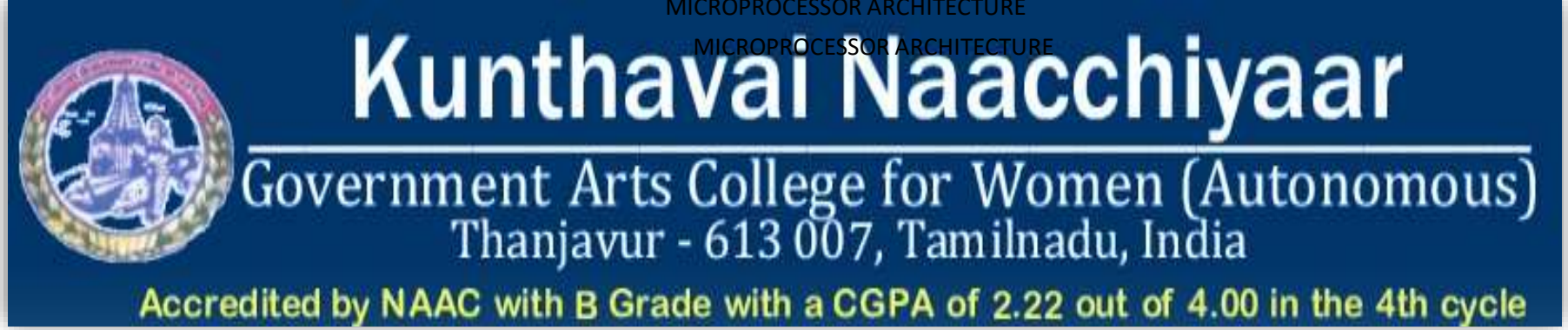
MICROPROCESSOR ARCHITECTURE



Microprocessor Architecture

Subject code:18K5CS07

Faculty: 1. Mrs. R. Valarmathy.
2. Mrs. K. Sharmila.

UNIT III

Examples of Assembly Language Programs: Introduction – simple Examples-Addition of two 8-bit Numbers; sum 8-bit -8 bit subtraction- Addition of two 8-bit Numbers; sum 16-bit- Move a block of Memory from one section of memory to Another Section of Memory.

UNIT IV

Intel 8086 and Intel's other 16-bit microprocessor: Introduction-Intel 8086: pin Description-Operating Modes of 8086-Pin Description for minimum and pin Description for Maximum mode. Assembler Directives: Assembler directives of Intel 8086- Assembly language Programs using Assembler.

UNIT V

Microcontroller: Introduction- Intel 8051 series of Microcontrollers(MCS-51)Registers-Pins of Intel 8051- I/O Lines- The Intel 8051 Interrupts-Instruction set- Memory Organisation of Intel 8051-Addressing Modes.

Text Book:" Fundamentals of Microprocessor and Microcontrollers" – B.Ram & Sanjay Kumar

Introduction

- A microprocessor executes instructions given by the user
- Instructions should be in a language known to the microprocessor
- Microprocessor understands the language of 0's and 1's only
- This language is called **Machine Language**
- For e.g.01001111
  - Is a valid machine language instruction of 8085
  - It copies the contents of one of the internal registers of 8085 to another.

Machine language program to add two numbers

```
00111110    ;Copy value 2H in register A
00000010
00000110    ;Copy value 4H in register B
00000100    ;A = A + B
10000000
```

Assembly language of 8085

- It uses English like words to convey the action/meaning
- For e.g.

  - MOV to indicate data transfer to add two
  - ADD values
  - SUB to subtract two values

SIMPLE EXAMPLES

Move the content of the memory location FC50 H to register C

The content is 08

Memory address	Machine codes	Mnemonics	Operands	Comments
FC00	21,50FC	LXI	H, FC50	Get the memory address FC00 in H – L Pair
FC03	4E	MOV	C, M	Move the content of the memory location, whose address is in the H – L pair, to register C
FC04	76	HLT		Halt

Place the content FC50 in reg B and FC51 H in reg C

Memory address	Machine codes	Mnemonics	Operands	Comments
FC00	21,50,FC	LXI	H, FC50	Get the memory address FC00 in H – L Pair
FC03	46	MOV	B, M	Move the content of the memory locationTo register B
FC04	23	INX	H	Increement HL pair by 1
FC05	4E	MOV	C,M	Mov the content FC51 to C
FC04	76	HLT		Halt **OUTPUT: FC50—11H** **FC51—12H**

Load the content of memory location FC50 H directly to accumulator,then transfer it to regB

Memory Address	Machine codes	Mnemonics	**Operands**	**commands**
FCOO	3A,50,FC	LDA	FC50	Get the content of memory location FC50 H into Accumulator
FC03	47	MOV	B,A	Move the content of register A to register B
FCO4	76	HLT		HALT

Place 05 in reg B

Memory address	Machine codes	Mnemonics	Operands	Comments
FC00	06, 05	MVI	B, 05	Get 05 in register B
FC02	76	HLT		Stop

Addition of two 8-bit numbers .sum 8-bit

Memory address	Machine codes	Mnemonics	Operands	Comments
2000	21,01,25	LXI	H, 2501 H	Get address of 1st number H – L pair
2003	7E	MOV	A, M	1st number in accumulative
2004	23	INX	H	Increment content of H L pair
2005	86	ADD	M	Add 1st and 2nd number
2006	32,03,25	STA	2503H	Store sum in 2503 H
2009	76	HLT	**Input data 2501-49H 2502—56H Result=9F**	Stop

Subtraction of two 8-bit numbers

Memory address	Machine codes	Mnemonics	Operands	Comments
2000	21,01,25	LXI	H, 2501 H	Get address of 1st number H – L pair
2003	7E	MOV	A, M	1st number in accumulative
2004	23	INX	H	Increment content of H L pair
2005	96	SUB	M	SUB1st and 2nd number
2006	23	INX	H	Content of H-L pair becomes 2503H
2007	77	MOV	M,A	Store sum in 2503 H
2008	76	HLT		Stop **Input Data** **2501—49H** **2502—32H** **OUTPUT** **2503—17H**

Addition of two 16-bit number

Memory address	Machine codes	Labels	Mnemonics	Operands	Comments
2000	2A,01,25		LHLD	2501 H	1st 16-bit number in H-L pair.
2003	EB		XCHG		Get 1st number in D- pair.
2004	2A,03,25		LHLD	2503 H	2nd 16-bit number in H-L pair.
2007	0E,00		MVI	C,00	MSBs of the sum in register C. Initial value =00
2009	19		DAD	D	1st number +2nd number
200A	D2,0E,20		JNC	AHEAD	Is carry? No ,go to the label AHEAD.
200D	0C		INR	C	Yes , increment C.
200DE	22,05,25	AHEAD	SHLD	2505 H	Store LSBs of sum in 2025 and 2506 H.
2011	79		MOV	A,C	MSBs of sum in

Decimal addition of two 8- bit number

Memory address	Machine codes	Labels	Mnemonics	Operands	Comments
2000	21,01,25		LXI	H,2501 H	Address of 1st number in H-L pair.
2003	0E,00		MVI	C,00	MSDs of sum in register C. Initial value =00.
2005	7E		MOV	A,M	1st number in accumulator.
2006	23		INX	H	Address of 2nd number 2502 in H-L pair.
2007	86		ADD	M	1st number +2nd number
2008	27		DAA		Decimal adjust.
2009	D2,0D,20		JNC	AHEAD	Is carry? No ,go to the label AHEAD.
200C	0C		INR	C	Yes , increment C.
200D	32,03,25	AHEAD	STA	2503 H	LSDs of sum in

