MICROPROCESSORS & INTERFACING

By Vinayaka Kota
Asst. Prof., ECE
N.B.K.R.I.S.T

UNIT-1

Introduction to Microprocessor, Evaluation of Microprocessor, Types of Microprocessors, Architecture of 8085 Microprocessor, Pin Configuration, Instruction Cycle, Timing Diagrams, Stack and Subroutines.

UNIT-2

Instruction set of 8085 Microprocessor, Addressing Modes, Assembly Language Programs (8085) for Addition, Subtraction, Multiplication, Division etc., Interrupts of 8085, Memory and I/O Interfacing of 8085 Microprocessor.

UNIT-3

Architecture of 8086 Microprocessor, Instruction Set, Addressing Modes, Interrupt System, Minimum Mode and Maximum Mode Operations of 8086 and its Timing Diagrams, Assembler Directives, Assembly Language Programs (8086), Stages of Software Development.

UNIT-4

Data Transfer Schemes – Synchronous, Asynchronous, Interrupt Driven and DMA Type Schemes, Programmable Interrupt Controller (8259) and its Interfacing, Programmable DMA Controller (8257) and its Interfacing, Programmable Interval Timer (8253) and its Interfacing, Programmable Communication Interface (8251 USART) and its Interfacing.

UNIT-5

Memory Intefacing to 8086 – Interfacing Various Types of RAM and ROM Chips, PPI (8255) and its Interfacing, ADC and DAC Interfacing, Waveform Generation, Traffic Light Controller, Stepper Motor Control, Temperature Measurement and Control.

Text Books :

- Ram B, “Fundamentals of Microprocessors and Microcontrollers”, DhanpatRai Publications
- A.K. Ray and K.M. Bhurchandi, “Advanced Microprocessor and Peripherals”, TMH

UNIT-I

UNIT - I

Introduction to Microprocessor: Microprocessor is an electronic chip that functions as the central processing unit (CPU) of a computer. The microprocessor based systems with limited resources are called as microcomputers. Now-a-days microprocessors are found in almost all electronic machines and appliances in its different form. Some common devices using microprocessors are computer, printers, automobiles, washing machines, microwave ovens, mobile phones, fax machines, Xerox machines and advanced instruments like radar, satellites, flights etc.

Almost all microprocessors use the basic concept of "stored program execution". By this concept, programs are stored sequentially in memory locations. The microprocessor will fetch the instructions one after the other and execute them in its arithmetic and logic unit. So it is necessary for the user to know about the internal resources and features of the microprocessor. The programmers must also understand the instructions that a microprocessor can support. Every microprocessor will have its own associated set of instructions and this list is given by all the microprocessor manufacturers. Programs are written using mnemonics called the assembly level language and then they are converted into binary machine level language. This conversion can be done manually or using an application called assembler.

In general, the programs are written by the user for a microprocessor to work with real world data. These data are available in many forms and are from many sources. A microprocessor based system need a set of memory units, set of interfacing circuits for inputs and a set of interfacing circuits for outputs. All circuits put together along with microprocessor are called as microcomputer system. The physical components of the microcomputer system are in general called as hardware. The program which makes this hardware useful is called as software.

Origin of Microprocessor

The breakthrough in transistor technology led to the introduction of minicomputers of the 1960s and the personal computer revolution of the 1970s.Microprocessors evolution is categorized into five generations i.e.first, second, third, fourth, and fifth generations.

First Generation (1971-73)

The microprocessors that were introduced in 1971 to 1972 were referred to as the first generation systems. Intel Corporation introduced 4-bit 4004 at 108 kHz, the first microprocessor in 1971, co-developed by Busicom, a Japanese manufacturer of calculators. In 1972, Intel made the 8-bit 8008 and 8080 microprocessors.

Second Generation (1974-78)

The second generation marked the beginning of very efficient 8-bit microprocessors. Some of the popular processors are Motorola's 6800 and 6809 and Intel's 8085, Zilog's Z80. The distinction between the first and second generation devices is primarily the use of newer semiconductor technology to fabricate the chips. They were manufactured using NMOS technology.

Third Generation (1979-80)

Introduced in 1978, dominated by Intel's 8086 and the Zilog Z8000, which were 16-bit processors, have 16-bit arithmetic and pipelined instruction processing.

Fourth Generation (1981-95)

Intel introduced 32 bit processor, 80386 and Motorola 68020/68030. Fabricated using low-power version of the HMOS technology called HCMOS.

Fifth Generation (1995 till date)

Chips carry on-chip functionalities and improvements in the speed of memory and I/O devices. Design surpassed 10 million transistors per chip. Introduction of 64-bit processors.Intel leads the show with Pentium, Celeron and dual and quad core processors working with up to 3.5GHz speed.

Intel 8085 Microprocessor

Introduction

8085 is an eight bit microprocessor of Intel Corporation, usually called as a general purpose 8-bit processor. It is upward compatible with microprocessor 8080, which is the earlier product of Intel. Several faster versions of 8085 microprocessor are 8085AH, 8085AH-1, and 8085AH-2.

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.

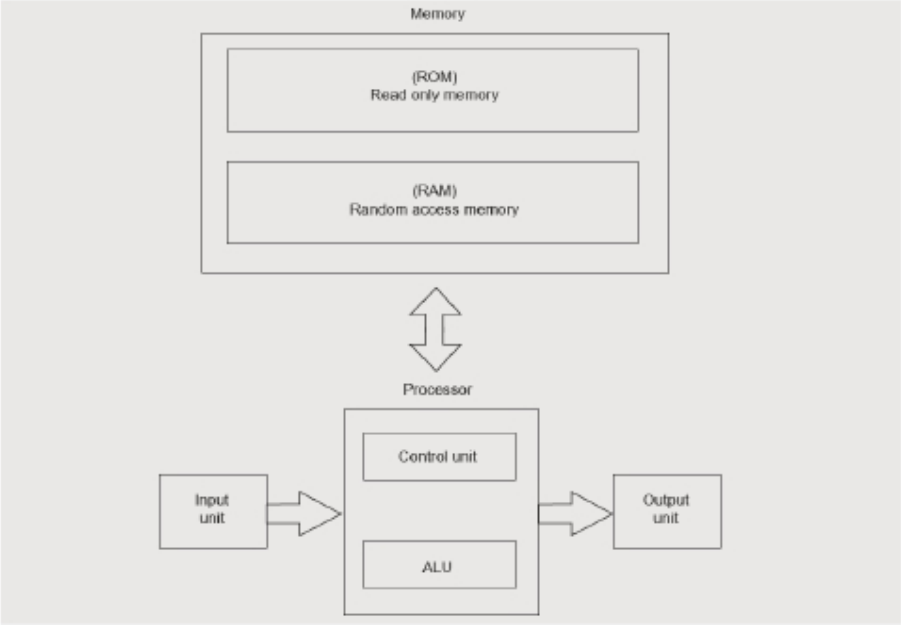


Figure1.1: Microprocessor System

Microprocessor is an integrated chip that functions as the central processing unit of a computer. The microprocessor basically performs

- Memory Read: Accept data (instruction) from memory.
- Memory Write: Send data to memory.
- I/O Read: Accept data from input device.
- I/O Write: Send data to output device.
- Controls timing of instruction flow.

Memory includes ROM (read only memory) and RAM (random access memory or read write memory).The memory

- Stores instructions and data.
- Provides the instructions and data to processor.
- Stores results.

The input devices enter instructions and data to processor. The input device includes Key-Board (Hexadecimal / ASCII), Switches, and Analog-to-Digital (A/D) convertor.

The output devices accept data from processor. This includes LED (Light Emitting Diode) display, LCD (Liquid Crystal Diode) display, CRT (Cathode Ray Tube) Screen.

Architecture

The internal block diagram of 8085 is shown in figure 1.2. It is a 40 pin IC package and uses +5V for power. It can run at a maximum frequency of 3MHz. It is a 8-bit processor which has a data bus of 8-bits wide. It has addressing capability of 16-bit.That is it can address 2^{16} = 64K Bytes of memory (1Kbyte =1024 byte).

The architecture of 8085 processor consists of five functional units: Arithmetic and logic Unit, General purpose registers, Special Purpose Registers, Instruction register/decoder and Timing and control unit.

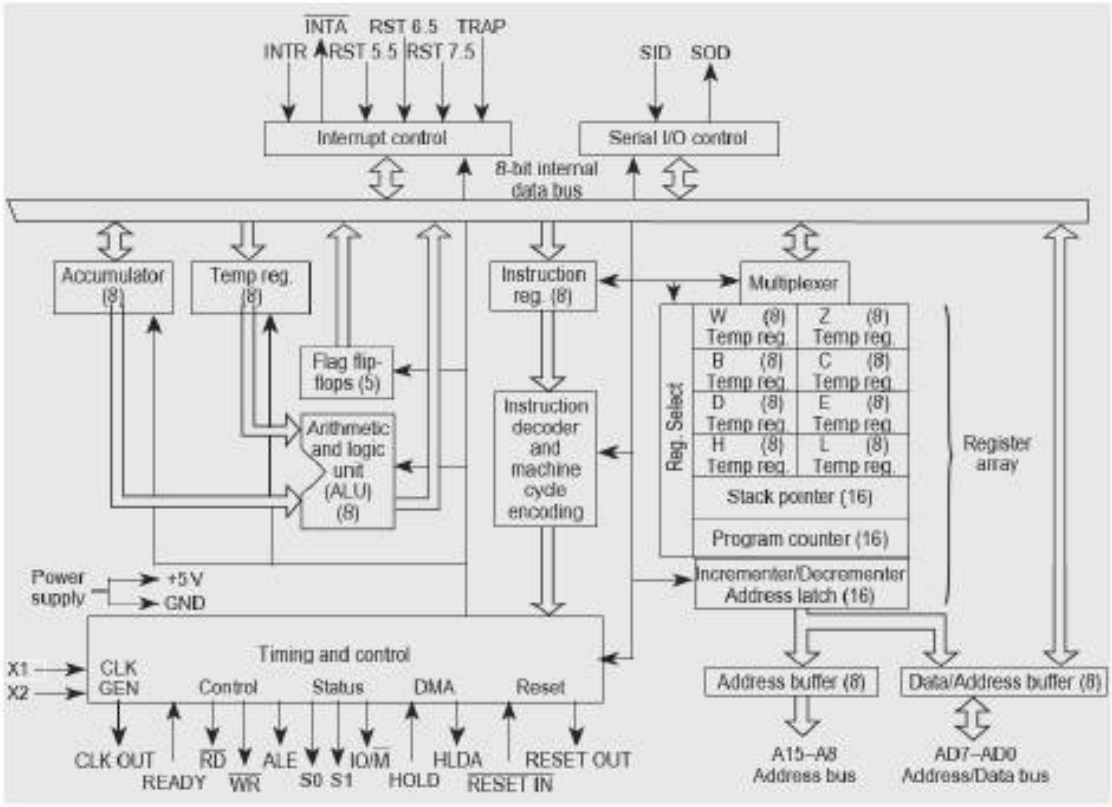
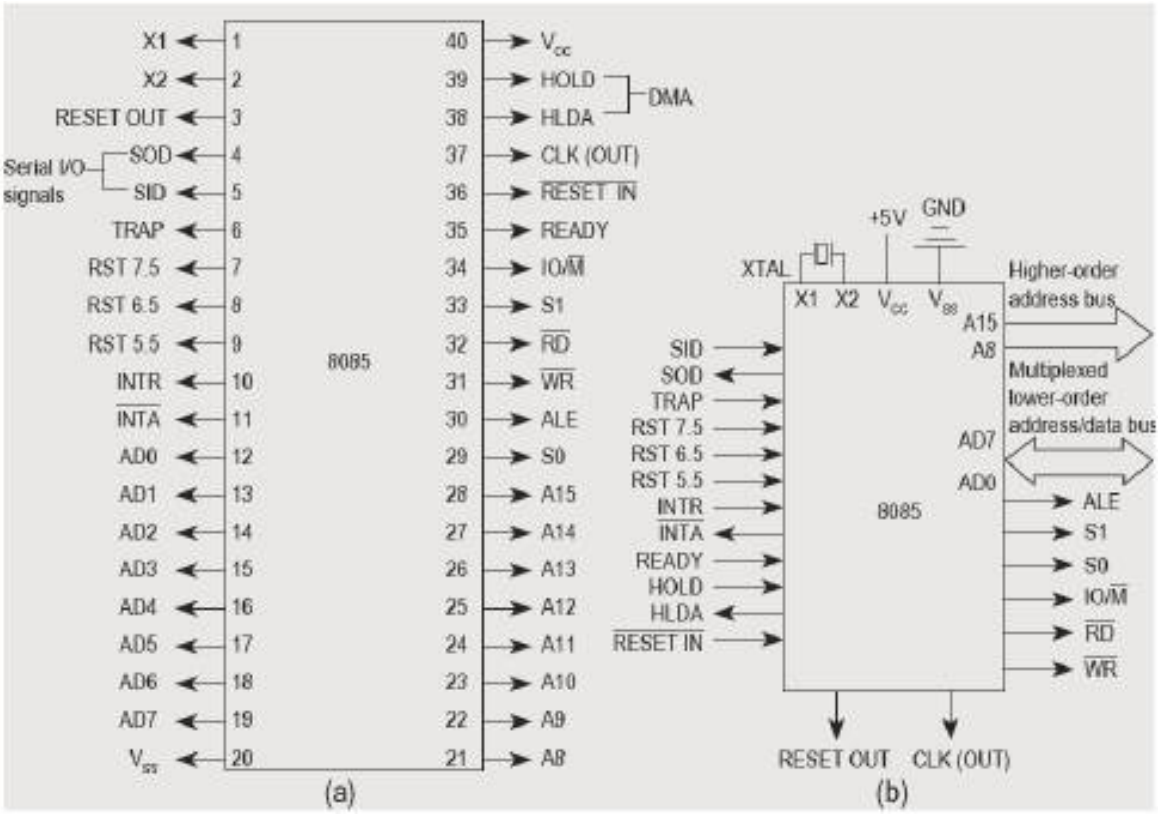


Figure 1.2: Architecture of 8085



Pin Diagram of 8085

Register Organization

The register organization of 8085 is shown in figure 1.3.

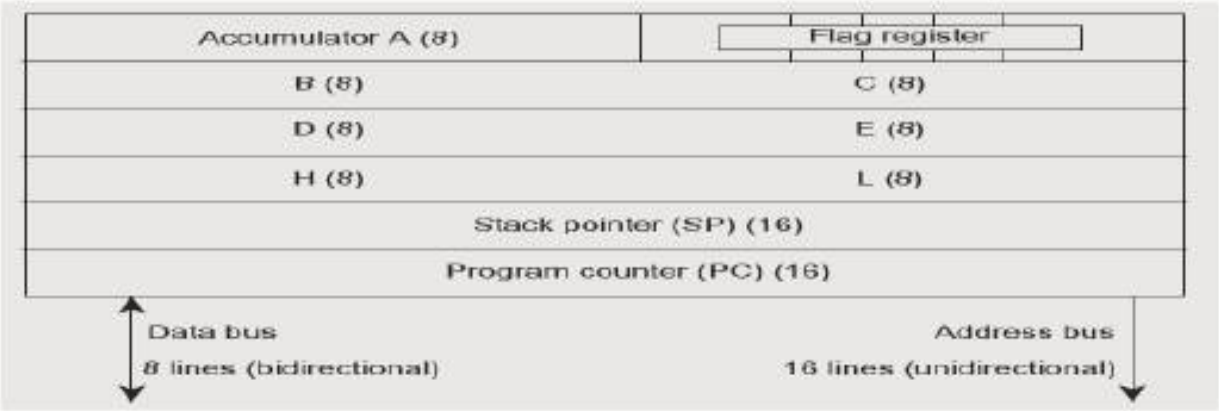


Figure1.3: Register Organization of 8085

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: 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 1.4.

S	Z	X	AC	X	P	X	CY
D7	D6	D5	D4	D3	D2	D1	D0

Figure1.4: 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 (Figure 1.5).

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

Figure1.5: 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.

Instruction Timing and Execution:

An instruction is a command to a microprocessor to perform a specific task on data. Each instruction consists of two parts: one is operation code (OPCODE), and second one is Operand.

Instruction Cycle:

It is defined as the time taken by the processor to complete fetch and execution of an instruction. Instruction cycle consists of both fetch cycle and execute cycle. Instruction cycle consists of 1 to 6 machine cycles.

Fetch Cycle:

It is defined as the time taken by the processor to fetch an operation code from memory.

Execution Cycle:

It is defined as the time taken by the processor to decode and execute an instruction.

Machine Cycle:

The time required to complete one operation of accessing memory, I/O. or acknowledging an external request. The machine cycle consists of 3 to 6 T-states.

T state:

The T state is equal to one clock period. The T state is defined as one sub division of an operation performed in one clock period.

8085 Machine Cycles

The 8085 microprocessor has 5 (Five) basic machine cycles. They are Opcode Fetch Cycle (4T), Memory Read Cycle (3 T), Memory Write Cycle (3 T), I/O Read Cycle (3 T), and I/O Write Cycle (3 T).In Opcode Fetch Cycle processor get a Opcode from memory. The processor read data from memory in Memory Read Cycle. In Memory Write Cycle, the data is stored in memory. The processors get data from input device in I/O Read Cycle and send data to output device in I/O Write Cycle.

Timing Diagram:

The timing diagram of an instruction is obtained by drawing the binary levels on the various signals of 8085. It is drawn with respect to the clock input of the microprocessor. It explains the execution of the instruction with the basic machine cycles of that instruction, one by one in the order of execution.

Opcode Fetch Cycle (4T):

It is defined as the time taken by the processor to fetch an operation code from memory. The program counter places 16-bit memory address on the address bus. The 8085 sends memory read control signal to enable memory chip. The operation code from memory location is placed on the data bus. The operation code from data bus is stored in the instruction register to decode and execute.

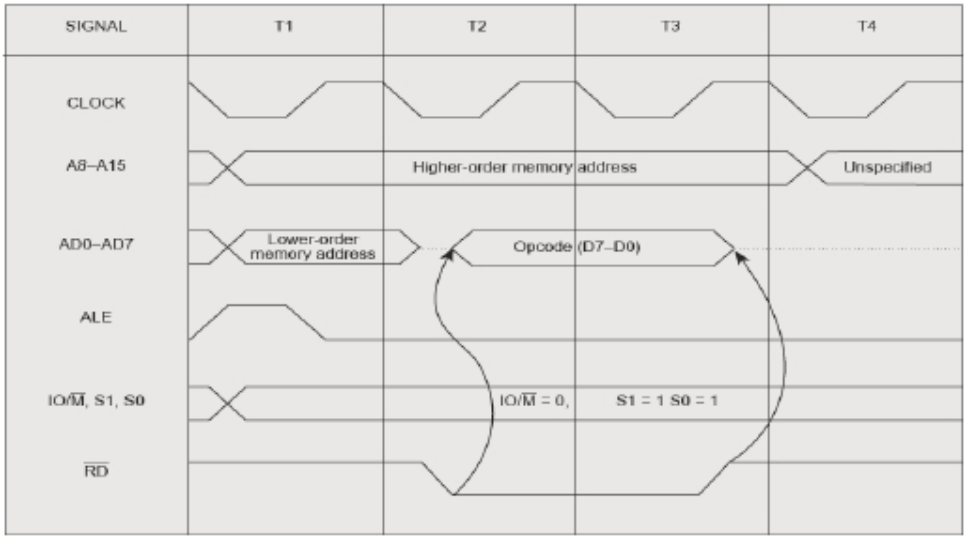


Figure1.6: Timing Diagram of Opcode Fetch Cycle

- At T1, the high order 8 address bits are placed on the address lines A8 – A15 and the low order bits are placed on AD7–AD0. The ALE signal goes high to indicate that AD0 – AD8 are carrying an address. At exactly the same time, the IO/M signal goes low to indicate a memory access operation.
- At the beginning of the T2 cycle, the low order 8 address bits are removed from AD7– AD0 and the processor sends the Read (RD) signal to the memory. The RD signal remains low (active) for two clock periods to allow for reading slow devices. During T2, memory places the operation code byte from the memory location on the lines AD7– AD0.

- During T3 the RD signal is Disabled (goes high). This turns off the output Tri-state buffers in the memory. That makes the AD7– AD0 lines go to high impedance state.
- During T4, the opcode is decoded by the processor and necessary action or control is initiated for the execution of the instruction fetched.

Memory Read Cycle (3 T) :

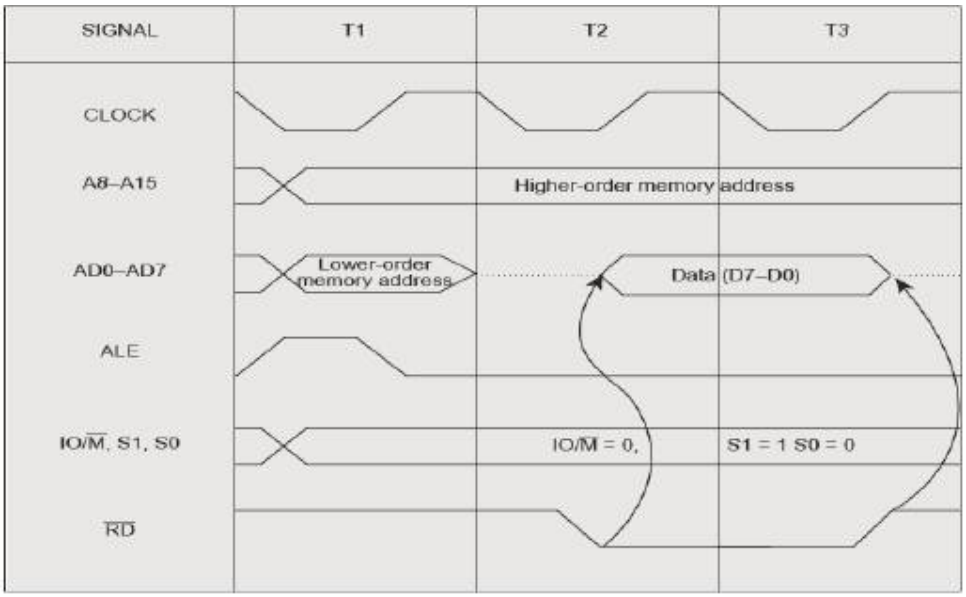


Figure1.7: Timing Diagram of Memory Read Cycle

The memory read cycle is similar as the opcode fetch cycle except: It only has 3 T-states. The S0 signal is 0. S1 is set to 1.In memory read cycle processor read a data byte from memory.

UNIT-II

UNIT-II

Instruction Set of 8085

An instruction is a command to a microprocessor to perform a specific task on data. The instruction set is the total number of instructions supported by processor.

Code	Registers
000	B
001	C
010	D
011	E
100	H
101	L
110	A
111	M

Code	Register Pair
00	B-C
01	D-E
10	H-L
11	AF or SP

Each instruction consists of two parts: one is operation code (OPCODE), and second one is Operand.

The operation code specifies the type of operation to be performed. The operand is the data to be operated on. The operand includes 8-bit/16-bit data, an internal register, memory location, and 8-bit/16-bit address.

In the design of 8085, all the instructions, and registers are identified with a binary code.

Instruction Word Size

The word size of 8085 instruction can be

1-byte instruction: ADD B

The opcode and operand are in the same byte.

2-byte instruction: MVI A, 05H.

The first byte specifies opcode and second byte specify the operand.

3-byte instruction: LDA 2000H.

The first byte specifies opcode and the following two bytes specify 16-bit address.

Addressing Mode

The various techniques to specify data for instruction is called addressing mode.

Direct addressing: The address of the data is specified in the instruction itself.

Ex: STA 2000H
IN 07H.

Register addressing: The operands are in general purpose register.

Ex: MOV A, B
ADD B.

Register indirect addressing: The address of the operand is specified by register pair indirectly.

Ex: MOV A, M
ADD M

Immediate addressing: The operand is specified within the instruction itself.

Ex: MVI A,03H
ADI 12H

Implicit addressing: The operand is in the accumulator.

Ex: CMA
RAL
RAR

INSTRUCTION SET of 8085 :

The entire group of instructions is called the instruction set, and this determines the functionalities the microprocessor can perform.

The 8085 Instructions can be classified into five functional categories: Data Transfer (copy) operations, Arithmetic operations, Logical operations, Branching operations, and Machine-control operations.

1) Data transfer instructions

This group of instructions copy data from a location called a source to another location called a destination without modifying the contents of the source. The various type of data transfer are:

Between registers: MOV B, D

Data byte to register/memory: MVI A, 45H, MVI M, 34H.

Between memory and register: LDA 2100H, STA 2400H.

Between I/O device and accumulator: IN 16H, OUT 12H.

2) Arithmetic instruction

The arithmetic instructions are addition, subtraction, increment, and decrement.

Addition

Any 8-bit number, or the content of a register, or the content of a memory location can be added to the content of accumulator. The sum is stored in the accumulator.

The DAD instruction adds 16-bit numbers in register pairs.

Subtraction

Any 8-bit number, or the content of a register, or the content of a memory location can be subtracted from the content of accumulator. The result is stored in the accumulator. The subtraction is performed in 2's complement form.

Mnemonics	Tasks performed on execution	Addressing mode	Length of the Instruction	Example
ADI 8-bit	Add immediate to accumulator	Immediate	Two bytes	ADI 30H
ACI 8-bit	Add immediate to accumulator with carry	Immediate	Two bytes	ACI 4FH
SUI 8-bit	Subtract immediate from accumulator	Immediate	Two bytes	SUI 2AH
SBI 8-bit	Subtract immediate from accumulator with borrow	Immediate	Two bytes	SBI 5CH

ADD R	Add register content to accumulator	Direct	One byte	ADD C
ADC R	Add register content to accumulator with carry	Direct	One byte	ADC E
SUB R	Subtract register content from accumulator	Direct	One byte	SUB B
SBB R	Subtract register content and borrow from accumulator	Direct	One byte	SBB C
DAD Rp	Add register pair to H and L registers	Direct	One byte	DAD B
INR R	Increment register by 1	Direct	One byte	INR B
INX Rp	Increment register pair by 1	Direct	One byte	INX B
DCR R	Decrement register by 1	Direct	One byte	DCR E
DCX Rp	Decrement register pair by 1	Direct	One byte	DCX D
ADD M	Add memory content pointed by HL register pair to accumulator	Indirect	One byte	ADD C
ADC M	Add memory content pointed by HL register pair to accumulator	Indirect	One byte	ADC E
SUB M	Subtract memory content pointed by HL register pair from accumulator	Indirect	One byte	SUB B
SBB M	Subtract memory content pointed by HL pair and borrow from accumulator	Indirect	One byte	SBB C
INR M	Increment a memory content pointed by HL register pair once	Indirect	One byte	INR M
DCR M	Increment a memory content pointed by HL register pair once	Indirect	One byte	INR M
DAA	Decimal adjust accumulator	Implicit	One byte	DAA

Figure1.8: Arithmetic Instructions of 8085

3) Logical instruction

These instructions perform logical operation with content of accumulator. These instructions are AND, OR, Ex-OR, rotate, compare, and complement.

Logical AND

Any 8-bit number, or the content of a register, or the content of a memory location can be logically ANDed with the content of accumulator. The result is stored in the accumulator.

