CT 215-1

Computer Organization

&

Assembly Language

Contents

1. Introduction

Basic concepts , machine language, numbering systems ,Hardware organization IBM PC , element of an assembly language program.

2. Assembly language fundamentals

Assembly language syntax , Program data , Variables , Named constants , Basic instruction , program structure

3. The processor status and Flags register

4. Flow control instruction

5. Logic shift and rotate instructions

6. The stack and introduction to procedures

7. Multiplication and Division instructions

8. Arrays and Addressing modes

9. The string instructions

10. Text Display and Keyboard programming

11. Macros

12. BIOS and DOS interrupts

13. Color Graphics

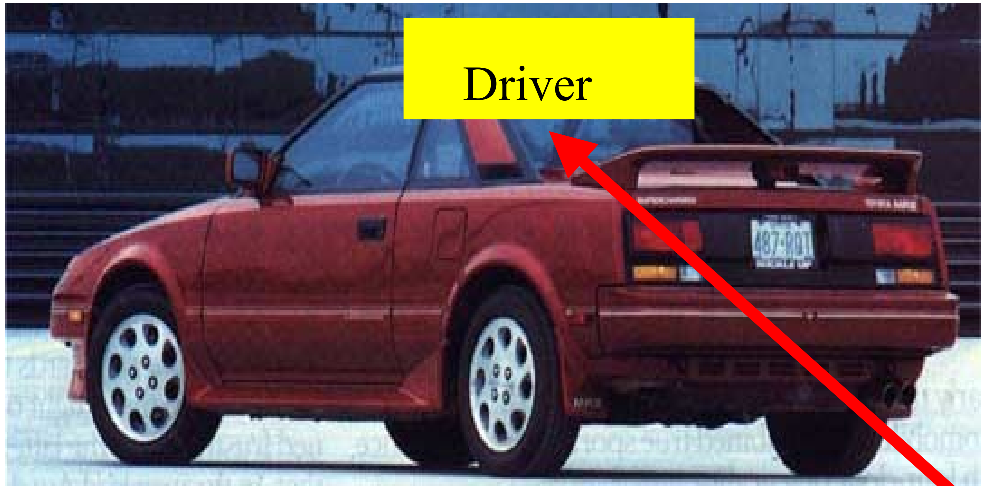
14. Advanced Arithmetic

Computer Organization

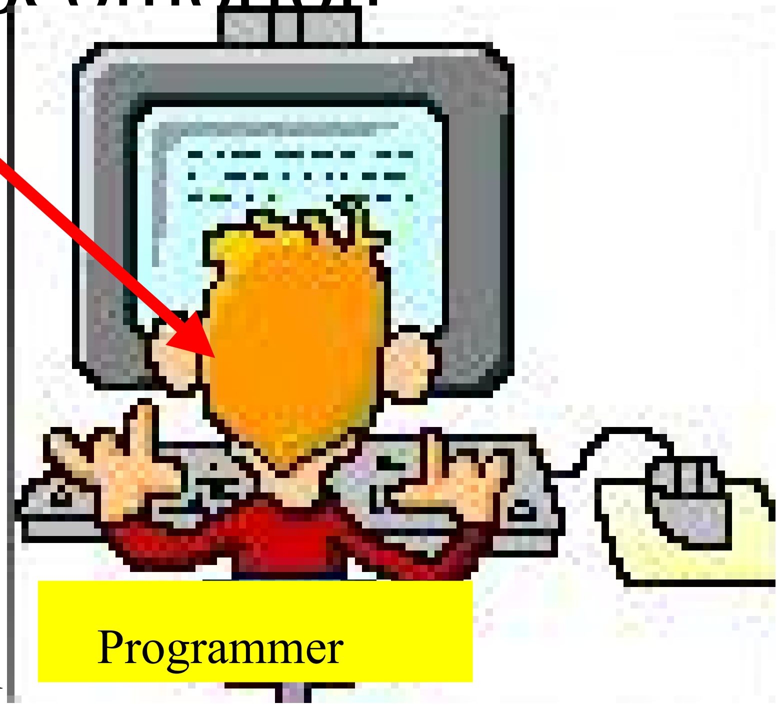
Computer organization refers to the operational units and their interconnections that realized the architectural specifications. Examples of architectural attributes include the instruction set, the number of bits used to represent various data type(e,g, numbers and character) , I/O mechanisms, and techniques for addressing mode Organizational attributes include those hardware details transparent to the programmer, such as control signals , interfaces between the computer and peripherals and the memory technology used.

Assembly Language is a programming language with one to one correspondence between its statements and a computer's machine language . There is no single assembly language because there is no single type of computer CPU. Each assembly language is directly influenced by a computer's machine instruction set and hardware architecture..

Computer: A machine



Example 1: An automobile augments our power of locomotion

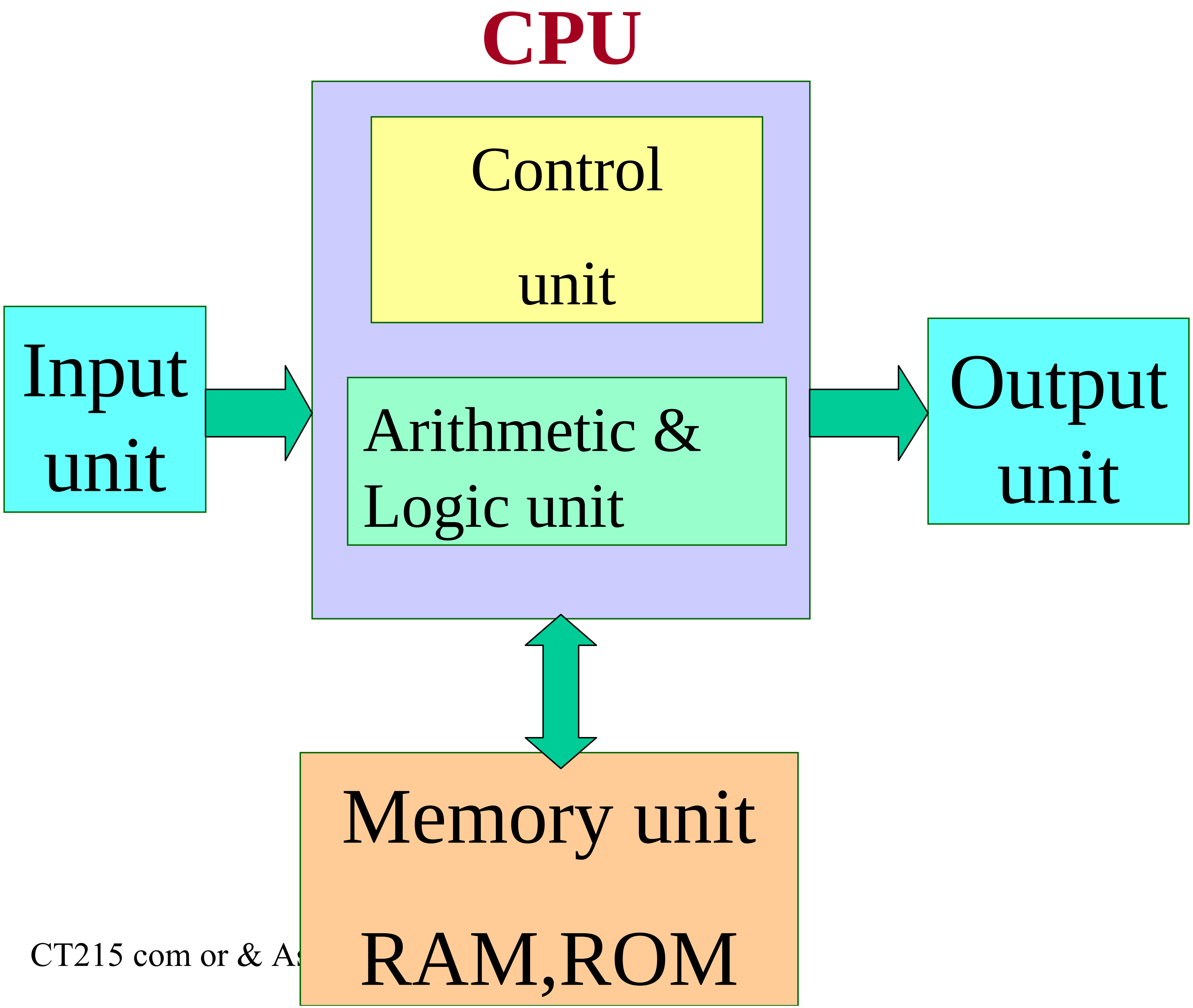


A computer is a device capable of solving problems according to designed program. It simply augments our power of storage and speed of calculation.