8—bit decimal subtraction

Memory address	Machine codes	Mnemonics	Operands	Comments
2000	21,02,25	LXI	H,2502 H	Get address of 2nd number in H-L pair .
2003	3E,99	MVI	AA,99	Place 99 in accumulator.
2005	96	SUB	M	9's complement of 2nd number.
2006	3C	INR	A	10's complement of 2nd number.
2007	2B	DCX	H	Get address of 1st number .
2008	86	ADD	M	Add 1st number and 10's complement of 2nd number.
2009	27	DAA		Decimal adjustment.
200A	32,03,25	STA	2503 H	Store result in 2502 H.
200D	76	HL,T		Halt

Find 1's complement of 8-bit number

EXAMPLE: Find one's complement of 96

96= 1001 0110
(9) (6)
1's com= 0110 1001 =69
(6) (9)

Output
2501—96H
2502—69H

Memory address	Machine codes	Mnemonics	Operands	Comments
2000	3A,01,25	LDA	2501 H	Get data in accumulator.
2003	2F	CMA		Take it 1's complement.
2004	32,02,25	STA	2502 H	Store result in 2502 H.
2007	76	HLT		Stop.

EXAMPLE: Find two's complement of 96

	96= 1001 0110	
	(9)	(6)
1's com=	0110	1001 =69
	(6)	(9)
	+0000	0001
2,s =	0110	1010=6A
	(6)	(A)

Output
2501—96H
2502—6AH

Memory address	Machine codes	Mnemonics	Operands	Comments
2000	3A,01,25	LDA	2501 H	Get data in accumulator.
2003	2F	CMA		Take it 1's complement.
2004	3C	INR	A	Take it 2's complement.
2005	32,02,25	STA	2502 H	Store result in 2502 H.
2008	76	HLT		Stop.

TO FIND LARGER OF TO NUMBERS

Memory address	Machine codes	Labels	Mnemonics	Operands	Comments
2000	21,01,25		LXI	H,2501 H	Address of 1st number in H-L pair.
2003	7E		MOV	A,M	1st number in accumulator .
2004	23		INX	H	Address of 2nd number in H-L pair.
2005	BE		CMP	M	Compare 2nd number with 1st number. Is the 2nd number >1st ?
2006	D2,0A,20		JNC	AHEAD	No, larger

- The first number 98H is placed in the memory location 2501H
- The second number 87H is placed in the memory location 2502H

❖Data
2501—98H
2502—87H
result is 98 H and it is stored in the memory location 2503H
2503—98H

Memory address	Machine codes	Mnemonics **To find square from Lookup table**	Operands	Comments
2000	3A,00,25	LDA	2500 H	Get data in accumulator.
2003	6F	MOV	L,A	Get data in register L.
2004	26,26	MVI	H,26 H	Get 26 in register H.
2006	7E	MOV	A,M	Square of data in accumulator.
2007	32,01,25	STA	2501 H	Store square in 2501 H.
200A	76	HLT		Stop **Data:** **2500—07D** **result:** **2501—49D**

Ex: find square of 07(decimal)using lookup table technique.

- The no 07 D is in the memory location 2500H
- The result is to be stored in the memory location 2501H`
- The table of square is stored from 2600 to 2609.

Masking off MS4BITS OF AN 8-BIT NUMBER

Ex: num=A6=1010 0110
(A) (6)
RESULT=06=0000 0110
(0) (6)

Memory address	Machine codes	Mnemonics	Operands	Comments
2000	3A,01,25	LDA	2501H	Get data in accumulator.
2003	E6,0F	ANI	0F	Mask off the most significant 4 bit.
2005	32,02,25	STA	2502 H	Store result in 2502 H.
2008	76	HLT		Stop.

Masking off LS4BITS OF AN 8-BIT NUMBER

Ex: num=A6=1010 0110
(A) (6)
RESULT=06=0000 0110
(A) (0)

Memory address	Machine codes	Mnemonics	Operands	Comments
2000	3A,01,25	LDA	2501H	Get data in accumulator.
2003	E6,0F	ANI	F0	Mask off the LEAST significant 4 bit.
2005	32,02,25	STA	2502 H	Store result in 2502 H.
2008	76	HLT		Stop.

Shift an 8-bit num left by one bit

Memory address	Machine codes	Mnemonics	Operands	Comments
2000	3A	LDA	2501 H	Get data in accumulato r.
2003	87	ADD	A	Shift it left by one bit.
2004	32,02,25	STA	2502 H	Store result in 2502 H.
2007	79	HLT		Halt. **output 2501—65H 2502—CA H**

Shift an 8-bit num left by two bit

Memory address	Machine codes	Mnemonics	Operands	Comments
2000	3A	LDA	2501 H	Get data in accumulato r.
2003	87	ADD	A	Shift it left by one bit.
2004	87	ADD	A	AGAIN SHIFT BY ONE BIT
2005	32,02,25	STA	2502 H	Store result in 2502 H.
2008	79	HLT		Halt.

SHIFT 16-BIT NUMBER LEFT BY TWO BIT

Memory address	Machine codes	Mnemonics	Operands	Comments
2000	2A,01,25	LHLD	2501 H	Get data in H-L pair.
2003	29	DAD	H	Shift it left by one bit.
2004	29	DAD	H	AGAIN SHIFT BY ONE BIT
2005	22,03,25	SHLD	2503 H	Store result in 2503 and 2504H.
2006	76	HLT		Halt.

SHIFT 16-BIT NUMBER LEFT BY ONE BIT

Memory address	Machine codes	Mnemonics	Operands	Comments
2000	2A,01,25	LHLD	2501 H	Get data in H-L pair.
2003	29	DAD	H	Shift it left by one bit.
2004	22,03,25	SHLD	2503H	Store result in 2503 and 2504H.
2007	76	HLT		STOP

Find 1's complement of a 16-bit number

Memory address	Machine codes	Mnemonics	Operands	Comments
2000	21,01,25	LXI	H,2501 H	Address of LSBs of the number.
2003	7E	MOV	A,M	8 LSBs of the number in accumulator.
2004	2F	CMA		Complement of 8 LSBs of result.
2005	32,03,25	STA	2503 H	Store 8 LSBs of result.
2008	23	INX	H	Address of MSBs of the number.
2009	7E	MOV	A,M	8 MSBs of the number in accumulator.
200A	2F	CMA		Complement of 8 MSBs of the result.
200B	32,04,25	STA	2504	Store 8 MSBs of the result.
200E	76	HLT		Halt **Output 2501—85H,LSB of the number**

Memory address	Machine codes	Mnemonics **2,s complement of 16-bit number**	Operands	Comments
2000	21,01,25	LXI	H,2501 H	Address of LSBs of the number.
2003	06,00	MVI	B,00	USE THE REGISTER B to the carry.
2005	7E	MOV	A,M	8 LSBs IN ACCUMULATOR.
2006	2F	CMA		1's complement of 8 lsb
2007	C6,01	ADI	01	2'S COMPLEMENT OF 8 LSB
2009	32,03,25	STA	2503 H	STORE 8 LSB OF THE RESULT
200C	D2,10,20	JNC	GO	
200F	04	INR	B	STORE CARRY
2010	23	GO INX	H	ADDR OF 8 MSB

