UNIT-I

Evolution of processors:

1974 – 4004 - 4bit Intel
1976 – 8008 - 8bit "
1978 – 8085 - 8bit "

} 1st generation Processors

1979 – 8086 20bit "
80386 32bit "
80486 32bit "
8051 16bit "
ARM 64bit "
cotex 64bit "

} 2nd generation processors

→ There were few drawbacks in the 1st generation processors given by
→ Less addressing capability
→ Less number of Registers
→ Less powerful ALU
→ low Speed of execution
→ These drawbacks are overcomed by 2nd genera-

-tion processors whose addressing capability is more, more no. of registers, more powerful to perform Arithmetic & logic operations & high Speed of execution.

Note:- processor is core of the embeded system

Micro processor:

→ In micro processor only small amount of memory is used when compared to the total memory that is why the name micro processor come into existence.

8086 µp:

→ It is a 20bit processor Out of which only 16 bits are used effectively for addressing capabi-lity.

→ The memory mapping of 8086 µp is given by

→ 20bit µp = 2^{20} = 1MB

→ effectively used 16 bits ⇒ 2^{16} = 64KB

$= 2^{10} \times 2^{6}$

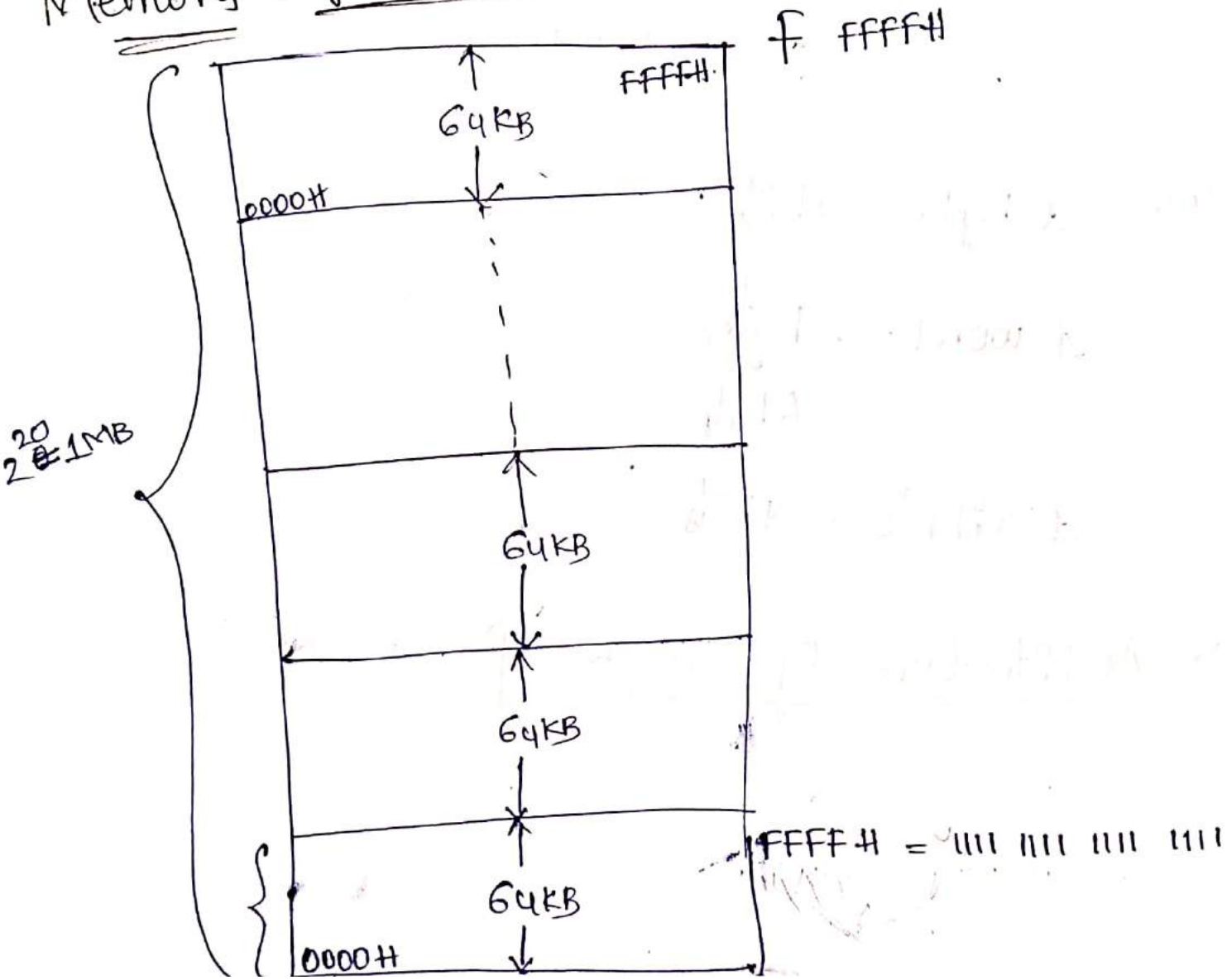
$\Rightarrow 1KB \times 64 = 64KB$

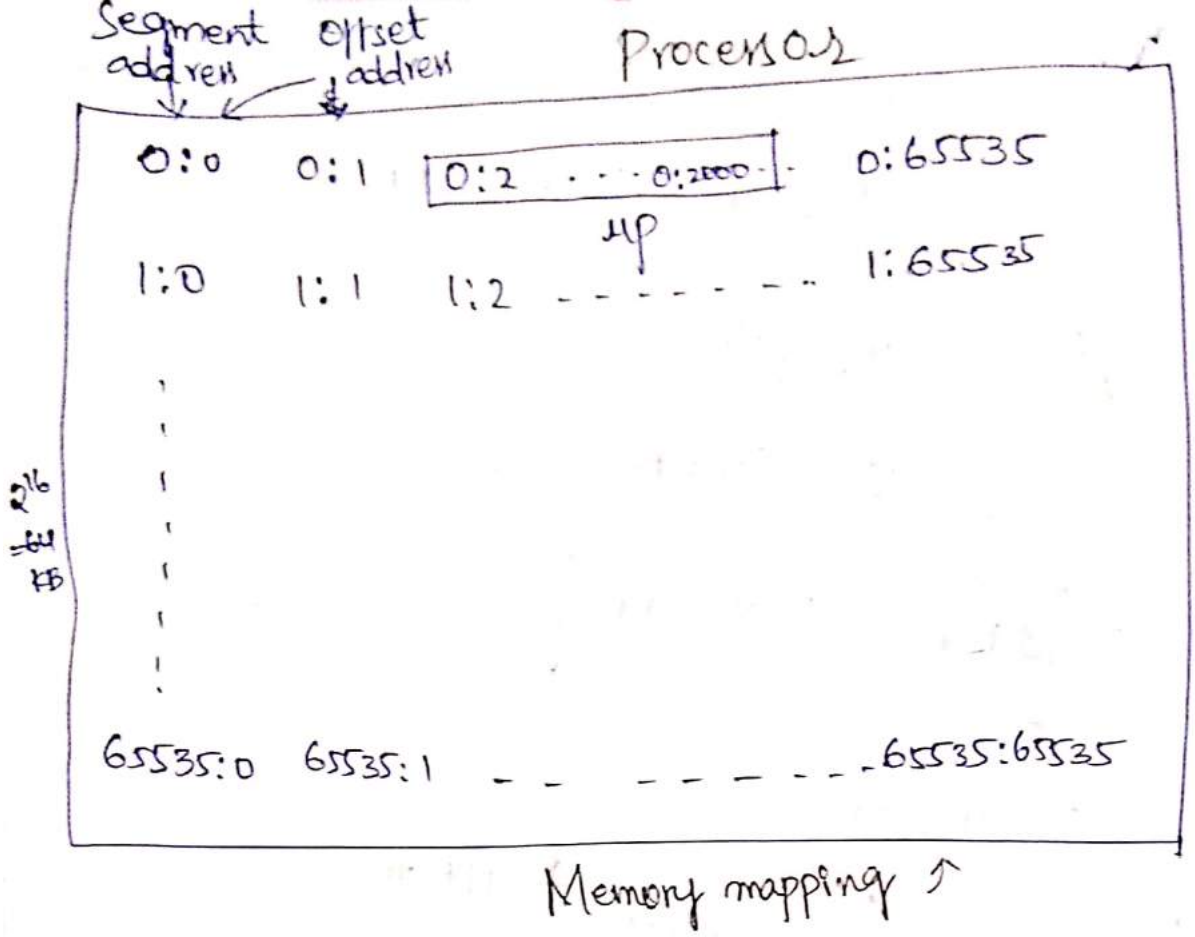
$2^{20} = 2^{4} \times 2^{16}$

$= 16 \times 64\,KB$

$\Rightarrow$ **1MB = 16 x 64 KB**

Memory Organisation:



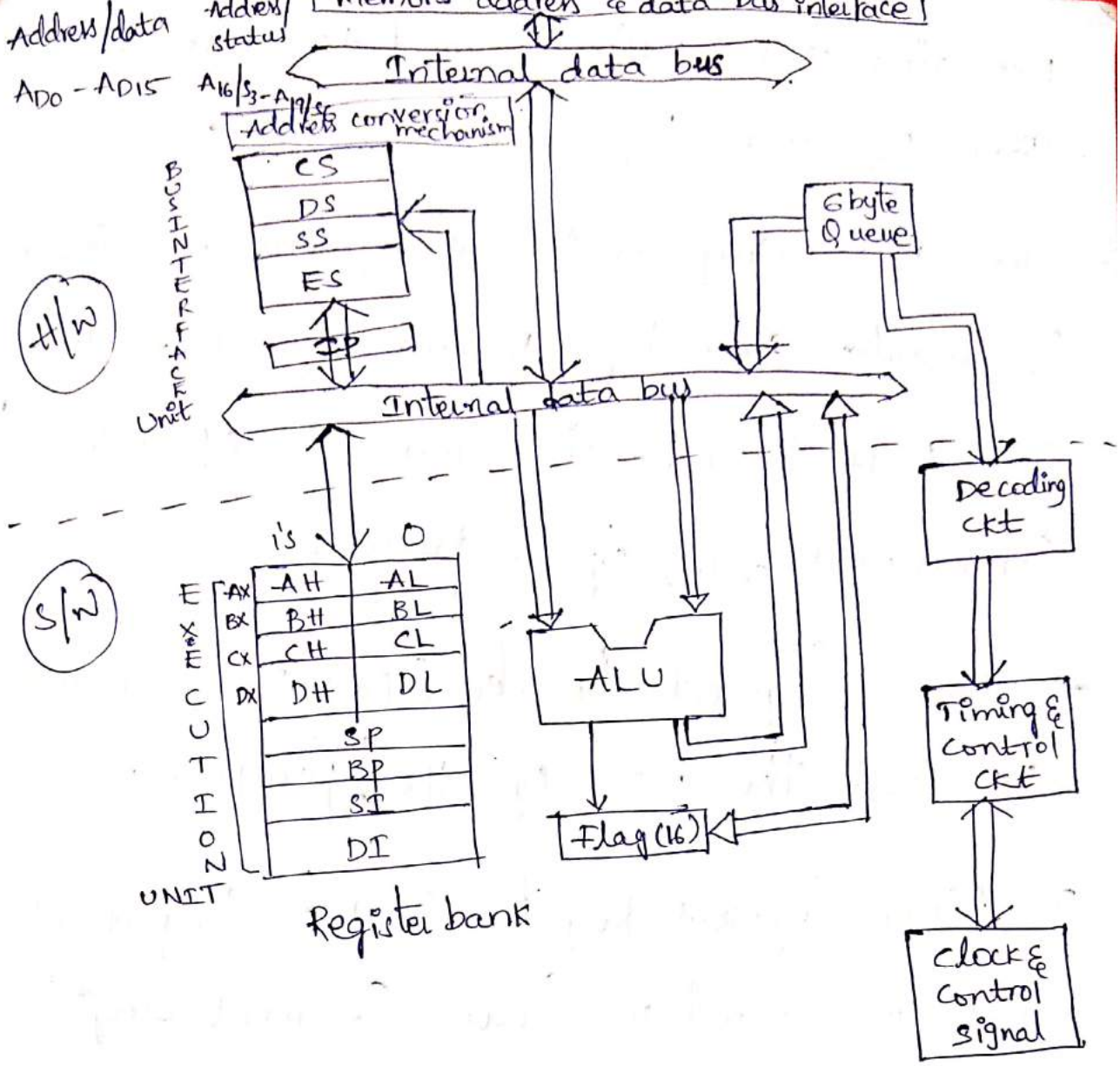


Note: 1 byte = 8 bit

1 word = 2 bytes
= 16 bit

1 nibble = 4 bit

→ Architecture of 8086 up



→ 8086 µp is divided into two unit given by

1) Bus interface unit (or) Hardware unit

2) Execution unit (or) Software unit

→ BIU is used to establish the connection between processors & peripherals by means of two buses Given by Address bus & data bus.

→ Execution unit (EU) is used to execute the given program by means of ALU.

→ The BIU comprises of 4 segment registers each of 16-bits given by :1, code segment register (CS) : It is used to store the OP-code (operational code) of the program.

2, Data Segment Register (DS) : It is used to store the data of the given program.

3, Stack Segment Register (SS) : Important data is stored in Stack Segment register & whenever CPU want to retrieve the data it can retrieve the data from Stack Segment.

4) Extra Segment Register (ES): when data Segment is fully filled with the data then extra data is stored in extra data Segment.

→ Instruction pointer (IP) is a 16 bit register which is used to fetch the starting address of the program.

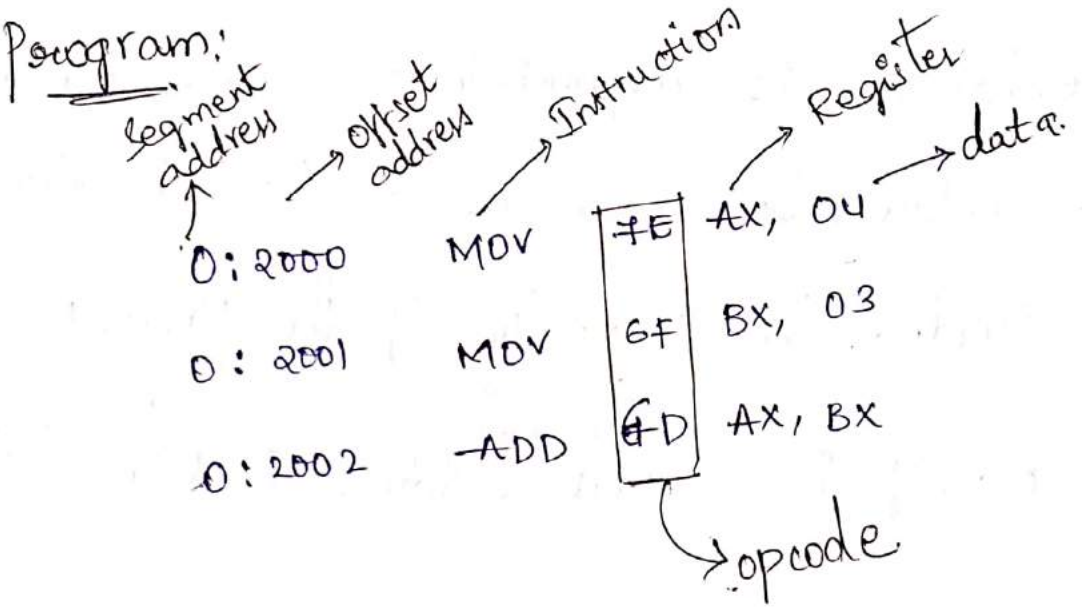
→ The decoding ckt is used to convert the Assembly level language to machine level language comprising of 0's & 1's (operational code).

→ The execution is performed by ALU

Note: Every instruction is a combination of 3 steps:

1, Fetching 2, Decoding 3, Execution.

Program:



→ In 8085 there is no 6 byte Queue & remaining architecture remains same but in 8086 the presence of 6 byte Queue makes multi tasking Operations.

Note:

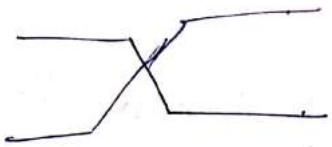
→ In 8085 instruction pointer (IP) remain idle until & unless decoding & execution is completed.

→ In 8086 IP doesnot remain idle but instead fetches the address data & puts into 6 byte queue.

→ In 8086 up architecture advance conversion mechanism is responsible for demultiplexing the address bus, data bus & status bus inorder to reduce the complexity of the processor.

→ 8086 up is a 20 bit ($A_0 - A_{19}$) address & 16 bit ($D_0 - D_{15}$) data.

~~$AD_0 - AD_{15}$~~ →



Multiplexed address data bus

Demultiplexing

— $A_0 - A_{15}$

— $D_0 - D_{15}$

$A16/S_3 - A19/S_6 \rightarrow$ 

Multiplexed Address & status bus

= $A_{16} - A_{19}$ (Address bus)

= $S_3 - S_6$ (status bus)

→ In execution unit there are various components given by

→ Decoder
→ Timing & control ckt
→ ALU
→ Flag
→ General purpose & Index register

Register Organisation Of 8086 µp:

→ It has general purpose & Special purpose registers which are 16 bit registers.

→ General purpose registers which are used for holding data, variables & intermediate results where as Special purpose registers are used for indirect addressing.

Register organisation of 8086

AX	AH	AL
BX	BH	BL
CX	CH	CL
DX	DH	DL

General purpose Register (16-bit)

SP
BP
SI
DI

Index & Pointer registers (16 bit)

Note: The Registers can be used either 16 bit (or) 8 bit

for example

AX is called as accumulator Since the results are Stored in accumulator.

→ All the Operations are performed from right to left.

→ SI & DI are called as index registers, SI – Source index which is used at i/p. DI – Destination index which is used at o/p.

→ SP → Stack Pointer which is used to point the top of the Stack.

→ BP- Base pointer which is used to point the Base address

ALU:

It Performs Various arithemetic operations like addition, subtraction, multiplication & division.
It Performs various logical operations like OR, AND, NAND, NOR, EX-OR, EX-NOR, NOT etc.

→ Timing & control ckt:

→ 8086 µp is available in plastic package (or) Dual in package (DIP) & is a 40 pin IC. out of that 40 pins two pins are alloted for crystall oscillators for providing clock frequency to the processor.

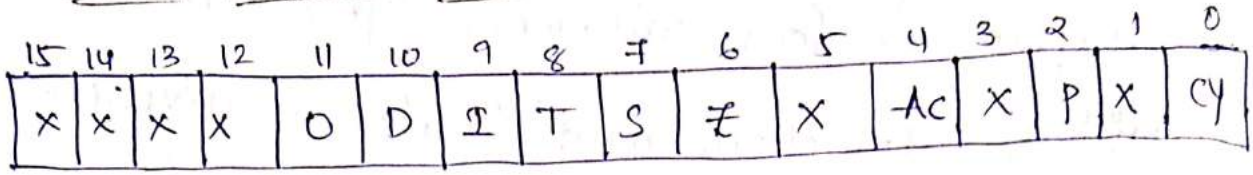
→ µp is available in 5MHz, 8MHz, 10MHz clock frequencies.

→ Flag register:

→ Flag register is of 16bit register & flag register is used to show the results of the Arithmetic operations being Performed.

→ It has brought two types of flags given by 1, Status flags 2, control flags.

Flag register of 8086 µp:



Status flags

CY – carry flag
P – Parity flag
Ac – Auxilary Carry flag
Z – Zero flag
S – Sign flag
O – overflow flag

control flags

→ T – Trap flag
→ I – Interrupt flag
→ D – Directional flag

Carry flag:

→ This flag is said when there is a carry generated from the 7th bit (or) MSB (or) otherwise Reset.

7th bit (MSB) 0th bit (LSB)

```
  1000  0100
  0001  0100
 ------------
  1001  1000
 ------------
```

CY = 0 (Reset)

lags given

```
  1  <-  1000  0100
      1   1001  0100
      ---------------
      0001  1000
      ---------------
```

CY = 1 (set)

2	1	0
P	X	CY

flags

flag

ypt flag

nal flag

y generated

set.

Zero flag:

If the result of the Computation is zero then zero flag is 'Set' otherwise Reset.

Parity flag:

If in the results of the Computation there are even no. of one's then parity flag is Set otherwise Reset.

$P \neq 1$

Sign flag: If the result of the computation has MSB = 1 then it is a negative number & if MSB = 0 then it is a +ve number

MSB = 1 (−ve)

MSB = 0 (+ve)

Auxilary Carry flag: If the Carry is generated from the nibble (from 3rd bit to 4th bit) then auxilary Carry is set otherwise reset.

ex:

4th bit	3rd bit
1000	1000
0000	1100
1001	0100

AC = 1 (set)

Overflow flag:

This flag is set if the result of the magnitude is greater than 255 then overflow flag is set otherwise reset.

> 255 OV = set = 1.

Control flags:

→ control flags are also called as Machine flags which are executed by the machine.

Trap flag:-

This flag is set when there is a Single step debugging Otherwise reset.

T = 1 = set
single step debugging

Interrupt flag:

This flag is set when Maskable Interrupts are recognized Otherwise reset

I = 1 = set Maskable Interrupts Recognized.

There are two types of interrupts 1, Maskable Interrupts (least Priority) 2, non maskable interrupt

(highest priority).

Directional flag: This flag is Set when auto increment is done & this flag is reset when auto decrement is done.

D = 1 = set
auto increment

D = 0 = Reset
auto decrement.

→ Interrupts of 8086 up:

An interrupt is a Sudden halt in a given program.

→ In 8086 up there are a total of 256 interrupts & each interrupt is executed by going to it respective capital ISR (Interrupt Service Routine) address

→ The ISR address of each interrupt is of 4-bit & thus the total 256 interrupts consume $256 \times 4 = 1024 = 2^{10} = 1KB$

→ Out of 256 interrupts 8 interrupts are hardware interrupts & remaining interrupts are software interrupts.

→ Hardware interrupts are the interrupts generated by the peripherals where as Software interrupts are generated by the programmer himself for debugging Purpose.

→ NOP (No operation) is an example of software interrupt.

→ The interrupts are also classified into two types based on their priorities.

(1) Maskable interrupts which has least priority & (2) Non Maskable interrupts which has highest Priority.

Note:

Trap is a example of NMI (non Maskable Interrupt).

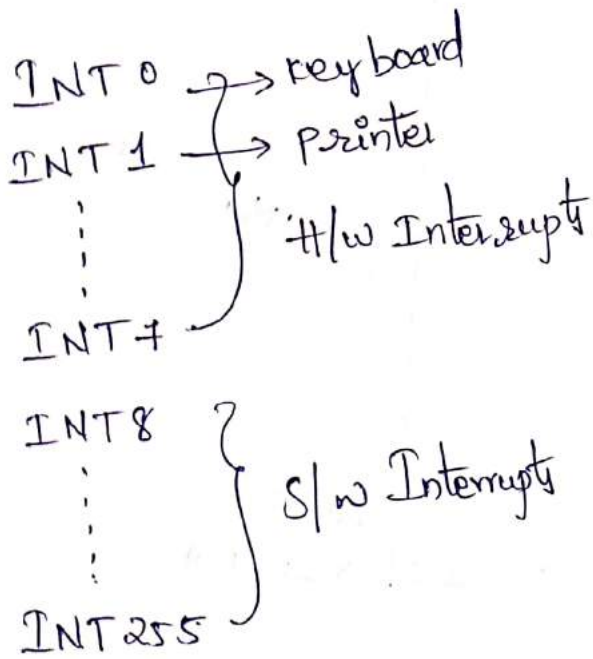
→ The interrupts are executed based on their priorities by going to its respective ISR address

→ The program which is being generated before the execution of interrupts is stored in PSW (Program Status Word). And next byte of the

instruction that has to be executed after the completion of interrupt is stored in 'PC' (program counter).

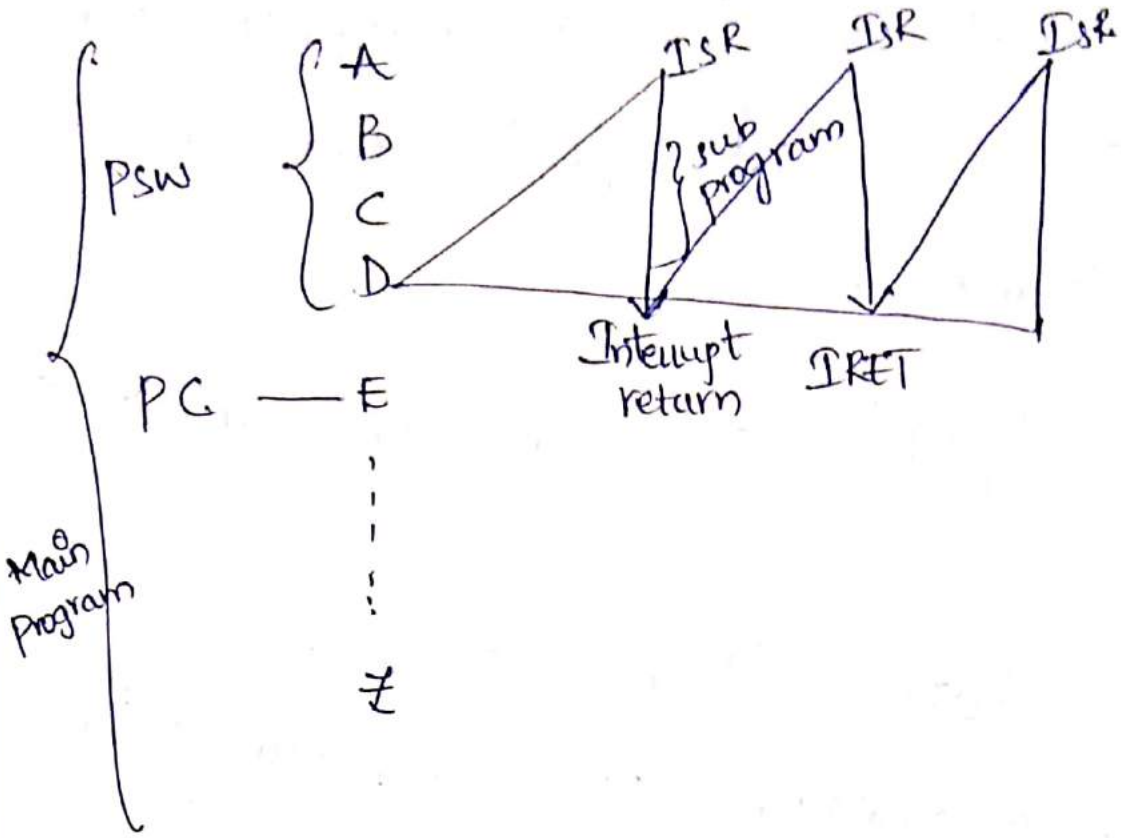
→ Both these 'PSW' & 'PC' are stored in stack segment.

→ Important data will be stored in stack segment & whenever CPU wants to retrieve the data it can retrieve the data from the stack segment.

