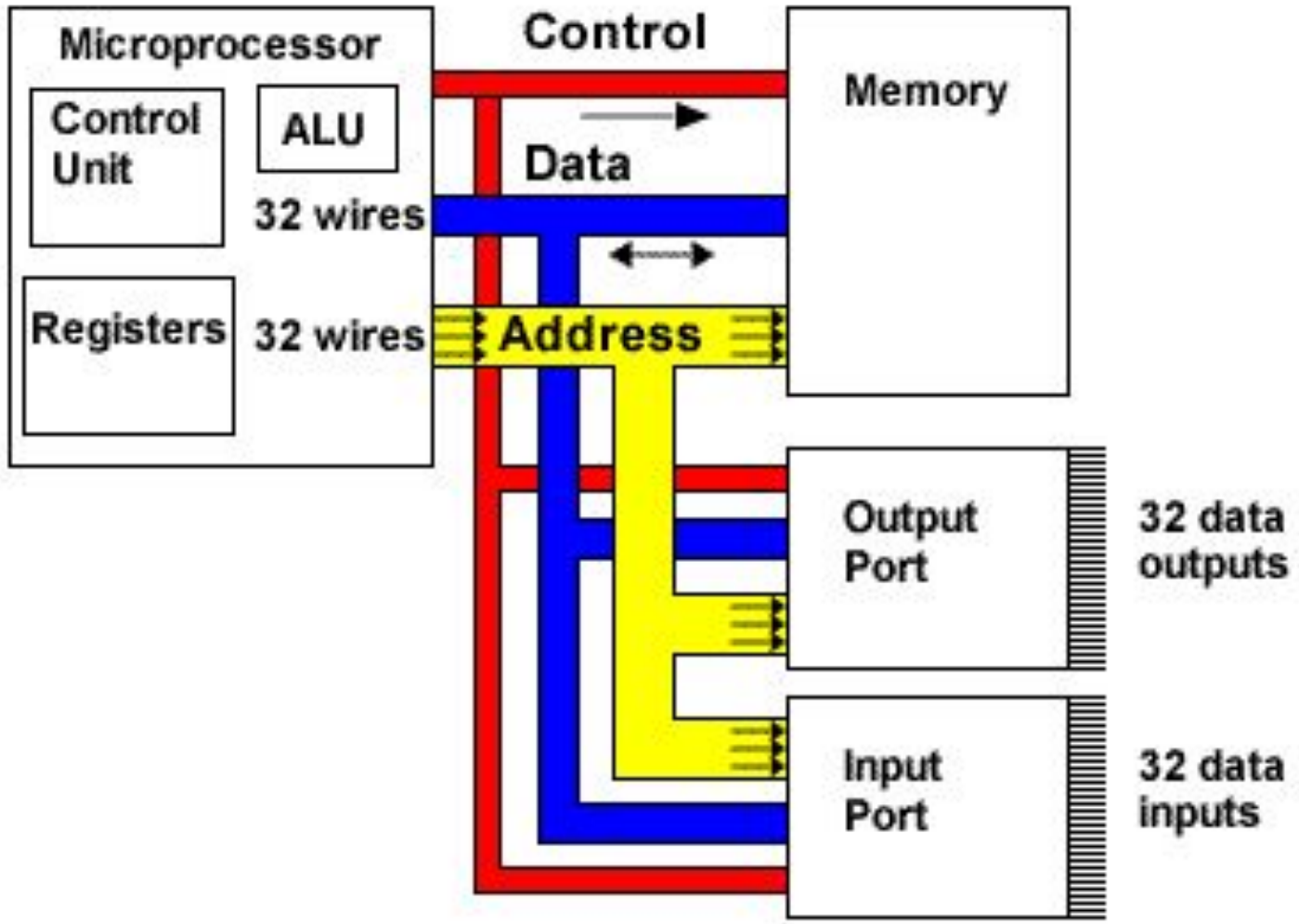
A modern computer is an electronic, digital, general purpose computing machine.

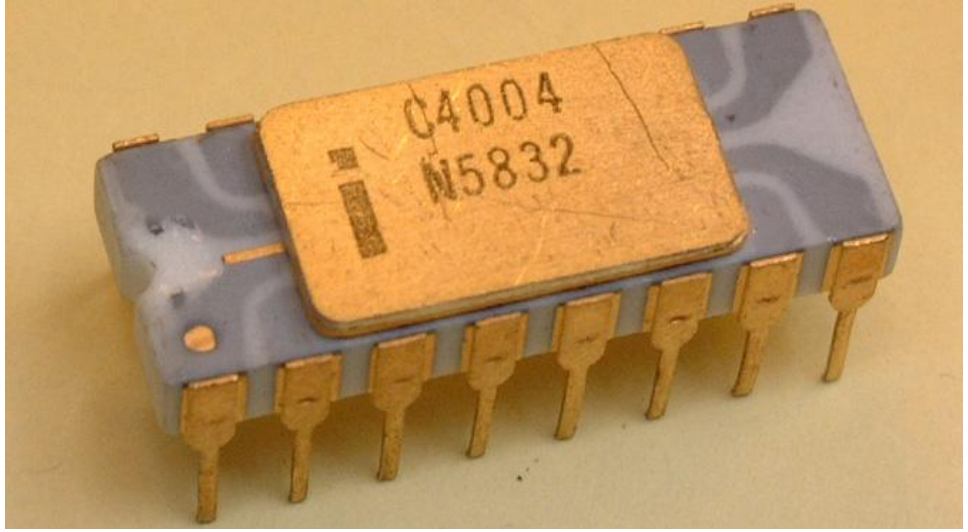


- **CPU**: Central Processing Unit. CPU is considered as the brain of the computer. CPU performs all types of data processing **operations**. It stores data, intermediate results, and instructions (program). It controls the operation of all parts of the computer.
- **Memory**: Holds program and data in binary form
- **Input and Output devices**
- **Bus system**: A system bus is a single computer bus that connects the major components of a computer system.



- Three types of bus are used.
  - Address bus - carries memory addresses from the processor to other components such as primary storage and input/output devices.
  - Data bus - carries the data between the processor and other components.
  - Control bus - carries control signals from the processor to other components.

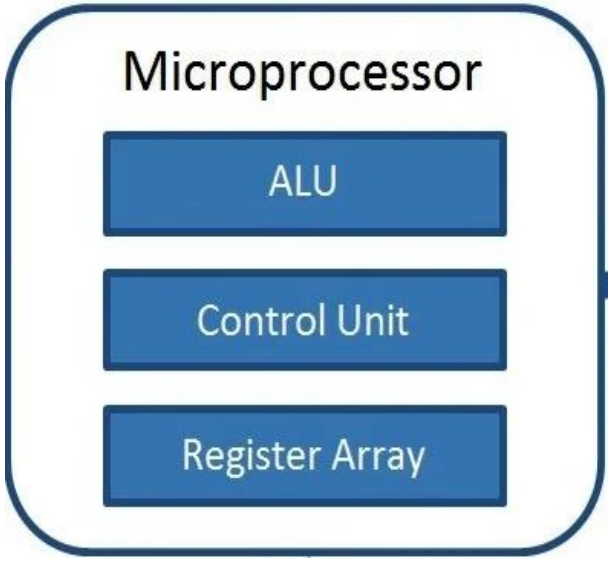




Intel 4004, the first CPU, is 40 years old today (November 15, 1971)

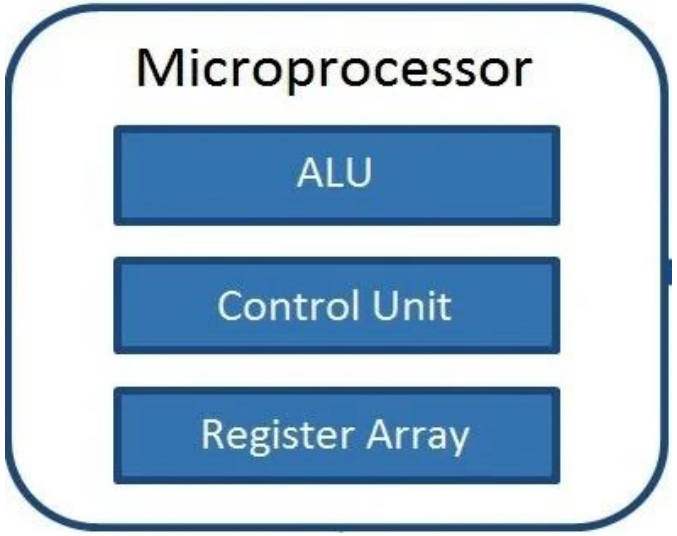
A **4-bit,** 16-pin microprocessor that operated at 740KHz — and at roughly eight clock cycles per instruction cycle (fetch, decode, execute), that means the chip was capable of executing up to 92,600 instructions per second. We can't find the original list price, but one source indicates that it cost around $5 to manufacture. It had 2,300 transistors and a feature size of 10 micron (a micron is one-millionth of a meter)

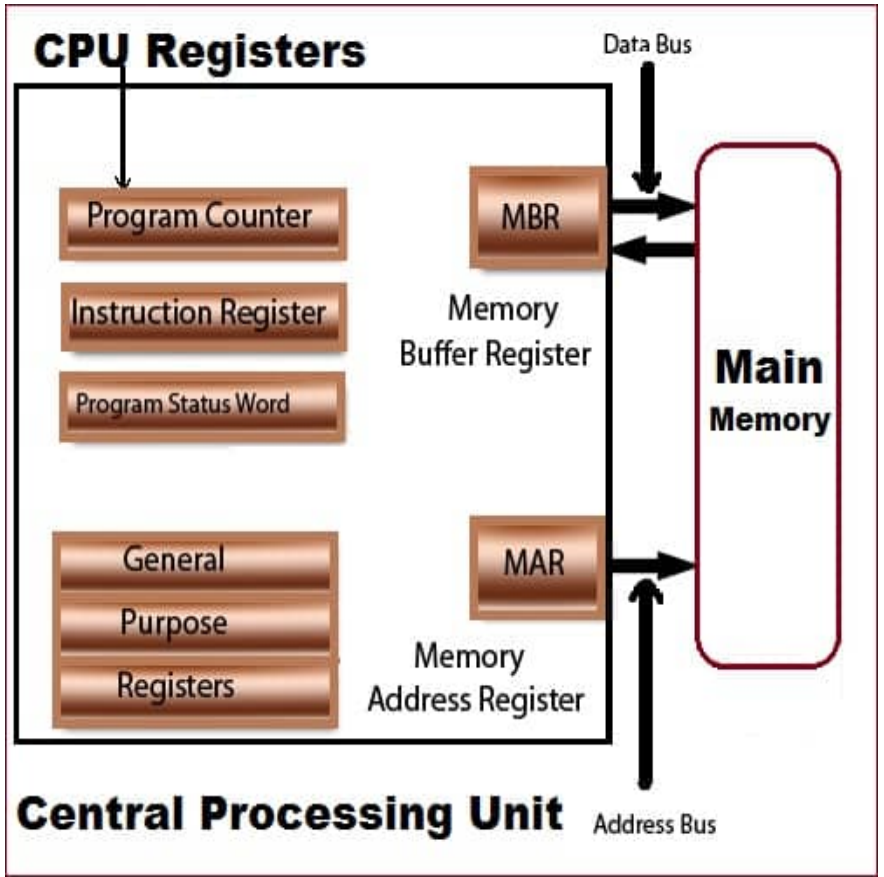
A family of 64-bit x86 CPUs with up to 18 cores from Intel (introduced in 2017). Designed for high-performance computing and gaming, the 3.3 GHz i9 chip can be overclocked to 4.5 GHz.
It contains around 7 billion transistors using 14 nm technology. CPU can carry out 3000,000000 (three thousand million) instructions per second!



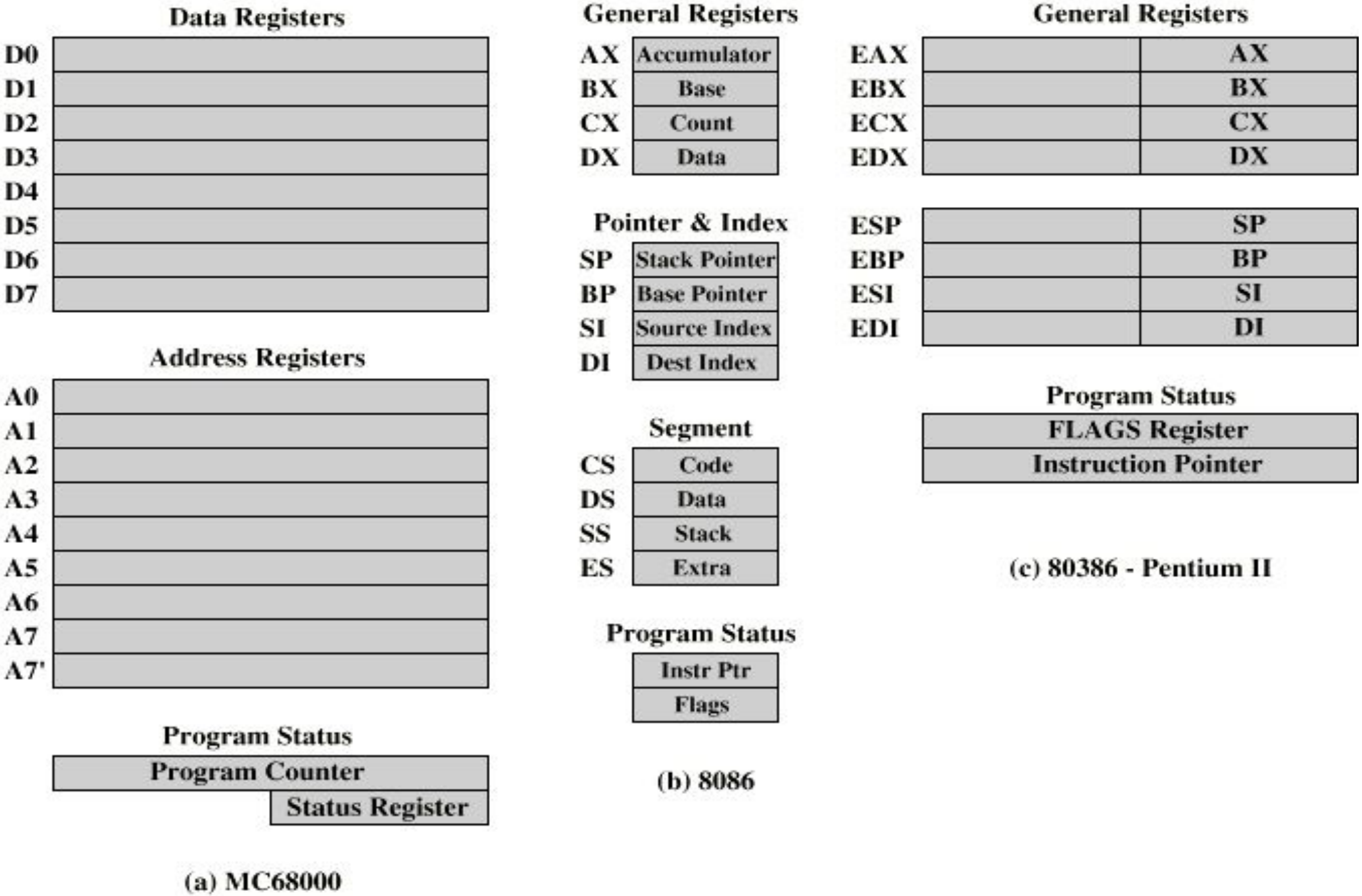
Major functional units of a CPU

CPU

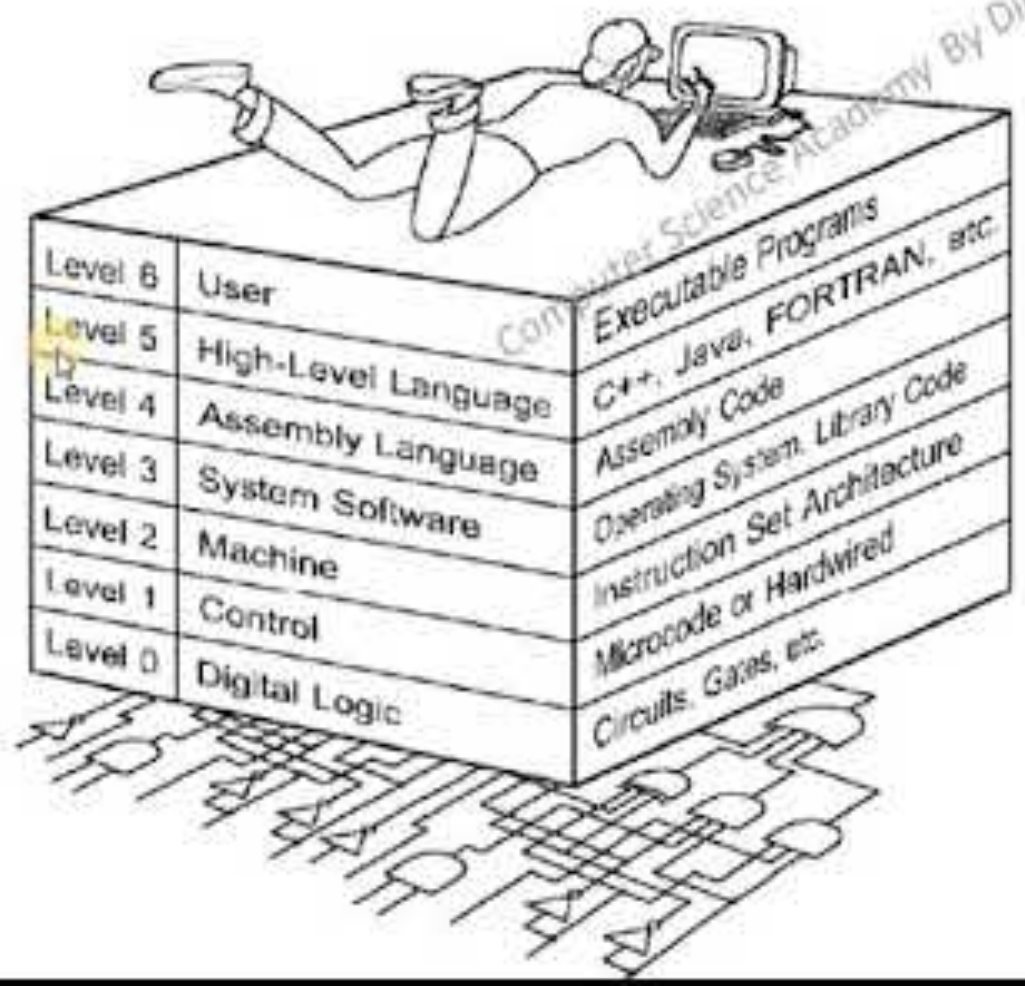




- ALU: an arithmetic logic unit (*ALU*) is a combinational digital circuit that performs arithmetic and logical operations on binary numbers.
- The **control unit** of the central processing **unit** regulates and integrates the operations of the **computer**. It selects and retrieves instructions from the main memory in proper sequence and interprets them so as to activate the other functional elements of the system at the appropriate moment.
- Register Array: **Registers** are small amounts of high-speed electronic storage contained within the **CPU**. They are used by the **processor** to store small amounts of data that are needed during processing, such as: the address of the next instruction to be executed. the current instruction being decoded. **Register array** consists of **registers** identified by letters like B, C, D, E, H, L and accumulator.


