Microcontroller

Microcontroller

- A single chip computer or a CPU with all the peripherals like RAM, ROM, I/O, Timers. ADCs etc. on the same chip.
- Microcontroller is meant to be more self contained and independent and functions as s a tiny, dedicated computer.
- Microcontroller can be used to perform control function so it is comparable with a microcomputer.
- The microcontroller is an embedded computer chip that controls most of the electronic gadgets and appliances people use on a daily basis, right from washing machine to anti lock brakes in a cars.
- It is also known as **SOC (System on Chip).**
- Examples : Motorola’s 6811, Intel’s 8051, Zilog’s Z8 and PIC16X

Evolution of Microcontroller/Microprocessor

FIRST GENERATION OF COMPUTERS
- ❑ Vacuum Tubes
- ❑ Magnetic Drums

1940-1956

SECOND GENERATION OF COMPUTERS
- ❑ Transistors
- ❑ High Level Language

1956-1963

THIRD GENERATION OF COMPUTERS
- ❑ Integrated Chips

1967-1971

INTEL 4004

1971

INTEL 8008

1972

INTEL 8085

1974

INTEL 8031

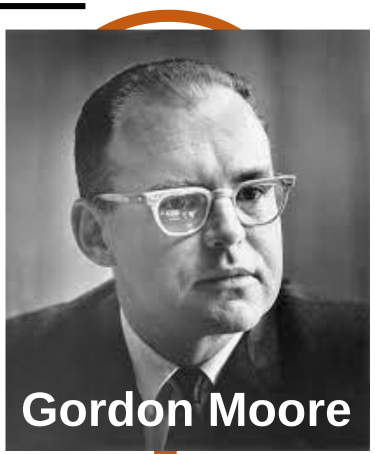
1976

INTEL 8086

1978

1980

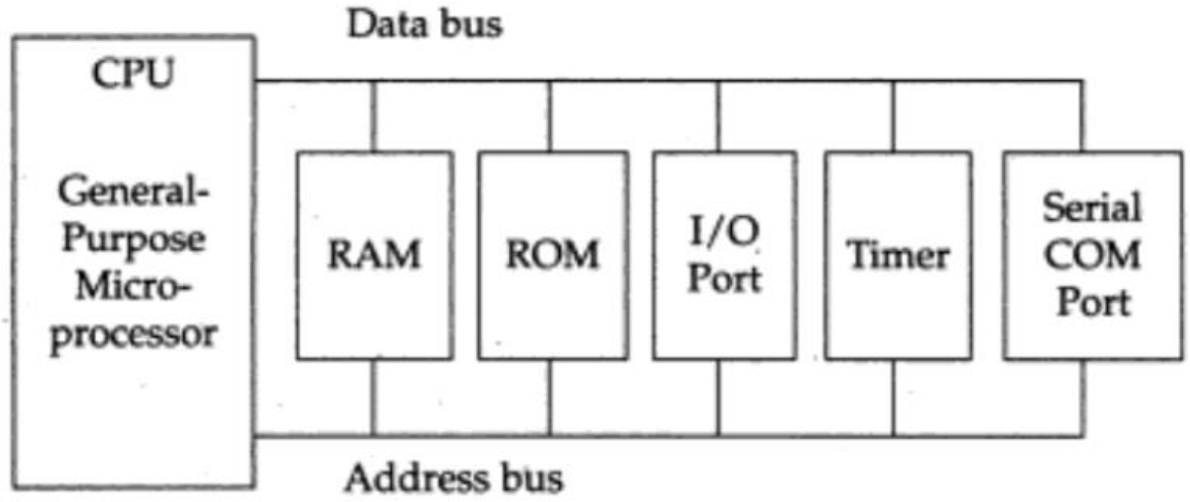
2000



Gordon Moore

Gordon E. Moore observed that the number of transistors per square inch on a computer chip was doubling about every 18–24 months.

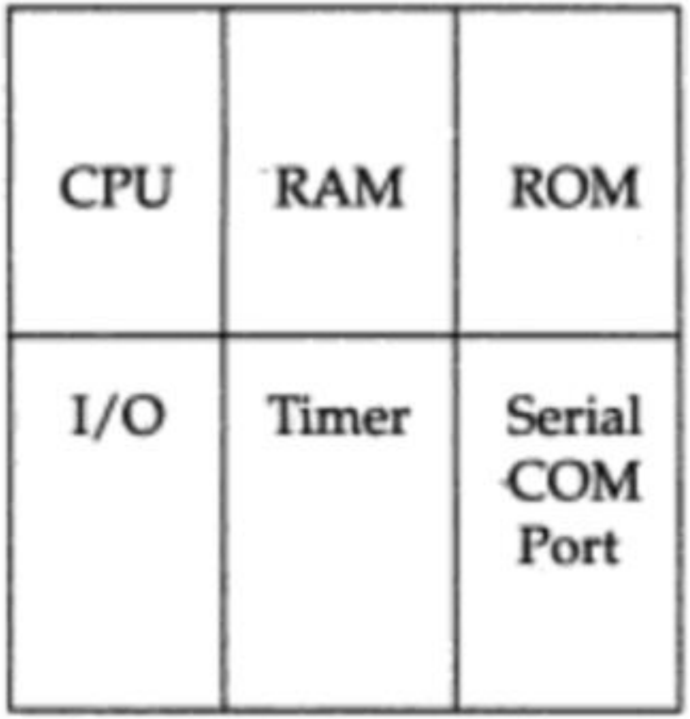
Microprocessor



General-Purpose Microprocessor System

- Microprocessor acts as the heart of computer system.
- Consist of ALU, CU and Registers.
- memory and I/O output component is connected externally.
- It is a processor in which memory and I/O output component is connected externally.
- The operations are memory-based.
- It is mainly used in personal computers.
- Used for general purpose.
- Example: 8085,8088, 80x86, Intel P1

Microcontroller



Microcontroller

- The microcontroller acts as the heart of the embedded system.
- A single chip computer or a CPU with all the peripherals like RAM, ROM, I/O, Timers. ADCs etc. on the same chip.
- It is also known as **SOC (System on Chip).**
- The operations are register-based.
- Pins are multi functioned.
- Primary use is to control the operations of a machine using a fixed program that is stored in ROM and that does not change over the lifetime of the system.
- Used for specific application.
- Example: 8051, PIC 16X.

8051 Microcontroller

- Intel introduced 8051, referred as MCS- 51, in 1981.
- 8051 is housed in a **40 Pin** DIP.
- Constructed using N-channel Metal oxide Silicon (NMOS) and Complementary Metal Oxide Silicon (CMOS).
- The 8051 is an **8-bit processor.** The CPU can work on only 8 bits of data at a time
- The 8051 became widely popular after allowing other manufactures to make and market any flavor of the 8051.

Features of 8051 Microcontroller

- 8-bit CPU with register A and B.
- 16- bit program counter (PC) and Data Pointer (DPTR)
- 8 bit Program Status word.
- Internal ROM or EPROM (8751) of 0 (8031) to 4K (8051).
- Internal RAM of 128 bytes.
  - 4 Register bank.
  - 16 bytes bit addressable memory
  - Eighty bytes of general purpose data memory

34 General purpose resisters

A
8 bit

B
8 bit

Features of 8051 Microcontroller

- 8-bit CPU with register A and B.
- 16- bit program counter (PC) and Data Pointer (DPTR)
- 8 bit Program Status word.
- Internal ROM or EPROM (8751) of 0 (8031) to 4K (8051).
- Internal RAM of 128 bytes.
  - 4 Register bank.
  - 16 bytes bit addressable memory
  - Eighty bytes of general purpose data memory

Address resisters

PC 16 bit

DPH 8 bit	DPL 8 bit

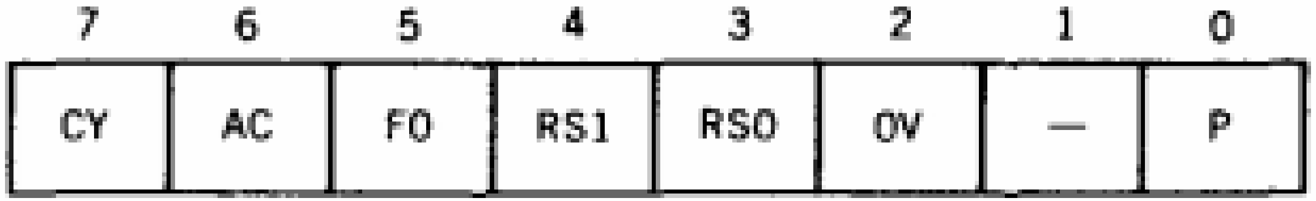
DPTR

16-bit

Features of 8051 Microcontroller

- 8-bit CPU with register A and B.
- 16- bit program counter (PC) and Data Pointer (DPTR)
- 8 bit Program Status word (PSW).
- Internal ROM or EPROM (8751) of 0 (8031) to 4K (8051).
- Internal RAM of 128 bytes.
  - 4 Register bank.
  - 16 bytes bit addressable memory
  - Eighty bytes of general purpose data memory

- Parity flag **P**
- Overflow Flag **OV**
- Register Bank select **RS1, RS0**
- User flag **F0**
- Auxiliary Carry flag **AC**
- Carry Flag **CY**



7	6	5	4	3	2	1	0
CY	AC	F0	RS1	RS0	OV	—	P

Flag Register

8-bit

Features of 8051 Microcontroller

- 8-bit CPU with register A and B.
- 16- bit program counter (PC) and Data Pointer (DPTR)
- 8 bit Program Status word (PSW).
- Internal ROM or EPROM (8751) of 0 (8031) to 4K (8051).
- Internal RAM of 128 bytes.
  - 4 Register bank.
  - 16 bytes bit addressable memory
  - Eighty bytes of general purpose data memory

- Size of the internal ROM is 4 KB
- 4 KB = 2^2 x 2^{10}
- 12 bits are used to generate the address.

0000 **0000 0000 0000**

Address provided by the PC – 16 bit

Address Range

0FFF

.

.

0000

Features of 8051 Microcontroller

- 8-bit CPU with register A and B.
- 16- bit program counter (PC) and Data Pointer (DPTR)
- 8 bit Program Status word (PSW).
- Internal ROM or EPROM (8751) of 0 (8031) to 4K (8051).
- Internal RAM of 128 bytes.
  - 4 Register bank.
  - 16 bytes bit addressable memory
  - Eighty bytes of general purpose data memory

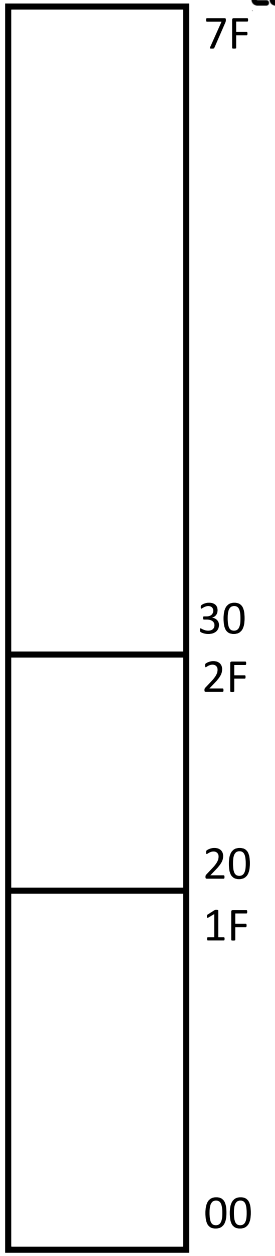
7F (0111 1111)

Address Range

.

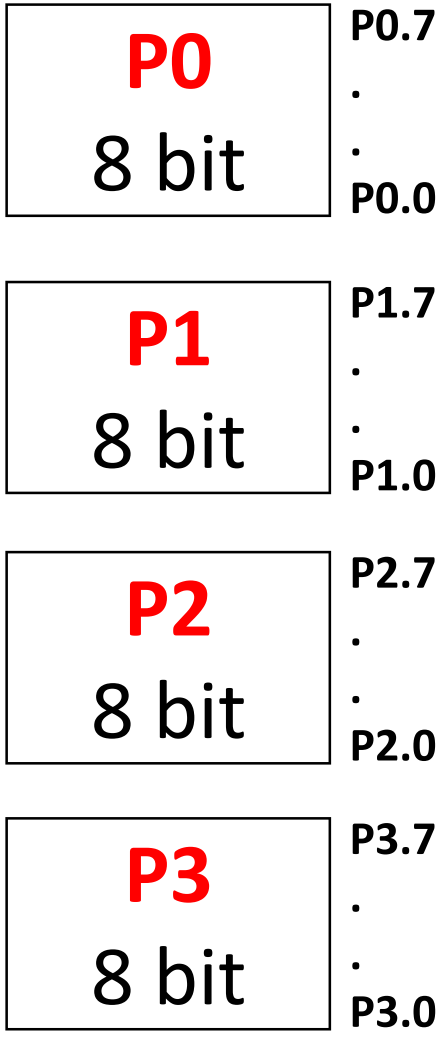
.

00 (0000 0000)



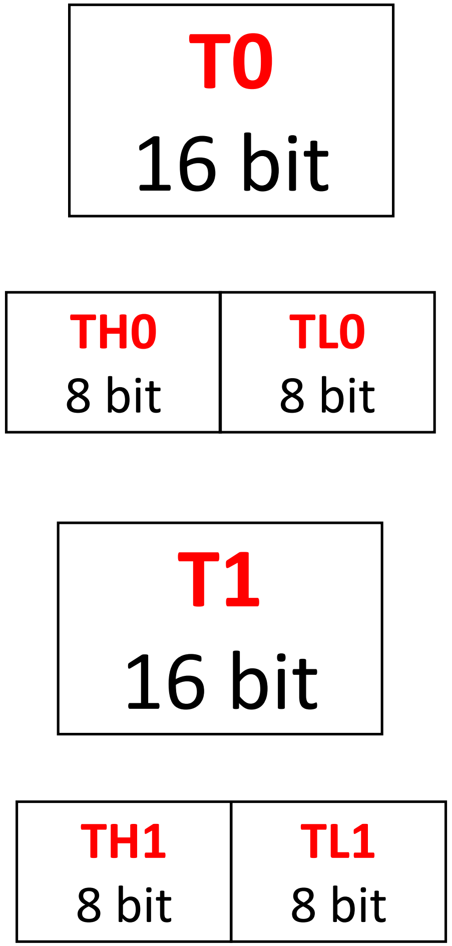
Features of 8051 Microcontroller

- 32 I/O pins arranged as four 8 bits ports i.e. P0-P3.
- Two 16-bit Timer/counter i.e. T0 and T1.
- On chip full duplex serial data receiver/ transmitter: SBUF
- 2 External and 3 internal interrupt sources.
- On Chip Clock Oscillator
- Control registers: TCON, TMOD, SCON, PCON, IP and IE



Features of 8051 Microcontroller

- 32 I/O pins arranged as four 8 bits ports i.e. P0-P3.
- Two 16-bit Timer/counter i.e. T0 and T1.
- On chip full duplex serial data receiver/ transmitter: SBUF
- 2 External and 3 internal interrupt sources.
- On Chip Clock Oscillator
- Control registers: TCON, TMOD, SCON, PCON, IP and IE



T0
16 bit

TH0
8 bit

TL0
8 bit

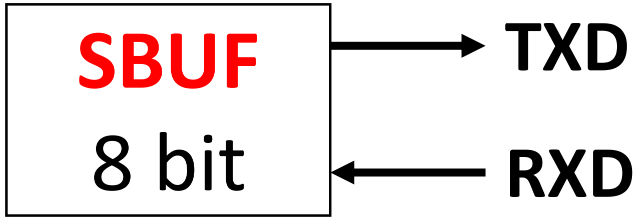
T1
16 bit

TH1
8 bit

TL1
8 bit

Features of 8051 Microcontroller

- 32 I/O pins arranged as four 8 bits ports i.e. P0-P3.
- Two 16-bit Timer/counter i.e. T0 and T1.
- On chip full duplex serial data receiver/ transmitter: SBUF



- 2 External and 3 internal interrupt sources.
- On Chip Clock Oscillator
- Control registers: TCON, TMOD, SCON, PCON, IP and IE

Features of 8051 Microcontroller

- 32 I/O pins arranged as four 8 bits ports i.e. P0-P3.
- Two 16-bit Timer/counter i.e. T0 and T1.
- On chip full duplex serial data receiver/ transmitter: SBUF
- 2 External and 3 Internal interrupt sources.
- On Chip Clock Oscillator
- Control registers: TCON, TMOD, SCON, PCON, IP and IE

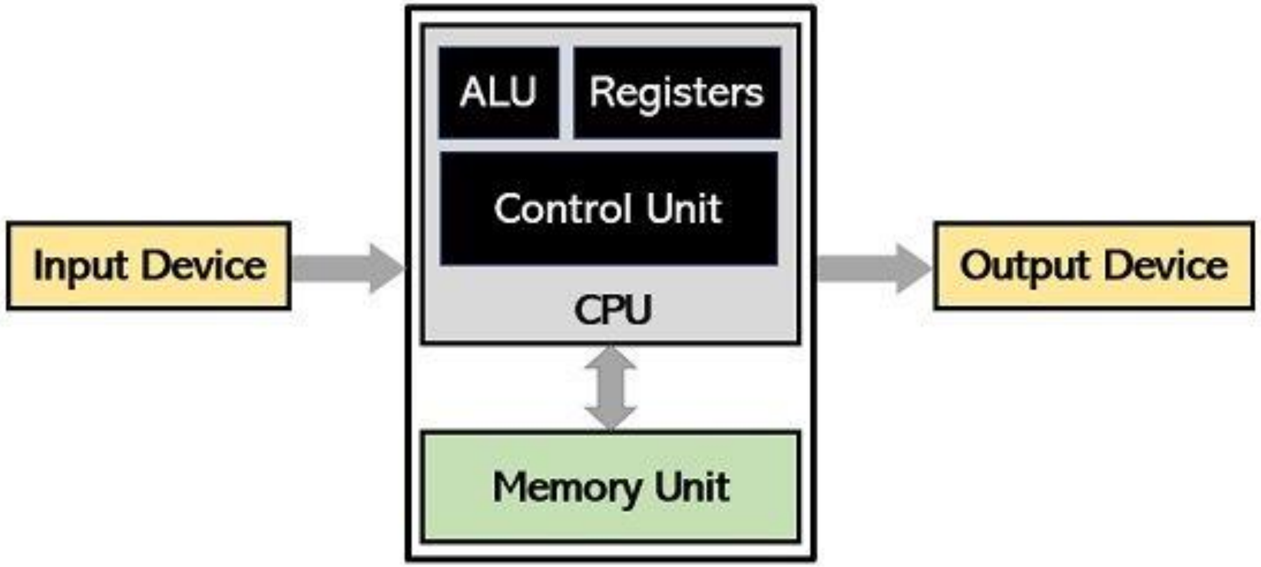
External interrupt sources.

- **INTERUPT 0 (INT 0)**
- **INTERUPT 1 (INT 1)**

Internal interrupt sources.

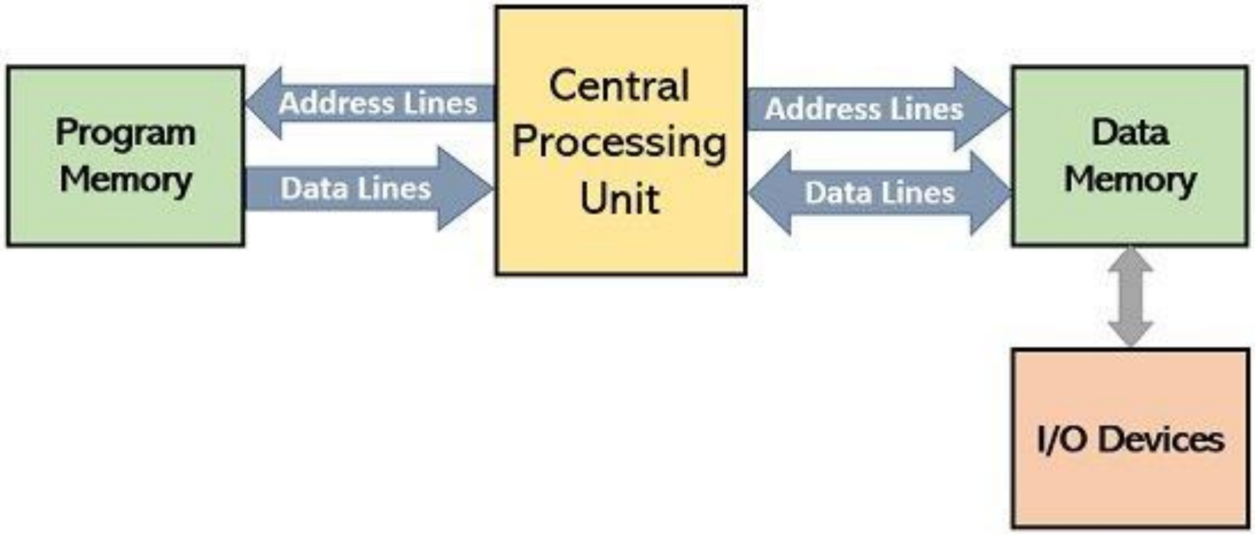
- **TIMER/COUNTER 0**
- **TIMER/COUNTER 1**
- **SERIAL COMMUNICATION**

Von Neumann Architecture



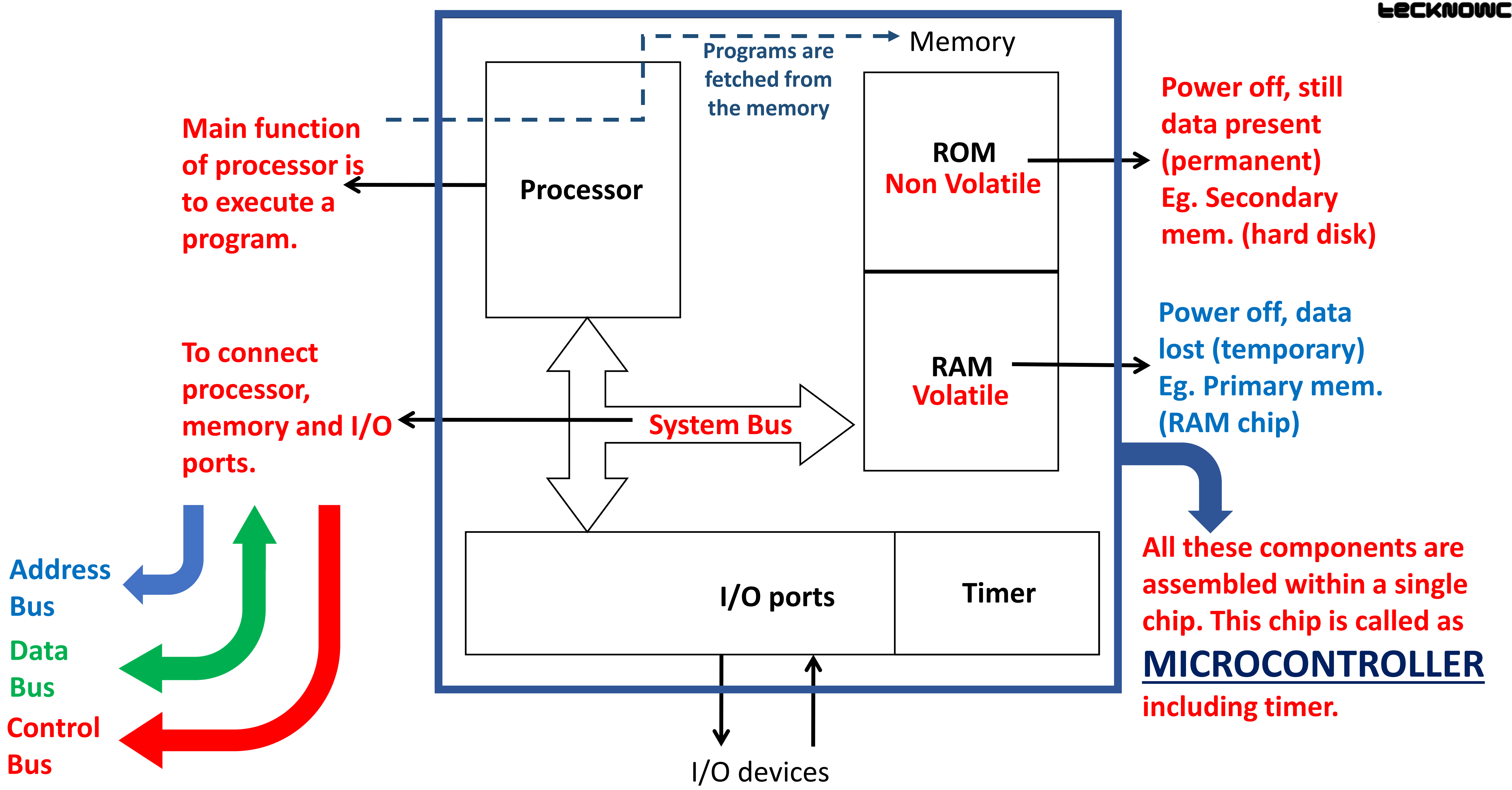
- Data and instructions reside within a single memory unit.
- The control unit retrieves instruction and data in the same way from one memory unit.
- This requires less hardware since only a common memory needs to be reached.
- Von-Neumann Architecture requires less space.
- Speed of execution is slower since it cannot fetch the data and instructions at the same time.
- Space is not wasted because the space of the data memory can be utilized by the instructions memory and vice-versa.
- Controlling becomes simpler since either data or instructions are to be fetched at a time.

Harvard Architecture



- The CPU is connected with both the data memory (RAM) and program memory (ROM), separately.
- Instructions and data can be accessed the same way.
- It requires more hardware since it will be requiring separate data and address bus for each memory.
- This requires more space.
- Speed of execution is faster because the processor fetches data and instructions simultaneously.
- Controlling becomes complex since data and instructions are to be fetched simultaneously.
- It results in wastage of space since if the space is left in the data memory then the instructions memory cannot use the space of the data memory and vice-versa.

Processor based system



Main function of processor is to execute a program.

