18BEBME404 MICROPROCESSOR AND MICROCONTROLLER 3 0 0 3 100

OBJECTIVES:

The student should be made to:

- Study the Architecture of 8086 microprocessor.
- Learn the design aspects of I/O and Memory Interfacing circuits.
- Study about communication and bus interfacing.
- Study the Architecture of 8051 microcontroller.

OUTCOMES:

At the end of the course, the student should be able to:

- Design and implement programs on 8086 microprocessor.
- Design I/O circuits.
- Design Memory Interfacing circuits.
- Design and implement 8051 microcontroller based systems.

UNIT -I THE 8085 MICROPROCESSOR 9

Introduction to 8085 – Microprocessor architecture – Instruction set – Programming the 8085 – Code conversion.

UNIT II THE 8086 MICROPROCESSOR 9

Introduction to 8086 – Microprocessor architecture – Addressing modes - Instruction set and assembler directives – Assembly language programming – Modular Programming – Interrupts and interrupt service routines – Byte and String Manipulation.

UNIT III I/O INTERFACING 9

Memory Interfacing and I/O interfacing - Parallel communication interface – Serial communication interface – D/A and A/D Interface - Timer – Keyboard /display controller – Interrupt controller – DMA controller – Programming and applications Case studies: Traffic Light control, LED display , LCD display, Keyboard display interface and Alarm Controller.

UNIT IV MICROCONTROLLER 9

Architecture of 8051 – Special Function Registers(SFRs) - I/O Pins Ports and Circuits – Instruction set - Addressing modes - Assembly language programming.

UNIT V INTERFACING MICROCONTROLLER 9

Programming 8051 Timers - Serial Port Programming - Interrupts Programming – LCD & Keyboard Interfacing - ADC, DAC & Sensor Interfacing - External Memory Interface- Stepper Motor and Waveform generation.

TOTAL: 45

TEXT BOOKS:

S.NO.	Author(s) Name	Title of the book	Publisher	Year of publication
1	Yu-Cheng Liu, Glenn A.Gibson	Microcomputer Systems: The 8086 / 8088 Family Architecture, Programming and Design	Second Edition, Prentice Hall of India	2007
2	Mohamed Ali Mazidi, Janice Gillispie Mazidi, Rolin	The 8051 Microcontroller and Embedded Systems: Using Assembly and C	Second Edition, Pearson education	2011

REFERENCE:

S.NO.	Author(s) Name	Title of the book	Publisher	Year of publication
1	Doughlas V.Hall	Microprocessors and Interfacing, Programming and Hardware	TMH	2012

KARPAGAM ACADEMY OF HIGHER EDUCATION
COIMBATORE-21
Faculty of Engineering
Department of Electronics and Communication Engineering

LECTURE PLAN

Name of the staff : M.Daranikumar
Designation : Assistant Professor.
Class : B.E-II Year BME
Subject : Microprocessor and Microcontroller
Subject code : 18BEBME404

Sl.No	Topics to be covered	Time Duration	Teaching aids
	INTRODUCTION		
1	Introduction to Microprocessors and Microcontrollers	01	
UNIT –I THE 8085 MICROPROCESSOR			
2	Introduction to 8085 – Microprocessor architecture	01	R2 105-108
3	8085 Pin configuration.	01	R2 97
4	Instruction and types.	01	R2 35-37
5	Addressing modes and interrupts	02	R2 375-395
6	Instruction set	02	R2 47-50
7	Programming the 8085.	01	R2 173-274
8	Code conversion	01	R2 329-335
UNIT II THE 8086 MICROPROCESSOR			
9	Introduction to 8086 – Microprocessor architecture.	01	R1 2.10-2.16
10	Minimum and maximum modes of operation	01	R17.6-7.7
11	Addressing modes.	01	W1
12	Instruction set.	01	R1 3.7-3.12
13	Assembler directives.	01	R1 6.1-6.34
14	Assembly language programming – Modular Programming.	02	R1 3.1-3.29
15	Interrupts and interrupt service routines.	01	R1 8.1-8.34
16	Byte and String Manipulation.	01	W1
UNIT III I/O INTERFACING			
17	Memory Interfacing and I/O interfacing.	01	R1 11.10-11.23
18	Parallel communication interface.	01	R1 9.7-9.16
19	Serial communication interface.	01	R1 14.1- 14.14
20	D/A and A/D Interface.	01	R1 10.13-10.23
21	Timer and Interrupt controller.	01	R1 8.1-8.34
22	Keyboard /display controller.	01	R1 9.27-9.34
23	DMA controller – Programming and applications.	01	R1 11.5-11.10
24	Case studies: Traffic Light control, LED, LCD display.	01	R1 9.35-9.43
25	Keyboard display interface and Alarm Controller.	01	R1 9.27-9.34
UNIT IV MICROCONTROLLER			

26	Architecture of 8051.	02	T2 23-26
27	Special Function Registers(SFRs).	01	T2 92-94
28	I/O Pins Ports and Circuits.	01	T2 76-85
29	Instruction set.	02	T2 115-135
30	Addressing modes.	01	T2 90-91
31	Assembly language programming.	02	-
UNIT V INTERFACING MICROCONTROLLER			
32	Programming 8051 Timers.	01	T2 184-187
33	Serial Port Programming.	01	T2 244-255
34	Interrupts Programming.	01	T2 271-279
35	LCD & Keyboard Interfacing.	01	T2 299-310
36	ADC, DAC & Sensor Interfacing.	02	T2 322-347
37	External Memory Interface.	01	T2 356-366
38	Stepper Motor.	01	T2 432-440
39	Waveform generation.	01	T2 417-418

W1- https://www.tutorialspoint.com/microprocessor/microprocessor_8086_addressing_modes.htm

TEXT BOOK

Sl.No.	Author(s)	Title of the Book	Publisher	Year of Publication
1	Yu-Cheng Liu, Glenn A.Gibson	Microcomputer Systems: The 8086 / 8088 Family Architecture, Programming and Design	Second Edition Prentice Hall of India	2007
2	Mohamed Ali Mazidi, Janice Gillispie Mazidi, Rolin McKinlay	The 8051 Microcontroller and Embedded Systems: Using Assembly and C	Second Edition, Pearson education	2011

REFERENCES

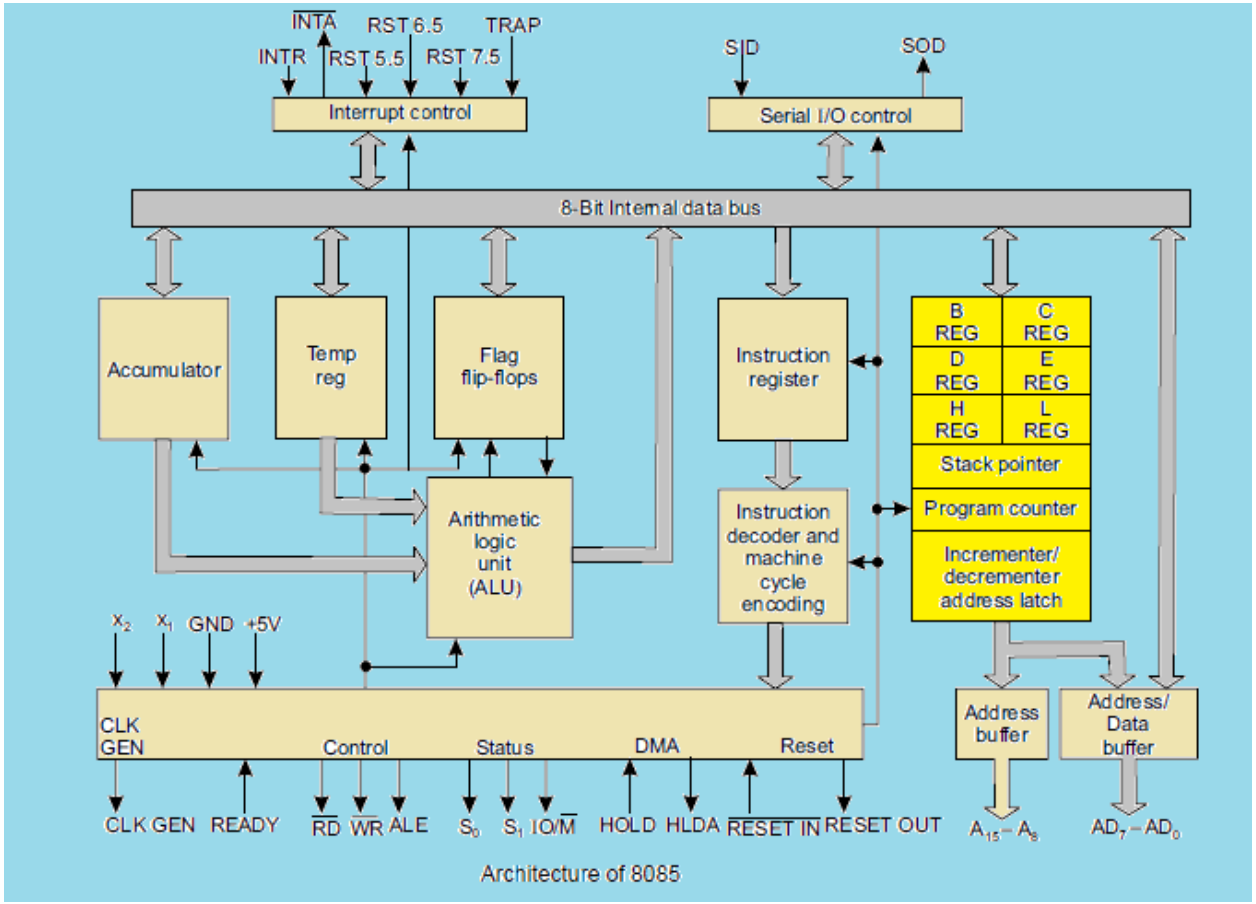
Sl.No.	Author(s)	Title of the Book	Publisher	Year of Publication
1	Doughlas V.Hall	Microprocessors and Interfacing, Programming and Hardware	TMH	2012
2	Ramesh Gaonkar	Microprocessor Architecture and Applications with the 8085	PRI	2009

UNIT 1

8085 Microprocessor Architecture

It has the following configuration −

- 8-bit data bus
- 16-bit address bus, which can address upto 64KB
- A 16-bit program counter
- A 16-bit stack pointer
- Six 8-bit registers arranged in pairs: BC, DE, HL
- Requires +5V supply to operate at 3.2 MHZ single phase clock



Architecture of 8085

8085 consists of the following functional units −

Accumulator

It is an 8-bit register used to perform arithmetic, logical, I/O & LOAD/STORE operations. It is connected to internal data bus & ALU.

Arithmetic and logic unit

As the name suggests, it performs arithmetic and logical operations like Addition, Subtraction, AND, OR, etc. on 8-bit data.

General purpose register

There are 6 general purpose registers in 8085 processor, i.e. B, C, D, E, H & L. Each register can hold 8-bit data.

These registers can work in pair to hold 16-bit data and their pairing combination is like B-C, D-E & H-L.

Program counter

It is a 16-bit register used to store the memory address location of the next instruction to be executed. Microprocessor increments the program whenever an instruction is being executed, so that the program counter points to the memory address of the next instruction that is going to be executed.

Stack pointer

It is also a 16-bit register works like stack, which is always incremented/decremented by 2 during push & pop operations.

Temporary register

It is an 8-bit register, which holds the temporary data of arithmetic and logical operations.

Flag register

It is an 8-bit register having five 1-bit flip-flops, which holds either 0 or 1 depending upon the result stored in the accumulator.

These are the set of 5 flip-flops −

- Sign (S)
- Zero (Z)
- Auxiliary Carry (AC)
- Parity (P)

- Carry (C)

Its bit position is shown in the following table −

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

Instruction register and decoder

It is an 8-bit register. When an instruction is fetched from memory then it is stored in the Instruction register. Instruction decoder decodes the information present in the Instruction register.

Timing and control unit

It provides timing and control signal to the microprocessor to perform operations. Following are the timing and control signals, which control external and internal circuits −

- Control Signals: READY, RD', WR', ALE
- Status Signals: S0, S1, IO/M'
- DMA Signals: HOLD, HLDA
- RESET Signals: RESET IN, RESET OUT

INSTRUCTUION SET:

Data Transfer Group — Moves data between registers or between memory locations and registers. Includes moves, loads, stores, and exchanges.

Opcode	Operand	Meaning	Explanation
MOV	Rd, Sc M, Sc Dt, M	Copy from the source (Sc) to the destination(Dt)	This instruction copies the contents of the source register into the destination register without any alteration. **Example** − MOV C, B

MVI	Rd, data M, data	Move immediate 8-bit	The 8-bit data is stored in the destination register or memory. **Example** − MVI B, 55L
LDA	16-bit address	Load the accumulator	The contents of a memory location, specified by a 16-bit address in the operand, are copied to the accumulator. **Example** − LDA 2034

Arithmetic Group — Adds, subtracts, increments, or decrements data in registers or memory.

Opcode	Operand	Meaning	Explanation
ADD	R M	Add register or memory, to the accumulator	The contents of the register or memory are added to the contents of the accumulator and the result is stored in the accumulator. **Example** − ADD B.
ADC	R M	Add register to the accumulator with carry	The contents of the register or memory & M the Carry flag are added to the contents of the accumulator and the result is stored in the accumulator. **Example** − ADC D

Logic Group — ANDs, ORs, XORs, compares, rotates, or complements data in registers or between memory and a register.

Opcode	Operand	Meaning	Explanation
CMP	R M	Compare the register or memory with the accumulator	The contents of the operand (register or memory) are M compared with the contents of the accumulator.
ANA	R M	Logical AND register or memory with the accumulator	The contents of the accumulator are logically AND with M the contents of the register or memory, and the result is placed in the accumulator.
ANI	8-bit data	Logical AND immediate with the accumulator	The contents of the accumulator are logically AND with the 8-bit data and the result is placed in the accumulator.
RAL	None	Rotate the accumulator left through carry	Each binary bit of the accumulator is rotated left by one position through the Carry flag. Bit D7 is placed in the Carry flag, and the Carry flag is placed in the least significant position D0. CY is modified according to bit D7.

Branch Group — Initiates conditional or unconditional jumps, calls, returns, and restarts.

Opcode	Operand	Meaning	Explanation
JMP	**16-bit address**	Jump unconditionally	The program sequence is transferred to the memory address given in the operand.

. **Stack, I/O, and Machine Control Group** — Includes instructions for maintaining the stack, reading from input ports, writing to output ports, setting and reading interrupt masks, and setting and clearing flags.

Opcode	Operand	Meaning	Explanation
NOP	None	No operation	No operation is performed, i.e., the instruction is fetched and decoded.
HLT	None	Halt and enter wait state	The CPU finishes executing the current instruction and stops further execution. An interrupt or reset is necessary to exit from the halt state.
DI	None	Disable interrupts	The interrupt enable flip-flop is reset and all the interrupts are disabled except TRAP.
EI	None	Enable interrupts	The interrupt enable flip-flop is set and all the interrupts are enabled.

Programming in 8085

8 bit addition program:

MEMORY ADDRESS	MNEMONICS	COMMENT
2000	LDA 2050	A<-[2050]
2003	MOV B, A	B<-A
2004	LDA 2051	A<-[2051]

MEMORY ADDRESS	MNEMONICS	COMMENT
2007	ADD B	A<-A+B
2008	STA 2052	[2052]<-A
200B	HLT	Terminate

8 bit subtraction program with or without borrow –

MEMORY	MNEMONICS	OPERANDS	COMMENT
2000	MVI	C, 00	[C] <- 00
2002	LHLD	2500	[H-L] <- [2500]
2005	MOV	A, H	[A] <- [H]
2006	SUB	L	[A] <- [A] – [L]
2007	JNC	200B	Jump If no borrow
200A	INR	C	[C] <- [C] + 1
200B	STA	2502	[A] -> [2502], Result
200E	MOV	A, C	[A] <- [C]
2010	STA	2503	[A] -> [2503], Borrow
2013	HLT		Stop

8 bit multiplication Program –

MEMORY ADDRESS	MNEMONICS	COMMENT
2000	LHLD 2050	H←2051, L←2050
2003	XCHG	H↔D, L↔E
2004	MOV C, D	C←D
2005	MVI D 00	D←00
2007	LXI H 0000	H←00, L←00
200A	DAD D	HL←HL+DE
200B	DCR C	C←C-1
200C	JNZ 200A	If Zero Flag=0, goto 200A
200F	SHLD 3050	H→3051, L→3050
2012	HLT	

Division of two 8 bits numbers

MEMORY ADDRESS	MNEMONICS	COMMENT
2000	LXI H, 3050	H←30, L←50

MEMORY ADDRESS	MNEMONICS	COMMENT
2003	MOV B, M	B←M
2004	MVI D, 00	D←0
2005	INX H	H=H+1
2006	MOV A, M	A←M
2007	CMP B	Compare B and A
2008	JC 2012	If Carry is 1 goto 2012
200C	SUB B	A=A-B
200D	INR C	C=C+1
200E	JMP 2007	Jump to 2007
2012	STA 3052	Store A at 3052
2015	MOV A, C	Copy C to A
2016	STA 3053	Store A at 3053
2019	**HLT**	

16 bit addition

Program –

MEMORY ADDRESS	MNEMONICS	COMMENTS
2000	LHLD 2050	H-L ← 2050
2003	XCHG	D H & E L
2004	LHLD 2052	H-L ← 2052
2007	DAD D	H ← H+D & L ← L+E
2008	SHLD 3050	A → 3050
200B	HLT	Stops execution

16 bit subtraction

Program –

MEMORY ADDRESS	MNEMONICS	COMMENTS
2000	LHLD 2050	Load H-L pair with address 2050
2003	XCHG	EXCHANGE H-L PAIR WITH D-E PAIR
2004	LHLD 2052	Load H-L pair with address 2052
2007	MVI C, 00	C<-00H
2009	MOV A, E	A<-E
200A	SUB L	A<-A-L
200B	STA 2054	2054<-A
200E	MOV A,	A<-D

MEMORY ADDRESS	MNEMONICS	COMMENTS
	D	
200F	SBB H	SUBTRACT WITH BORROW
2010	STA 2055	2055<-A
2013	HLT	TERMINATES THE PROGRAM

UNIT 2

8086 architecture:

8086 Microprocessor is an enhanced version of 8085Microprocessor that was designed by Intel in 1976.

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.

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

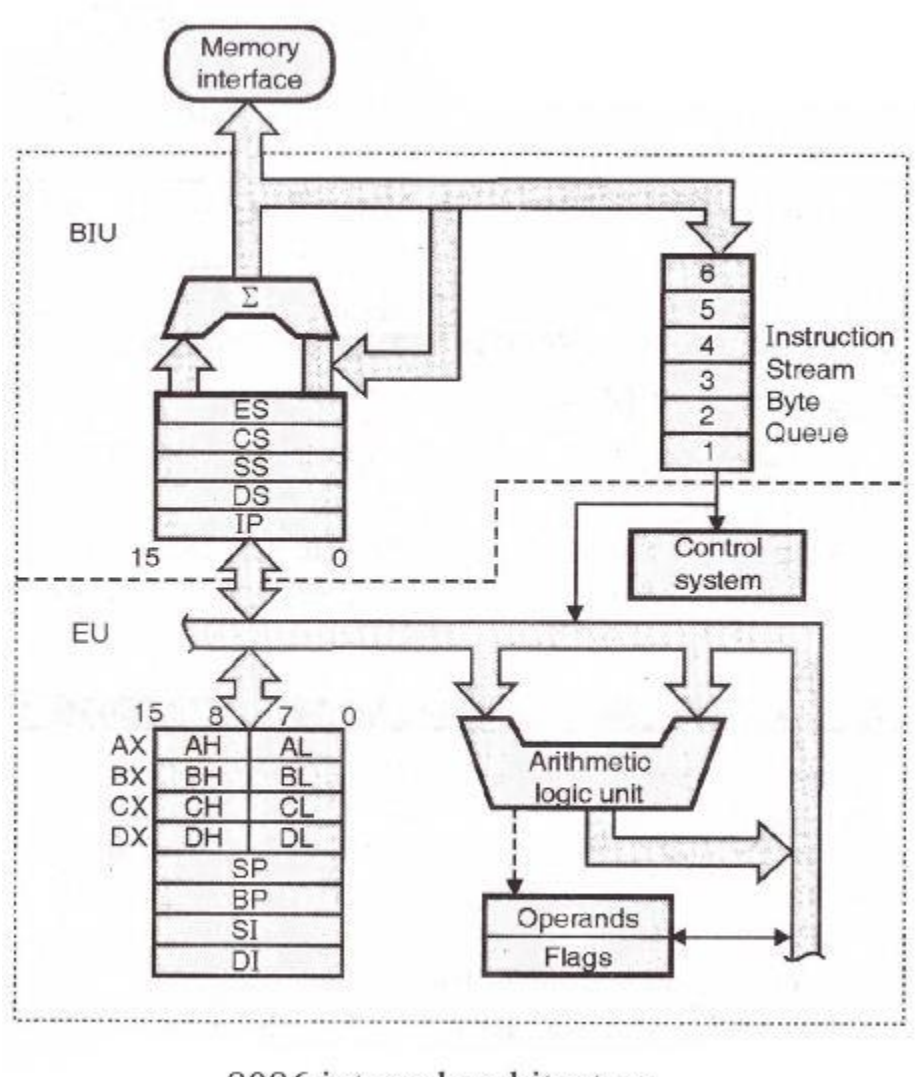
It is a 16-bit register that behaves like a flip-flop, i.e. it changes its status according to the result stored in the accumulator. It has 9 flags and they are divided into 2 groups − Conditional Flags and Control Flags.

