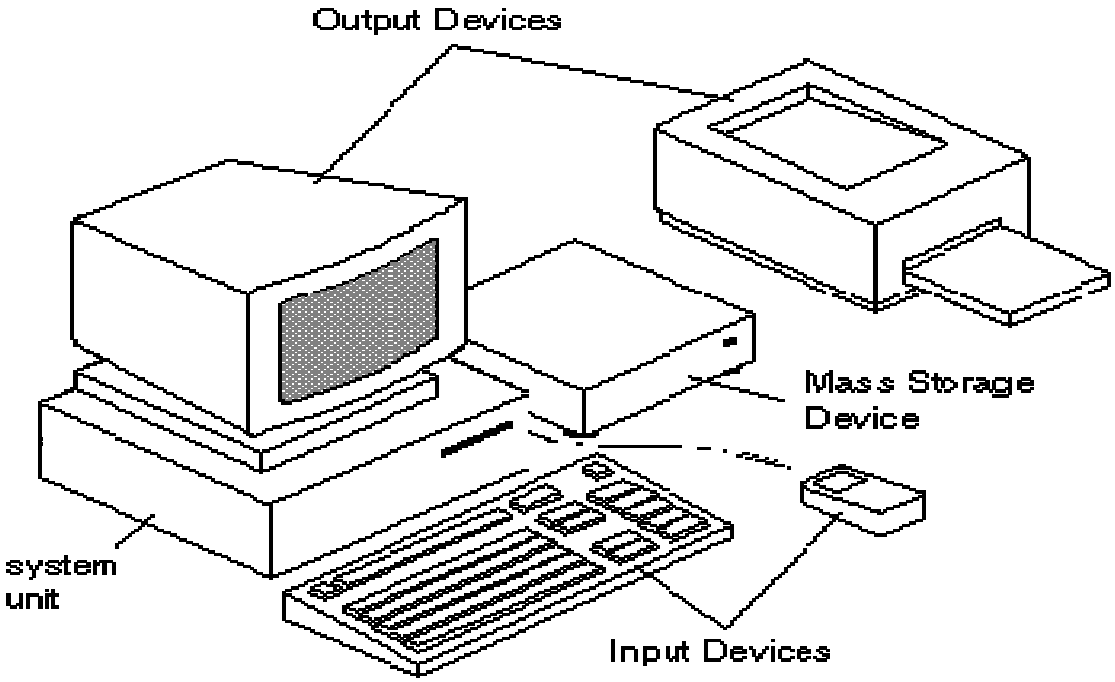
Module 1 learning unit 1

- A **Computer** is a programmable machine.
- The two principal characteristics of a computer are:
- It responds to a specific set of instructions in a well-defined manner.
- It can execute a prerecorded list of instructions (a program).
- Modern computers are electronic and digital.
- The actual machinery wires, transistors, and circuits is called hardware. the instructions and data are called software.
-



- All general-purpose computers require the following hardware components:
- **Memory**: Enables a computer to store, at least temporarily, data and programs.
- **Mass storage device**: Allows a computer to permanently retain large amounts of data. Common mass storage devices include disk drives and tape drives.
- **Input device**: Usually a keyboard and mouse are the input device through which data and instructions enter a computer.
- **Output device**: A display screen, printer, or other device that lets you see what the computer has accomplished.
- **Central processing unit (CPU):** The heart of the computer, this is the component that actually executes instructions.
- In addition to these components, many others make it possible for the basic components to work together efficiently.
- For example, every computer requires a bus that transmits data from one part of the computer to another.

- Computers can be generally classified by size and power as follows, though there is considerable overlap:
- **Personal computer**: A small, single-user computer based on a microprocessor.
- In addition to the microprocessor, a personal computer has a keyboard for entering data, a monitor for displaying information, and a storage device for saving data.
- **Working station**: A powerful, single-user computer. A workstation is like a personal computer, but it has a more powerful microprocessor and a higher-quality monitor.
- **Minicomputer**: A multi-user computer capable of supporting from 10 to hundreds of users simultaneously.
- **Mainframe**: A powerful multi-user computer capable of supporting many hundreds or thousands of users simultaneously.
- **Supercomputer:** An extremely fast computer that can perform hundreds of millions of instructions per second.

Minicomputer:

- A midsized computer. In size and power, minicomputers lie between workstations and mainframes.
- A minicomputer, a term no longer much used, is a computer of a size intermediate between a microcomputer and a mainframe.
- Typically, minicomputers have been stand-alone computers (computer systems with attached terminals and other devices) sold to small and mid-size businesses for general business applications and to large enterprises for department-level operations.
- In recent years, the minicomputer has evolved into the "mid-range server" and is part of a network. IBM's AS/400e is a good example.
- The AS/400 - formally renamed the "IBM iSeries," but still commonly known as AS/400 - is a midrange server designed for small businesses and departments in large enterprises and now redesigned so that it will work well in distributed networks with Web applications.
- The AS/400 uses the PowerPC microprocessor with its reduced instruction set computer technology. Its operating system is called the OS/400.
- With multi-terabytes of disk storage and a Java virtual memory closely tied into the operating system, IBM hopes to make the AS/400 a kind of versatile all-purpose server that can replace PC servers and Web servers in the world's businesses, competing with both Wintel and Unix servers, while giving its present enormous customer base an immediate leap into the Internet.

Workstation:

1) A type of computer used for engineering applications (CAD/CAM), desktop publishing, software development, and other types of applications that require a moderate amount of computing power and relatively high quality graphics capabilities.
- Workstations generally come with a large, high- resolution graphics screen, at least 64 MB (mega bytes) of RAM, built-in network support, and a graphical user interface.

- Most workstations also have a mass storage device such as a disk drive, but a special type of workstation, called a diskless workstation, comes without a disk drive.
- The most common operating systems for workstations are UNIX and Windows NT.
- In terms of computing power, workstations lie between personal computers and minicomputers, although the line is fuzzy on both ends.
- High-end personal computers are equivalent to low-end workstations. And high-end workstations are equivalent to minicomputers.
- Like personal computers, most workstations are single-user computers. However, workstations are typically linked together to form a local-area network, although they can also be used as stand-alone systems.

2) In networking, *workstation* refers to any computer connected to a local-area network. It could be a workstation or a personal computer.

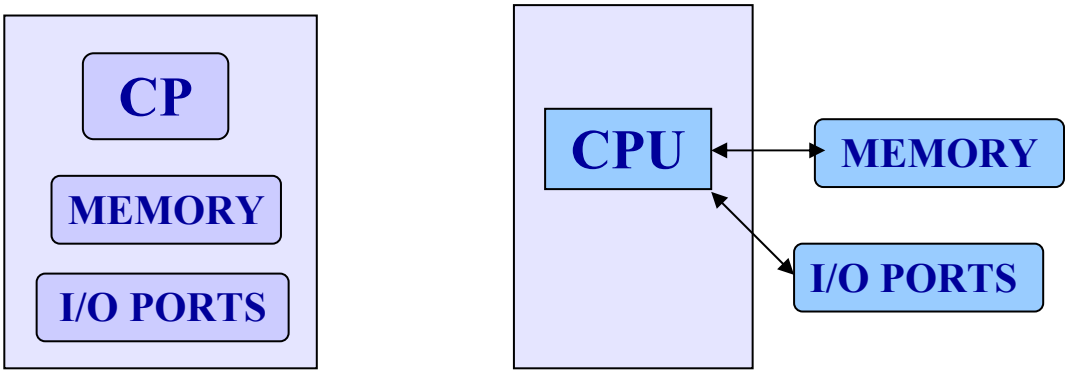
- **Mainframe:** A very large and expensive computer capable of supporting hundreds, or even thousands, of users simultaneously. In the hierarchy that starts with a simple microprocessors (in watches, for example) at the bottom and moves to supercomputer at the top, mainframes are just below supercomputers.
- In some ways, mainframes are more powerful than supercomputers because they support more simultaneous programs.
- But supercomputers can execute a single program faster than a mainframe. The distinction between small mainframes and minicomputers is vague, depending really on how the manufacturer wants to market its machines.
- **Microcomputer:** The term *microcomputer* is generally synonymous with personal computer, or a computer that depends on a microprocessor.
- Microcomputers are designed to be used by individuals, whether in the form of PCs, workstations or notebook computers.
- A microcomputer contains a CPU on a microchip (the microprocessor), a memory system (typically ROM and RAM), a bus system and I/O ports, typically housed in a motherboard.
- **Microprocessor**: A silicon chip that contains a CPU. In the world of personal computers, the terms *microprocessor* and CPU are used interchangeably.
- A **microprocessor** (sometimes abbreviated **µP**) is a digital electronic component with miniaturized transistors on a single semiconductor integrated circuit (IC).
- One or more microprocessors typically serve as a central processing unit (CPU) in a computer system or handheld device.
- Microprocessors made possible the advent of the microcomputer.
- At the heart of all personal computers and most working stations sits a microprocessor.
- Microprocessors also control the logic of almost all digital devices, from clock radios to fuel-injection systems for automobiles.
- Three basic characteristics differentiate microprocessors:
- **Instruction set**: The set of instructions that the microprocessor can execute.
- **Bandwidth**: The number of bits processed in a single instruction.
- **Clock speed**: Given in megahertz (MHz), the clock speed determines how many instructions per second the processor can execute.

- In both cases, the higher the value, the more powerful the CPU. For example, a 32 bit microprocessor that runs at 50MHz is more powerful than a 16-bit microprocessor that runs at 25MHz.
- In addition to bandwidth and clock speed, microprocessors are classified as being either RISC (reduced instruction set computer) or CISC (complex instruction set computer).
- **Supercomputer**: A supercomputer is a computer that performs at or near the currently highest operational rate for computers.
- A supercomputer is typically used for scientific and engineering applications that must handle very large databases or do a great amount of computation (or both).
- At any given time, there are usually a few well-publicized supercomputers that operate at the very latest and always incredible speeds.
- The term is also sometimes applied to far slower (but still impressively fast) computers.
- Most supercomputers are really multiple computers that perform parallel processing.
- In general, there are two parallel processing approaches: symmetric multiprocessing (SMP) and massively parallel processing (MPP).
- **Microcontroller:** A highly integrated chip that contains all the components comprising a controller.
- Typically this includes a CPU, RAM, some form of ROM, I/O ports, and timers.
- Unlike a general-purpose computer, which also includes all of these components, a microcontroller is designed for a very specific task - to control a particular system.
- A microcontroller differs from a microprocessor, which is a general-purpose chip that is used to create a multi-function computer or device and requires multiple chips to handle various tasks.
- A microcontroller is meant to be more self-contained and independent, and functions as a tiny, dedicated computer.
- The great advantage of microcontrollers, as opposed to using larger microprocessors, is that the parts-count and design costs of the item being controlled can be kept to a minimum.
- They are typically designed using CMOS (complementary metal oxide semiconductor) technology, an efficient fabrication technique that uses less power and is more immune to power spikes than other techniques.
- Microcontrollers are sometimes called *embedded microcontrollers,* which just means that they are part of an embedded system that is, one part of a larger device or system.
- **Controller:** A device that controls the transfer of data from a computer to a peripheral device and vice versa.
- For example, disk drives, display screens, keyboards and printers all require controllers.
- In personal computers, the controllers are often single chips.
- When you purchase a computer, it comes with all the necessary controllers for standard components, such as the display screen, keyboard, and disk drives.

- If you attach additional devices, however, you may need to insert new controllers that come on expansion boards.
- Controllers must be designed to communicate with the computer's expansion bus.
- There are three standard bus architectures for PCs - the AT bus, PCI (Peripheral Component Interconnect) and SCSI.
- When you purchase a controller, therefore, you must ensure that it conforms to the bus architecture that your computer uses.
- Short for ***P**eripheral **C**omponent **I**nterconnect,* a local bus standard developed by Intel Corporation.
- Most modern PCs include a PCI bus in addition to a more general IAS expansion bus.
- PCI is also used on newer versions of the Macintosh computer.
- PCI is a 64-bit bus, though it is usually implemented as a 32 bit bus. It can run at clock speeds of 33 or 66 MHz.
- At 32 bits and 33 MHz, it yields a throughput rate of 133 MBps.
- Short for ***s**mall **c**omputer **s**ystem **i**nterface*, a parallel interface standard used by Apple Macintosh computers, PCs, and many UNIX systems for attaching peripheral devices to computers.
- Nearly all Apple Macintosh computers, excluding only the earliest Macs and the recent iMac, come with a SCSI port for attaching devices such as disk drives and printers.
- SCSI interfaces provide for faster data transmission rates (up to 80 megabytes per second) than standard serial and parallel ports. In addition, you can attach many devices to a single SCSI port, so that SCSI is really an I/O bus rather than simply an interface
- Although SCSI is an ANSI standard, there are many variations of it, so two SCSI interfaces may be incompatible.
- For example, SCSI supports several types of connectors.
- While SCSI has been the standard interface for Macintoshes, the iMac comes with *IDE*, a less expensive interface, in which the controller is integrated into the disk or CD-ROM drive.
- The following varieties of SCSI are currently implemented:
- SCSI-1**:** Uses an 8-bit bus, and supports data rates of 4 MBps.
- SCSI-2**:** Same as SCSI-1, but uses a 50-pin connector instead of a 25-pin connector, and supports multiple devices. This is what most people mean when they refer to plain *SCSI*.
- Wide SCSI**:** Uses a wider cable (168 cable lines to 68 pins) to support 16-bit transfers.
- Fast SCSI**:** Uses an 8-bit bus, but doubles the clock rate to support data rates of 10 MBps.
- Fast Wide SCSI**:** Uses a 16-bit bus and supports data rates of 20 MBps.
- Ultra SCSI: Uses an 8-bit bus, and supports data rates of 20 MBps.
- Wide Ultra2 SCSI**:** Uses a 16-bit bus and supports data rates of 80 MBps.
- SCSI-3**:** Uses a 16-bit bus and supports data rates of 40 MBps. Also called *Ultra Wide SCSI.*
- Ultra2 SCSI**:** Uses an 8-bit bus and supports data rates of 40 MBps.