MOVE A BLOCK OF DATA FROM ONE LOCATION TO ANOTHER

PROGREAM			
2400	21,00,20	LXI H,2000H	Get memory addre of count
2403	4E	MOV C,M	Count in Register C
2404	23	INX H	Source address of d in H-L pair
2405	11,01,22	LXI D,2201	Destiny address of data in D-E pair
2408	7E LOOP	MOV A,M	Data from source address to ACC
2409	EB	XCHG	Destiny address HL pair
240A	77	MOV M,A	Data to destiny address
240B	EB	XCHG	Source address in H pair
240C	23	INX H	Source address of n data
240D	13	INX D	Destiny address of next data

MULTIBYTE ADDTION

PROGREAM			
2400	21,00,25	LXI H,2500H	Get memory addre of count IN H-L pai
2403	4E	MOV C,M	Count in Register C
2404	23	INX H	Address of 1st byte 1st number
2405	11,01,26	LXI D,2601	Address of 1st byte 2nd number
2408	B7	ORA A	Clear carry
2409	1A LOOP	LDAX D	Get byte of 2nd nun in accumulator
240A	8E	ADC M	Add 2nd and 1st witl carry
240B	27	DAA	Decimal adjust
240C	77	MOV M,A	Store sum in H-L pa
240D	13	INX D	Increment D-E pair
240E	23	INX H	Increment H-L pair
240F	OD	DCR C	Decrement count
2410	C2,09,24	JNZ LOOP	Is count 00?, No, go

MULTIBYTE SUBTRACTION

PROGREAM			
2400	21,00,25	LXI H,2500H	Get memory addre of count IN H-L pai
2403	4E	MOV C,M	Count in Register C
2404	23	INX H	Address of 1st byte 1st number
2405	11,01,26	LXI D,2601	Address of 1st byte 2nd number
2408	B7	ORA A	Clear carry
2409	1A LOOP	LDAX D	Get byte of 2nd nun in accumulator
240A	9E	SBB M	SUB 2nd and 1st witl BORROW
240B	77	MOV M,A	Store sum in H-L pa
240C	23	INX D	Increment D-E pair
240D	13	INX H	Increment H-L pair
240E	OD	DCR C	Decrement count
241F	C2,09,24	JNZ LOOP	Is count 00?, No, go loop

To find the largest number in a data array

Memory address	Machine codes	Labels	Mnemonics	Operands	Comments
2000	21,00,25		LXI	H,2500 H	Address of count in H-L pair.
2003	4E		MOV	C,M	Count in register C
2004	23		INX	H	Addr of 1st num n H-L pair
2005	7E		MOV	A,M	1st num in acc.
2006	0D		DCR	C	Decrement C
2007	23	LOOP	INX	H	Addr of next number
2008	BE		CMP	M	Icompare next num with previous is next num> previous.
2009	D2,0D,20		JNC	AHEAD	No,larger number is in acc.Goto Ahead
200C	7E	AHEAD	MOV	A,M	Yes, get larger in acc.
200D	0D		DCR	C	Decr ement C
200E	32,07,20		JNZ	LOOP	

To find the smallest number in a data array

Memory address	Machine codes	Labels	Mnemonics	Operands	Comments
2000	21,00,25		LXI	H,2500 H	Address of count in H-L pair.
2003	4E		MOV	C,M	Count in register C
2004	23		INX	H	Addr of 1st num n H-L pair
2005	7E		MOV	A,M	1st num in acc.
2006	0D		DCR	C	Decrement C
2007	23	LOOP	INX	H	Addr of next number
2008	BE		CMP	M	Icompare next num with previous is next num> previous.
2009	D2,0D,20		JC	AHEAD	No,larger number is in acc.Goto Ahead
200C	7E	AHEAD	MOV	A,M	Yes, get larger in acc.
200D	0D		DCR	C	Decr ement C
200E	32,07,20		INZ	LOOP	

Sum of series

Memory address	Machine codes	Labels	Mnemonics	Operands	Comments
2400	21,01,25		LXI	H,2500 H	Address of 1st number in H-L pair.
2403	4E		MOV	C,M	COUNT IN REG C
2404	3E,00		MVI	A,00	Initial value of sum =00
2406	23		INX	H	Addr of next value inH-L pair
2407	86		ADD	M	Previous sum +next number
2408	0D		DCR	C	Decrement count
2409	C2,06,24	AHEAD	JNZ	LOOP	Is count=0? No, jump to loop
240C	32,50,24		STA	2450	Store sum in 2450H
240F	76		HLT		stop

UNIT IV

Introduction

- Intel ‘s 16-bit Microprocessor are 8086, 80186, 80286. These Microprocessors have 16-bit Internal architecture and contain 16 data lines.

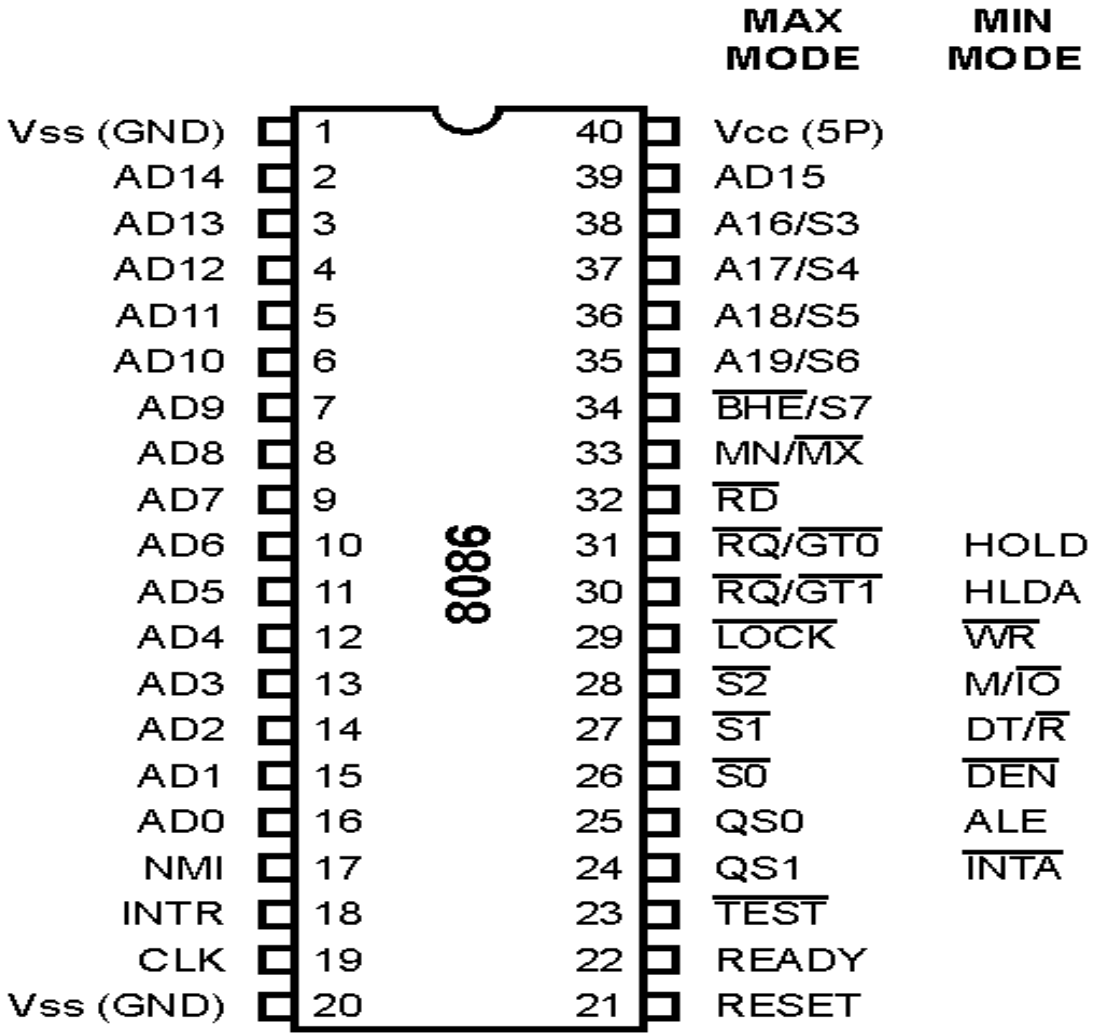
- The Intel 8086, 80186, 8088 and 80188 microprocessors have same basic set of registers, instructions and addressing modes.

