UNIT-1

SOME IMPORTANT TERMS:

Microprocessor: It is a semiconductor device which is manufactured by using LSI or VLSI technology, which includes ALU, Control unit and a group of Registers in a single Integrated circuit.

Microcontroller: It is a device that includes microprocessor, memory, and I/O signal lines on a single chip, fabricated using VLSI technology.

Microcomputer: A digital computer having a microprocessor as its Central Processing Unit is called Microcomputer. So, a microprocessor combined with memory, an input device and an output device forms a microcomputer.

Bus: A group of wires or lines used to transfer bits between the microprocessor and other components of the computer system. Or a path used to carry signals, such as connection between memory and the microprocessor in a digital computer.

Hardware: The physical devices and circuitry of the computer is called Hardware.

Software: The programs written for the computer is referred to as software.

Firmware: The programs stored in ROMs or in other devices which permanently keep their stored information are referred as Firmware.

- ✓ In general the width of the data bus is equal to the bit capacity of the microprocessor.
- ✓ In general the internal architecture of the microprocessor depends on the bit capacity of the microprocessor.
- ➢ **Bit**-a binary digit.0 or 1
- ➢ **Nibble**-a group of four bits
- ➢ **Byte**-a group of eight bits
- ➢ **Word**-a group of 16 bits or a group of bits the computer recognizes and processes at a time.
- ➢ **Double word**-a group of 32 bits.

Instruction: A command in binary that is recognized and executed by the computer to accomplish a task. Some instructions are designed with one word, and some require multiple words.

Mnemonic: A combination of letters to suggest the operation of an instruction.

Program: A set of instructions written in specific sequence for the computer to accomplish a given task.

Machine language: The binary medium of communication with a computer through a designed set of instructions specific to each computer.

Assembly language: A medium of communication with a computer in which programs are written in mnemonics.

Low-level language: A medium of communication that is machine-dependant or specific to a given computer. The machine and the assembly languages of a computer are considered low level languages. Programs written in these languages are not transferable to different types of machines.

High-level language: A medium of communication that is independent of a given computer. Programs are written in English like words, and they can be executed on a machine using translator (a compiler or interpreter).

Source code: A program written either in mnemonics of as assembly language or in English like statements of high level language (before it is assembled or compiled).

Assembler: A computer program that translates an assembly language program from mnemonics to the binary machine code of a computer.

Compiler: A program that translates English-like words of a high level language into the machine language of a computer.

Interpreter: A program that translates the English –like statements of a high level language into the machine language of a computer.

Operating System: A set of programs that manages interaction between hardware and software.

ASCII: American Standard Code for Information Interchange. This is 7-bit alphanumeric code with 128 combinations.

MIPS: Million instructions per second is a measure of the speed at which the instructions are executed in a processor.

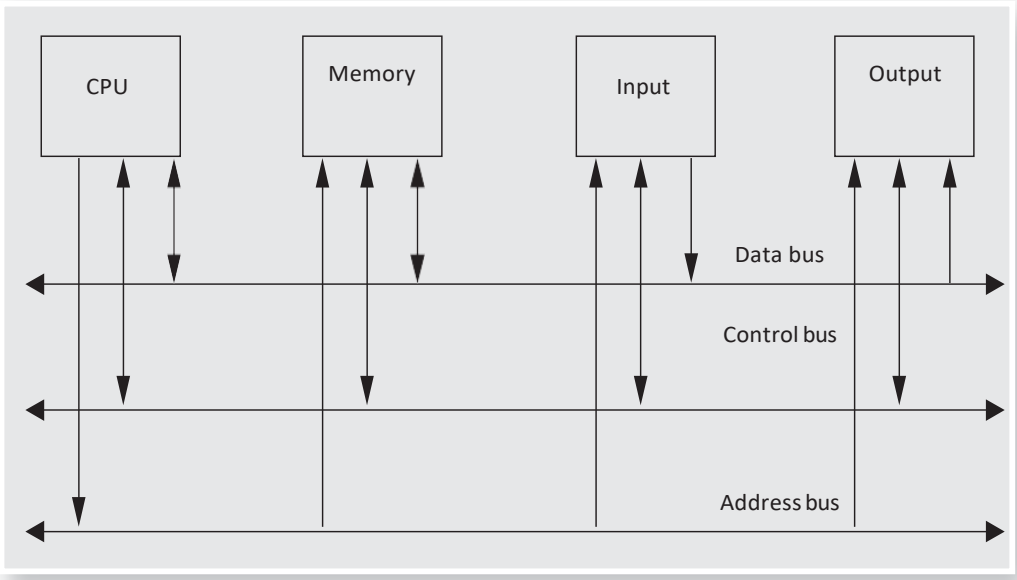
MICROPROCESSORS AND MICROCONTROLLERS:

Microprocessor: Microprocessor is a multi-purpose programmable device that reads the binary instructions from the memory, accepts the data as its input, process the data according to those instructions and provides the result as output. These microprocessors need additional circuitry elements such as memory devices and I/O ports to connect the input and output devices. All microprocessor-based systems need two types of memories—RAM and ROM.

Microcontroller: Microcontrollers are microprocessors designed especially for control applications. Microcontrollers contain memory units and I/O ports inside a chip, in addition to the CPU. Microcontrollers are otherwise called embedded controllers; they are generally used to control and operate smart machines. Some of the machines using microcontrollers are microwave ovens, washing machines, sewing machines, automobile ignition systems, computer printers, and fax machines.

MICROPROCESSOR-BASED SYSTEM:

A basic general-purpose microprocessor is called a microcomputer system. The system consists of CPU,



memory, and I/O ports as shown in Fig.

Memory: It is a medium that stores binary information. The memory section usually consists of a mixture of RAM and ROM. It may also have magnetic floppy disks, magnetic hard disks, or optical disks. Memory has two purposes .The first purpose is to store the binary code for the sequences instructions you want the computer to carry out. The second purpose of the memory is to store the binary coded data with which the computer is going to be working.

Input: It is a device that allows the computer to take in data from the outside world.

Output: It is a device that allows the computer to send date to the outside world. Peripherals such as keyboards, video display terminals, printers, and modems are connected to the I/O section. These allow the user and the computer to communicate with each other. The actual physical devices used to interface the computer buses to external systems are often called **Ports.** An ***input port*** allows data from keyboard, an A/D converter, or some other source to be read into the computer under control of the CPU. An ***output port*** is used to send data from the computer to some peripheral, such as video display terminal printer, or a D/A converter.

CPU: The Central Processing Unit or CPU controls the operation of the computer. In a microcomputer the CPU is a microprocessor. The CPU or Microprocessor fetches binary ceded instructions from the memory, decodes the instructions into a series of simple actions, and carries out these actions in sequence of steps.

Address bus: The address bus consists of 16, 20, 24, or 32… parallel signal lines. These address lines are used to send a address of the memory location or a device address from the microprocessor unit to the memory or the peripheral. The address bus is always unidirectional. Address always goes out of the microprocessor. The number of memory locations that the microprocessor can address is determined by the number of address lines. If the microprocessor has N address lines, then it can directly address 2^N memory

locations. For ex: An address bus of eight bits corresponds to eight lines of addresses and can thus address 2^8 different memory locations. These addresses are written in hexadecimal number format as 00H– FFH and can be used to for 256 different locations.

Data bus: The data bus consists of 8,16,or 32…parallel lines .A group of lines used to transfer a data between the microprocessor and peripherals(or memory).The data bus is always bi-directional. The width of the data bus determines the data size that can be transferred. An 8-bit processor will generally have an 8-bit data bus and a 16-bit processor will have a 16-bit data bus.

Control bus: The control bus consists of 4 to 10 parallel signal lines. The microprocessor sends out signals on the control bus to enable the outputs of addressed memory devices or port devices. Typical control bus signals are Memory Read, Memory Write, I/O Read, and I/O Write.

ORIGIN OF MICROPROCESSORS:

The microprocessor is the greatest invention of the 20^{th} century. Its evolution started from the earlier mechanical calculating devices, in the 1930s. These devices used mechanical relays. Later, in the 1950s, these devices were replaced by vacuum tubes. The vacuum tubes were quickly replaced by transistors. The breakthrough in transistor technology led to the introduction of minicomputers in the 1960s and the personal computer revolution in the 1970s.

The transistor technology led to the development of complex devices called integrated circuits (ICs). The microprocessor, or microprocessing unit (MPU), later evolved as an IC and was designed to fetch instructions and execute the predefined arithmetic and logic functions.

1. First generation (1971 – 1973)

Intel Corporation introduced 4004, the first microprocessor in 1971. It is evolved from the development effort while designing a calculator chip.

There were three other microprocessors in the market during the same period:

- Rockwell International's PPS-4 (4 bits)
- Intel's 8008 (8 bits)
- National Semiconductor's IMP-16 (16 bits)

They were fabricated using PMOS technology which provided low cost, slow speed and low output currents. They were not compatible with TTL.

2. Second Generation (1974 – 1978)

Marked the beginning of very efficient 8-bit microprocessors. Some of the popular processors were:

- Motorola's 6800 and 6809
- Intel's 8085
- Zilog's Z80

They were manufactured using NMOS technology.

This technology offered faster speed and higher density than PMOS

It is TTL compatible.

3. Third generation microprocessors (1979 – 80)

This age is dominated by 16 – bits microprocessors

Some of them were:

- Intel's 8086/80186/80286
- Motorola's 68000/68010

They were designed using HMOS technology

HMOS provides some advantages over NMOS as

Speed-power-product of HMOS is four times better than that of NMOS

HMOS can accommodate twice the circuit density compared to NMOS

Intel used HMOS technology to recreate 8085A and named it as 8085AH with a higher price tag.

4. Fourth Generation (1981 – 1995)

- This era marked the beginning of 32 bits microprocessors
- Intel introduced 432, which was bit problematic
- Then a clean 80386 in launched.
- Motorola introduced 68020/68030.

They were fabricated using low-power version of the HMOS technology called HCMOS.

Motorola introduced 32-bit RISC processors called MC88100.

5. Fifth Generation (1995 – till date)

This age the emphasis is on introducing chips that carry on-chip functionalities and improvements in the speed of memory and I/O devices along with introduction of 64-bit microprocessors.

Intel leads the show here with Pentium, Celeron and very recently dual and quad core processors working with up to 3.5GHz speed.

Table Comparison of general-purpose processors

General-purpose processors	Transistors	CPU speed	Data length (bits)
8080	6,000	2 MHz	8
8085	6,500	3 MHz	8
8088	29,000	3 MHz	16
8086	30,000	4 MHz	16
80286	1,34,000	6 MHz	16
80386	2,75,000	16 MHz	16/32
80486	12,00,000	33 MHz	16/32
Athalon XP	37,00,000	2.8 GHz	16/32/64
Celeron	75,00,000	1.06–2 GHz	32
Pentium II	75,00,000	233–450 MHz	32
Pentium III	95,00,000	450 MHz–1 GHz	32
Pentium III Xeon	2,81,00,000	500 MHz–1 GHz	32
Pentium 4	5,50,00,000	1.4–2.2 GHz	32
IBM PowerPC G3	65,00,000	233–333 MHz	32
PowerPC G4	1,05,00,000	400–800 MHz	32

CLASSIFICATION OF MICROPROCESSORS:

Microprocessors can be classified based on their specifications, applications, and architecture.

Based on the size of the data that the microprocessors can handle, they are classified as 4-bit, 8-bit, 16-bit, 32-bit, and 64-bit microprocessors.

Based on the application of the processors, they are classified as follows:

(i) General-purpose processors
(ii) Microcontrollers
(iii) Special-purpose processors

General-purpose processors are those that are used in general computer system integration and can be used by the programmer for any application. Common microprocessors from Intel 8085 to Intel Pentium are examples of general-purpose processors.

Microcontrollers are microprocessor chips with built-in hardware for the memory and ports. These chips can be programmed by the user for any generic control application.

Special-purpose processors are designed specifically to handle special functions required for an application. Digital signal processors are examples of special-purpose processors; these have special instructions to handle signal processing. Application-specific integrated circuit (ASIC) chips are also examples of this category of microprocessors.

Based on the architecture and hardware of the processors, they are classified as follows:

(i) RISC processors
(ii) CISC processors
(iii) VLIW processors
(iv) Superscalar processors

RISC is a processor architecture that supports limited machine language instructions. *RISC processors* can execute programs faster than CISC processors.

CISC *processors* have about 70 to a few hundred instructions and are easier to program. However, CISC processors are slower and more expensive than RISC processors.

Very long instruction word (VLIW) *processors* have instructions composed of many machine operations. These instructions can be executed in parallel. This parallel execution is called instruction-level

parallelism. VLIW processors also have a large number of registers.
Superscalar processors use complex hardware to achieve parallelism. It is possible to have overlapping of instruction execution to increase the speed of execution.

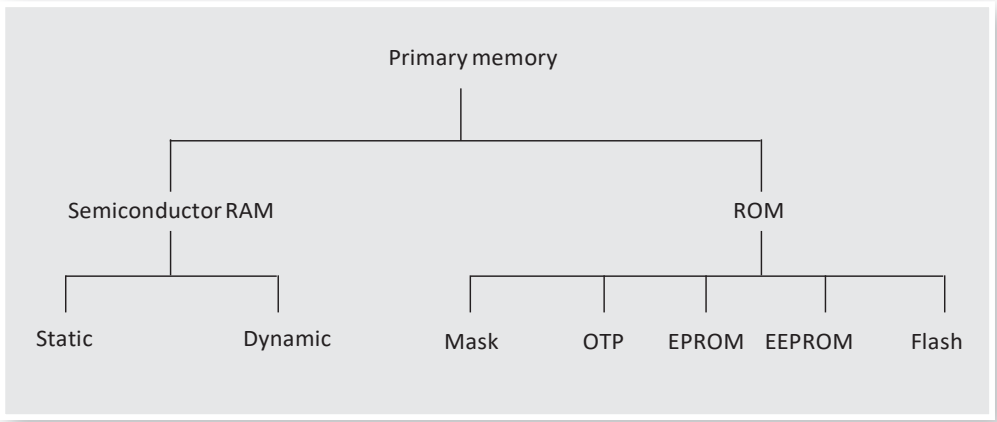
TYPES OF MEMORY:

A microcomputer memory can be logically divided into four groups:

(i) Processor memory/register
(ii) Cache memory
(iii) Primary or main memory
(iv) Secondary memory

Processor memory refers to a set of CPU registers. Processor registers are the first set of storage devices available for the programmers to store any data, but they are generally few in number—up to a few tens or hundreds. As these registers are available within the processor, they are the fastest memory registers. The main disadvantage is the cost involved, which restricts the number of registers and their bytes.
Cache memory is the fastest external memory; it is placed close to the processor. The instructions to be executed are placed in the cache memory for access by the processor. These are a few kilobytes in size. Cache memory contains volatile semiconductor RAMs. The processor fetches instructions from the cache memory and if an instruction is not in cache, it refers to the primary memory.
Primary memory is the storage area from which all the programs are executed. All the programs and corresponding data for execution must be within the primary memory. The primary memory is much larger than the processor memory and the cache memory but its operating speed is slower. The primary memory in a system varies from few KB to a few MB.
Secondary memory refers to the storage medium for huge files such as program source codes, compilers, operating systems, etc. These are not accessed directly or very frequently by the microprocessor in a computer system. Secondary memory consists of slow devices such as magnetic tapes and optical disks. Sometimes, they are referred to as auxiliary or backup store. Stored information in a magnetic tape or magnetic disk is not lost when power is turned off. Therefore, these storage devices are called non-volatile memories.
Classification of primary memory Primary memory normally includes ROM and RAM, which are further classified as shown in Fig. Microprocessor-based systems have at least one RAM and one ROM chip.
RAM devices allow both reading and writing to their memory cells. In static RAM devices, bits are stored as the status of on/off switches. There are no charges involved and hence, no charges to leak. However, **static RAM** devices have complex construction and hence larger size per unit storage. So they are more expensive. Static RAMs are comparatively faster and are used in cache memories.

Fig. Classification of primary memories



In **dynamic RAM** devices, the data bits are stored as charge in capacitors. Since capacitor charge has a tendency to leak, these devices need refreshing even when they are powered. However, they have simpler construction and smaller size per unit storage. These devices are less expensive and comparatively slower.
As the name implies, a ROM permits only read access. There are many kinds of ROMs:

(i) Mask programmable ROMs (MPROMs) are custom-made for the customer; their contents are programmed by the manufacturer. Since they are mass produced, they are inexpensive. The customer cannot erase or program it afterwards.

(ii) Programmable ROMs (PROMs) or one-time programmable (OTP) ROMs are devices that can be programmed by the user in his/her place using special equipments. The main disadvantage of PROMs is that they cannot be erased and reprogrammed.