Data Registers
D0
D1
D2
D3
D4
D5
D6
D7
Address Registers
A0
A1
A2
A3
A4
A5
A6
A7
A7'
Program Status
Program Counter
Status Register
(a) MC68000
General Registers
AX
Accumulator
BX
Base
CX
Count
DX
Data
Pointer & Index
SP
Stack Pointer
BP
Base Pointer
SI
Source Index
DI
Dest Index
Segment
CS
Code
DS
Data
SS
Stack
ES
Extra
Program Status
Instr Ptr
Flags
(b) 8086
General Registers
EAX
AX
EBX
BX
ECX
CX
EDX
DX
ESP
SP
EBP
BP
ESI
SI
EDI
DI
Program Status
FLAGS Register
Instruction Pointer
(c) 80386 - Pentium II

Computer Level Hierarchy



Level 6	User	Executable Programs
Level 5	High Level Language	C++ , Java
Level 4	Assembly Language	Assembly Code
Level 3	System Software	Operating System
Level 2	Machine	Instruction Set Architecture
Level 1	Control	Microcode
Level 0	Digital Logic	Circuits , Gates

Computer Architecture: visible to programmer

How do I design a computer?

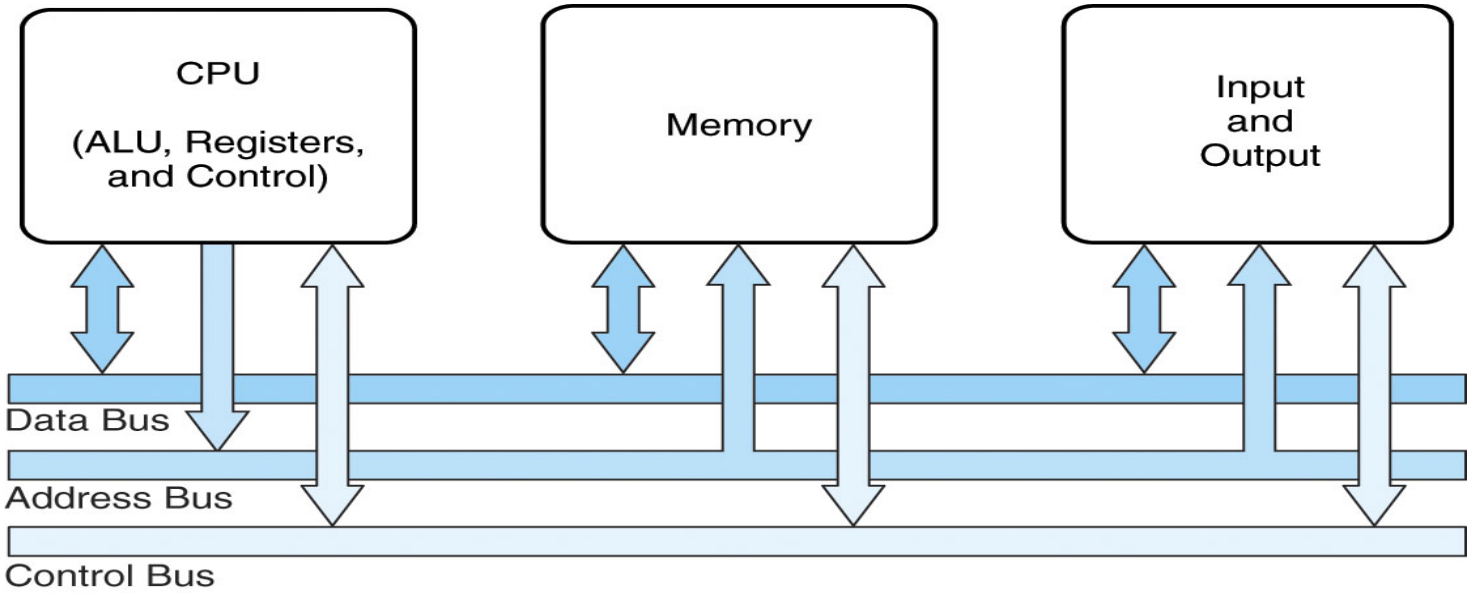
- Focuses on the **structure:** the way in which the components are interrelated
- Computer architecture includes many elements such as
  - Instruction sets and formats
  - Operation codes
  - Data types & Data Representation
  - Register architecture with CPU
  - Addressing modes
  - Input/Output Mechanism
- The architecture of a system directly affects the logical execution of programs.
- The computer architecture for a given machine is the combination of its hardware components plus its instruction set architecture (ISA).
- The ISA is the interface between all the software that runs on the machine and the hardware

Computer organization: *How does a computer work?*

- **Computer Organization** refers to the level of abstraction above the digital logic level, but below the operating system level.

- Encompasses all physical aspects of computer systems
  - Hardware details of the system
  - Circuit design
  - Control signals
  - Memory types & Technologies

A modern computer is an electronic, digital, general purpose computing machine.



At the most basic level, a computer consists of 3 pieces

- A processor to interpret and execute programs
- A memory (Includes Cache, RAM, ROM) to **store both data and program instructions**
- A mechanism for transferring data to and from the outside world.

I/O to communicate between computer and the world

Bus to move info from one computer component to another

Levels of Program code

Program: Display the sum of A,B & C.

C++: cout << (A+B+C)

Assembly Language:
mov eax, A
add eax, B
add eax, C
call WriteInt

Machine Language(Intel):
A1 00000000
F7 25 00000004
03 05 00000008
E8 00500000

Figure 2.1 Translation of L1 to L0

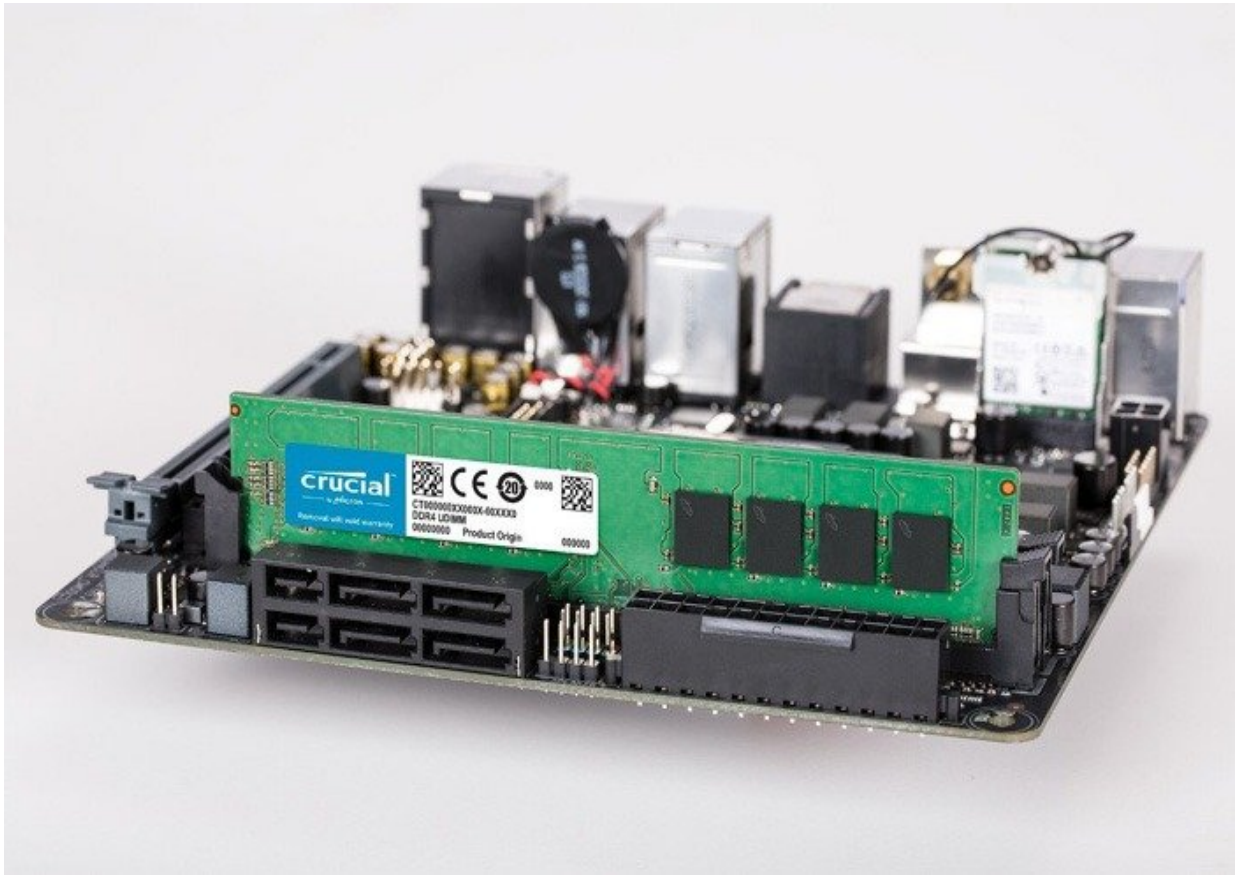
How the machine code is formed?

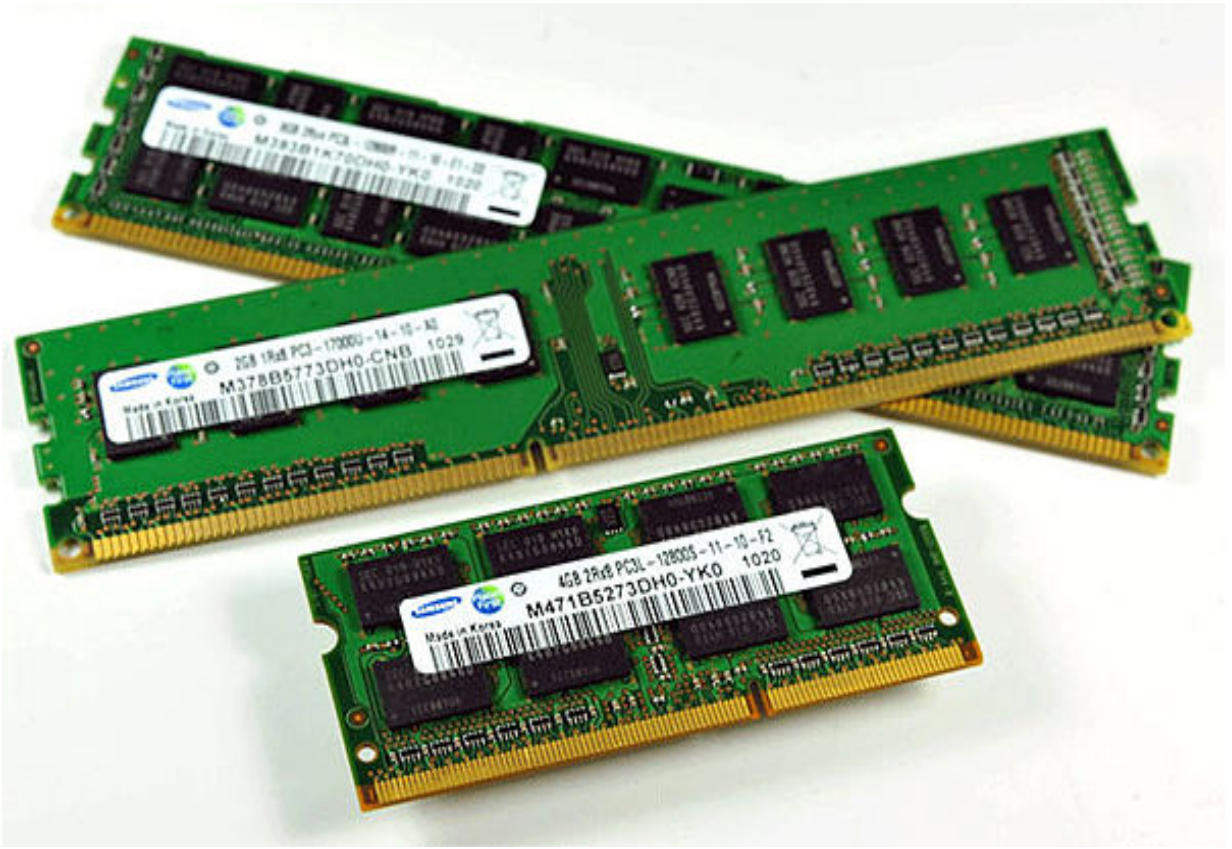
Machine code in binary

0101000001111111
1100000111100000
0101010100001111
1110000000111111
0101000001111111
1100000111100000
0101010100001111
1110000000111111

Stored in RAM

Main Memory: Semiconductor ICs
RAM(Random Access Memory)



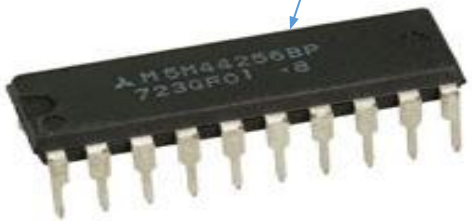


- Memory is an array of storage, each having capacity of 8 bits or so and holds program and data in binary format

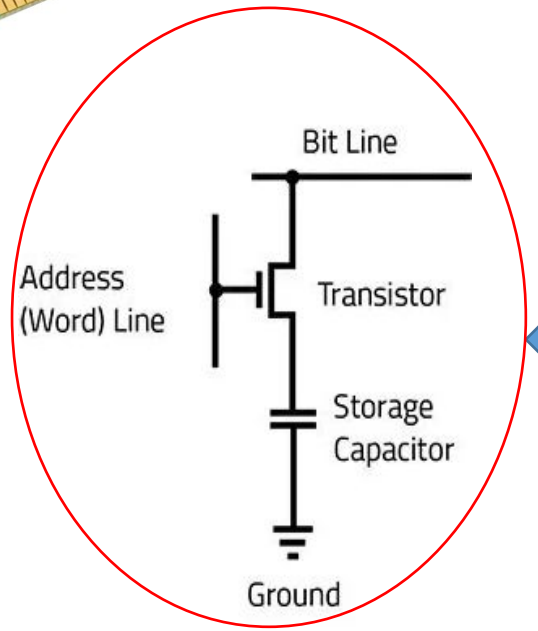
User Program & Data

0101000001111111
1100000111100000
0101010100001111
1110000000111111
0101000001111111
1100000111100000
0101010100001111
1110000000111111