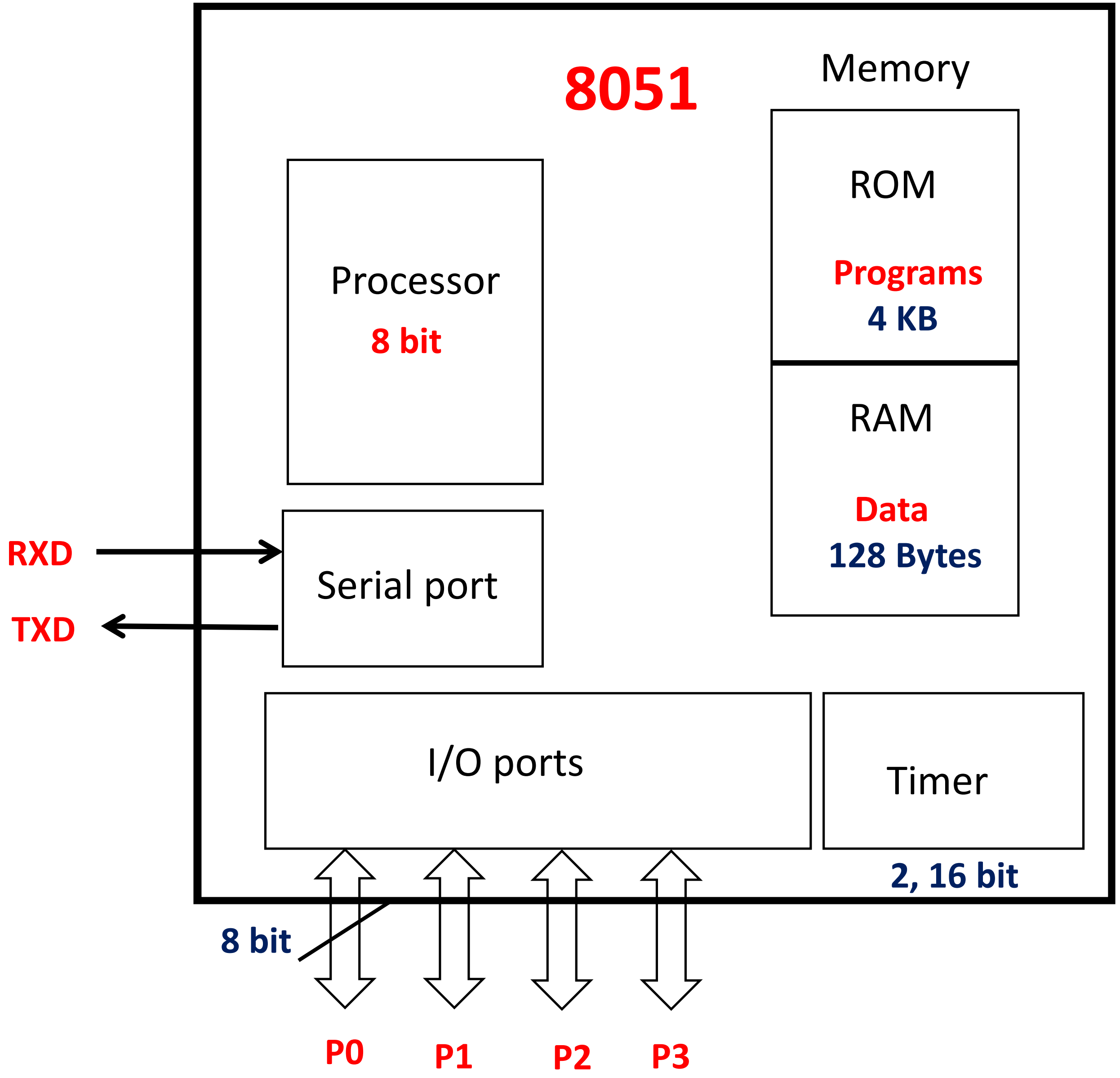
To connect processor, memory and I/O ports.

Power off, still data present (permanent) Eg. Secondary mem. (hard disk)

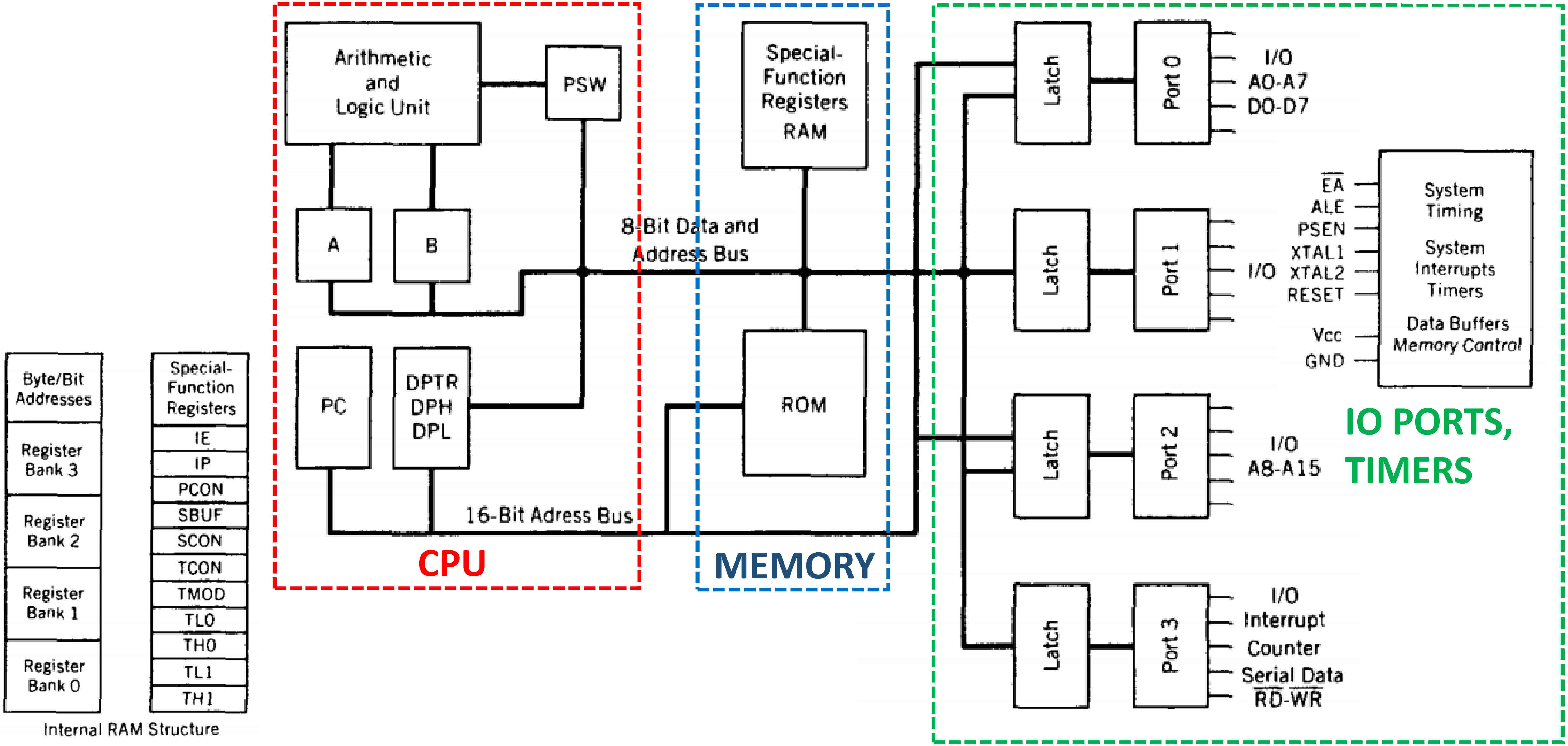
Power off, data lost (temporary) Eg. Primary mem. (RAM chip)

All these components are assembled within a single chip. This chip is called as **MICROCONTROLLER** including timer.

8051 Microcontroller

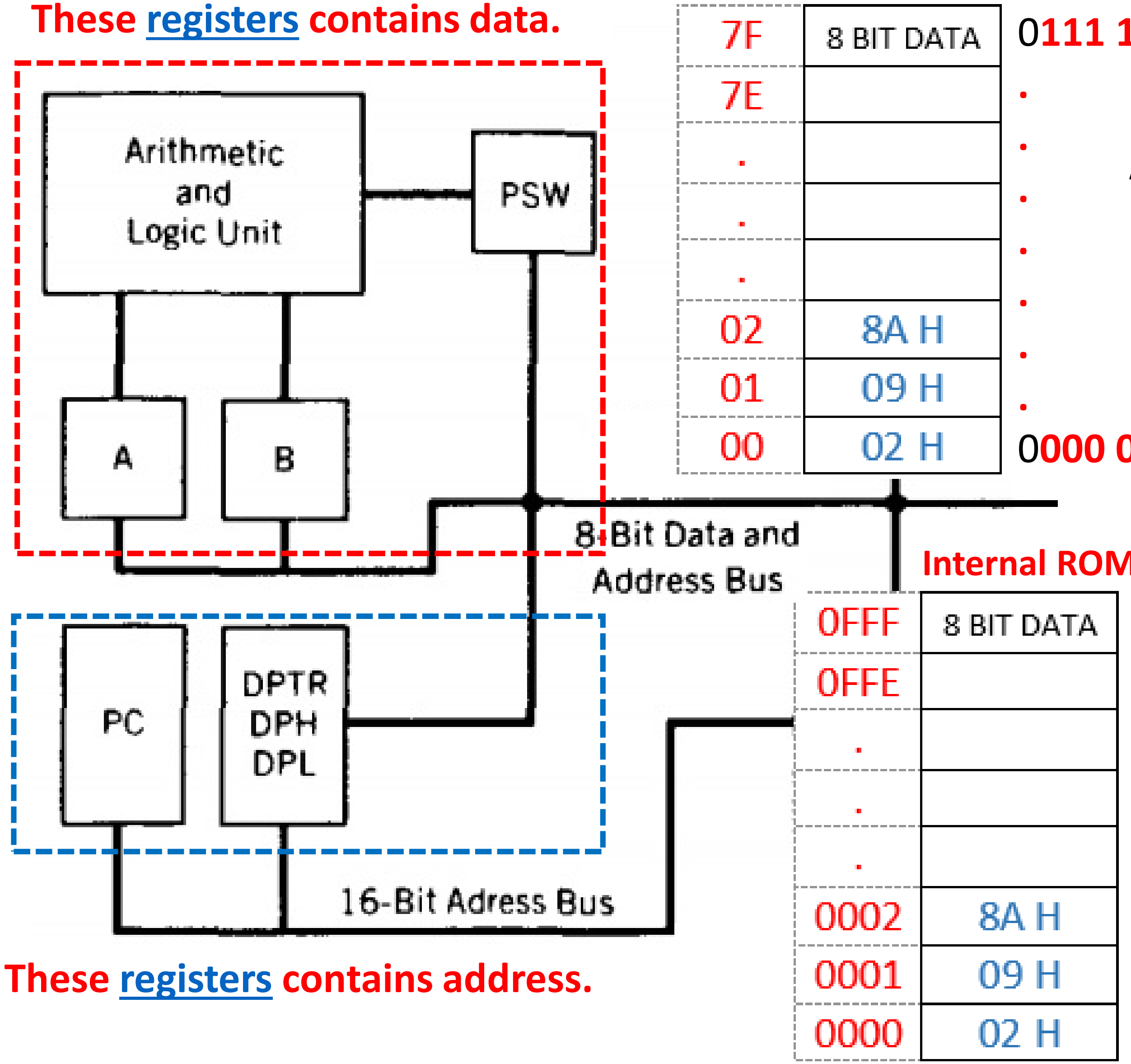


8051 Microcontroller



8051 Microcontroller

These registers contains data.



These registers contains address.

Internal RAM 128 Bytes

7F	8 BIT DATA
7E	
.	
.	
.	
02	8A H
01	09 H
00	02 H

0111 1111
.
.
.
.
.
.
.
0000 0000

ADDRESS BUS - 8 BITS
DATA BUS - 8 BITS

Internal ROM 4 KB

0FFF	8 BIT DATA
0FFE	
.	
.	
.	
0002	8A H
0001	09 H
0000	02 H

0000 1111 1111 1111
.
.
.
.
.
.
.
0000 0000 0000 0000

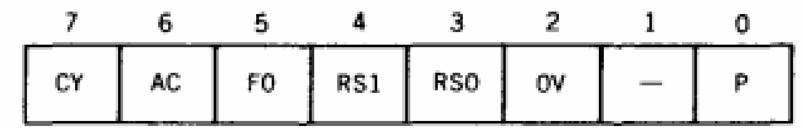
ADDRESS BUS - 16 BITS
DATA BUS - 8 BITS

Data Registers

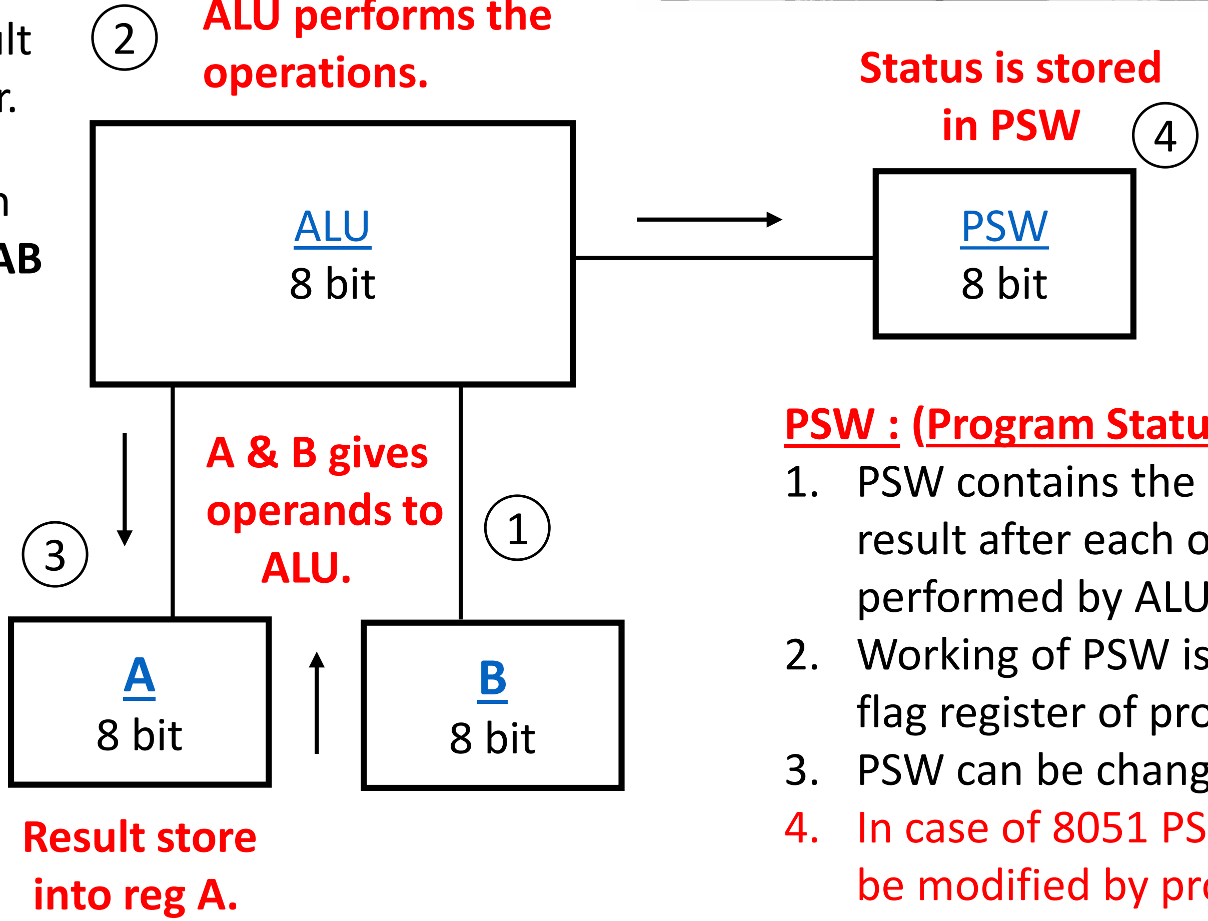
Reg A: (Accumulator): It is used to hold the first operand as well as result. It is accumulating the result therefore it is called as accumulator.

Reg B: it is used with multiplication and division instructions i.e. **MUL AB**

ALU: A and B gives operand to the ALU, then ALU will perform all arithmetic, logical operations on given operand and result will store into accumulator and status will store into PSW.



7	6	5	4	3	2	1	0
CY	AC	F0	RS1	RS0	OV	—	P

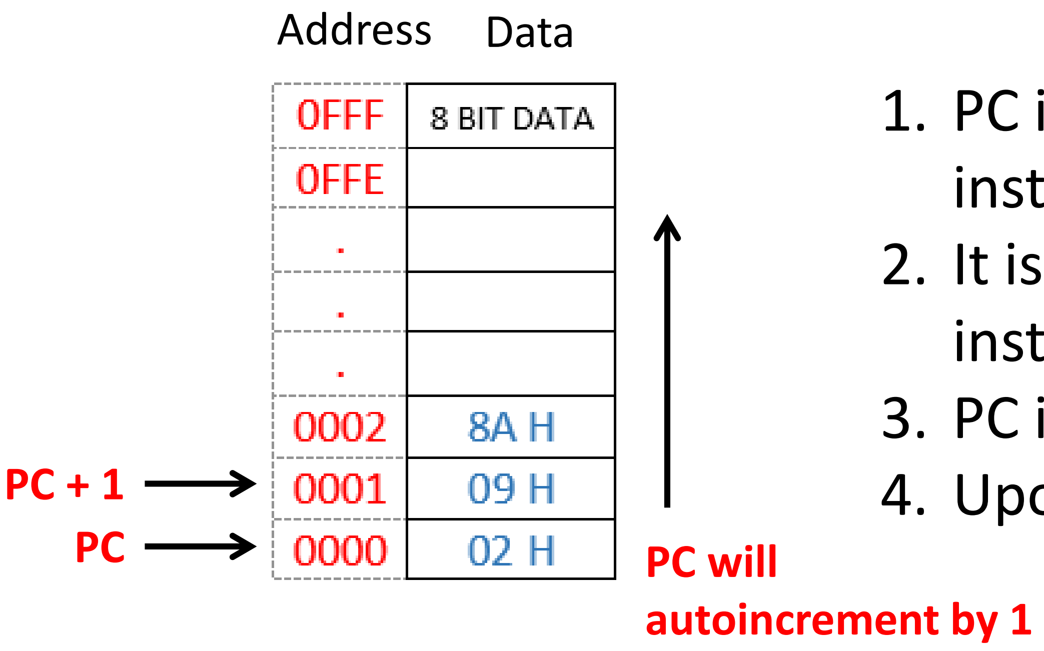


PSW : (Program Status WORD) :

1. PSW contains the status of the result after each operation performed by ALU.
2. Working of PSW is similar with flag register of processor.
3. PSW can be changed by ALU.
4. In case of 8051 PSW can also be modified by programmer (i.e. **MOV PSW, 10h**)

Address Registers

PC (Program Counter):



Address	Data
0FFF	8 BIT DATA
0FFE	
.	
.	
.	
0002	8A H
0001	09 H
0000	02 H

PC will autoincrement by 1

1. PC is used to hold the address of next instruction.
2. It is always used to hold the address of an instruction.
3. PC is **16 bit** register.
4. Upon reset, it points to **0000** H.

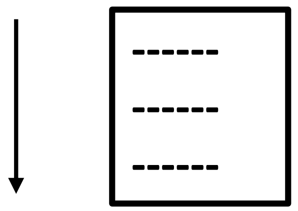
Address Registers

DPTR (Data Pointer 16 bit):

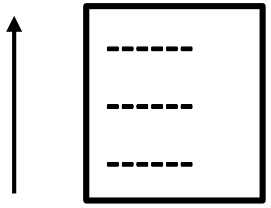
1. Data pointer is used to gives the address of data.
2. Data can be accessed by following manner.

- **It is not autoincrement.**
- **It is controlled by programmer.**

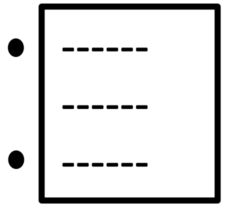
a. In sequence



b. In reverse



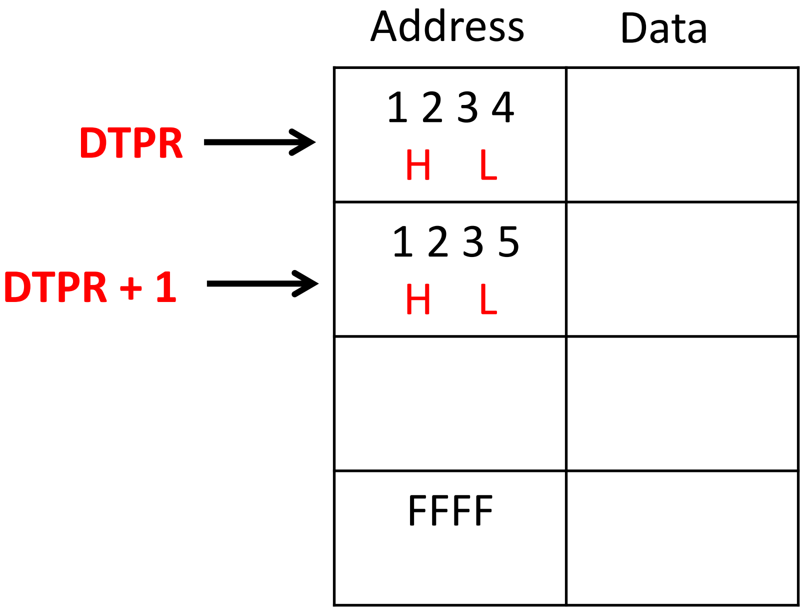
c. Random



- DTPR gives the 16 bit address of 8 bit data because 8051 has 16 bit address.
- To modify DPTR we have to perform 16 bit operation because DPTR is 16 bit.
- But 8051 works on 8 bit data it means 8051 performs 8 bit operation in one cycle.
- So to modify DPTR it requires more than one cycle.
- To avoid above problem two registers are used i.e. DPH and DPL.

Address Registers

DPH (data pointer higher byte) & DPL (data pointer lower byte) :



	Address	Data
DTPR →	1 2 3 4 H L	
DTPR + 1 →	1 2 3 5 H L	
	FFFF	

- If DPTR is 1234 and we want to increment it by 1.
- It means actually we want to increment lower byte.
- So instead of incrementing whole number, only lower byte is needed to perform operation.
- Therefore choice is given to programmer to increment lower byte or higher byte individually.

So programmer can use full DTPR for 16 bit operation or DPH / DPL for 8 bit operation.

Program Status Word

MSB							LSB
7	6	5	4	3	2	1	0
CY	AC	F0	RS1	RS0	OV	-	P
PSW.7	PSW.6	PSW.5	PSW.4	PSW.3	PSW.2	PSW.1	PSW.0

- 6 flags used by the 8051.
- 4 maths flags (CY,AC,OVR,P).
- 1 user flag (F0).

- **SFR (D0 H)**, 8-bit register.
- PSW contains the status of the result after each operation performed by ALU.
- PSW can be changed by ALU.
- In case of 8051 PSW can also modified by programmer (i.e. mov psw, 10 h)
- Working of PSW is similar with flag register of processor.

Program Status Word

MSB							LSB
D7	D6	D5	D4	D3	D2	D1	D0
CY	AC	F0	RS1	RS0	OV	X	P
PSW.7	PSW.6	PSW.5	PSW.4	PSW.3	PSW.2	PSW.1	PSW.0

CY (CARRY FLAG)

- This flag is set whenever there is a carry out from the D7 bit.
- This flag bit is affected after an 8 bit addition or subtraction.
- It can be set to 1 or 0 directly by an instruction such as **SET C** and **CLR C**.

AC (AUXILIARY CARRY FLAG)

- If there is a carry from D3 to D4 during an ADD or SUB operation, this bit is set; otherwise it is cleared.
- This flag is used by instructions that perform BCD (Binary Coded Decimal) arithmetic

Program Status Word

MSB							LSB
D7	D6	D5	D4	D3	D2	D1	D0
CY	AC	F0	RS1	RS0	OV	X	P
PSW.7	PSW.6	PSW.5	PSW.4	PSW.3	PSW.2	PSW.1	PSW.0

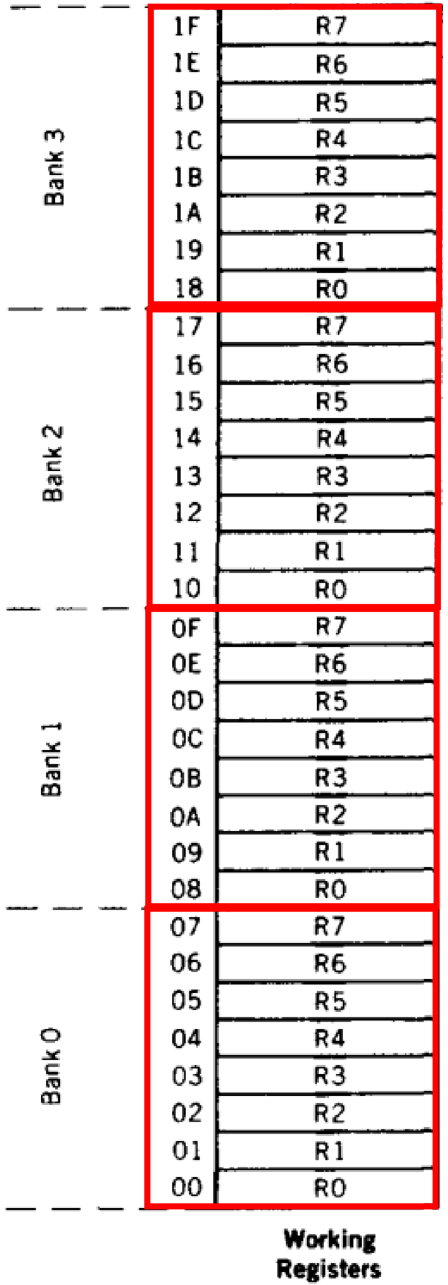
F0

- This flag is an user flag like GF0 and GF1 in PCON.
- It can be set to 1 or 0 directly by an instruction such as **SET PSW.5** and **CLR PSW.5** or **SET D5** and **CLR D5.**

RS1 and RS0 (REGISTER BANK SELECT BITS)

- These bits are used to select working register bank.