- **Embedded system**: A specialized computer system that is part of a larger system or machine.
- Typically, an embedded system is housed on a single microprocessor board with the programs stored in ROM.
- Virtually all appliances that have a digital Interface- watches, microwaves, VCRs, cars -utilize embedded systems.
- Some embedded systems include an operating system, but many are so specialized that the entire logic can be implemented as a single program.

MICRO CONTROLLER	MICRO PROCESSER
• It is a single chip • Consists Memory, I/o ports	• It is a CPU • Memory, I/O Ports to be connected externally



Definitions:

- A **Digital Signal Processor** is a special-purpose CPU (Central Processing Unit) that provides ultra-fast instruction sequences, such as shift and add, and multiply and add, which are commonly used in math-intensive signal processing applications.
- A **digital signal processor** (**DSP**) is a specialized microprocessor designed specifically for ***digital signal processing***, generally in real time.

Digital

- *operating by the use of discrete signals to represent data in the form of numbers.*

Signal

- *a variable parameter by which information is conveyed through an electronic circuit.*

Processing

- *to perform operations on data according to programmed instructions.*

Digital Signal processing

- changing or analysing information which is measured as discrete sequences of numbers.

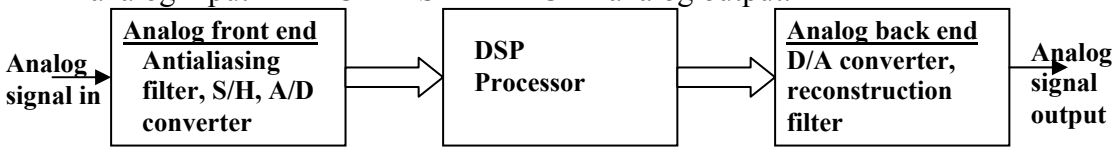
- **Digital signal processing** (DSP) is the study of signals in a digital representation and the processing methods of these signals.
- DSP and analog signal processing are subfields of signal processing.

DSP has three major subfields:

- Audio signal processing, Digital image processing and Speech processing.
- Since the goal of DSP is usually to measure or filter continuous real-world analog signals, the first step is usually to convert the signal from an analog to a digital form, by using an analog to digital converter.
- Often, the required output signal is another analog output signal, which requires a digital to analog converter.

Characteristics of Digital Signal Processors:

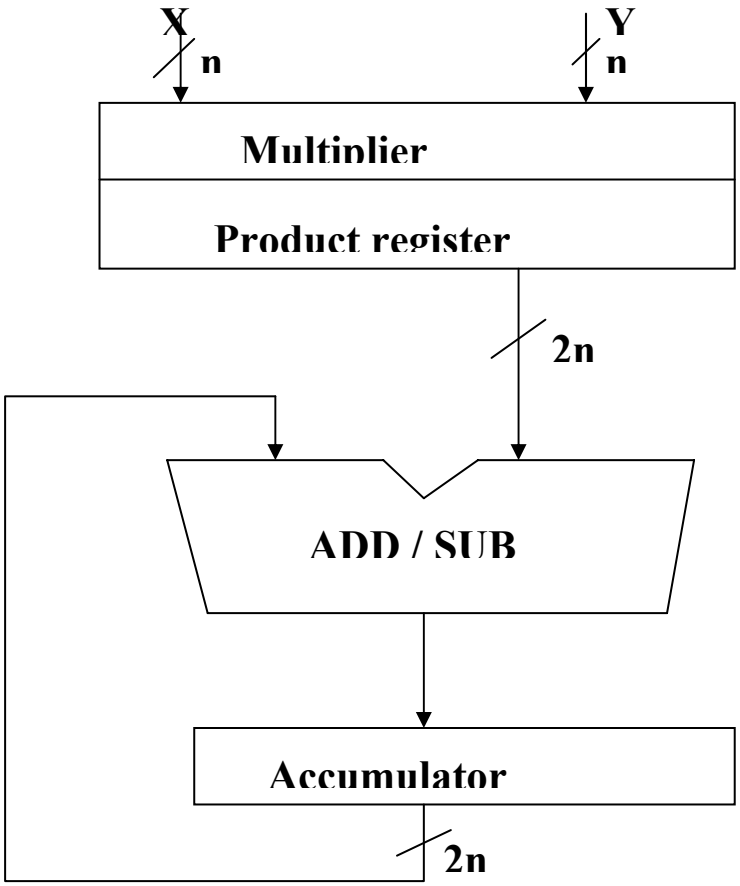
- Separate program and data memories (Harvard architecture).
- Special Instructions for SIMD (Single Instruction, Multiple Data) operations.
- Only parallel processing, no multitasking.
- The ability to act as a direct memory access device if in a host environment.
- Takes digital data from ADC (Analog-Digital Converter) and passes out data which is finally output by converting into analog by DAC (Digital-Analog Converter).
- analog input-->ADC-->DSP-->DAC--> analog output.



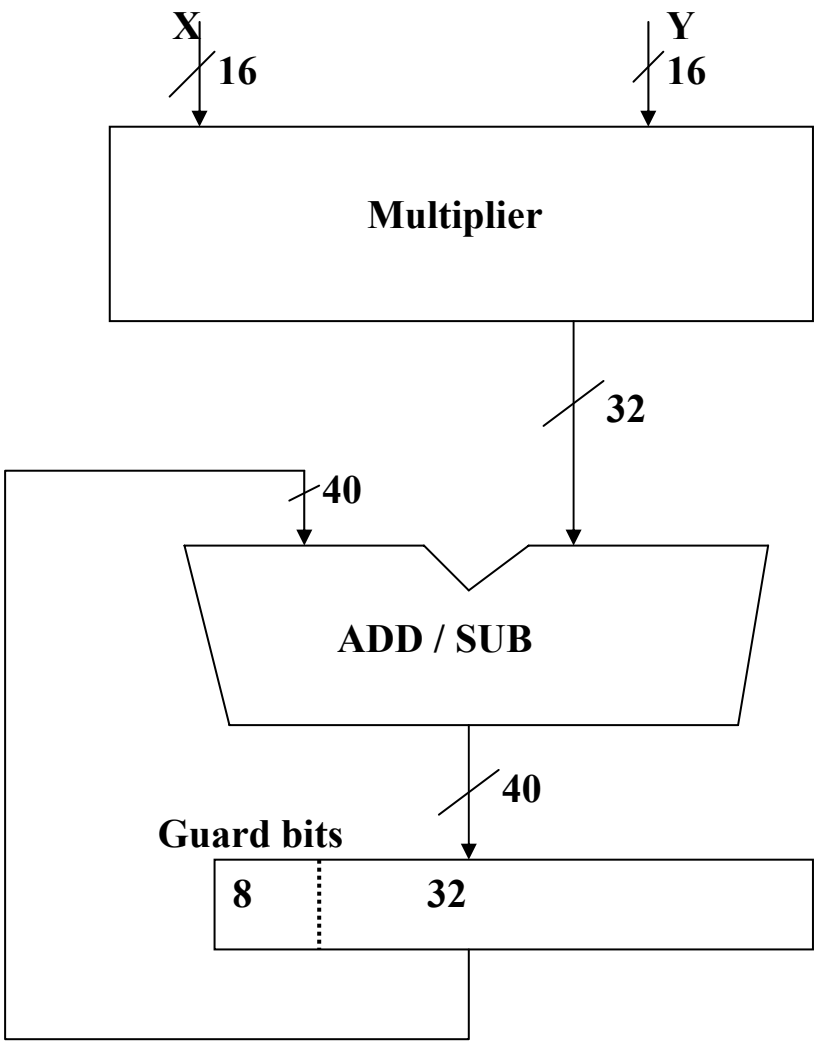
DAP System

Multiply-accumulate hardware:

- Multiply accumulate is the most frequently used operation in digital signal processing.
- In order to implement this efficiently, the DSP has an hardware multiplier, an accumulator with an adequate number of bits to hold the sum of products and at explicit multiply-accumulate instructions.
- ***Harvard architecture***: in this memory architecture, there are two memory spaces. Program memory and data memory.

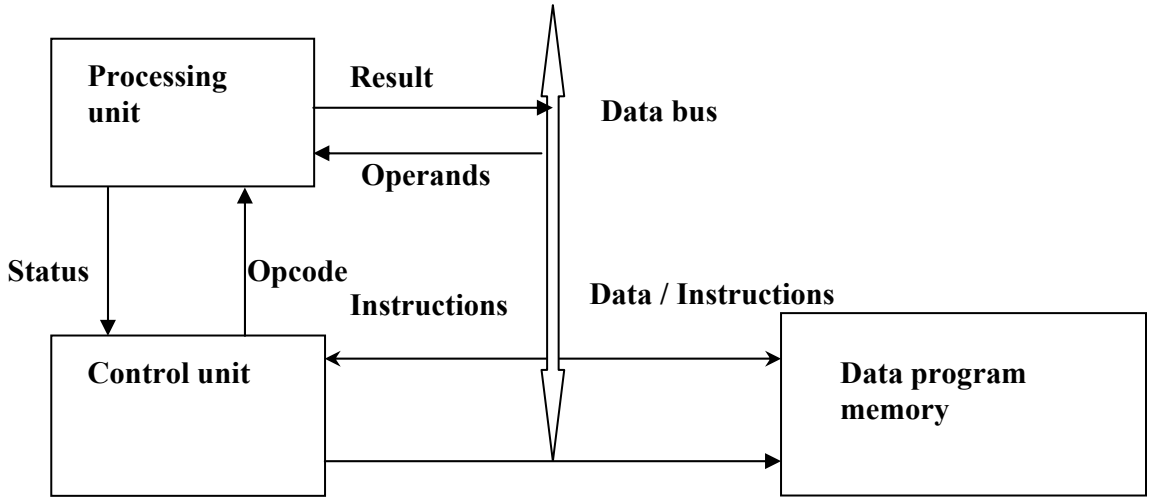


A MAC

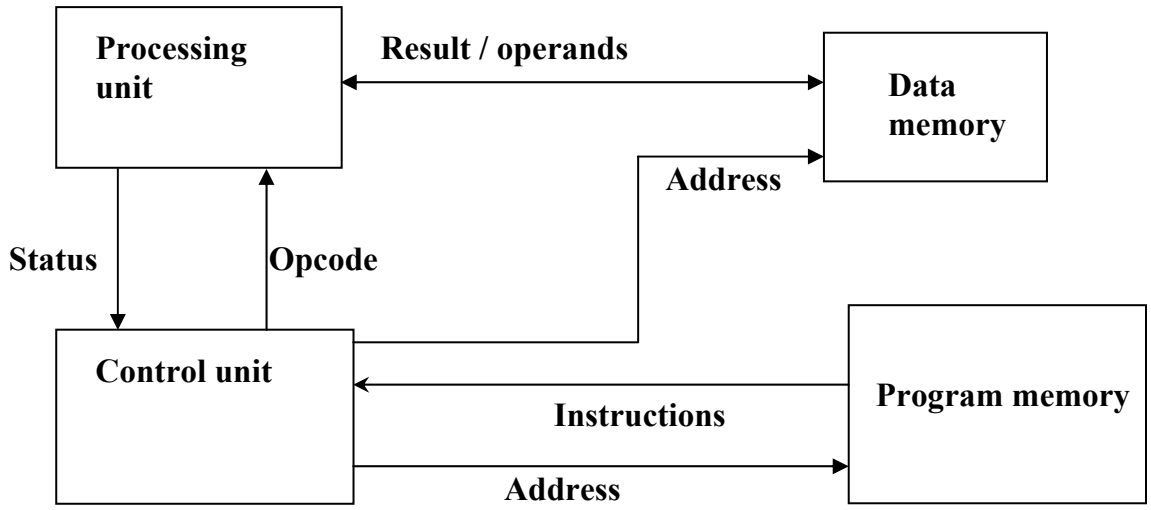


A MAC unit with accumulator guard bits

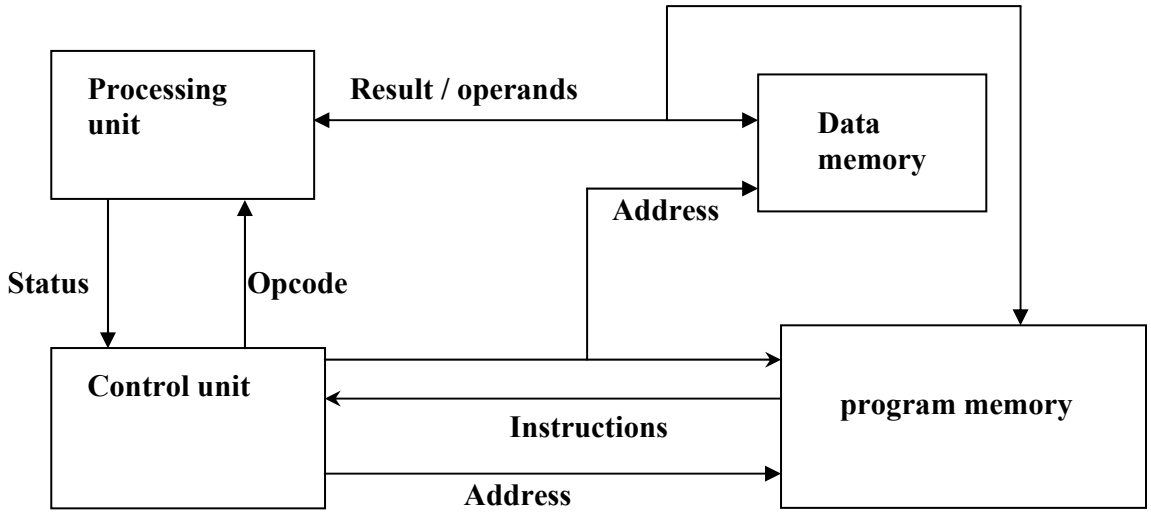
- The processor core connects to these memory spaces by two separate bus sets, allowing two simultaneous access to memory. This arrangement doubles the processor memory bandwidth.
-
- ***Zero-overhead looping***: one common characteristics of DSP algorithms is that most of the processing time is split on executing instructions contained with relatively small loops.
- The term zero overhead looping means that the processor can execute loops without consuming cycles to test the value of the loop counter, perform a conditional branch to the top of the loop, and decrement the loop counter.



