Revision - 1st internals MODULE 1

Class	MP&CO
Created	@Feb 27, 2021 12:00 AM
Materials	PMCO - Module 1 Notes.pdf

Evolution of Microprocessors

- 1971 - World's first microprocessor : **Intel 4004** , a 4 bit microprocessor was developed. It was a programmable controller on a chip.

  It has 12 bit address lines that could access 4096 (2^12) memory locations. The 4004 instruction set contained only 45 instructions.

  Main problems → Speed, word width and memory size.

- Later in 1971, Intel released the **Intel 8008** , an extended 8-bit version of the 4004. It addressed a larger memory size and contained additional instructions (48)
- 1974 - Intel introduced the **8080** which was the first of the modern 8-bit microprocessors.
- 1997 - 8085 - 8bit up

The modern microprocessor

- 1978 - Intel released the **8086** microprocessor, and the **8088** about a year later. Both were 16-bit microprocessors. They addressed 1M bytes of memory, 16 times more memory than the 8085
- The 16 bit microprocessor evolved mainly due to the need for larger memory systems.

The 32 bit microprocessor

- The **80386** was intel's first 32 bit microprocessor that addressed upto 4G bytes of memory

The Pentium microprocessor

- The pentium, introduced in 1993, was similar to the 80386 and 80486
- It operated with a clocking frequency of 60-66MHz
- The memory system contained upto 4Gbytes, with an increased cache size of 16K.
- **Pentium II** → placed on a small circuit board rather than an IC. It was rated at 355-450MHz
- **Pentium III** → Faster core than Pentium II, and was available with clock frequencies upto 1GHz
- **Pentium 4 and Core2** → 3.2 GHz and faster

64 bit multiple core microprocessors

Intel introduced modifications to the Pentium 4 and Core2 processors to include a 64-bit core and multiple cores.

Each core executes a different task in a program which increases the speed of execution. The programs that do this are called **multithreaded** applications.

The future of microprocessors

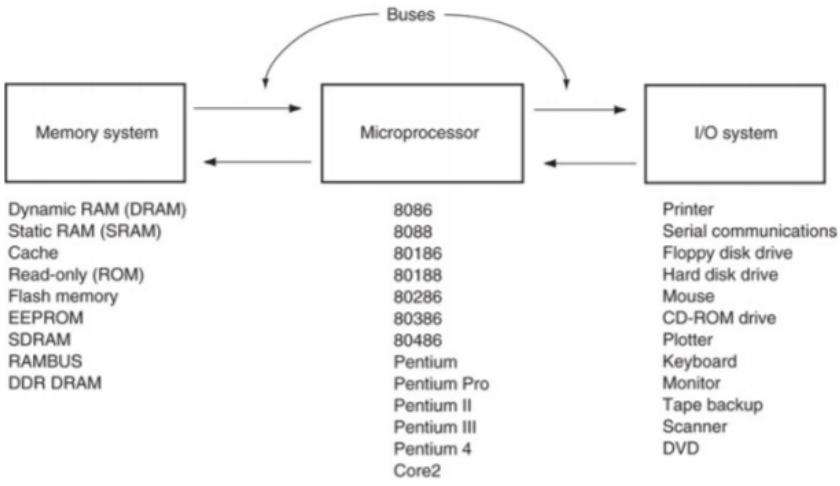
What may occur is a change to RISC technology, but more likely improvements to new
hyper-threading technology being developed jointly by Intel and HP.

The basic idea behind this technology is that many microprocessors communicate directly
with each other. Allowing parallel processing without any change to the instruction set or
program.

The Microprocessor based Personal Computer system

Figure below shows the block diagram of the personal computer. This diagram also applies to
any computer system.

The block diagram is composed of three blocks that are interconnected by buses.



A **bus** is a set of common connections that carry the same type of information.
ex : The address bus conveys the memory address to the memory.

The memory and I/O system

The memory structure of all Intel 80X86 and pentium based PC systems are similar. The memory is divided into three parts, **Transient program area(TPA), system area and extended memory system(XMS)**

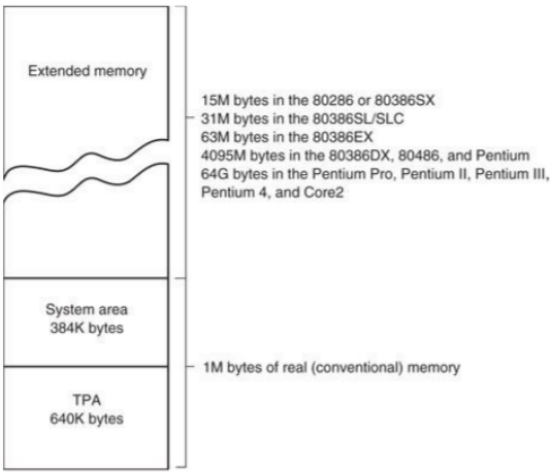
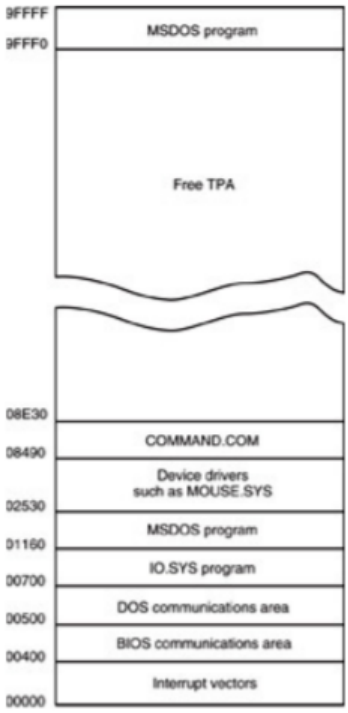


Figure: The memory map of a personal computer.

