



Pune Institute of Computer Technology, Pune

Session on Introduction to Microcontroller



Mr.D.M.Shinde

Assistant Professor,

Department of E&TC, PICT, Pune

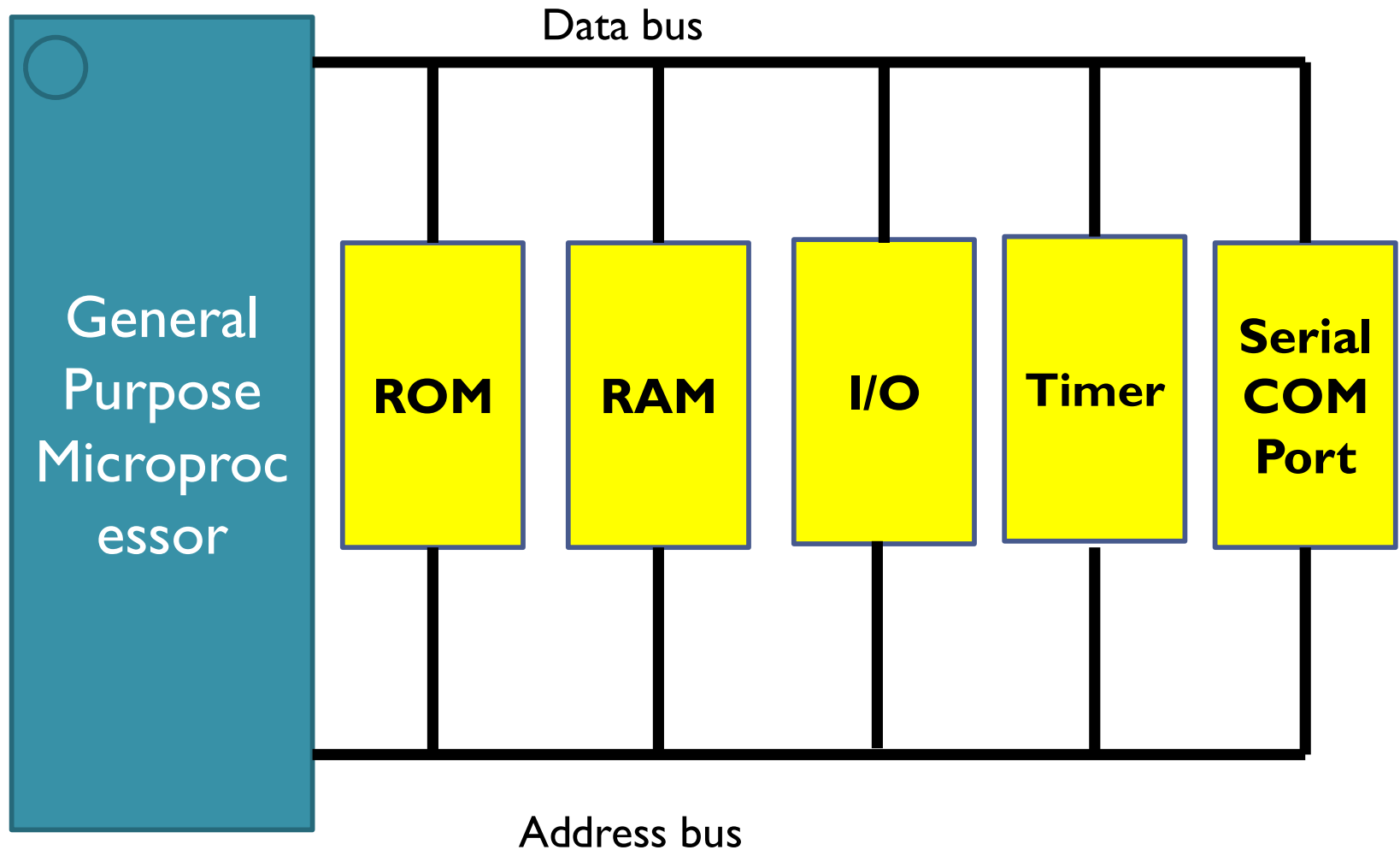
dmshinde@pict.edu



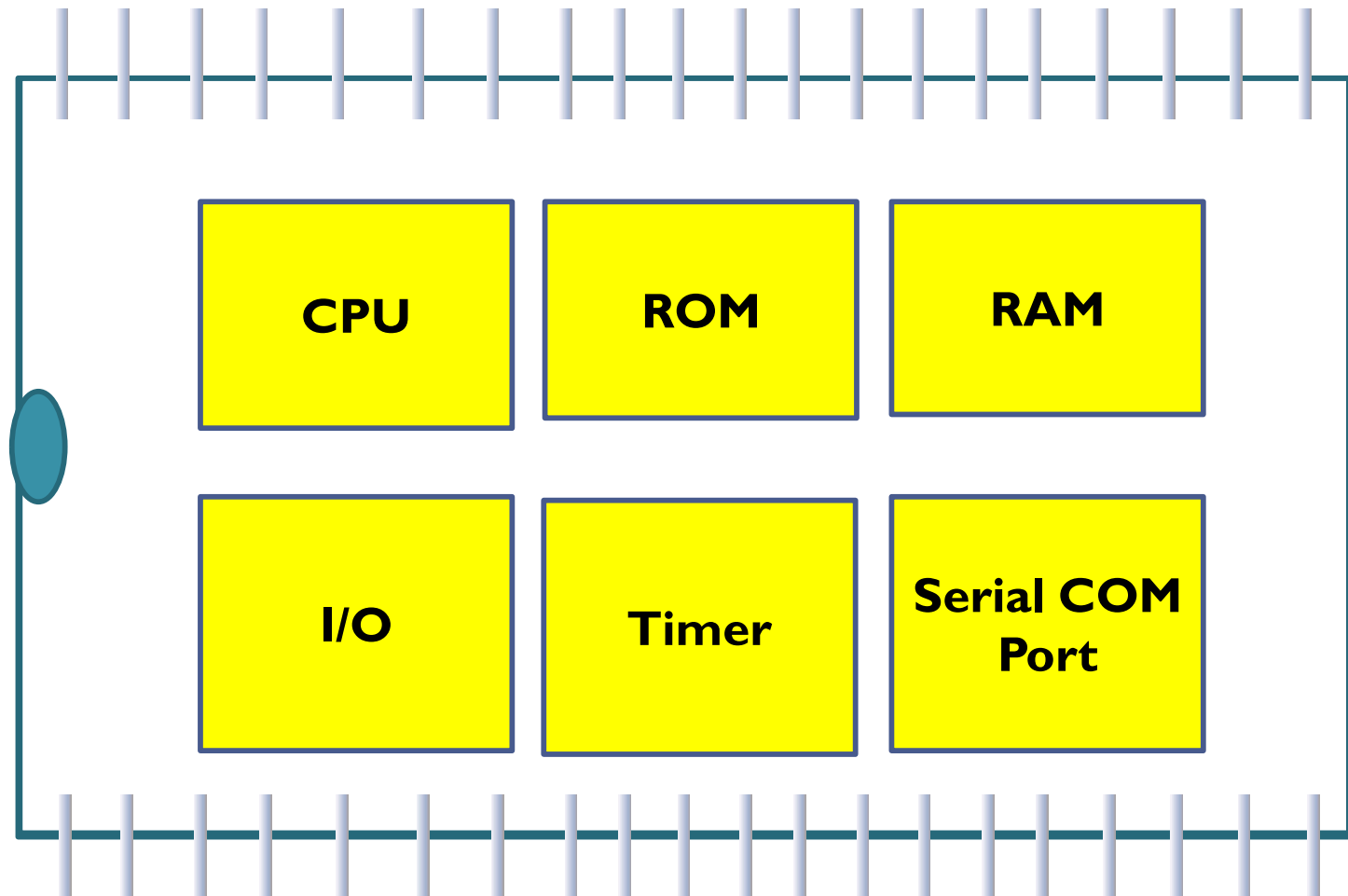
Session contents

- Introduction
- 8051 Assembly Language Programming
- 8051 Flag Bits And The PSW Register
- 8051 Register Banks and STACK
- 8051 Addressing Modes
- The 8051 Instruction Set

What is Microprocessor



What is microcontroller





Criteria for Choosing a μ C

- Meeting the computing needs of the task at hand efficiently and cost effectively.
- Availability of software development tools.
- Wide availability and reliable sources of μ C.