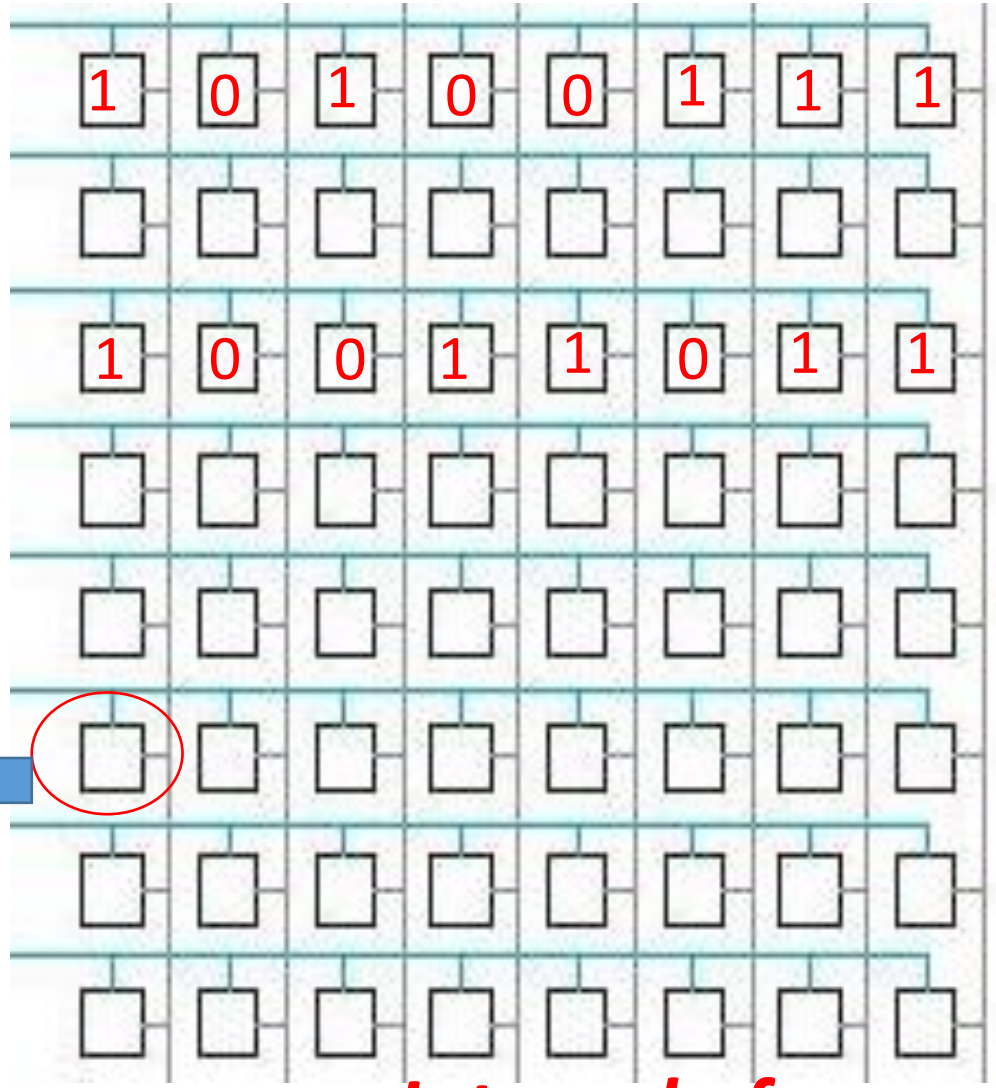
Module



IC



Single Memory Cell

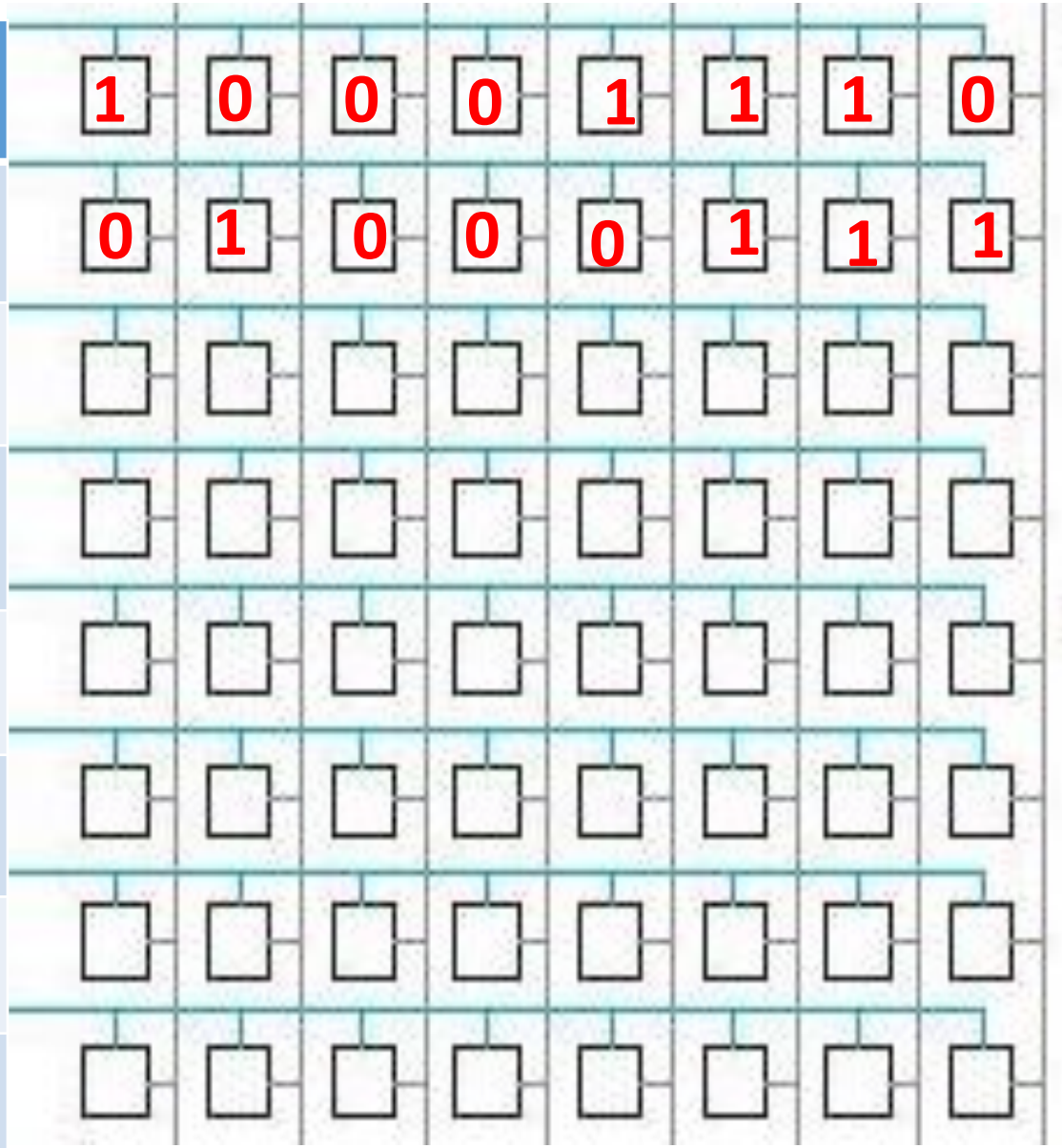


Internal of Module/IC

Each row is uniquely identified by a number/code, starting from ‘0’, called Address, usually represented in Hexadecimal form and used in Assembly language programming

Address in HEX	Address in Binary			
0H	0	0	0	0
1H	0	0	0	1
2H	0	0	1	0
3H	0	0	1	1
7H	0	1	1	1

Contents (Machine code & data)

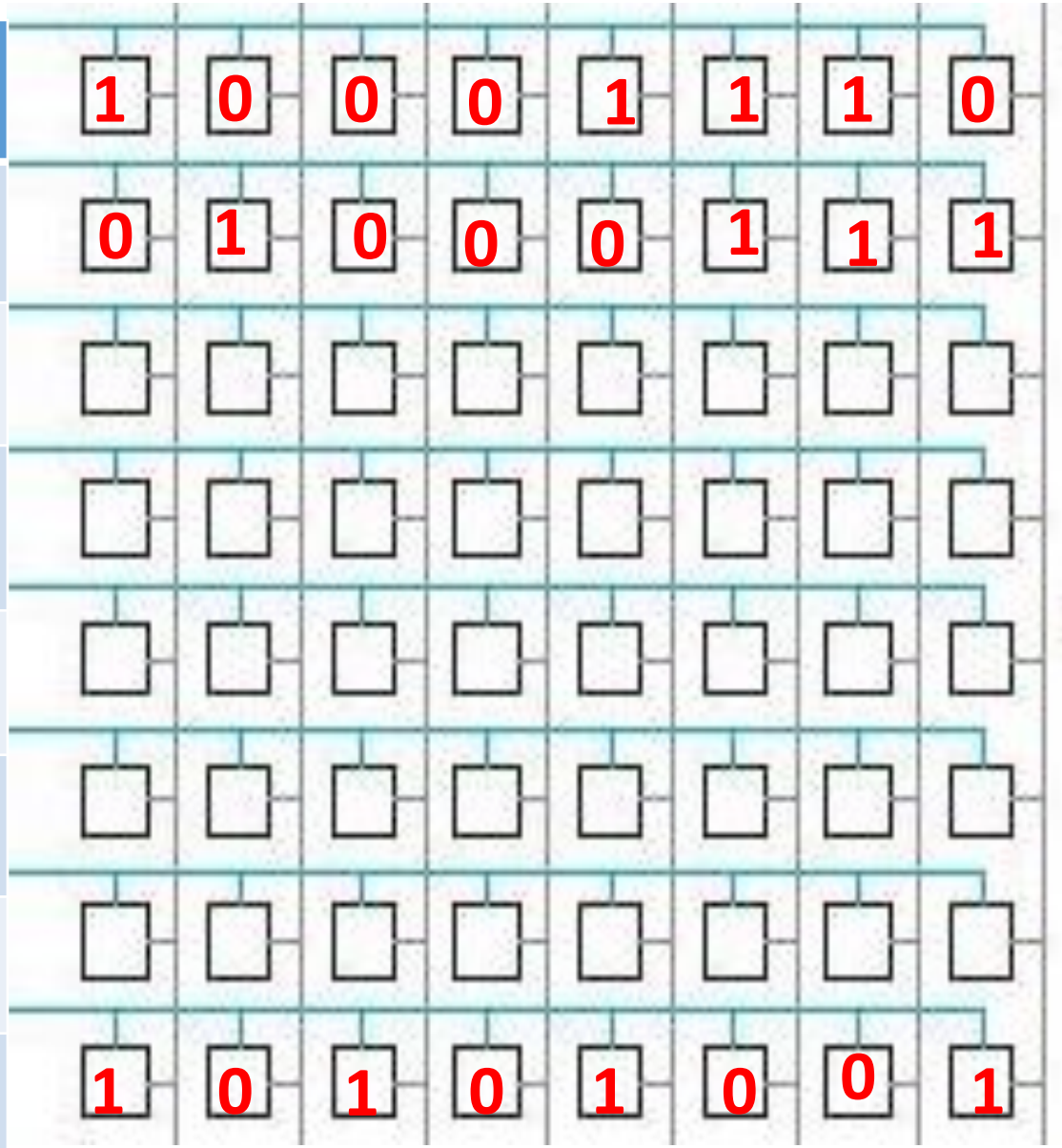


If each location of memory contains 8 bits or 1 byte, then it is called

Byte-Addressable memory

Address in HEX	Address in Binary			
0H	0	0	0	0
1H	0	0	0	1
2H	0	0	1	0
3H	0	0	1	1
7H	0	1	1	1

Contents (Machine code or data)



Capacity of Memory

- Example: 1KB: Approximately 1 Thousand (exact value 1024) locations each having capacity of 8 bits/1Byte
- Address starts at 0 and ends at 1 less than 1 Thousand, actually encoded in BINARY
- In Binary, first address requires 1 bit (0) and final addressable location requires 10 bits (all 1's: 11…11), since $\mathbf{2^{10} = 1K}$
- For ease of Decoder design, uniform address format is used for all the locations; Maximum number of bits!
- For convenience/ease of representation/programming/discussion, Hexadecimal number system is used to represent Memory address

Address of Memory (for 1KB)

Address (20 bits in binary)	Content (8 bits)
1111111111B	11001100 (machine code/data)
…	
0000000000B	00110101(machine code/data)

Address (3 digits in hexadecimal)	Content(8 bits)
3FFH	11001100 (machine code/data)
000H	00110101(machine code/data)

Capacity of Memory

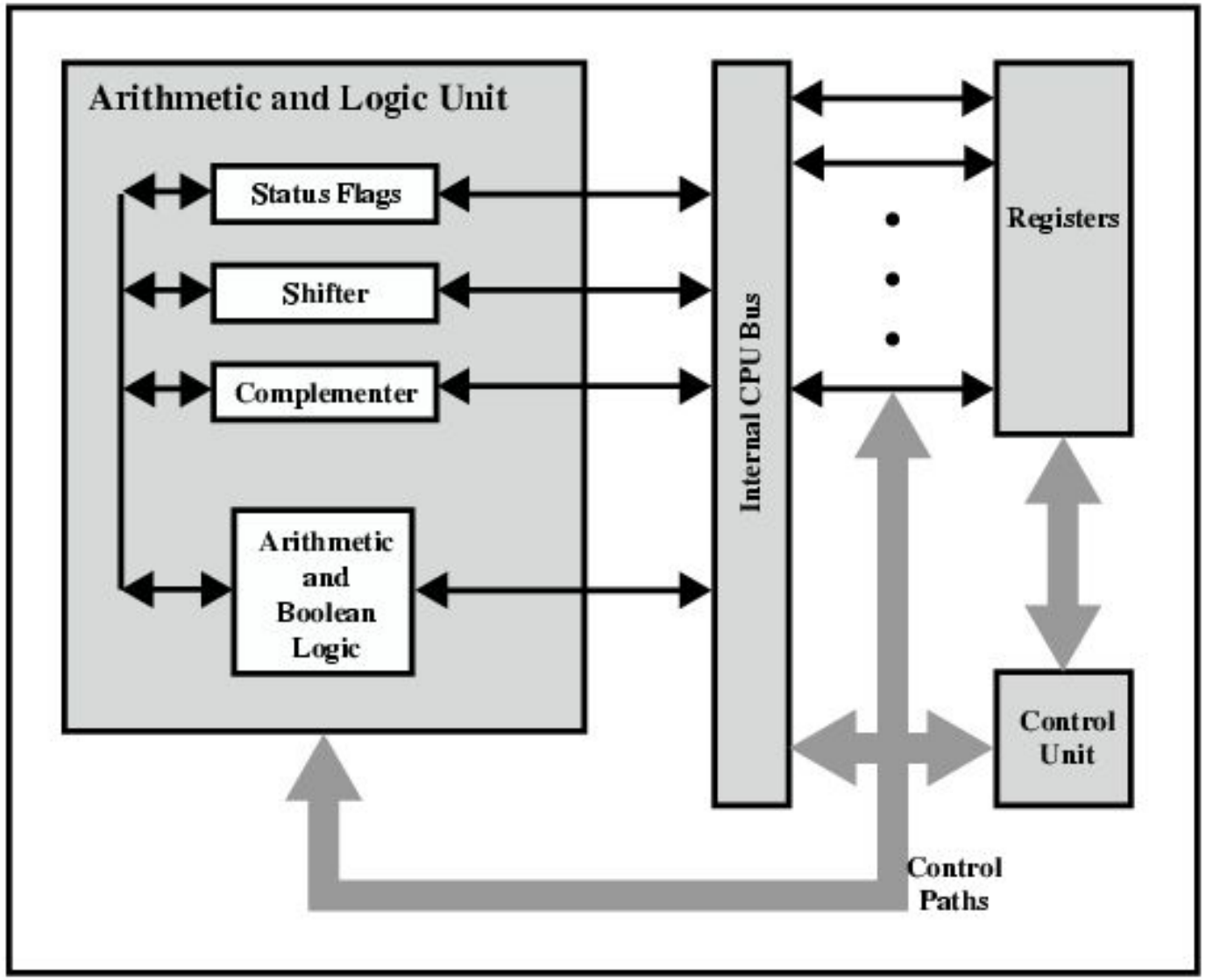
- Example: 1MB: Approximately 1 Million (exact value 1024 x 1024) locations each having capacity of 1Byte
- Address starts at 0 and ends at 1 less than 1 Million, actually encoded in BINARY
- In Binary, first address requires 1 bit (0) and final addressable location requires 20 bits (all 1's: 11…11), since $\mathbf{2^{20} = 1M}$
- For ease of Decoder design, uniform address format is used for all the locations; Maximum number of bits!
- For convenience/ease of representation/programming/discussion, Hexadecimal number system is used to represent Memory address

Address of Memory (for 1MB)

Address (20 bits in binary)	Content (8 bits)
11111111111111111111B	11001100 (machine code/data)
…	
00000000000000000000B	00110101(machine code/data)

Address (5 digits in hexadecimal)	Content(8 bits)
FFFFFH	11001100 (machine code/data)
00000H	00110101(machine code/data)

Internal Architecture of CPU



- **Register** is an electronic storage inside the CPU
- Used to hold data/address RAM where program is stored
- Number & size of registers vary from CPU to CPU
- Size means how many bit can be stored in a register

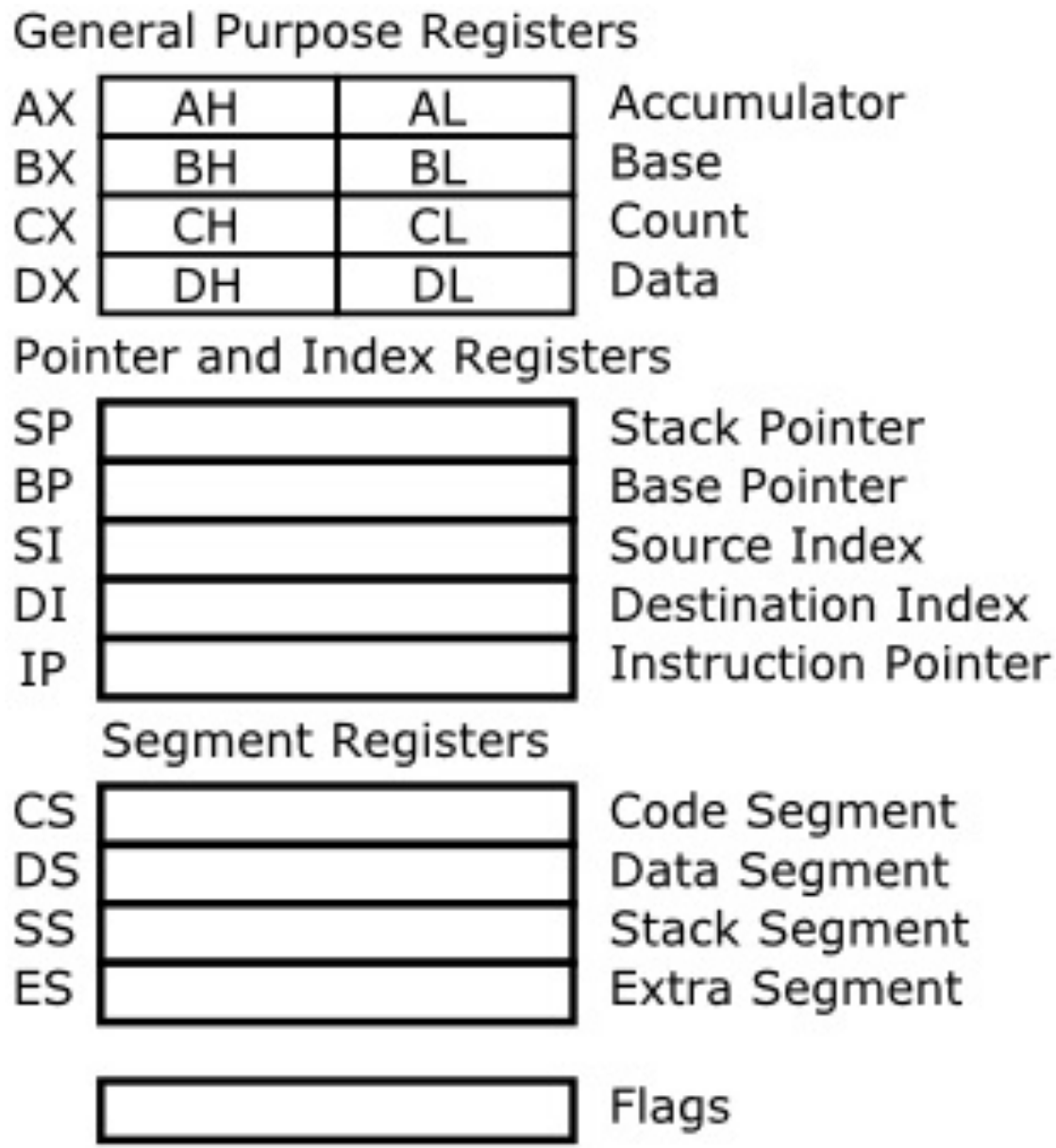
Registers

- Size: 4 bits/8 bits/16 bits/32 bits/64 bits/128 bits…
- In modern processors: Register size and number both are increasing
- Register types: General purpose & special purpose
- Must know register architecture for Assembly Language Programming
- Since registers are few in numbers, they are addressed by Names in Assembly language Programs
- Naming depends on processor series, usually upper case one/two/three letters are used. Example: A, AX, EAX etc