First 1M bytes of memory is often called the real or conventional memory system, because each Intel microprocessor is designed to function in this area in its real mode of operation.

Transient program area

The TPA holds the DOS operating system and the other programs that control the computer. It also stores currently active or inactive DOS application programs. The length of the TPA is 640K bytesv



The interrupt vectors access various features of the DOS and BIOS.

The system BIOS communications area and DOS communications area contain transient
data used by programs to access I/O devices and the internal features of the computer system.

IO.sys is a program that loads into the TPA from the disk whenever an MSDOS system is started. It allows programs that allow the DOS to use keyboard, video display and other I/O devices.

The MSDOS program area occupies 2 parts of the TPA

Drivers are files with the extension .sys and are programs that control the installable devices like mouse , scanner etc.

The COMMAND.com program controls the operation of the computer through keyboard when started in the DOS mode.

The free TPA area holds DOS application programs as they are executed.

The System area

The system area is smaller than the TPA. It contains programs for data storage. It contains programs on either a ROM or flash memory and also in some areas of the RAM.

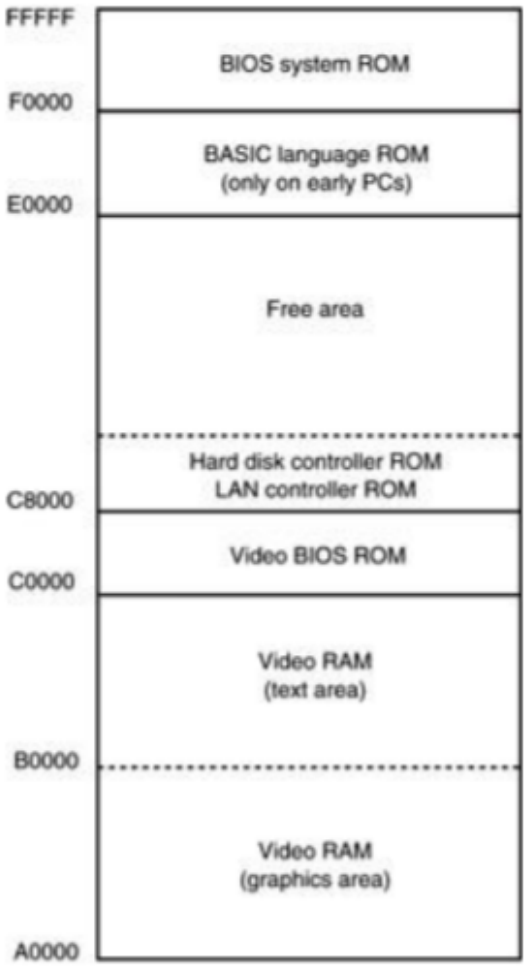
The first area of system space extends from A0000H - C7FFFH and contains video display RAM and

video control programs on ROM or flash memory. The video BIOS ROM located on a ROM or flash memory contains the programs that control the DOS video display.

Hard disk controller ROM holds low level format software.

The free area is used for the expanded memory in a PC system or for the upper memory system in an AT system. The next memory locations contain cassette BASIC language on ROM found in early IBM PC systems.

The system BIOS ROM controls the operation of basic I/O devices connected to the PC system.



I/O Space

IO devices allow the computer to communicate with the outside world. An I/O port is similar to a memory address, but addresses an I/O device. The I/O space contains 2 major sections. The area below 0400H is considered reserved for system devices. The remaining space is available for expansion on newer systems.

Microprocessor

- Can be called as the heart of microprocessor based personal computer system. Also known as CPU - Central Processing Unit
- The microprocessor connects memory to I/O devices through the buses. Memory and I/O are controlled using instructions that are stored in memory and executed by the microprocessor.

- The microprocessor performs three main tasks:
    1. Data transfer between itself and the memory or I/O systems
    2. Simple arithmetic & logic operations
    3. Program flow via simple decisions

The strength of the microprocessor lies in its ability to perform millions of instructions per minute from the software or programs.

The arithmetic and logical instructions executed by the microprocessor are:

1. Addition
2. Subtraction
3. Multiplication
4. Division
5. AND (logical multiplication)
6. OR (logic addition)
7. NOT (logical inversion)
8. NEG (arithmetic inversion)
9. Shift
10. Rotate

Data is stored in the memory or internal registers.The width of the data is either a byte (8 bits) or a word (16 bits) or a double word (32 bits). Only the 80386 and above are able to execute all three.

