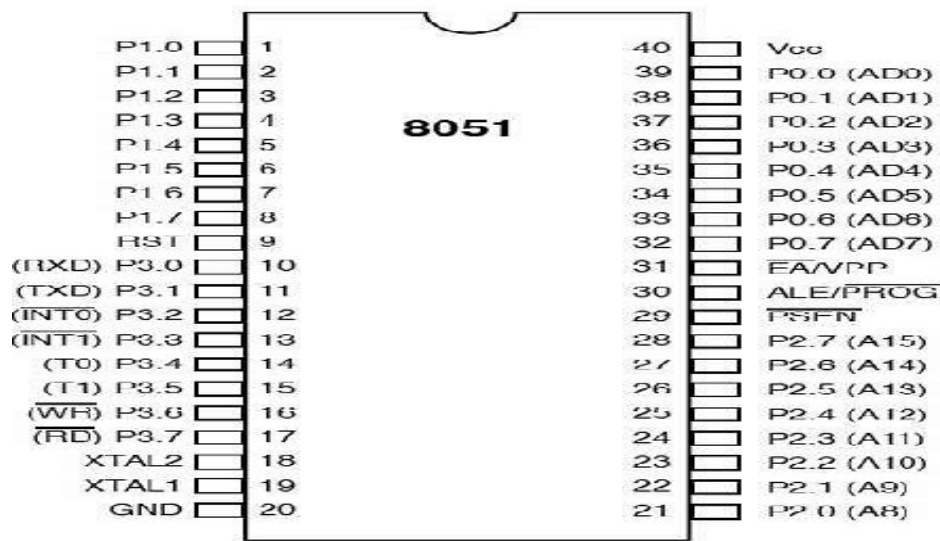


USN

Internal Assessment Test – I

Sub:	<b>Microcontrollers</b>						Code:	18EE52	
Date:	11/11/2021	Duration:	90 mins	Max Marks:	50	Sem:	5 <sup>th</sup>	Branch:	EEE
Answer <b>Any FIVE FULL</b> Questions									
							Marks	OBE	
								CO	RBT
2.	Explain the memory organization of 8051 with suitable diagrams. <b>Internal RAM Organization diagram- 3 marks</b> <b>Working register bank-2 marks</b> <b>Bit Addressable -2 marks</b> <b>Scratch pad-2 Marks</b> <b>ROM- 1 marks</b>						10	CO2	L1
4	Explain with suitable example, the different types of addressing modes used in 8051. <b>Five Addressing modes along with examples</b> <b>1. Immediate Addressing mode-2 marks</b> <b>2.Register addressing mode-2 marks</b> <b>3.Indirect Addressing mode-2 marks</b> <b>4. Register Indirect Addressing Mode- 2 marks</b> <b>5. Indexed Addressing mode- 2Marks</b>						10	CO1	L1
5	With a neat block diagram of 8051, brief about its salient features. <b>Block Diagram- 4 Marks</b> <b>Salient Features-6 Marks</b>						10	CO2	L1
6	Explain in brief for the following 1. Program Counter & DPTR 2. PSW3.Stack Pointer 4.Register Bank <b>Each Brief explanation – 2.5 marks</b>						10	CO1	L4
7	Compare the following <b>i)CISC &amp; RISC –any five difference -5M</b> <b>ii) Microprocessor &amp; Microcontroller- any five difference -5M</b>						10	CO1	L3
8	Write an ALP to toggle all the bits of P0 & P1- <b>ALP -5M</b> b. Write an ALP to take the complement of value 68h for 1000 times. <b>ALP -5M</b>						10	CO2	L4
1.	Explain the four ports of 8051 in detail. <b>Four ports explanation-2.5m each for ports</b>						10	CO4	L2
3.	Explain the logical instructions of 8051 with examples <b>AND, OR, XOR, CPL explanation with example-2.5M each</b>						10	CO2	L2

**1. Ports of 8051**



- To access the pins of port 0 as input & output ports, each pin must be connected externally to a 10K 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