Carry flag, Auxiliary flag, Parity flag, Zero flag, Sign flag, Overflow flag

Control Flags

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

Trap flag, Interrupt flag, Direction flag



8086 internal architecture

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 16bit 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.

Program

8 bit addition

```
MOV AL, 05H                    Move 1st 8-bit number to AL.
```

```
MOV BL, 03H          Move 2nd 8-bit number to BL.
ADD AL, BL           Add BL with AL.
HLT                  END
```

8 bit subtraction

8 bit addition

```
MOV AL, 05H          Move 1st 8-bit number to AL.
MOV BL, 03H          Move 2nd 8-bit number to BL.
SUB AL, BL           subtract BL with AL.
HLT                  END
```

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.

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.

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.

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.

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.

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.

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.

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.

Example

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

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.

Instructions for input and 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.

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.

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.

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.

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.
- **MOVS/MOVSB/MOVSW** − Used to move the byte/word from one string to another.
- **COMS/COMPSB/COMPSW** − Used to compare two string bytes/words.
- **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 and 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 −

- **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
- **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

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

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

Interrupts in 8086 microprocessor

An interrupt is a condition that halts the microprocessor temporarily to work on a different task and then return to its previous task. Interrupt is an event or signal that request to attention of CPU. This halt allows peripheral devices to access the microprocessor.

Whenever an interrupt occurs the processor completes the execution of the current instruction and starts the execution of an Interrupt Service Routine (ISR) or Interrupt Handler. ISR is a program that tells the processor what to do when the interrupt occurs. After the execution of ISR, control returns back to the main routine where it was interrupted.

In 8086 microprocessor following tasks are performed when microprocessor encounters an interrupt:

1. The value of flag register is pushed into the stack. It means that first the value of SP (Stack Pointer) is decremented by 2 then the value of flag register is pushed to the memory address of stack segment.
2. The value of starting memory address of CS (Code Segment) is pushed into the stack.
3. The value of IP (Instruction Pointer) is pushed into the stack.
4. IP is loaded from word location (Interrupt type) * 04.
5. CS is loaded from the next word location.
6. Interrupt and Trap flag are reset to 0.

The different types of interrupts present in 8086 microprocessor are given by:

1. **Hardware Interrupts –**
   Hardware interrupts are those interrupts which are caused by any peripheral device by sending a signal through a specified pin to the microprocessor. There are two hardware interrupts in 8086 microprocessor. They are:
   - *(A) NMI (Non Maskable Interrupt)* – It is a single pin non maskable hardware interrupt which cannot be disabled. It is the highest priority interrupt in 8086 microprocessor. After its execution, this interrupt generates a TYPE 2 interrupt. IP is loaded from word location 00008 H and CS is loaded from the word location 0000A H.
   - *(B) INTR (Interrupt Request)* – It provides a single interrupt request and is activated by I/O port. This interrupt can be masked or delayed. It is a level triggered interrupt. It can receive any interrupt

type, so the value of IP and CS will change on the interrupt type received.

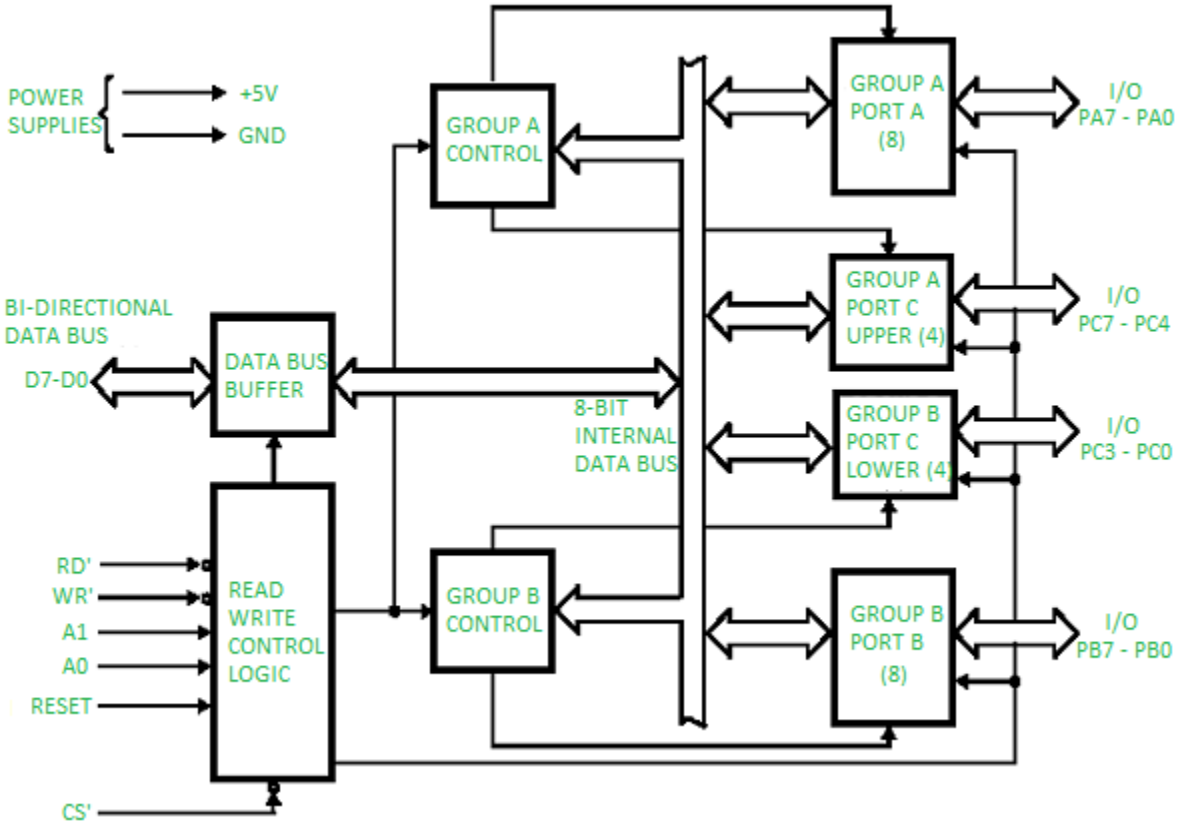
2. **Software Interrupts –** These are instructions that are inserted within the program to generate interrupts. There are 256 software interrupts in 8086 microprocessor. The instructions are of the format INT type where type ranges from 00 to FF. The starting address ranges from 00000 H to 003FF H. These are 2 byte instructions. IP is loaded from type * 04 H and CS is loaded from the next address give by (type * 04) + 02 H. Some important software interrupts are:
    - (A) *TYPE 0* corresponds to division by zero(0).
    - (B) *TYPE 1* is used for single step execution for debugging of program.
    - (C) *TYPE 2* represents NMI and is used in power failure conditions.
    - (D) *TYPE 3* represents a break-point interrupt.
    - (E) *TYPE 4* is the overflow interrupt.

Programmable peripheral interface 8255

PPI 8255 is a general purpose programmable I/O device designed to interface the CPU with its outside world such as ADC, DAC, keyboard etc. We can program it according to the given condition. It can be used with almost any microprocessor.

It consists of three 8-bit bidirectional I/O ports i.e. PORT A, PORT B and PORT C. We can assign different ports as input or output functions.

Block diagram –



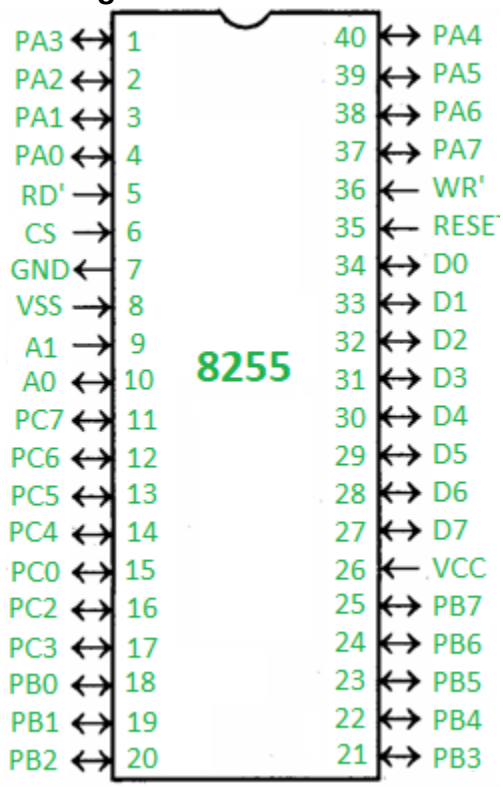
It consists of 40 pins and operates in +5V regulated power supply. Port C is further divided into two 4-bit ports i.e. port C lower and port C upper and port C can work in either BSR (bit set rest) mode or in mode 0 of input-output mode of 8255. Port B can work in either mode or in mode 1 of input-output mode. Port A can work either in mode 0, mode 1 or mode 2 of input-output mode.

It has two control groups, control group A and control group B. Control group A consist of port A and port C upper. Control group B consists of port C lower and port B.

Depending upon the value if CS', A1 and A0 we can select different ports in different modes as input-output function or BSR. This is done by writing a suitable word in control register (control word D0-D7).

CS'	A1	A0	SELECTION	ADDRESS
0	0	0	PORT A	80 H
0	0	1	PORT B	81 H
0	1	0	PORT C	82 H
0	1	1	Control Register	83 H
1	X	X	No Seletion	X

Pin diagram –

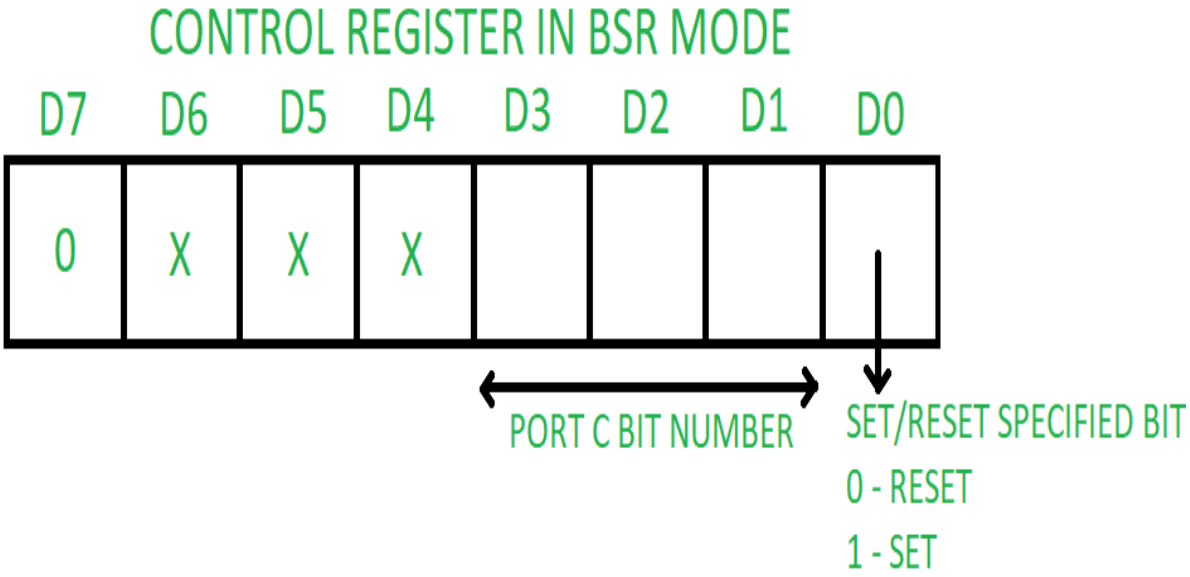


- **PA0 – PA7 –** Pins of port A
- **PB0 – PB7 –** Pins of port B
- **PC0 – PC7 –** Pins of port C
- **D0 – D7 –** Data pins for the transfer of data

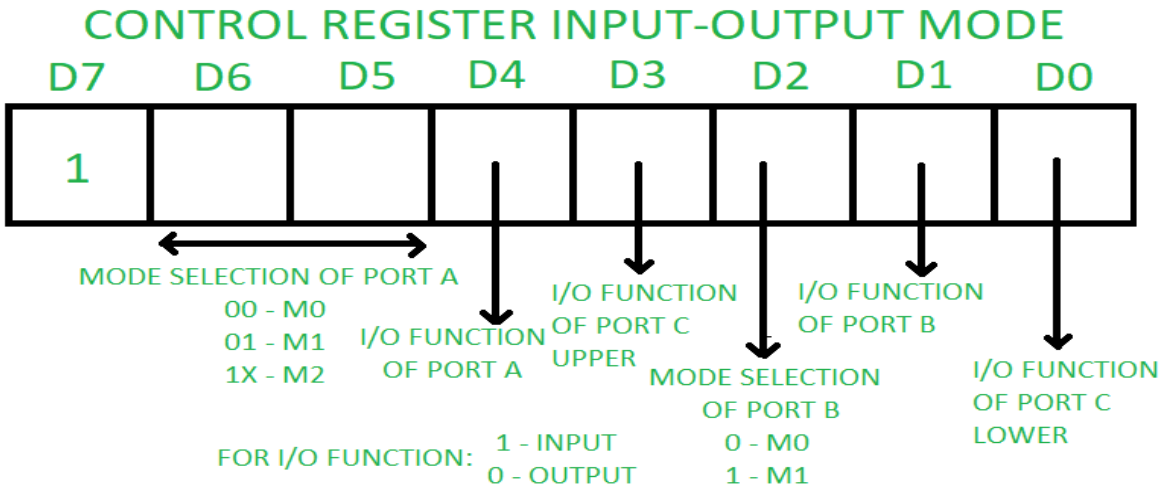
- **RESET –** Reset input
- **RD' –** Read input
- **WR' –** Write input
- **CS' –** Chip select
- **A1 and A0 –** Address pins

Operating modes –

1. **Bit set reset (BSR) mode –**
   If MSB of control word (D7) is 0, PPI works in BSR mode. In this mode only port C bits are used for set or reset.



2. **Input-Outpt mode –**
   If MSB of control word (D7) is 1, PPI works in input-output mode. This is further divided into three modes:



- **Mode 0 –**In this mode all the three ports (port A, B, C) can work as simple input function or simple output function. In this mode there is no interrupt handling capacity.

- **Mode 1 –** Handshake I/O mode or strobbed I/O mode. In this mode either port A or port B can work as simple input port or simple output port, and port C bits are used for handshake signals before actual data transmission. It has interrupt handling capacity and input and output are latched.
  Example: A CPU wants to transfer data to a printer. In this case since speed of processor is very fast as compared to relatively slow printer, so before actual data transfer it will send handshake signals to the printer for synchronization of the speed of the CPU and the peripherals.

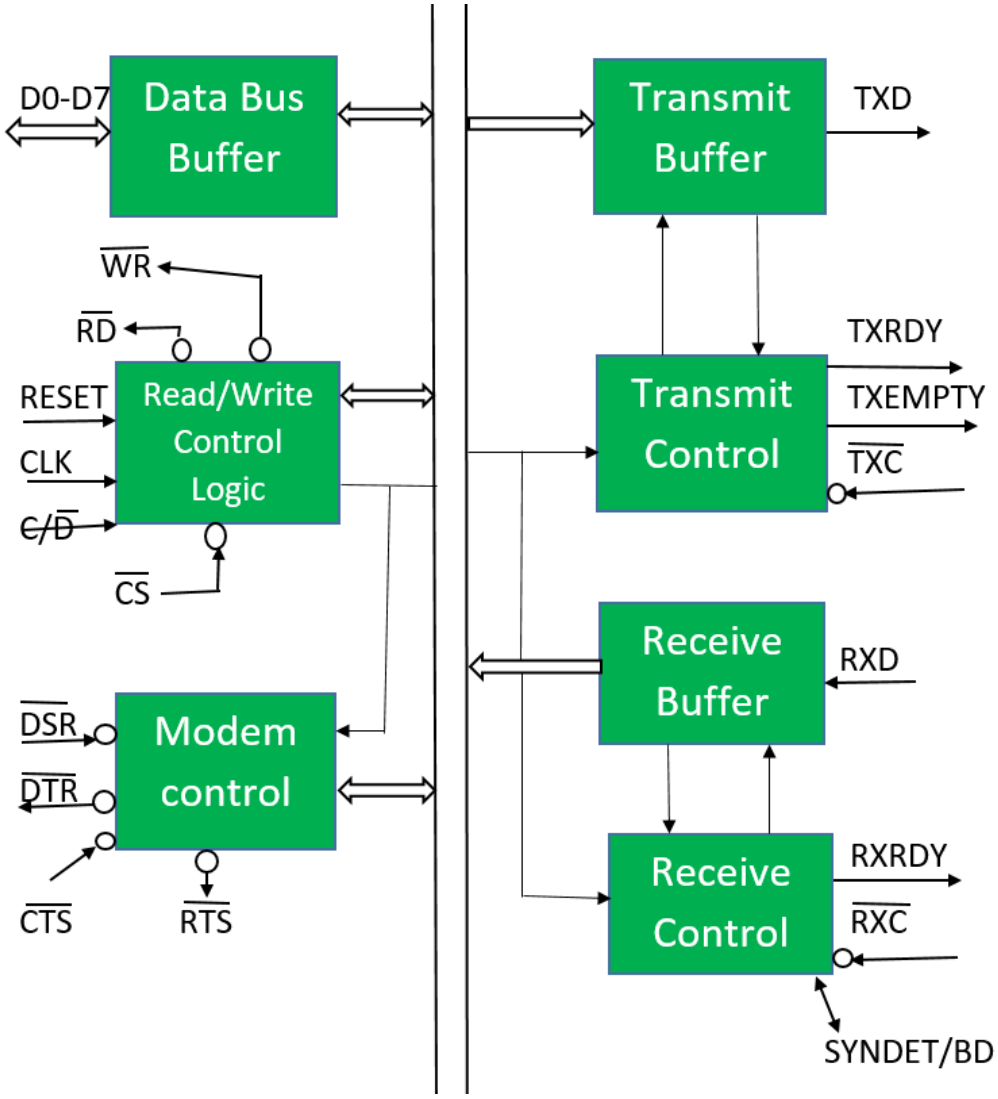
- **Mode 2 –** Bi-directional data bus mode. In this mode only port A works, and port B can work either in mode 0 or mode 1. 6 bits port C are used as handshake signals. It also has interrupt handling capacity.

8251 USART

8251 universal synchronous asynchronous receiver transmitter (USART) acts as a mediator between microprocessor and peripheral to transmit serial data into parallel form and vice versa.

1. It takes data serially from peripheral (outside devices) and converts into parallel data.
2. After converting the data into parallel form, it transmits it to the CPU.
3. Similarly, it receives parallel data from microprocessor and converts it into serial form.
4. After converting data into serial form, it transmits it to outside device (peripheral).

