CMR
INSTITUTE OF
TECHNOLOGY

USN | | | | | | | | | | |

Internal Assessment Test – I

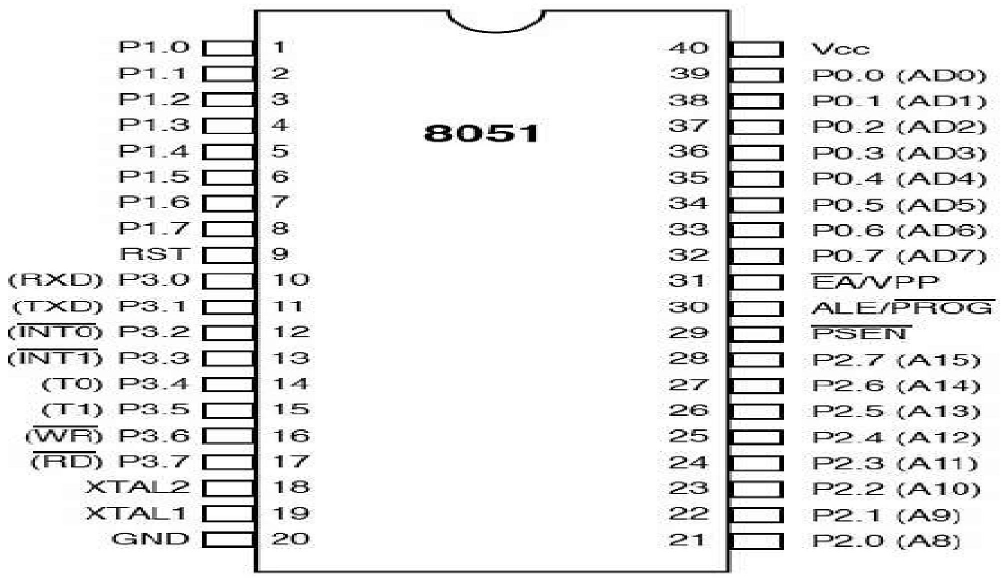
Sub:	**Microcontrollers**							Code:	18EE52
Date:	11/11/2021	Duration:	90 mins	Max Marks:	50	Sem:	5th	Branch:	EEE

Answer **Any FIVE FULL** Questions

		Marks	OBE	
			CO	RBT
1	Draw the pin diagram of 8051 with a neat diagram and brief about its features. **Pin diagram- 4 marks** **Brief description- 6 marks**	10	CO4	L1
2	a) Define assembler directive and with suitable example explain the commonly used assembler directives. (5) **Assembler Directive- 1 marks** **Org directive- 1 marks** **End directive-1 marks** **Count directive-1 marks** **Examples- 1marks** b) Write briefly about the steps involved in creating, assembling and running the program in 8051. (5) **Flow chart- 2 marks** **Steps explanation- 3 marks**	10	CO3	L2
3	Explain the internal RAM organization of 8051 with suitable diagrams. **Internal RAM Organization diagram- 3 marks** **Working register bank-2 marks** **Bit Addressable -2 marks** **Scratch pad-3 Marks**	10	CO2	L1
4	Explain with suitable example, the different types of addressing modes used in 8051. **Five Addressing modes along with examples** **1. Immediate Addressing mode-2 marks** **2.Register addressing mode-2 marks** **3.Indirect Addressing mode-2 marks** **4. Register Indirect Addressing Mode- 2 marks** **5. Indexed Addressing mode- 2Marks**	10	CO1	L1
5	With a neat block diagram of 8051, brief about its salient features. **Block Diagram- 4 Marks** **Salient Features-6 Marks**	10	CO2	L1
6	Explain in brief for the following 1. Program Counter & DPTR 2. PSW3.Stack Pointer 4.Register Bank **Each Brief explanation – 2.5 marks**	10	CO1	L4

7	Explain the program ROM space allocation for the following: 1.EA=0 for 8751 chip **- block diagram and explanation- 5 marks** 2. EA= Vcc with both on-chip and off-chip ROM for 8751.- **block diagram and explanation- 5 marks**	10	CO1	L3
8	Explain the arithmetic instruction used for 8051 **Explanation Addition with examples- 2.5marks** **Explanation Subtraction with examples -2.5marks** **Explanation Multiplication with examples – 2.5 marks** **Explanation Division with examples-2.5 marks**	10	CO2	L4

1. **Pin diagram of 8051**



- To access the pins of port O as input & output ports, each pin must be connected externally to a 10KΩ pull-up resistor.
- Port 0 is designated as AD0-AD7, allowing it to be used for both address & data.
- When Port 0 is connected to an external memory, port 0 provides address and data.

Port-1

- Total of 8 pins.
- Can be used as an input or output.
- Does not require any pull up resistors
- If port 1 has been configured as an output port, to make it an input port again, it can programmed by writing 1 to all its bits.

Port-2

- Total of 8 pins.
- Do not require pull up resistor.
- On reset, port 2 is configured as an input port.

Port-3

- Total of 8 pins.
- Do not require pull up resistor
- Port 3 has the additional function of providing extremely important signal such as interrupts.

2. Data Byte Directive:

- The DB directive is the most widely used data directive in the assembler.