(iii) Erasable and programmable ROMs (EPROMs) allow the erasure and reprogramming of the content by the user. In an EPROM, programs are entered using electrical impulses and the stored information is erased using ultraviolet rays.

(iv) Electrically erasable PROMs (EEPROMs) or electrically alterable ROMs (EAROMs) allow the users to electrically erase and reprogram its contents. EEPROMs are different from RAMs in that electrical signals are required to erase and program them. EEPROMs require a higher voltage for erasing and programming than the normal 5V supply.

(v) Flash memory devices are a group of single transistor cell EPPROMs. Cell sizes are about half the size of a two-transistor EEPROM. The operation requires bulk erasure of a large portion of the memory array.

INPUT AND OUTPUT DEVICES:

Input and output devices permit the user to feed data to the computer and retrieve the computed result from it. Sometimes, the input and output devices can communicate among themselves. In general, computer systems have I/O ports; I/O devices are connected to these ports for data transfer. Basically, the ports are digital registers that allow the computer to transfer data between the I/O devices using additional control signals. These control signals allow error-free transfer of data.

The common input devices used in almost all systems is the keypad. Microprocessor-based basic microcomputer systems use simple numeric keypads. However, advanced computer systems use keyboards with a large number of keys involving alphabets, numbers, and special characters. Nowadays, a number of optical devices and scanners such as mouse, joystick, and bar code scanners are also being used as input devices. Microcomputer systems also use different types of sensors for data input. These sensors need data converters such as analog to digital converters. Any introductory course on microprocessors should cover the interfacing of data converters, keypads, and switches.

An output device is a device through which the user can receive the results from the computer. The output can be a rapidly changing display or printed material. Other forms of output are sounds and alarms. The simplest output devices, used in almost all microprocessor-based systems and computer systems, are LEDs, seven- segment LED displays, and LCD displays. The advanced video display terminals (either cathode-ray tubes or LCDs) and ink-jet and laser printers are the common output devices nowadays. Some output devices can be used to directly control machineries. Some devices, such as display terminals with touch screen, may provide both input and output. Modems and other network interface cards can also be called output devices as they enable the transmission and reception of data between computers.

TECHNOLOGY IMPROVEMENTS ADAPTEDTO MICROPROCESSORS AND COMPUTERS:

Technological improvements are taking place rapidly in microprocessor, microcomputer, and personal computer systems. Some of these improvements are listed here:

(i) Increase in data bus/address bus width: The processing capability of the microprocessor can be drastically improved by increasing data size. This development can be seen clearly from the advancements in microprocessors (Section 1.5).

(ii) Increase in speed: As the data to be processed by the microprocessors and computers increased in volume, it became necessary to increase the speed of the processor. With high speed processors, the user can get results quickly, even with large data volumes.

(iii) Reduction in size and increase in capability: The trend in microprocessor technology is to include a large number of peripherals such as memory and I/O ports within a single chip. Microcontrollers are manufactured in this fashion. In addition, developments in large scale integration have led to the manufacture of small microprocessor chips with large built-in peripherals. Processors with a large amount of flash memory are now available in the market.

(iv) Development of external peripherals: The use of computers in all fields have resulted in the development of many fast and advanced peripheral devices. For example, the application of microprocessors in medicine has resulted in the development of many handheld electronic devices with

specialized input sensors, output printers, etc. Faster peripherals can increase the speed of processor execution and provide a good user interface.

(v) Increase in memory unit size and speed: The developments in IC technology have led to a reduction in the size of the memory units and an increase in memory speed. This reduces the memory access time of the processor and results in higher speed of execution. More amount of memory per unit area is possible.

(vi) Microprocessors are largely used in handheld devices operated from a battery source. This has resulted in research on the reduction of power consumption in microprocessor chips. As power consumption is reduced, these devices work for more time once the batteries are fully charged. There are many devices operating at 3.3 V or even lower voltages and have low power consumption.

8085 microprocessor Review (brief details only)

ARCHITECTURE OF 8085 MICROPROCESSOR:

The functional block diagram or architecture of 8085 Microprocessor is very important as it gives the complete details about a Microprocessor. Fig. shows the Block diagram of a Microprocessor.

The main features of 8085 μp are:

1) It is a 8 bit microprocessor.
2) It is manufactured with N-MOS technology.
3) It has 16-bit address lines and hence can address up to 2^{16} = 65536 bytes (64KB) memory locations through A_0-A_{15}.
4) Data bus is a group of 8 lines $D_0 - D_7$.
5) In 8085, the lower 8-bit address bus (A_0-A_7) and data bus (D_0-D_7) are multiplexed to reduce number of external pins. But due to this, external hardware (latch) is required to separate address lines and data lines.
6) It can operate with a 3MHz clock frequency. The 8085A-2 version can operate at the maximum frequency of 5MHz.
7) It supports 74 instructions with the following addressing modes:

a) Immediate b) Register c) Direct d) Indirect e) Implied

8) It provides five hardware interrupts: TRAP, RST 7.5, RST 6.5, RST 5.5and INTR.
9) It has serial I/O control which allows serial communication.
10) It has 8-bit accumulator, flag register, instruction register, six 8-bit general purpose registers (B,C,D,E,H,and L) and two 16-bit registers (SP and PC). Getting the operand from the general purpose registers is more faster than from memory.

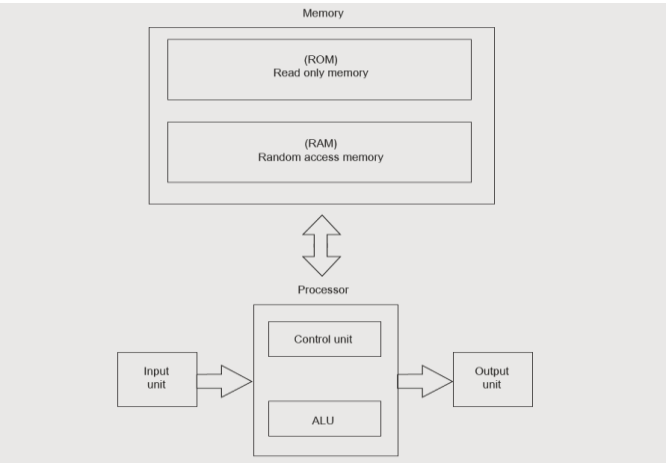


Figure: Microprocessor System

11) It requires a signal +5V power supply.
12) It is enclosed with 40 pins DIP (Dual in line package).

A microprocessor system consists of three functional blocks: central processing unit (CPU), input and output units, memory units as shown in figure1.1. The central processing unit contains several registers, arithmetic logic unit (ALU) and control unit.

Architecture

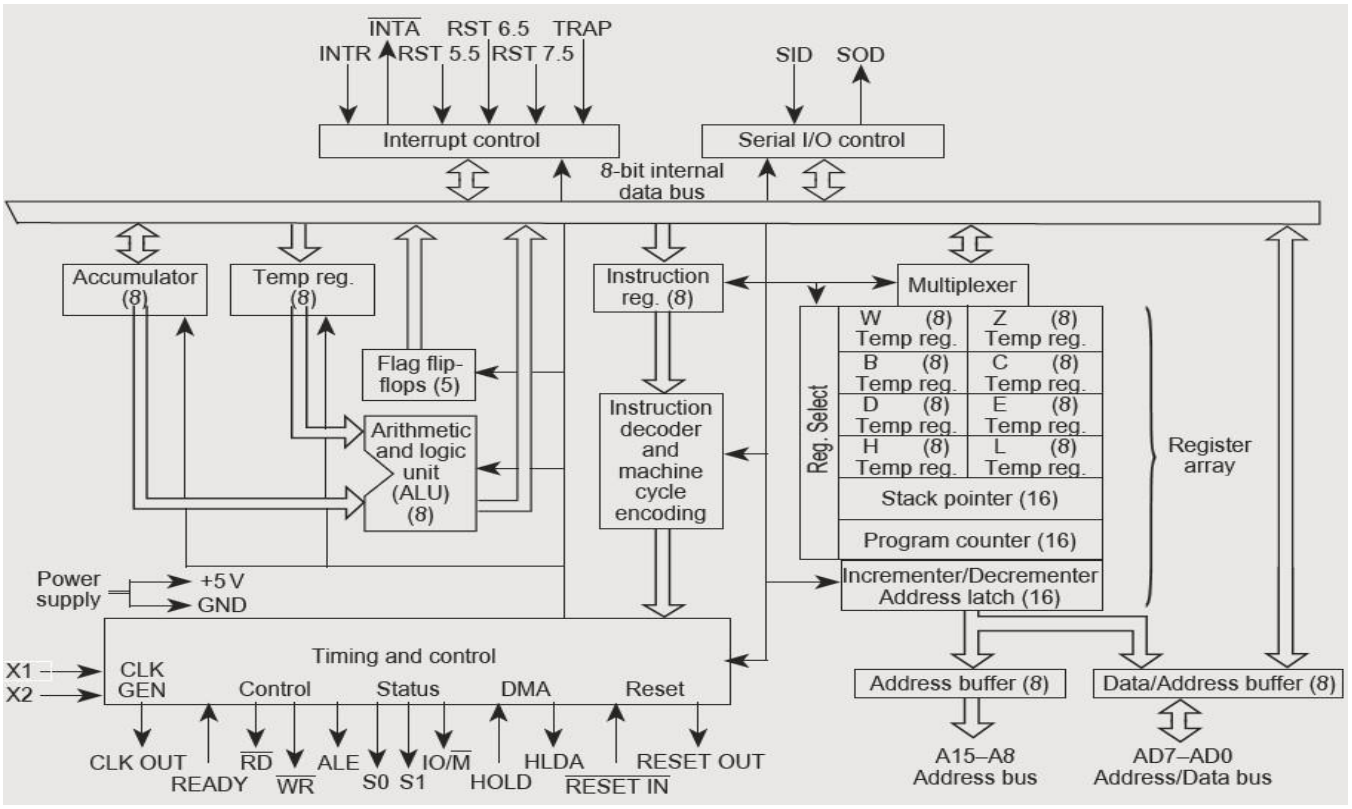
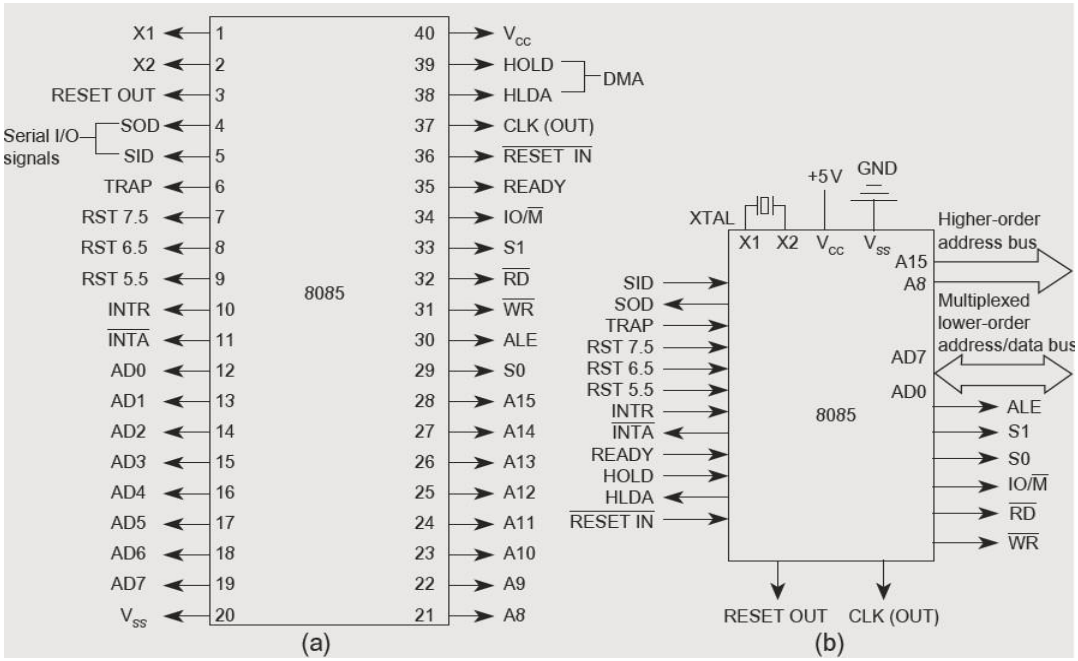


Figure: Architecture of 8085



Pin Diagram of 8085

Register Organization

The register organization of 8085 is shown in figure

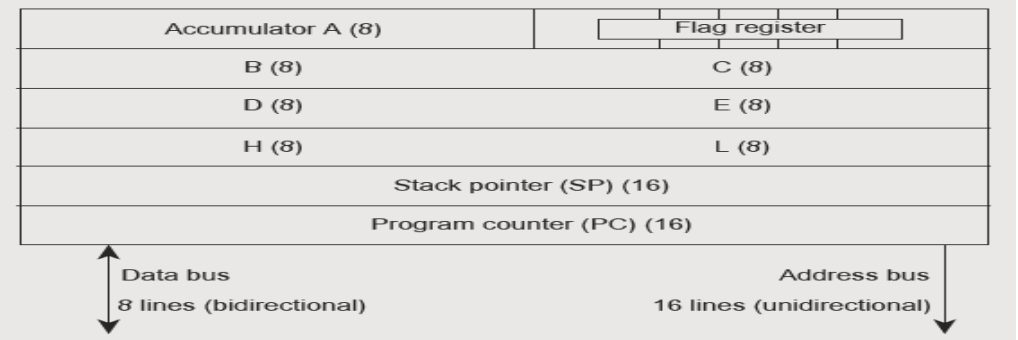


Figure: Register Organization of 8085

Registers: The 8085 has six general purpose registers to store 8-bit data. They are identified as B, C, D, E, H and L. They can be combined as register pairs: B-C, D-E, and H-L to store and perform 16-bit operations.

The 8085 special purpose registers are the Accumulator, Flag register, Program Counter (PC) and Stack Pointer (SP).

Accumulator: This is an 8 bit register that is part of arithmetic and logical unit (ALU).This is used to store 8-bit data and perform arithmetic and logical operations. The result of an operation is stored in the accumulator.

Program Counter (PC): This is a 16-bit register, which always points to the address of next instruction to be fetched.

Stack Pointer (SP): This is a 16-bit register that points to a memory location in R/W memory, called as stack.

Flags register: The ALU includes five flip-flops, which are set or reset after an operation according to data conditions of result in the accumulator. They are called as Zero (Z), Carry (CY), Sign (S), Parity (P), and Auxiliary Carry (AC) flags .The flag register format of 8085 is shown in figure

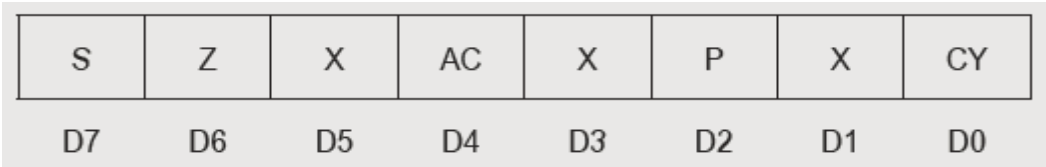


Figure: Flag Register Format of 8085

Z-Zero: The zero flag is set if the result is zero, otherwise it is reset.

CY-Carry: If an arithmetic operation results in a carry, the carry flag is set otherwise it is reset.

S-Sign: The sign flag is set if bit D7 of the result is one, otherwise it is reset.

AC- Auxiliary Carry: In a BCD arithmetic operation, when a carry results from digit D3 and passes on to digit D4, the AC flag is set.

P-Parity: If the result has an even number of 1's, the parity flag is set. For odd number of 1's, the flag is reset.

Two additional 8-bit temporary registers W and Z are included in the register array. They are not programmable.

The ALU (arithmetic and logic unit)

The arithmetic logic unit of 8085 performs addition, subtraction, increment, decrement and comparison arithmetic operation and logical operations such as AND, OR, exclusive-OR, complement. The ALU includes the accumulator, the temporary register, the arithmetic logic circuits, and five flip-flops. The temporary register is used to hold data during arithmetic logic operation. The result is stored in accumulator, and flags are set or reset according to result of operation.

The 8085 is also called accumulator oriented processor as one of the data in the ALU operation is stored in the accumulator. The other data may be in memory or in register.

Timing and Control unit

The timing and control unit synchronizes all microprocessor operations with clock and generates control and status signals (RD, WR) for communication between microprocessor and peripherals.

Instruction Register and Decoder

The Instruction register/decoder is an 8-bit register that is part of the ALU. When an instruction is fetched from memory, it is loaded in the instruction register. The decoder decodes the instruction. The instruction register is not programmable.