Mnemonics	Tasks performed on execution	Addressing mode	Length of the Instruction	Example
ANI 8-bit	Logical AND immediate with accumulator	Immediate	Two bytes	ANI 0FH
XRI 8-bit	Exclusive OR immediate with accumulator	Immediate	Two bytes	XRI 01H
ORI 8-bit	Logical OR immediate with accumulator	Immediate	Two bytes	ORI 80H
ANA R	Logical AND register or memory with accumulator	Direct	One byte	ANA C
XRA R	Exclusive OR register or memory with accumulator	Direct	One byte	XRA D
ORA R	Logical OR register or memory with accumulator	Direct	One byte	ORA E
ANA M	Logical AND memory pointed by HL register pair with accumulator	Indirect	One byte	ANA M
XRA M	Logical XOR memory pointed by HL register pair with accumulator	Indirect	One byte	XRA M
ORA M	Logical OR memory pointed by HL register pair with accumulator	Indirect	One byte	ORA M
RLC	Rotate accumulator left	Implicit	One byte	RLC
RRC	Rotate accumulator right	Implicit	One byte	RRC
RAL	Rotate accumulator left through carry	Implicit	One byte	RAL
RAR	Rotate accumulator right through carry	Implicit	One byte	RAR
CPI 8-bit	Compare immediate with accumulator	Immediate	Two bytes	CPI FFH
CMP R	Compare register or memory with accumulator	Direct	One byte	CMP B
CMP M	Compare memory pointed by HL register pair with accumulator	Indirect	One byte	CMP M
CMA	Complement accumulator	Implicit	One byte	CMA
CMC	Complement carry	Implicit	One byte	CMC
STC	Set carry	Implicit	One byte	STC

Figure 1.9: Logical Instructions of 8085

4) Branching Instructions

This instruction alters the sequence of program execution either conditionally or unconditionally. These instructions are jump, call, return, and restart.

Mnemonics	Tasks performed on execution	Addressing mode	Length of the Instruction	Example	
JMP 16-bit	Jump unconditionally	Immediate	Three bytes	JMP 9500	
JC 16-bit	Jump if carry is set	Immediate	Three bytes	JC 9500	
JNC 16-bit	Jump on no carry	Immediate	Three bytes	JNC 9500	
JP 16-bit	Jump on positive	Immediate	Three bytes	JP 9500	
JM 16-bit	Jump on minus	Immediate	Three bytes	JM 9500	
JZ 16-bit	Jump on zero	Immediate	Three bytes	JZ 9500	
JNZ16-bit	Jump on no zero	Immediate	Three bytes	JNZ 9500	
JPE 16-bit	Jump on parity even	Immediate	Three bytes	JPE 9500	
JPO 16-bit	Jump on parity odd	Immediate	Three bytes	JPO 9500	
CALL 16-bit	Call unconditionally	Immediate	Three bytes	CALL 9500	
CC 16-bit	Call on carry	Immediate	Three bytes	CC 9500	
CNC 16-bit	Call on no carry	Immediate	Three bytes	CNC 9500	
CP 16-bit	Call on positive	Immediate	Three bytes	CP 9500	
CM 16-bit	Call on minus	Immediate	Three bytes	CM 9500	
CZ 16-bit	Call on zero	Immediate	Three bytes	CZ 9500	
CNZ 16-bit	Call on no zero	Immediate	Three bytes	CNZ 9500	
CPE 16-bit	Call on parity even	Immediate	Three bytes	CPE 9500	
CPO 16-bit	Call on parity odd	Immediate	Three bytes	CPO 9500	
RET	Return unconditionally	Implicit	One byte	RET	
RC	Return on carry	Implicit	One byte	RC	
RNC	Return on no carry	Implicit	One byte	RNC	
RP	Return on positive	Implicit	One byte	RP	

RM	Return on minus	Implicit	One byte	RM	
RZ	Return on zero	Implicit	One byte	RZ	
RNZ	Return on no zero	Implicit	One byte	RNZ	
RPE	Return on parity even	Implicit	One byte	RPE	
RPO	Return on parity odd	Implicit	One byte	RPO	
PCHL	Copy HL contents to t program	Implicit	One byte	PCHL	
RST 0/1/2/3/4/5/6/7	Restart	Implicit	One byte	RST 5	

Figure1.10: Branch Instructions of 8085

Machine control operation

These instruction are halt, interrupt, and do nothing etc.

Mnemonics	Tasks performed on execution	Addressing mode	Length of the Instruction
NOP	No operation	Implicit	One byte
HLT	Halt the microprocessor execution	Implicit	One byte
DI	Disable interrupts	Implicit	One byte
EI	Enable interrupts	Implicit	One byte
RIM	Read interrupt mask	Implicit	One byte
SIM	Set interrupt mask	Implicit	One byte

Assembly Language Programming using 8085 instruction set

Statement: Add the content of memory location 4000H with content of memory location 4001H and place the result in memory location 4002H.

(4000H) = 14H

(4001H) = 89H

Result = 14H + 89H = 9DH

Source program

LXI H 4000H : HL points 4000H

MOV A, M : Get first operand

INX H : HL points 4001H

ADD M : Add second operand

INX H : HL points 4002H

MOV M, A : Store result at 4002H

HLT : Terminate program execution

Statement: Subtract the contents of memory location 4001H from the memory location 2000H and place the result in memory location 4002H.

Program - 4: Subtract two 8-bit numbers

Sample problem:

(4000H) = 51H

(4001H) = 19H

Result = 51H - 19H = 38H

LXI H, 4000H : HL points 4000H

MOV A, M : Get first operand

INX H : HL points 4001H

SUB M : Subtract second operand

INX H : HL points 4002H

MOV M, A : Store result at 4002H.

HLT : Terminate program execution

Memory interfacing

Introduction: RAM, ROM, EPROM

The programs and data which are executed by the microprocessor have to be stored in ROM/EPROM and RAM which are basically semiconductor memory chips.

The programs and data which are stored in ROM/EPROM are not erased even-though the power supply to the ROM/EPROM chip is removed. Hence the ROM/EPROM are called non-volatile memory and we can use them to store permanent programs such as monitor program and data such as look up table, which are needed in microprocessor based systems. The difference between ROM and EPROM is that ROM chip is programmable only one time whereas an EPROM chip can be programmed many times.

The programs and data which are stored in RAM are erased when the power supply to the RAM chip is removed and hence RAM is called volatile memory. The program written during the learning of assembly language programming and data entered while testing the above programs are stored in RAM.

Interfacing memory chips with 8085

8085 has 16 address lines namely (A15 to A0), a maximum of 64Kbytes (=2^{16}) memory can be interfaced with 8085. The 64 KB memory address space of 8085 has the value from 0000H to FFFFH when represented in hexadecimal form.

The 8085 access memory to read the instructions and data stored in memory and also to store the result in to memory.

The 8085 initiates a set of signals and when it wants to read from and write into memory and the memory chip has certain signals such as chip Enable or Chip Select, or Output Enable or Read and or Write Enable or Write.

Interfacing 2764 EPROM chip with 8085

There are 13 address lines (Since 2^{13} =8K) namely A12 to A0 present in IC 2764 (Fig. 2.1) where A0 is the least significant bit of the address and A12 is the most significant bit of the address.

In order to read the content of a memory location in EPROM chip, the following steps are done

- The address of the memory location from where data has to be read is placed in the address lines of EPROM.
- CE signal is made logic low (i.e. 0)
- OE signal is made logic low(i.e. 0)
- Now the data in the selected memory location will be available in the data lines (D7-D0) of EPROM.

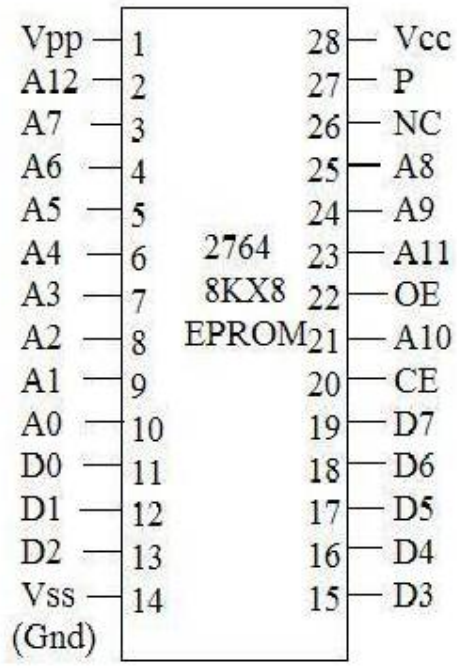


Figure 2.2: Pin diagram of IC 2764

Whenever many number of same capacity memory chips have to be interfaced with 8085, decoder IC with active low outputs such as 74LS138 is very useful.

By using a single 74LS138 IC, maximum of eight memory chips (RAM and EPROM) can be interfaced with 8085.

Partial address decoding:

If the entire address bus of 8085 (A15 to A0) is used to interface memory chips with 8085 than this technique is called absolute address decoding. Using the absolute address decoding method, maximum of 64 Kbytes of memory can be interfaced with 8085.

There is another method known as partial address decoding which is used when the amount of memory needed in an 8085 based system is less than 64 Kbytes such as 8K or 16K or 32K.

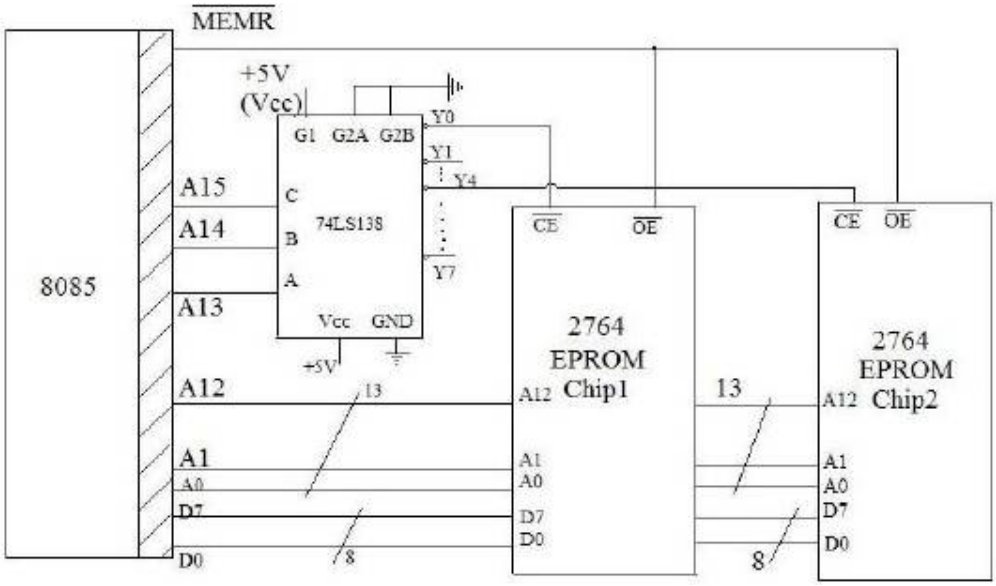


Figure2.3: Interfacing EPROM chips using 74LS138 Decoder

Interfacing 6264 RAM chip with 8085

The interfacing of RAM chip with 8085 is same as that of EPROM chip except that one more signals namely write of 8085 is used.6264 RAM has 13 address lines.

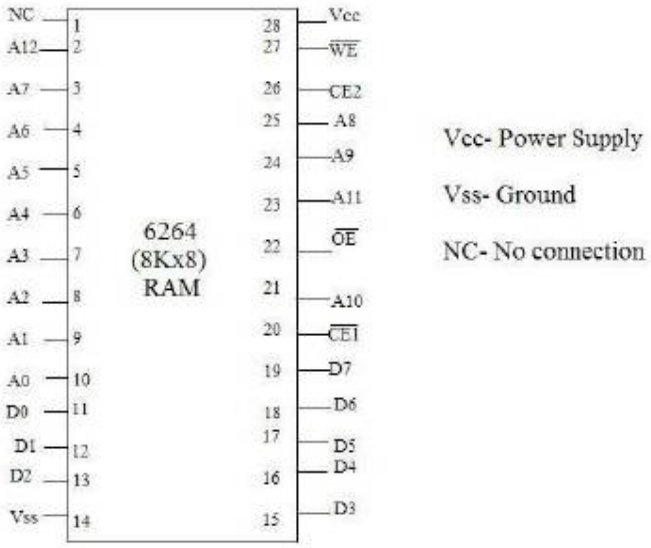


Figure2.4: Pin diagram of IC 6264 RAM

Stack and Subroutines

The stack is an area of R/W memory identified by the programmer for temporary storage of information.

- The stack is a Last in First out (LIFO) structure.
- The stack normally grows backwards into memory.

In other words, the programmer defines the bottom of the stack and the stack grows up into reducing address range.

Given that the stack grows backwards into memory, it is customary to place the bottom of the stack at the end of memory to keep it as far away from user programs as possible.

In the 8085, the stack is defined by setting the SP (Stack Pointer) register.

LXI SP, FFFFH

This sets the Stack Pointer to location FFFFH (end of memory for the 8085).

The Size of the stack is limited only by the available memory

Saving Information on the Stack

Information is saved on the stack by PUSH ing it on. It is retrieved from the stack by POPing it off.

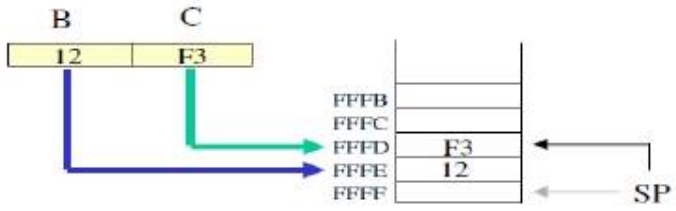
The 8085 provides two instructions: PUSH and POP for storing information on the stack and retrieving it back. Both PUSH and POP work with register pairs only.

The PUSH Instruction

PUSH B (1 Byte Instruction).

Decrement SP, Copy the contents of register B to the memory location pointed to by SP.

Decrement SP, Copy the contents of register C to the memory location pointed to by SP



The POP Instruction

POP D (1 Byte Instruction).

Copy the contents of the memory location pointed to by the SP to register E, Increment SP.

Copy the contents of the memory location pointed to by the SP to register D, Increment SP.

Operation of the Stack

During pushing, the stack operates in a "decrement then store" style.

The stack pointer is decremented first, and then the information is placed on the stack.

During popping, the stack operates in a "use then increment" style.

The information is retrieved from the top of the, the stack and then the pointer is incremented.

The SP pointer always points to "the top of the stack".

LIFO

The order of PUSHs and POPs must be opposite of each other in order to retrieve information back into its original location.

PUSH B

PUSH D

...

POP D

POP B

Reversing the order of the POP instructions will result in the exchange of the contents of BC and DE.

The PSW Register Pair

The 8085 recognizes one additional register pair called the PSW (Program Status Word).

This register pair is made up of the Accumulator and the Flags register.

It is possible to push the PSW onto the stack, do whatever operations are needed, then POP it off of the stack.

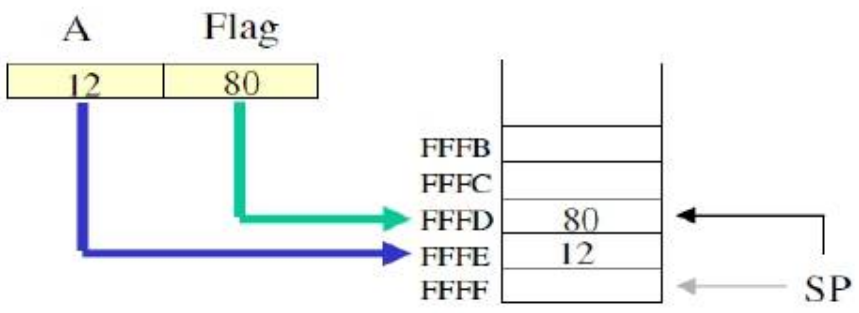
The result is that the contents of the Accumulator and the status of the Flags are returned to what they were before the operations were executed.

PUSH PSW Register Pair

PUSH PSW (1 Byte Instruction).

Decrement SP, Copy the contents of register A to the memory location pointed to by SP.

Decrement SP, Copy the contents of Flag register to the memory location pointed to by SP.



Pop PSW Register Pair

POP PSW (1 Byte Instruction).

Increment SP, Copy the contents of the memory location pointed to by the SP to Flag register

Increment SP, Copy the contents of the memory location pointed to by the SP to register A

Subroutines

A subroutine is a group of instructions that will be used repeatedly in different locations of the program.

Rather than repeat the same instructions several times, they can be grouped into a subroutine that is called from the different locations.

In Assembly language, a subroutine can exist anywhere in the code. However, it is customary to place subroutines separately from the main program.

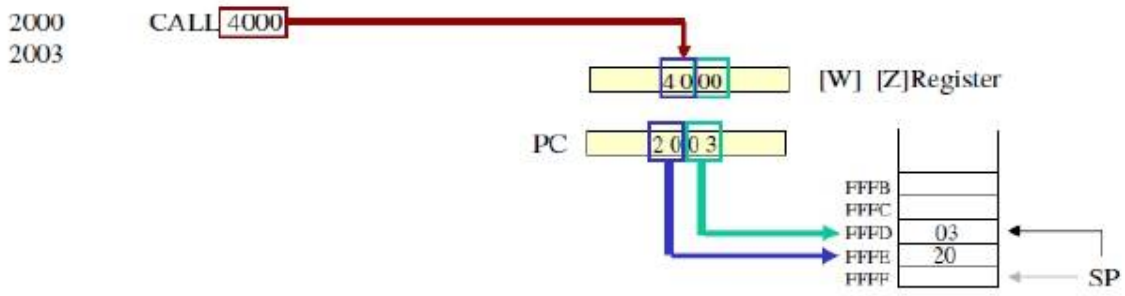
The 8085 has two instructions for dealing with subroutines.

The CALL instruction is used to redirect program execution to the subroutine.

The RET instruction is used to return the execution to the calling routine.

CALL 4000H (3 byte instruction)

When CALL instruction is fetched, the Processor knows that the next two memory location contains 16bit subroutine address in the memory.



The Processor Reads the subroutine address from the next two memory location and stores the higher order 8-bit of the address in the W register and stores the lower order 8-bit of the address in the Z register

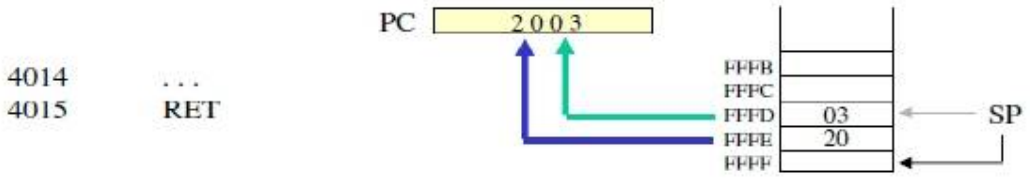
Push the address of the instruction immediately following the CALL onto the stack.

Loads the program counter with the 16-bit address supplied with the CALL instruction from WZ register.

RET (1 byte instruction)

Retrieve the return address from the top of the stack

Load the program counter with the return address.



Subroutine provides flexibility in writing program and uses memory efficiently.

Nesting:

The programming technique of a subroutine calling another subroutine is called nesting of subroutine. This process is limited only by available stack locations.

Recursive Subroutine:

A subroutine which is called by itself is called recursive subroutine.

Reentrant Subroutine:

In nested subroutine, if a latter subroutine calls an earlier subroutine, it is known as reentrant subroutine.

Multiple Ending Subroutines:

The subroutine which has more than one return is called multiple ending subroutines.

Multiple Calling of subroutine:

Calling a subroutine more than once by main program are called multiple calling subroutines.

Parameter passing:

It is necessary pass data and address variable of main program to subroutine. This passing of data and address is called parameter passing. Parameter passing can be done by using register, pointer, memory and stack.

8085 Interrupts

Introduction

Interrupts are mainly used for interrupt driven data transfer between processor and slower input/ output device. The primary function of the microprocessor is to accept data from input devices such as keyboards and A/D converters, read instruction from memory, process data according to the instructions, and send the results to output device such as LEDs, printers and video monitors. The 8085 has five number of interrupt pins: TRAP, RST 7.5, RST 6.5, RST 5.5, and INTR.

When the 8085 is interrupted by external devices, it is called hardware interrupt. In software interrupt, the program execution is interrupted by using software instruction.

Types of Interrupts

- Vectored and Non-vectored Interrupts
- Maskable and Non-maskable Interrupts
- Software and Hardware Interrupt

Interrupt Handling Procedure

The Processor will have to store the information about the current program when an interrupt signal is recognized before executing the ISR. The processor checks for the Interrupt request signals at the end of every instruction execution. If the interrupt is masked, then the interrupt will not be recognized until interrupts are re-enabled. The sequence of operations that take place when an interrupt signal is recognized is as followed. The CPU responds to an interrupt request by a transfer of control to another program in a manner similar to a subroutine call. Save the PC (Program Counter) contents (address of the next instruction) and supplementary information about current state (flags, registers, etc.) to the stack. Load PC with the beginning address of an Interrupt Service Routine (ISR) and start to execute it. Finish ISR when return instruction is executed. Return to the interrupted program, exactly to the same point from which it left.

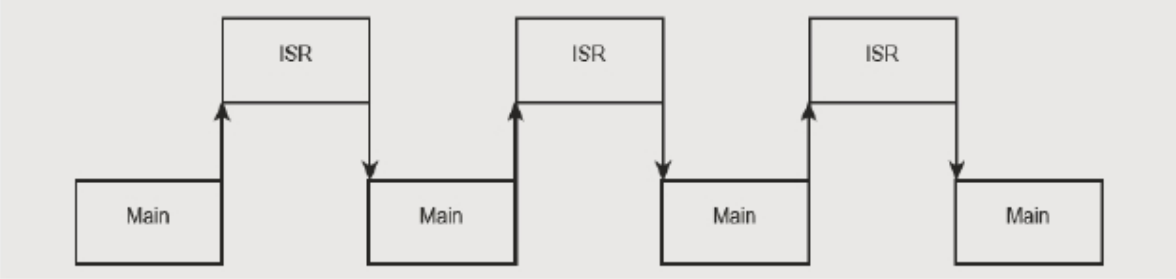


Figure2.5: Transfer of control from Main memory to ISR

Interrupt Sources and their Vector Addresses in 8085

Intel 8085 has the facility for both software and hardware interrupts. The software interrupts are in the form of instructions and the hardware interrupts are applied as signals from the external devices.

Hardware Interrupts and Priorities

Intel 8085 has 5 hardware interrupts: INTR, RT 5.5, RST 6.6, RST 7.5, and TRAP. The details of the five interrupts are given in the table 2.1. Five pins of 8085 are reserved for the five hardware interrupts. All the five interrupts are active high signals. This means that in order to apply an interrupt, logic 1 or high level signal should be applied at these pins. The processor checks the voltage on these pins after the execution of every instruction. If the signal level on any of these 5 pins is at logic 1 and the corresponding interrupt is not masked, then the processor will suspend the current program and execute the corresponding interrupt service routine. RST 7.5 interrupt alone is edge triggered. That means a logic 0 to 1 transition will be treated as an interrupt input on this line. The rising edge interrupt can be applied at any time and this will set a flip flop inside the processor. The processor will check this flip flop while checking the signal level on other hardware interrupts**.**

Table 2.1: Hardware interrupts in 8085

Interrupt	Interrupt vector address	Maskable or Non Maskable	Edge-or Level-triggered	priority
Trap	0024H	Non-maskable	Level-triggered	1
RST 7.5	003CH	Maskable	Rising edge triggered	2
RST 6.5	0034H	Maskable	Level-triggered	3
RST 5.5	002CH	Maskable	Level-triggered	4
INTR	Decided by Hardware	Maskable	Level-triggered	5

Software Interrupts

8085 microprocessor instruction set has eight software interrupts instruction called Restart (RST) instructions. These are one-byte call instruction that transfer program execution to subroutine at predefined address. Table2.2 gives the eight software instructions and their corresponding interrupt vector address.

The vector address for a software interrupt is calculated as follows:

Vector address = interrupt number x 8.

The vector address for RST 5 is calculated as

5x 8=40 $_{10}$=28H. So vector address of RST 5 is 0028H

Table 2.2: Software interrupts and their Vector address

Instruction	Machine hex Code	Interrupt Vector address
RST 0	C7	0000H
RST 1	CF	0008H
RST 2	D7	0010H
RST 3	DF	0018H
RST 4	E7	0020H
RST 5	EF	0028H
RST 6	F7	0030H
RST 7	FF	0038H

Masking of Interrupts

The 8085 interrupt structure is shown in figure 2.5.The maskable interrupts are by default masked by the RESET signal. So, any interrupt will not be recognized by the hardware reset. The interrupts can be enabled by the execution of the instruction, EI – Enable interrupts. The three RST interrupts can be selectively masked by having proper word in Accumulator and executing the SIM (Set Interrupt Mask) instruction. This is called software masking. All the maskable interrupts are disabled whenever an interrupt is recognized. So, it is necessary to execute EI instruction every time the interrupts are recognized and serviced by the processor. All the maskable interrupts can be disabled by executing an instruction DI – Disable Interrupts. This instruction will reset an interrupt enable flip flop in the processor and the interrupts will be disabled. To enable interrupts, EI instruction has to be executed.

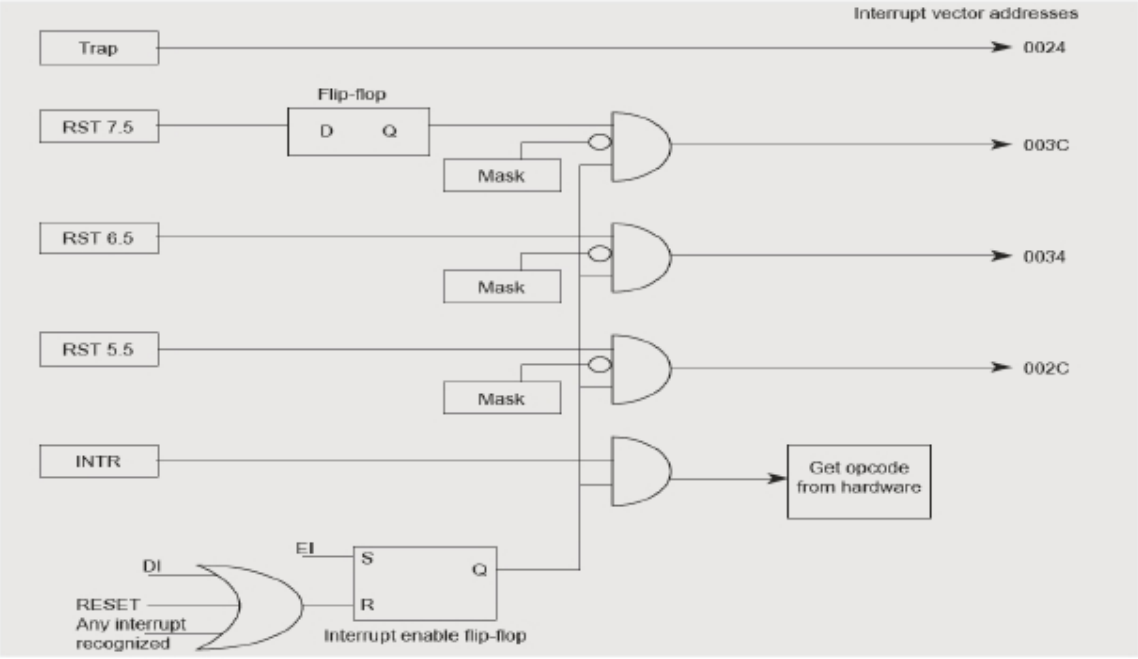


Figure2.6: interrupts structure of 8085

SIM Instruction

The SIM instruction is used to mask or unmask the restart RST hardware interrupts. Figure 2.7 shows the SIM instruction format. The SIM instruction when executed will read the contents of the accumulator and based on that will mask or unmask the interrupts. So, SIM instruction must be executed after storing having proper control word in accumulator. The format of the control word is to be stored in accumulator before executing SIM instruction.