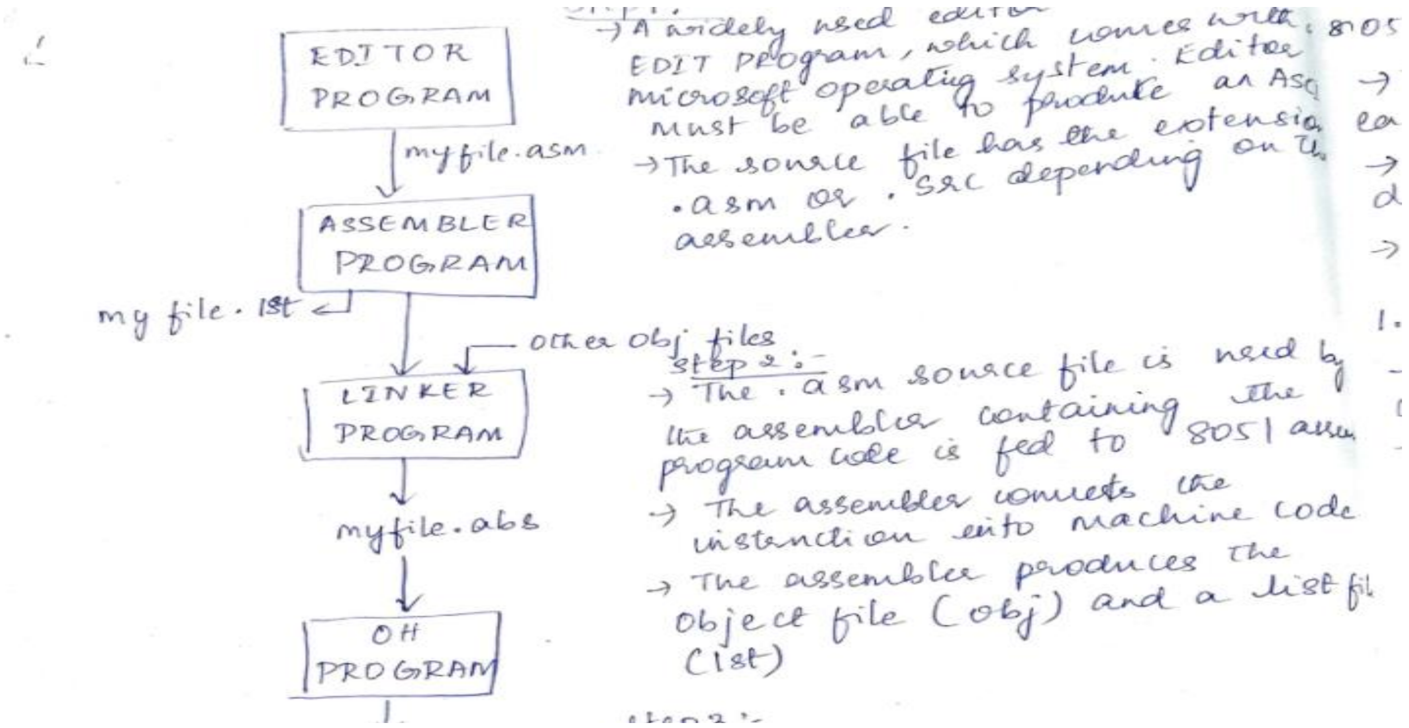
- It is used to define the 8-bit data.
- When DB is used to define data, the numbers can be in decimal, binary, hex, or ASCII formats. For decimal, the “D” after the decimal number is optional, but using “B” (binary) and “H” (hexadecimal) for the others is required
- ORG
- The ORG directive is used to indicate the beginning of the address.
- The number that comes after ORG can be either in hex or in decimal.0000 or 0000D MOV A,30
- If the number is not followed by H, it is decimal and the assembler will convert it to hex.
- Some assemblers use “. ORG” (notice the dot) instead of “ORG” for the origin directive. Check your assembler.
- EQU
- This is used to define a constant without occupying a memory location.
- The EQU directive does not set aside storage for a data item but associates a constant value with a data label so that when the label appears in the program, it constant value will be substituted for the label.
- The following uses EQU for the counter constant and then the constant is used to load the R3 register.

  Eg: COUNT EQU 25

  ……………….

  Mov r3, #COUNT

2.b.



→ A widely used editor [illegible] comes with 8051 EDIT program, which [illegible] microsoft operating system. Editor must be able to produce an Asc[illegible]

→ The source file has the extension .asm or .src depending on the assembler.

step 2:-

→ The .asm source file is need by the assembler containing the 8051 assem[illegible] program code is fed to

→ The assembler converts the instruction into machine code

→ The assembler produces the object file (obj) and a list fil[illegible] (lst)