Pins and Signals

Intel 8085 has 40 pins, operates at 3MHz clock and requires +5V for power supply. The pin diagram of 8085 is shown in figure 1.6.The signals can be classified into six groups i.e. Address Bus, Data Bus, Control and Status Signals, Power supply and System Clock, Externally Initiated Signals, and Serial I/O signals.

Address and Data Buses

The 8085 has 16 bit (A15-A0) address lines which are unidirectional and 8-bit (D7-D0) data lines which are bidirectional. A8 - A15 (Output-3 state: higher order address bus): The most significant 8 bits of the memory address or the 8 bits of the I/0 addresses, tri-stated during Hold and Halt modes.

AD0-AD7 (Input/output- 3 state: multiplexed address/data bus): Lower 8 bits of the memory address (or I/0 address) appear on the bus during the first clock cycle of a machine state. It then becomes the data bus during the second and third clock cycles. Tri- stated during Hold and Halt modes.

Control and Status Signals

ALE (Output) Address Latch Enable:

This output signal indicates the availability of the valid address on the address/data lines. It occurs every time during the first clock cycle of the 8085 machine cycle operation. It is used to latch the low order address from multiplexed bus and generate a separate address (A7-A0).

RD (Output- 3 state) READ:

This is a Read control signal (active low).This indicates that the selected memory or I/O device is to be read and data are available on the data bus.

WR (Output- 3state) WRITE:

This is a Write control signal (active low).This indicates that the data on the data bus is to be written into the selected memory or I/O location.

IO/M (Output: 3 state):

This is a status signal used to differentiate between I/O and memory operation. When it is high, it indicates an I/O operation. When it is low, it indicates a memory operation.

SO, S1 (Output): These are status signals and identify various operations.

S1	S0	States
0	0	Halt
0	1	Write
1	0	Read
1	1	Fetch

Figure: 8085 Status Signals

Externally initiated signals

HOLD (Input):

It is an active high signal used in the direct transfer of data between a peripheral device and memory locations. This type of data transfer is called as direct memory access (DMA).During this transfer, the microprocessor loses control over the address and data buses and these buses are tri-stated.

Logic 1 on the Hold pin indicates that another controller, generally the DMA controller, is requesting the use of the address and data buses.

HLDA (Output): Hold Acknowledge

This is an active high signal, and acknowledges HOLD request.

INTR (Input): Interrupt Request

The INTR is used as a general purpose interrupt.

INTA (Output): Interrupt Acknowledge

This is an active low signal and is used to acknowledge an interrupt

RST 7.5, RST 6.5, and RST 5.5 : Restart Interrupts (input)

These three are hardware vectored interrupt signals.

TRAP (Input) :

This is a non-maskable interrupt and has highest priority.

RESET IN (Input):

When this goes low, the Program Counter is set to zero (0000H),the buses are tri-stated and reset the 8085.

RESET OUT (Output):

This indicates that 8085 is being reset. This can be used to reset other devices.

Power supply and Clock Frequency

X1, X2 (Input):

A Crystal or R-C or L-C network is connected to these two pins. The crystal frequency is internally divided by two to give the operating system frequency.So, to run the microprocessor at 3 MHz, a clock running at 6 MHz should be connected to the X1and X2 pins.

CLK (Output): Clock Output:

This output clock pin is used to provide the clock signal for other devices.

Power supplies:Vcc : +5 V supply; Vss : Ground Reference

Serial I/O Signals

SID and SOD implement serial data transmission.

SID (Input):

Serial input data line. The data on this line is loaded into accumulator bit 7 whenever a RIM instruction is executed.

SOD (output):

Serial output data line. The accumulator bit 7 output on SOD line by the SIM instruction.

ARCHITECTURE OF *8086* MICROPROCESSOR:

Features:

Intel released its first 16-bit microprocessor 8086 in 1978.The Intel 8086 is a 16-bit processor ,which is fabricated using HMOS technology and it has 40 pins ,packaged in DIP .

- The 8086 is a 16-bit microprocessor. The term 16-bit means that its arithmetic logic unit, internal registers and most of its instructions are designed to work with 16-bit binary words.
- The 8086 has a 16-bit data bus, so it can read data from or write data to memory and ports either 16 bits or 8 bits at a time. The 8088, however has an 8-bit data bus, so it can only read data from or write data to memory and ports 8 bits at a time.
- The 8086 has a 20-bit address bus, so it can directly access 2^{20} or 10, 48,576 (1MB) memory locations. Each of the 10, 48,576 (1MB) memory locations is byte wide. Therefore, a 16-bit word is stored in two consecutive memory locations. The 8088 also has a 20-bit address bus, so it can also address 2^{20} memory locations.
- The 8086 can generate 16-bit I/O address, hence it can access 2^{16}=65536 I/O ports.
- The 8086 provides fourteen 16-bit registers.
- The 8086 has multiplexed address and data bus which reduces the number of pins needed, but does slow down the transfer of data.
- The 8086 requires clock with a 33% duty cycle to provide optimized internal timing.
- The clock frequency of 8086 is 5 MHz.
- The Intel 8086 is designed to operate in two modes: minimum mode and maximum mode.
- *Minimum mode:* When only one 8086 CPU is to be used in a microcomputer system, the 8086 is used in the minimum mode of operation.
- *Maximum mode:* More than one processor (multiprocessor) is used in the system, the 8086 is used in the maximum mode of operation.
- An interesting feature of the 8086 is that it fetches up to 6 instruction bytes from memory and queue stores them in order to speed up instruction execution.
- It requires +5V single power supply.

Architecture:

Before we can talk about how to write programs for the 8086, we need to discuss its specific internal features, such as its ALU, Flags, Registers, instruction byte queue, and segment registers. The internal architecture of 8086 microprocessor is shown in figure below.

As shown by the block diagram in figure, the 8086 processor is divided into two independent functional parts, the *Bus Interfacing Unit (BIU),* and the *Execution Unit (EU).* Dividing the work between these two units speeds up processing. These two functional units work simultaneously to increase system speed and throughput. *Throughput* is a measure of number of instructions executed per unit time.

1. Bus Interfacing Unit: The bus interfacing unit in 8086 provides the interface to the outside world. This unit is responsible for performing all external bus operations like

a) It sends the address of the memory or I/O.
b) It fetches instructions from memory.
c) It reads data from port/memory.
d) It writes data into port/memory.
e) It supports instruction queuing.
f) It provides address relocation facility.

To implement these functions, Bus Interface Unit contains Bus Interface Logic, Segment registers, Memory addressing logic (Adder) and a Six byte instruction object code queue.

The queue: To speed up program execution, the BIU fetches *six instruction bytes* ahead of time from the memory. The BIU stores these perfected bytes in a group of registers called a **queue**. With the help of queue, it is possible to fetch next instruction when current instruction is in execution. The queue operates on the principle first-in first-out (FIFO). Feature of fetching the next instruction while the current instruction is in execution is called **pipelining.**

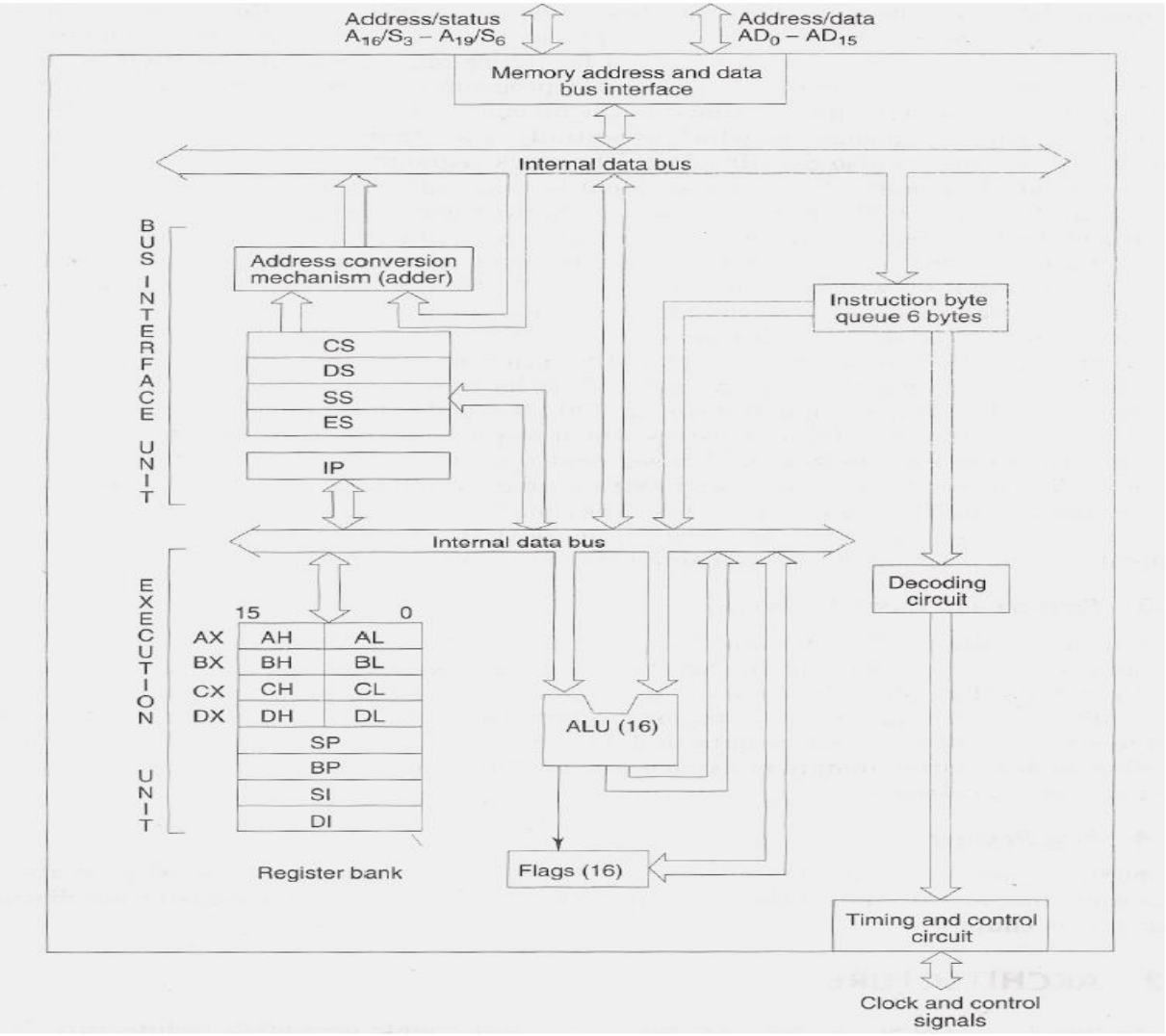


Fig.: 8086 internal block diagram

For example: The current instruction in execution is multiplication instruction. In 8086, the operands for multiplication operation are within registers. Still it requires 100 clock cycles to execute multiply instruction. Like multiplication there are number of other instructions in 8086 which need quite a large number of clock cycles for execution. During this execution time, the BIU fetches the next instruction or instructions from memory into the instruction queue instead of remaining idle. The BIU continues this process as long as the queue is not full. During this, execution unit gets the ready instruction bytes in the queue and instruction fetch time is eliminated.

2. Execution Unit: The EU of 8086 tells the BIU from where to fetch instructions or data, decodes instructions and executes instructions. As shown in figure, the EU contains: control unit, decoder, ALU and a registers.

Control unit: This directs internal operations.

Decoder: A decoder in the EU translates instructions fetched from memory into a series of actions which the EU carries out.

ALU: The EU has a 16-bit arithmetic logic unit which can add, subtract, AND, OR, XOR, increment, decrement, complement and shift binary numbers.

MEMORY SEGMENTATION:

Two types of memory organizations are commonly used. These are linear addressing and segmented addressing. In linear addressing the entire memory space is available to the processor in one linear array. Where as in the segmented addressing the available memory space is divided into "chunks" called segments. Such a memory is known as segmented memory. The memory in an 8086 based system is organized as segmented memory. In this scheme, the complete physically available 1MB memory may be divided into a number of logical segments. Each segment is 64KB in size and is addressed by one of the segment registers. However, at any given time the 8086 works with only four 64KB segments within this 1,048,576-byte (1MB) range. Four segment registers in the BIU are used to hold the upper 16-bits of the starting addresses of four memory segments that the 8086 is working with at a particular time.

Figure shows how these four segments might be positioned in memory at a given time. The four segments can be separated as shown, or for small programs which do not need all 64 K bytes ion each segment, they can overlap.

Rules for memory segmentation:

- The four segments can overlap for small programs. In a minimum system all the four segments can start at the address 00000H.
- The segment can start at any memory address which is divisible by 16.

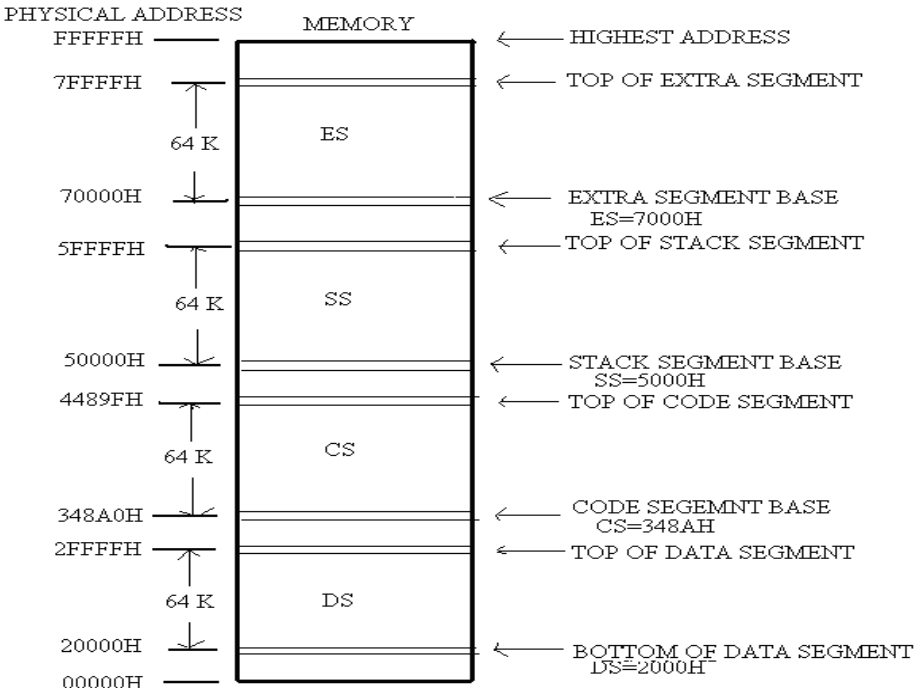


Fig.5. One way four 64-Kbyte segments might be positioned within the 1-Mbyte address Space of and 8086

Advantages:

- It allows the memory addressing capacity to be 1 MB even though the address associated with individual instruction is only 16-bit.
- It allows instruction code, data, stack, and portion of program to be more than 64 KB long by using more than one code, data, stack and extra segment.
- It provides use of separate memory areas for program, data and stack.

REGISTER ORGANIZATION OF *8086* MICROPROCESSOR: The 8086 processor has a powerful set of registers. It includes general purpose registers, segment registers, pointers and index registers, and flag register. The figure below shows the register organization of 8086 processor.