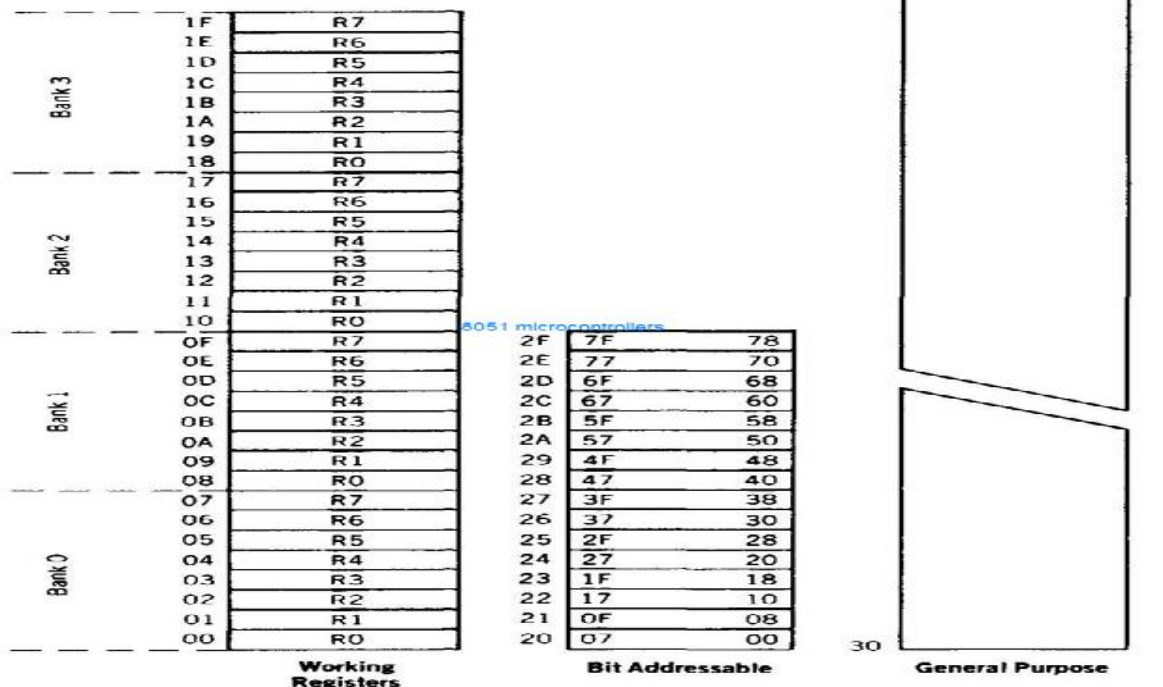
- Total of 8 pins.
- Can be used as an input or output.
- Does not require any pull up resistors
- If port 1 has been configured as an output port, to make it an input port again, it can be programmed by writing 1 to all its bits.

#### Port-2

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

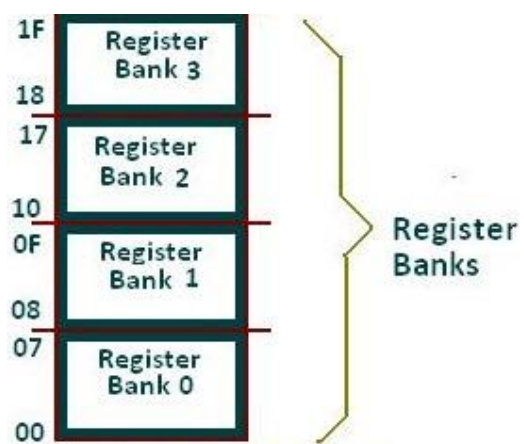
#### Port-3

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



2.

- 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	Bank 3
18	
17	Bank 2
10	
0F	Bank 1
08	
07	Default register bank for R0-R7
00	

} Bit-addressable locations

## Addressing Modes.

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 five 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 into any of the registers including the DPTR register.

→ The immediate data must be preceded by the pound 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, #4512H ; 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 ~~copied~~ 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 & 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  
Destination 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 ; Copy 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 registers R0 & R1 are used for internal RAM indirect data transfer.

When they hold the address of ROM locations they must be 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.

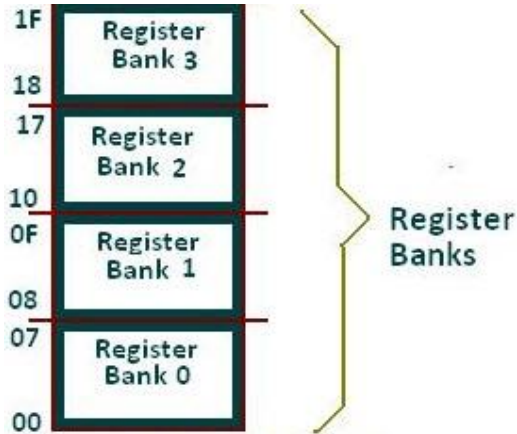
### 5. Indexed Addressing Mode

Indexed addressing mode is 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 MC 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	Auxiliary flag
FO	PSW.5	Available to the user for general purpose
RS1	PSW.4	Register Bank selector bit 1.
RS0	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 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.

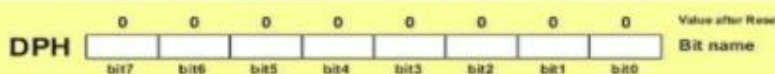
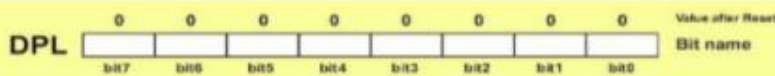
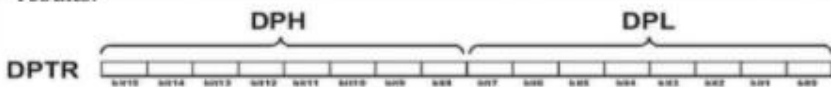
- In the 8051, the stack pointer (SP) points to the last used location of the stack.
- When the data is pushed onto the stack, the stack pointer (SP) is incremented by one.