↓
myfile.hex

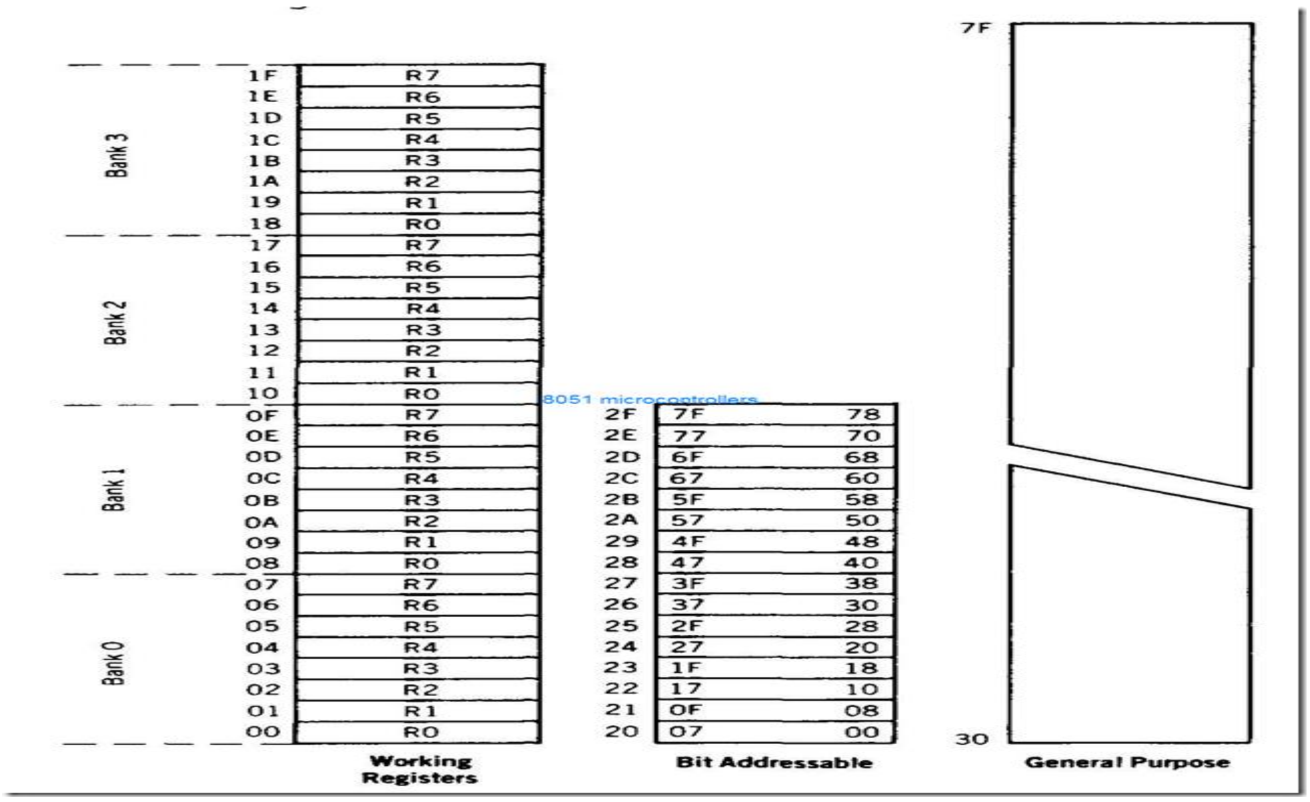
step 3:-

→ The link program takes one or more object files & produces an absolute object file (.abs)

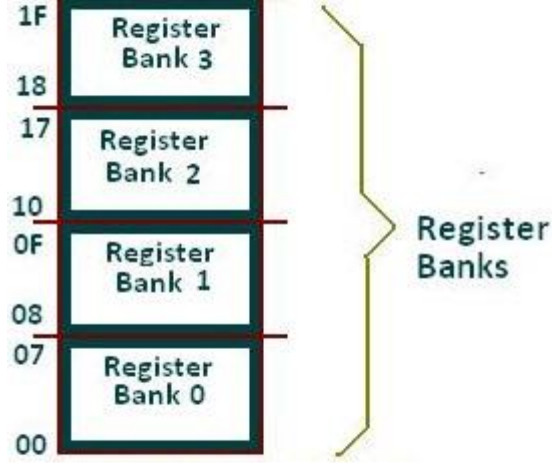
Step 4:-

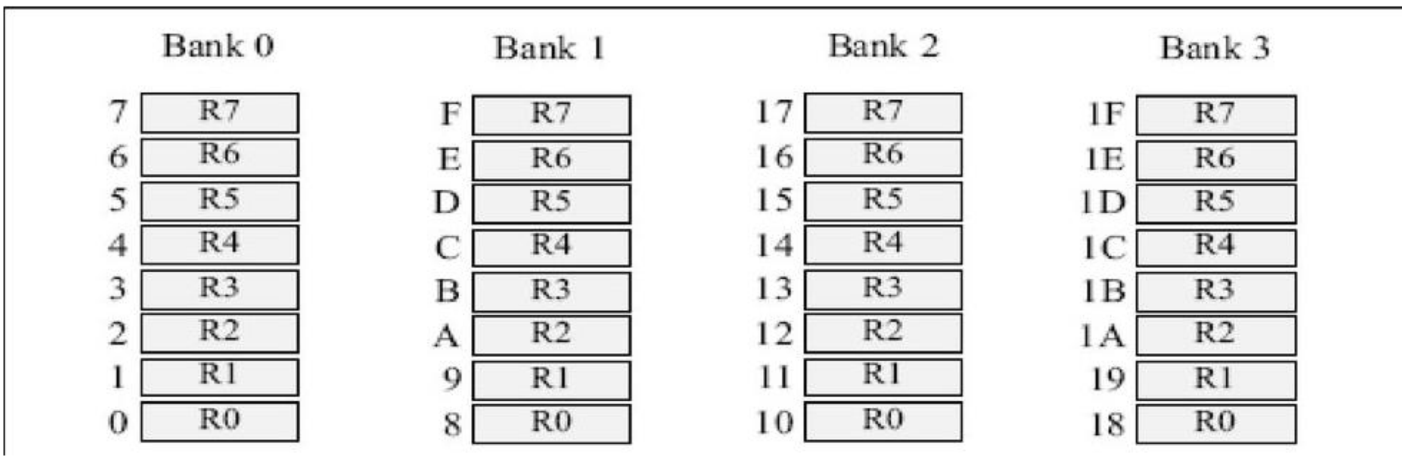
→ The "abs" file is fed into a program called OH (object fi hex converter) which creates the file with extension " he ready to burn into ROM.

3.



- Four register banks.
- Each bank containing 8 registers labeled from R0-R7.
- Total of 32 working registers labeled from 00-1FH.
- By default bank 0 will be selected on reset





Bank 0		Bank 1		Bank 2		Bank 3	
7	R7	F	R7	17	R7	1F	R7
6	R6	E	R6	16	R6	1E	R6
5	R5	D	R5	15	R5	1D	R5
4	R4	C	R4	14	R4	1C	R4
3	R3	B	R3	13	R3	1B	R3
2	R2	A	R2	12	R2	1A	R2
1	R1	9	R1	11	R1	19	R1
0	R0	8	R0	10	R0	18	R0

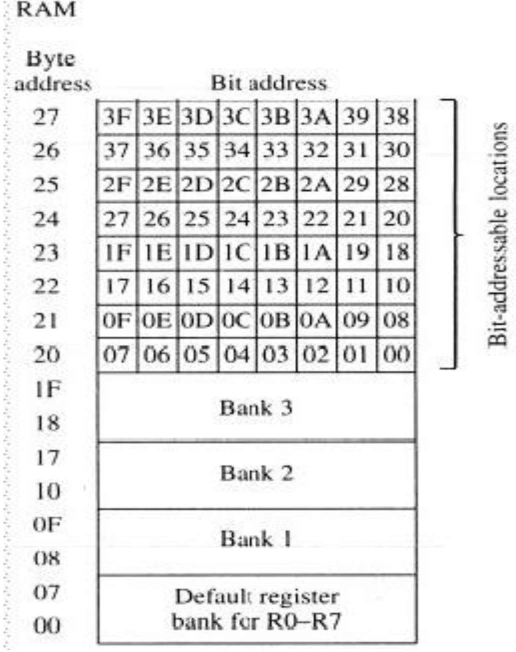
Bit Addressable RAM

16 bytes of RAM from address 20h to 2fh are bit addressable.