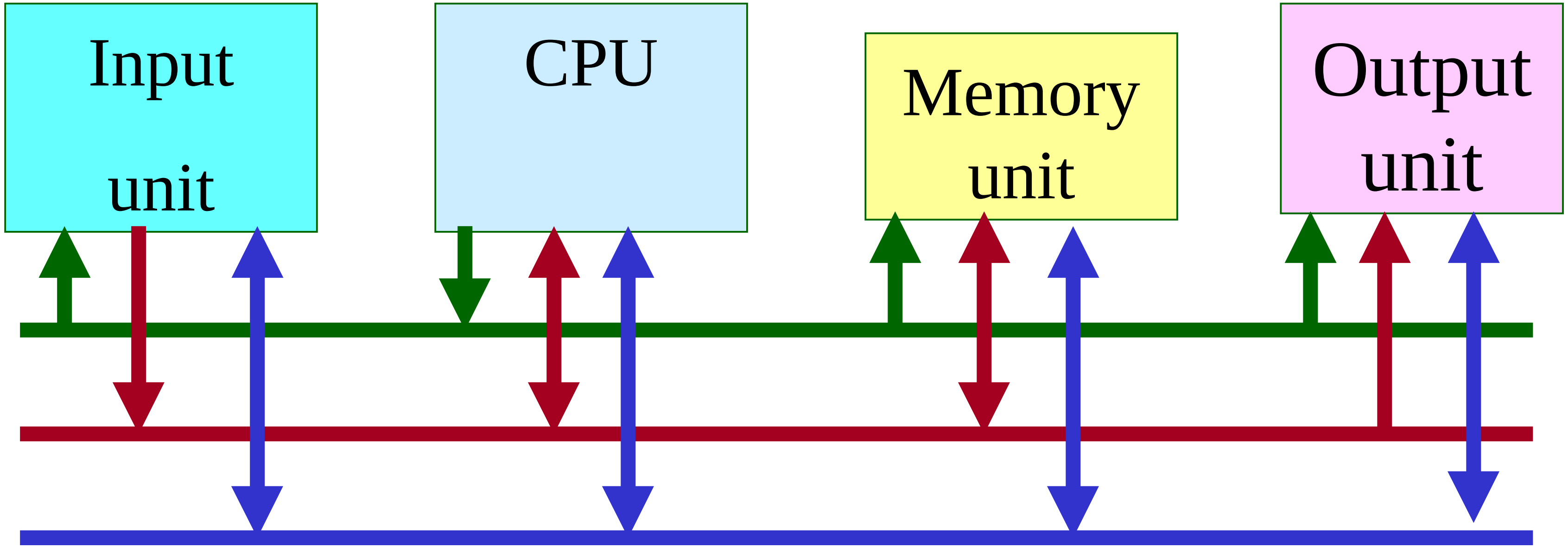
Basic Parts of a Computer

The five basic parts are:

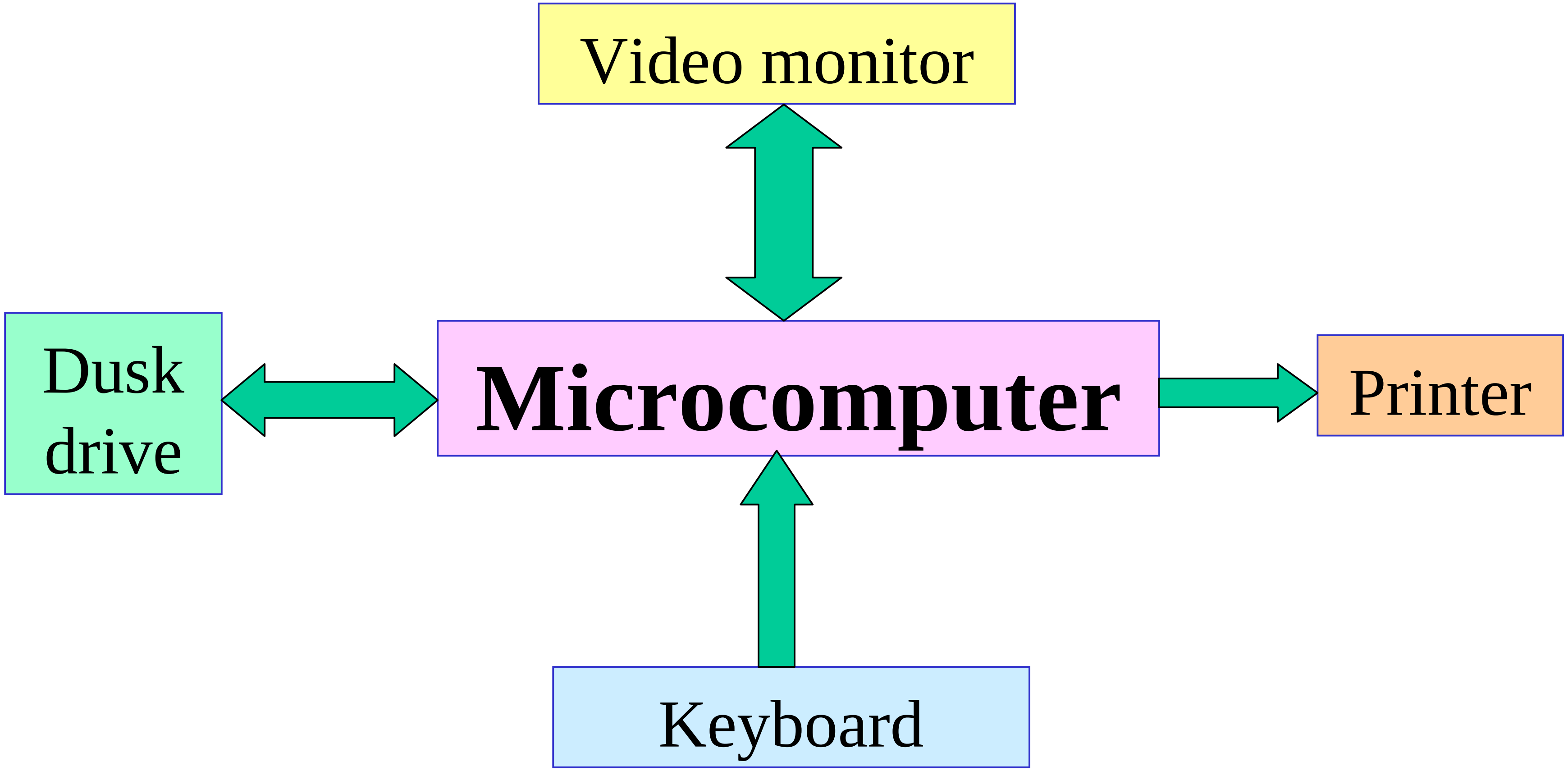
- The arithmetic & logic unit. (ALU)
- The control unit.
- The memory unit.
- The input unit.
- The Output unit.


CPU
Control
unit
Input
unit
Arithmetic &
Logic unit
Output
unit
Memory unit
RAM,ROM