Core i7(2009)

64-bit register	Lower 32 bits	Lower 16 bits	Lower 8 bits
rax	eax	ax	al
rbx	ebx	bx	bl
rcx	ecx	cx	cl
rdx	edx	dx	dl
rsi	esi	si	sil
rdi	edi	di	dil
rbp	ebp	bp	bpl
rsp	esp	sp	spl
r8	r8d	r8w	r8b
r9	r9d	r9w	r9b
r10	r10d	r10w	r10b
r11	r11d	r11w	r11b
r12	r12d	r12w	r12b
r13	r13d	r13w	r13b
r14	r14d	r14w	r14b
r15	r15d	r15w	r15b

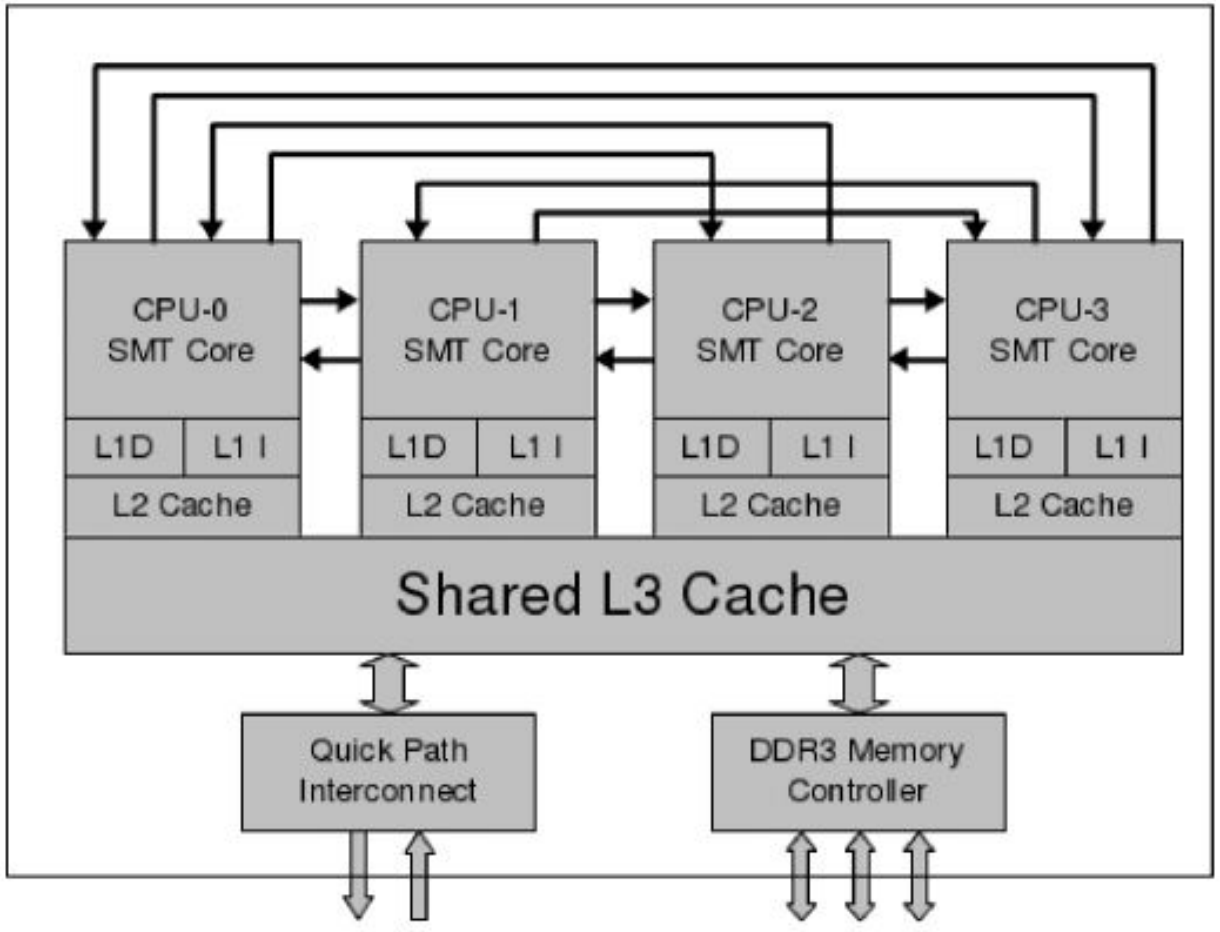
8086 (1978)



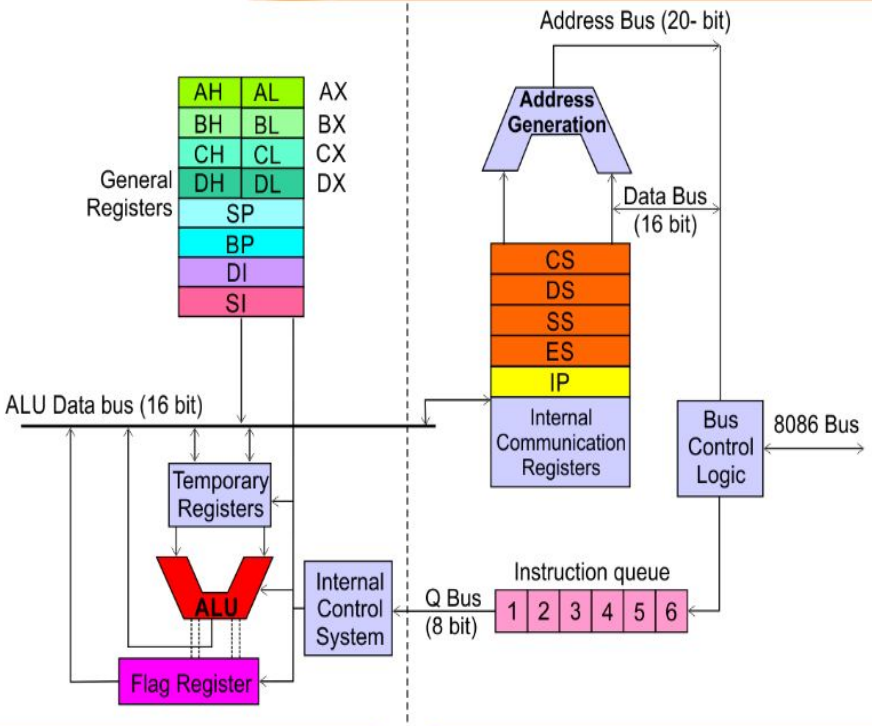
Types of Registers

- General purpose: Holds data to be used in ALU. Also holds partial results or final results generated by ALU.
- Special Purpose:
  - Holds address of main memory locations where instructions are loaded by Operating system.
  - Holds address of main memory locations where data are stored
  - Holds address of main memory locations used for special purposes
  - Holds machine code of Instructions
  - Holds status of CPU

Intel Core i7 Processor



Intel 8086



Execution Unit (EU)

EU executes instructions that have already been fetched by the BIU.

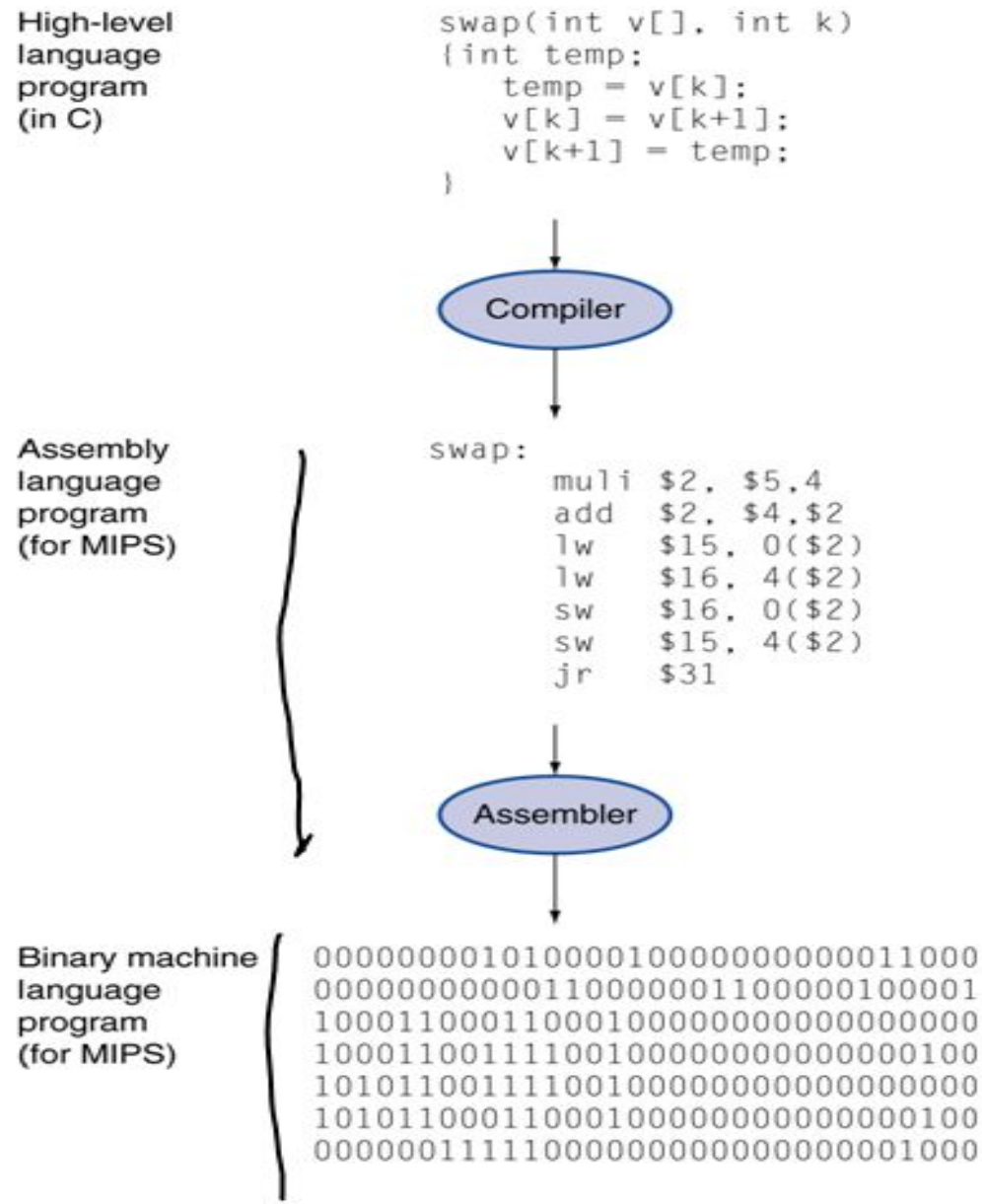
BIU and EU functions separately.

Bus Interface Unit (BIU)

BIU fetches instructions, reads data from memory and I/O ports, writes data to memory and I/ O ports.

Levels of Program Code

- **High-level language**
  - Level of abstraction closer to problem domain
  - Provides productivity and portability
- **Assembly language**
  - Textual representation of instructions
- **Hardware representation**
  - Binary digits (bits)
  - Encoded instructions and data



Levels of Program code

Program: Display the sum of A,B & C.

C++: cout << (A+B+C)

Assembly Language:
mov eax, A
add eax, B
add eax, C
call WriteInt

Machine Language(Intel):
A1 00000000
F7 25 00000004
03 05 00000008
E8 00500000

Figure 2.1 Translation of L1 to L0

How the instructions are designed?

How the machine codes are formed?

Machine code in binary

0101000001111111
1100000111100000
0101010100001111
1110000000111111
0101000001111111
1100000111100000
0101010100001111
1110000000111111

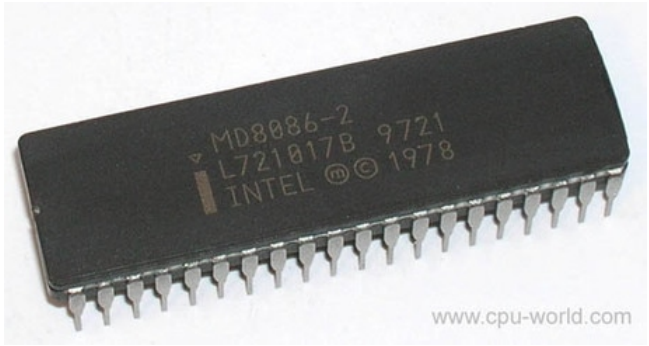
Stored in RAM

- Two key benefits of assembly language programming
  - It takes up less memory
  - It executes much faster

- Assembly language is good for controlling hardware devices

Program and Instructions

- The sequence of commands used to tell a microcomputer what to do is called a program,
- Each command in a program is called an instruction
- 8086 understands and performs operations for 117 basic instructions



IBM 5150, 8086 Processor 4,77 MHz, 640KB RAM, 10MB Hard Disk, 5,25" Floppy (August 12, 1981)

Instruction—An operation performed by the CPU and assigned a specific number
I**nstruction set**—The list of CPU instructions for the operations

Complete 8086 instruction set

Quick reference:

	CMPSB				MOV		
AAA	CMPSW	JAE	JNBE	JPO	MOVSB	RCR	SCASB
AAD	CWD	JB	JNC	JS	MOVSW	REP	SCASW
AAM	DAA	JBE	JNE	JZ	MUL	REPE	SHL
AAS	DAS	JC	JNG	LAHF	NEG	REPNE	SHR
ADC	DEC	JCXZ	JNGE	LDS	NOP	REPNZ	STC
ADD	DIV	JE	JNL	LEA	NOT	REPZ	STD
AND	HLT	JG	JNLE	LES	OR	RET	STI
CALL	IDIV	JGE	JNO	LODSB	OUT	RETF	STOSB
CBW	IMUL	JL	JNP	LODSW	POP	ROL	STOSW
CLC	IN	JLE	JNS	LOOP	POPA	ROR	SUB
CLD	INC	JMP	JNZ	LOOPE	POPF	SAHF	TEST
CLI	INT	JNA	JO	LOOPNE	PUSH	SAL	XCHG
CMC	INTO	JNAE	JP	LOOPNZ	PUSHA	SAR	XLATB
CMP	IRET	JNB	JPE	LOOPZ	PUSHF	SBB	XOR
	JA				RCL		

The 8086 microprocessor supports 8 types of instructions –

- Data Transfer Instructions
- Arithmetic Instructions
- Bit Manipulation Instructions
- String Instructions
- Program Execution Transfer Instructions (Branch & Loop Instructions)
- Processor Control Instructions
- Iteration Control Instructions
- Interrupt Instructions

Instruction format

Example: STD ; clear direction flag bit

Example: NOT AX ; complement content of AX

Example: MOV AX, BX; copy the contents of BX into AX

Instruction format

- Instructions having only opcodes, everything is indicated within.
- Some instructions may require only one operand
- To reduce the size of instructions, only one operand is indicated in instructions whereas a default register/memory location is used for other operand.
- Example: MUL BX ; AX holds other operand
- PUSH AX; contents of AX is copied into memory location within Stack segment.

Machine Codes

- An instruction can be coded with 1 to 6 bytes
- Byte 1 contains three kinds of information
  - Opcode field (6 bits) specifies the operation (add, subtract, move)
  - Register Direction Bit (D bit) Tells the register operand in REG field in byte 2 is source or destination operand

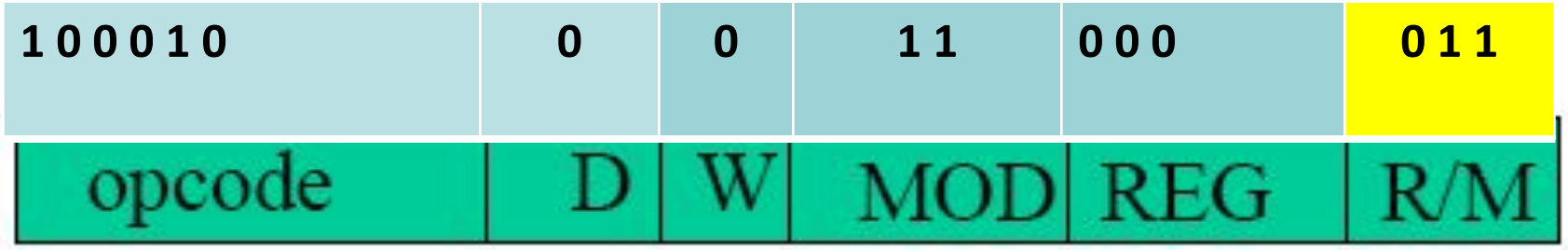
    1: destination 0: source
  - Data Size Bit (W bit) Specifies whether the operation will be performed on 8-bit or 16-bit data

    0: 8 bits 1: 16 bits

Example: MOV BL, AL

(machine code: 10001000 11000011)

(Hexcode: 88 C8H**)**



Opcode = 100010 ⇒ MOV data transfer

D = 0 ⇒ AL is source operand

W = 0 ⇒ 8-bit data transfer

Therefore byte 1 is $10001000_2=88_{16}$

MOD = 11 ⇒ register mode

REG = 000 ⇒ code for AL

R/M = 011 ⇒ destination is BL

Therefore Byte 2 is $11000011_2=C3_{16}$

Assembly Language

- In assembly language, a mneumonic (i.e. memory aid) is used as a short notation for the instruction to be used.