Each bit of has an address and can be accessed individually

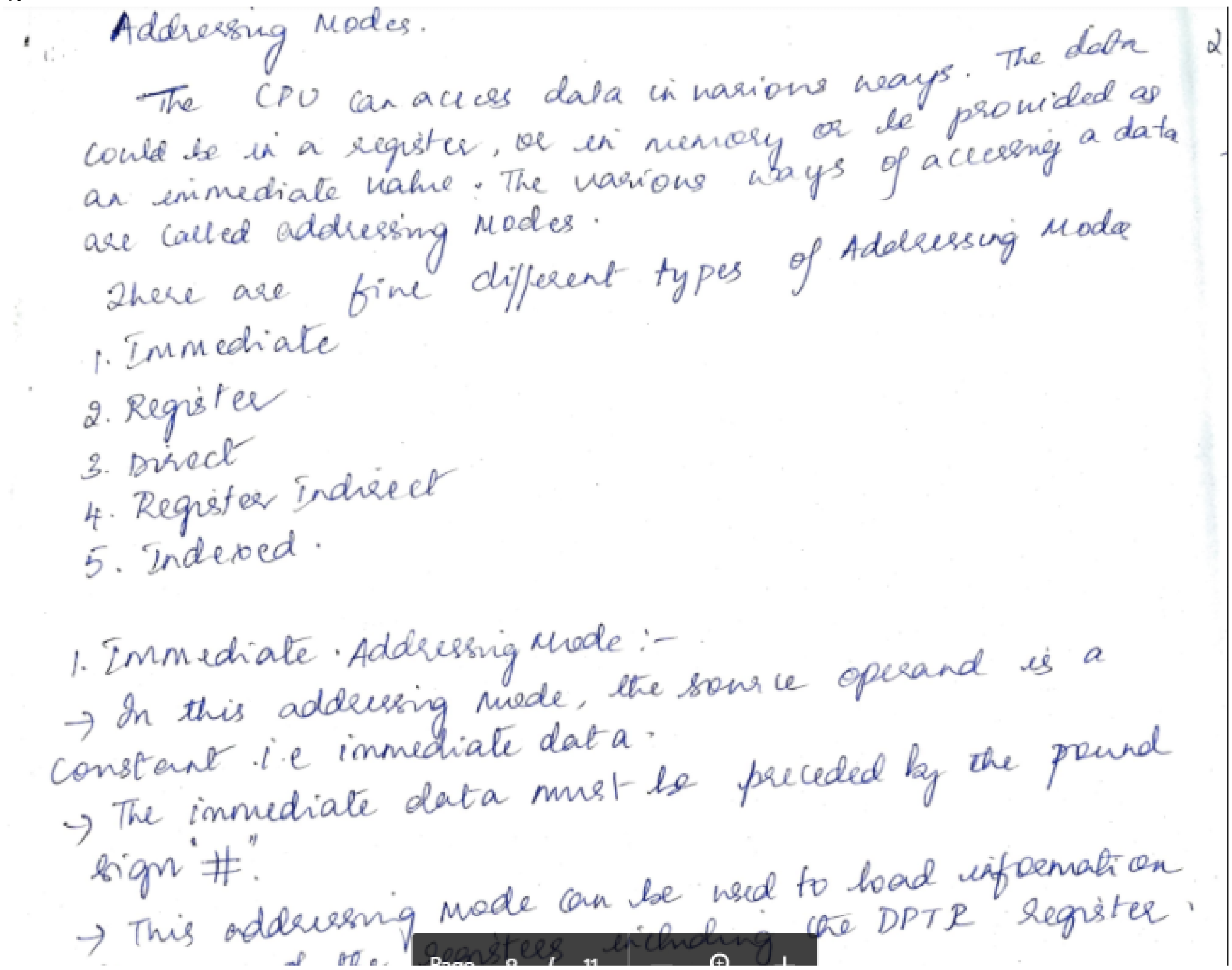
Total of 128 bits address is given in the register bank.

General purpose RAM-registers above bit addressable registers are called Scratch pad registers. Used for temporary


RAM

Byte address	Bit address							
27	3F	3E	3D	3C	3B	3A	39	38
26	37	36	35	34	33	32	31	30
25	2F	2E	2D	2C	2B	2A	29	28
24	27	26	25	24	23	22	21	20
23	1F	1E	1D	1C	1B	1A	19	18
22	17	16	15	14	13	12	11	10
21	0F	0E	0D	0C	0B	0A	09	08
20	07	06	05	04	03	02	01	00
1F 18	Bank 3							
17 10	Bank 2							
0F 08	Bank 1							
07 00	Default register bank for R0–R7							

4.


Addressing Modes.

2

The CPU can access data in various ways. The data could be in a register, or in memory or be provided as an immediate value. The various ways of accessing a data are called addressing modes.

There are fine different types of Addressing mode

1. Immediate
2. Register
3. Direct
4. Register Indirect
5. Indexed.

1. Immediate Addressing mode :-

→ In this addressing mode, the source operand is a constant i.e immediate data.

→ The immediate data must be preceded by the pound sign "#".

→ This addressing mode can be used to load information ~~~~ the registers including the DPTR register.

→ The immediate data must be p[illegible]
sign "#".

→ This addressing mode can be used to load information into any of the registers including the DPTR register.