Overview of 8051

Feature	8051	8052	8031
ROM(on-chip program space in bytes)	4K	8K	0K
RAM(bytes)	128	256	128
Timers	2	3	2
I/O pins	32	32	32
Serial Port	1	1	1
Interrupt sources	6	8	6



Various 8051 Microcontroller

8051 offers different memory types, such as

UV-EPROM

Flash

NV-RAM

OTP

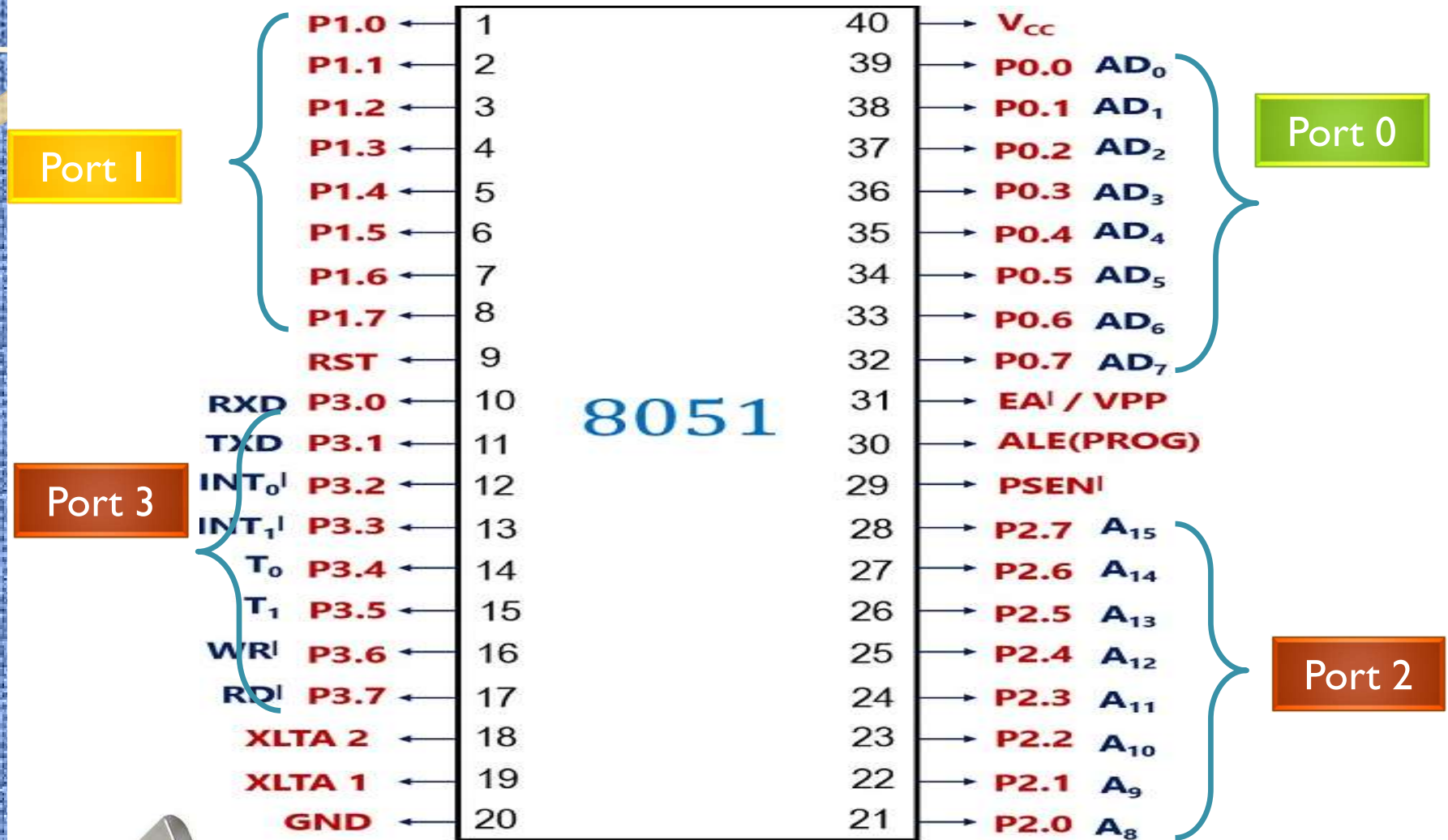
AT89C51 from Atmel corporation:

This 8051 chip has on-chip ROM in the form of flash memory.

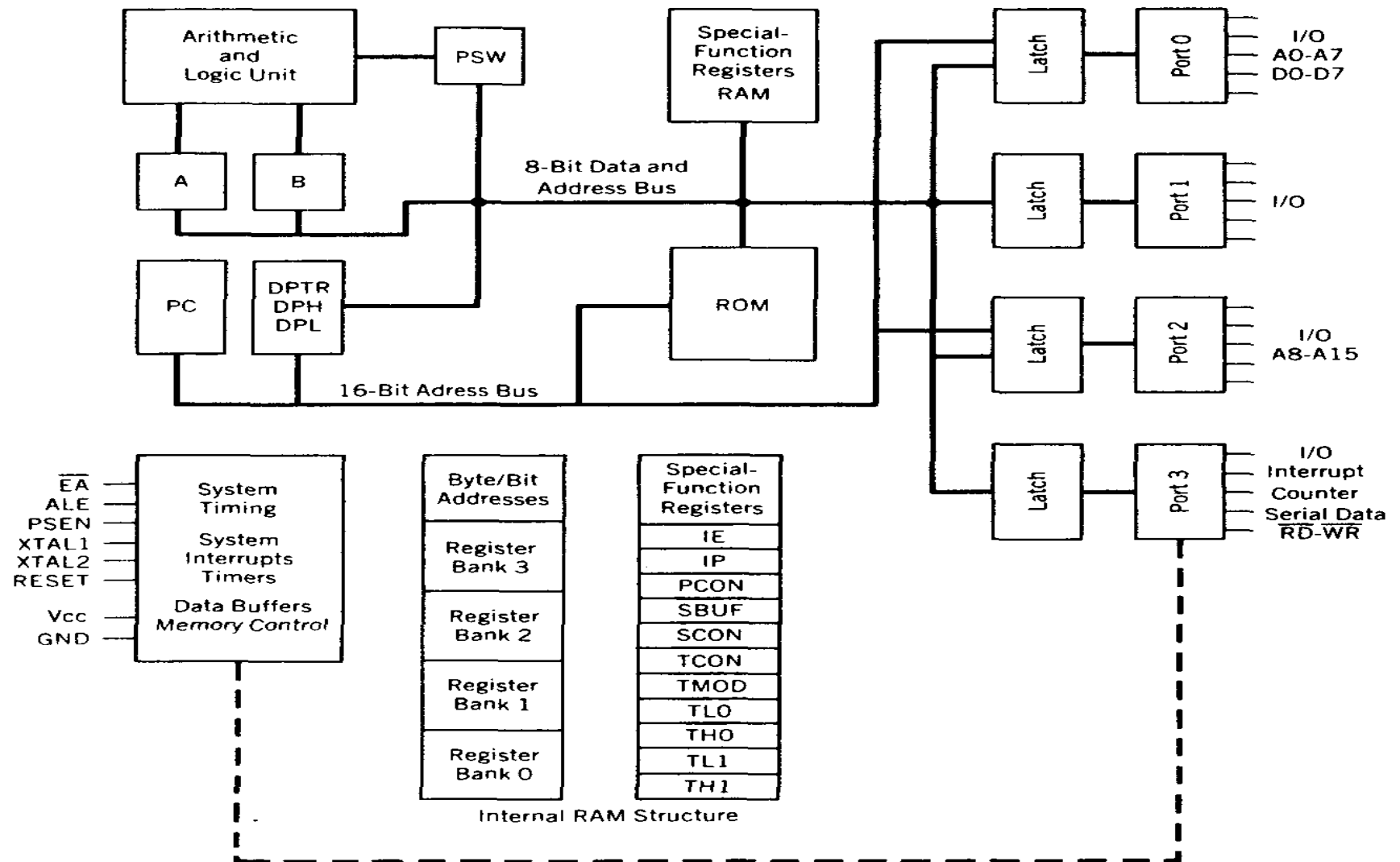
P89C51RD2BN from Phillips:

It is another major producer of 8051 family from Phillips Corporation

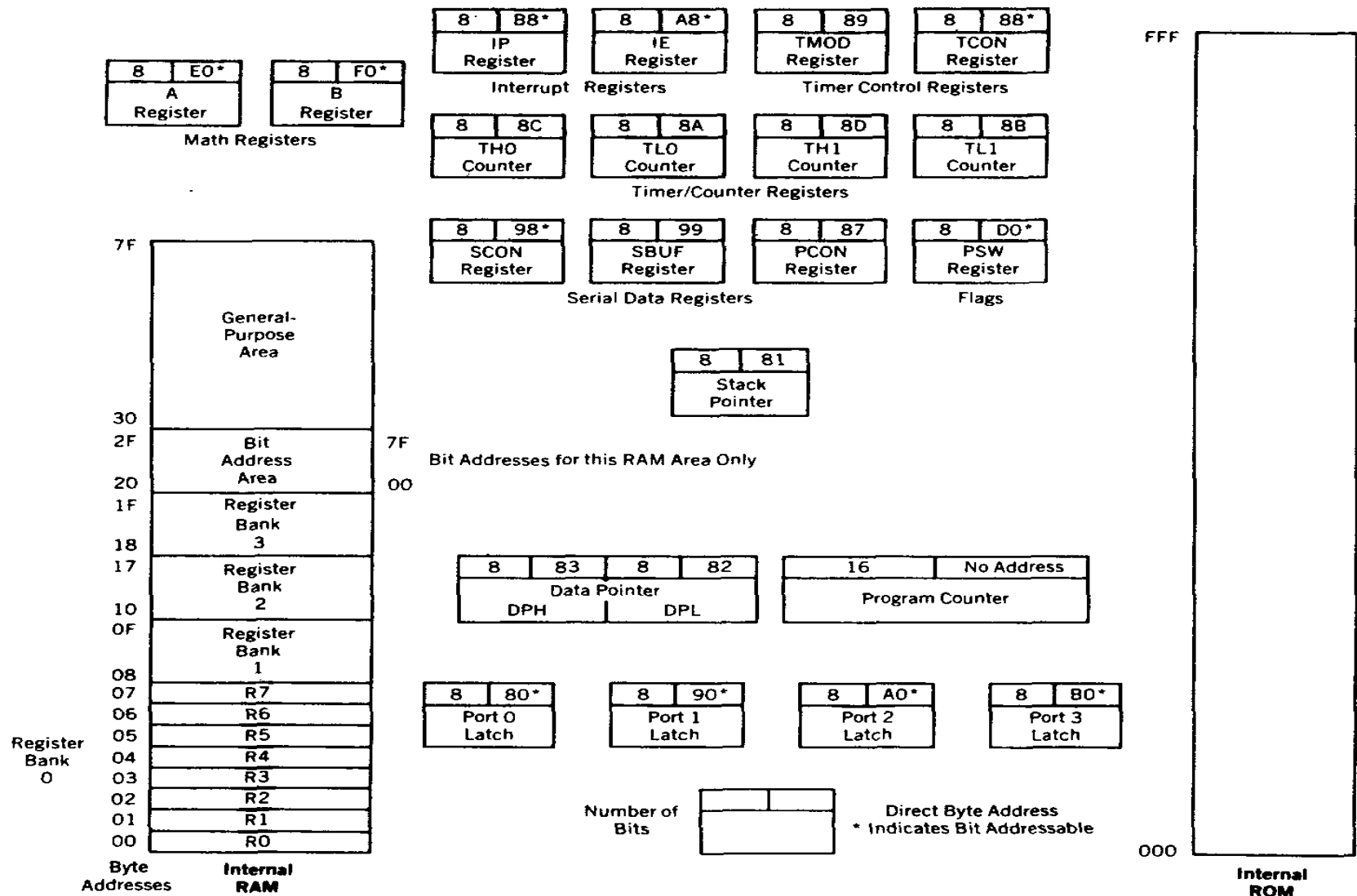
8051 Pin Configuration



Architecture of 8051

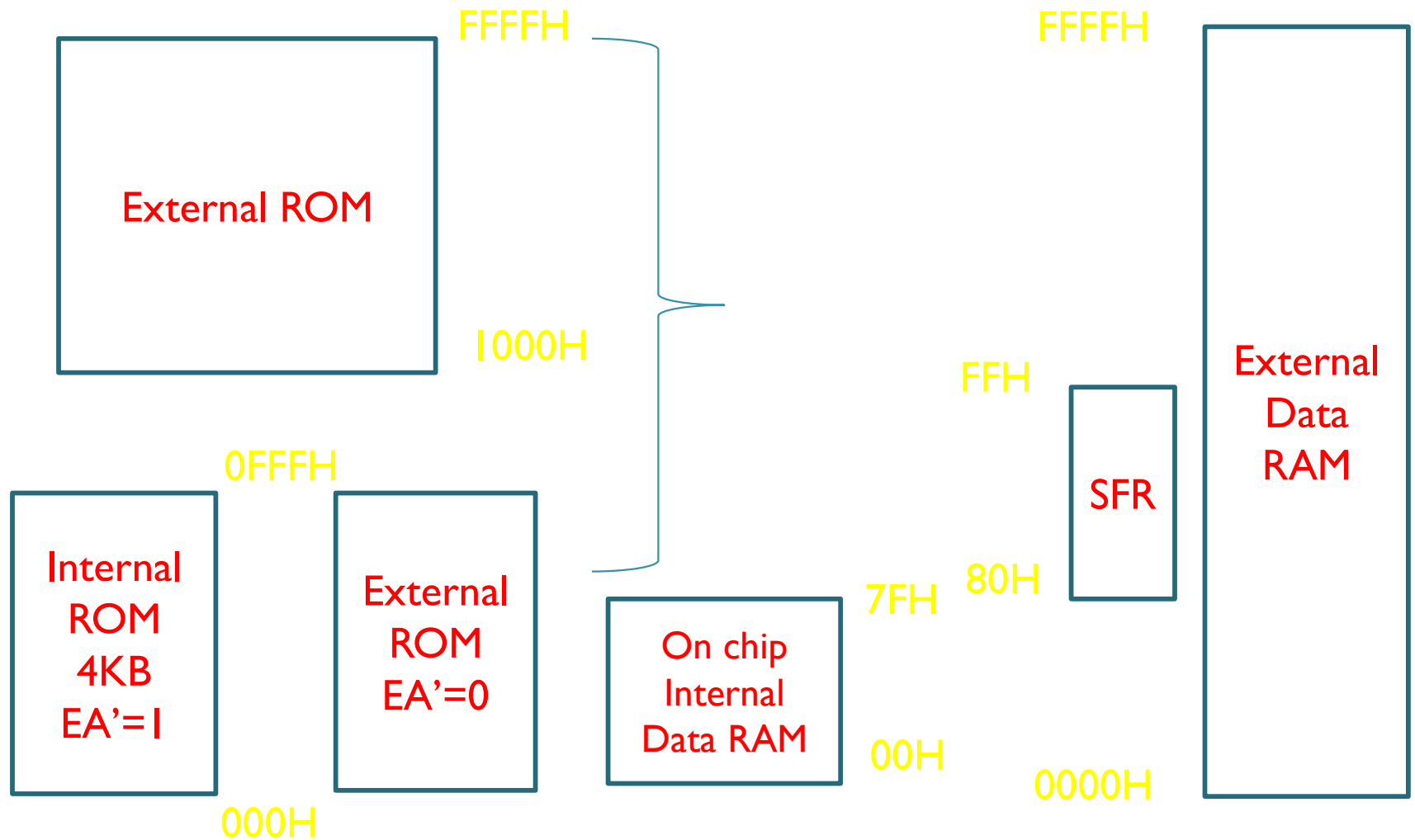


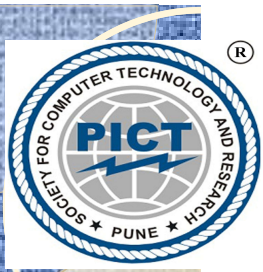
Programmable Model of 8051





Memory map of 8051





Introduction to 8051 Assembly language

- CPU works on: **0 & 1**
- What is Machine language?