Bit position	D7	D6	D5	D4	D3	D2	D1	D0
Name	SOD	SDE	X	R7.5	MSE	M7.5	M6.5	M5.5
Explanation	Serial data to be sent	Serial data enable—set to 1 for sending	Not used	Reset RST 7.5 flip-flop	Mask set enable—Set to 1 to mask interrupts	Set to 1 to mask RST 7.5	Set to 1 to mask RST 6.5	Set to 1 to mask RST 5.5

Figure2.7: SIM instruction Format

The instructions required for execution of SIM are

MVI A, control word

SIM

RIM Instruction

RIM stands for 'Read Interrupt Mask' and its format is shown in figure 2.8. When RIM instruction is executed, the status of serial input data (SID), pending interrupts and interrupt masks are loaded into accumulator.

Bit position	D7	D6	D5	D4	D3	D2	D1	D0
Name	SID	I7.5	I6.5	I5.5	IE	M7.5	M6.5	M5.5
Explanation	Serial input data in the SID pin	Set to 1 if RST 7.5 is pending	Set to 1 if RST 6.5 is pending	Set to 1 if RST 5.5 is pending	Set to 1 if interrupts are enabled	Set to 1 if RST 7.5 is masked	Set to 1 if RST 6.5 is masked	Set to 1 if RST 5.5 is masked

Figure2.8: RIM instruction Format

The instructions required for execution of RIM are

MVI A, control word

RIM

UNIT-III

7. 8086 – Overview

8086 Microprocessor is an enhanced version of 8085Microprocessor that was designed by Intel in 1976.It is a 16-bit Microprocessor having 20 address lines and16 data lines that provides up to 1MB storage. It consists of powerful instruction set, which provides operations like multiplication and division easily.

It supports two modes of operation, i.e. Maximum mode and Minimum mode. Maximum mode is suitable for system having multiple processors and Minimum mode is suitable for system having a single processor.

Features of 8086

The most prominent features of a 8086 microprocessor are as follows:

- It has an instruction queue, which is capable of storing six instruction bytes from the memory resulting in faster processing.
- It was the first 16-bit processor having 16-bit ALU, 16-bit registers, internal data bus, and 16-bit external data bus resulting in faster processing.
- It is available in 3 versions based on the frequency of operation:
  - 8086 -> 5MHz
  - 8086-2 ->8MHz
  - (c)8086-1 ->10 MHz
- It uses two stages of pipelining, i.e. Fetch Stage and Execute Stage, which improves performance.
- Fetch stage can prefetch up to 6 bytes of instructions and stores them in the queue.
- Execute stage executes these instructions.
- It has 256 vectored interrupts.
- It consists of 29,000 transistors.

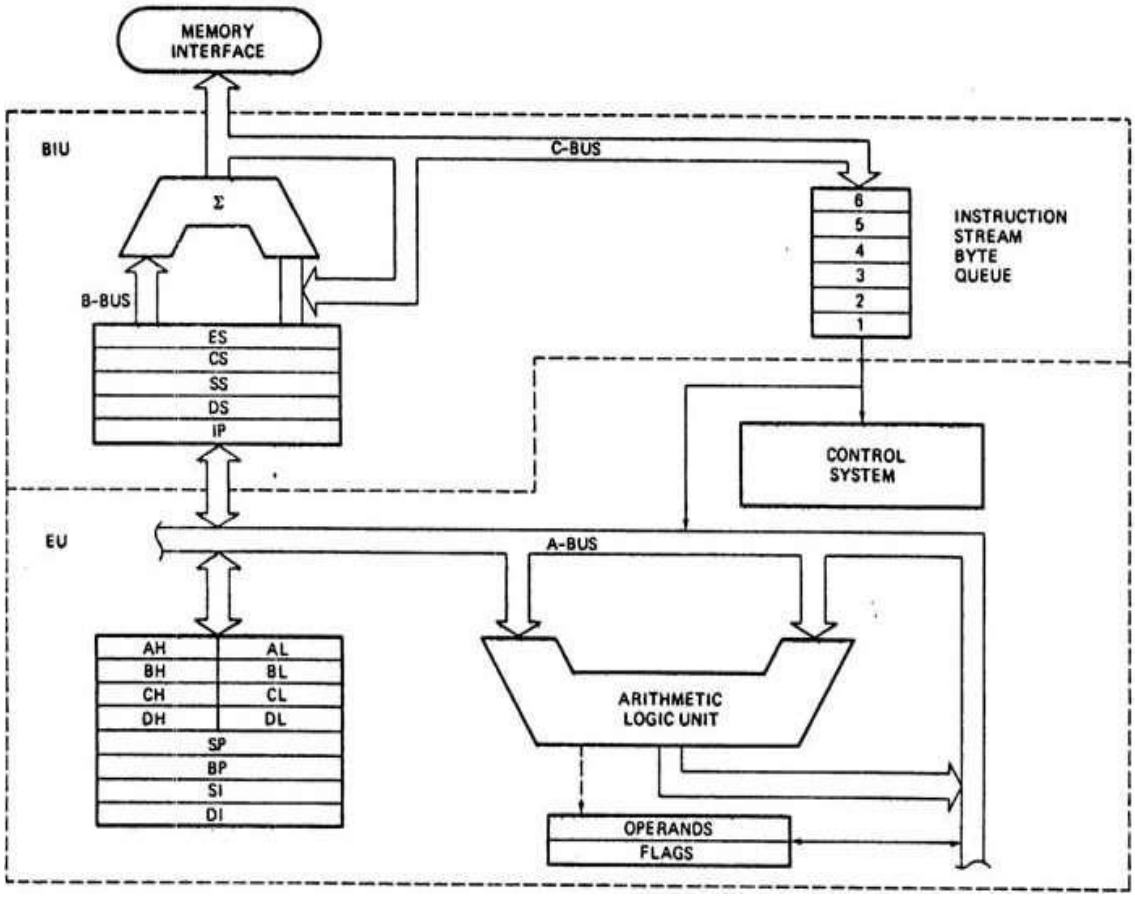
Comparison between 8085 &8086 Microprocessor

- **Size**: 8085 is 8-bit microprocessor, whereas 8086 is 16-bit microprocessor.
- **Address Bus**: 8085 has 16-bit address bus while 8086 has 20-bit address bus.
- **Memory**: 8085 can access up to 64Kb, whereas 8086 can access up to 1 Mb of memory.
- **Instruction**: 8085 doesn't have an instruction queue, whereas 8086 has an instruction queue.

- **Pipelining**: 8085 doesn't support a pipelined architecture while 8086 supports a pipelined architecture.
- **I/O**: 8085 can address 2^8 = 256 I/O's, whereas 8086 can access 2^16 = 65,536 I/O's.
- **Cost**: The cost of 8085 is low whereas that of 8086 is high.

Architecture of 8086

The following diagram depicts the architecture of a 8086 Microprocessor:



8. 8086 – Functional Units

8086 Microprocessor is divided into two functional units, i.e., **EU** (Execution Unit) and **BIU** (Bus Interface Unit).

EU (Execution Unit)

Execution unit gives instructions to BIU stating from where to fetch the data and then decode and execute those instructions. Its function is to control operations on data using the instruction decoder & ALU. EU has no direct connection with system buses as shown in the above figure, it performs operations over data through BIU.

Let us now discuss the functional parts of 8086 microprocessors.

ALU

It handles all arithmetic and logical operations, like +,-,×,/, OR, AND, NOT operations.

Flag Register

It is a 16-bit register that behaves like a flip-flop, i.e. it changes its status according to the result stored in the accumulator. It has 9 flags and they are divided into 2 groups: Conditional Flags and Control Flags.

Conditional Flags

It represents the result of the last arithmetic or logical instruction executed. Following is the list of conditional flags:

- **Carry flag**: This flag indicates an overflow condition for arithmetic operations.
- **Auxiliary flag**: When an operation is performed at ALU, it results in a carry/barrow from lower nibble (i.e. D0 – D3) to upper nibble (i.e. D4 – D7), then this flag is set, i.e. carry given by D3 bit to D4 is AF flag. The processor uses this flag to perform binary to BCD conversion.
- **Parity flag**: This flag is used to indicate the parity of the result, i.e. when the lower order 8-bits of the result contains even number of 1's, then the Parity Flag is set. For odd number of 1's, the Parity Flag is reset.
- **Zero flag**: This flag is set to 1 when the result of arithmetic or logical operation is zero else it is set to 0.
- **Sign flag**: This flag holds the sign of the result, i.e. when the result of the operation is negative, then the sign flag is set to 1 else set to 0.
- **Overflow flag**: This flag represents the result when the system capacity is exceeded.

Control Flags

Control flags controls the operations of the execution unit. Following is the list of control flags:

- **Trap flag**: It is used for single step control and allows the user to execute one instruction at a time for debugging. If it is set, then the program can be run in a single step mode.

- **Interrupt flag**: It is an interrupt enable/disable flag, i.e. used to allow/prohibit the interruption of a program. It is set to 1 for interrupt enabled condition and set to 0 for interrupt disabled condition.

- **Direction flag**: It is used in string operation. As the name suggests when it is set then string bytes are accessed from the higher memory address to the lower memory address and vice-a-versa.

General purpose register

There are 8 general purpose registers, i.e., AH, AL, BH, BL, CH, CL, DH, and DL. These registers can be used individually to store 8-bit data and can be used in pairs to store 16-bit data. The valid register pairs are AH and AL, BH and BL, CH and CL, and DH and DL. It is referred to the AX, BX, CX, and DX respectively.

- **AX register**: It is also known as accumulator register. It is used to store operands for arithmetic operations.

- **BX register**: It is used as a base register. It is used to store the starting base address of the memory area within the data segment.

- **CX register**: It is referred to as counter. It is used in loop instruction to store the loop counter.

- **DX register**: This register is used to hold I/O port address for I/O instruction.

Stack pointer register

It is a 16-bit register, which holds the address from the start of the segment to the memory location, where a word was most recently stored on the stack.

BIU (Bus Interface Unit)

BIU takes care of all data and addresses transfers on the buses for the EU like sending addresses, fetching instructions from the memory, reading data from the ports and the memory as well as writing data to the ports and the memory. EU has no direction connection with System Buses so this is possible with the BIU. EU and BIU are connected with the Internal Bus.

It has the following functional parts:

- **Instruction queue**: BIU contains the instruction queue. BIU gets upto 6 bytes of next instructions and stores them in the instruction queue. When EU executes

instructions and is ready for its next instruction, then it simply reads the instruction from this instruction queue resulting in increased execution speed.

- Fetching the next instruction while the current instruction executes is called **pipelining**.

- **Segment register**: BIU has 4 segment buses, i.e. CS, DS, SS& ES. It holds the addresses of instructions and data in memory, which are used by the processor to access memory locations. It also contains 1 pointer register IP, which holds the address of the next instruction to executed by the EU.

    - **CS**: It stands for Code Segment. It is used for addressing a memory location in the code segment of the memory, where the executable program is stored.

    - **DS**: It stands for Data Segment. It consists of data used by the program andis accessed in the data segment by an offset address or the content of other register that holds the offset address.

    - **SS**: It stands for Stack Segment. It handles memory to store data and addresses during execution.

    - **ES**: It stands for Extra Segment. ES is additional data segment, which is used by the string to hold the extra destination data.

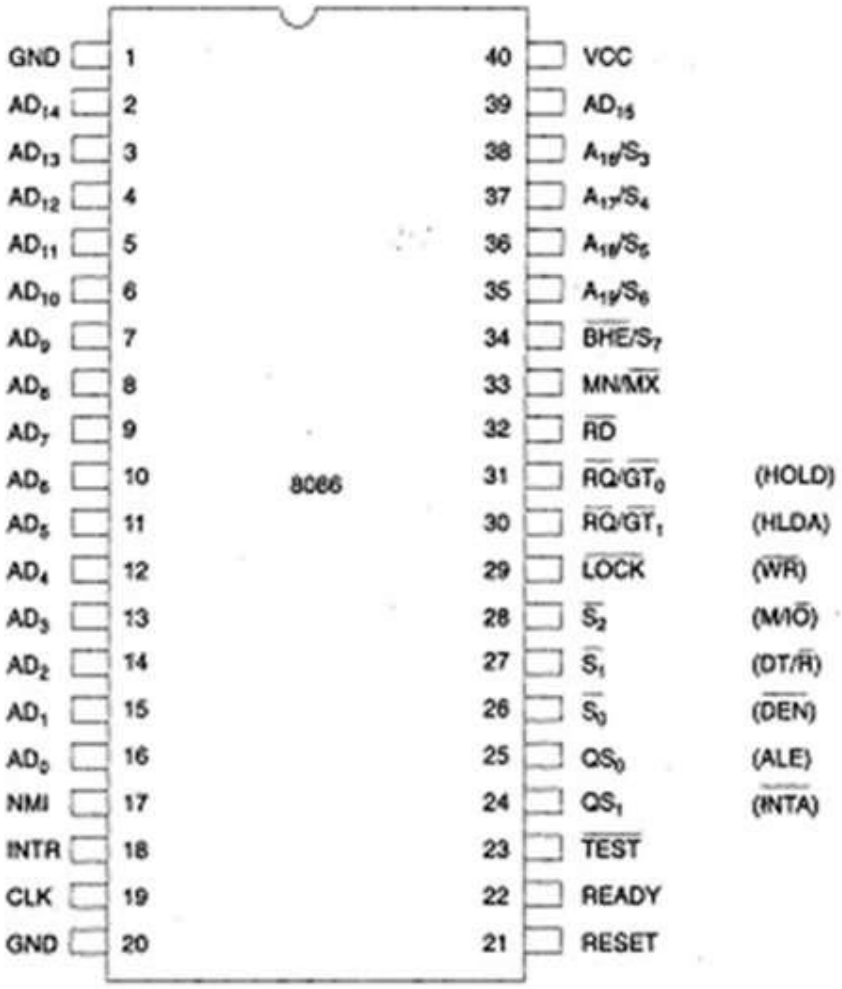
- **Instruction pointer**: It is a 16-bit register used to hold the address of the next instruction to be executed.

9. 8086 – Pin Configuration

8086 was the first 16-bit microprocessor available in 40-pin DIP (Dual Inline Package) chip. Let us now discuss in detail the pin configuration of a 8086 Microprocessor.

8086 Pin Diagram

Here is the pin diagram of 8086 microprocessor:



Let us now discuss the signals in detail:

Power supply & frequency signals

It uses 5V DC supply at V_{CC} pin 40, and uses ground at V_{SS} pin 1 and 20 for its operation.

Clock signal

Clock signal is provided through Pin-19. It provides timing to the processor for operations. Its frequency is different for different versions, i.e. 5MHz, 8MHz and 10MHz.

Address/data bus

AD0-AD15. These are 16 address/data bus. AD0-AD7 carries low order byte data and AD8-AD15 carries higher order byte data. During the first clock cycle, it carries 16-bit address and after that it carries 16-bit data.

Address/status bus

A16-A19/S3-S6. These are the 4 address/status buses. During the first clock cycle, it carries 4-bit address and later it carries status signals.

S7/BHE

BHE stands for Bus High Enable. It is available at pin 34 and used to indicate the transfer of data using data bus D8-D15. This signal is low during the first clock cycle, thereafter it is active.

Read ( )

It is available at pin 32 and is used to read signal for Read operation.

Ready

It is available at pin 32. It is an acknowledgement signal from I/O devices that data is transferred. It is an active high signal. When it is high, it indicates that the device is ready to transfer data. When it is low, it indicates wait state.

RESET

It is available at pin 21 and is used to restart the execution. It causes the processor to immediately terminate its present activity. This signal is active high for the first 4 clock cycles to RESET the microprocessor.

INTR

It is available at pin 18. It is an interrupt request signal, which is sampled during the last clock cycle of each instruction to determine if the processor considered this as an interrupt or not.

NMI

It stands for non-maskable interrupt and is available at pin 17. It is an edge triggered input, which causes an interrupt request to the microprocessor.

This signal is like wait state and is available at pin 23. When this signal is high, then the processor has to wait for IDLE state, else the execution continues.

MN/

It stands for Minimum/Maximum and is available at pin 33. It indicates what mode the processor is to operate in; when it is high, it works in the minimum mode and vice-a-versa.

INTA

It is an interrupt acknowledgement signal and id available at pin 24. When the microprocessor receives this signal, it acknowledges the interrupt.

ALE

It stands for address enable latch and is available at pin 25. A positive pulse is generated each time the processor begins any operation. This signal indicates the availability of a valid address on the address/data lines.

DEN

It stands for Data Enable and is available at pin 26. It is used to enable Transreceiver 8286. The transreceiver is a device used to separate data from the address/data bus.

DT/R

It stands for Data Transmit/Receive signal and is available at pin 27. It decides the direction of data flow through the transreceiver. When it is high, data is transmitted out and vice-a-versa.

M/IO

This signal is used to distinguish between memory and I/O operations. When it is high, it indicates I/O operation and when it is low indicates the memory operation. It is available at pin 28.

WR

It stands for write signal and is available at pin 29. It is used to write the data into the memory or the output device depending on the status of M/IO signal.

HLDA

It stands for Hold Acknowledgement signal and is available at pin 30. This signal acknowledges the HOLD signal.

HOLD

This signal indicates to the processor that external devices are requesting to access the address/data buses. It is available at pin 31.

QS_1 & QS_0

These are queue status signals and are available at pin 24 and 25. These signals provide the status of instruction queue. Their conditions are shown in the following table:

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

S_0, S_1, S_2

These are the status signals that provide the status of operation, which is used by the Bus Controller 8288 to generate memory & I/O control signals. These are available at pin 26, 27, and 28. Following is the table showing their status:

S_2	S_1	S_0	Status
0	0	0	Interrupt acknowledgement
0	0	1	I/O Read
0	1	0	I/O Write
0	1	1	Halt
1	0	0	Opcode fetch
1	0	1	Memory read
1	1	0	Memory write
1	1	1	Passive

LOCK

When this signal is active, it indicates to the other processors not to ask the CPU to leave the system bus. It is activated using the LOCK prefix on any instruction and is available at pin 29.

RQ/GT_1 & RQ/GT_0

These are the Request/Grant signals used by the other processors requesting the CPU to release the system bus. When the signal is received by CPU, then it sends acknowledgment. RQ/GT_0 has a higher priority than RQ/GT_1.

10. 8086 – Instruction Sets

The 8086 microprocessor supports 8 types of instructions:

- Data Transfer Instructions
- Arithmetic Instructions
- Bit Manipulation Instructions
- String Instructions
- Program Execution Transfer Instructions (Branch & Loop Instructions)
- Processor Control Instructions
- Iteration Control Instructions
- Interrupt Instructions

Let us now discuss these instruction sets in detail.

Data Transfer Instructions

These instructions are used to transfer the data from the source operand to the destination operand. Following are the list of instructions under this group:

Instruction to transfer a word

- **MOV**: Used to copy the byte or word from the provided source to the provided destination.
- **PPUSH**: Used to put a word at the top of the stack.
- **POP**: Used to get a word from the top of the stack to the provided location.
- **PUSHA**: Used to put all the registers into the stack.
- **POPA**: Used to get words from the stack to all registers.
- **XCHG**: Used to exchange the data from two locations.
- **XLAT**: Used to translate a byte in AL using a table in the memory.

Instructions for input & output port transfer

- **IN**: Used to read a byte or word from the provided port to the accumulator.
- **OUT**: Used to send out a byte or word from the accumulator to the provided port.

Instructions to transfer the address

- **LEA**: Used to load the address of operand into the provided register.
- **LDS**: Used to load DS register and other provided register from the memory
- **LES**: Used to load ES register and other provided register from the memory.

Instructions to transfer flag registers

- **LAHF**: Used to load AH with the low byte of the flag register.
- **SAHF**: Used to store AH register to low byte of the flag register.
- **PUSHF**: Used to copy the flag register at the top of the stack.
- **POPF**: Used to copy a word at the top of the stack to the flag register.

Arithmetic Instructions

These instructions are used to perform arithmetic operations like addition, subtraction, multiplication, division, etc.

Following is the list of instructions under this group:

Instructions to perform addition

- **ADD**: Used to add the provided byte to byte/word to word.
- **ADC**: Used to add with carry.
- **INC**: Used to increment the provided byte/word by 1.
- **AAA**: Used to adjust ASCII after addition.
- **DAA**: Used to adjust the decimal after the addition/subtraction operation.

Instructions to perform subtraction

- **SUB**: Used to subtract the byte from byte/word from word.
- **SBB**: Used to perform subtraction with borrow.
- **DEC**: Used to decrement the provided byte/word by 1.
- **NPG**: Used to negate each bit of the provided byte/word and add 1/2's complement.
- **CMP**: Used to compare 2 provided byte/word.
- **AAS**: Used to adjust ASCII codes after subtraction.
- **DAS**: Used to adjust decimal after subtraction.

Instruction to perform multiplication

- **MUL**: Used to multiply unsigned byte by byte/word by word.
- **IMUL**: Used to multiply signed byte by byte/word by word.
- **AAM**: Used to adjust ASCII codes after multiplication.

Instructions to perform division

- **DIV**: Used to divide the unsigned word by byte or unsigned double word by word.
- **IDIV**: Used to divide the signed word by byte or signed double word by word.
- **AAD**: Used to adjust ASCII codes after division.
- **CBW**: Used to fill the upper byte of the word with the copies of sign bit of the lower byte.
- **CWD**: Used to fill the upper word of the double word with the sign bit of the lower word.

Bit Manipulation Instructions

These instructions are used to perform operations where data bits are involved, i.e. operations like logical, shift, etc.

Following is the list of instructions under this group:

Instructions to perform logical operation

- **NOT**: Used to invert each bit of a byte or word.
- **AND**: Used for adding each bit in a byte/word with the corresponding bit in another byte/word.
- **OR**: Used to multiply each bit in a byte/word with the corresponding bit in another byte/word.
- **XOR**: Used to perform Exclusive-OR operation over each bit in a byte/word with the corresponding bit in another byte/word.
- **TEST**: Used to add operands to update flags, without affecting operands.

Instructions to perform shift operations

- **SHL/SAL**: Used to shift bits of a byte/word towards left and put zero(S) in LSBs.
- **SHR**: Used to shift bits of a byte/word towards the right and put zero(S) in MSBs.
- **SAR**: Used to shift bits of a byte/word towards the right and copy the old MSB into the new MSB.

Instructions to perform rotate operations

- **ROL**: Used to rotate bits of byte/word towards the left, i.e. MSB to LSB and to Carry Flag [CF].
- **ROR**: Used to rotate bits of byte/word towards the right, i.e. LSB to MSB and to Carry Flag [CF].
- **RCR**: Used to rotate bits of byte/word towards the right, i.e. LSB to CF and CF to MSB.

- **RCL**: Used to rotate bits of byte/word towards the left, i.e. MSB to CF and CF to LSB.

String Instructions

String is a group of bytes/words and their memory is always allocated in a sequential order.

Following is the list of instructions under this group:

- **REP**: Used to repeat the given instruction till CX≠0.
- **REPE/REPZ**: Used to repeat the given instruction until CX=0 or zero flag ZF=1.
- **REPNE/REPNZ**: Used to repeat the given instruction until CX=0 or zero flag ZF=1.
- **MOVS/MOVSB/MOVSW**: Used to move the byte/word from one string to another.
- **COMS/COMPSB/COMPSW**: Used to compare two string bytes/words.
- **INS/INSB/INSW**: Used as an input string/byte/word from the I/O port to the provided memory location.
- **OUTS/OUTSB/OUTSW**: Used as an output string/byte/word from the provided memory location to the I/O port.
- **SCAS/SCASB/SCASW**: Used to scan a string and compare its byte with a byte in AL or string word with a word in AX.
- **LODS/LODSB/LODSW**: Used to store the string byte into AL or string word into AX.

Program Execution Transfer Instructions (Branch & Loop Instructions)

These instructions are used to transfer/branch the instructions during an execution. It includes the following instructions:

Instructions to transfer the instruction during an execution without any condition:

- **CALL**: Used to call a procedure and save their return address to the stack.
- **RET**: Used to return from the procedure to the main program.
- **JMP**: Used to jump to the provided address to proceed to the next instruction.

Instructions to transfer the instruction during an execution with some conditions:

- **JA/JNBE**: Used to jump if above/not below/equal instruction satisfies.
- **JAE/JNB**: Used to jump if above/not below instruction satisfies.
- **JBE/JNA**: Used to jump if below/equal/ not above instruction satisfies.
- **JC**: Used to jump if carry flag CF=1

- **JE / JZ**: Used to jump if equal/zero flag ZF=1
- **JG / JNLE**: Used to jump if greater/not less than/equal instruction satisfies.
- **JGE / JNL**: Used to jump if greater than/equal/not less than instruction satisfies.
- **JL / JNGE**: Used to jump if less than/not greater than/equal instruction satisfies.
- **JLE / JNG**: Used to jump if less than/equal/if not greater than instruction satisfies.
- **JNC**: Used to jump if no carry flag (CF=0)
- **JNE / JNZ**: Used to jump if not equal/zero flag ZF=0
- **JNO**: Used to jump if no overflow flag OF=0
- **JNP / JPO**: Used to jump if not parity/parity odd PF=0
- **JNS**: Used to jump if not sign SF=0
- **JO**: Used to jump if overflow flag OF=1
- **JP / JPE**: Used to jump if parity/parity even PF=1
- **JS**: Used to jump if sign flag SF=1

Processor Control Instructions

These instructions are used to control the processor action by setting/resetting the flag values.

Following are the instructions under this group:

- **STC**: Used to set carry flag CF to 1
- **CLC**: Used to clear/reset carry flag CF to 0
- **CMC**: Used to put complement at the state of carry flag CF.
- **STD**: Used to set the direction flag DF to 1
- **CLD**: Used to clear/reset the direction flag DF to 0
- **STI**: Used to set the interrupt enable flag to 1, i.e., enable INTR input.
- **CLI**: Used to clear the interrupt enable flag to 0, i.e., disable INTR input.