Another powerful feature is the ability to make simple decisions based upon numerical facts.
For example a microprocessor can decide if a number is zero, positive, and so forth. These
decisions allow the microprocessor to modify the program flow, so programs appear to think
through these simple decisions

Buses

A bus is a common group of wires that interconnect components in a computer system. In the microprocessor based system, three buses exist for the transfer of information between the microprocessor and its memory and I/O systems. These are **Address bus, Data bus, Control bus.**

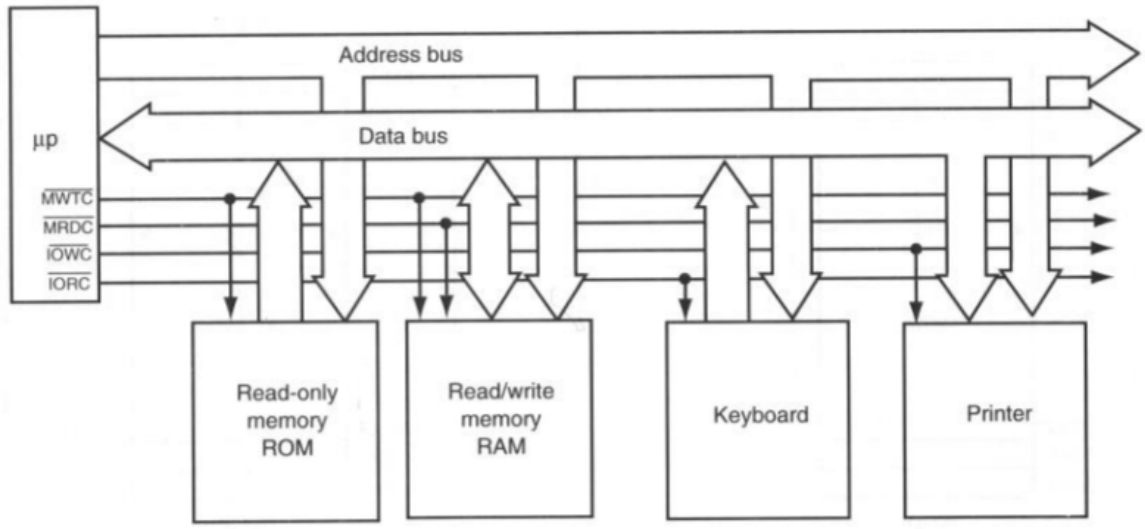


FIGURE The block diagram of a computer system showing the address, data, and control bus structure

The address bus Requests a memory location from the memory or an I/O location from the I/O devices. If I/O is addressed, the address bus contains a 16bit IO address. If memory is addressed, the bus contains a memory address instead. This address varies in width with the type of microprocessor.

The address bus is a unidirectional bus because address bus sends out the address to the memory location or I/O location.

The data bus transfers information between the microprocessor and its memory and I/O address space. It is bidirectional.

Control bus contains lines that select the memory or I/O and cause them to perform a read or write operation. In most computers, there are four control bus connections: Memory Read Control MRDC, Memory Write Control MWTC, IORC I/O Read Control, and IOWC I/O Write Control.

Internal Microprocessor Architecture

- The programming model of the 8086 through Core2 is considered to be program visible because its registers are used doing programming and are specified by the instructions. Other registers are considered program invisible because they cannot be directly addressed during application programming. But they can be used indirectly during system programming
- 80286 and upwards have program invisible registers to control and operate protected memory and other features.
- The earlier 8086/88/286 microprocessors contain 16bit internal architecture whereas the 80386 through Core2 contain full 32bit internal architecture.

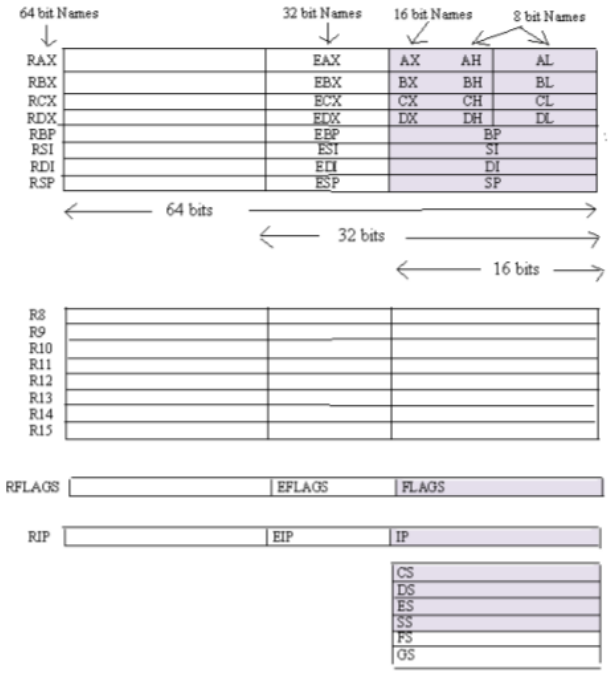


Figure. The programming model of the 8086 through the Core2 microprocessors

The architectures of the 8086/88/286 are fully upward compatible to the 80386 through Core2. The shaded areas represent registers that are not available in the 8086/88/286 microprocessors but are provided in the 80386 through Core2., for compatibility to the older versions.

The programming model contains 8,16,32,64 bit registers. The 8-bit registers are **AH, AL, BH, BL, CH, CL, DH and DL** . These are referred to when instructions is formed using two-letter designations.

The 16 bit registers are **AX, BX, CX, DX, SP, BP, DI, SI, IP, FLAGS, CS, DS, ES, SS, FS and GS**.

The first 4 16 bit registers contain a pair of 8 bit register. An example is AX which contain AH, AL. The 16 bit registers are referenced with the two letter designations
such as AX

The extended 32 bit registers are **EAX, EBX, ECX, EDX, ESP, EBP, EDI, ESI, EIP and EFLAGS.** These and the 16-bit registers FS and GS are available only on the 80386 and above. They are referenced with three letter designations.

Some registers are general purpose while others are multipurpose. The multipurpose registers include EAX, BAX, ECX, EDX, EBP, EDI and ESI. They hold various data sizes (bytes words or double words) and are used for almost any purpose.