A program consists:

0 & 1

- Is it suitable for human?

Answer is :

No

Introduced Assembly Language which provides mnemonics for the machine codes

Assembler : is a program which converts assembly language into machine level language



Major 8051 Register

MOV A,B
MOV A,RI
MOV R2,A

MOV RI,R2(Not Allowed)



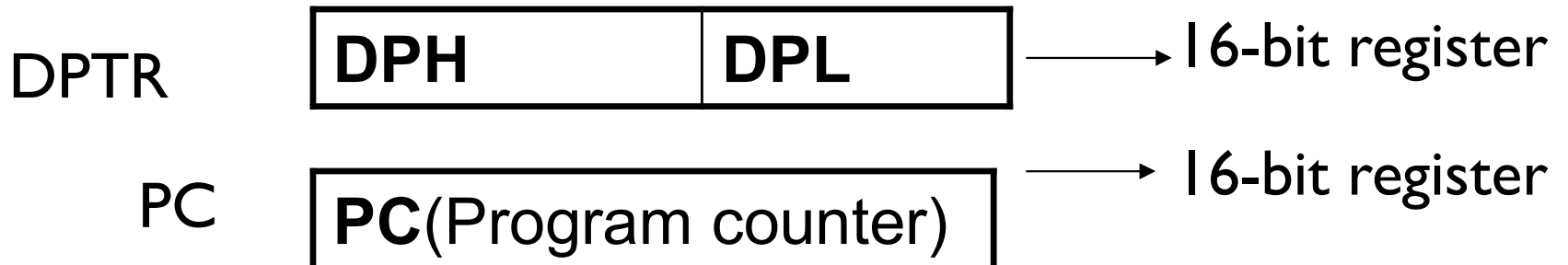
MSB bit ↑

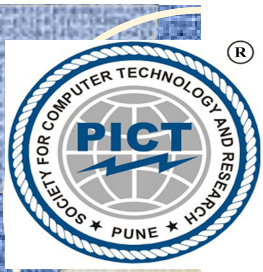
↑ LSB bit

Registers



8- bit Registers of the 8051





Structure of Assembly Language

An ***assembly language*** program consists of series of assembly language instructions .

An ***assembly language*** instruction consist of *mnemonic* , optionally followed by one or more *operands*.

Example:

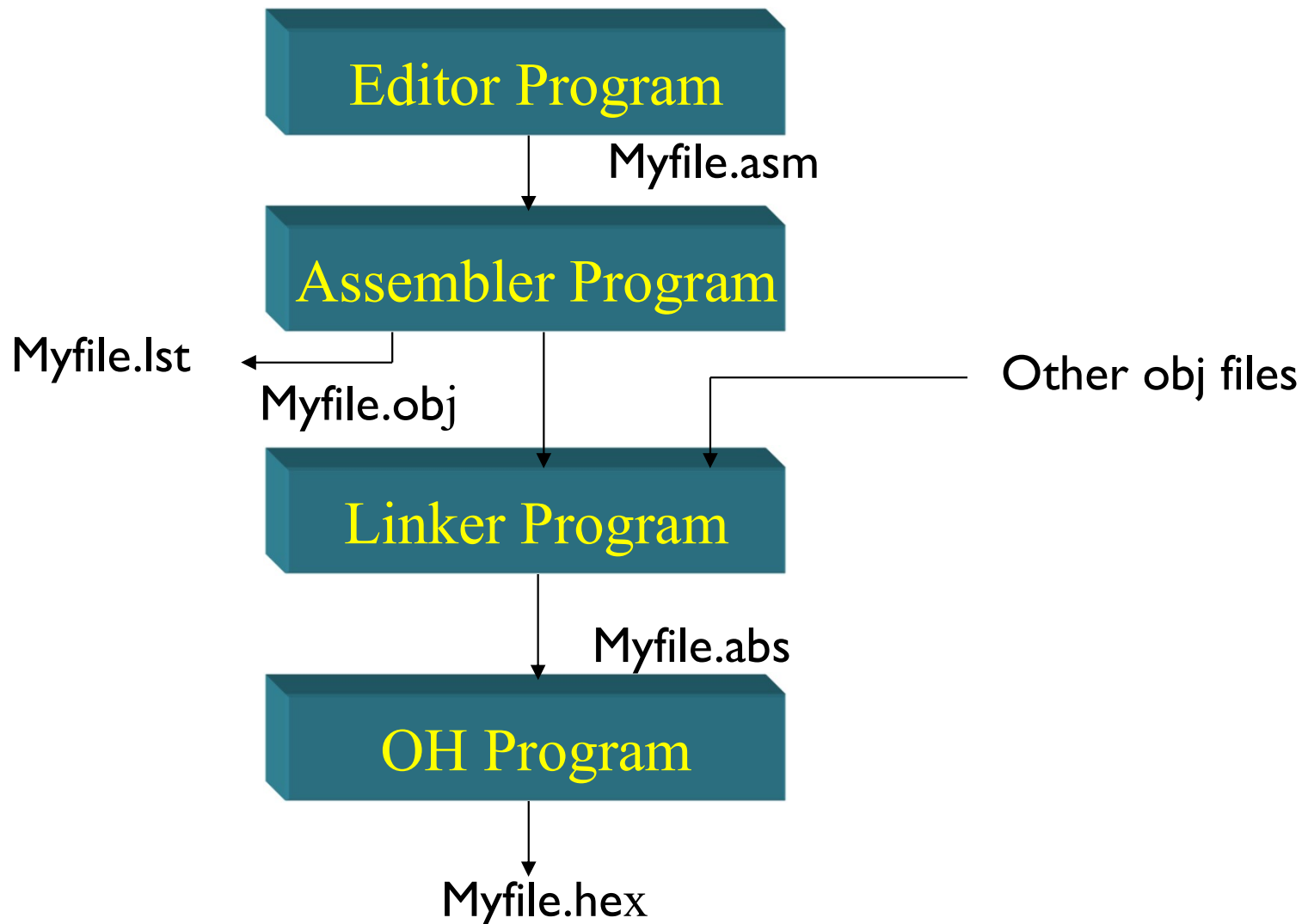
ORG 0000H

MOV A,B

ADD A,B

END

Assembling and Running An 8051 Program



8051 Data Types and Directives

Data Type

8051 micro controller has only one data type. It is of 8-bits, and the size of each register is also 8 bits.

DB(Define byte)

DB directive is used to define data, the numbers can be in decimal, binary, hex, or ASCII Formats

Examples:

DB 25;

DB 25H;

DB 00001B;

Assembler Directives

ORG(origin)

The **ORG** directive is used to indicate the beginning of the address

EQU(equate)

This is used to define a constant without occupying a memory location.

