# 8086 Microprocessor

- Memory Bank
- Piping
- Segmentation
- Architecture

→ Enchanced version of 8085 mp.
→ 16 bit mp (binary data)
→ 20 address lines $A_0$ - $A_{19}$
→ 16 data lines $D_0$ - $D_{15}$
→ It can read/write data to a memory/port either 16/8 bit at a time.
→ $2^{20}$ = 1MB
→ 40 pins
→ +5V power supply
→ 6 to 10 MHz frequency
→ It can provide div & mul operations.

- 8086 has two main blocks
① Bus interface Unit (BIU)
② Execution Unit (EU)

- BIU handles system buses. BIU connects EU with the memory or I/O circuits. It is responsible for transmitting data, addresses and control signals on the bus.

- EU receives from instruction codes & data from BIU, execute them & stores them in general registers. Receives its all inputs & outputs from BIU.

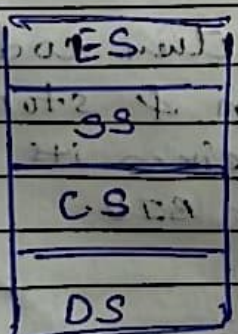
- Both BIU and EU operates asynchronously to give 8086 an overlapping instruction fetch and execution mechanism which is called pipelining.
- fetching the next instruction while the current instruction is still executing is called pipelining.

* Segmentation :-

It is a process in which the main memory of the computer is divided into different segments. Each segment has its own base address.
It is done for faster execution of fetching & executing of data.

→ (BIU) 16 bit registers :-

1. Code segment registor (CS)
2. Data segment registor (DSS)
3. Extra segment registor (ES)
4. Stack segment registor (SS)



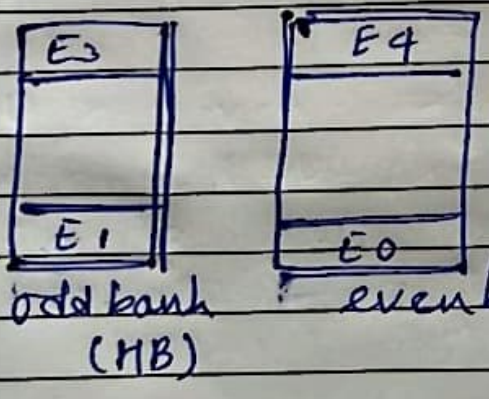
## Memory Banking in 8086 :-

8086 is a 16 bit data bus mp. So capabale of 16 bits of data in one cycle but each memory location is of 8 bit only, so we need two cycles to access 16 bit (8 bits each) from two memory locations, which is done by memory banking. So two consecutive memory locations are executed in one cycle.

Memory chip is divided into two equal parts.
One of the banks contain even addresses called even bank and other is odd bank.

Even bank always gives lower byte so it is lower bank (LB) and odd bank is higher bank (HB).

It just facilates a 16 bit transfer doesn't makes it compulsary.



2. **Explain in detail the architecture of Intel 8086.**

Salient features:

The 8086 processor is a 16-bit processor internally as well as externally – which means that its data bus width as well as the bit size of its internal data registers is 16. Thus it can access an external source 16 bits at a time through its data bus. However, it also has the capability to access and work on 8-bit (byte) data. It's a pipelined processor having two stages of pipelining.
The memory in 8086 is segmented into four parts namely, code segment, stack segment, data segment and extra segment to prevent overlap of memory and as a result, overwrite of data.

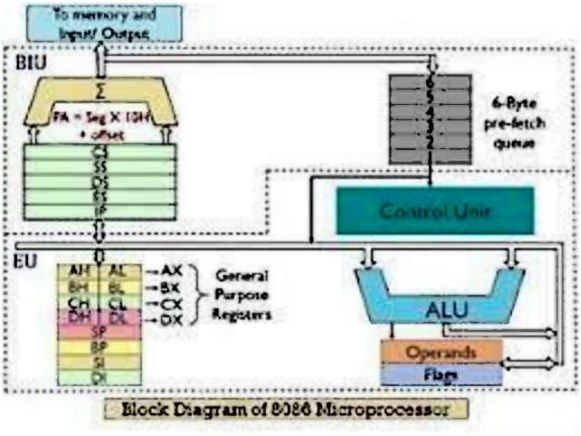



Figure 6: 8086 architecture

The internal block diagram has been partitioned into two logical units, namely, the Bus Interface Unit (BIU) and the Execution Unit. They interact directly with each other through the internal bus, but perform separately as two units with well-defined functions for each unit.