The 64 bit registers **RAX, RBX** and so on. There are also additional 64 bit registers called **R8 through R15**

Multipurpose registers

RAX (Accumulator)

RAX is referenced as 64 bit register RAX, a 32 bit register EAX, a 16 bit register AX, or as either of the two 8-bit registers AH and AL. The accumulator is used for instructions such as multiplication, division and some of the adjustment instructions. In 80386 and above, EAX might also hold the offset address of a location in the memory system.

RBX (Base index)

RBX is addressable as RBX, EBX, BX, BH or BL. EBX registers sometimes hold offset address of a location in the memory system.

RCX (Count)

It is addressable as RCX, ECX, CX, CH, CL and is a general purpose register that also holds the count for various instructions. The shift and rotate instructions use CX as the count, and the LOOP/LOOPD instructions use either CX or ECX.

RDX (Data)

It is a general purpose register that holds a part of the result from a multiplication or a part of the dividend before a division.

RBP (Base pointer)

Points to a memory location.

RDI (Destination index)

Addresses string destination data for the string instructions.

RSI (Source index)

Often addresses source string data for the string instructions. Like RDI, it also functions as a general purpose register.

R8 through R15

These registers are only found on the Pentium4 and Core2 if extensions are enabled.

Special Purpose registers

The special-purpose registers include RIP, RSP, RFLAGS; and the segment registers include
CS, DS, ES, SS, FS, and GS.

RIP (Instruction pointer)

Addresses the next instruction in a section of memory defined as a code segment. The instruction pointer is used by the microprocessor to find the next sequential instruction in a program located within the code segment.

RSP (Stack pointer)

Addresses an area of memory called the stack. The stack memory stores data through this pointer.

RFLAGS

Indicates the condition of the microprocessor and controls its operation.

C (Carry)

Holds the carry after addition or the borrow after subtraction. The carry flag also indicates error conditions.

P (Parity)

Parity is logic 0 for odd parity and logic 1 for even parity. Parity is the count of ones in a number expressed as even or odd. For example if a number contains three binary one bits, it
has odd parity. If a number contains no one bits, it has even parity.

A (Auxiliary carry)

The auxiliary carry holds the carry (half-carry) after addition or the borrow after subtraction
between bits positions 3 and 4 of the result.

Z (zero)
The zero flag shows that the result of an arithmetic or logic operation is zero. If Z=1, the result is zero; if Z= 0, the result is not zero.

S (sign)
The sign flag holds the arithmetic sign of the result after arithmetic or logic instruction executes. If S=1, the sign bit (leftmost hit of a number) is set or negative; if S=0, the sign bit is cleared or positive.

I (interrupt)
The interrupt flag controls the operation of the INTR (interrupt request) input pin. If I=1, the INTR pin is enabled: if I= 0, the INTR pin is disabled. The state of the I flag bit is controlled by the STI (set I flag) and CLI (clear I flag) instructions.

D (direction)
The direction flag selects either the increment or decrement mode for the Dl and/or SI registers during string instructions. If D=1, the registers are automatically decremented: if D=1, the registers are automatically incremented. The D flag is set with the STD (set direction) and cleared with the CLD (clear direction) instructions.

O (overflow)
Overflows occur when signed numbers are added or subtracted. An overflow indicates that the result has exceeded the capacity of the machine. For unsigned operations, the overflow flag is ignored.

IOPL (I/0 privilege level)
IOPL is used in protected mode operation to select the privilege level for I/O devices.

NT (nested task)
The nested task flag indicates that the current task is nested within another task in protected mode operation. This line is set when the task is nested by software.

RF (resume)
The resume flag is used with debugging to control the resumption of execution after the next instruction.

VM (virtual mode)
The VM flag bit selects virtual mode operation in a protected mode system.

AC (alignment check)
The alignment check flag bit activates if a word or double word is addressed on a non-word or non-double word boundary

VIF (virtual interrupt flag)
The VIF is a copy of the interrupt flag bit available to the Pentium-Pentium IV microprocessors.

VIP (virtual interrupt pending)
VIP provides information about a virtual mode interrupt for the Pentium—Pentium IV microprocessors.

ID (identification)
The ID flag indicates that the Pentium—Pentium IV microprocessors support the CPUID instruction. The CPUID instruction provides the system with information about the Pentium microprocessor, such as its version number and manufacturer.

Segment registers

Additional registers, called segment registers, generate memory addresses when combined
with other registers in the microprocessor. There are either four or six segment registers in
various versions of the microprocessor.

CS (Code)

The code segment is a section of memory that holds the code (programs and procedures) used by the microprocessor. It defines the starting address of the section of memory holding code.

DS (Data)

Contains mostly data used by a program. Data are addressed by an offset address or the contents of another register that holds the offset address.

ES (Extra)

It is an additional data segment that is used by some of the string instructions to hold destination data

SS (Stack)

The stack segment defines the area of memory used for the stack. The stack entry point is defined by the Stack segment and Stack pointer registers. The BP register also addresses data within the stack segment.

FS and GS

These are supplemental segment registers available in the 80386 - Core2 microprocessors to allow 2 additional memory segments for access by programs.

Real Mode Memory Addressing

The 80826 and above operate in either the real or protected mode. Only the 8086 and 88 operate exclusively in the real mode. Real mode addressing allows the microprocessor to address only the first 1M byte or memory, called the real memory or the conventional memory system.