END Directive

This indicates to the *assembler* the end of the source(asm) file



8051 Flag bits and PSW Register

- PSW is a flag register in the 8051
- It is used to indicate arithmetic conditions
- The PSW register is an 8-bit register
- The two unused bits are user-definable flags
- Four of the flags are conditional flags
These are CY,AC,P,OV

PSW Register

D7	D6	D5	D4	D3	D2	D1	D0
CY	AC	F0	RSI	RS0	OV	--	P
• CY		PSW.7					Carry flag
• AC		PSW.6					Auxiliary carry flag
• F0		PSW.5					Future used
• RSI		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

Register Bank Selection

RS1	RS0	Register Bank	Address
0	0	0	00H-07H
0	1	1	08H-0FH
1	0	2	10H-17H
1	1	3	18H-1FH

Example:

- Show the status of the CY,AC,OV and P flags after the addition of ABH and CD H in the following instructions

MOV A,#ABH

ADD A,#CDH

Solution:

AB	1 0 1 0 1 0 1 1
+ <u>CD</u>	1 1 0 0 1 1 0 1
	<hr/>
	1 0 1 1 1 1 0 0 0

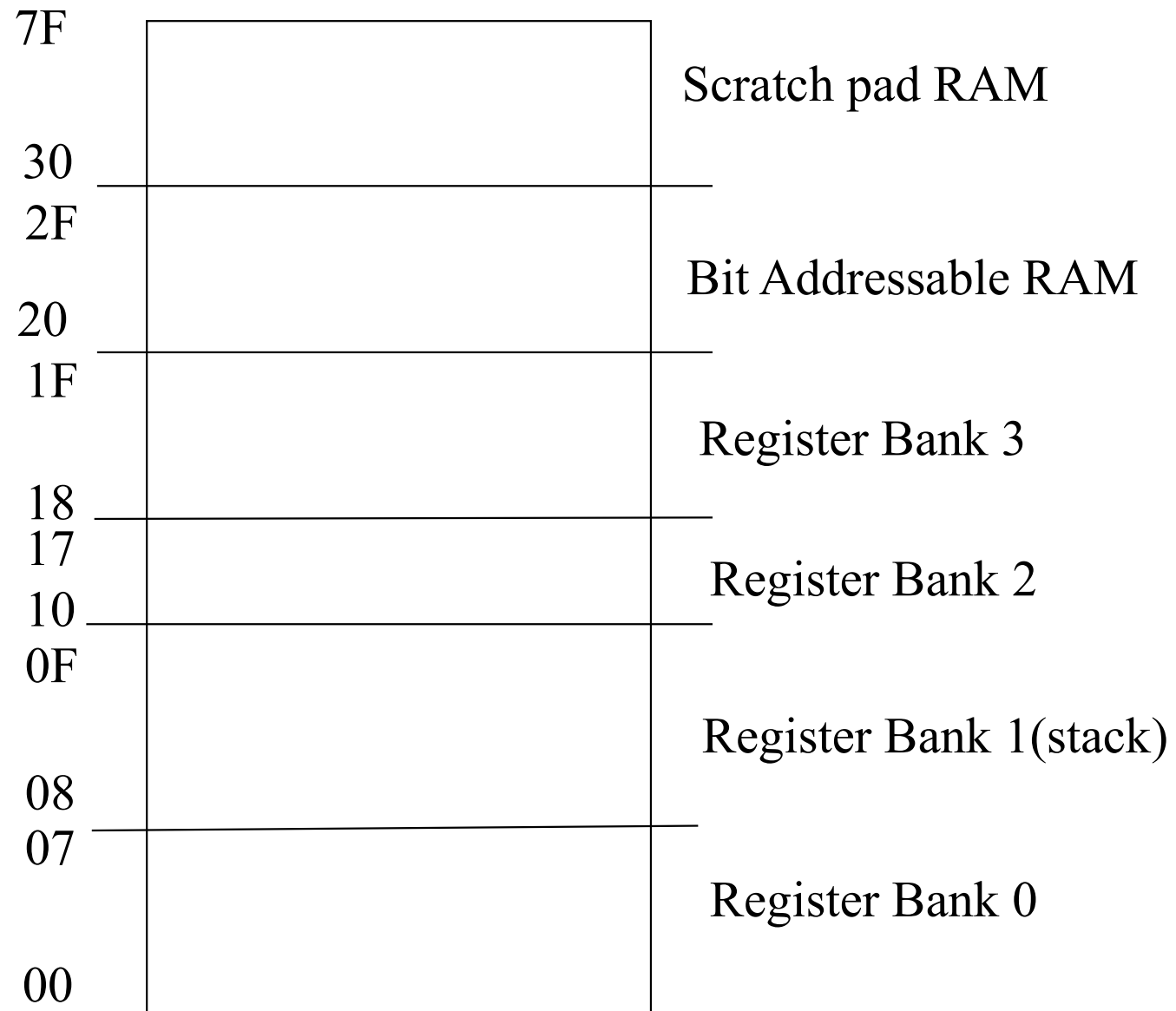
CY=1

AC=1

OV =1

P =0

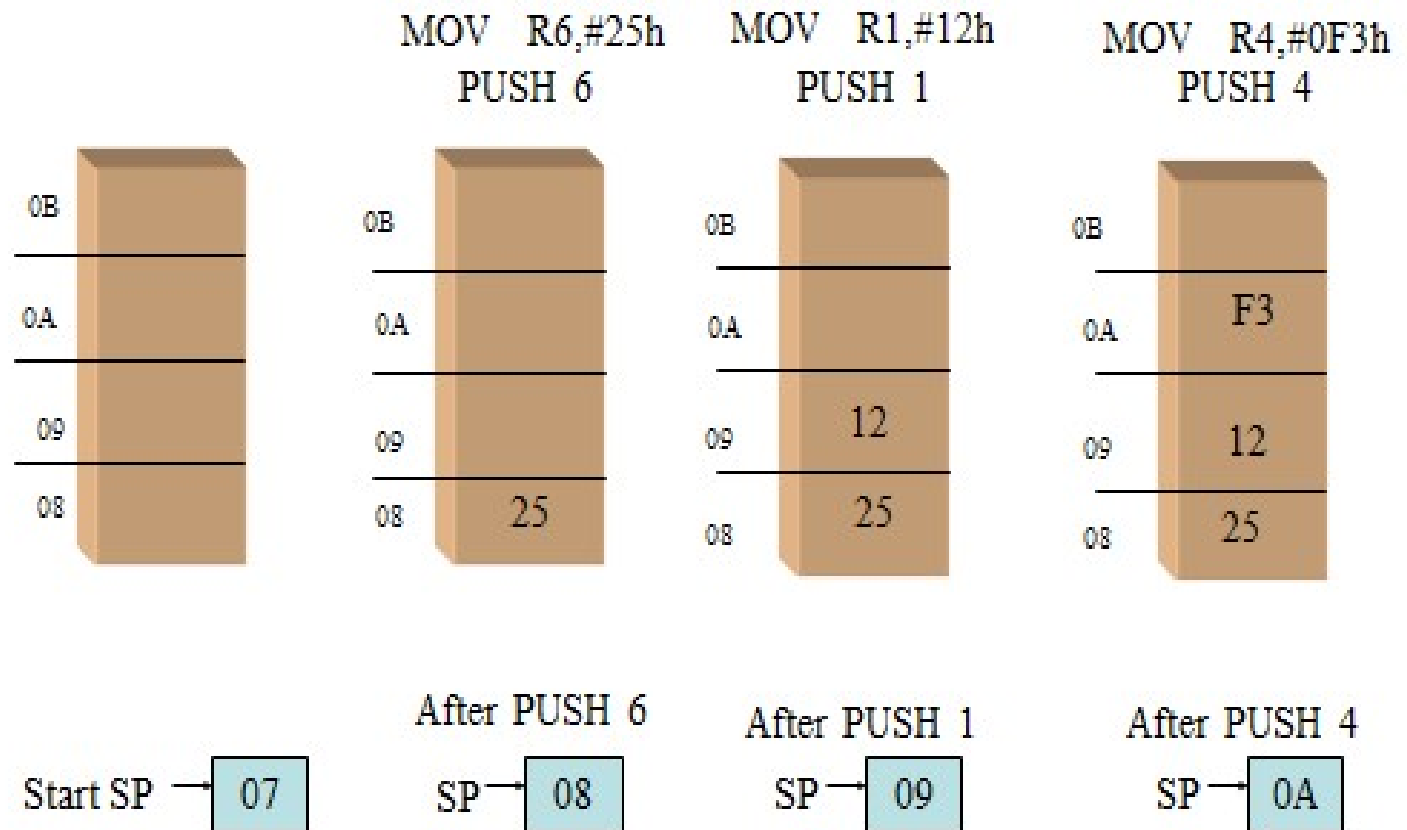
RAM Allocation in 8051



Bit Addressable Locations	7F 30 2F	General purpose RAM							
		7F	7E	7D	7C	7B	7A	79	78
	⋮								
	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	Bank3							
	18 17 10	Bank2							
	0F 08	Bank1							
	07 00	Default register bank for R0-R7							