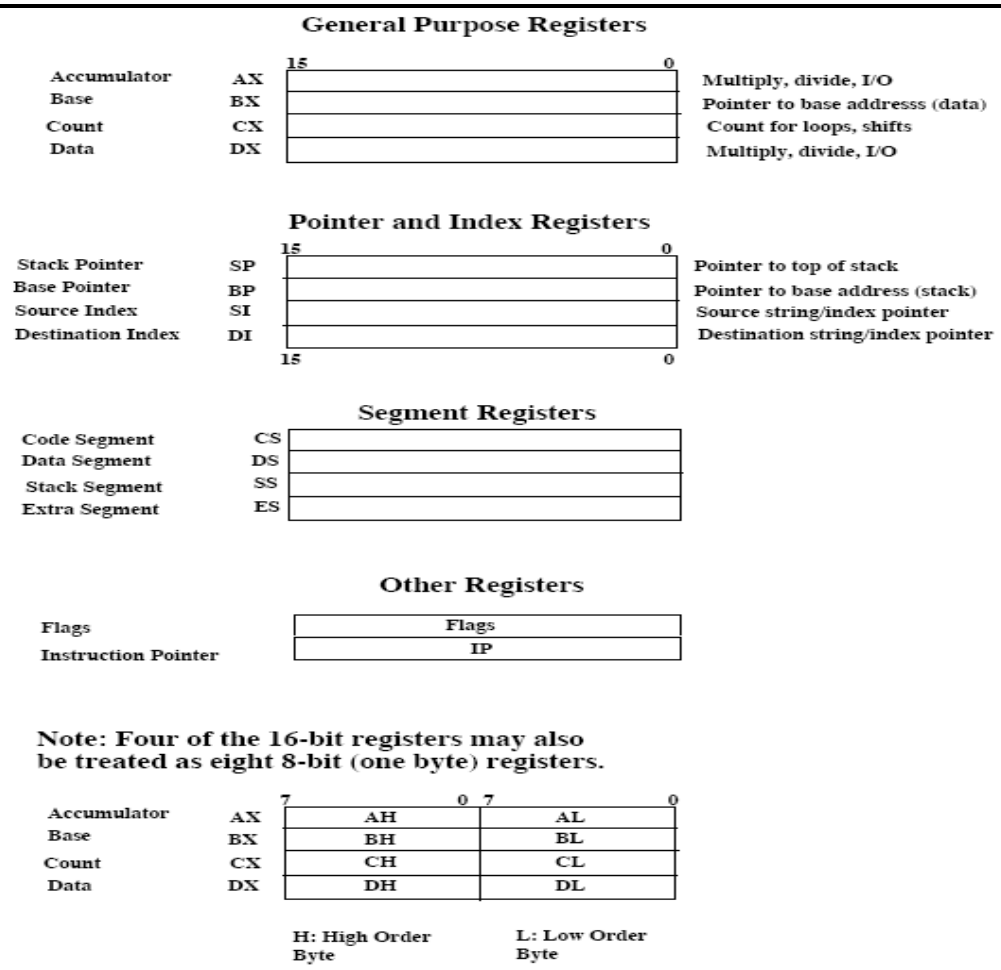
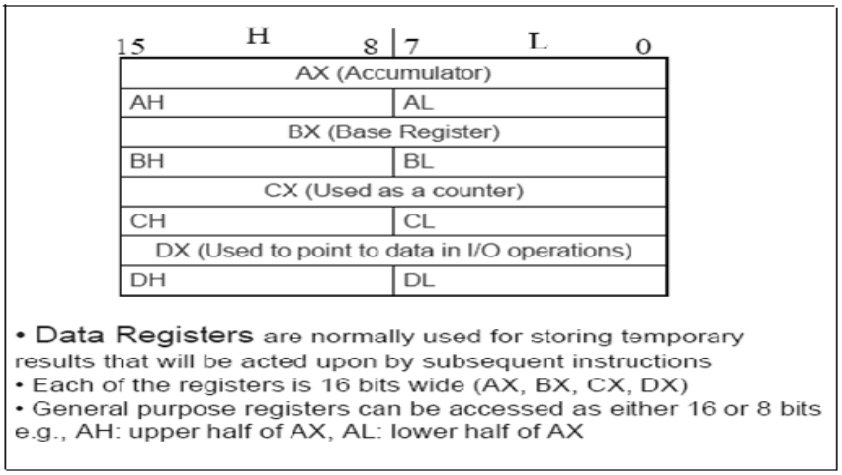


Fig.6. Register Organization of an 8086

General purpose registers:

8086 has a powerful set of registers containing general purpose and special purpose registers. All the registers of 8086 are 16-bit registers. The general purpose registers, can be used either 8-bit registers or 16-bit registers. The general purpose registers are either used for holding the data, variables and intermediate results temporarily or for other purpose like counter or for storing offset address for some particular addressing modes etc.



Segment Registers:

The 8086 BIU sends out 20-bit physical addresses, so it can address any of 2^{20} or 1,048,576 bytes in memory. However at any given time the 8086 works with only four 64Kbytes (65,536) segments within this 1Mbytes (1,048,576 byte) range. Four segment registers in the BIU are used to hold the upper 16 bits of the starting addresses of four memory segments that the 8086 is working with at a particular time. The

four segment registers are the ***code segment (CS) register***, the ***stack segment (SS) register***, the ***extra segment (ES) register***, and the ***data segment (DS) register***. So, the segment register is used to hold the upper 16 bits of the starting address for each of the segments.

For example, the *code segment register* holds the upper 16 bits of the starting address for the segment from which the BIU is currently fetching instruction code bytes. The BIU always inserts zeros for the lowest 4 bits (nibble) of the 20-bit starting address for a segment. For example, if the code segment register contains 348AH, then the code segment will start at address 348A0H. In other words, a 64Kbyte segment can be located anywhere within the 1-Mbyte address space, but the segment will always start at an address with zeros in the lowest 4 bits. The part of a segment starting address stored in a segment register is often called the ***segment address (or) Base address.***

A ***stack*** is a section of memory set aside to store addresses and data while a *subprogram executes*. The *stack segment register* is used to hold the upper 16 bits of the starting address for the program stack.

The *extra segment register* and *data segment register* are used to hold the upper 16 bits of the starting addresses of two memory segments that are used for data.

Code segment (CS) is a 16-bit register containing address of 64 KB segment with processor instructions. The processor uses CS segment for all accesses to instructions referenced by instruction pointer (IP) register. CS register cannot be changed directly. The CS register is automatically updated during far jump, far call and far return instructions. It is used for addressing a memory location in the code segment of the memory, where the executable program is stored.

Stack segment (SS) is a 16-bit register containing address of 64KB segment with program stack. By default, the processor assumes that all data referenced by the stack pointer (SP) and base pointer (BP) registers is located in the stack segment. SS register can be changed directly using POP instruction. It is used for addressing stack segment of memory. The stack segment is that segment of memory, which is used to store stack data.

Data segment (DS) is a 16-bit register containing address of 64KB segment with program data. By default, the processor assumes that all data referenced by general registers (AX, BX, CX, DX) and index register (SI, DI) is located in the data segment. DS register can be changed directly using POP and LDS instructions. It points to the data segment memory where the data is resided.

Extra segment (ES) is a 16-bit register containing address of 64KB segment, usually with program data. By default, the processor assumes that the DI register references the ES segment in string manipulation instructions. ES register can be changed directly using POP and LES instructions. It also refers to segment which essentially is another data segment of the memory. It also contains data

Pointer and Index registers:

SP	Stack Pointer
BP	Base Pointer
SI	Source Index
DI	Destination Index
IP	Instruction Pointer

The registers in this group are all 16 bits wide
Low and high bytes are not accessible
These registers are used as memory pointers
- Example: MOV AH, [SI]
  Move the byte stored in memory location whose address is contained in register SI to register AH

IP is not under direct control of the programmer

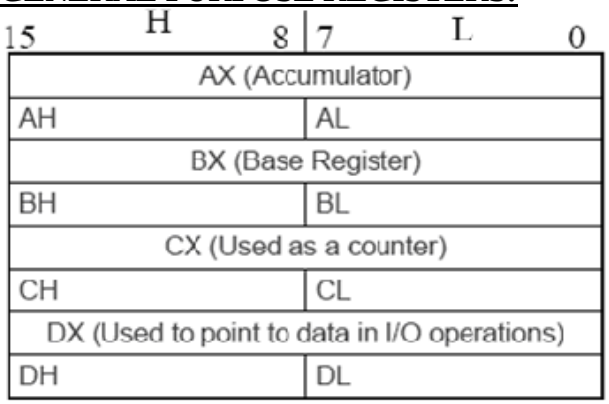
The EU contains a 16-bit *instruction pointer (IP) register, stack pointer register (SP)* and *base pointer (BP) register*. It also contains a 16-bit *source index (SI) register* and a 16-bit *destination index (DI) register*. The main use of these pointers and index registers is to hold the 16-bit offset address of a data word in one of the segments. For example SI can be used to hold the offset address of a data word in the data segment. The physical address of the data in memory will be generated in this case by adding the contents of SI to the segment base address represented by the 16-bit number in the DS register.

The three registers BP, SI, and DI can also be used for temporary storage of data just as the general purpose registers described above.

Flag registers: A flag is a flip-flop which indicates some condition produced by the execution of an instruction or controls certain operations of EU. The flag register contains nine active flags.

Offset registers: The registers which are used to hold 16-bit offset address of a particular segment are called *offset registers* and such registers are **BP, SP, BX, IP, SI, DI.**

SPECIAL FUNCTIONS OF GENERAL PURPOSE REGISTERS:



AX (Accumulator): The register AX is used to store the result produced by the ALU. So, it is called as Accumulator.

BX (Base Register): The register BX is used as offset storage for generating physical addresses in case of certain addressing modes.

CX (Counter Register): The register CX is used as default counter in case of string and loop instructions.

DX (Data Register): The register DX is used as destination (store the result data) in case of multiplication and division instructions. DX register is also used as I/O address pointer in case of I/O instructions.

***8086* FLAG REGISTER AND FUNCTION OF *8086* FLAGS:**

A **flag** is flip-flop that indicates some condition produced by the execution of an instruction or controls certain operations of the EU. A 16-bit flag register in the EU contains *nine* active flags. The figure shows the location of the nine flags in the flag register. Six of the nine flags are used to indicate some condition produced by an instruction. The six conditional flags in this group are the *carry flag (CF)*, the *parity flag (PF)*, the *auxiliary carry flag (AF)*, the *zero flag (ZF)*, the *sign flag (SF)*, and the *overflow flag (OF)*. The names of these flags should give you hints as to what conditions affect them.Cetain 8086 instructions check these flags to determine which of two alternative actions should be done in executing the instruction. The six conditional flags are set or reset by the EU on the basis of the results of some arithmetic or logic operation.

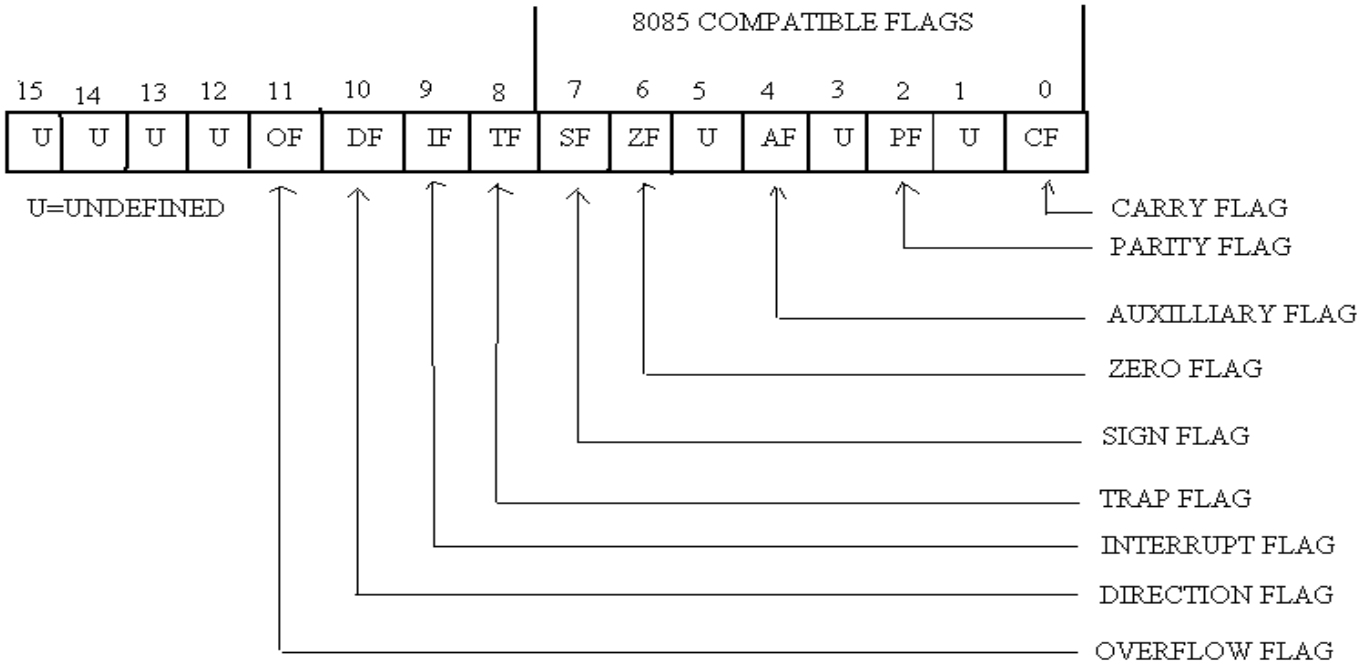


Fig.7. 8086 flag register format

The description of each flag bit is as follows:

Carry flag: This flag is set when there is a carry out of MSB in case of addition or a borrow in case of subtraction. For example, when two numbers are added, a carry may be generated out of the MSB position. The carry flag, in this case will be set to '1 '. In case no carry is generated it will be '0 '.

Parity flag: This flag is set to 1 if the lower byte of the result contains even number of 1s, other wise it is zero.

Auxiliary flag: This is set if there is a carry from the lowest nibble during addition or borrow for the lowest nibble during subtraction.

Zero flag: This flag is set if the result of operation in ALU is zero and the flag resets if the result is nonzero.

Sign flag: After execution of arithmetic or logical operations, if the MSB of the result is 1, the sign flag is set. Sign flag equals the MSB of the result. Sign bit 1 indicates the result is negative.

Overflow flag: This flag is set if result is out of range. For addition this flag is set when there is a carry into the MSB and no carry out of the MSB or vice-versa. For subtraction, it is set when the MSB needs a borrow and there is no borrow from the MSB, or vice-versa.

The three remaining flags in the flag register are used to control certain operations of the processor. These flags are different from the six conditional flags described above. The control flags are the *trap flag (TF*), the *interrupt flag (IF),* the *direction flag (DF).* The descriptions of these flags are explained as follows:

Trap flag: If this flag is set, the processor enters the single step execution mode. In other words, a trap interrupt is generated after execution of each instruction. The processor executes the current instruction and the control is transferred to the trap interrupt service routine.

Single stepping: One way to debug a program is to run the program one instruction at a time and see the contents of used registers and memory variables after execution of every instruction. This process is called *single stepping* through a program.

Interrupt flag: If this flag is set, the maskable interrupts are recognized by the processor, otherwise they are ignored.

Direction flag: This is used by string manipulation instructions. If this flag bit is '0', the string is processed beginning from the lowest address to the highest address, i.e. auto incrementing mode. Otherwise, the string is processed from the highest address towards the lowest address, i.e. auto decrementing mode.

Example:

1. Give the contents of flag register after execution of following addition.

```
            0110 0101 1101 0001
            0010 0011 0101 1001
      +     -------------------------------
            1000 1001 0010 1010
```

Solution: SF=1, ZF=0, PF=1, CF=0, AF=0, OF=1

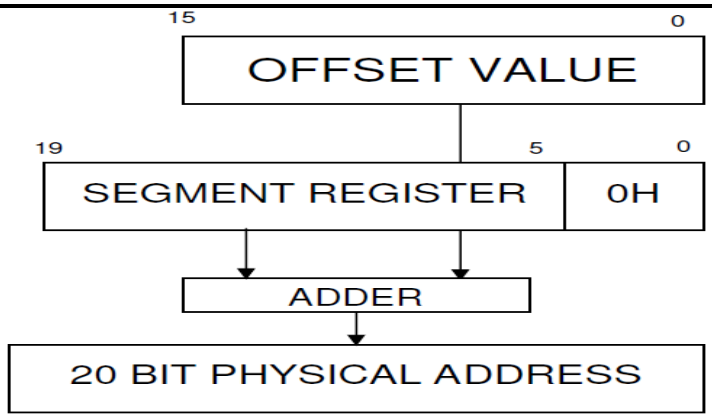
2. Give the contents of flag register after execution of following subtraction.

```
            0110 0111 0010 1001
      -     0011 0101 0100 1010
            -------------------------------
            0011 0001 1101 1111
```

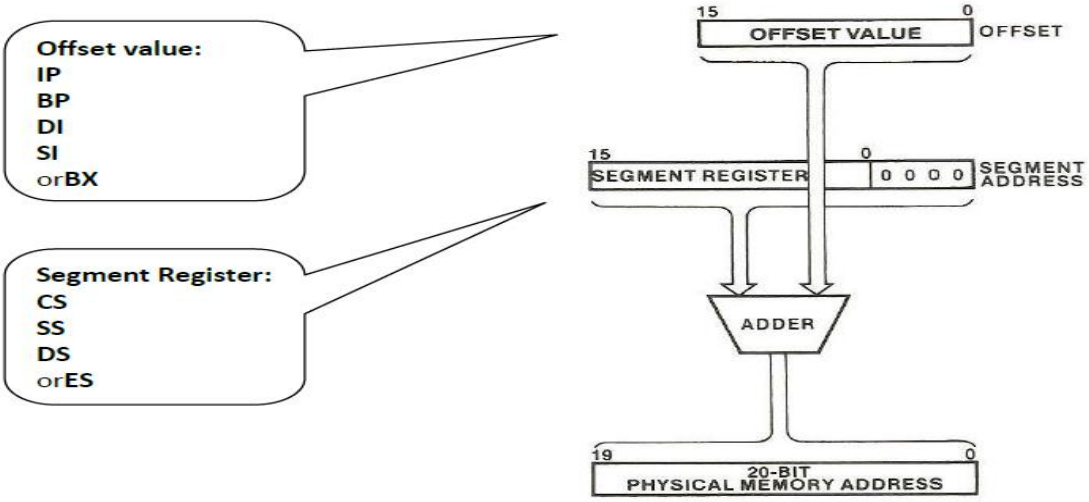
Solution: SF=0, ZF=0, PF=1, CF=0, AF=1, OF=0

Accessing Memory locations or GENERATION OF 20-BIT PHYSICAL ADDRESS:

To access a specific memory location from any segment we need 20-bit physical address. The 8086 generates this address using the contents of segment register and the offset register associated with it. The figure below shows the way of calculating the physical address.