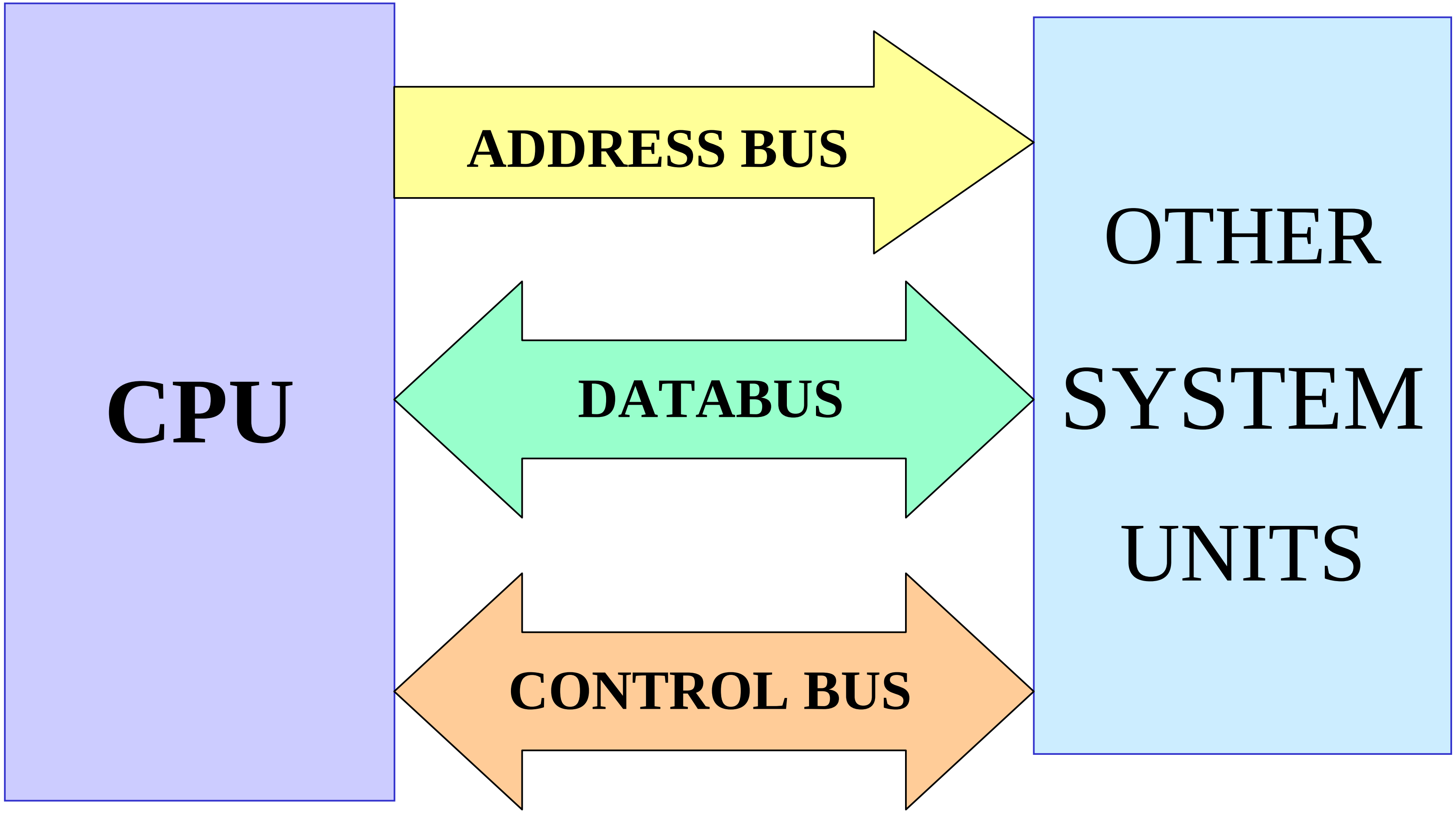
The Bus structure of a computer



A Typical Microcomputer system



THE CPU INTERFACE AND COMMUNICATIONS

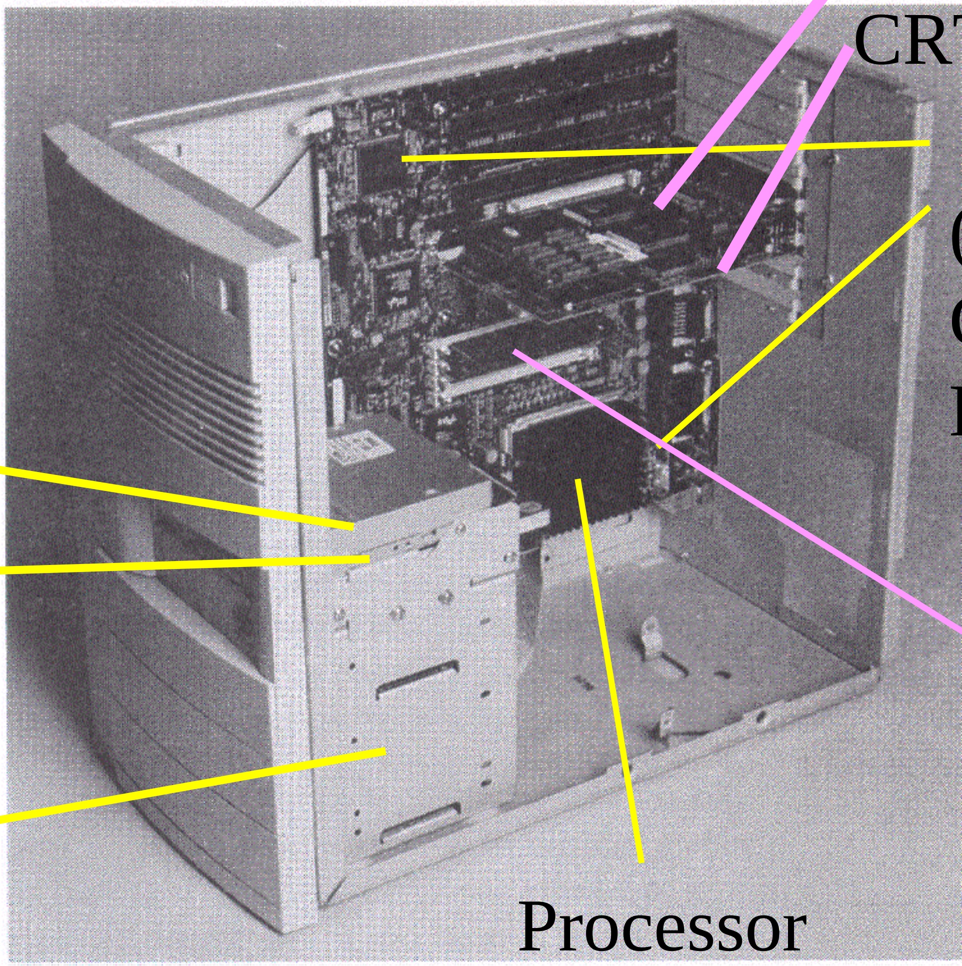


Various components of a computer

Where are the components in my Computer?

Network card and CRT card

Motherboard (Printed Circuit Board)



Floppy disk drive and Hard disk drive

Cage for mounting drives

Processor

Slots for RAM chip

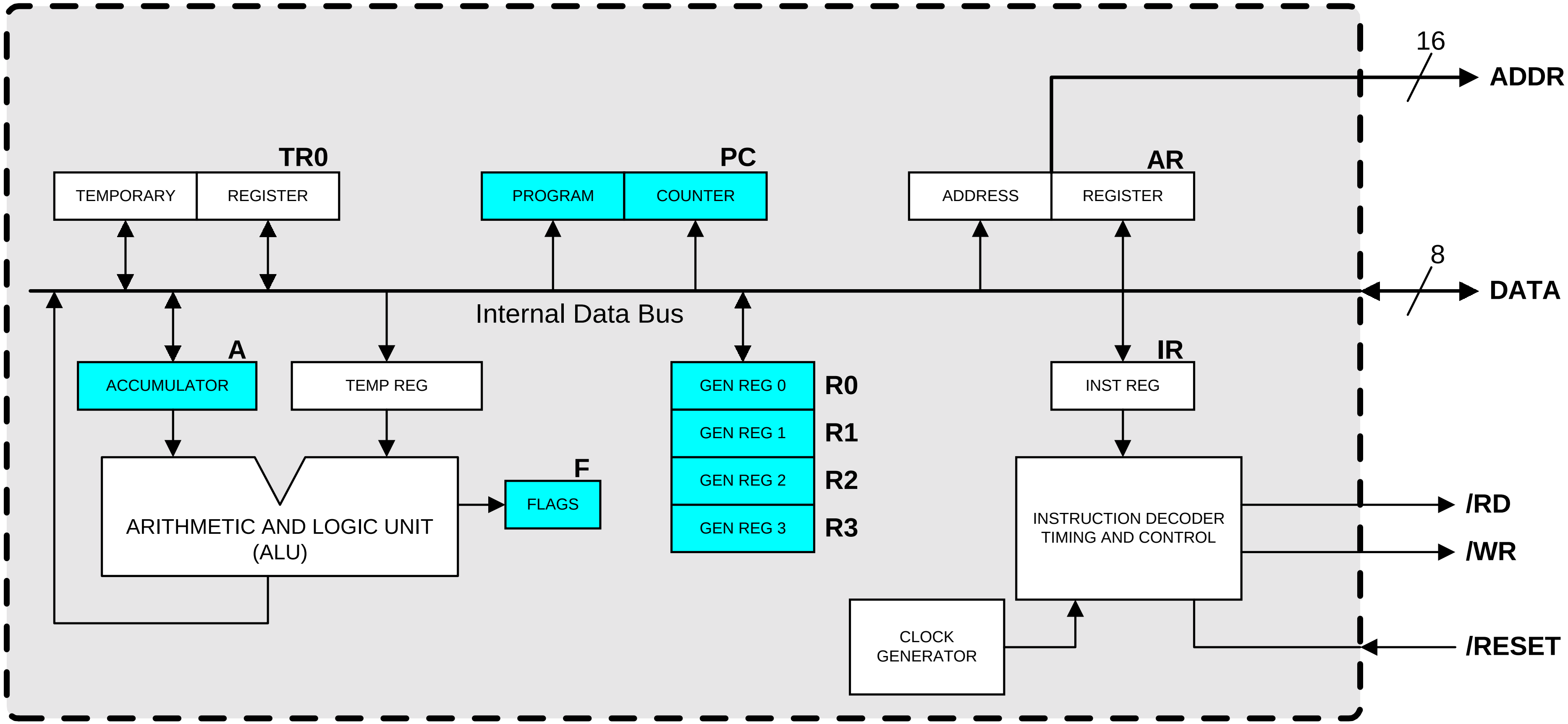
© above picture: Patterson and Hennessy

Intel Microprocessor

on

IBM PC

Simple MP Architecture



สถาปัตยกรรมของ 8088

	Internal Bus	External Bus	Memory
MP	16	16	1 MB
MP 8088	16	8	1 MB
MP 80286	16	16	16 MB
MP 80386DX	32	32	4 GB
MP 80386 SX	16	24	16 MB
MP 80486 DX	32	32	4 GB
Pentium	32/64	32/64	4 GB

Microprocessor 8 bit

1972 บริษัทอินเทล ได้ผลิตจำหน่ายออกสู่ตลาดคือ **8008** มีหน่วยความจำขนาดเวิร์ดละ 8 บิตความจุ 16 กิโลไบต์ มีคำสั่งทั้งหมด 48 คำสั่ง

1974 ได้ผลิต **8080** เป็น 8 บิตรุ่นใหม่ มีความจุ 64 กิโลไบต์ ทำงานเร็วกว่า 8008 มากกว่า 10 เท่า มีคำสั่ง 78 คำสั่ง

1977 ได้ผลิต **8085** มีความเร็วมากกว่า 8080 โดยการรวมเอาส่วนควบคุมและส่วนกำเนิดสัญญาณนาฬิกาไว้ในชิพเดียวกันและเป็นที่นิยมใช้ ผลิตชิพชนิดนี้ขายได้มากกว่า 1 ร้อยล้านชิพ

Microprocessor 16 bit

1978 : บริษัทอินเทล ได้ผลิต **8086** และปีต่อมาได้ผลิต **8088** ไมโครโปรเซสเซอร์ทั้งสองมีขนาด 16 บิต สามารถอ้างอิงหน่วยความจำได้ถึง 1 MB

1982 : **80286** เป็นการพัฒนามาจาก 8086 สามารถอ้างอิงแอดเดรสได้ 16 MB ความเร็วจาก 8 เมกกะเฮิทซ์ เป็น 16 เมกกะเฮิทซ์

Microprocessor 32 bit

1985 : บริษัทอินเทล ได้ผลิตไมโครโปรเซสเซอร์ 32 บิต คือ **80386**

1986 : **80486** มีความเร็ว 33 เมกกะเฮิทซ์ มีความจุหน่วยความจำ 4 GB

1993 : **Pentium Microprocessor** : เป็นไมโครโปรเซสเซอร์ขนาด 32/64 บิต ปัจจุบันได้พัฒนาจาก Pentium I , II , III , เป็นที่นิยมใช้งานในปัจจุบันและยังมี ไมโครโปรเซสเซอร์ที่ทำงานร่วมกันได้ (Compatible) อีกหลายบริษัท