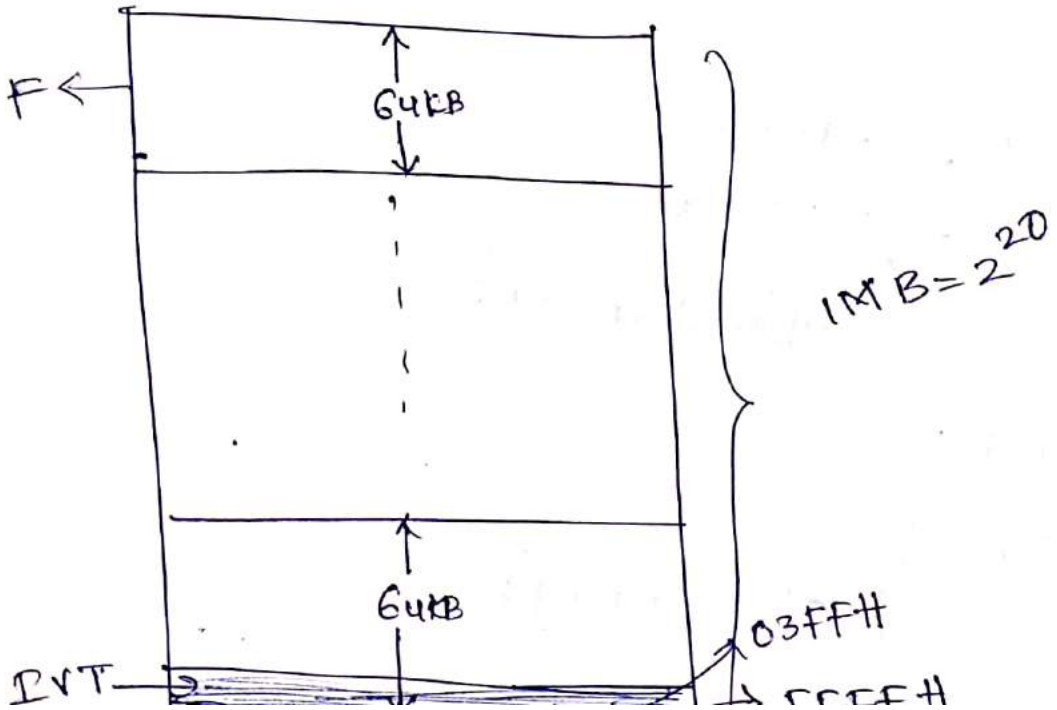


IVR address = $256 \times 4 = 1024$

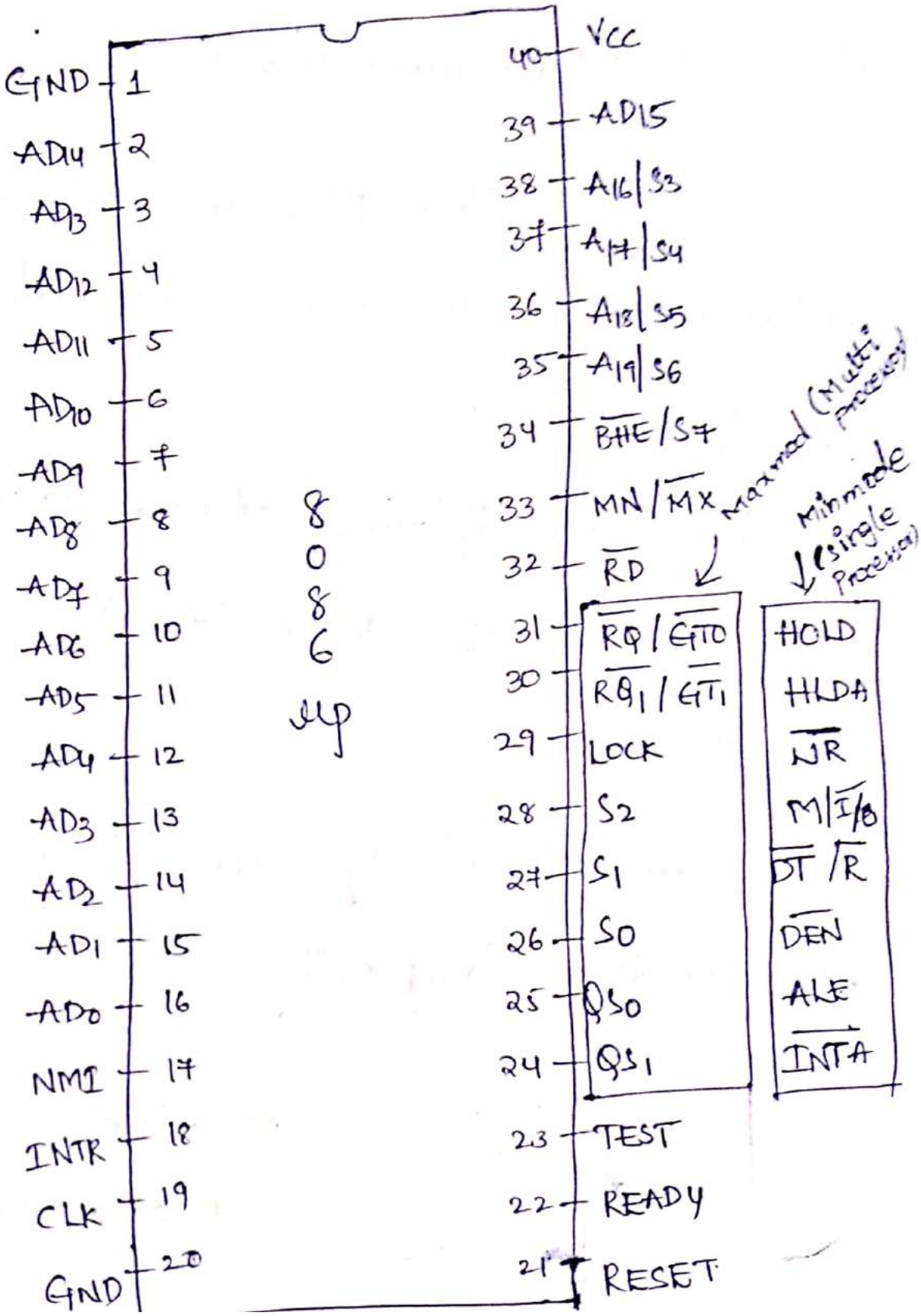
$= 2^{10} = 1KB$



PSW & PC = SS



Signal description of microprocessor 8086 :-

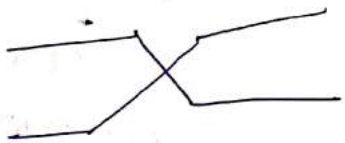


→ The pins of 8086 µp are broadly categorized into '3' categories. 1, common pins 2, Minimum mode pins & 3, Maximum mode pins.

1, Common pins: This pins are common to both minimum mode pins & maximum mode pins given by " $AD_0 - AD_{15}$.

→ It is a multiplexed Address Data bus Comprising of 16-bit address ($A_0 - A_{15}$) & 16-bit data ($D_0 - D_{15}$).

Note: The Concept of multiplexed bus is used to reduce Complexity



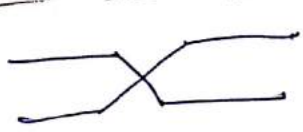
Multiplexed Address / Data

$AD_0 - AD_{15}$

= $A_0 - A_{15}$

= $D_0 - D_{15}$

2, $A_{16}/S_3 - A_{19}/S_6$



Multiplexed Address / Status.

= $A_{16} - A_{19}$

= $S_3 - S_6$

S_4	S_3	Indication
0	0	Alternate data / extra data
0	1	stack data
1	0	code data
1	1	Data

3, GND & Vcc:

There are two ground pins & external voltage of 5v is applied to run IC.

4) NMI: It is a non-maskable level triggered interrupt which has got highest priority

5, INTR: (Interrupt request): this pin is used to access the level triggered maskable interrupts which has least priority.

6, clock: crystal oscillators are used to provide the clock frequency to the processor in the range of 2MHz, 5MHz & 10MHz &

BHE	A0	Indication
0	0	whole word
0	1	upper byte
1	0	lower byte
1	1	None

9, $\overline{RD}$: If this pin is active low reading operation is performed by CPU from the peripherals.

→ reading operation is similar to recieve operation.

10, Test pin: This pin is actively used to test whether the processor is working properly or not by introducing wait states

11, READY: This pin is used to indicate the Processor to make the execution ready for it Operation

12, RESET: This pin makes the processor to starte it operation from the begining.

Note: The time period of execution of instruction in up is measured in machine cycles.

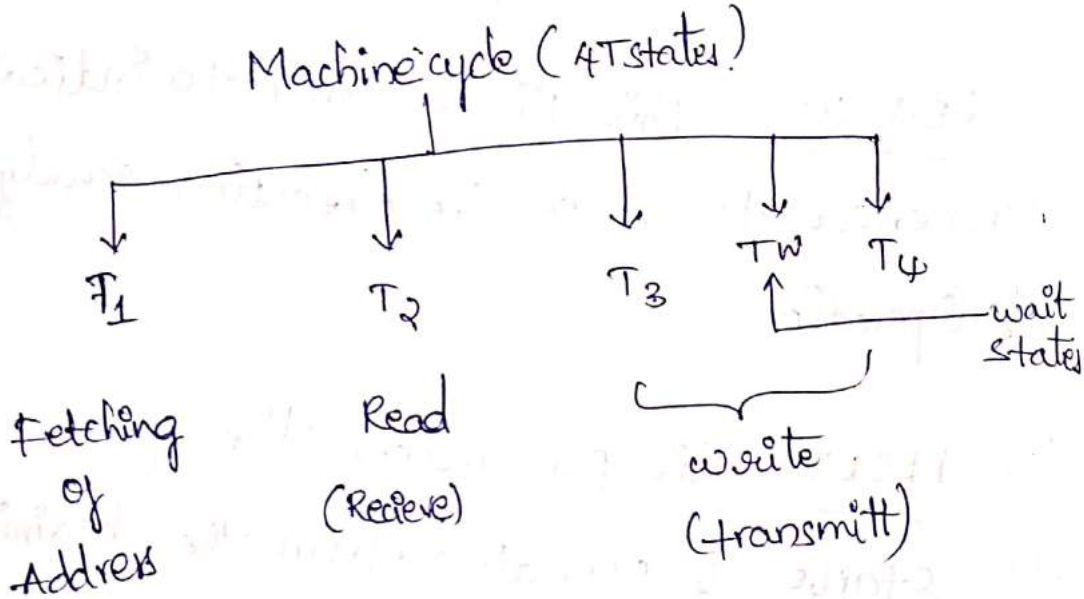
1 Machine Cycle = 4 Tstates

during T_1 Tstate fetching Operation is done wrt addres of the program.

During T_2 Tstate read operation (recieve) is performed.

During T_3 & T_4 Tstates write operation (transmitt) is performed.

→ For the devices which are slow to respond to the processor WAIT States are introduced during write Operation.



(ii) Minimum Mode Pins:

1) HOLD: when an interrupt is generated the processor will hold its current operation

2) HLDA: (Hold Acknowledgement) After holding the operation cpu acknowledges the interrupt

& hardoves the address bus & data bus to the interrupt.

3) M/I/O : (Memory / Input output devices).

If M/$\overline{I/O}$ = 0 → Input output devices are communicated with cpu.

If M/$\overline{I/O}$ = 1 → Memory devices are Communicated with cpu/processor.

4) $\overline{DEN}$ & DT/$\overline{R}$:-

↓ Data enable ↓ Data transmitt / Recieve.

→ If $\overline{DEN}$ = 0 (data enabled)

DT/$\overline{R}$ = 0 (Read (or) recieve)

DT/$\overline{R}$ = 1 (write (or) Transmitt)

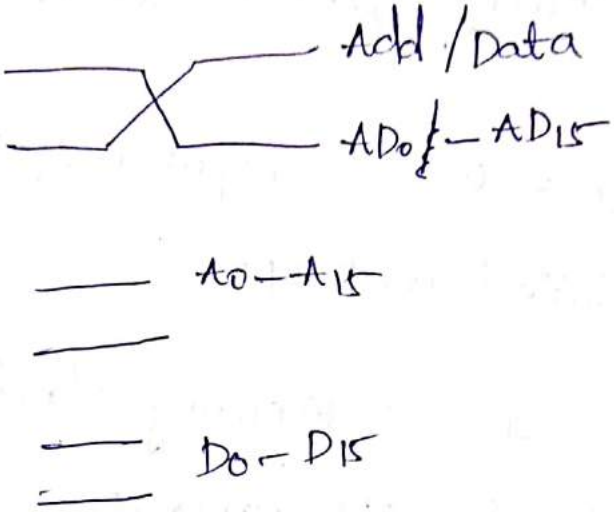
→ If $\overline{DEN}$ = 1 (Data disabled)

Note:

→ For data transmission $\overline{DEN}$ must be always '0'.

5, ALE: (Address latch Enable):

This is used to demultiplex the buses into various buses.



6, $\overline{INTA}$: (Interrupt Acknowledgement):

when maskable interrupts are generated with least priority interrupt acknowldgement is given by means of $\overline{INTA}$

Note: HOLD is for Non maskable Interrupts (NMI) & $\overline{INTA}$ is for MI.

7, $\overline{WR}$: If $\overline{WR}=0$ write Operation is performed from cpu to Peripherals. It is Similar to transmitt Operation.

(iii) Maximum mode pins:

1)

QS_1	QS_0	Indication
0	0	No operation
0	1	first byte from queue
1	0	Empty Queue.
1	1	Subsequent byte from queue

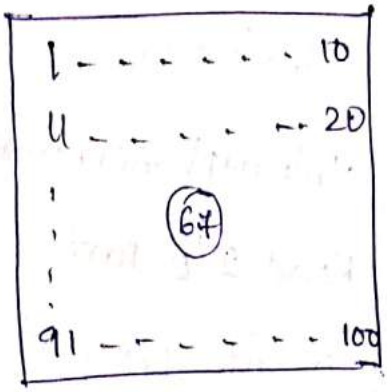
→ QS_0 & QS_1 gives information about Queue Status which is used for Multitasking operations.

$\overline{S_2}$	$\overline{S_1}$	$\overline{S_0}$	Indication.
0	0	0	Interrupt Acknowledge
0	0	1	Read I/O part
0	1	0	write I/O part
0	1	1	Halt
1	0	0	Code access
1	0	1	Read memory
1	1	0	write memory
1	1	1	Passive.

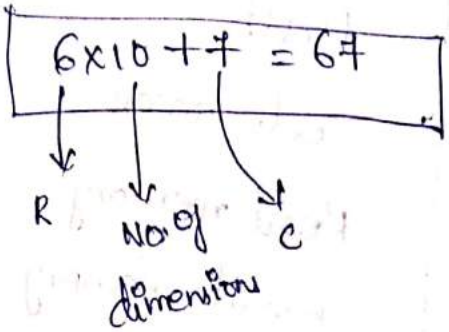
Lock: By means of this pin the current program that is being written is locked (or) saved.

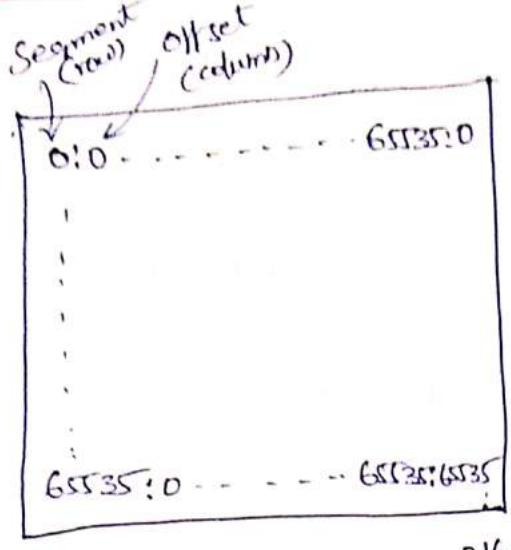
$\overline{RQ_0}/\overline{GT_0}$ & $\overline{RQ_1}/\overline{GT_1}$:- These pins are called as request & grant pins. & they are similar to HOLD & HOLD Acknowldgement pins in minimum mode in mode of Operation.

Calculation of physical address;



10X10=100





$2^{16} \times 2^{16}$

Segment address $\times 16_d$ + Offset address = PA (or) EA

(or)

Segment address $\times 10_H$ + Offset address = PA (or) EA

$\because 16 \mid 16$ (1 — 0)

$16_d = 10_H$

Calculate physical address (or) effective address if Segment address is 1005H & offset address is 5555H

Segment add 1005H

Offset add 5555H

PA (or) EA = 155A5

Segment add →	0001	0000	0000	0101
Segment add $\times 16_d$ →	0001	0000	0000	0101 0000
Offset add →	0101	0101	0101	0101
	0001	0101	0101	1010 0101
	1	5	5	A 5

Note: 1, Multiplying with 16 is equal to shifting the whole bit stream into left side by 4 bit.

2, when the bits are shifted left wards vacant position is created in LSB bit which is appended by zero.

calculate PA if the Segment add 4305 H and offset address is 945CH

Segment add = 4305 H

offset add = 945CH

PA = ?

Segment add = 0100 0101 0000 0101

segment add + 16d = 0100 0101 0000 0101 0000

offset add = 1001 1010 0101 1100

0100 1100 1010 1010 1100

4 C A A C

Addressing

The way

two oper

various a

address, da

→ According

modes are

addressing

(i) Sequentia

→ In thi

Sequenti

given b

data i

ex:

Q, Direct

In this

Specifi

Addressing modes:

The way the communication is performed b/n any two operands is known as addressing. There are various addressing modes that takes place register, address, data etc.

→ According to the flow of instructions the addressing modes are categorized into Sequential flow addressing modes & control flow addressing modes

(i) Sequential flow addressing modes:

→ In this addressing mode operations are executed Sequentially & they are divided into various types given by,

1, immediate addressing mode. The data is immediately moved to this register.

ex: MOV AX, 05H

05H is the immediate data.

2, Direct addressing mode:

In this mode the data is moved from Specified address to register.

ex: MOV AX, (5000H) → address 5000H, data 04H

AX = 04H

The physical address for this mode is 10H × DS + 5000H

3, Register addressing mode: In this addressing mode the data is moved from one register to another register

MOV AX, BX

4, Register indirect addressing mode: In this mode the register is acting as a pointer to some address & from that address data is moved to register ex: MOV AX, [BX]

Physical address is given by 10H × DS + (BX)
↳ acting as pointer

5, Indexed addressing mode: In this mode source index SI is pointed to some address & from that address & the data is moved to register

ex; MOV AX, [SI] ⇒ SI → 60004H [05H]
AX = 05H

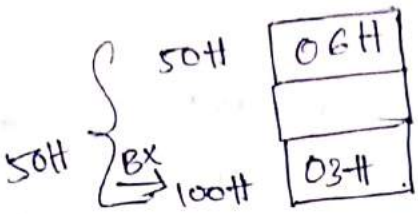
Physical address is given by 10H × DS + [SI]

6, Register Relative addressing mode:
The data is available at an address which is formed by adding register pointer address & that

address, the data is moved from that address to register

ex: MOV AX, 50H [BX]

physical address is given by 10H x DS + 50H + [BX]

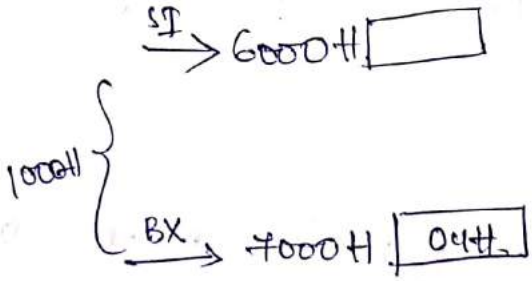


7) Based indexed addressing mode:

The effective address is formed by adding the contents of register & Source index.

MOV AX, [SI] [BX]

PA = 10H x DS + [SI]



8) Relative based Indexed addressing mode:-

The effective address is formed by adding offset address, register pointer address & Sourced index address.

ex: MOV AX, 50H [BX] [SI]

$PA = 10H \times DS + 50H + [BX] + [SI]$

50H [?]

AX = 04H

BX → 60H [?]

SI → 70H [04H]

(ii) Control Flow Addressing modes:

In this addressing mode the sequence of operations is not performed sequentially. It is divided into two types given by

1) Inter Segment addressing mode: when addressing takes place b/n one segment register to another segment register it is called Intersegment addressing mode.

2) Intra Segment addressing mode: when addressing takes place within the segment it is called intra Segment addressing mode.

Note:

Both Inter & Intra Segment Support direct & indirect addressing mode.

Assembler directives:

Microprocessor related programs can be executed in two Software environments given by T-ASM (Turbo Assembler) & M-ASM (Micro Assembler).

→ Assembler is used to convert assembly level programming (ALP) which is a high level language to machine language (or) low level language for its compatibility.

→ Assembler directives are the directions given to the assembler to perform various Operations

→ There are few assembler directives are given by

→ Data Segment

```
Num1  db  04H
Num2  db  06H
Res   db  Dup(1)
```

Data ends

```
ASSUME CS: code, DS: data
code segment
START:
        MOV AX, data   } Initializing data in data segment in code segment
        MOV DS, AX
code ends
End start
```

1) db: (define byte) (8-bits)

2) dw (define word): (16-bits)

3) DT (define ten bytes): (80-bits)

4) DQ (define Quad word): (4 words/64-bits)

5) ASSUME
CS is assumed as codesegment
DS is assumed as data segment

(6) ENDS (End of segment)

(7) START: Start the program.

(8) end START: ending of the program

(9) DUP(?): Duplicate memory to store the result

(10) EQU: equate an array of numbers.

(11) ORG: Originate to that address
(or) go to that address

12) PROC: It is a Sub program in a given main program

13) ENDP: End of Procedure.

14) GLOBAL: Can be accessed in any Part of the Program

15) LOCAL: Pertaining to that Sub program.

16) PTR : Pointer

Instruction Set of 8086

8086 µp has got 'n' no. of instructions given by 1, data-transfer / copy Instructions:

- All the transfer, move, copy, exchange, swap instructions come under this category

ex: MOV AX, 05H

MOV AX, BX

XCHG AX, BX

2) Arithmetic & logic instructions:

- Arithmetic operations like ADD, SUB, MUL, DIV & logical operations like AND, OR, NAND, NOR, EXOR, EX-NOR, NOT & ~~NOR~~ come under this ~~instructions~~ Category.

3) Branch instruction:

* There are three types of jumps: 1, short jump:
If there is a jump within 256 address locations then it is called short jump

2, Absolute jump: If there is a jump between 64K address locations

3, Long Jump:- If there is a jump within 64K address location.

Note:
The same implies for call instruction also.

4) Loop Instructions:-

Instructions like loop, loop nz, loop z comes under this category

loop not equal to zero — loop equal to zero

5, Machine Control Instructions:

These instructions are being executed by the processor. Hence they are called machine controlled instructions

NOP, HALT (Halt), WAIT, ESC, LOCK

6, Flag manipulation instructions:

1, CLC: (clear carry flag)
2, CMC: (complement carry flag)
3, STC: (set carry flag)
4, CLD: (clear directional flag)
5, CLI: (clear interrupt flag)
6, STI: (set Interrupt flag)

7, String Instructions:-

→ All the instructions which involved String manipulations come under this category

→ A String is defined as a collection of characters.

→ Each character is recognized by its ASCII (American Standard code for International Interchange).

ex: "HELLO", "MADAM"

→ each letter in String is called character

→ SCAN (scan String byte)
CMPS (compare String byte)

LODS (Load String byte)

STOS (Store String byte)

Note:

Compare Instruction involves three Steps.

1) Subtraction

2) checking Zero flag

3) Giving the result

8) Rotate & shift Instruction:

These instructions are used either to rotate (or) shift the contents in the register leftwards rightwards depending upon the count.

→ They are divided into various types given by

1) SHL /SAL (Shift logical /Arithemetic Left):

This instruction Shifts the whole Stream of bits leftwards by onebit for count = 1

→ MSB is moved to carry flag & LSB is appended with '0'

Ex: count=1

Bit position CF 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

Number 1 1 0 1 1

1. 1 0 1 0

appended with zero

MSB = CF

LSB = 0

Note:

After 16 counts we will get all zero's.

2, SHR (shift logical Right):

This instruction shifts the whole stream of bits rightwards by one bit for count=1

→ LSB is moved to carry flag & MSB is appended with

Ex: count=1

Bit position 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0 CF

Number 1 1 0 1 ..... 1 → 1

→ 0

appended with zero

Note: After 16 count we will get all zeros

3, SAR (Shift Arithmetic Right):

This instruction shifts the whole stream of bits rightwards by 1 bit position for count=1.

→ LSB is moved to carry flag & MSB is appended with '1'

LSB=CF
MSB=1

Bit position	15 14 13 12	11 10 9 8	7 6 5 4	3 2 1 0	CF
Number	1 1 0 1	- - - -	- - - -	- - - -	1

↳ 1 appended with 1.

Note: After 16 count we will get all one's.

1, ROR (Rotate Right without carry):

→ In this Instruction rotates whole stream of bits rightwards by one bit position for count=1

→ LSB is moved to carry flag & the same LSB bit is moved to MSB

Bit position	15 14 13 12	11 10 9 8	7 6 5 4	3 2 1 0	CF
Number	1 1 0 1	- - - -	- - - -	- - - -	1

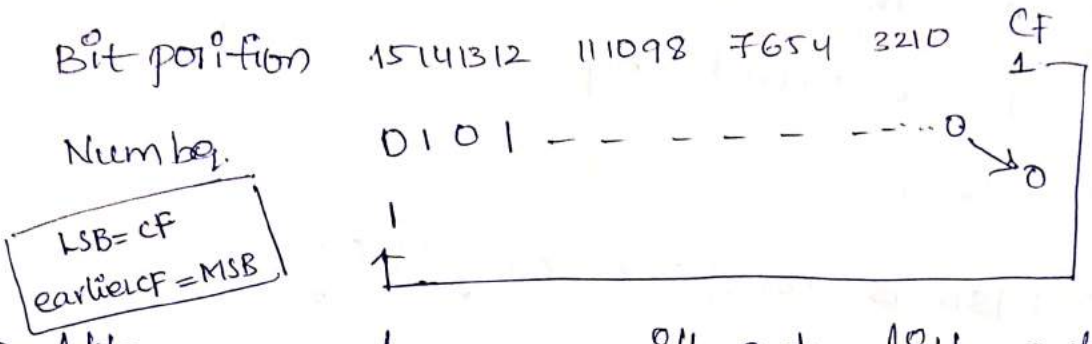
LSB=CF
LSB=MSB

Note: After 16 counts we will get the same number.

2, RCR (Rotate right through carry):

→ This instruction rotates the whole stream of bits rightwards by one bit position for count = 1.

→ LSB is moved to carry flag & earlier carry flag is moved to MSB.

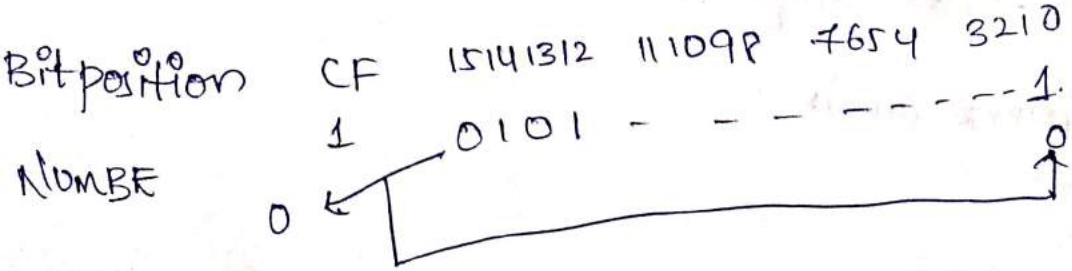


→ After 16 counts we will get different number.

3, ROL (Rotate left without carry):

→ This instruction rotates the whole stream of bits leftwards by one bit position for count = 1.

→ MSB is moved to carry flag & same MSB is moved to LSB



MSB = CF

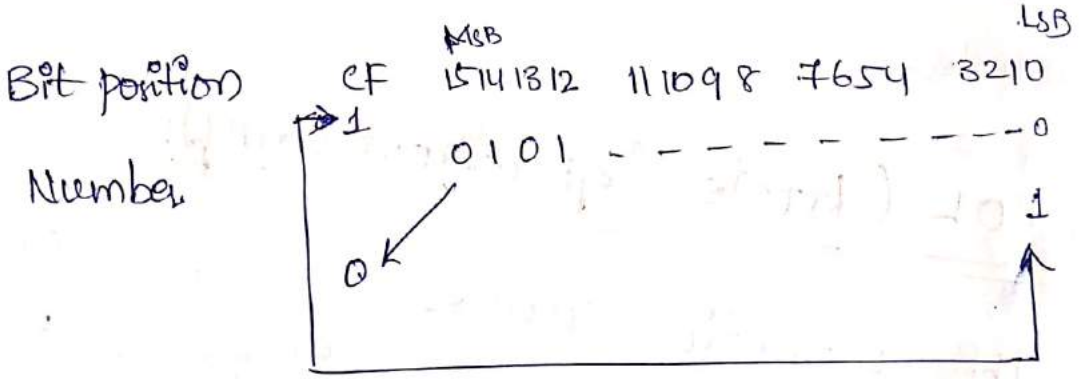
MSB = LSB

Note: After 16 counts we get the same number.

4) RCL: (Rotate Left through Carry):

→ This instruction rotates the whole stream of bits rightwards by one bit position for Count = 1.

→ MSB is moved to carry flag & earlier carry is moved to LSB.



MSB = CF

Earlier CF = LSB

Note: After compl 16 counts we will get different number.

DAA (Decimal Adjust Accumulator):

This instruction converts the unpacked BCD numbers to packed BCD number.

Note: BCD number is in the range of 0-9

Steps:

1) To the lower nibble (B) Add '6'.
2) TO the upper nibble (E) Add '6' along with auxiliary carry flag.

Result:

EB is converted to 51

Note: '6' is added to unpacked BCD since it is least number to get first BCD number

EB – unpacked BCD

B – 1011

Add '6' – 0110

AC=1	0001	→ 1

(1)

F9 – 1110

Add6 – 0110

CF=1 0101 → 5

CF=1 | 51 → packed BCD

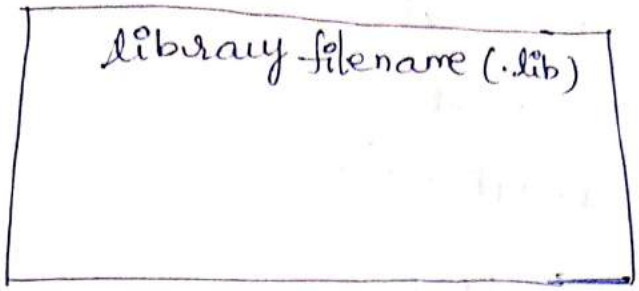
Macro Assembler programming:

There are various steps involved in coriting a program in Macro Assembler (MASM) by

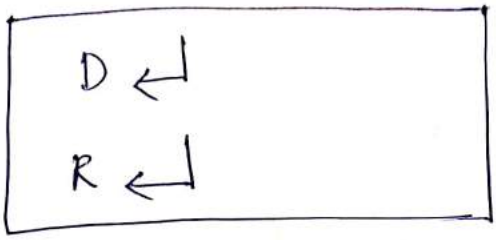
1) Entering a program

Enter file name: (.Prj)
Press any key to Continue

3) linking a program



4) Debug & Run



→ write an ALP to find the sum of even & odd numbers from a given series of numbers

```
ASSUME  CS:code  ds:data
→Data Segment
list dw 2357H, 0A52H, C322H, 4242H
Count equ 04 H
→Data ends
Code Segment
START: XOR BX, BX ↵
       XOR DX, DX ↵
```

```
        MOV AX, data
        MOV ds, AX

        MOV CL, count
        MOV SI, offset list
Again:  MOV AX, [SI]
        ROR AX, 01
        JC odd
        INC BX
        Jmp Next
Odd:    INC DX
Next:   Add SI, 02
        Dec CL
        JNZ again

        MOV AH, 4CH
        INT 21H
Cod ends
end start
```

1) ASSUME → Assembler directive

2) dw → define word

3) equ → equate an array of number.

4) perform XOR operation to clear the previous content

5) Initialize data in data Segment in code Segment

6) SI is pointed to offset address list

7) The content of SI is moved to AX

8) Rotate right without carry the contents of Ax ~~with~~ by count 1.

9) If Carry is generated treat it as odd number & increment Dx register

10) If Carry is not generated treat it as even number & increment BX

11) Get the 2nd number

12) Decrement CL

13) If counter is not equal to zero jump to the label again & Repeat the same process till the count is zero.