Iteration Control Instructions

These instructions are used to execute the given instructions for number of times. Following is the list of instructions under this group:

- **LOOP**: Used to loop a group of instructions until the condition satisfies, i.e., CX=0

- **LOOPE / LOOPZ**: Used to loop a group of instructions till it satisfies ZF=1 & CX=0
- **LOOPNE / LOOPNZ**: Used to loop a group of instructions till it satisfies ZF=0 & CX=0
- **JCXZ**: Used to jump to the provided address if CX=0

Interrupt Instructions

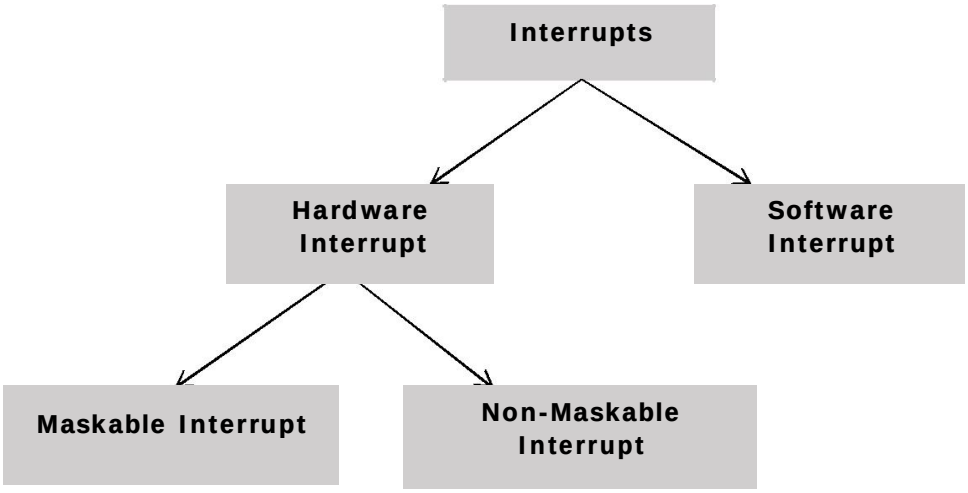
These instructions are used to call the interrupt during program execution.

- **INT**: Used to interrupt the program during execution and calling service specified.
- **INTO**: Used to interrupt the program during execution if OF=1
- **IRET**: Used to return from interrupt service to the main program

11. 8086 – Interrupts

Interrupt is the method of creating a temporary halt during program execution and allows peripheral devices to access the microprocessor. The microprocessor responds to that interrupt with an **ISR** (Interrupt Service Routine), which is a short program to instruct the microprocessor on how to handle the interrupt.

The following image shows the types of interrupts we have in a 8086 microprocessor:



Hardware Interrupts

Hardware interrupt is caused by any peripheral device by sending a signal through a specified pin to the microprocessor.

The 8086 has two hardware interrupt pins, i.e. NMI and INTR. NMI is a non-maskable interrupt and INTR is a maskable interrupt having lower priority. One more interrupt pin associated is INTA called interrupt acknowledge.

NMI

It is a single non-maskable interrupt pin (NMI) having higher priority than the maskable interrupt request pin (INTR)and it is of type 2 interrupt.

When this interrupt is activated, these actions take place:

- Completes the current instruction that is in progress.
- Pushes the Flag register values on to the stack.
- Pushes the CS (code segment) value and IP (instruction pointer) value of the return address on to the stack.
- IP is loaded from the contents of the word location 00008H.

- CS is loaded from the contents of the next word location 0000AH.
- Interrupt flag and trap flag are reset to 0.

INTR

The INTR is a maskable interrupt because the microprocessor will be interrupted only if interrupts are enabled using set interrupt flag instruction. It should not be enabled using clear interrupt Flag instruction.

The INTR interrupt is activated by an I/O port. If the interrupt is enabled and NMI is disabled, then the microprocessor first completes the current execution and sends '0' on INTA pin twice. The first '0' means INTA informs the external device to get ready and during the second '0' the microprocessor receives the 8 bit, say X, from the programmable interrupt controller.

These actions are taken by the microprocessor:

- First completes the current instruction.
- Activates INTA output and receives the interrupt type, say X.
- Flag register value, CS value of the return address and IP value of the return address are pushed on to the stack.
- IP value is loaded from the contents of word location X x 4
- CS is loaded from the contents of the next word location.
- Interrupt flag and trap flag is reset to 0

Software Interrupts

Some instructions are inserted at the desired position into the program to create interrupts. These interrupt instructions can be used to test the working of various interrupt handlers. It includes:

INT- Interrupt instruction with type number

It is 2-byte instruction. First byte provides the op-code and the second byte provides the interrupt type number. There are 256 interrupt types under this group.

Its execution includes the following steps:

- Flag register value is pushed on to the stack.
- CS value of the return address and IP value of the return address are pushed on to the stack.
- IP is loaded from the contents of the word location 'type number' x 4
- CS is loaded from the contents of the next word location.

- Interrupt Flag and Trap Flag are reset to 0

The starting address for type0 interrupt is 000000H, for type1 interrupt is 00004H similarly for type2 is 00008H and ……so on. The first five pointers are dedicated interrupt pointers. i.e.:

- **TYPE 0** interrupt represents division by zero situation.
- **TYPE 1** interrupt represents single-step execution during the debugging of a program.
- **TYPE 2** interrupt represents non-maskable NMI interrupt.
- **TYPE 3** interrupt represents break-point interrupt.
- **TYPE 4** interrupt represents overflow interrupt.

The interrupts from Type 5 to Type 31 are reserved for other advanced microprocessors, and interrupts from 32 to Type 255 are available for hardware and software interrupts.

INT 3-Break Point Interrupt Instruction

It is a 1-byte instruction having op-code is CCH. These instructions are inserted into the program so that when the processor reaches there, then it stops the normal execution of program and follows the break-point procedure.

Its execution includes the following steps:

- Flag register value is pushed on to the stack.
- CS value of the return address and IP value of the return address are pushed on to the stack.
- IP is loaded from the contents of the word location 3x4 = 0000CH
- CS is loaded from the contents of the next word location.
- Interrupt Flag and Trap Flag are reset to 0

INTO - Interrupt on overflow instruction

It is a 1-byte instruction and their mnemonic **INTO**. The op-code for this instruction is CEH. As the name suggests it is a conditional interrupt instruction, i.e. it is active only when the overflow flag is set to 1 and branches to the interrupt handler whose interrupt type number is 4. If the overflow flag is reset then, the execution continues to the next instruction.

Its execution includes the following steps:

- Flag register values are pushed on to the stack.

- CS value of the return address and IP value of the return address are pushed on to the stack.
- IP is loaded from the contents of word location 4x4 = 00010H
- CS is loaded from the contents of the next word location.
- Interrupt flag and Trap flag are reset to 0

12. 8086 – Addressing Modes

The different ways in which a source operand is denoted in an instruction is known as **addressing modes**. There are 8 different addressing modes in 8086 programming:

Immediate addressing mode

The addressing mode in which the data operand is a part of the instruction itself is known as immediate addressing mode. **For example**:

```
MOV CX, 4929 H, ADD AX, 2387 H,  MOV AL, FFH
```

Register addressing mode

It means that the register is the source of an operand for an instruction. **For example**:

```
MOV CX, AX        ; copies the contents of the 16-bit AX register into
                  ; the 16-bit CX register),
ADD BX, AX
```

Direct addressing mode

The addressing mode in which the effective address of the memory location is written directly in the instruction. **For example**:

```
MOV AX, [1592H], MOV AL, [0300H]
```

Register indirect addressing mode

This addressing mode allows data to be addressed at any memory location through an offset address held in any of the following registers: BP, BX, DI& SI. **For example**:

```
MOV AX, [BX]     ; Suppose the register BX contains 4895H, then the contents
                 ; 4895H are moved to AX
ADD CX, {BX}
```

Based addressing mode

In this addressing mode, the offset address of the operand is given by the sum of contents of the BX/BP registers and 8-bit/16-bit displacement. **For example**:

```
MOV DX, [BX+04], ADD CL, [BX+08]
```

Indexed addressing mode

In this addressing mode, the operands offset address is found by adding the contents of SI or DI register and 8-bit/16-bit displacements. **For example**:

```
MOV BX, [SI+16], ADD AL, [DI+16]
```

Based-index addressing mode

In this addressing mode, the offset address of the operand is computed by summing the base register to the contents of an Index register. **For example**:

```
ADD CX, [AX+SI], MOV AX, [AX+DI]
```

Based indexed with displacement mode

In this addressing mode, the operands offset is computed by adding the base register contents. An Index registers contents and 8 or 16-bit displacement**. For example**:

```
MOV AX, [BX+DI+08], ADD CX, [BX+SI+16]
```

MINIMUM MODE 8086 SYSTEM AND TIMINGS

In a minimum mode 8086 system, the microprocessor 8086 is operated in minimum mode by strapping its MN/MX* pin to logic1. 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.

The 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 signal. They are controlled by two signals, namely, DEN* and DT/R*. The DEN* signal indicates that the valid data is available on the data bus, while DT/R indicates the direction of data, i.e. from or to the processor. The system contains memory for the monitor and users program storage. Usually, EPROMS are used for monitor storage, while RAMs for users program storage. A system may contain I/O devices for communication with the processor as well as some special purpose I/O devices. The clock generator generates the clock from the crystal oscillator and then shapes it and divides to make it more precise so that it can be used as an accurate timing reference for the system. The clock generator also synchronizes some external signals with the system clock. The general system organization is shown in Fig. 1.1. Since it has 20 address lines and 16 data lines, the 8086 CPU requires three octal address latches and two octal data buffers for the complete address and data separation.

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.

Fig 1.2 shows the read cycle timing diagram. The read cycle begins in T_1 with the assertion of the 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 T2 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 addressed 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.

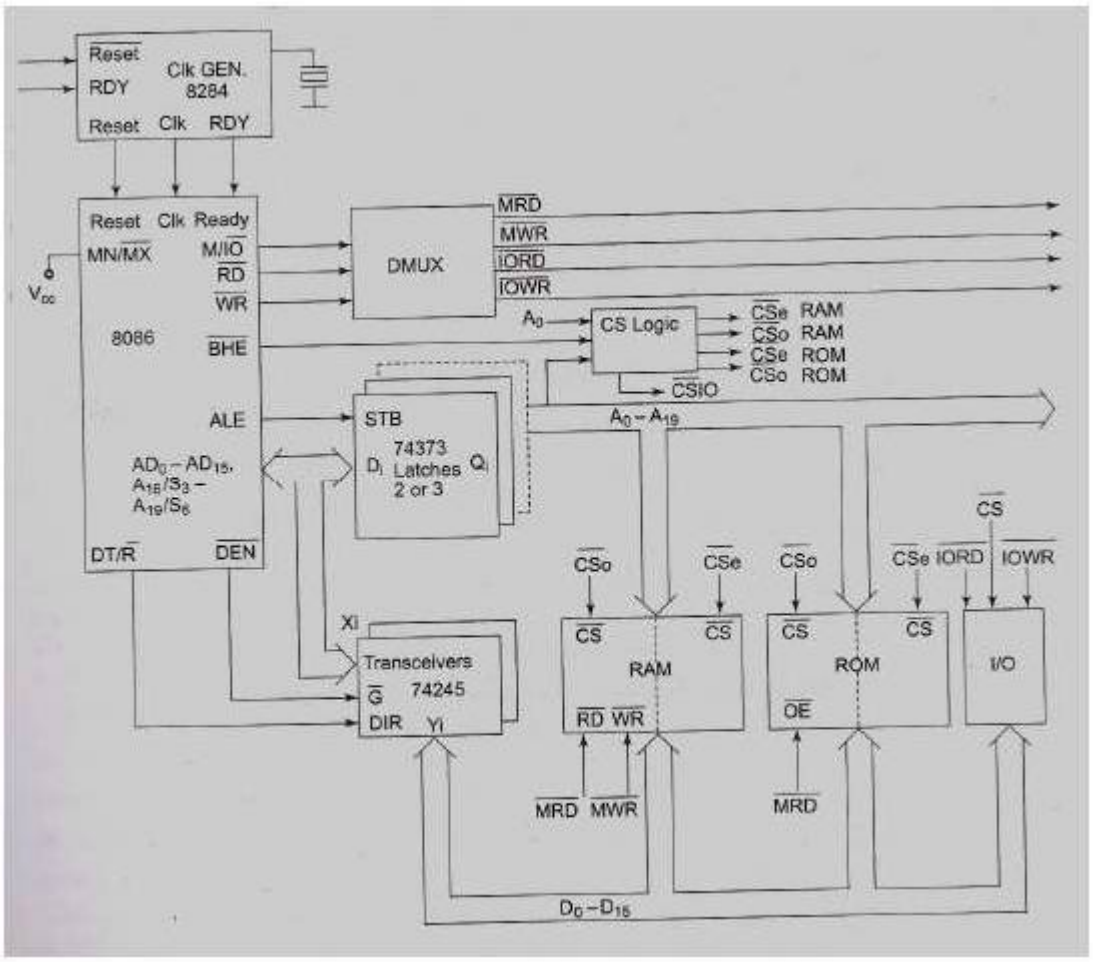


Fig 1.1. Minimum Mode 8086 System

Fig 1.3 shows the write cycle timing diagram. 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 ofT_2 (unlike RD* is somewhat delayed in T_2 to provide time for floating).

M/IO	RD	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

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 written. The M/IO*, RD* and WR* signals indicate the types of data transfer as specified in Table

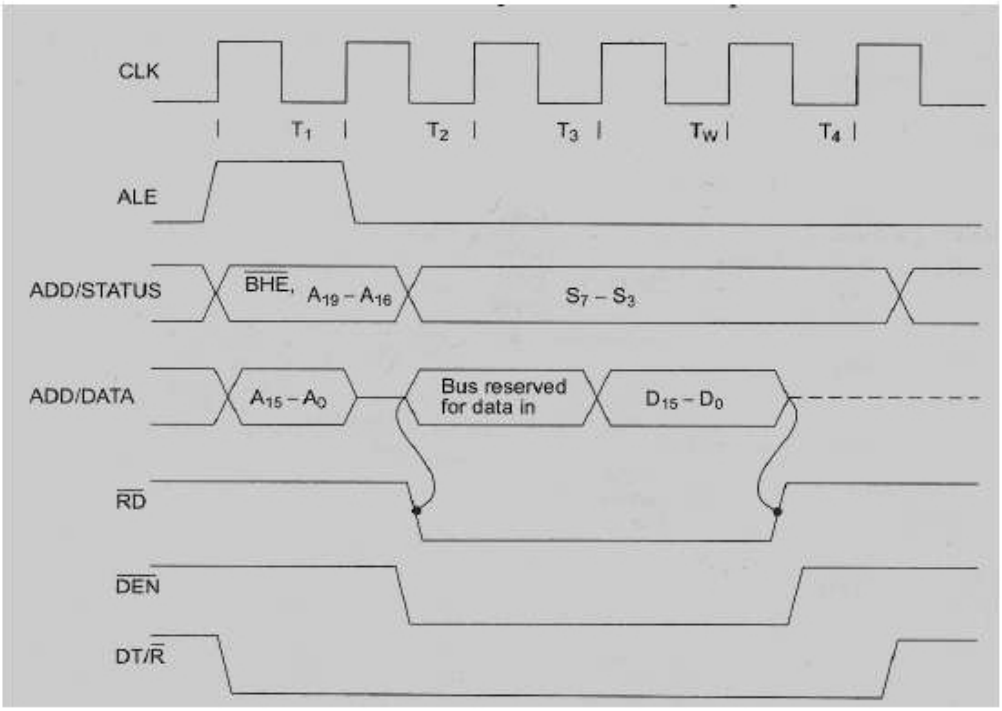


Fig.1.2. Read Cycle Timing Diagram for Minimum Mode

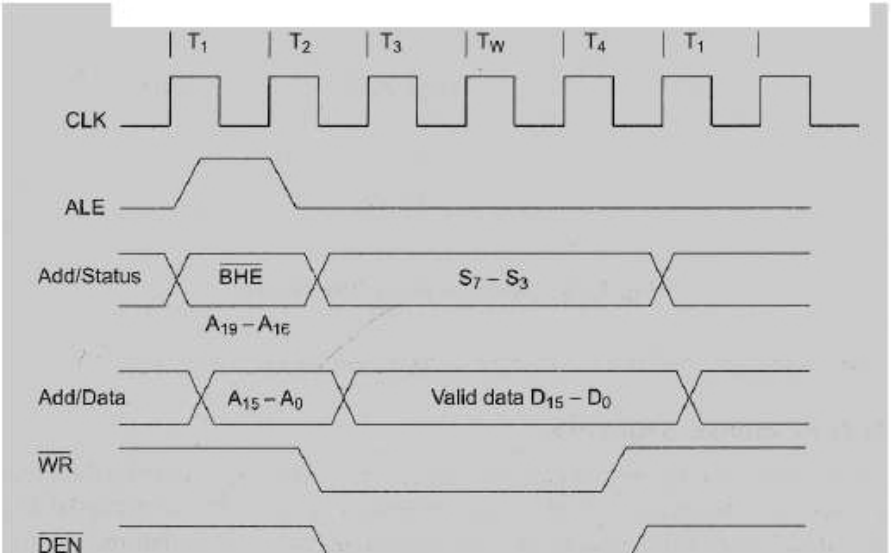


Fig.1.3. Write Cycle Timing Diagram for Minimum Mode

HOLD Response Sequence

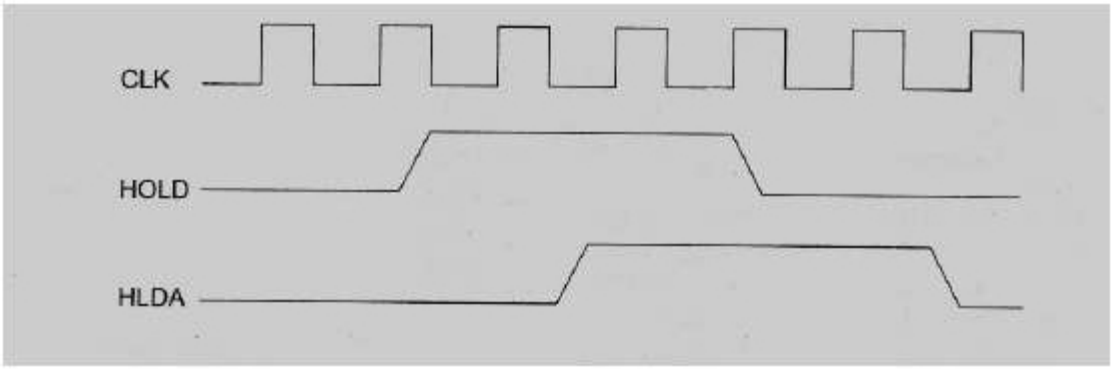


Fig 1.4. Bus Request and Bus Grant Timings in Minimum Mode System

The HOLD pin is checked at the end of the each bus cycle. If it is received active by the processor before T4 of the previous cycle or during T1 state of the current cycle, the CPU activities HLDA in the next clock cycle and for the succeeding bus cycles, the bus will be given to another requesting master The control 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 as shown in fig 1.4.

MAXIMUM MODE 8086 SYSTEM AND TIMINGS

In the maximum mode, the 8086 is operated by strapping the MN/MX* pin to ground. In this mode, the processor derives the status signals S_2*, S_1* and S_0*. Another chip called bus controller derives the control signals using this status information. In the maximum mode, there may be more than one microprocessor in the system configuration. The other components in the system are the same as in the minimum mode system. The general system organization is as shown in the fig1.1

The basic functions 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 made available by the processor on the status lines. The bus controller chip has input lines S_2*, S_1* and S_0* and CLK. These inputs to 8288 are driven by the CPU. It derives the outputs ALE, DEN*, DT/R*, 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.

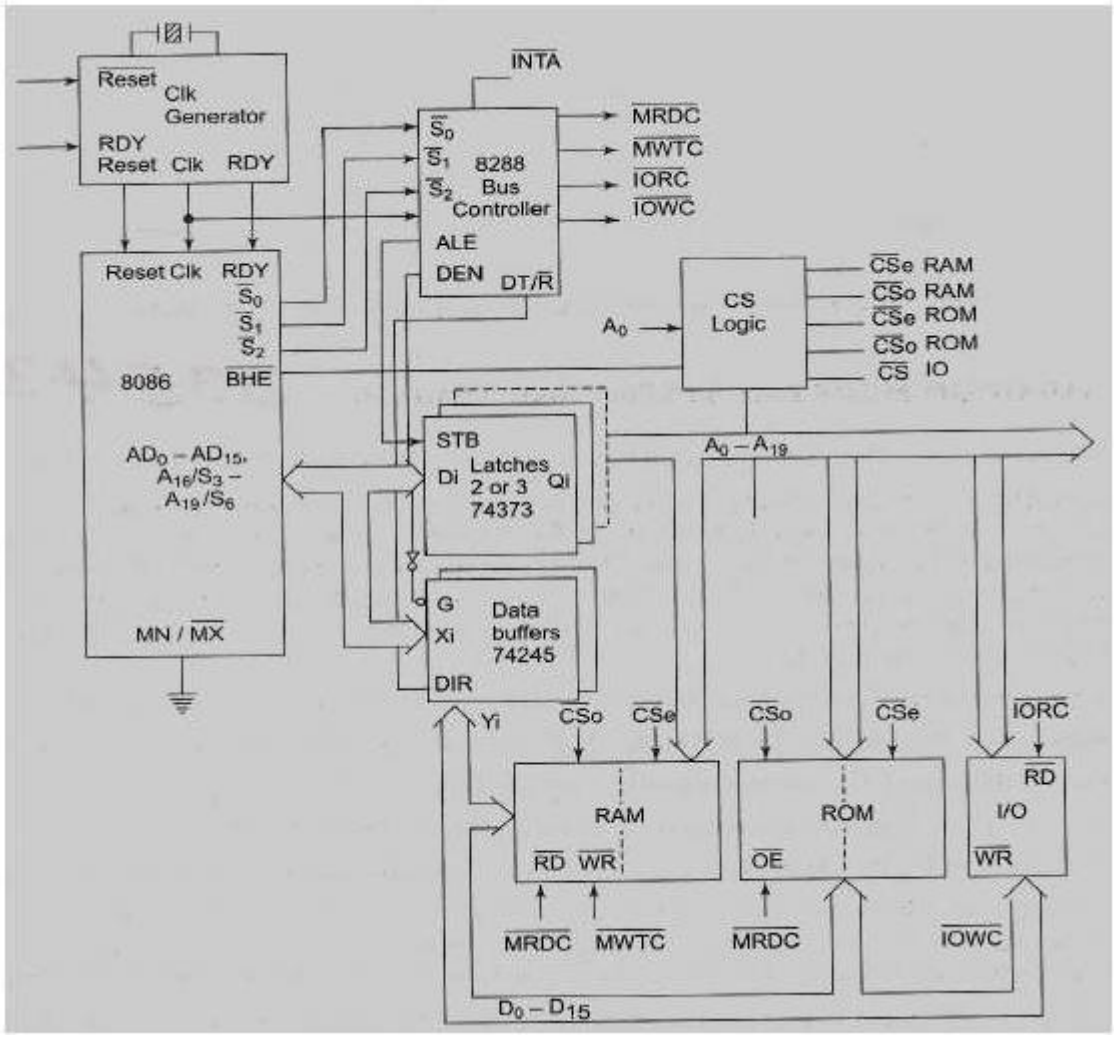


Fig 1.1 Maximum Mode 8086 System

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 cascaded 8259A; else it acts as peripheral data enable used in the multiple bus configurations. INTA* pin is 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 addressed port. The MRDC*, MWTC* are memory read command and memory write command signals respectively and may be used as memory read and write signals. All these command signals instruct 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. They also serve the same purpose, but are activated one clock cycle earlier than the IOWC* and MWTC* signals, respectively. The maximum mode system is shown in fig. 1.1.

The maximum mode system timing diagrams are also divided in two portions as read (input) and write (output) timing diagrams. The address/data and address/status timings are similar to the minimum mode. ALE is asserted in T_1, just like minimum mode. The only difference lies in the status signals used and the available control and advanced command signals. The fig. 1.2 shows the maximum mode timings for the read operation while the fig. 1.3 shows the same for the write operation.

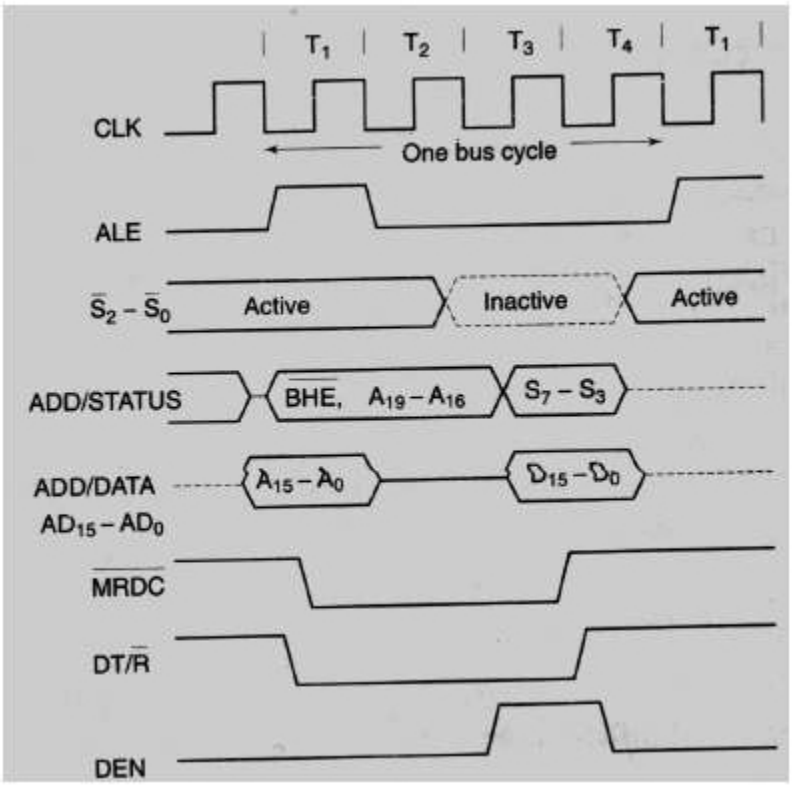
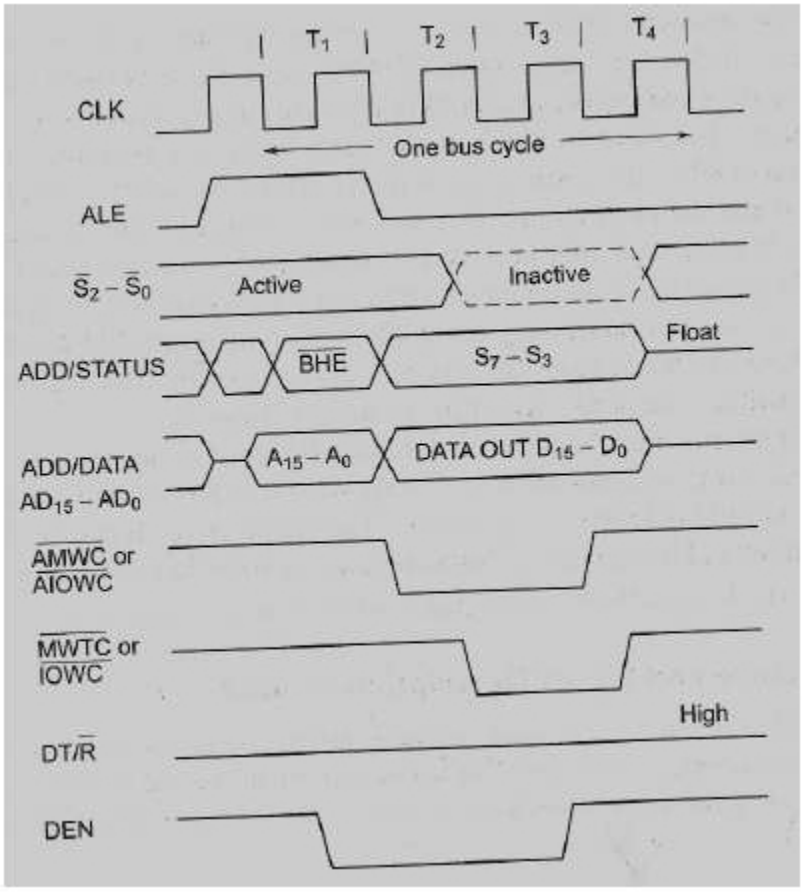


Fig. 1.2 Memory Read Timing in Maximum Mode



Timings for RQ*/GT1.3 Memory Write Timing in Maximum Mode

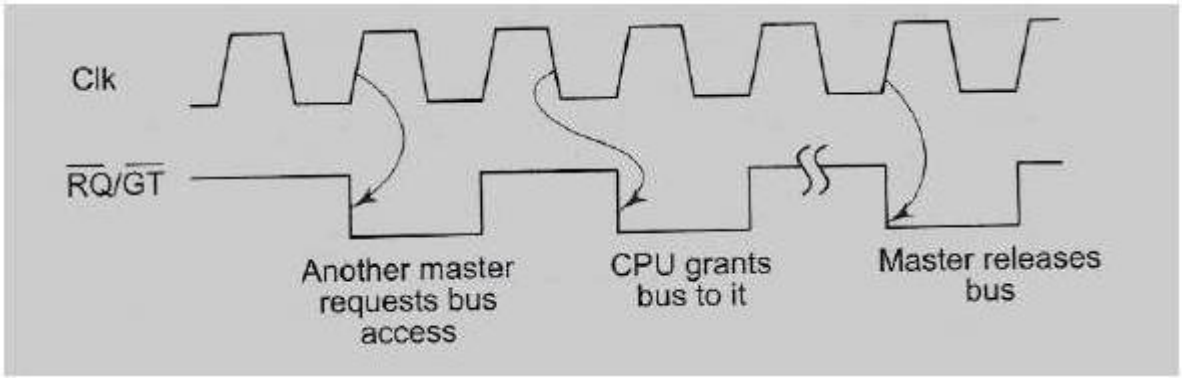


Fig1.4. RQ*/GT* Timings in Maximum Mode

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 detected the processor issues grant pulse over RQ*/GT* pin immediately during T4 (current) or T1(next) state. When the requesting master receives this pulse, it accepts the control the control of the bus. The requesting master uses the bus till it requires. When it is ready to hand over. The bus, it sends a release pulse to the processor (host) using RQ/GT pin. The sequence is shown in fig 1.4.