RS1	RS0	BANK	ADDRESS
0	0	Bank 0	00 H – 07 H
0	1	Bank 1	08 H – 0F H
1	0	Bank 2	10 H – 17 H
1	1	Bank 3	18 H – 1F H



Program Status Word

MSB							LSB
D7	D6	D5	D4	D3	D2	D1	D0
CY	AC	F0	RS1	RS0	OV	X	P
PSW.7	PSW.6	PSW.5	PSW.4	PSW.3	PSW.2	PSW.1	PSW.0

OV (OVERFLOW FLAG)

- This flag is set whenever the result of a signed number operation is too large, causing the high order bit to overflow into sign bit.
- Carry flag is used to detect errors in unsigned arithmetic operation. The overflow flag is only used to detect errors in signed arithmetic operations.

P (PARITY FLAG)

- The parity flag reflects the numbers of 1’s in the A register only.
- If the A register contains an odd number of 1’s, then P=1, otherwise P=0.

Examples

ADD A, #3E H

MOV A, #79 H ; A = 79 HEX

ADD A, #3E H ; A = 79 + 3E

```
   79 H          0 1111 000
+                 0111 1001
   3E H        +
                  0011 1110
 ------        --------------
   D7 H           1011 0111
```

0	1						0
CY	AC	F0	RS1	RS0	OV	-	P

Examples

ADD A, #0B H

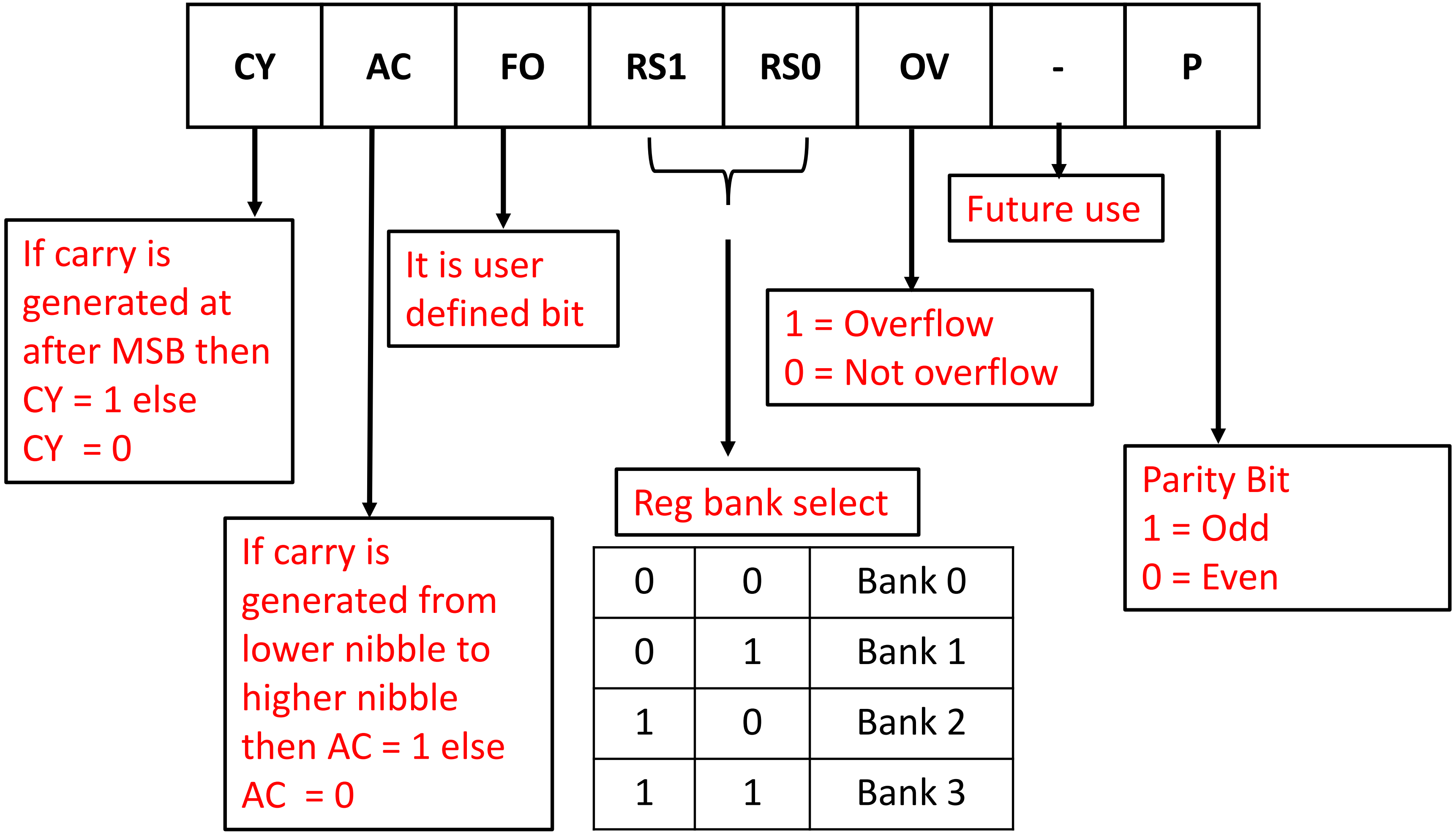
MOV A, #0F5 H ; A = F5 HEX

ADD A, #0B H ; A = F5 + 0B

```
   F5 H          1 1111 111
+                  1111 0101
   0B H       +
                   0000 1011
 --------     -----------------
  100 H            0000 0000
```

1	1						0
CY	AC	F0	RS1	RS0	OV	-	P

Program Status Word



CY	AC	FO	RS1	RS0	OV	-	P

If carry is generated at after MSB then CY = 1 else CY = 0

If carry is generated from lower nibble to higher nibble then AC = 1 else AC = 0

It is user defined bit

Reg bank select

0	0	Bank 0
0	1	Bank 1
1	0	Bank 2
1	1	Bank 3

1 = Overflow
0 = Not overflow

Future use

Parity Bit
1 = Odd
0 = Even

Program Status Word

MSB							LSB
D7	D6	D5	D4	D3	D2	D1	D0
CY	AC	F0	RS1	RS0	OV	X	P
PSW.7	PSW.6	PSW.5	PSW.4	PSW.3	PSW.2	PSW.1	PSW.0

OV (OVERFLOW FLAG)

- This flag is set whenever the result of a signed number operation is too large, causing the high order bit to overflow into sign bit.
- Carry flag is used to detect errors in unsigned arithmetic operation. The overflow flag is only used to detect errors in signed arithmetic operations.

Number System

▪ Unsigned Number	**+ve**	**0 to 255**
▪ Signed Number	**+ve and -ve**	**-128 to +127**

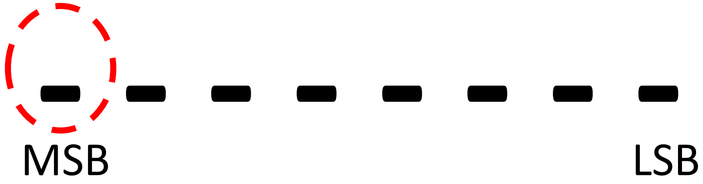
Overflow flag

UNSIGNED NUMBERS

- All positive numbers.
- For Unsigned numbers if magnitude is 8 then, $\mathbf{2^8 = 256}$

SIGNED NUMBERS

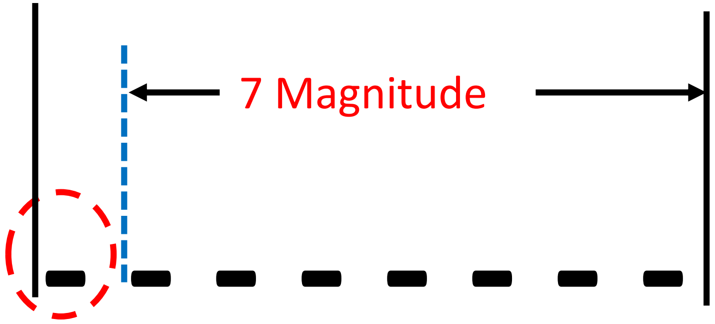
- Positive and Negative numbers.



Range for unsigned numbers

00	0000 0000
01	0000 0001
..	..
..	..
FF	1111 1111

If number is 8 bit



S

0 + ve

1 - ve

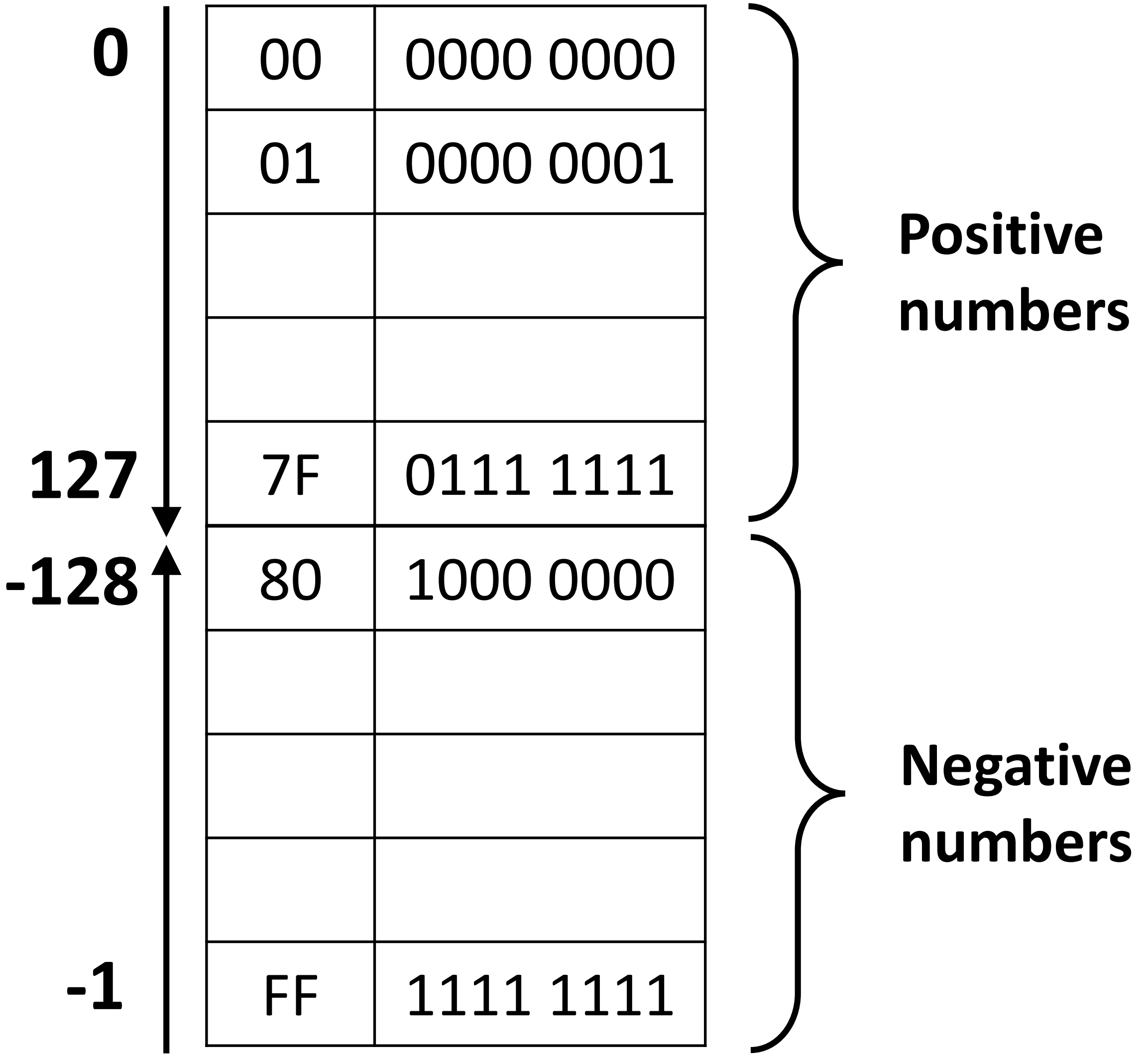
Overflow flag

SIGNED NUMBERS

For signed numbers if magnitude is then 7 ,
$2^7 = 128$
i.e. 128 **- ve** numbers & 128 **+ ve** numbers

-128 to -1

0 to 127



0	00	0000 0000
	01	0000 0001
127	7F	0111 1111
-128	80	1000 0000
-1	FF	1111 1111

Positive numbers

Negative numbers

Overflow flag

USE OF 2's COMPLIMENT

1 0 0 0 0 0 1 1

7 bit magnitude then **-03 H**

If we consider 1 for –ve number

~~**1 0 0 0 0 0 0 0**~~

~~7 bit magnitude then~~ **- 00 H**

If we consider 1 for –ve number

Which is not possible

- **2's Complement** is a universal method to store –ve number.

0 0 0 0 0 0 0 0

2's complement of 0 is :

0 0 0 0 0 0 0 0

Shortcut for 2's complement :

copy number as it is from right side till gets first 1 after 1 complement all numbers.

~~0~~1 ~~0~~1 ~~0~~1 1 0 0 0 0

Overflow Flag

MSB							LSB
D7	D6	D5	D4	D3	D2	D1	D0
CY	AC	F0	RS1	RS0	OV	X	P
PSW.7	PSW.6	PSW.5	PSW.4	PSW.3	PSW.2	PSW.1	PSW.0

- This flag is set whenever there is a carry out of bit 6 or out of bit 7, but not both.
- Processor will come to know the overflow flag using carry.

Formula used by processor is :

$$\boxed{OV = C7 \oplus C6}$$

X-OR table

0	0	0
0	1	1
1	0	1
1	1	0

Overflow Flag

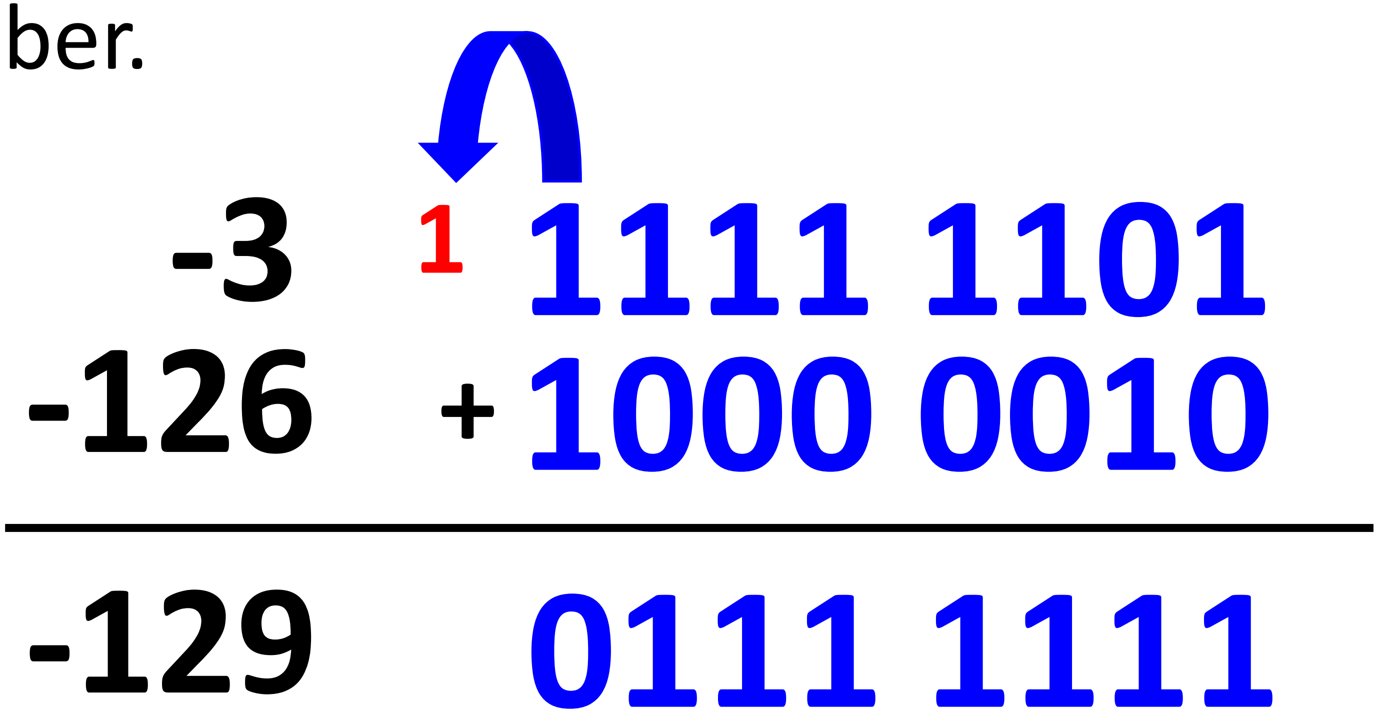
- In an 8-bit signed number operations, the number ranges from **-128** to **+127.**
- Any number outside this range can not be represented by a 8-bit signed number.

Lets try -2 add -127

-3 + (-126) = -129

OV = 1

The result is out of range of the 8 bit signed number.



```
  -3     1  1111 1101
-126     +  1000 0010
---------------------
-129        0111 1111
```

Overflow Flag

Lets try -2 add -127

-4 + (-124) = -128

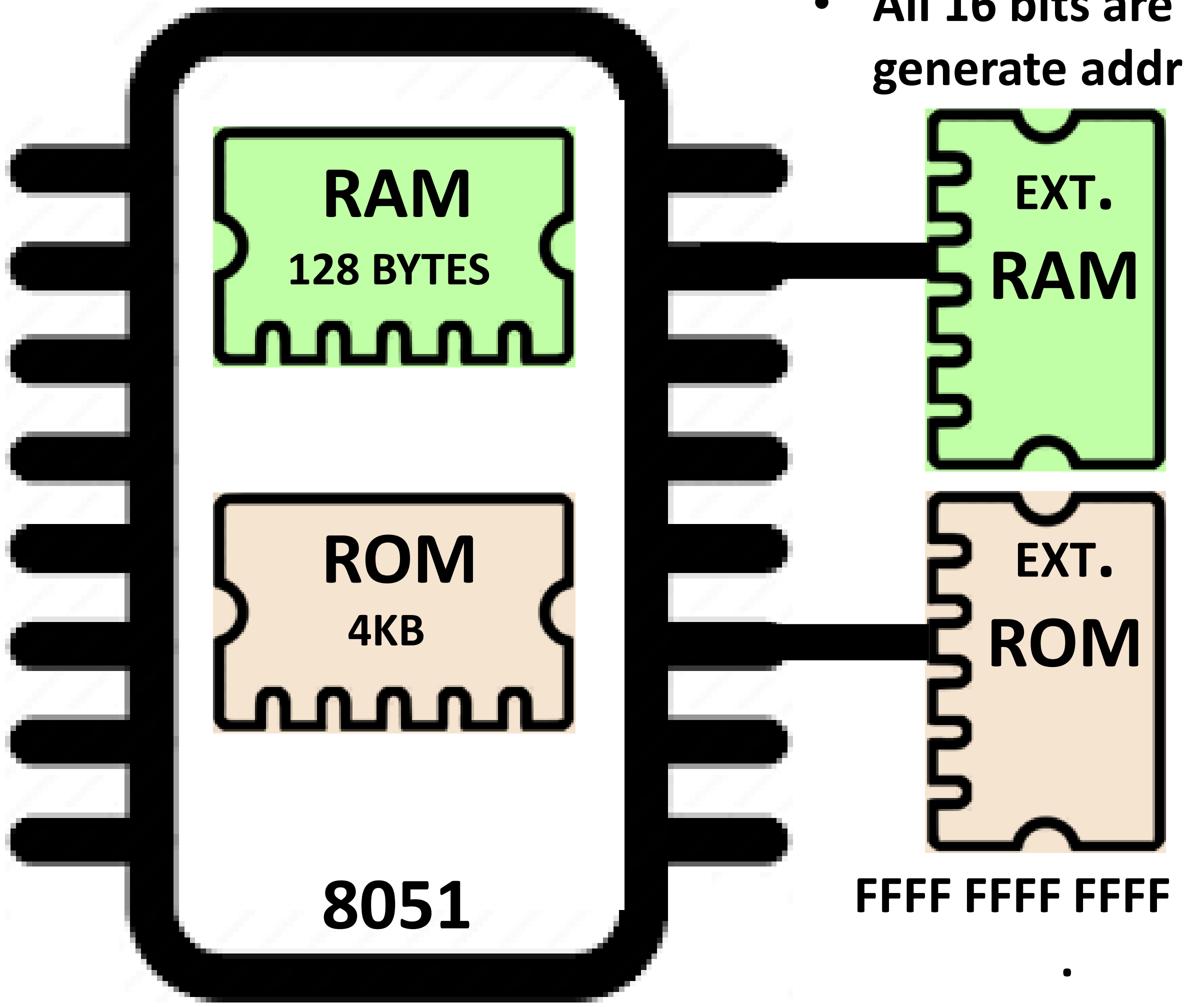
OV = 0

The result is within the range of the 8 bit signed number.

```
              1 1 1 1 1   1
  -4            1111 1100
-124         +  1000 0100
-------------------------
-128            1000 0000
```

Memory

- PC and DPTR is 16 bit.
- 2^{16} = 64KB
- **All 16 bits are used to generate address.**



Up to 64 KB

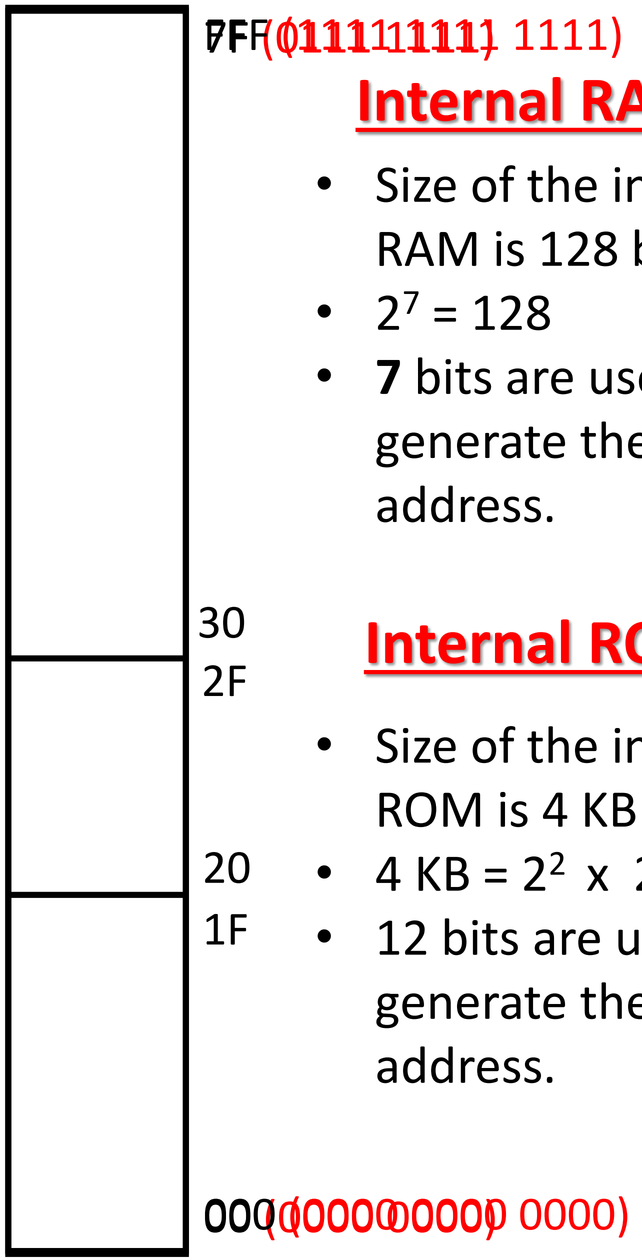
Up to 64 KB

FFFF FFFF FFFF FFFF

.

.

0000 0000 0000 0000



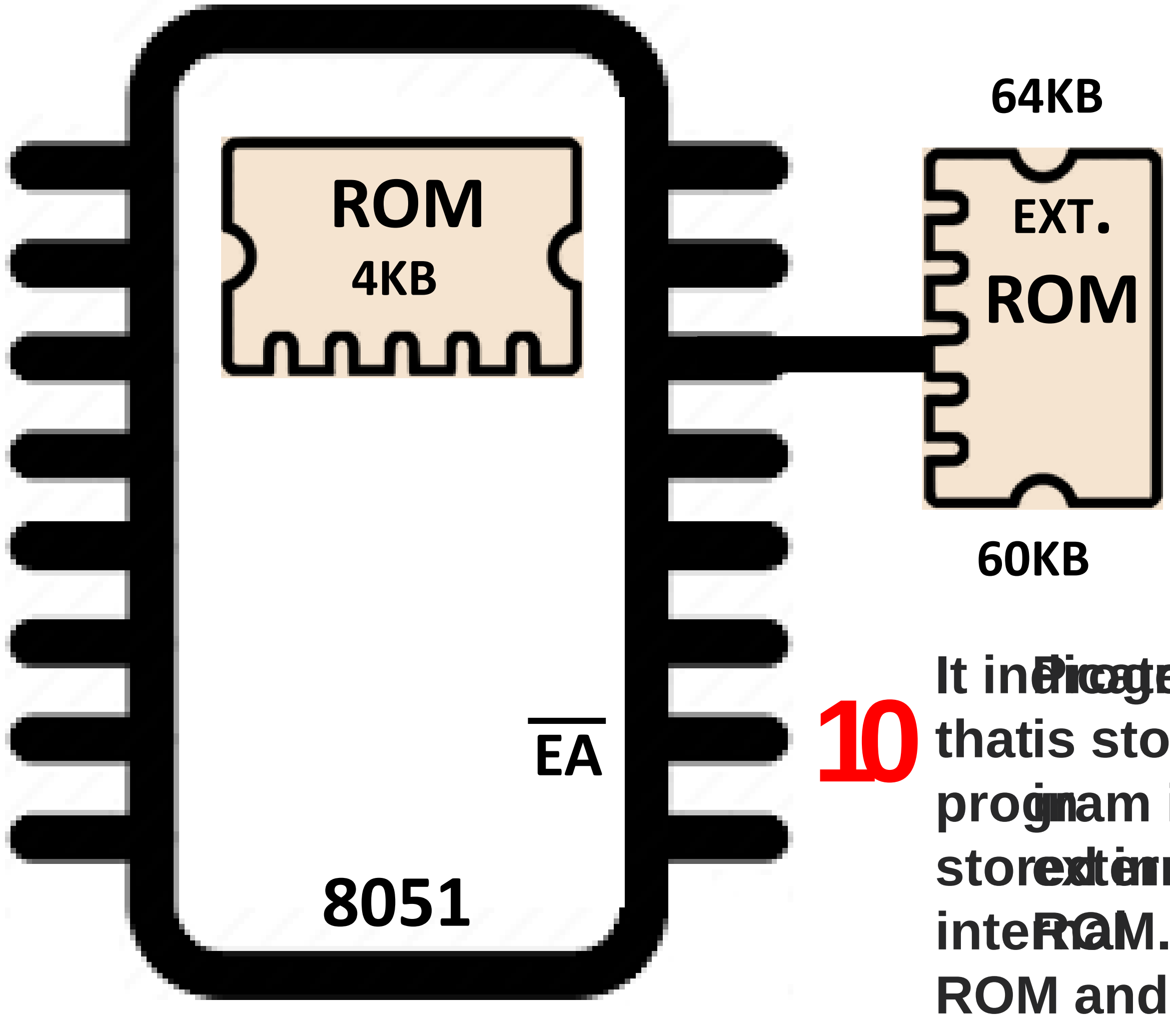
Internal RAM

- Size of the internal RAM is 128 bytes
- 2^7 = 128
- **7** bits are used to generate the address.