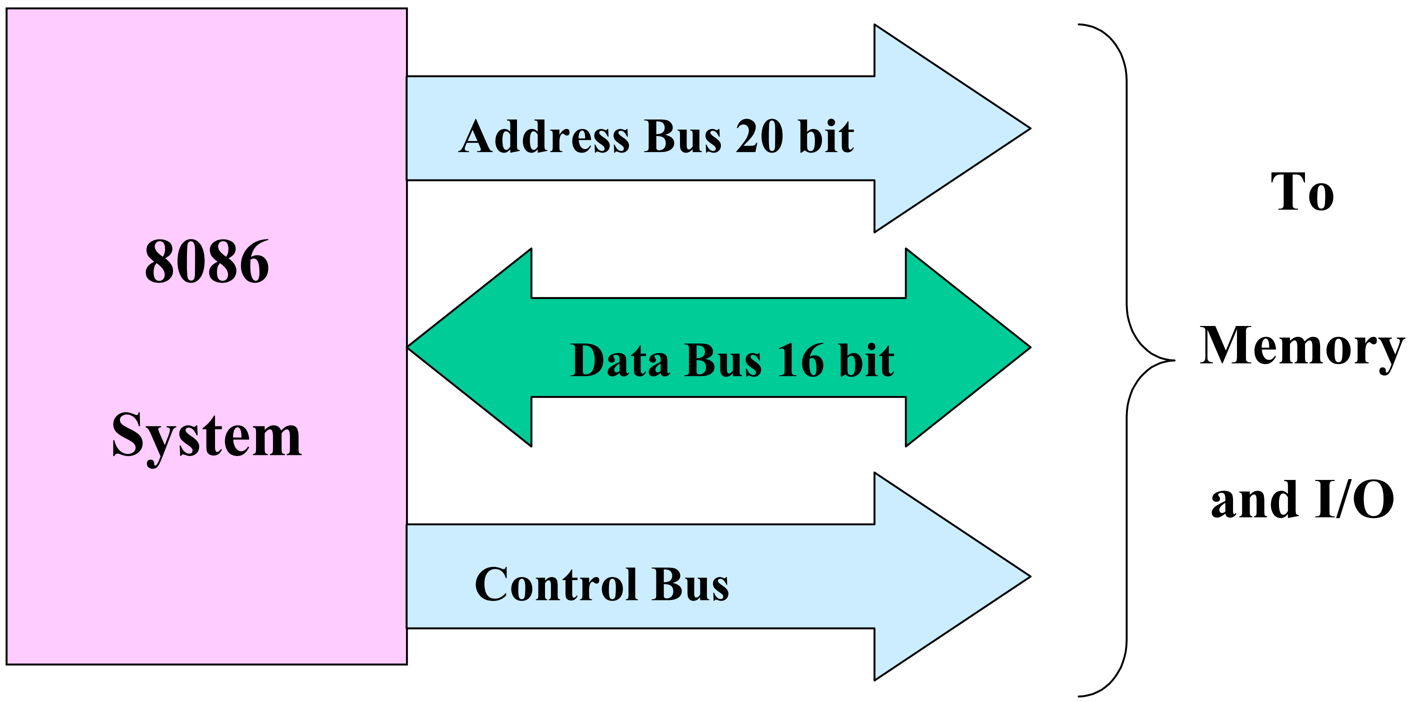
2000: **Pentium 4 Processor** based Pcs can create professional quality movies ;deliver TV-like video via communicate with real time video and voice; quick encode music for MP3 players and run multimedia applications

2001 : Intel Xeon Processor

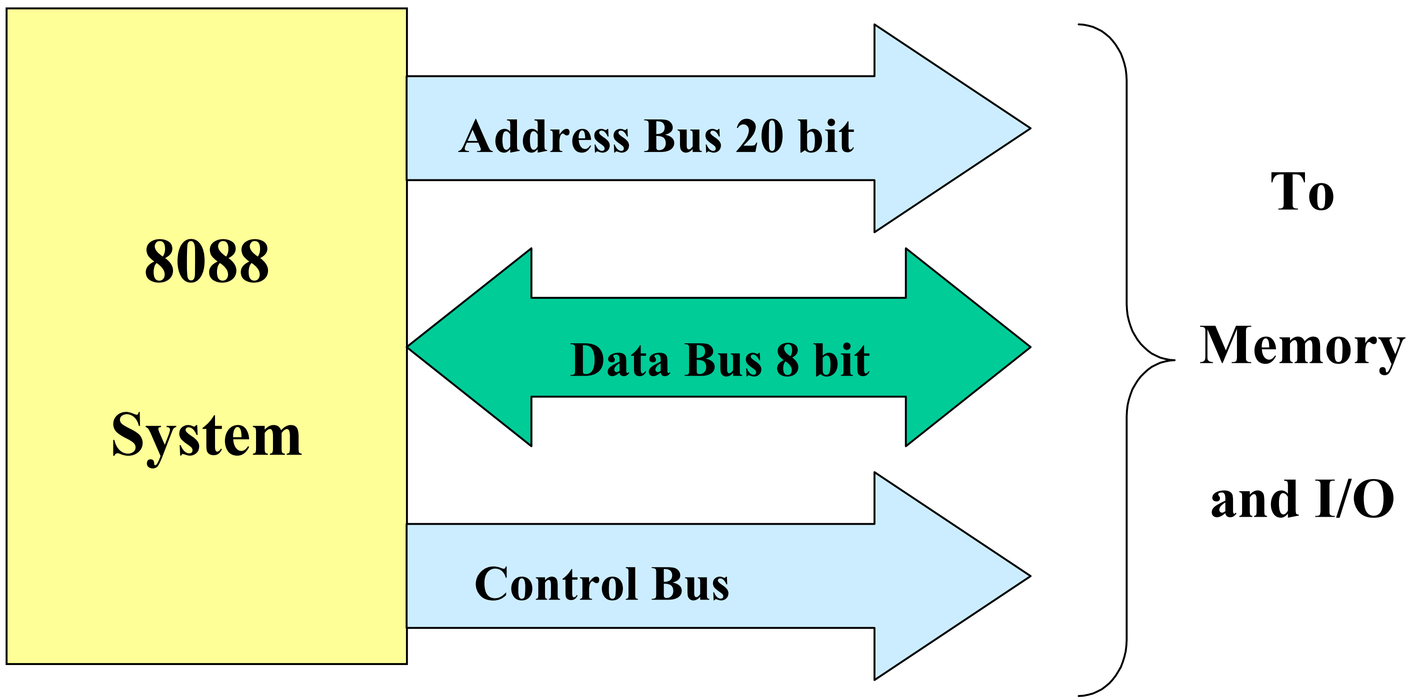
2002 : Intel Itanium Processor

designed for high end ,enterprise servers and workstations was built new architecture (parallel instruction design)