UNIT-IV

UNIT-IV

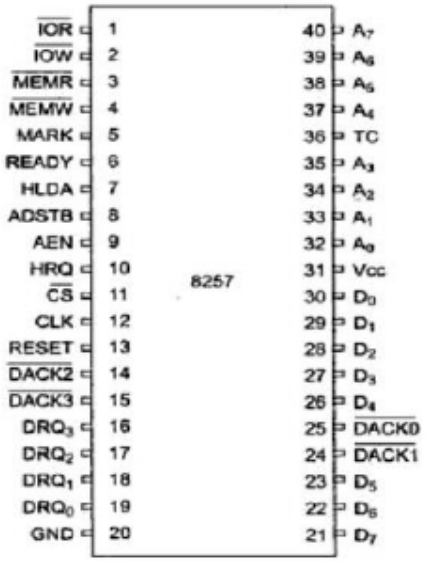
PROGRAMMABLE DMA CONTROLLER 8257

Introduction

8257 is Intel direct memory access controller (DMA) IC. Peripheral devices access memory directly by passing processor. So it is faster method of memory access over programmed data transfer for memory access. The features of 8257 are as follows:

- It is a four channel DMA controller IC.
- A maximum 16 KB data transfer can be done sequentially at a time.
- 8257 is initialized for each channel by starting address, number of bytes of data to be transferred, mode of operation.
- It executes 3 DMA cycles: DMA read, DMA write, DMA verify.
- It generates a TC signal to indicate the peripheral that the programmed numbers of data bytes have been transferred.
- It generates MARK signal to indicate the peripheral that 128 bytes have been transferred.
- It provide AEN signal that can be used to isolate CPU and other devices from the system bus.
- It operates in two modes: Master Mode, Slave Mode.

Figure 3.7 shows block diagram of 8257 DMA controller. The internal block diagram of 8257 consists of eight blocks: data bus buffer, read/write block, control logic and mode set register, priority resolver and four channel blocks.



Pin Diagram of 8257

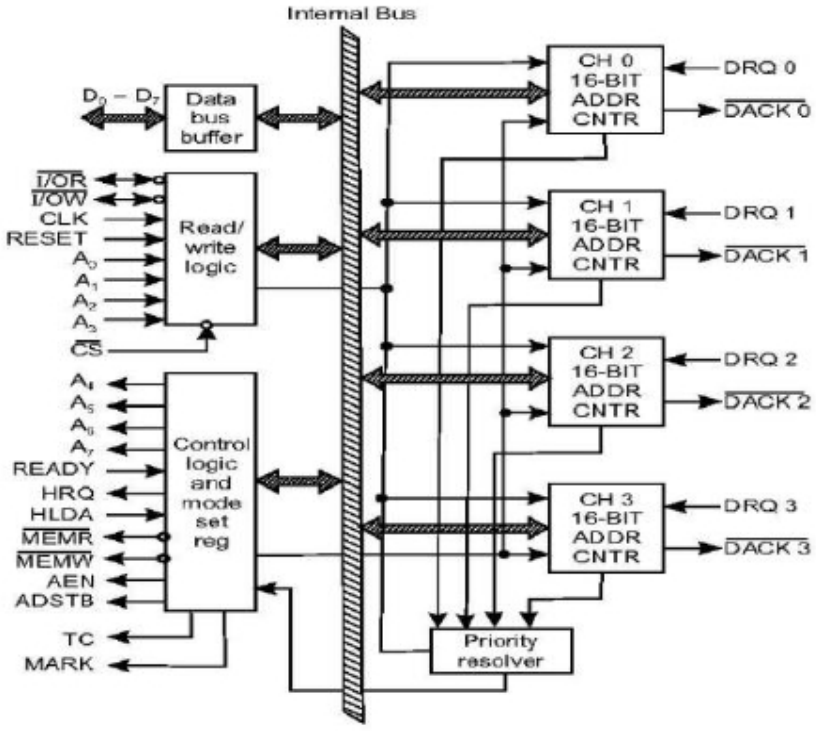
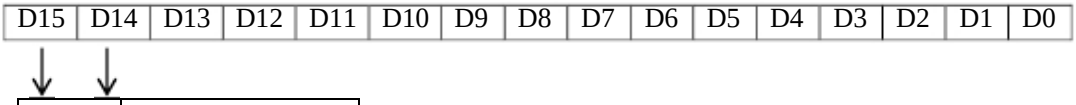


Figure 3.7: Block Diagram of 8257

- Each channel of 8257 Block diagram has two programmable 16-bit registers named as address register and count register.

- Address register is used to store the starting address of memory location for DMA data transfer.
- The address in the address register is automatically incremented after every read/write/verify transfer.
- The count register is used to count the number of byte or word transferred by DMA. The format of count register is shown in figure 3.8.



0 0	Verify transfer
0 1	Write transfer
1 0	Read transfer
1 1	Illegal

Figure 3.8: Count Register Format

- 14-bits B0-B13 is used to count value and a 2-bits is used for indicate the type of DMA transfer (Read/Write/Veri1 transfer).
- In read transfer the data is transferred from memory to I/O device.
- In write transfer the data is transferred from I/O device to memory.
- Verification operations generate the DMA addresses without generating the DMA memory and I/O control signals.

The 8257 has two eight bit registers called mode set register and status register. The format of mode set register is shown in figure 3.9.

B7	B6	B5	B4	B3	B2	B1	B0
Auto-load	TC	EW	RP	CH-3	CH-2	CH-1	CH-0

Figure 3.9: Mode Set Register Format

The use of mode set register is

- Enable/disable a channel.
- Fixed/rotating priority
- Stop DMA on terminal count.
- Extended/normal writes time.
- Auto reloading of channel-2.

- The bits B0, B1, B2, and B3 of mode set register are used to enable/disable channel -0, 1, 2 and 3 respectively. A one in these bit position will enable a particular channel and a zero will disable it.
- If the bit B4 is set to one, then the channels will have rotating priority and if it zero then the channels wilt have fixed priority.
    1. In rotating priority after servicing a channel its priority is made as lowest.
    2. In fixed priority the channel-0 has highest priority and channel-2 has lowest priority.

- If the bit B5 is set to one, then the timing of low write signals (MEMW and IOW) will be extended.
- If the bit B6 is set to one then the DMA operation is stopped at the terminal count.
- The bit B7 is used to select the auto load feature for DMA channel-2.
- When bit B7 is set to one, then the content of channel-3 count and address registers are loaded in channel-2 count and address registers respectively whenever the channel-2 reaches terminal count. When this mode is activated the number of channels available for DMA reduces from four to three.

Figure 3.10 shows the eight bit status register format of 8257.It is used to read the status of terminal count of the four channels (CH0-CH3).

B7	B6	B5	B4	B3	B2	B1	B0
0	0	0	UF	CH-3	CH-2	CH-1	CH-0

Figure 3.10: Status Register Format

- The bit B0, B1, B2, and B3 of status register indicates the terminal count status of channel-0, 1,2 and 3 respectively. A one in these bit positions indicates that the particular channel has reached terminal count.
- These status bits are cleared after a read operation by microprocessor.
- The bit B4 of status register is called update flag and a one in this bit position indicates that the channel-2 register has been reloaded from channel-3 registers in the auto load mode of operation.

PROGRAMMABLE INTERRUPT CONTROLLER: 8259

Introduction

The Intel 8259 is a Programmable Interrupt Controller (PIC) designed for use with the 8085 and 8086 microprocessors. The 8085 has only five number of hardware interrupts: TRAP,RST 7.5.RST 6.5,RST 5.5 and INTR. The 8259 can be used for applications that use more than five numbers of interrupts from multiple sources.

Features and Architecture of 8259

The main features of 8259 are listed below.

- Manage eight levels of interrupts.
- Eight interrupts are spaced at the interval of four or eight locations.
- Resolve eight levels of priority in fully nested mode, automatic rotation mode or specific rotation mode.
- Mask each interrupt individually.
- Read the status of pending interrupt, in-service interrupt, and masked interrupt.
- Accept either the level triggered or edge triggered interrupt.
- Can be expanded to 64 priority levels by cascading additional interrupts.Eight slave 8259s may be cascaded to a master 8259 to provide up to 64 priority levels.

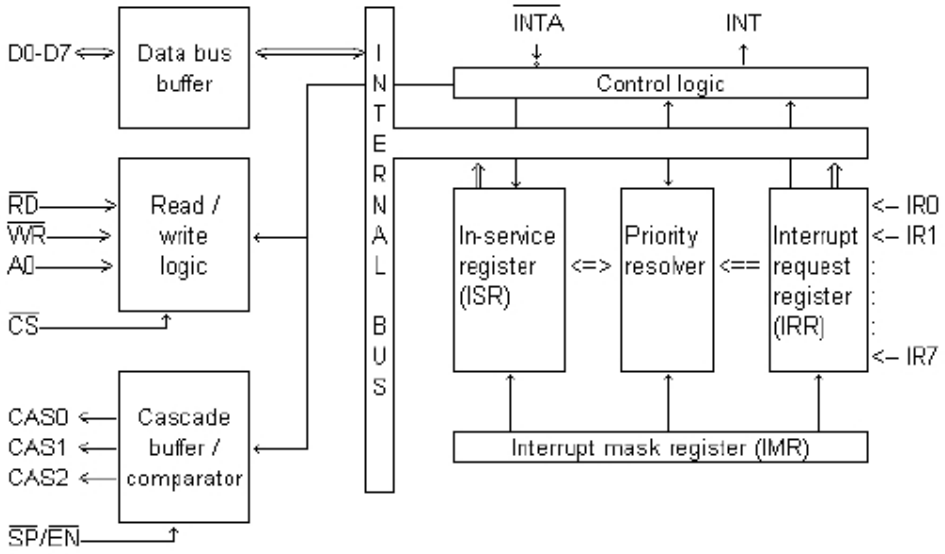
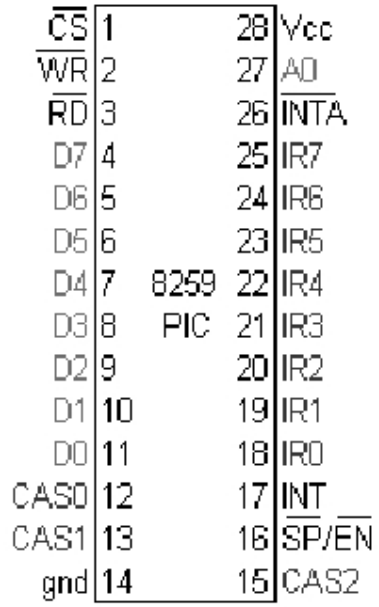


Figure 3.11: 8259 Internal Block Diagram

Figure 3.11 shows the internal block diagram of 8259. It includes eight blocks: control logic, read/write logic, data bus buffer, three registers (IRR, ISR, and IMR), priority resolver and cascade buffer.



Pin Diagram of 8259

- A0: Address input line, used to select control register.
- CAS 0 - CAS 2: Bi-directional, 3 bit cascade lines. In master mode, PIC places slave ID number on these lines. In slave mode, the PIC reads slave ID number from master on these lines. It may be regarded as slave-select.
- SP/EN (slave program / enable): In non-buffered mode, it is SP- input, used to distinguish master/slave PIC. In buffered mode, it is output line used to enable buffers
- INT: Interrupt line, connected to INTR of microprocessor.

Interrupt Operation

The Interrupt Enable (IE) flip-flop is enabled by writing the EI instruction. The 8259 is initialized by writing control words in the control register: Initialization Command Words (ICWs) and Operational Command Word (OCWs). ICWs are used to specify interrupt vector address. OCWs perform masking of interrupts, status read operation. After 8259 is initialized, the following sequence of events occurs when one interrupt request line go high:

- The IRR stores the request.
- The priority resolver checks three registers: the IRR, IMR, ISR. It resolves the priority and sets the INT pin high.
- The processor acknowledges the interrupt by sending INTA.
- After INTA is received, the priority bit in the ISR is set, the corresponding bit in IRR is reset. Then the opcode for the CALL instruction is placed on the data bus.
- When the processor decodes the CALL instruction, it places two more INTA signals on the data bus.
- When the 8259 receives the second INTA, it places the low order byte of the CALL address on the data bus. At the third INTA, it places the high-order byte on the data bus. This address is placed in the control register during initialization.
- During the third INTA, the ISR bit is reset.
- The program sequence is transferred to specified CALL address.

ICW1 (Initialization Command Word 1):

ICW1 specify lower byte of ISR CALL address.

A0	D7	D6	D5	D4	D3	D2	D1	D0
0	A7	A6	A5	1	LTIM	ADI	SNGL	IC4

D0: IC4: 0=no ICW4, 1=ICW4 required

D1: SNGL: 1=Single PIC, 0=Cascaded PIC

D2: ADI: Address interval used only in 8085, not 8086. 1=ISR's are 4 bytes apart (0200, 0204, etc)

0=ISR's are 8 byte apart (0200, 0208, etc)

D3: LTIM: level triggered interrupt mode: 1=All IR lines level triggered, 0=edge triggered

D4-D7: A5-A7: 8085 only. ISR address lower byte segment.

The lower byte is

A7	A6	A5	A4	A3	A2	A1	A0

of which A7, A6, A5 are provided by D7-D5 of ICW1 (if ADI=1), or A7, A6 are provided if ADI=0. A4-A0
(or A5-A0) is set by 8259 itself:

	ADI=1 (spacing 4 bytes)							
IRQ	A7	A6	A5	A4	A3	A2	A1	A0
IR0	A7	A6	A5	0	0	0	0	0
IR1	A7	A6	A5	0	0	1	0	0
IR2	A7	A6	A5	0	1	0	0	0
IR3	A7	A6	A5	0	1	1	0	0
IR4	A7	A6	A5	1	0	0	0	0
IR5	A7	A6	A5	1	0	1	0	0
IR6	A7	A6	A5	1	1	1	0	0
IR7	A7	A6	A5	1	1	1	0	0

	ADI=0 (spacing 8 bytes)							
IRQ	A7	A6	A5	A4	A3	A2	A1	A0
IR0	A7	A6	0	0	0	0	0	0
IR1	A7	A6	0	0	1	0	0	0
IR2	A7	A6	0	1	0	0	0	0
IR3	A7	A6	0	1	1	0	0	0
IR4	A7	A6	1	0	0	0	0	0
IR5	A7	A6	1	0	1	0	0	0
IR6	A7	A6	1	1	0	0	0	0
IR7	A7	A6	1	1	1	0	0	0

ICW2 (Initialization Command Word 2):

ICW2 specify higher byte of ISR CALL address (8085), or 8 bit vector address (8086).

A0	D7	D6	D5	D4	D3	D2	D1	D0
1	A15	A14	A13	A12	A11	A10	A9	A8

ICW3 (Initialization Command Word 3):

ICW3 is required only if several 8259's are used in cascaded form.ICW3 can operate in master mode and slave mode.

		D7	D6	D5	D4	D3	D2	D1	D0
A0	Master	S7	S6	S5	S4	S3	S2	S1	S0
1	Slave	0	0	0	0	0	ID3	ID2	ID1

- Master mode: 1 indicates slave is present on that interrupt, 0 indicates direct interrupt
- Slave mode: ID3-ID2-ID1 is the slave ID number. Slave 4 on IR4 has ICW3=04h (0000 0100)

ICW4 (Initialization Command Word 4)

A0	D7	D6	D5	D4	D3	D2	D1	D0
1	0	0	0	SFNM	BUF	M/S	AEOI	Mode

- SFNM: 1=Special Fully Nested Mode, 0=FNM
- M/S: 1=Master, 0=Slave
- AEOI: 1=Auto End of Interrupt, 0=Normal
- Mode: 0=8085, 1=8086

OCW1 (Operational Command Word 1)

OCW1 specify masking of interrupts.

A0	D7	D6	D5	D4	D3	D2	D1	D0
1	M7	M6	M5	M4	M3	M2	M1	M0

IRn is masked by setting Mn to 1; mask cleared by setting Mn to 0 (n=0..7)

OCW2 (Operational Command Word 2)

OCW2 control the rotate and End of interrupt modes.

A0	D7	D6	D5	D4	D3	D2	D1	D0
1	R	SL	EOI	0	0	L3	L2	L1

	R	SL	EOI	Action
EOI	0	0	1	Non specific EOI (L3L2L1=000)
	0	1	1	Specific EOI command (Interrupt to clear given by L3L2L1)
Auto rotation of priorities (L3L2L1=000)	1	0	1	Rotate priorities on non-specific EOI
	1	0	0	Rotate priorities in auto EOI mode set
	0	0	0	Rotate priorities in auto EOI mode clear
Specific rotation of priorities (Lowest priority ISR=L3L2L1)	1	1	1	Rotate priority on specific EOI command (resets current ISR bit)
	1	1	0	Set priority (does not reset current ISR bit)
	0	1	0	No operation

OCW3 (Operational Command Word 3)

OCW3 read the status of register, set/reset special mask and polled modes.

A0	D7	D6	D5	D4	D3	D2	D1	D0
1	D7	ESMM	SMM	0	1	MODE	RIR	RIS

ESMM	SMM	Effect
0	X	No effect
1	0	Reset special mask
1	1	Set special mask

Priority Modes:

- Fully Nested Mode: All IRs are arranged from highest to lowest priority.
  IR0=highest priority, IR7=lowest priority
- Automatic Rotation Mode: A device after being serviced receives the lowest priority.
- Specific Rotation Mode: User selects any IR for lowest priority.

PROGRAMMABLE INTERVAL TIMER: 8253

INTRODUCTION

In software programming of 8085, it has been shown that a delay subroutine can be programmed to introduce a predefined time delay. The delay is achieved by decrementing a count value in a register using instructions. The disadvantage of this software approach is that the processor is locked in the delay loop and the precious processor time is wasted in just counting. This disadvantage can be overcome by using the hardware timer and interrupts. IC 555 can be used to generate the timing signals, but only at a fixed time interval. This can't be easily interfaced with the microprocessor. So, Intel has produced programmable timer devices namely IC 8253 and IC 8254. These devices can be programmed to generate different types of delay signals and also count external signals. Other counter/timer functions that are also common to be implemented with the 8253 are Programmable frequency square wave Generator, Event Counter, Real Time Clock, Digital One-Shot and Complex Motor Controller.

Features of 8253

Timer ICs 8253 and 8254 are manufactured by Intel with the similar operating functions. 8254 can be operated at frequency of up to 8MHz whereas 8253 can be operated only up to a maximum frequency of 2 MHZ.

- Generation of accurate time delay
- Three independent 16-bit down counters.
- Six different programmable operating modes

- Timer or counter operation.
- Can count in binary or BCD
- Can be used to interrupt the processor.
- Single +5V supply
- Can operate from DC to 2 MHZ.

Block Diagram of 8253

Figure 3.12 is the block diagram of 8253. It include three counters (counter 0, 1and 2), a data bus buffer, Read/write control logic and control register. Each counter has two input signals-clock (CLK) and GATE- and one output signal-OUT.

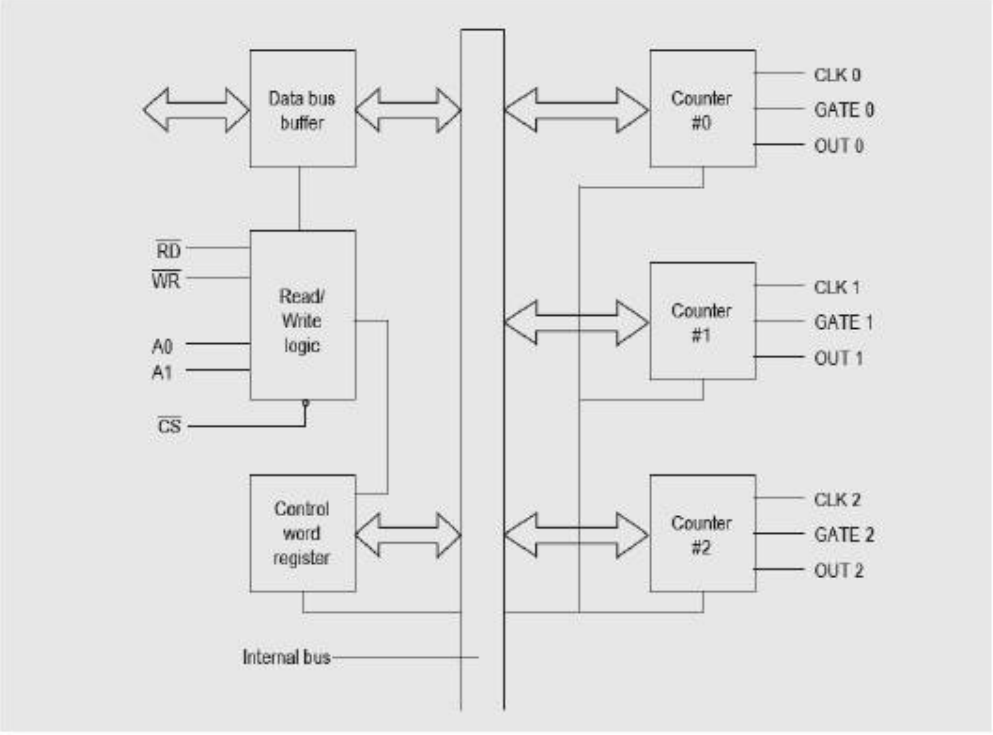
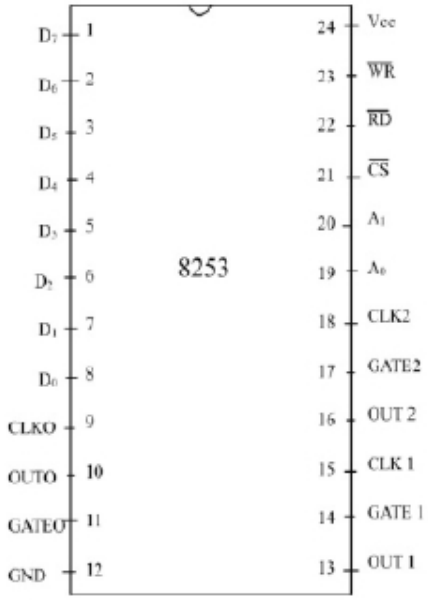


Figure 3.12: Block Diagram of 8253



Pin Diagram of 8253

The control word register and counters are selected according to the signals on lines A0 and A1 as follows:

A1	A0	Selection
0	0	Counter0
0	1	Counter1
1	0	Counter2
1	1	Control Register

$\overline{CS}$	$\overline{RD}$	$\overline{WR}$	A_1	A_0	Operation
0	1	0	0	0	Load count Value in Counter 0
0	1	0	0	1	Load count Value in Counter 1
0	1	0	1	0	Load count Value in Counter 2
0	1	0	1	1	Write Control Word
0	0	1	0	0	Read Counter value from Counter 0
0	0	1	0	1	Read Counter value from Counter 0
0	0	1	1	0	Read Counter value from Counter 0
0	0	1	1	1	No operation
0	1	1	X	X	No operation
1	X	X	X	X	Disable chip

Control Word Register

The control word format is shown in figure 3.13. This register is accessed when A0 and A1 at logic 1. It is used to write a command word which specifies the counter to be used, its mode, and either a Read or write operation.

Bit position	D7	D6	D5	D4	D3	D2	D1	D0
Name	SC1	SC0	RL1	RL0	M2	M1	M0	Binary /BCD
Explanation	Select Counter 0 0 Counter 0 0 1 Counter 1 1 0 Counter 2 1 1 Illegal		Read/ Load option 0 0 Latch count 0 1 Load LS byte only 1 0 Load MS byte only 1 1 Load LSB and then MSB		Mode selection bits 0 0 0 Mode 0 0 0 1 Mode 1 X 1 0 Mode 2 X 1 1 Mode 3 1 0 0 Mode 4 1 0 1 Mode 5			0-Binay counter 1-BCD counter

Figure 3.13: 8253 Control Word Format

Mode:

Figure 3.14 shows the operating modes of 8253. It operates in 6 different modes: mode0, mode1, mode2, mode3, mode4, and mode5.

M2	M1	M0	Operating modes		GATE control	Reloading of Count value
0	0	0	Mode 0	**Interrupt on Terminal Count**	0- Disables counting 1- Enables counting	No
0	0	1	Mode 1	**Programmable One-Shot.**	0 to 1 transition initiates counting	Yes, if triggered
X	1	0	Mode 2	**Rate Generator. Divide by N counter**	0- Disables counting 1- Enables counting 0 to 1 transition will reload counter and initiate counting	Yes
X	1	1	Mode 3	**Square Wave Rate Generator**	0- Disables counting 1- Enables counting 0 to 1 transition will reload counter and initiate counting	Yes
1	0	0	Mode 4	**Software Triggered Strobe**	0- Disables counting 1- Enables counting	No
1	0	1	Mode 5	**Hardware Triggered Strobe**	0 to 1 transition initiates counting	Yes if Gate input goes from 0 to1.