Internal ROM

- Size of the internal ROM is 4 KB
- 4 KB = 2^2 x 2^{10}
- 12 bits are used to generate the address.

Read Only Memory



1 It indicates that program is stored internal ROM and external 60 KB ROM.

0 Program is stored in external ROM.

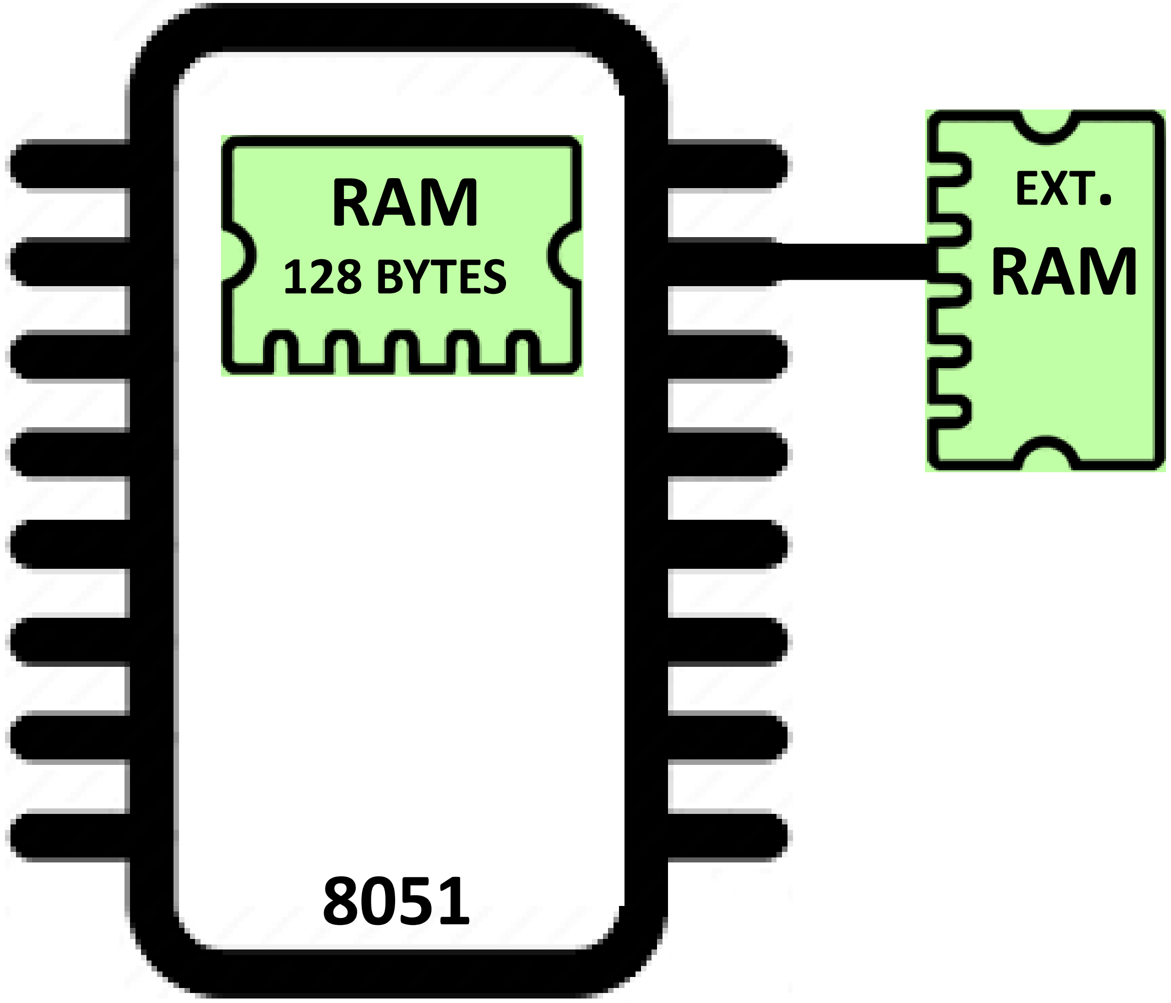
8051 ROM is used to store programs.
ROM is read only memory it is not used to write, so cannot change the programs.
If you want to change or update the system or programs external ROM is required.

EA stands for external access enable.
This pin is designed to tell the information about program storage.

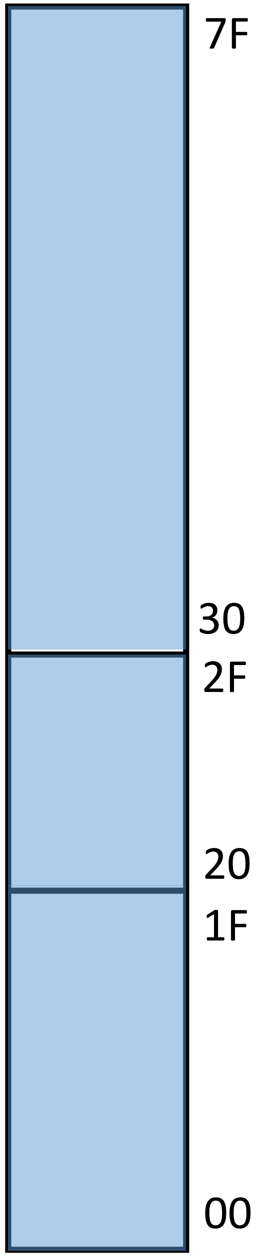
INT	0000	
	0FFF	
EXT	1000	
	FFFF	

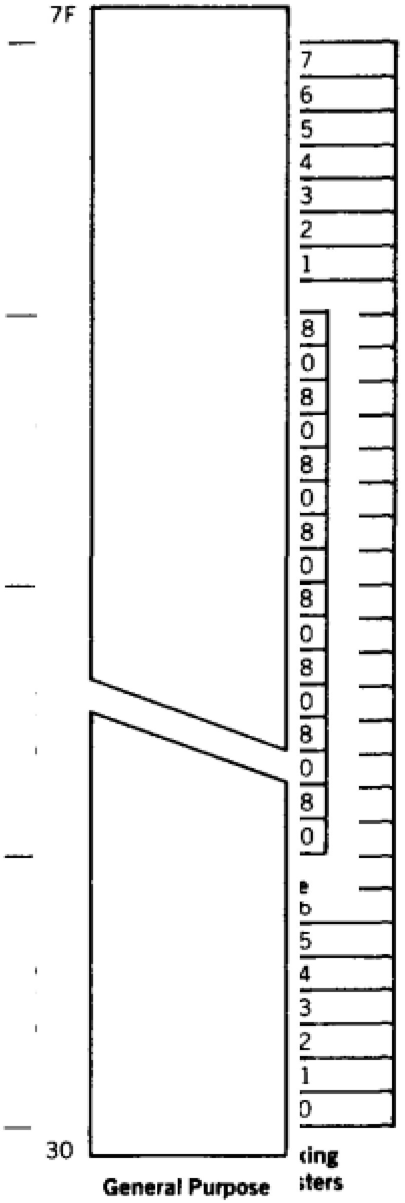
Total 64 KB

Random Access Memory



Up to
64 KB

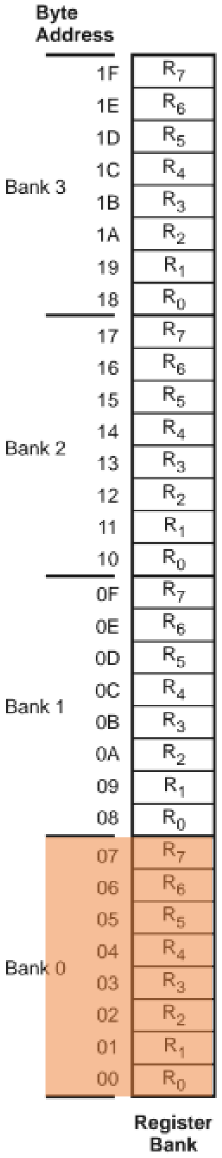




Random Access Memory

Register Banks:

- 32 bytes
- There are 4 banks used to store the data i.e. bank 0 to bank 3.
- Bank 0 default bank
- Each bank consist 8 register, from R0-R7 in each Bank.
- RS1 and RS0 used to select bank



(Default bank)

	Byte Address	Register Bank
Bank 3	1F	R_7
	1E	R_6
	1D	R_5
	1C	R_4
	1B	R_3
	1A	R_2
	19	R_1
	18	R_0
Bank 2	17	R_7
	16	R_6
	15	R_5
	14	R_4
	13	R_3
	12	R_2
	11	R_1
	10	R_0
Bank 1	0F	R_7
	0E	R_6
	0D	R_5
	0C	R_4
	0B	R_3
	0A	R_2
	09	R_1
	08	R_0
Bank 0	07	R_7
	06	R_6
	05	R_5
	04	R_4
	03	R_3
	02	R_2
	01	R_1
	00	R_0

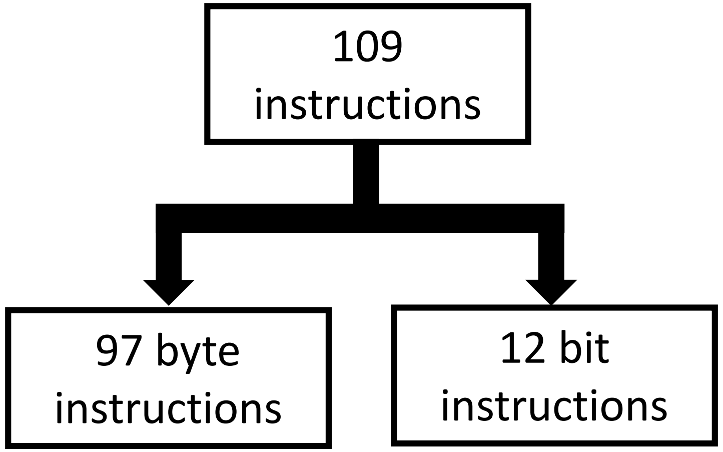
- size of individual register is 1 byte. i.e. 8 bit
- Used to perform Byte Operation.
- These registers can be accessed by name or address
- By Name : Need to activate register bank. Eg. MOV A, R0
- By Address : No need to activate register bank. Any activated bank does not put any restriction.

  Eg. MOV A, 13H

Random Access Memory

BIT ADDRESSABLE AREA

- **16 bytes**
- **Bit operation.**
- **Each byte has 8 addressable bits.**
- **16 X 8 = 128 bit addressable bits.**



109 instructions

97 byte instructions

12 bit instructions

B_7	B_6	B_5	B_4	B_3	B_2	B_1	B_0	Byte Addresses
7F	7E	7D	7C	7B	7A	79	78	2F
77	76	75	74	73	72	71	70	2E
6F	6E	6D	6C	6B	6A	69	68	2D
67	66	65	64	63	62	61	60	2C
5F	5E	5D	5C	5B	5A	59	58	2B
57	56	55	54	53	52	51	50	2A
4F	4E	4D	4C	4B	4A	49	48	29
47	46	45	44	43	42	41	40	28
3F	3E	3D	3C	3B	3A	39	38	27
37	36	35	34	33	32	31	30	26
2F	2E	2D	2C	2B	2A	29	28	25
27	26	25	24	23	22	21	20	24
1F	1E	1D	1C	1B	1A	19	18	23
17	16	15	14	13	12	11	10	22
0F	0E	0D	0C	0B	0A	09	08	21
07	06	05	04	03	02	01	00	20

Bit Addresses

How to access bit addressable area?

Register bank and bit addressable area both are use same address.

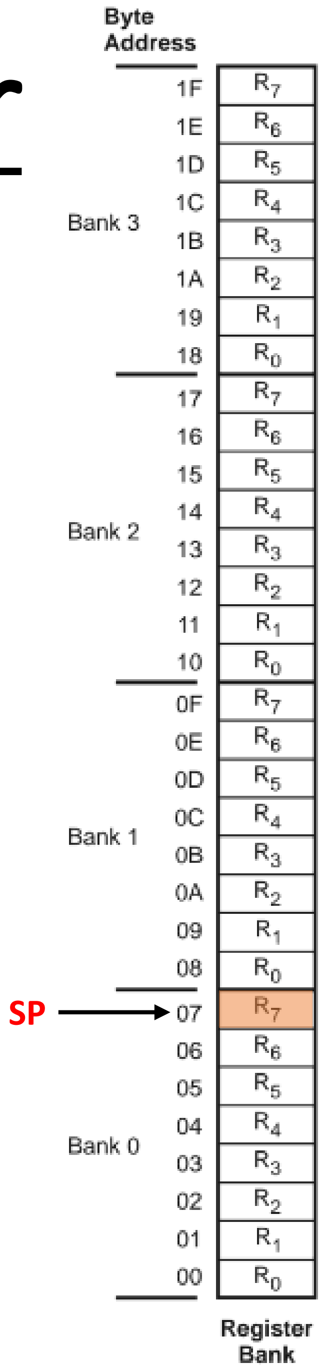
Instructions are used to differentiate between register and bit area.

MOV 00, 25H

SETB 25H
CLR 25H

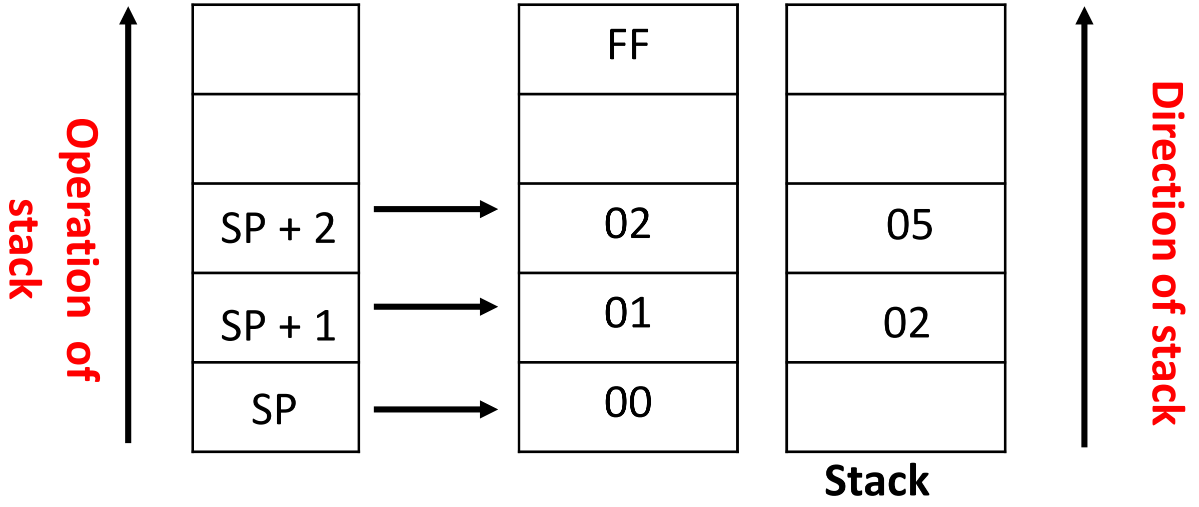
Stack Pointer

- Used to access stack.
- Stack is a section of Internal RAM.
- SP is 8 bit wide.
- Can point to address in the range of 00 H – FF H
- Upon reset SP is set to 07 H.
- Used for PUSH and POP operation.
- SFR address is 81H.



	Byte Address	Register Bank
Bank 3	1F	R_7
	1E	R_6
	1D	R_5
	1C	R_4
	1B	R_3
	1A	R_2
	19	R_1
	18	R_0
Bank 2	17	R_7
	16	R_6
	15	R_5
	14	R_4
	13	R_3
	12	R_2
	11	R_1
	10	R_0
Bank 1	0F	R_7
	0E	R_6
	0D	R_5
	0C	R_4
	0B	R_3
	0A	R_2
	09	R_1
	08	R_0
Bank 0	SP → 07	R_7
	06	R_6
	05	R_5
	04	R_4
	03	R_3
	02	R_2
	01	R_1
	00	R_0

To insert into stack PUSH instruction is used with FILO.



Operation of stack

SP + 2 → 02 | 05

SP + 1 → 01 | 02

SP → 00

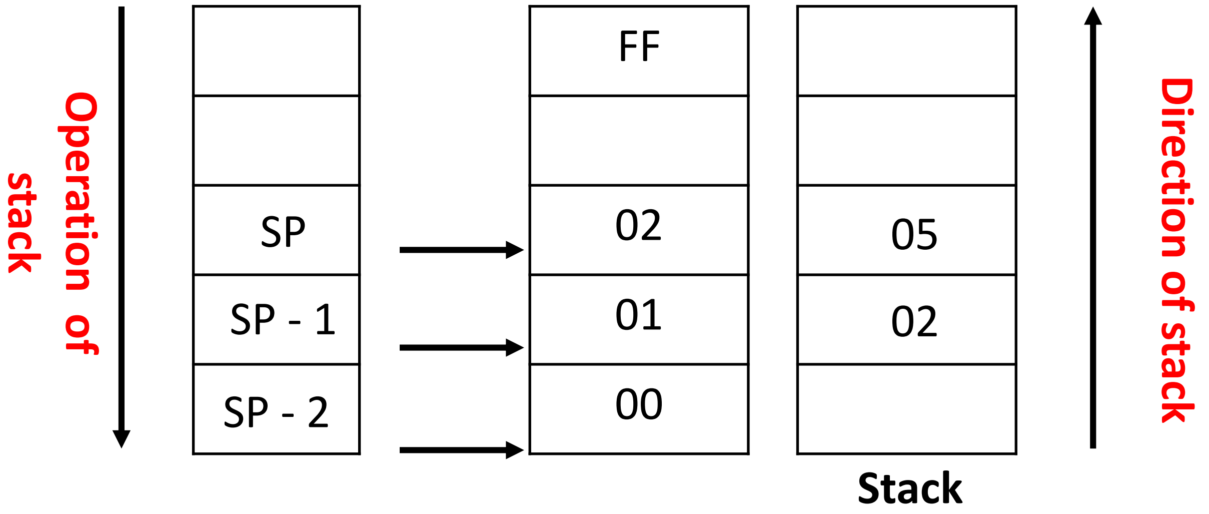
FF

Stack

Direction of stack

Stack pointer is always first incremented by 1 to insert.

To access from stack POP instruction is used with LIFO.



Operation of stack

SP → 02 | 05

SP - 1 → 01 | 02

SP - 2 → 00

FF

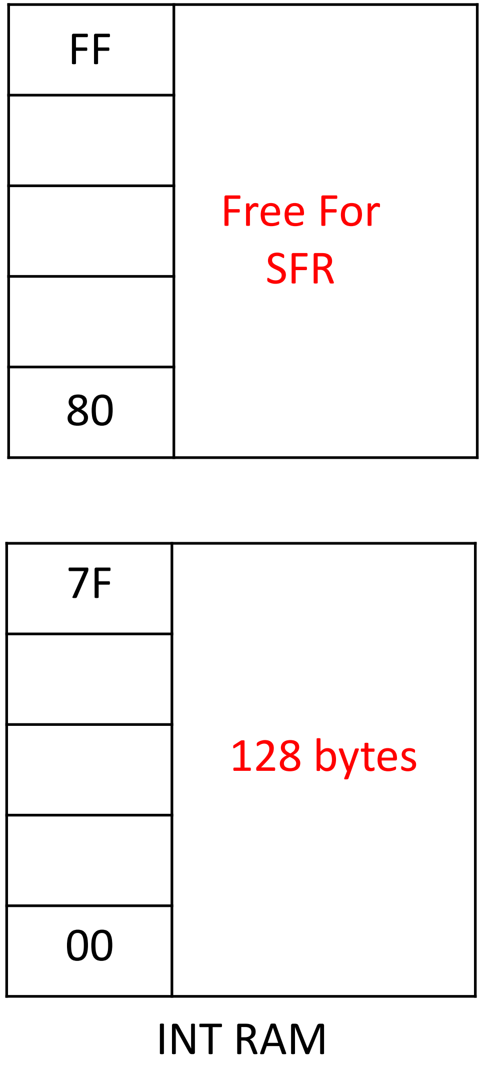
Stack

Direction of stack

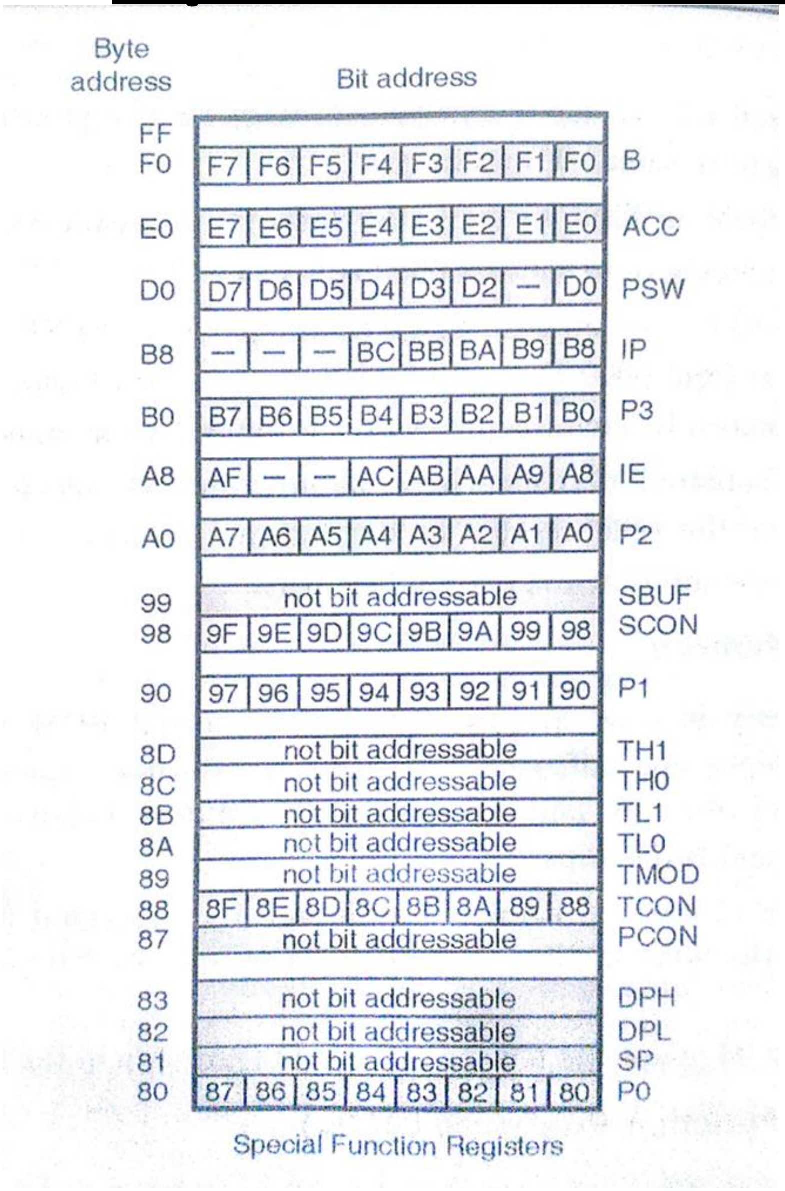
Stack pointer is access data pointed by SP and at the end always decremented by 1.