14) Return to DOS prompt

15) Stop the program

16) End of the code Segment

17) code ends

→ Write an ALP to find sum of +ve & -ve Numbers

Note: For the above program Replace

ROR AX, 01 with SHL AX, 01

JC odd with JC NEG

Change label ODD with NEG

SHL (shift logical left) the contents of Accumulator by count 1

4322H

CF MSB LSB

0100 0011 0010 0010

0 ← appended with 0

Cf = 0 = +ve

Cf = 1 = -ve

3) Write ascendin

ASSUM

Data

List d

Count

Data

code

START

Again

Again

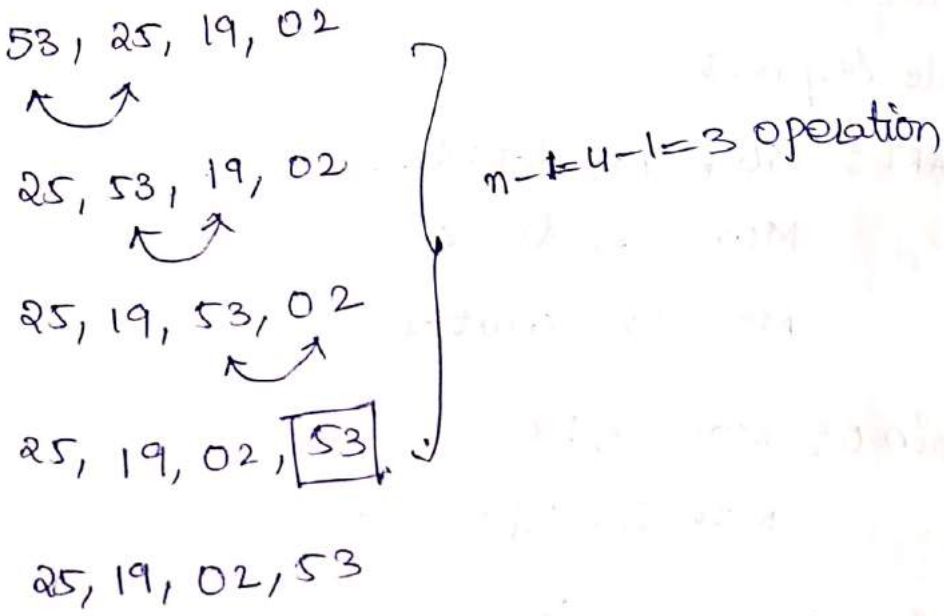
PR 1

8) Write an ALP to sort the numbers in ascending order

```
ASSUME  cs: code  ds: data
Data Segment
List dw 53H, 25H, 19H, 02H
count equ 04
Data ends
code segment
START: MOV AX, data
       MOV ds, AX
       MOV DX, count-1
Again 0: MOV CX, DX
         MOV SI, offset list
Again 1: MOV AX, [SI]
         CMP AX, [SI+2]
         JL PR1
         XCHG [SI+2], AX
         XCHG [SI+1], AX
PR1: ADD SI, 02
     loop again 1
     Dec CX
     JNZ Again 0
```

```
MOV AH,4CH
INT 21H
Code ends
end START
```

Bubble Sort Algorithm:



→ In n-1 operations the first largest num is found

→ In n-2 operations 2nd largest number is fo

→ In n-n operations last largest number c can find

→ If the 1st number is less than second number Jump to the label PR1. Otherwise exchange the contents lower byte with lower byte & upper byte with upper byte of 1st & 2nd numbers respectively.

→ loop performs two steps internally given by, Incrementing SI 2, decrementing CX 3, checking whether the counter is zero (or) not

→ Decrement CX from 3 to 2 to find the 2nd largest number

→ check if CX=0 (or) not
↙ counter

if counter is zero it will come out of the loop

if counter is not zero it will jump to the label again '0' & repeats the same process.

→ MOV AH, 4CH DAAS Prompt

→ INT 21H → software interrupt

Note:

To Sort numbers in Descending order Replace JL with JG

Write an ALP to display the message of a string

```
ASSUME: CS: code  ds: data

Data Segment
Message db 0DH, 0AH, "study of Microprocessor is intresting, "$"
Data ends

code Segment
START: MOV AX, data
       MOV ds, AX

       MOV AH, 09H
       MOV DX, offset message

       MOV AH, 4CH
       INT 21H
code ends
end START
```

DB → define byte since the characters are represented ASCII values.

0DH → cursor points to the 1st line.

0AH → cursor points to the next line.

$ → indicates end of the string.

→ Initialize data in data segment in code segment

→ MOV AH, 09H → set function for display

→ MOV DX, offset message → DX is pointed to offset address with label 'message' & start reading the ASCII values

→ MOV AH, 4CH → return to DOS prompt

→ INT 21H → interrupt

→ Questions:

1) Explain about 8086up architecture.

2) Explain the pin configuration of 8086up

3) Explain about memory segmentation

4) Explain about Interrupts & Interrupt vector table (IVT)

5 → Explain about various addressing modes with examples.

6 → How do you calculate physical address? Give with an example

7 → Explain about various assembler directives.

8 → Explain about the instruction set of 8086.

9 → Explain about the programming in MASM with programs.

11/1/19

UNIT-II

INTRODUCTION TO Micro Controllers (8051 µc)

The basic difference b/n µc & µp is that in µc all the Supporting IC's like

8255 PPI (programable peripheral Interference)

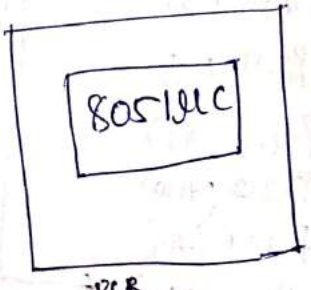
8257 DMA (Direct Memory Accessable)

8251 USART (Universal Synchronous Asynchrono-us transmitter Reciever)

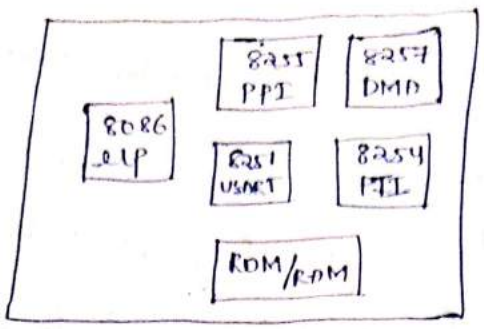
8254 PTI (programable timer Interrupt)

along with inbuilt memory or RAM & ROM is present in a single IC. Where as in 8086 µp they have to be provided externally.

As a result in 8086 µp size is big, cost is high, computational complexibility is more, Speed decreases.

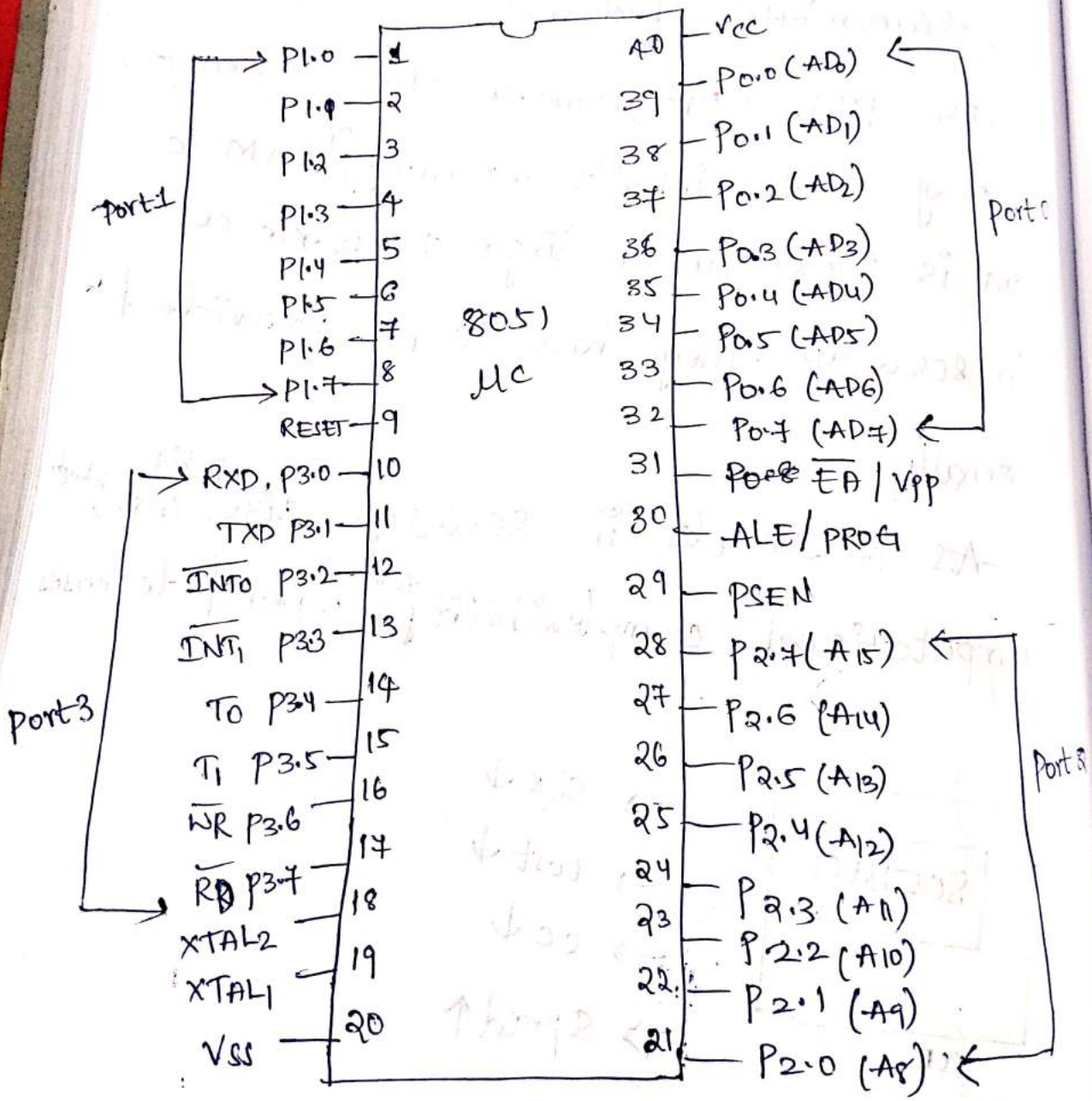


→ size↓
→ cost↓
→ cc↓
→ speed↑



→ size ↑
→ cost ↑
→ cc ↑
→ speed ↓

→ Pin diagram of 8051 µc



→ 8051 is available in 40pin IC in plastic package & Dual in package (DIP)

→ It has got 4 ports given by port '0', port '1', port '2', port '3'

→ Each port has got '8' pins. Thus a total of $4 \times 8 = 32$ I/P & O/P devices can be connected.

→ port '0' acts as multiplexed address data bus as well as Input output ports.

→ port '1' is acting as only Input & output ports.

→ port '2' is acting as ~~both~~ only address & I/O ports.

→ port '3' is called as Multifunctional Port because apart from I/O operations it performs the following things:

→ RXD recieves the Serial data
→ TXD transmits Serial data

} These two are acting as 8251 IC (USART)

→ $\overline{INT_0}$ is interrupt '0' } These two pins are
→ $\overline{INT_1}$ is interrupt '1' } acting as 8259A Priority Interrupt Controller (PIC)

→ T_0 → Timer '0' } These two are
→ T_1 → Timer '1' } working as 8254 IC (PTI)

→ If $\overline{WR} = 0$ then write operation takes place from controller to peripherals.
→ If $\overline{RD} = 0$ read operation takes place from peripherals to controller

→ XTAL1, XTAL2 are crystall oscillators.

There are two clock crystal oscillators which provide Clock frequency in the range of 8MHz & 12MHz

→ From Clock frequency we get the time period of the execution of the controller

→ V_{cc} = 5V/12V DC is enough to run the IC

→ V_{ss} = ground

→ $\overline{EA}$ = external axis of memory.

If $\overline{EA} = 0$ external memory is accessed

If $\overline{EA} = 1$ external memory is not accessed

→ $\overline{PSEN}$ → program store enable

If $\overline{PSEN} = 0$ program is stored

$\overline{PSEN} = 1$ Program is not stored.

→ ALE → Address Latch enable

→ It is used to demultiplex the address & data bus.

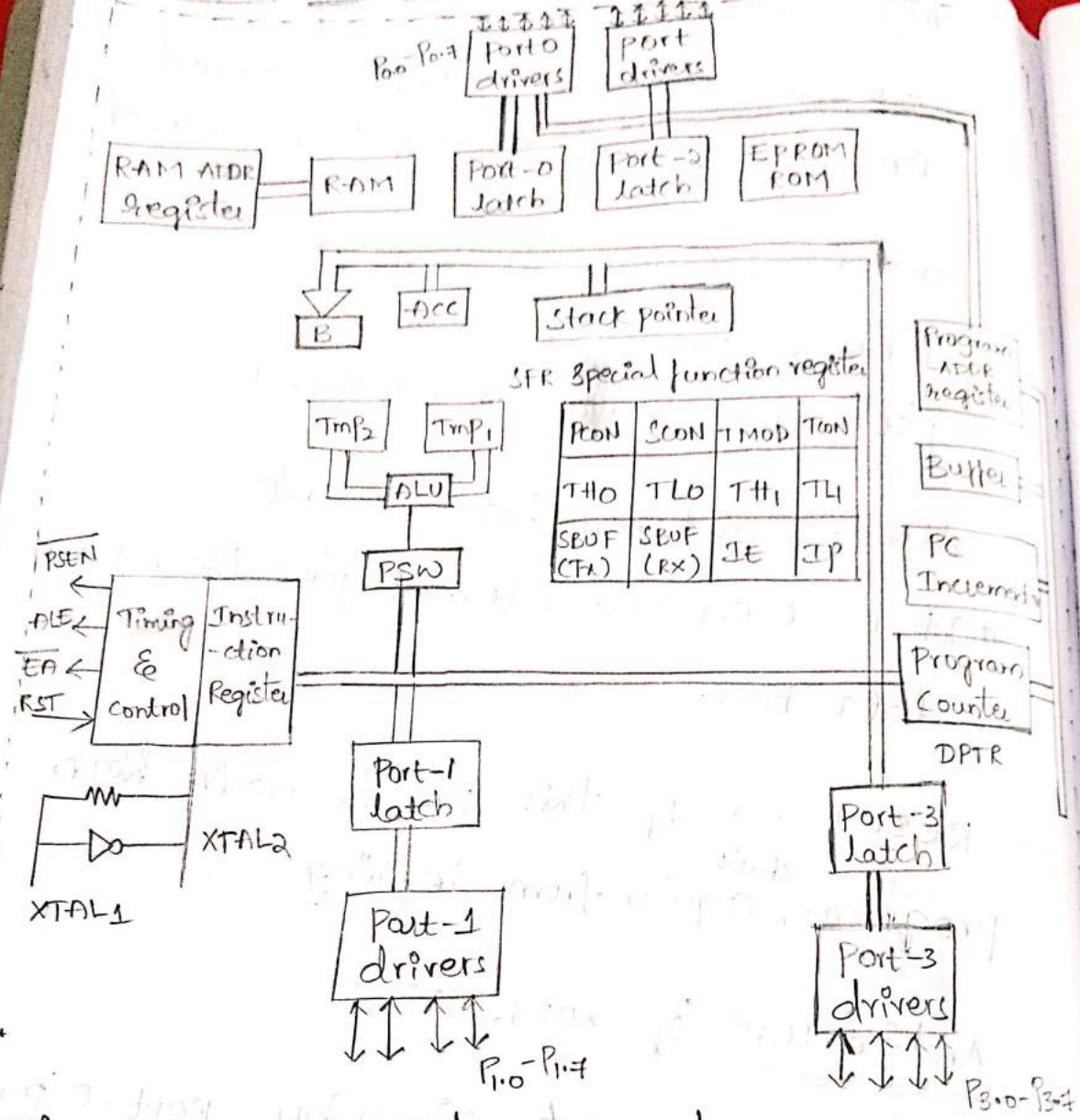
→ RESET → If this pin is active high program starts again from begining.

→ Architecture of 8051 µC:-

→ There are 4-ports given by port-0, port-1, port-2, port-3 & each port has 8 I/O pins. Thus a total of 4x8 = 32 I/O pins for interfacing purpose.

→ All these 4-ports are drived by 4-latches respectively

Architecture of 8051 IC:



→ There are 4-ports given by port-0, port-1, port-2, port-3

→ A latch is a one-bit memory cell which is used for input & output operations & for better safety.

→ Microcontroller has got inbuilt RAM of 128 bytes

which is accessed by RAM address register

→ In RAM data of the program is saved.

→ RAM is a volatile memory

→ Microcontroller has got inbuilt 4K ROM which is accessed by program address register.

→ In ROM opcode of the program is stored.

→ ROM is a non-volatile memory

→ There are various registers in microcontroller given by

- Two temporary registers Tmp_1, Tmp_2
- B register
- Accumulator

All these register are of 8-bits. Accumulator is used to store the result.

→ It comprises of ALU to perform all arithmetic & logical operations.

→ It Comprises of controlling & timing circuit.

1) $\overline{PSEN}$: (program store enable): If $\overline{PSEN}=0$ program is stored.

2, ALE (Address Latch enable): It is used to demultiplex the buses

3, $\overline{EA}$ (External Access): If $\overline{EA}=0$, External Access is used otherwise not

4) RST (Reset the operation & start from the beginning.

→ Stack Pointer: Stack pointer pointes to the top of the Stack.

→ There are 2- crystal oscillators XTAL1 & XTAL2 which provides clock frequency in the range of 8MHz - 12MHz. From clock frequency time period of the execution of the instruction is achieved. $T = 1/f$

→ program Counter:- The next byte is incremented by means of pc incrementor

→ DPTR (Data pointer): It is similar to Source Index in 8086µp.

→ Buffer: Microcontroller comprises of buffer to store intermediate data.

→ Microcontrol comprises of SFR given by

PCON : power control

SCON : Serial control

TCON : Timer / counter control

TMOD : Timer / counter mode registe

TH_0 & TL_0 : of timer '0'

TH_1 & TL_1 : of timer '1'

Note :

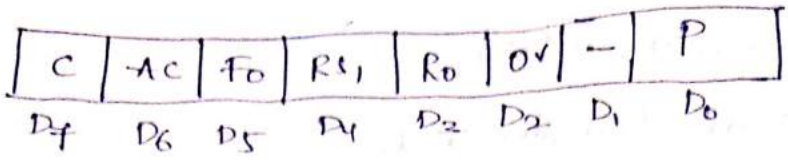
Timmer-0 & Timmer-1 are of 16-bit

→ SBUF : Serial Buffer (Transmitter / reciever)

→ IE : Interrupt Enable control

→ Ip : Interrupt priority control

Program Status word: (PSW)



→ It is a 8-bit register which is similar to flag register of 8086 up

C – carry flag

AC – Auxiliary carry flag

f0 – Special purpose flag

RS1 & RS0 – Register bank Selection

OV – overflow flag

P – priority flag.

Register Bank Selection

RS1	RS0	Bank Selection
0	0	Bank 0
0	1	Bank 1
1	0	Bank 2
1	1	Bank 3

Note:

-Ao-

Reg

→ 8

by

1)

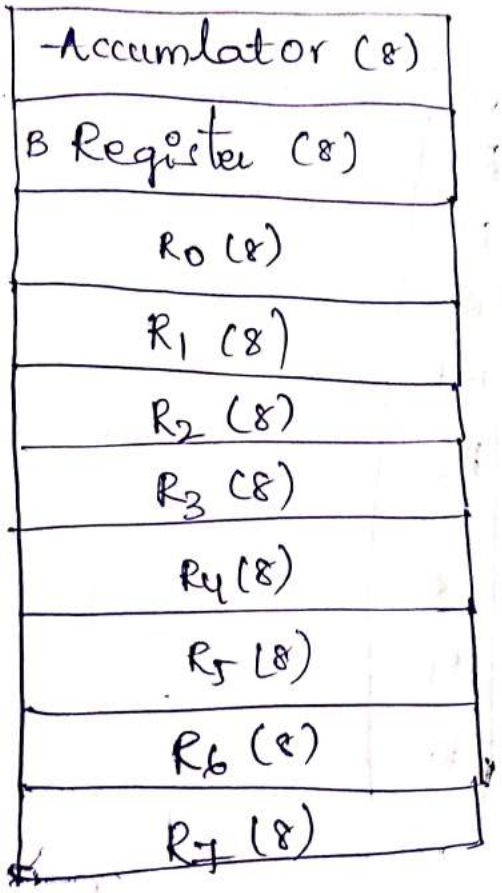
b is similar

Note: 8051 µc is a 16 bit address line A_0-A_{15} & 8 bit data lines D_0-D_7 Ic

Register Organisation of 8051 µc:

→ 8051 µc has got various registers is given by

1) General purpose (or) working Registers:-

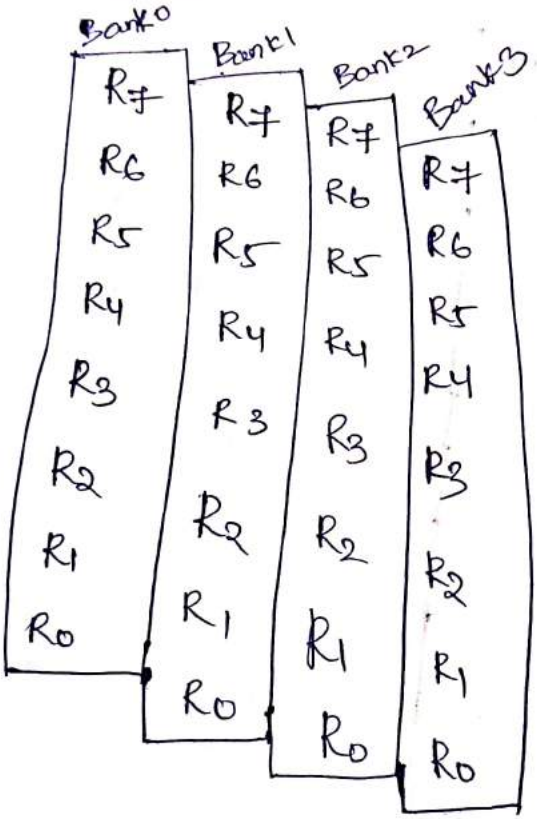


There are 8 registers R_0-R_7 which are of 8-bits

2, Pointer Register

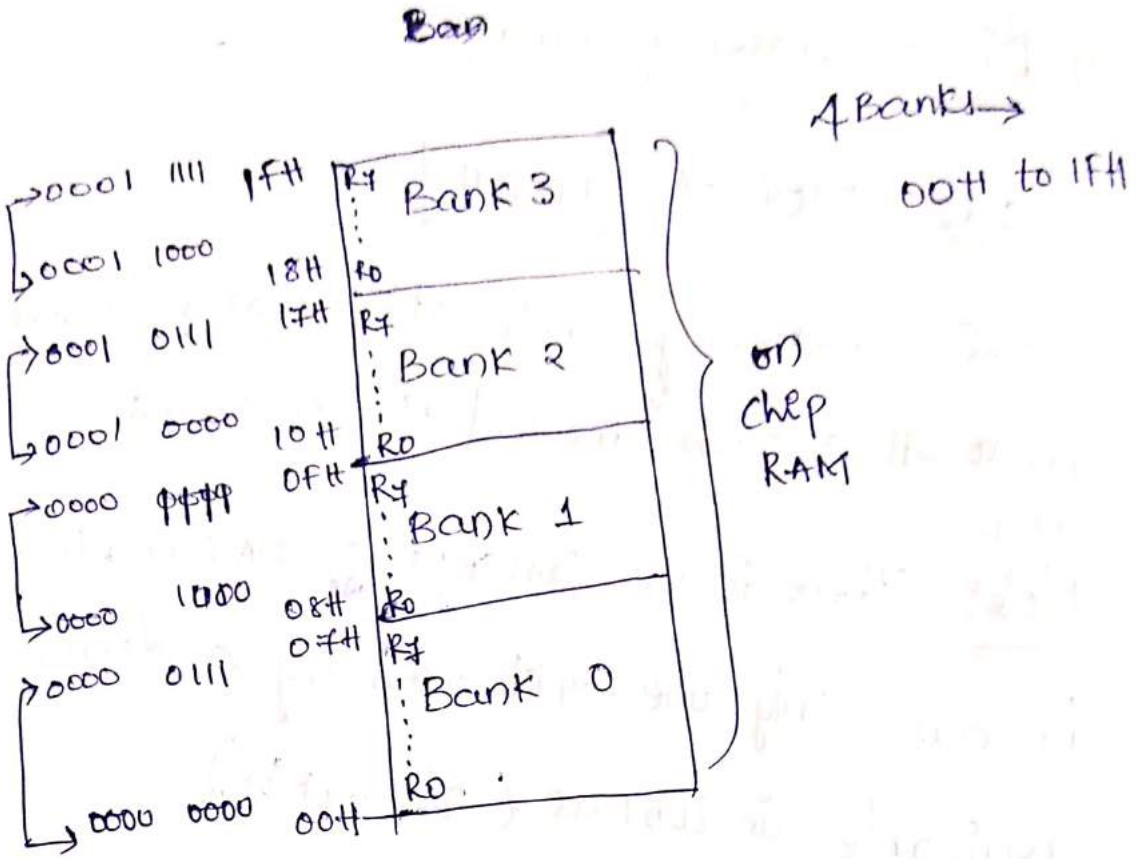
Program counter (pc) (8)
Data pointer DPTR (16)
DPH (8) (or) DPL (8)
SP Stack pointer (8)

3, Bank Registers.



→ There are 4-banks given by Bank0, Bank1, Bank2, Bank-3. Each Bank has got 8- Registers $R_0 - R_7$. Each of 8-bit.

Thus the total of 32 Registers are available.



Note: On chip memory is also known as inbuilt memory.

Addressing Modes of 8051 µc:

Addressing means the way the communication is being performed b/n any two Operands.

→ The operands can be either Register, data (or) address

There are various addressing modes in 8051 µc given by

1) Direct addressing mode:

Ex:- MOV A, [5000H]

In this addressing mode, the data present in 5000H address is moved to accumulator.

Note: There is no concept of PA (or) EA because only one 64kb memory address is available in 8051 µc (2^{16} = 64Kb)

2) Immediate addressing Mode:

Move immediate ← address → Data

Ex: MVI A, 54H

↓ Register

→ In this mode the immediate data 54H is moved to register

Note: # & @ are used to denote data

3) Register indirect addressing mode:-

ex: MOV R_1, R_2 → Registers

→ In this addressing mode R_1 & R_2 belongs to different banks.

4) Register instruction addressing mode:

→ In this mode the contents of R_7 is added with accumulator & result is stored in accumulator

ex: ADD A, R_7

5) Register Specific Instruction addressing mode:

ex: RL A

Rotate Left the contents of Accumulator.

Note: In this there is only one operand.

6) Base Register Indexed Addressing mode:

ex: MOV A, @A+DPTR.

A → Register, @A+DPTR → Register, DPTR → Address

→ In this addressing mode DPTR is pointing to some address & the data present in that address is added with previous content of accumulator & result is stored in accumulator.

→ Instruction set of μc (8051):-

There are various instruction sets in 8051 μc given by

1) Data transfer & Copy Instructions:-

Ex: MOV A, Rn

MVI A, #54H

XCHG A, Rn

MOV A, @A+DPTR

n=0 to 7

2) Arithmetic Instructions:

Ex:

ADD A, Rn
MUL A, Rn
SUB A, Rn
DIV A, Rn

} n = 0 to 7

3) Logical I

Ex: A

The conte

4) Boo

Ex:

(3) Logical Instructions:-

ex: ANL A,Rn

ORL A,Rn

XRL A,Rn

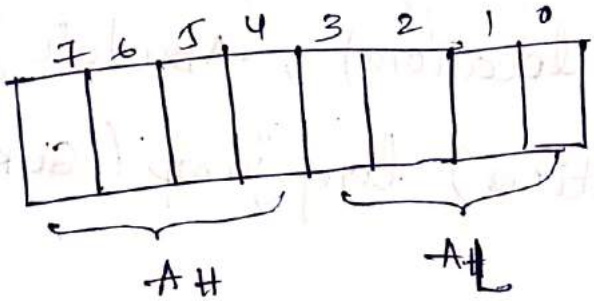
RL A

RR A

Swap A

↙

The contents of A_L & A_H are interchanged.



(4) Boolean instructions:-

ex: CLR C

JC → jump carry

JNC → jump no carry

JB → jump below

JNB → jump not below

SETB P1.1 → Set bit of port 1 1.1 pin with '1'.

(5) Program Branch Instructions:

All the Branch instructions like

JMP

RET

CALL

INT

Note:

There are three types of jumps : short jump (256 address locations) ; Absolute jump (2k address locations), long jump (64K address locations.

(2K)

// There are '3' calls short Call, Absolute Call, (64K) long Call (64K)

Special function register (SFR):

In 8051 MC there is a Special function register called SFR which performs the following functions.

1, Timer / counter:

In MC there are two timers & two counters

which are decided by the pin $C/\overline{T}$

→ If $C/\overline{T} = 0$ Timer Operation is performed

$C/\overline{T} = 1$ counter operation is performed

→ Two timers are given by T_0 & T_1 which are 16-bit timers

→ They also can be altered by 8-bit timers given by T_{H0}, T_{L0}, T_{H1}, T_{L1}

→ There are two Counters given by C_0, C_1 which are 16-bit counters.

→ They also can be operated by 8-bit counters given by C_{H0}, C_{L0}, C_{H1}, C_{L1}

Definition Of timer:

→ The time taken to complete a given instruction which is measured in machine cycles and where there are no boundaries Specified is called timer.

Definition Of counter:-

→ The time taken to complete a given instruction which is measured in machine cycles

and where there are boundaries Specified is called a counter

Note: For up counter the task is completed when it reaches the maximum value.

→ For down Counter the task is Completed when it reaches the '0' value.