Figure 3.14: Operating Mode of 8253

Programming the 8253

The 8253 can be programmed to provide various types of outputs through Write operation, or to check a count while counting through Read operations.

Write operation

To initialize a counter, the following steps are necessary

- Write a control word into the control register.
- Load the low –order byte of a count in the counter register.
- Load the high –order byte of a count in the counter register.

Read Operation

In event counters application, it is necessary to read count value in progress. This can be done by either two methods. One method involves reading a count after stopping the counter to be read. The second method involves reading a count while the count is in progress.

In the first method, counting is stopped by controlling the gate input or the clock input of the selected counter, and two I/O read operation are performed by CPU. The first I/O operation reads the lower byte and the second I/O operation reads the higher byte. In the second method an appropriate control word is written into the control register to latch a count in output latch, and two I/O read operation are performed by the CPU.

Mode 0: Interrupt on Terminal Count

In this mode, initially the OUT is low. Once a count is loaded in the register, the counter is decremented every cycle. When count reaches zero, the OUT goes high. This can be used as an interrupt. The OUT remains until a new count or a command is loaded.

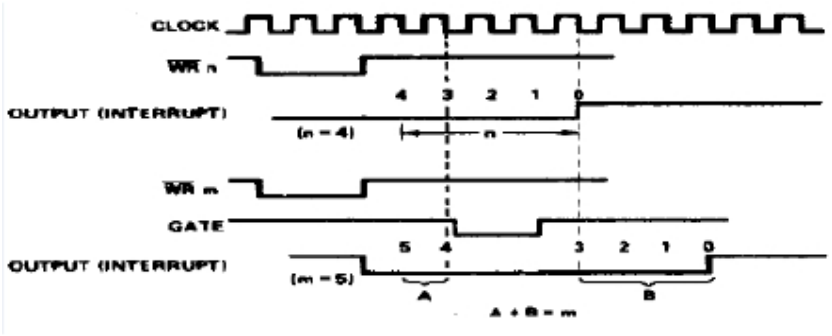


Figure 3.15: Mode 0: Interrupt on Terminal Count

Mode 1: Hardware-Triggered One Shot

In this mode, the OUT is initially high. When the Gate is triggered, the OUT goes low, and at the end of count, the OUT goes high again, generating one shot pulse.

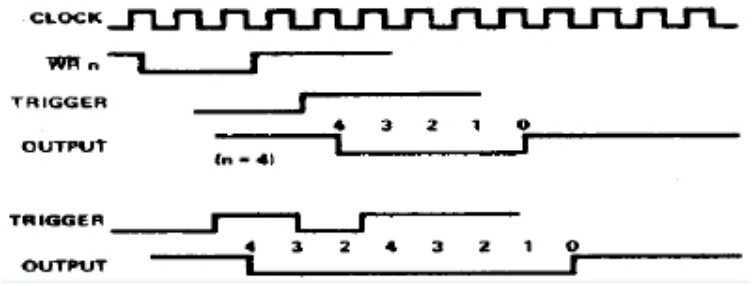


Figure 3.16: Mode 1: Programmable one shot

Mode 2: Rate Generator

This mode is used to generate a pulse equal to the clock period at a given interval. When a count is loaded, the OUT stays high until the count reaches 1, and then the count goes low for one clock period. The count is reloaded automatically, and the pulse is generated continuously. The count = 1 is illegal in this mode.

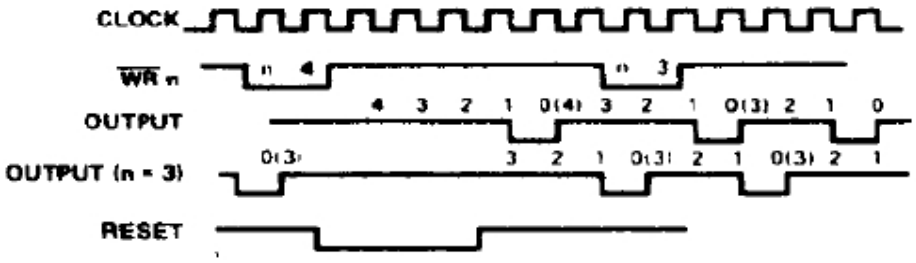


Figure 3.17: Mode 2: Rate Generator.

Mode 3: Square Wave Generator

In this mode, when count is loaded, the OUT is high. The count is decremented by two at every clock cycle. When it reaches zero, the OUT goes low, and the count is reloaded again. This is repeated continuously to generate a square wave with period equal to period of count generated. The frequency of the square wave is equal to the frequency of the clock divided by count.

- High for N/2 counts, and low for N/2 counts, if count N is even.
- High for (N+1)/2 counts, and low for (N-1)/2 counts, if N is odd.

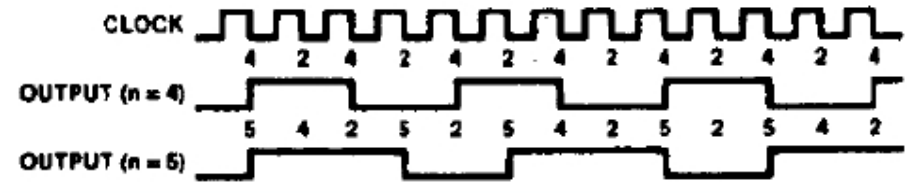


Figure 3.18: Mode 3: Square Wave Generator

Mode 4: Software Triggered Strobe

In this mode the OUT is initially high. It goes low for one clock period at the end of count. The count is reloaded for subsequent outputs.

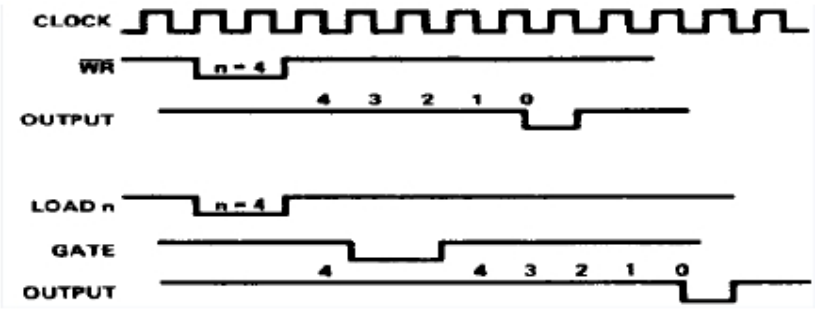


Figure 3.19: Mode 4: Software Triggered Strobe

Mode 5: Hardware Triggered Strobe

This mode is similar to mode 4, except that it is triggered by the rising edge of GATE input. Initially the OUT is low, and when the Gate pulse is triggered from low to high, the count begins. At the end of count, the OUT goes low for one clock period.

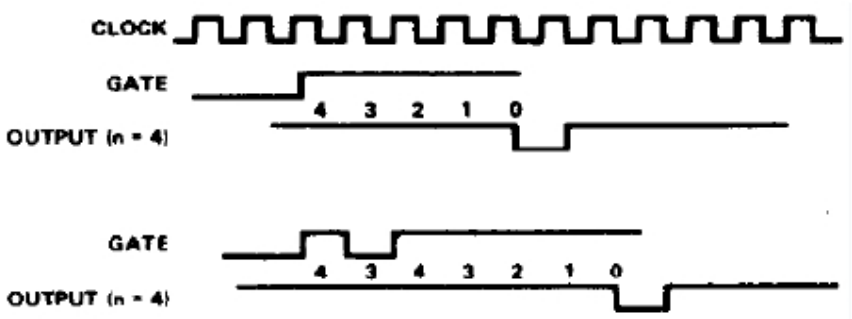
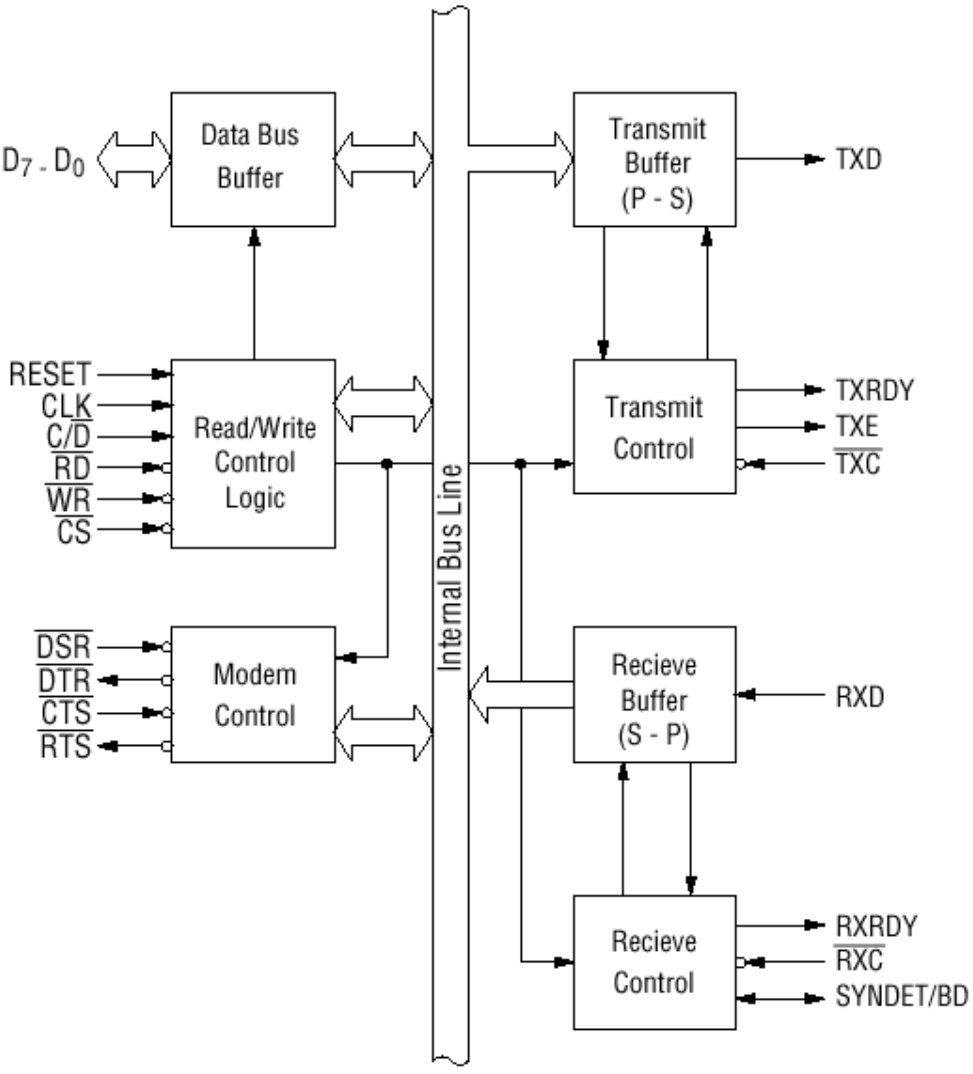


Figure 3.20: Mode5: Hardware Triggered Strobe

8251 UNIVERSAL SYNCHRONOUS ASYNCHRONOUS RECEIVER TRANSMITTER (USART)

The 8251 is a USART (Universal Synchronous Asynchronous Receiver Transmitter) for serial data communication. As a peripheral device of a microcomputer system, the 8251 receives parallel data from the CPU and transmits serial data after conversion. This device also receives serial data from the outside and transmits parallel data to the CPU after conversion.



Block diagram of the 8251 USART (Universal Synchronous Asynchronous Receiver Transmitter)

The 8251 functional configuration is programed by software. Operation between the 8251 and a CPU is executed by program control. Table 1 shows the operation between a CPU and the device.

$\overline{CS}$	C/$\overline{D}$	$\overline{RD}$	$\overline{WR}$	
1	×	×	×	Data Bus 3-State
0	×	1	1	Data Bus 3-State
0	1	0	1	Status → CPU
0	1	1	0	Control Word ← CPU
0	0	0	1	Data → CPU
0	0	1	0	Data ← CPU

Table 1 Operation between a CPU and 8251

Control Words

There are two types of control word.

1. Mode instruction (setting of function)

2. Command (setting of operation)

1) Mode Instruction

Mode instruction is used for setting the function of the 8251. Mode instruction will be in "wait for write" at either internal reset or external reset. That is, the writing of a control word after resetting will be recognized as a "mode instruction."

Items set by mode instruction are as follows:

• Synchronous/asynchronous mode

• Stop bit length (asynchronous mode)

• Character length

• Parity bit

• Baud rate factor (asynchronous mode)

• Internal/external synchronization (synchronous mode)

• Number of synchronous characters (Synchronous mode)

The bit configuration of mode instruction is shown in Figures 2 and 3. In the case of synchronous mode, it is necessary to write one-or two byte sync characters. If sync characters were written, a function will be set because the writing of sync characters constitutes part of mode instruction.

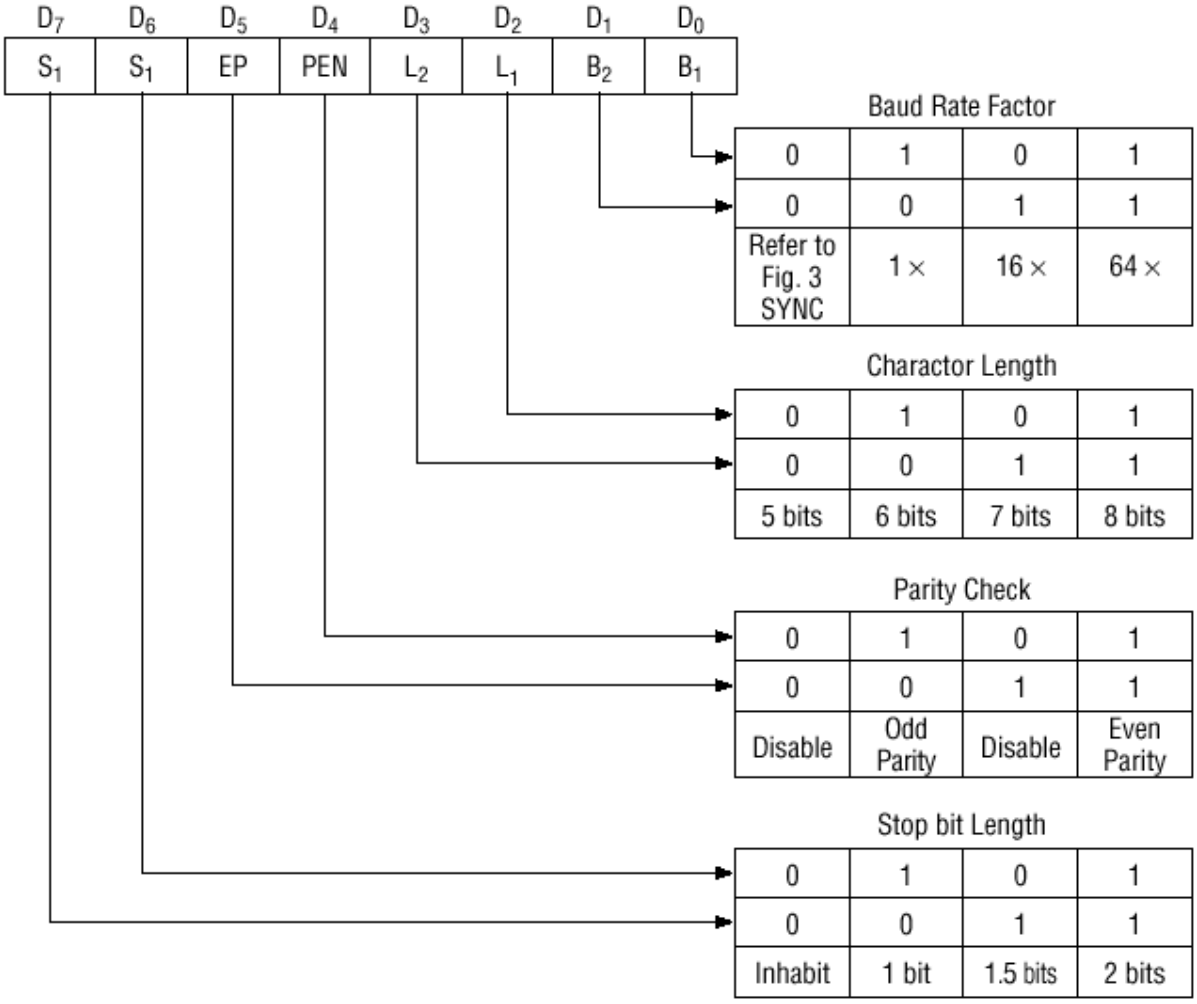


Fig. 2 Bit Configuration of Mode Instruction (Asynchronous)

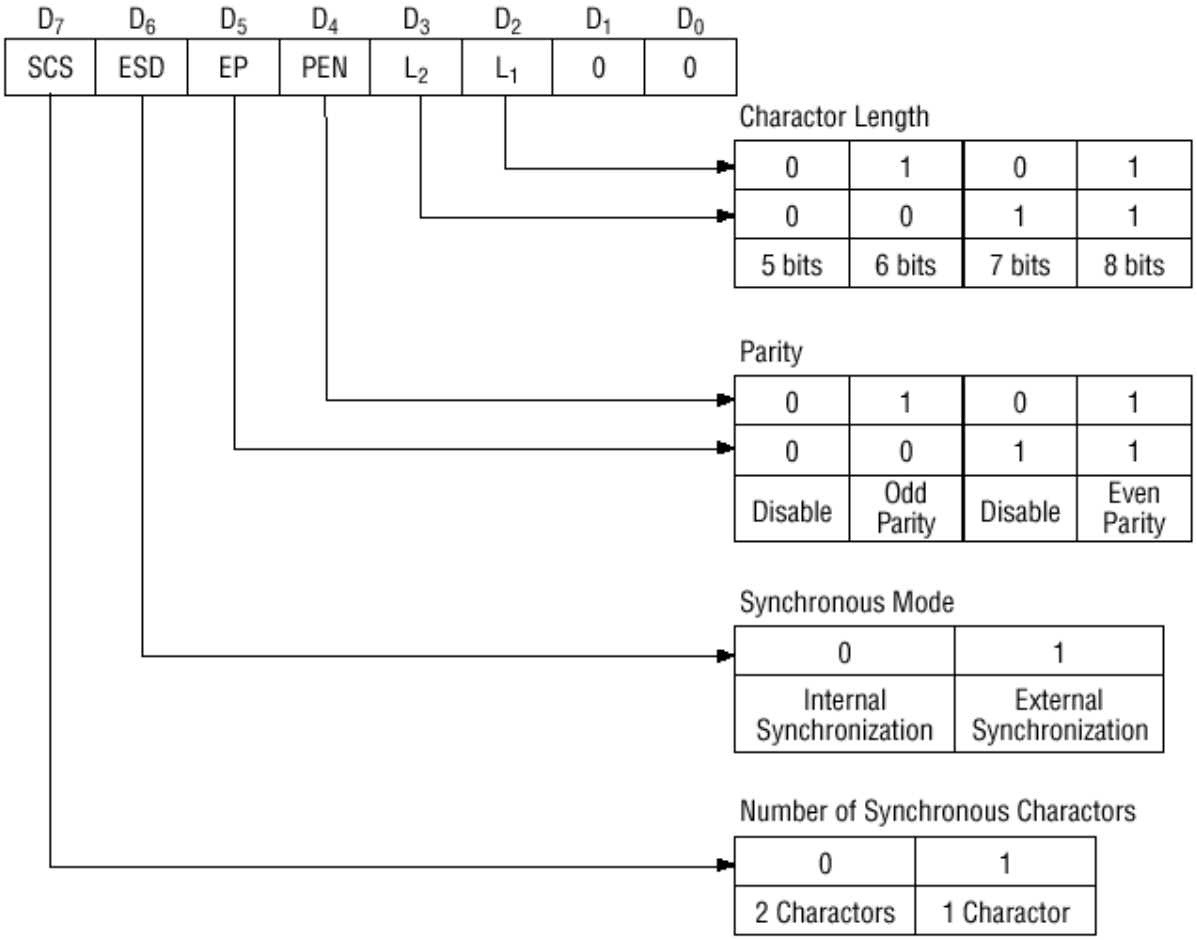


Fig. 3 Bit Configuration of Mode Instruction (Synchronous)

2) Command

Command is used for setting the operation of the 8251. It is possible to write a command whenever necessary after writing a mode instruction and sync characters.

Items to be set by command are as follows:

• Transmit Enable/Disable

• Receive Enable/Disable

• DTR, RTS Output of data.

• Resetting of error flag.

• Sending to break characters

• Internal resetting

• Hunt mode (synchronous mode)

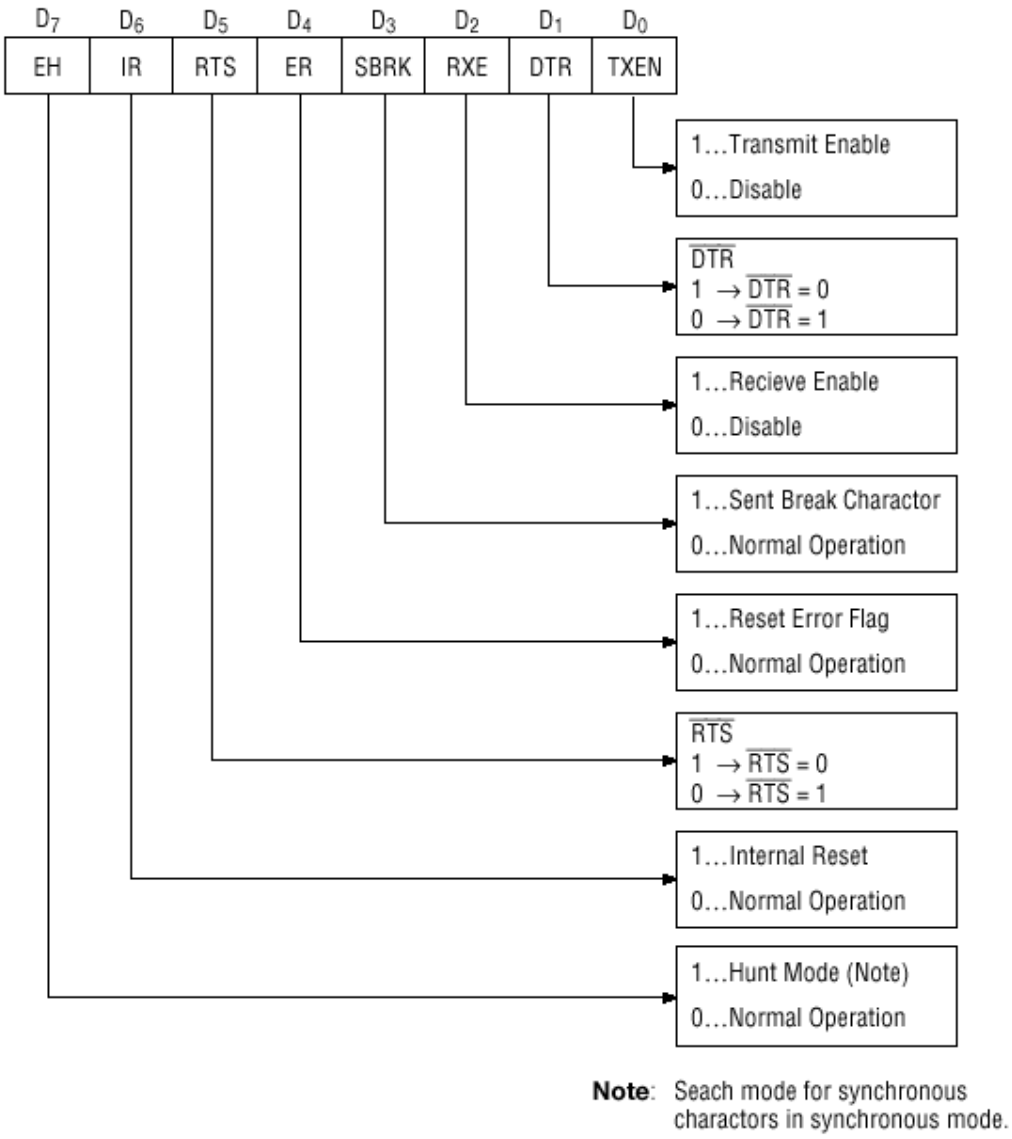


Fig. 4 Bit Configuration of Command

Status Word

It is possible to see the internal status of the 8251 by reading a status word. The bit configuration of status word is shown in Fig. 5.

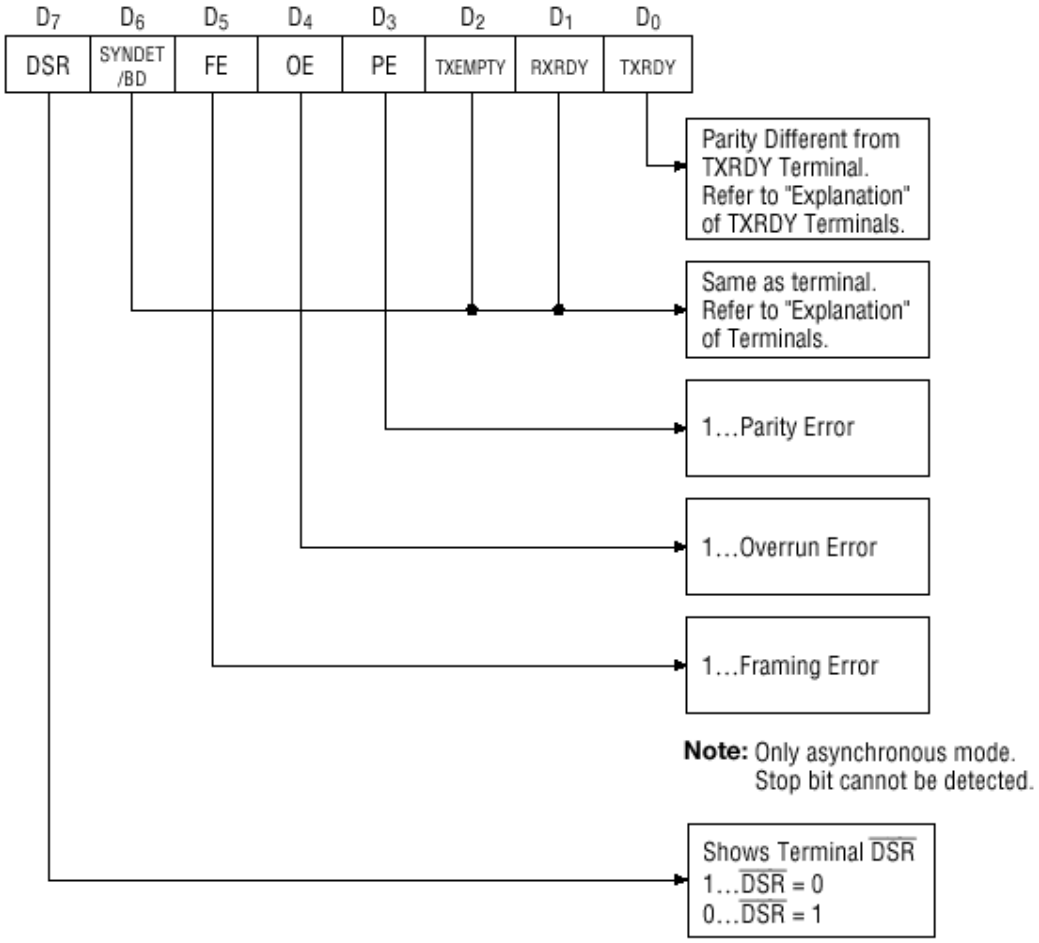


Fig. 5 Bit Configuration of Status Word

Pin Description

D 0 to D 7 (l/O terminal)

This is bidirectional data bus which receive control words and transmits data from the CPU and sends status words and received data to CPU.