(Or)



Let us see the following examples:

1. Code Segment and Instruction Pointer:

The Code segment register holds the upper 16-bits of the starting address of the segment from which the BIU is currently fetching instruction code bytes. The instruction pointer register holds the 16-bit address or offset of the next code byte within this code segment. The value contained in the IP is referred to as an offset, because this value must be offset from (added to) the segment base address in CS to produce the required 20-bit physical address sent out by BIU. The figure below shows the way of calculating physical address by adding CS and IP addresses.

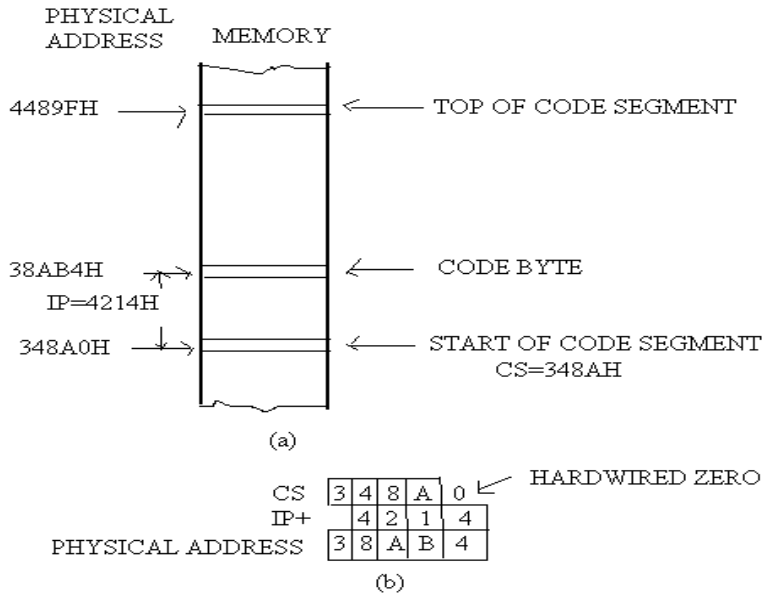


Fig.8. Addition of IP to CS to produce the physical address of the code byte,
(a).Diagram (b). Computation

The CS registers points to the base or start of the current code segment. The IP contains the **distance or offset** from this base address to the next instruction byte to be fetched. The fig.(b) shows how the 16-bit in

IP is added to the 16-bit segment base address in CS to produce the **20-bit physical** address. Note that the two 16-bit numbers are not added directly in line, because the CS register contains only the upper 16-bits of the base address for the code segment. The BIU automatically inserts zeros for the lowest 4 bits of the segment base address.

For example, if the CS register contains 348AH and IP contains offset of 4214H, then the result of 20-bit physical address is 38AB4H. The alternative way of representing a 20-bit physical address is the ***segment base: offset form.*** For the address of a code byte, the format for this alternative form will be **CS: IP.** So, the above physical address can also be represented as 348A:4214.

In brief, the CS register contains the upper 16 bits of the starting address of the code segment in the 1 MB address range of the 8086. The instruction pointer register contains a 16-bit offset, which tells where the next instruction byte is to be fetched in that 64KB code segment. The actual physical address sent to memory is produced by adding the offset contained in the IP register to the segment base represented by the upper 16 bits in the CS register.

So, at any time to access a memory, the BIU produces the required 20-bit physical address by adding an offset to a segment base value represented by the contents of one of the segment registers

2. Stack Segment register and Stack Pointer register:

A stack is a section of memory set aside to store addresses and data while a subprogram is executing. The 8086 allows you to set aside an entire 64-Kbyte segment as a stack. The upper 16-bits of the starting address for this segment are kept in the stack segment register. So, the *stack segment register* holds the upper 16 bits of starting address of stack segment. The *stack pointer (SP)* register in the execution unit holds the 16-bit offset from the start of the segment to the memory location where a word was most recently stored on the stack. The memory location where a word was most recently stored is called the ***top of stack.*** The figure shows calculation of physical address using SS and SP.

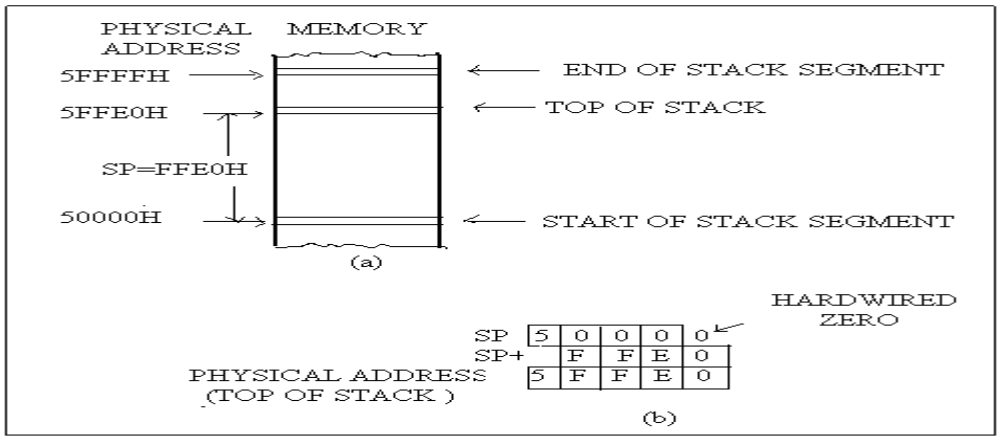


Fig.9. Addition of SS and SP to produce the physical address of the top of the stack, (a). Diagram (b). Computation.

The physical address for a stack read or stack write is produced by adding the contents of the stack pointer register to the segment base address represented by the upper 16 bits of the base address in SS.In the above example the 5000H in SS represents a segment base address of 50000H.When the FFE0H in the SP is added to this the resultant physical address for the top of the stack will be 5FFE0H. The physical address can be represented either as a single number, 5FFE0H or in SS: SP form as 5000:FFE0H.

DEFAULT AND ALTERNATE REGISTER ASSIGNMENTS:

Below table shows that some memory references and their default and alternate segment definitions. For example, instruction codes can only be stored in the code segment with IP used as an offset. Similarly, for stack operations only SS and SP or BP registers can be used to give segment and offset addresses respectively. On the other hand, for accessing general data, string source, data pointed by BX and BP registers it is possible to use alternate segments by using segment override prefix.

Type of memory reference	***Default Segment***	***Alternate Segment***	***Offset (logical address)***
Instruction fetch	CS	None	IP
Stack operation	SS	None	SP, BP

General data	DS	CS,ES,SS	Effective address
String source	DS	CS,ES,SS	SI
String destination	ES	None	DI
BX used as pointer	DS	CS,ES,SS	Effective address
BP used as pointer	SS	CS,ES,DS	Effective address

Example: Calculate the physical address for the following instructions.

1. MOV AL,[BP]
2. MOV CX,[BX]
3. MOV AL, [BP+SI]
4. MOV CS:[BX],AL

Assume: CS=1000H, DS=2000H, SS=3000H, ES=4000H, BP=0010H,
BX=0020H, SP=0030H, SI=0040H, DI=0050H

Solution:

1.
3000 0H SS
+0010H BP

30010H---Physical address

2.
2000 0H DS
+0020H BX

20020H---Physical address
3.
0010H BP
+0040H SI

0050H---Effective address
3000 0H SS
+0050H EA

30050H----Physical address
4.
1000 0H CS
+0020H BX

10020H----Physical address

Segment Override Prefix: The segment override prefix allows the programmer to deviate from the default segment

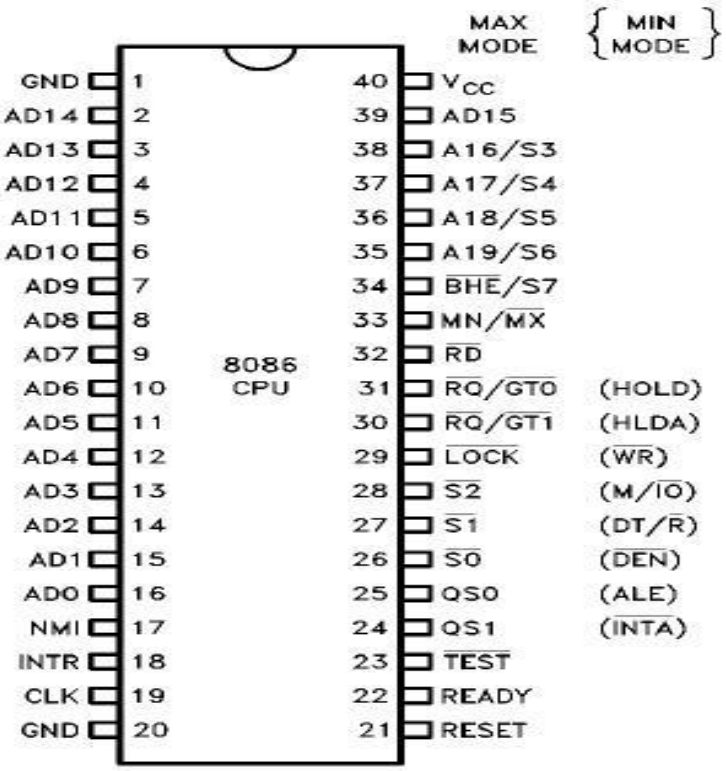
PIN DIAGRAM OF 8086 MICROPROCESSOR

The Microprocessor 8086 is a 16-bit CPU available in different clock rates and packaged in a 40 pin CERDIP or plastic package. The 8086 operates in single processor or multiprocessor configuration to achieve high performance. The pins serve a particular function in minimum mode (single processor mode) and other function in maximum mode configuration (multiprocessor mode).

The minimum mode is selected by applying logic 1 to the MN / $\overline{MX}$ input pin. This is a single microprocessor configuration.

The maximum mode is selected by applying logic 0 to the MN / $\overline{MX}$ input pin. This is a multi micro processors configuration.

The figure below shows the pins/signals of 8086 processor. Here the pins within the brackets (minimum mode pins) are minimum mode pins.



Signal description:

The 8086 signals can be categorized in three groups.

- The first are the signal having common functions in minimum as well as maximum mode.
- The second are the signals which have special functions for minimum mode.
- Third are the signals which have special functions for maximum mode.

✓ **The following signal descriptions are common for both modes:**

Vcc: It requires +5V single power supply for the operation of the internal circuit.

GND: ground for the internal circuit.

AD15-AD0: These are the time multiplexed memory I/O address and data lines. These lines serve two functions. The 16 data bus lines D0 through D15 are actually multiplexed with address lines A0 through A15 respectively. By multiplexed we mean that the bus work as an address bus during first machine cycle and as a data bus during next machine cycles. D15 is the MSB and D0 LSB. When acting as a data bus, they carry read/write data for memory, input/output data for I/O devices, and interrupt type codes from an interrupt controller.

A19/S6, A18/S5, A17/S4, and A16/S3: These are the time multiplexed address and status lines. During T1 these are the most significant address lines for memory operations. During I/O operations, these lines are low. During memory or I/O operations, status information is available on those lines for T2, T3, Tw and T4.

The status of the interrupt enable flag bit is updated at the beginning of each clock cycle. The status is displayed on $\mathbf{S_5}$ pin.

The **S4** and **S3** combinedly indicate which segment register is presently being used for memory accesses as in below fig.

The last status bit **S6** is always at the logic 0 level.

The address bits are separated from the status bit using latches controlled by the ALE signal.

S_4	S_3	Segment Register
0	0	Extra
0	1	Stack
1	0	Code / none
1	1	Data

$\overline{BHE}$/S7: The bus high enable is used to indicate the transfer of data over the higher order (D15-D8) data bus as shown in table. It goes low for the data transfer over D15-D8 and is used to derive chip selects of odd address memory bank or peripherals. $\overline{BHE}$ is low during T1 for read, write and interrupt acknowledge cycles, whenever a byte is to be transferred on higher byte of data bus.

BHE	A_0	Indication
0	0	Whole word
0	1	Upper byte from or to even address
1	0	Lower byte from or to even address
1	1	None

$\overline{RD}$ (Read): This signal on low indicates the peripheral that the processor is performing s memory or I/O read operation. $\overline{RD}$ is active low and shows the state for T2, T3, and Tw of any read cycle. The signal remains tristated during the hold acknowledge.

READY: This is the acknowledgement from the slow device or memory that they have completed the data transfer. This signal is provided by an external clock generator device and can be supplied by the memory or I/O subsystem to signal the 8086 when they are ready to permit the data transfer to be completed.

NMI-Non maskable interrupt: This is an edge triggered input which causes a type2 interrupt. The NMI is not maskable internally by software. A transition from low to high initiates the interrupt response at the end of the current instruction.

INTR: INTR is an input to the 8086 that can be used by an external device to signal that it needs to be serviced. Logic 1 at INTR represents an active interrupt request. When an interrupt request has been recognized by the 8086, it indicates this fact to external circuit with pulse to logic 0 at the $\overline{INTA}$ output.

TEST: This input is examined by a 'WAIT' instruction. If the TEST pin goes low, execution will continue, else the processor remains in an idle state.

CLK: Clock Input: The clock input provides the basic timing for processor operation and bus control activity. Its an asymmetric square wave with 33% duty cycle.

RESET: This input causes the processor to terminate the current activity and start execution from FFFF0H.

MN/$\overline{MX}$: The logic level at this pin decides whether the processor is to operate in either minimum or maximum mode.

✓ **The following pin functions are for the minimum mode operation of 8086:**

M/$\overline{IO}$: This is a status line logically equivalent to S2 in maximum mode. The logic level of M/$\overline{IO}$ tells external circuitry whether a memory or I/O transfer is taking place over the bus. When it is low, it indicates the processor is having an I/O operation, and when it is high, it indicates that the processor is having a memory operation.

$\overline{WR}$: The signal write $\overline{WR}$ indicates that a write bus cycle is in progress. The 8086 switches $\overline{WR}$ to logic 0 to signal external device that valid write or output data are on the bus.

$\overline{INTA}$: (Interrupt Acknowledge)-This signal is used as a read strobe for interrupt acknowledge cycles. i.e. when it goes low, the processor has accepted the interrupt.

ALE – Address Latch Enable: This output signal indicates the availability of the valid address on the address/data lines, and is connected to latch enable input of latches. This signal is active high and is never tristated.

DT/$\overline{R}$ – Data Transmit/Receive: This output is used to decide the direction of data flow through the transreceivers (bidirectional buffers). When the processor sends out data, this signal is high and when the processor is receiving data, this signal is low.

***DEN* – Data Enable**: This signal indicates the availability of valid data over the address/data lines. It is used to enable the transreceivers (bidirectional buffers) to separate the data from the multiplexed address/data signal.

HOLD and HLDA: The direct memory access DMA interface of the 8086 minimum mode consist of the HOLD and HLDA signals. When the HOLD line goes high, it indicates to the processor that another master is requesting the bus access. The processor, after receiving the HOLD request, issues the hold acknowledge signal on HLDA pin, in the middle of the next clock cycle after completing the current bus cycle.

The following pin functions are applicable for maximum mode operation of 8086:

$\overline{S2}$,$\overline{S1}$, **and** $\overline{S0}$ **– Status Lines**: These are the status lines which reflect the type of operation, being carried out by the processor.

Status Inputs			CPU Cycles
$\overline{S}_2$	$\overline{S}_1$	$\overline{S}_0$	
0	0	0	Interrupt Acknowledge
0	0	1	Read I/O Port
0	1	0	Write I/O Port
0	1	1	Halt
1	0	0	Instruction Fetch
1	0	1	Read Memory
1	1	0	Write Memory
1	1	1	Passive

LOCK: This output pin indicates that other system bus master will be prevented from gaining the system bus, while the LOCK signal is low.