TCON: (Timer/counter control register):

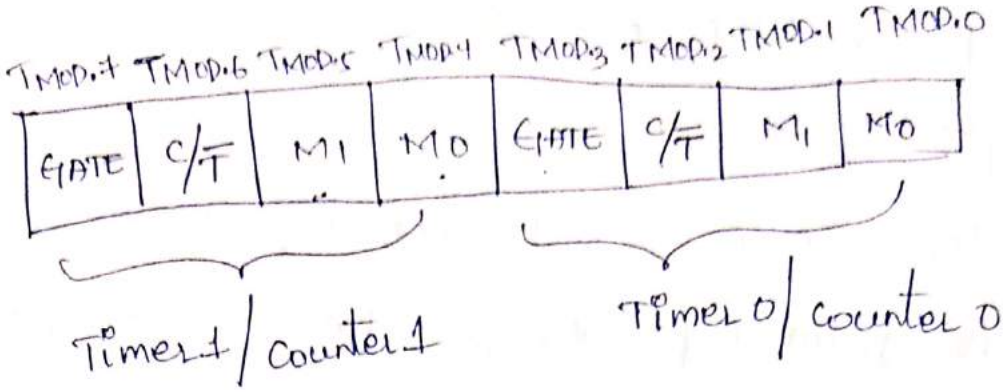
→ The timer & Counter Operations are decided by two 8-bit registers given by TCON & TMOD

→ TCON is a 8-bit register given by

TCON.7	TCON.6	TCON.5	TCON.4	TCON.3	TCON.2	TCON.1	TCON.0
TF_1	TR_1	TF_0	TR_0	IE_1	IT_1	IE_0	IT_0

TCON.7	TF_1	Timer overflow of timer 1
TCON.6	IE_0	Interrupt enable of timer 0
TCON.5	IT_1	Interrupt of timer 1
TCON.4	IE_1	Interupt enable of timer 1
TCON.3	TR_0	Timer run of timer 0
TCON.2	TF_0	Timer overflow of timer 0
TCON.1	TR_1	Timer run of timer 1
TCON.0	~~IT_0~~	Interrupt Timer overflow of timer 0

TMOD (Timer Mode control Register):



M_0	M_1	Mode
0	0	Mode1
0	1	Mode2
1	0	Mode3
1	1	Mode4

There are four modes of operations of timers & counters decided by M_0 & M_1 bits.

→ If $C/\overline{T} = 0$ (timer)

$C/\overline{T} = 1$ (counter)

→ GATE: It is a triggering pulse which is of either level triggering or edge triggering

Note: Triggering is used to change the state.

Mode 0 :

C/$\overline{T}$ is used to decide timer & counter Operation.

→ The timer is initiated by means of run bit given by TR_0 for timer 0 TR_1 for timer 1.

→ Interrupt is enabled by the means of a gate pulse either by edge triggering (or) level triggering. IE_0 for IT_0 & IE_1 for IT_1

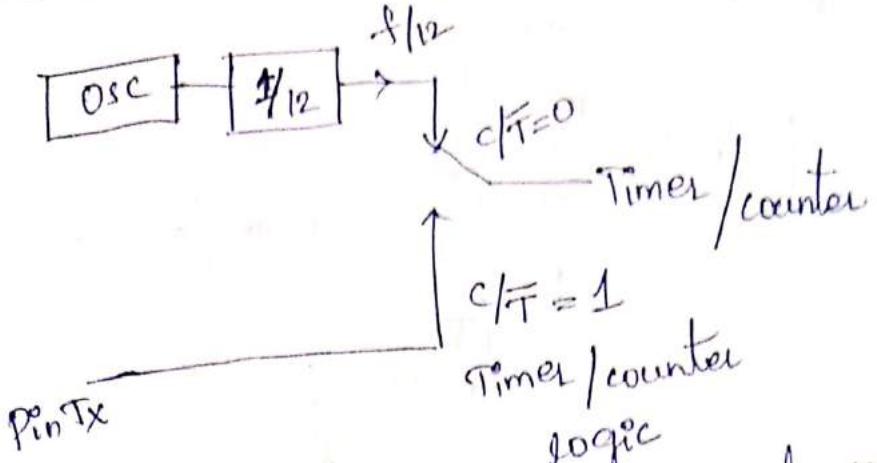
→ This is an 13-bit timer obtained by taking the upper byte of T_H & lower 5-bits of T_L

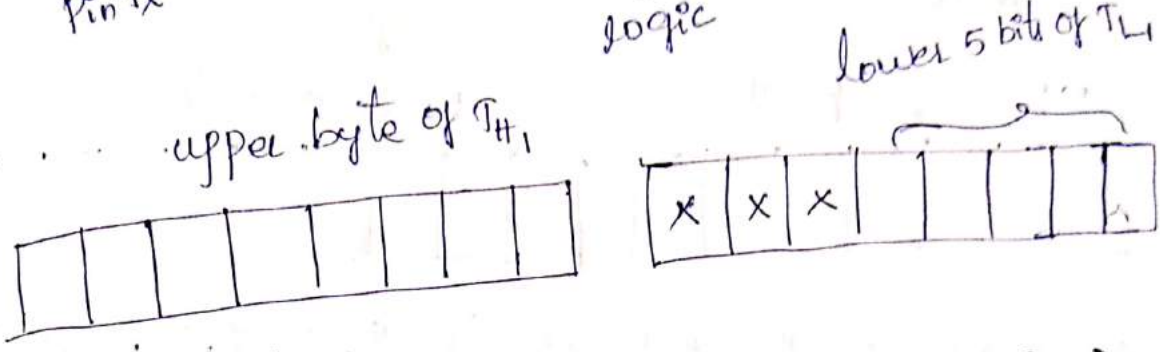
→ The task is indicated by timer overflow for its completion.

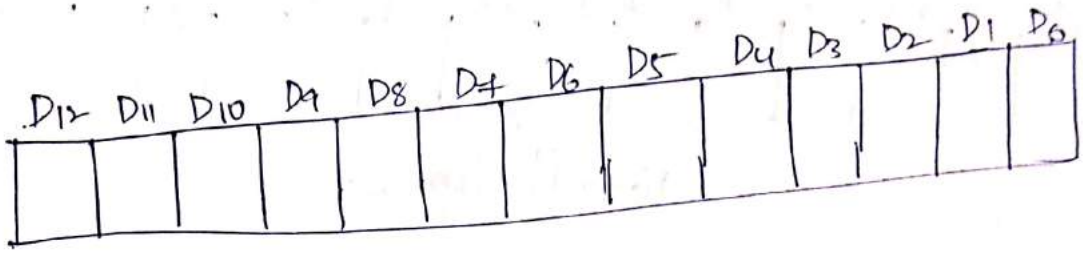
→ Since it is a 13-bit timer there is no overflow.

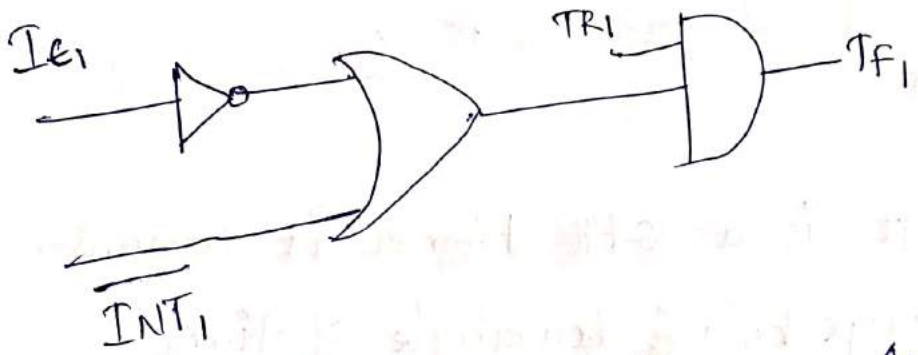
→ A crystal oscillator which provides 12MHz is intern divided by 12 to get 1MHz clock frequency whose time period is 1 μs.

Mode '0' Timer '1' operation



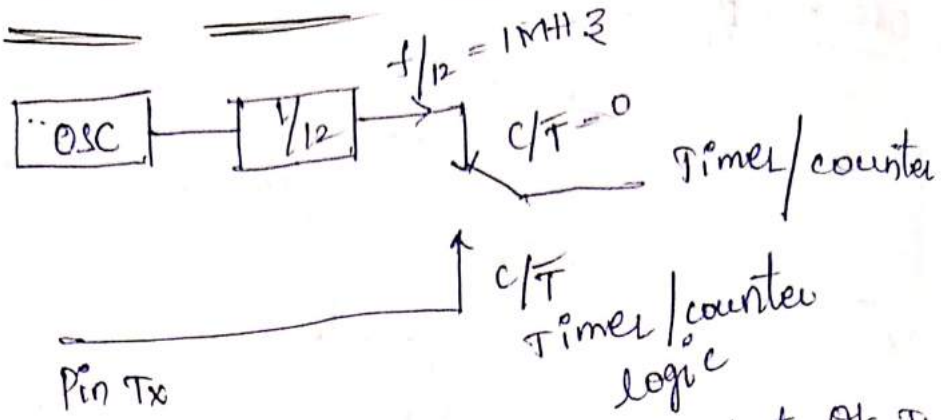


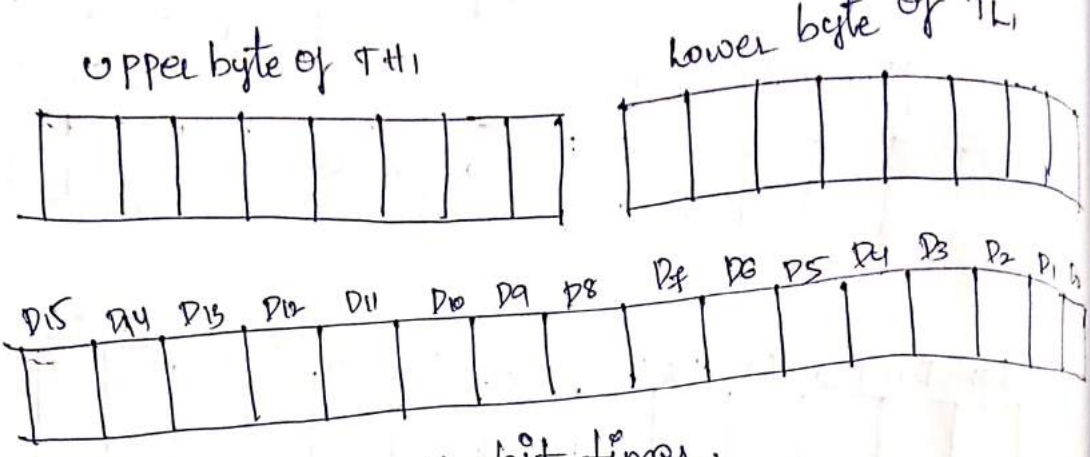




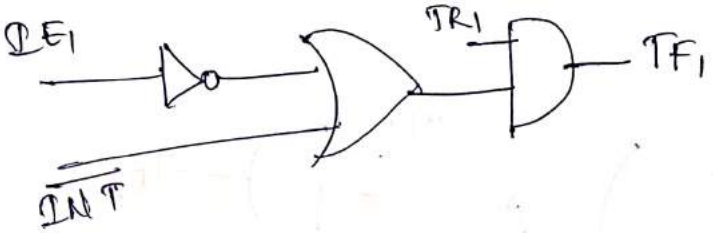
Note: From the above diagram 4-diagrams are generated Mode '0' Timer 1 operation, Mode '0' Timer 0, Mode '0' counter 1, mode '0' counter 0.

Mode '1' Timer '1' Operation:





16-bit timer.



Note: It is a 16 bit timmer it comprises of both upperbyte & lowerbyte of timer.

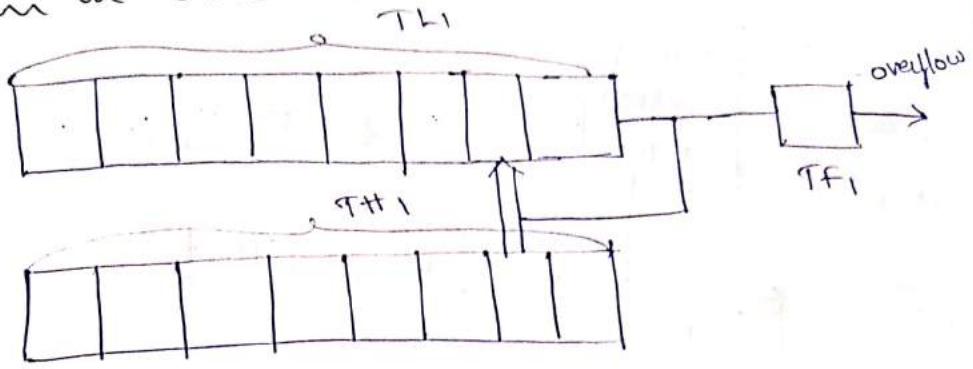
→ In this mode there is timer overflow since it is a 16-bit timer.

→ from the above diagrams 4-diagrams can be generated.

Mode-1 Timer1, Mode-1 Timer0, Mode-1 counter1

Mode-1 counter0

Mode 2 Timer 1 operation.



→ It is an 8-bit timer (or) counter

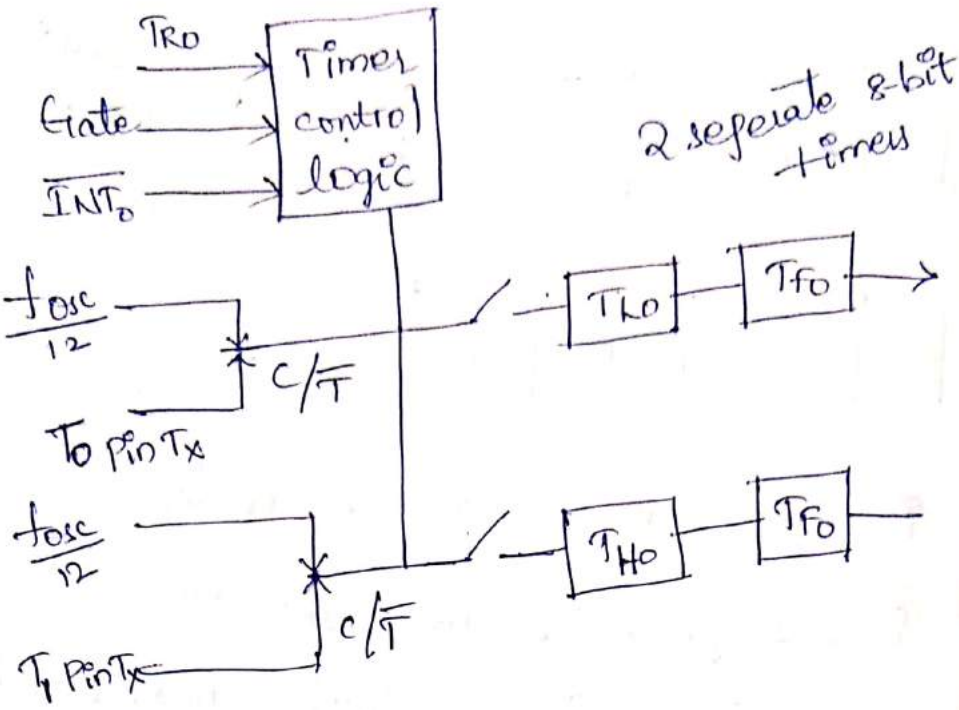
→ The 8-bits from T_L are sent to T_F & T_F is used to indicate whether timer overflow is there are not

→ A separate Mechanism is there to Auto Reload the contents of T_L from T_H

→ Since it is a 8-bit timer there is timer overflow.

→ from the above diagram we can generate four diagrams. Mode2 timer 1, Mode2 timer '0', Mode2 Counter 1, Mode 2 counter 0

Mode 3 timers :



→ Two seperate 8-bit timers are available in mode '3'

→ Since it is a 8-bit timer there is a timer overflow

→ From the above diagram we can generate 4 diagrams given by Mode '3' timer 0, Mode '3' timer '1', Mode '3' Counter '0', Mode '3' counter

M0 M1	Mode	Timer
0 0	Mode 0	13-bit timer
0 1	Mode 1	16-bit timer
1 0	Mode 2	8-bit timer
1 1	Mode 3	Two 8-bit timers

Serial communication control:-

Serial Communication is performed by means of few registers given by 1, SBUF (Tx) → Serial buffer in transmitter mode 2, SBUF (Rx) → Serial buffer in reciever mode. 3, PCON → power control 4) SCON → Serial Control.

S-CON Register:

It is an 8-bit register which gives the information about various modes of operation of Serial data given by

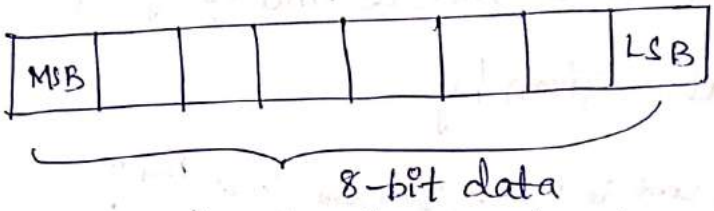
SCON.7	SCON.6	SCON.5	SCON.4	SCON.3	SCON.2	SCON.1	SCON.0
SM0	SM1	SM2	REN	TB8	RB8	TI	RI

Address	SCON bit	Description.
9FH	SM0	Serial communication mode
9EH	SM1	serial communication mode.
9DH	SM2	If this bit is set it operates as Multiprocesor in mode 2 & 3
9CH	REN	Recieve Enable
9BH	TB8	Transmitt 9th bit
9AH	RB8	Recieve 9th bit
99H	TI	Transmitt Interrupt flag
98H	RI	Recieve Interrupt flag

SM0	SM1	Mode	Description	Baud rate
0	0	Mode 0	8 bit shift Register	$f_{osc}/12$
0	1	Mode 1	8 bit UART	variable (T_1)
1	0	Mode 2	8 bit UART	$f_{osc}/64$ (or) $f_{osc}/32$
1	1	Mode 3	9 bit UART	variable (T_1)

Mode '0':

→ In this mode an 8-bit shift register comprizing of LSB & MSB data is transmitted given by

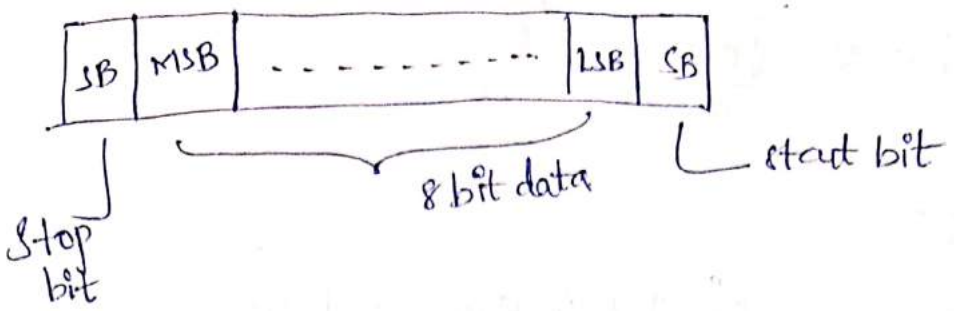


→ TXD is for transmitting Serial data
RXD is used Recieving Serial data.

→ The Baud rate is given by $f_{osc}/12 = 12/12$
$1MHz \Rightarrow T = 1\mu s$ which is fixed.

Mode '1':

→ In this mode 10 bits are used for data transmission which comprises of 8 bit data, one start bit & one stop bit given by



→ TxD is used for transmitting Serial data
RxD is used for recieving Serial data

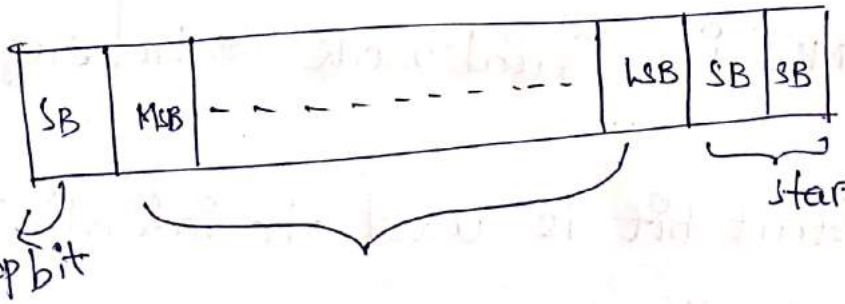
→ Baud rate is variable which is set by timer '0' & given by

$$\text{Baud rate} = \frac{2^{SMOD}}{32} \times (T_{F0})$$

where SMOD = Serial mode Control register

Mode '2' :-

→ In this mode 11 bits are transmitted which comprises of 8-bit data, two start bits & one stop bit given by

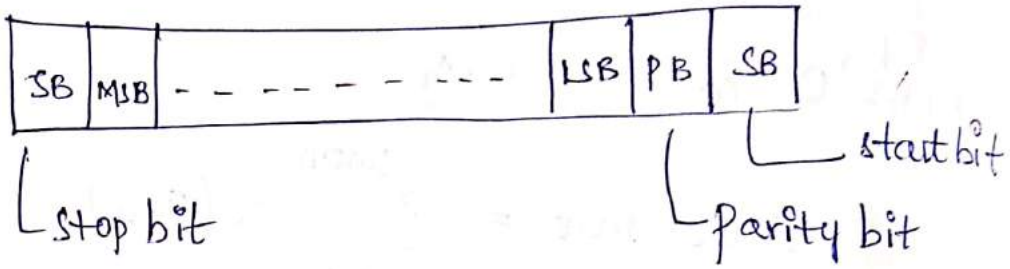


→ TxD is used to transmitt the serial data
→ RxD is used to recieve the serial data.

Baud rate = $\frac{f_{osc}}{64}$ (or) $\frac{f_{osc}}{32}$

Mode '3':-

→ In this mode 11 bits are used for transmission which comprises of 8 bits of for data, one start bit, one stop bit, one parity bit given by



→ T_{XD} is used to transmitt the Serial data
R_{XD} is used to recieve the Serial data

→ Baud rate is variable by timer 1 is given by

$$\text{Baud rate} = \frac{2^{SMOD}}{64} \times (T\&F_1)$$

where SMOD is Serial mode control register

Note: Start bit is used to indicate the Starting of the frame.

→ Stop bit is used to indicate the ending of frame.

→ Parity bit is used to indicate errors

→ Mode '3' is preferable over all modes for Serial communication.

Interrupts of 8051 uC:-

→ 8051 has got 5 vectored interrupts.

→ These vectors are executed by going to their respective ISR address.

→ After completion of interrupts the interrupts are return to the by means of IRET

→ There are two external interrupts given by ($\overline{INT_0}$, $\overline{INT_1}$), two timer Interrupts & one Serial interrupt (RI (or) TI)

Note:-

External interrupts has got highest priority followed by timer interrupts followed by serial interrupts.

Interrupts	flag affected	vector	cause of Inter
→ external interr-upt 0 $\overline{INT_0}$ pin	IE0	0003H	A high to l transition $\overline{INT_0}$
→ Timer/counter 0 Interrupts	TF0	000BH	overflow counter
→ External interr-upts 1 $\overline{INT_1}$ pin	IE1	0013H	A high to l transition
→ Timer/counter interrupts	TF1	001FH	overflow Timer Counter 1
→ serial port	RI+TI	0023H	when either TI (or) RI is set

Interrupts Enable (IE) Register:

IE.7 1 = enable 0 = disable IE.0

EA	–	–	ES	ET_1	EX_1	ET_0	EX_0

EA – Enable / disable all interrupts

ES – Enable / disable Serial interrupts

ET_1 – Enable / disable timer interrupts 1

EX_1 – Enable / disable external interrupts 1

ET_0 – Enable / disable timer interrupts 0

EX_0 – Enable / disable external interrupts 0

Interrupt priority register (IP):-

IP_7							IP_0
X	X	PT_2	PS	PT_1	PX_1	PT_0	PX_0

X → don't care

PT_2 – Timer priority – (2)

PS – Serial Interrupt priority (3)

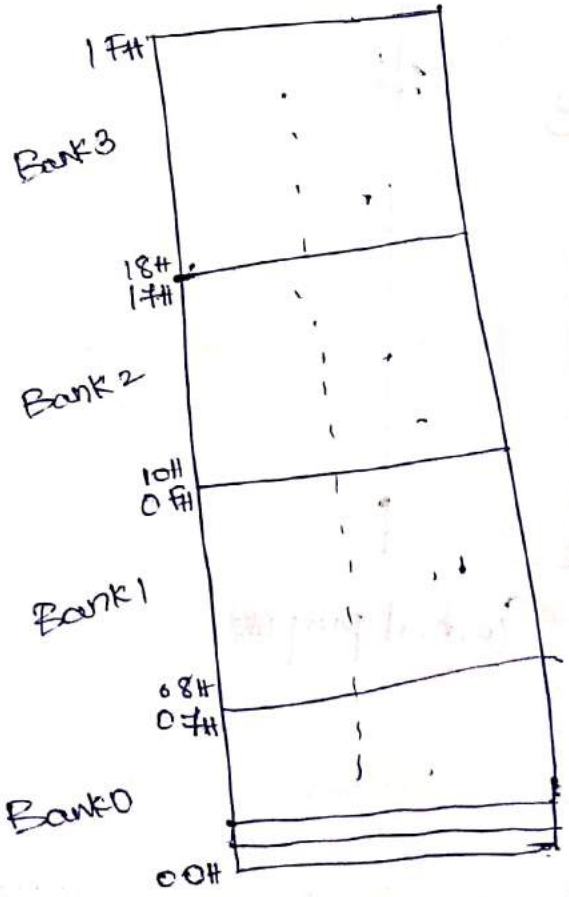
PT_1 – Timer 1 Interrupt priority – (2)

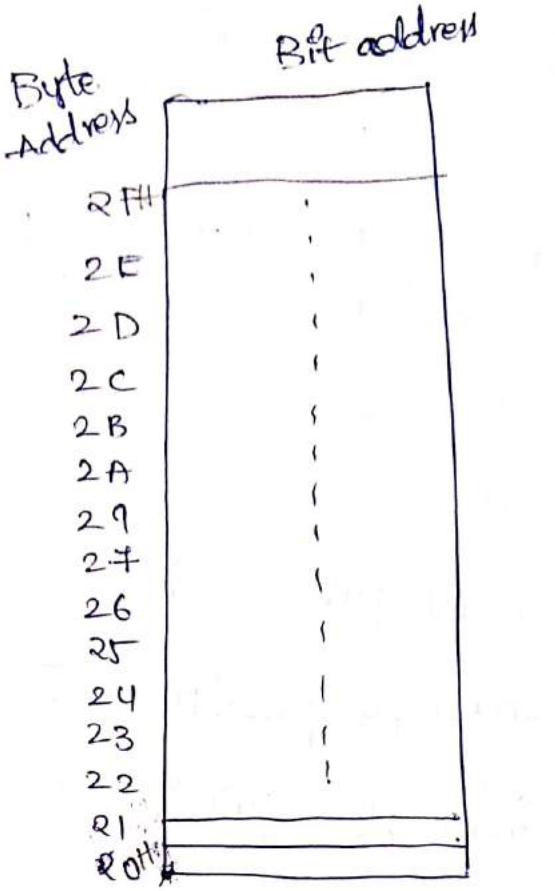
PX_1 – External interrupt 1 priority – (1)

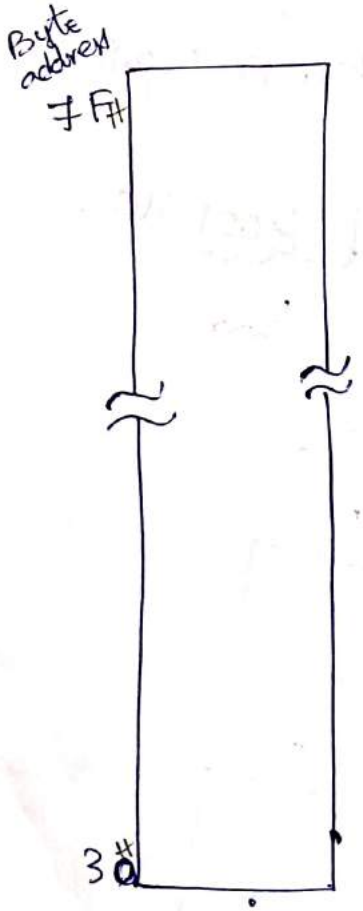
PT_0 – Timer 0 interrupt priority (2)

PX_0 – External interrupt 0 priority – (1)

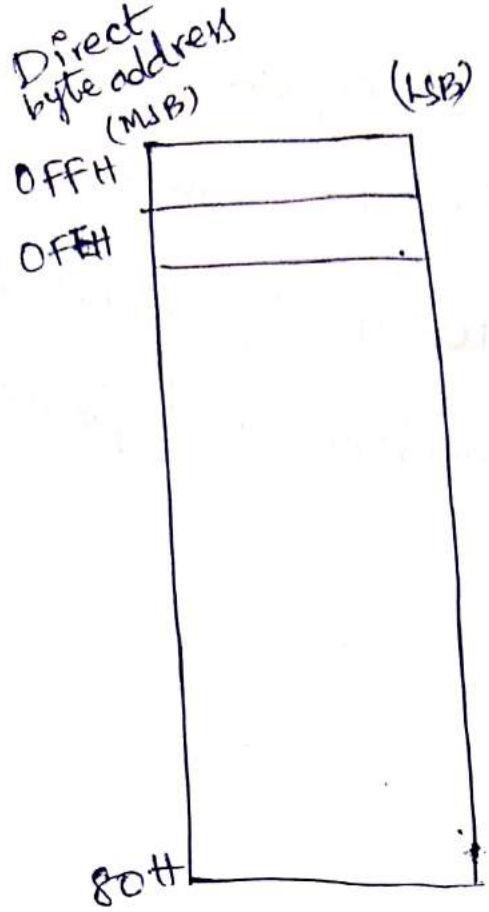
Memory allocation organisation of 8051 uc





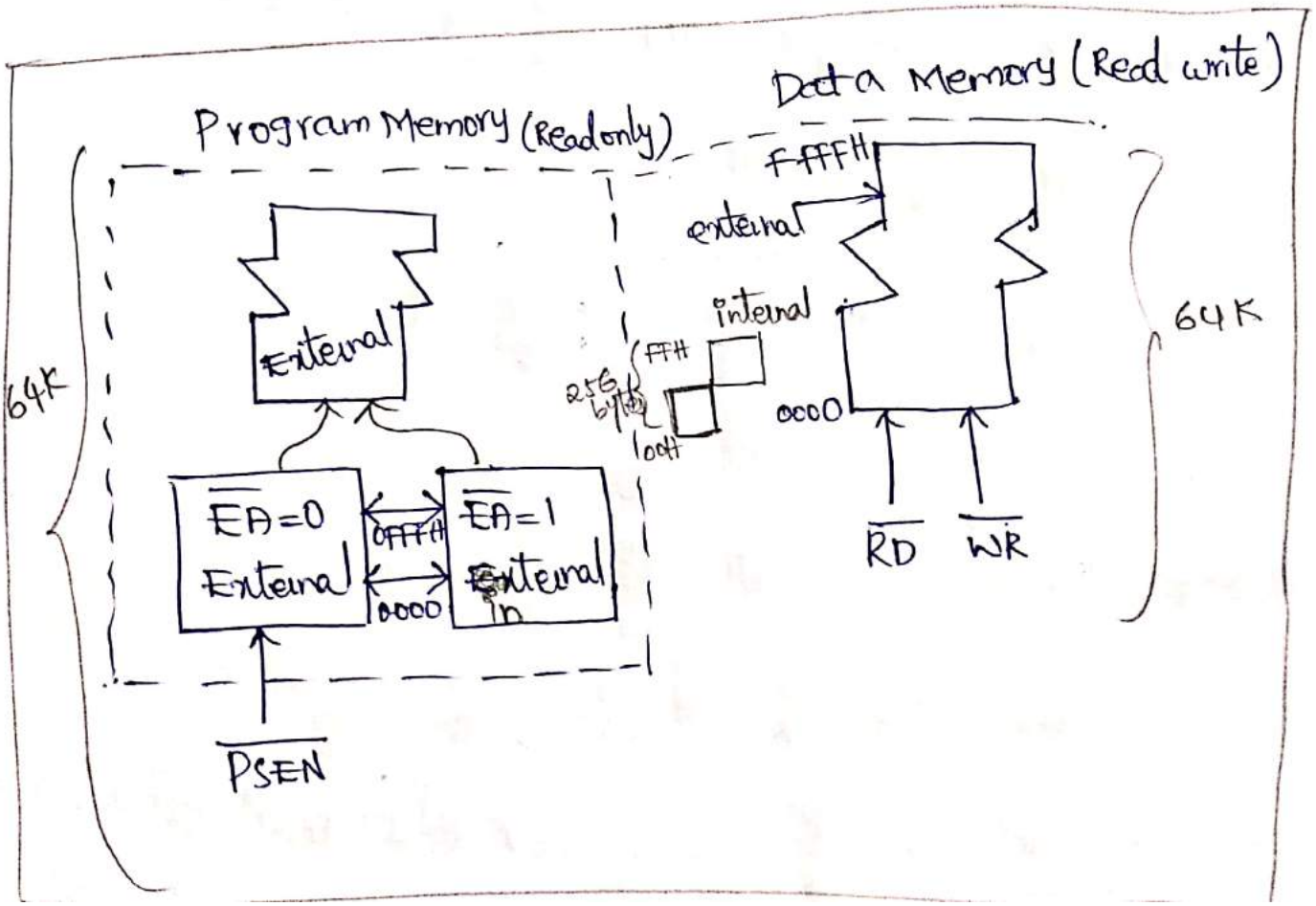


General purpose



SFR bit address

Memory organization in 8051



→ There are four banks & in each bank there are 8 Registers (R0-R7) which are of 8-bit (1 byte). Thus a total of 32 Bank registers are available which are byte addressable.