Microprocessor 8086



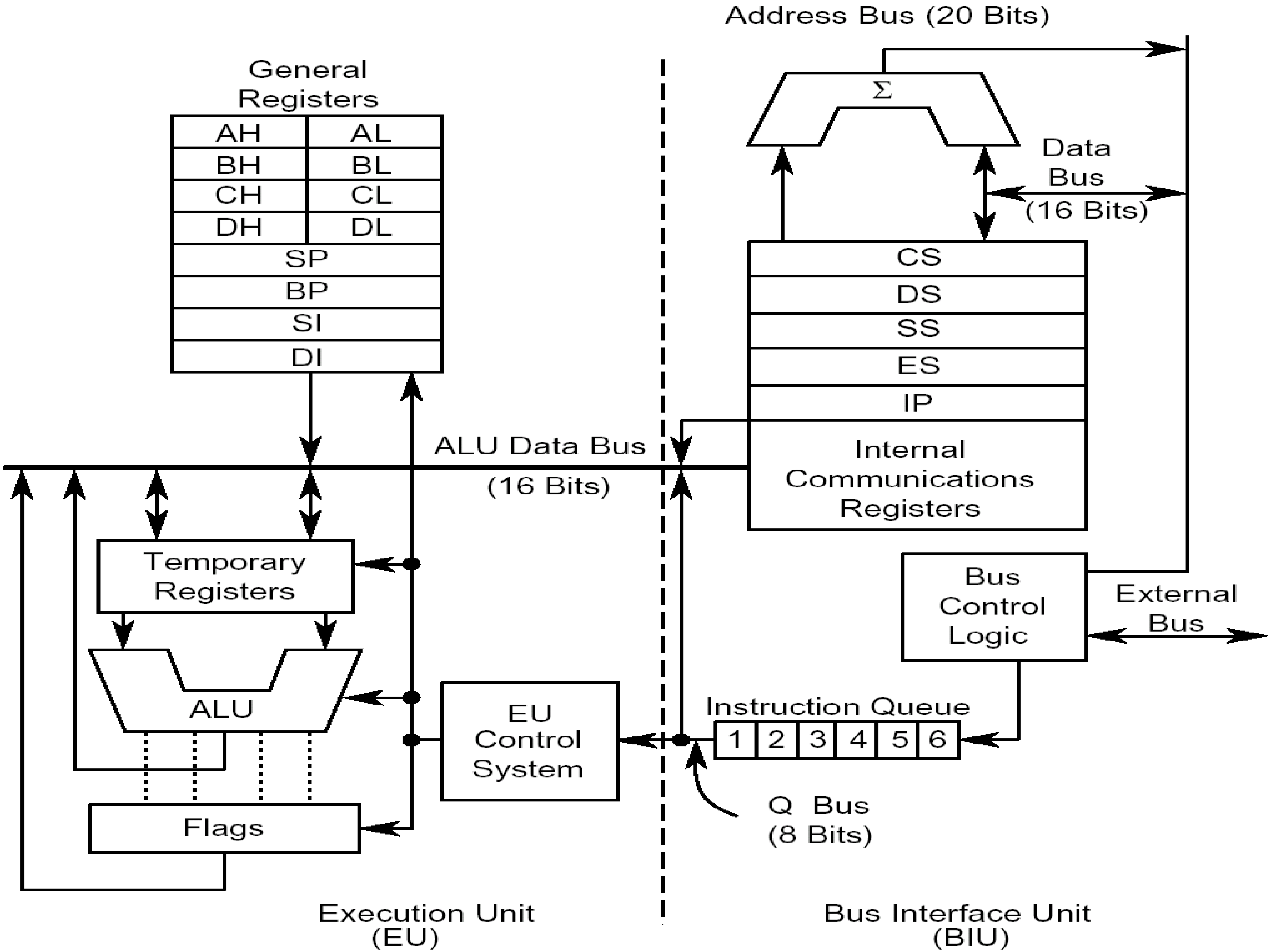
Microprocessor 8088



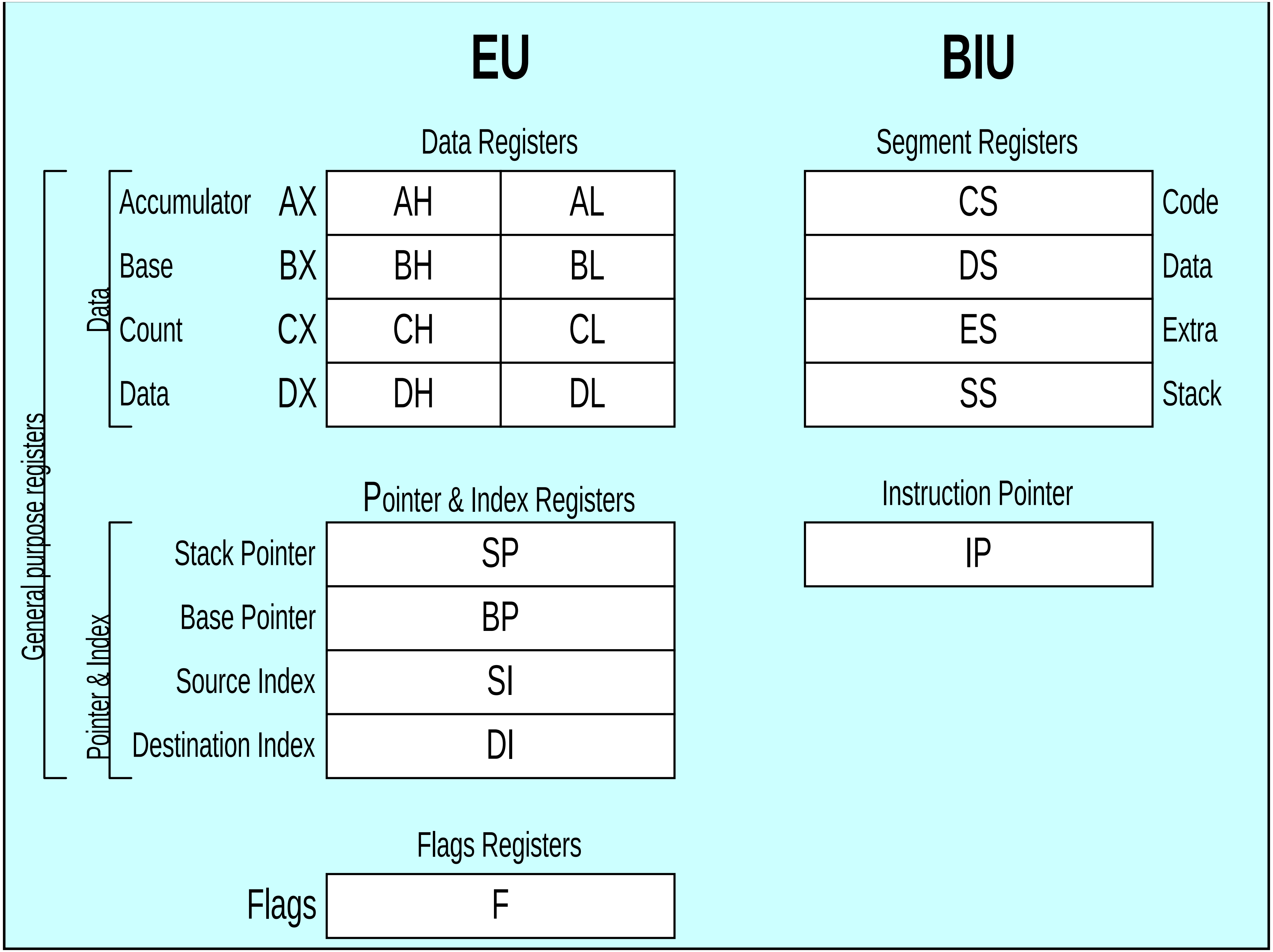
Organization of the IBM PC

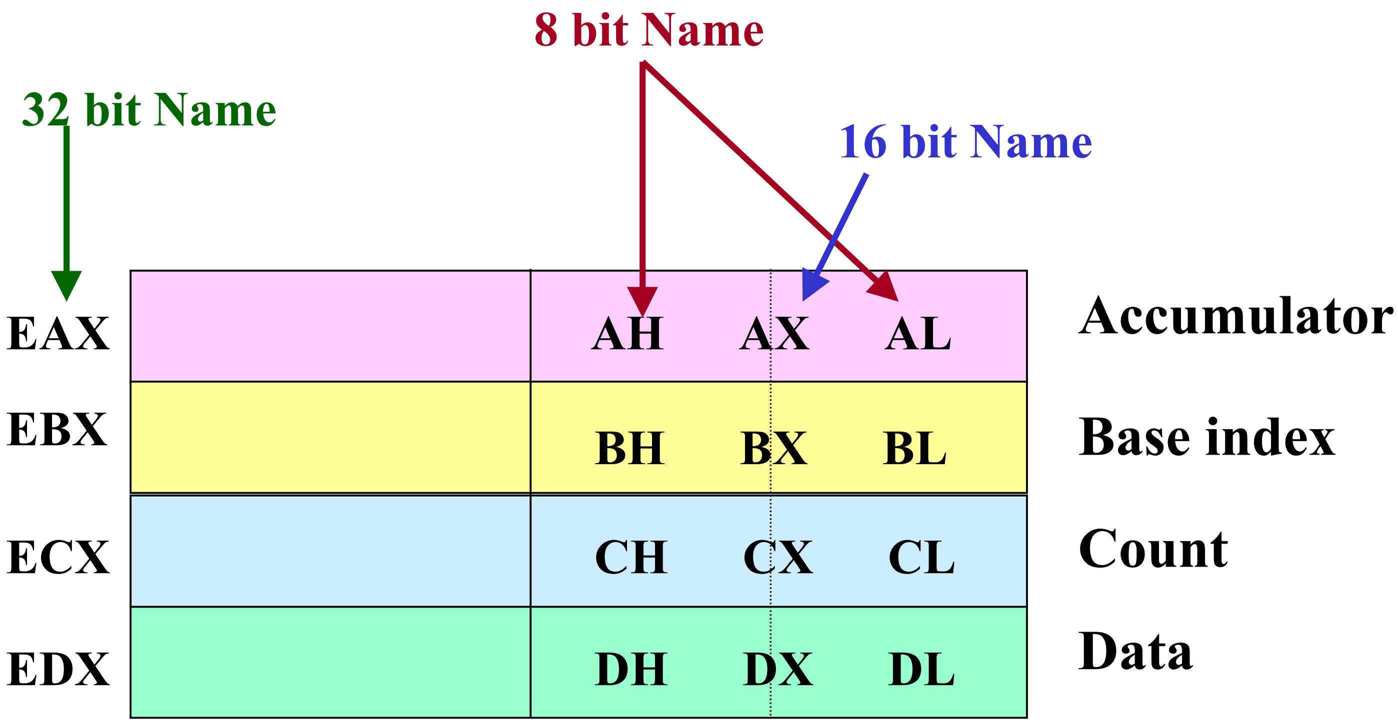
Programming Model

การเขียนโปรแกรมจำเป็นต้องทำความเข้าใจถึง โครงสร้างทางคอมพิวเตอร์ (Computer Organization) ว่าโครงสร้างประเภทใดที่นักเขียนโปรแกรมมองเห็นในคอมพิวเตอร์ที่เราต้องการศึกษาการทำงานของชุดคำสั่ง IBM PC ได้แบ่งโครงสร้างของรีจีสเตอร์ออกเป็น 3 กลุ่ม ได้แก่ รีจีสเตอร์ใช้งานทั่วไป รีจีสเตอร์ที่เป็นพอยเตอร์และอินเดกซ์ รีจีสเตอร์เซคเม้นต์ นอกจาก 3 ประเภทนี้แล้วยังมี แฟลกรีจีสเตอร์เพื่อแสดงสถานะการทำงานของซีพียู



Address Bus (20 Bits)
General Registers
AH
AL
BH
BL
CH
CL
DH
DL
SP
BP
SI
DI
Σ
Data Bus (16 Bits)
CS
DS
SS
ES
IP
Internal Communications Registers
ALU Data Bus (16 Bits)
Temporary Registers
ALU
Flags
EU Control System
Bus Control Logic
External Bus
Instruction Queue
1
2
3
4
5
6
Q Bus (8 Bits)
Execution Unit (EU)
Bus Interface Unit (BIU)


EU
BIU
Data Registers
Accumulator AX AH AL
Base BX BH BL
Count CX CH CL
Data DX DH DL
Data
General purpose registers
Pointer & Index Registers
Stack Pointer SP
Base Pointer BP
Source Index SI
Destination Index DI
Pointer & Index
Flags Registers
Flags F
Segment Registers
CS Code
DS Data
ES Extra
SS Stack
Instruction Pointer
IP



General Purpose Registers

Registers :

- Data register
- Address register
- Status register
- Index register
- Segment register

ESP		SP	Stack pointer
EBP		BP	Base pointer
EDI		DI	Destination index
ESI		SI	Source index

EIP		IP	Instruction pointer
FLAGS		FLAGS	FLAGS

Base & Index Registers

CS	**CODE**
DS	**DATA**
ES	**EXTRA**
SS	**STACK**
FS	
GS	

Note : 1. No special names are driven to the FS and GS registers.