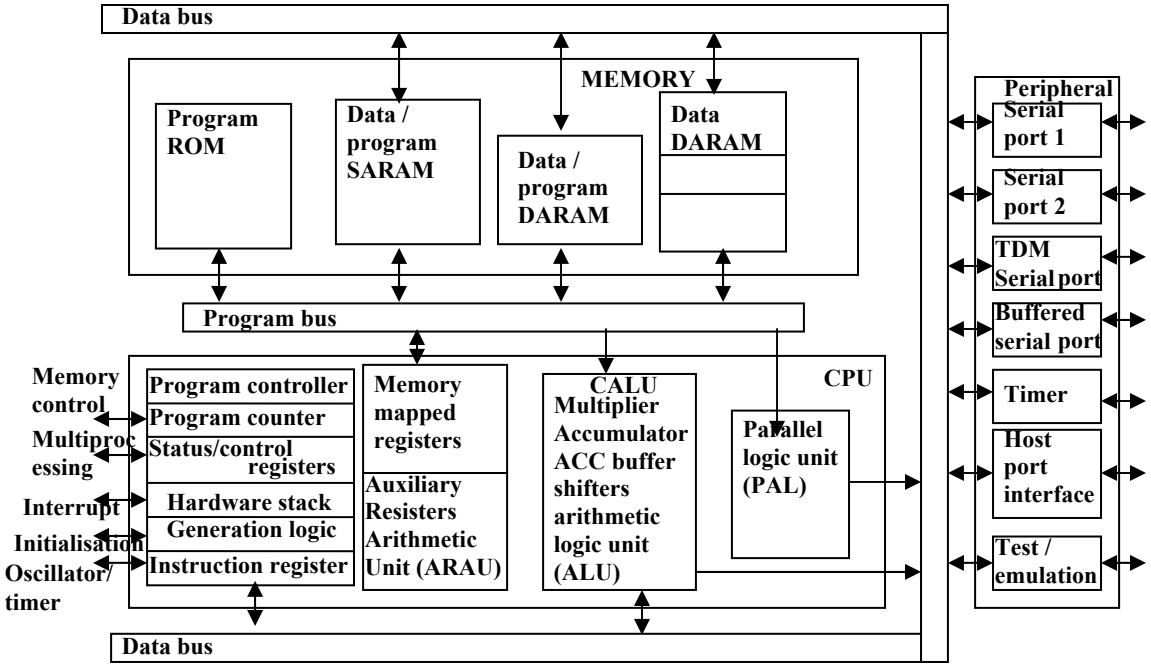
Von Neuman Architecture



Harvard Architecture



Modified Harvard Architecture



Internal Architecture of the TMS320C5X DSP

- The advantages of DSP are:

Versatility:

- digital systems can be reprogrammed for other applications (at least where programmable DSP chips are used)
- digital systems can be ported to different hardware (for example a different DSP chip or board level product)

Repeatability:

- digital systems can be easily duplicated
- digital system responses do not drift with temperature

- digital systems do not depend on strict component tolerances.
-

Simplicity:

- some things can be done more easily digitally than with analogue systems
- DSP is used in a very wide variety of applications but most share some common features:
- they use a lot of multiplying and adding signals.
- they deal with signals that come from the real world.
- they require a response in a certain time.
-

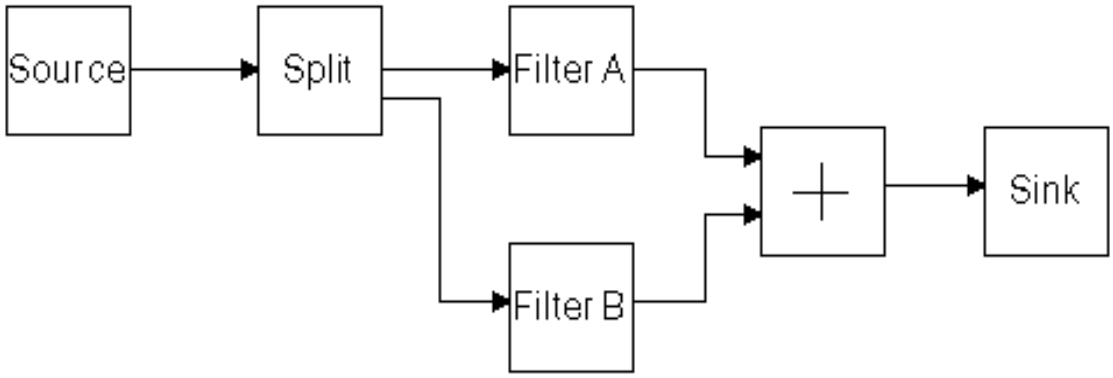


Figure: A block diagram (or dataflow graph)

- What is the difference between a DSP and a microprocessor ?
- The essential difference between a DSP and a microprocessor is that a DSP processor has features designed to support high-performance, repetitive, numerically intensive tasks.
- In contrast, general-purpose processors or microcontrollers (GPPs / MCUs for short) are either not specialized for a specific kind of applications (in the case of general-purpose processors), or they are designed for control-oriented applications (in the case of microcontrollers).
- Features that accelerate performance in DSP applications include:
- Single-cycle multiply-accumulate capability; high-performance DSPs often have two multipliers that enable two multiply-accumulate operations per instruction cycle; some DSP have four or more multipliers.
-
- Specialized addressing modes, for example, pre- and post-modification of address pointers, circular addressing, and bit-reversed addressing.
- Most DSPs provide various configurations of on-chip memory and peripherals tailored for DSP applications. DSPs generally feature multiple-access memory architectures that enable DSPs to complete several accesses to memory in a single instruction cycle.

- Specialized execution control. Usually, DSP processors provide a loop instruction that allows tight loops to be repeated without spending any instruction cycles for updating and testing the loop counter or for jumping back to the top of the loop
- DSP processors are known for their irregular instruction sets, which generally allow several operations to be encoded in a single instruction.
- For example, a processor that uses 32-bit instructions may encode two additions, two multiplications, and four 16-bit data moves into a single instruction.
- In general, DSP processor instruction sets allow a data move to be performed in parallel with an arithmetic operation. GPPs / MCUs, in contrast, usually specify a single operation per instruction.
- What is really important is to choose the processor that is best suited for your application.
- If a GPP/MCU is better suited for your DSP application than a DSP processor, the processor of choice is the GPP/MCU.
- It is also worth noting that the difference between DSPs and GPPs/MCUs is fading: many GPPs/MCUs now include DSP features, and DSPs are increasingly adding microcontroller features.

Module 1: learning unit 2

8085 Microprocessor

ContentsGeneral definitions

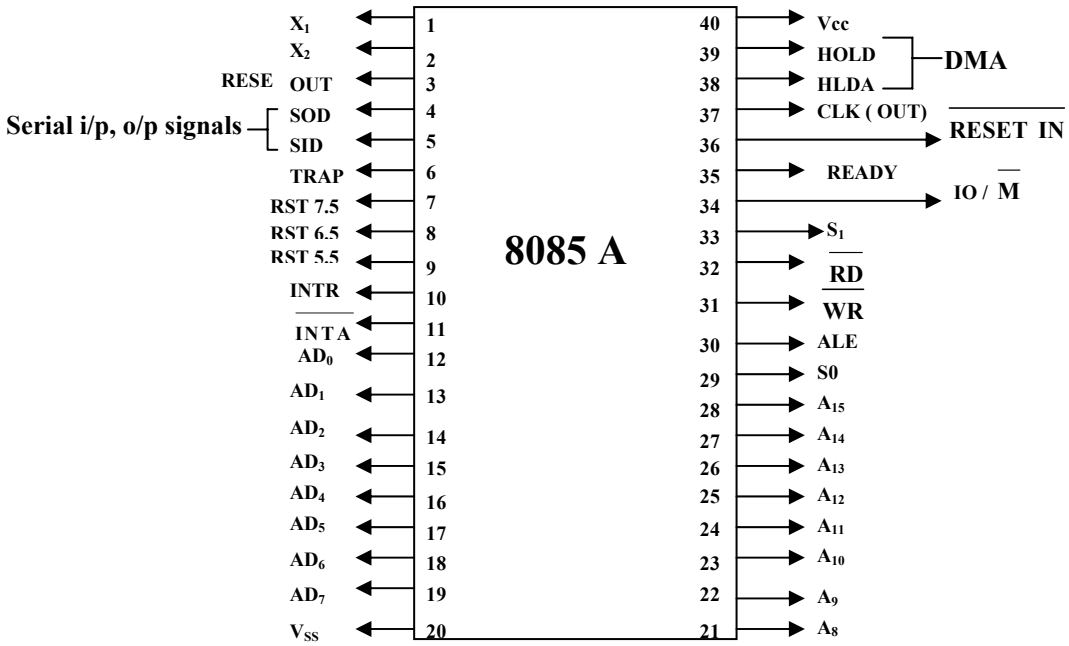
- Overview of 8085 microprocessor
- Overview of 8086 microprocessor
- Signals and pins of 8086 microprocessor

The salient features of 8085 μp are:

- It is a 8 bit microprocessor.
- It is manufactured with N-MOS technology.
- It has 16-bit address bus and hence can address up to 216 = 65536 bytes (64KB) memory locations through A_0-A_{15}.
- The first 8 lines of address bus and 8 lines of data bus are multiplexed $AD_0 – AD_7$.
- Data bus is a group of 8 lines $D_0 – D_7$.
- It supports external interrupt request.
- A 16 bit program counter (PC)
- A 16 bit stack pointer (SP)
- Six 8-bit general purpose register arranged in pairs: BC, DE, HL.
- It requires a signal +5V power supply and operates at 3.2 MHZ single phase clock.
- It is enclosed with 40 pins DIP (Dual in line package).