RESET (Input terminal)

A "High" on this input forces the 8251 into "reset status." The device waits for the writing of "mode instruction." The min. reset width is six clock inputs during the operating status of CLK.

CLK (Input terminal)

CLK signal is used to generate internal device timing. CLK signal is independent of RXC or TXC. However, the frequency of CLK must be greater than 30 times the RXC and TXC at Synchronous mode and Asynchronous "x1" mode, and must be greater than 5 times at Asynchronous "x16" and "x64" mode.

WR (Input terminal)

This is the "active low" input terminal which receives a signal for writing transmit data and control words from the CPU into the 8251.

RD (Input terminal)

This is the "active low" input terminal which receives a signal for reading receive data and status words from the 8251.

C/D (Input terminal)

This is an input terminal which receives a signal for selecting data or command words and status words when the 8251 is accessed by the CPU. If C/D = low, data will be accessed. If C/D = high, command word or status word will be accessed.

CS (Input terminal)

This is the "active low" input terminal which selects the 8251 at low level when the CPU accesses. Note: The device won't be in "standby status"; only setting CS = High.

TXD (output terminal)

This is an output terminal for transmitting data from which serial-converted data is sent out. The device is in "mark status" (high level) after resetting or during a status when transmit is disabled. It is also possible to set the device in "break status" (low level) by a command.

TXRDY (output terminal)

This is an output terminal which indicates that the 8251is ready to accept a transmitted data character. But the terminal is always at low level if CTS = high or the device was set in "TX disable status" by a command. Note: TXRDY status word indicates that transmit data character is receivable, regardless of CTS or command. If the CPU writes a data character, TXRDY will be reset by the leading edge or WR signal.

TXEMPTY (Output terminal)

This is an output terminal which indicates that the 8251 has transmitted all the characters and had no data character. In "synchronous mode," the terminal is at high level, if transmit data characters are no longer remaining and sync characters are automatically transmitted. If the CPU writes a data character, TXEMPTY will be reset by the leading edge of WR signal. Note : As the transmitter is disabled by setting CTS "High" or command, data written before disable will be sent out. Then TXD and TXEMPTY will be "High". Even if

a data is written after disable, that data is not sent out and TXE will be "High".After the transmitter is enabled, it sent out. (Refer to Timing Chart of Transmitter Control and Flag Timing)

TXC (Input terminal)

This is a clock input signal which determines the transfer speed of transmitted data. In "synchronous mode," the baud rate will be the same as the frequency of TXC. In "asynchronous mode", it is possible to select the baud rate factor by mode instruction. It can be 1, 1/16 or 1/64 the TXC. The falling edge of TXC sifts the serial data out of the 8251.

RXD (input terminal)

This is a terminal which receives serial data.

RXRDY (Output terminal)

This is a terminal which indicates that the 8251 contains a character that is ready to READ. If the CPU reads a data character, RXRDY will be reset by the leading edge of RD signal. Unless the CPU reads a data character before the next one is received completely, the preceding data will be lost. In such a case, an overrun error flag status word will be set.

RXC (Input terminal)

This is a clock input signal which determines the transfer speed of received data. In "synchronous mode," the baud rate is the same as the frequency of RXC. In "asynchronous mode," it is possible to select the baud rate factor by mode instruction. It can be 1, 1/16, 1/64 the RXC.

SYNDET/BD (Input or output terminal)

This is a terminal whose function changes according to mode. In "internal synchronous mode." this terminal is at high level, if sync characters are received and synchronized. If a status word is read, the terminal will be reset. In "external synchronous mode, "this is an input terminal. A "High" on this input forces the 8251 to start receiving data characters.

In "asynchronous mode," this is an output terminal which generates "high level"output upon the detection of a "break" character if receiver data contains a "low-level" space between the stop bits of two continuous characters. The terminal will be reset, if RXD is at high level. After Reset is active, the terminal will be output at low level.

DSR (Input terminal)

This is an input port for MODEM interface. The input status of the terminal can be recognized by the CPU reading status words.

DTR (Output terminal)

This is an output port for MODEM interface. It is possible to set the status of DTR by a command.

CTS (Input terminal)

This is an input terminal for MODEM interface which is used for controlling a transmit circuit. The terminal controls data transmission if the device is set in "TX Enable" status by a command. Data is transmitable if the terminal is at low level.

RTS (Output terminal)

This is an output port for MODEM interface. It is possible to set the status RTS by a command.

UNIT-V

UNIT - V MEMORY AND IO INTERFACING

SEMICONDUCTOR MEMORY INTERFACING

Semiconductor memories are of two types, viz. RAM (Random Access Memory) and ROM (Read Only Memory).

The semiconductor RAMs are of broadly two types-static RAM and dynamic RAM. The semiconductor memories are organized as two dimensional arrays of memory locations. For example, 4Kx8 or 4K byte memory contains 4096 locations, where each location contains 8-bit data and only one of the 4096 locations can be selected at a time.

The general procedure of static memory interfacing with 8086 is briefly described as follows:

1. Arrange the available memory chips so as to obtain 16-bit data bus width. The upper 8-bit bank is called "odd address memory bank" and the lower 8-bit bank is called "even address memory bank".

2. Connect available memory address lines of memory chips with those of the microprocessor and also connect the memory and inputs to the corresponding processor control signals. Connect 16-bit data bus of the memory bank with that of the microprocessor 8086.

3. The remaining address lines of the microprocessor, and A_0 are used for decoding the required chip select signals for the odd and even memory banks. The of memory is derived from the O/P of the decoding circuit.

Relation between number of address pins and memory capacity

Number of address pins	Memory capacity			Range of address in hexa
	in decimal	in kilo	in hexa	
10	$2^{10} = 1024$	1 k	400	000 to 3FF
11	$2^{11} = 2 \times 2^{10} = 2048$	2 k	800	000 to 7FF
12	$2^{12} = 2^2 \times 2^{10} = 4 \times 2^{10} = 4096$	4 k	1000	000 to FFF
13	$2^{13} = 2^3 \times 2^{10} = 8 \times 2^{10} = 8192$	8 k	2000	0000 to 1FFF
14	$2^{14} = 2^4 \times 2^{10} = 16 \times 2^{10} = 16384$	16 k	4000	0000 to 3FFF
15	$2^{15} = 2^5 \times 2^{10} = 32 \times 2^{10} = 32768$	32 k	8000	0000 to 7FFF
16	$2^{16} = 2^6 \times 2^{10} = 64 \times 2^{10} = 65536$	64 k	10000	0000 to FFFF
17	$2^{17} = 2^7 \times 2^{10} = 128 \times 2^{10} = 131072$	128 k	20000	00000 to 1FFFF
18	$2^{18} = 2^8 \times 2^{10} = 256 \times 2^{10} = 262144$	256 k	40000	00000 to 3FFFF
19	$2^{19} = 2^9 \times 2^{10} = 512 \times 2^{10} = 524288$	512 k	80000	00000 to 7FFFF
20	$2^{20} = 2^{10} \times 2^{10} = 1024 \times 2^{10} = 1048576$	1024 k=1M	100000	00000 to FFFFF

Problem 1

Interface two 4Kx8 EPROM and two 4Kx8 RAM chips with 8086. Select suitable maps.

Solution:

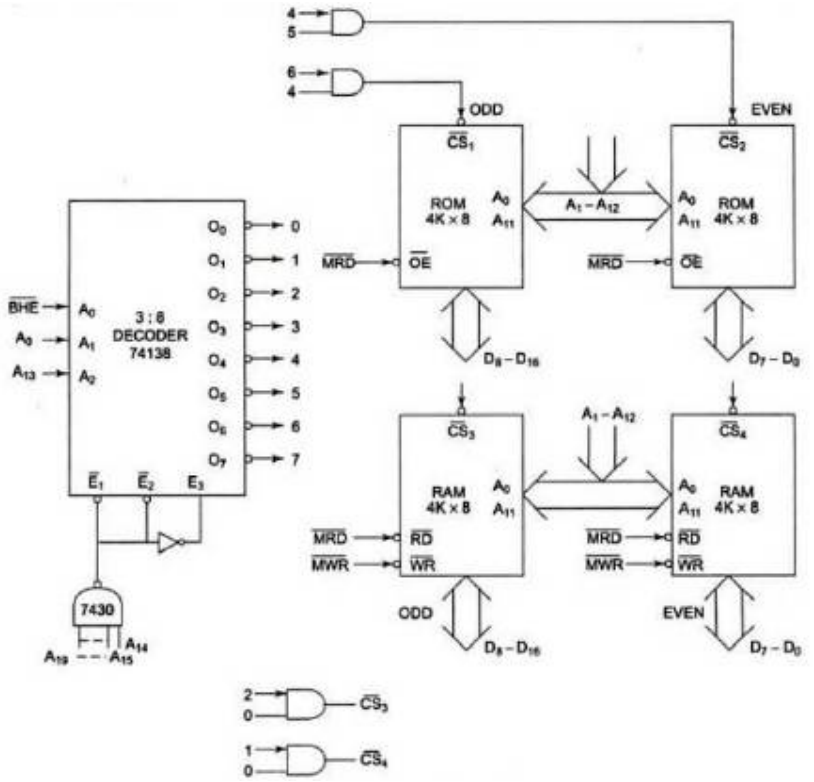
We know that, after reset, the IP and CS are initialized to form address FFFF0H. Hence, this address must lie in the EPROM. The address of RAM may be selected anywhere in the 1MB address space of 8086, but we will select the RAM address such that the address map of the system is continuous.

Courtesy: *Advanced Microprocessors and Peripherals, A.K. Ray, Tata McGraw Hill*

Memory Map Table

Address	A19	A18	A17	A16	A15	A14	A13	A12	A11	A10	A09	A08	A07	A06	A05	A04	A03	A02	A01	A00
FFFFFH	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
EPROM								8K X 8												
FE000H	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0
FDFFFH	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1
RAM								8K X 8												
FC000H	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Total 8K bytes of EPROM need 13 address lines A0-A12 (since z13 = 8K).
Address lines A13 - A19 are used for decoding to generate the chip select.
The signal goes low when a transfer is at odd address or higher byte of data is to be accessed.
Let us assume that the latched address, and demultiplexed data lines are readily available for interfacing.
The memory system in this problem contains in total four 4K x 8 memory chips.
The two 4K x 8 chips of RAM and ROM are arranged in parallel to obtain 16-bit data bus width. If A0 is 0, i.e., the address is even and is in RAM, then the lower RAM chip is selected indicating 8-bit transfer at an even address. If A0 is i.e., the address is odd and is in RAM, the goes low, the upper RAM chip is selected, further indicating that the 8-bit transfer is at an odd address. If the selected addresses are in ROM, the respective ROM chips are selected. If at a time A0 and both are 0, both the RAM or ROM chips are selected, i.e., the data transfer is of 16 bits. The selection of chips here takes place as shown in table below.



Courtesy: *Advanced Microprocessors and Peripherals, A.K. Ray, Tata McGraw Hill*

Memory Chip Selection Table:

Decoder I/P --> Address/ -->	A2 A13	A1 A0	A0	Selection/ Comment
Word transfer on D0 - D15	0	0	0	Even and odd address in RAM
Byte transfer on D7 - D0	0	0	1	Only even address in RAM
Byte transfer on D8 - D15	0	1	0	Only odd address in RAM
Word transfer on D0 - D15	1	0	0	Even and odd address in RAM
Byte transfer on D7 - D0	1	0	1	Only even address in RAM
Byte transfer on D8 - D15	1	1	0	Only odd address in ROM

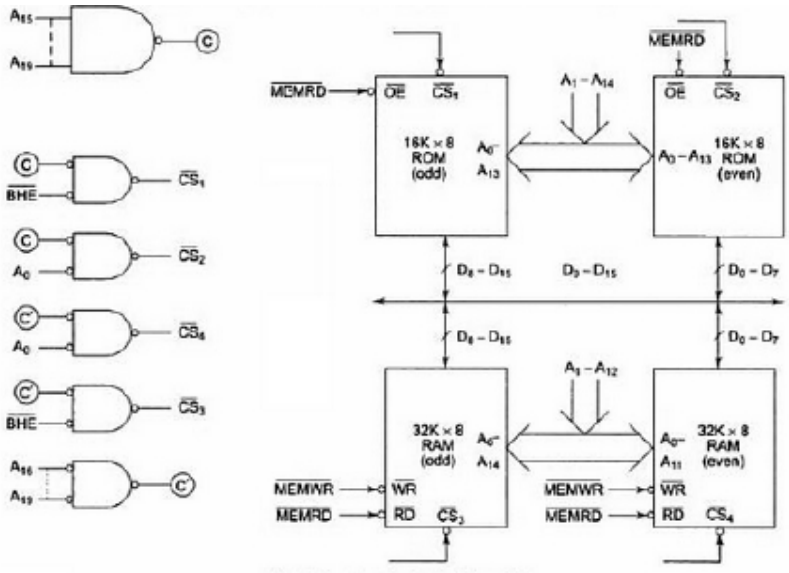
Problem2: Design an interface between 8086 CPU and two chips of 16K×8 EPROM and two chips of 32K×8 RAM. Select the starting address of EPROM suitably. The RAM address must start at 00000 H.

Solution: The last address in the map of 8086 is FFFFF H. after resetting, the processor starts from FFFF0 H. hence this address must lie in the address range of EPROM.

Address Map for Problem

Addresses	A_{19}	A_{18}	A_{17}	A_{16}	A_{15}	A_{14}	A_{13}	A_{12}	A_{11}	A_{10}	A_{09}	A_{08}	A_{07}	A_{06}	A_{05}	A_{04}	A_{03}	A_{02}	A_{01}	A_{00}
FFFFFH	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
						32KB EPROM														
F8000H	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0FFFFH	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
						64KB RAM														
00000H	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

It is better not to use a decoder to implement the above map because it is not continuous, i.e. there is some unused address space between the last RAM address (0FFFF H) and the first EPROM address (F8000 H). Hence the logic is implemented using logic gates.

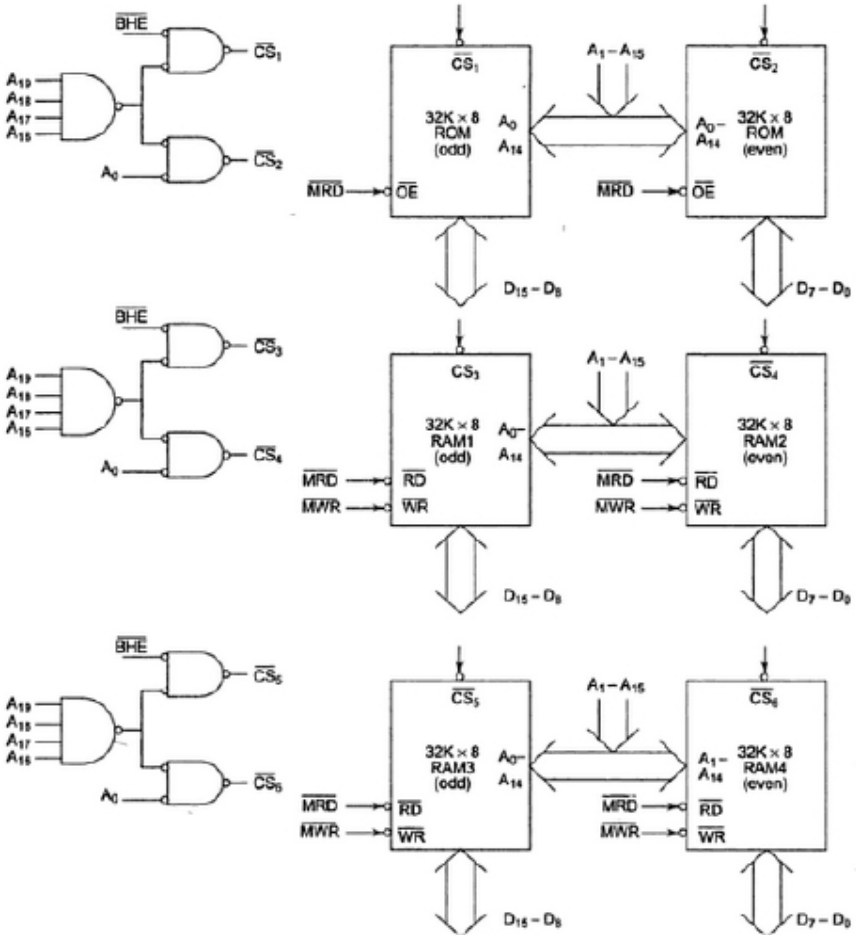


Courtesy: *Advanced Microprocessors and Peripherals, A.K. Ray, Tata McGraw Hill*

Problem3: It is required to interface two chips of 32K×8 ROM and four chips of 32K×8 RAM with 8086, according to following map.
ROM 1 and ROM 2 F0000H - FFFFFH, RAM 1 and RAM 2 D0000H - DFFFFH, RAM 3 and RAM 4 E0000H - EFFFFH. Show the implementation of this memory system.
Solution:

Address Map for Problem

Address	A_{19}	A_{18}	A_{17}	A_{16}	A_{15}	A_{14}	A_{13}	A_{12}	A_{11}	A_{10}	A_{09}	A_{08}	A_{07}	A_{06}	A_{05}	A_{04}	A_{03}	A_{02}	A_{01}	A_{00}
F0000H	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ROM 1and2										64K										
FFFFFH	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
D0000H	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
RAM 1and2										64K										
DFFFFH	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
E0000H	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
RAM 3and4										64K										
EFFFFH	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0



Courtesy: *Advanced Microprocessors and Peripherals, A.K. Ray, Tata McGraw Hill*

INTERFACING I/O PORTS

I/O ports or Input/output ports are the devices through which the microprocessor communicates with other devices or external data source/destinations.

Input activity, as one may expect, is the activity that enables the microprocessor to read data from external devices, and for example keyboards. These devices are known as input devices as they feed data into microprocessor system.

Output activity transfers data from the microprocessor to the external devices, for example CRT display. These devices which accept the data from a microprocessor system are called output devices.

Thus for a microprocessor the input activity is similar to read operation, while the output activity is similar to write operation.

Steps in Interfacing an I/O Device

Connect the data bus of the microprocessor system with the data bus of the I/O port.

Derive a device address pulse by decoding the required address of the device and use it as the chip select of the device.

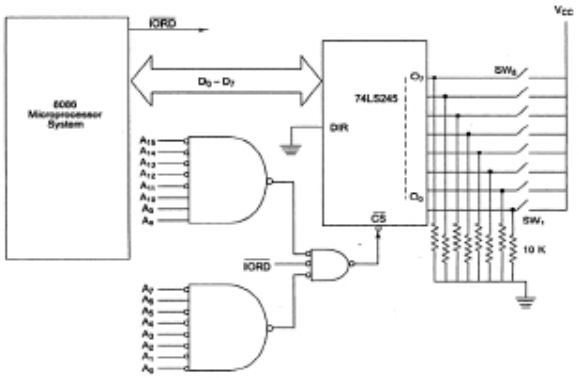
Use a suitable control signal i.e. to carry out device operations.

Methods of Interfacing I/O Devices

Memory Mapping	IO mapping
1. 20-bit addresses are provided for IO devices.	1. 8-bit or 16-bit address are provided for IO devices
2. The IO ports or peripherals can be treated like memory locations and so all instructions related to memory can be used for data transfer.	2. Only IN and OUT instructions can be used for data transfer between IO device and the processor.
3. In memory mapped ports, the data can be moved from any register to port and vice versa	3. In IO mapped ports, the data transfer can take only between the accumulator and the ports
4. When memory mapping is used for IO devices, the full memory address space cannot be used for addressing memory.	4. When IO mapping is used for IO devices, then the full address space can be used for addressing memory.

Problem: Interface an input port 74LS245 to read the status of switches SW1 to SW8. The switches, when shorted, input a 1 else input a 0 to the microprocessor system. Store the status in register BL. The address of the port is 0740H.

Solution:

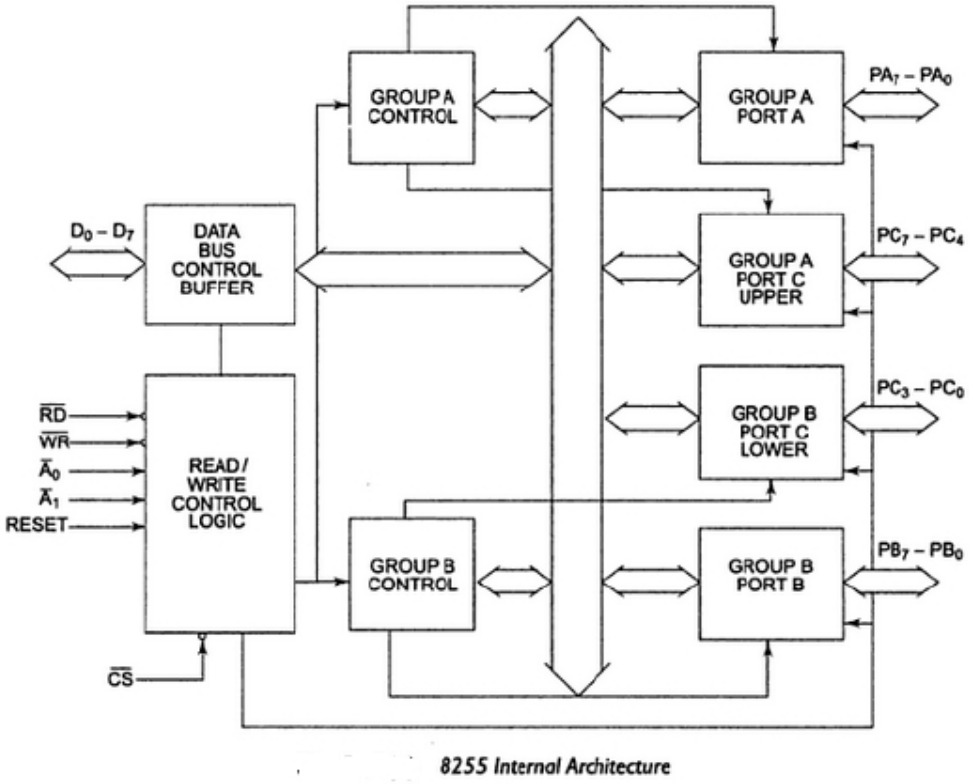


Courtesy: *Advanced Microprocessors and Peripherals, A.K. Ray, Tata McGraw Hill*

The ALP is given as follows:

```
MOV BL, 00          ; Clear BL for status
MOV DX, 0740H       ; 16-bit port address in DX
IN AL, DX           ; Read port 0740H for switch positions
MOV BL, AL          ; Store status of switches from AL into BL
HLT                 ; Stop
```

Programmable Input-Output 8255



8255 Internal Architecture

The parallel input-output port chip 8255 is also known as programmable peripheral input-output port.

It has 24 input/output lines which may be individually programmed in two groups of twelve lines each, or three groups of eight lines.

The two groups of I/O pins are named as Group A and Group B.

Each of these two groups contains a subgroup of eight I/O lines called as 8-bit port and another subgroup of four I/O lines or a 4-bit port.

Thus Group A contains an 8-bit port A along with a 4-bit port, C upper. The port A lines are identified by symbols PA_0 –PA_7 while the port C lines are identified as PC_4-PC_7.

Similarly Group B contains an 8-bit port and a 4-bit port C with lower bits.

The port C upper and port C lower can be used in combination as an 8-bit port C.

All of these ports function independently either as input or as output ports.

Courtesy: *Advanced Microprocessors and Peripherals, A.K. Ray, Tata McGraw Hill*

This can be achieved by programming the bits of internal register of 8255 called as Control Word Register.
The 8-bit data bus buffer is controlled by read/write control logic.
The read/write control logic manages all of the internal and external transfer of both data and control words.

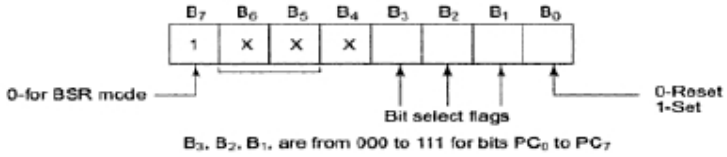
Modes of Operation of 8255

There are two basic modes of operation of 8255- I/O mode and Bit Set-Reset mode(BSR).
In the I/O mode, the 8255 ports work as programmable I/O ports, while in BSR mode only port C(PC0-PC7) can be used to set or reset its individual port bits.
Under the IO mode of operation, further there are three modes of operation of 8255 so as to support different types of applications- mode0, mode1 and mode2.

Bit Set-Reset Mode

In this mode, any of the 8 bits of port C can be set or reset depending on B0 of the control word.
The individual bits of port C can be set or reset by sending out a single OUT instruction to the control register.
When port C is used for control/status operation, this feature can be used to set or reset individual bits.
The bit to be set or reset is selected by bit select flags B3, B2 and B1 of the CWR.

BSR Mode Control Word Register Format



B_3, B_2, B_1, are from 000 to 111 for bits PC_0 to PC_7

B_3	B_2	B_1	*Selected Bits of port C*
0	0	0	B_0
0	0	1	B_1
0	1	0	B_2
0	1	1	B_3
1	0	0	B_4
1	0	1	B_5
1	1	0	B_6
1	1	1	B_7

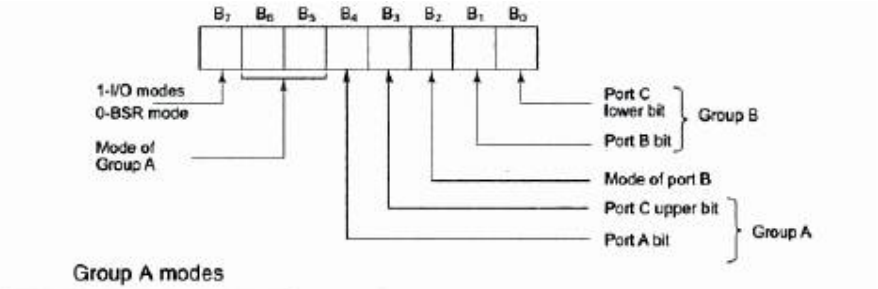
IO Modes

MODE 0: This mode is also known as basic input/output mode. This mode provides simple input and output capability using each of the three ports.
The salient features of this mode are

1. Two 8 bit ports (port A and port B) and two 4-bits ports (port C upper and lower) are available. The two 4-bit ports can be combinedly used as a third 8-bit port.
2. Any port can be used as an input and output port
3. Output ports are latched. Input ports are not latched.
4. A maximum of four ports are available so that overall 16 I/O configurations are possible.

Courtesy: *Advanced Microprocessors and Peripherals, A.K. Ray, Tata McGraw Hill*

All these modes can be selected by programming a register internal to 8255 known as Control Word Register (CWR) which has two formats.