Starting address is 00H
ending address is 1FH

∴ Size = 1FH − 00H

```
  0001  1111
  0000  0000
-------------
  0001  1111
```

⇒ $2^5 = 32$

→ There are 16-byte (or) 128 bit addressable registers

Starting address = 20H
Ending address = 2FH

Size = 2FH − 20H

```
  0010  1111
  0010  0000
-------------
  0000  1111
```

⇒ $2^4 = 16$

⇒ 16 × 8 ⇒ 128 bit addressable registers

→ There are General purpose registers which are ranging from 30H to FFH

Note: For

General

& ending

→ SFR

∴

∴ T

Note

In

If

→

→

each bank
which are of
of 32 Bank
ue byte addressable.

essable registers

t addressable
registers
hich are

Note: for working registers (Bank registers) & General purpose registers starting address is 00H & ending address is FFH. Therefore size = FFH-00H

$$\begin{array}{r} 0111\ \ 1111 \\ 0000\ \ 0000 \\ \hline 0111\ \ 1111 \end{array}$$

$2^7 = 128$ bytes

→ SFR ranges from 80H to FFH

∴ size = FFH − 80H

$$\begin{array}{r} 1111\ \ 1111 \\ 1000\ \ 0000 \\ \hline 0111\ \ 1111 \end{array}$$

$2^7 = 128$ bytes

∴ Total internal RAM = 128+128 = 256 bytes

Note: Internal ROM = 4K bytes, [0FFFH − 0000H]

$$\begin{array}{r} 0000\ \ 1111\ \ 1111\ \ 1111 \\ 0000\ \ 0000\ \ 0000\ \ 0000 \\ \hline 0000\ \ 1111\ \ 1111\ \ 1111 \end{array} \rightarrow 2^{12} = 2^2 \times 2^{10} = 4 \times 1K = 4K$$

In this case $\overline{EA} = 1$

If external memory is accessed then $\overline{EA} = 0$

→ ROM performs only Read operation ($\overline{PSEN}$)

→ Maximum size is of $2^{16} = 64K$

Internal RAM is of 256 bytes

FFH - 00H $\Rightarrow 2^8 = 256$ bytes

In this case $\overline{EA} = 1$

→ In case of external memory $\overline{EA} = 0$

→ The maximum size is $2^{16} = 64K$

→ RAM performs both read & write operation

Questions:

→ Explain about the pin configuration of 8051 uc

→ Explain about the architecture of 8051 uc

→ Mention the differences between up & uc

→ Explain the register Organisation of 8051 uc

→ Explain the addressing modes of 8051 uc

→ Explain Instruction Set of 8051 uc

→ Explain the memory organisation of 8051 uc

→ Explain about TMOD & TCON register

→ Ex Draw the timers & counters of various modes

→ Explain about Serial communical control SCON register.

→ Explain various modes of serial communication data

→ Explain about vectored interrupts & IE & IP

2/2/19

UNIT-III

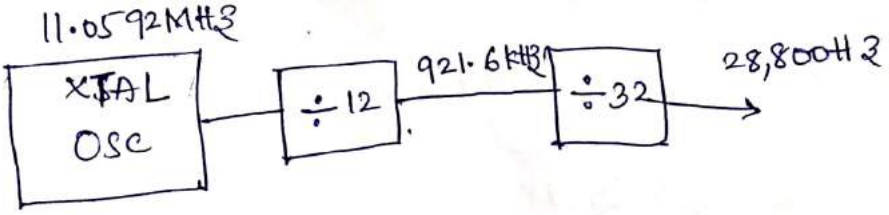
→ 8051 µc has got crystal oscillator frequency given by 11.0592 MHz

→ It is internally divided by 12 in timers to give $\frac{11.0592\ MHz}{12} = 921.6\ KHz$.

→ The UART circuitry inside the µc again divides by 32 to give $\frac{921.6\ KHz}{32} = 28,800\ Hz$.

→ There are various baud rates selected by registers given by

Baud rate	TH
9600	-3
4800	-6
2400	-12
1200	-24

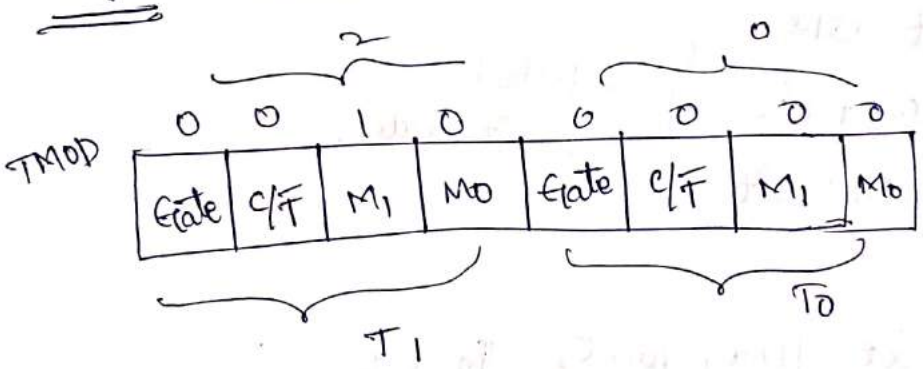


Write a program for 8051 µC to transfer letter 'A' serially at 4,800 baud rate continuously ?-

```
MOV  TMOD, #20H
MOV  TH1, -6
MOV  SCON, #50H
SETB TR1
Again: MOV SBUF, #"A"
Here: JNB TI, Here
      CLR TI
      SJMP Again
```

Jump No below

Step:1 Select the timer



→ T1 (8 bit timer)

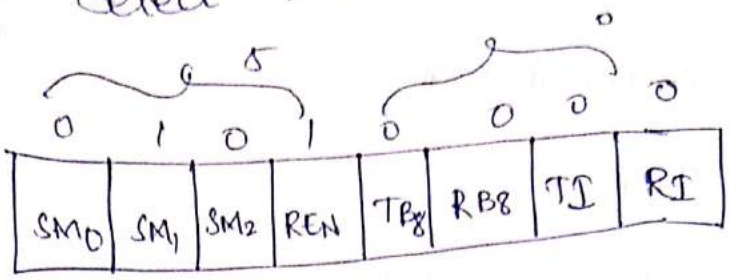
M1	M0	Mode
0	0	Mode0
0	1	Mode1
1	0	Mode2
1	1	Mode3

→ Mode 2 8 bit timer is selected.

Step:2 Select the baudrate of 4,800

→ $\frac{28,800}{6} = 4,800$

Step:3 Select the data size



SM0	SM1	Mode
0	0	Mode0
0	1	Mode1
1	0	Mode2
1	1	Mode3

8 bit data
1 start bit
1 stop bit
} Mode 1
10 bit data.

Step:4 Set timer run of timer 1.

Step:5 transmitt ASCII value of 'A' serially to SBUF

Step:6 transmitt serially till TI = 1

Step:7 Clear timer interrupt

step:8 Short Jump to the label again to repeat the process.

Note: For recieving the data Serially 8 modifications are done in the above Program

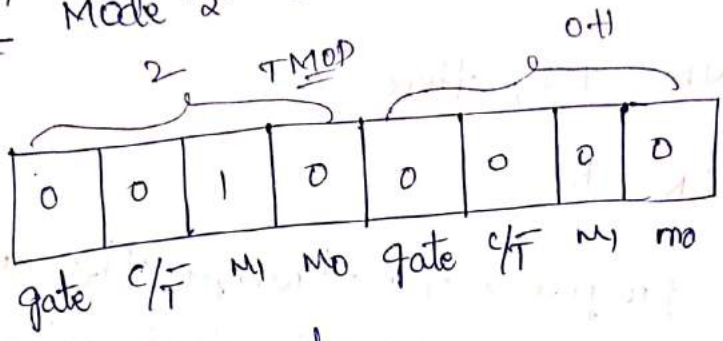
```
MOV #"A", SBUF
JNB RI, Here
CLR RI
```

Write a program to transfer the message "YES" Serially at 9600 baud rate, with 8-bit data one start & stop bit.

```
→ MOV TMOD, #20H
  MOV TH1, #-3
  MOV SCON, #50H
  SETB TR1
Again: MOV A, #"Y"
  ACALL TRANS
  MOV A, #"E"
  ACALL TRANS
  MOV A, #"S"
  ACALL TRANS
  SJMP Again
```

```
TRANS: MOV SBUF, A
Here : JNB TI, Here
       CLR TI
       RET
```

Step:1 Mode '2' 8 bit timer 1 is selected.

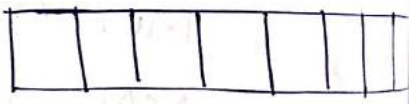


M1	M0	Mode
0	0	Mode0
0	1	Mode1
1	0	Mode2 → 8 bit timer
1	1	Mode3

Step:2 Select 9,600 baud rate

$$\frac{28,800}{3} = 9,600 \text{ bps}$$

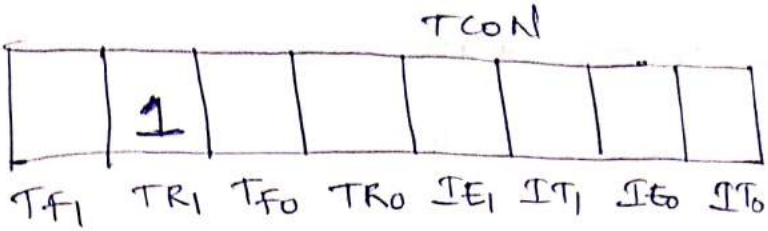
Step:3 Data size is Selected (Mode 1 10-bit data).



1 start bit

1 stop bit.

Step:4 Set bit of timer '1'



Step:5 Move ASCII value of '4' to the accumulator & CALL the label TRANS

Step:6; Move the data serially from accumulator to SBUF tim TI=1 after that clear TI & return back to the main program to transmitt serially. ~~for~~ lly for "E" & "S" also

Short JUMP to the label again to transmit the word 'YES' continuously.

Timer Programming:-

Write a program to create a Square wave of 50% Duty cycle on port-1 5th bit & select timer 0 mode 1

```
        MOV   TMOD, #01H
Here:   MOV   TL0, 0F2H
        MOV   TH0, 0FFH
        CPL   P1.5
        ACALL Delay
        SJMP  Here
Delay:  SETB  TR0
Again:  JNB   TF0, Again
        CLR   TR0
        CLR   TF0
        RET
```

Step: 1

TMOD

0	0	0	0	0	0	0	1

CPL → complement $T_{1:1}$ to generate positive & negative edges of square wave

→ Absolute call delay program.

→ short jump to the label "Here" to repeat the process

→ Set bit of timer run of timer 0

→ Generate the Square wave till Timer Overflow of timer 0 is set

→ clear timer run of timer '0'

→ clear timer Overflow of timer '0'

→ Return back to the main program to repeat the process.

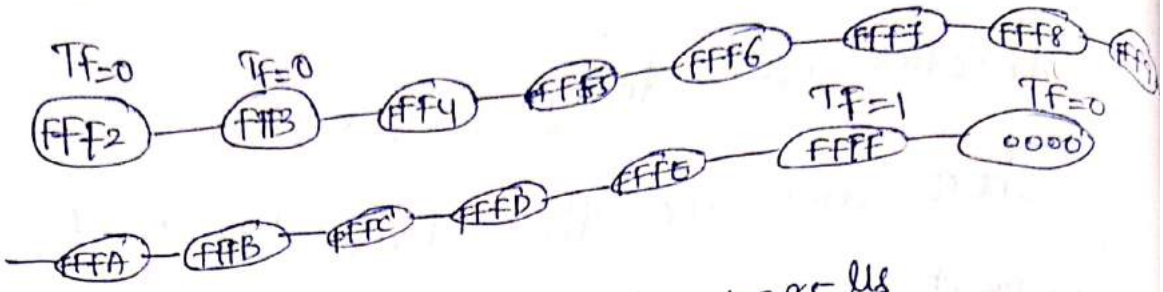
Note: Duty cycle = $\frac{T_{ON}}{T_{ON}+T_{OFF}}$

for Square wave Duty cycle = 50%.

for Square wave $T_{ON} = T_{OFF}$

$$\frac{T_{ON}}{2T_{ON}} = 1/2 \times 100 = 50\%.$$

For the above program the time period of the Square wave is calculated as follows.



Delay = No. of Counts × 1.085 μs

D = 14 × 1.085 μs

D = T/2 = 15.19 μs

T = 2 × 15.19 μs

T = 30.38 μs

[∵ $\frac{11.0592\ MHz}{12}$

f = 921.6 kHz

T = $\frac{1}{921.6\ kHz}$

T = 1.085 μs]

No. of counts = 65536 − Initial value

(or)

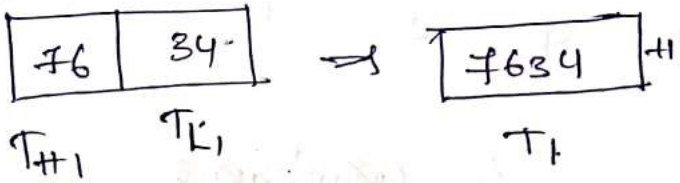
FFFFH − Initial value

→

```
        MOV   TMOD, #10H
Again:  MOV   TL1, #34H
        MOV   TH1, #76H
        SETB  TR1
BACK:   JNB   TF1, BACK
        CLR   TR1
        CPL   P1.5
        CLR   TF1
        SJMP  Again
```

Step:1 Timer 1, Mode 1 (16-bit) is selected

Step:2



Step:3 Set timer run of timer 1.

Step:4 Jump not below until TF=1

Step:5 clear timer run of timer 1

Step:6 Port 1 5th pin is used to generate the Square wave by means of complement function.

~~CLR TF1~~

Step:7 Clear Timer Overflow of timer 1

Step:8 Short Jump to the label again to repeat the same process to generate continuous Square wave.

⇒

No. of counts = FFFFH − 7634H

```
 1111   1111   1111   1111
 0111   0110   0011   0100
-----------------------------
 1000   1001   1100   1011
   8      9      C      B      ⇒ 89CBH
```

→ Perform 2's complement

⇒ 89CBH + 1 = 89CCH

= 35276 (Decimal value)

D = 35276 × 1.085 μs

T/2 = 38.274 ms

T = 76.548 ms

Memory

→ Inter

size 8KB with 8

→ EPROM

8000H

Hex Addr

Hex address

0000H

OFFFH

RAM address

8000H

9FFFH

Memory Interfacing:

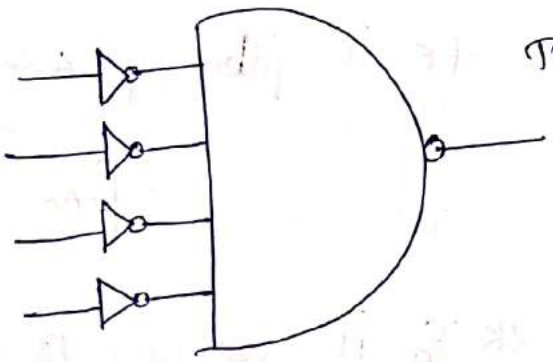
→ Interface an EPROM of size 4KB & RAM of size 8KB with 8051 µC

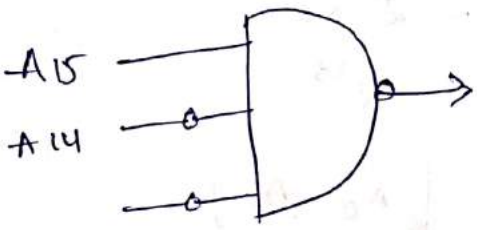
→ EPROM Starts at 0000H & RAM Starts at 8000H address

~~Hex Address A15 A14 A13 A12 A11 A10 A9 A8 A7 A6 A5~~

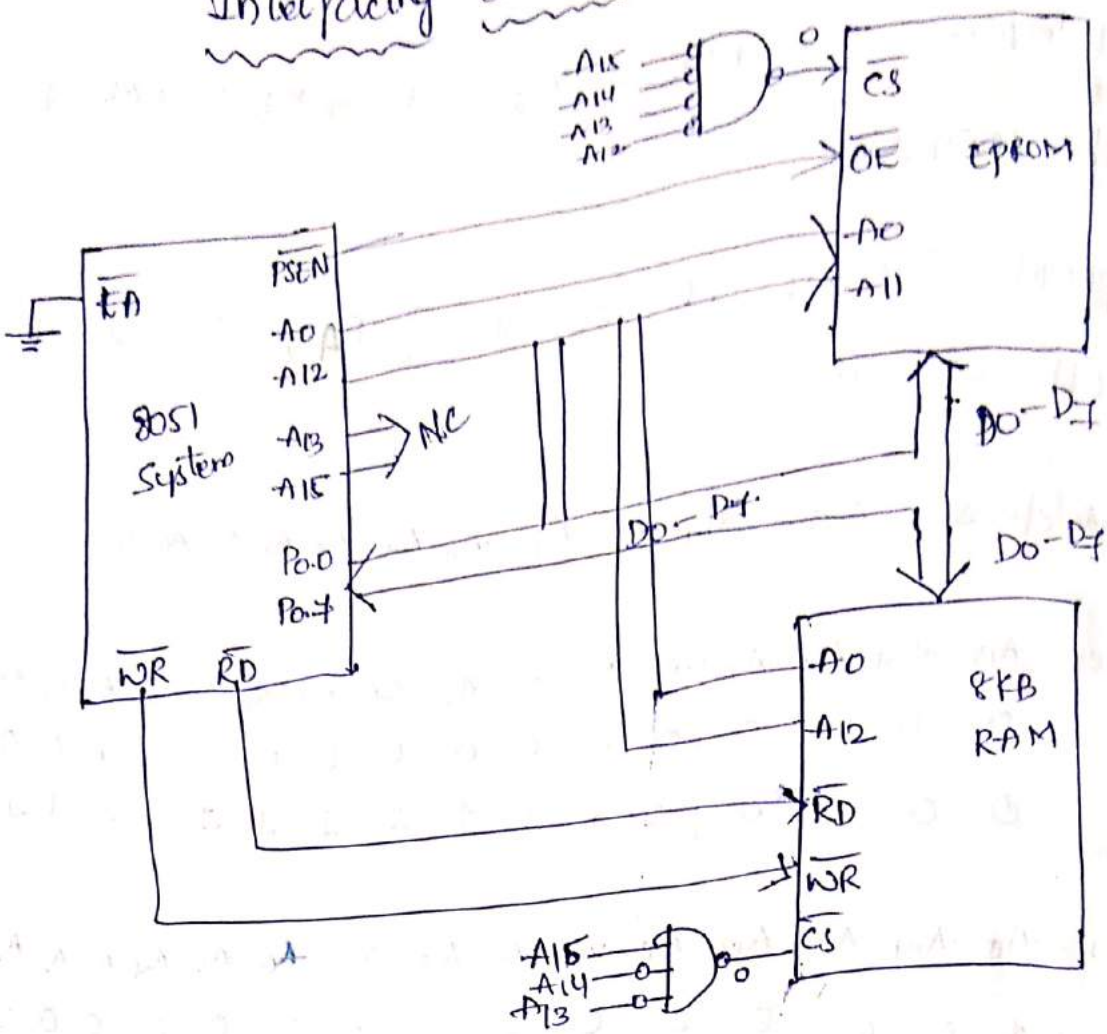
Hex address	A_{15}	A_{14}	A_{13}	A_{12}	A_{11}	A_{10}	A_9	A_8	A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0	Size
0000H	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4KB
0FFFH	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	

RAM address	A_{15}	A_{14}	A_{13}	A_{12}	A_{11}	A_{10}	A_9	A_8	A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0	Size
8000H	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	8KB
9FFFH	1	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	





Interfacing circuit



→ EPROM is of size 4K is given by $4K = 2^2 \times 2^{10}$

$= 2^{12}$

[A0 – A11]

→ RAM is of size 8K & it requires 13 address lines given by $8K = 2^3 \times 2^{10}$

$= 2^{13}$

[A0 – A12]

→ The chip select of RAM & EPROM should be zero which is generated by NAND gate

→ $\overline{EA}$ should be zero to access external memory

→ $\overline{OE}$ of EPROM is connected to $\overline{PSEN}$ of μC. (because it is read only memory).

→ $A_0 - A_{11}$ of EPROM is connected to $A_0 - A_{11}$ of μC

→ $\overline{CS}$ of EPROM is zero.

→ $A_0 - A_{12}$ of RAM is connected to $A_0 - A_{12}$ of μC respectively.

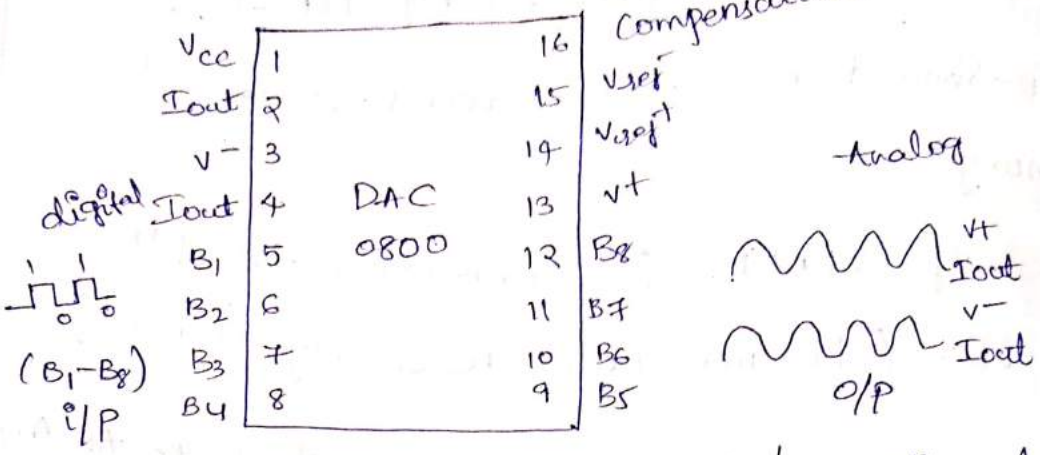
→ $\overline{RD}$ & $\overline{WR}$ of RAM I/s are connected $\overline{RD}$ & $\overline{WR}$ of μC respectively.

→ $\overline{CS}$ of RAM is zero.

→ The 8-bit data lines ($D_0 - D_7$) of RAM & EPROM are connected to port '0' respectively. (It is multiplexed Address data bus given by $AD_0 - AD_7$).

Note: A_{13}, A_{14}, A_{15} of μC are given NC.

DAC Interfacing:

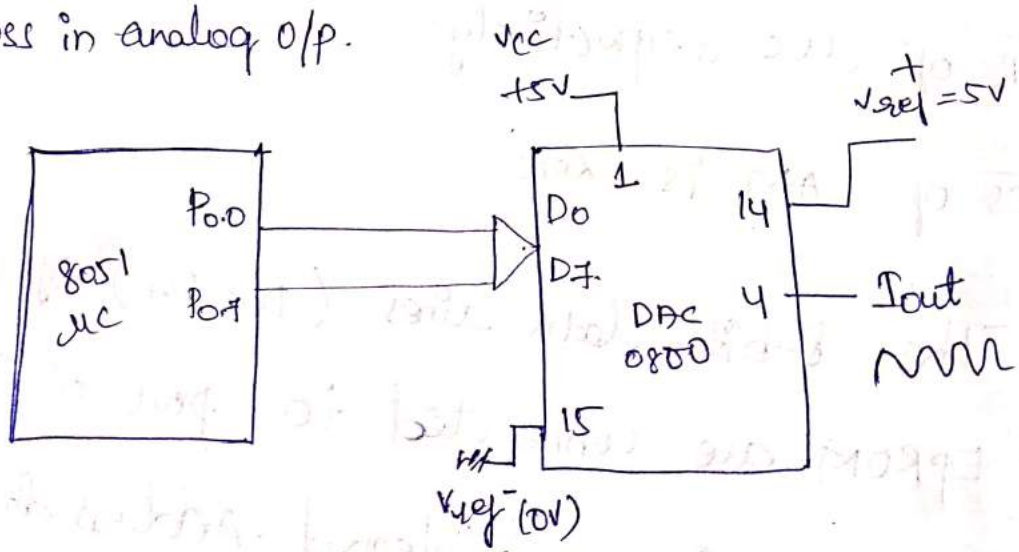


→ It has got two two analog o/p's given by V^+ & V^- whose currents are Iout & Iout

→ It has got 8-bit digital output given by B_1 to B_8

→ It has got Supply voltage of Vcc & two reference voltages V_{ref}^+ & V_{ref}^-.

→ It has got compensation Resistor to avoid any loss in analog o/p.



Interfacing circuit

→ The digital data from port 0 ($P_{0.0}$ to $P_{0.7}$) is sent to DAC for DAC conversion from B_1 to B_8 respectively.

→ Digital data is converted into Analog voltage which is shown by Iout at pin.no:4

→ $V_{cc} = 5V$, $V_{ref}^- = 0V$, $V_{ref}^+ = 5V$

Interface DAC with 8051 & write ALP to generate (i) triangular wave with from 0 to 3V with frequency 100Hz.

(ii) Sawtooth wave of 0–5V with frequency 500Hz.

(i) Since reference voltage is from 0–5V the maximum voltage swing is 5V.

→ Resolution of DAC is given by $= \frac{5}{2^8 - 1} = \frac{5}{255}$

$= 0.0196V$

$= 19.6mV$

→ Binary equivalent of '3V' is equal to $3/19.6mV$

$= 153 = 99H$

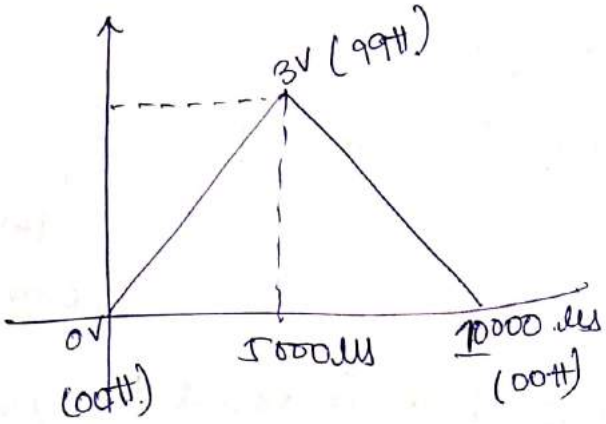
$f = 100Hz$

$T = \frac{1}{100} = 10^{-2}\ sec = 10^4\ \mu sec$

Program:

```
        Org 0000H
Again : MOV A, #00H
continue  MOV P0, A
          INC A
          CJNE A, #99H, continue

continue1 : DEC A
          MOV P0, A
          CJNE A, #00H, continue1
          SJMP Again
          end.
```



→ originate to the Starting address.