- Besides CPU, they also contain on-chip clock generator, programmable memory and chip select logic, programmable interrupt controller, high speed DMA channels, programmable 16-bit timer, local bus controller etc.,

INTEL 8086

- The 8086 is a 16 bit Microprocessor.
- It consumes less power. It requires +5V power supply.
- Its clock frequency for its different versions are 5, 8, 10 MHz it was introduced in 1978. it contains an electronic circuitry of 29000 transistors.
- A 40 pin dual in line package(DIP).
- 8086 has 20 address lines and 16 data lines. It can access up to 2^{20} memory locations (1MB).
- The 16-bit data word is divided into a low-order byte and a higher-order byte.
- The 20 address lines are time multiplexed. 16 low-order address lines are multiplexed with data and 4 higher-order address lines are multiplexed with status signal.
- It provides 16-bit registers.
- AD 0- AD15 are 16 low-order address lines . 8 LSB of data are transmitted onAD0-AD7 and 8 MSB of data are transmitted on AD8-AD15.
- 8086 is designed to operate in two modes, Minimum and Maximum.

Pin Diagram of 8086 Microprocessor



Pin Description of 8086

A0-A15: It's a Bidirectional Address/data lines. These are lower-order address bus. They are multiplexed with data. AD lines are used to transmit memory address the symbol A is used instead of AD.

A10-A19: High order Address lines. These are multiplexed with status signal.

A16/S3, A17/S4: A16 and A17 are multiplexed with segment identifier signals s3 and s4.

A18/S5: A18 are multiplexed with interrupt status S5.

A19/S6: A19 is multiplexed with status signal S6.

BHE/S7: The bus high enable (BHE) signal is used to indicate the transfer of data over the higher order data bus.

$\overline{\text{RD}}$**(Read) : This** signal is used for read operation. It is an output signal. It is active when LOW.

READY(Input): The addressed I/O or memory sends acknowledgment through this pin when HIGH it indicates that the peripheral is ready to transfer data.

RESET (Input): System is reset when the signal is active HIGH.

CLK(Input): Clock 5,8,or 10 MHZ

INTR (Interrupt Request)

NMI (Input). Non-Maskable Interrupt Request.

$\overline{\text{TEST}}$(Input). Wait for test control. When it is low the microprocessor continues execution otherwise waits.

VCC. Power supply, +5V d.c

GND. Ground.

Operation Modes of 8086

There are two modes of Operation

1.Minimum mode.

2. Maximum mode.

- When only one 8086 CPUis to be used in a microcomputer system the 8086 used minimum mode of operation. In this mode the COU issues the control signals required by memory and I/O devices.
- In a multiprocessor system it uses maximum mode.

Pin Description for Minimum Mode

The Minimum mode of operation the pin MN/MX is connected to 5 V d.c supply. The description of the pins from 24 to 31 for the minimum mode is as follows.

$\overline{\text{INTA}}$(Output): Pin No.24. Interrupt Acknowledgement. On receiving Interrupt signal the processor issues an acknowledge signal. It is active LOW .

ALE (Output). Pin No.25. Address Latch Enable. It goes High during T1. the microprocessor sends this signal to latch the address into the Intel 8282/8283 latch.

$\overline{DEN}$ (Output). Pin No.26. Data enable. When Intel 8286/8287 octal bus transceiver is used this signal acts as an output enable signal. It is active LOW.

DT/$\overline{R}$ (Output). Pin No 27. Data Transmit/Receive. When Intel 8286/8287 octal bus transceiver is used this signal controls the direction of data flow through the transceiver. When it is HIGH data are sent out. When it is LOW data are received.

M/$\overline{IO}$(Output). Pin No.28. Memory or I/O access. When it is HIGH the CPU wants to access memory. When it is LOW the CPU wants to access I/O device.

$\overline{WR}$(Output). Pin No.29. Write. When it is LOW the CPU performs memory or I/O Write operation.

HLDA(Output).Pin No.30. HOLD acknowledge. It is issued by the processor when it receives HOLD signal. It is active HIGH signal. When HOLD request is removed HLDA goes LOW.

HOLD(Input).Pin No.31. Hold. When another device in microprocessor system wants to use the address data bus, it sends a HOLD request to CPU through this pin. It is an active HIGH signal.

Pin Description for Maximum Mode

The Maximum mode of operation the pin MN/$\overline{MX}$ is made LOW. It is grounded. The description of the pins from 24 to 31.

QS1, QS0 (Output) Pin No.24,25. Instruction Queue Status. Logic are given below.

QS1	*QS0*	
0	0	No Operation
0	1	1st byte of the opcode from queue
1	0	Empty the queue
1	1	Subsequent byte from the queue.

***S*2 , *S*1, *S*0 (Output). Pin No. 26,27,28** Status lines: These are the status lines are connected to the bus controller. The bus controller generates memory and I/O access control signals.
These status lines logic are as follows

*S*2	*S*1	*S*0	Function
0	0	0	Interrupt Acknowledge
0	0	1	Read I/O port
0	1	0	Write I/O port
0	1	1	Halt
1	0	0	Code Access
1	0	1	Read memory
1	1	0	Write memory
1	1	1	Passive (In active)

***LOCK* : Pin No.29.** When it active LOW signal. All interrupts are masked and no HOLD request are granted.

***RQ / GT*1 and *RQ / GT* 0(**Bidirectional)Pin No 30,31. request/Grant: The request/grant pins are used by other local bus masters to force the processor to release the local bus at the end of the processors current bus cycle. These lines are bi- directional and are used to both request and grant a *DMA* operation. *RQ* / *GT* 0 is having higher priority than *RQ* / *GT*1

Assembler Directives:

- An Assembler directive is a statement to give direction to the assembler to perform the task of assembly Process.

- The assembler Directives control organization of the program and provide necessary information to the assembler to understand assembly language program to generate machine codes.

- An Assembly language program consists of two types of statements: Instructions and Directives.

- The instructions are translated to machine code by the assembler whereas the directives are not translated to machine code.