The DOS operating system requires the microprocessor to operate in the real mode. Real mode operation also allows software written for the 8086/88 to function in the 80286 and above without changing the software.

All microprocessors begin operation on real mode whenever power is applied or the microprocessor is reset.

Segments and Offsets

A combination of a segment address and an offset address, access a memory location in the real mode. The segment address, located in one of the segment registers defines the beginning address of any 64K-byte memory system.

The offset memory address locates any location in the 64K-byte memory system, by being added to the segment address.

Default segment and Offset registers

For example, the code segment register is always used with the instruction pointer to address
the next instruction in a program. This combination is CS:IP or CS:EIP, depending upon the
microprocessor's mode of operation. The code segment register defines the start of the code
segment and the instruction pointer locates the next instruction within the code segment.
This combination (CS: IP or CS: EIP) locates the next instruction executed by the microprocessor.

Segment	Offset	Special Purpose
CS	IP	Instruction address
SS	SP or BP	Stack address
DS	BX, DI, SI, an 8-bit number, or a 16-bit number	Data address
ES	DI for string instructions	String destination address

Table : Default 16-bit segment and offset address combinations

Segment	Offset	Special Purpose
CS	EIP	Instruction address
SS	ESP or EBP	Stack address
DS	EBX, EDI, ESI, EAX, ECX, EDX an 8-bit number, or a 32-bit number	Data address
ES	EDI for string instructions	String destination address
FS	No default	General address
GS	No default	General address

Table : Default 32-bit segment and offset address combinations

Suppose that an application program requires 1000H bytes of memory for its code,
190H bytes of memory for its data, and 200H bytes of memory for its stack. This application
does not require an extra segment. When this program is placed in the memory system by
DOS, it is loaded in the TPA at the first available area of memory above the drivers and other
TPA programs. This area is indicated by a free-pointer that is maintained by DOS. Program
loading is handled automatically by the program loader located within DOS.

The scheme of segment plus offset addressing allows programs to be relocated in the memory system. A relocatable program is one that can be placed into any area of memory and be executed without change.

Physical address calculation

Physical memory address pointed by SEGMENT:OFFSET pair is calculated as:
Physical address = (<segment address> * 10H) + <offset address>
Example: If CS = 24F6H and IP = 634AH , then segment address=24F6H and offset
address=634AH, then
PA = 24F6H*10H +634AH = 24F60H +634AH =34F5FH

Memory paging

The memory paging mechanism located within the 80386 and above allows for any physical memory to be assigned to any linear address. The linear address is defined as the address generated by the program. the physical address is the actual memory location accessed by a program.

Paging registers

The paging unit is controlled by the contents of the microprocessors control units.

8086 Pin diagram

AD15-AD8

Address/Data bus lines

These are multiplexed lines, and contain address bits A15 - A8 whenever ALE is a logic 1, and data bus connections D15-D8 when ALE is logic 0.

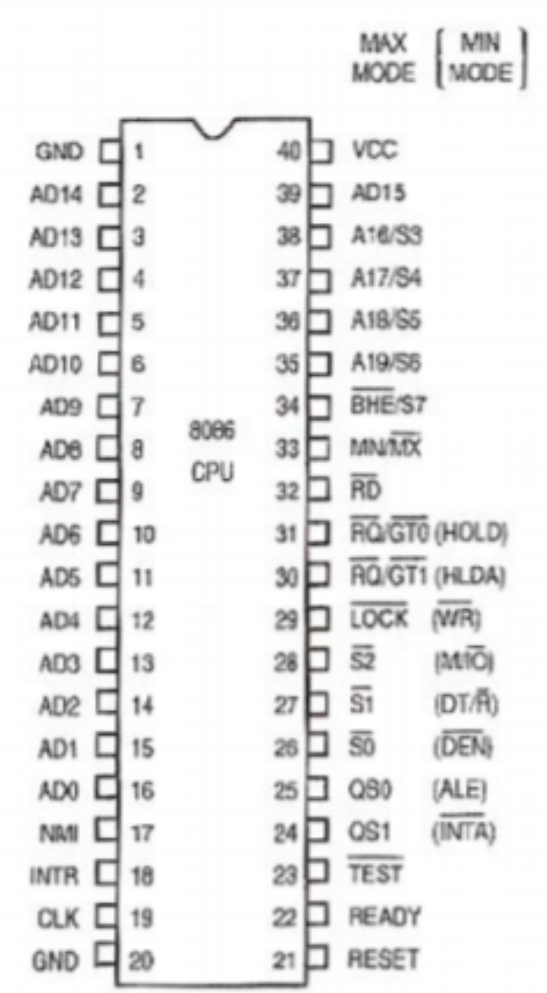
A19/S6 - A16/S3

Address and status bus bits are multiplexed to give address signals A19-A16 and status bits S6-S3.

S6 → Always remains 0

S5 → Indicates the condition of the interrupt flag

S4 and S3 → Indicate the segment being accessed during current bus cycle.



S4	S3	Function
0	0	Extra segment
0	1	Stack segment
1	0	Code or no segment
1	1	Data segment

Ready

Used to enforce a waiting state.

0 → The microprocessor goes into idle state

1 → The microprocessor does normal operation.

RD

When read signal is logic 0, the data bus is receptive to data from memory or I/O devices

INTR

Interrupt request is used to request a hardware interrupt. If INTR is held high when IF is 1, 8086/88 goes into an interrupt acknowledge cycle after the current instruction has completed execution.