Block Diagram of 8251 USART –



It contains the following blocks:

1. **Data bus buffer –**
   This block helps in interfacing the internal data bus of 8251 to the system data bus. The data transmission is possible between 8251 and CPU by the data bus buffer block.
2. **Read/Write control logic –**
   It is a control block for overall device. It controls the overall working by selecting the operation to be done. The operation selection depends upon input signals as:

$\overline{CS}$	C/$\overline{D}$	$\overline{RD}$	$\overline{WR}$	Operation
1	X	X	X	Invalid
0	0	0	1	data CPU< ----- 8251
0	0	1	0	data CPU ----- > 8251
0	1	0	1	Status word CPU < ----------8251
0	1	1	0	Control word CPU---------- > 8251

In this way, this unit selects one of the three registers- data buffer register, control register, status register.

3. **Modem control (modulator/demodulator) –**
   A device converts analog signals to digital signals and vice-versa and helps the computers to communicate over telephone lines or cable wires. The following are active-low pins of Modem.
   - **DSR:** Data Set Ready signal is an input signal.
   - **DTR:** Data terminal Ready is an output signal.
   - **CTS:** It is an input signal which controls the data transmit circuit.
     RTS: It is an output signal which is used to set the status RTS.
4. **Transmit buffer –**
   This block is used for parallel to serial converter that receives a parallel byte for conversion into serial signal and further transmission onto the common channel.
   - **TXD:** It is an output signal, if its value is one, means transmitter will transmit the data.
5. **Transmit control –**
   This block is used to control the data transmission with the help of following pins:
   - **TXRDY:** It means transmitter is ready to transmit data character.
   - **TXEMPTY:** An output signal which indicates that TXEMPTY pin has transmitted all the data characters and transmitter is empty now.
   - **TXC:** An active-low input pin which controls the data transmission rate of transmitted data.
6. **Receive buffer –**
   This block acts as a buffer for the received data.
   - **RXD:** An input signal which receives the data.

7. **Receive control –**
   This block controls the receiving data.
   - **RXRDY:** An input signal indicates that it is ready to receive the data.
   - **RXC:** An active-low input signal which controls the data transmission rate of received data.
   - **SYNDET/BD:** An input or output terminal. External synchronous mode-input terminal and asynchronous mode-output terminal.

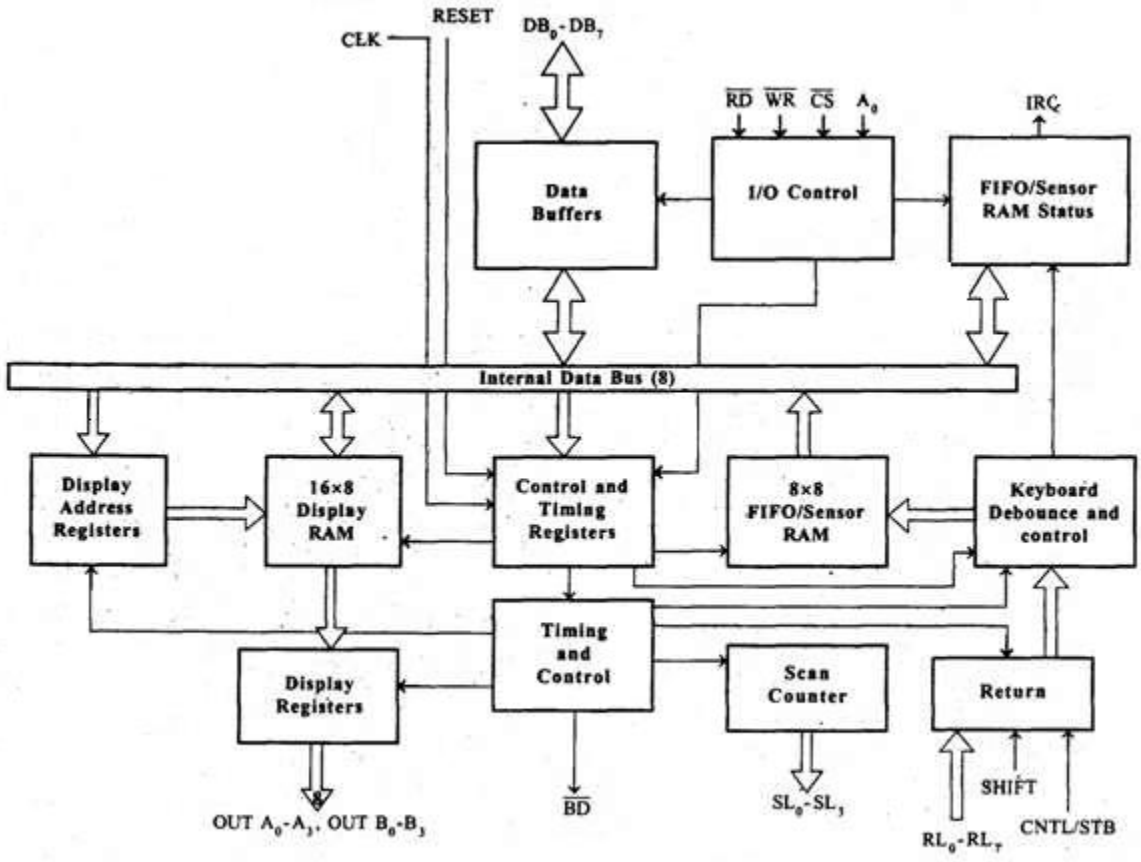
Keyboards Interface;

8279 programmable keyboard/display controller is designed by Intel that interfaces a keyboard with the CPU.

The keyboard consists of maximum 64 keys, which are interfaced with the CPU by using the key-codes. These key-codes are de-bounced and stored in an 8-byte FIFORAM, which can be accessed by the CPU. If more than 8 characters are entered in the FIFO, then it means more than eight keys are pressed at a time. This is when the overrun status is set.

If a FIFO contains a valid key entry, then the CPU is interrupted in an interrupt mode else the CPU checks the status in polling to read the entry. Once the CPU reads a key entry, then FIFO is updated, and the key entry is pushed out of the FIFO to generate space for new entries.

Architecture and Description



I/O Control and Data Buffer

This unit controls the flow of data through the microprocessor. It is enabled only when D is low. Its data buffer interfaces the external bus of the system with the internal bus of the microprocessor. The pins A0, RD, and WR are used for command, status or data read/write operations.

Control and Timing Register and Timing Control

This unit contains registers to store the keyboard, display modes, and other operations as programmed by the CPU. The timing and control unit handles the timings for the operation of the circuit.

Scan Counter

It has two modes i.e. **Encoded mode** and Decoded mode. In the encoded mode, the counter provides the binary count that is to be externally decoded to provide the scan lines for the keyboard and display.

In the **decoded scan mode**, the counter internally decodes the least significant 2 bits and provides a decoded 1 out of 4 scan on SL_0-SL_3.

Return Buffers, Keyboard Debounce, and Control

This unit first scans the key closure row-wise, if found then the keyboard debounce unit debounces the key entry. In case, the same key is detected, then the code of that key is directly transferred to the sensor RAM along with SHIFT & CONTROL key status.

FIFO/Sensor RAM and Status Logic

This unit acts as 8-byte first-in-first-out (FIFO) RAM where the key code of every pressed key is entered into the RAM as per their sequence. The status logic generates an interrupt request after each FIFO read operation till the FIFO gets empty.

In the scanned sensor matrix mode, this unit acts as sensor RAM where its each row is loaded with the status of their corresponding row of sensors into the matrix. When the sensor changes its state, the IRQ line changes to high and interrupts the CPU.

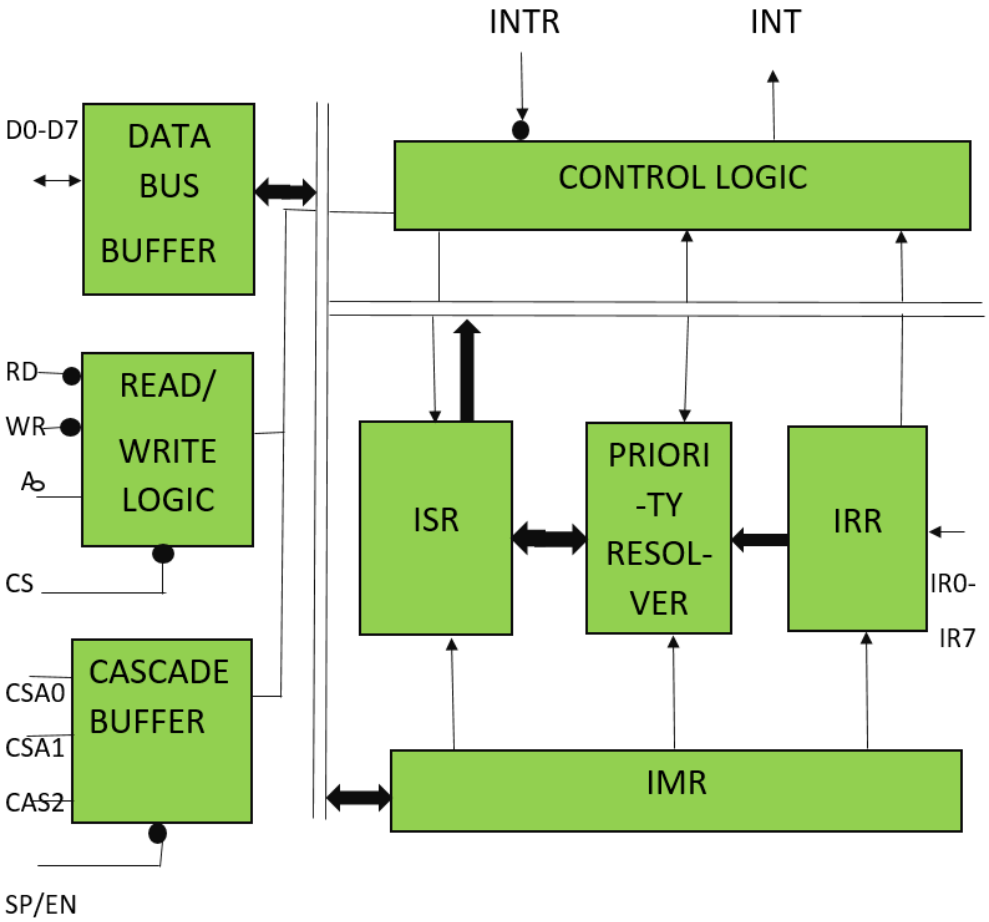
Display Address Registers and Display RAM

This unit consists of display address registers which holds the addresses of the word currently read/written by the CPU to/from the display RAM.

Programmable Interrupt Controller

8259 microprocessor is defined as Programmable Interrupt Controller **(PIC)** microprocessor. There are 5 hardware interrupts and 2 hardware interrupts in 8085 and 8086 respectively. But by connecting 8259 with CPU, we can increase the interrupt handling capability. 8259 combines the multi interrupt input sources into a single interrupt output. Interfacing of single PIC provides 8 interrupts inputs from IR0-IR7.

1. Features of 8259 PIC microprocessor – Intel 8259 is designed for Intel 8085 and Intel 8086 microprocessor.
2. It can be programmed either in level triggered or in edge triggered interrupt level.
3. We can masked individual bits of interrupt request register.
4. We can increase interrupt handling capability upto 64 interrupt level by cascading further 8259 PIC.
5. Clock cycle is not required.



The Block Diagram consists of 8 blocks which are – Data Bus Buffer, Read/Write Logic, Cascade Buffer Comparator, Control Logic, Priority Resolver and 3 registers- ISR, IRR, IMR.

1. **Data bus buffer –**
   This Block is used as a mediator between 8259 and 8085/8086 microprocessor by acting as a buffer. It takes the control word from the 8085 (let say) microprocessor and transfer it to the control logic of 8259 microprocessor. Also, after selection of Interrupt by 8259 microprocessor, it transfer the opcode of the selected Interrupt and address of the Interrupt service sub routine to the other connected microprocessor. The data bus buffer consists of 8 bits represented as D0-D7 in the block diagram. Thus, shows that a maximum of 8 bits data can be transferred at a time.
2. **Read/Write logic –**
   This block works only when the value of pin CS is low (as this pin is active low). This block is responsible for the flow of data depending upon the inputs of RD and WR. These two pins are active low pins used for read and write operations.
3. **Control logic –**
   It is the centre of the microprocessor and controls the functioning of every block. It

has pin INTR which is connected with other microprocessor for taking interrupt request and pin INT for giving the output. If 8259 is enabled, and the other microprocessor Interrupt flag is high then this causes the value of the output INT pin high and in this way 8259 responds to the request made by other microprocessor.

4. **Interrupt request register (IRR) –**
   It stores all the interrupt level which are requesting for Interrupt services.
5. **Interrupt service register (ISR) –**
   It stores the interrupt level which are currently being executed.
6. **Interrupt mask register (IMR) –**
   It stores the interrupt level which have to be masked by storing the masking bits of the interrupt level.
7. **Priority resolver –**
   It examines all the three registers and set the priority of interrupts and according to the priority of the interrupts, interrupt with highest priority is set in ISR register. Also, it reset the interrupt level which is already been serviced in IRR.
8. **Cascade buffer –**
   To increase the Interrupt handling capability, we can further cascade more number of pins by using cascade buffer. So, during increment of interrupt capability, CSA lines are used to control multiple interrupt structure.

SP/EN (Slave program/Enable buffer) pin is when set to high, works in master mode else in slave mode. In Non Buffered mode, SP/EN pin is used to specify whether 8259 work as master or slave and in Buffered mode, SP/EN pin is used as an output to enable data bus.

11.10.4 Analog to Digital Converter (ADC) Interfacing

In this section, we will see how we can interface ADC 0808 family and ADC 7109 to the 8085 system using 8255. Before going to see interfacing details we see the details of ADC 0808 and ADC 7109.

11.10.4.1 ADC 0808/0809 Family

The ADC 0808 and ADC 0809 are monolithic CMOS devices with an 8-channel multiplexer. These devices are also designed to operate from common microprocessor control buses, with tri-state output latches driving the data bus. The main features of these devices are :

Features

- 8-bit successive approximation ADC.
- 8-channel multiplexer with address logic.
- Conversion time 100 μs
- It eliminates the need for external zero and full-scale adjustments.

- Easy to interface to all microprocessors.
- It operates on single 5 V power supply.
- Output meet TTL voltage level specifications.

Fig. 11.33 shows pin diagram of 0808/0809 ADC.

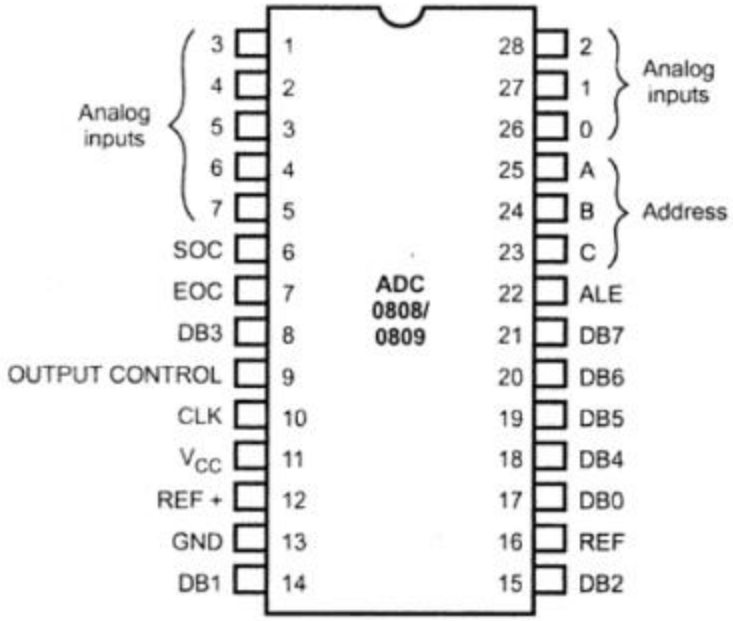
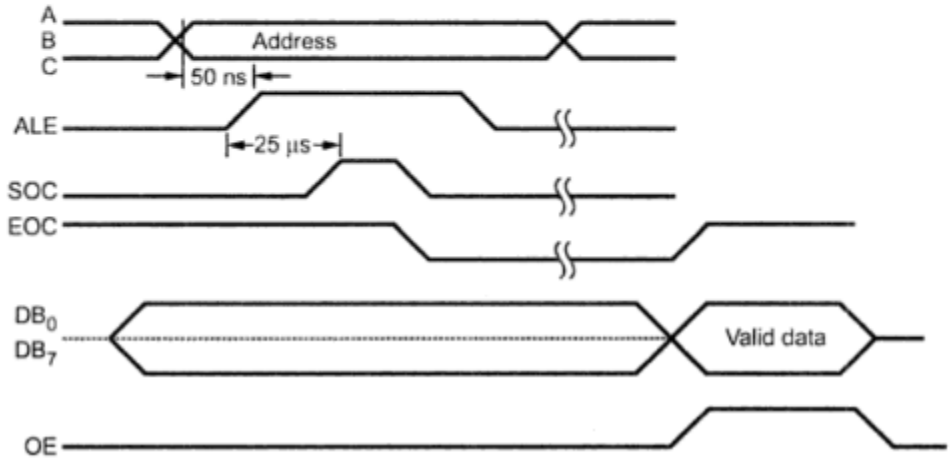


Fig. 11.33 Pin diagram of 0808/0809

ADC 0808/0809 has eight input channels, so to select desired input channel, it is necessary to send 3-bit address on A, B and C inputs. The address of the desired channel is sent to the multiplexer address inputs through port pins. After atleast 50 ns, this address must be latched. This can be achieved by sending ALE signal. After another 2.5 μs, the start of conversion (SOC) signal must be sent high and then low to start the conversion process. To indicate end of conversion ADC 0808/0809 activates EOC signal. The microprocessor system can read converted digital word through data bus by enabling the output enable signal after EOC is activated. This is illustrated in Fig. 11.34.



IC 1408 D/A Converter Devices

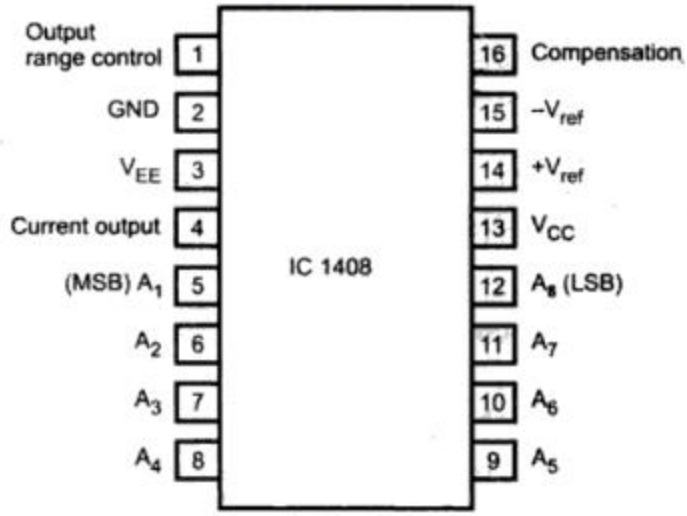
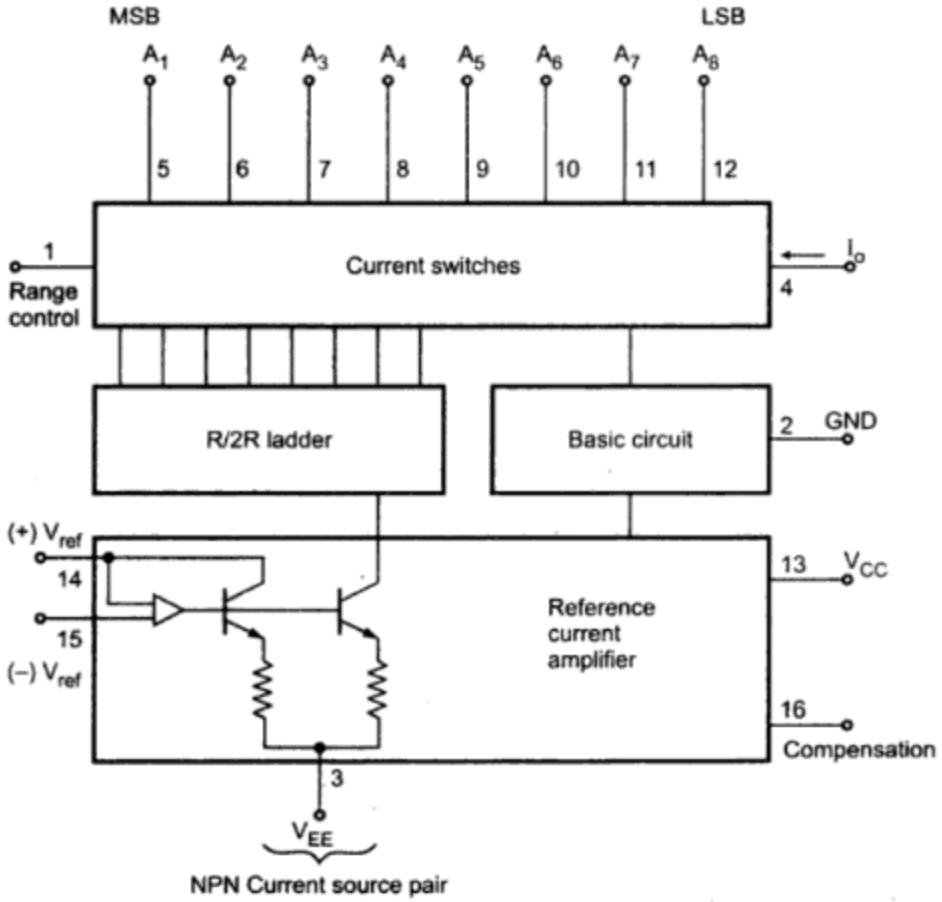


Fig. 11.13 (a) Pin diagram

The 1408 is an 8 bit R/2R ladder type D/A converter compatible with TTL and CMOS logic. It is designed to use where the output current is linear product of an eight bit digital word.