Segment Register

EAX		AH AX AL	Accumulator
EBX		BH BX BL	Base index
ECX		CH CX CL	Count
EDX		DH DX DL	Data
ESP		SP	Stack pointer
EBP		BP	Base pointer
EDI		DI	Destination index
ESI		SI	Source index
EIP		IP	Instruction pointer
FLAGS		FLAGS	FLAGS

EU

8086 Microprocessor

There are two main components

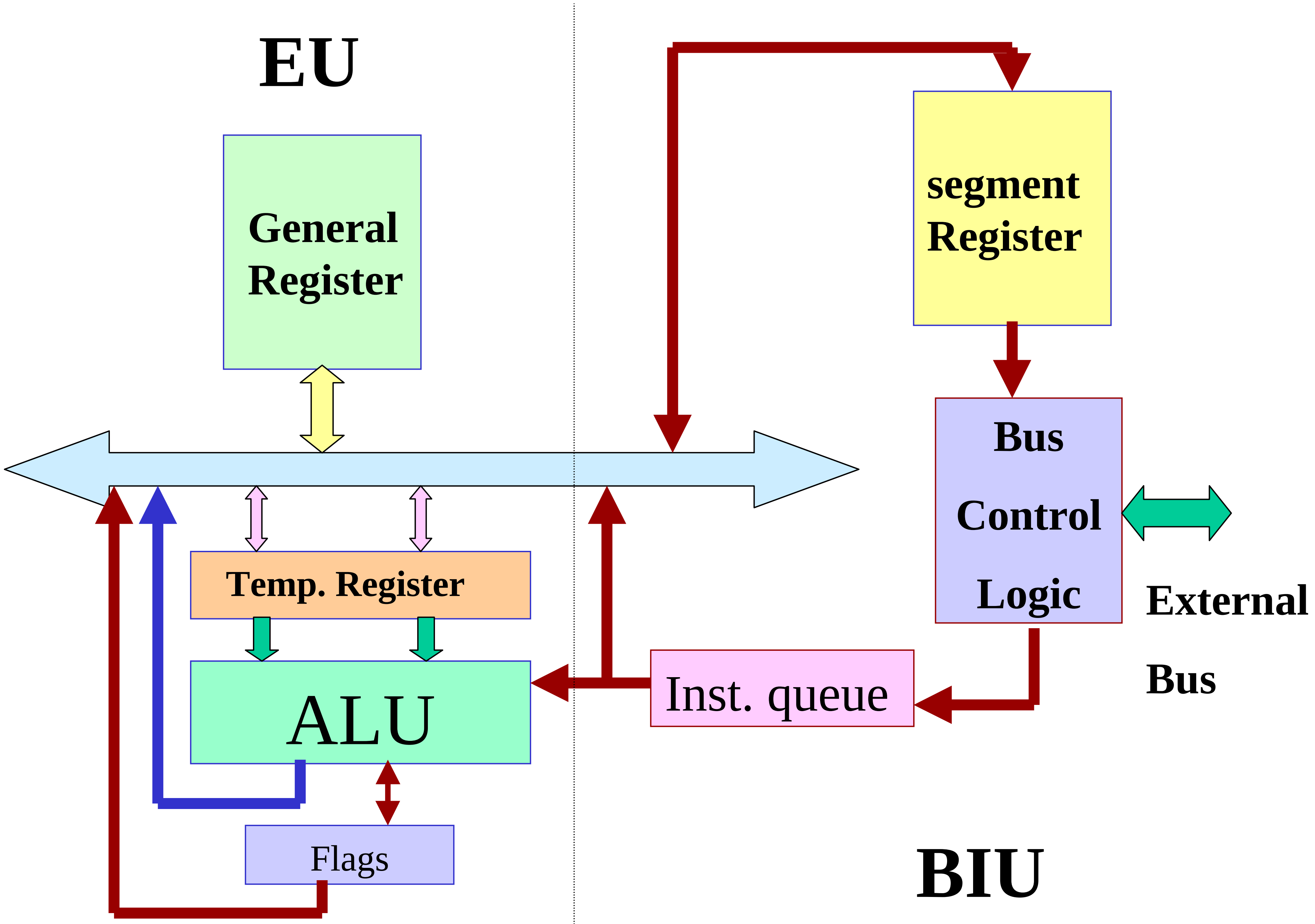
- The Execution unit
- The Bus interface unit

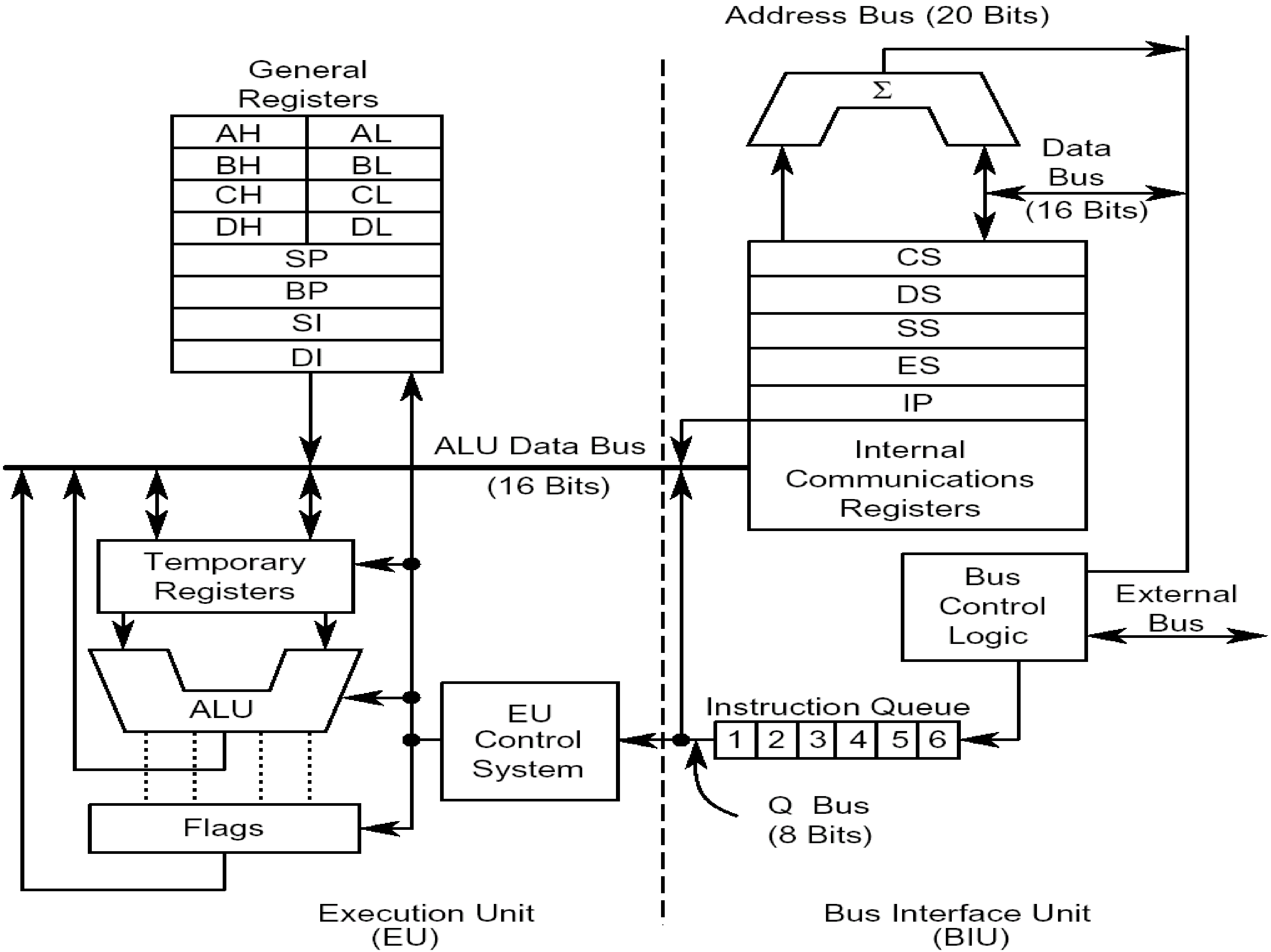
The Execution unit (EU) is to execute instructions. It contains a circuit called the arithmetic and logic unit (ALU) . The ALU can perform arithmetic (+ , - , * , /) and logic (AND , OR , NOT) The data for operations are stored in circuit called Registers A register is like a memory location except that we normally refer to it by a name rather than a number. The EU has eight registers for storing data; their names are AX , BX , CX , DX , SI , DI , BP ,SP and FLAGS register

Bus interface Unit (BIU)

Bus interface unit (BIU) facilities communication between the EU and memory or I/O circuits. It responsible for transmitting address , data , and control signals on the buses. Its registers are named CS , DS , ES , SS , IP ; they hold addresses of memory locations. The IP contains the address of next instruction to be executed by the EU.

The EU and The BIU are connected by an internal bus. And they work together. While the EU is executing and instruction, the BIU fetches up to six bytes of the next instruction and places them in the instruction queue. This operation is called ***Instruction prefetch***. The purpose is to speed up the processor .