QS1, QS0 – Queue Status: These lines give information about the status of the code-prefetch queue. Two new signals that are produced by the 8086 in the maximum-mode system are queue status outputs QS0 and QS1. Together they form a 2-bit queue status code, QS1QS0. Following table shows the four different queue status.

QS_1	QS_0	Queue Status
0 (low)	0	No Operation. During the last clock cycle, nothing was taken from the queue.
0	1	First Byte. The byte taken from the queue was the first byte of the instruction.
1 (high)	0	Queue Empty. The queue has been reinitialized as a result of the execution of a transfer instruction.
1	1	Subsequent Byte. The byte taken from the queue was a subsequent byte of the instruction.

RQ0/0GT, RQ1/GT1 – Request/Grant: These pins are used by the other local bus master in maximum mode, to force the processor to release the local bus at the end of the processor current bus cycle. Each of the pin is bidirectional with RQ/GT0 having higher priority than RQ/GT1.

ADDRESSING MODES

Before we can teach you ALP techniques, we need to discuss some of the different ways in which an 8086 can access the data that it operates on. *The different ways in which a processor can access data are referred to as its* ***addressing modes.(or)*** *Different ways of specifying the operands in an instruction is known as* ***Addressingmodes****.*

In assembly language statements, the addressing mode is indicated in the instruction. We will use the 8086 MOV instruction to illustrate some of the 8086 addressing modes. The MOV instruction has the format

MOV Destination, Source

When executed, this instruction copies a word or a byte from the specified source location to the specified destination location. The *source* can be a number written directly in the instruction, a specified register, or a memory location specified in 1 of 24 different ways. The *destination* can be a specified register or a memory location specified in any 1 of 24 different ways. The source and the destination cannot both be memory locations in an instruction. Remember that the destination location is specified in the instruction before the comma, and the source is specified in the instruction after the comma.

The addressing modes of 8086 processor are:

The 8086 provides total of eight addressing modes for instructions to specify the operands.

Two addressing modes are provided for instructions that operate on immediate and register operands.

1. *Immediate operandmode*
2. *Register operandmode*

Immediate Operand Mode: In this mode the operand is included in the instruction. Suppose that in a program you need to put the number 4203H in the CX register. The MOV CX, 4203H instruction can be used to do this. When it executes, this instruction will put the immediate hexadecimal number 4203H in the 16-bit CX register. This is referred to as immediate addressing mode. So, here the immediate data is a part ofinstruction.

Example: MOV AX,0004H

MOV AL, 04H
MOV CX,
437BH

Register Operand Mode: In this mode the operand is located in one of the 8 or 16-bit general purpose register. Register addressing mode means that a register is the source of an operand for an instruction. All the registers, except IP, may be used in this mode.

Example: MOV AX,
BX MOV
CX,AX

The instruction MOV CX, AX, copies the contents of the 16-bit AX register into the 16-bit CX register. Remember that the destination location is specified in the instruction before the comma, and the source is specified in the instruction after the comma. Also note that the contents of AX are just copied to CX, not actually moved. In other words, the previous contents of CX are written over, but the contents of AX are not changed. For example, if CX contains 4301H and AX contains 8470H before the MOV CX, AX instruction executes, then after the

instruction executes CX will contain 8470H and AX will still contain 8470H.

Six modes are provided to specify the location of an operand in memory segment. A memory operand address consists of two 16-bit components: segment selector (segment base) and offset. The offset is calculated by summing any combination of the following three address elements.

the **displacement** (an 8 or 16-bit immediate value contained in the instruction) the
base (contents of either BX or BP baseregisters)

the **index** (contents of either SI or DI indexregisters)

Combination of these three address elements defines the following **six** addressing modes, described below.

1. *DirectMode*
2. *Register IndirectMode*
3. *Register RelativeMode*
4. *IndexedMode*
5. *Based IndexedMode*
6. *Based Indexed Mode withdisplacement*

Direct Mode: The operand's offset is contained in the instruction as 8 or 16-bit displacement element.

Example: MOV BL,[437AH]

The square brackets around the 437AH are shorthand for "the contents of the memory location(s) at a displacement from the segment base of". When executed, this instruction will copy 'the contents of the memory location at a displacement from the data segment base of '437AH into the BLregister.

Register Indirect Mode: The operand's offset is in one of the registers SI, DI, BX or BP.

Example: MOV AX,[BX]

MOV CX,[BP]

Register Relative Mode: The operand's offset is sum of 8 or 16-bit displacement and the contents of registers SI, DI, BX or BP.

Example: MOV AX, 50H[BX]

Indexed Mode: The operand's offset is contents of index register SI or DI.

Example: MOV AX,[SI]

MOV AX,[DI]

Based Indexed Mode: The operand's offset is sum of the contents of base register and index register.

Example: MOV AX, [BX] [SI] or MOV AX,[BX+SI]

Based Indexed Mode with displacement: The operand's offset in sum of base register contents, an index register contents, and an 8 or 16-bit displacement.

Example: MOV AX, 60D [BX] [SI] or MOV AX, 60D[BX+SI]

Example: Calculate the physical address for the following instructions.

1. MOVAL,[BP]
2. MOVCX,[BX]
3. MOV AL,[BP+SI]
4. MOVCS:[BX],AL

Assume: CS=1000H, DS=2000H, SS=3000H, ES=4000H, BP=0010H,
BX=0020H, SP=0030H, SI=0040H, DI=0050H.

Solution:

1. 3000 0HSS

   +0010HBP

   30010H---Physical address

2. 2000 0HDS

   +0020HBX

   20020H---Physical address

3. 0010H BP

   +0040HSI

   0050H---Effective address

   3000 0HSS

   +0050HEA

   30050H-----Physicaladdress

4. 1000 0HCS

   +0020H BX

   10020H-----Physicaladdress

Segment Override Prefix:The segment override prefix allows the programmer to deviate from the default segment.

Interrupts

Definition: The meaning of 'interrupts' is to break the sequence of operation. While the CPU is executing a program, on 'interrupt' breaks the normal sequence of execution of instructions, diverts its execution to some other program called Interrupt Service Routine (ISR).After executing ISR , the control is transferred back again to the main program. Interrupt processing is an alternative to polling.

Need for Interrupt: Interrupts are particularly useful when interfacing I/O devices that provide or require data at relatively low data transfer rate.

Types of Interrupts: There are two types of Interrupts in 8086. They are: (i) Hardware Interrupts and

(ii)Software Interrupts

(i) **Hardware Interrupts** (External Interrupts). The Intel microprocessors support hardware interrupts through:

- Two pins that allow interrupt requests, INTR and NMI
- One pin that acknowledges, INTA, the interrupt requested on INTR.
- INTR and NMI
- INTR is a maskable hardware interrupt. The interrupt can be enabled/disabled using STI/CLI instructions or using more complicated method of updating the FLAGS register with the help of the POPF instruction.

- When an interrupt occurs, the processor stores FLAGS register into stack, disables further interrupts, fetches from the bus one byte representing interrupt type, and jumps to interrupt processing routine address of which is stored in location 4 * <interrupt type>. Interrupt processing routine should return with the IRET instruction.

- NMI is a non-maskable interrupt. Interrupt is processed in the same way as the INTR interrupt. Interrupt type of the NMI is 2, i.e. the address of the NMI processing routine is stored in location 0008h. This interrupt has higher priority than the maskable interrupt.

- – Ex: NMI, INTR.

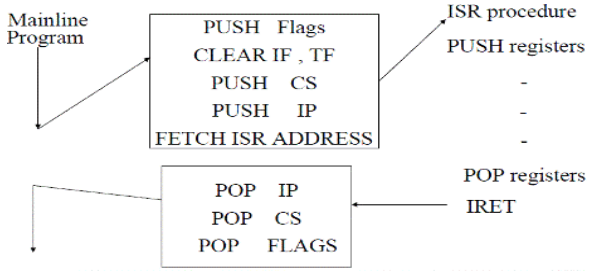
(ii) **Software Interrupts** (Internal Interrupts and Instructions) .Software interrupts can be caused by:

- INT instruction - breakpoint interrupt. This is a type 3interrupt.
- INT <interrupt number> instruction - any one interrupt from available 256interrupts.
- INTO instruction - interrupt on overflow
- Single-step interrupt - generated if the TF flag is set. This is a type 1 interrupt. When the CPU processes this interrupt it clears TF flag before calling the interrupt processing routine.

- Processor exceptions: Divide Error (Type 0), Unused Opcode (type 6) and Escape opcode (type7).

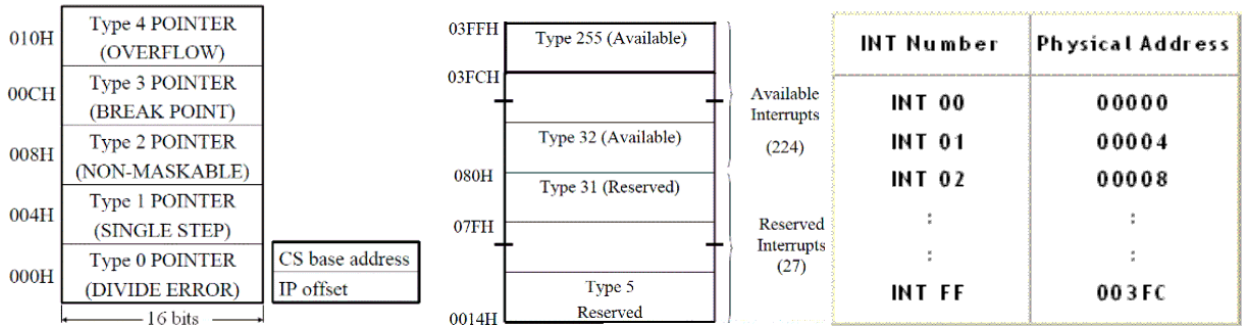
- Software interrupt processing is the same as for the hardware interrupts.
- - Ex: INT n (Software Instructions)
- Control is provided through:
    - IF and TF flag bits
    - IRET and IRETD

Processing of Interrupts



1. It decrements SP by 2 and pushes the flag register on the stack.
2. Disables INTR by clearing the IF.
3. It resets the TF in the flag Register.
5. It decrements SP by 2 and pushes CS on the stack.
6. It decrements SP by 2 and pushes IP on the stack.
6. Fetch the ISR address from the interrupt vector table.

Interrupt Vector Table



Functions associated with INT00 to INT04 INT 00 (divide error)

- INT00 is invoked by the microprocessor whenever there is an attempt to divide a number by zero.
- ISR is responsible for displaying the message “Divide Error” on the screen

INT 01

- For single stepping the trap flag must be1
- After execution of each instruction, 8086 automatically jumps to 00004H to fetch 4 bytes for CS: IP of the ISR.
- The job of ISR is to dump the registers on to the screen

INT 02 (Non maskable Interrupt)

- Whenever NMI pin of the 8086 is activated by a high signal (5v), the CPU Jumps to physical memory location 00008 to fetch CS:IP of the ISR associated with NMI.

INT 03 (break point)

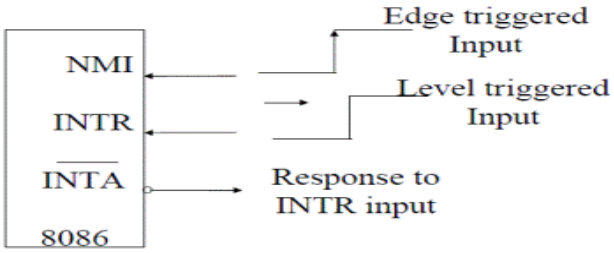
- A break point is used to examine the CPU and memory after the execution of a group of Instructions.
- It is one byte instruction whereas other instructions of the form “INT n” are 2 byte instructions.

INT 04 (Signed number overflow)

- There is an instruction associated with this INT 0 (interrupt on overflow).
- If INT 0 is placed after a signed number arithmetic as IMUL or ADD the CPU will activate INT 04 if 0F =1.
- In case where 0F = 0, the INT 0 is not executed but is bypassed and acts as a NOP.
-

Performance of Hardware Interrupts

- NMI : Non maskable interrupts - TYPE 2Interrupt
- INTR : Interrupt request - Between 20H and FFH



Interrupt Priority Structure

Interrupt	Priority
Divide Error, INT(n),INTO	Highest
NMI	↓
INTR	
Single Step	Lowest

TIMINGS:

Timing plays a crucial role, not only in sports like cricket_but also in digital electronic equipments like microprocessors. Timing and Timing diagram plays a vital role in microprocessors. The timing diagram is the diagram which provides information about the various conditions of signals such as high/low, when a machine cycle is being executed. Without the knowledge of timing diagram it is not possible to match the peripheral devices to the microprocessors. These peripheral devices includes memories, ports etc. Such devices can only be matched with microprocessors with the help of timing diagram.

Before dealing with timing diagram, we have to make ourselves familiar with certain terms.

Machine cycle: A basic microprocessor operation such as reading a byte from memory or writing a byte to a port is called a machine cycle (Bus cycle). A machine cycle consists of at least 4 clock cycles/ clock states (T-states) for accessing the data.

Instruction cycle: The time a microprocessor requires to fetch and execute an entire instruction is referred to as an instruction cycle. An instruction cycle consists of one or more machine cycles.

T-state: T-state is nothing but one subdivision of the operation performed in one clock period. These subdivisions are internal state of the microprocessor synchronized with system clock.

So, an instruction cycle is made up of machine cycles, and a machine cycle is made up of states. The time for the state is determined by the frequency of the clock signal.

GENERAL BUS OPERATION CYCLES:

The 8086 has a combined address and data bus commonly referred as a time multiplexed address and data bus. The main reason behind multiplexing address and data over the same pins is the maximum utilization of processor pins and it facilitates the use of 40 pin standard DIP package. The bus can be demultiplexed using a few latches and transreceivers, when ever required. Basically, all the processor bus cycles consist of at least four clock cycles. These are referred to as T1, T2, T3, and T4. The address is transmitted by the processor during T1. It is present on the bus only for one cycle.

The above figs shows the signal activities on the 8086 microcomputer buses during simple read and write operations. The first line look at is the clock waveform, CLK, at the top. This represent s the crystal controlled clock signal sent to the 8086 from an external clock generator device such as the 8284. One clock cycle of this clock is called a state. The time interval labeled T1 in the figure is an example of a state. Different versions of the 8086 have maximum clock frequencies of between 5 MHz and 10 MHz, so the minimum time for one sate will be between 100 and 200ns, depending on the part use and the crystal used.

The negative edge of this ALE pulse is used to separate the address and the data or status information. In maximum mode, the status lines $\overline{S0}$, $\overline{S1}$ and $\overline{S2}$ are used to indicate the type of operation. Status bits S3 to S7 are multiplexed with higher order address bits and the BHE signal. Address is valid during T1 while status bits S3 to S7 are valid during T2 through T4.