Symbols, Variables and Constants:

- The symbols whose values can be varied while running a program, is called variable.
- The name of the variable should be meaningful to increase readability and to make program maintenance easy.
- E x: Upper case and Lower case Letter A to Z; a to z
  - Digits ; 0 to 9
  - Special characters: $, ?, @, _(underscore)
  - A variable can use of the following:
  - A to Z : a to z; 0 to 9; @ and _(underscore)
  - Digits cannot be used as a first character of an assembler variable.
  - $ and ? Are not used in a variable because they are used for other purposes in an assembly language .
  - The variable must begin with an alphabet or an underscore. No difference between upper case and lower case.
  - The length of the assembler variable depends on the particular assembler.

- A numeric constant may be a binary, decimal, or hexadecimal number.

- To differentiate them, symbols B,D,H are used at the end of the binary, decimal, and hexadecimal number. A number without any identification symbol is taken as decimal number.

- The decimal and hexadecimal numbers are converted to their binary equivalent for processing.

- A hexadecimal number with its first digit between A to F, must begun with 0; otherwise it will be taken as a symbol. A hexadecimal number AFB6 must be written as 0A5B6H.

To assign Names to Variables, Constants and Addresses:

- ALP contains three general types of data: Variables, Constants and addresses.

- If names are assigned to variables, constants and addresses, The assembler uses these names to get the corresponding data items or address when the programmer refers to them in instructions.

- Assembler directives are used to assign names to variables and constants. Label are used to give names to addresses.

- The directives DB, DW, DD, DQ, and DT are used to assign names to the variable and specify their types. The Directive EQU is used to give names constants.

Assembler Directives for Intel 8086 Microprocessor:

ASSUME

DB - Defined Byte.

DD - Defined Double Word

DQ - Defined Quad Word

DT - Define Ten Bytes

DW - Define Word

ASSUME Directive: The ASSUME directive is used to inform the assembler that the name of the logical segment should be assigned for a different segment used in as assembly language program.

The 8086 works directly with only 4 physical segments: a Code segment, a data segment, a stack segment, and an extra segment.

Example:

ASSUME CS: CODE: This tells the assembler that the logical segment named CODE contains the instruction statements for the program and should be treated as a code segment.

ASUME DS:DATA ;This tells the assembler that for any instruction which refers to a data in the data segment, data will found in the logical segment DATA.

DB: (Define byte). DB directive is used to declare a byte- type variable or to store a byte in memory location.

Example:

WEIGHT DB 85 ; This directive informs the assembler to reserve one byte of memory space for the variable named WEIGHT and initialize the value 85.

ARRAY **DB 32,42,59,67,83** ;This directive informs the assembler to reserve five Bytes of consecutive memory space for the variable named ARRAY. The memory location are to be initialized with the values 32, 42, 59, 67 and 83.

TEMP DB 100 DUP(?) ;Set 100 bytes of storage in memory and give it the name as TEMP, but leave the 100 bytes uninitialized. Program instructions will load values into these locations.

DW: (Define Word) The DW directive is used to define a word type variable(ie variable which occupies two-bytes of memory space)

Example:

MULTIPLIER DW 437Ah ; this declares a variable of type word and named it as MULTIPLIER. This variable is initialized with the value 437Ah when it is loaded into memory to run.

EXP1 DW 1234h, 3456h, 5678h ; this declares an array of 3 words and initialized with specified values.

STOR1 DW 100 DUP(0); Reserve an array of 100 words of memory and initialize all words with 0000.Array is named as STOR1.

END(End of the Program): END directive is placed after the last statement of a program to tell the assembler that this is the end of the program module. The assembler will ignore any statement after an END directive. Carriage return is required after the END directive.

ENDP(End procedure): ENDP directive is used along with the name of the procedure to indicate the end of a procedure to the assembler

Example:

```
SQUARE_NUM    PROCE              ; It start the procedure
              :          ;Some steps to find the square root of a number
SQUARE_NUM    ENDP       ;Hear it is the End for the procedure
```

ENDS(Ends the segment) This ENDS directive is used with name of the segment to indicate the end of that logic segment.

Example:

```
CODE    SEGMENT                 ;Hear it Start the logic
                :               ;segment containing code
                        ; Some instructions statements to perform the logical
                        ;operation
CODE    ENDS                    ;End of segment named as
                                ;CODE
```

EQU (Equate) This EQU directive is used to give a name to some value or to a symbol. Each time the assembler finds the name in the program, it will replace the name with the value or symbol you given to that name.

Example:

FACTOR EQU 03H ; you has to write this statement at the starting of your program and later in the program you can use this as follows

ADD AL, FACTOR ; When it codes this instruction the assembler will code it as ADD AL, 03H

EVEN: This **EVEN** directive instructs the assembler to increment the location of the counter to the next even address if it is not already in the even address.

GROUP: The **GROUP** directive is used to group the logical segments named after the directive into one logical group segment.

INCLUDE: This **INCLUDE** directive is used to insert a block of source code from the named file into the current source module.

PROC(procedure). The **PROC** directive is used to identify the start of a procedure. The term near or far is used to specify the type of the procedure.

TYPE:TYPE operator instructs the assembler to determine the type of a variable and determines the number of bytes specified to that variable.

Assembly Language Program using assembler :

```
                                Addition of two 16-bit numbers
SSEG    SEGMENT         `STACK`   :Stack segments starts.
        DB 100 DUP      (`STACK----) :100 Bytes allocated for stack.
SSEG    ENDS                                    :End of the stack segment.
OP1     DW      1234H                           ;op1 type word
OP2     DW      2356H                           ;OP2 type2
RESULT DW       0000,0000                       ;MEMORY ALLOCATION FOR RESULT
DSEG    ENDS                                    ;End of the data segment
CSEG    SEGMENT         'CODE'                  ;Code segment starts.
ASSUME   CS:CSEG, DS:DSEG, SS:SSEG              ;directive for segment register.
MAIN    PROC                                     ;starts of the procedure named MAIN
;statements for startup codes for proper return in OS environment
        PUSH    DS                              ;Save present value of DS in stack.
        MOV AX, 00                              ;initialize AX=00 for return.
        PUSH AX                                 ;save AX in stack.
;statements for initialization of data segment register
        MOV AX,DSEG                             ;AX be loaded with segment register
        MOV DS,AX                               ; load data segment register with
segment address.
```

```
; start of the addition program
                    MOV AX,OP1          ;mov operand 1 to AX
                    MOV BX,OP2          ;mov operand 2 to BX
                    ADD AX,BX           ;add AX to BX,result in BX
                    MOV RESULT,AX       ;save result in memory
                    EXIT                ; insert exit code for return from
                                        the procedure
                    RET
MAIN                ENDP                ; end of the procedure main
CSEG      ENDS                          ;end of the code segment
          END MAIN                      ; End of the program.
```

UNIT-V Microcontrollers

Introduction:

❖ A microcomputer built on a single semiconductor chip is called single-chip microcomputer.

❖It is used for dedicated application such as automatic control of equipment, machines, and process of industry, instrumentation, commercial and control application.

❖Microprocessor are generally used in control applications , they are called microcontrollers. Microcontroller are embedded in the system which they control and they are called embedded controllers.

❖Microcomputer contains the component such as CPU, RAM, ROM/EPROM,I /O lines etc., A program for microcontroller is developed in a laboratory and tested then it is stored in ROM/EPROM of the microcontroller.

The ROM/EPROM of the microcontroller is known as program memory. Data and intermediate results are stored in RAM, it is called data memory.

The ROM/EPROM of the microcontroller is known as program memory. Data and intermediate results are stored in RAM, it is called data memory.