EU
General Register
segment Register
Bus Control Logic
Temp. Register
ALU
Inst. queue
External Bus
Flags
BIU


Address Bus (20 Bits)
Σ
Data Bus (16 Bits)
General Registers
AH
AL
BH
BL
CH
CL
DH
DL
SP
BP
SI
DI
CS
DS
SS
ES
IP
Internal Communications Registers
ALU Data Bus (16 Bits)
Temporary Registers
ALU
Flags
EU Control System
Bus Control Logic
External Bus
Instruction Queue
1
2
3
4
5
6
Q Bus (8 Bits)
Execution Unit (EU)
Bus Interface Unit (BIU)

I/O Ports : I/O devices connected to the computer through I/O circuits. Each of these circuits contains several register called *I/O Ports* .Some are used for data while others are used control commands. Like memory locations , the I/O ports have address and are connected to the bus system. These address are known as I/O address and can only be use in input or output instructions.

Serial and Parallel Ports:

The data transfer between an I/O port and an I/O devices can be 1 bit at a time (serial) or 8 bit or 16 bit at a time (parallel). A parallel port requires more wiring connections, while serial port tends to slower . Slow devices , like the keyboard, always connect to a serial port, and fast devices , like the disk drive , always connect to a parallel port. But some devices like the printer , can connect to either a serial or a parallel port.

Instruction Execution :

To understand how the CPU operates, let's look at how an instruction is executed. First of all , a machine instruction has two parts: an opcode and Operands. The opcode specifics the type of operation and operands are often given as memory address to the data to be operated on. The CPU poes through the following steps to execute a machine instruction.(Fetch-Execute cycle)

Fetch:

1. Fetch an instruction from memory

2. Decode the instruction to determine the operation

3. Fetch data from memory necessary.

Execute:

1. Perform the operation on the data

2. Store the result in memory if needed.

I/O Devices:

- Magnetic disk (Hard disk, Floppy disks)
- Keyboard
- Display monitor
- Printers

Programming Languages:

The operations of computer's hardware are controlled by its software. When the computer is on, it is always in the process of executing instructions. To fully understand the computer's operation, we must also study instructions.

Machine Language :

A CPU can only execute machine language instructions. As we've seen, they are bit strings. The following is a short machine language program for IBM PC.

Machine instruction	Operation
10100001 00000000 00000000	Fetch the content of memory word 0 and put it in Register AX
00000101 00000100 00000000	Add 4 to AX
10100011 00000000 00000000	Store AX in memory word 0.

Assembly Language :

More convenient language to use is assembly language . In assembly language, we use symbolic names to represent operations, registers, and memory locations. If location 0 is symbolized by A , The preceding program expressed in IBM PC assembly language would look like this:

Assembly Language		**Comment**
MOVE	AX,A	; Fetch content of location A and put it in AX
ADD	AX,4	; Add 4 to AX
MOV	A,AX	;move content Ax to location A

High Level Languages :

<u>Advantages</u>:

• Closer to natural language , It's easier to convert a natural language algorithm to a high level language program than to an assembly language program.

• An assembly language program generally contains more statements than an equivalent high level language program , so more time is needed to code the assembly language program.

• Because each computer has its own unique assembly language, Assembly language program can be executed on any machine that has a compiler for the language.

Advantage of Assembly languages:

- Assembly language is close to machine language.
- Faster
- Some operation, such as reading or writing to specific memory locations and I/O ports can be done easily
- ASM is necessary for subprograms in high level language.

Basic Features

of

PC Hardware

Objective

- Information Representation
- PC Computer Hardware Organization

Information Representation

- Size
  - Bits and Bytes
- ASCII Code
- Numerical code
  - Unsigned Binary System
  - Signed and Magnitude System
  - 1's Complement System
  - 2's Complement System
  - Hexadecimal System

Information Representation

Computers use binary number system to store information as 0’s and 1’s

Bits

- A *bit* is the fundamental unit of computer storage
- A bit can be 0 (off) or 1 (on)
- Related bits are grouped to represent different types of information such as numbers, characters, pictures, sound, instructions

Bytes

Bytes

- A *byte* is a group of 8 bits that is used to represent numbers and characters
- An additional bit is for parity check (error checking for storage/transmission)
- A byte consists of 8 data bits and 1 parity bit
- A standard code for representing numbers and characters is ASCII (**A**merican **S**tandard **C**ode for **I**nformation **I**nterchange)

Byte Size

Bytes

- How many different combinations of 0's and 1's with 8 bits can form?
- In general, how many different combinations of 0's and 1's with N bits can form?
- How many different characters that a byte (8 bits) can represent?

Related Bytes

– A ***nibble*** is a half-byte (4-bit) - hex representation
– A ***word*** is a 2-byte (16-bit) data item
– A ***doubleword*** is a 4-byte (32-bit) data item
– A ***quadword*** is an 8-byte (64-bit) data item
– A ***paragraph*** is a 16-byte (128-bit) area
– A ***kilobyte*** (KB) is 2^{10} = 1,024 bytes (K bytes)
– A ***megabyte*** (MB) is 2^{20} = 1,048,576 ? 1 MB
– A ***Gigabyte*** (GB) is 2^{30} = 1,073,741,824 ? 1 GB

Representation Codes

- ASCII code
- Numerical codes
  - Unsigned binary code
  - Signed binary code
  - Hexadecimal notation

ASCII Code

– **ASCII: American Standard Code for Information Interchange.**
– **Used to represent characters and textual information**
– **Each character is represented with 1 byte**
  - **upper and lower case letters: a..z and A..Z**
  - **decimal digits -- 0,1,…,9**
  - **punctuation characters -- ; , . :**
  - **special characters --$ & @ / {**
  - **control characters -- carriage return (CR) , line feed (LF), beep**

Examples of ASCII Code

Bit contents (S):	**01010011**
Bit position:	**76543210**

S ⟷ 83 (binary) , 53 (hex)

Bit contents (8):	**00111000**
Bit position:	**76543210**

8 ⟷ 56 (binary) , 38 (hex)

ASCII Code in Binary and Hex

Character	Binary	Hex
A	0100 0001	41
D	0100 0100	44
a	0110 0001	61
?	0011 1111	3F
2	0011 0010	32
DEL	0111 1111	7F

Numerical Codes

- Unsigned number system
- Signed and magnitude system
- 1's complement system
- 2's complement system
- Hexadecimal system

Binary Number System

- base 10 -- has ten digits: 0,1,2,3,4,5,6,7,8,9
  - positional notation

$$2401 = 2 ? 10^3 + 4 ? 10^2 + 0 ? 10^1 + 1 ? 10^0$$

- base 2 -- has two digits: 0 and 1
  - positional notation

$$1101_2 = 1 ? 2^3 + 1 ? 2^2 + 0 ? 2^1 + 1 ? 2^0$$

$$= 8 + 4 + 0 + 1 = 13$$

Binary Positional Notation

If

$$N = b_{n-1} b_{n-2} ??? b_1 b_0$$

then

$$N = b_{n-1} ? 2^{n-1} + b_{n-2} ? 2^{n-2} + ??? + b_0 ? 2^0$$

Unsigned Binary Code

Use for representing integers without signed (natural numbers)

0	0000	8	1000
1	0001	9	1001
2	0010	10	1010
3	0011	11	1011
4	0100	12	1100
5	0101	13	1101
6	0110	14	1110
7	0111	15	1111

Number of Bits Required in Unsigned Binary Code

- What is the range of values that can be represented with *n* bits in the Unsigned Binary Code?

$$[0, 2^n\text{-}1]$$

- How many bits are required to represent a given number N in decimal?

Min. Number of Bits = $\mathbf{\log_2(N+1)}$

Unsigned Conversion

- Convert an unsigned binary number to decimal

  use positional notation (polynomial expansion)

- Convert a decimal number to unsigned Binary

  use successive division by 2

Examples

- Represent 26_{10} in unsigned Binary Code

$$26_{10} = 11010_2$$

- Represent 26_{10} in unsigned Binary Code using 8 bits

$$26_{10} = 00011010_2$$

- Represent $(26)_{10}$ in Unsigned Binary Code using 4 bits -- not possible

Signed Binary Codes

These are codes used to represent positive and negative numbers.

- Signed and Magnitude System
- 1’s Complement System
- 2’s Complement System

Signed and Magnitude

- The most significant (left most) bit represent the sign bit
  - 0 is positive
  - 1 is negative
- The remaining bits represent the magnitude

Examples of Signed & Magnitude

Decimal	5-bit Sign and Magnitude
+5	00101
-5	10101
+13	01101
-13	11101

Signed and Magnitude in 4 bits

0	0000	-0	1000
1	0001	-1	1001
2	0010	-2	1010
3	0011	-3	1011
4	0100	-4	1100
5	0101	-5	1101
6	0110	-6	1110
7	0111	-7	1111

Examples

Decimal	Signed	8-bit Signed
26_{10}	011010_2	00011010_2
-26_{10}	111010_2	10011010_2

1's Complement System

- Positive numbers:
  - same as in unsigned binary system
  - pad a 0 at the leftmost bit position
- Negative numbers:
  - convert the magnitude to unsigned binary system
  - pad a 0 at the leftmost bit position
  - complement every bit

Examples of 1's Complement

Decimal	5-bit 1's complement
5	00101
-5	11010
13	01101
-13	10010

1's Complement in 4 bits

0	0000	-0	1111
1	0001	-1	1110
2	0010	-2	1101
3	0011	-3	1100
4	0100	-4	1011
5	0101	-5	1010
6	0110	-6	1001
7	0111	-7	1000

Examples

Decimal	Signed	8-bit Signed
26_{10}	011010_2	00011010_2
-26_{10}	100101_2	11100101_2

2’s Complement System

- Positive numbers:
  - same as in unsigned binary system
  - pad a 0 at the leftmost bit position
- Negative numbers:
  - convert the magnitude to unsigned binary system
  - pad a 0 at the leftmost bit position
  - complement every bit
  - add 1 to the complement number