Assembly Language	Machine Code
SUB AX,BX	001010111000011
MOV CX,AX	100010111001000
MOV DX,0	10111010000000000000000

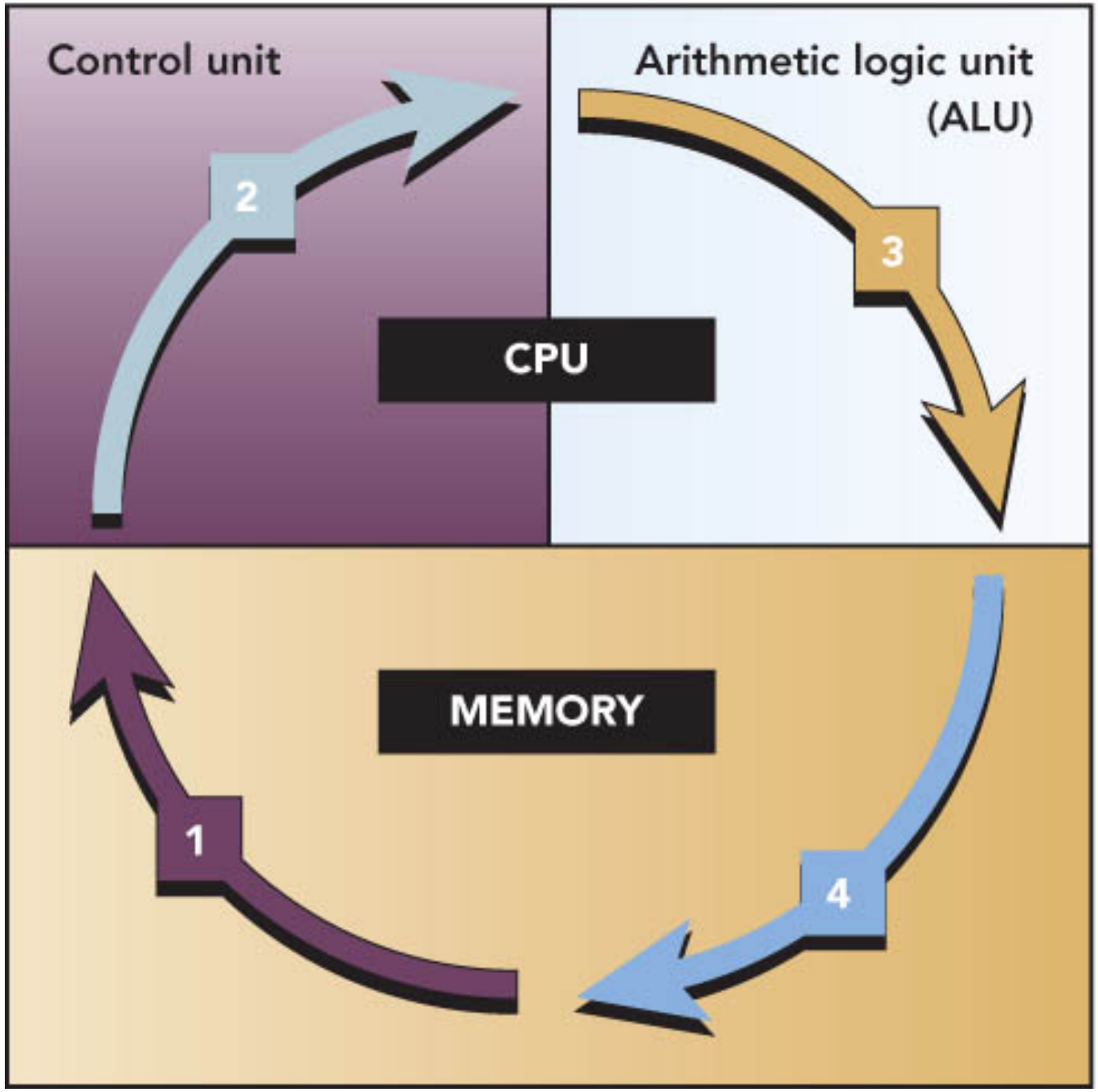
Assembly language is an intermediate step between high level languages and machine code. Most features present in HLL are not present in Assembly Language as type checking etc.

RAM(16 bits)

Address	Contents
1256H	0010101**11000011**
1257H	1000101**11001000**

RAM(8 bits)

Address	Contents
1256H	**11000011**
1257H	**0010101**
1258H	**11001000**
1259H	**1000101**



INSTRUCTION CYCLE

1 Fetch

Retrieves the next program instruction from memory

2 Decode

Determines what the program is telling the computer to do

EXECUTION CYCLE

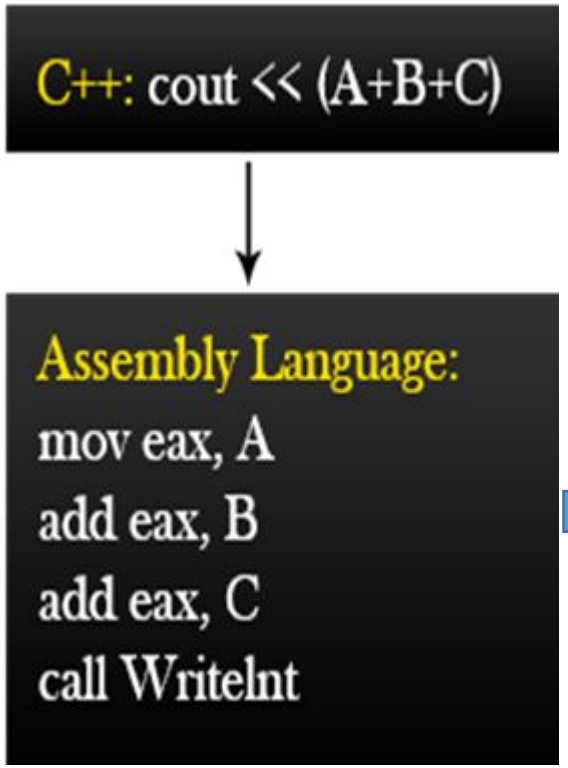
3 Execute

Performs the requested instruction

4 Store

Stores the results to an internal register (a temporary storage location) or to memory

How a program & data are stored in Memory?



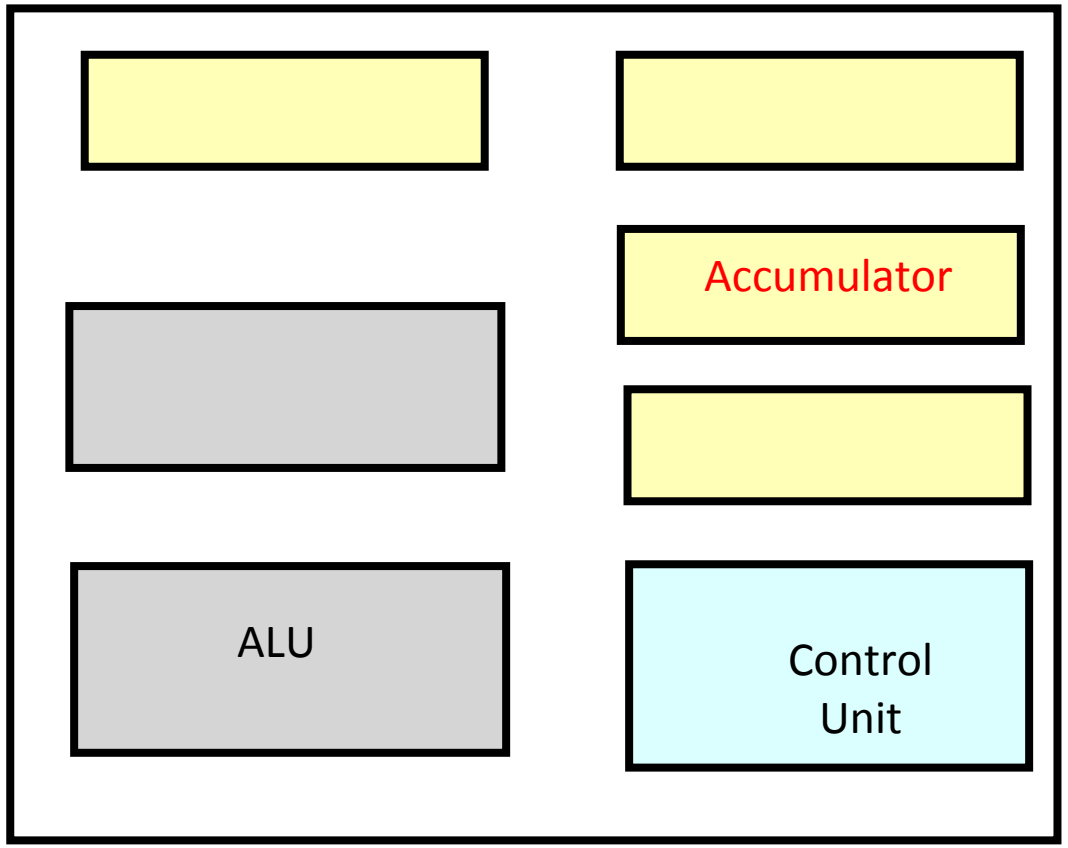
Machine codes
11001101
10001101
11101101
11011101

Stored in RAM

ADDRESS	CONTENTS
000100100101 (125H)	11001101 (Machine code of Instruction-1)
000100100110 (126H)	10001101 (Machine code of Instruction-2)
000100100111 (127H)	11101101 (Machine code of Instruction-3)
000100101000 (128H)	11011101 (Machine code of Instruction-4)
200H	11000101 (DATA-1)
201H	11000001 (DATA-2)
202H	11000111 (DATA-3)

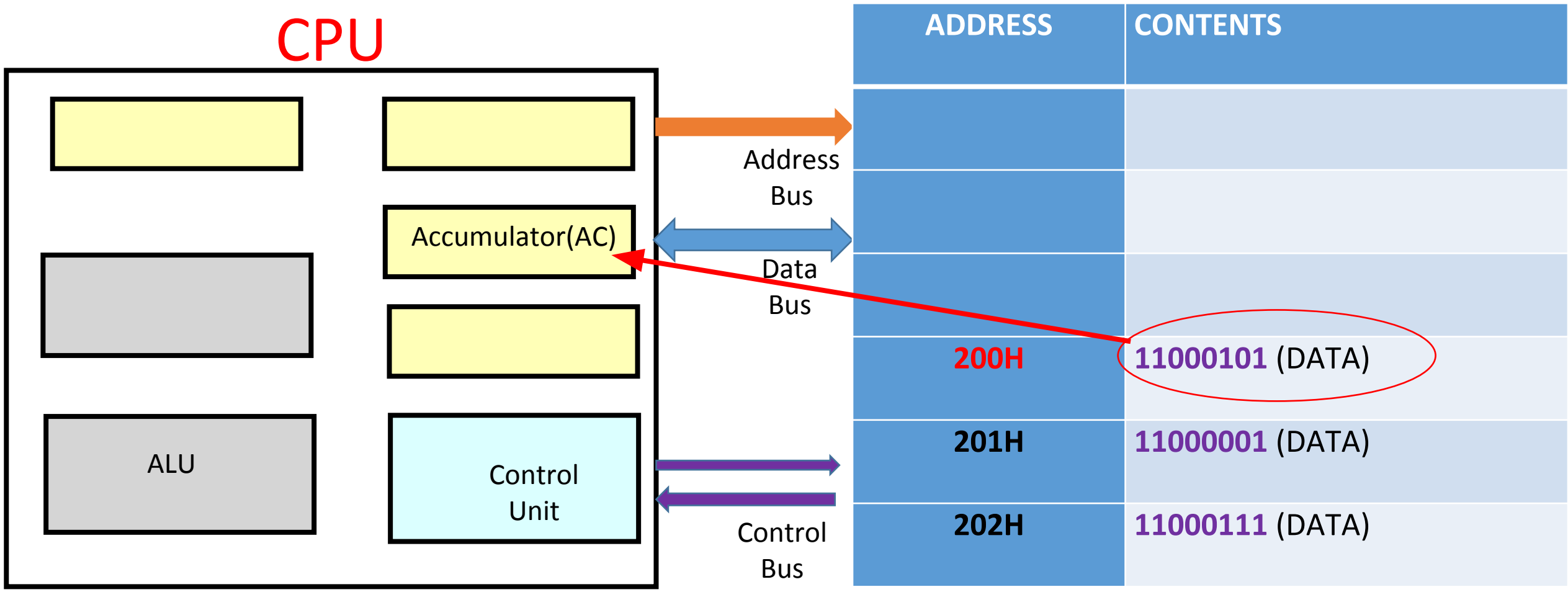
- Starting address of the program is set by operating system/user depending on system. Here starting address of memory (RAM) is chosen randomly as 125H.
- Machine code of 1st Instruction is stored at 125H.
- Following instructions are stored in consecutive locations of RAM.
- Data, if used and if required to store in RAM, could be stored in a different segment but in consecutive locations of RAM

Accumulator-based CPU



- Accumulator is a special purpose register within CPU
- Commonly indicated by AC
- CPU uses this register in almost all instructions by default and it remains implicit (not written/indicated in instructions)
- Results/partial results are also stored in AC by default