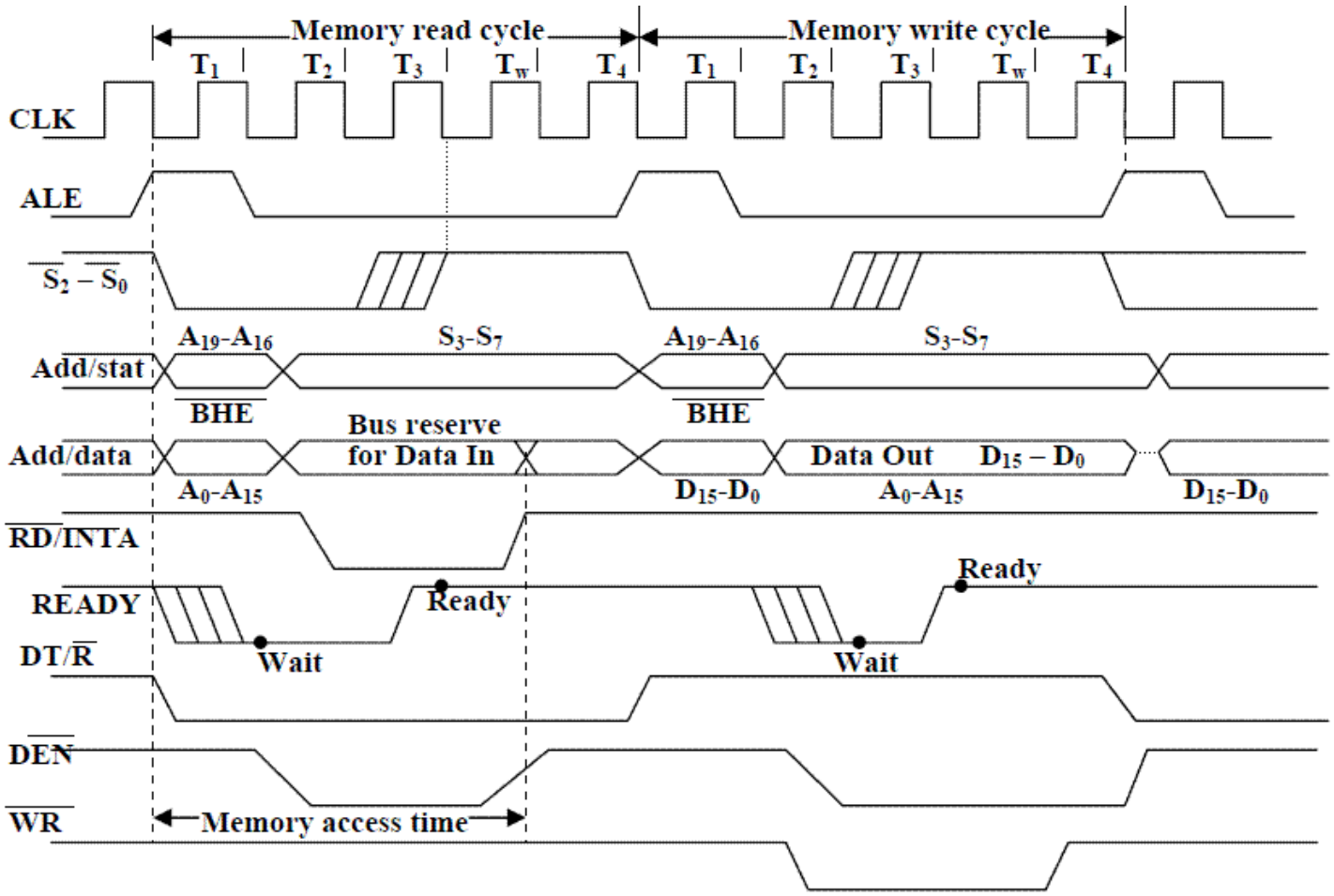


Fig. General bus operation timing diagram

8086 Bus activities during a Read machine cycle:

Fig above (left half portion) shows the timing diagram of 8086 read machine cycle with WAIT state. The clock (CLK) signal is obtained from the clock-generator 8284. Each cycle of the clock is referred to as a state. Minimum number of states to access a data is four. They are T1, T2, T3, and T4 states.

During T1 state of a read machine cycle an 8086 first asserts the M/$\overline{IO}$ signal. It will assert this signal high if it is going to read from memory during memory read cycle and it will assert M/$\overline{IO}$ low if it is going to do a read from an Input port during its read cycle. The timing diagram in fig shows two lines for the M/$\overline{IO}$ signal, because the signal may be going LOW or going HIGH for a read cycle. The point where the two lines cross indicate the time at which the signal becomes valid for this machine cycle.

After asserting M/$\overline{IO}$, the 8086 sends out a high on the address latch enable signal, ALE. The microprocessor sends out on AD0-AD15, A16 through A19 and $\overline{BHE}$ lines, and the address of the memory location that it wants to read. Since the latches are enabled by ALE being high, this address information passes through the latches to their outputs. The 8086 then makes the ALE output low. This disables the latches (8282) and holds the address information latched on the latch outputs. The address information latched on the latch outputs can now be used to select the desired memory or port location.

In the timing diagram, the first point at which the two ($AD_0 - AD_{15}$) cross represents the time at which the 8086 has put a valid address on these lines. Two lines DO NOT indicate that all 16 lines are going high or going low at this point. The crossed lines indicate the time at which a valid address is on the bus.

Since the address information is now held on the latch, the 8086 does not need to send it out any more. As shown in fig. 12 the 8086 floats the AD0 - AD15 lines so that they can be used to input data from memory or from a port. At about the same time the 8086 also remove the $\overline{BHE}$ and A16-A19 information from the upper lines and sends out some status information on these lines.

The 8086 is now ready to read data from the addressed memory locations or port. During T2-state the 8086 asserts its $\overline{RD}$ signal low. This signal is used to enable the addressed memory device or port device.

At the end of T3 state the microprocessor makes the $\overline{RD}$ signal high and reads the data available on the data bus, provided the READY input signal is high. It is the duty of the external circuit to see that valid data is made available on the data bus.

If the READY input pin is not high at the sampled time in a machine cycle, the 8086 will insert one or more WAIT states between T3 and T4 states in that machine cycle. An external hardware device is set up to pulse READY low before the rising edge of the clock in T2 state. After the 8086 finishes T3 of the machine cycle, it enters a WAIT state.

If the READY input is still low at the end of a WAIT state, then the 8086 will insert another WAIT state. The 8086 will continue inserting WAIT states until the READY input is sampled high again. If the READY input is sampled high again during T3 or during the WAIT state, the microprocessor comes out of the WAIT state and will initiate T4 of the machine cycle.

The $\overline{DEN}$ signal is used to enable bi-directional buffers on the data bus. The data enable signal, $\overline{DEN}$, from the 8086 will enable the data buffer when it is asserted LOW. The data transmit / receive signal DT/$\overline{R}$ from the 8086 is used to specify the direction in which the buffers are enabled. When DT/$\overline{R}$ is asserted high, the buffers will, if enabled by $\overline{DEN}$, transmit data from the 8086 to Memory or I/O ports. When DT/$\overline{R}$ is asserted low, the buffers, if enabled by $\overline{DEN}$, will allow data to be received from Memory or I/O ports of the 8086. DT/$\overline{R}$ is asserted during T1 of the machine cycle. The $\overline{DEN}$ is asserted after the 8086 finishes using the data bus to send the lower 16 address bits

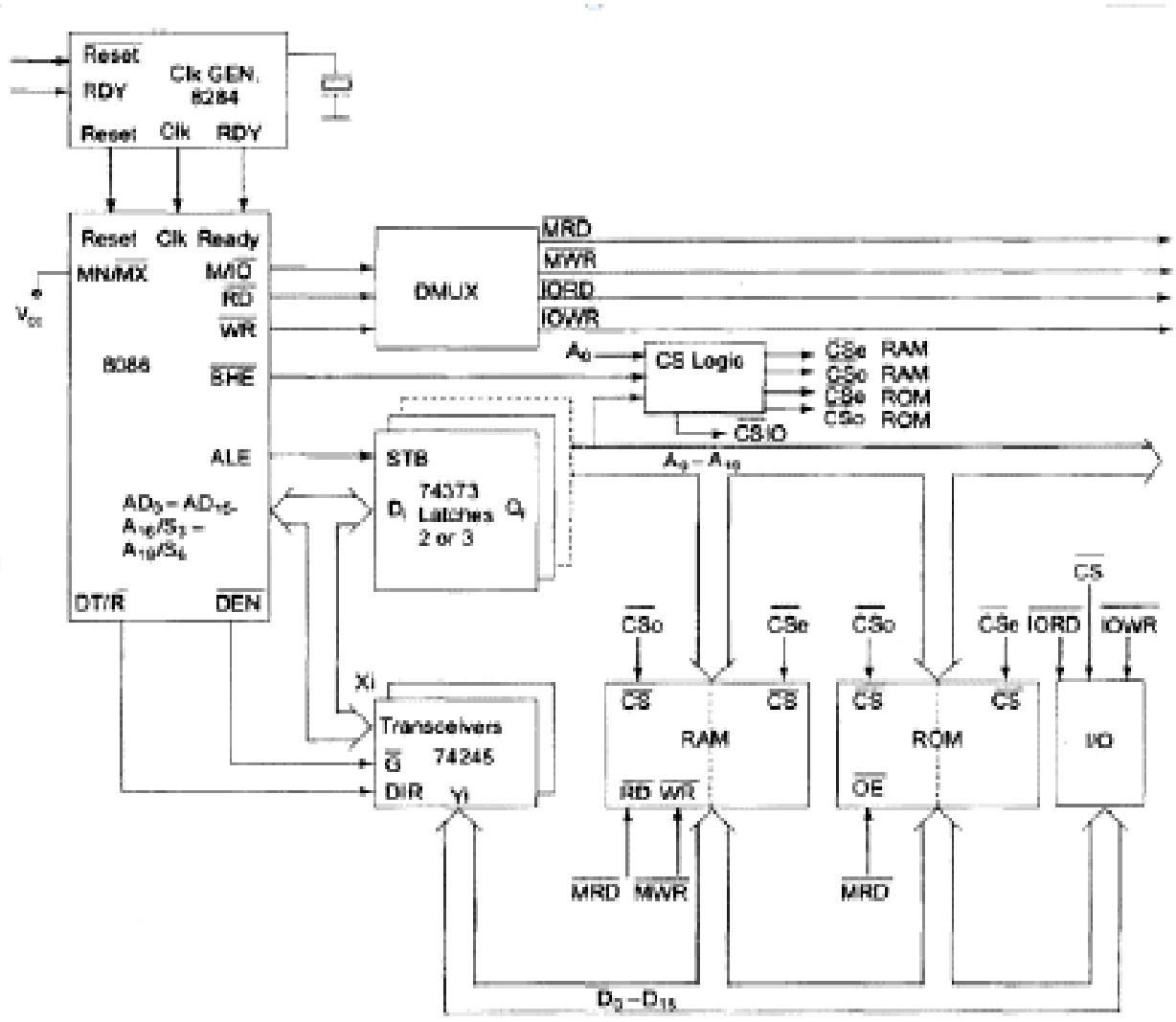
8086 Bus activities during write machine cycle:

The 8086 write operation is very similar to the read cycle. During T1 of a write machine cycle the 8086 asserts M/$\overline{IO}$ low if the write is going to a port and it asserts M/$\overline{IO}$ high if the write is going to memory. At about the same time the 8086 raises ALE high to enable the address latches. The 8086 then assert $\overline{BHE}$ and on the lines AD0 - AD19, it output the address that it will be writing to. When writing to a port, line A16 - A19 will always be low, because the 8086 only sends out 16-bits port addresses. The 8086 brings ALE low again to latch the address on the outputs of the latches. In addition to holding the address, the latches also function as buffers for the address lines. After the address information is latched, the 8086 remove the address information from AD0 - AD15 and outputs the desired data on these lines.

If the READY input is sampled LOW by the 8086 before or during T2 of the machine cycle, the 8086 will insert a WAIT state after T3. If the READY input is sampled high before the end of the WAIT state, the 8086 will go on with state T4 as soon as it completes the WAIT state. The 8086 will continue to insect wait states for as long as the READY is sampled low just before the end of each WAIT state.

MINIMUM MODE 8086 SYSTEM AND TIMINGS

Fig.1: Minimum mode 8086 system



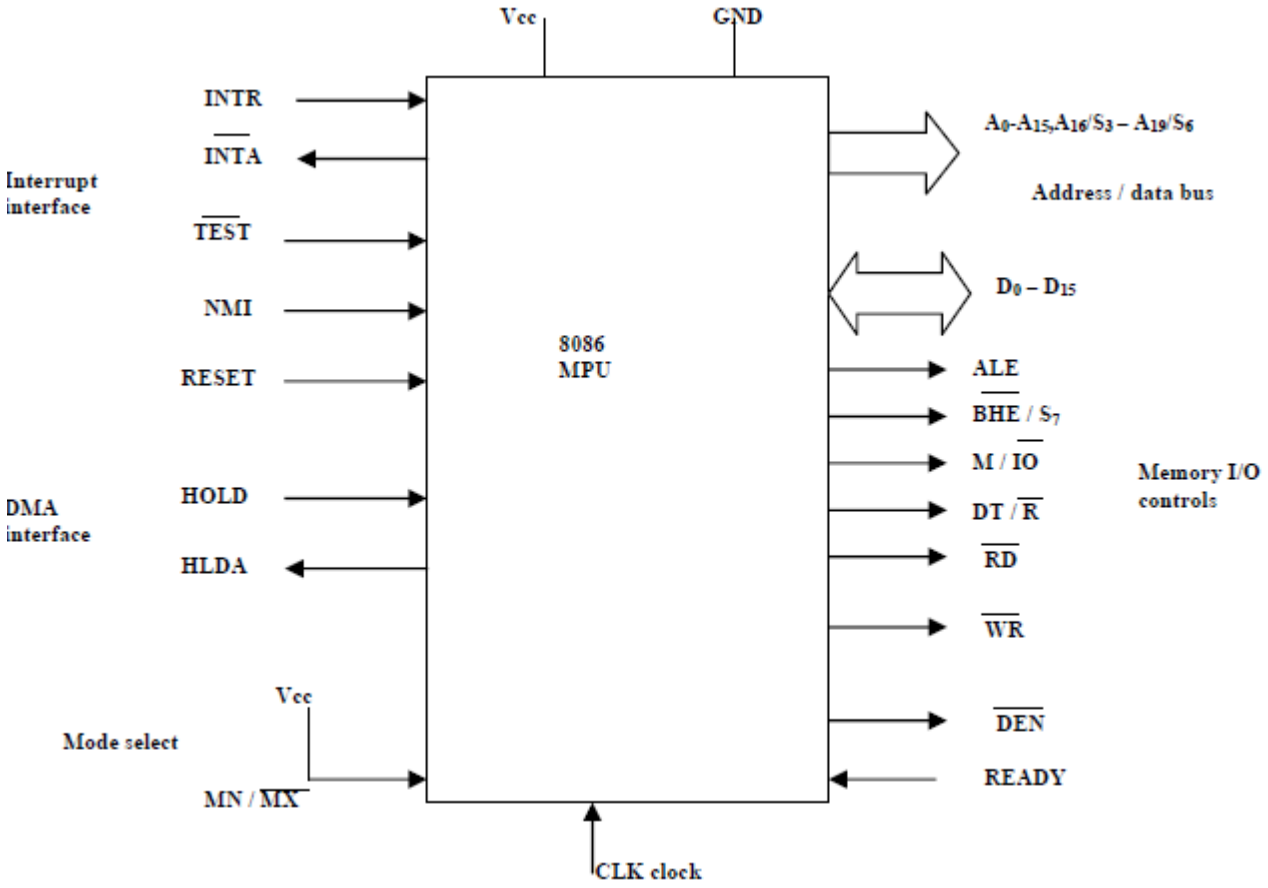


Fig.2: Pin diagram of minimum mode 8086 system

•In a minimum mode 8086 system, the microprocessor 8086 is operated in minimum mode by strapping its MN/$\overline{MX}$ pin to logic 1. In this mode, all the control signals are given out by the microprocessor chip itself. There is a single microprocessor in the minimum mode system. The remaining components in the system are latches, transreceivers, clock generator, memory and I/O devices. Some type of chip selection logic may be required for selecting memory or I/O devices, depending upon the address map of the system.

•Latches are generally buffered output D-type flip-flops like 74LS373 or 8282. They are used for separating the valid address from the multiplexed address/data signals and are controlled by the ALE signal generated by 8086.

•Transreceivers are the bidirectional buffers and sometimes they are called as data amplifiers. They are required to separate the valid data from the time multiplexed address/data signals. They are controlled by two signals namely, $\overline{DEN}$ and DT/$\bar{R}$. The DT/$\bar{R}$ signal indicates the direction of data, i.e. from or to the processor. The system contains memory for the monitor and users program storage.

•Usually, EPROM is used for monitor storage, while RAM for user's program storage. A system may contain I/O devices.

•The working of the minimum mode configuration system can be better described in terms of the timing diagrams rather than qualitatively describing the operations.

•The opcode fetch and read cycles are similar. Hence the timing diagram can be categorized in two parts, the first is the timing diagram for **read cycle** and the second is the timing diagram for **write cycle.**

Read Cycle:

The read cycle begins in T1 with the assertion of address latch enable (ALE) signal and also M / $\overline{IO}$ signal. During the negative going edge of this signal, the valid address is latched on the local bus.

•The $\overline{BHE}$ and A0 signals address low, high or both bytes. From T1 to T4 , the M/$\overline{IO}$ signal indicates a memory or I/O operation.
•At T2, the address is removed from the local bus and is sent to the output. The bus is then tristated. The read ($\overline{RD}$) control signal is also activated in T2.
•The read ($\overline{RD}$) signal causes the address device to enable its data bus drivers. After $\overline{RD}$ goes low, the valid data is available on the data bus.
•The addressed device will drive the READY line high. When the processor returns the read signal to high level, the addressed device will again tristate its bus drivers.