→ MOVe digital value 8' 0's to Accumlator

→ From Accumlator, move to port zero

→ And from port zero to

→ Increment the Accumulator value. (01H)

→ Compare this incremented value with final value 99H & if it is not equal jump to the label continue & repeat the same process until it reaches 99H (positive ramp is generated).

→ Present value is 99H & after decrementing we get 98H

→ compare this decremented value with final value. 00H & if it is not equal jump to the label continue 1 & repeat the same process until it reaches 00H (-ve ramp is generated.

→ By this one triangular wave is generated.

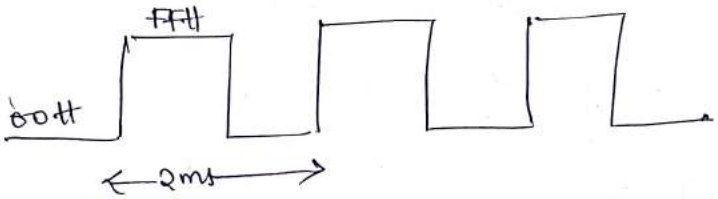
→ Short Jump to the label Again to generate continuous triangular wave.

(ii)

```
        Org  0000H
        MOV  A, 00H
Again : MOV  P0, A
        CPL  A
        SJMP Again
        END
```

$f = 500Hz$

$T = 1/500 = 2ms$



→ Originate to starting adress

→ Digital bits of all zero's are moved to accumulator

→ The contents of accumulator are moved to port zero & from their they are moved to DAC

→ complement the Accumulator values

→ To generate continuous Square wave we have to short jump to the label Again

Write a Program to display the message 'HELLO' on LCD

```
        ORG 3000H
MY COM: DB 38H, 0EH, 01, 06, 85H, 0 command and & NULL
MY data: DB "Hello", 0
        End
```

$P_{0.0} - P_{0.7} = D_0 - D_7$ $P_{2.0} = RS$ $P_{2.1} = R/W$ $P_{2.2} = E$

```
        ORG 0000H
        MOV DPTR, #Mycom
C1 : CLR A
        MOVC A, @A+DPTR
        ACALL COMNwrt
        Acall delay
        INC DPTR
        JZ Send_dat
        SJMP C-1
COMNwrt : Send command to LCD
        MOV P0,A
        CLR P2.0
        CLR P2.1
        SETB P2.2
        ACALL delay
        CLR P2.2
        RET
```

Send_dat Send data to LCD

```
MOV DPTR, # My data
D1: CLR A
MOVC A, @A+DPTR
ACall datawrt
Acall delay
INC DPTR
JZ Again
SJMP D1
Again: SJMP Again
```

Datawrt: write data to LCD

```
MOV P0, A
SETB P2.0
CLR P2.1
SETB P2.2
Acall delay
CLR P2.2
RET
```

```
DELAY:
        MOV R4, #FFH
Here: DJNZ R4, Here; stay until R4 becomes zero
        RET
```

Note:

→ In my command & my data labels '0' is inserted intentionally to come out of the loop

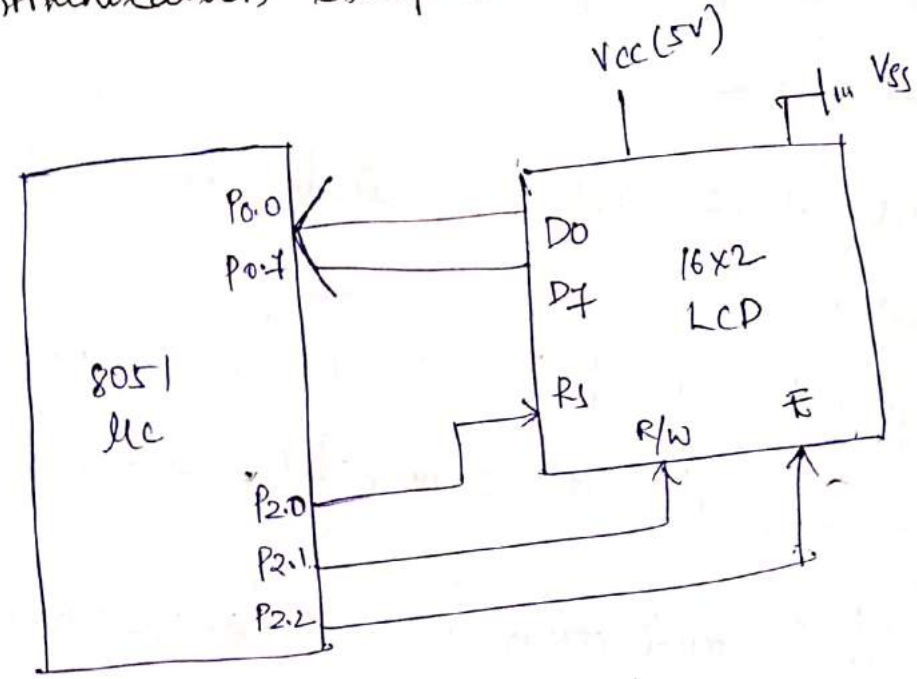
→ For command mode 1, RS=0 2, R/W=0 3, E=1, 4, E=0

→ for data mode 1, RS=1, 2, R/W=0, 3, E=1, 4, E=0

→ Note:

whenever DPTR is used while moving the data use MOVC instead of MOV

communication ~~Interfaces~~



Interfacing ckt

Communication Interface:-

There are two types of communication interface given by 1, Serial communication Interface
2, Parallel communication interface.

1) Serial communication Interface:-

1) In this communication data is transmitted serially one bit after another & while transmission one start bit two stop bits & one parity bits is used for transmission.

→ These 12 bits together are called as a transmission frame.

→ start bit is used to indicate the starting of the frame.

→ Stop bit is used to indicate the end of the frame & parity bit is used for error detection.

→ Serial Communication is similar to Asynchronous data transfer which requires no. of clock signals equals to no. of bits

→ The o/p of one device acts as clock for another device in asynchronous mode.

→ It is slow in operation when compared to parallel communication.

2, Parallel communication:

In this Communication, the bits are transmitted simultaneously & one clock is given enough to initiate the task.

→ It is Similar to Synchronous Communication.

→ It is fast in operation.

→ Error detection is not possible in parallel communication.

→ Modes of communication:

There are three types of communications given by

(1) Simplex communication:

In this Communication transmission takes place only in one direction.

ex; TV, FM radio etc

(2) Half duplex:

In this Communication transmission takes place

in both directions but not simultaneously.

ex: walkie-talkie.

3) Full duplex communication: In this transmission takes place in both directions simultaneously

ex: mobile.

Communication interfaces:-

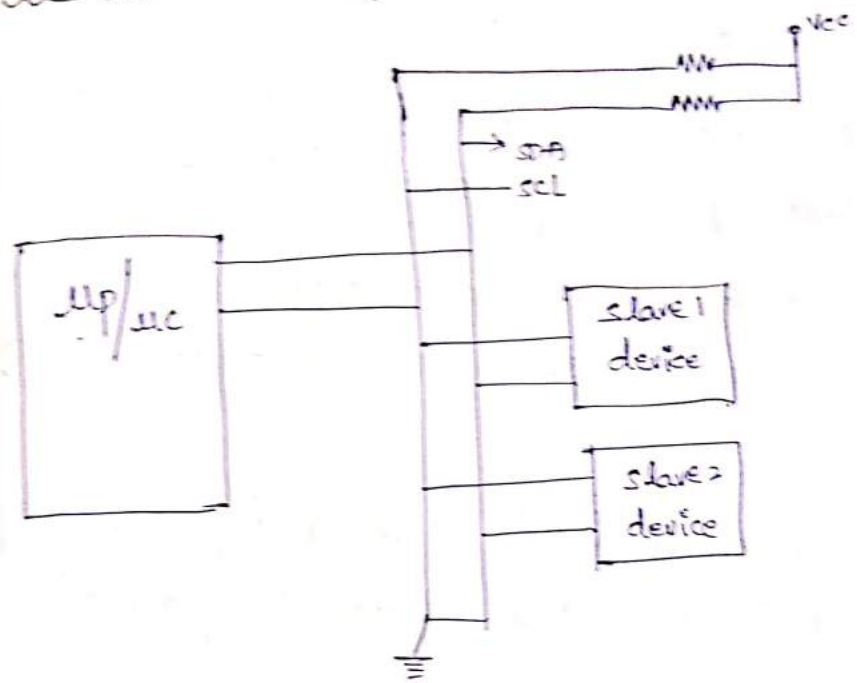
There are two types of communication interfaces given by 1) on-board communication (device board level)

In this communication various peripherals in the sub system of the embeded products (or) communicated by means of various interfaces like I^2C, SPI and UART.

2) External communication:

In this communication the subsystem of the embeded products communicate with the external world by means of interfaces like RS232, USB etc.

I²C Bus Interfacing :-



→ It is a bidirectional Synchronous half duplex communication interface used to communicate various peripherals in the system.

→ It is a two wire Serial bus interface.

→ It was developed by philips Semiconductor in the year 1980 to perform communication between processor and peripheral chips in TV.

→ It has got two lines given by SCL (synchronous clock) & SDA (Serial data).

→ In this communication there is only one master

& it can have n- no. of slaves & it follows master slave communication.

→ The communication is initiated by master by making SCL high & SDA low.

→ The master verifies the address of the slave by means of 7-bit or 10-bit address assigned to slave.

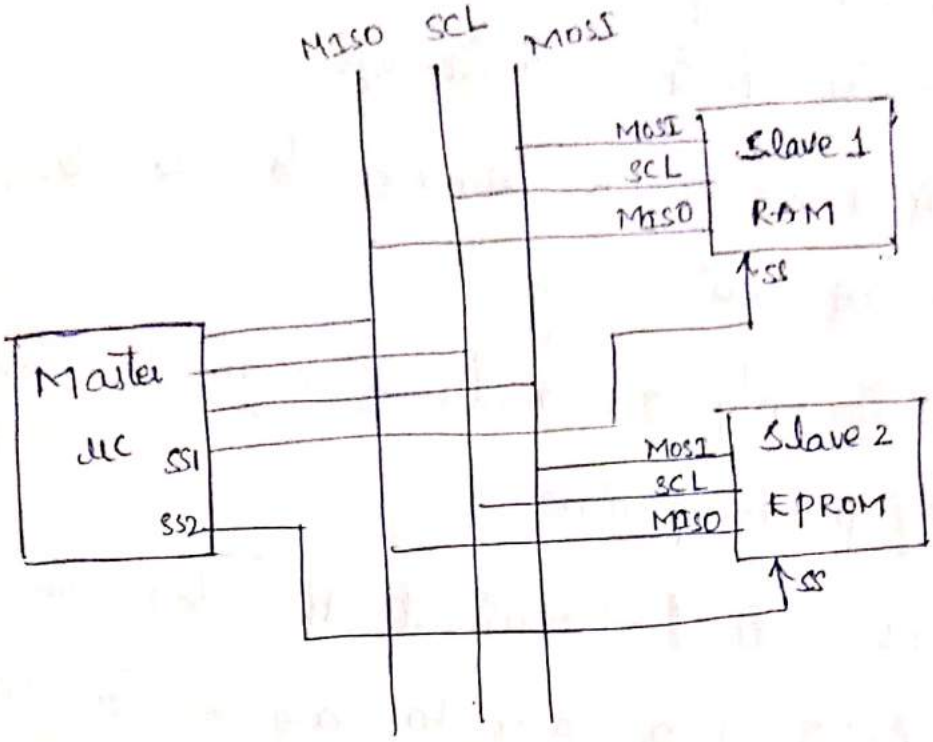
→ The communication is achieved by means of open drain (or) open collector configuration using 2.2 kΩ pull up resistors.

→ When master detects the address of the slave & after recieving acknowledgement it starts transmitting the data through SDA line.

→ Various baud rates are supported by this communication interface given by 200kbps (1st gen), 400kbps, 3.4 Mbps (present gen) (present gen).

→ The Communication is terminated by making SCL low & SDA high.

Serial Peripheral Interface:



SPI Bus Interfacing

→ It is a bidirectional full duplex Synchron-ous Serial communication.

→ It has got 4 lines given by MOSI (Master Out ~~Slaving~~ Slave In) → Master is transmitting data to Slave.

MISO (Master in Slave Out) → Slave is transmitting to master

SCL → Synchronous clock

SS → Slave Select

→ It follows master Slave Communication in which master 1st identifies which Slave wants

to communicate with it.

→ After recieving the acknowledgement from the slave master starts communicating.

→ At times master can become slave & slave can become master.

→ There is only one master & 'n' no. of slaves at any point of time

→ The SCL is used to initiate the data transmission either from master to slave (or) from slave to master.

→ Internally an 8-bit shift register is used from which the data is transmitted.

→ It comprises of two registers given by; control register :

This register holds the information of clock signal, baud rate & slave rate. The rate at which the master switches one slave to the another for transmission purpose is called slave rate.

2, Status register: It contains the information of status of data transmitted (or) recieved.

→ The master identifies the slave by the slave select

→ The draw back of this SPI is it doesnot support multiple acknowledgement signals.

UART:

Every microcontroller has got an inbuilt UART which is used for Serial communication & at port 3.0 & 3.1 given by TXD & RXD.

→ The Output of one device will act as clock to the other device.

→ Along with the 8-bit of ASCII value of the character one start bit, one parity bit & two stop bits are appended for Serial communication. This is also known as transmission frame.

→ Start bit is used to indicate the Starting of the frame.

→ Parity bit is used for error detection

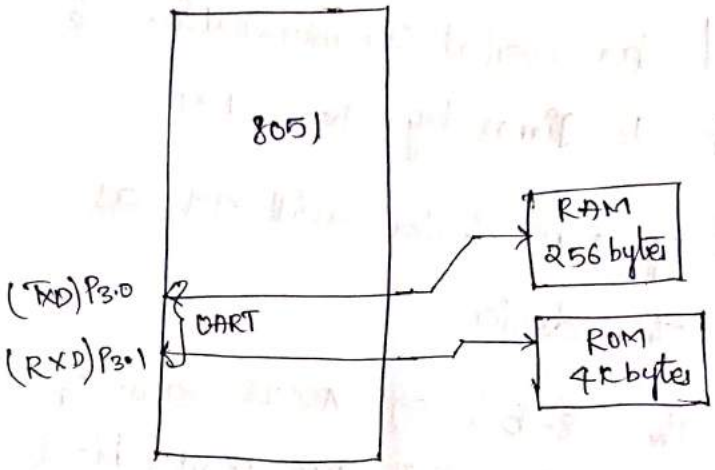
→ Stop bit is used to ending of the frame.

→ Two stop bits are used for more authentication

& Security.

→ UART Supports network handshaking Signals for controlling the Serial data flow.

→ National Semiconductors has developed first UART by the name 8250 for Serial communicatio



External Communication Interface:

There are two types of External communication interfaces. They are:

(1) RS232:

1. It is available in 9-pin & 25-pin configuration.
2. It is used to interface the computer System

with the peripheral.

→ It was 1st developed in the year 1960 by EIA (Electronics industry association).

→ RS Stands for Recomended Stardard & it is a full duplex wired asynchronous Serial communication.

→ RS uses two logics given by logic 0 b/n 3-25 V & it is termed as Space (OFF) & logic 1 b/n -3 to -25 V & it is termed as mark. (ON)

→ RS-232 has got two ends given by DTE (Data terminal equipment) & DCE (Data communication equipment.

→ RS-232 Supports different baud rates of 20kbps, 300bps, 9,600 bps

→ The range of Communication of RS-232 is 5 feet distance.

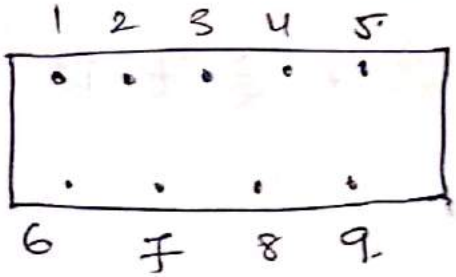
→ RS-422 is enhanced Version of RS-232 which is a Multi dropped Communication & Supports 400 feet distance where as RS-232 is one-one

Communication. RS-422 acts as a modem for longer distance communication & it Supports 10 devices.

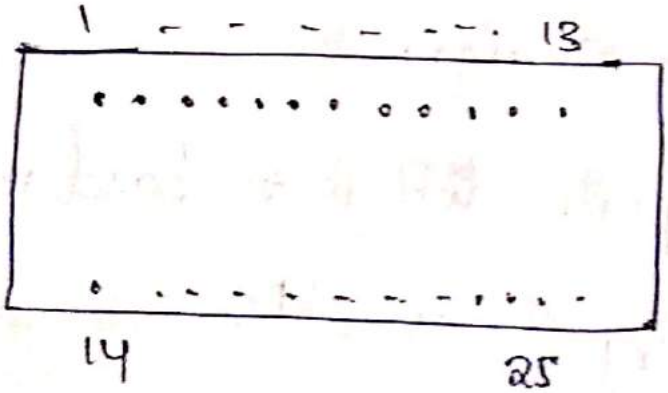
→ RS-485 is enhanced version of RS-422 which Supports 32 devices.

→ The pin configuration of RS-232 is 9 pin & 25 pin is given by

9 pin RS-232



25pin RS-232



Pin configuration of RS-232:

Pin no 1-Pin RS-232	Pin no 25 Pin RS-232	Description
3	2	Transmitt serial data
2	3	Recieve Serial data
7	4	Request to send
8	5	Clear to send
6	6	Data set ready
5	7	Ground
1	8	Data career detect
4	20	Data terminal Ready
9	22	Ring indicator
—	1	frame ground
—	12	Secondary data career detect
—	13	secondary clear to send
—	14	Secondary transmitt data
—	15	Timing clock (transmitter)
—	16	Secondary Recieve data
—	17	Rec Timing clk (Recievers)
—	18	Secondary request to send
—	19	Signal Quality detector
—	20	NO connection
—	21	"
—	22	"
—	23	"
—	24	"
—	25	"

Note:1 Ring indicator is normally used for indicating the incoming call on telephone line.

Note 2: The Sequence of operations in RS-232 is given by 1, RTS
2, DTR
3, CTS
4, DSR
5, TXD
6, TC
7, RXD
8, RC

USB: (Universal Synchronous Bus):

→ USB is a wired high Speed Serial bus for data Communication.

→ The first version was developed in the year 1995 by group consisting of INTEL, Microsoft, IBM, Compaque & digital Northern Telecom

→ USB has a Star topology & it can Support 127 devices

→ USB transmits data in packets & the Communication is initiated by host USB to other devices using OHCI i.e., Openhost control interface &

UHCI i.e., Universal host control Interface.

→ In the topology we can have n.no.of hosts but only one host among them acts as master

→ In USB there are two types of connectors given by 1, type-A connector 2, type-B connector.

→ In type-A connector the communication takes place from host to Slave device.

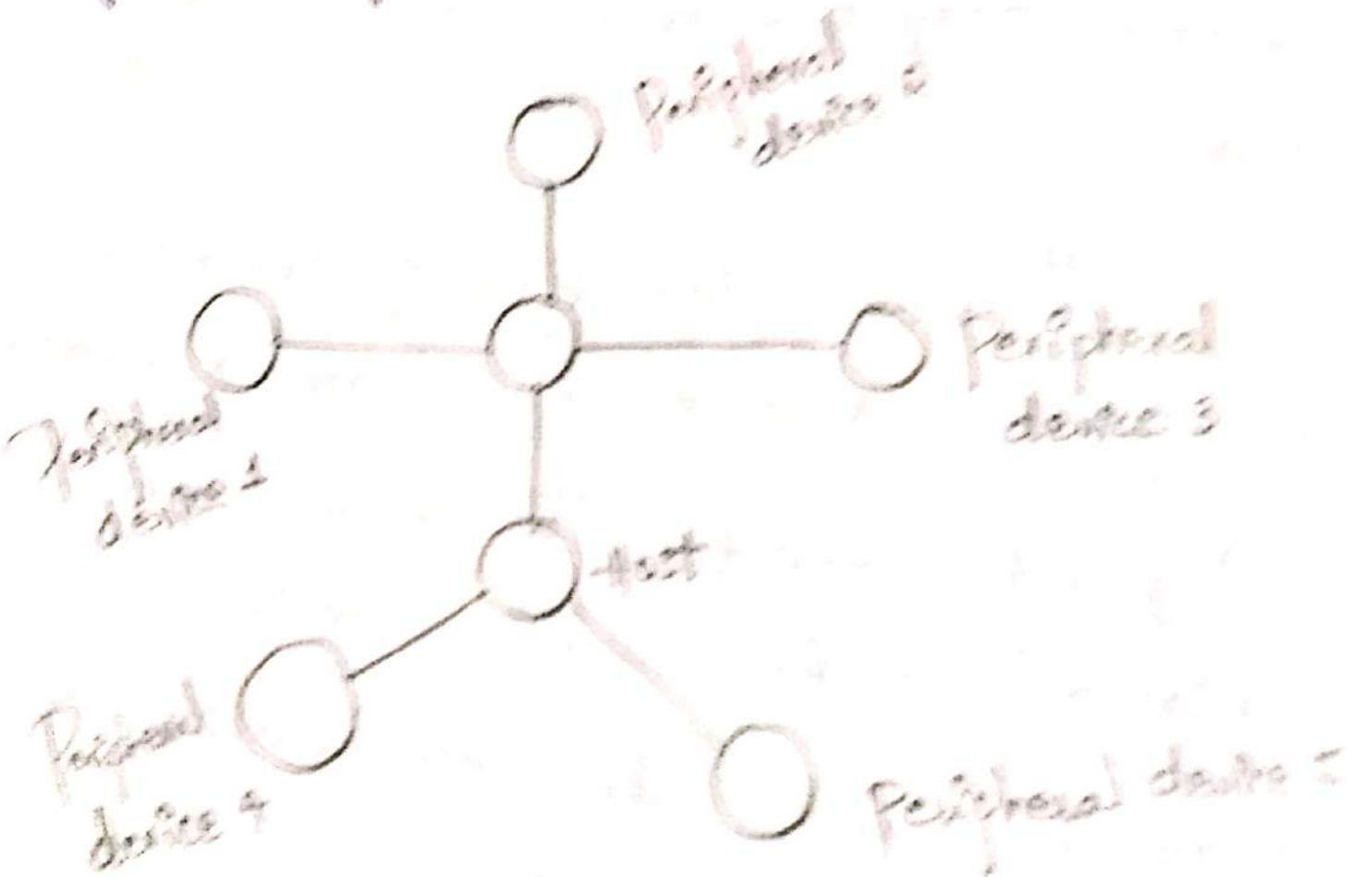
→ In type-B connector the communication takes place from Slave to host

→ It has got four pins given by

Pin.no	Name	Description
1	V_{BUS}	carries power of 5v
2	D	Differential data carrier line
3	D^+	Differential data carrier line
4	GND	Ground

→ Each USB has got two products 1, PID (product ID), 2, VID (vendor ID.)

Note: These two Bits have to be configured by installing the selected software



Use device connection topology.

4) Explain interfacing of 8051 with LCD to display message "HELLO".

5, Differentiate between Serial communication & parallel communication

6, Explain modes of Communication

7, Explain I^2C

8) Explain SPI

9, Explain UART

10) Explain RS-232

11, Explain USB.

1/3/19

UNIT-IV

ARM processors

Advanced Risk Machine (ARM) Processors

developed in the year 1990's 'Acron' Com

→ There are various Series of ARM processors given by ARM 1 to ARM 11.

→ ARM '7' is also known as LPC 2148 which has got 3 stage pipe line given by 1, fetch 2, decode 3, execute.

→ ARM '9' is a '5' stage pipe line given by 1, fetch 2, decode 3, data buffer 4, data write 5, execute.

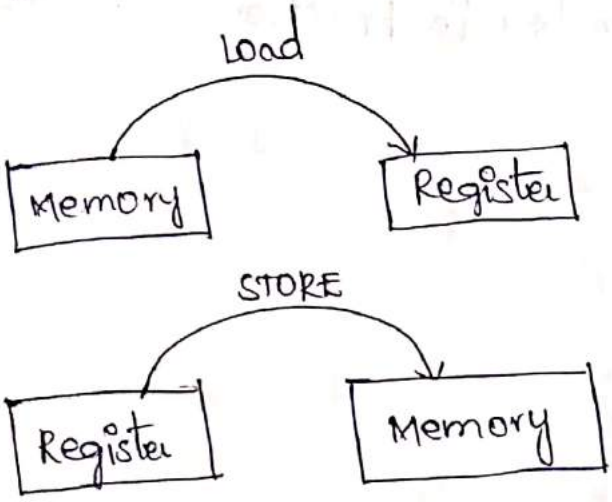
→ ARM is 32 bit micro controller which has 32 address lines & 32 data lines.

→ It can also Operate as a 16-bit microcontroll which is known by ~~THUMB~~ 'THUMB Instruction

→ The Speed of execution in ARM processor

is very fast, its code effeciency is very good. & also it has low power consumption.

→ ARM processor is a load & store architecture in which the contents from one memory to another memory are not directly moved but they are moved from cpu registers.



→ ARM processor has got an inbuilt JTAG & "embedded ICE" codes for downloading & debugging the programs.

→ ARM instructions are executed in a single cycle.

→ ARM processors are used in high end applications like wireless telecommunications, portable computers, Smart cards, data communications etc.

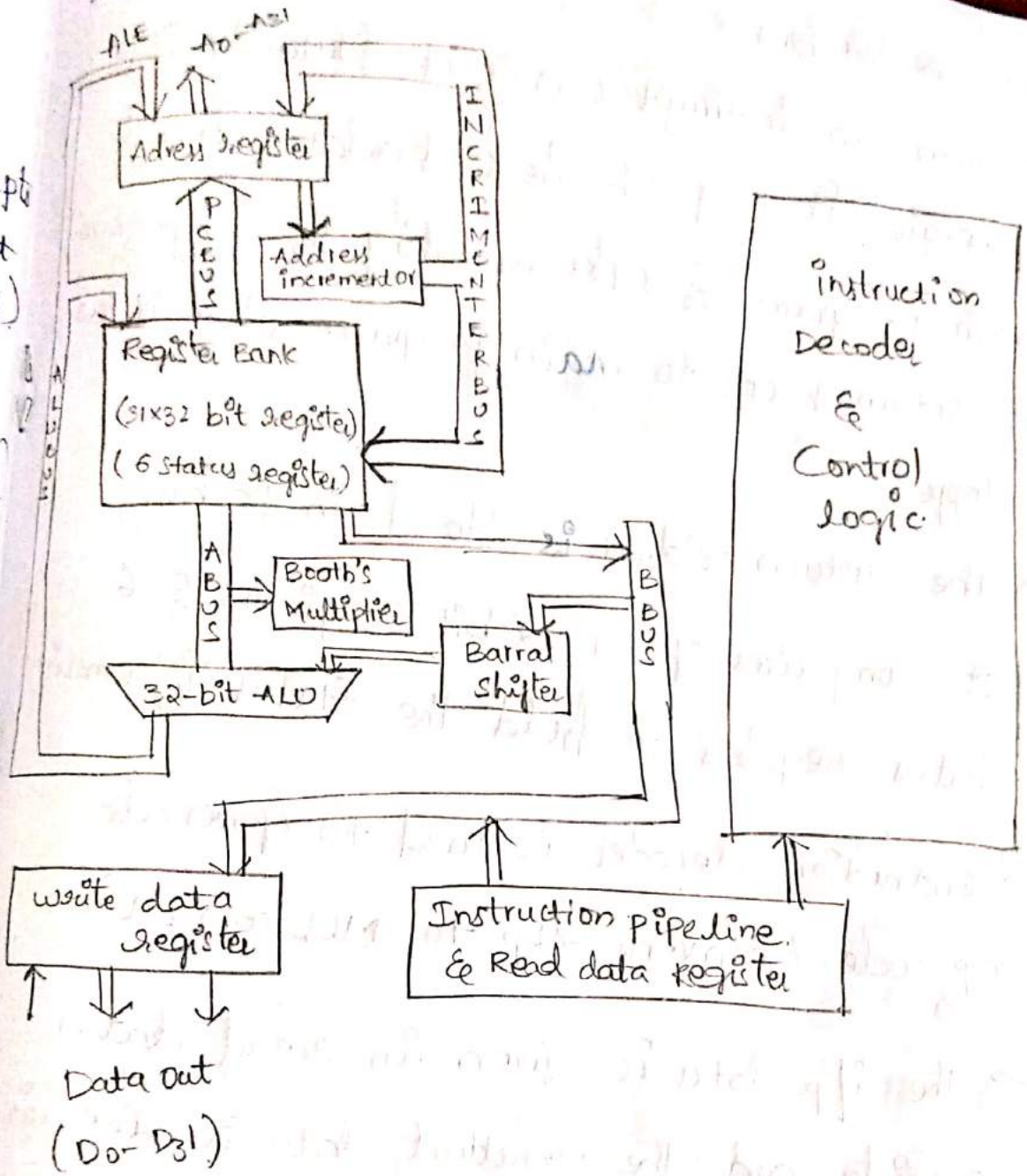
where as microcontroller are used in low end applications like home automachine, home security etc.

→ ARM processors are based on RISC Concept :

(Reduced instruction set computing)

where as microprocessors & micro controllers uses CISC (complex instruction set computing

ARM Processor -Architecture:



→ ARM procesor has 32 address lines & 32 data lines given by A0-A31 & D0-D31 hence it is called as 32-bit micro controller.