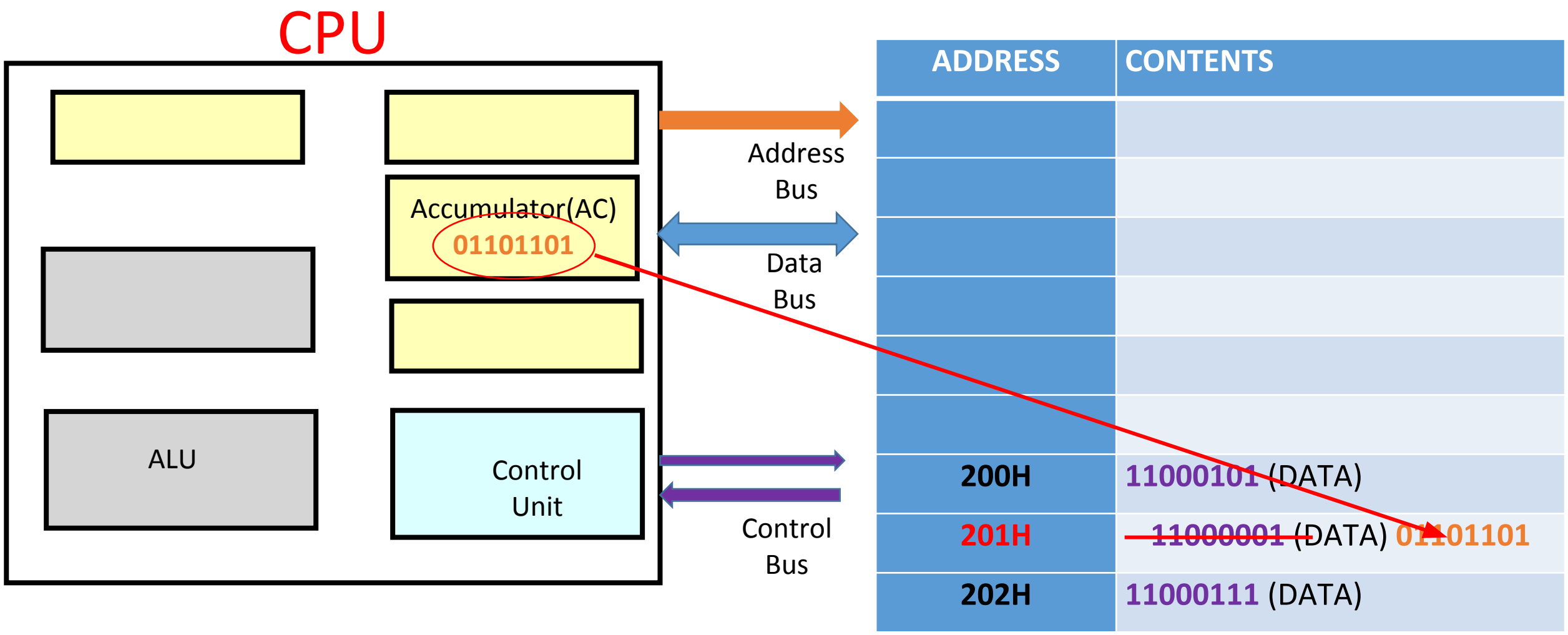
Accumulator-based



Example: **LOAD M1** **;M1= 200H**

The content of memory location 200H is transferred/copied into AC

Accumulator-based

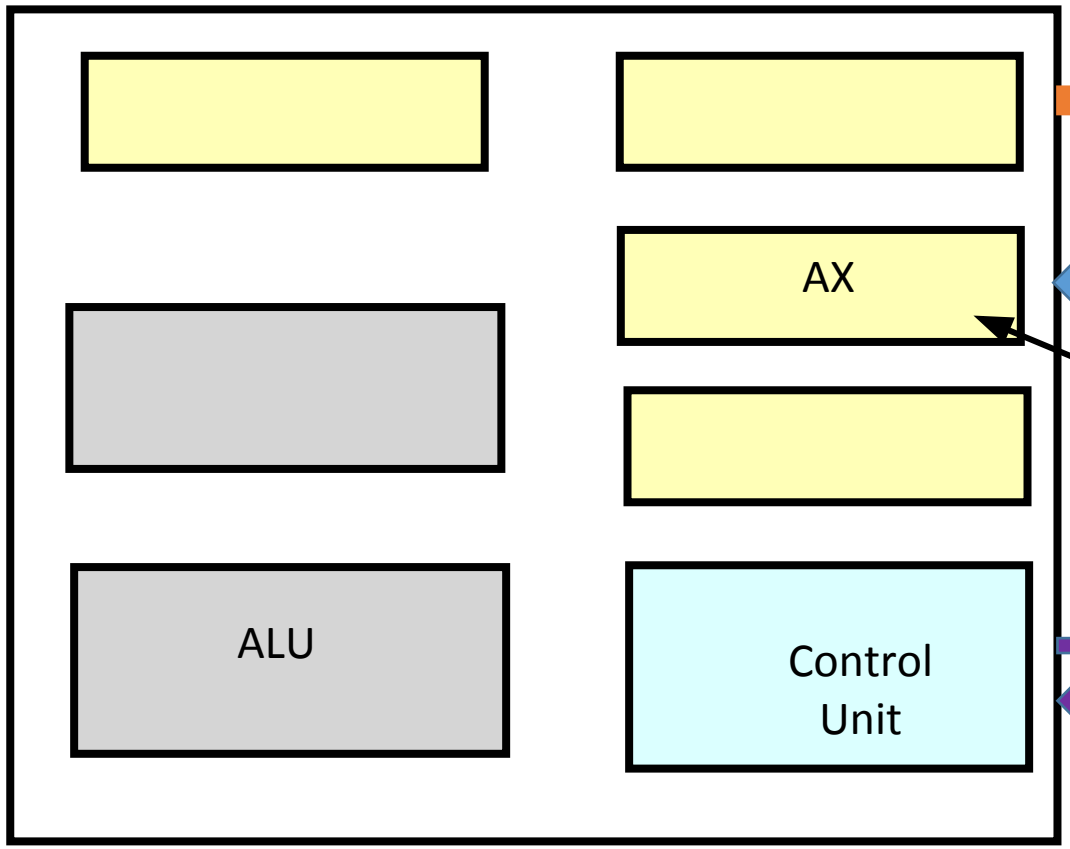


Example: **STOR M2** ;M2 = 201H and [AC] = 01101101

Current content of AC is saved into RAM at 201H

General purpose

CPU



Address Bus

Data Bus

Control Bus

Memory

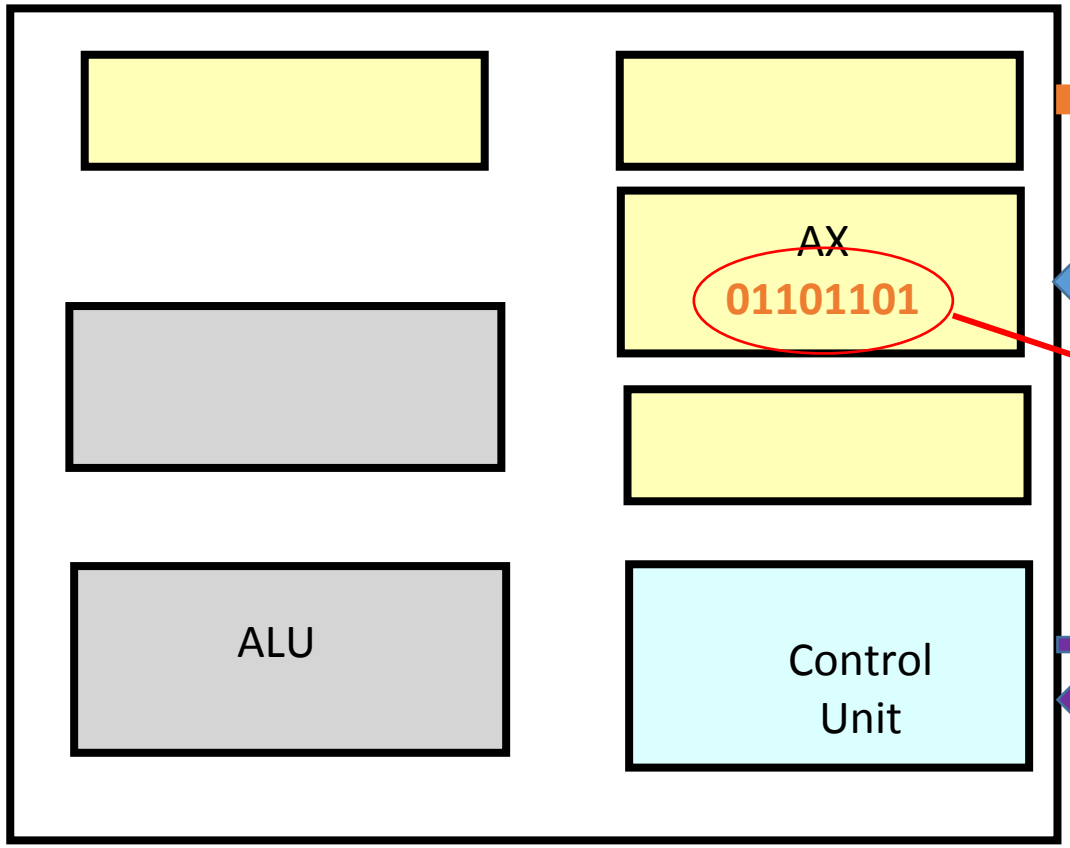
ADDRESS	CONTENTS
200H	**11000101** (DATA-1)
201H	**11000001** (DATA-2)
202H	**11000111** (DATA-3)

Example: **MOV AX, M1** ;M1= 200H

The content of memory location 200H is copied into AX register

General purpose

CPU



Address Bus

Data Bus

Control Bus

Memory

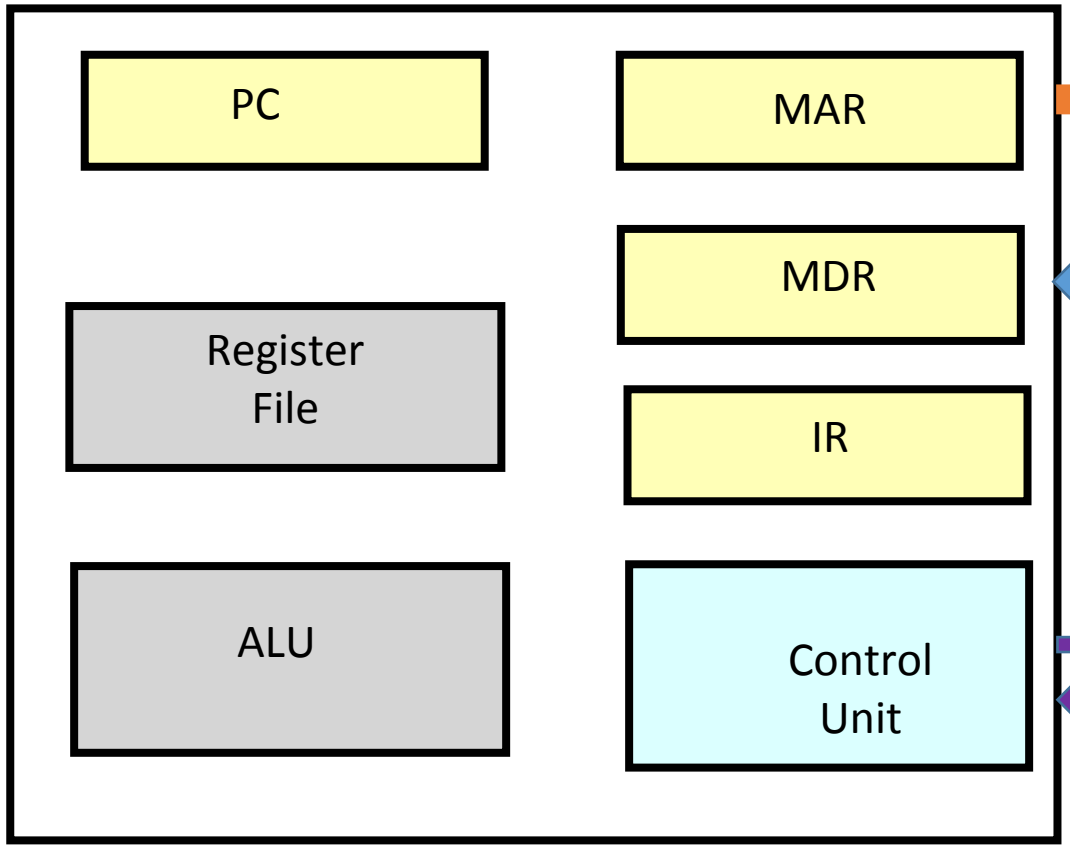
ADDRESS	CONTENTS
200H	11000101 (DATA-1)
201H	~~11000001~~ (DATA-2) 01101101
202H	11000111 (DATA-3)

MOV M2, AX ;M2 = 201H

; [AX] = 01101101

Computer Components

CPU



Address Bus

Data Bus

Control Bus

Memory

ADDRESS	CONTENTS
000100100101 (125H)	11001101 (Machine code of Instruction-1)
000100100110 (126H)	10001101 (Machine code of Instruction-2)
000100100111 (127H)	11101101 (Machine code of Instruction-3)
000100101000 (128H)	11011101 (Machine code of Instruction-4)
200H	11000101 (DATA-1)
201H	11000001 (DATA-1)
202H	11000111 (DATA-1)

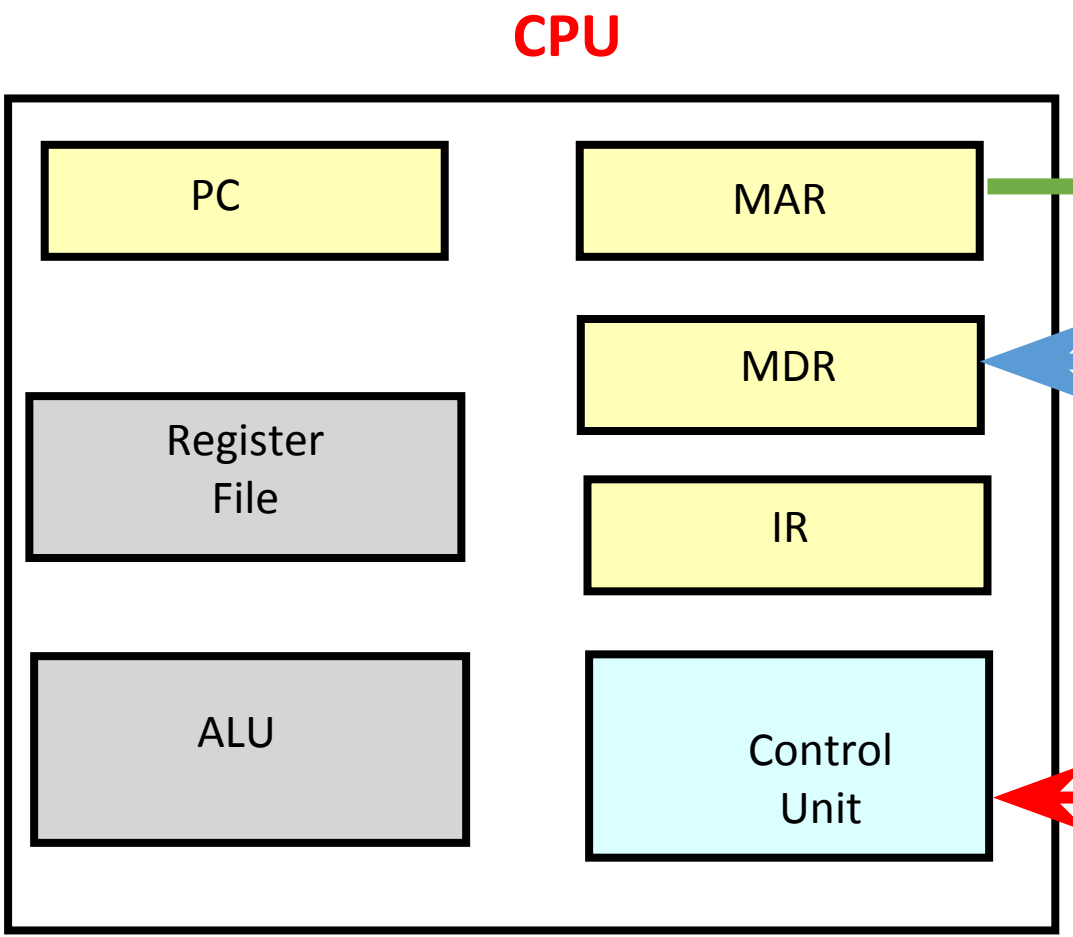
PC-Program counter
MAR-Memory Address Register

IR-Instruction Register
MDR-Memory Data Register

Control Unit decodes Instructions & Generates control signals

How does computer work?

User program
SUB AX, BX ;0010101**11000011**
MOV CX, AX ;1000101**11001000**
MOV DX, 0

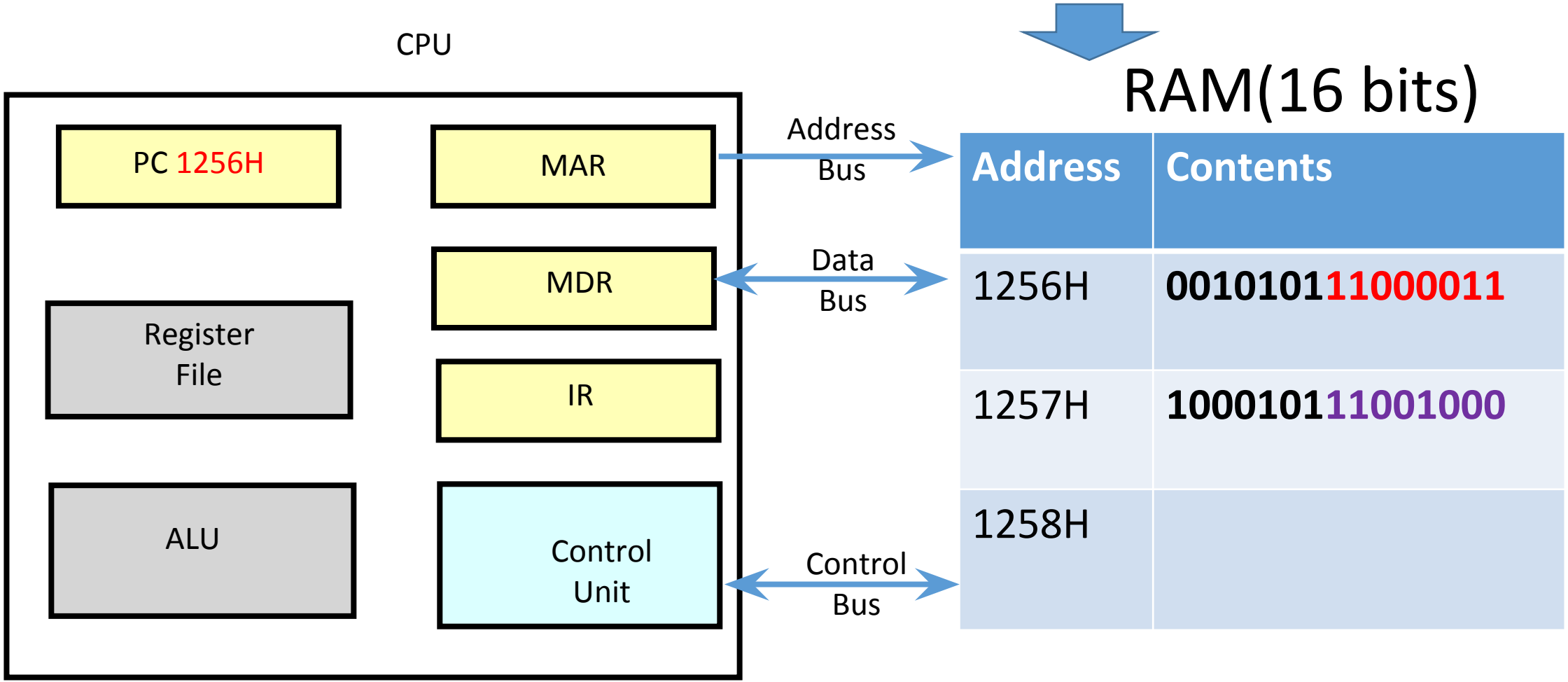


Stored in RAM(16 bits)

Address	Contents
0001001001010110 (1256H)	**0010101****11000011**
0001001001010111 (1257H)	**1000101****11001000**

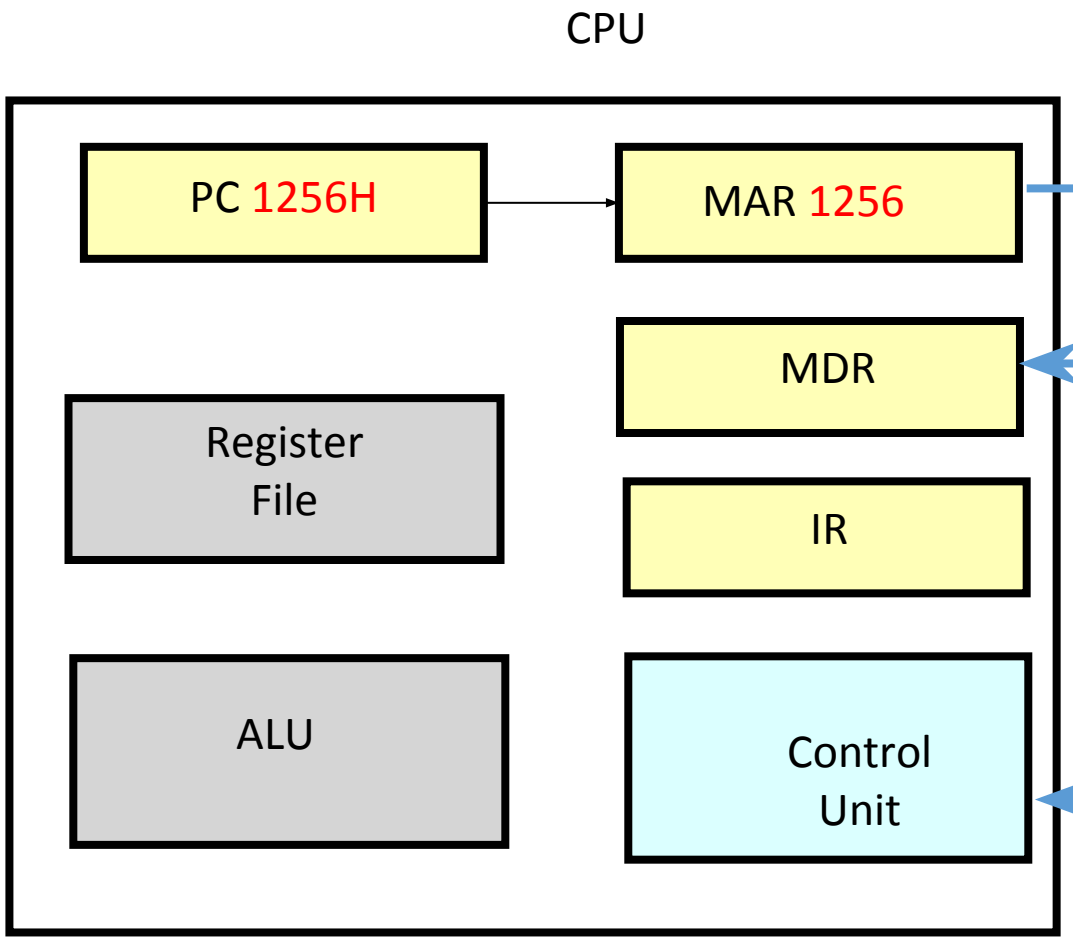
STEP-1: PC is loaded with address of 1st instruction of program