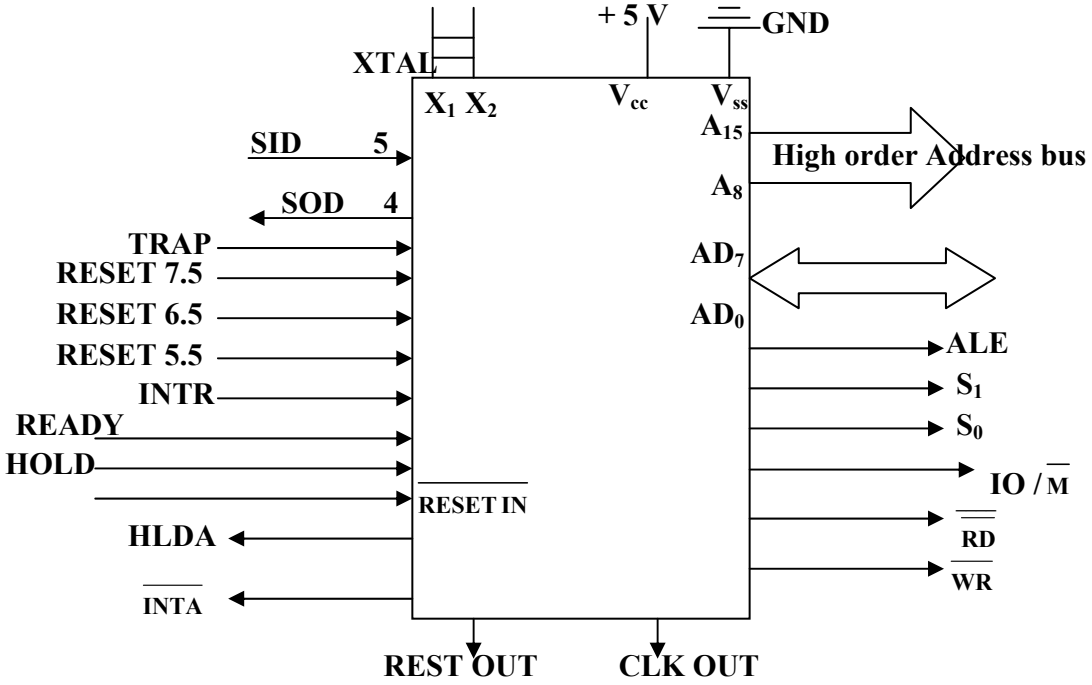
Overview of 8085 microprocessor

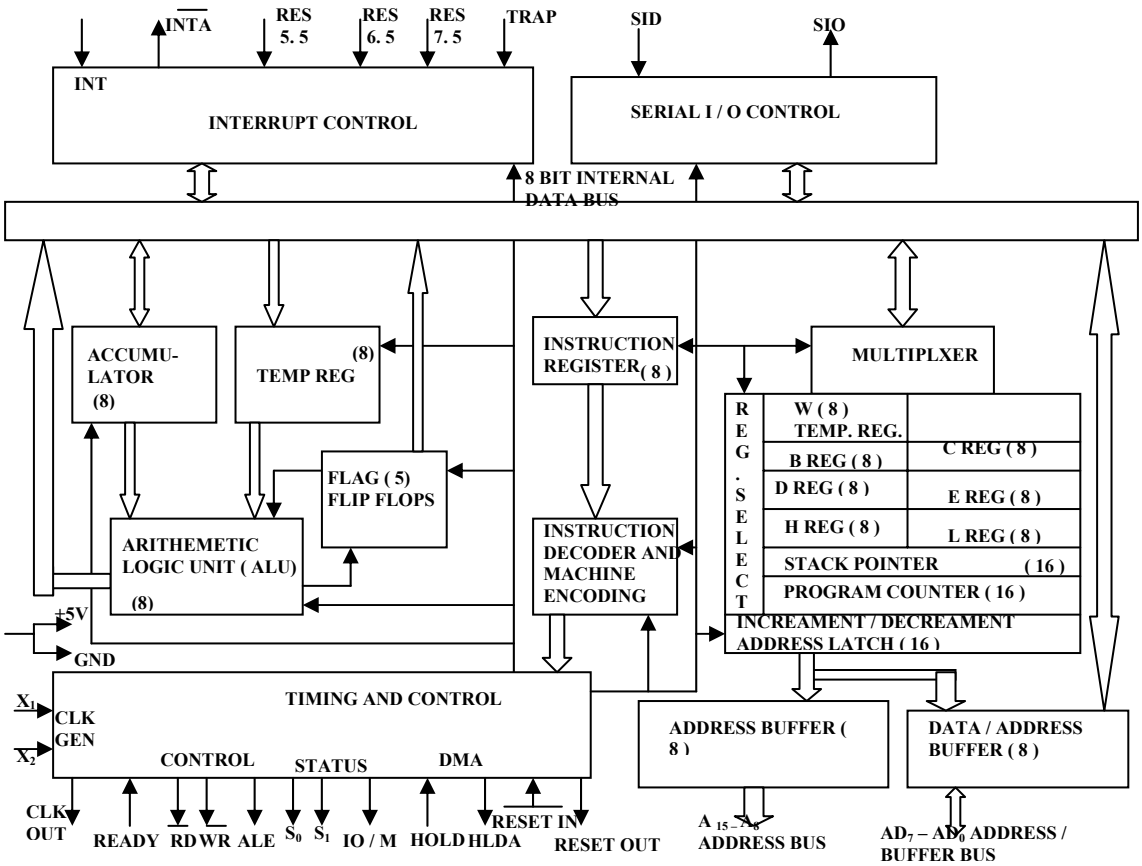
- 8085 Architecture
- Pin Diagram
- Functional Block Diagram



Pin Diagram of 8085

Signal Groups of 8085





Block Diagram

Flag Registers

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
S	Z		AC		P		CY

General Purpose Registers

INDIVIDUAL	B, C, D, E, H, L
COMBININATON	B & C, D & E, H & L

Memory

- Program, data and stack memories occupy the same memory space. The total addressable memory size is 64 KB.
- **Program memory** - program can be located anywhere in memory. Jump, branch and call instructions use 16-bit addresses, i.e. they can be used to jump/branch anywhere within 64 KB. All jump/branch instructions use absolute addressing.
- **Data memory** - the processor always uses 16-bit addresses so that data can be placed anywhere.
- **Stack memory** is limited only by the size of memory. Stack grows downward.
- First 64 bytes in a zero memory page should be reserved for vectors used by RST instructions.

Interrupts

- The processor has 5 interrupts. They are presented below in the order of their priority (from lowest to highest):
-
- **INTR** is maskable 8080A compatible interrupt. When the interrupt occurs the processor fetches from the bus one instruction, usually one of these instructions:
- One of the 8 RST instructions (RST_0 - RST_7). The processor saves current program counter into stack and branches to memory location N * 8 (where N is a 3-bit number from 0 to 7 supplied with the RST instruction).
- **CALL** instruction (3 byte instruction). The processor calls the subroutine, address of which is specified in the second and third bytes of the instruction.
- **RST5.5** is a maskable interrupt. When this interrupt is received the processor saves the contents of the PC register into stack and branches to 2CH (hexadecimal) address.
- **RST6.5** is a maskable interrupt. When this interrupt is received the processor saves the contents of the PC register into stack and branches to 34H (hexadecimal) address.
- **RST7.5** is a maskable interrupt. When this interrupt is received the processor saves the contents of the PC register into stack and branches to 3CH (hexadecimal) address.
- **TRAP** is a non-maskable interrupt. When this interrupt is received the processor saves the contents of the PC register into stack and branches to 24H (hexadecimal) address.
- All maskable interrupts can be enabled or disabled using EI and DI instructions. RST 5.5, RST6.5 and RST7.5 interrupts can be enabled or disabled individually using SIM instruction.

Reset Signals

- **RESET IN**: When this signal goes low, the program counter (PC) is set to Zero, µp is reset and resets the interrupt enable and HLDA flip-flops.
- The data and address buses and the control lines are 3-stated during RESET and because of asynchronous nature of RESET, the processor internal registers and flags may be altered by RESET with unpredictable results.
- RESET IN is a Schmitt-triggered input, allowing connection to an R-C network for power-on RESET delay.

- Upon power-up, RESET IN must remain low for at least 10 ms after minimum Vcc has been reached.
- For proper reset operation after the power – up duration, RESET IN should be kept low a minimum of three clock periods.
- The CPU is held in the reset condition as long as RESET IN is applied. Typical Power-on RESET RC values $R_1 = 75K\Omega$, $C_1 = 1\mu F$.
- **RESET OUT**: This signal indicates that µp is being reset. This signal can be used to reset other devices. The signal is synchronized to the processor clock and lasts an integral number of clock periods.

Serial communication Signal

- **SID - Serial Input Data Line**: The data on this line is loaded into accumulator bit 7 whenever a RIM instruction is executed.
- **SOD – Serial Output Data Line**: The SIM instruction loads the value of bit 7 of the accumulator into SOD latch if bit 6 (SOE) of the accumulator is 1.

DMA Signals

- **HOLD**: Indicates that another master is requesting the use of the address and data buses. The CPU, upon receiving the hold request, will relinquish the use of the bus as soon as the completion of the current bus transfer.
- Internal processing can continue. The processor can regain the bus only after the HOLD is removed.
- When the HOLD is acknowledged, the Address, Data RD, WR and IO/M lines are 3-stated.
- **HLDA: Hold Acknowledge**: Indicates that the CPU has received the HOLD request and that it will relinquish the bus in the next clock cycle.
- HLDA goes low after the Hold request is removed. The CPU takes the bus one half-clock cycle after HLDA goes low.
- **READY:** This signal Synchronizes the fast CPU and the slow memory, peripherals.
- If READY is high during a read or write cycle, it indicates that the memory or peripheral is ready to send or receive data.
- If READY is low, the CPU will wait an integral number of clock cycle for READY to go high before completing the read or write cycle.
- READY must conform to specified setup and hold times.

Registers

- **Accumulator** or A register is an 8-bit register used for arithmetic, logic, I/O and load/store operations.
- **Flag Register** has five 1-bit flags.
- **Sign** - set if the most significant bit of the result is set.
- **Zero** - set if the result is zero.
- **Auxiliary carry** - set if there was a carry out from bit 3 to bit 4 of the result.
- **Parity** - set if the parity (the number of set bits in the result) is even.
- **Carry** - set if there was a carry during addition, or borrow during subtraction/comparison/rotation.

General Registers

- 8-bit B and 8-bit C registers can be used as one 16-bit BC register pair. When used as a pair the C register contains low-order byte. Some instructions may use BC register as a data pointer.
- 8-bit D and 8-bit E registers can be used as one 16-bit DE register pair. When used as a pair the E register contains low-order byte. Some instructions may use DE register as a data pointer.
- 8-bit H and 8-bit L registers can be used as one 16-bit HL register pair. When used as a pair the L register contains low-order byte. HL register usually contains a data pointer used to reference memory addresses.
- **Stack pointer** is a 16 bit register. This register is always decremented/incremented by 2 during push and pop.
- **Program counter** is a 16-bit register.

Instruction Set

- 8085 instruction set consists of the following instructions:
- Data moving instructions.
- Arithmetic - add, subtract, increment and decrement.
- Logic - AND, OR, XOR and rotate.
- Control transfer - conditional, unconditional, call subroutine, return from subroutine and restarts.
- Input/Output instructions.
- Other - setting/clearing flag bits, enabling/disabling interrupts, stack operations, etc.

Addressing mode

- **Register** - references the data in a register or in a register pair.
  Register indirect - instruction specifies register pair containing address, where the data is located.
  Direct, Immediate - 8 or 16-bit data.

Module 1: learning unit 3

8086 Microprocessor

•It is a 16-bit µp.

•8086 has a 20 bit address bus can access up to 2^{20} memory locations (1 MB).

•It can support up to 64K I/O ports.

•It provides 14, 16 -bit registers.

•It has multiplexed address and data bus AD_0- AD_{15} and $A_{16} – A_{19}$.

•It requires single phase clock with 33% duty cycle to provide internal timing.

•8086 is designed to operate in two modes, Minimum and Maximum.

•It can prefetches upto 6 instruction bytes from memory and queues them in order to speed up instruction execution.

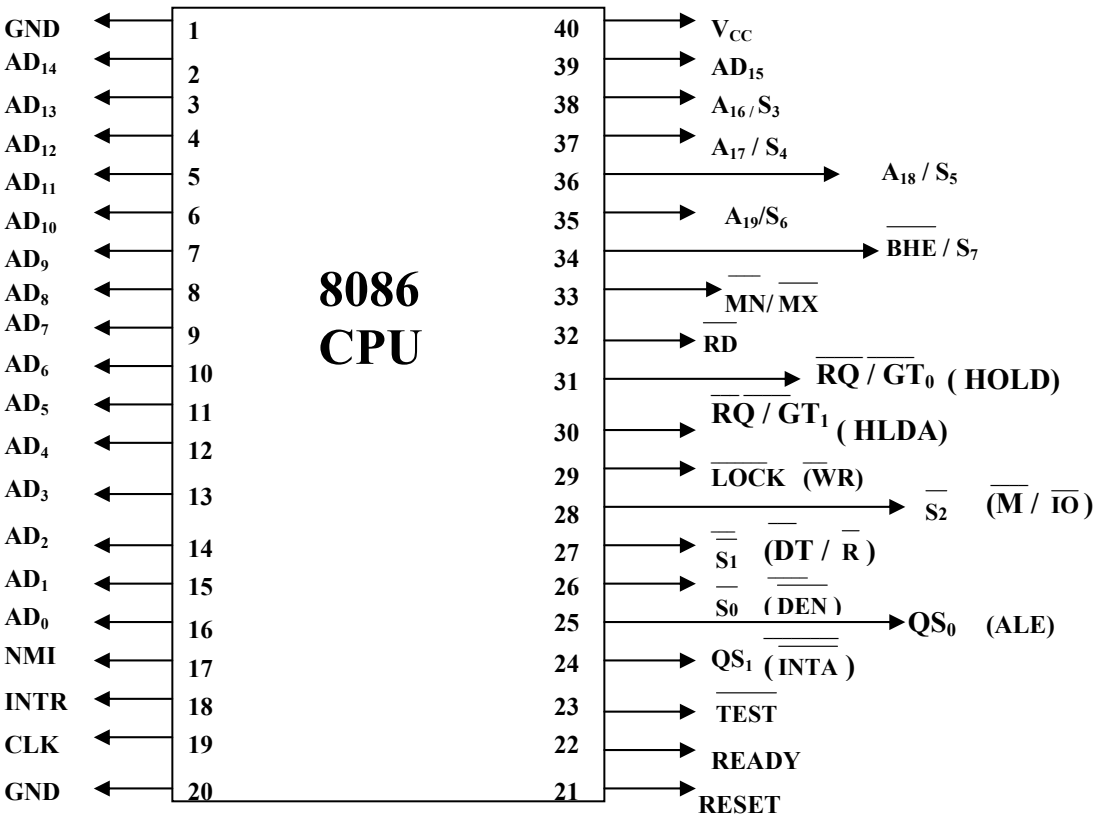
•It requires +5V power supply.

•A 40 pin dual in line package

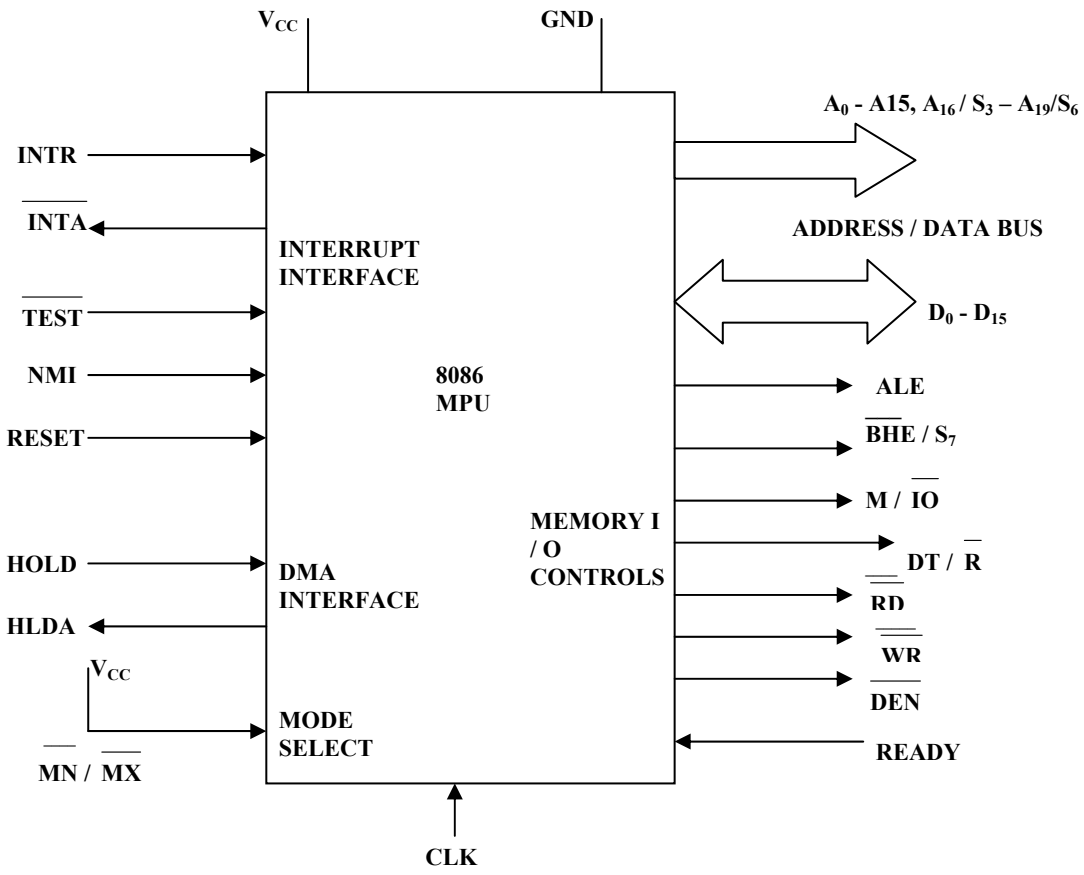
Minimum and Maximum Modes:

