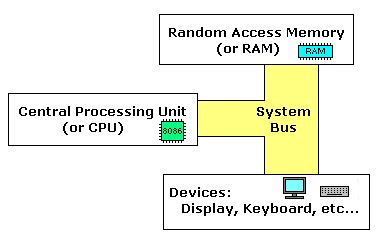
**Microcomputer Architecture**

A computer system has three main components: a Central Processing Unit (CPU) or processor, a Memory Unit, and Input Output Units (devices). In any microcomputer system, the component which actually processes data is entirely contained on a single chip called Microprocessor unit (MPU). This MPU can be programmed using assembly language. Writing a program in assembly language requires a knowledge of the computer hardware (or Architecture) and the details of its instruction set.

The main internal hardware features of a computer are the processor, memory and registers (registers are special processor components for holding address and data).

The external hardware features are the computer Input/Output

devices such as keyboard, monitor…

Software consists of the operating system (O.S) and various programs and data files stored on disk, Figure (1) show how a compute component work together.

Figure(1)

A **MICROPROCESSOR** is a multipurpose programmable logic device that reads binary instructions from a storage device called **memory** accepts binary data as input and processes data according to those instructions and provides results as input.

Intel introduced its first 4-bit microprocessor 4004 in 1971 and 8-bit microprocessor 8008 in 1972. These microprocessors could not survive as general purpose microprocessors due to their design and performance limitations. Launching of the first general purpose 8-bit microprocessor 8080 in 1974 by Intel is considered to be the first major stepping stone towards the development of advanced microprocessors. The microprocessor 8085 followed 8080, with a few more features added to its architecture, which resulted in a functionally complete microprocessor. The **main limitations** of the 8-bit microprocessors were their l**ow speed of execution,** **low memory addressing capability**, limited **number of general-purpose registers** and a **less powerful instruction set**. All these limitations of the 8-bit microprocessors tempted the designers to go for more powerful processors in terms of advanced architecture, more processing capability, larger memory addressing capability and a more powerful instruction set. The 8086 was a result of such developmental design efforts.

In the family of 16-bit microprocessors, Intel's 8086 was the first one launched in 1978. The introduction of the 16-bit processor was a result of the **increasing** demand for more and more **powerful** and **high speed computational resources.** 8086 microprocessor has a much more **powerful instruction set along** with the architectural developments which imparted substantial **programming flexibility** and improvement in **speed over the 8-bit** microprocessors.

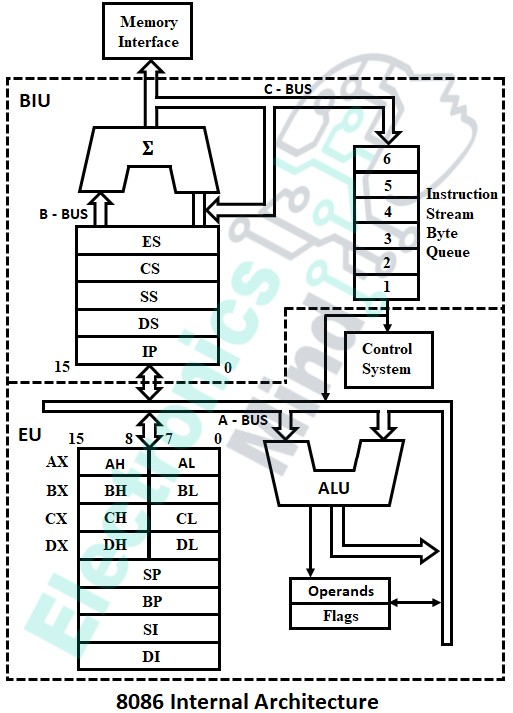
The architecture of 8086 provides a number of improvements over 8085 architecture. It supports a 16-bit ALU, a set of 16-bit registers and provides segmented memory addressing capability, a rich instruction set, powerful interrupt structure, fetched instruction queue for overlapped fetching and execution etc.

1. **CPU:Bus Interface Unit and Execution Unit**

The complete architecture of 8086 can be divided into two parts (a) **Bus Interface Unit (BIU)** and (b) **Execution Unit (EU).**

The bus interface unit contains the **circuit for physical address calculations** and a **predecoding instruction byte queue (6 bytes long).** The bus interface unit makes the system bus signals available for **external interfacing** of the devices. In other words, this unit **is responsible for establishing communications with external devices and peripherals including memory via the bus**. As already stated, the 8086 addresses a segmented memory. The complete physical address which is 20-bits long is generated using segment and offset registers, each 16-bits long.

The internal function of 8086 processor are partitioned logically into processing units, Bus Interface Unit(BIU) and Execution Unit (EU). general block diagram of 8086 processor is shown in figure (2).

Figure(2)

**Execution Unit (EU) :** Execution unit receives program instruction codes and data from the BIU, executes them and stores The results in the general registers. It can also store the data in a memory location or send them to an I/O device by passing the data back to the BIU. This unit, EU, has no connection with the system Buses. It receives and outputs all its data through BIU.

**Bus Interface Unit :** As the EU has no connection with the system Busses, this job is done by BIU. BIU and EU are connected with an internal bus. BIU connects EU with the memory or I/O circuits. It is responsible for transmitting data, addresses and control signal on the busses.