**Bus Interface Unit:**

This unit is responsible for address calculations, pre-fetching instructions for the queue and sequencing instructions one by one.
Pre-fetch Queue:
In 8086, there is a queue which fetches instructions ahead of the execution time and places them in a 6-byte first-in-first-out (FIFO) queue. This pre-fetching is done when the buses are free i.e., not being used for the execution of the current instruction. The advantage of pre-fetching is that when a particular instruction is to be executed, there is a good chance of finding it in the queue (which is on-chip), rather than having to go to memory to fetch it.
Special Registers:
IP is the instruction pointer analogous to the program counter, it contains the offset address of the instruction in code segment. The other special purpose registers hold the addresses of different segments of memory. They are named as: Code Segment (CS), Stack Segment (SS), Data Segment (DS) and Extra Segment (ES).

**Execution Unit:**

The 8086 has data registers, address registers, segment registers and also a flag register.
8086 has four 16-bit general-purpose registers labeled as AX, BX, CX and DX. Each of these registers can also be used as two separate and exclusive 8-bit registers also.
AX This is a 16-bit data register, but it can also be used as two unrelated 8-bit registers AH and AL. AL /AX is sometimes called the accumulator, but the relevance of the accumulator is less for 8086 compared to the earlier 8085 in which one operand is implied to be in the A register for many instructions. only for certain specialized instructions is the AL or AX register implied to contain one of the operands. Base register BX is frequently used as an address register in many based addressing modes. Counting register CX is used as a counter in many instructions. Data register DX is used in I/O instructions as a pointer to data by storing the address of the I/O port. Also, multiply and divide operations involving large numbers assume DX and AX as a pair. Pointer and Index Registers SP, BP, SI and DI are address registers, and can be used only as 16-bit registers. They are very important in programming, as they facilitate the use of various addressing modes. Arithmetic Logic Unit: It is the part of a computer that performs all arithmetic and logic computations. The ALU is the most important

unit of the processor. Instructions that are fetched and decoded, are executed in the ALU. Thus the ALU has direct access to the general purpose registers and flags.
Operands register has temporary values that are used by the processor.

Both BIU and EU have mathematical circuits, the circuit in BIU is for address calculation and the one in EU is the ALU.
Control unit sends timing and control signals.

# IMP

# 8086

* Addressing modes
The manner in which an operand is given to an instruction
5 types: Immediate
Register
Implied
Direct
Indirect

1 Immediate
Value is directly provided to register
E.g MOV CL, 54H

2 Register
Value is stored in register then provided to another register
E.g MOV BX, 1234H
MOV CX, BX

3 Implied
Operand is implied / understood that instruction is working on certain part
SLC → Set Carry Flag
CLC → Clear Carry Flag
DAA → Decimal adjust after

4 Direct
Directs the processor towards an address
E.g MOV CX, [2001H]

5 Indirect

Value is given indirectly from address to address

Most important, powerful and used mode

Eg MOV CL, [BX]

4 types : Register Indirect
Register Relative
Base indexed
Base relative + indexed

a Register indirect - Address is given by register

Most basic addressing mode

In 8086, register used for storing address is BX - Rule given by Intel

Syntax : MOV CL, [BX]

Apart from BX, BP (Base pointer), SI (Source Index), DI (destination index) can be used

b Register Relative - Displacement can be above (-ve) or below (+ve) in memory. Displacement can be of 8 or 16 byte

Value of address given should be within the segment

Address = Register + Displacement

Eg BX + D3H

c Base Index - To access subset of an array

Address = Base register + Index register

E.g BX + SI

d Base Index + Relative

Address = Base + Index + Displacement

Eg BX + SI ± 03H

- Data Memory Addressing Mode (MOV AL, [2000H])
- Program Memory Addressing Mode (JMP 0006H)
- Stack Memory Addressing Mode (Push BX)

* 8086 Programmer's Model
Program visible programming module

- A Register : Accumulator
Acts as an accumulator for multiply and divide operations
Also used for DAA operations and string operations

- B Register : Memory Pointer
Used for indirect addressing mode

- C Register : Counter Register
Used for registers where counting is implied

- D Register : Extension for Accumulator
Used to store Higher significant bits (HSB) in 32 bit value

- Code Segment
  Stack Segment
  Data Segment
  Extra Segment
  } are also used

- Flag Register
8085 - 8 bit flag register
8086 - 16 bit flag register

DF, IF, TF : Control Flag
Can be controlled by programmer

OF, SF, ZF, AF, PF, CF : Status flags
Changes after every operation by ALU

X: Don't care condition
Doesn't make changes if it is 0 or 1

CF: Carry flag
When a carry is generated at most significant bit, it is stored in Carry flag

PF: Parity flag
Used in error detection or correction

AF: Auxiliary flag

ZF: Zero flag
Used in comparision (for subtracting)

SF: Sign flag
Tells us if value is signed or unsigned

* Operating Modes of 8086

1) Real mode memory addresing
Uses real address of memory location
Also called Real address mode
All adresses given in real mode correspond to real memory location
It has no support for memory protection, multitasking or code priviledge level