→ The starting address of the program is fetched by address register and the next address is obtained by address incrementor by means of

Incrementor bus.

→ When an interrupt (or) call statement is generated it jumps to the call address from the main program & after executing the subprogram it return backs to main program where it has stopped.

→ The return address is stored in pc bus.

→ It comprises of 31-32 bit registers & 6 status registers to hold the status information.

→ Instruction decoder is used to generate op-code. (converts ALP to MLL (or) LLL)

→ The i/p data is given in read data register and the resultant data is given in write data register.

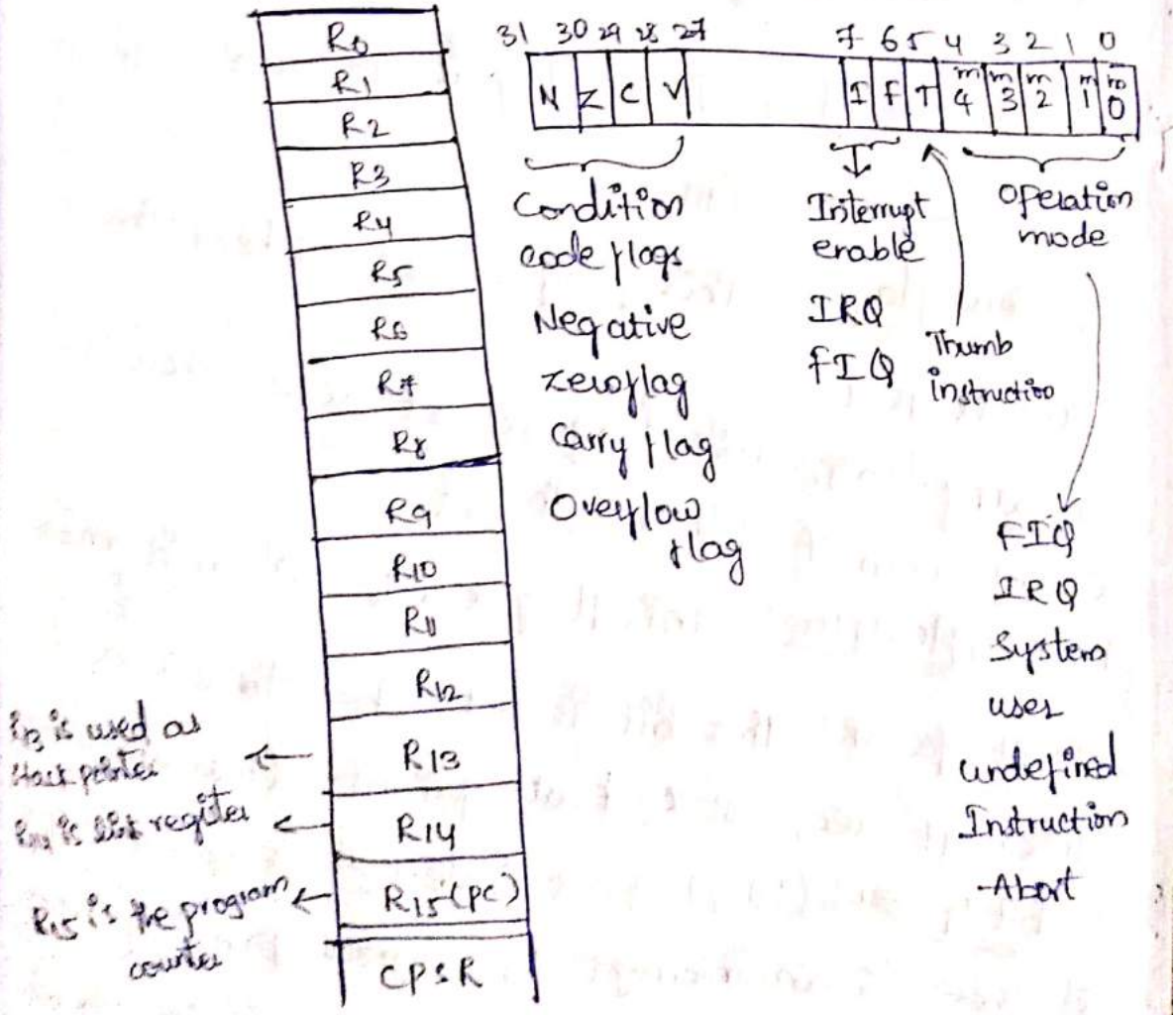
→ A Bus & B Bus are used to share the work among them by means of Booth's multiplier & Barral shifter respectively before they are applied to ALU

→ Barral Shifter, Booth's Multiplier & ALU together work as combinational ckt.

Note: 1) Booth's multiplier & Barral shifter act as preprocessing device.

2) Booth's multiplier is used to perform multiplications & Barral shifter is used to perform Rotate & shift operations

Register Model of ARM7:-



Current program Status Register & flags

→ It consists 15-general purpose register a pc.

→ Apart from it R_{13} acts as Stack pointer which points to the top of the Stack.

→ R_{14} acts as link register which has return address of Sub program.

→ R_{15} acts as pc which comprises of return address data.

→ ARM processor comprises a status register called as cpsr which saves the current program.

→ It comprises of 4 conditional flags given by

1) Negative flag: This flag is set when result is -ve number (MSB=1)

2) Zero flag:- This flag is set when the result is '0'.

3) carry flag: This flag is set when there is a carry from 4th bit to 8th bit.

4) overflow flag: This flag exists when it has 2^{32}.

7th bit of cpsr: This bit is set when there is a interrupt request of least priority (maskable Interrupt)

6th bit of cpsr: (FIQ) fast interrupt request is set if there is an interrupt of highest priority (NMI)

→ 5th bit of cpsr: If T=0 it acts as 16-bit (uc thumb instruction). If T=1 32 bit Mc which is arm controller

Operating modes of ARM:

→ ARM processor has total of 7- operating modes.

1) System mode:

This mode is used to run the application code.

→ cpsr is used to store the current program.

2) User mode:

→ This mode is also similar to System mode to run the application code.

3) FIQ Mode:

→ This mode is used to enable the fast interrupt request which has highest priority (NMI) while executing the application code.

→ cpsr is used to store the current program.
& spsr is used to store the saved program (or) previous program.

4) IRQ Mode:

→ This mode is used to enable the interrupt request of least priority (MI) while executing the application code.

→ cpsr & spsr are used to store the current & previous program respectively.

5) Supervision Mode:

→ This mode is used to access hardware (or)

OS calls

→ CPSR & SPSR are used to store current & Previous program respectively.

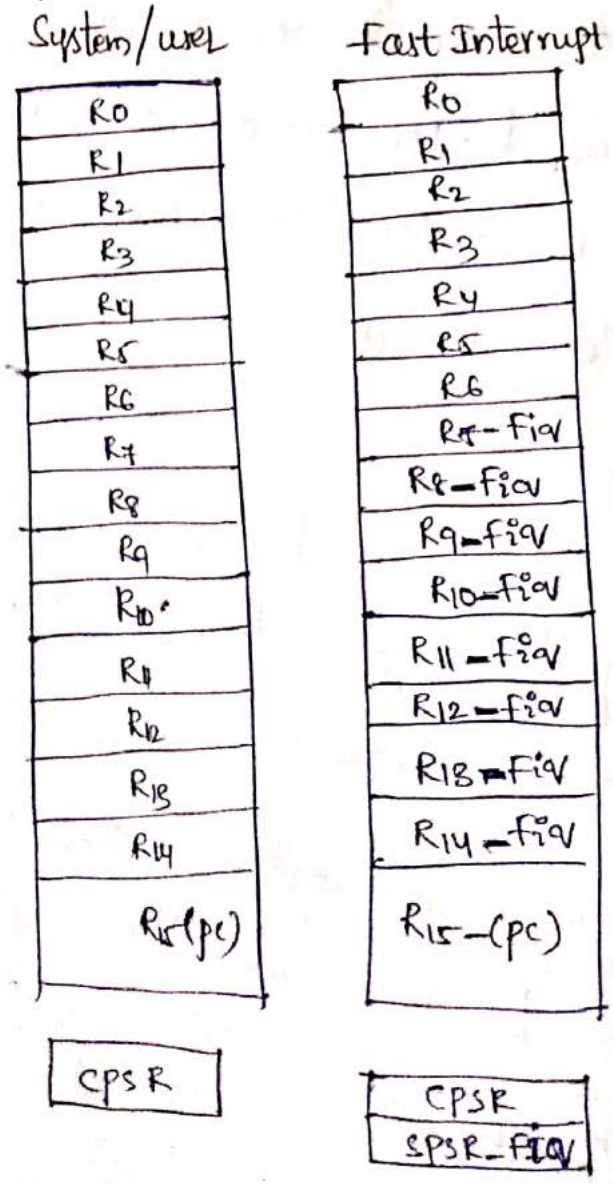
(c) Abort:

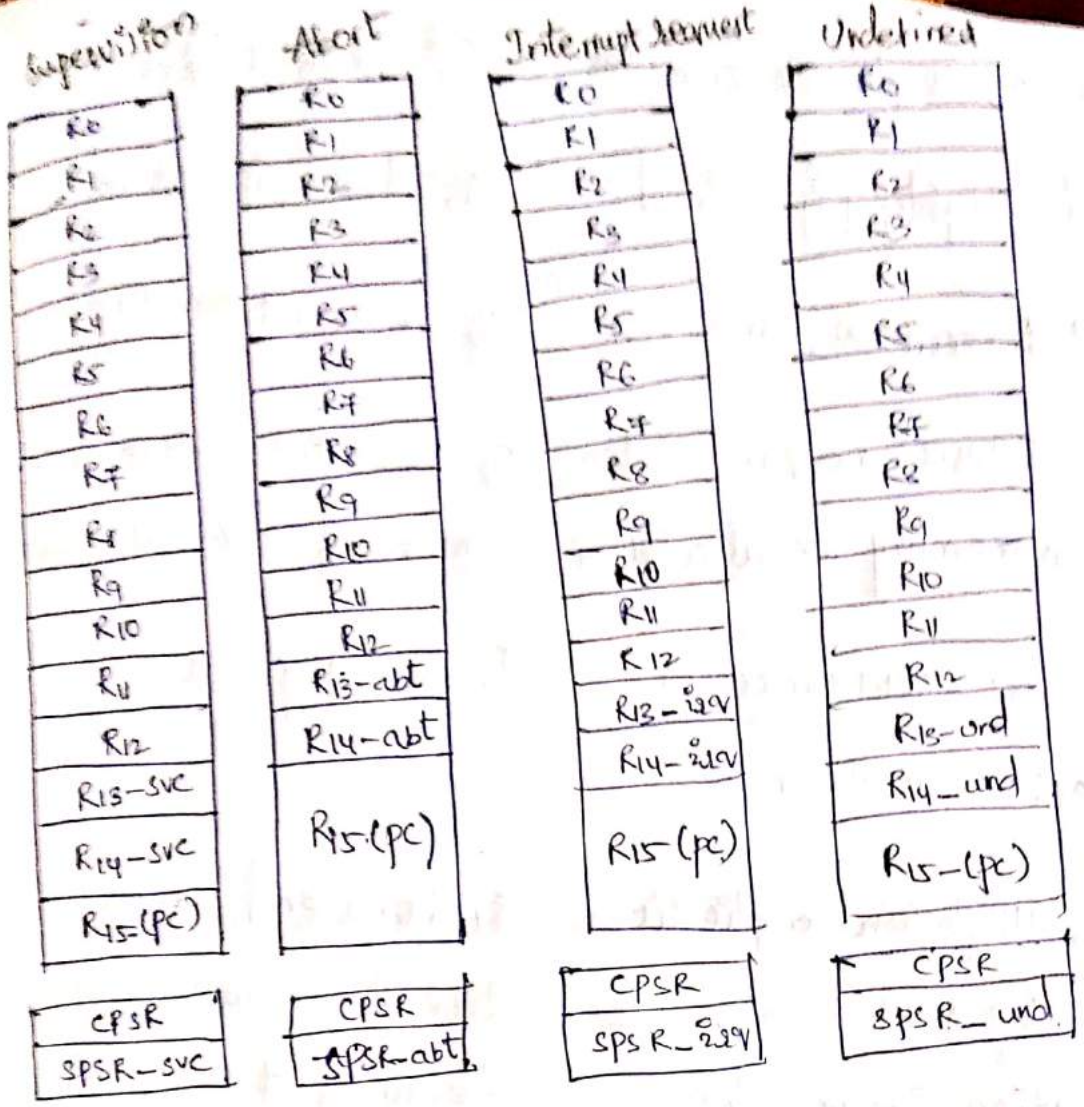
→ If an instruction (or) data is fetched from invalid memory an abort exception will be gen

(d) undefined Instruction mode:

→ If a fetched opcode is not an ARM instruct & undefined exception will be generated.

→ CPSR & SPSR are used to store current &





Difference between RISC & CISC processors:

RISC	CISC
1, Simple instructions with one clock cycle (Average clock per instruction, CPI is 1.5).	1, Complex instructions with multiple clock. Avg CPI is 2.
2, Few instructions (very primitive).	2, More instructions (very complex)

3, Multiple Register set	3, Single Register set
4) Highly pipe lined	4) less pipe lined
5) Execution-time is less	5) Execution time is more
6) Doesnot require external memory for calculations	6) Require external memory for Calculations.
7) ex: ARM processor, COTEX processor	7) ex: µp, µc.
8) High end applications like Mobile communication, image-video applications, telecommunications etc.	8) Low end applications like Home automation, Security etc.

13/05/19

Pipe line:

→ ARM processor utilized pipe lining (or) multitasking concept and there by it speeds up the execution process with respective to various instructions

→ The Speed of the ARM processor is directly proportional to pipe lining that takes place in

ARM processor.

→ for example, Let ARM processor performs 3 instructions given by add, Sub, compare.

→ The processor fetches the address of the addition instruction. The other instructions like Subtraction, compare remains idle.

→ when the processor decodes the addition instruction fetching of address of subtraction takes place.

→ when processor executes the addition instruction Simultaneously decoding of subtraction instruction and fetching the address of compare instruction takes place.

→ Each instruction takes one cycle and to execute all these 3- instructions a total of 5 cycles is required.

→ ARM 7 executes 3-Stage pipe line given by fetching, decoding and executing.

→ ARM-9 processor executes 5-stage pipe line given fetching, decoding, data buffer, data write and executing.

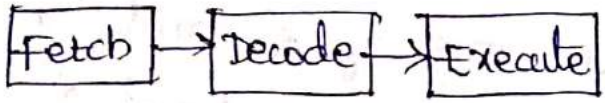
ARM-10 Executes 6-stage pipeline given by fetching, issue, decoding, data buffer, data write and execute.

Note:

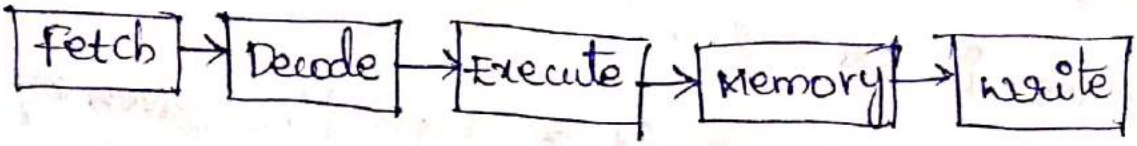
→ Th of ARM-9, ARM-10 Processor is very high when compared to ARM-7 Processor.

→ Ex: ARM-9 Processor is 13% high than ARM-7 processor and ARM-10 is 34% high when compared to ARM-7 processor.

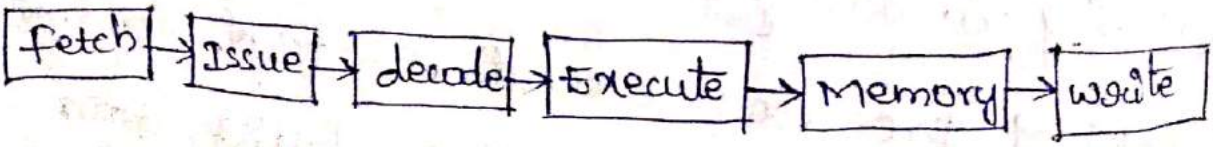
ARM-7

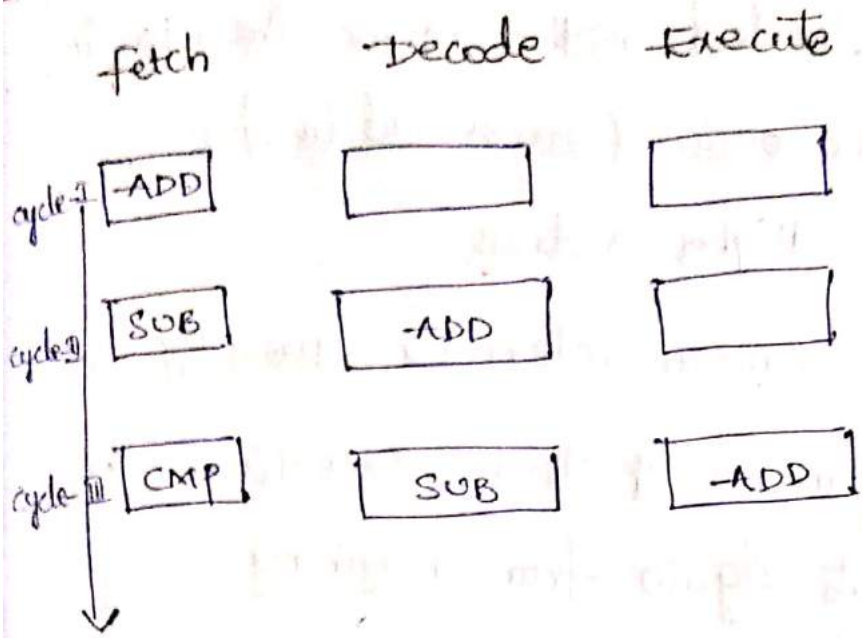


ARM-9



ARM-10





Pipe line Instruction Sequence.

Exceptions / Interrupts & Interrupt vector table.

→ Exceptions are the interrupts that are been performed by the processor other than the Operations.

→ The interrupts are executed by going to their respective interrupt Service routine address.

→ After the interrupt is executed they are returned back to the main program where these have Stoped.

→ All the interrupts of ARM processor are stored in a common place called interrupt vector table.

→ They are located either from the Starting address 0x0000 0000 (lower address) (or) from 0xffff 0000 (Higher address).

→ There are various interrupts given by

1) Reset: By means of these exception the Program starts again from begining.

2) Undefined: By means of this exception ARM Processor is unable to identify opcode.

3) Software Interrupt: By means of this exception Programmer himself will introduce the interrupt for debugging purpose.

4) (i) Prefetch Abort: By means of this exception invalid address is selected.

(ii) Data Abort: By means of this exception invalid data Search is done.

5) Reserved: By means of these exception Special purpose interrupts are generated.

6) Interrupt request:- By means of this exception least poriority (or) Maskable interrupts are

the Starting
ss) (or) from

en by

on the

ARM

exception
e

xception

tion

tion

generated

* Fast interrupt request (FIRQ): By means of these exception highest priority (or) Non-maskable interrupts are generated.

Note: Each interrupt take 4x8=32 bits of address and thus for a total of 8x32=256 address space is allocated for IVT

IVT Exception/Interrupt	Shortend	Address
Reset & Programmer	Reset	0x00000000 0x00000003
undefined & op-code	UNDEF	0x00000004 0x00000007
Software interrupts & PS	SWI	0x00000008 0x0000000B
Prefetch Abort } Invalid search	PABT	0x0000000C 0x0000000F
Data Abort } Invalid search	DABT	0x00000010 0x00000013
Reserved		0x00000014 0x00000017
Interrupt Request	IRQ	0x00000018 0x0000001B
Fast Interrupt Request	FIRQ	0x0000001C 0x0000001F

ARM Instruction SET:

→ ARM processor are responsible for condition -al execution Statements which there by reduces the no. of instructions, computation time period & increases the code efficiency.

→ ARM Processor when acting as 16 bit uc is called as thumb instruction & the conditional execution in 'IT' is denoted by 'IT' instruction.

→ The presence of parallel Shifter in barrel shifter in ARM processor is responsible for reducing the instructions.

→ TO multiply two numbers & to store the result as taken only one instruction.

→ The Standard Syntax used in ARM processor as MNEMONIC {S}, {condition} {Rd}, operand1 & Operand 2.

MNEMONIC → Short name of the instruction

{S}

→ If S is Specified, the conditional flags are updated.

→ {condition} - condition to be met inorder for execution.

→ {Rd} - Destination register the results is stored.

→ Operand 1 - It is the 1st operand which can be either a register (or) an immediate value.

→ operand 2 - It is the 2nd operand which can be immediate value (or) a register with optional shift.

→ few instructions are given by

ADD, R_0, R_1, R_2

→ The contents of R_1 & R_2 are added & stored in R_0 ADD R_0, R_1, #2

→ The contents of R_1 is added with immediate value 2 & the result is stored in R_0

MOV R_0, R_1, LSL #1

→ The content of R_1 is logically shifted leftwards by 1 bit & the result is stored in R_0

→ If $R_1 = 2$

R_1 is multiplied with 2 after LSL operation & result 4 is stored in R_0.

Note: The whole stream are moved leftwards by 1 bit which is equivalent to multiply by 2.

Most often used -ARM instructions:

1, MOV - Move data
2, MVN - Move & negate
3, ADD - Addition
4, SUB - Subtraction
5, MUL - Multiplication
6, LSL - Logical shift left
7, ASR - Arithemetic Shift right
8, ROR - Rotate right without carry
9, cmp - compare
10, AND - Bitwise AND
11, OR - Bitwise OR
12, EOR - Bitwise exclusive OR
13, LDR - Load data (similar to write operation)
14, STR - Store data (similar to read operation)
15, LDM - load multiple.

16, STM -
17, Push -
18, POP -
19, B -
20, BL -
21, BX -
22, BLX -
23, SWI -

Data proc

→ Manipu
like MO
multiply
-gory.

→ ARM p
for prep

→ The ca
result o
bit shift

→ N-

(or) MSB

16, STM — Store multiple.

17, Push — push on stack

18, Pop — pop on stack

19, B — Branch

20, BL — Branch with link

21, BX — Branch with exchange

22, BLX — Branch with link & exchange

23, SWI — System call.

Data processing Instructions:

→ Manipulating data within the register like Move, Arithmetic, logical, compare and multiply instructions comes under this category.

→ ARM processor utilizes barral shifter for preprocessing operations.

→ The carry flag is set from the result of the barral shifter as the last bit shifted out.

→ N-flag is determined by the 31st bit (or) MSB bit.

→ Move Instructions:

MOV Move 32 bit value to the register

$R_d = N$

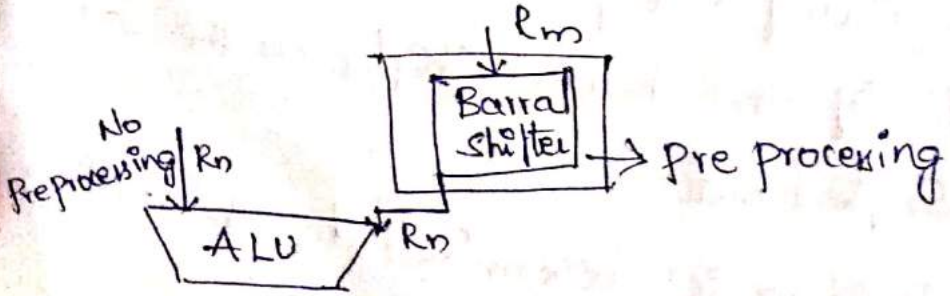
MVN Move the not of 32 bit value into the register.

$R_d = -N$

→ Preprocessing operations in barral shifter are Completed within one cycle.

→ If barral Shifter Shifts the logical bits by one bit is equivalent to multiplying by $2^1 = 2$ (2 bits $2^2 = 4$}, 3-bits $2^3 = 8$ ....)

→ If barral Shifter shifts the bits logically by right by one bit it is equivalent to dividing by $2^1 = 2$ (for 2-bits $2^2 = 4$, 3-bits $2^3 = 8$



MOV r4, r5 LSL #1

Previous data

r4 = 8

r5 = 5

present data

r5 = 5

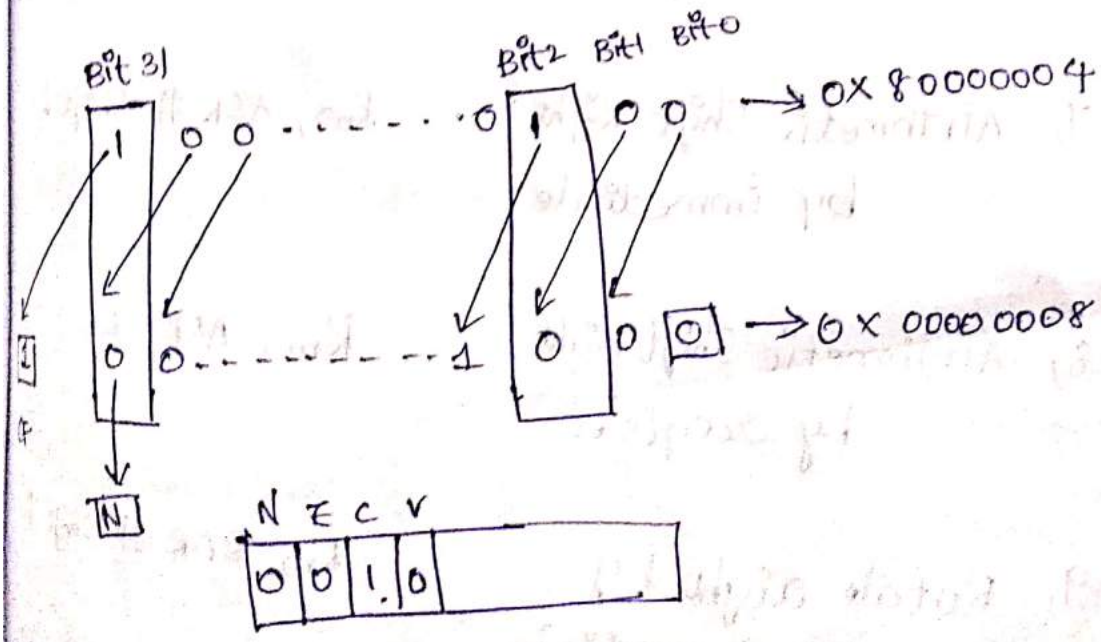
r4 = 20

4 → 0100

01000 → 8 → 4 × 2¹ = 8

CF N

Barral shifter operation: (LSL #1)



↘ Barral shift operation Syntax for data processing instructions.

Shift operation	Syntax
1) Immediate	#
2) Register	Rm
3) Logical shift left by immediate	Rm, LSL # shift_value
4) Logical shift left by Register	Rm, LSL Rs
5) Logical shift Right by immediate	Rm, LSR # shift_m
6) Logical shift Right with register	Rm, LSR Rs
7) Airthmetic shift Right by immediate	Rm, ASR # shift_m
8) Airthmetic shift Right by register	Rm, ASR Rs
9) Rotate right by immediate	Rm, ROR # shift_im
10) Rotate right by register	Rm, ROR Rs

Program Status Register instruction:-

→ There are two types of Status register given by CPSR & SPSR.

→ when there is a transfer of data either from CPSR (or) SPSR into a register in reverse direction it is called MRS instruction.

→ It is denoted by $R_d = PSR$

→ when there is a transfer of data from register to CPSR (or) SPSR it is called MSR instruction.

→ The data transfer can be either from register (or) immediate value. It is denoted by PSR (fields) = Rm

PSR (fields) = imm

→ Together MRS & MSR perform read write operation.

→ The fields comprises of the Syntax given by

c (constant) x (extension) } fields.
s (status) f (flags)

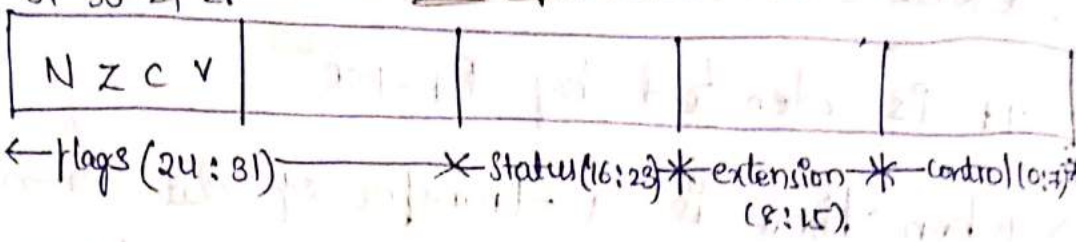
→ Syntax:

MRS {<cond>} Rd, <cpsR/spsR>

MSR {<cond>} <cpsR/spsR>_ <fields>, Rm

MSR {<cond>} <cpsR/spsR>_ <fields>, imm

PSR byte fields:



Software interrupt instructions:

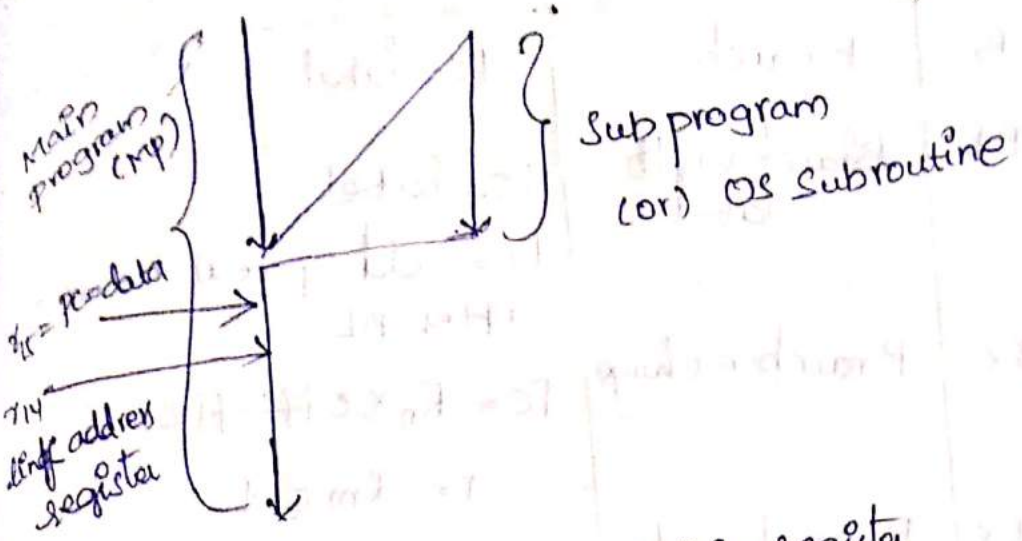
The Software interrup instructions is denoted by SWI.