- The first microcontroller TMS1000 was introduced by Texas Instruments in the year 1974.
- Later the Intel company produced its first Microcontroller 8048 with a CPU and 1K bytes of EPROM, 64 Bytes of RAM an 8-Bit Timer and 27 I/O pins in 1976.
- the most popular controller 8051 in the year 1980 with 4K bytes of ROM,128 Bytes of RAM , a serial port, two 16-bit Timers , and 32 I/O pins.
- The 8051 family has many additions and improvements over the years .
- INTEL introduced a 16 bit microcontroller 8096 in the year 1982 .
- The 32-bit microcontrollers have been developed by IBM and Motorola.

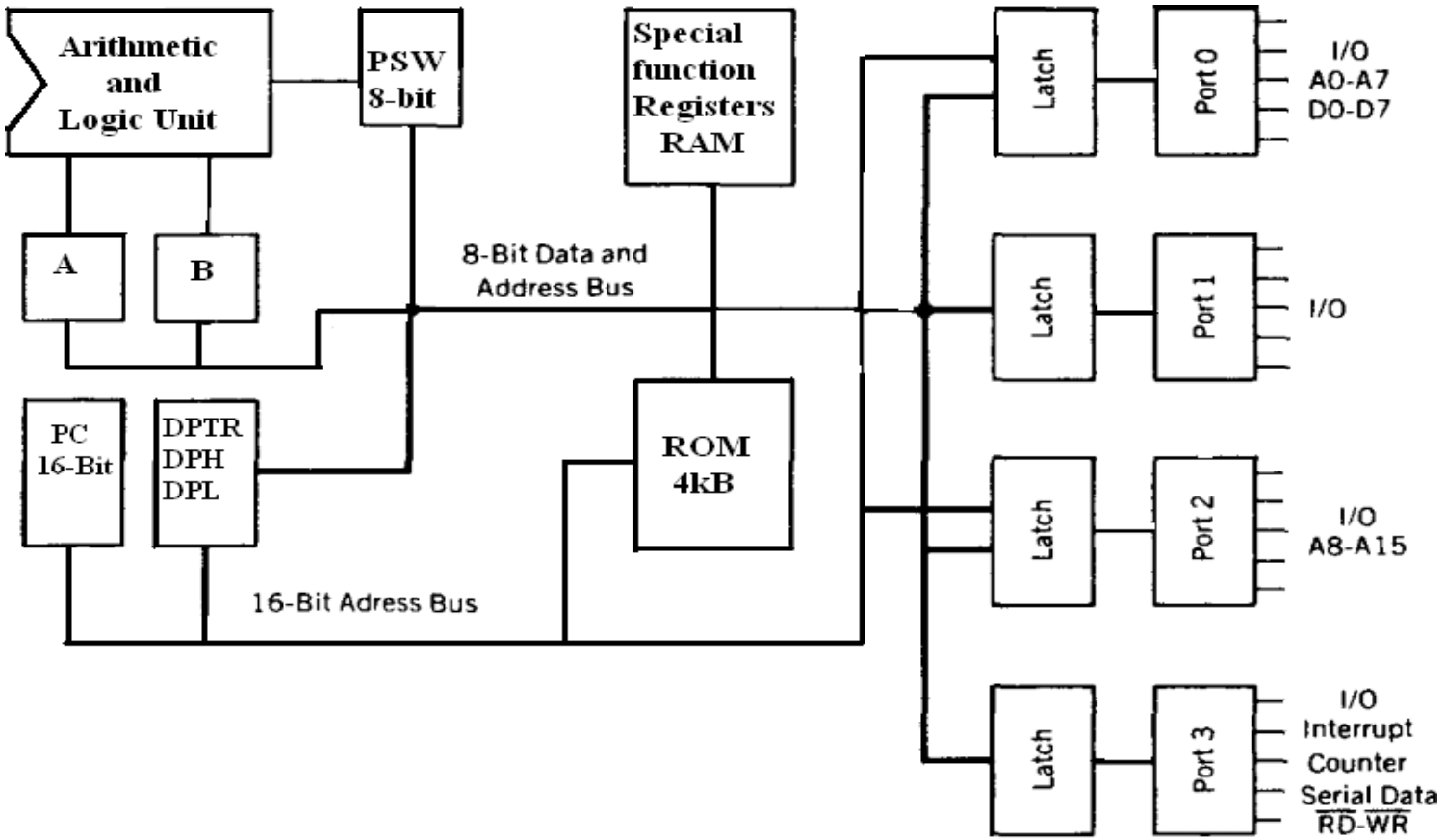
INTEL 8051 MICRCONTROLLER :

The 8051 microcontroller is a very popular 8-bit microcontroller introduced by Intel in the year 1981 and it has become almost the academic standard now a days. The 8051 is based on an 8-bit CISC core with Harvard architecture. Its 8-bit architecture is optimized for control applications with extensive Boolean processing. It is available as a 40-pin DIP chip and works at +5 Volts DC. The salient features of 8051 controller are given below.

SALIENT FEATURES : The salient features of 8051 Microcontroller are

i. 4 KB on chip program memory (ROM or EPROM)).
ii. 128 bytes on chip data memory(RAM).
iii. 8-bit data bus
iv. 16-bit address bus
v. 32 general purpose registers each of 8 bits
vi. Two -16 bit timers T_0 and T_1
vii. Five Interrupts (3 internal and 2 external).Four Parallel ports each of 8-bits (PORT0, PORT1,PORT2,PORT3) with a total of 32 I/O lines.
ix. One 16-bit program counter and One 16-bit DPTR (data pointer)
x. One 8-bit stack pointer
xi. One Microsecond instruction cycle with 12 MHz Crystal.
xii. One full duplex serial communication port.

BLOCK DIAGRAM OF INTEL 8051



- It consists of an 8-bit ALU, one 8- bit PSW(Program Status Register), A and B registers , one 16-bit Program counter , one 16-bit Data pointer register(DPTR),128 bytes of RAM and 4kB of ROM and four parallel I/O ports each of 8-bit width.

- 8051 has 8-bit ALU which can perform all the 8-bit arithmetic and logical operations in one machine cycle. The ALU is associated with two registers A & B

A and B Registers : The A and B registers are special function registers which hold the results of many arithmetic and logical operations of 8051.

The A register is also called the **Accumulator** and as it's name suggests, is used as a general register to accumulate the results of a large number of instructions. By default it is used for all mathematical operations and also data transfer operations between CPU and any external memory.

The B register is mainly used for multiplication and division operations along with A register.

MUL AB : DIV AB.

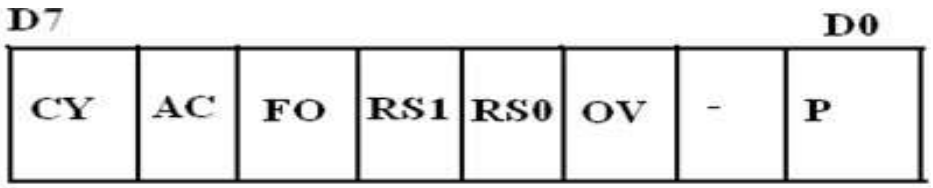
It has no other function other than as a location where data may be stored.

Program Counter(PC) : 8051 has a 16-bit program counter .The program counter always points to the address of the next instruction to be executed. After execution of one instruction the program counter is incremented to point to the address of the next instruction to be executed.