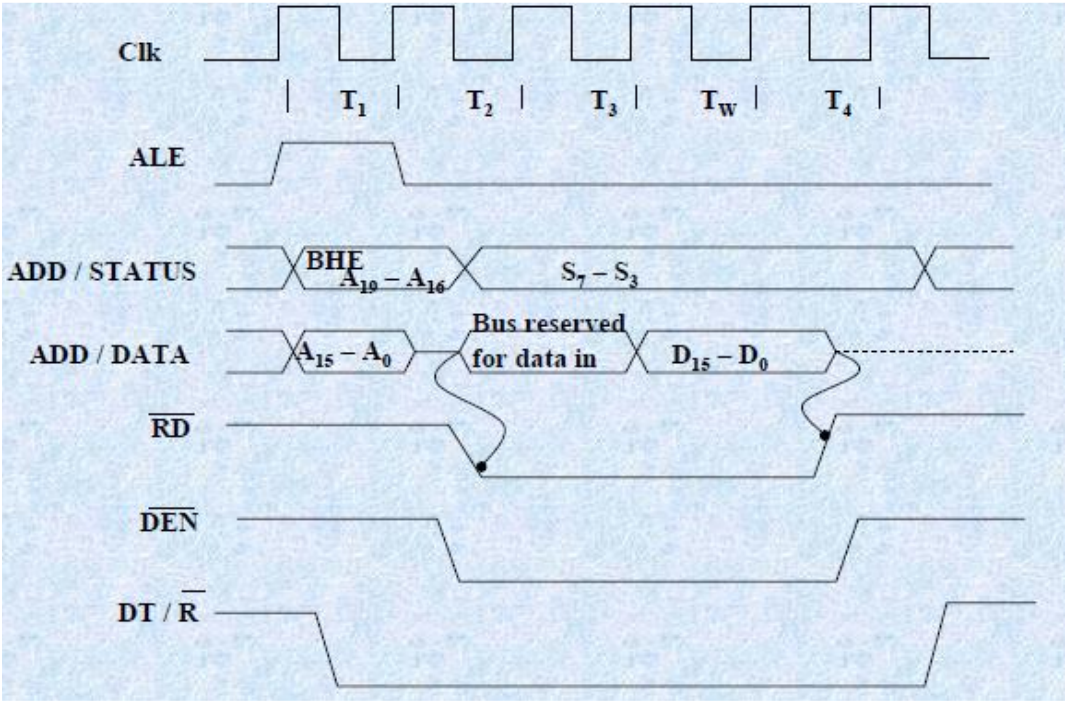


Fig. Read cycle timing diagram for minimum mode operation

Write Cycle:

•A write cycle also begins with the assertion of ALE and the emission of the address. The M/$\overline{IO}$ signal is again asserted to indicate a memory or I/O operation. In T2, after sending the address in T1, the processor sends the data to be written to the addressed location.
•The data remains on the bus until middle of T4 state. The $\overline{WR}$ becomes active at the beginning of T2 (unlike $\overline{RD}$ is somewhat delayed in T2 to provide time for floating).
•The $\overline{BHE}$ and A0 signals are used to select the proper byte or bytes of memory or I/O word to be read or write.

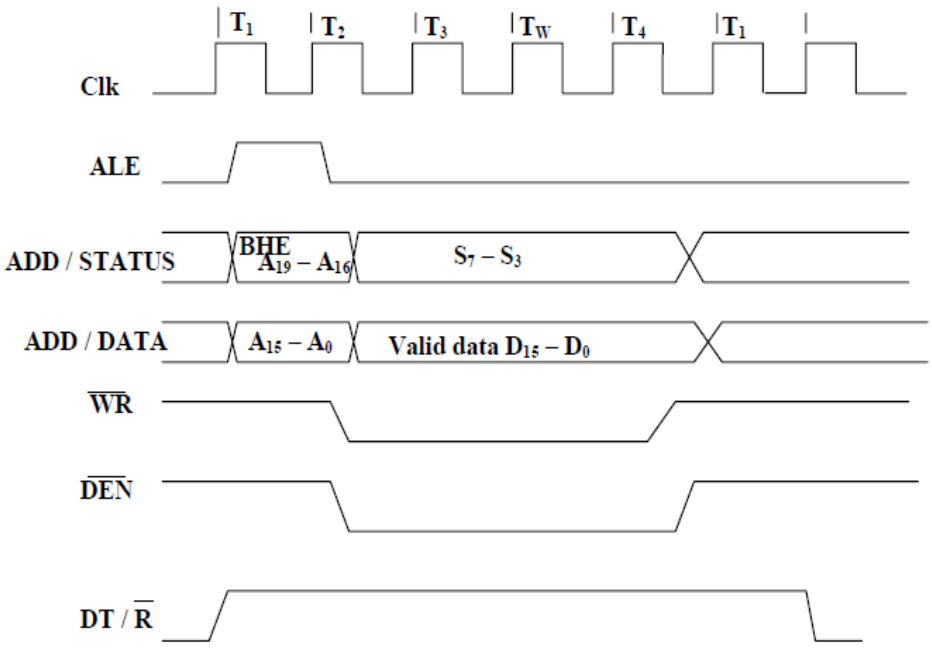


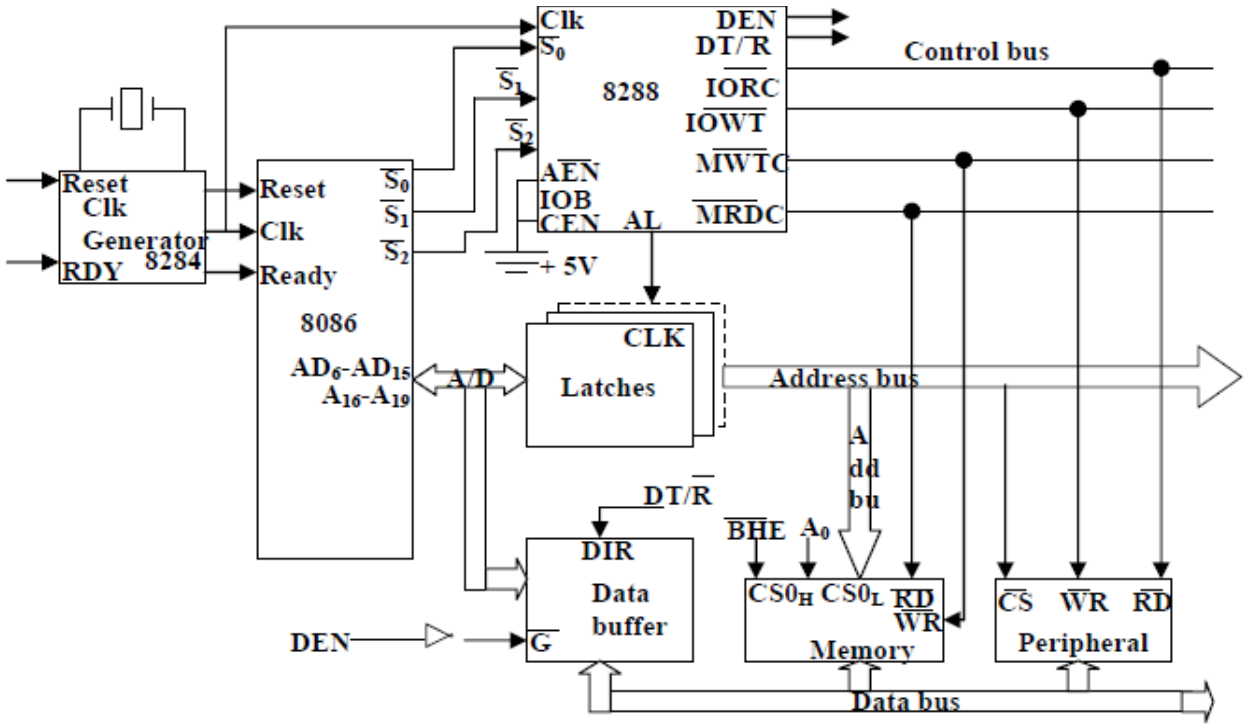
Fig. Write cycle timing diagram for minimum mode operation

The M/$\overline{IO}$, $\overline{RD}$ and $\overline{WR}$ signals indicate the type of data transfer as specified in table below.

M / $\overline{IO}$	$\overline{RD}$	$\overline{WR}$	Transfer Type
0	0	1	I / O read
0	1	0	I/O write
1	0	1	Memory read
1	1	0	Memory write

MAXIMUM MODE 8086 SYSTEM AND TIMINGS

When the 8086 is set for the maximum-mode configuration, it provides signals for implementing a multiprocessor / coprocessor system environment. By multiprocessor environment we mean that one microprocessor exists in the system and that each processor is executing its own program. Usually in this type of system environment, there are some system resources that are common to all processors. They are called as global resources. There are also other resources that are assigned to specific processors. These are known as local or private resources. Coprocessor also means that there is a second processor in the system. In this two processor does not access the bus at the same time. One passes the control of the system bus to the other and then may suspend its operation. In the maximum-mode 8086 system, facilities are provided for implementing allocation of global resources and passing bus control to other microprocessor or coprocessor.



Maximum Mode 8086 System.

•In the maximum mode, the 8086 is operated by strapping the MN/$\overline{MX}$ pin to ground. In this mode, the processor derives the status signal $\overline{S2}$, $\overline{S1}$, $\overline{S0}$. Another chip called bus controller derives the control signal using this status information. In the maximum mode, there may be more than one microprocessor in the system configuration. The components in the system are same as in the minimum mode system.

•The basic function of the bus controller chip IC8288 is to derive control signals like $\overline{RD}$ and $\overline{WR}$ (for memory and I/O devices), $\overline{DEN}$, DT/$\bar{R}$, ALE etc. using the information by the processor on the status lines.

•The bus controller chip has input lines $\overline{S2}$, $\overline{S1}$, $\overline{S0}$ and CLK. These inputs to 8288 are driven by CPU.

•It derives the outputs ALE, $\overline{DEN}$, DT/$\bar{R}$, $\overline{MRDC}$, $\overline{MWTC}$, $\overline{AMWC}$, $\overline{IORC}$, $\overline{IOWC}$ and $\overline{AIOWC}$. The AEN, IOB and CEN pins are specially useful for multiprocessor systems.

•AEN and IOB are generally grounded. CEN pin is usually tied to +5V. The significance of the MCE/PDEN output depends upon the status of the IOB pin.

•If IOB is grounded, it acts as master cascade enable to control cascade 8259A, else it acts as peripheral data enable used in the multiple bus configurations.

•$\overline{INTA}$ pin used to issue two interrupt acknowledge pulses to the interrupt controller or to an interrupting device.

•$\overline{IORC}$, $\overline{IOWC}$ are I/O read command and I/O write command signals respectively. These signals enable an IO interface to read or write the data from or to the address port.

•The $\overline{MRDC}$, $\overline{MWTC}$ are memory read command and memory write command signals respectively and may be used as memory read or write signals.

•All these command signals instructs the memory to accept or send data from or to the bus.

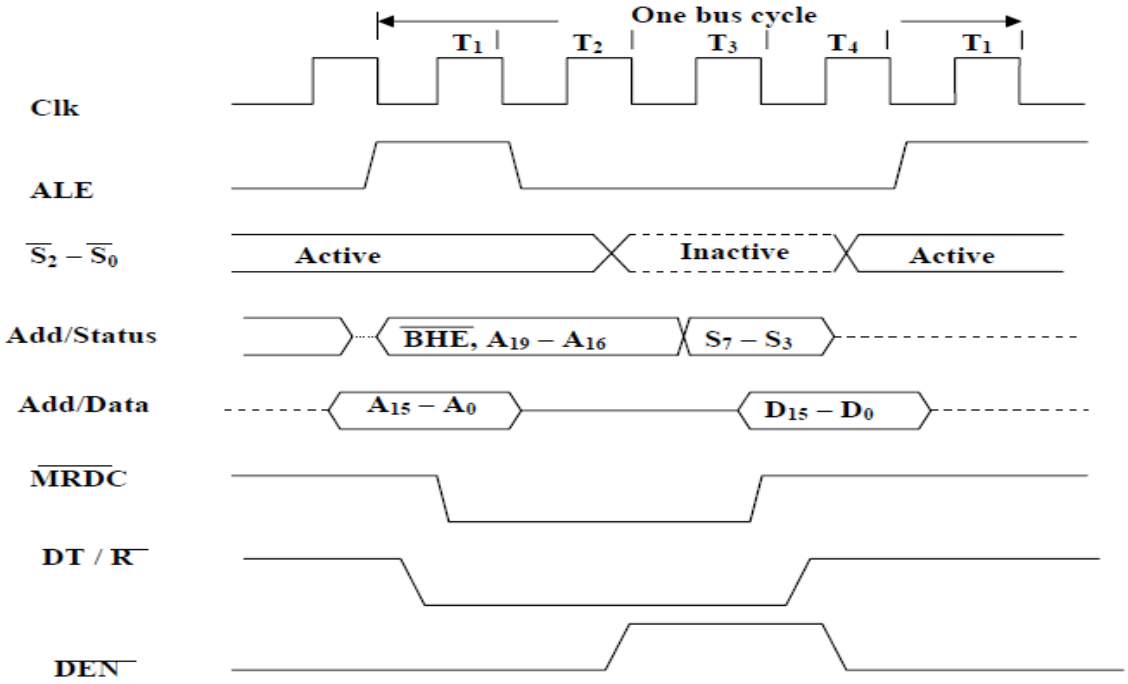
•For both of these write command signals, the advanced signals namely $\overline{AIOWC}$ and $\overline{AMWTC}$ are available.

•Here the only difference between in timing diagram between minimum mode and maximum mode is the status signals used and the available control and advanced command signals.

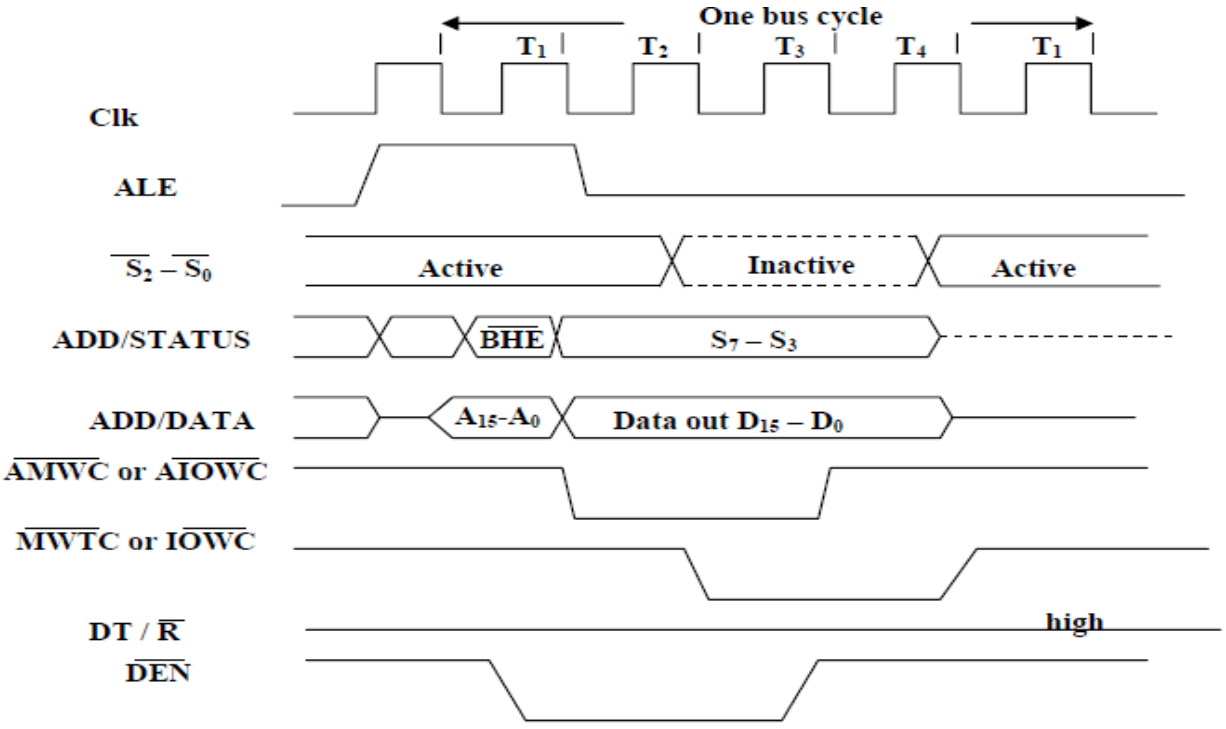
•R0, S1, S2 are set at the beginning of bus cycle.8288 bus controller will output a pulse as on the ALE and apply a required signal to its DT / R pin during T1.

•In T2, 8288 will set DEN=1 thus enabling transceivers, and for an input it will activate MRDC or IORC. These signals are activated until T4. For an output, the AMWC or AIOWC is activated from T2 to T4 and MWTC or IOWC is activated from T3 to T4.

•The status bit S0 to S2 remains active until T3 and become passive during T3 and T4.
•If reader input is not activated before T3, wait state will be inserted between T3 and T4.



Memory Read Timing in Maximum Mode



Memory Write Timing in Maximum mode.