Special Function Register

- **SFR are placed in the address space immediately above the 128 bytes of RAM, from address 80H to FFH.**
- **The SFR memory consist of important registers like accumulator, B register, Interrupt control registers, PSW, timer/counter, Power control, four I/O ports, serial control.**
- **There are total 21 SFR.**
- **Some of these registers are bit addressable and remaining are byte addressable.**



Special Function Register



Byte address	Bit address								
FF									
F0	F7	F6	F5	F4	F3	F2	F1	F0	B
E0	E7	E6	E5	E4	E3	E2	E1	E0	ACC
D0	D7	D6	D5	D4	D3	D2	—	D0	PSW
B8	—	—	—	BC	BB	BA	B9	B8	IP
B0	B7	B6	B5	B4	B3	B2	B1	B0	P3
A8	AF	—	—	AC	AB	AA	A9	A8	IE
A0	A7	A6	A5	A4	A3	A2	A1	A0	P2
99	not bit addressable								SBUF
98	9F	9E	9D	9C	9B	9A	99	98	SCON
90	97	96	95	94	93	92	91	90	P1
8D	not bit addressable								TH1
8C	not bit addressable								TH0
8B	not bit addressable								TL1
8A	not bit addressable								TL0
89	not bit addressable								TMOD
88	8F	8E	8D	8C	8B	8A	89	88	TCON
87	not bit addressable								PCON
83	not bit addressable								DPH
82	not bit addressable								DPL
81	not bit addressable								SP
80	87	86	85	84	83	82	81	80	P0

Special Function Registers

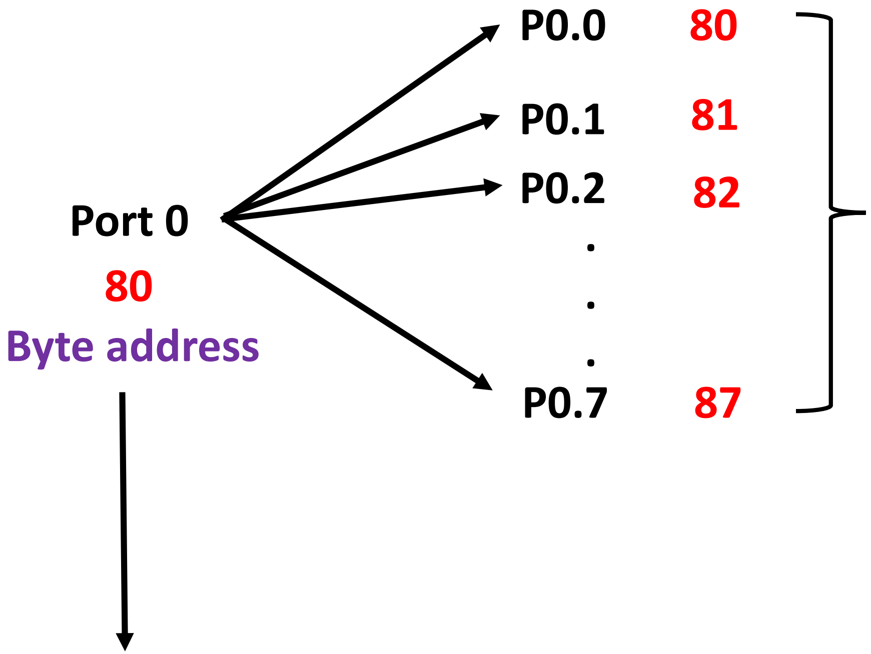
1. **SFR is not part of internal RAM.**
2. **SFR is a combination of different registers which are used for special purpose.**
3. **Those registers are refers the addresses from 80H which are not consecutive.**

SFR are always access by using address.

MOV A, 88 H
MOV A, 87 H
MOV A, 98 H

Special Function Register

For PORT Addresses:



Bit address

SETB 80

SETB 81

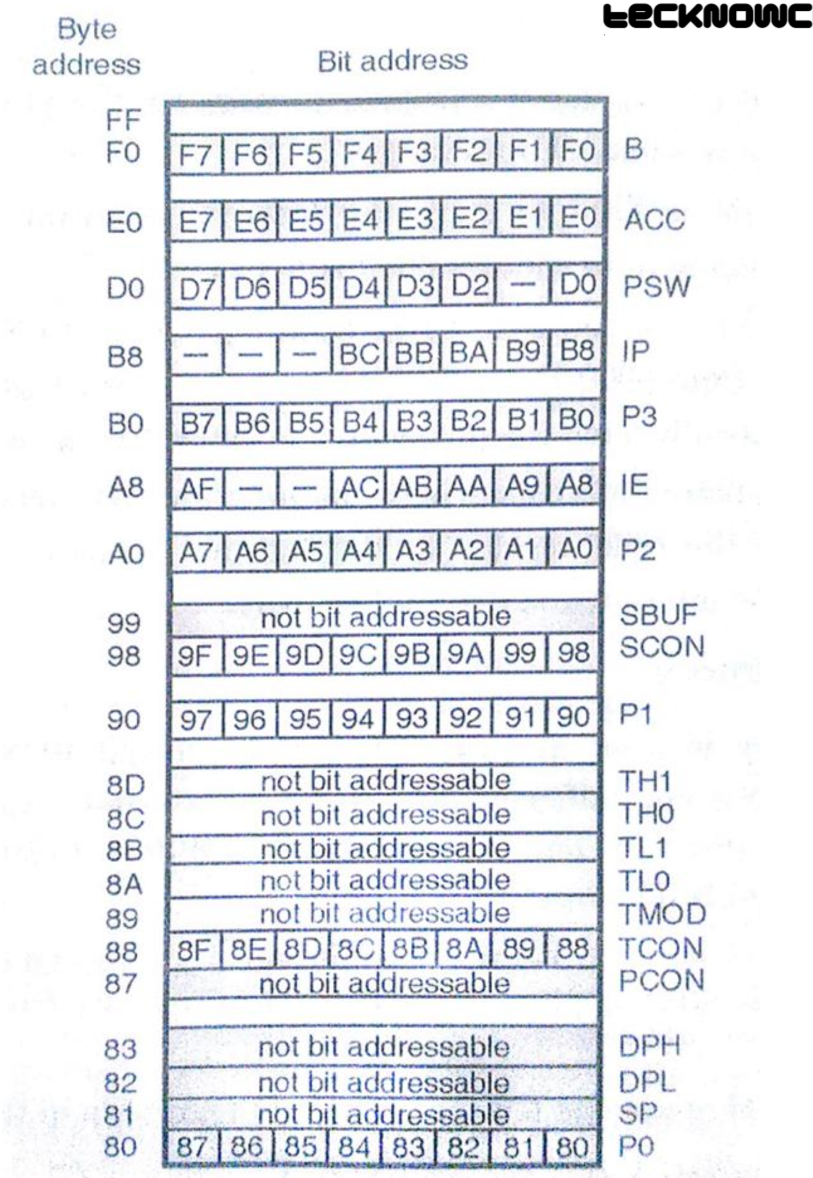
.

.

SETB 87

If we want to access whole port so we have to use 80 as a byte address

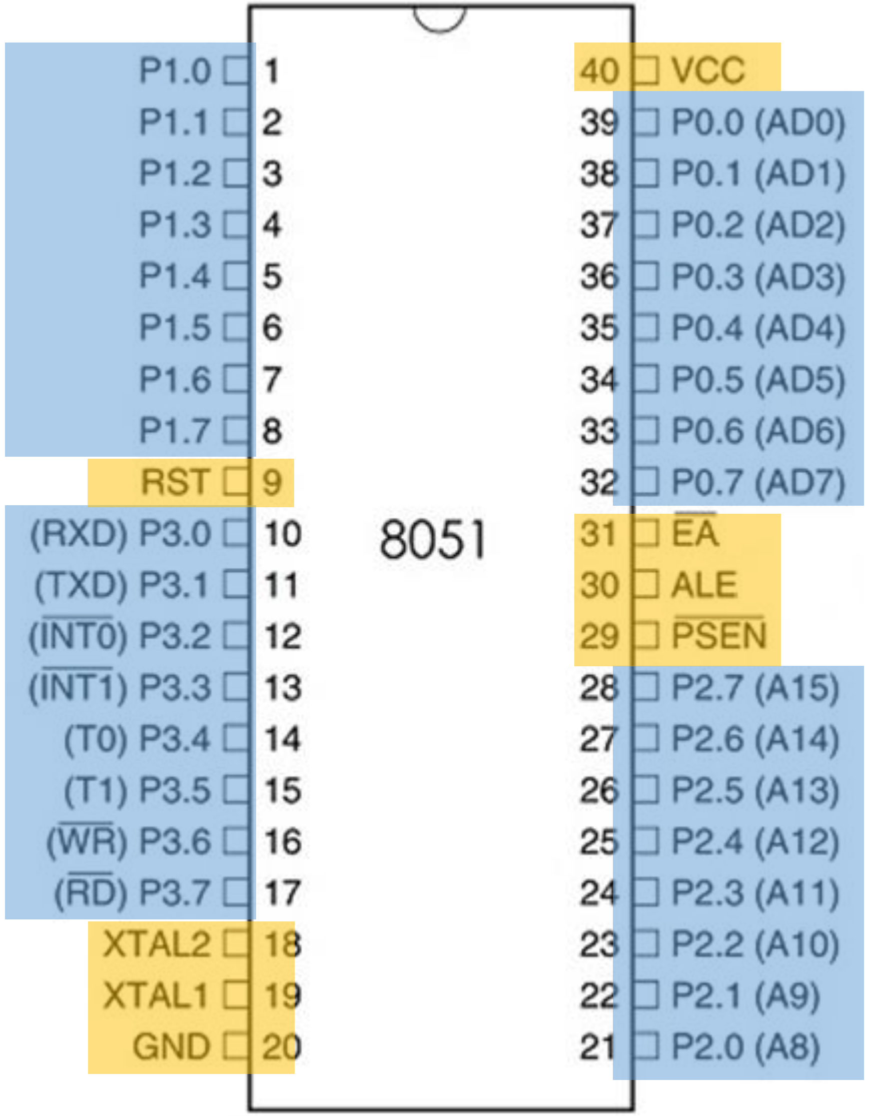
MOV A, 80 H



Byte address	Bit address								
FF									
F0	F7	F6	F5	F4	F3	F2	F1	F0	B
E0	E7	E6	E5	E4	E3	E2	E1	E0	ACC
D0	D7	D6	D5	D4	D3	D2	—	D0	PSW
B8	—	—	—	BC	BB	BA	B9	B8	IP
B0	B7	B6	B5	B4	B3	B2	B1	B0	P3
A8	AF	—	—	AC	AB	AA	A9	A8	IE
A0	A7	A6	A5	A4	A3	A2	A1	A0	P2
99	not bit addressable								SBUF
98	9F	9E	9D	9C	9B	9A	99	98	SCON
90	97	96	95	94	93	92	91	90	P1
8D	not bit addressable								TH1
8C	not bit addressable								TH0
8B	not bit addressable								TL1
8A	not bit addressable								TL0
89	not bit addressable								TMOD
88	8F	8E	8D	8C	8B	8A	89	88	TCON
87	not bit addressable								PCON
83	not bit addressable								DPH
82	not bit addressable								DPL
81	not bit addressable								SP
80	87	86	85	84	83	82	81	80	P0

Special Function Registers

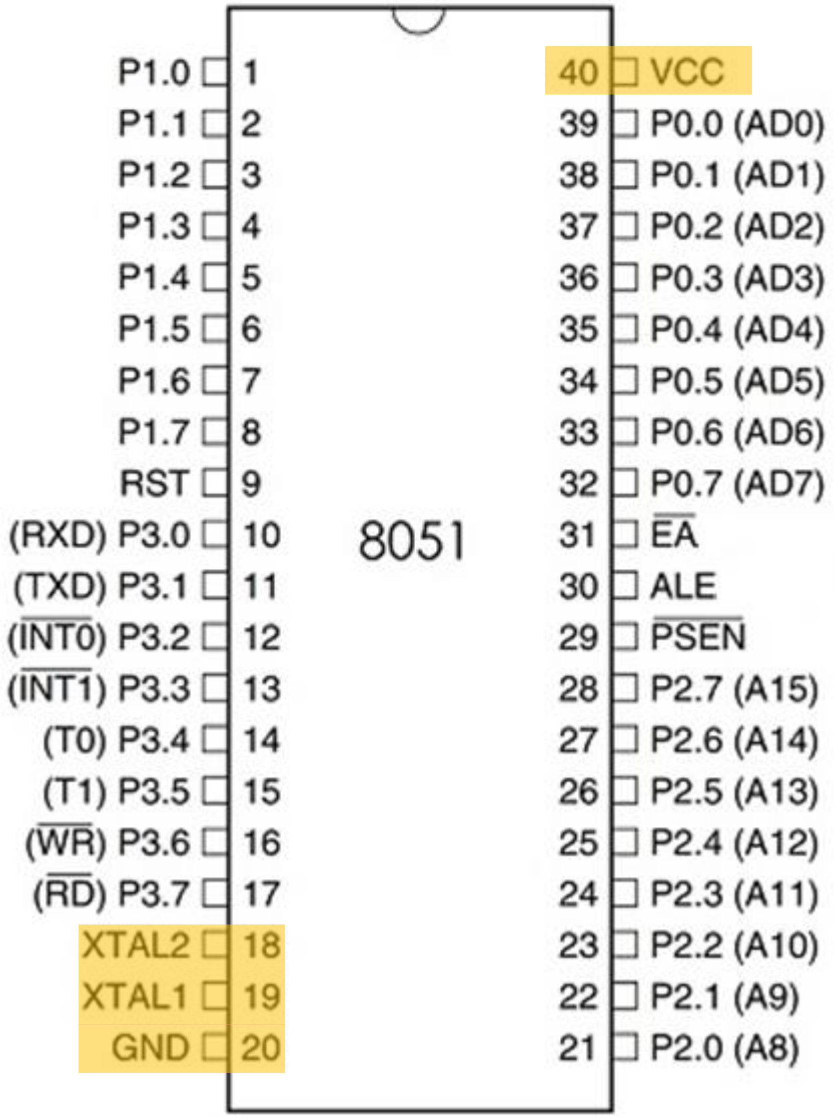
Pin Diagram of 8051



8051 microcontroller is a 40 pin Dual Inline Package (DIP).
These 40 pins serve different functions like read, write, I/O operations, interrupts etc.
It offers functionalities of 72 pins.

32 pins out of these 40 pins are dedicated to I/O ports.
The rest of the pins are dedicated to VCC, GND, XTAL1, XTAL2, RST, ALE, EA and PSEN.

Pin Diagram of 8051



Pin 40 (VCC) –

This pin provides power supply voltage i.e. +5 Volts to the circuit.

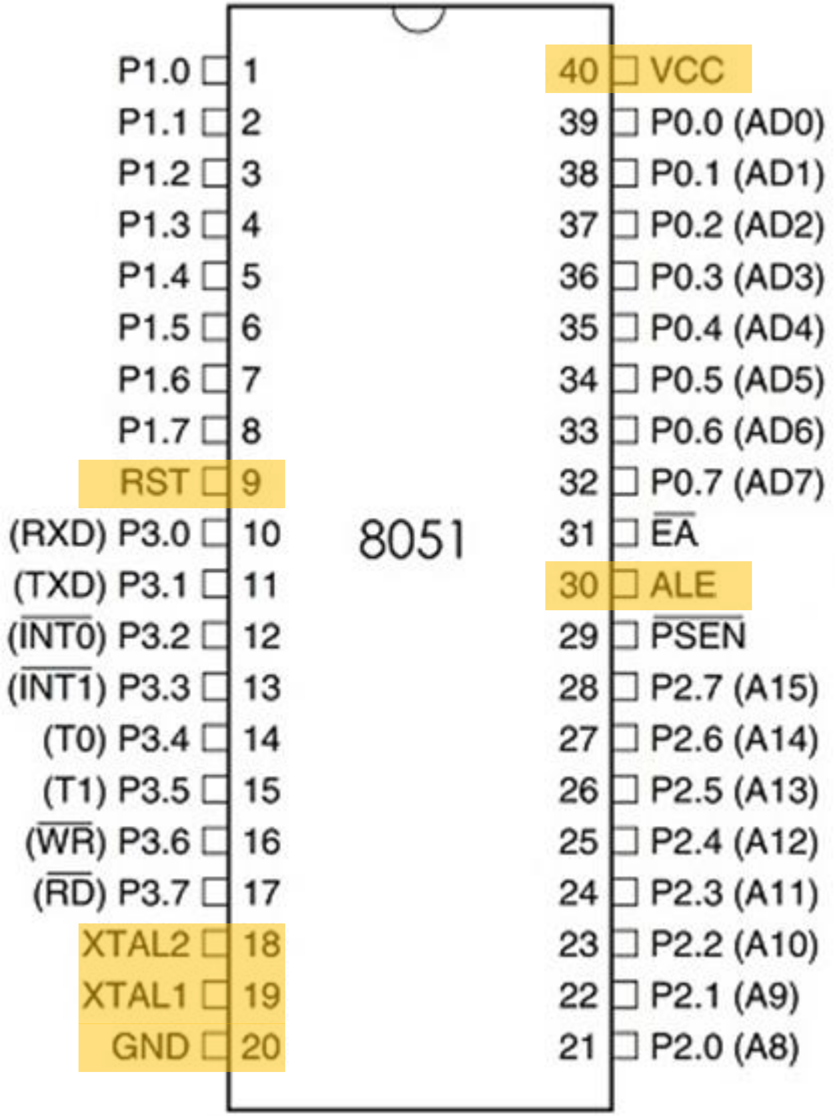
Pin 20 (GND) –

This pin is connected to the ground. It has to be provided with 0 V power supply. Hence it is connected to the negative terminal of the power supply.

Pin 18 and Pin 19 (XTAL2 And XTAL1) –

These pins are connected to an external oscillator which is generally a quartz crystal oscillator. They are used to provide an external clock frequency.

Pin Diagram of 8051



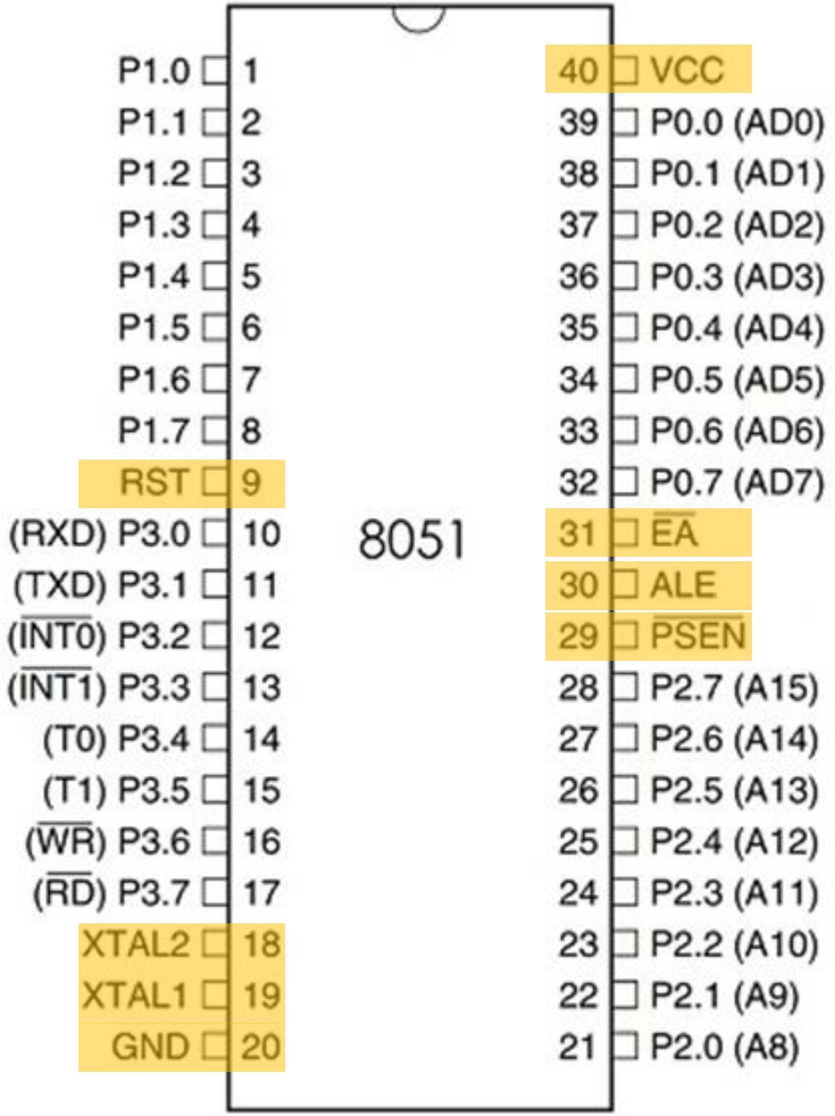
Pin 9 (RST) –

It is an active-high, input pin.
The RST pin is high for a minimum of 2 machine cycles, the microcontroller will reset i.e. it will close and terminate all activities. It is often referred as "power-on-reset" pin because it is used to reset the microcontroller to it's initial values when power is on (high). It only reset RAM not ROM.

Pin 30 (ALE) –

ALE stands for Address Latch Enable.
It is input, active-high pin. It is also used to de-multiplex the multiplexed address and data signals available at port 0. 8051 has 16 bit address bus and 8 bit data bus. When ALE = 0 , AD0-AD7 work as data bus and when ALE = 1 this lines work as a address

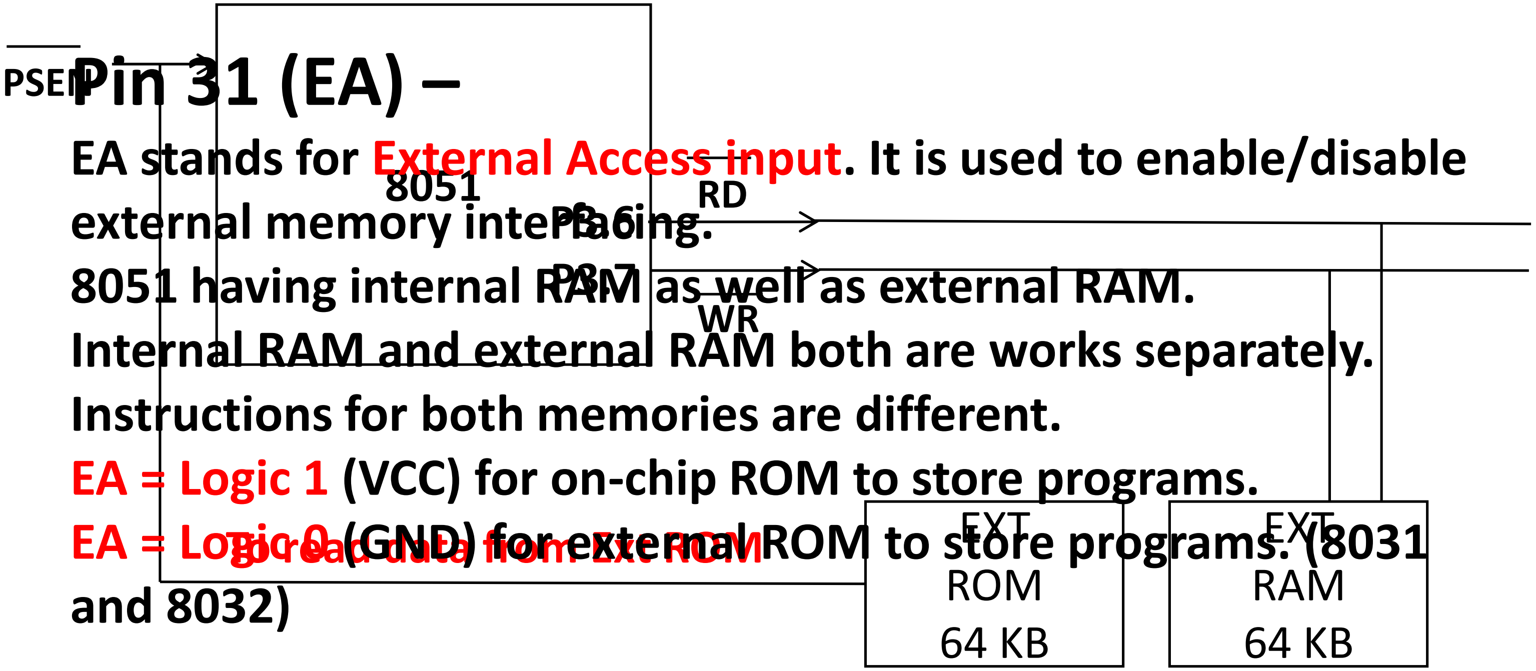
Pin Diagram of 8051



Pin 29 (PSEN) –

PSEN stands for Program Store Enable.

It is output, active-low pin. This is used to read external ROM memory. In 8031 based system where external ROM holds the program code, this pin is connected to the OE pin of the ROM.



Pin 31 (EA) –

EA stands for External Access input. It is used to enable/disable external memory interfacing.

8051 having internal RAM as well as external RAM.

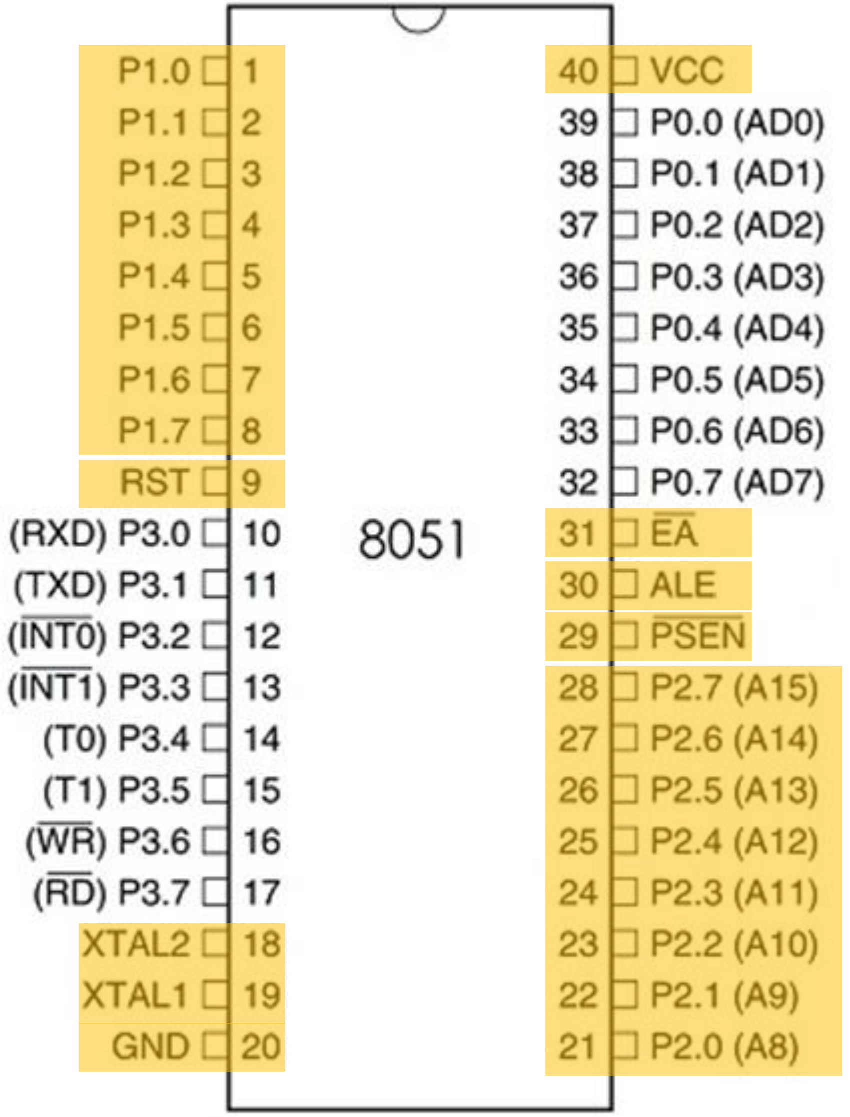
Internal RAM and external RAM both are works separately.