Examples of 2's Complement

Decimal	5-bit 2's complement
5	00101
-5	11011
13	01101
-13	10011

2’s Complement in 4 bits

0	0000	-1	1111
1	0001	-2	1110
2	0010	-3	1101
3	0011	-4	1100
4	0100	-5	1011
5	0101	-6	1010
6	0110	-7	1001
7	0111	-8	1000

Examples

Decimal	Signed	8-bit Signed
26_{10}	011010_2	00011010_2
-26_{10}	100110_2	11100110_2

More Examples

- Represent 65 in 2's complement

  $65 = 0100\ 0001_2$

- Represent -65 in 2's complement

  $-65 = 1011\ 1111_2$

Convert 2's Complement to decimal

Positive 2's complement numbers

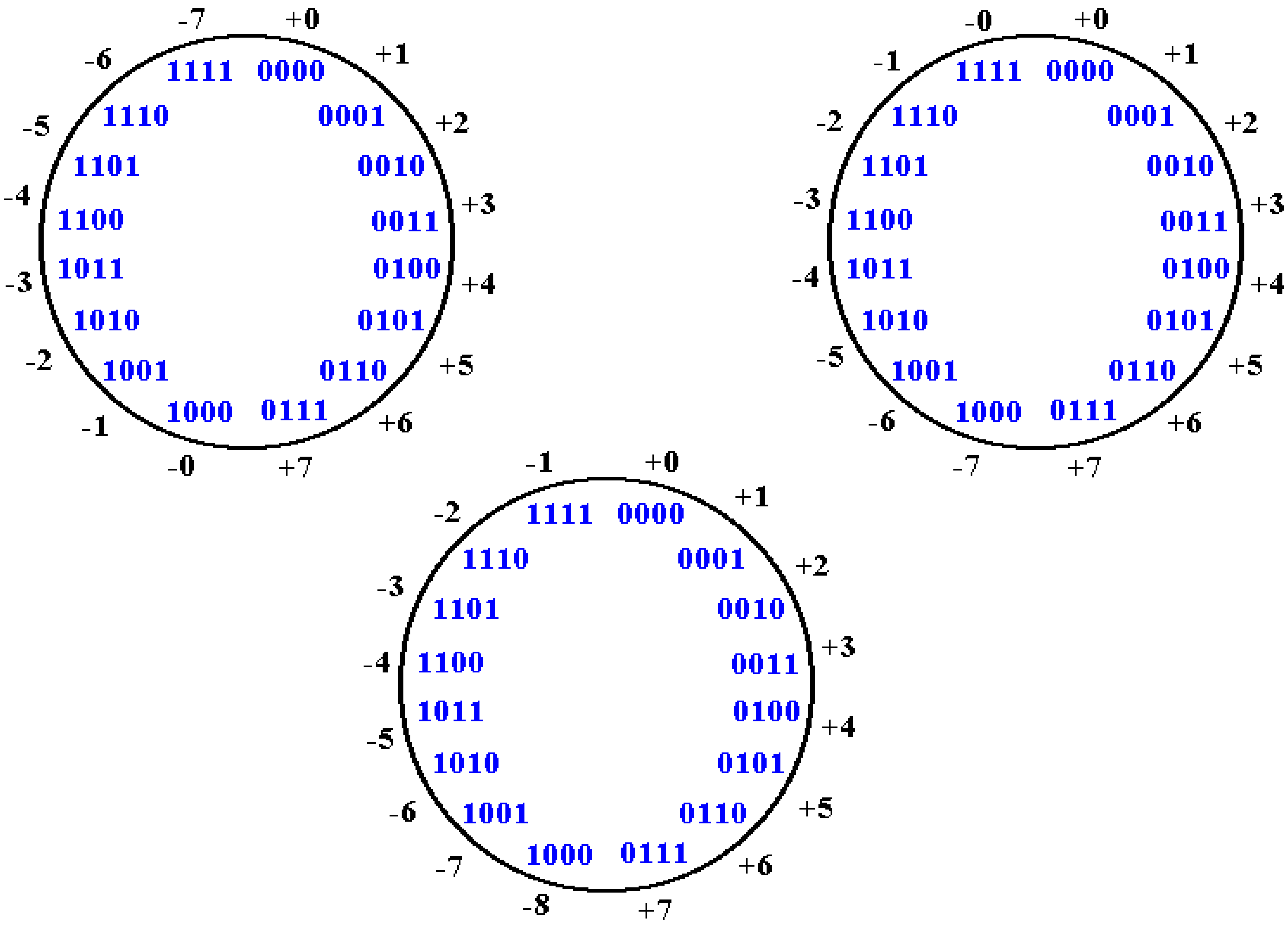
- convert the same as in unsigned binary

Negative 2's complement numbers

- complement the 2's complement number
- add 1 to the complemented number
- convert the same as in unsigned binary

Examples

2's complement	Decimal
00101	4 + 1 = 5
11011 → 00100 + 1	4 + 1 = 5 → -5
01101	8 + 4 + 1 = 13
10011 → 01100 + 1	8 + 4 + 1 = 13 → -13



-7 +0
-6 1111 0000 +1
-5 1110 0001 +2
1101 0010
-4 1100 0011 +3
-3 1011 0100 +4
1010 0101
-2 1001 0110 +5
-1 1000 0111 +6
-0 +7
-0 +0
-1 1111 0000 +1
-2 1110 0001 +2
1101 0010
-3 1100 0011 +3
-4 1011 0100 +4
1010 0101
-5 1001 0110 +5
-6 1000 0111 +6
-7 +7
-1 +0
-2 1111 0000 +1
1110 0001 +2
-3 1101 0010
-4 1100 0011 +3
1011 0100 +4
-5 1010 0101
-6 1001 0110 +5
-7 1000 0111 +6
-8 +7

Mathematical Formula

- **Formula to convert a decimal number to a 1's complement --**

$$N' = 2^n - N - 1$$

- **Formula to convert a decimal number to a 2's complement --**

$$N' = 2^n - N$$

where *N* is the binary number representing the decimal with *n* number of bits

Hexadecimal Notation

- base 16 -- has 16 digits:

  0 1 2 3 4 5 6 7 8 9 A B C D E F

- each Hex digit represents a group of 4 bits (i.e. half of a byte)   0000 to 1111
- use as a shorthand notation for convenient

Convert Binary ⟷ Hex

Binary	Hex
1111 0110b	F6h
1001 1101 0000 1010b	9D0Ah
1111 0110 1110 0111b	F6E7h
1011011b	5Bh

Examples

– ASCII value of character ‘D’ in Hex

$$D = 0100\ 0100b_{\mathbf{ASCII}} = 44h_{\mathbf{ASCII}}$$

– Represent 37D in 2’s complement using Hex.

$$37d = 010\ 0101b_{\mathbf{2's}} = 0010\ 0101b_{\mathbf{2's}}$$

$$= 25h_{\mathbf{2's}}$$

– Represent -37d in 2’s complement using Hex.

$$-37d = 101\ 1011b_{\mathbf{2's}} = 1101\ 1011b_{\mathbf{2's}} = DBh_{\mathbf{2's}}$$

Convert Hex $\longleftrightarrow$ Decimal

- Convert Hex to decimal
  - use positional (polynomial expansion) notation

3BAh $= 3 * 16^2 + B * 16^1 + A * 16^0$

$= 3 * 256 + 11 * 16 + 10 * 1 = 954$d

- Convert decimal to Hex
  - Use successive divisions by 16

359/16 $\Rightarrow$ 22 R 7, 22/16 $\Rightarrow$ 1 R 6, 1/16 $\Rightarrow$ 0 R 1

359d = 167h

Covert Large Binary to Decimal

Convert 1001 0011 0101 1100b to decimal

Method 1:

- Use polynomial expansion methods

Method 2:

- Convert number to hex, then convert it to decimal.

1001 0011 0101 1100b = 935Ch

935Ch = 37724d

Addition and Subtraction in Signed and Magnitude

(a)	5	0101
	+2	+0010
	7	0111
(b)	-5	1101
	-2	+1010
	-7	1111
(c)	5	0101
	-2	+1010
	3	0011
(d)	-5	1101
	+2	+0010
	-3	1011

Addition and Subtraction in 1's Complement

(a)

```
   5      0101
  +2     +0010
 ----    -----
   7      0111
```

(b)

```
  -5       1010
  -2      +1101
 ----    ------
  -7     1 0111
              1
         ------
           1000
```

(c)

```
   5       0101
  -2      +1101
 ----    ------
   3     1 0010
              1
         ------
           0011
```

(d)

```
  -5       1010
  +2      +0010
 ----    ------
  -3       1100
```

Addition and Subtraction in 2's Complement

	Decimal	Binary
(a)	5	0101
	+2	+0010
	7	0111
(b)	-5	1011
	-2	+1110
	-7	1 1001
(c)	5	0101
	-2	+1110
	3	1 0011
(d)	-5	1011
	+2	+0010
	-3	1101

Theoretical Facts

- Why is the carry out from adding 1’s complements added to the sum?
- Why is the carry out from adding 2’s complements dropped?

Overflow Conditions

Carry-in ? carry-out

```
      0111                  1000
 5    0101            -5    1011
+3  + 0011            -4   +1100
-8    1000             7   10111
```

Carry-in = carry-out

```
      0000                  1110
+5    0101            -2    1110
+2   +0010            -6   +1010
 7    0111            -8   11000
```

Addition and Subtraction in Hexadecimal System

Addition

$(9F1B)_{16} + (4A36)_{16}$:

$$\begin{array}{r} \scriptstyle 1\ 1\ \ \ \\ 9F1B \\ +\ 4A36 \\ \hline E951 \end{array}$$

Subtraction

$(9F1B)_{16} - (4A36)_{16}$:

$$\begin{array}{r} \scriptstyle 16\ \ \\ 9F1B \\ -\ 4A36 \\ \hline 54E5 \end{array}$$