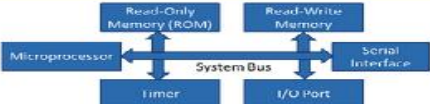

## POPPING from the stack

- 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.



Microprocessor	Micro Controller
	
Microprocessor is heart of Computer system.	Micro Controller is a heart of embedded system.
It is just a processor. Memory and I/O components have to be connected externally	Micro controller has external processor along with internal memory and i/O components
Since memory and I/O has to be connected externally, the circuit becomes large.	Since memory and I/O are present internally, the circuit is small.
Cannot be used in compact systems and hence inefficient	Can be used in compact systems and hence it is an efficient technique
Cost of the entire system increases	Cost of the entire system is low
Due to external components, the entire power consumption is high. Hence it is not suitable to use with devices running on stored power like batteries.	Since external components are low, total power consumption is less and can be used with devices running on stored power like batteries.
Most of the microprocessors do not have power saving features.	Most of the micro controllers have power saving modes like idle mode and power saving mode. This helps to reduce power consumption even further.
Since memory and I/O components are all external, each instruction will need external operation, hence it is relatively slower.	Since components are internal, most of the operations are internal instruction, hence speed is fast.
Microprocessors have less number of registers, hence more operations are memory based.	Micro controller have more number of registers, hence the programs are easier to write.
Microprocessors are based on von Neumann model/architecture where program and data are stored in same memory module	Micro controllers are based on Harvard architecture where program memory and Data memory are separate
Mainly used in personal computers	Used mainly in washing machine, MP3 players

RISC	CISC
It is a Reduced Instruction Set Computer.	It is a Complex Instruction Set Computer.
It emphasizes on software to optimize the instruction set.	It emphasizes on hardware to optimize the instruction set.
It is a hard wired unit of programming in the RISC Processor.	Microprogramming unit in CISC Processor.
It requires multiple register sets to store the instruction.	It requires a single register set to store the instruction.
RISC has simple decoding of instruction.	CISC has complex decoding of instruction.
Uses of the pipeline are simple in RISC.	Uses of the pipeline are difficult in CISC.
It uses a limited number of instruction that requires less time to execute the instructions.	It uses a large number of instruction that requires more time to execute the instructions.
It uses LOAD and STORE that are independent instructions in the register-to-register a program's interaction.	It uses LOAD and STORE instruction in the memory-to-memory interaction of a program.
RISC has more transistors on memory registers.	CISC has transistors to store complex instructions.
The execution time of RISC is very short.	The execution time of CISC is longer.
RISC architecture can be used with high-end applications like telecommunication, image processing, video processing, etc.	CISC architecture can be used with low-end applications like home automation, security system, etc.

## 1. AND

ANL Destination, Source

- Performs a logical AND operation of two operands and place the result in the destination.
- The source operand can be register, memory or immediate.
- The destination: Accumulator

Eg: MOV A, #35H      0011 0101

ANL A, #0FH      0000 1111

0000 0101

XRL destination, source

- Performs a logical XOR operation of two operands and place the result in the destination.
- The source operand can be register, memory or intermediate.
- The destination: Accumulator
- Used to check if two registers have the same value.

Eg: XRL A,R1

XRL destination, source

- Performs a logical XOR operation of two operands and place the result in the destination.
- The source operand can be register, memory or intermediate.
- The destination: Accumulator
- Used to check if two registers have the same value.

Eg: XRL A,R1

- This instruction complements the contents of registers A

Eg 1: MOV A, #85H    1000 0101 -85

CPL A              0111 1010 – 7A

ADD A,#1          0111 1011-7B

Eg 2:MOV A, #55H

CPL A ;now A=AAH ;    0101 0101(55H)

                          ;becomes 1010 1010(AAH)

## 8.

```
ORG 0H ; START PROGRAM FROM 0H
```

```
START: MOV A, #0 ;
```

```
MOV P1, A ;
```

```
Mov p0,a
```

```
MOV R0, #30H ;
```

```
ACALL DELAY ; CALL DELAY
```

```
CPL A ; COMPLEMENT A TO TOGGLE
```

```
MOV P1, A ; TOGGLE P1 BY TRANSFERING A
```

```
Mov p0,a  
MOV R0, #OFFH ; CHANGE R0 VALUE TO CHANGE DELAY  
ACALL DELAY ; CALL DELAY WITH DIFFERENT R0  
SJMP START ; JUMP BACK TO START
```

```
DELAY: NOP ; DELAY SUBROUTINE  
LOOP: DJNZ R0, LOOP ;  
RET ;
```

END

ORG 00H

Mov a,#68H

MOV R3,#10

NEXT: MOV R2,#10

AGAIN: CPL A

DJNZ R2, AGAIN

DJNZ R3, NEXT

END