Fig. 11.13 shows the pin diagram and block diagram for IC 1408 DAC.

The IC 1408 consists of a reference current amplifier, an R/2R ladder and eight high speed current switches. It has eight input data lines A_1 (MSB) through A_8 (LSB) which control the positions of current switches.



It requires 2 mA reference current for full scale input and two power supplies V_{CC} = + 5 V and V_{EE} = – 15 V (V_{EE} can range from – 5 V to – 15 V).

The voltage V_{ref} and resistor R_{14} determines the total reference current source and R_{15} is generally equal to R_{14} to match the input impedance of the reference current amplifier.

Fig. 11.14 (See Fig. 11.14 on next page) shows a typical circuit for IC 1408.

The output current I_o can be given as

$$I_o = \frac{V_{ref}}{R_{14}}\left(\frac{A_1}{2} + \frac{A_2}{4} + \frac{A_3}{8} + \frac{A_4}{16} + \frac{A_5}{32} + \frac{A_6}{64} + \frac{A_7}{128} + \frac{A_8}{256}\right) \quad \ldots(16)$$

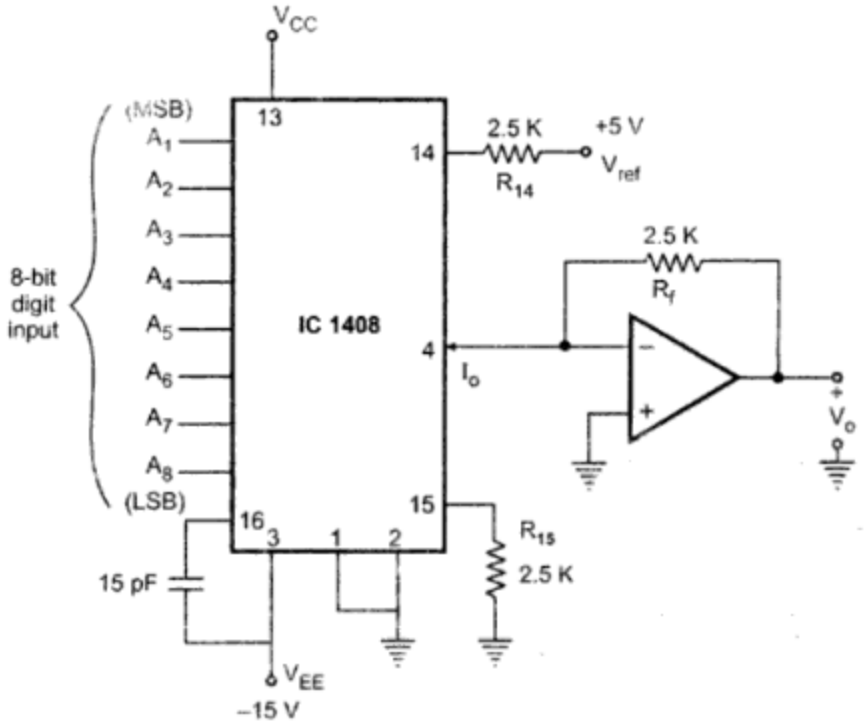
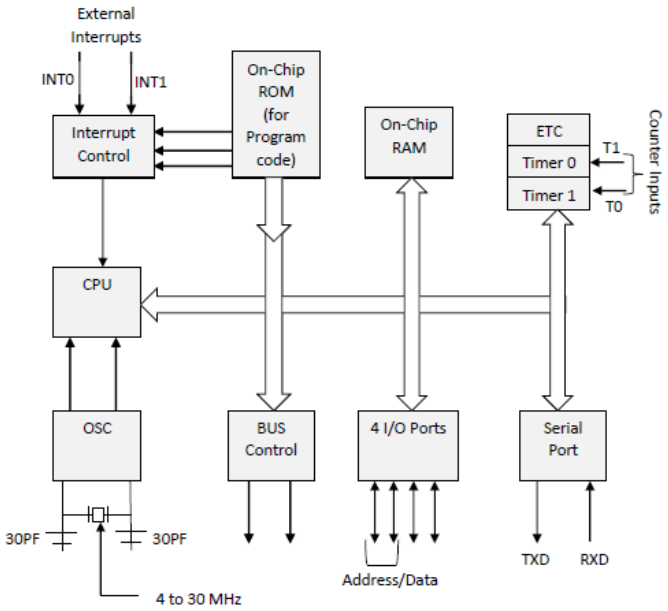


Fig. 11.14 Typical circuit for IC 1408

8051 Microcontroller Architecture

The 8051 Microcontroller was designed in 1980's by Intel. Its foundation was on Harvard Architecture and was developed principally for bringing into play in Embedded Systems. There are two buses in 8051 Microcontroller one for program and other for data. As a result, it has two storage rooms for both program and data of 64K by 8 size. The microcontroller comprise of 8 bit accumulator & 8 bit processing unit. It also consists of 8 bit B register as majorly functioning blocks and 8051 microcontroller programming is done with embedded C language using Keil software. It also has a number of other 8 bit and 16 bit registers.

For internal functioning & processing Microcontroller 8051 comes with integrated built-in RAM. This is prime memory and is employed for storing temporary data. It is unpredictable memory i.e. its data can get be lost when the power supply to the Microcontroller switched OFF.



CPU (Central Processor Unit):

As you may be familiar that Central Processor Unit or CPU is the mind of any processing machine. It scrutinizes and manages all processes that are carried out in the Microcontroller. User has no power over the functioning of CPU. It interprets program printed in storage space (ROM) and carries out all of them and do the projected duty. CPU manages different types of registers in 8051 microcontroller.

Interrupts:

As the heading put forward, Interrupt is a sub-routine call that reads the Microcontroller's key function or job and helps it to perform some other program which is extra important at that point of time. The characteristic of 8051 Interrupt is extremely constructive as it aids in emergency cases. Interrupts provides us a method to postpone or delay the current process, carry out a sub-routine task and then all over again restart standard program implementation.

The Micro-controller 8051 can be assembled in such a manner that it momentarily stops or break the core program at the happening of interrupt. When sub-routine task is finished then the implementation of core program initiates automatically as usual. There are 5 interrupt supplies in 8051 Microcontroller, two out of five are peripheral interrupts, two are timer interrupts and one is serial port interrupt.

Memory:

Micro-controller needs a program which is a set of commands. This program enlightens Microcontroller to perform precise tasks. These programs need a storage space on which they can be accumulated and interpret by Microcontroller to act upon any specific process. The memory which is brought into play to accumulate the program of Microcontroller is recognized as Program memory or code memory. In common language it's also known as Read Only Memory or ROM.

Microcontroller also needs a memory to amass data or operands for the short term. The storage space which is employed to momentarily data storage for functioning is acknowledged as Data Memory and we employ Random Access Memory or RAM for this principle reason. Microcontroller 8051 contains code memory or program memory 4K so that is has 4KB Rom and it also comprise of data memory (RAM) of 128 bytes.

Bus:

Fundamentally Bus is a group of wires which functions as a communication canal or mean for the transfer Data. These buses comprise of 8, 16 or more cables. As a result, a bus can bear 8 bits, 16 bits all together. There are two types of buses:

1. **Address Bus:** Microcontroller 8051 consists of 16 bit address bus. It is brought into play to address memory positions. It is also utilized to transmit the address from Central Processing Unit to Memory.
2. **Data Bus:** Microcontroller 8051 comprise of 8 bits data bus. It is employed to cart data.

Oscillator:

As we all make out that Microcontroller is a digital circuit piece of equipment, thus it needs timer for its function. For this function, Microcontroller 8051 consists of an on-chip oscillator which toils as a time source for CPU (Central Processing Unit). As the productivity thumps of oscillator are steady as a result, it facilitates harmonized employment of all pieces of 8051 Microcontroller. Input/output Port: As we are acquainted with that Microcontroller is employed in embedded systems to manage the functions of devices.

Applications of 8051 Microcontroller:

The microcontroller 8051 applications include large amount of machines, principally because it is simple to incorporate in a project or to assemble a machine around it. The following are the key spots of spotlight:

1. **Energy Management:** Competent measuring device systems aid in calculating energy consumption in domestic and industrialized applications. These meter systems are prepared competent by integrating microcontrollers.
2. **Touch screens:** A high degree of microcontroller suppliers integrate touch sensing abilities in their designs. Transportable devices such as media players, gaming devices & cell phones are some illustrations of micro-controller integrated with touch sensing screens.
3. **Automobiles:** The microcontroller 8051 discovers broad recognition in supplying automobile solutions. They are extensively utilized in hybrid motor vehicles to control engine variations. In addition, works such as cruise power and anti-brake mechanism has created it more capable with the amalgamation of micro-controllers.
4. **Medical Devices:** Handy medicinal gadgets such as glucose & blood pressure monitors bring into play micro-controllers, to put on view the measurements, as a result, offering higher dependability in giving correct medical results.
5. **Medical Devices:** Handy medicinal gadgets such as glucose & blood pressure monitors bring into play micro-controllers, to put on view the measurements, as a result, offering higher dependability in giving correct medical results.

Addressing mode:

In this section, we will see different addressing modes of the 8051 microcontrollers. In 8051 there are 1-byte, 2-byte instructions and very few 3-byte instructions are present. The opcodes are 8-bit long. As the opcodes are 8-bit data, there are 256 possibilities. Among 256, 255 opcodes are implemented.

The clock frequency is12MHz, so 64 instruction types are executed in just 1 µs, and rest are just 2 µs. The Multiplication and Division operations take 4 µsto to execute.

In 8051 There are six types of addressing modes.

- Immediate AddressingMode
- Register AddressingMode
- Direct AddressingMode
- Register IndirectAddressing Mode
- Indexed AddressingMode
- Implied AddressingMode

Immediate addressing mode

In this Immediate Addressing Mode, the data is provided in the instruction itself. The data is provided immediately after the opcode. These are some examples of Immediate Addressing Mode.

```
MOVA, #0AFH;
```

```
MOVR3, #45H;

MOVDPTR, #FE00H;
```

In these instructions, the # symbol is used for immediate data. In the last instruction, there is DPTR. The DPTR stands for Data Pointer. Using this, it points the external data memory location. In the first instruction, the immediate data is AFH, but one 0 is added at the beginning. So when the data is starting with A to F, the data should be preceded by 0.

Register addressing mode

In the register addressing mode the source or destination data should be present in a register (R0 to R7). These are some examples of RegisterAddressing Mode.

```
MOVA, R5;

MOVR2, #45H;

MOVR0, A;
```

In 8051, there is no instruction like **MOVR5, R7**. But we can get the same result by using this instruction **MOV R5, 07H**, or by using **MOV 05H, R7**. But this two instruction will work when the selected register bank is **RB0**. To use another register bank and to get the same effect, we have to add the starting address of that register bank with the register number. For an example, if the RB2 is selected, and we want to access R5, then the address will be (10H + 05H = 15H), so the instruction will look like this **MOV 15H, R7**. Here 10H is the starting address of Register Bank 2.

Direct Addressing Mode

In the Direct Addressing Mode, the source or destination address is specified by using 8-bit data in the instruction. Only the internal data memory can be used in this mode. Here some of the examples of direct Addressing Mode.

```
MOV80H, R6;

MOVR2, 45H;

MOVR0, 05H;
```

The first instruction will send the content of registerR6 to port P0 (Address of Port 0 is 80H). The second one is forgetting content from 45H to R2. The third one is used to get data from Register R5 (When register bank RB0 is selected) to register R5.

Register indirect addressing Mode

In this mode, the source or destination address is given in the register. By using register indirect addressing mode, the internal or external addresses can be accessed. The R0 and R1 are used for 8-bit addresses, and DPTR is used for 16-bit addresses, no other registers can be used for addressing purposes. Let us see some examples of this mode.

```
MOV0E5H, @R0;

MOV@R1, 80H
```

In the instructions, the @ symbol is used for register indirect addressing. In the first instruction, it is showing that theR0 register is used. If the content of R0 is 40H, then that instruction will take the data which is located at location 40H of the internal RAM. In the second one, if the content of R1 is 30H, then it indicates that the content of port P0 will be stored at location 30H in the internal RAM.

```
MOVXA, @R1;

MOV@DPTR, A;
```

In these two instructions, the X in MOVX indicates the external data memory. The external data memory can only be accessed in register indirect mode. In the first instruction if the R0 is holding 40H, then A will get the content of external RAM location40H. And in the second one, the content of A is overwritten in the location pointed by DPTR.

Indexed addressing mode

In the indexed addressing mode, the source memory can only be accessed from program memory only. The destination operand is always the register A. These are some examples of Indexed addressing mode.

```
MOVCA, @A+PC;

MOVCA, @A+DPTR;
```

The C in MOVC instruction refers to code byte. For the first instruction, let us consider A holds 30H. And the PC value is1125H. The contents of program memory location 1155H (30H + 1125H) are moved to register A.

Implied Addressing Mode

In the implied addressing mode, there will be a single operand. These types of instruction can work on specific registers only. These types of instructions are also known as register specific instruction. Here are some examples of Implied Addressing Mode.

```
RLA;

SWAPA;
```

These are 1- byte instruction. The first one is used to rotate the A register content to the Left. The second one is used to swap the nibbles in A.

Instructions in 8051 Microcontroller Instruction Set

Before seeing the types of instructions, let us see the structure of the 8051 Microcontroller Instruction. An 8051 Instruction consists of an Opcode (short of Operation – Code) followed by Operand(s) of size Zero Byte, One Byte or Two Bytes.

The Op-Code part of the instruction contains the Mnemonic, which specifies the type of operation to be performed. All Mnemonics or the Opcode part of the instruction are of One Byte size.

Coming to the Operand part of the instruction, it defines the data being processed by the instructions. The operand can be any of the following:

- No Operand
- Data value
- I/O Port
- Memory Location
- CPU register

There can multiple operands and the format of instruction is as follows:

MNEMONIC DESTINATION OPERAND, SOURCE OPERAND

A simple instruction consists of just the opcode. Other instructions may include one or more operands. Instruction can be one-byte instruction, which contains only opcode, or two-byte instructions, where the second byte is the operand or three byte instructions, where the operand makes up the second and third byte.

Based on the operation they perform, all the instructions in the 8051 Microcontroller Instruction Set are divided into five groups. They are:

- Data Transfer Instructions
- Arithmetic Instructions
- Logical Instructions
- Boolean or Bit Manipulation Instructions
- Program Branching Instructions

Data Transfer Instructions

The Data Transfer Instructions are associated with transfer with data between registers or external program memory or external data memory. The Mnemonics associated with Data Transfer are given below.

- *MOV*
- *MOVC*
- *MOVX*
- *PUSH*
- *POP*
- *XCH*
- *XCHD*

The following table lists out all the possible data transfer instruction along with other details like addressing mode, size occupied and number machine cycles it takes.

Mnemonic	Instruction	Description	Addressing Mode	# of Bytes	# of Cycles
MOV	A, #Data	A ← Data	Immediate	2	1
	A, Rn	A ← Rn	Register	1	1
	A, Direct	A ← (Direct)	Direct	2	1
	A, @Ri	A ← @Ri	Indirect	1	1
	Rn, #Data	Rn ← data	Immediate	2	1
	Rn, A	Rn ← A	Register	1	1
	Rn, Direct	Rn ← (Direct)	Direct	2	2
	Direct, A	(Direct) ← A	Direct	2	1
	Direct, Rn	(Direct) ← Rn	Direct	2	2
	Direct1, Direct2	(Direct1) ← (Direct2)	Direct	3	2
	Direct, @Ri	(Direct) ← @Ri	Indirect	2	2
	Direct, #Data	(Direct) ← #Data	Direct	3	2
	@Ri, A	@Ri ← A	Indirect	1	1
	@Ri, Direct	@Ri ← Direct	Indirect	2	2
	@Ri, #Data	@Ri ← #Data	Indirect	2	1
	DPTR, #Data16	DPTR ← #Data16	Immediate	3	2
MOVC	A, @A+DPTR	A ← Code Pointed by A+DPTR	Indexed	1	2
	A, @A+PC	A ← Code Pointed by A+PC	Indexed	1	2
	A, @Ri	A ← Code Pointed by Ri (8-bit Address)	Indirect	1	2
MOVX	A, @DPTR	A ← External Data Pointed by DPTR	Indirect	1	2
	@Ri, A	@Ri ← A (External Data 8-bit Addr)	Indirect	1	2
	@DPTR, A	@DPTR ← A (External Data 16-bit Addr)	Indirect	1	2
PUSH	Direct	Stack Pointer SP ← (Direct)	Direct	2	2
POP	Direct	(Direct) ← Stack Pointer SP	Direct	2	2
XCH	Rn	Exchange ACC with Rn	Register	1	1
	Direct	Exchange ACC with Direct Byte	Direct	2	1
	@Ri	Exchange ACC with Indirect RAM	Indirect	1	1
XCHD	A, @Ri	Exchange ACC with Lower Order Indirect RAM	Indirect	1	1

Arithmetic Instructions

Using Arithmetic Instructions, you can perform addition, subtraction, multiplication and division. The arithmetic instructions also include increment by one, decrement by one and a special instruction called Decimal Adjust Accumulator.

The Mnemonics associated with the Arithmetic Instructions of the 8051 Microcontroller Instruction Set are:

- *ADD*
- *ADDC*
- *SUBB*
- *INC*
- *DEC*
- *MUL*
- *DIV*
- *DA A*

The arithmetic instructions has no knowledge about the data format i.e. signed, unsigned, ASCII, BCD, etc. Also, the operations performed by the arithmetic instructions affect flags like carry, overflow, zero, etc. in the PSW Register.

All the possible Mnemonics associated with Arithmetic Instructions are mentioned in the following table.

Mnemonic	Instruction	Description	Addressing Mode	# of Bytes	# of Cycles
ADD	A, #Data	A ← A + Data	Immediate	2	1
	A, Rn	A ← A + Rn	Register	1	1
	A, Direct	A ← A + (Direct)	Direct	2	1
	A, @Ri	A ← A + @Ri	Indirect	1	1
ADDC	A, #Data	A ← A + Data + C	Immediate	2	1
	A, Rn	A ← A + Rn + C	Register	1	1
	A, Direct	A ← A + (Direct) + C	Direct	2	1
	A, @Ri	A ← A + @Ri + C	Indirect	1	1
SUBB	A, #Data	A ← A – Data – C	Immediate	2	1
	A, Rn	A ← A – Rn – C	Register	1	1
	A, Direct	A ← A – (Direct) – C	Direct	2	1
	A, @Ri	A ← A – @Ri – C	Indirect	1	1
MUL	AB	Multiply A with B (A ← Lower Byte of A*B and B ← Higher Byte of A*B)	--	1	4
DIV	AB	Divide A by B (A ← Quotient and B ← Remainder)	--	1	4
DEC	A	A ← A – 1	Register	1	1
	Rn	Rn ← Rn – 1	Register	1	1
	Direct	(Direct) ← (Direct) – 1	Direct	2	1
	@Ri	@Ri ← @Ri – 1	Indirect	1	1
INC	A	A ← A + 1	Register	1	1
	Rn	Rn ← Rn + 1	Register	1	1
	Direct	(Direct) ← (Direct) + 1	Direct	2	1
	@Ri	@Ri ← @Ri + 1	Indirect	1	1
	DPTR	DPTR ← DPTR + 1	Register	1	2
DA	A	Decimal Adjust Accumulator	--	1	1

Logical Instructions

The next group of instructions are the Logical Instructions, which perform logical operations like AND, OR, XOR, NOT, Rotate, Clear and Swap. Logical Instruction are performed on Bytes of data on a bit-by-bit basis.