User program
SUB AX, BX ;0010101**11000011**
MOV CX, AX ;1000101**11001000**
MOV DX, 0



STEP-2: content of PC is loaded into MAR

User program
SUB AX, BX
MOV CX, AX
MOV DX, 0



Address Bus

Data Bus

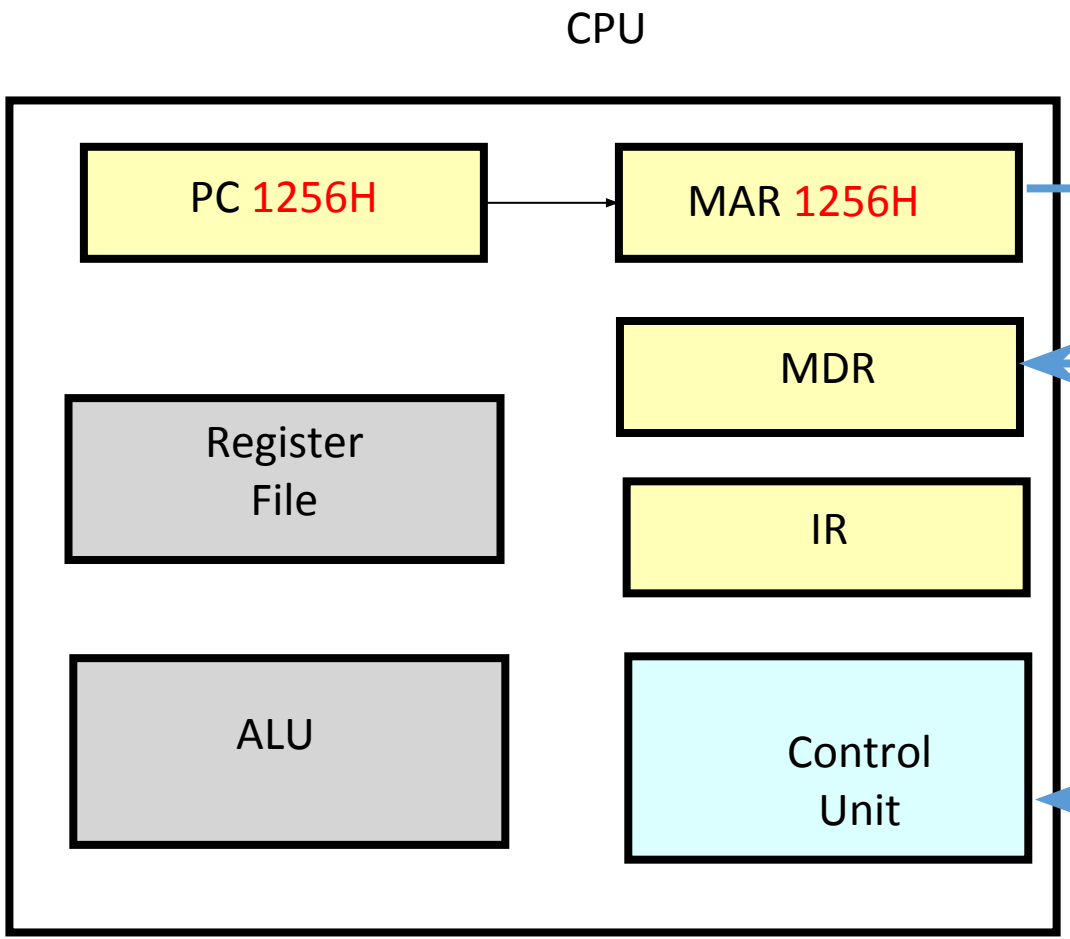
Control Bus

RAM(16 bits)

Address	Contents
1256H	**0010101**11000011
1257H	**1000101**11001000
1258H	**1001111**11001011

STEP-3: content of MAR is placed on Address bus and applied to Memory, as a result memory location 1256H is selected

User program
SUB AX, BX
MOV CX, AX
MOV DX, 0



Address Bus 1256H

Data Bus

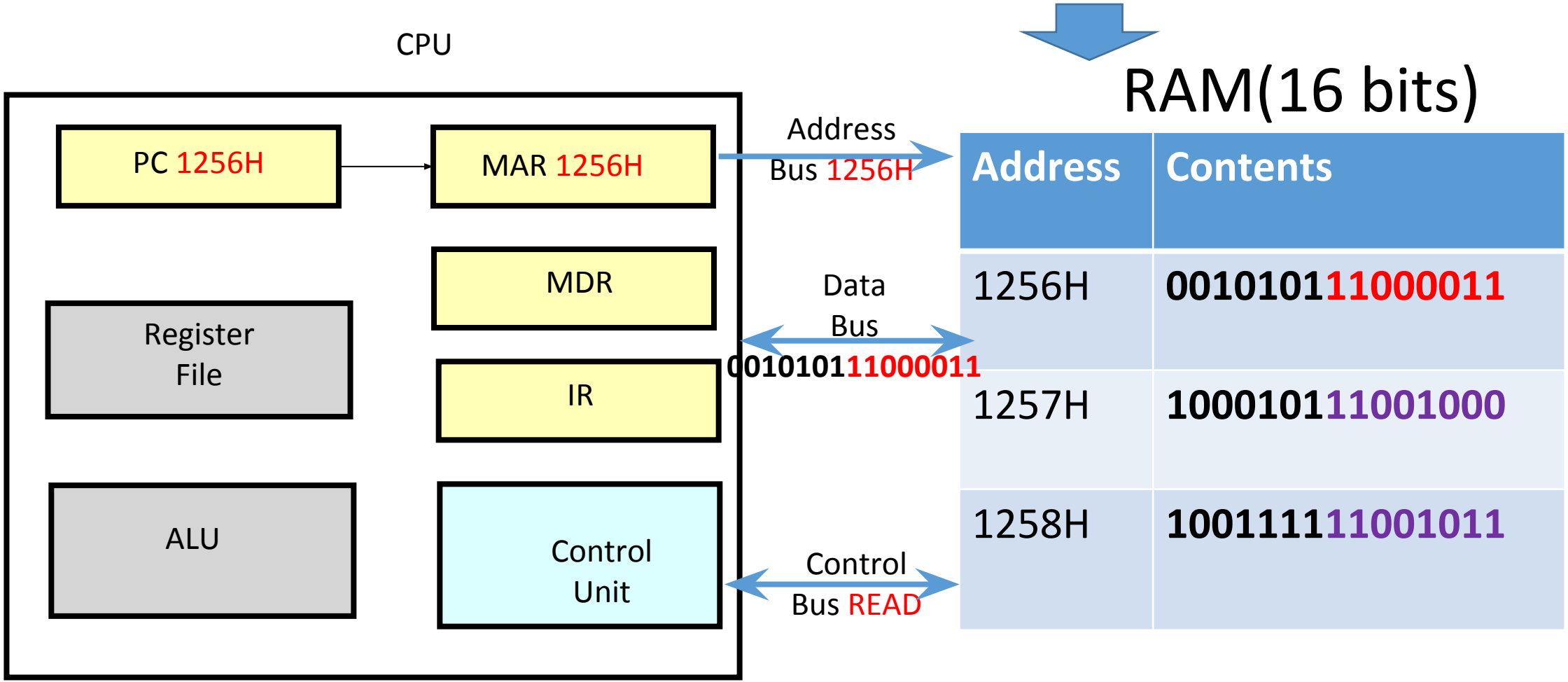
Control Bus

RAM(16 bits)

Address	Contents
1256H	0010101**11000011**
1257H	1000101**11001000**
1258H	1001111**11001011**

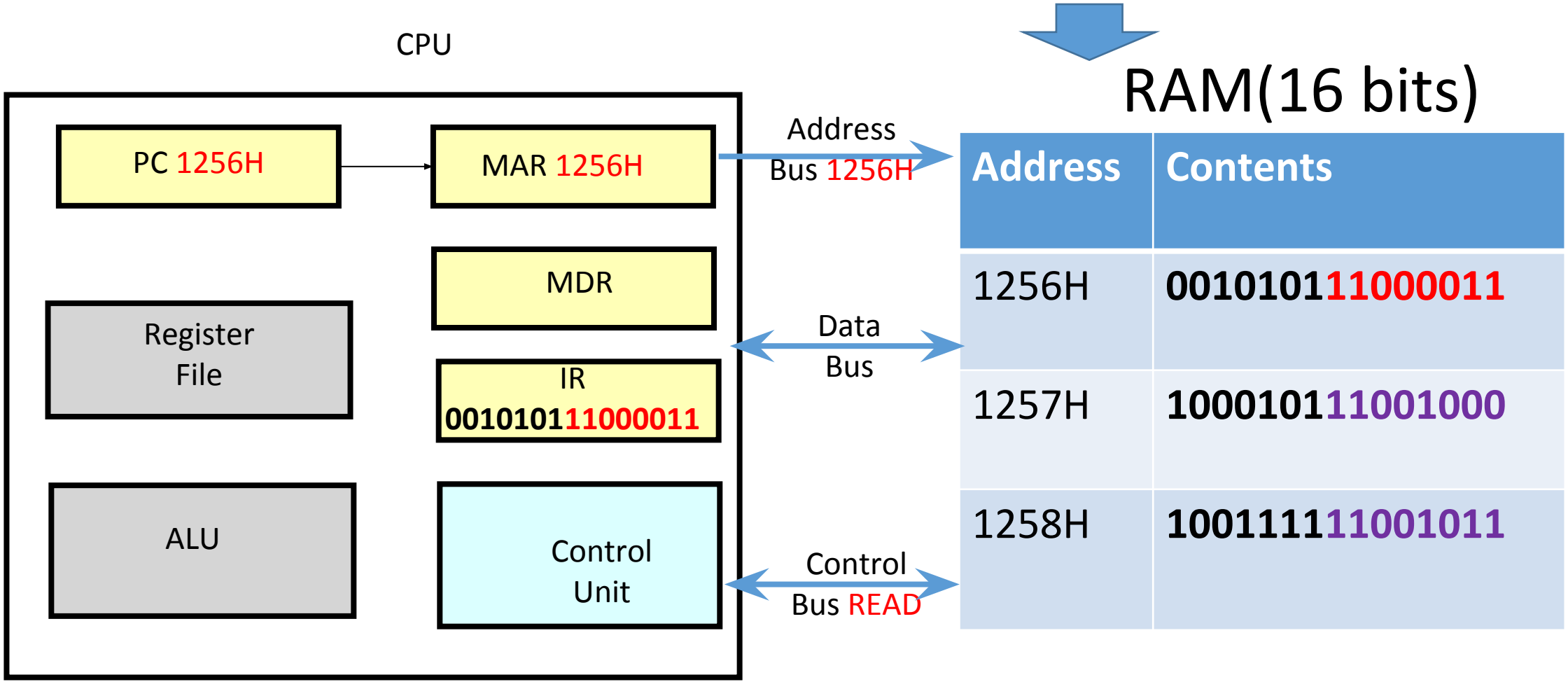
STEP-4: Control unit send READ control signal to RAM, as a result machine code of 1st instruction is available on Data Bus

User program
SUB AX, BX
MOV CX, AX
MOV DX, 0



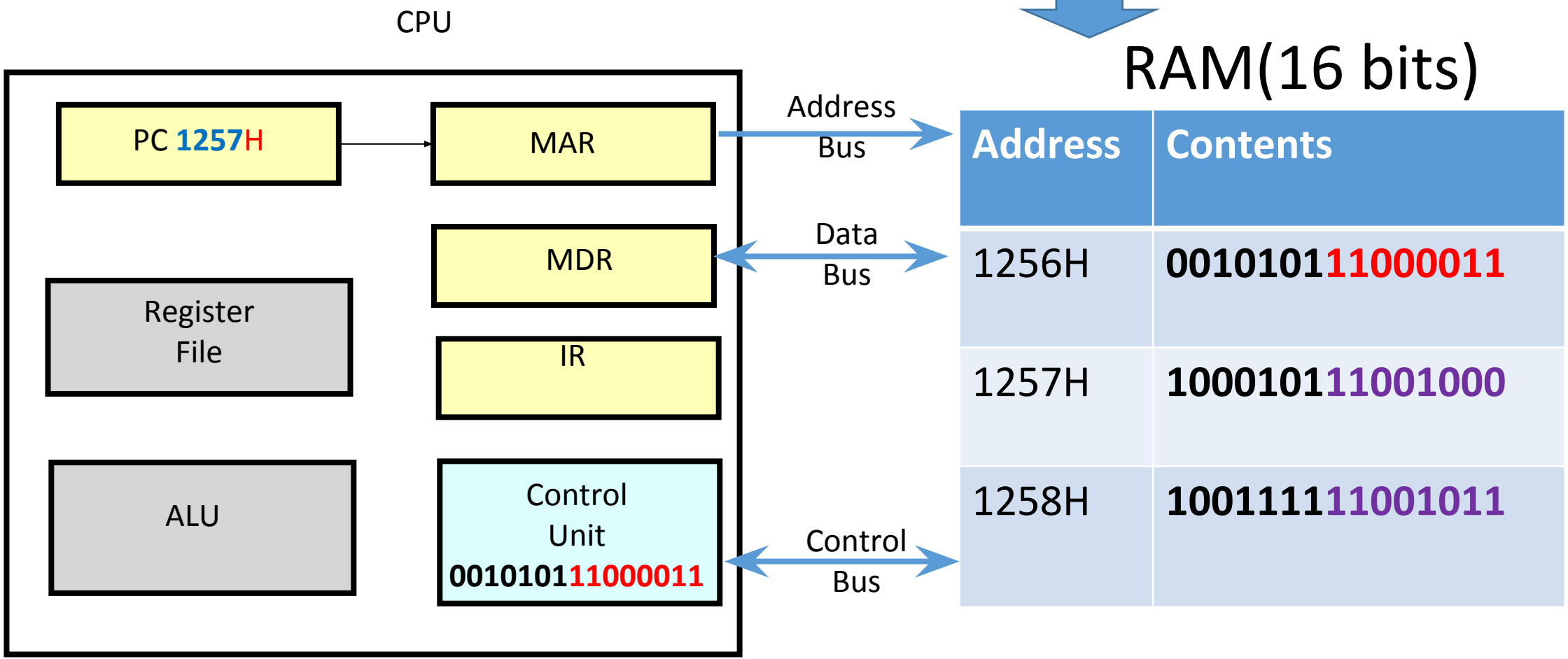
STEP-5: Machine code of 1st instruction is loaded into IR

User program	
SUB AX, BX	;0010101**11000011**
MOV CX, AX	;**10001011**1001000
MOV DX, 0	



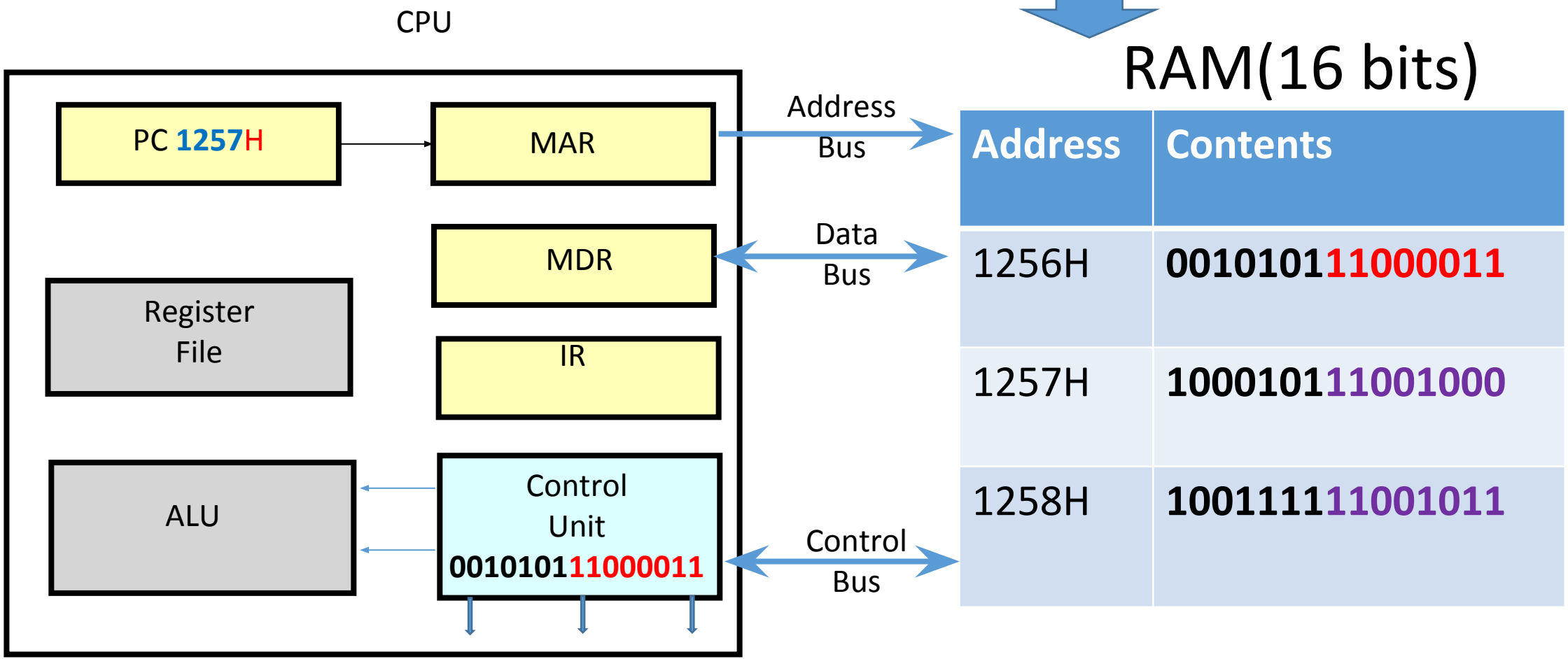
STEP-6: PC is incremented by 1 to point next instruction to be executed. Contents of IR is fed to Control Unit