8051 STACK

Push operation

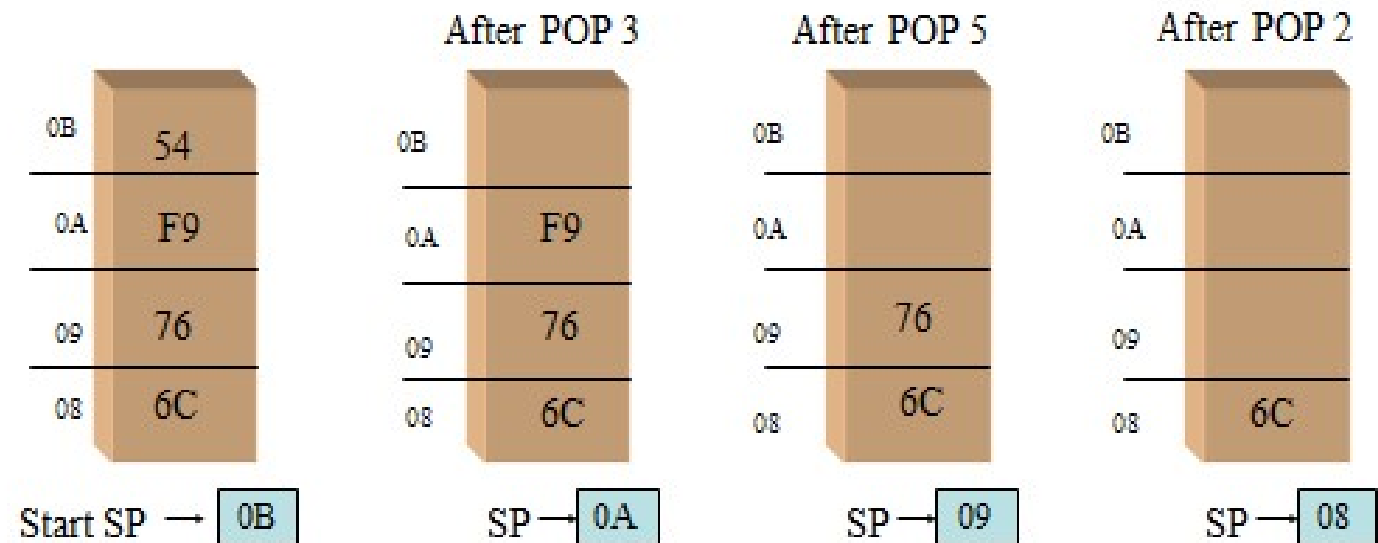


8051 STACK

POP 3 ; POP stack into R3

POP 5 ; POP stack into R5

POP 2 ; POP stack into R2



Addressing Modes

There are Five addressing modes available in the 8051:

- Register
- Direct
- Register Indirect
- Immediate
- Indexed

Register Addressing mode

It involves the use of Registers to hold the data to be manipulated.

Example:

```
MOV    A,R0  
MOV    R2,A
```

Note:

We can move the data between Accumulator and register, but movement of data between registers is not allowed.

Review:

- 8-bit registers
- General purpose: R0, R1, ... R7
- Special function: A, B, PSW, SP, I/O ports...
- 16-bit registers
- DPTR (=DPH, DPL)

Direct Addressing mode

Allocation of RAM

00 – 1FH are assigned to the register banks and stacks

20 – 2FH are set aside as bit-addressable space to save single bit data

30 – 7FH are available as a place to save byte sized data

- In direct addressing mode, the data is in a RAM memory location whose address is known, and this address is given as a part of instruction.

MOV R0,40H

Register indirect addressing mode

A register is used as a pointer to the data. If the data is inside the CPU, only Registers **R0** and **RI** are used as pointers and they must Precede by '@' and **R2 –R7** cannot be used to hold the address of an operand located in RAM.

Example:

```
MOV R0,#05H;
```

```
MOV A,@R0;
```



Address	Value
05H	40
06H	30
07H	20
08H	10

Immediate Addressing Mode

This addressing mode is used to load information into any of the Registers, including **DPTR** Register. Source operand is a constant preceded by the sign '#'.

Example:

MOV A,#25H

MOV **R4**,#62

MOV **B**,#40H

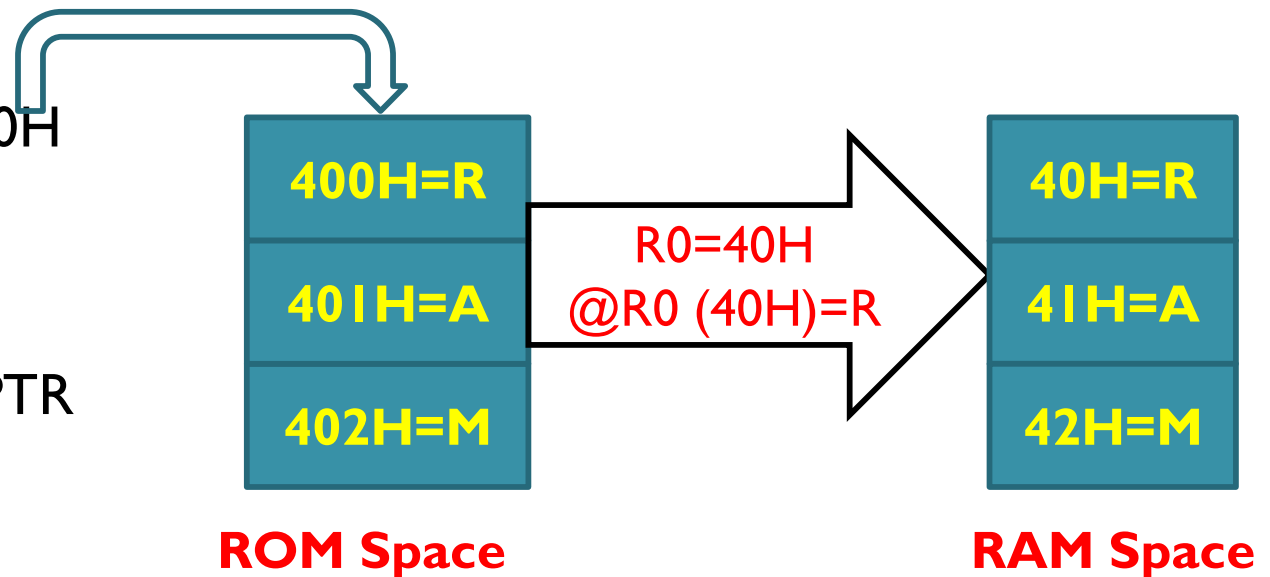
MOV **DPTR**,#4512H ;DPTR=4512H

Indexed addressing mode and on-chip ROM access

It is widely used in accessing data elements of look-up table entries located in the program ROM space of the 8051 by the instruction **MOVC A,@A+DPTR**

Example:

```
MOV DPTR,#0400H
MOV R2,#03H
MOV R0,#40H
LI: CLR A
MOVC A,@A+DPTR
MOV @R0,A
INC DPTR
INC R0
DJNZ R2,LI
SJMP $
```



Special function registers

Special function registers can be accessed by their names or by their addresses.