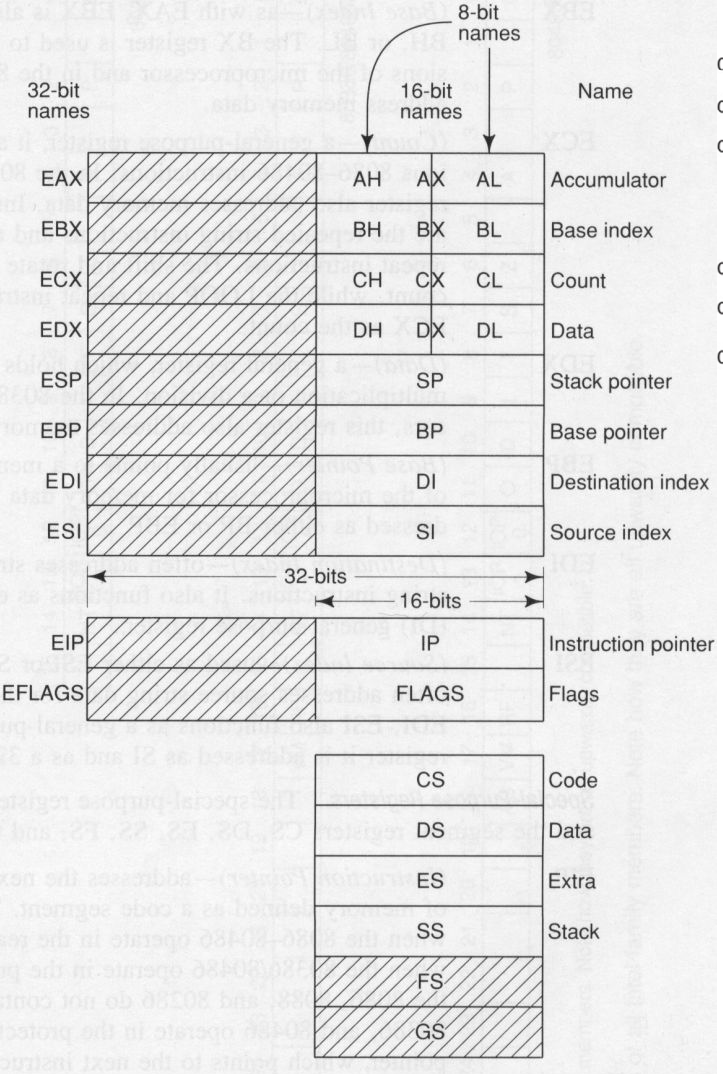
EU is used mainly to execute instructions. It contains a circuit called the arithmetic and logic unit (ALU). The data for operations are stored in circuit called Registers. The EU has eight registers for storing data; their names are AX, BX, CX, DX, SI, DI, BP, SP and FLAGS register. The EU accepts instructions and data that have been fetched by the BIU and then processes the information. Data processed by the EU can be transmitted to the memory or peripheral devices through the BIU. EU has no direct connection with the outside world and relies solely on the BIU to feed it with instruction and data. It is here that instructions are received, decoded, and executed from the instruction queue portion of BIU. The instructions are taken from the top of the instruction queue on the first-in, first-out, or FIFO, basis.

**ALU (Arithmetic & Logic Unit) :** This unit can perform various arithmetic and logical operation, if required, based on the instruction to be executed. It can perform arithmetical operations, such as add, subtract, increment, decrement, convert byte/word and compare etc and logical operations, such as AND, OR, exclusive OR, shift/rotate and test etc.

**Registers**

Registers are 8, 16, or 32-bit high speed storage locations directly

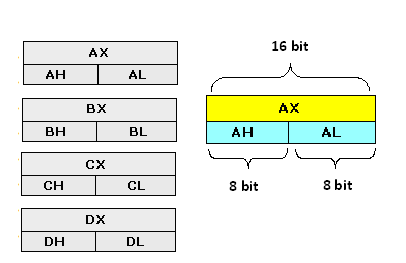
inside the CPU, figure (5) designed to be accessed at much higher speed than conventional memory. The CPU has an internal data bus that is generally twice as wide as its external data bus.



**Figure(3): Registers 8086 blockdaigram**

* **General Registers:**

The general purpose registers, are used for arithmetic and data movement. Each register can be addressed as either 16-bit or 8 bit value. Example, **AX** register is a 16-bit register, its upper 8-bit is called **AH**, and its lower 8-bit is called **AL**. Bit 0 in AL corresponds to bit 0 in AX and bit 0 in AH corresponds to bit 8 in AX. See Figure (6).



Figure(6): General Registers of 8086 MPU

All general registers of the 8086 microprocessor can be used for arithmetic and logic operations. The general registers are:

1. **AX (Accumulator):** This is accumulator register. It gets used in arithmetic, logic and data transfer instructions. In manipulation and division, one of the numbers involved must be in AX or AL.
2. **BX (Base Register):** This is base register. BX register is an address register. It usually contain a data pointer used for based, based indexed or register indirect addressing.
3. **CX (Count register):** This is Count register. This serves as a loop counter. Program loop constructions are facilitated by it. Count register can also be used as a counter in string manipulation and shift/rotate instruction.
4. **DX (Data Register):** This is data register. Data register can be used as a port number in I/O operations. It is also used in multiplication and division.