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

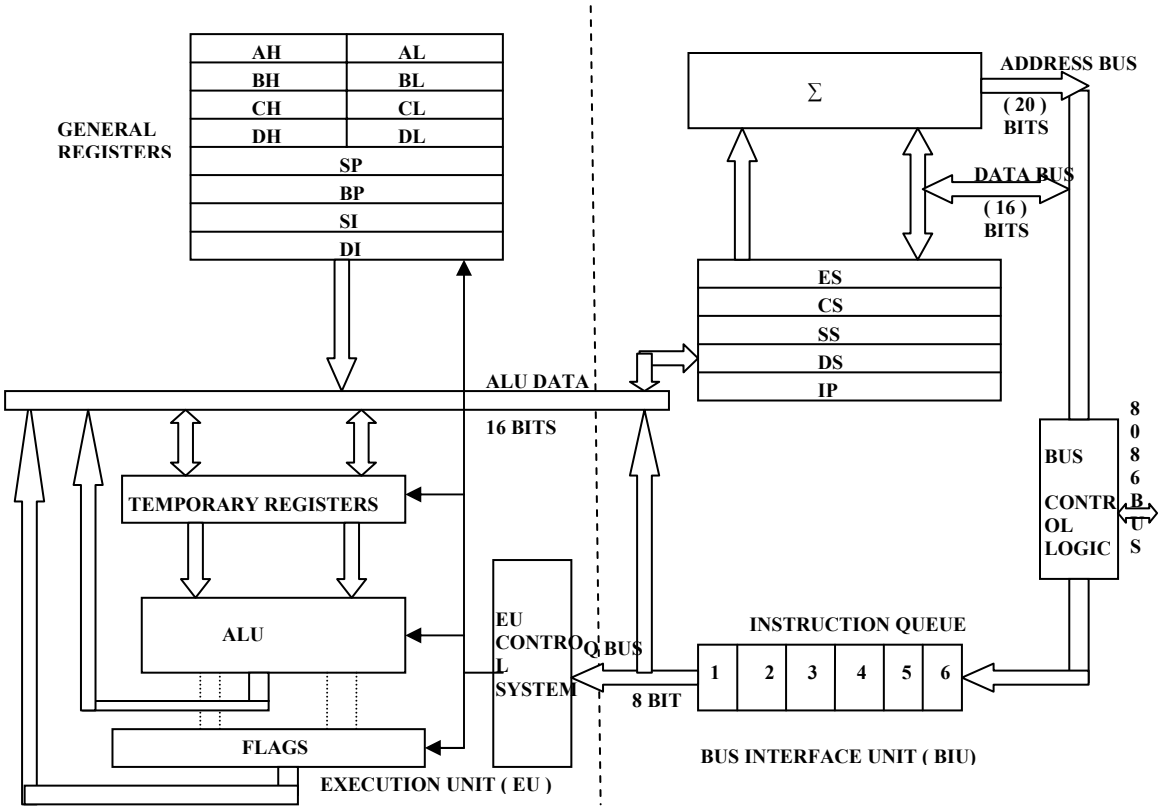
•The maximum mode is selected by applying logic 0 to the MN / $\overline{MX}$ input pin. This is a multi micro processors configuration.



Pin Diagram of 8086



Signal Groups of 8086



Block Diagram of 8086

Internal Architecture of 8086

•8086 has two blocks BIU and EU.

•The BIU performs all bus operations such as instruction fetching, reading and writing operands for memory and calculating the addresses of the memory operands. The instruction bytes are transferred to the instruction queue.

•EU executes instructions from the instruction system byte queue.

•Both units operate asynchronously to give the 8086 an overlapping instruction fetch and execution mechanism which is called as Pipelining. This results in efficient use of the system bus and system performance.

•BIU contains Instruction queue, Segment registers, Instruction pointer, Address adder.

•EU contains Control circuitry, Instruction decoder, ALU, Pointer and Index register, Flag register.

BUS INTERFACR UNIT:

•It provides a full 16 bit bidirectional data bus and 20 bit address bus.

•The bus interface unit is responsible for performing all external bus operations.

Specifically it has the following functions:

•Instruction fetch, Instruction queuing, Operand fetch and storage, Address relocation and Bus control.

•The BIU uses a mechanism known as an instruction stream queue to implement a ***pipeline architecture.***

•This queue permits prefetch of up to six bytes of instruction code. When ever the queue of the BIU is not full, it has room for at least two more bytes and at the same time the EU

is not requesting it to read or write operands from memory, the BIU is free to look ahead in the program by prefetching the next sequential instruction.
•These prefetching instructions are held in its FIFO queue. With its 16 bit data bus, the BIU fetches two instruction bytes in a single memory cycle.
•After a byte is loaded at the input end of the queue, it automatically shifts up through the FIFO to the empty location nearest the output.
•The EU accesses the queue from the output end. It reads one instruction byte after the other from the output of the queue. If the queue is full and the EU is not requesting access to operand in memory.
•These intervals of no bus activity, which may occur between bus cycles are known as ***Idle state.***
•If the BIU is already in the process of fetching an instruction when the EU request it to read or write operands from memory or I/O, the BIU first completes the instruction fetch bus cycle before initiating the operand read / write cycle.
•The BIU also contains a dedicated adder which is used to generate the 20bit physical address that is output on the address bus. This address is formed by adding an appended 16 bit segment address and a 16 bit offset address.
•For example**:** The physical address of the next instruction to be fetched is formed by combining the current contents of the code segment CS register and the current contents of the instruction pointer IP register.
•The BIU is also responsible for generating bus control signals such as those for memory read or write and I/O read or write.

EXECUTION UNIT

The Execution unit is responsible for decoding and executing all instructions.
•The EU extracts instructions from the top of the queue in the BIU, decodes them, generates operands if necessary, passes them to the BIU and requests it to perform the read or write bys cycles to memory or I/O and perform the operation specified by the instruction on the operands.
•During the execution of the instruction, the EU tests the status and control flags and updates them based on the results of executing the instruction.
•If the queue is empty, the EU waits for the next instruction byte to be fetched and shifted to top of the queue.
•When the EU executes a branch or jump instruction, it transfers control to a location corresponding to another set of sequential instructions.
•Whenever this happens, the BIU automatically resets the queue and then begins to fetch instructions from this new location to refill the queue.

Module 1 and learning unit 4:

Signal Description of 8086•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 8086 signals can be categorised 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 and third are the signals having special functions for maximum mode.

•**The following signal descriptions are common for both modes.**

•**AD_{15}-AD_0**: These are the time multiplexed memory I/O address and data lines.

• Address remains on the lines during T_1 state, while the data is available on the data bus during T_2, T_3, T_W and T_4.

•These lines are active high and float to a tristate during interrupt acknowledge and local bus hold acknowledge cycles.

•**A_{19}/S_6,A_{18}/S_5,A_{17}/S_4,A_{16}/S_3:** These are the time multiplexed address and status lines.

•During T_1 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 T_2,T_3,T_W and T_4.

•The status of the interrupt enable flag bit is updated at the beginning of each clock cycle.

•The S_4 and S_3 combinedly indicate which segment register is presently being used for memory accesses as in below fig.

•These lines float to tri-state off during the local bus hold acknowledge. The status line S_6 is always low.

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

S_4	S_3	Indication
0	0	Alternate Data
0	1	Stack
1	0	Code or none
1	1	Data

• $\overline{\text{BHE}}$ **/S7:** The bus high enable is used to indicate the transfer of data over the higher order (D_{15}-D_8) data bus as shown in table. It goes low for the data transfer over D_{15}-D_8 and is used to derive chip selects of odd address memory bank or peripherals. BHE is low during T_1 for read, write and interrupt acknowledge cycles, whenever a byte is to be transferred on higher byte of data bus. The status information is available during T_2, T_3 and T_4. The signal is active low and tristated during hold. It is low during T_1 for the first pulse of the interrupt acknowledges cycle.

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

• $\overline{\text{RD}}$ **Read:** This signal on low indicates the peripheral that the processor is performing s memory or I/O read operation. RD is active low and shows the state for T_2, T_3, T_W 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. The signal made available by the devices is synchronized by the 8284A clock generator to provide ready input to the 8086. the signal is active high.
•**INTR-Interrupt Request**: This is a triggered input. This is sampled during the last clock cycles of each instruction to determine the availability of the request. If any interrupt request is pending, the processor enters the interrupt acknowledge cycle.
•This can be internally masked by resulting the interrupt enable flag. This signal is active high and internally synchronized.
• $\overline{\textbf{TEST}}$ 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. The input is synchronized internally during each clock cycle on leading edge of clock.
•**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.
•**MN/** $\overline{\textbf{MX}}$: 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/** $\overline{\textbf{IO}}$ **– Memory/IO**: This is a status line logically equivalent to S_2 in maximum mode. When it is low, it indicates the CPU is having an I/O operation, and when it is high, it indicates that the CPU is having a memory operation. This line becomes active high in the previous T_4 and remains active till final T_4 of the current cycle. It is tristated during local bus "hold acknowledge ".
• $\overline{\textbf{INTA}}$ **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/** $\overline{\textbf{R}}$ **– 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. It is active from the middle of T_2 until the middle of T_4. This is tristated during ' hold acknowledge' cycle.
•**HOLD, HLDA- Acknowledge**: 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.•At the same time, the processor floats the local bus and control lines. When the processor detects the HOLD line low, it lowers the HLDA signal. HOLD is an asynchronous input, and is should be externally synchronized.
•If the DMA request is made while the CPU is performing a memory or I/O cycle, it will release the local bus during T_4 provided:
1.The request occurs on or before T_2 state of the current cycle.
2.The current cycle is not operating over the lower byte of a word.
3.The current cycle is not the first acknowledge of an interrupt acknowledge sequence.

4. A Lock instruction is not being executed.

•***The following pin function are applicable for maximum mode operation of 8086***.

•**S_2, S_1, S_0 – Status Lines**: These are the status lines which reflect the type of operation, being carried out by the processor. These become activity during T_4 of the previous cycle and active during T_1 and T_2 of the current bus cycles.

S_2	S_1	S_0	Indication
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	Code Access
1	0	1	Read memory
1	1	0	Write memory
1	1	1	Passive

• $\overline{\text{LOCK}}$ This output pin indicates that other system bus master will be prevented from gaining the system bus, while the LOCK signal is low.

•The LOCK signal is activated by the 'LOCK' prefix instruction and remains active until the completion of the next instruction. When the CPU is executing a critical instruction which requires the system bus, the LOCK prefix instruction ensures that other processors connected in the system will not gain the control of the bus.

•The 8086, while executing the prefixed instruction, asserts the bus lock signal output, which may be connected to an external bus controller.

•**QS_1, QS_0 – Queue Status:** These lines give information about the status of the code-prefetch queue. These are active during the CLK cycle after while the queue operation is performed.

•This modification in a simple fetch and execute architecture of a conventional microprocessor offers an added advantage of pipelined processing of the instructions.

•The 8086 architecture has 6-byte instruction prefetch queue. Thus even the largest (6-bytes) instruction can be prefetched from the memory and stored in the prefetch. This results in a faster execution of the instructions.

•In 8085 an instruction is fetched, decoded and executed and only after the execution of this instruction, the next one is fetched.

•By prefetching the instruction, there is a considerable speeding up in instruction execution in 8086. This is known as ***instruction pipelining***.

•At the starting the CS:IP is loaded with the required address from which the execution is to be started. Initially, the queue will be empty an the microprocessor starts a fetch operation to bring one byte (the first byte) of instruction code, if the CS:IP address is odd or two bytes at a time, if the CS:IP address is even.

•The first byte is a complete opcode in case of some instruction (one byte opcode instruction) and is a part of opcode, in case of some instructions (two byte opcode instructions), the remaining part of code lie in second byte.

•The second byte is then decoded in continuation with the first byte to decide the instruction length and the number of subsequent bytes to be treated as instruction data.

•The queue is updated after every byte is read from the queue but the fetch cycle is initiated by BIU only if at least two bytes of the queue are empty and the EU may be concurrently executing the fetched instructions.
•The next byte after the instruction is completed is again the first opcode byte of the next instruction. A similar procedure is repeated till the complete execution of the program.•The fetch operation of the next instruction is overlapped with the execution of the current instruction. As in the architecture, there are two separate units, namely Execution unit and Bus interface unit.
•While the execution unit is busy in executing an instruction, after it is completely decoded, the bus interface unit may be fetching the bytes of the next instruction from memory, depending upon the queue status.