Example :

Register **A** has address **E0H**

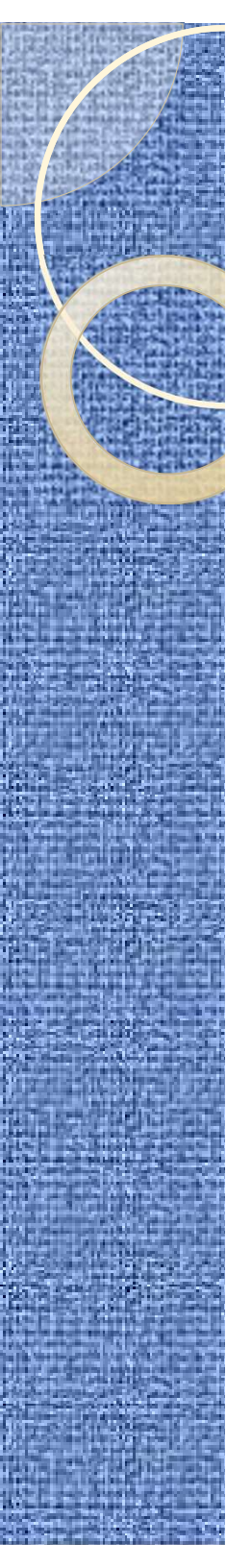
Register **B** has address **F0H**

MOV 0E0H ,#55H OR MOV **A**,#55H

MOV 0F0H,#25H OR MOV **B**,#25H

Special function register (SFR) addresses

<u>Symbol</u>	<u>Name</u>	<u>Address</u>
P0	Port0	80H
P1	Port1	90H
P2	Port2	0A0H
P3	Port3	0B0H
PSW	Program status word	0D0H
ACC	Accumulator	0E0H
B	B register	0F0H
SP	Stack Pointer	81H
DPTR	Data pointer 2 bytes	
DPL	Low byte	82H
DPH	High byte	83H
IP	Interrupt priority control	0B8H
IE	Interrupt enable control	0A8H



<u>Symbol</u>	<u>Name</u>	<u>Address</u>
TMOD	Timer/counter mode control	89H
TCON	Timer/counter control	88H
T2CON	Timer/counter 2 control	0C8H
T2MOD	Timer/Counter 2 mode control	0C9H
TH0	Timer/Counter 0 high byte	8CH
TL0	Timer/Counter 0 low byte	8AH
TH1	Timer/Counter 1 high byte	8DH
TL1	Timer/Counter 1 low byte	8BH
TH2	Timer/counter 2 high byte	0CDH
TL2	Timer/Counter 2 low byte	0CCH
RCAP2HT/C 2	capture register high byte	0CBH
RCAP2L T/C 2	capture register low byte	0CAH
SCON	Serial control	98H
SBUF	Serial data buffer	99H
PCON	Power control	87H



Instruction Set

Instruction set are divided into Four types :

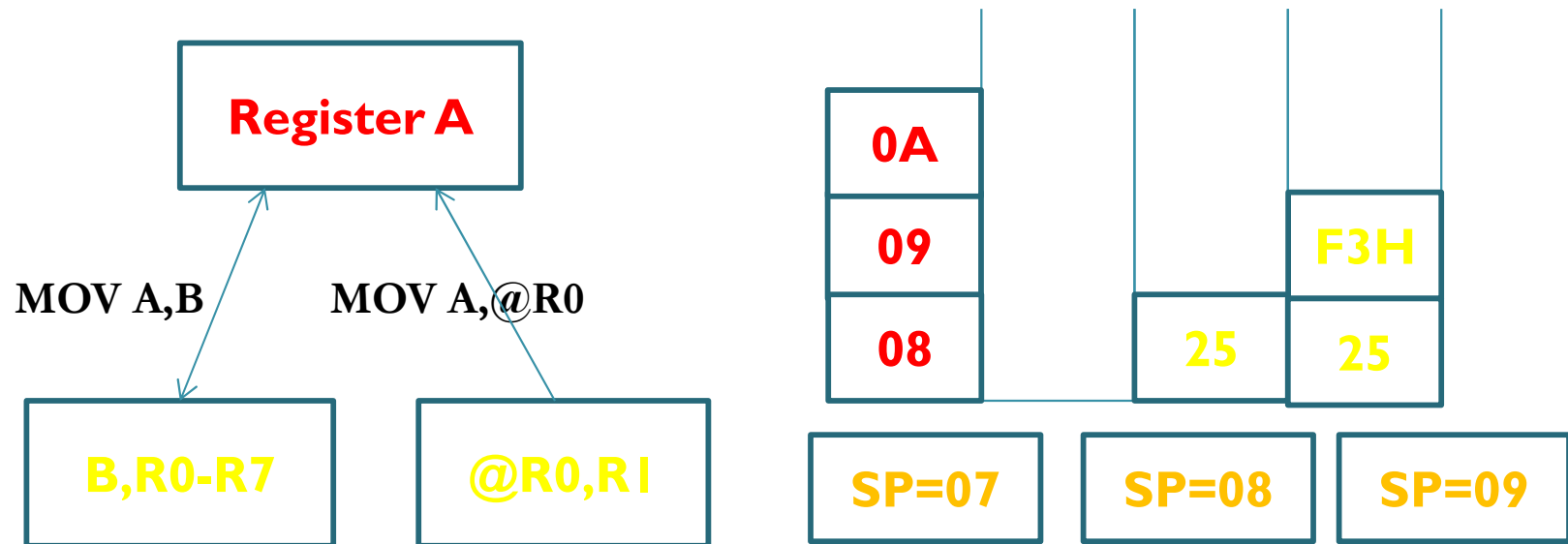
- Data Transfer
- Logical
- Arithmetic
- Control Transfer

Instruction Format:

[Label:] mnemonic [operands] [;comment]

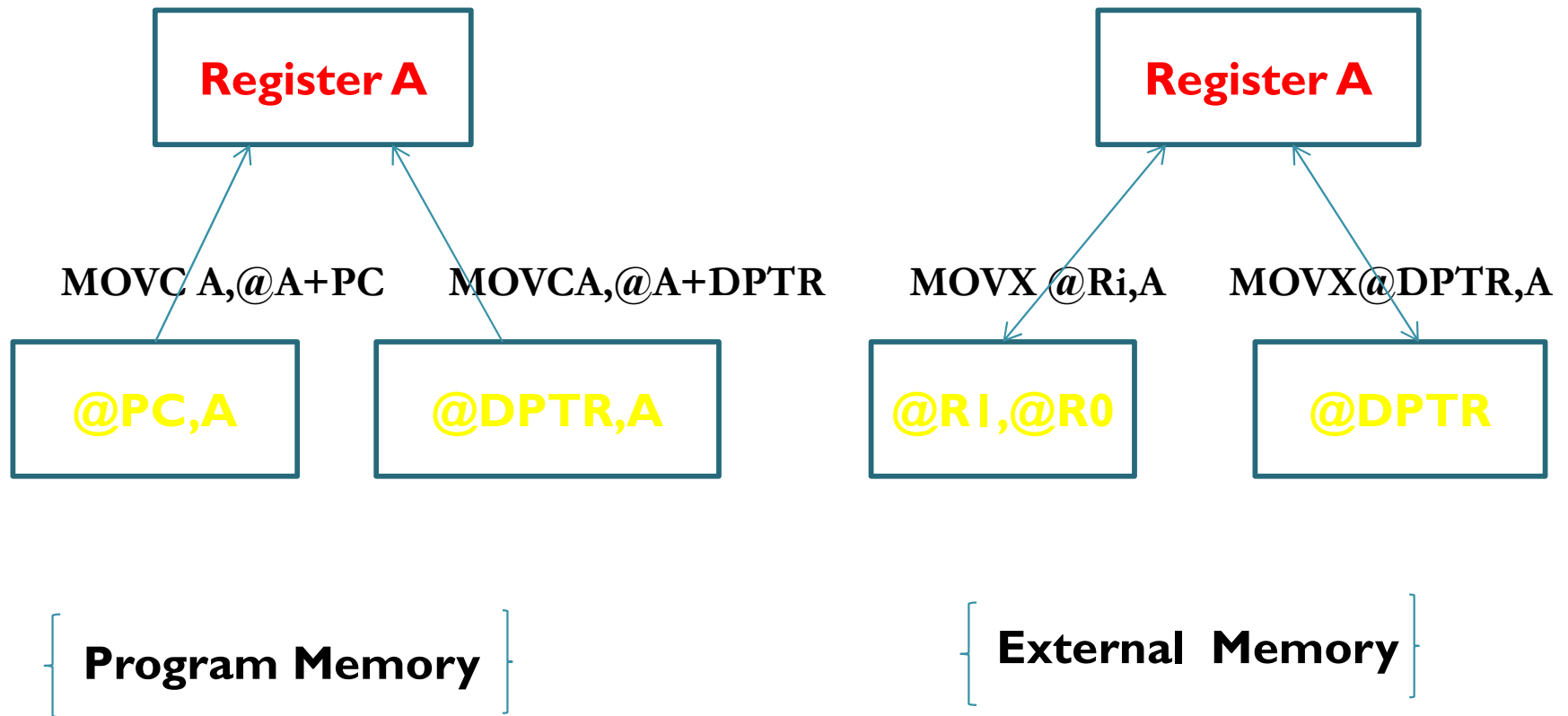
LI:MOV A,B; Copy the content from B to A