* **Index Registers**

1. **SP (Stack Pointer):** This is stack pointer register pointing to program stack. It is used in conjunction with SS for accessing the stack segment.
2. **BP (Base Pointer):** This is base pointer register pointing to data in stack segment. Unlike SP, we can use BP to access data in the other segments.
3. **SI (Source Index):** This is source index register which is used to point to memory locations in the data segment addressed by DS. By incrementing the contents of SI one can easily access consecutive memory locations.
4. **DI (Destination Index):** This is destination index register performs the same function as SI. There is a class of instructions called string operations.

* **Segment Registers :** BIU has 4 segment busses, CS, DS, SS, ES. These all 4 segment registers holds the addresses of instructions and data in memory. These values are used by the processor to access memory locations. It also contains 1 pointer register IP. IP contains the address of the next instruction to execute by the EU.

**1- CS (Code Segment):** The code segment register holds the base location of all executable instructions (code) in a program.

**2- DS (Data Segment):** the data segment register is the default base location for variables. The CPU calculates their location using the segment value in DS.

**3- SS (Stack Segment):** the stack segment register contain the base location of the stack.

**4- ES (Extra Segment):** The extra segment register is an additional base location for memory variables.

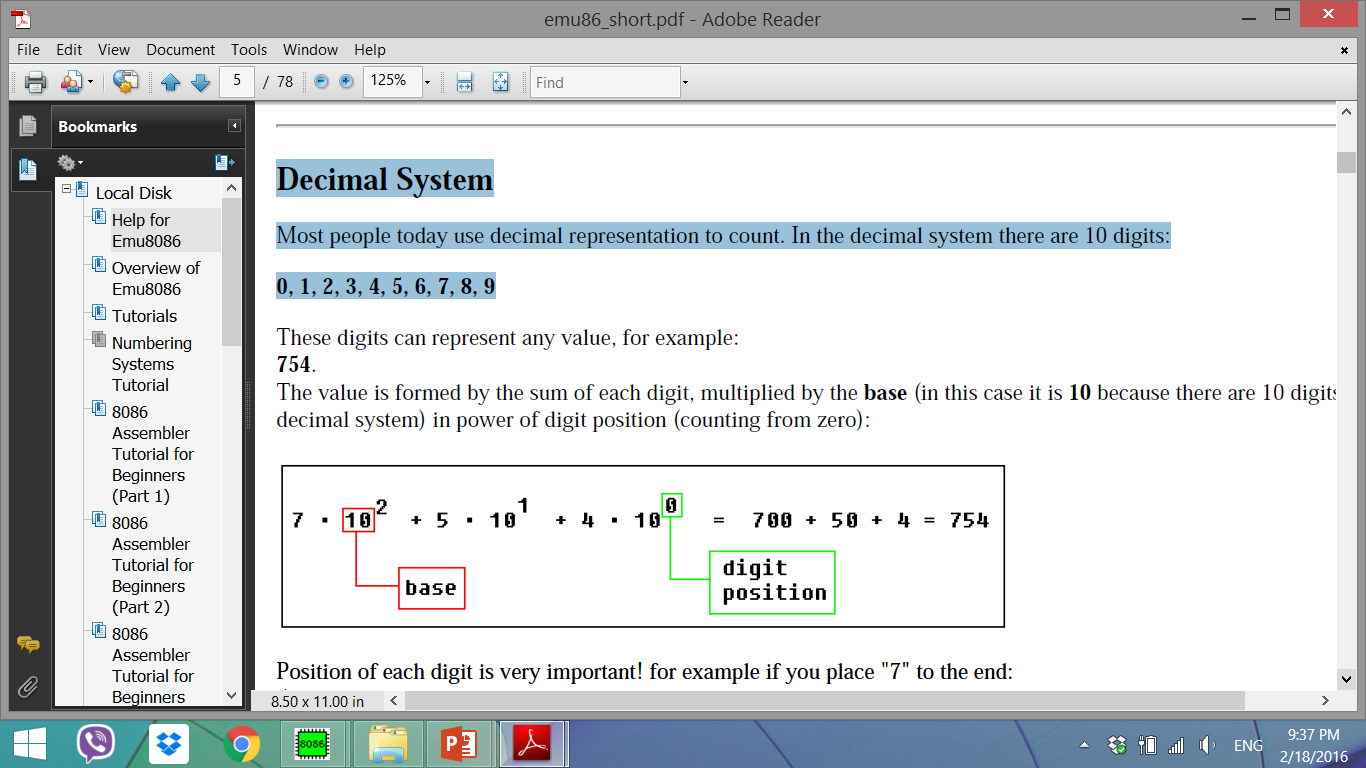
**Numbering System**

* **Decimal System**

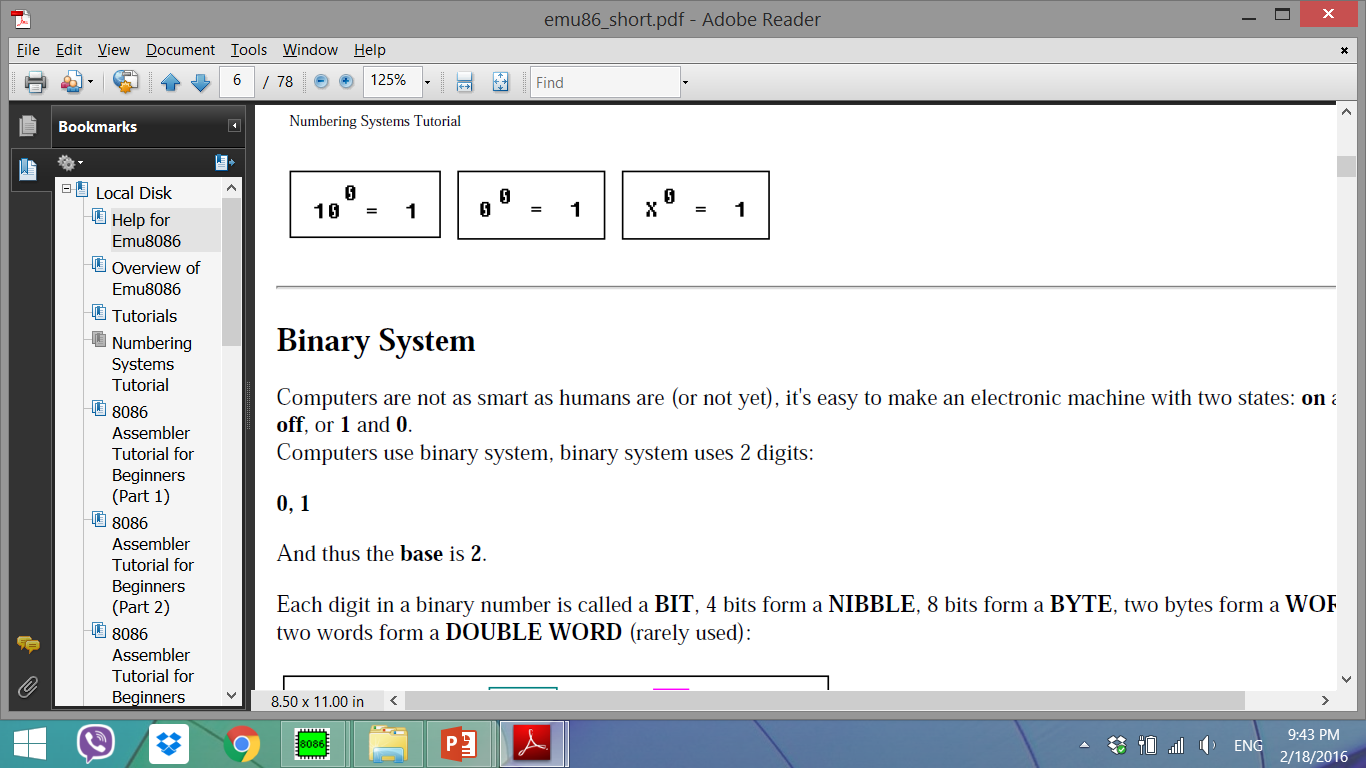
In the decimal system there are **10 digits**:

**0, 1, 2, 3, 4, 5, 6, 7, 8, 9**

**EX: 754**



**Important note:** any number in power of zero is 1, even zero in power of zero is 1:

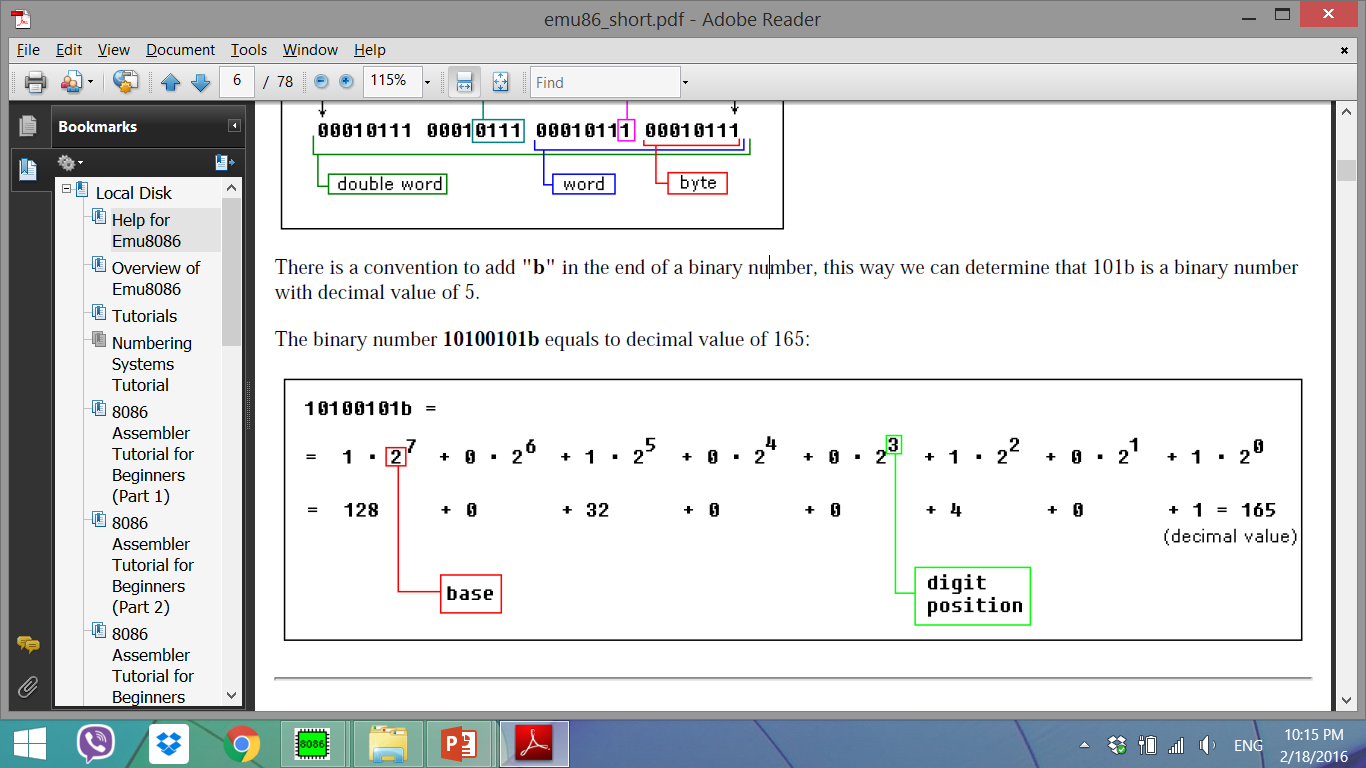


* **Binary System**

Computers are not as smart as humans, it's easy to make an electronic machine with two states: **on** and **off**, or **1** and **0**.

Computers use binary system, binary system uses 2 digits:

**0, 1**. And thus the **base** is **2**.

10100101 b

* **Hexadecimal System**

Hexadecimal System uses **16 digits**:

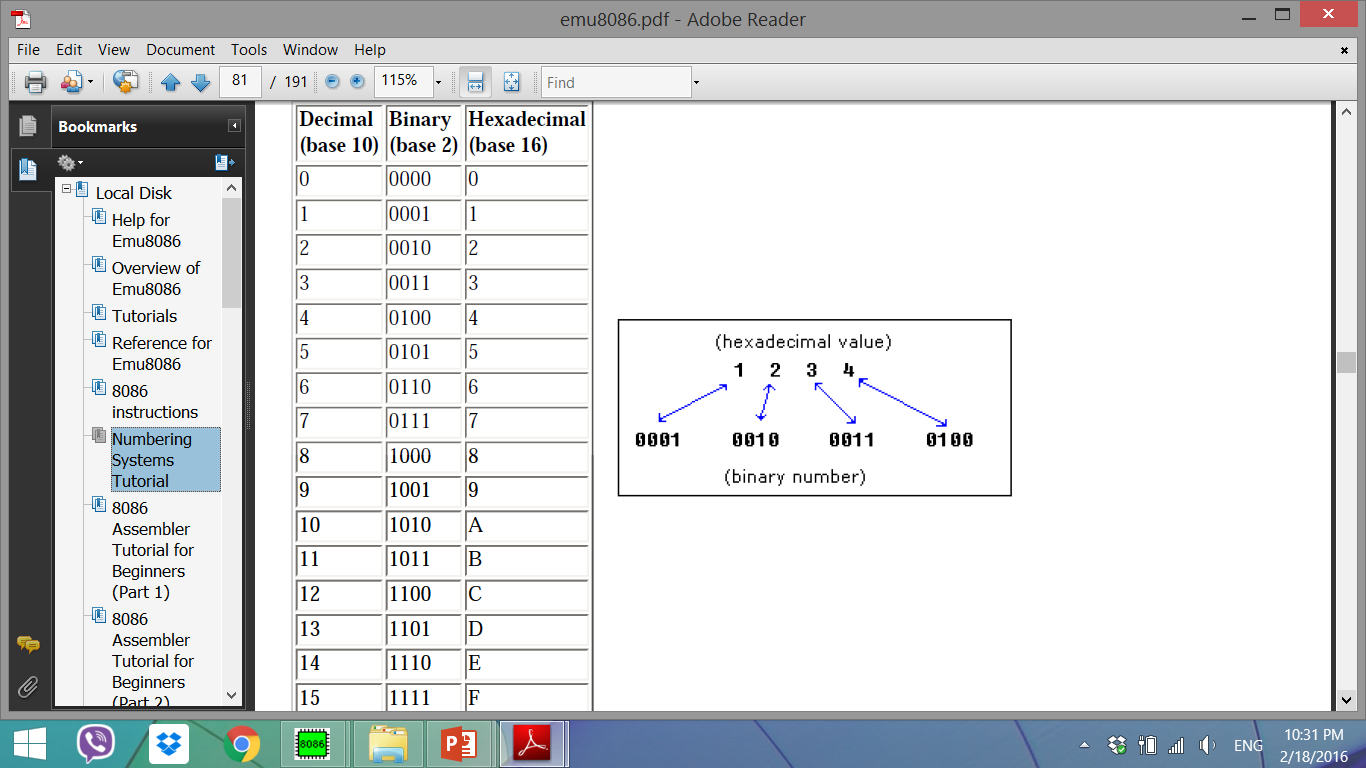
**0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F**

And thus the **base** is **16**.

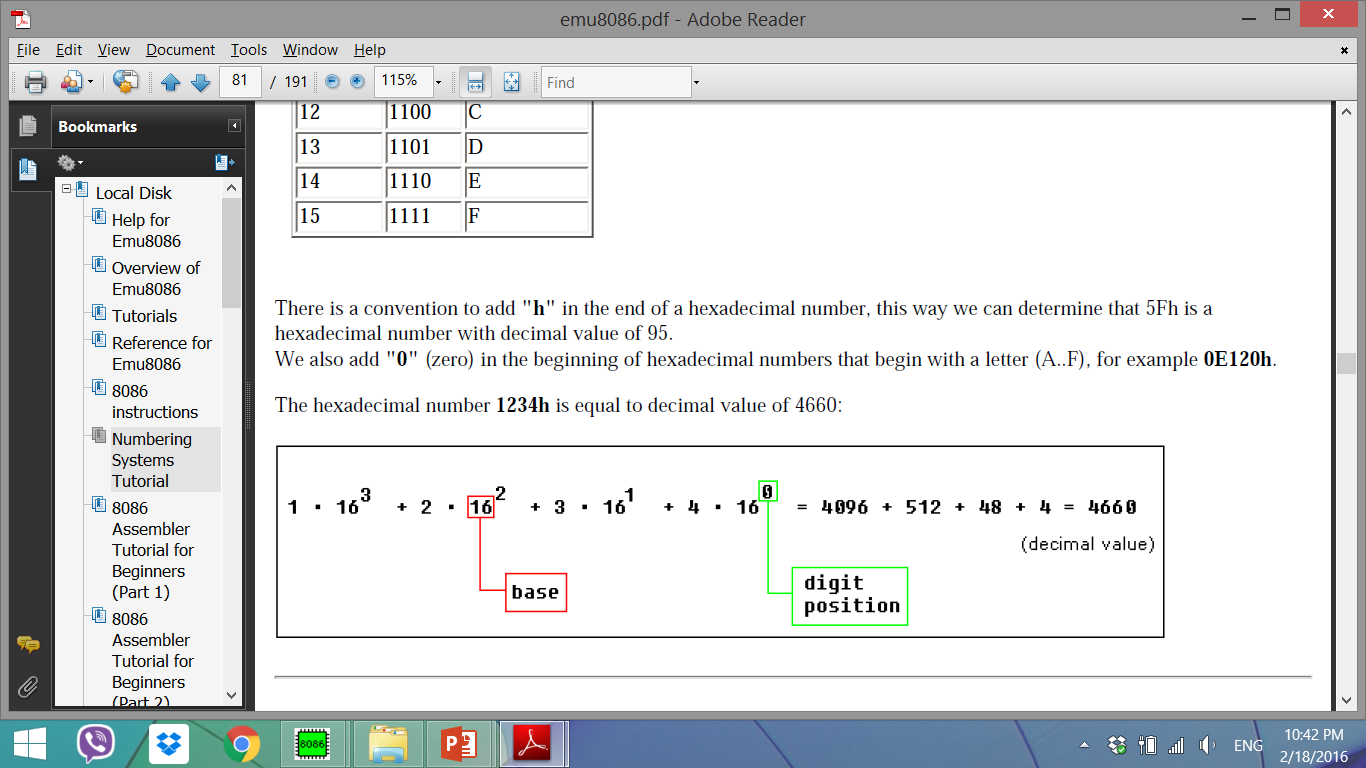
Note:

* Hexadecimal numbers are compact and easy to read.
* It is very easy to convert numbers from binary system to hexadecimal system and vice-versa, every nibble (4 bits)

**converted to a hexadecimal digit using this table:**



* There is a convention to add **"h"** in the end of a hexadecimal number, this way we can determine that 5Fh is a hexadecimal number with decimal value of 95.
* We also add **"0"** (zero) in the beginning of hexadecimal numbers that begin with a letter (A..F), for example **0E120h**.
* The hexadecimal number **1234h** is equal to decimal value of 4660:

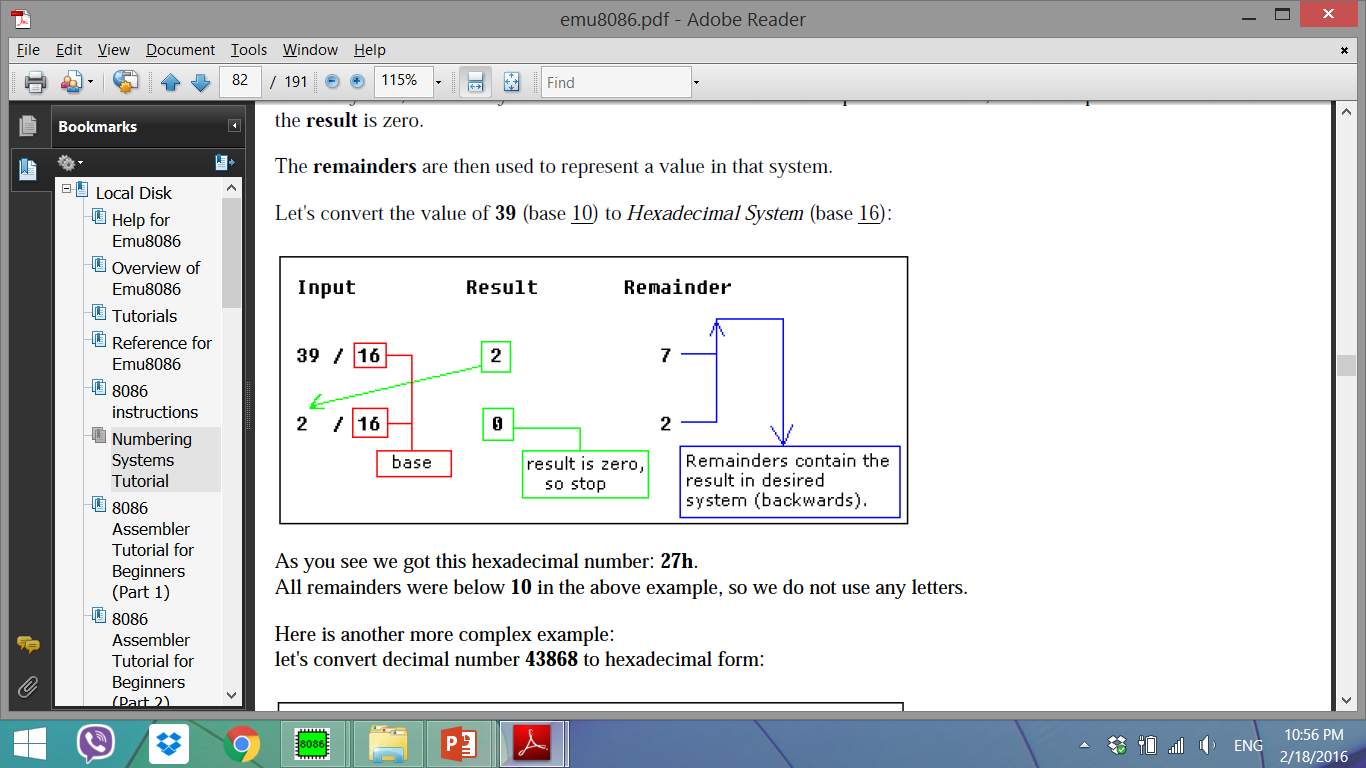


**Converting from Decimal System to Any Other**

In order to convert from decimal system, to any other system, it is required to divide the decimal value by the **base** of the desired system, each time you should remember the **result** and keep the **remainder**, the divide process continues until the **result** is zero.

The **remainders** are then used to represent a value in that system.

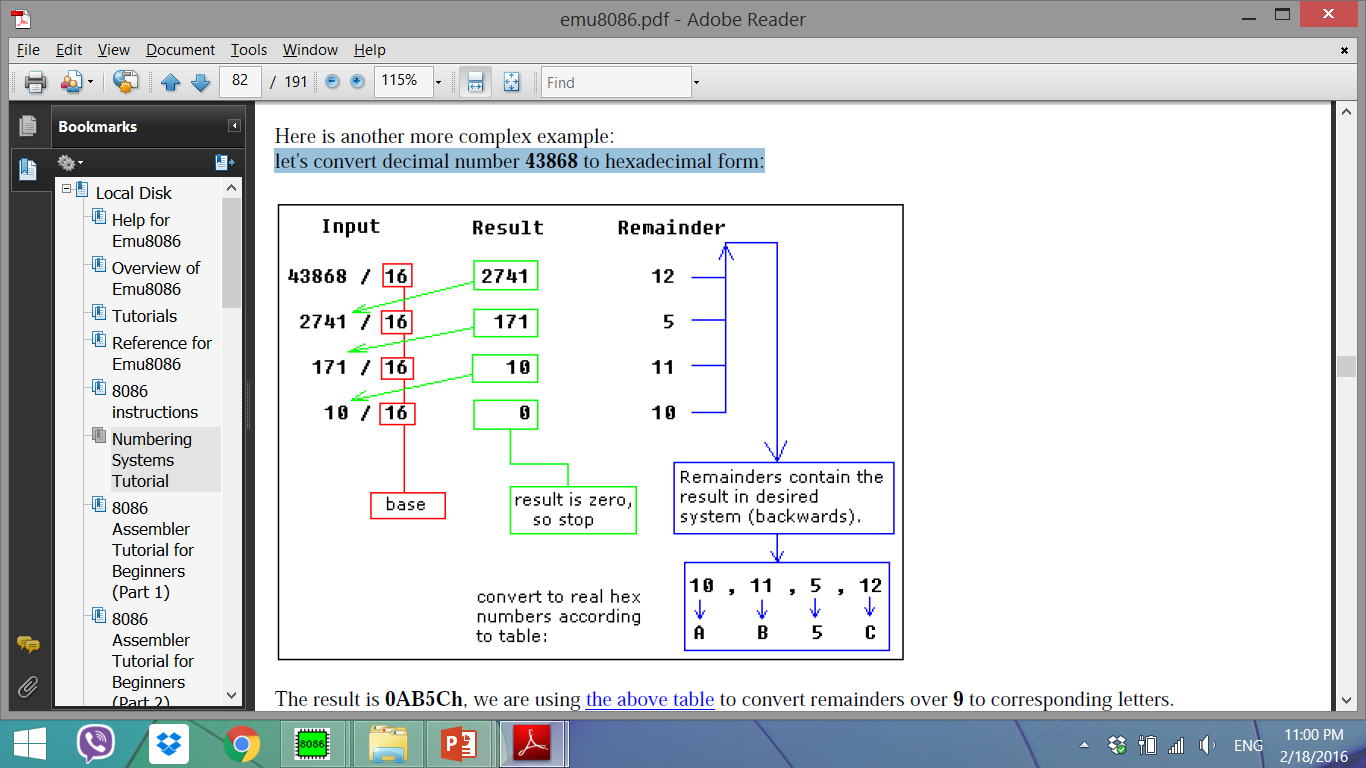
Let's convert the value of **39** (base 10) to *Hexadecimal System* (base 16):



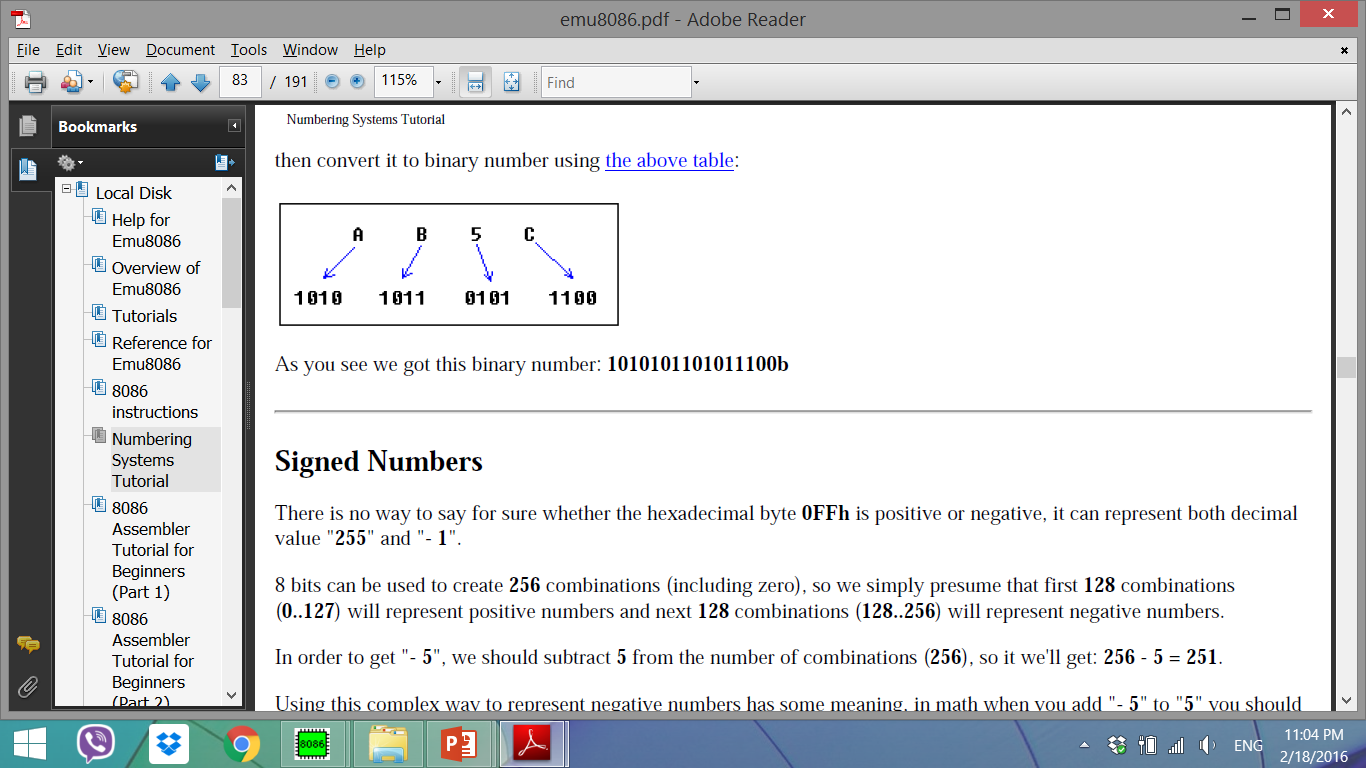
As you see we got this hexadecimal number: **27h**.

All remainders were below **10** in the above example, so we do not use any letters.

**let's convert decimal number 43868 to hexadecimal form:**

****

**Convert hexadecimal to Binary number**

****

**Notes:**

* Each digit in a binary number is called a **BIT**,
* 4 bits form a **NIBBLE**,
* 8 bits form a **BYTE**,
* two bytes form a **WORD**,
* two words form a **DOUBLE WORD** (rarely used)
* There is a convention to add **"b"** in the end of a binary number, this way we can determine that 101b is a binary number with decimal value of 5.
* Binary numbers must have "**b**" suffix, example: 00011011b
* Hexadecimal numbers must have "**h**" suffix, and start with a zero

when first digit is a letter (A..F), example: 0ABCDh

* Octal (base 8) numbers must have "**o**" suffix, example: 77o

………………………………………………

**H/W**

1. **convert 13d to b**
2. **convert 24d to b**
3. **convert 314 d to h**
4. **convert 126 d to h**