Mnemonics associated with Logical Instructions are as follows:

- *ANL*
- *ORL*
- *XRL*
- *CLR*
- *CPL*
- *RL*
- *RLC*
- *RR*
- *RRC*
- *SWAP*

The following table shows all the possible Mnemonics of the Logical Instructions.

Mnemonic	Instruction	Description	Addressing Mode	# of Bytes	# of Cycles
ANL	A, #Data	A ← A AND Data	Immediate	2	1
	A, Rn	A ← A AND Rn	Register	1	1
	A, Direct	A ← A AND (Direct)	Direct	2	1
	A, @Ri	A ← A AND @Ri	Indirect	1	1
	Direct, A	(Direct) ← (Direct) AND A	Direct	2	1
	Direct, #Data	(Direct) ← (Direct) AND #Data	Direct	3	2
ORL	A, #Data	A ← A OR Data	Immediate	2	1
	A, Rn	A ← A OR Rn	Register	1	1
	A, Direct	A ← A OR (Direct)	Direct	2	1
	A, @Ri	A ← A OR @Ri	Indirect	1	1
	Direct, A	(Direct) ← (Direct) OR A	Direct	2	1
	Direct, #Data	(Direct) ← (Direct) OR #Data	Direct	3	2
XRL	A, #Data	A ← A XRL Data	Immediate	2	1
	A, Rn	A ← A XRL Rn	Register	1	1
	A, Direct	A ← A XRL (Direct)	Direct	2	1
	A, @Ri	A ← A XRL @Ri	Indirect	1	1
	Direct, A	(Direct) ← (Direct) XRL A	Direct	2	1
	Direct, #Data	(Direct) ← (Direct) XRL #Data	Direct	3	2
CLR	A	A← 00H	--	1	1
CPL	A	A ← A	--	1	1
RL	A	Rotate ACC Left	--	1	1
RLC	A	Rotate ACC Left through Carry	--	1	1
RR	A	Rotate ACC Right	--	1	1
RRC	A	Rotate ACC Right through Carry	--	1	1
SWAP	A	Swap Nibbles within ACC	--	1	1

Boolean or Bit Manipulation Instructions

As the name suggests, Boolean or Bit Manipulation Instructions will deal with bit variables. We know that there is a special bit-addressable area in the RAM and some of the Special Function Registers (SFRs) are also bit addressable.

The Mnemonics corresponding to the Boolean or Bit Manipulation instructions are:

- *CLR*
- *SETB*
- *MOV*
- *JC*
- *JNC*
- *JB*
- *JNB*
- *JBC*
- *ANL*
- *ORL*
- *CPL*

These instructions can perform set, clear, and, or, complement etc. at bit level. All the possible mnemonics of the Boolean Instructions are specified in the following table.

Mnemonic	Instruction	Description	# of Bytes	# of Cycles
CLR	C	C ← 0 (C = Carry Bit)	1	1
	Bit	Bit ← 0 (Bit = Direct Bit)	2	1
SET	C	C ← 1	1	1
	Bit	Bit ← 1	2	1
CPL	C	C ← $\overline{C}$	1	1
	Bit	Bit ← $\overline{Bit}$	2	1
ANL	C, /Bit	C ← C. $\overline{Bit}$ (AND)	2	1
	C, Bit	C ← C . Bit (AND)	2	1
ORL	C, /Bit	C ← C + $\overline{Bit}$ (OR)	2	1
	C, Bit	C ← C + Bit (OR)	2	1
MOV	C, Bit	C ← Bit	2	1
	Bit, C	Bit ← C	2	2
JC	rel	Jump is Carry (C) is Set	2	2
JNC	rel	Jump is Carry (C) is Not Set	2	2
JB	Bit, rel	Jump is Direct Bit is Set	3	2
JNB	Bit, rel	Jump is Direct Bit is Not Set	3	2
JBC	Bit, rel	Jump is Direct Bit is Set and Clear Bit	3	2

Program Branching Instructions

The last group of instructions in the 8051 Microcontroller Instruction Set are the Program Branching Instructions. These instructions control the flow of program logic. The mnemonics of the Program Branching Instructions are as follows.

- *LJMP*
- *AJMP*
- *SJMP*
- *JZ*
- *JNZ*
- *CJNE*
- *DJNZ*
- *NOP*
- *LCALL*
- *ACALL*
- *RET*
- *RETI*
- *JMP*

All these instructions, except the NOP (No Operation) affect the Program Counter (PC) in one way or other. Some of these instructions has decision making capability before transferring control to other part of the program.

The following table shows all the mnemonics with respect to the program branching instructions.

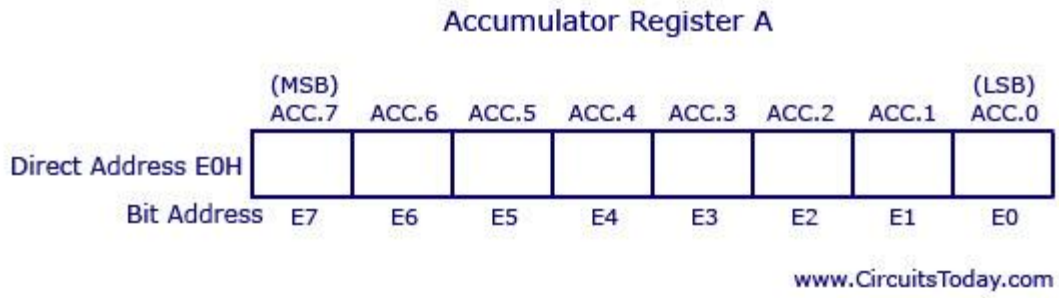
Mnemonic	Instruction	Description	# of Bytes	# of Cycles
ACALL	ADDR11	Absolute Subroutine Call PC + 2 → (SP); ADDR11 → PC	2	2
LCALL	ADDR16	Long Subroutine Call PC + 3 → (SP); ADDR16 → PC	3	2
RET	--	Return from Subroutine (SP) → PC	1	2
RETI	--	Return from Interrupt	1	2
AJMP	ADDR11	Absolute Jump ADDR11 → PC	2	2
LJMP	ADDR16	Long Jump ADDR16 → PC	3	2
SJMP	rel	Short Jump PC + 2 + rel → PC	2	2
JMP	@A + DPTR	A + DPTR → PC	1	2
JZ	rel	If A=0, Jump to PC + rel	2	2
JNZ	rel	If A ≠ 0, Jump to PC + rel		
CJNE	A, Direct, rel	Compare (Direct) with A. Jump to PC + rel if not equal	3	2
	A, #Data, rel	Compare #Data with A. Jump to PC + rel if not equal	3	2
	Rn, #Data, rel	Compare #Data with Rn. Jump to PC + rel if not equal	3	2
	@Ri, #Data, rel	Compare #Data with @Ri. Jump to PC + rel if not equal	3	2
DJNZ	Rn, rel	Decrement Rn. Jump to PC + rel if not zero	2	2
	Direct, rel	Decrement (Direct). Jump to PC + rel if not zero	3	2
NOP		No Operation	1	1

8051 Special Function Registers and Ports

There **are 21 Special function registers (SFR)** in 8051 micro controller and this includes Register A, Register B, Processor Status Word (PSW), PCON etc etc. There are 21 unique locations for these 21 special function registers and each of these register is of 1 byte size. Some of these special function registers ***are bit addressable*** (which means you can access 8 individual bits inside a single byte), while some others are ***only byte addressable***. Let's take a look at them in detail.

Register A/Accumulator

The most important of all special function registers, that's the first comment about **Accumulator** which is also known as **ACC** or **A.** The Accumulator (sometimes referred to as Register A also) holds the result of most of arithmetic and logic operations. ACC is usually accessed by direct addressing and its physical address is **E0H.** Accumulator is ***both byte and bit addressable.*** You can understand this from the figure shown below. To access the first bit (i.e bit 0) or to access accumulator as a single byte (all 8 bits at once), you may use the same physical address E0H. Now if you want to access the second bit (i.e bit 1), you may use E1H and for third bit E2H and so on.



Register B

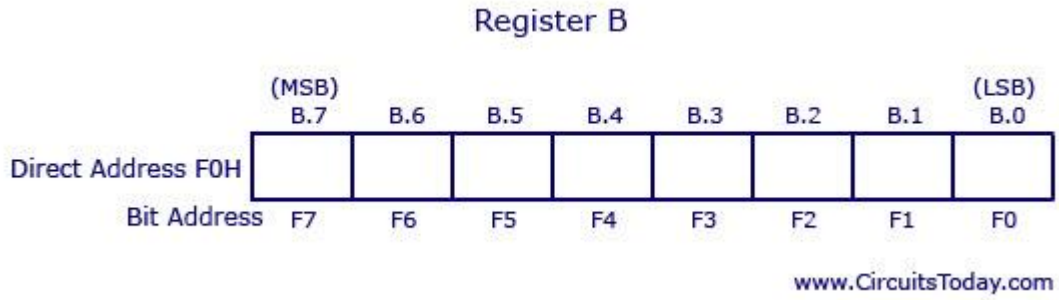
The major purpose of this register is in executing multiplication and division. The 8051 micro controller has a single instruction for multiplication **(MUL)** and division **(DIV)**. If you are familiar with 8085, you may now know that multiplication is repeated addition, where as division is repeated subtraction. While programming 8085, you may have written a loop to execute repeated addition/subtraction to perform multiplication and division. Now here in 8051 you can do this with a single instruction.

Ex: MUL A,B – When this instruction is executed, data inside A and data inside B is multiplied and answer is stored in A.

Note: For MUL and DIV instructions, it is necessary that the two operands must be in A and B.

Note: Follow this link if you are interested in knowing about **differences between a microprocessor and microcontroller**.

Register B is also byte addressable and bit addressable. To access bit o or to access all 8 bits (as a single byte), physical address F0 is used. To access bit 1 you may use F1 and so on. Please take a look at the picture below.

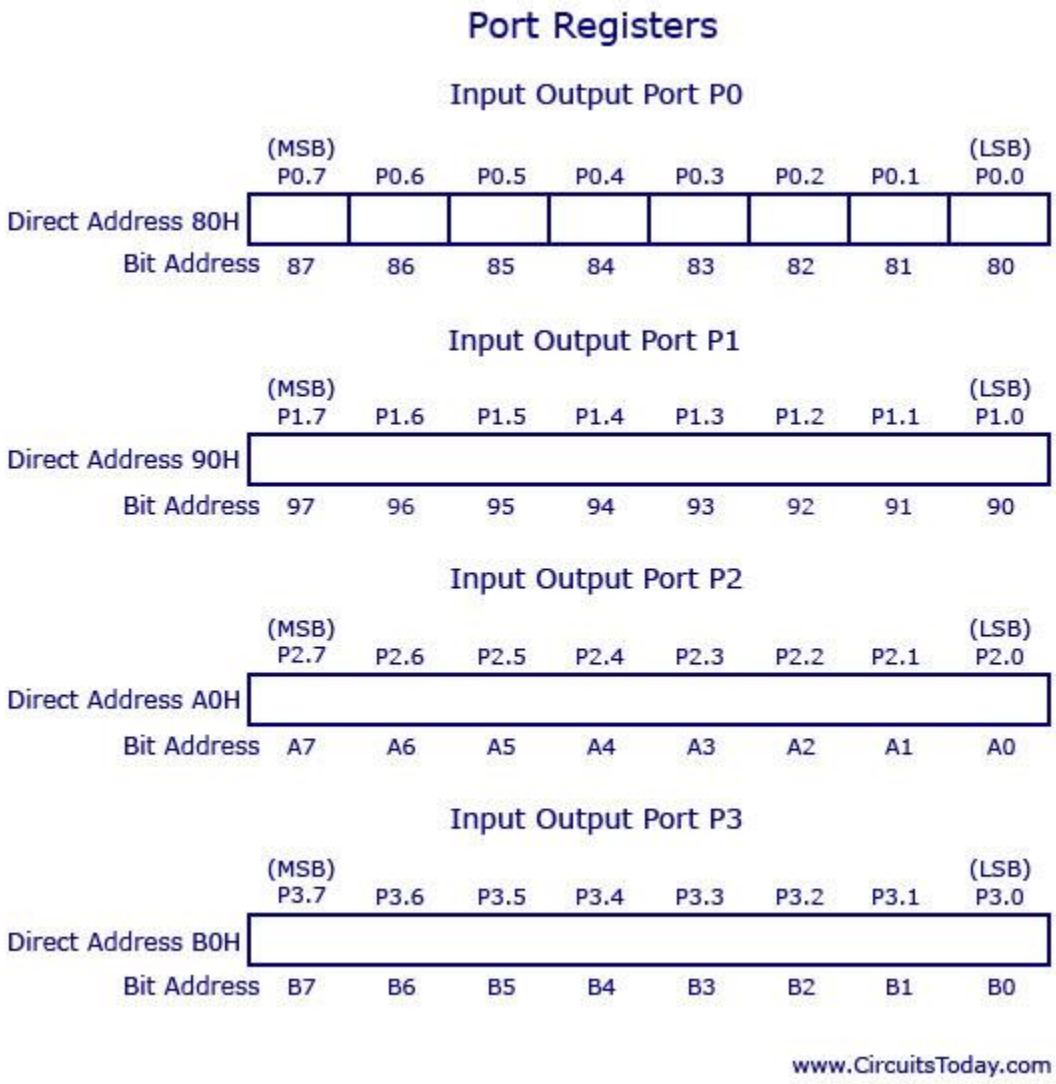


Note: Register B can also be used for other general purpose operations.

Port Registers

As you may already know, there are 4 ports for 8051. If you are unfamiliar of the architecture of 8051 please read the following article:- **The architecture of 8051**

So 4 Input/Output ports named P0, P1, P2 and P3 has got four corresponding port registers with same name P0, P1, P2 and P3. Data must be written into port registers first to send it out to any other external device through ports. Similarly any data received through ports must be read from port registers for performing any operation. All 4 port registers are bit as well as byte addressable. Take a look at the figure below for a better understanding of port registers.



From the figure:-

- The physical address of port 0 is 80
- The physical address of port 1 is 90
- And that of port 2 is A0
- And that of port 3 is B0

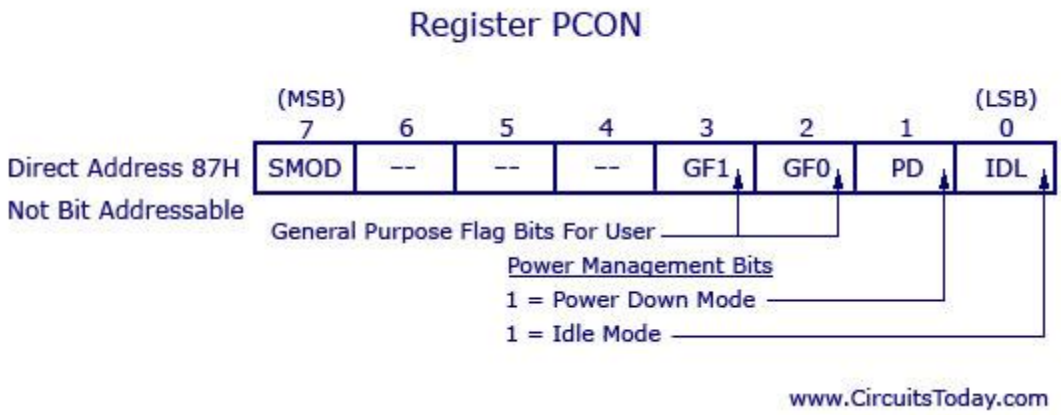
Stack Pointer

Known popularly with an acronym SP, stack pointer represents a pointer to the the system stack. Stack pointer is an 8 bit register, the direct address of SP is 81H and it is only byte addressable, which means you cant access individual bits of stack pointer. The content of the stack pointer points to the last stored location of system stack. To store something new in system stack, the SP must be incremented by 1 first and then execute the "store" command. Usually after a system reset SP is initialized as 07H and data can be stored to stack from 08H onwards. This is usually a default case and programmer can alter values of SP to suit his needs.

Power Management Register (PCON)

Power management using a microcontroller is something you see every day in mobile phones. Haven't you noticed and got wondered by a mobile phone automatically going into stand by mode when not used for a couple of seconds or minutes ? This is achieved by power management feature of the controller used inside that phone.

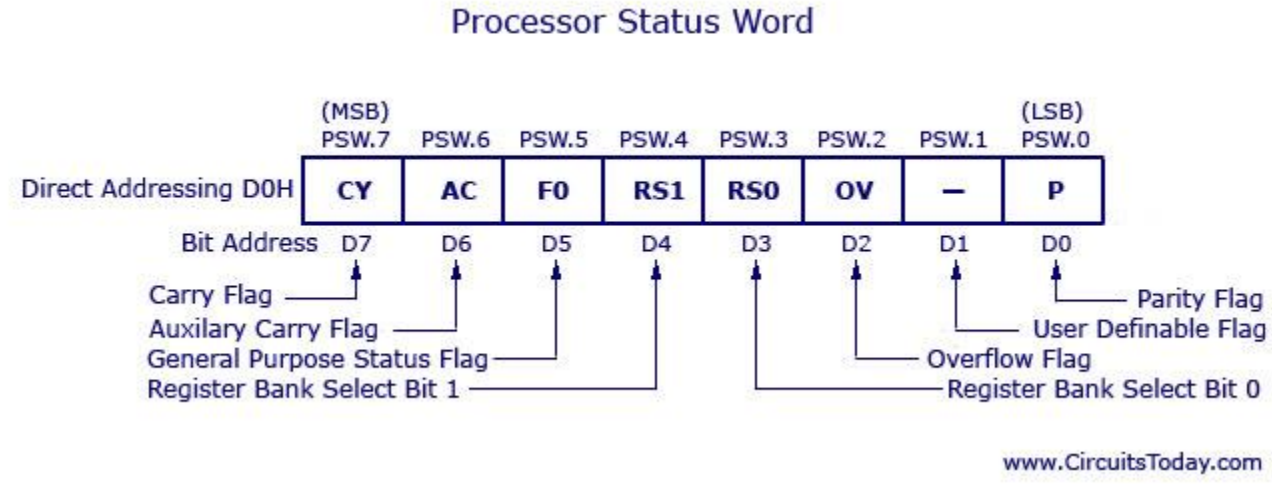
As the name indicates, this register is used for efficient power management of 8051 micro controller. Commonly referred to as PCON register, this is a dedicated SFR for power management alone. From the figure below you can observe that there are 2 modes for this register :- ***Idle mode and Power down mode.***



Setting bit 0 will move the micro controller to Idle mode and Setting bit 1 will move the micro controller to Power down mode.