Instructions for both memories are different.

EA = Logic 1 (VCC) for on-chip ROM to store programs.

EA = Logic 0 (GND) for external ROM to store programs. (8031 and 8032)

Pin Diagram of 8051



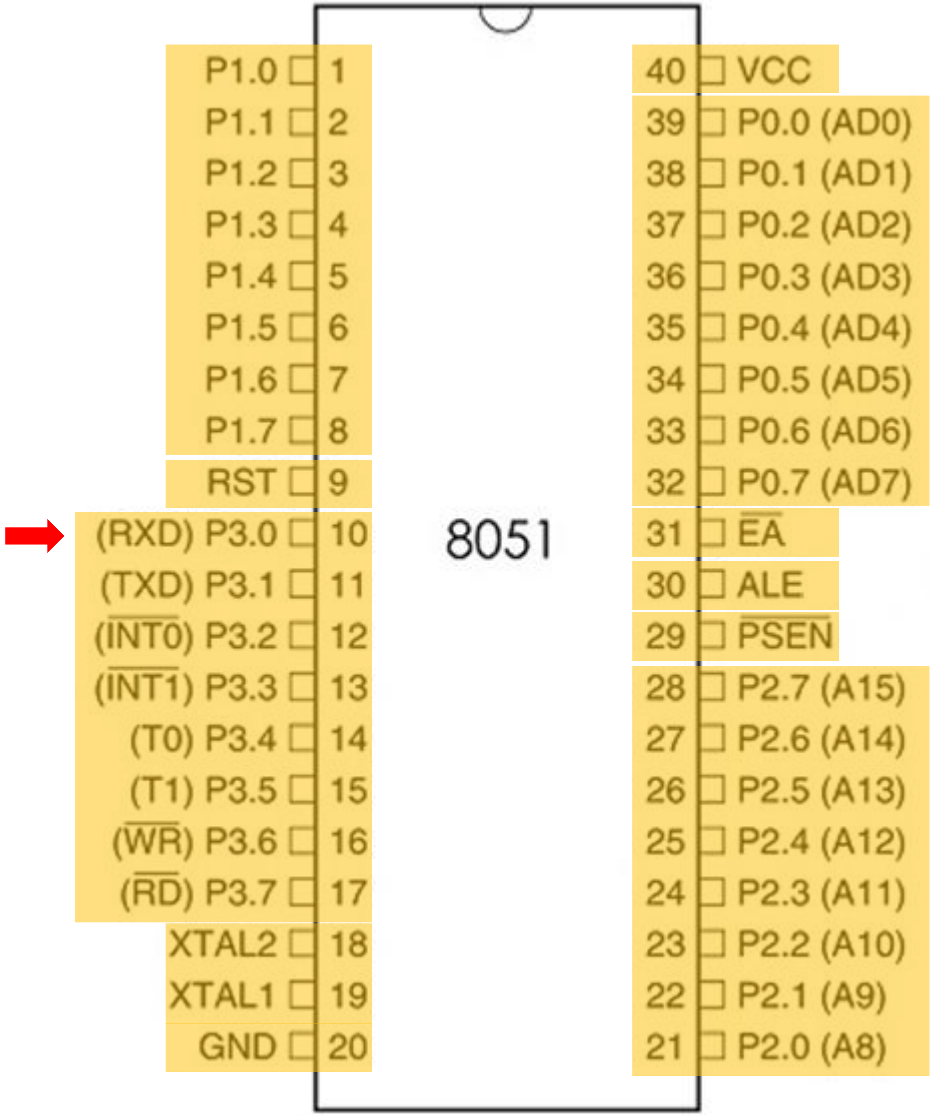
Pin 1 to Pin 8 (Port 1) –

These pins are bidirectional and bit addressable.
P1 doesn’t have any alternative functions.
It has a built-in pull-up resistor.
They can be configured as input or output pins depending on the logic i.e. if logic zero (0) is applied to the I/O port it will act as an output pin and if logic one (1) is applied the pin will act as an input pin.
These pins are also referred to as P1.0 to P1.7

Pin 21 to Pin 28 (Port 2) –

Pin 21 to pin 28 are port 2 pins also referred to as P2.0 to P2.7.
These pins are bidirectional and bit addressable.
PORT 2 pins act as Input or Output pins
When additional external memory is interfaced with the 8051 microcontroller, pins of port 2 act as higher-order address bytes.

Pin Diagram of 8051



Pin 32 to Pin 39 (Port 0) –

These pins are bidirectional IO pins and bit addressable.
They don’t have any internal pull-ups.
10 K ohm pull-up registers are used as external pull-ups.
Port 0 is also designated as AD0-AD7 because 8051 multiplexes low order address and data through port 0 to save pins.

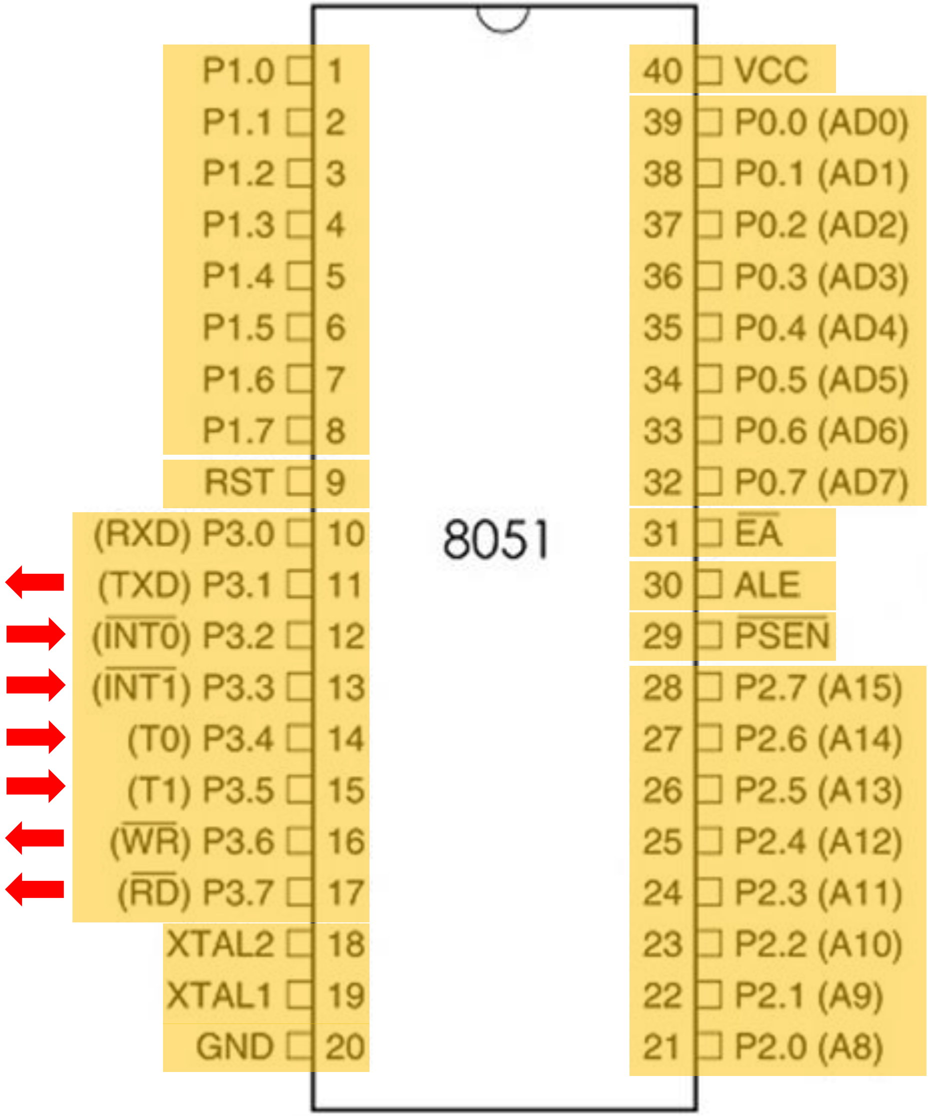
Pin 10 to Pin 17 (Port 3) –

Pin 10 to pin 17 are port 3 pins which are also referred to as P3.0 to P3.7.
These pins are bidirectional and bit addressable.
PORT 3 pins act as Input or Output pins.
These pins also have some special functions.

P3.0 (RXD) :

10th pin is RXD (serial data receive pin) which is for serial input.
Microcontroller receives data for serial communication.

Pin Diagram of 8051



P3.1 (TXD) :

11th pin is TXD (serial data transmit pin) which is serial output pin. Microcontroller transmits data for serial communication.

P3.2 and P3.3 (INT0′, INT1′) :

12th and 13th pins are for External Hardware Interrupt 0 and Interrupt 1 respectively. When this interrupt is activated(i.e. when it is low), 8051 gets interrupted and jumps to the vector value of the interrupt and starts performing ISR from that vector location.

P3.4 and P3.5 (T0 and T1) :

14th and 15th pin are for Timer 0 and Timer 1 external input. They can be connected with 16 bit timer/counter.

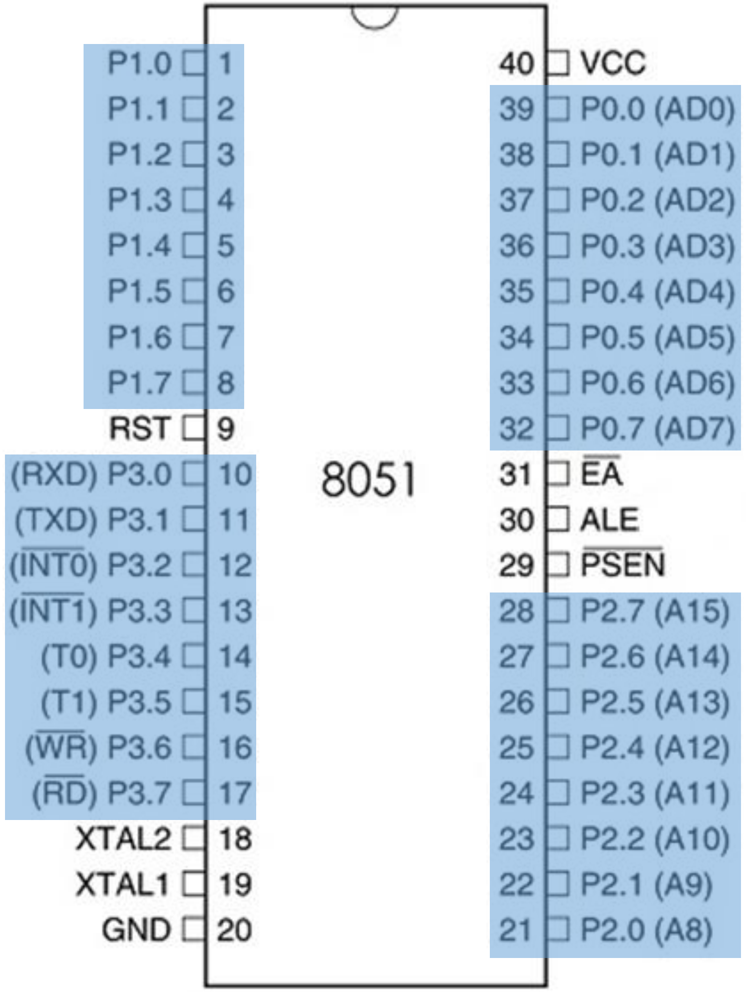
P3.6 (WR’) :

16th pin is for Ext. memory write i.e. writing data to the Ext. memory.

P3.7 (RD’) :

17th pin is for Ext. memory read i.e. reading data from Ext. memory.

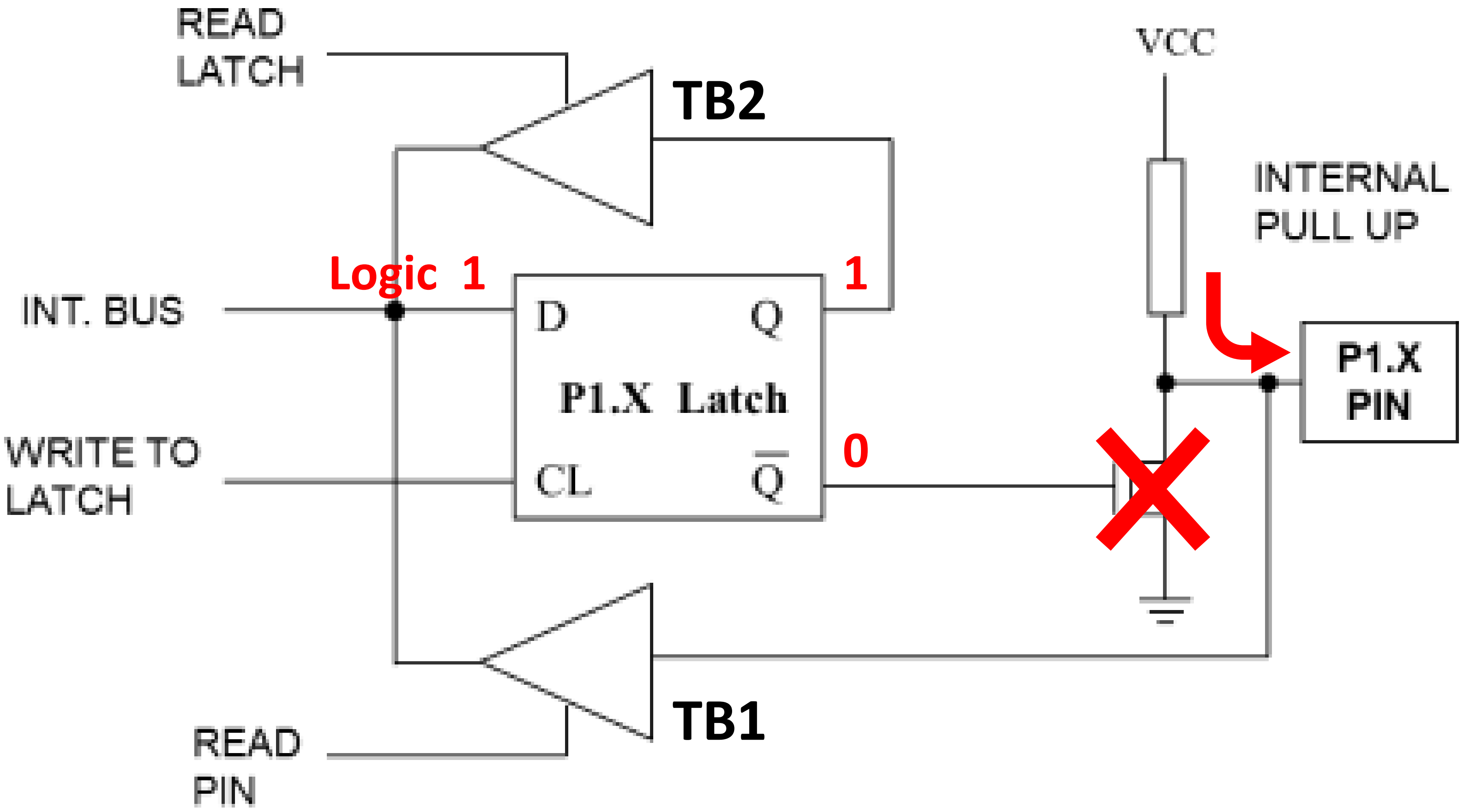
Ports of 8051



- 4 bidirectional IO ports. P0-P3
- Each of 8 bits.
- Can be configured of input and output.
- P1, P2 and P3 have internal pull up resistor.
- P0 is not having internal pull resistor.
- Port 0 (Pins 32-39): P0（P0.0~P0.7）
- I/O + Lower address + data bus
- Port 1 (Pins 1-8): P1（P1.0~P1.7）I/O Bus
- Port 2 (Pins 21-28): P2（P2.0~P2.7）
  I/O + Higher Address
- Port 3 (pins 10-17): P3（P3.0~P3.7)
  I/O + Special function

Port 1 Structure

WRITE OPERATION: Write ‘1’ to pin P1.X



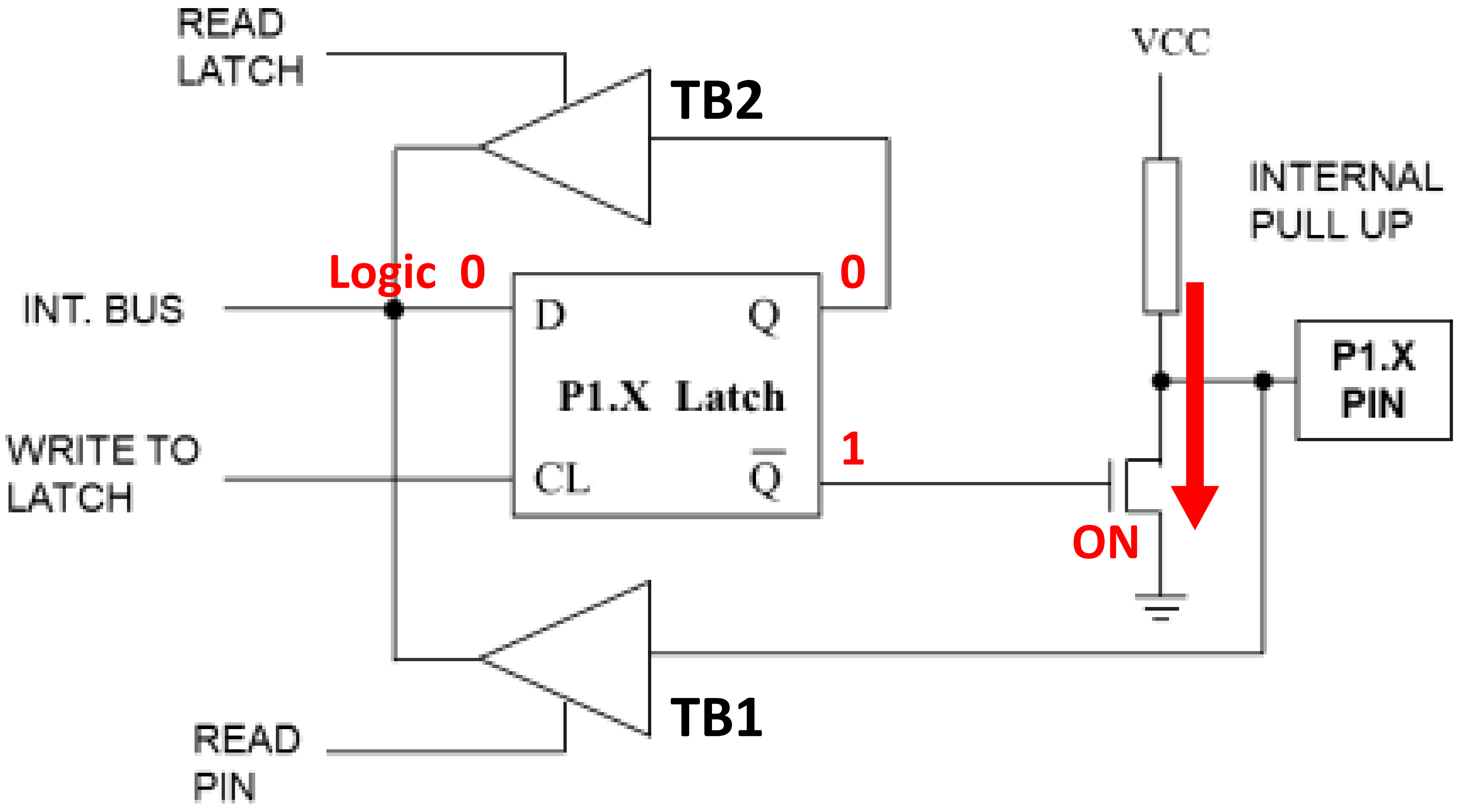
When D = 1, Q = 1, $\overline{Q}$ = 0.

Switch is OFF.

P1.X is connected to VCC.

Port 1 Structure

WRITE OPERATION: Write '0' to pin P1.X



= 0

When D = 0, Q = 0, $\overline{Q}$ = 1.

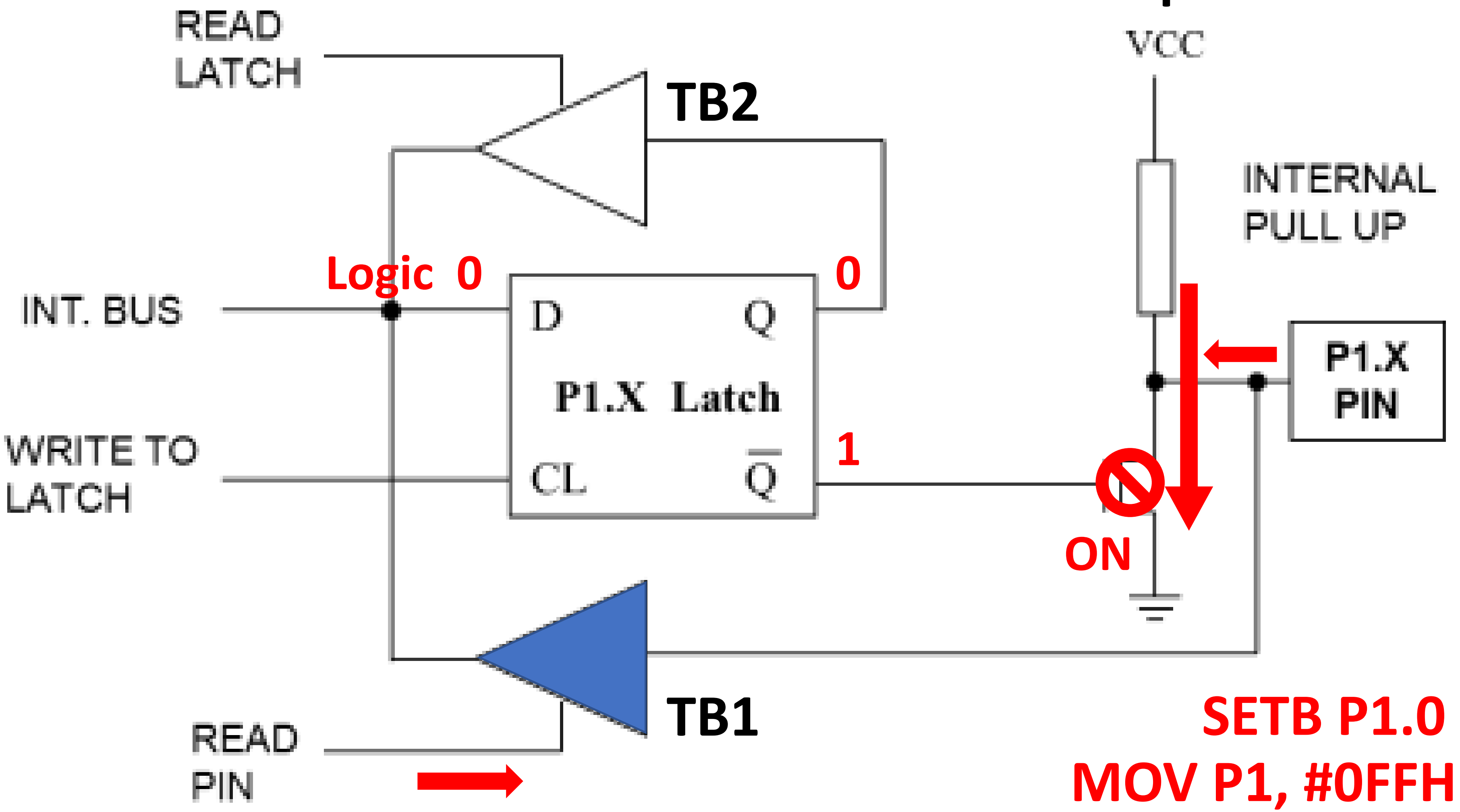
Switch is ON.

P1.X is grounded.

Port 1 Structure

READ OPERATION: (PRECAUTION)

Write '1' to port before read operation.



When D = 0, Q = 0, $\overline{Q}$ = 1.

= 1

Switch is ON.

VCC is grounded.

P1.X is at logic 1, supplying current.

Read signal is given.