General Purpose Data Transfer



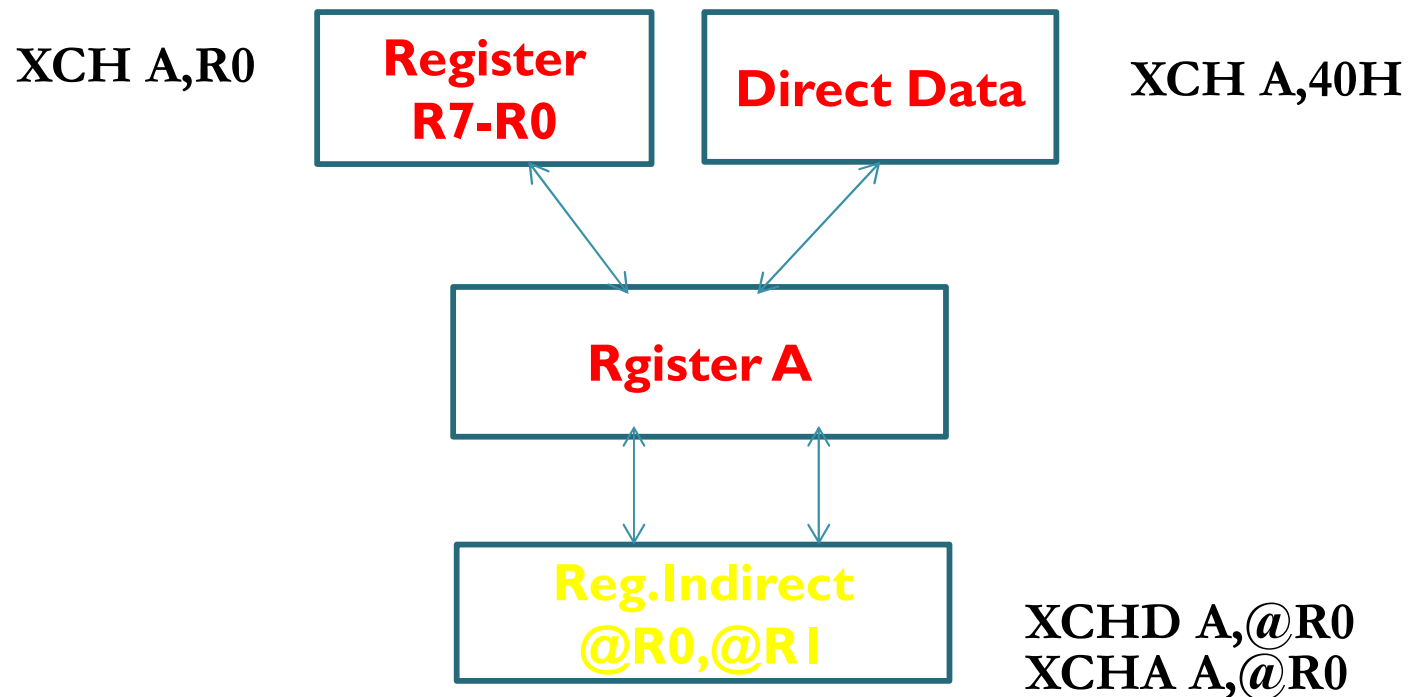
MOV DPL, #25H
MOV DPH, #0F3H
PUSH DPL
PUSH DPH

Memory Data Transfer Instruction



Accumulator Specific Data Transfer

Accumulator Specific Transfer



Logical Instruction

Single Operand

Clear- CLR A

SET-SETB P1.0

Complement- CPL A

Rotate Left- RL A

Rotate Left through Carry –RL C

Rotate Right through Carry-RR A

Swap nibble-SWAP A

Logical Instruction

Two Operand

AND - ANL A,R0

OR- ORL A,R0

XOR- XRL A,R0

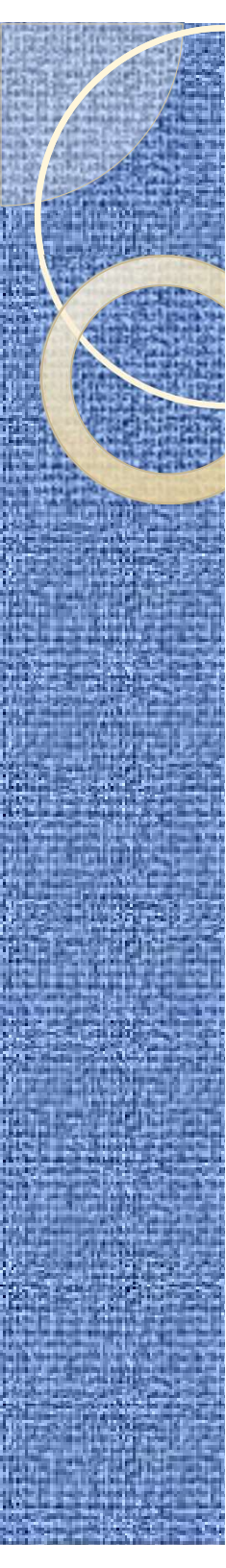
Logical Instruction

ANL dest byte,source byte

Example:

```
MOV  A,#39H
ANL  A,#09H
```

39 H	0	0	1	1	1	0	0	1
09 H	0	0	0	0	1	0	0	1
09 H	0	0	0	0	1	0	0	1



Arithmetic Instruction

INCA

DECA

ADD

ADDC

DA A

SUB B

MUL AB

DIV AB

Arithmetic Instruction

ADD A, source byte

Example:

```
MOV    A,#03H
ADD    A,#05H
```

ADDC A, source byte : Used in multibyte addition

Example: (25F2H to 3189H)

```
MOV    A,# 89H
ADDC   A,#0F2H      ;A=89H+F2H+0 =17BH
MOV    R3,A
MOV    A,#31H ;
ADDC   A,#25H ;A=31H+25H+1=57H
```

Therefore Result is : 577BH (A=57H & R3=7BH)

Arithmetic Instruction

DA A (Decimal –adjust accumulator after addition)

Two possible cases :

- It adds 6 to the lower 4 bits of A if it is greater than 9 or if AC=1
- It also adds 6 to the upper 4 bits of A if it is greater than 9 or if CY=1

Example:

```
MOV    A,#47H ;A=0100 0111
ADD    A,#38H ;A=7FH
DA      A      ;A= 1000 0101 =85H,valid BCD
```

47H	
+ 38 H	
<hr/>	
7FH	(Invalid BCD)
+6H	(After DA A)
<hr/>	
85H	(Valid BCD)

Control Transfer Instruction

Function : Transfer control to a subroutine

Two Types of Call:

- ACALL
- LCALL

Example:

Create a square wave of 50% duty cycle on bit 0 of port I

```
ORG 0000H
SETB  P1.0,
ACALL Delay
CLRB  P1.0
ACALL Delay
SJMP $
Delay: MOV R3,# 0FFH;
LI:    DJNZ R3,LI
RET
END
```



References

[1] Muhammad Mazidi- The 8051 Microcontroller and Embedded Systems, Pearson Education 2nd Edition.



End
Session



Questions if any!
Thanks

**Please write for queries, suggestions and
feedback : dmshinde@pict.edu**