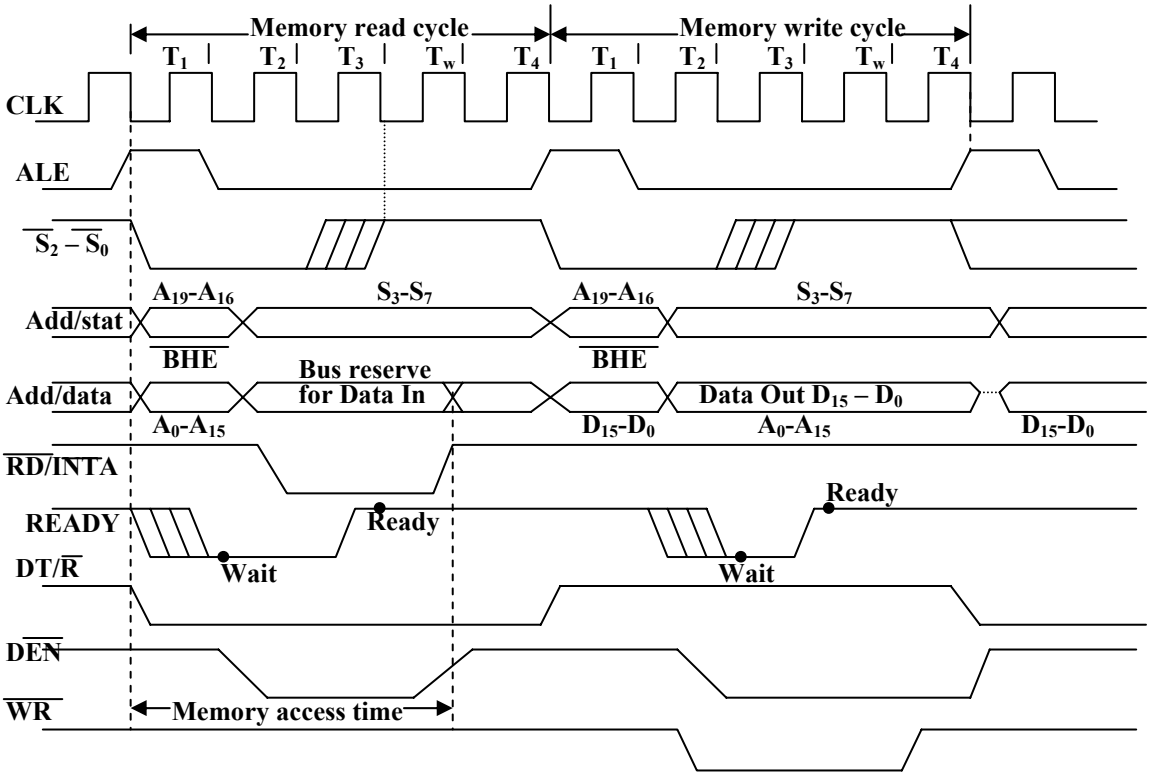
QS_1	QS_0	Indication
0	0	No operation
0	1	First byte of the opcode from the queue
1	0	Empty queue
1	1	Subsequent byte from the queue

• $\overline{RQ}/\overline{GT_0}$, $\overline{RQ}/\overline{GT_1}$ – **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/GT_0 having higher priority than RQ/GT_1.
•RQ/GT pins have internal pull-up resistors and may be left unconnected.
•**Request/Grant sequence is as follows:**
1.A pulse of one clock wide from another bus master requests the bus access to 8086.
2.During T_4(current) or T_1(next) clock cycle, a pulse one clock wide from 8086 to the requesting master, indicates that the 8086 has allowed the local bus to float and that it will enter the 'hold acknowledge' state at next cycle. The CPU bus interface unit is likely to be disconnected from the local bus of the system.
3.A one clock wide pulse from the another master indicates to the 8086 that the hold request is about to end and the 8086 may regain control of the local bus at the next clock cycle. Thus each master to master exchange of the local bus is a sequence of 3 pulses. There must be at least one dead clock cycle after each bus exchange.
•The request and grant pulses are active low.
•For the bus request those are received while 8086 is performing memory or I/O cycle, the granting of the bus is governed by the rules as in case of HOLD and HLDA in minimum mode.

General Bus Operation:

•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 utilisation 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 T_1, T_2, T_3, T_4. The address is transmitted by the processor during T_1. It is present on the bus only for one cycle.
•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 S_0, S_1 and S_2 are used to indicate the type of operation.
•Status bits S_3 to S_7 are multiplexed with higher order address bits and the BHE signal. Address is valid during T_1 while status bits S_3 to S_7 are valid during T_2 through T_4.

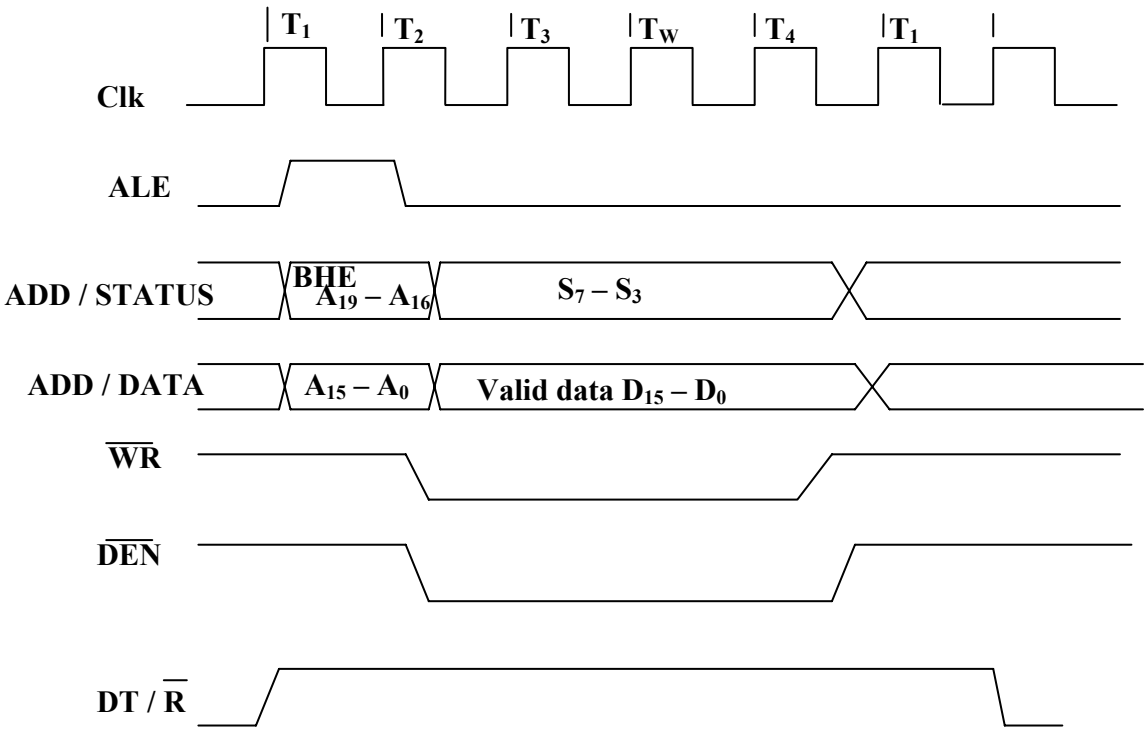


General Bus Operation Cycle in Maximum Mode

Minimum Mode 8086 System

•In a minimum mode 8086 system, the microprocessor 8086 is operated in minimum mode by strapping its MN/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.

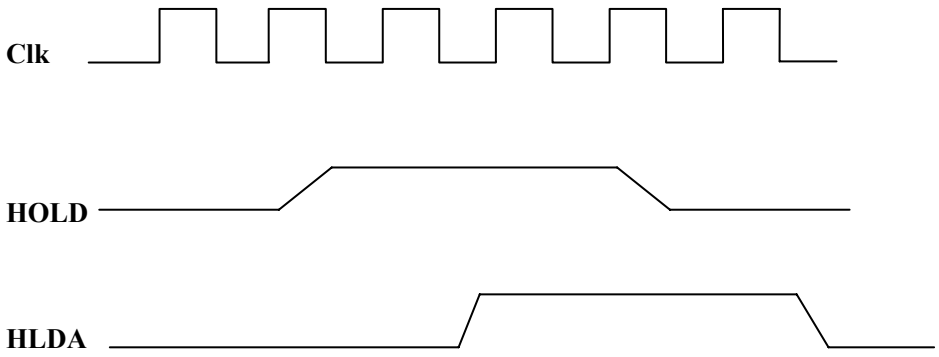
•Transreceivers are the bidirectional buffers and some times they are called as data amplifiers. They are required to separate the valid data from the time multiplexed address/data signals.
•They are controlled by two signals namely, DEN and DT/R.
•The DEN signal indicates the direction of data, i.e. from or to the processor. The system contains memory for the monitor and users program storage.
•Usually, EPROM are used for monitor storage, while RAM for users 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.
•The read cycle begins in T_1 with the assertion of address latch enable (ALE) signal and also M / IO signal. During the negative going edge of this signal, the valid address is latched on the local bus.
•The BHE and A_0 signals address low, high or both bytes. From T_1 to T_4 , the M/IO signal indicates a memory or I/O operation.
•At T_2, the address is removed from the local bus and is sent to the output. The bus is then tristated. The read (RD) control signal is also activated in T_2.
•The read (RD) signal causes the address device to enable its data bus drivers. After 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.
•A write cycle also begins with the assertion of ALE and the emission of the address. The M/IO signal is again asserted to indicate a memory or I/O operation. In T_2, after sending the address in T_1, the processor sends the data to be written to the addressed location.
•The data remains on the bus until middle of T_4 state. The WR becomes active at the beginning of T_2 (unlike RD is somewhat delayed in T_2 to provide time for floating).
•The BHE and A_0 signals are used to select the proper byte or bytes of memory or I/O word to be read or write.
•The M/IO, RD and WR signals indicate the type of data transfer as specified in table below.



Write Cycle Timing Diagram for Minimum Mode

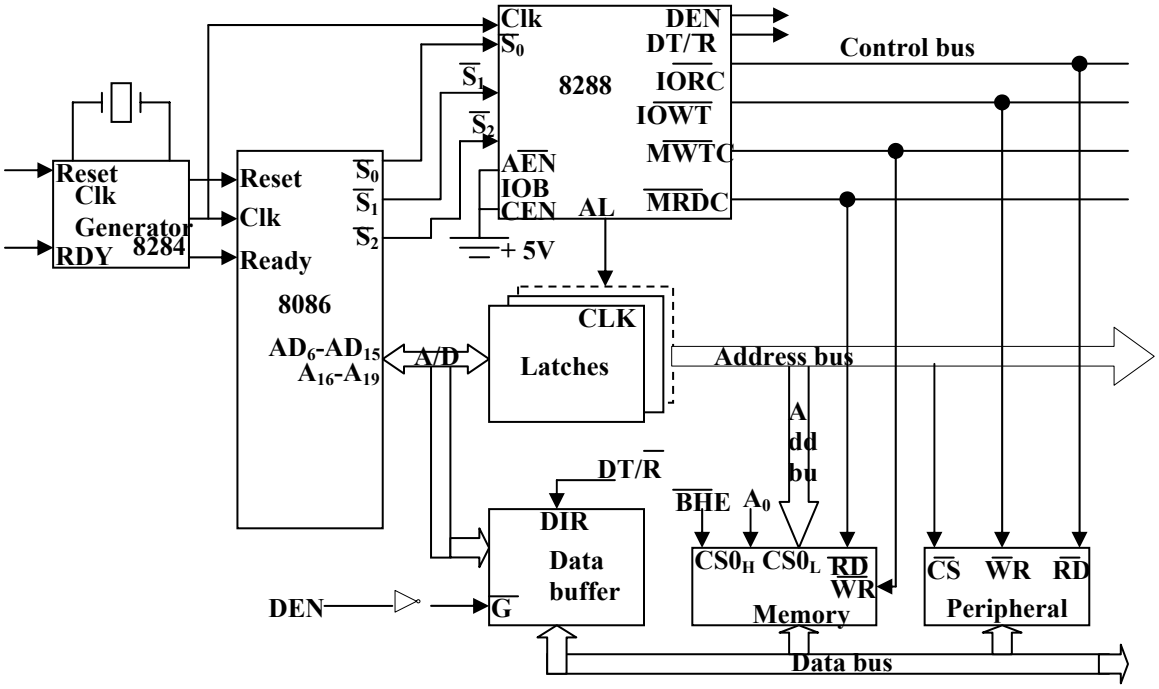
•***Hold Response sequence***: The HOLD pin is checked at leading edge of each clock pulse. If it is received active by the processor before T_4 of the previous cycle or during T_1 state of the current cycle, the CPU activates HLDA in the next clock cycle and for succeeding bus cycles, the bus will be given to another requesting master.

•The control of the bus is not regained by the processor until the requesting master does not drop the HOLD pin low. When the request is dropped by the requesting master, the HLDA is dropped by the processor at the trailing edge of the next clock.



Bus Request and Bus Grant Timings in Minimum Mode System

Maximum Mode 8086 System •In the maximum mode, the 8086 is operated by strapping the MN/MX pin to ground.
•In this mode, the processor derives the status signal S_2, S_1, S_0. 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 RD and WR (for memory and I/O devices), DEN, DT/R, ALE etc. using the information by the processor on the status lines.
•The bus controller chip has input lines S_2, S_1, S_0 and CLK. These inputs to 8288 are driven by CPU.
•It derives the outputs ALE, DEN, DT/R, MRDC, MWTC, AMWC, IORC, IOWC and 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.
•INTA pin used to issue two interrupt acknowledge pulses to the interrupt controller or to an interrupting device.
•IORC, 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 MRDC, 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 AIOWC and 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.



Maximum Mode 8086 System.

•R_0, S_1, S_2 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 T_1.

•In T_2, 8288 will set DEN=1 thus enabling transceivers, and for an input it will activate MRDC or IORC. These signals are activated until T_4. For an output, the AMWC or AIOWC is activated from T_2 to T_4 and MWTC or IOWC is activated from T_3 to T_4.

•The status bit S_0 to S_2 remains active until T_3 and become passive during T_3 and T_4.