→ It is the mechanism for calling operating System Subroutine applications (or) other hardware pulgin's.

Syntax:-

SWI {<cond>} SWI_number

→ There are 'n' no. of Software interrupts which is given by the SWI_number.



→ The Program Counter (or) PC register is responsible for storing the next byte of instruction that has to be executed after SWI.

→ R14 register (Link address register) is responsible for storing the return address.

Branch instructions:

ARM-5 processor in general has 4 types of branch instruction sets whose syntax is given by

B { <cond> } label.

BL { <cond> } label

BX { <cond> } Rm → m=0-12 register acts as a pointer which points to some address

BLX { <cond> } label/Rm

B	Branch	PC = label
BL	Branch with label	PC = label lr = add of next instruction after BL
BX	Branch exchange	PC = Rn & 0xfffffffe T = Rm & 1
BLX	Branch exchange with link	PC = label T = 1 PC = Rn & 0xfffffffe T = Rm & 1 lr = add of next instruction after BLX

Note; when T = Rm & 1 → It operates in Thumb mode (16-bit mc) & Rm = r14 = link address

→ In pc = r15 → next byte of data is stored after interrupt (or) Branch.

→ In pc = r14 ≠ Link address → next byte of address after interrupt are executed

Load & Store architecture; Instructions:

→ In general for memory to memory data transfer the time taken is more to avoid this (or) to increase the efficiency of the instructions (or) to reduce the timeperiod Load & Store instructions are used.

→ first the data is transfered from memory to cpu register which is called load & then cpu register to memory which is called store.

→ The Overall time period is taken is very less when compared to memory to memory data transfer.

→ There are 3-types of load & Store instructions given by.

1) Single register transfer (LDR/STR)
2) Multiple register transfer. (LDM/STM)
3) SWAP

→ The Syntax for load & Store is given by

<LDR/STR>{<cond>}{B} Rd, addressing1

LDR{<cond>}SB|H|SH Rd, addressing2

STR{<cond>}H Rd, addressing2.

where SH is Signed Half word

H is Half word

SB is Signed byte

Questions:

1) Explain about the introduction of ARM processor.

2) Explain ARM processor architecture.

3) Explain the Register organisation of ARM processor.

4) Explain the Operating modes of ARM.

5) Differentiate b/n RISC & CISC.

6) Explain the pipelining

7) Explain about exceptions & IVT

8) Explain about PSR instructions

9) Explain about Data processing instructions

10, Explain SWI

11, Explain about branch instructions

12, " " Load & Store "

23/3/19

UNIT- V

CORTEX Processor

→ The enhanced version of ARM processor is called as CORTEX processor.

→ It is a 64 bit RISC with high speed of execution more no. of instruction sets & high 5- stage pipe lining like 9-stage & 11-stage pipe lining etc.

→ It uses the concept of RISC technology & mainly used in high end applications like Smartphone i-phone 6s, FS series, Satellite applications, IOT application etc

→ The hardware & Software associate with the CORTEX processor is costly when compared to the ARM processor.

→ CORTEX processor is used in wide range of applications like Mobile phones, Gaming Consoles, automative navigation & entertainment systems.

→ It consumes less power in the range of 300mW. As a result heat dissipation is very less resulting in lower packaging & integration cost.

→ COTEX processor use THUMB-2 technology in which instruction set combines 16 & 32-bit instructions to improve code density & performance.

→ In THUMB-2 technology it adds 130 additional instructions to thumb & removes the need to switch b/n ARM processor & THUMB-1 Operation mode.

→ COTEX processor uses NEON technologies for media & signal processing applications like audio, vedio, & 3D graphics.

→ It is a 64/128 bit hybrid SIMD technology & it has its own register files & execution pipe lines which are separate from the main ARM integer pipeline.

→ It can handle both integer & floating for more accuracy & precision.

→ CORTEX-A8 Proces Can decode MPEG EGA videos at 30 frames per sec at 275 MHz & H-~~point~~ H.264 ~~video~~ video at 350 MHz

→ CORTEX Proces uses trust zone technolo TO ensure data privacy & protection.

→ NEON technology enables Software Solution to data processing applications like installin new Softwares & device drivers.

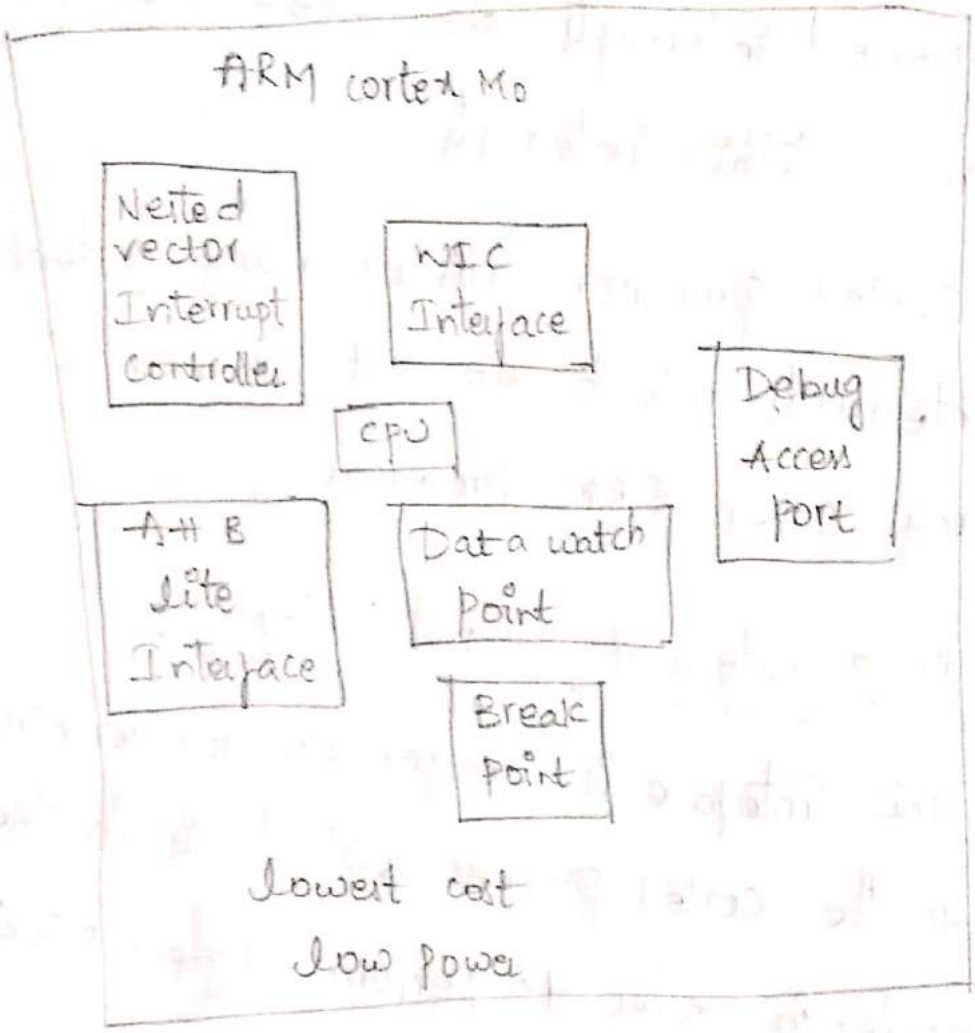
→ It can handle both integer & floating for more accuracy & precision.

→ CORTEX-A8 proces can decode MPEG4 VGA videos at 30 frames per sec at 275 MHz & H.264 video at 350 MHz

→ CORTEX proces uses trust zone technology TO ensure data privacy & protection.

→ NEON technology enable Software Solutions to data processing applications like installing new Softwares & device drivers.

Cortex Processor architecture:



→ Cortex processor is available in various versions like cortex Mo, cortex M1.---. Cortex M8.

→ Cortex Mo- Cortex M6 are 32-bit uc's & from cortex M7 onwards 64-bit uc's

Cortex - Mo

Nested vector Interrupt controller:

→ It is similar like IVT where all the interrupts are executed by going to the

respective ISR address.

→ Nested interrupts are where interrupts are within interrupts.

→ Cortex processor allows many nested interrupts which are not allowed in any other ARM processor.

Wakeup Interrupt Control Interface :-

→ This interface is responsible for waking up the cortex processor which is in sleep mode in order to perform highest priority instructions.

Note: Each instruction consists less than 1 cycle in cortex processor. As a result there is feasibility for the cortex processor to go into sleep mode.

Advanced high Bus lite Interface:

→ By the means of advanced high bus for data there is a feasibility of data transfer to take place for high speed applicat

-ons. &

→ In normal applications where data rate is very low. Normal cortex processor data bus is enough to cater the requirements.

Data watch point:

→ This provides the feasibility for monitoring & checking the data at any instant of time in the available registers.

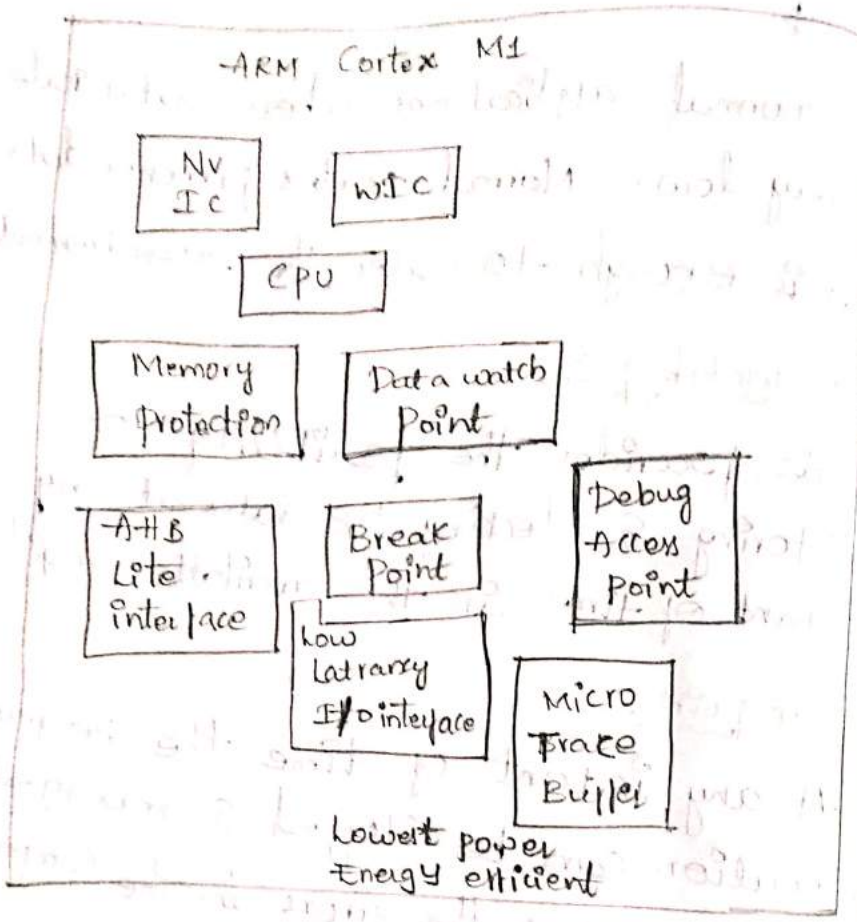
Break point:

→ At any instant of time the program execution can be stopped & debugged for finding out the errors in the computation.

Debug Access Port:-

Externally also it can debug the program using JTAG port connector.

Note: -ARM cortex M0 Processor is used for low Power Consumption & lowest cost among the available cortex processor



Memory Protection Unit:

→ Flash memory is used in most of the real time applications & Mpu protects the data from getting corrupted.

→ SRAM is used to store the data of the program.

Note: Erasing of bit of information in flash memory is done at single

where as
bit by bit

Low latrancy

→ latrancy
states.

→ wait stat
peripherals
to the proc

→ The less
better the

Micro trace

→ Tracing
in the buff
CORTEX proce

→ This CORTE
like lowest p

where as in other memories it is done bit by bit. It is a time consuming proces.

Low latrancy I/o interface:-

→ latrancy refers to the no. of wait states.

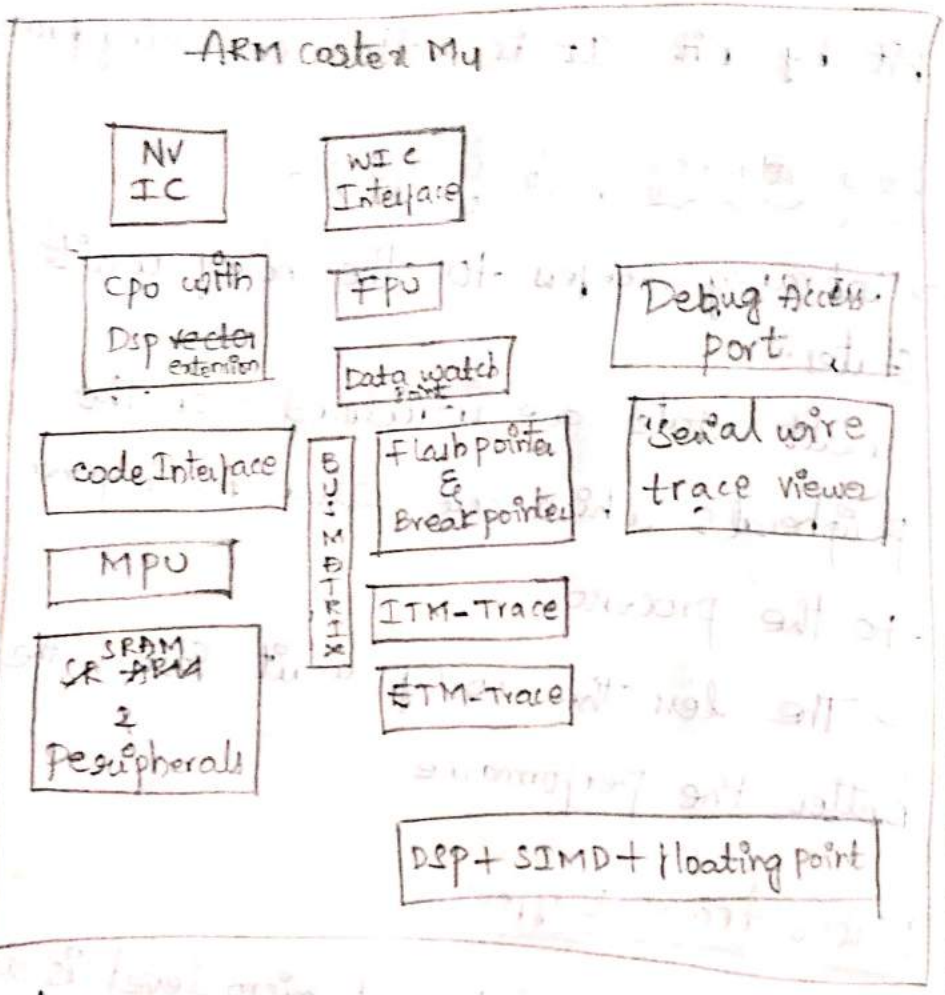
→ wait states are introduced for the peripherals which are slow to respond to the processor.

→ The less the no. of wait states the better the performance.

Micro trace Buffer:

→ Tracing of data at micro level is done in the buffer which stores the data in CORTEX processor.

→ This CORTEX processor is used in applications like lowest power consumption & energy efficient.



→ Cortex M4 processor is an enhanced version of cortex M0, M1, M2, M3 processor with DSP extension

→ It can perform Signal processing applications like Noise removal, Dividing the signal into different frequency bands, Amplifying the signal strength etc.

→ The Op-code of cortex processor is stored in flash memory.

→ The
in S
→ Both
protect
→ cort
opera
→ SIM
→ ITM
→ ETM
→ Seria
→ These
level

2) It can perform signal processing applications like noise removal, dividing the signal into different frequency band amplifying signal strength etc.

3) The opcode of cortex processor is stored in flash memory

4) The data of the cortex processor is stored in SRAM.

5) Both this flash memory & S-RAM are protected from getting corrupted by MPU.

6) Cortex M4 processor performs floating point operations by means of FPU.

7) SIMD stands for Single instruction multipledata

8) ITM Trace – Instrumention trace micro viewer

9) ETM Trace Embedded trace micro viewer

10) Serial wire trace viewer.

11) These 3 traces are responsible for microlevel analysis of data performance.

01/04/19

* Register organisation of CORTEX Processor:

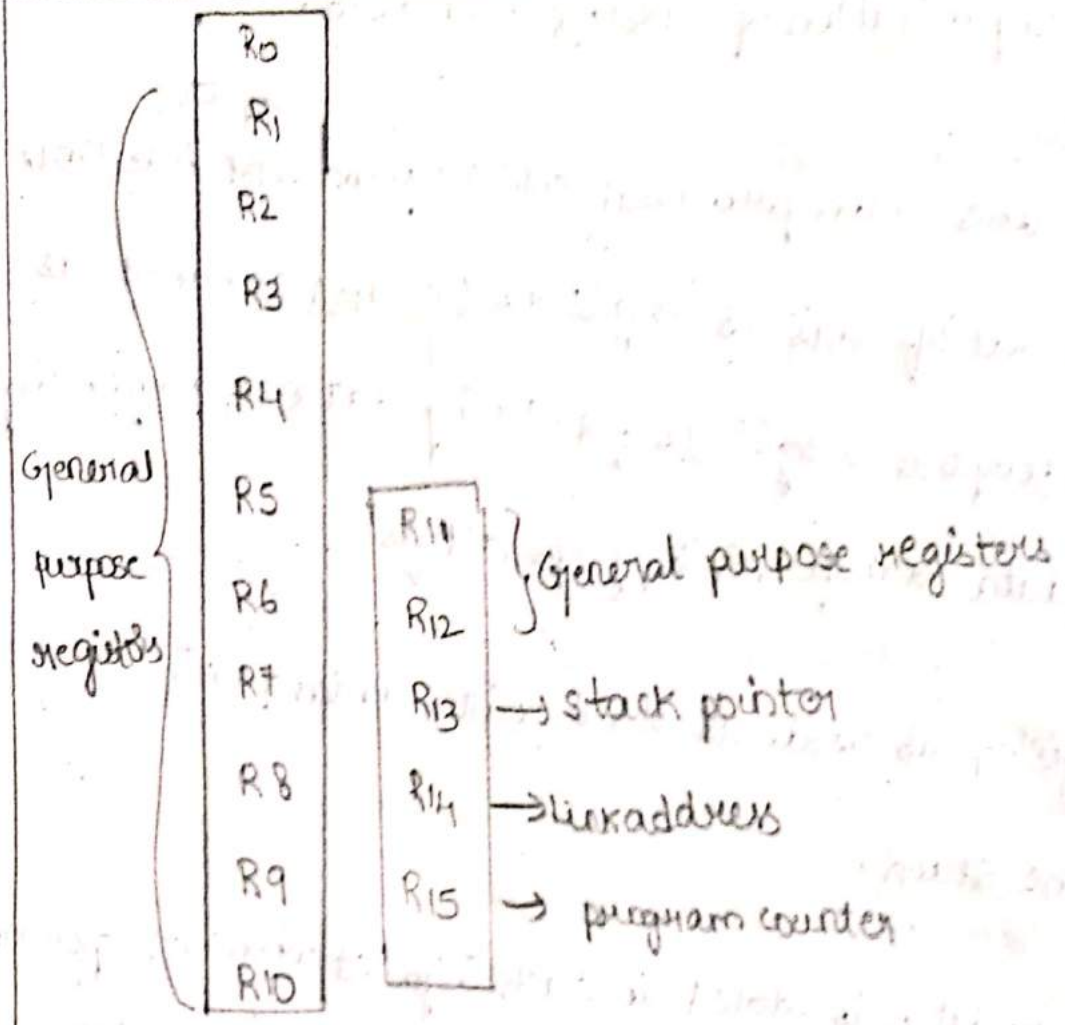
1) Similar like arm processor CORTEX processor has 16bit register out of this 13 registers (r_0-r_{12}) are used as general purpose registers for storing data & performing various data transfer & copy operations.

2) R_{13} register is used as stack pointer which points to the top of the stack.

3) Important data is stored in stack Segment whenever program wants to retrieve the data it can retrieve the data from stack.

4) R_{14} register is used as link address which specifies the written address of external or internal interrupt.

5) R_{15} register is the program counter which stores the next byte of instruction that has to be executed after the completion of interrupt.



Note:

1) CORTEX M3-M6 processors are 32 bit microcontrollers which uses 32 bit registers & from CORTEX M7 onwards they are 64 bit microcontrollers.

2) It uses 64 bit registers.

02/04/19.

* OMAP Processor (open multimedia application platform):

1) Texas instruments have developed these omap processors which has a series of image & video processors.

2) SOC (System on chip) was developed for portable & mobile multimedia applications which is a part of omap processors.

3) Omap processor includes general purpose ARM processor with 1 or 2 specialized co-processors.

4) It is similar to TMS-320C DSP processors.

5) ST microelectronics & TI have developed omap processor in the year 2003 which facilitates 2.5 & 3G mobilephones.

6) omap processors ruled till 2011, when it lost its credibility to qualcom snap drag on Sep 26th 2012.

7) Its last omap-5 chips were released in the year 2013.

Note:

Platform is an environment where hardware & software can be executed.

8) OMAP processors can be divided into 3 groups based on

i, high performance application processors

ii, Basic multimedia application processors

iii, Integrated modern application processor

i) High performance application processors:

In this there are 5 types of omap processor given by:

a) OMAP1

b) OMAP2

c) OMAP3

d) OMAP4

e) OMAP5.

a) OMAP 1:

1) They are used in smartphones

2) they are used with operating systems like Symbian, Linux & android.

3) It is used in personal computers, audio & video application.

4) Eg: Nokia 770, Tablets

5) There are various versions in OMAP1 given by:

i) OMAP 171 X – operating frequency is 220mHz & 90nanometer technology.

ii) OMAP162X – 204 megaHz operating frequency & 130nanometer technology

iii) OMAP161X – 204 mHz operating frequency & 130 nano meter technology.

iv) OMAP1510 – 168megaHz operating frequency & 130nano meter technology

b) OMAP2:-

1) This processor is used in internet tablets & mobile phones.

2) It has got various versions given by:-

OMAP 2431 — 330MHz & 90nm technology.

OMAP 2430 — 330mHz op freq & 90nm technology

OMAP 2420 — 330mHz & 90nm technology

c) OMAP-3:

1) This processor is used in image, video, audio (IVA2 application)

2) It comprises of 12megapixels resolutions

Note: pixel is the minute part of picture element & more the resolution better the quality.

3) OMAP-3 is available in various versions given by:

OMAP3410, OMAP 3420, OMAP3440, OMAP3503, OMAP3515, OMAP 3525, OMAP 3620 all these versions uses 65nm technology.

4) OMAP 3611, OMAP3621, OMAP3622, OMAP3630, OMAP3640, all these versions uses 45nm technology

d) OMAP-4:

1) It comprises of dual core ARM cotter A9 with two ARM cotter M3 processors.

2) It is used in IVA3 accelerator.

3) It is used in video encoding & decoding applications.

4) By means of encoding & decoding security & authentication comes into picture

5) omap-4 is available in various versions given by: OMAP 4430, OMAP 4460, OMAP4470 which uses 45nm technology.

e) OMAP-5:

1) It comprises of dual core ARM cotter M15 processors.

2) It has got 2 versions given by:-

omAp5430 & omAp 5432, which uses 28nm technology.

03/04/19

* omAp processor Architechture:

II) Serial interface:-

1) UART: (universal asynchronous receiver transmitter)

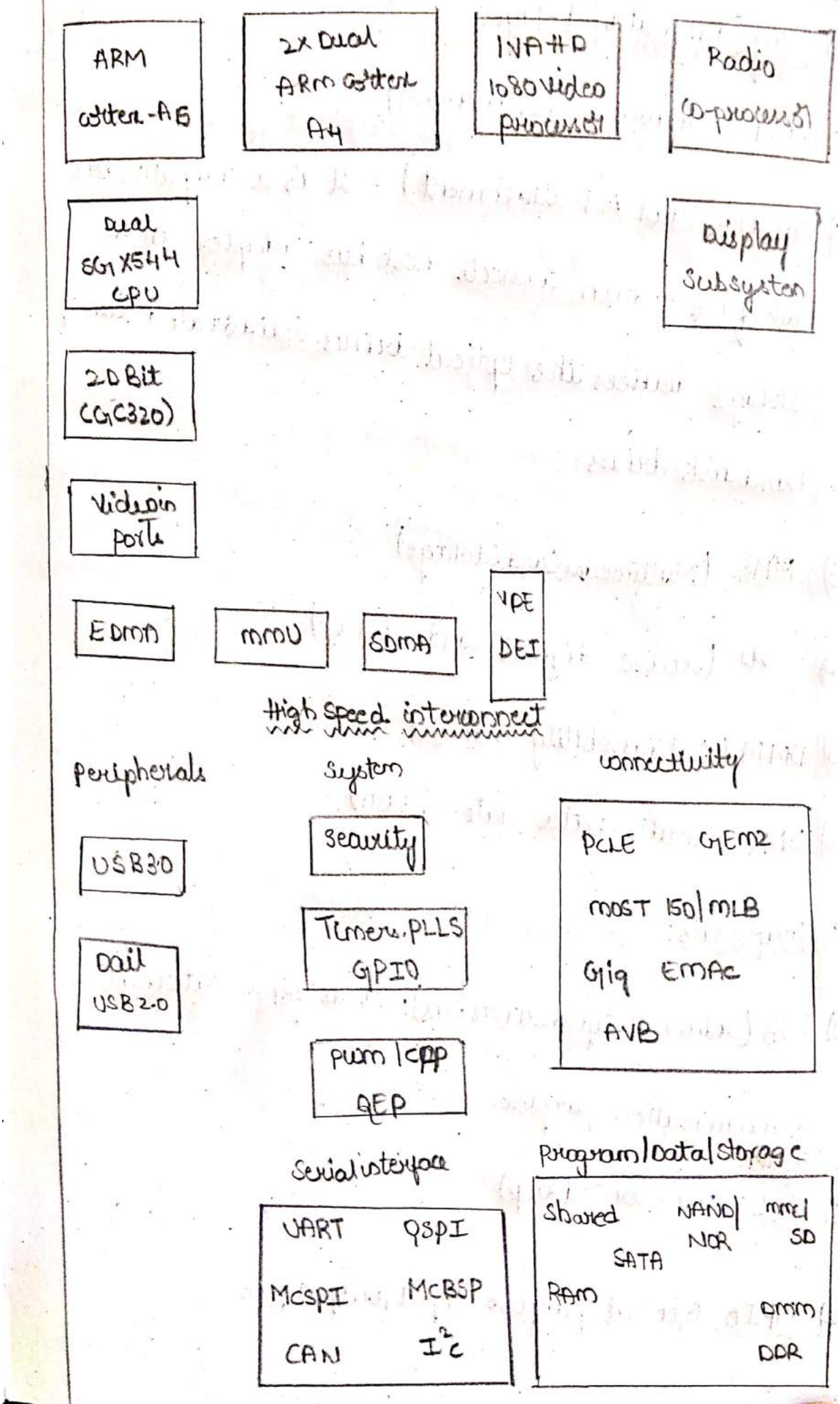
2) QSPI: (Quad Serial peripheral interface): It provides interface with flash memory to read & write data

3) MCSPI: (Master slave controller serial peripheral interface): It can interface a maximum of 4 slaves & 1 master. It is used for full duplex serial communication.

4) McBSP (Multi channel buffered serial port): It can interface with DSP kits, codecs etc.

5) I^2C: (Inter integrated circuit):

6) CAN (controlled area network)



III) Program / Data / storage :-

1) RAM (Random access memory)

2) SATA (Serial AT attachment) : It is a computer bus interface which connects host bus adapter to m storage devices like optical drives, Solid state drives & hard disk drives

3) MMC (Multimedia Card storage)

4) SD (Secure digital card storage)

5) DMM (Data mobility manager

6) DDR (Double Data rate SDRAM)

IV) Peripherals :-

1) USB (universal synchronus bus): It is for a external communication purpose.

V) system :

1) PLL (phase locked loop)

2) GPIO General purpose Input / output pins

3) PWM (pulse width modulation)

4) CAP (carrierless amplitude phase modulation) : It is one type of QAM.

5) QEP (Quadrature encoder used in pulse width modulation)

V) connectivity:

1) AVB (audio video Bridging)

2) EMAC (Ethernet media access control)

3) Gig : (Giga bit)

4) MLB (Major league baseball video content) : It contains 40 Gbyte

VI) omap processor comprises of 1 general purpose ARM processor i.e., A15 (cortex A15) & 2 Specialized cortex processors (A4)

VII) IVAHD (Image video audio high dimension) : the resolution is 1080 pixel resolution

VIII) MMU (Memory managment unit)

Note:

Effective utilization of memory is called memory managment

SDMA(Spacial division multiple axis): It is a channel axis method used in mobile

Questions:

1) Explain fundamentals of cortex processor?
2) With neat diagram Explain the architechture of cortex m3?
3) With neat diagram Explain the architechture of cortex m1?
4) With neat diagram Explain the architechture of cortex m4?
5) Explain the fundamentals of omap processor
6) With neat diagram Explain the architechture of omap processor.
7) Explain the register organisation of cortex processor?