Group A modes

B_6	B_5	Mode
0	0	mode 0
0	1	mode 1
1	0	mode 2
1	1	x

(i) Port B mode is either 0 or 1 depending upon B2 bit.
(ii) A port is an output port if the port bit is 0 else it is input port

MODE 1: This mode is also known as strobed input/output mode. In this mode the handshaking signals control the input or output action of the specified port.
The salient features of this mode are

1. Two groups- group A and group B are available for strobed data transfer.
2. Each group contains one 8-bit data I/O port and one 4-bit control/data port.
3. The 8-bit port can be either used as input or an output port.
4. Out of 8-bit port C, PC_0-PC_2 are used to generate control signals for port B and PC_3-PC_5 are used to generate control signals for port A. the lines PC_6,PC_7 may be used as independent data lines.

MODE 2: This mode is also known as strobed Bidirectional input/output mode.
The salient features of this mode are

1. The single 8-bit port in group A is available.
2. The 8-bit port is bidirectional and additionally a 5-bit control port is available.
3. Three I/O lines are available at port C
4. Input and output ports are both latched
5. The 5-bit control port C is used for generating/accepting handshake signals for 8-bit data transfer on port A.

Interfacing 8255 to 8086

Problem: Interface an 8255 with 8086 to work as an I/O port. Initialize port A as output port, port B as input port and port C as output port. Port A address should be 0740H. Write a program to sense switch positions SW0-SW7 connected to port B. the sensed pattern is to display on port A, to which 8 LEDs are connected, while the port C lower displays number of on switches out of the total eight switches.

Solution:

Thus 82H is the control word for the requirements in the problem. The port address can be done as given below.

Courtesy: *Advanced Microprocessors and Peripherals, A.K. Ray, Tata McGraw Hill*

8255 Ports	I/O Address lines A_{15}	A_{14}	A_{13}	A_{12}	A_{11}	A_{10}	A_{09}	A_{08}	A_{07}	A_{06}	A_{05}	A_{04}	A_{03}	A_{02}	A_{01}	A_{00}	Hex. Port Addresses
PortA	0	0	0	0	0	1	1	1	0	1	0	0	0	0	0	0	0740H
Port B	0	0	0	0	0	1	1	1	0	1	0	0	0	0	1	0	0742H
8255 Ports	I/O Address lines A_{15}	A_{14}	A_{13}	A_{12}	A_{11}	A_{10}	A_{09}	A_{08}	A_{07}	A_{06}	A_{05}	A_{04}	A_{03}	A_{02}	A_{01}	A_{00}	Hex. Port Addresses
Port C	0	0	0	0	0	1	1	1	0	1	0	0	0	1	0	0	0744H
CWR	0	0	0	0	0	1	1	1	0	1	0	0	0	1	1	0	0746H

The 8255 is to be interfaced with lower order data bus, i.e. D_0-D_7

The A_0 and A_1 pins of 8255 are connected to A_{01} and A_{02} pins of microprocessor respectively.

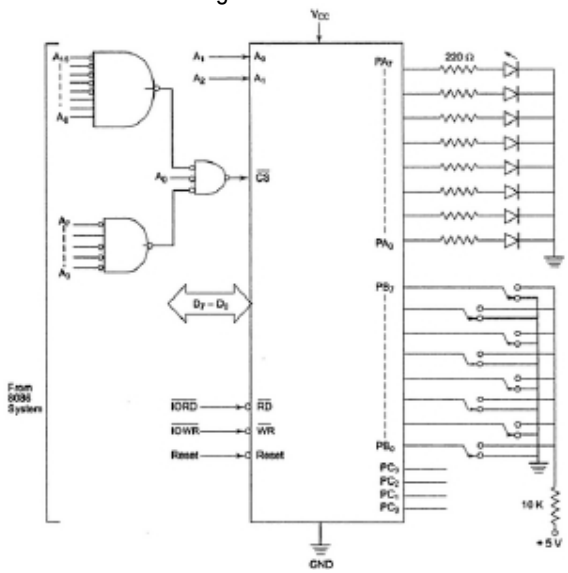
The A_{00} pin of the microprocessor is used for selecting lower byte of data bus.

Hence any change in the status of A_{00} does not affect the port to be selected.

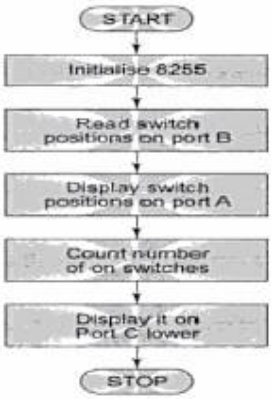
Let us use absolute decoding scheme that uses all the 16 address lines for deriving the device address pulse.

Out of A_0-A_{15} lines, two address lines A_{02} and A_{01} are directly required by 8255 for three port and CWR address decoding.

Hence only A_3 to A_{15} are used for decoding address.



Flow Chart



Courtesy: *Advanced Microprocessors and Peripherals, A.K. Ray, Tata McGraw Hill*

Program

```
MOV DX, 0746H      ; Initialize CWR with
MOV AL, 82H        ; control word 82H
OUT DX, AL         ;
SUB DX, 04         ; Get address of port B in DX
IN AL, DX          ; Read port B for switch
SUB DX, 02         ; positions in to AL and get port A address in DX
OUT DX, AL         ; Display switch positions on port A
MOV BL, 00H        ; Initialize BL for switch count
MOV CH, 08H
YY: ROL AL         ; Rotate AL through carry to check,
JNC XX             ; whether the switches are on or
INC BL             ; off, i.e. either 1 or 0
XX: DEC CH         ; Check for next switch. If all
JNZ YY             ; switch are checked, the
MOV AL, BL         ; number of on switches are
ADD DX, 04         ; in BL. Display it on port C
OUT DX, AL         ; lower
HLT
```

Interfacing to Keyboard

In most keyboards, the key switches are connected in a matrix of rows and columns.
We use simple mechanical switches but the principle is same for other types of switches.
Getting a meaningful data from a keyboard requires three major tasks.

1. Detect a key press.
2. Debounce the key press
3. Encode the key press

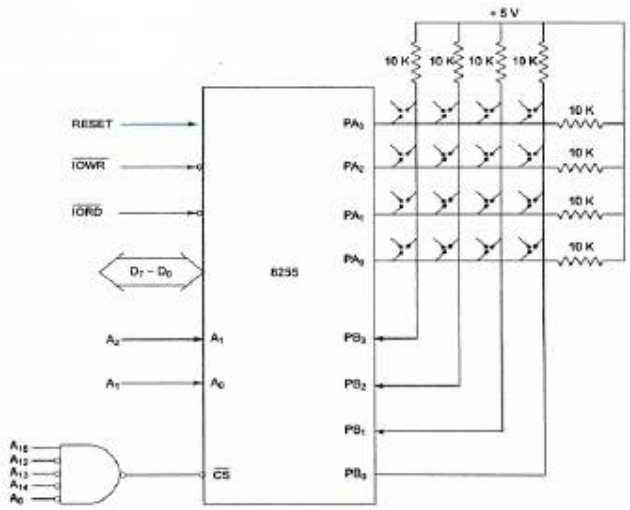
The three tasks can be done with hardware, software, or a combination of two depending on application.

Problem: Interface a 4*4 keyboard with 8086 using 8255 and write an ALP for detecting a key closure and return the key code in AL. The debouncing period for a key is 10ms. Use software key debouncing technique. DEBOUNCE is an available 10ms delay routine.

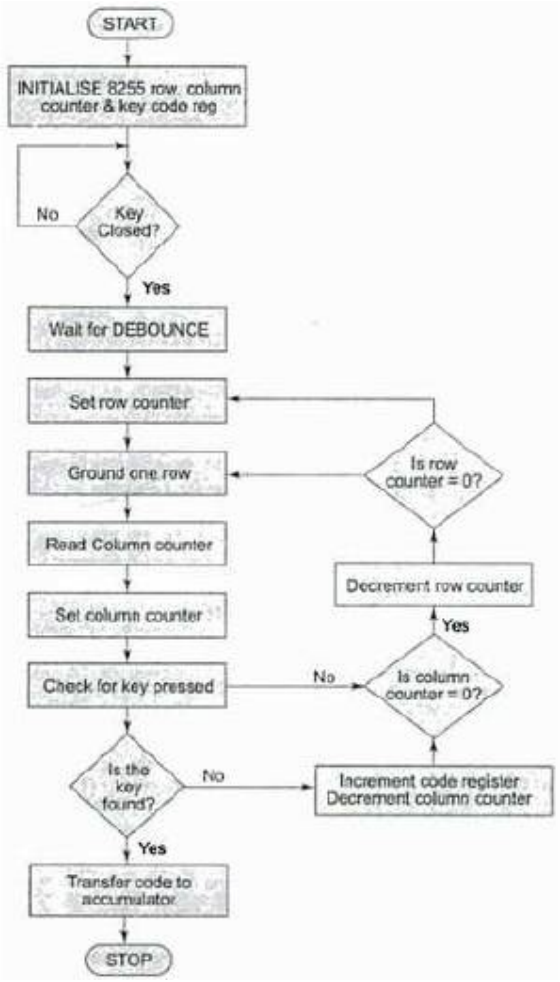
Solution:

Port A is used as output port for selecting a row of key while port B is used as an input port for sensing a closed key.
Thus keyboard lines are selected one by one through port A and the port B lines are polled continuously till a key closure is sensed.
Then routine DEBOUNCE is called for debouncing.
The key code is decided depending upon the selected row and a low sensed column.
The higher order lines of port A and port B are left unused.
The address of port A and port B will be respectively 8000H and 8002H while the address of CWR will be 8006H.

Courtesy: *Advanced Microprocessors and Peripherals, A.K. Ray, Tata McGraw Hill*



Flow chart:



Courtesy: *Advanced Microprocessors and Peripherals, A.K. Ray, Tata McGraw Hill*

Check 1: whether any key is pressed or not?

1. Make all column lines zero by sending low on all output lines. This activates all keys in the keyboard matrix.
2. Read the status of return lines. If status of all lines is logic high, key is not pressed; otherwise pressed.

Check 2:

1. Activate keys from any one column by making any one column line zero.
2. Read the status of return lines. The zero on any return line indicates key is pressed from the corresponding row and selected column.
3. Activate the keys from next column and repeat 2 and 3 for all columns.

Program:

```
ASSUME CS: CODE
CODE    SEGMENT
START:  MOV AL, 82H          ; Load CWR with
        MOV DX, 8006H        ; control word
        OUT DX, AL           ; required
        MOV BL, 00H          ; Initialize BL for key code
        XOR AX, AX           ; Clear all flags
        MOV DX, 8000H        ; Port Address in AX
        OUT DX, AL           ; Ground all rows
        ADD DX, 02           ; Port B address in DX
WAIT:   IN AL, DX            ; Read all columns
        AND AL, 0F H         ; Mask data lines D7-D4
        CMP AL, 0F           ; any key closed?
        JZ WAIT              ; if not, wait till key
        CALL DEBOUNCE        ; closure else wait for 10ms
        MOV AL, F7 H         ; Load data byte to ground
        MOV BH, 04 H         ; a row and set row counter
NXTROW: ROL AL, 01 H         ; rotate AL to ground next row
        MOV CH, AL           ; save data byte to ground next row
        SUB DX, 02           ; output Port address is in DX
        OUT DX, AL           ; ground one of the rows
        ADD DX, 02           ; input port address is in DX
        IN AL, DX            ; read input for key closure
        AND AL, 0F H         ; mask D4-D7
        MOV CL, 04 H         ; set column counter
NXTCOL: ROR AL, 01           ; move D0 in CF
        JNC CODEKY           ; key closure is found, if CF=0
        INC BL               ; increment BL for next binary key code
        DEC CL               ; decrement column counter, if no key closure found
        JNZ NXTCOL           ; check for key closure key in next column
        MOV AL, CH           ; load data byte to ground next row
        DEC BH               ; if no key closure found in column get ready to ground next row
        JNZ NXTROW           ; go back to ground next row
        JMP WAIT             ; jump back to check for key closure again
CODEKY: MOV AL, BL           ; key code is transferred to AL
        MOV AH, 4C H         ; return to DOS prompt
        INT 21 H
```


Courtesy: *Advanced Microprocessors and Peripherals, A.K. Ray, Tata McGraw Hill*

Procedure to generate 10ms delay at 5 MHz operating frequency

```
DEBOUNCE    PROC  NEAR
            MOV CL, 0E2 H
BACK:       NOP
            DEC CL
            JNZ BACK
            RET
DEBOUNCE    ENDP
CODE        ENDS
END         START
```

Interfacing to Alphanumeric Displays

Most of the microprocessor controlled instruments and machines need to display letters of the alphabet and numbers to give directions or data values to users.
This can be displayed using CRT, LED or LCD displays.
CRT displays are used when a large amount of data is to be displayed.
In systems where only a small amount of data is to be displayed, simple LED and LCD displays are used.

Interfacing to Seven Segment Displays

7 segment displays are generally used as numerical indicators and consists of a number of LEDs arranged in seven segments.
Any number between 0--9 can be indicated by lighting the appropriate segments.
The seven segments are labeled as a-g and dot is labeled as h.
By forward biasing different LED segments, we can display the digits 0 through 9.

Problem: Interface an 8255 with 8086 at 80H as an I/O address of port A. interface five 7 segment displays with 8255. Write a sequence of instructions to display 1, 2, 3, 4 and 5 over five displays continuously as per their positions starting with 1 at the least significant position.
Solution: In this scheme. I/O port A is multiplexed to carry data to all the 7-segment displays. The port B selects one of the displays at a time. The displays used are common anode type.

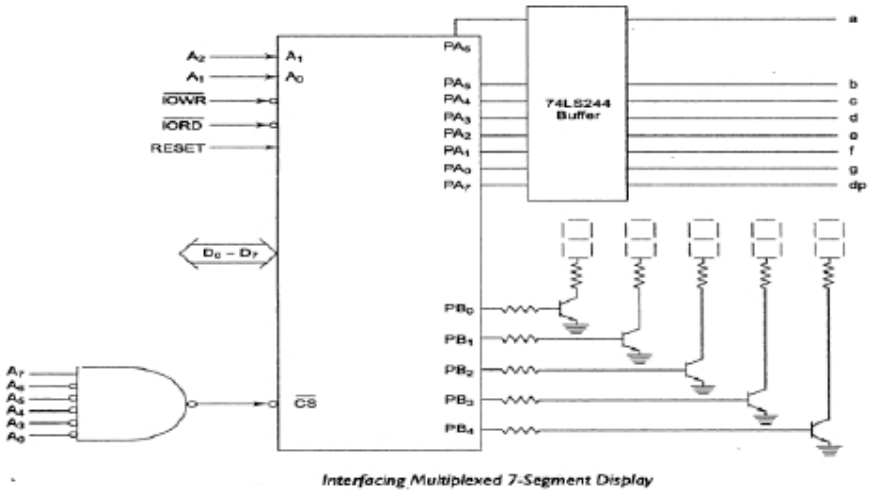
Number to be displayed	PA_7 dp	PA_6 a	PA_5 b	PA_4 c	PA_3 d	PA_2 e	PA_1 f	PA_0 g	Code
1	1	1	0	0	1	1	1	1	CF
2	1	0	0	1	0	0	1	0	92
3	1	0	0	0	0	1	1	0	86
4	1	1	0	0	1	1	0	0	CC
5	1	0	1	0	0	1	0	0	A4

All these codes, decided above, are stored in a look up table starting at 2000:0000
Program:

```
ASSUME CS: CODE
CODE   SEGMENT
AGAIN: MOV CL, 05H          ; Count for displays
       MOV BX, 2000H        ; Initialize data segment
```


Courtesy: *Advanced Microprocessors and Peripherals, A.K. Ray, Tata McGraw Hill*

```
        MOV DS, BX          ; for look up table
        MOV CH, 01H         ; 1st number to be displayed
        MOV AL, 80H         ; Load control word in the
        OUT 86H, AL         ; CWR
        MOV DL, 01H         ; Enable code for least significant 7-seg display
NXTDGT: MOV BX, 0000H       ; Set pointer to look up table
        MOV AL, CH          ; First no to display Store number to be displayed in AL
        XLAT                ; Find code from look up table
        OUT 80H, AL         ; Display the code
        MOV AL, DL          ; Enable the display
        OUT 82H, AL;
        ROL DL              ; Go for selecting next display
        INC CH              ; Next number to display
        DEC CL
        JNZ NXTDGT          ; Go for next digit display
        JMP AGAIN
CODE        ENDS
```



Interfacing Multiplexed 7-Segment Display

Interfacing to Stepper Motor

A stepper motor is a device used to obtain an accurate position control of rotating shafts.

It employs rotation of its shafts in terms of steps, rather than continuous rotation as in case of AC or DC motors.

To rotate the shafts of the stepper motor, a sequence of pulses is needed to be applied to the windings of the stepper motor in a proper sequence.

The numbers of pulses required for one complete rotation of the shaft of the stepper motor are equal to its number of internal teeth or its rotor.

A typical stepper motor may have parameters like torque 3Kg-cm, operating voltage 12V, current rating 0.2A and a step angle 1.8^0, i.e. 200 steps/revolution (number of rotor teeth).

A simple scheme for rotating the shaft of stepper motor is called a wave scheme.

Courtesy: *Advanced Microprocessors and Peripherals, A.K. Ray, Tata McGraw Hill*

In this scheme, the windings W_a, W_b, W_c and Wd are applied with required voltage pulses, in a cyclic fashion.

Excitation Sequences of a Stepper Motor Using Wave Switching Scheme

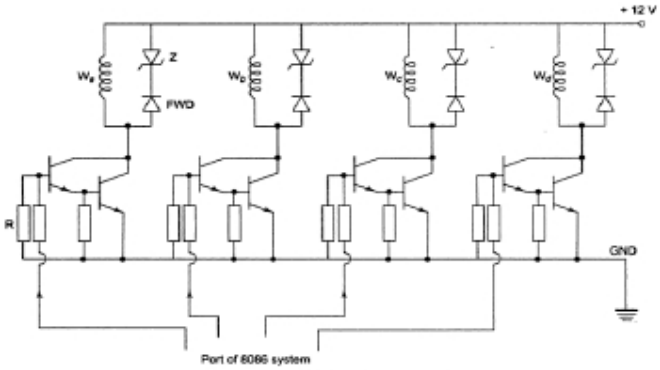
Motion	Step	A	B	C	D
Clockwise	1	1	0	0	0
	2	0	1	0	0
	3	0	0	1	0
	4	0	0	0	1
	5	1	0	0	0
Anticlockwise	1	1	0	0	0
	2	0	0	0	1
	3	0	0	1	0
	4	0	1	0	0
	5	1	0	0	0

Problem: Design a stepper motor controller and write an ALP to rotate shaft of 4-phase stepper motor:

a. In clock wise 5 rotations
b. In anticlockwise 5 rotations

The 8255 port A address is 0740H. The stepper motor has 200 rotor teeth. The port A bit PA_0 drives winding Wa, PA_1 drives Wb and so on. The stepper motor has an inertial delay of 10ms. Assume that the routine for this delay is already available.

Solution: The stepper motor connections for all the four windings are



Stepper Motor Windings Connections

Program:

```
ASSUME CS: CODE
CODE SEGMENT
START: MOV AL, 80H
       OUT CWR, AL
       MOV AL, 88H          ; Bit pattern 10001000 to start the sequence of excitation from Wa.
       MOV CX, 1000         ; for clockwise rotations the count is 200*5=1000
AGAIN1: OUT PORTA, AL       ;
       CALL DELAY           ; Excite Wa, Wb, Wc, Wd in sequence with delay.
       ROL AL, 01           ;
       DEC CX               ;
       JNZ AGAIN1           ; Excite till count=0
       MOV AL, 88H          ; Bit pattern to excite Wa
       MOVCX, 1000          ; count for 5 rotations
AGAIN2: OUT PORTA, AL       ; excite Wa, Wb, Wc, Wd
       CALL DELAY           ; Wait
        ROR AL, 01      ; anticlockwise
```


Courtesy: *Advanced Microprocessors and Peripherals, A.K. Ray, Tata McGraw Hill*

```
        DEC CX
        JNZ AGAIN2
        MOV AH, 4CH
        INT 21H
CODE    ENDS
END     START
```

Interfacing to A/D Converter

A general algorithm for ADC interfacing contains:

1. Ensure the stability of analog input applied to the ADC
2. Issue start of conversion (SOC) pulse to ADC
3. Read end of conversion (EOC) signal to mark the end of conversion process
4. Read digital data output of the ADC as equivalent digital output

It may be noted that the analog input voltage must be a constant at the input of the ADC right from the beginning to end of the conversion to get correct results.

ADC 0808/0809

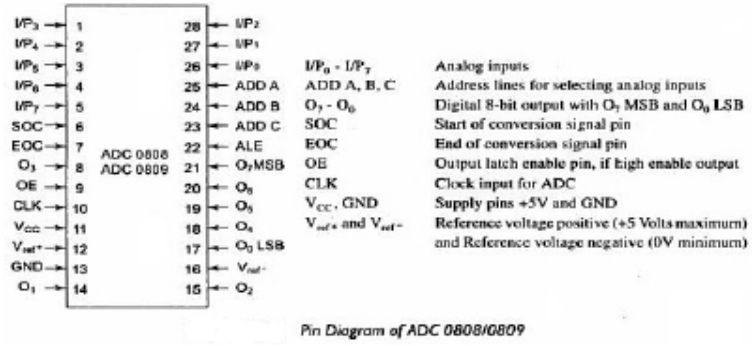
The analog to digital converter chips 0808 and 0809 are 8-bit CMOS, successive approximation converters.

These converters internally have 3:8 analog multiplexer so that at a time eight different analog inputs can be connected to chip.

Out of these eight inputs only one can be selected for conversion using address lines ADD A, ADD B and ADD C

Analog I/P selected	Address lines		
	C	B	A
I/P 0	0	0	0
I/P 1	0	0	1
I/P 2	0	1	0
I/P 3	0	1	1
I/P 4	1	0	0
I/P 5	1	0	1
I/P 6	1	1	0
I/P 7	1	1	1

These are unipolar analog to digital converters, i.e. they are able to convert only positive analog input voltages to their digital equivalents.



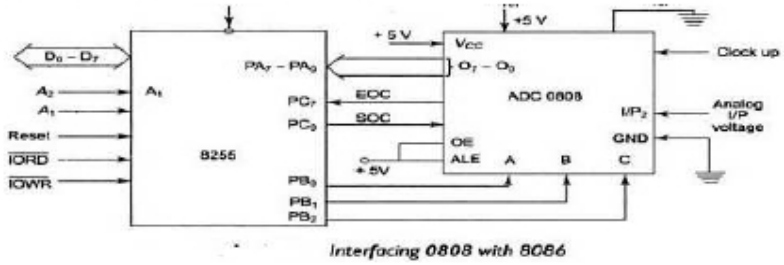
Pin Diagram of ADC 0808/0809

Problem: Interface ADC 0808 with 8086 using 8255 ports. Use port A of 8255 for transferring digital data output of ADC to the CPU and port C for control signals. Assume that an analog input is present at I/P_2 of the ADC and a clock input of suitable frequency is available for ADC. Draw the schematic and write required ALP.

Courtesy: *Advanced Microprocessors and Peripherals, A.K. Ray, Tata McGraw Hill*

Solution:

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	Control word
1	0	0	1	1	0	0	0	= 98 H



Interfacing 0808 with 8086

The analog input I/P_2 is used and therefore address pins A, B, C should be 010 respectively to select I/P_2.

The OE and ALE pins are already kept at +5V to select the ADC and enable the outputs.

Port C upper acts as the input port to receive the EOC signal while port C lower acts as output port to send SOC to the ADC.

Port A acts as a 8-bit input data port to receive the digital data output from the ADC.

Program:

```
        ASSUME CS: CODE
        CODE SEGMENT
START:  MOV AL, 98H          ; Initialize 8255
        OUT CWR, AL
        MOV AL, 02H          ; Select I/P2
        OUT PORT B, AL       ; analog input
        MOV AL, 00H          ; Give start of conversion
        OUT PORT C, AL       ; pulse to the ADC
        MOV AL, 01H
        OUT PORTC, AL
        MOV AL, 00H
        OUT PORT C, AL
WAIT:   IN AL, PORT C        ; Check EOC by
        RCL                  ; reading port C upper and
        JNC WAIT             ; rotating through carry
        IN AL, PORT A        ; If EOC, read digital equivalent in AL
        HLT                  ; Stop
CODE        ENDS
END         START
```

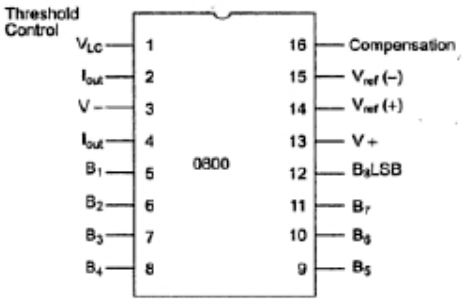
Interfacing to D/A Converters

The digital to analog converters converts binary numbers into their corresponding analog equivalent voltages.

The DAC find applications in areas like digitally controlled gains, motor speed controls, programmable gain amplifiers etc.

The DAC 0800 is a monolithic 8-bit DAC manufactured by National Semiconductor.

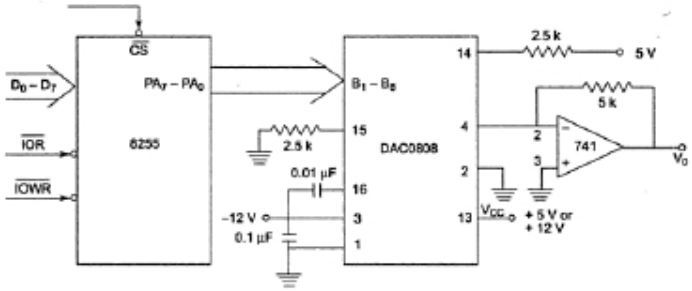
Courtesy: *Advanced Microprocessors and Peripherals, A.K. Ray, Tata McGraw Hill*



Pin Diagram of DAC 0800

Problem: Write an ALP to generate a triangular wave of frequency 500 Hz using the DAC 0800 interfacing. The 8086 system operates at 8 MHz the amplitude of the triangular wave should be +5V.

Solution:



Interfacing DAC0800 with 8086

The V_{ref+} should be tied to +5V to generate a wave of +5V amplitude.
The required frequency of the output is 500 Hz i.e. the period is 2ms.
Assuming the wave to be generated is symmetric; the waveform will rise for 1ms and fall for 1ms. This will be repeated continuously.

Program:

```
    ASSUME CS: CODE
    CODE SEGMENT
START: MOV AL, 80H          ; Initialize 8255 Ports suitably
       OUT CWR, AL
       MOV AL, 00H          ; Start raising ramp from 0V by sending 00h to DAC
BACK: OUT PORT A , AL
       INC AL               ; Increment ramp till 5V i.e. FFH
       CMP AL, FF H
       JB BACK              ; if it is FFH then
BACK1: OUT PORT A, AL       ; Output it and start the falling
       DEC AL               ; ramp by decrementing the
       CMP AL, 00H          ; counter till it reaches
       JA BACK1             ; zero then start again
       JMP BACK             ; for the next cycle
CODE        ENDS
END         START
```


Courtesy: *Advanced Microprocessors and Peripherals, A.K. Ray, Tata McGraw Hill*