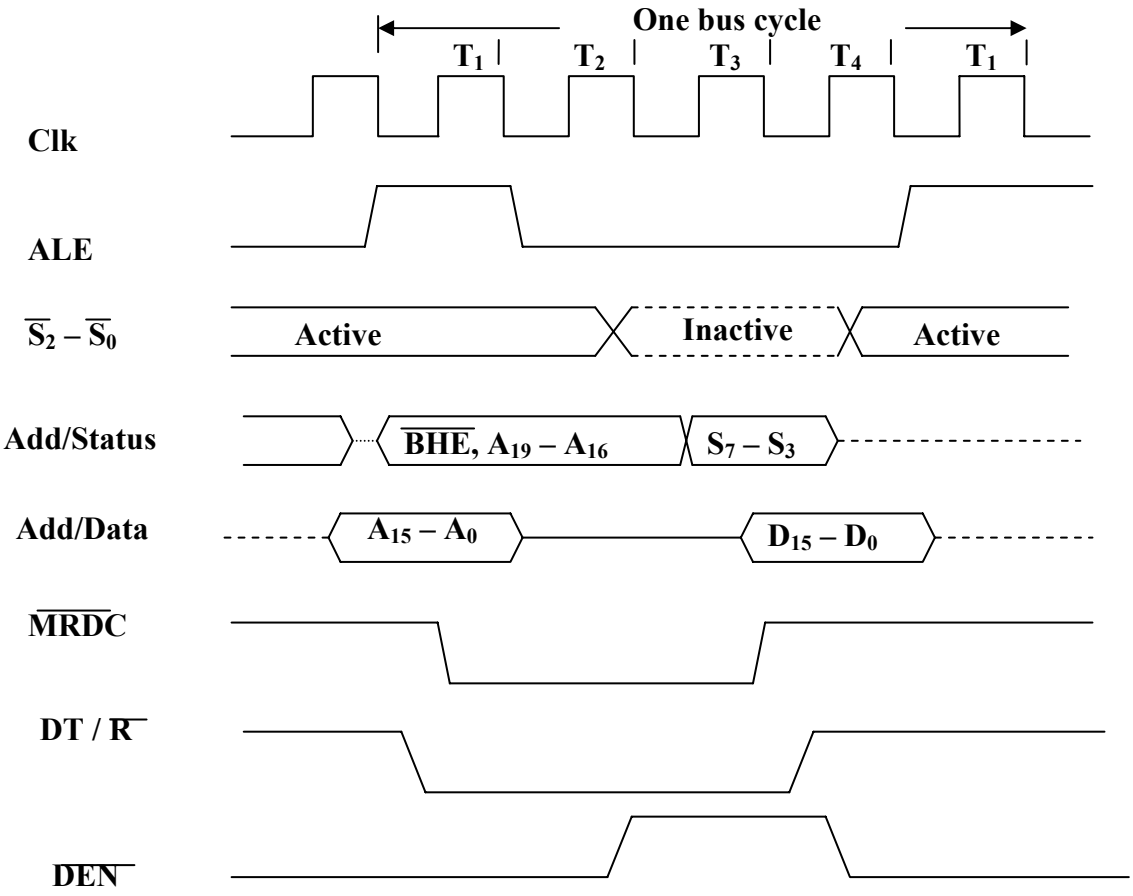
•If reader input is not activated before T_3, wait state will be inserted between T_3 and T_4.

•**Timings for RQ/ GT Signals:**

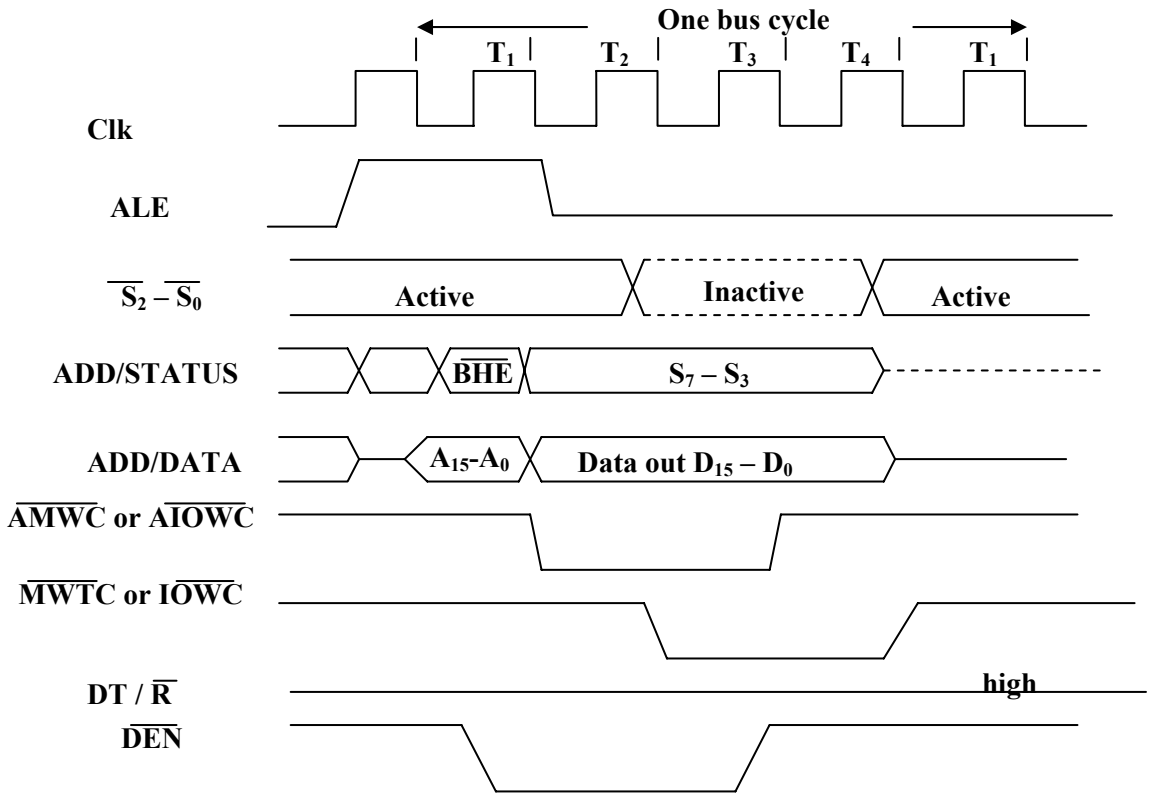
The request/grant response sequence contains a series of three pulses. The request/grant pins are checked at each rising pulse of clock input.

•When a request is detected and if the condition for HOLD request are satisfied, the processor issues a grant pulse over the RQ/GT pin immediately during T_4 (current) or T_1 (next) state.

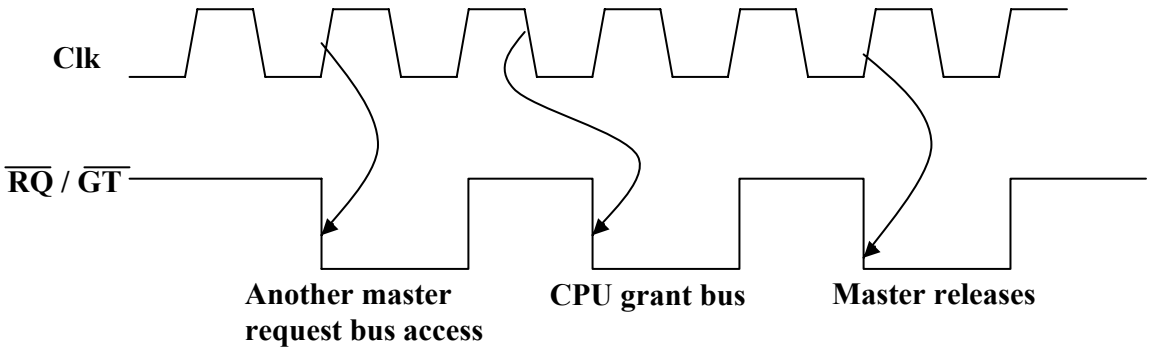
•When the requesting master receives this pulse, it accepts the control of the bus, it sends a release pulse to the processor using RQ/GT pin.



Memory Read Timing in Maximum Mode



Memory Write Timing in Maximum mode.



RQ/GT Timings in Maximum Mode.

Minimum Mode Interface

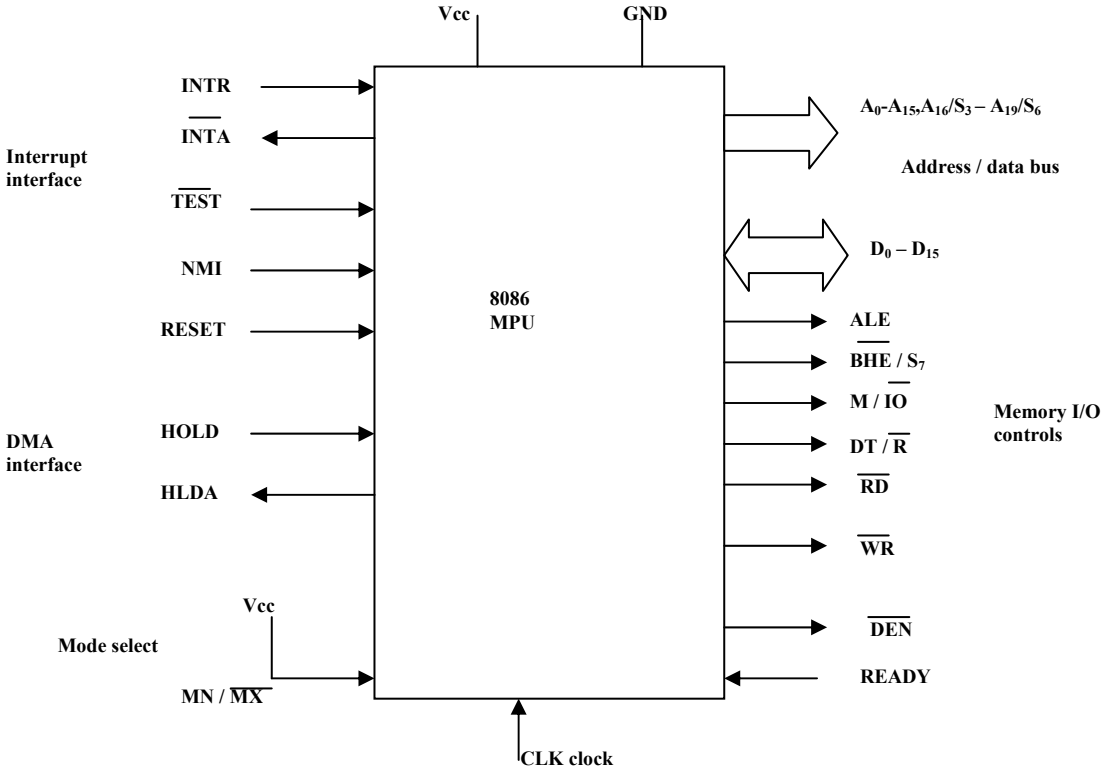
•When the Minimum mode operation is selected, the 8086 provides all control signals needed to implement the memory and I/O interface.

•The minimum mode signal can be divided into the following basic groups: address/data bus, status, control, interrupt and DMA.

•**Address/Data Bus**: these lines serve two functions. As an address bus is 20 bits long and consists of signal lines A_0 through A_{19}. A_{19} represents the MSB and A_0 LSB. A 20bit address gives the 8086 a 1Mbyte memory address space. More over it has an independent I/O address space which is 64K bytes in length.

•The 16 data bus lines D_0 through D_{15} are actually multiplexed with address lines A_0 through A_{15} 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. D_{15} is the MSB and D_0 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.



Block Diagram of the Minimum Mode 8086 MPU

•**Status signal:**

The four most significant address lines A_{19} through A_{16} are also multiplexed but in this case with status signals S_6 through S_3. These status bits are output on the bus at the same time that data are transferred over the other bus lines.

•Bit S_4 and S_3 together from a 2 bit binary code that identifies which of the 8086 internal segment registers are used to generate the physical address that was output on the address bus during the current bus cycle.

•Code $S_4S_3 = 00$ identifies a register known as ***extra segment register*** as the source of the segment address.

•Status line S_5 reflects the status of another internal characteristic of the 8086. It is the logic level of the internal enable flag. The last status bit S_6 is always at the logic 0 level.

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

Memory segment status codes.