NMI

The Non-Maskable Interrupt input is similar to INTR.

- Does not check for IF flag bit logic 1
- If activated, uses interrupt vector 2.

TEST

The Test pin is an input that is tested by the WAIT instruction.

- If TEST is a logic 0, the WAIT instruction functions as an NOP.
- If TEST is a logic 1, the WAIT instruction waits for TEST to become a logic 0.
- The TEST pin is most often connected to the 8087 numeric coprocessor
-

RESET

Causes the microprocessor to reset itself if held high for a minimum of four clocking periods.

- When 8086/8088 is reset, it executes instructions at memory location FFFFOH

- Also disables future interrupts by clearing IF flag

CLK

The **clock** pin provides the basic timing signal.

VCC

This **power supply** input provides a +5.0 V, signal to the microprocessor.

GND

The **ground** connection is the return for the power supply.

MN/$\overline{MX}$

Minimum/maximum mode pin selects either minimum or maximum mode operation.

if minimum mode selected, the MN/$\overline{MX}$ pin must be connected directly to +5.0 V

$\overline{BHE}$ / S_7

The **bus high enable** pin is used in 8086 to enable the most-significant data bus bits(D_{15}–D_8) during a read or a write operation.

- The state of S_7 is always logic 1.

Minimum Mode Pins

Minimum mode operation is obtained by connecting the MN/$\overline{MX}$ pin directly to +5.0 V

$\overline{IO}$/M or M/$\overline{IO}$

The **$\overline{IO}$/M** (8088) or **M/$\overline{IO}$** (8086) pin selects memory or I/O.

33

Indicates the address bus contains either a memory address or an I/O port address.

$\overline{WR}$

Write line indicates 8086/8088 is outputting data to a memory or I/O device.

During the time $\overline{WR}$ is a logic 0, the data bus contains valid data for memory or I/O

INTA

The **interrupt acknowledge** signal is a response to the INTR input pin.

- Normally used to gate the interrupt vector number onto the data bus in response to an interrupt

ALE

Address latch enable shows the 8086/8088 address/data bus contains an address.

- can be a memory address or an I/O port number

DT/$\overline{R}$

The **data transmit/receive** signal shows that the microprocessor data bus is transmitting (**DT/$\overline{R}$ = 1**) or receiving (**DT/$\overline{R}$ = 0**) data.

$\overline{\text{DEN}}$

Data bus enable activates external data bus buffers.

HOLD

Hold input requests a direct memory access (DMA).

if HOLD signal is a logic 1, the microprocessor stops executing software and places address, data, and control bus at high-impedance

if a logic 0, software executes normally

HLDA

Hold acknowledge indicates the 8086/8088 has entered the hold state.

$\overline{\text{SS0}}$

The $\overline{\text{SS0}}$ status line is equivalent to the S0 pin in maximum mode operation.

Signal is combined with **IO/$\overline{\text{M}}$** and **DT/$\overline{\text{R}}$** to decode the function of the current bus cycle.

Maximum Mode Pins

In order to achieve maximum mode for use with external coprocessors, connect the MN/MX pin to ground.

$\overline{S2}$, $\overline{S1}$, and **$\overline{S0}$**

Status bits indicate function of the current bus cycle.

$\overline{S2}$	$\overline{S1}$	$\overline{S0}$	*Function*
0	0	0	Interrupt acknowledge
0	0	1	I/O read
0	1	0	I/O write
0	1	1	Halt
1	0	0	Opcode fetch
1	0	1	Memory read
1	1	0	Memory write
1	1	1	Passive

- Normally decoded by the 8288 bus controller

$\overline{RQ}$/GT1

The **request/grant** pins request direct memory accesses (DMA) during maximum mode operation.

- Bidirectional; used to request and grant a DMA operation

$\overline{LOCK}$

The **$\overline{lock}$** output is used to lock peripherals of the system. This pin is activated by using the LOCK prefix on any instruction.

QS_1 and **QS_0**

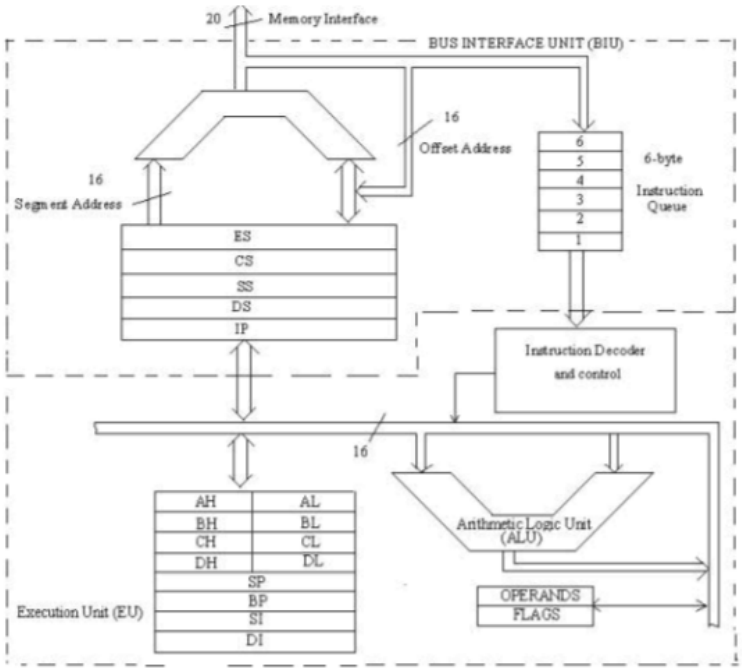
The **queue status** bits show the status of the internal instruction queue.