Eg:- ①
```
MOV A, #56H        ; load 56H into Accumulator
MOV R3, #62        ; load the decimal value 62
                     into R3
MOV B, #40H        ; load 40H into B
MOV DPTR, #4521H   ; DPTR = 4512H.
```

②
```
MOV DPTR, #2550H
   Or
MOV  DPL, #50H
MOV  DPH, #25H.
```

2. Register Addressing Mode:-

It involves the use of registers to hold the data to be Manipulated.

Eg:-
```
MOV A, R0     ; copy the contents of R0 into A
MOV R2, A     : copy the contents of A into R2
ADD A, R5     : Add the contents of A to the
                contents of R5
ADD A, R7     ; add the contents of R7 to
                contents of A.
MOV R6, A     ; Save the accumulator in R6
```

→ The size of the source & destination should be matched.

Eg:-① MOV DPTR, #25F5H → valid.

② MOV DPTR, A → invalid

Source is A → 8 bit register

Destinatination is → 16 bit register

→ The data can be moved between the accumulator & Rn (for n=0 to 7), but the movement of data between Rn is not allowed.

Eg :- MOV A, R0 valid.

MOV R0, R7 invalid.

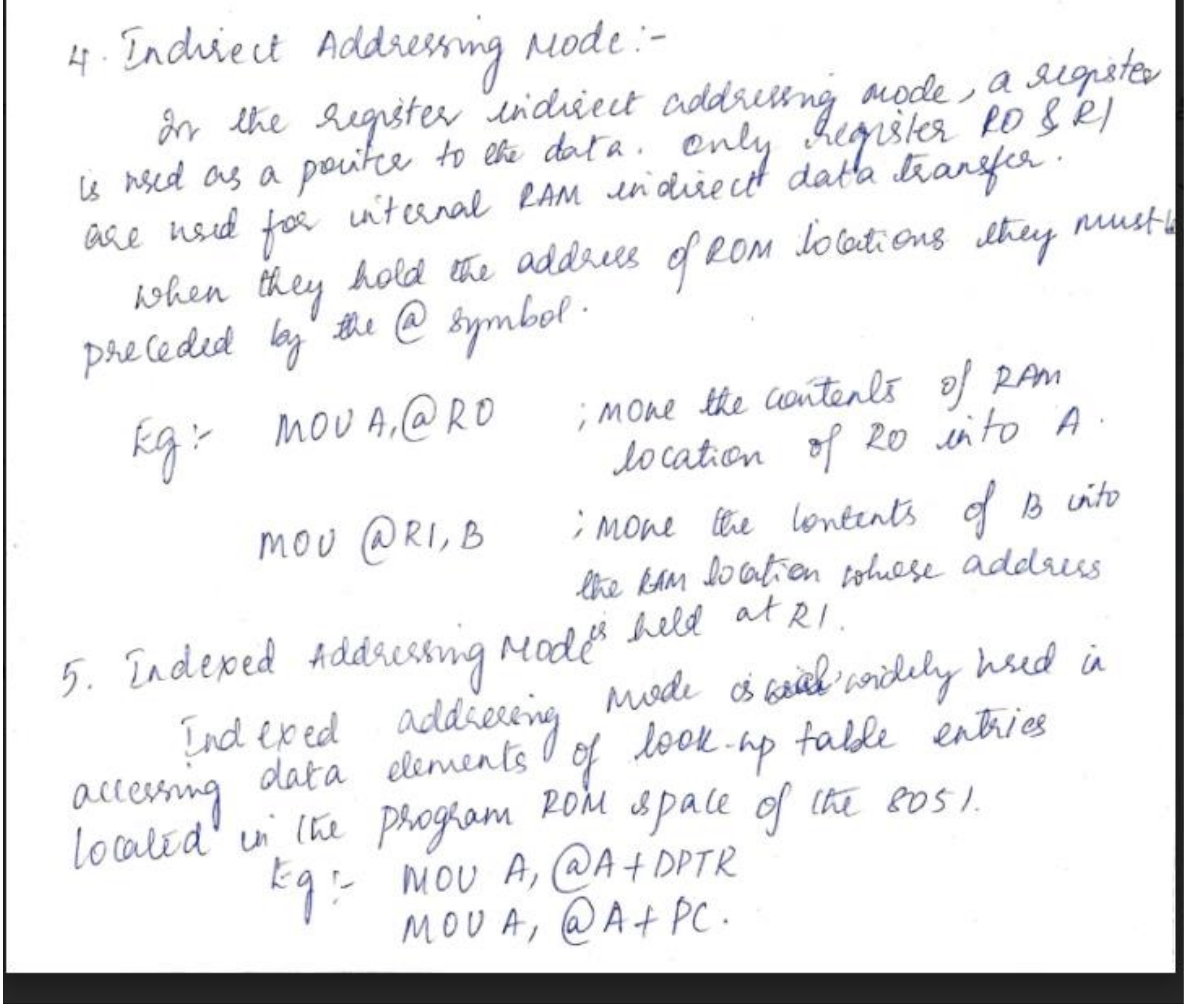
③ Direct Addressing Mode :-

→ The data is in a RAM memory location whose addressing is known & this address is given as part of the instruction.

→ Although the entire 128 bytes of RAM can be addressing using direct addressing mode.

→ It is most often used to access RAM locations 30 - 7FH.

```
Eg :- MOV R0, 40H    ; ~~lop~~ Save the contents of RAM
                       location 40H to R0
      MOV 7FH, A     ; Save the contents of A into
                       RAM location 7FH.
```



4. Indirect Addressing Mode :-

In the register indirect addressing mode, a register is used as a pointer to the data. Only register R0 & R1 are used for internal RAM indirect data transfer.

When they hold the address of ROM locations they must preceded by the @ symbol.

Eg:- MOV A,@R0 ; move the contents of RAM location of R0 into A.

MOV @R1, B ; move the contents of B into the RAM location whose address is held at R1.