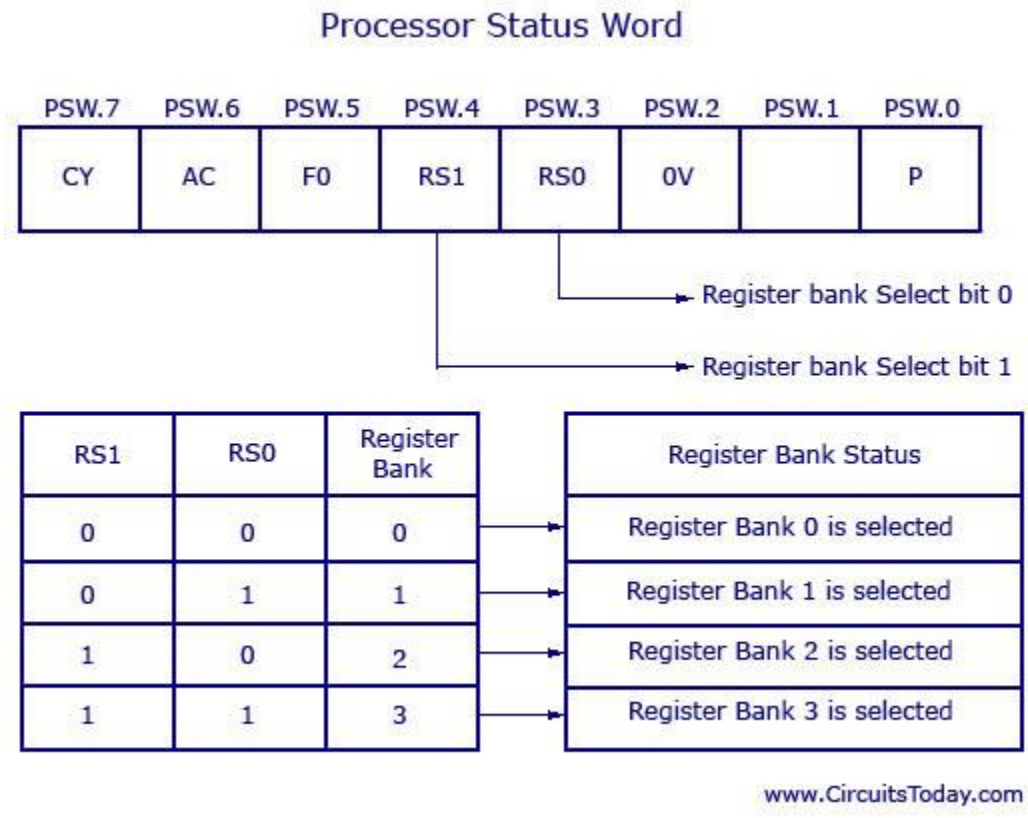
Processor Status Word (PSW)

Commonly known as the PSW register, this is a vital SFR in the functioning of micro controller. This register reflects the status of the operation that is being carried out in the processor. The picture below shows PSW register and the way register banks are selected using PSW register bits – RS1 and RS0. PSW register is both bit and byte addressable. The physical address of PSW starts from D0H. The individual bits are then accessed using D1, D2 ... D7. The various individual bits are explained below.



Bit No	Bit Symbol	Direct Address	Name	Function
0	P	D0	Parity	This bit will be set if ACC has odd number of 1's after an operation. If not, bit will remain cleared.
1	–	D1		User definable bit
2	OV	D2	Overflow	OV flag is set if there is a carry from bit 6 but not from bit 7 of an Arithmetic operation. It's also set if there is a carry from bit 7 (but not from bit 6) of Acc
3	RS0	D3	Register Bank select bit 0	LSB of the register bank select bit. Look for explanation below this table.
4	RS1	D4	Register Bank select bit 1	MSB of the register bank select bits.
5	F0	D5	Flag 0	User defined flag
6	AC	D6	Auxiliary carry	This bit is set if data is coming out from bit 3 to bit 4 of Acc during an Arithmetic operation.
7	CY	D7	Carry	Is set if data is coming out of bit 7 of Acc during an Arithmetic operation.

At a time registers can take value from R0,R1...to R7. You may already know there are 32 such registers. ***So how you access 32 registers with just 8 variables to address registers?*** Here comes the use of register banks. There are 4 register banks named 0,1,2 and 3. Each bank has 8 registers named from R0 to R7. At a time only one register bank can be selected. Selection of register bank is made possible through PSW register bits PSW.3 and PSW.4, named as RS0 and RS1.These two bits are known as register bank select bits as they are used to select register banks. The picture will talk more about selecting register banks.



So far we have discussed about all major SFR's in 8051. There many other still waiting! Please remember there are 21 SFR's and we have discussed only 9 specifically. The table below lists all other 12 SFR's.

SFR	Address	Function
DPH	83	Data pointer registers (High). Only byte addressing possible.
DPL	82	Data pointer register (Low). Only byte addressing possible.
IP	B8	Interrupt priority. Both bit addressing and byte addressing possible.
IE	A8	Interrupt enable. Both bit addressing and byte addressing possible.
SBUF	99	Serial Input/Output buffer. Only byte addressing is possible.
SCON	98	Serial communication control. Both bit addressing and byte addressing possible.
TCON	88	Timer control. Both bit addressing and byte addressing possible.
TH0	8C	Timer 0 counter (High). Only byte addressing is possible.
TL0	8A	Timer 0 counter (Low). Only byte addressing is possible.

TH1	8D	Timer 1 counter (High). Only byte addressing is possible.
TL1	8B	Timer 1 counter (Low). Only byte addressing is possible.
TMOD	89	Timer mode select. Only byte addressing is possible.

Statement 1: – exchange the content of FFh and FF00h

Solution: – here one is internal memory location and other is memory external location. so first the content of ext memory location FF00h is loaded in acc. then the content of int memory location FFh is saved first and then content of acc is transferred to FFh. now saved content of FFh is loaded in acc and then it is transferred to FF00h.

```
Mov dptr, #0FF00h     ; take the address in dptr
   Movx a, @dptr          ; get the content of 0050h in a
   Mov r0, 0FFh           ; save the content of 50h in r0
   Mov 0FFh, a            ; move a to 50h
   Mov a, r0              ; get content of 50h in a
   Movx @dptr, a          ; move it to 0050h
```

Duplication and Subtraction

Statement 2: – store the higher nibble of r7 in to both nibbles of r6

Solution: –first we shall get the upper nibble of r7 in r6. Then we swap nibbles of r7 and make OR operation with r6 so the upper and lower nibbles are duplicated

```
Mov a, r7              ; get the content in acc
   Anl a, #0F0h           ; mask lower bit
   Mov r6, a              ; send it to r6
   Swap a          ; xchange upper and lower nibbles of acc
   Orl a, r6              ; OR operation
   Mov r6, a              ; finally load content in r6
```

Statement 3: – treat r6-r7 and r4-r5 as two 16 bit registers. Perform subtraction between them. Store the result in 20h (lower byte) and 21h (higher byte).

Solution: – first we shall clear the carry. Then subtract the lower bytes afterward then subtract higher bytes.

```
Clr c                  ; clear carry
   Mov a, r4              ; get first lower byte
   Subb a, r6             ; subtract it with other
   Mov 20h, a             ; store the result
   Mov a, r5              ; get the first higher byte
   Subb a, r7             ; subtract from other
```

```
        Mov 21h, a          ; store the higher byte
```

Division & Data Transfer

Statement 4: – divide the content of r0 by r1. Store the result in r2 (answer) and r3 (reminder). Then restore the original content of r0.

Solution:-after getting answer to restore original content we have to multiply answer with divider and then add reminder in that.

```
        Mov a, r0           ; get the content of r0 and r1
        Mov b, r1           ; in register A and B
        Div ab              ; divide A by B
        Mov r2, a           ; store result in r2
        Mov r3, b           ; and reminder in r3
        Mov b, r1           ; again get content of r1 in B
        Mul ab              ; multiply it by answer
        Add a, r3           ; add reminder in new answer
        Mov r0, a           ; finally restore the content of r0
```

Statement 5: – transfer the block of data from 20h to 30h to external location 1020h to 1030h.

Solution: – here we have to transfer 10 data bytes from internal to external RAM. So first, we need one counter. Then we need two pointers one for source second for destination.

```
        Mov r7, #0Ah        ; initialize counter by 10d
        Mov r0, #20h        ; get initial source location
        Mov dptr, #1020h    ; get initial destination location
Nxt:    Mov a, @r0          ; get first content in acc
        Movx @dptr, a       ; move it to external location
        Inc r0              ; increment source location
        Inc dptr            ; increase destination location
        Djnz r7, nxt    ; decrease r7. if zero then over otherwise move next
```

Various Comparison Programs

Statement 6: – find out how many equal bytes between two memory blocks 10h to 20h and 20h to 30h.

Solution: – here we shall compare each byte one by one from both blocks. Increase the count every time when equal bytes are found

```
        Mov r7, #0Ah        ; initialize counter by 10d
        Mov r0, #10h        ; get initial location of block1
        Mov r1, #20h        ; get initial location of block2
        Mov r6, #00h        ; equal byte counter. Starts from zero
Nxt:    Mov a, @r0          ; get content of block 1 in acc
```

```
                Mov b, a                ; move it to B
                Mov a, @r1              ; get content of block 2 in acc
                Cjne a, b, nomatch      ; compare both if equal
                Inc r6                  ; increment the counter
Nomatch:        inc r0                  ; otherwise go for second number
                Inc r1
                djnz r7, nxt    ; decrease r7. if zero then over otherwise move next
```

Statement 7: – given block of 100h to 200h. Find out how many bytes from this block are greater then the number in r2 and less then number in r3. Store the count in r4.

Solution: – in this program, we shall take each byte one by one from given block. Now here two limits are given higher limit in r3 and lower limit in r2. So we check first higher limit and then lower limit if the byte is in between these limits then count will be incremented.

```
            Mov dptr, #0100h    ; get initial location
                Mov r7, #0FFh       ; counter
                Mov r4, #00h        ; number counter
                Mov 20h, r2         ; get the upper and lower limits in
                Mov 21h, r3         ; 20h and 21h
        Nxt:    Movx a, @dptr       ; get the content in acc
                Cjne a, 21h, lower  ; check the upper limit first
                Sjmp out            ; if number is larger
        Lower: jnc out              ; jump out
                Cjne a, 20h, limit  ; check lower limit
                Sjmp out            ; if number is lower
        Limit:  jc out              ; jump out
                Inc r4              ; if number within limit increment count
        Out:    inc dptr        ; get next location
                Djnz r7, nxt        ; repeat until block completes
```

Interrupt Counting, Subroutines & Scan n Mulitply

Statement 8:- the crystal frequency is given as 12 MHz. Make a subroutine that will generate delay of exact 1 ms. Use this delay to generate square wave of 50 Hz on pin P2.0

Solution: – 50 Hz means 20 ms. And because of square wave 10 ms ontime and 10 ms offtime. So for 10 ms we shall send 1 to port pin and for another 10 ms send 0 in continuous loop.

```
    <_x0021_xml:namespace prefix="st1″ ns="urn:schemas-microsoft-
com:office:smarttags"/>Loop:  Setb p2.0          ; send 1 to port pin
                Mov r6, #0Ah        ; load 10d in r6
                Acall delay         ; call 1 ms delay ×10 = 10 ms
                Clr p2.0            ; send 0 to port pin
                Mov r6, #0Ah        ; load 10d in r6
```

```
        Acall delay          ; call 1 ms delay ×10 = 10 ms
        Sjmp loop            ; continuous loop
    Delay:                   ; load count 250d
Lp2:  Mov r7, #0FAh
Lp1:  Nop                    ; 1 cycle
      Nop                    ; 1+1=2 cycles
      Djnz r7, lp1           ; 1+1+2 = 4 cycles
      Djnz r6, lp2           ; 4×250 = 1000 cycles = 1000 µs = 1 ms
        ret
```

Statement 9:-count number of interrupts arriving on external interrupt pin INT1. Stop whencounter overflows and disable the interrupt. Give the indication on pinP0.0

Solution: –as we know whenever interrupt occurs the PC jumps to one particular location where it's ISR is written. So we have to just write one ISR that will do the job

```
      Movr2, #00h          ; initialize the counter
      Movie, #84h          ; enable external interrupt 1
Here: Sjmp here            ; continuous loop

Org 0013h                  ; interrupt 1location
      Incr2                ; increment the count
      Cjner2, #00h, out    ; check whether it overflows
      Movie, #00h          ; if yes then disable interrupt
      Clr   p0.0           ; and give indication
Out   : reti               ; otherwise keep counting
```

Statement 10: –continuously scan port P0. If data is other then FFh write a subroutine that will multiply it with 10d and send it to port P1

Solution: –here we have to use polling method. We shall continuously pole port P0 if there is any data other then FFh. If there is data we shall call subroutine

```
Again:  Mov p0, #0ffh  ; initialize port P0 as input port
Loop:   Mov a, p0      ; get the data in acc
        Cjne a, #0FFh, dat  ; compare it with FFh
        Sjmp loop      ; if same keep looping
Dat:    acall multi;     if different call subroutine
        Sjmp again       ; again start polling
Multi
        Mov b,#10d     ; load 10d in register B
        Mul ab         ; multiply it with received data
        Mov p1, a      ; send the result to P1
        Ret            ;return to main program
```

KEYBOARD INTERFACING

Keyboards and LCDs are the most widely used input/output devices of the 8051, and a basic understanding of them is essential. In this section, we first discuss keyboard fundamentals, along with key press and key detection mechanisms. Then we show how a keyboard is interfaced to an 8051.

Interfacing the keyboard to the 8051

At the lowest level, keyboards are organized in a matrix of rows and columns. The CPU accesses both rows and columns through ports; therefore, with two 8-bit ports, an 8 x 8 matrix of keys can be connected to a microprocessor. When a key is pressed, a row and a column make a contact; otherwise, there is no connection between rows and columns. In IBM PC keyboards, a single microcontroller (consisting of a microprocessor, RAM and EPROM, and several ports all on a single chip) takes care of hardware and software interfacing of the keyboard. In such systems, it is the function of programs stored in the EPROM of the microcontroller to scan the keys continuously, identify which one has been activated, and present it to the motherboard. In this section we look at the mechanism by which the 8051 scans and identifies the key.

Scanning and identifying the key

Figure 12-6 shows a 4 x 4 matrix connected to two ports. The rows are connected to an output port and the columns are connected to an input port. If no key has been pressed, reading the input port will yield 1s for all columns since they are all connected to high (V_{CC}). If all the rows are grounded and a key is pressed, one of the columns will have 0 since the key pressed provides the path to ground. It is the function of the microcontroller to scan the keyboard continuously to detect and identify the key pressed. How it is done is explained next.

Grounding rows and reading the columns

To detect a pressed key, the microcontroller grounds all rows by providing 0 to the output latch, then it reads the columns. If the data read from the columns is D3 - D0 = 1111, no key has been pressed and the process continues until

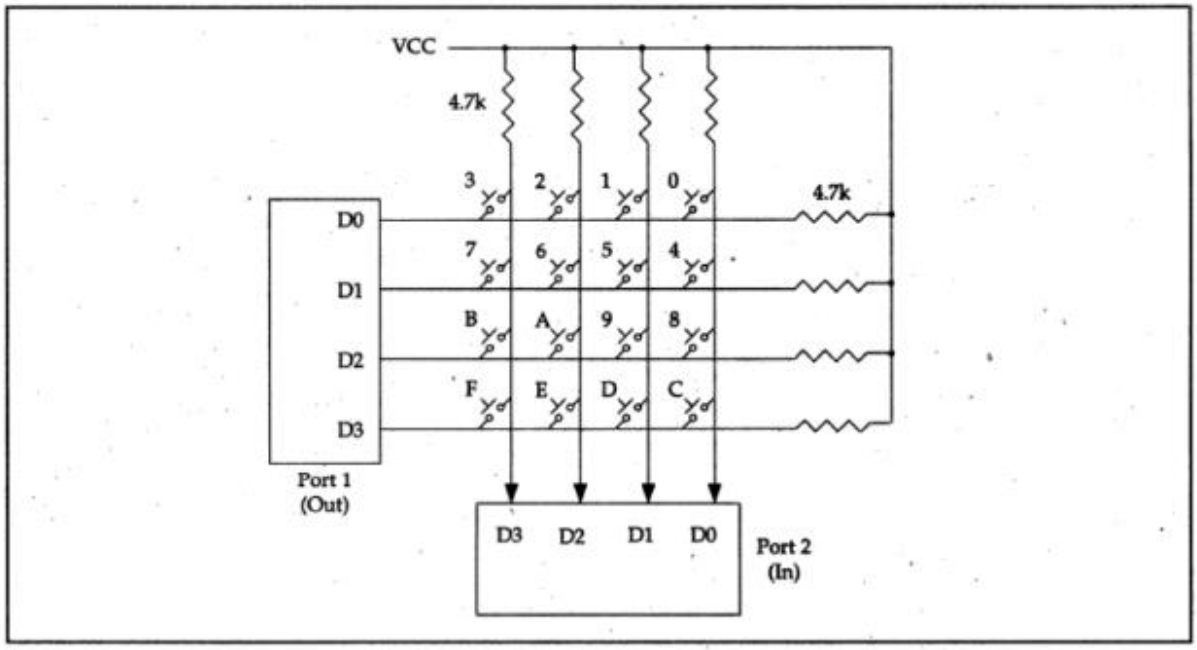


Figure 12-6. Matrix Keyboard Connection to Ports

Example 12-3

From Figure 12-6, identify the row and column of the pressed key for each of the following.
(a) D3 - D0 = 1110 for the row, D3 - D0 = 1011 for the column
(b) D3 - D0 = 1101 for the row, D3 - D0 = 0111 for the column

Solution:

From Figure 12-6 the row and column can be used to identify the key.
(a) The row belongs to D0 and the column belongs to D2; therefore, key number 2 was pressed.
(b) The row belongs to D1 and the column belongs to D3; therefore, key number 7 was pressed.

a key press is detected. However, if one of the column bits has a zero, this means that a key press has occurred. For example, if D3 - D0 = 1101, this means that a key in the D1 column has been pressed. After a key press is detected, the microcontroller will go through the process of identifying the key. Starting with the top row, the microcontroller grounds it by providing a low to row D0 only; then it reads the columns. If the data read is all 1s, no key in that row is activated and the process is moved to the next row. It grounds the next row, reads the columns, and checks for any zero. This process continues until the row is identified. After identification of the row in which the key has been pressed, the next task is to find out which column the pressed key belongs to. This should be easy since the microcontroller knows at any time which row and column are being accessed. Look at Example 12-3.

Program 12-4 is the 8051 Assembly language program for detection and identification of key activation. In this program, it is assumed that P1 and P2 are initialized as output and input, respectively. Program 12-4 goes through the following four major stages:

1. To make sure that the preceding key has been released, 0s are output to all rows at once, and the columns are read and checked repeatedly until all the columns are high. When all columns are found to be high, the program waits for a short amount of time before it goes to the next stage of waiting for a key to be pressed.
2. To see if any key is pressed, the columns are scanned over and over in an infinite loop until one of them has a 0 on it. Remember that the output latches connected to rows still have their initial zeros (provided in stage 1), making them grounded. After the key press detection, the microcontroller waits 20 ms for the bounce and then scans the columns again. This serves two functions: (a) it ensures that the first key press detection was not an erroneous one due to a spike noise, and (b) the 20-ms delay prevents the same key press from being interpreted as a multiple key press. If after the 20-ms delay the key is still pressed, it goes to the next stage to detect which row it belongs to; otherwise, it goes back into the loop to detect a real key press.
3. To detect which row the key press belongs to, the microcontroller grounds one row at a time, reading the columns each time. If it finds that all columns are high, this means that the key press cannot belong to that row; therefore, it grounds the next row and continues until it finds the row the key press belongs to. Upon finding the row that the key press belongs to, it sets up the starting address for the look-up table holding the scan codes (or the ASCII value) for that row and goes to the next stage to identify the key.
4. To identify the key press, the microcontroller rotates the column bits, one bit at a time, into the carry flag and checks to see if it is low. Upon finding the zero, it pulls out the ASCII code for that key from the look-up table; otherwise, it increments the pointer to point to the next element of the look-up table. Figure 12-7 flowcharts this process.

While the key press detection is standard for all keyboards, the process for determining which key is pressed varies. The look-up table method shown in Program 12-4 can be modified to work with any matrix up to 8 x 8. Figure 12-7 provides the flowchart for Program 12-4 for scanning and identifying the pressed key.