S**tack Pointer Register (SP) :** It is an 8-bit register which stores the address of the stack top. i.e the Stack Pointer is used to indicate where the next value to be removed from the stack should be taken from.

Data Pointer Register(DPTR) : It is a 16-bit register which is the only user-accessible. DPTR, as the name suggests, is used to point to data. It is used by a number of commands which allow the 8051 to access external memory.

Program Status Register (PSW) : The 8051 has a 8-bit PSW register which is alsoknown as Flag register.In the 8-bit register only 6-bits are used by 8051.The two unused bits are user definable bits.In the 6-bits four of them are conditional flags .They are Carry –CY,Auxiliary Carry-AC, Parity-P,and Overflow-OV .These flag bits indicate some conditions that resulted after an instruction was executed.



Program status word register

CY	PSW.7	Carry Flag
AC	PSW.6	Auxiliary Carry Flag
FO	PSW.5	Flag 0 available for general purpose .
RS1	PSW.4	Register Bank select bit 1
RS0	PSW.3	Register bank select bit 0
OV	PSW.2	Overflow flag
---	PSW.1	User difinable flag
P	PSW.0	Parity flag .set/cleared by hardware.

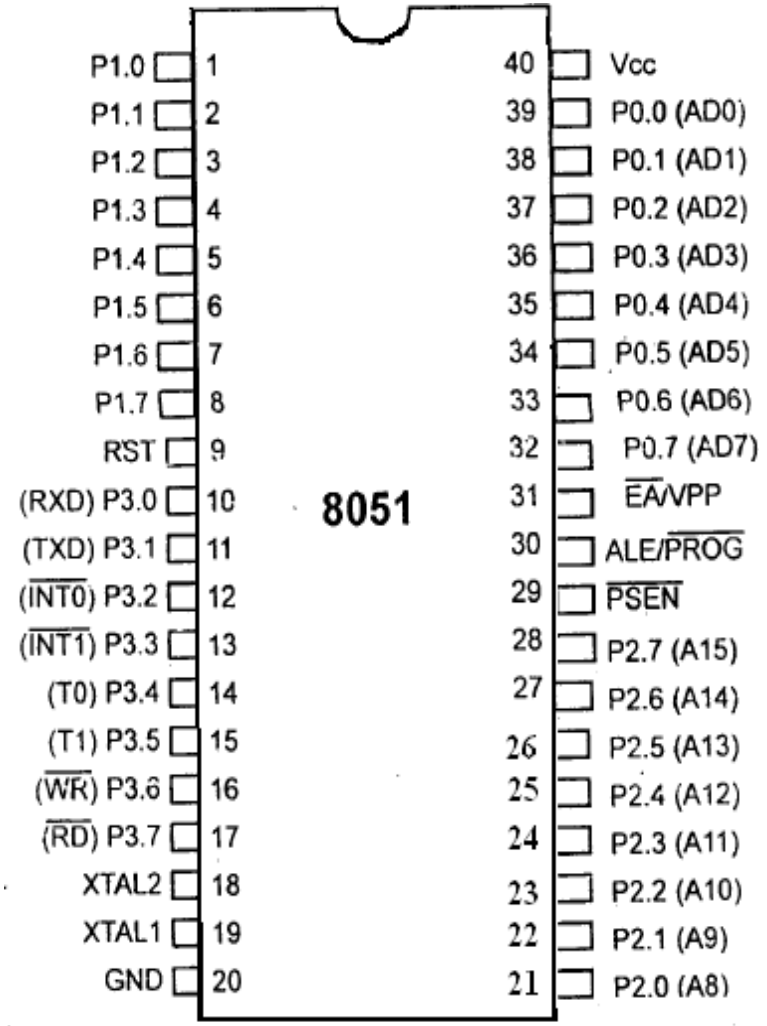
PIN Diagram of 8051 Microcontroller : The 8051 microcontroller is available as a 40 pin DIP chip and it works at +5 volts DC. Among the 40 pins , a total of 32 pins are allotted for the four parallel ports P0,P1,P2 and P3 i.e each port occupies 8-pins .The remaining pins are VCC, GND, XTAL1, XTAL2, RST, EA ,PSEN.

XTAL1,XTAL2: These two pins are connected to Quartz crystal oscillator which runs the on- chip oscillator. The quartz crystal oscillator is connected to the two pins along with a capacitor of 30pF as shown in the circuit. If we use a source other than the crystal oscillator, it will be connected to XTAL1 and XTAL2 is left unconnected.

VSS:It is Circuit ground. All the voltages are specified with respect to it.

VCC: It is for power supply.

Pin diagram of Intel 8051 Microcontroller



RST: The RESET pin is an input pin and it is an active high pin. When a high pulse is applied to this pin the microcontroller will reset and terminate all activities.

$\overline{EA}$(External Access): This pin is an active low pin.

$\overline{PSEN}$(Program Store Enable) : This is an output pin which is active low. **ALE (Address latch enable):** This is an output pin, which is active high**.** When connected to external memory , port 0 provides both address and data.

P0.0- P0.7(AD0-AD7) : The port 0 pins multiplexed with Address/data pins .If the microcontroller is accessing external memory these pins will act as address/data pins otherwise they are used for Port 0 pins.

P2.0- P2.7(A8-A15) : The port2 pins are multiplexed with the higher order address pins **.**When the microcontroller is accessing external memory these pins provide the higher order address byte otherwise they act as Port 2 pins.

P1.0- P1.7 : These 8-pins are dedicated for Port1 to perform input or output port operations.

P3.0- P3.7 : These 8-pins are meant for Port3 operations and also for some control operations like Read,Write,Timer0,Timer1 ,INT0,INT1 ,RxD and TxD

I/O Lines:

- **8051** Microcontrollers contain four ports: P0,P1,P2,P3.
- There are 32 I/O lines. All ports in 8051 are bi-directional and they are also multifunctional lines.

Alternate function of port pins are:

P1.0:T2(Timer/Counter 2 external input).

P1.1: T2EX(Timer/Counter2 capture/reload trigger)

P3.0: RXD(Serial Input port)

P3.1: TXD(Serial output port)

P3.2:INT0(External Interrupt)

P3.3:INT1(External Interrup

P3.4:T0(Timer/Counter 0 external input)

P3.5:T1(Timer/Counter 1 external input)

P3.6:WR(External data memory write strobe)

P3.7:RD(External data memory Read strobe)

8051 Interrupts:

- **Interrupt-** Facility provided in microprocessor using which attention of microprocessor may be drawn for some specific purpose.
- Microprocessor suspends its current job- saves the status.
- Microprocessor attends to the device/system/event causing interrupt- ISR is executed.
- Microprocessor goes back to suspended job and starts executing from the point where it was suspended.

Other issues in Interrupts

- Hardware/Software Interrupts
- Interrupt Priority
- Enabling / Disabling of Interrupts.
- Masking of Interrupts.
- Edge/Level Triggered Hardware Interrupts

- 8051 has five interrupt sources.
- Each interrupt can be programmed to two priority levels.

INT0 – External Request from P3.2pin

Timer 0 – Overflow from Timer 0 activates interrupt request flag TF0.

INT1 – External Request from P3.3

Timer1- Overflow from Timer1 activates interrupt request flag TF1

Serial Port – completion of transmission or reception of a serial frame activates the flags TI or RI.