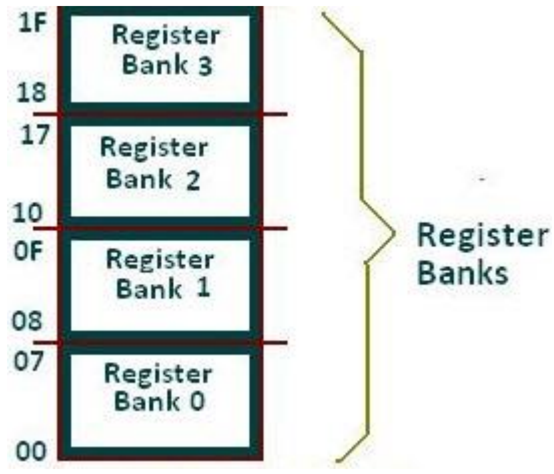
5. Indexed Addressing Mode

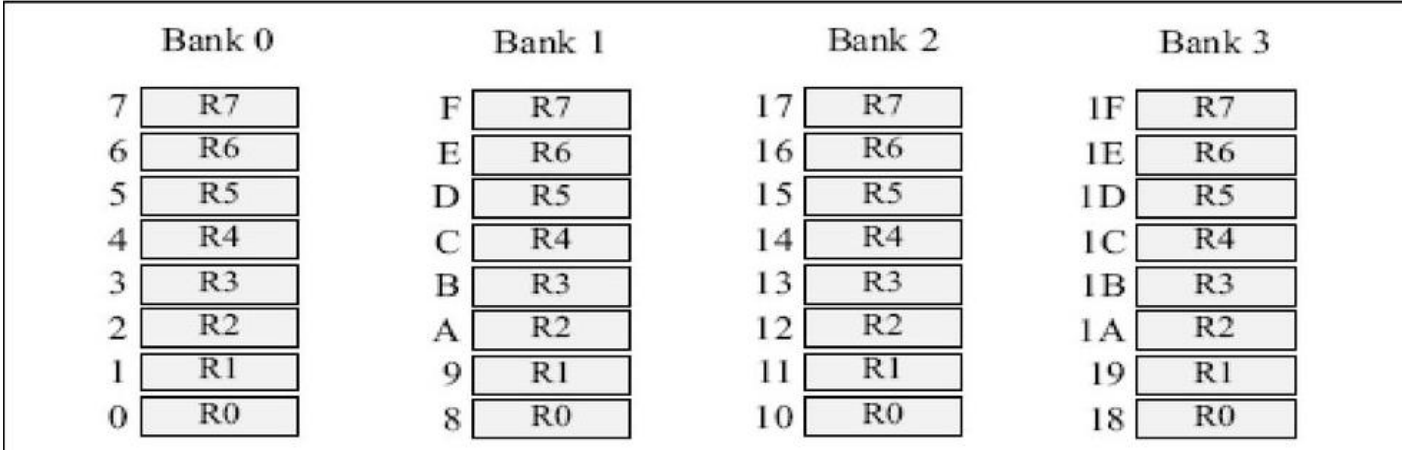
Indexed addressing mode is ~~[illegible]~~ widely used in accessing data elements of look-up table entries located in the program ROM space of the 8051.

Eg:- MOV A, @A+DPTR
MOV A, @A+PC.

6.Register Bank

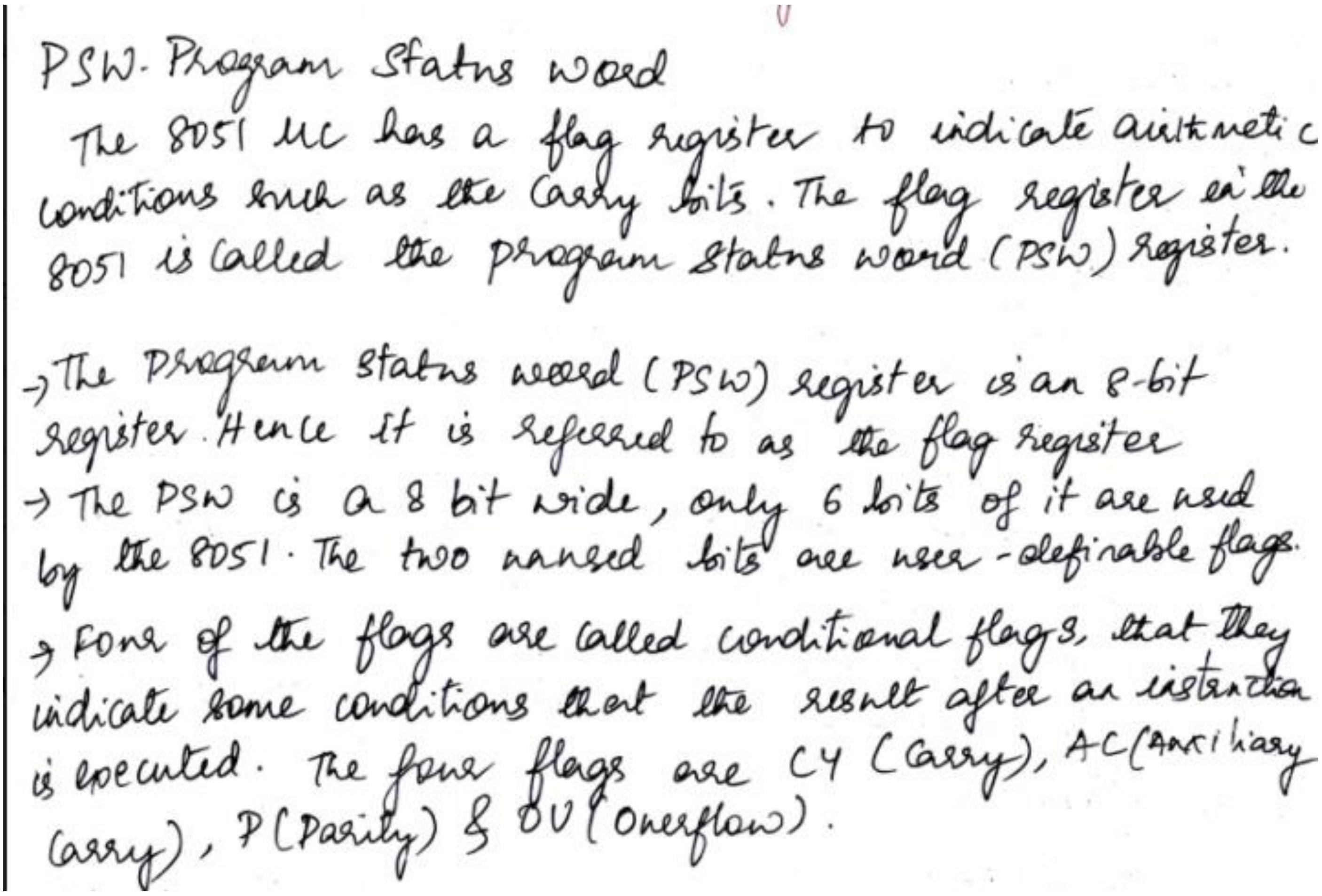
- Four register banks.
- Each bank containing 8 registers labeled from R0-R7.
- Total of 32 working registers labeled from 00-1FH.
- By default bank 0 will be selected on reset