User program
SUB AX, BX
MOV CX, AX
MOV DX, 0



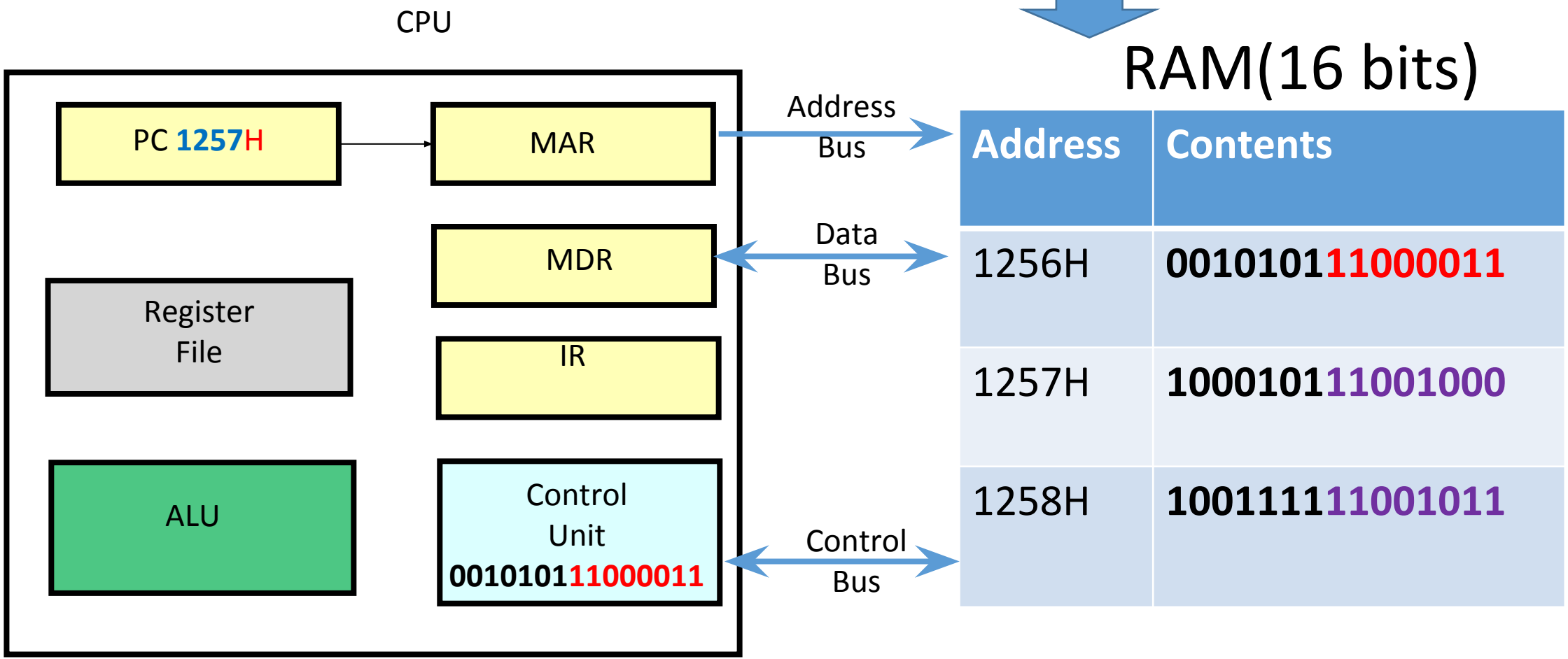
STEP-7: 1st instruction is decoded at control unit: control signals are generated to activate ALU for specific operation as per instruction

User program
SUB AX, BX
MOV CX, AX
MOV DX, 0



STEP-8: 1st instruction is Executed (that includes Data read from RAM followed by ALU operation, result stored as per instruction)

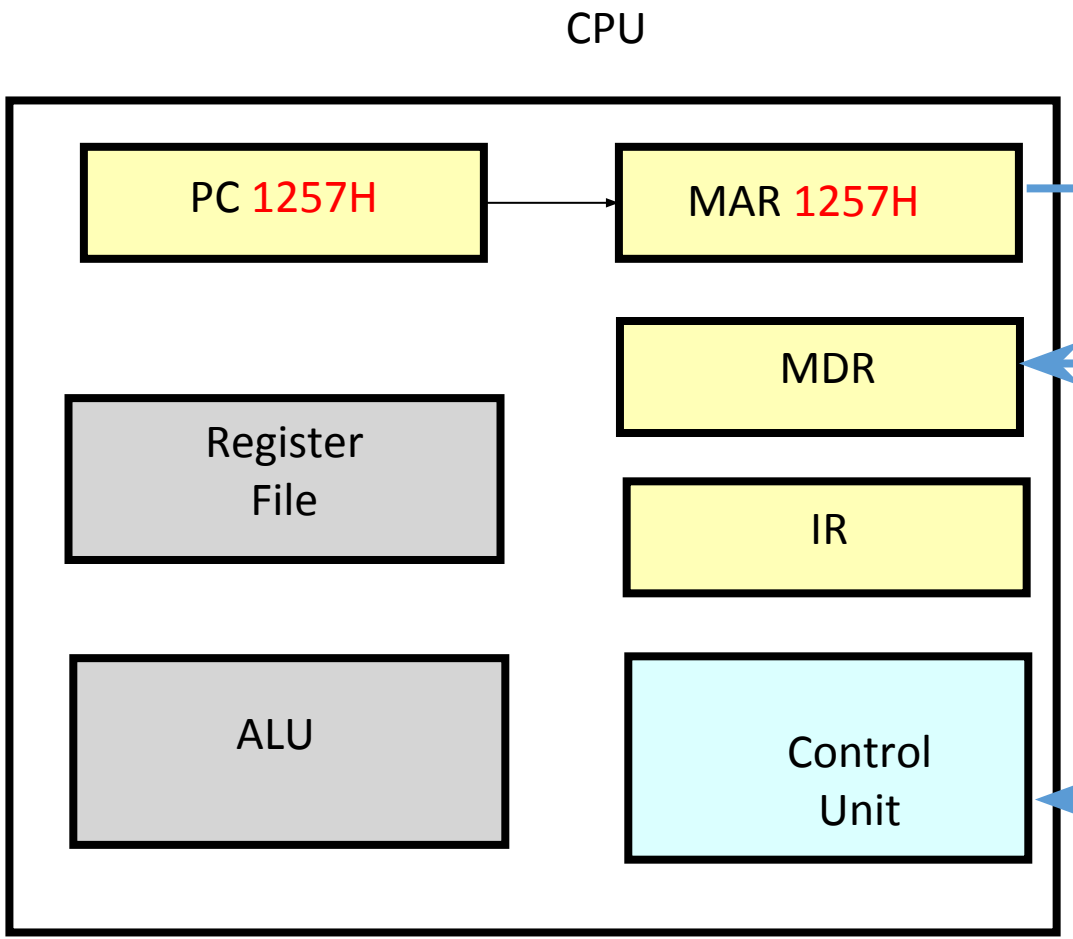
User program
SUB AX, BX
MOV CX, AX
MOV DX, 0



STEPS-2:8 repeated for next instruction

content of PC is loaded into MAR

User program
SUB AX, BX
MOV CX, AX
MOV DX, 0



Address Bus

Data Bus

Control Bus

RAM(16 bits)

Address	Contents
1256H	**0010101**11000011
1257H	**1000101**11001000
1258H	**1001111**11001011

Instruction Set Design

 # Types of Operations /Instructions and

Complete list of Instructions

 # Mnemonics of Instructions or operations (Usually represented by assembly codes)

 # Formats of Instructions (fields/subfields/size in binary bits/data format/memory address formats/addressing modes etc)

 # Machine Codes in Binary

Instruction Set Architecture

- How many instructions and types of instructions processors can understand/decode/process/execute
- How to perform any operation: Accumulator based or General?
- Instruction Format?
- Number of binary bits used to form Instructions
- Same number of bits for all instructions or not?
- How arithmetic/logical/data processing/data transfer operations are encoded in instructions
- How data/main memory address/register names are indicated in instructions
- Data types/formats/number of bits used/positive/negative numbers/ranges of numbers etc
- Memory: Address format/addressing scheme/content of each addressable locations etc

Instruction Set Architecture

Assembly Language

3

- In assembly language, a mneumonic (i.e. memory aid) is used as a short notation for the instruction to be used.

Assembly Language	Machine Code
SUB AX,BX	001010111000011
MOV CX,AX	100010111001000
MOV DX,0	10111010000000000000000

Assembly language is an intermediate step between high level languages and machine code. Most features present in HLL are not present in Assembly Language as type checking etc.

Instruction Formats (3-operand fields)

ADD M, D1, D2 [M] ⇐ D1+ D2

Binary code for ALU operation (Op Code)	Memory address to store result (Binary)	Data -1 (Binary)	Data-2 (Binary)

ADD M1, M2, M3 [M1] ⇐ [M2]+ [M3]

Binary code for ALU operation (Op Code)	Memory address to store result (Binary)	Memory address of Data -1 (Binary)	Memory address of Data-2 (Binary)

ADD R1, R2, R3 R1 ⇐ R2+ R3

Binary code for ALU operation (Op Code)	CPU Register to store result (Binary)	CPU register containing Data -1 (Binary)	CPU register containing Data -1 (Binary)

Instruction Formats(2-operand field)

ADD M1, M2 [M1] ⇐ [M1]+ [M2]

Binary code for ALU operation (Op Code)	CPU register or memory address that contains Data-1 (Binary) Result is stored in same CPU register or memory address after operation	CPU register or memory address that contains Data-2 (Binary)

ADD R1, R2 R1 ⇐ R1 + R2

ADD M, R1 [M] ⇐ R1 + [M]

Instruction Formats(one-operand field)

ADD M AC ⇐ [M] + AC

Binary code for ALU operation (Op Code)	CPU register or memory address that contains Data-2 (Binary)

Here Data-1 should be loaded into a predefined register, namely Accumulator (AC) prior to use this instruction. Moreover, the result is stored into Accumulator (AC) as well. Interestingly, the Accumulator (AC) is not explicitly indicated in the Instruction sub-field!

ADD R1 AC ⇐ R1 + AC

Instruction Formats(Opcode only!)

CLC Clear carry Flag bit

Binary code for ALU /special operation

Data to be used should be initially loaded into default CPU register. Result is also stored there as well.

Example: Opcode of Instructions

Operation	Mnemonic	Opcode
Addition	ADD	001
Subtraction	SUB	010
Data Transfer	MOV	011
Load Accumulator	LOAD	100
Multiply	**MUL**	**101**
Read from Input device	READ	110
Send to output device	STORE	111

Format of an Instruction

OP CODE 4-BITS	OPERAND FIELD-1 4-BITS	OPERAND FIELD-2 4-BITS	OPERAND FIELD-3 4-BITS

OP CODE 4-BITS	OPERAND FIELD-1 4-BITS	OPERAND FIELD-2 4-BITS

OP CODE 8-BITS	OPERAND FIELD-1 16-BITS	OPERAND FIELD-2 16-BITS

Instruction Set Architecture

- RISC (Reduced Instruction Set Computer) Architectures
  - **Memory accesses are restricted to load and store instruction, and data manipulation instructions are register to register.**
  - **Addressing modes are limited in number.**
  - **Instruction formats are all of the same length.**
  - **Instructions perform elementary operations**

- CISC (Complex Instruction Set Computer) Architectures
  - **Memory access is directly available to most types of instruction.**
  - **Addressing mode are substantial in number.**
  - **Instruction formats are of different lengths.**
  - **Instructions perform both elementary and complex operations.**

Sample questions

- What do you understand by computer architecture?
- What are the architectural attributes?
- What are the main features of von Neumann architecture?
- What is the concept of stored program computer?
- What is computer organization?
- What are the organizational attributes?
- State/explain difference between computer architecture and organization.
- What do you understand by Register and Register file? What do registers contain?
- State the types of register. State the functions of Registers. What do you understand by size of Register?
- What do you understand by Instruction and Instruction set and Instruction format?
- State the classification of Processors based on instruction set architecture. Give examples. Also state relative merits and demerits.
- What do you understand by machine code? Show general format of machine code/Instruction format.
- What do you understand by main memory? Is it volatile or non volatile?
- What do you understand by Random Access Memory? Why it is named so?
- What do you understand by Memory address? How it is formed? How it is represented? Give examples.

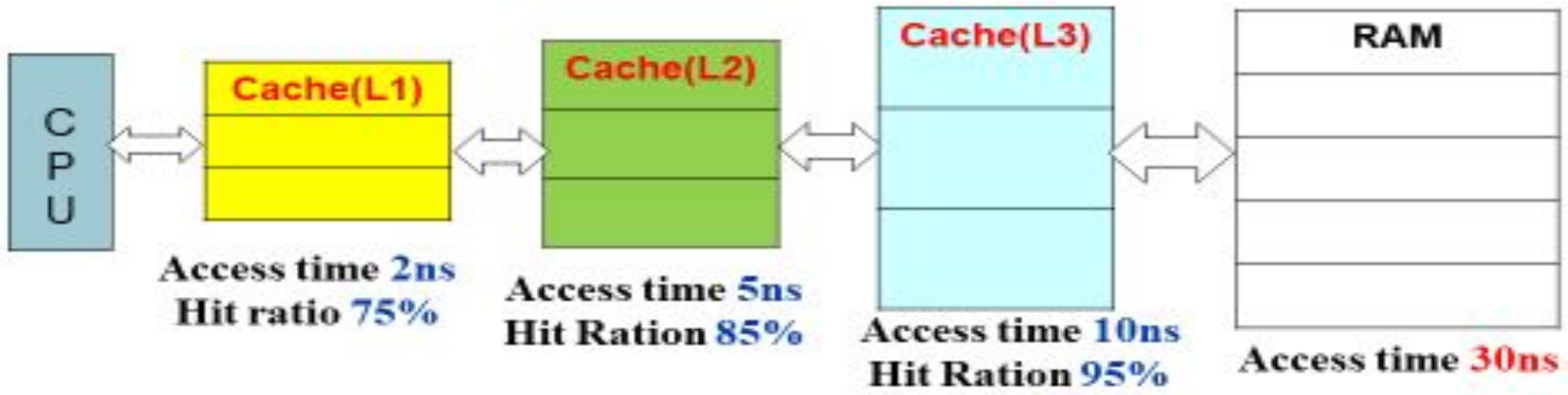
- Show the address formats of (i) 1KB, (ii) 1MB, (iii) 1K x 16 bits, 4K x 32 bits memory ICs.
- What do memory locations contain? Give examples
- What do you understand by Byte-addressable memory and Word-addressable memory? State differences, if any.
- What do you understand by bus system? Discuss types, sizes of types and their impacts on performance of a computer.
- Draw a simple schematic diagram of a computer and identify its main functional units.
- Draw a simple schematic diagram of a CPU/Microprocessor and identify its main functional units.
- List the steps usually followed by a general purpose processor to run a program.
- List the steps usually followed by a general purpose processor to run a program.
- List the steps usually followed by a general purpose processor to process a single instruction. Use suitable diagram.

- What do you understand by Instruction fetch and Instruction Execution?
- How does a CPU/Microprocessor run a program? Use a suitable schematic diagram and list the steps in chronological order.
- What is program counter (pc)? What is Instruction register (IR)? Discuss the functions of each.
- What do you understand by Port address? Why it is used?
- Briefly explain Read / Write operation.
- List the four main structural components of a computer and briefly state their functions.
- Briefly describe IAS structure with a suitable diagram.
- List the registers of IAS structure and state their functions.
- Show the instruction and data format of IAS structure.
- Briefly describe how IAS structure works?
- List the steps how IAS structure runs a program?

Quiz-1(Summer-2020)

- A program containing 2 Million instructions with the following instruction mix is run on a RISC machine having following CPI values:
- Op Freq CPI
- ALU 20% 2
- Load 40% 6
- Store 30% 5
- Branch 10% 2
- If the CPU runs at 200MHz, calculate the execution time of the program.
- If a CPU design enhancement reduces the CPI of Load and Store instructions by 2 and 3 respectively. Calculate the resulting performance improvement from this enhancement using Amdahl's law. Also calculate the execution time with enhancement.

Quiz-2(Summer-2020)



Suppose a program, initially loaded into RAM, contains 1500 instructions. The Hit ratios of L1, L2 and L3 Caches are 75%, 85% and 95% respectively. Calculate average access time. Also calculate total access time.

Solution: