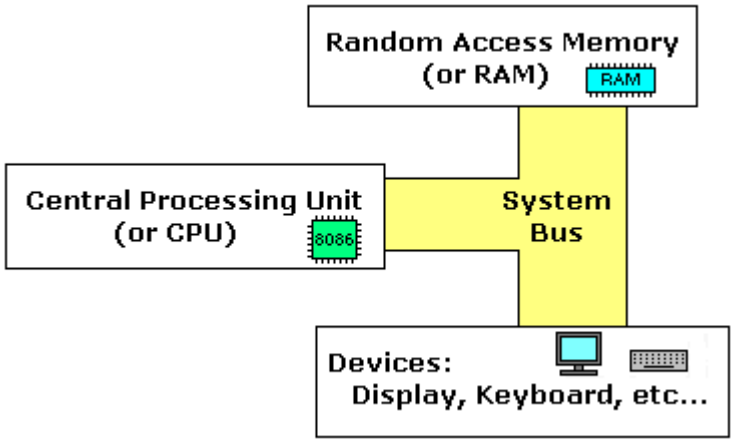
Assembly language

Assembly language is a low level programming language. you need to get some knowledge about computer structure in order to understand anything. the simple computer model as shown in figure below:



System bus: (shown in yellow) connects the various components of a computer.
CPU: is the heart of the computer, most of computations occur inside the **CPU**.
RAM: is a place to where the programs are loaded in order to be executed.

inside the CPU

General purpose registers

8086 CPU has 8 general purpose registers, each register has its own name:

- **AX** - the accumulator register (divided into **AH / AL**).
- **BX** - the base address register (divided into **BH / BL**).
- **CX** - the count register (divided into **CH / CL**).
- **DX** - the data register (divided into **DH / DL**).
- **SI** - source index register.
- **DI** - destination index register.
- **BP** - base pointer.
- **SP** - stack pointer.

The size of the above registers is 16 bit, it's something like: **0011000000111001b.** The programmer who determines the usage for each general purpose registers.

4 general purpose registers (AX, BX, CX, DX) are made of two separate 8 bit registers, for example if AX= **0011000000111001b**, then AH=**00110000b** and AL=**00111001b**, "H" is for high and "L" is for low part.

The registers are located inside the cpu, they are much faster than memory. accessing a memory location requires the use of a system bus, so it takes much longer. accessing data in a register usually takes no time.

Special purpose registers

- **IP** - the instruction pointer.
- **flags register** - determines the current state of the microprocessor.

Flags register is modified automatically by CPU after mathematical operations, this allows to determine the type of the result, and to determine conditions to transfer control to other parts of the program.

Assembly Language instruction set (8086)

1) MOV

Copy operand2 to operand1 (operand1 = operand2)
Example: MOV AL, 2

2) ADD

operand1 = operand1 + operand2
Example: ADD AL, BL

*Use emu8086 software to execute the instructions above.

Exercise: Add 3 to 4 to 2 by using three different general registers. and then show your result by emu8086.

3) **Subtract.**

Algorithm:

operand1 = operand1 - operand2

Example:

```
Org 1000h
MOV AL, 5
MOV BL, 2
SUB AL, BL
RET
```

---

4) **Logical AND** between all bits of two operands. Result is stored in operand1.

1 AND 1 = 1
1 AND 0 = 0
0 AND 1 = 0
0 AND 0 = 0

Example:

```
Org 1000h
MOV AL, 00000111b
MOV BL, 00000011b
AND AL, BL
RET
```

H.W

AH = 00001111b, BL = 00000111b, CH = 2

Use the "Logical AND" between AH and BL, and then subtract CH from the result of AND operation. Show your result by emu8086.

5) Invert each bit of the operand.

Algorithm:

- if bit is 1 turn it to 0.
- if bit is 0 turn it to 1.

Example:

```
MOV AL, 11111011b
NOT AL
RET
```

---

6) Logical OR between all bits of two operands. Result is stored in first operand.

1 OR 1 = 1
1 OR 0 = 1
0 OR 1 = 1
0 OR 0 = 0

Example:

```
MOV AL, 00001010b
MOV BL, 00001111
OR AL, BL
RET
```