Bank 0		Bank 1		Bank 2		Bank 3	
7	R7	F	R7	17	R7	1F	R7
6	R6	E	R6	16	R6	1E	R6
5	R5	D	R5	15	R5	1D	R5
4	R4	C	R4	14	R4	1C	R4
3	R3	B	R3	13	R3	1B	R3
2	R2	A	R2	12	R2	1A	R2
1	R1	9	R1	11	R1	19	R1
0	R0	8	R0	10	R0	18	R0

Data Pointer



PSW- Program Status word

The 8051 µc has a flag register to indicate arithmetic conditions such as the Carry bits. The flag register in the 8051 is called the program status word (PSW) register.

→ The Program status word (PSW) register is an 8-bit register. Hence it is referred to as the flag register

→ The PSW is a 8 bit wide, only 6 bits of it are used by the 8051. The two unused bits are user-definable flags.

→ Four of the flags are called conditional flags, that they indicate some conditions that the result after an instruction is executed. The four flags are CY (Carry), AC (Auxiliary carry), P (Parity) & OV (Overflow).

CY	AC	FO	RS1	RS0	OV	--	P

CY	PSW.7	carry flag
AC	PSW.6	Anxiliary flag
Fo	PSW.5	Available to the user for general purpose
RS1	PSW.4	Register Bank selector bit 1.
RS2	PSW.3	Register Bank Selector bit 0.
OV	PSW.2	overflow flag.
--	PSW.1	user-definable bit
P	PSW.0	Parity flag. set/ cleared by hardware each instruction cycle to indicate an odd/even number of 1 bits in the accumulator.

Stack in the 8051.

→The stack is a section of RAM used by the CPU to store information temporarily. This information could be data or an address.

→ The register used to access the stack is called the stack pointer register. The stack pointer in the 8051 is only 8 bits wide, which means that it can take values of 00 to FFH.

→ when the 8051 is powered up, SP register contains value 07, which means that RAM location 08 is the first location used for ~~fl~~ the stack by the 8051

⇒ The storing of a CPU register in the stack is called PUSH & pulling the contents off the stack back into a CPU register is called a POP.

Pushing onto the Stack.

→ In the 8051, the stack pointer (SP) points to the last used location of the stack.

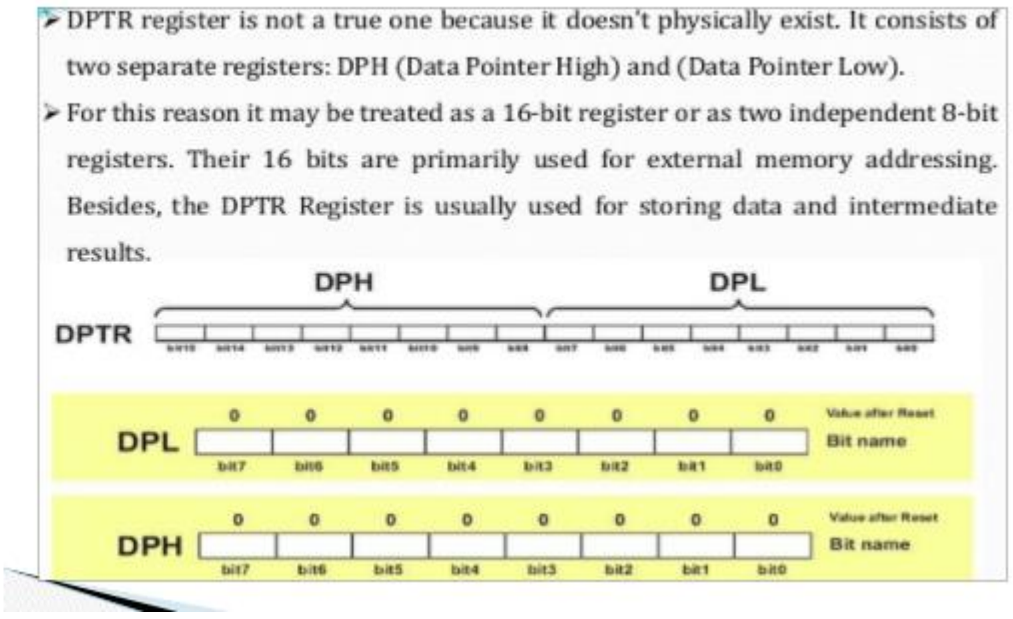
→ when the data is pushed onto the stack, the stack pointer (SP) is incremented by one.

POPPING from the stack

→ Popping the contents of the stack back into a given register is the opposite process of pushing.

→ with every pop, the top byte of the stack is copied to the register specified by the instruction & the stack pointer is decremented once.