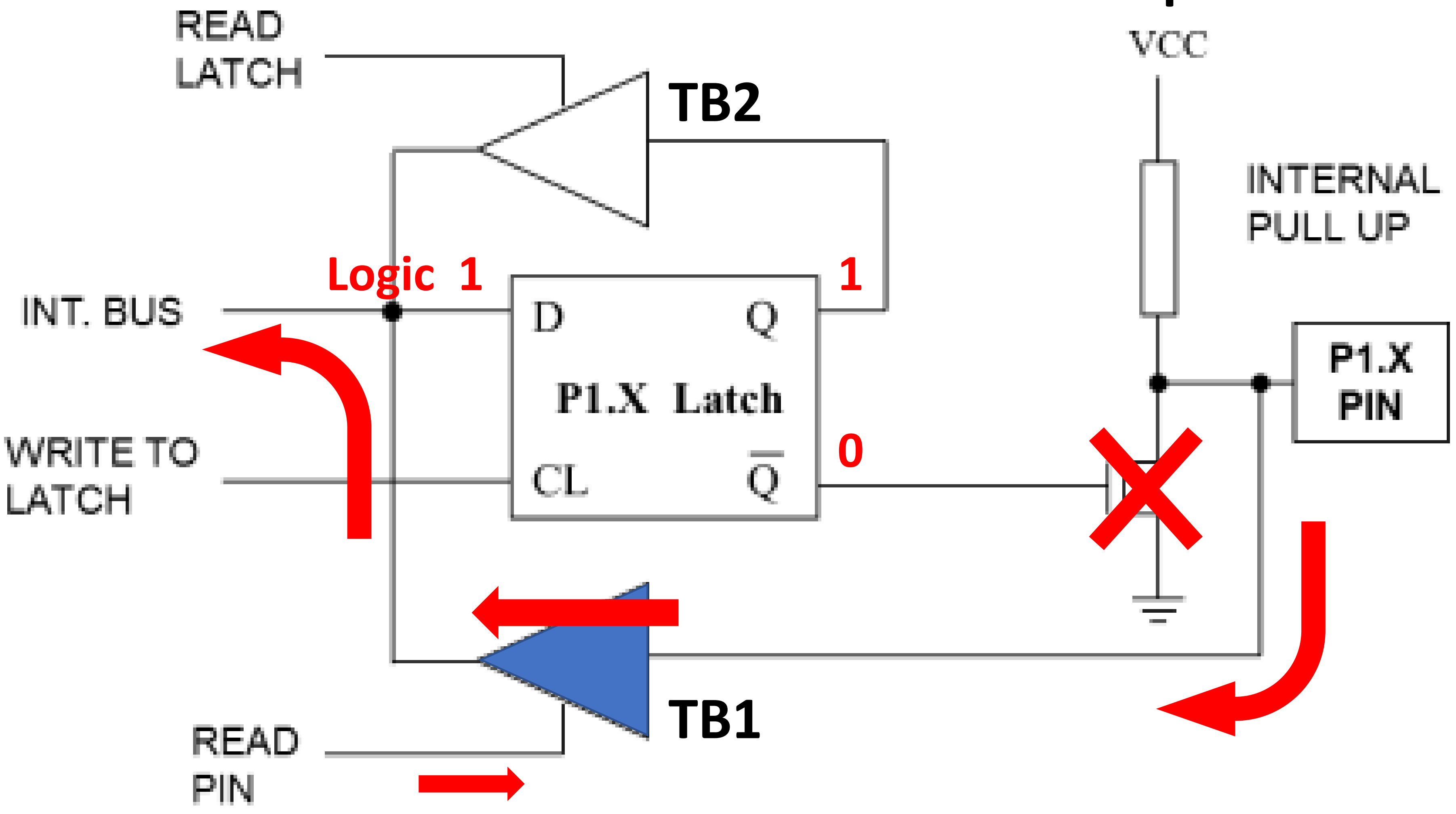
SETB P1.0

MOV P1, #0FFH

Port 1 Structure

READ OPERATION: (PRECAUTION)

Write ‘1’ to port before read operation.



= 1

When D = 1, Q = 1, Q̄ = 0.

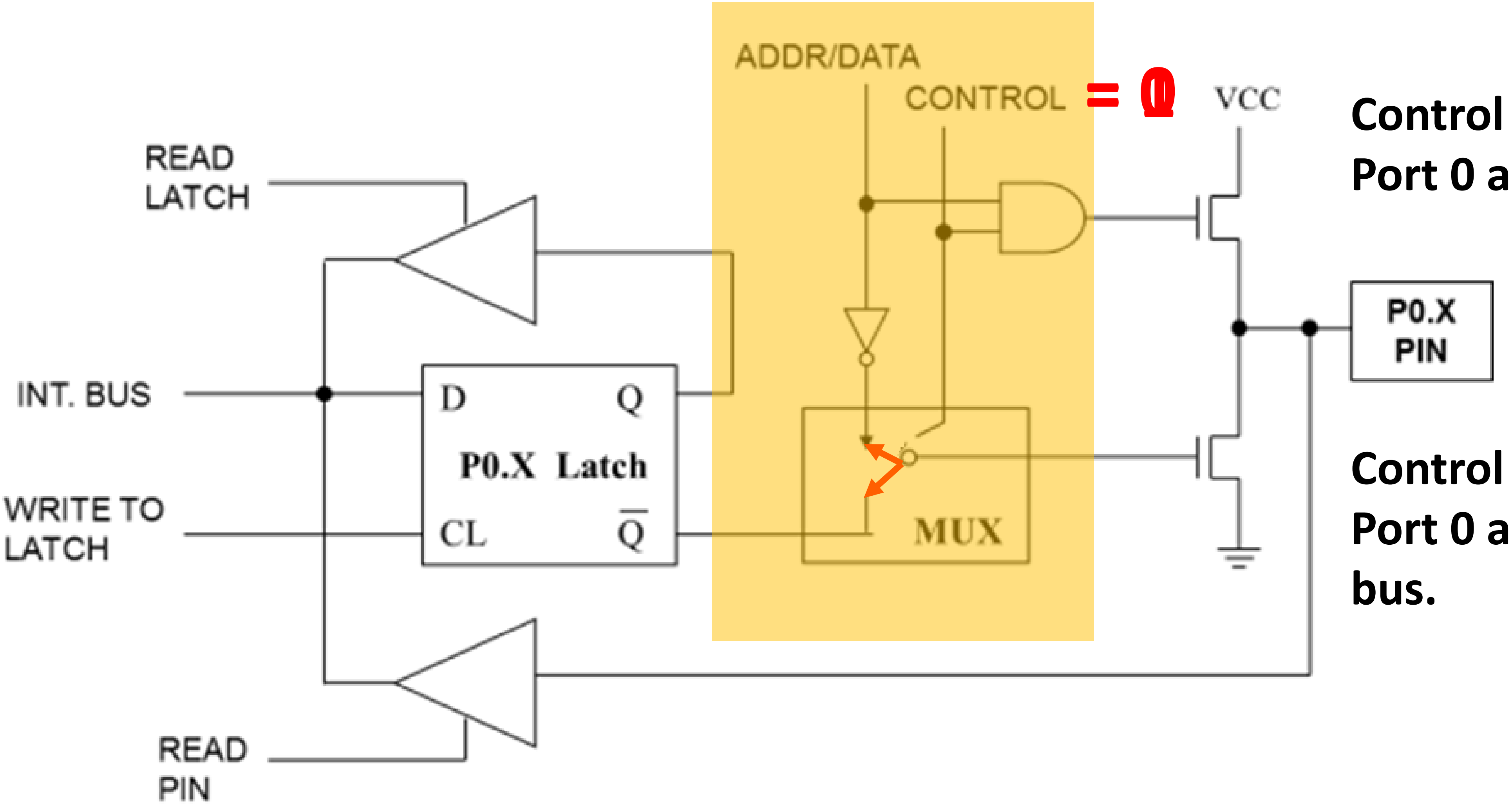
Switch is OFF.

P1.X is at logic 1.

Read signal is given.

Data is placed on internal bus.

Port 0 Structure



Control = 0

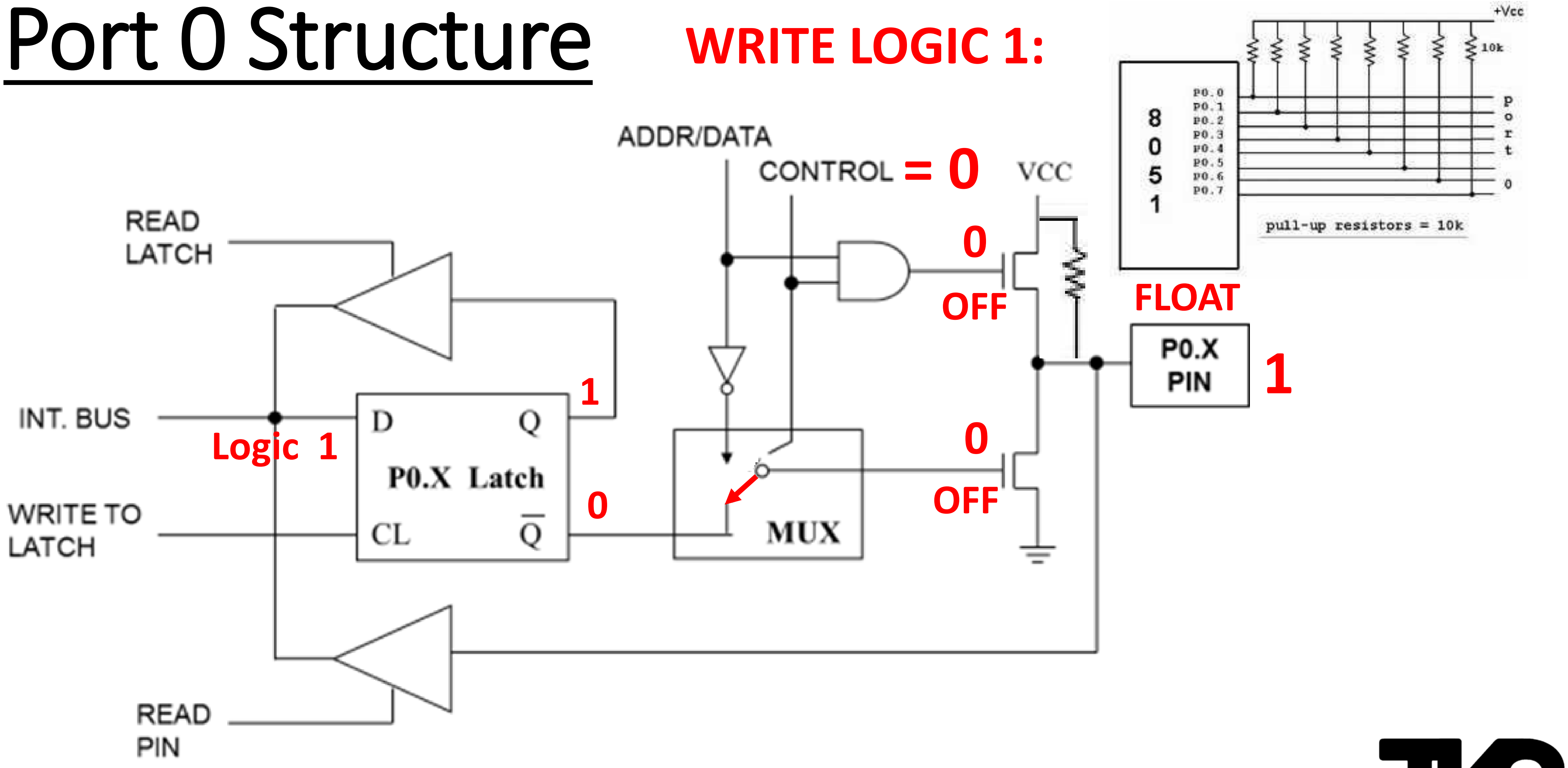
Port 0 acts as IO bus.

Control = 1

Port 0 acts as Address bus.

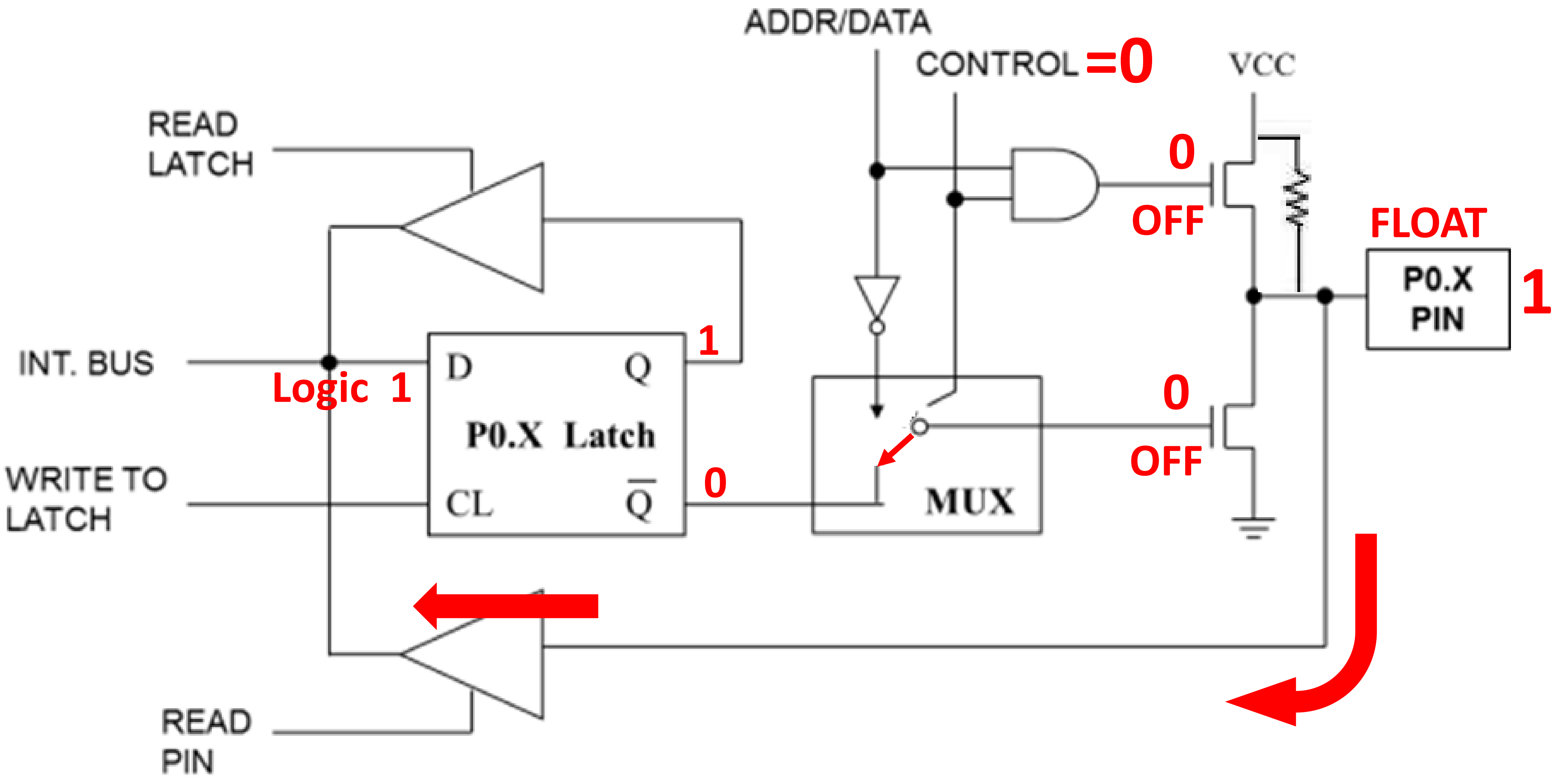
Port 0 Structure

WRITE LOGIC 1:

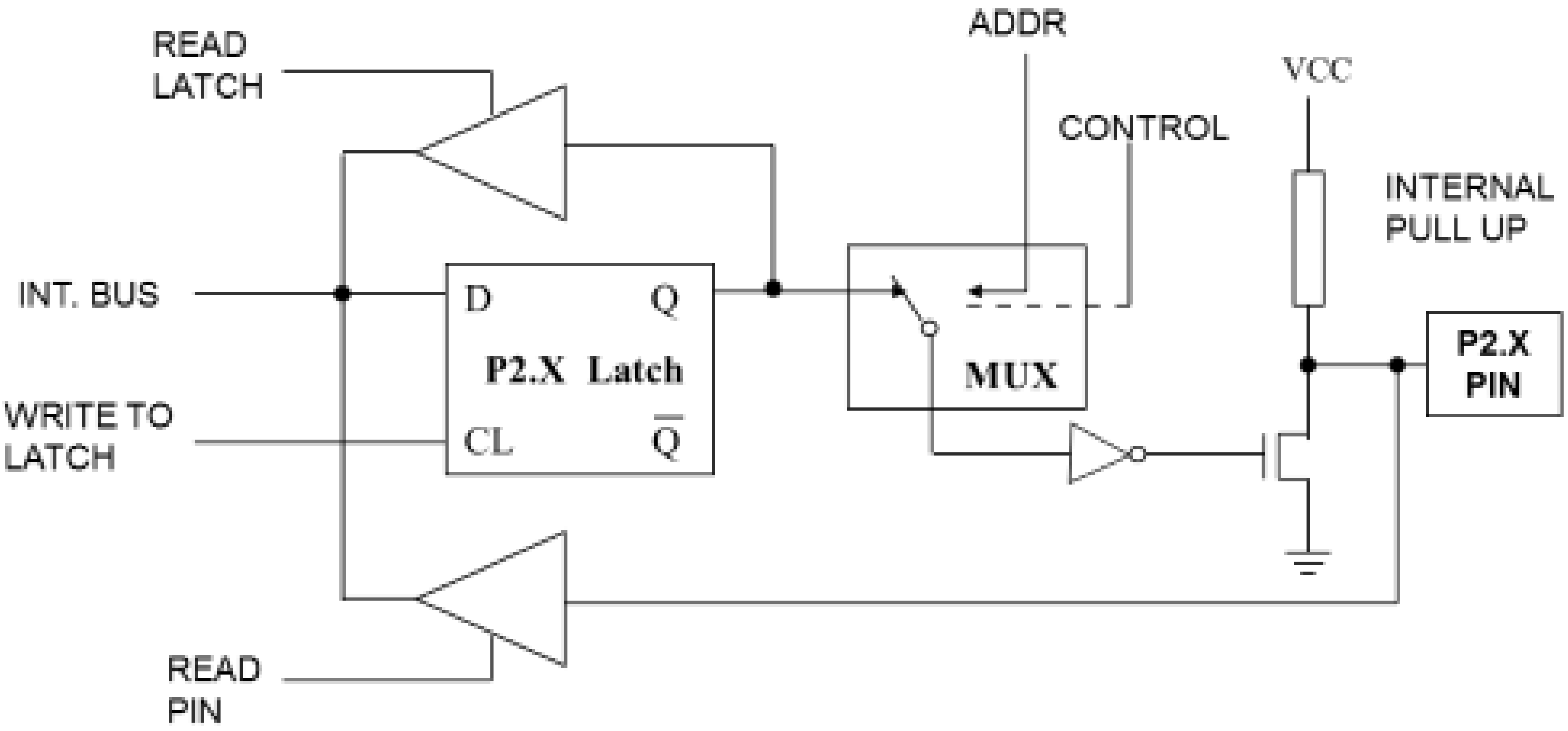


Port 0 Structure

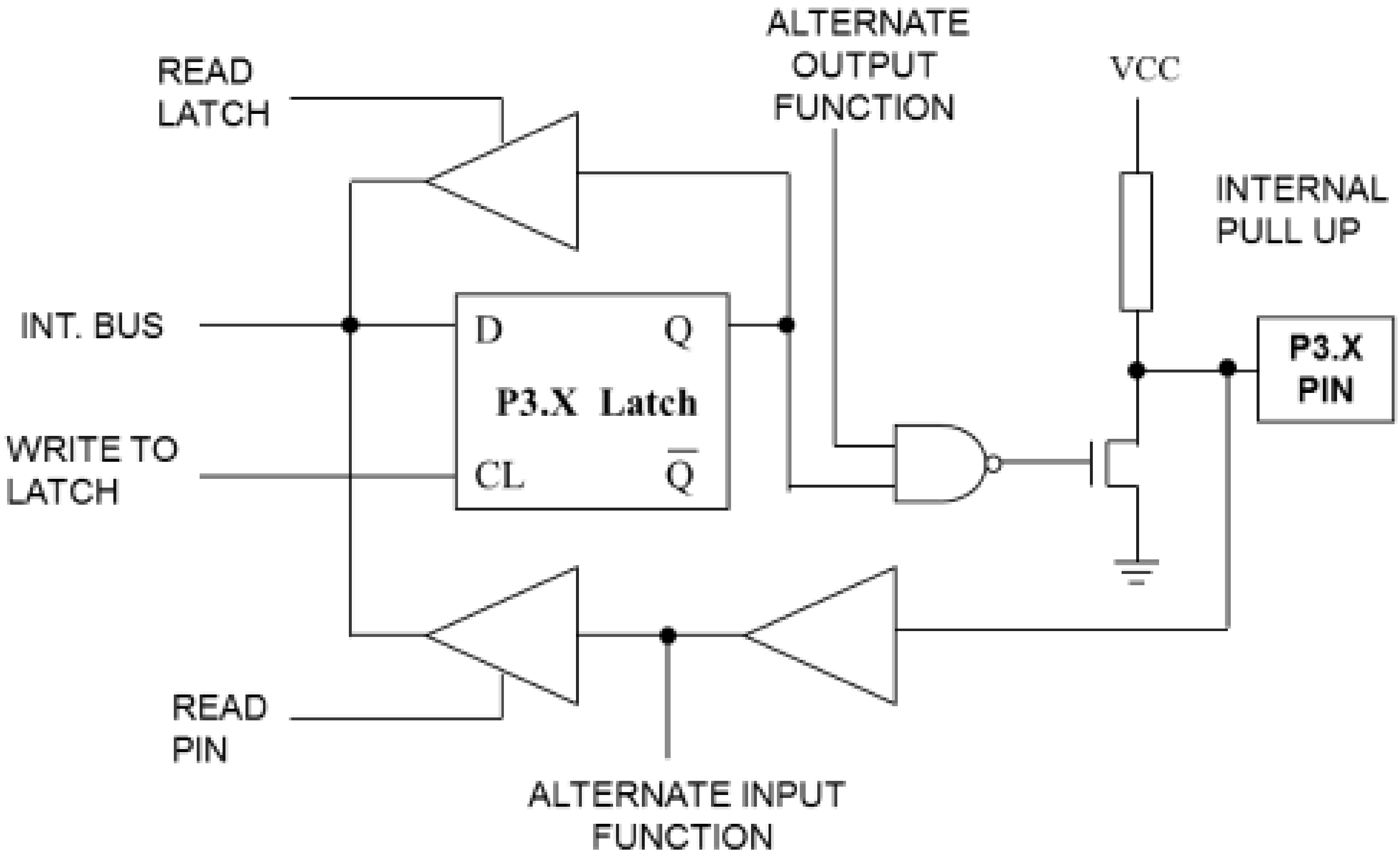
READ OPERATION LOGIC 0



Port 2 Structure



Port 3 Structure



Addressing Modes of 8051

- Every instruction is divided into 2 parts.
- Opcode: The opcode is the instruction that is executed by the CPU.
- Operand: The operand is the data or memory location used to execute that instruction.

- Example:- ADD A,B
- ADD is opcode.
- Reg. A and Reg. B are operands.

Addressing Modes of 8051

- **Addressing mode is a way to address an operand.**
- **Operand means the data we are operating upon (in most cases source data).**
- **It can be a direct address of memory, it can be register names, it can be any numerical data.**
- **It is a method to specify operand in an instruction.**

1) **Immediate addressing mode.**
2) **Register addressing mode.**
3) **Direct addressing mode.**
4) **Register indirect addressing mode.**
5) **Indexed addressing mode.**

Immediate Addressing Mode

- **In this Immediate Addressing Mode, the data is provided in the instruction itself.**
- **The data is provided immediately after the opcode.**
- **This addressing mode is named as "immediate" because it transfers an 8-bit data immediately to the accumulator.**

Example: MOV A, #07H

The # in the operand indicates that it is a data and not the address of a Register.
The Accumulator is loaded with 07 Hexadecimal.
A = 07H

More examples:

MOV B, #64H
MOV R5, #39H
MOV DPTR, #0A234H

Immediate data can not be a destination.

Register Addressing Mode

- In the register addressing mode the source or destination data should be present in a register (R0 to R7) or A.
- It is important to select the appropriate Bank with the help of PSW Register.

Example: MOV A, R0

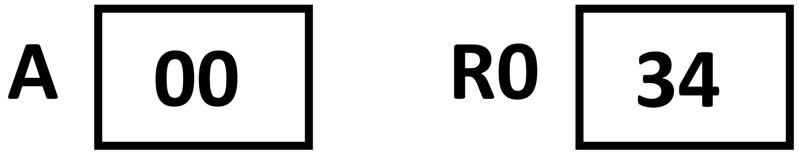
More examples:

MOV R5, A

~~MOV R3, R1~~

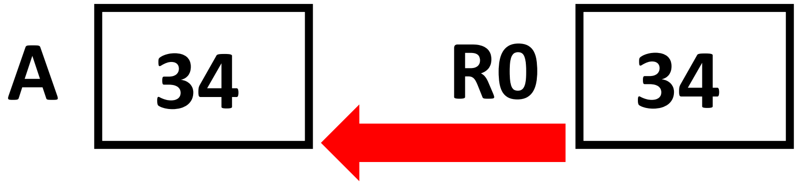
The 8-bit content of the Register R0 of Bank0 is moved to the Accumulator.

Before Execution:


A 00

R0 34

After Execution:


A 34

R0 34

Direct Addressing Mode

- All 128 bytes of internal RAM and the SFRs can be addressed.
- Data movement is between register to memory location or vice versa.
- Address is the part of the instruction as source or destination

Syntax:

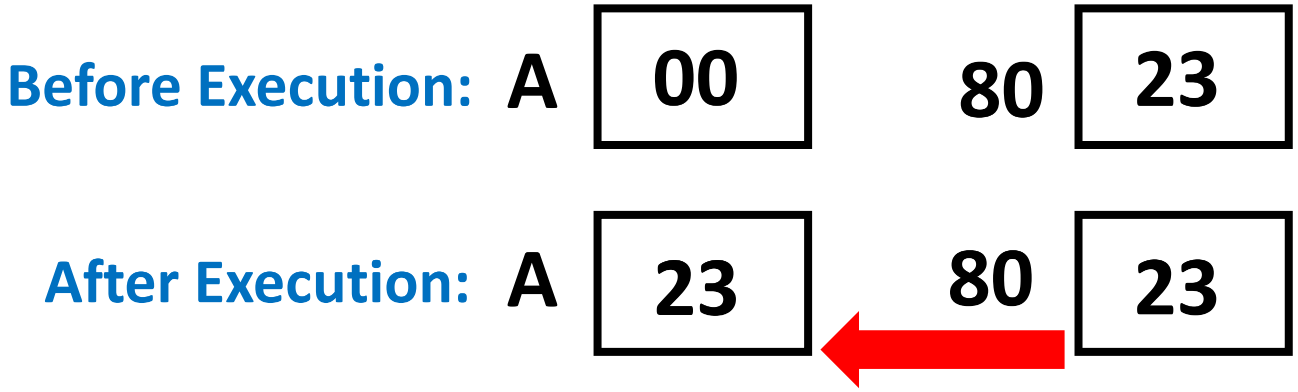
MOV A, Add	Copy data from memory to A.
MOV Add, A	Copy data from A to memory.
MOV Rr, Add	Copy data from memory to register.
MOV Add, Rr	Copy data from register to memory.
MOV Add, #n	Copy immediate data to memory.
MOV Add1, Add2	Copy data from memory 2 to memory 1.