- provided for access by the 8087 coprocessor

Internal Architecture of 8086

The basic architecture of the 8086 is shown below. It consist of two independent units:

1. The Bus Interface unit
2. The execution unit



The Bus Interface Unit

The BIU primarily interacts with the System Bus. It performs all activities related to fetch cycle such as:

- Calculating the physical address of the next instruction
- Fetching the instruction
- Reading or writing data memory or I/O port from memory or Input/Output

The instructions/data are then passed to the execution unit

BIU consists of :

(a) The Instruction Queue

It is used to store the instruction "bytes" fetched. It uses underlying queue data structure

The advantage of this queue would only be if the next
expected instructions are fetched in advance, thus, allowing a pipeline of fetch and execute
cycles.

(b) The Segment Registers

In 8086, the memory is byte organized, ie the memory address is a byte address. The number of bits fetched is 16 at a time.

The BIU contains four sixteen-bit registers, viz., the CS: Code Segment, the DS: Data Segment, the SS: Stack Segment, and the ES: Extra Segment. But what is the need of the segments: Segments logically divide a program into logical entities of Code, Data and Stack
each having a specific size of 64 K

The segment register holds the upper 16 bits of the starting address of a logical group of memory, called the segment. The main advantages of using segments are:

- Logical division of program thus enhancing overall possible memory use
- The addresses are relocatable as they are the offsets. Thus segmentation supports relocatability.
- Although the size of address is 20 bits, only the maximum segment size, 16 bits needs to be kept in instruction thereby reducing instruction length.

(c) Instruction Pointer

Points to the offset of the current instruction in the code segment. It is used for calculating address of the instruction

Execution Unit

Execution unit performs all the ALU operations. In 8086, the EU is 16bits. It also contains the control unit which instructs the bus interface unit about which memory location to access and what to do with the data.

It also performs decoding and execution of instructions. It consists of the following:

(a) Control circuitry, Instruction decoder and ALU

The 8086 control unit is primarily micro-programmed control. In addition it has an
instruction decoder, which translates an instruction into sequence of micro operations. The
ALU performs the required operations under the control of CU which issues the necessary
timing and control sequences

(b) Registers

All CPUs have a defined number of operational registers. 8086 has several general and special purpose registers. They are all 16-bit registers, and the gen. purpose registers can be used as 8 bit or 16bit registers.

The register set of 8086 can be categorized into 4 groups

General Purpose Registers

AX, BX, CX ,DX → 16bit gen. purpose registers.

AX → 16bit accumulator

All data register can be used as either 16 bit or 8 bit. BX is a 16 bit register, but BL
indicates the lower 8-bit of BX and BH indicates the higher 8-bit of BX.
The register BX is used as offset storage for forming physical address in case of certain
addressing modes

CX → default counter in case of string and loop instructions.

DX → may be used as implicit operand or destination.

Segment Registers

There are 4 segment registers in 8086

- Code segment register CS, used for addressing memory location where the executable program is stored in the code segment
- Data segment DS, points to the segment of the memory where the data is stored
- Extra segment ES, also refers to a segment of memory which is another data segment
- Stack segment SS, used for addressing stack segment of the memory.

While addressing any location in the memory bank, the physical address is calculated
from two parts:
- The first is segment address, the segment registers contain 16-bit segment base addresses,
related to different segment.
- The second part is the offset value in that segment.

Pointers and Index Registers

The pointers contain offset within the particular segments

→ The pointer register IP contains offset within the code segment

→ The pointer register BP contains offset within the data segment.

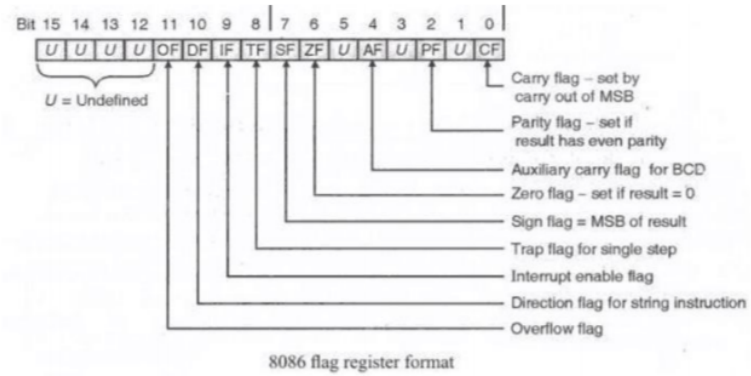
→ The pointer register SP contains offset within the stack segment.

The index registers are used as general purpose registers as well as offset storage in case of indexed, base indexed, and relative base indexed addressing modes.

SI → Used to store the offset of source data in data segment

DI → Used to store the offset of destination in data or extra segment.

8086 Flag Register



8086 flag register format

4. Show how the flag register is affected by the addition of 38H and 2FH

MOV BH, 38H

ADD BH, 2FH

38H = 0011 1000

+2FH = 0010 1111

67H = 0110 0111 AC=1, CF=0, ZF=0, SF=0, PF=0

Addressing modes

Addressing mode specifies the way in which the operand of the instruction is specified.

1. Immediate

In this type of addressing, immediate data is part of instruction, and appears in the form of successive byte or bytes

ex: MOV AX, 0005H , where 0005H is the immediate data.