There are IC chips such as National Semiconductor's MM74C923 that incorporate keyboard scanning and decoding all in one chip. Such chips use combinations of counters and logic gates (no microcontroller) to implement the underlying concepts presented in Program 12-4. Example 12-4 shows keypad programming in 8051 C.

```
;Keyboard subroutine. This program sends the ASCII code
;for pressed key to P0.1
;P1.0-P1.3 connected to rows P2.0-P2.3 connected to columns
        MOV   P2,#0FFH              ;make P2 an input port
K1:     MOV   P1,#0                 ;ground all rows at once
        MOV   A,P2                  ;read all col. ensure all keys open
        ANL   A,#00001111B          ;masked unused bits
        CJNE  A,#00001111B,K1       ;check til all keys released
K2:     ACALL DELAY                 ;call 20 ms delay
        MOV   A,P2                  ;see if any key is pressed
        ANL   A,#00001111B          ;mask unused bits
        CJNE  A,#00001111B,OVER     ;key pressed, await closure
        SJMP  K2                    ;check if key pressed
OVER:   ACALL DELAY                 ;wait 20 ms debounce time
        MOV   A,P2                  ;check key closure
        ANL   A,#00001111B          ;mask unused bits
        CJNE  A,#00001111B,OVER1    ;key pressed, find row
        SJMP  K2                    ;if none, keep polling
OVER1:  MOV   P1,#11111110B         ;ground row 0
        MOV   A,P2                  ;read all columns
        ANL   A,#00001111B          ;mask unused bits
        CJNE  A,#00001111B,ROW_0    ;key row 0, find the col.
        MOV   P1,#11111101B         ;ground row 1
        MOV   A,P2                  ;read all columns
        ANL   A,#00001111B          ;mask unused bits
        CJNE  A,#00001111B,ROW_1    ;key row 1, find the col.
        MOV   P1,#11111011B         ;ground row 2
        MOV   A,P2                  ;read all columns
        ANL   A,#00001111B          ;mask unused bits
        CJNE  A,#00001111B,ROW_2    ;key row 2, find the col.
        MOV   P1,#11110111B         ;ground row 3
        MOV   A,P2                  ;read all columns
        ANL   A,#00001111B          ;mask unused bits
        CJNE  A,#00001111B,ROW_3    ;key row 3, find the col.
        LJMP  K2                    ;if none, false input, repeat
```

```
ROW 0:  MOV   DPTR,#KCODE0          ;set DPTR=start of row 0
        SJMP  FIND                  ;find col. key belongs to
ROW 1:  MOV   DPTR,#KCODE1          ;set DPTR=start of row 1
        SJMP  FIND                  ;find col. key belongs to
ROW 2:  MOV   DPTR,#KCODE2          ;set DPTR=start of row 2
        SJMP  FIND                  ;find col. key belongs to
ROW 3:  MOV   DPTR,#KCODE3          ;set DPTR=start of row 3
FIND:   RRC   A                     ;see if any CY bit is low
        JNC   MATCH                 ;if zero, get the ASCII code
        INC   DPTR                  ;point to next col. address
        SJMP  FIND                  ;keep searching
MATCH:  CLR   A                     ;set A=0 (match is found)
        MOVC  A,@A+DPTR             ;get ASCII code from table
        MOV   P0,A                  ;display pressed key
        LJMP  K1
;ASCII  LOOK-UP TABLE FOR EACH ROW
        ORG   300H
KCODE0: DB    '0','1','2','3'       ;ROW 0
KCODE1: DB    '4','5','6','7'       ;ROW 1
KCODE2: DB    '8','9','A','B'       ;ROW 2
KCODE3: DB    'C','D','E','F'       ;ROW 3
        END
```

Program 12-4: Keyboard Program

Selecting an analog channel

The following are the steps we need to take for data conversion by the ADC0848 chip.

1. While CS = 0 and RD = 1 provide the address of the selected channel (see Table 13-6) to the DB0 - DB7 pins, and apply a low-to-high pulse to the WR pin to latch in the address and start the conversion. The channel's addresses are 08H for CH1, 09H for CH2, 0AH for CH3, and so on, as shown in Table 13-6. Notice that this process not only selects the channel, but also starts the conversion of the analog input at the selected channel.

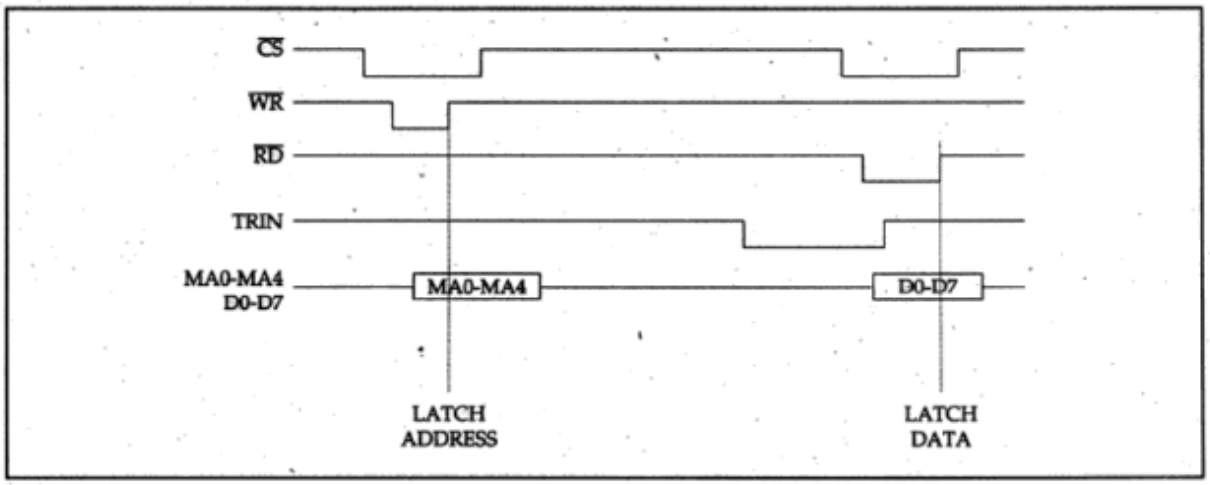


Figure 13-10. Selecting a Channel and Read Timing for the ADC0848

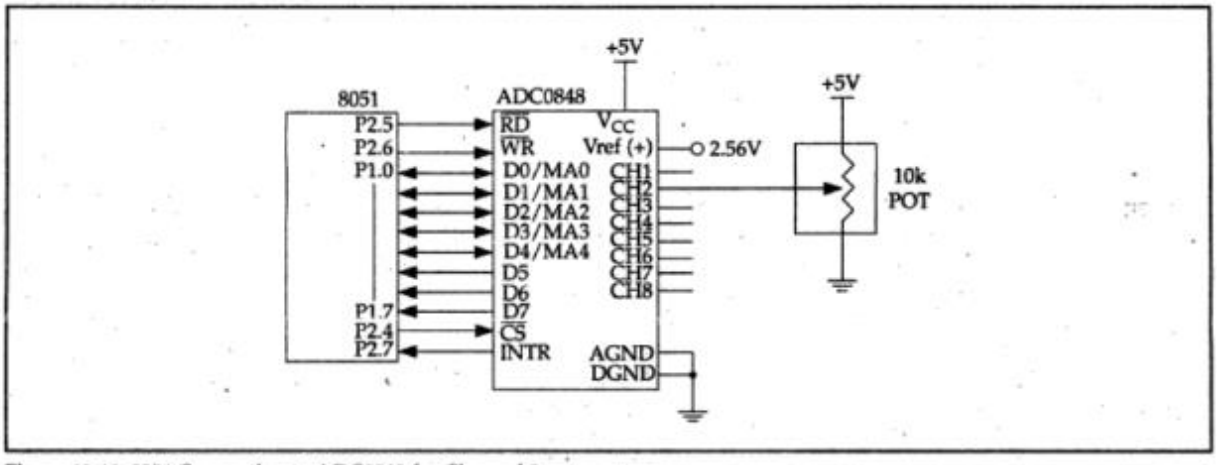


Figure 13-11. 8051 Connection to ADC0848 for Channel 2

2. While WR = 1 and RD = 1 keep monitoring the INTR pin. When INTR goes low, the conversion is finished and we can go to the next step. If INTR is high, we keep the ADC0848 polling until it goes low, signalling end-of-conversion.
3. After the INTR has become low, we must make CS = 0, WR = 1, and apply a low pulse to the RD pin to get the data out of the 848 IC chip.

ADC0848 connection to 8051

The following is a summary of the connection between the 8051 and the ADC0848 as shown in Figure 13-11.

P1.0-P1.7 D0 - D7 of ADC:	Channel selection (out), data read (in)
P2.7 to INTR	P2.7 as input
P2.6 to WR	P2.6 as output
P2.5 to RD	P2.5 as output

Notice the following facts about Figure 13-11.

1. P2 is an output when we select a channel, and it is an input when we read the converted data.
2. We can monitor the INTR pin of the ADC for end-of-conversion or we can wait a few milliseconds and then read the converted data.

Displaying ADC0848 data

In order to display the ADC result on a screen or LCD, it must be converted to ASCII. To convert it to ASCII, however, it must first be converted to decimal. To convert a 00 - FF hex value to decimal, we keep dividing it by 10 until the remainder is less than 10. Each time we divide it by 10 we keep the quotient as one of our decimal digits. In the case of an 8-bit data, dividing it by 10 twice will do the job. For example, if we have FFH it will become 255 in decimal. To convert from decimal to ASCII format, we OR each digit with 30H (see Chapter 6). Now all we have to do is to send the digits to the PC screen using a serial port, or send them to the LCD, as was shown in the Chapter 12.

Programming ADC0848 in Assembly

The following program selects channel 2, reads the data, and calls conversion and display subroutines.

```
CS      BIT P2.4
RD      BIT P2.5
```

```
      WR      BIT P2.6
      INTR    BIT P2.7
      ORG     0H
      SETB    INTR            ;make INTR an input
      SETB    CS              ;set Chip Select high
      SETB    RD              ;set Read high
      SETB    WR              ;set Write high
BACK:
      MOV     P1,#0AH         ;Chan 2 address(Table 13-6)
      NOP                     ;wait
      CLR     CS              ;chip select (CS=0)
      CLR     WR              ;write = LOW
      NOP                     ;make pulse width wide enough
      NOP                     ;for DS89C4x0 you might need a delay
      SETB    WR              ;latch the address and start conv
      SETB    CS              ;de-select the chip
      MOV     P1,#0FFH        ;make P1 an input
HERE:
      JB      INTR,HERE       ;wait for EOC
      CLR     CS              ;chip select (CS=0)
      CLR     RD              ;read RD=0
      NOP                     ;make pulse width wide enough
      NOP
      SETB    RD              ;bring out digital data
      MOV     A,P1            ;get the value
      SETB    CS              ;de-select for next round
      ACALL   CONVERT         ;convert to ASCII (Chap 6)
      ACALL   DATA_DISPLAY    ;display the data (Chap 12)
```

INTERFACING STEPPER MOTOR WITH 8051

ASSEMBLY PROGRAM TO RUN THE STEPPER MOTOR USING 8051

ADDRESS	OP CODE	MNEMONICS	COMMENTS
8500	74 80	MOV A, #80H	Initialize 80HEX value to Accumulator
8502	90 40 03	MOV DPTR, #4003	Move 4003 to R1 register
8505	F0	MOVX @DPTR, A	Store ACC value into R1 register
8506	90 40 00	MOV DPTR, #4000	Move 4000 to R1 register

8509	74 66	MOV A, #66H	Move 66HEX value to Accumulator
850B	F0	AGAIN:MOVX@DPTR, A	In AGAIN, Store ACC value into R1
850C	03	RR A	Rotate sequence for clockwise
850D	B1 11	ACALL DELAY	wait
850F	80 FA	SJMP AGAIN	Short Jump into AGAIN
8511	7D 05	DELAY: MOV R5, #5	In DELAY, Load 5 value into R5
8513	7C 0F	H3: MOV R4, #0F	In H3, Load 0F value into R4 register
8515	7B 43	H2: MOV R3, #43	In H2, Load 43 value into R3 register
8517	DB FE	H1 : DJNZ R3, H1	H1, Decrement R0& check if it's not 0
8519	DC FA	DJNZ R4, H2	Decrement R4& Check if it's not 0
851B	DD F6	DJNZ R5, H3	Decrement R5& Check if it's not 0
851D	22	RET	Return

Questions	opt1	opt2	opt3	opt4
An 8- bit Microprocessor signifies that it has	8- bit address bus	8- bit control bus	8- bit data bus	None of these
Microprocessors are	Programmable	General purposee	Single chip CPU	All the above
Microprocessor contains	Control and arithmetic unit	RAM, ROM	peripherals	all the above
The register which keeps track of the execution of program is	program counter	Stack pointer	Instruction Register	Accumulator
Microprocessors are usually programmed using	High level language	Assembly language	Machine language	None of these
The slower peripherals and processor are synchronized using____ signal	RESET	READY	HOLD	HLDA
The status signal IO/ M bar = 1, S1=0, S0=1 corresponds to	opcode fetch	memory read	I/O write	I/O read
MVI C,50 is a _____ instruction	one byte	two byte	thee byte	zero byte
MOV C,M belongs to _____ addressing	Direct	Implied	Register	Register indirect
Which of the following flag is not available in 8085	zero	parity	overflow	auxiliary carry
ALE stands for	Address Latch Enable	Arithmetic Logic Enable	Address Logic enable	None of the above
DAA instruction is used for _____ addition	binary	BCD	Hexadecimal	octal
After executing “ADD reg” instruction, result will be in	ALU	Temporary register	Program counter	Accumulator
The “INR reg” can be used to increment the contents of	Accumulator	HL pair	B,C registers	All the registers
NOP is used for	delay	troubleshooting	halting	none of these
RIM stands for	Reset Interrupt Mask	delay	Restart Interrupt Mask	Register Interrupt Mask
The instruction which is used to complement the accumulator contents	CMA	CMP	CMC	CMP A
The microprocessor consists of __ bit address bus.	8	16	24	32
The instruction which affects only flag is	ADC M	INX rp	CMP M	CMA
“POP reg” operation is similar to	CALL	PUSH	RET	JUMP
Which of the following is a 16-bit register	Accumulator	program counter	decoder	none of the above
Which of the following is not a 16-bit register	program counter	Accumulator	stack pointer	none of the above
_____ is special purpose register	BC	DE	HL	PC
_____ flag is set for negative numbers	Sign	Aux. carry	Carry	parity
When there is a overflow after D3, _____ flag is set	Sign	Aux. carry	Carry	parity
When there is a overflow after D7, _____ flag is set	Sign	Aux. carry	Carry	parity
When there are even no. of ones, _____ flag is set	Sign	Aux. carry	Carry	parity
The fetched opcode is first placed in _____ register	IR	PC	SP	HL
_____ signal is used for DMA operation	READY	RESET	HOLD	SOD
_____ signal is used to clear the flipflops	READY	RESET	HOLD	SOD
_____ signal is used for communicating peripherals	READY	RESET	HOLD	SOD
_____ signal is used for serial communication	READY	RESET	HOLD	SOD
The status signal for opcode fetch	IO/M-=0, S1=1,S0=1	IO/M-=0, S1=1,S0=0	IO/M-=0, S1=0,S0=1	IO/M-=1, S1=1,S0=0
The status signal for memory read	IO/M-=0, S1=1,S0=1	IO/M-=0, S1=1,S0=0	IO/M-=0, S1=0,S0=1	IO/M-=1, S1=1,S0=0
The status signal for memory write	IO/M-=0, S1=1,S0=1	IO/M-=0, S1=1,S0=0	IO/M-=0, S1=0,S0=1	IO/M-=1, S1=1,S0=0
The status signal for I/O read	IO/M-=0, S1=1,S0=1	IO/M-=0, S1=1,S0=0	IO/M-=0, S1=0,S0=1	IO/M-=1, S1=1,S0=0
The status signal for I/O write	IO/M-=0, S1=1,S0=1	IO/M-=0, S1=1,S0=0	IO/M-=1, S1=0,S0=1	IO/M-=1, S1=1,S0=0
The vectored address for RST 4 interrupt	0020H	0028H	0030H	0038H
The vectored address for RST 5 interrupt	0020H	0028H	0030H	0038H
The vectored address for RST 6 interrupt	0020H	0028H	0030H	0038H
The vectored address for RST 7 interrupt	0020H	0028H	0030H	0038H
MVI C,50 belongs to _____ addressing mode	Immediate	Direct	Reg. indirect	Implied
LDA 4800 belongs to _____ addressing mode	Immediate	Direct	Reg. indirect	Implied
MOV A,M belongs to _____ addressing mode	Immediate	Direct	Reg. indirect	Implied
CMA belongs to _____ addressing mode	Immediate	Direct	Reg. indirect	Implied
LXIH,5000 belongs to _____ addressing mode	Immediate	Direct	Reg. indirect	Implied
SHLD 5000 belongs to _____ addressing mode	Immediate	Direct	Reg. indirect	Implied
STAX rp belongs to _____ addressing mode	Immediate	Direct	Reg. indirect	Implied
RAL belongs to _____ addressing mode	Immediate	Direct	Reg. indirect	Implied
MOV B,C is a _____ byte instruction	1	2	3	4
MVI C, 60 is a _____ byte instruction	1	2	3	4
SHLD 4700 is a __ byte instruction	1	2	3	4
LDAX B is a _____ byte instruction	1	2	3	4
SUI , 20 is a _____ byte instruction	1	2	3	4
JMP 2000 is a _____ byte instruction	1	2	3	4
_____ instruction is used to complement the accumulator	CMA	CMC	CMP	COM
_____ instruction is used to complement the carry	CMA	CMC	CMP	COM
_____ instruction is used to compare the register contents	CMA	CMC	CMP	COM
The instruction which is used to increment the content of register	INR	INX	INC	INT
The instruction which is used to increment the content of register pair	INR	INX	INC	INT

UNIT II

Questions	opt1	opt2	opt3	opt4
The instruction which is used to decrement the content of register pair	DCR	DCX	DEC	DEN
8086isa_____bit microprocessor	8	16	20	32

8086 is capable of addressing_____memory locations	1Mb	64Kb	32Kb	All the above
In8086,external memoryi s divided into_____segments	2	4	6	8
The register which keeps track of the execution of program in 8086 is	program counter	Stack pointer	Instruction Pointer	Accumulator
The status signal S2=1,S1=0,S0=1 corresponds to	In active	Memory read	I/O write	I/O read
Each segment of memory is of_____long	1Mb	64Kb	32Kb	Alltheabove
The slower peripherals and processor are synchronized using____signal	LOCK	READY	HOLD	HLDA
The size of the instruction queue is	6byte	8 byte	three byte	zero byte
Physical address of instruction is of_____bits	16	20	8	32
In string manipulation,ifDF=1,the data segment is	incremented by 1	decremented by 2	decremented by 1	None of the above
The registers in 8086 are of_____bit wide	4	8	32	16
In8086,the lower_____addresslines are multiplexed with data	4	8	32	16
Addresslines A16-A19 are multiplexed with status lines	S0-S4	S1-S5	S3-S6	None of the above
.The instruction used to increment the contents of registers in 8086 is	INR	INC	INT	None of the above
The 16bit base address is stored in	segment registers	IP	SI	general purpose registers
.Subroutine programs are also referred to as	Macros	Procedures	String	None of the above
The instruction which is used to complement the carry	CMA	CLC	CMC	CC
The pin which is not available in Maximum mode	RQ/GT	LOCK	QS1	INTA
The pin which is not available in Minimum mode	DT/R	HOLD	ALE	LOCK
The offset value means	Effective address	Memory address	Segmen taddress	Physical address
_____flag is set for carry into MSB	Overflow flag	Trap flag	interrupt flag	Direcion flag
._____flag is used for single stepping	Overflow flag	Trap flag	interrupt flag	Direcion flag
._____flag is used for interrupts	Overflow flag	Trap flag	interrupt flag	Direcion flag
_____flag is set for string instructions	Overflow flag	Trap flag	interrupt flag	Direcion flag
MOVBX,CX belongs to_____addressing mode	Register	Immediate	Direct	Register indirect
MOVBX,4354 belongs to_____addressing mode	Register	Immediate	Direct	Register indirect
.MOVBL,[5431] belongs to_____addressing mode	Register	Immediate	Direct	Register indirect
.MOV[DI],CX belongs to_____addressing mode	Register	Immediate	Direct	Register indirect
The instruction which is used to increment the content of register in 8086	INR	INX	INC	INT
._____instruction is used to set the carry in 8086	STC	CMC	CMP	COM
._____instruction is used to complement the carry in 8086	STC	CMC	CMP	COM
._____instruction is used to compare the register contents in 8086	CMA	CMC	CMP	COM
The instruction which is used to decrement thec ontent of register in 8086	DCR	DCX	DEC	DEN
Source program is converted into object code using	assembler	loader	coder	linker
Several object files are combined using	assembler	loader	coder	linker
Object program is placed in system memory using	assembler	loader	coder	linker
Mnemonics codes are replaced by binary codes using	assembler	loader	coder	linker
Large function in gmodule is created using	assembler	loader	coder	linker
._____allows to examine register and memory contents	assembler	loader	coder	linker
._____concept is used in 8086 execution	LIFO	FIFO	random	Noneoftheabove
.Clockfrequency of 8086	1MHz	3MHz	5MHz	7MHz
During instruction fetch,_____cycle is performed	memoryread	emorywrite	I/owrite	I/Oread
Bus is idle while executing_____instruction	ADD	AND	CMP	INC
._____is not a pipelined processor	8085	8086	80286	80386