The complete interrupt system may be disabled or enabled by storing 0 or 1 in EA bit of IE SFR

- IE.7. IE.7, - Enable Interrupts IE.7,

SETB - Disable Interrupts

CLR

- When Interrupt system is enabled i.e. IE.7=1, then each of the five interrupts can be enabled/disabled individually by making a specified bit in IE register as 1 or 0.

- Interrupt masking

bit = 0 – Interrupt is disabled i.e. masked bit = 1 – Interrupt is

IE Register – Interrupt Enable Register

7	6	5	4	3	2	1	0
EA	**X**	**X**	**ES**	**ET1**	**EX1**	**ET0**	**EX0**

- EX0- Enable/Disable – External Interrupt 0
- ET0- Enable/Disable – Timer 0 overflow
- EX1- Enable/Disable – External Interrupt 1
- ET1- Enable/Disable – Timer 1 overflow Interrupt
- ES -Enable/Disable – Serial port Interrupt
- EA - Enable/Disable – All interrupts

- There are two priority levels for interrupts in 8051- High and Low.
- Each interrupt can be programmed to a low or high priority level, by making bits of IP(Interrupt Priority) SFR as 0 or 1.

bit = 0 – Interrupt at Low priority level.

bit = 1 – Interrupt at high priority level.

Interrupt Priority Register

IP -

X	X	X	PS	PT1	PX1	PT0	PX0
7	6	5	4	3	2	1	0

PX0 – Priority – External Interrupt 0

PT0 – Priority – Timer 0 overflow

PX1 – Priority – External Interrupt 1

PT1 – Priority – Timer 1 overflow

PS - Priority – Serial port Interrupt

- A low priority interrupt can't interrupt a high priority interrupt but a high priority interrupt can interrupt low priority interrupt.

Source

- External Interrupt 0 Highest
- Timer 0 Overflow
- External Interrupt 1
- Timer 1 Overflow
- Serial Port Lowest

- When two or more interrupt requests come at the same time and all the interrupts at the same priority level the 8051 will select an interrupt for servicing based on above.
- All the interrupts are separately scanned during each machine cycle.
- An interrupt request will be serviced i.e. corresponding ISR will be executed unless .An interrupt of equal or higher level is already in progress. Current m/c cycle is not the final m/c cycle in the execution of instruction in progress i.e. no interrupt request will be responded to until the instruction in progress in completed.

Instruction Set

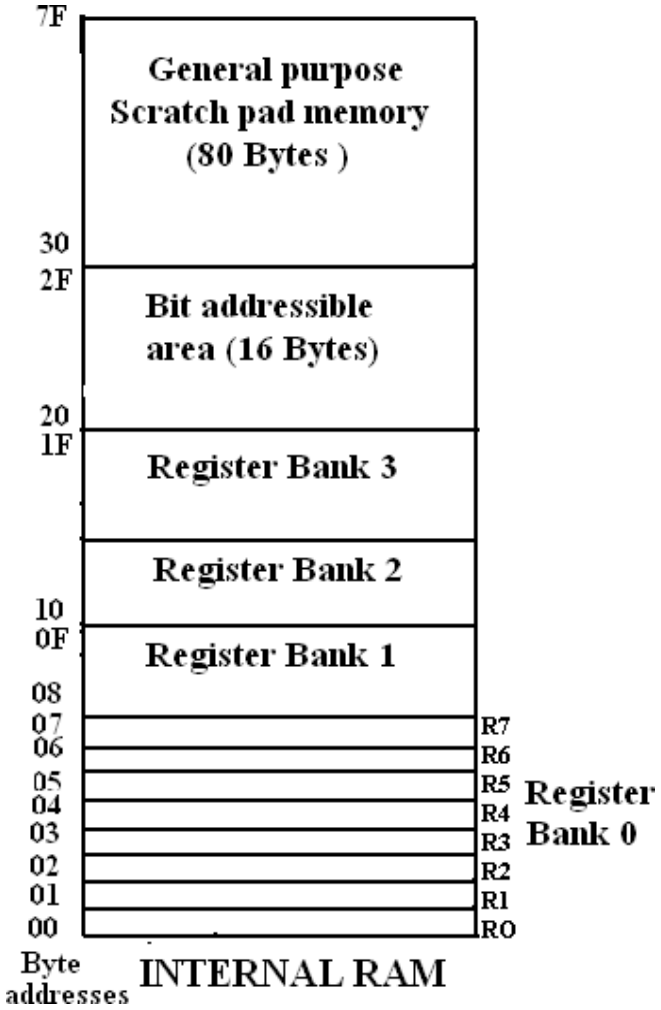
- The 8052 has 111 instructions: 49 single-byte,42 double-byte and 17 three-byte instruction.
- It is provided with multiplication and division instruction.
- It takes 1 microsecond for addition and four microsecond for multiplication or division.
- The instruction set of 8051 includes binary and BCD arithmetic operations, bit set/reset functions and logical functions.

Memory Organization of Intel 8051

The 8051 microcontroller has 128 bytes of Internal RAM and 4kB of on chip ROM .The RAM is also known as Data memory and the ROM is known as program memory. The program memory is also known as Code memory .In 8051 this memory is limited to 64K .Code memory may be found on-chip, as ROM or EPROM. It may also be stored completely off-chip in an external ROM or, more commonly, an external EPROM. The 8051 has only 128 bytes of Internal RAM but it supports 64kB of external RAM.

Internal RAM OF 8051 :

This Internal RAM is found on-chip on the 8051 .So it is the fastest RAM available, and it is also the most flexible in terms of reading, writing, and modifying it's contents. Internal RAM is volatile, so when the 8051 is reset this memory is cleared.



Internal ROM (On –chip ROM): The 8051 microcontroller has 4kB of on chip ROM but it can be extended up to 64kB.This ROM is also called program memory or code memory. The CODE segment is accessed using the program counter (PC) for opcode fetches and by DPTR for data.

ADDRESSING MODES OF 8051 :

There are various methods of denoting the data operands in the instruction. The 8051 microcontroller supports mainly 5 addressing modes. They are

1.Immediate addressing mode

2.Direct Addressing mode

3.Register addressing mode

4. Register Indirect addressing mode

5.Indexed addressing mode

Immediate addressing mode : The addressing mode in which the data operand is a constant and it is a part of the instruction itself is known as Immediate addressing mode.

Ex: MOV A , # 27 H: The data (constant) 27 is moved to the accumulator register

Direct addressing mode: The addressing mode in which the data operand is in the RAM location (00 -7FH) and the address of the data operand is given in the instruction is known as Direct addressing mode.

MOV R1, 42H : Move the contents of RAM location 42 into R1 register

Register addressing mode :The addressing mode in which the data operand to be manipulated lies in one of the registers is known as register addressing mode.

MOV A,R0 : Move the contents of the register R0 to the accumulator .

Register Indirect addressing mode : The addressing mode in which a register is used as a pointer to the data memory block is known as Register indirect addressing mode.

MOV A,@ R0 :Move the contents of RAM location whose address is in R0 into **A**

Indexed addressing mode : This addressing mode is used in accessing the data elements of lookup table entries located in program ROM space of 8051.

Ex : MOVC A,@ A+DPTR

The 16-bit register DPTR and register A are used to form the address of the data element stored in on-chip ROM. Here C denotes code .In this instruction the contents of A are added to the 16-bit DPTR register to form the 16-bit address of the data operand.