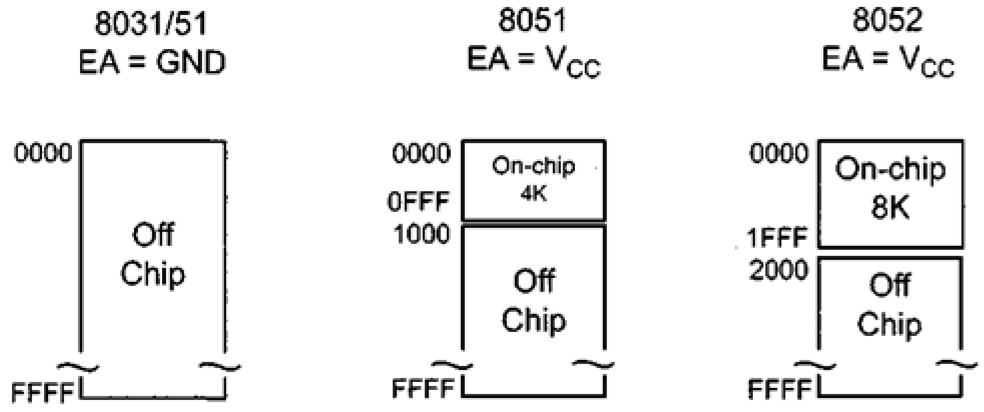
DATA POINTER



- DPTR register is not a true one because it doesn't physically exist. It consists of two separate registers: DPH (Data Pointer High) and (Data Pointer Low).
- For this reason it may be treated as a 16-bit register or as two independent 8-bit registers. Their 16 bits are primarily used for external memory addressing. Besides, the DPTR Register is usually used for storing data and intermediate results.

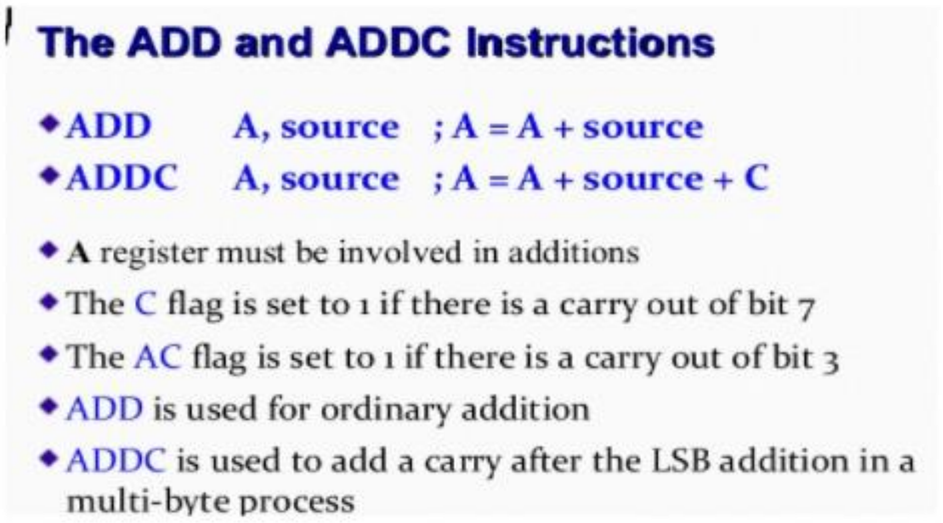
7.

1. When EA = 0, the EA pin is strapped to GND, and all program fetches are directed to external memory regardless of whether or not the 8751 has some on-chip ROM for program code. This external ROM can be as high as 64K bytes with address space of 0000 – FFFFH. In this case an 8751 (89C51) is the same as the 8031 system.
2. With the 8751 system where EA = V_{cc}, the microcontroller fetches the program code of addresses 0000 – OFFFH from on-chip ROM since it has 4K bytes of on-chip program ROM and any fetches from addresses 1000H – FFFFH are directed to external ROM.
3. With the 8752 system where EA = V_{cc}, the microcontroller fetches the program code of addresses 0000 – 1FFFH from on-chip ROM since it has 8K bytes of on-chip program ROM and any fetches from addresses 2000H – FFFFH are directed to external ROM.



8.

ADD & ADDC instruction



The ADD and ADDC Instructions

- **ADD   A, source   ; A = A + source**
- **ADDC   A, source   ; A = A + source + C**

- **A** register must be involved in additions
- The C flag is set to 1 if there is a carry out of bit 7
- The AC flag is set to 1 if there is a carry out of bit 3
- ADD is used for ordinary addition
- ADDC is used to add a carry after the LSB addition in a multi-byte process

Subtraction

- In many microprocessors there are two different instructions for subtraction:
- SUB and SUBB (subtract with borrow).
- In the 8051 we have only SUBB. To make SUB out of SUBB, we have to make CY = 0 prior to the execution of the instruction.
- Therefore, there are two cases for the SUBB instruction:

(1) with CY = 0,

(2) with CY = 1

SUBB with CY=0

- Three steps are performed for every SUBB instruction by the internal hardware of the 8051 CPU, regardless of the source of the operands, provided that the addressing mode is supported.
- Take the 2′s complement of the subtrahend (source operand).
- Add it to the minuend (A).
- Invert the carry.

Unsigned multiplication

- The two registers A & B are required for multiplication & Division.
- MUL AB ; AxB , place 16 bit result in B & A
- A-13, B-06
- A-FA, B-12

```
MOV  A,#25H    ;load 25H to reg. A
MOV  B,#65H    ;load 65H in reg. B
MUL  AB        ;25H * 65H = E99 where
               ;B = 0EH and A = 99H
```

Unsigned Division

- In the division of unsigned numbers, the 8051 supports byte over byte only. 10/02
- The syntax is as follows.
  DIV AB ; Divide A by B
  After the DIV instruction is performed, the quotient is in A and the remainder is in B

```
MOV  A,#95       ;load 95 into A
MOV  B,#10       ;load 10 into B
DIV  AB        . ;now A = 09 (quotient) and
                 ;B = 05(remainder)
```