2. Direct

In the direct addressing mode, a 16bit offset address is directly mentioned in the instruction

ex: MOV AX, [5000H]

Here, data resides in a memory location in the data segment, whose effective address may be found or computed using 5000H as the offset address and the content of DS as segment address.

The effective address here is 10H*DS+5000H

3. Register

The data is stored in a register and it is referred using the particular register. All registers except IP may be used in this mode.

4. Register Indirect

In this addressing mode, the offset address of data is stored either in BX, SI or DI registers. The default segment is DS or ES.

ex: MOV AX, [BX]

Here data is present in a memory location DS whose offset address is in BX. The effective address is thus 10H*DS + [BX]

5. Indexed

In this mode, offset of operand is stored in one of the index registers. DS and ES are the default segments for SI and DI respectively.

ex: MOV AX, [SI]

Here the data is present in DS and the offset value is stored in SI.

Effective address = 10H*DS+[SI]

6. Register Relative

In this mode, data is available at the effective address formed by adding an 8/16 bit displacement value to the content of any of the registers BX, BP, SI and DI in the default (DS or ES) segment.

ex: MOV AX, 50H [BX]

Here, effective address is 10H*DS + 50H + [BX]

7. Based indexed

The effective address of data in this mode is formed by adding the content of the base register to the content of an index register. The default segment may be DS or ES.

ex: MOV AX, [BX][SI]

Here BX is the base register and SI is the index register. The effective address is 10H*DS + [BX]+[SI]

8. Relative Based Indexed

The effective address is formed by adding a 8/16bit displacement value with the sum of contents of any of the base registers and any one of the index registers.

ex: MOV AX, 50H [BX] [SI]

Effective address = 10H*DS + [BX] + [SI] + 50H.

Machine Language Instruction Format

There are 2 machine language formats of the 8086-Core2 instructions

1. 16 bit instruction mode
2. 32 bit instruction mode

The first two bytes of 32bit instruction mode format are called override prefixes because they are not always present. The first modifies the size of the operand address used by the instruction and the second modifies the register size.

The first field is known as the opcode field which indicates the type of operation to be performed by the CPU. The second field is the operand field. The operand may consist of source/destination operand.

The source operand can be register/memory location or immediate data. The destination operand can be register or memory location or memory location with displacement.

The opcode

- Either one or two bytes long.
- The first 6 bytes of the opcode are the binary opcode

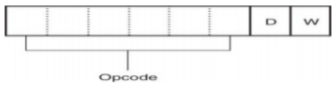


Figure: Byte 1 of Opcode

- D (direction) bit specifies the direction of data flow

  → If D=1, data flows *to* the register

  → If D=0, data flows to the R/M field from the REG field

- W bit:

  → W=1 ⇒ the operand is a word or doubleword.

  → W=0 ⇒ The operand is a byte.

The W bit appears in most instructions, while the D bit generally just appears in MOV and similar instructions.

Byte 2 of Machine language instructions

Byte 2 specifies the details about the operand. It contains 3 fields : MOD (mode field) → 2 bits, REG (register field) → 3 bits and R/M (Register/Memory) field → 3 bits.

REG field (3 bits)

Selects the register for the first operand which may be source or destination

REG	W=0 (byte)	W=1 (word)	W=1 (double word)
000	AL	AX	EAX
001	CL	CX	ECX
010	DL	DX	EDX
011	BL	BX	EBX
100	AH	SP	ESP
101	CH	BP	EBP
110	DH	SI	ESI
111	BH	DI	EDI

MOD field (2 bits)

The MOD field specifies the addressing mode for the selected instruction. It selects the type of addressing and whether a displacement is present with the selected type.

MOD	Explanation
00	No Displacement
01	8 bit displacement follows
10	16 bit displacement follows
11	R/M is a register

If the MOD field contains a 11, it selects the register-addressing mode. Register addressing uses the R/M field to specify a register instead of a memory location.

If the MOD field contains a 00, 01, or 10, the R/M field selects one of the data memory-addressing modes. When MOD selects a data memory-addressing mode, it indicates that the addressing mode contains no displacement (00), an 8-bit sign-extended displacement (01), or a 16-bit displacement (10).
For example, The MOV AL,[DI] instruction is an example showing no displacement;
a MOV AL,[DI + 2] instruction uses an 8-bit displacement (+ 2);
and a MOV AL,[DI + IOOOH] instruction uses a 16-bit displacement (+1000H).

R/M field (3 bits)

Specifies the second operand as a register or memory location based on the MOD field.

MOD=11		
R/M	W=0	W=1
000	AL	AX
001	CL	CX
010	DL	DX
011	BL	BX
100	AH	SP
101	CH	BP
110	DH	SI
111	BH	DI

When MOD=00, 01 or 10, R/M selects the second operand as memory location which may be source or destination.

R/M	MOD=00	MOD=11	MOD=10
000	[BX+SI]	[BX+SI+D8]	[BX+SI+D16]
001	[BX+DI]	[BX+DI+D8]	[BX+DI+D16]
010	[BP+SI]	[BP+SI+D8]	[BP+SI+D16]
011	[BP+DI]	[BP+DI+D8]	[BP+DI+D16]
100	[SI]	[SI+D8]	[SI+D16]
101	[DI]	[DI+D8]	[DI+D16]
110	Direct addressing	[BP+D8]	[BP+D16]
111	[BX]	[BX+D8]	[BX+D16]