PUSH instruction decrements stackpointer by	1	2	3	4
POP instruction increments stackpointer by	1	2	3	4
CALL instruction decrements stackpointer by	1	2	3	4
RET instruction increments stack pointer by	1	2	3	4
.IN AL,0F8H copy a_____from port to AL	byte	nibble	word	doubleword
.IN AX,0F8H copy a_____from port to AX	byte	nibble	word	doubleword
.IN AL,Dx copy a_____from port to AL	byte	nibble	word	doubleword
IN AX,DX copy a_____from port to AX	byte	nibble	word	doubleword
MOVAX,5654H is a_____byte instruction	1	2	3	4
.MOV AX,[5654H] is a_____byte instruction	1	2	3	4
LDS CX,[5654H] is a_____byte instruction	1	2	3	4
OUT DX,AL is a_____byte instruction	1	2	3	4
MUL BL is a_____byte instruction	1	2	3	4
NOP is a_____byte instruction	1	2	3	4
JNC [5000]H is a_____byte instruction	1	2	3	4

UNIT III

Questions	opt1	opt2	opt3	opt4
MOV AX, 5654H is a_____byte instruction	1	2	3	4
Asynchronous Transmission is _____ oriented	Character	Bit	Block	word
In8251,the data is send through __ _line	TXD	RXD	TXRDY	RXRD
How many error flags present in8251	4	3	2	1
If valid stop bit is not deducted at the end each character then _____ occurs	Over run error	Parity Error	framing error	over flow error
In 8251 the function of transmit buffer section is	To convert parallel date into serial data	To covert serial data into parallel date	just transmit the data	none of the above
In 8251 the function of Receives buffer section is	To convert parallel date into serial data	To covert serial data into parallel date	just transmit the data	none of the above
TXRD signal is used to Indicate that	Buffer register is empty	out put buffer is empty	buffer register is full	none of the above
TXE signal is used to indicate that	Buffer register is empty	out put buffer is empty	buffer register is full	none of the above
Synchronous Transmission is _____ oriented	Character	Bit	Block	word
In Synchronous mode , the actual data is deducted by receiving	Synchronous Characters	Start Bits	End Bits	None of the above
8255 is a general purpose programmable i/o device used for ---------data transfer.	several	parallel	synchronous	None of the above
8255 has --------i/o pins	40	32	24	28
8255 consists of ---------- ports	3	4	5	None of the above.
Each port of 8255 is of --------bits	8	16	24	32
Bidirectional I/O data transfer is meant only for.	Port A	port B	port C	port D
I/O with a handshake mode uses -------- as handshake signal.	Port A	port B	port C	port D
Simple I/O mode is suited _____	only for port C	only for port A	for all ports	None of the above.
The address of the ports are decided by the address lines ----------	A0-A1	A0–A7	A0-A3.	AO-A2
If the address of port A is 40 then the address of port B & port C are .	41 ,42	90,A0	43,A0	None of the above
The function of IN instruction is to move the data.	from port address to ACC.	from Accumulator to port address	from ACC to memory	from memory to ACC.
The function of OUT instruction is move the data from	from port address to ACC.	from Accumulator to port address	from ACC to memory	from memory to ACC.
All the data from the I/O device routed via----------	Accumulator	B register	Memory address	None of the above
To select BSR mode , the D7 bit of control word is made .	0	1	don't care.	z
If the control word is 01, then ------- bit of port C is high	D3	D2	D1	D0
BSR stands for	Bit set reset	Bit select	bit select register	bit select return.
In 8279 key board interface provides ----------modes.	3	2	4	none of the above
CPU interface unit consists of ----------bit data bus	8	16	32	4
8259 return lines are	RL4-RL7	RET 0- RET 7	RL0 –RL7	none of the above
To display data in 8279	out A3 –A0 , out B7-B0	out A7-A4, OUTB7 –B4	out A3 –A0 , outB3-B0	none
In 8251, to select the characters length as 6 make D3D2 bits of mode instruction format as .	0 0	0 1	1 0	1 1
In 8259 , all interrupt levels are stored in	IRR	ISR	IMR	Control logic

In 8259 , interrupts currently being serviced are stored in	IRR	ISR	IMR	Control logic
In 8259 , all masking bits are stored in	IRR	ISR	IMR	Control logic
The data flow is controlled by	IRR	ISR	Priority resolver	Read / Write logic
The highest priority DMA request is	DRQ0	DRQ1	DRQ2	DRQ3
The lowest priority DMA request is	DRQ0	DRQ1	DRQ2	DRQ3
In 8257, number of channels are	2	4	1	8
In 8257, each channel has _____ bit registers	24	42	16	8
In 8257, each channel has _____ 16 bit registers	2	4	1	8
Which is timer	8279	8235	8253	8523
flip flop is a _____ bit register.	1	2	5	6
Microcontroller has a _____ memory.	external.	In-built	RAM	PROM
The 8-bit microcontroller is called _____	8085	8086	8051	80286
8051 consists of _____ RAM.	64 Kbytes	64 bytes.	128 Kbytes	128 bytes.
8051 has _____ bytes of on-chip ROM.	4	8	4K	8K
8051 consists of _____ timers.	3	2	4	1
8051 consists of _____ serial port.	2	1	4	3
“System on the chip” is also the name given to _____	8051	80286	8086	8085
It has _____ bytes of flash ROM.	3k	3	4K	4
A general purpose microprocessor normally needs which of the follwing devices to be attached to it?	RAM	ROM	I/O	all the above.
A microcontroller normally has which of the following devices on chip?	RAM	ROM	I/O	All the above.
The vast majority of registers in 8051 are _____ bits	16	8	32	64
Pseudo-instructions are also called_____	assembler	assembler directive	compiler directive	code control instructions.
which one of the following is assembler directive	ADD A,R2	MOV A, #12	ORG 20000 H	SJMP HERE
the _____ register in 8051 is called PSW.	flag	accumulator	register banks.	RAM
The size of flag register in 8051 is_____ bits	8	16	12	32
the 32 bytes of RAM is divided into) banks.	3	4	5	2
The size of SP register is _____	8 bits.	8- bytes.	16 bits.	none.
with each POP instruction ,the SP is_____ by 1	incremented.	decremented.	shifted.	none.

UNIT IV

Questions	opt1	opt2	opt3	opt4
with each PUSH instruction ,the SP is _____ by 1	incremented	decremented.	Shifted.	none,.
on power-up the 8051 uses RAM location _____ as the first location of the stack.	8	10	16	none
LJMP is a _____-byte instruction.	2	3	4	none
JNC HERE is ____-byte instruction.	1	2	3	4
LCALL instruction is a _____-byte instruction.	2	3	4	none
the number of machine cycles needed to execute an 8051 instruction is ____	1	2	3	none
There are a total of _____ port in 8051.	3	4	6	none.
Each port in 8051 has_____ bits	8	16	12	32
Which 8051 ports need pull-up resistors to function as an I/O port?	p0	p1	p2	p3
The instruction “MOV A,#04H” uses _____ addressing mode.	indirect.	direct.	register	none`
in internal RAM only _____ are bit-addressable.	16bytes.	16 Kbytes.	64 bytes.	64 Kbytes.
_____ ports of 8051 are bit addressable.	2	1	3	all the ports.
Indicate which of the following registers are bit addressable.	A	B	PSW	all the above
_____ ports are bidirectional	p0 & p1	p2 & p4	p3 & p1	all the ports.
Normally mode ____ is called basic I\O mode	0	2	1	none
the method of connecting I/O chip to CPU is called _____	I/O mapped	Memory mapped	both	none
the transfer of data using parallel lines is _____	faster	slower	nominal	none
the start and stop bits are used in _____	synchronous	asynchronous	both	none
MAX232 uses _____ power supply	5V	15V	9V	none.
the two pins in 8051 that are used for transmitting and receiving a data are	T1 & T0	WR & RD	Txd &rxd	none
In synchronous mode the block of data are transferred at_____	different time.	same time.	sequentially	none
The 8051 has 128 bytes of on-chip RAM with addresses _____	00-FF H	00- 7F H	80-FF H	20-80 H
Special function registers are designated with addresses_____	00-7F H	00-FF H	80-FF H	none
Register banks, bit addressable and scratch pad are designated from_____ addresses	00-7F h	00-FF h	80-FF h	7F-80 H

Register banks are designated with address_____	00-7F h	00-FF h	80-FF h	00-1F H
Bit addressable RAM space has its address from_____	20-2F H	10-7F h	00-1F H	None
Scratch pad has its address address from_____	20-2F H	10-7F h	00-1F H	30-7F H
To access SFRs we use _____ addressing modes.	indirect	register	direct	None.
_____ are not bit addressable.	registers.	I/O ports.	controlling signals.	None.
All Four ports of 8051 uses _____pins.	8	16	10	12
The 8-pions of 8051 makes them _____ bit port.	16	8	12	64
Although 8051 can have _____ Kbytes,Manufacturers have put only 4Kbytes	64Kb	128b	128Kb	64 b
MCS-51 is the name referred by Intel corporation for_____	8085	8051	89086	80286
Binary representation of 0 to 9 is called _____	HEXA	BCD	octal	binary
For multiplying two bytes in 8051 one data is placed in_____ register and the other in _____register	A,B	SFR,A	temporary	None
The 8051 indicates the existence of the error by raising the _____ flag.	error	overflow	parity	sign
In an 8-bit operand, bit _____is used for the sign bit.	D2	D3	D7	D7
The range of Bit-sized signed operands is _____ to _____	-128 to +127	+128 to -127	128 to 127	None
The instruction _____ complements the contents of A register	CP A	CPL A	CMP A.	None
To mask the certain bits of the accumulator we must ANL it with_____	1	zeros	all zeros.	None
to set certain bits of accumulator to 1 we mustORL it with _____	zeros	one	all zeros	all the above
XRL of an operand with itself results in _____	particular bit zero.	all zeros.	all ones	None
The RS pin is an _____ pi in LCD.	output	input	bidirectional	none
For an LCD to recognize information at the data pins as data, RS must be set to _____	high	low	high impedence state	none
A stepper motor is a widely used device that translates _____ pulses into mechanical movement.	electronic pulses	electrical	sweep pulses.	ramp signal.
The controlling of the angle and the direction of the stepper motor is done by_____ language.	high level	machine level	assembly	none.
A _____ is electrically controllable switch	relay	capacitor	capacitor	none
optoisolator is also called as _____	optocoupler	photovoltaic	opto electronic	none
the step angle is the minimum degree of rotation associated with _____	single step	multiple step	rotating speed.	none
8051 has _____ bytes of on-chip ROM.	4	8	4k	8k
8051 consists of _____ timers	3	2	4	1
Pseudo-instructions are also called_____	assembler	assembler directive	compiler directive	code control instructions
Microcontroller has a _____ memory	external	In-built	RAM	PROM
The 8-bit microcontroller is called _____	8085	8086	8051	80286
The register which keeps track of the execution of program in 8086 is	program counter	Stack pointer	Instruction Pointer	Accumulator
Each segment of memory is of _____ long	1 Mb	64 Kb	32 Kb	all the above
The slower peripherals and processor are synchronized using____ signal	LOCK	READY	HOLD	HLDA
In string manipulation, if DF=1, the data segment is	incremented by 1	decremented by 2	decremented by 1	None of the above
The registers in 8086 are of _____ bit wide	4	8	32	16
In 8086, the lower _____ address lines are multiplexed with data	4	8	32	16

UNIT V

Questions	opt1	opt2	opt3	opt4
Address lines A16-A19 are multiplexed with status lir	S0-S4	S1-S5	S3-S6	None of the abov
The instruction used to increment the contents of regi	INR	INC	INT	None of the abov
Source program is converted into object code using	assembler	loader	coder	linker
Clock frequency of 8086	1MHZ	3MHZ	5MHZ	7MHZ
Several object files are combined using	assembler	loader	coder	linker
Object program is placed in system memory using	assembler	loader	coder	linker
Mnemonics codes are replaced by binary codes using	assembler	loader	coder	linker
Large functioning module is created using	assembler	loader	coder	linker
_____ allows to examine register and memory c	assembler	loader	coder	linker
_____ ports of 8051 are bit addressable.	2	1	3	all the ports
Indicate which of the following registers are bit addre	A	B	PSW	all the above

_____ ports are bidirectional	p0 & p1	p2 & p4	p3 & p1	all the ports
Normally mode ____ is called basic I\O mode	0	2	1	none
the method of connecting I/O chip to CPU is called __	I/O mapped	Memory mapped	both	none
the transfer of data using parallel lines is _____	faster	slower	nominal	none
_____ are not bit addressable	registers	I/O ports.	controlling signals.	None
All Four ports of 8051 uses _____pins.	8	16	10	12
The 8-pions of 8051 makes them _____ bit port.	16	8	12	64
The register which keeps track of the execution of pro	program counter	Stack pointer	Instruction Pointer	Accumulator
Each segment of memory is of _____ long	1 Mb	64MB	32Mb	All the above
How many types of control words in 8251	4	2	3	5
Writing of control word after resetting will be recognized as	command	mode instruction	parity bit	All the above
Write a command whenever necessary after writing a	command	mode instruction	parity bit	All the above
In 8051 which interrupt has highest priority	IE1	TF0	IE0	TF1
Serial port interrupt is generated, if ____ bits are set	IE	RI,IE	IP,TI	RI,TI
In 8051,After reset the SP register is initialized to add	8H	9H	7H	6H
When the 8051 is reset and the line is LOW, the progr	internal code memory	external code memory	both	none
The 8051 has _____ 16-bit counter/timers.	1	2	3	4
The internal RAM memory of the 8051 is:	32 bytes	64bytes	128 bytes	256 bytes
The 8051 can handle _____ interrupt sources.	3	4	5	6
An alternate function of port pin P3.4 in the 8051 is:	timer 0	timer 1	interupt 0	interupt 1
The I/O ports that are used as address and data for ext	ports 1 and 2	ports 1 and 3	ports 0 and 2	ports 0 and 3
The 8051 has _____ parallel I/O ports	2	3	4	5
The total external data memory that can be interfaced	32k	128k	64k	256k
Bit-addressable memory locations are:	10H through 1FH	20H through 2FH	30H through 3FH	40H through 4FH
The 8-bit address bus allows access to an address rang	0000 to FFFFH	000 to FFFH	00 to FFH	0 to FH
The start-conversion on the ADC0804 is done by using the	$\overline{SC}$	CS line	INTR line	$V_{ref/2}$ line
Which of the following commands will move the num	MOV A, P27	MOV A, #27H	MOV A, 27H	MOV A, @27
Which of the following commands will move the valu	MOV P2, R3	MOV R3, P2	MOV 3P, R2	MOV R2, P3
The number of data registers is:	8	16	32	64
What is the difference between the 8031 and the 8051	The 8031 has no interrupts	The 8031 is ROM-less	The 8051 is ROM-less.	The 8051 has 64 I
The I/O port that does not have a dual-purpose role is	port 0	port1	port 2	port 3
Which of the following commands will copy the conte	MOV @ P1, R0	MOV @ R0, P1	MOV P1, @ R0	MOV P1, R0
Which of the following commands will copy the conte	MOV A, 04H	MOV A, L4	MOV L4, A	MOV 04H, A
The ADC0804 has _____ resolution.	4 bit	8bit	16bit	32 bit
A HIGH on which pin resets the 8051 microcontroller	RESET	RST	PSEN	RSET
An alternate function of port pin P3.1 in the 8051 is:	serial port input	serial port output	memory write strobe	memory read stro
Which of the following instructions will move the con	MOV 6R, A	MOV R6, A	MOV A, 6R	MOV A, R6
An alternate function of port pin P3.0 (RXD) in the 8(	serial port input	serial port output	memory write strobe	memory read stro
Which of the following commands will move the valu	MOV P2, R3	MOV R3, P2	MOV 3P, R2	MOV R2, P3
8051 series of micro controllers are made by which of	Atmel	Philips	none of the mentioned	both of the mentio
AT89C2051 has RAM of:	128 byte	256 bytes	64 bytes	512 bytes
8051 series has how many 16 bit registers?	2	3	1	0
When 8051 wakes up then 0×00 is loaded to which re	DPTR	SP	PC	PSW
When the micro controller executes some arithmetic (	DPTR	SP	PC	PSW
On power up, the 8051 uses which RAM locations for	00-2F	00-07	00-7F	00-0F
How are the bits of the register PSW affected if we se	PSW.5=0 and PSW.4=1	PSW.2=0 and PSW.3=1	PSW.3=1 and PSW.4=1	PSW.3=0 and PS
If we push data onto the stack then the stack pointe	increases with every push	decreases with every pusl	none of the mentioned	both of the mentio
On-chip RAM is also called _____ memory?	direct memory	indirect memory	both	none
What is the width of address bus in 8051	8 bit	16 bit	32 bit	64 bit

Answer
8- bit data bus
All the above
Control and arithmetic unit
program counter
Assembly language
READY
I/O write
two byte
Register indirect
overflow
Address Latch Enable
BCD
Accumulator
All the registers
delay
delay
CMA
16
CMP M
RET
program counter
Accumulator
PC
Sign
Aux. carry
Carry
parity
IR
HOLD
RESET
READY
SOD
IO/M-=0, S1=1,S0=1
IO/M-=0, S1=1,S0=0
IO/M-=0, S1=0,S0=1
IO/M-=1, S1=1,S0=0
IO/M-=1, S1=0,S0=1
0020H
0028H
0030H
0038H
Immediate
Direct
Reg. indirect
Implied
Immediate
Direct
Reg. indirect
Implied
1
2
3
1
2
3
CMA
CMC
CMP
INR
INX

Answer
DCX

1Mb

4

Instruction Pointer

Memoryread

64Kb

READY
6 byte

20

decremented by 1

16

16

S3-S6

INC

segment registers

Procedures

CMC

INTA

LOCK
Effective address
Overflow flag
Trap flag
interrupt flag
Direcion flag

Register

Immediate

Direct

Register indirect

INC

STC

CMC

CMP

DEC

assembler
linker

loader

assembler
linker

loader

FIFO
5MHz

memoryread

INC
8085

2
2

2
2

byte

word
byte
word

3

4

4
1
1
1
4

Answer
3
Character

TXD
3
framing error

To convert parallel date into serial data
To covert serial data into parallel date
Buffer register is empty
out put buffer is empty
Block

Synchronous Characters

parallel

24
3

8
Port A
port C

for all ports

A0-A1

41 ,42

from port address to ACC.

from Accumulator to port address
Accumulator

0

D0

Bit set reset
3

8

RL0 –RL7
out A3 –A0 , outB3-B0
0 0

IRR

ISR

IMR
Read / Write logic
DRQ0
DRQ1
4
16
2

8253
1
In-built
8051
128 bytes.
4K
2
1
8051

4K
all the above.

All the above.

8

assembler directive

ORG 20000 H
flag
8
4
8 bits.
decremented.

Answer
incremented

8

3
2
3
1

4
8
p0

direct.

16bytes.

all the ports.
all the above

all the ports.
0
Memory mapped

faster
asynchronous
5V
Txd &rxd

same time.

00-FF H

80-FF H

00-7F h

00-1F H

20-2F H

30-7F H

direct

registers.
8
8

64Kb

8051

BCD

A,B

overflow

D7

-128 to +127

CPL A

zeros

one

all zeros.
input

high

electrical

assembly
relay
optocoupler

single step
4k
2

assembler directive
In-built
8051

Instruction Pointer
64 Kb

READY

decremented by 1
16

16

Answer
S3-S6
INC
assembler
5MHZ
linker
loader
assembler
linker
loader
all the ports
all the above

all the ports
0
Memory mapped
faster
registers
8
8
Instruction Pointer
64MB
2
Mode instruction

Command
IE0
RI,TI
7H
external code memory
2
128 bytes
5
timer 0
port 0 and 2
4
64Kb
20H through 2FH
00 to FFH
SC
MOV A, #27H
MOV R2, P3
32
The 8031 is ROM-less
port 1
MOV P1, @ R0
MOV A, 04H
8 bit
RST
serial port output
MOV R6, A
serial port input
MOV R2, P3
both of the mentioned
256 bytes
2
PC
PSW
00-07
PSW.3=0 and PSW.4=1
increases with every push
direct memory
16 bit