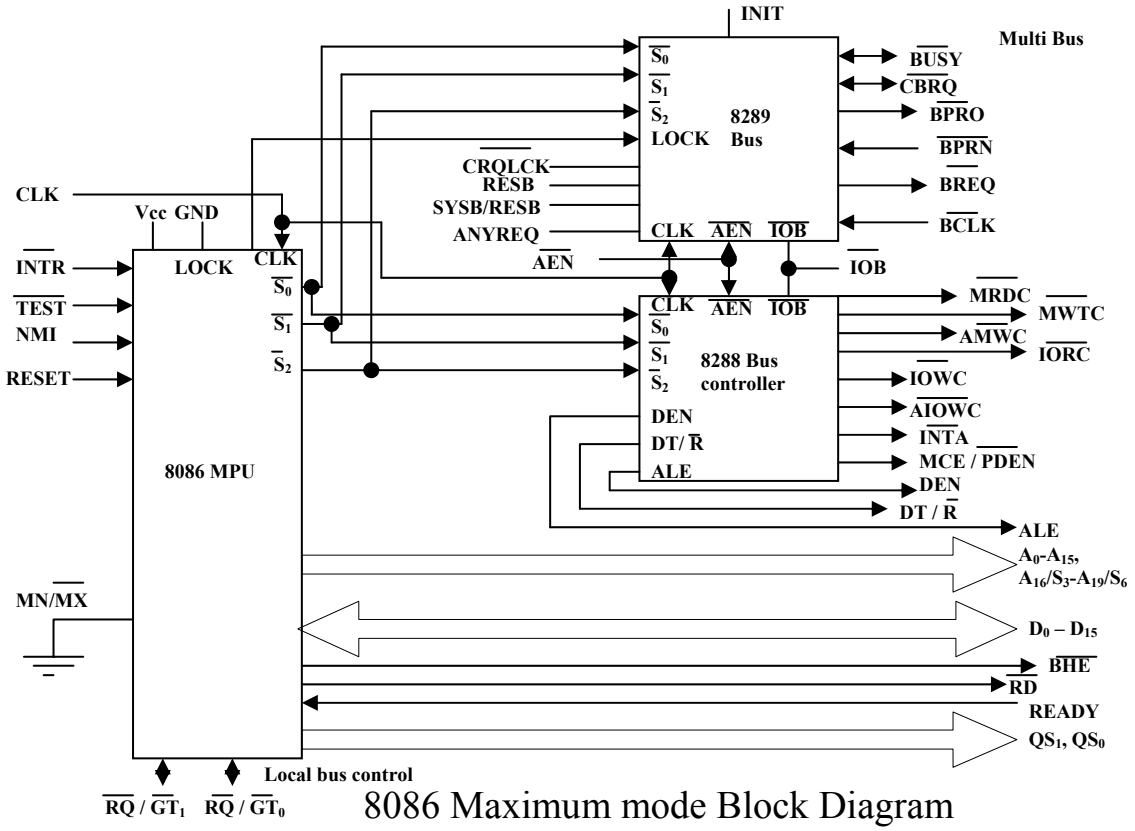
•**Control Signals**:
The control signals are provided to support the 8086 memory I/O interfaces. They control functions such as when the bus is to carry a valid address in which direction data are to be transferred over the bus, when valid write data are on the bus and when to put read data on the system bus.
•ALE is a pulse to logic 1 that signals external circuitry when a valid address word is on the bus. This address must be latched in external circuitry on the 1-to-0 edge of the pulse at ALE.
•Another control signal that is produced during the bus cycle is BHE bank high enable. Logic 0 on this used as a memory enable signal for the most significant byte half of the data bus D_8 through D_1. These lines also serves a second function, which is as the S_7 status line.
•Using the M/IO and DT/R lines, the 8086 signals which type of bus cycle is in progress and in which direction data are to be transferred over the bus.
•The logic level of M/IO tells external circuitry whether a memory or I/O transfer is taking place over the bus. Logic 1 at this output signals a memory operation and logic 0 an I/O operation.
•The direction of data transfer over the bus is signaled by the logic level output at DT/R. When this line is logic 1 during the data transfer part of a bus cycle, the bus is in the transmit mode. Therefore, data are either written into memory or output to an I/O device.

•On the other hand, logic 0 at DT/R signals that the bus is in the receive mode. This corresponds to reading data from memory or input of data from an input port.
•The signal read RD and write WR indicates that a read bus cycle or a write bus cycle is in progress. The 8086 switches WR to logic 0 to signal external device that valid write or output data are on the bus.
• On the other hand, RD indicates that the 8086 is performing a read of data of the bus. During read operations, one other control signal is also supplied. This is DEN (data enable) and it signals external devices when they should put data on the bus.
•There is one other control signal that is involved with the memory and I/O interface. This is the READY signal.
•READY signal is used to insert wait states into the bus cycle such that it is extended by a number of clock periods. This signal is provided by an external clock generator device and can be supplied by the memory or I/O sub-system to signal the 8086 when they are ready to permit the data transfer to be completed.
•**Interrupt signals**: The key interrupt interface signals are interrupt request (INTR) and interrupt acknowledge(INTA).
•INTR is an input to the 8086 that can be used by an external device to signal that it need 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 INTA output.
•The TEST input is also related to the external interrupt interface. Execution of a WAIT instruction causes the 8086 to check the logic level at the TEST input.
•If the logic 1 is found, the MPU suspend operation and goes into the idle state. The 8086 no longer executes instructions, instead it repeatedly checks the logic level of the TEST input waiting for its transition back to logic 0.
•As TEST switches to 0, execution resume with the next instruction in the program. This feature can be used to synchronize the operation of the 8086 to an event in external hardware.
•There are two more inputs in the interrupt interface: the nonmaskable interrupt NMI and the reset interrupt RESET.
•On the 0-to-1 transition of NMI control is passed to a nonmaskable interrupt service routine. The RESET input is used to provide a hardware reset for the 8086. Switching RESET to logic 0 initializes the internal register of the 8086 and initiates a reset service routine.
•**DMA Interface signals**:The direct memory access DMA interface of the 8086 minimum mode consist of the HOLD and HLDA signals.
•When an external device wants to take control of the system bus, it signals to the 8086 by switching HOLD to the logic 1 level. At the completion of the current bus cycle, the 8086 enters the hold state. In the hold state, signal lines AD_0 through AD_{15}, A_{16}/S_3 through A_{19}/S_6, BHE, M/IO, DT/R, RD, WR, DEN and INTR are all in the high Z state. The 8086 signals external device that it is in this state by switching its HLDA output to logic 1 level.

Maximum Mode Interface

•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.



8086 Maximum mode Block Diagram

•**8288 Bus Controller – Bus Command and Control Signals**:
8086 does not directly provide all the signals that are required to control the memory, I/O and interrupt interfaces.
•Specially the WR, M/IO, DT/R, DEN, ALE and INTA, signals are no longer produced by the 8086. Instead it outputs three status signals S_0, S_1, S_2 prior to the initiation of each bus cycle. This 3- bit bus status code identifies which type of bus cycle is to follow.
•$S_2S_1S_0$ are input to the external bus controller device, the bus controller generates the appropriately timed command and control signals.

Status Inputs			CPU Cycles	8288 Command
$\overline{S_2}$	$\overline{S_1}$	$\overline{S_0}$		
0	0	0	Interrupt Acknowledge	$\overline{INTA}$
0	0	1	Read I/O Port	$\overline{IORC}$
0	1	0	Write I/O Port	$\overline{IOWC}$, $\overline{AIOWC}$
0	1	1	Halt	None
1	0	0	Instruction Fetch	$\overline{MRDC}$
1	0	1	Read Memory	$\overline{MRDC}$
1	1	0	Write Memory	$\overline{MWTC}$, $\overline{AMWC}$
1	1	1	Passive	None

Bus Status Codes

•The 8288 produces one or two of these eight command signals for each bus cycles. For instance, when the 8086 outputs the code $S_2S_1S_0$ equals 001, it indicates that an ***I/O read cycle*** is to be performed.

•In the code 111 is output by the 8086, it is signaling that no bus activity is to take place.

•The control outputs produced by the 8288 are DEN, DT/R and ALE. These 3 signals provide the same functions as those described for the minimum system mode. This set of bus commands and control signals is compatible with the Multibus and industry standard for interfacing microprocessor systems.

•***The output of 8289 are bus arbitration signals***:

Bus busy (BUSY), *common bus request* (CBRQ), *bus priority out* (BPRO), *bus priority in* (BPRN), *bus request* (BREQ) and *bus clock* (BCLK).

•They correspond to the bus exchange signals of the Multibus and are used to lock other processor off the system bus during the execution of an instruction by the 8086.

•In this way the processor can be assured of uninterrupted access to common system resources such as ***global memory.***

•**Queue Status Signals**: Two new signals that are produced by the 8086 in the maximum-mode system are queue status outputs QS_0 and QS_1. Together they form a 2-bit queue status code, QS_1QS_0.

•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.

Queue status codes

•**Local Bus Control Signal – Request / Grant Signals**: In a maximum mode configuration, the minimum mode HOLD, HLDA interface is also changed. These two are replaced by request/grant lines RQ/ GT_0 and RQ/ GT_1, respectively. They provide a prioritized bus access mechanism for accessing the local bus.

Internal Registers of 8086

•The 8086 has four groups of the user accessible internal registers. They are the instruction pointer, four data registers, four pointer and index register, four segment registers.

•The 8086 has a total of fourteen 16-bit registers including a 16 bit register called the ***status register***, with 9 of bits implemented for status and control flags.

•Most of the registers contain data/instruction offsets within 64 KB memory segment. There are four different 64 KB segments for instructions, stack, data and extra data. To specify where in 1 MB of processor memory these 4 segments are located the processor uses four segment registers:

•**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.

•**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.

•**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.

•**Accumulator** register consists of two 8-bit registers AL and AH, which can be combined together and used as a 16-bit register AX. AL in this case contains the low-order byte of the word, and AH contains the high-order byte. Accumulator can be used for I/O operations and string manipulation.

•**Base** register consists of two 8-bit registers BL and BH, which can be combined together and used as a 16-bit register BX. BL in this case contains the low-order byte of the word, and BH contains the high-order byte. BX register usually contains a data pointer used for based, based indexed or register indirect addressing.
•**Count** register consists of two 8-bit registers CL and CH, which can be combined together and used as a 16-bit register CX. When combined, CL register contains the low-order byte of the word, and CH contains the high-order byte. Count register can be used in Loop, shift/rotate instructions and as a counter in string manipulation,.
•**Data** register consists of two 8-bit registers DL and DH, which can be combined together and used as a 16-bit register DX. When combined, DL register contains the low-order byte of the word, and DH contains the high-order byte. Data register can be used as a port number in I/O operations. In integer 32-bit multiply and divide instruction the DX register contains high-order word of the initial or resulting number.
•**The following registers are both general and index registers:**
•**Stack Pointer** (SP) is a 16-bit register pointing to program stack.
•**Base Pointer** (BP) is a 16-bit register pointing to data in stack segment. BP register is usually used for based, based indexed or register indirect addressing.
•**Source Index** (SI) is a 16-bit register. SI is used for indexed, based indexed and register indirect addressing, as well as a source data address in string manipulation instructions.
•**Destination Index** (DI) is a 16-bit register. DI is used for indexed, based indexed and register indirect addressing, as well as a destination data address in string manipulation instructions.
Other registers:
•**Instruction Pointer** (IP) is a 16-bit register.
•**Flags** is a 16-bit register containing 9 one bit flags.
•**Overflow Flag** (OF) - set if the result is too large positive number, or is too small negative number to fit into destination operand.
•**Direction Flag** (DF) - if set then string manipulation instructions will auto-decrement index registers. If cleared then the index registers will be auto-incremented.
•**Interrupt-enable Flag** (IF) - setting this bit enables maskable interrupts.
•**Single-step Flag** (TF) - if set then single-step interrupt will occur after the next instruction.
•**Sign Flag** (SF) - set if the most significant bit of the result is set.
•**Zero Flag** (ZF) - set if the result is zero.
•**Auxiliary carry Flag** (AF) - set if there was a carry from or borrow to bits 0-3 in the AL register.
•**Parity Flag** (PF) - set if parity (the number of "1" bits) in the low-order byte of the result is even.
•**Carry Flag** (CF) - set if there was a carry from or borrow to the most significant bit during last result calculation.
Addressing Modes
•**Implied** - the data value/data address is implicitly associated with the instruction.
•**Register** - references the data in a register or in a register pair.
•**Immediate** - the data is provided in the instruction.
•**Direct** - the instruction operand specifies the memory address where data is located.

•**Register indirect** - instruction specifies a register containing an address, where data is located. This addressing mode works with SI, DI, BX and BP registers.
•**Based**:- 8-bit or 16-bit instruction operand is added to the contents of a base register (BX or BP), the resulting value is a pointer to location where data resides.
•**Indexed**:- 8-bit or 16-bit instruction operand is added to the contents of an index register (SI or DI), the resulting value is a pointer to location where data resides
•**Based Indexed**:- the contents of a base register (BX or BP) is added to the contents of an index register (SI or DI), the resulting value is a pointer to location where data resides.
•**Based Indexed with displacement**:- 8-bit or 16-bit instruction operand is added to the contents of a base register (BX or BP) and index register (SI or DI), the resulting value is a pointer to location where data resides.
Memory •Program, data and stack memories occupy the same memory space. As the most of the processor instructions use 16-bit pointers the processor can effectively address only 64 KB of memory.
•To access memory outside of 64 KB the CPU uses special segment registers to specify where the code, stack and data 64 KB segments are positioned within 1 MB of memory (see the "Registers" section below).
•16-bit pointers and data are stored as:
address: low-order byte
address+1: high-order byte
•**Program memory** - program can be located anywhere in memory. Jump and call instructions can be used for short jumps within currently selected 64 KB code segment, as well as for far jumps anywhere within 1 MB of memory.
•All conditional jump instructions can be used to jump within approximately +127 to -127 bytes from current instruction.
•**Data memory** - the processor can access data in any one out of 4 available segments, which limits the size of accessible memory to 256 KB (if all four segments point to different 64 KB blocks).
•Accessing data from the Data, Code, Stack or Extra segments can be usually done by prefixing instructions with the DS:, CS:, SS: or ES: (some registers and instructions by default may use the ES or SS segments instead of DS segment).
•Word data can be located at odd or even byte boundaries. The processor uses two memory accesses to read 16-bit word located at odd byte boundaries. Reading word data from even byte boundaries requires only one memory access.
•**Stack memory** can be placed anywhere in memory. The stack can be located at odd memory addresses, but it is not recommended for performance reasons (see "Data Memory" above).
Reserved locations:
•0000h - 03FFh are reserved for interrupt vectors. Each interrupt vector is a 32-bit pointer in format segment: offset.
•FFFF0h - FFFFFh - after RESET the processor always starts program execution at the FFFF0h address.
Interrupts
The processor has the following interrupts:

•**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 then the maskable interrupt.
•**Software interrupts** can be caused by:
•INT instruction - breakpoint interrupt. This is a type 3 interrupt.
•INT <interrupt number> instruction - any one interrupt from available 256 interrupts.
•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 (type 7).
•Software interrupt processing is the same as for the hardware interrupts.