2) Protected mode memory addressing
Also called Protected virtual address mode
It allows system software to use features like virtual memory, paging and multitasking
This increases control of operating system over application programs

* Instruction Set of 8086

Type of Instructions: Data Transfer
Arithmetic
Bit Manipulation
String
Processor Control Instructions
Iteration Control Instructions
Interrupt Instructions
Branch and Loop

- Data Transfer

PUSH and POP

Push: adding a number to stack

Pop: removing a number from stack

XLAT

Translate / Transfer

Application: Lookup Table - stores value that are used to display digital

MOV

XCHG

Accumulator gets value from address and then translate it to it's own position

2 fundamental Addressing modes are required to be used with in, out instruction i.e Direct and Indirect

Indirect AM is used when I/O address is of 16 bits. DX register is used

Direct AM is also called Fixed port addressing

Indirect AM is also called Variable port addressing

| | Data size | Address size |
|---|---|---|
| IN AL, 30H | 8 | 8 |
| IN AX, 80H | 16 | 8 |
| IN AL, DX | 8 | 16 |
| IN AX, DX | 16 | 16 |

```
MOV AL, 80H
MOV DX, 2000H
OUT DX, AL
```

IN and OUT

Syntax: In or out Destination, source;

Port address is for I/O device - 16 bits

Data is transferred between Processor and I/O or memory

In and out deals with data transfer between I/O and processor

- Logical Instruction

| | Syntax |
|---|---|
| AND | AND destination, source; |
| OR | OR destination, source; |
| NOT | NOT destination; |
| XOR | XOR destination, source; |
| TEST | TEST destination, source; |

- Rotate and Shift Instruction

ROTATE instruction

ROL (Rotate left)
ROR (Rotate right)
RCR (Rotate right with carry)
RCL (Rotate left with carry)

~~register, count;~~

SHIFT instruction

SHL (Shift left)
SHR (Shift Right)
SAR (Shift arithmetic right)
SAL (Shift arithmetic left)

~~register, count;~~

- Arithmetic Instruction

| | |
|---|---|
| ADD | Addition |
| ADC | Addition with carry |
| SUB | Subtraction |
| SBB | Subtraction with borrow |
| INC | Increment |
| DEC | Decrement |
| NEG | Negation |
| CMP | Comparision |
| MUL | Multiplication |
| IMUL | Integer Multiplication |
| CBW | Convert byte to word |
| CWD | Convert word to doubleword |
| DIV | Division |
| DAA | Decimal Adjust after Addition |
| DAS | Decimal Adjust after Subtraction |

DAA (Decimal Adjust after Addition) adjusts the result of ~~adding two valid packed decimal~~ operands in AL. DAA must always follow the addition of two ~~pairs of packed decimal~~ numbers (one digit in each half-byte) to obtain a pair of valid ~~packed decimal digits as results.~~ The carry flag is set if carry was needed.

- Processor Control Instruction

Instructions for entire Flag register

PushF : Adds content of flag into stack
PopF : Removes content of flag from stack
LAHF : Takes lower nibble of flag and puts it in Accumulator
SAHF : Takes value from Accumulator and puts it in lower nibble

Instructions for Carry Flag :

STC (Set) - Change value to 1
CLC (Clear) - Change value to 0
CMC (Complement) - Complement the carry flag

Instructions for Direction Flag :

STD (Set) - Value set to 1
CLD (Clear) - Value set to 0

Instructions for Interrupt flag

STI (Set)
CLI (Clear)

- Setting Trap Flag procedure

```
PushF ;
Pop BX ;
Or BH, 01H ;
Push BX ;
PopF ;
```

Clearing Trap Flag procedure

```
PushF ;
Pop BX ;
AND BH, 0FEH ;
Push BX ;
PopF ;
```

* Assembly language for 8086

Programmer → Program → Assembler → Assembles → Microprocessor and stores

→ General structure of Assembly language program

```
Segment // Data segment
  Has variables // variable name DB (define byte) value;
Ends
Segment // Code
Ends
```

Datatypes:

| | | |
|---|---|---|
| DB | Define byte | 8 bits |
| DW | Define word | 16 bits |
| DD | Define doubleword | 32 bits |
| DQ | Define quadword | 64 bits |
| DT | Define Ten bytes | 80 bits |

* Parallel Communication

Expansion buses - connected at Southbridge
Used for connecting peripherals

ISA - Industry Standard Architecture
8 $MH_2$, 16 bit bus

EISA
32 bits at 8 $MH_2$

VL Bus

Developed for video by VESA - Video Electronics Standard Association

PCI - Peripheral Component Interconnect

Has replaced ISA bus on the motherboard

Transfer rate - 33 MHz and 32 bits at a time

Bandwidth - 133 MBps

Variations - 66 MHz, 32 bit PCI, 64 bit

64 bit 133 MHz PCI. The last version is used mostly in servers