Direct Addressing Mode

Example:

MOV A, 80H

The 8-bit content of the memory location having address 80H is moved to the Accumulator.



More examples:

MOV 3AH, #3AH

MOV R3, 12H

MOV 0AH, 77H

Register Indirect Addressing Mode

- Address of the data (source data to transfer) is given in the register operand.
- The @ symbol indicates that the addressing mode is indirect.
- Only R0 and R1 are allowed in Indirect Addressing Mode.
- R0 and R1 are called as Pointer registers.

Syntax:

MOV @Rp, #n	Copy immediate data to the memory in Rp.
MOV @Rp, Add	Copy data from memory to address in Rp.
MOV @Rp, A	Copy data from A to address in Rp.
MOV A, @Rp	Copy data from address in Rp to A
MOV Add, @Rp	Copy data from address in Rp to address.

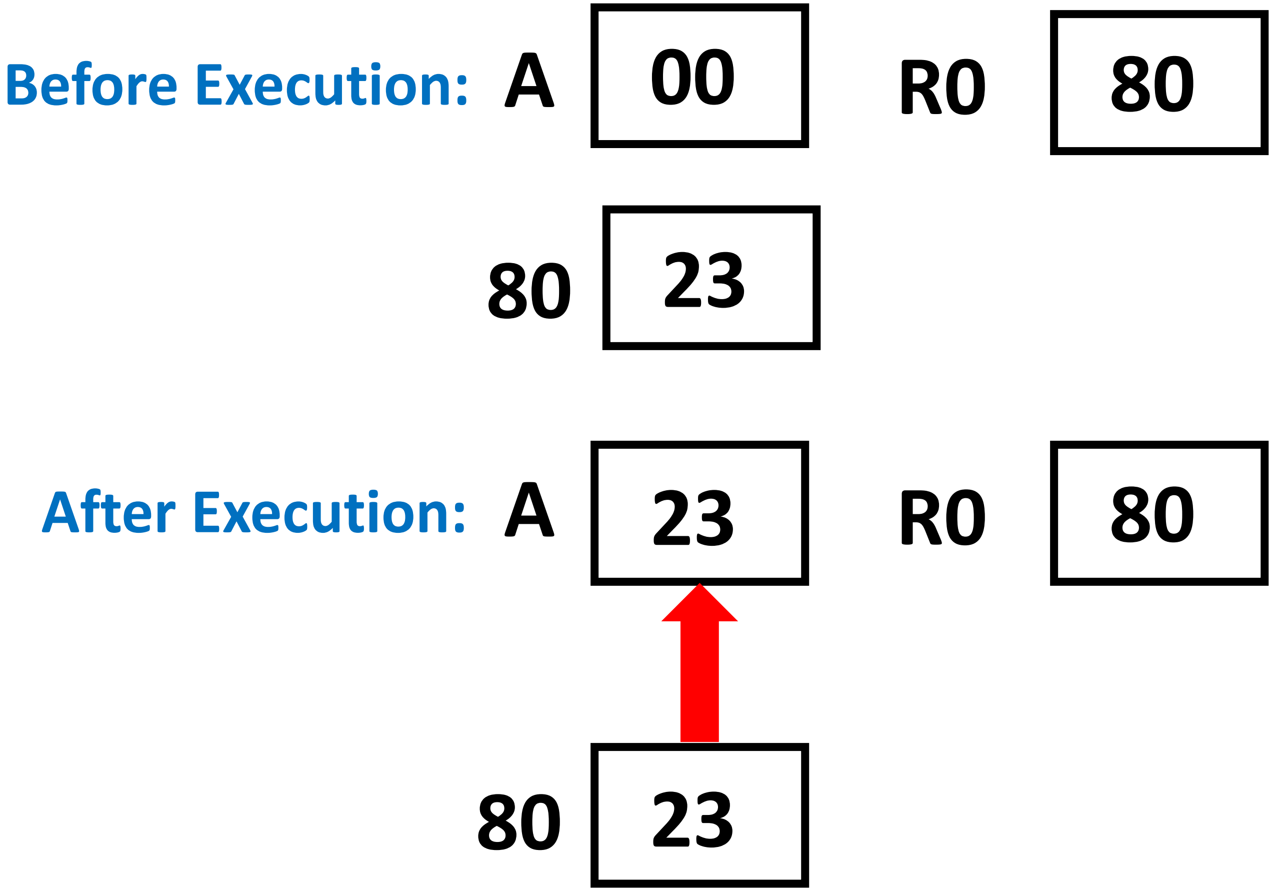
Register Indirect Addressing Mode

Example:

MOV A, @R0

The 8-bit content of the memory location given by R0 is moved to the Accumulator.

The number in register Rp must be RAM or SFR address.



Before Execution: A 00 R0 80

80 23

After Execution: A 23 R0 80

80 23

More examples:

MOV @R1, #3AH

MOV @R0, 18H

MOV 0AH, @R1

Indexed Addressing Mode

- It is widely used in accessing data elements of look-up table entries located in the program Rom space of the 8051.
- The effective address of the Operand is the sum of a base register(PC or DPTR) and an offset register (A).
- Only MOVC and JMP instructions can be used.
- The C in MOVC instruction refers to code byte.

Syntax:

MOVC A, @A+DPTR	Copy the code byte found at the ROM address by adding A and DPTR.
MOVC A, @A+PC	Copy the code byte found at the ROM address by adding A and PC.

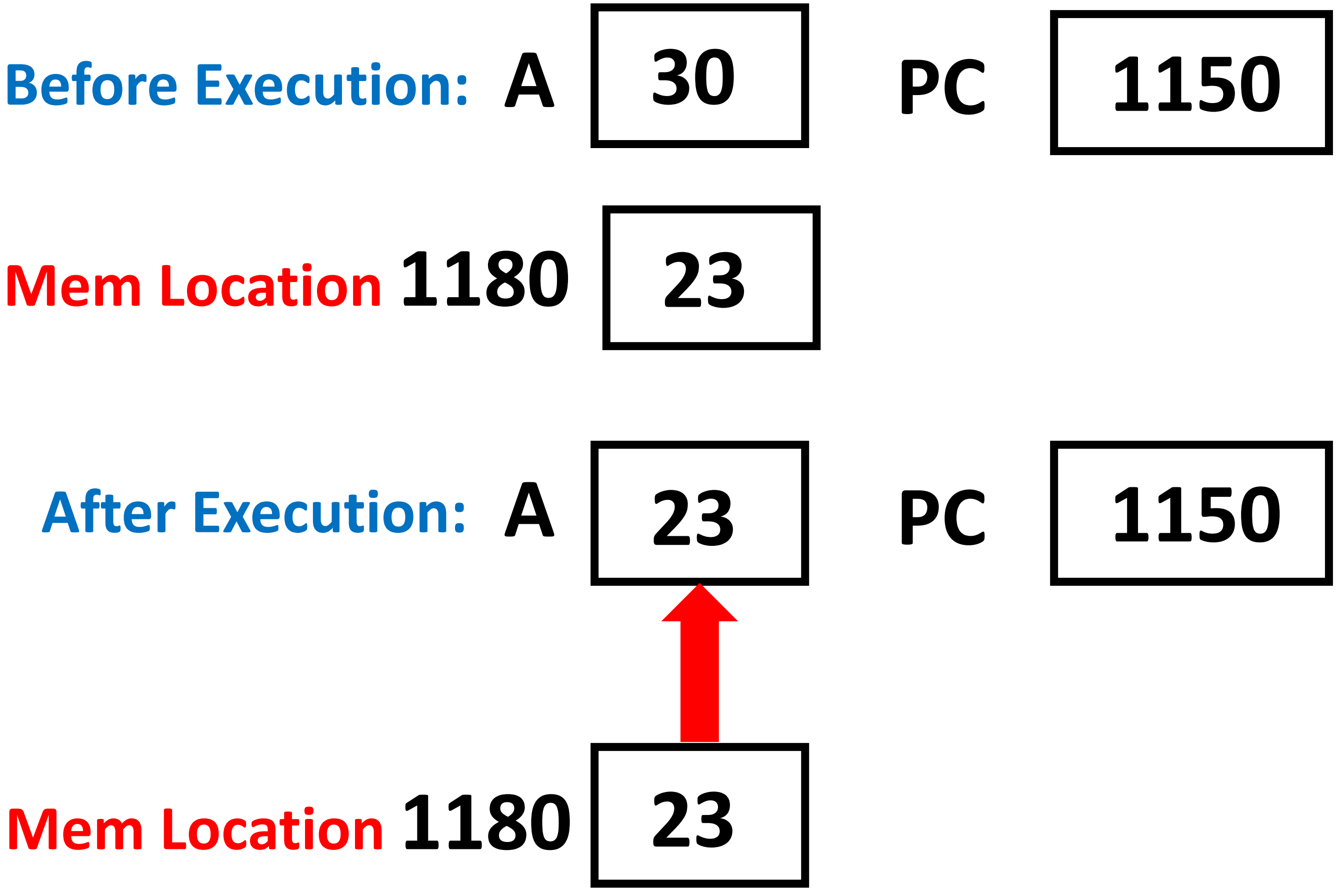
Indexed Addressing Mode

Example:

MOVC A, @A+PC

The 8-bit content of the memory location formed by adding 8 bit content of A and 16 bit PC is moved to the Accumulator.

MOVC is usually used with internal or external ROM and can address 4K of internal or 64K of external code.



Before Execution: **A** **30** **PC** **1150**

Mem Location 1180 **23**

After Execution: **A** **23** **PC** **1150**

Mem Location 1180 **23**

More examples:

MOVC A, @A+DPTR

Data transfer Instructions

Different types of instructions.

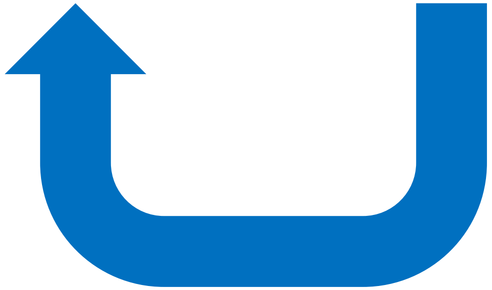
• Data transfer instructions	MOV, PUSH & POP XCHANGE
• Byte level Logical instructions	AND, OR, XOR, NOT
• Arithmetic instructions	ADD, SUB, INC, DCR, MUL, DIV, DAA
• Bit level instructions	AND, OR, XOR, NOT, CLR, CPL, SETB
• Jump and call instructions	CONDITIONAL JUMP
• Rotate and swap instructions	RR, RL, RLC, RRC, SWAP

Data transfer Instructions

MOV instruction

It copies data from one location to another.

MOV DESTINATION, SOURCE



After execution Source and Destination have same data.

It is not a MOV, but COPY operation.

Different types of MOV

- **MOV**
- **MOVX**
- **MOVC**

Data transfer Instructions

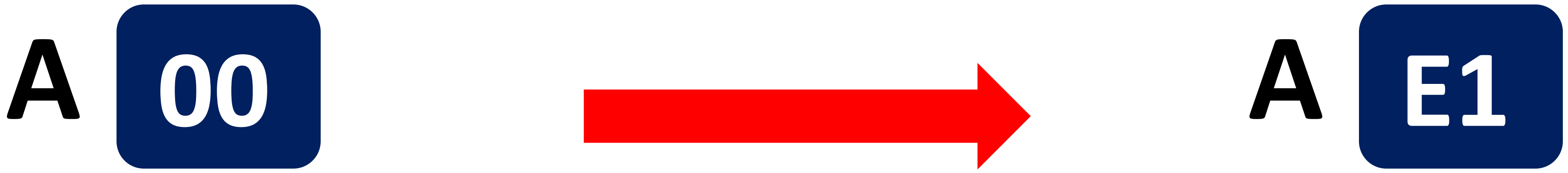
Mnemonic:

MOV A, #n	**Copy immediate data n to A.**
MOV A, Rr	**Copy data from register Rr to register A.**
MOV Rr, A	**Copy data from register A to register Rr.**
MOV Rr, #n	**Copy immediate data n to register Rr.**
MOV DPTR, #nn	**Copy immediate 16 bit data ‘nn‘ to the DPTR register.**

Data transfer Instructions

Example:

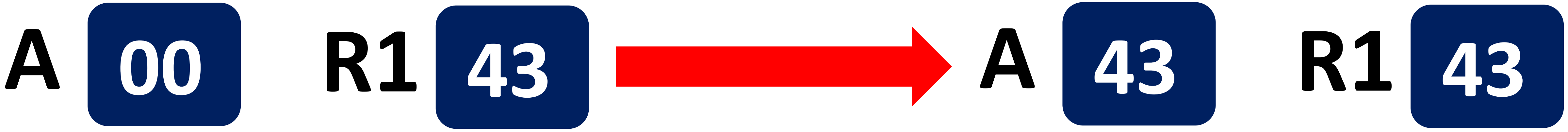
MOV A, #0E1 H — Load the immediate data E1 h to A register.



Before execution — After execution

Example

MOV A, R1 — Copy 8 bit data in register R1 to A register.



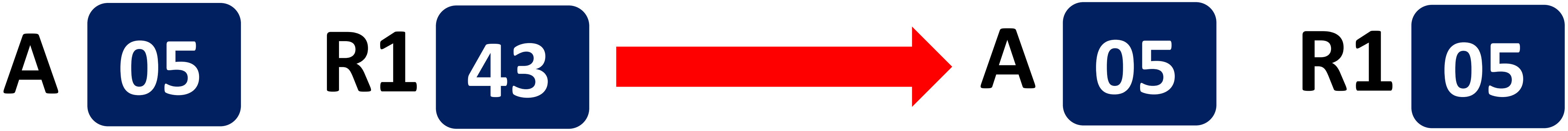
Before execution — After execution

Data transfer Instructions

Example:

MOV R1, A

Copy 8 bit data in A register to R1 register.



A 05 R1 43

A 05 R1 05

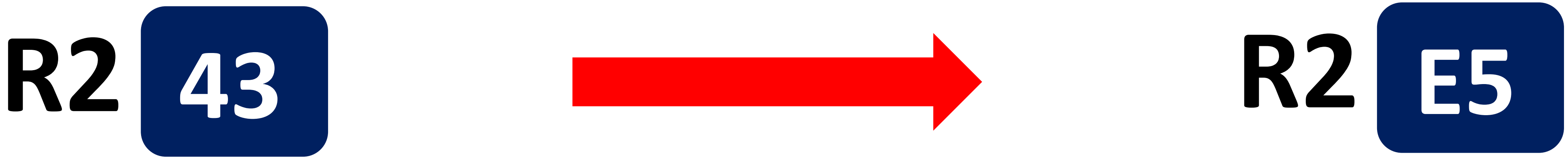
Before execution

After execution

Example:

MOV R2, #0E5 H

Load 8 bit immediate data to R2 register.



R2 43

R2 E5

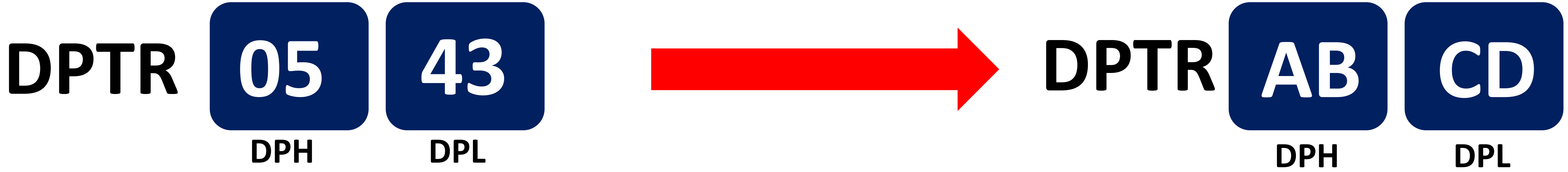
Before execution

After execution

Data transfer Instructions

Example:

MOV DPTR, #0ABCD H　　Load 16 bit immediate data in DPTR reg.



Before execution　　After execution

NOTES TO REMEMBER

1. It is impossible to have immediate data as a destination.
2. Register to register move occurs between reg. A and R0-R7 reg.

External data transfer Instruction

- This instruction is used to access external memory.
- External memory can be as large as 64K.
- Mnemonic used is ‘MOVX’
- ‘X’ is added to MOV to indicate that data move is external.

Mnemonic:

MOVX A, @Rp	Copy content of external address in Rp to A reg.
MOVX A, @DPTR	Copy content of external address in DPTR to A reg.

Other instructions:

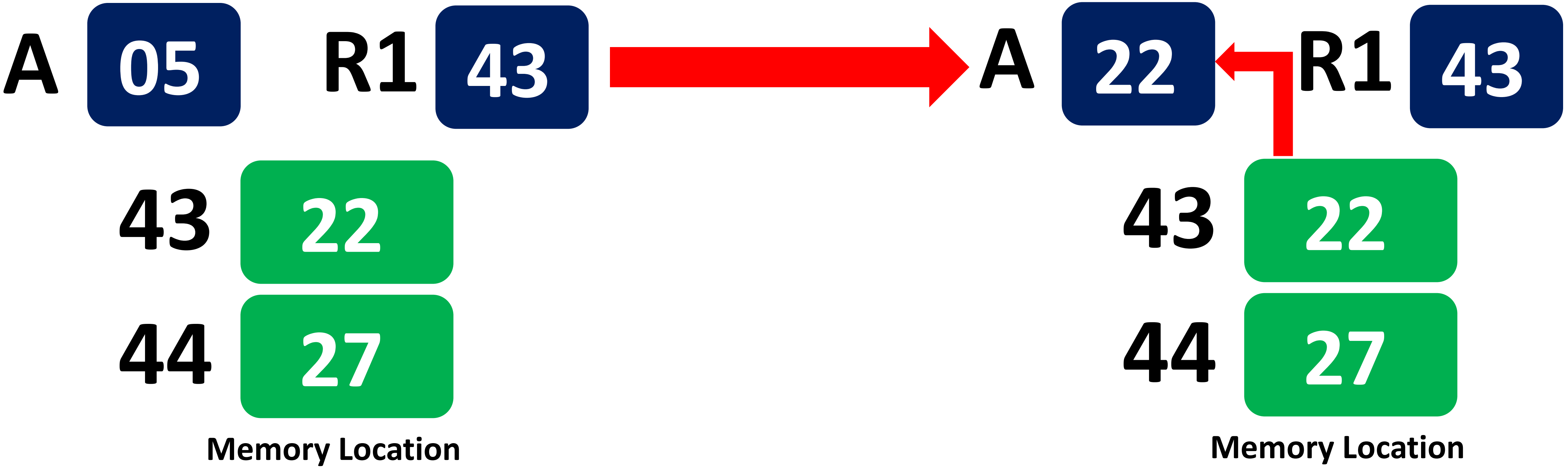
MOVX @Rp, A

MOVX @DPTR, A

External data transfer Instruction

Example:

MOVX A, @R1 (or R0) Copy content of external address in R1 to A.



A 05 R1 43

43 22

44 27

Memory Location

Before execution

A 22 R1 43

43 22

44 27

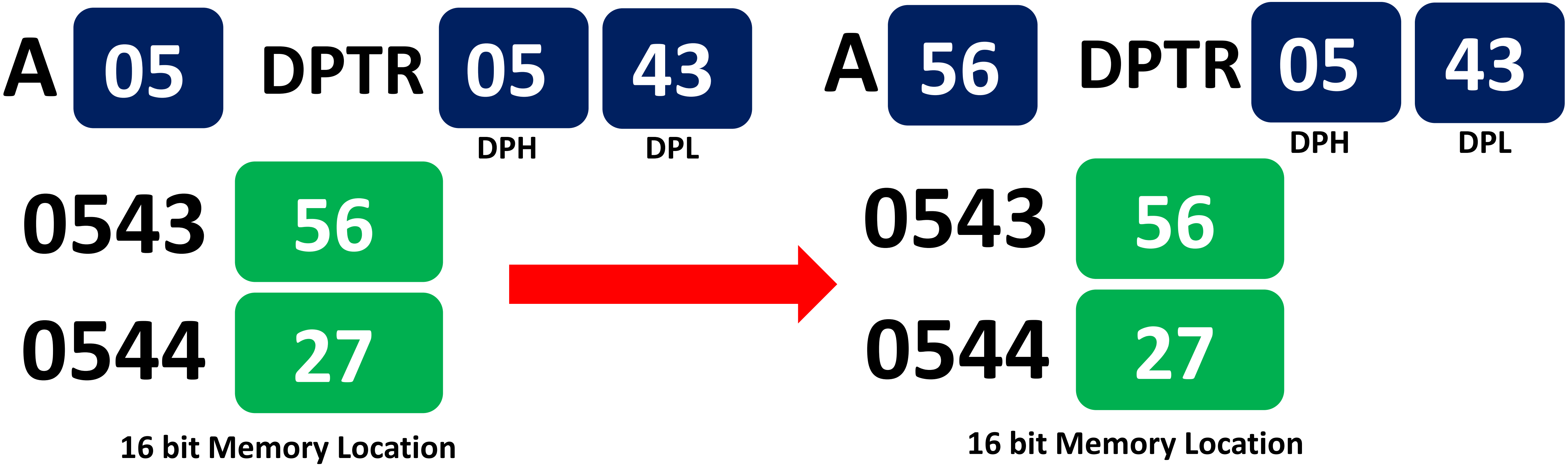
Memory Location

After execution

External data transfer Instruction

Example:

MOVX A, @DPTR Copy content of external address in DPTR to A.



A 05 DPTR 05 43

DPH DPL

0543 56

0544 27

16 bit Memory Location

A 56 DPTR 05 43

DPH DPL

0543 56

0544 27

16 bit Memory Location

Before execution

After execution

External data transfer Instruction

NOTES TO REMEMBER

1. **All external data moves always involves register A.**
2. **MOVX deals with external RAM and IO addresses.**
3. **Rp can address 256 bytes; DPTR can address 64K bytes.**
4. **MOVC instruction fetches the code from internal or external ROM.**
5. **Using MOVC, data is moved from the code memory to the A register.**

Push Operation

- Used to push data on the STACK.
- PUSH opcode copies the data from the source address to the stack.

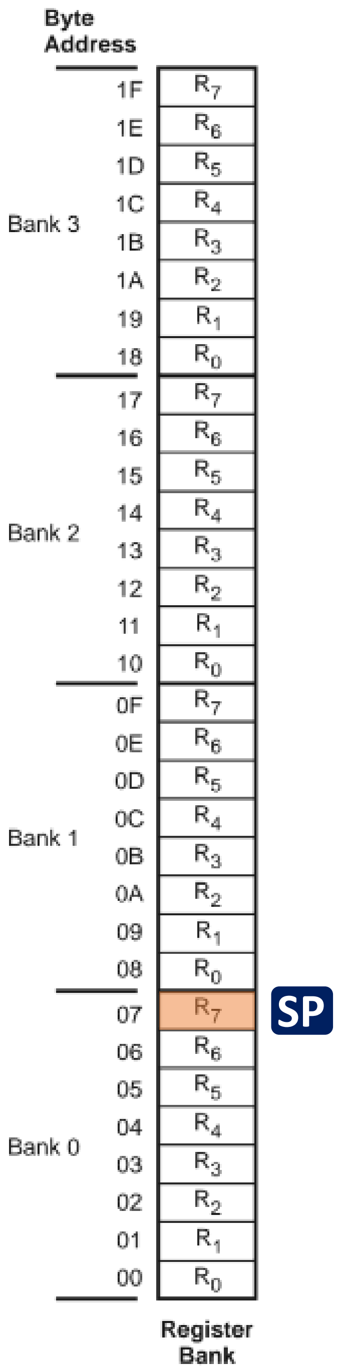
Mnemonic:

PUSH add

Increment SP;

Copy the data in the address to the internal RAM address contained in SP.

- Part of RAM in which data will store temporary during execution of program.
- SP (stack pointer) register is used to access.
- BY default SP register contains value 07.
- RAM location 08 is the first location.
- The stack can take a value between 00 to FFH.

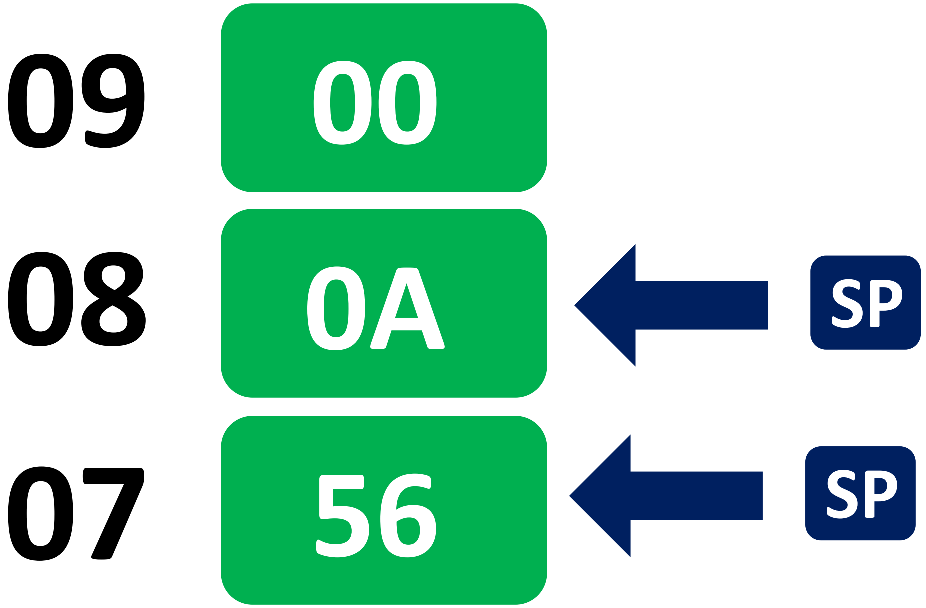


Push Operation

Example:

```
MOV R0, #0A H
PUSH 00 H
```

INTERNAL RAM



R0 00 SP 07

Before execution

R0 0A SP 08

After execution

00 H is the address of R0 of Register bank 0.

Pop Operation

- **Used to pop the data from stack.**
- **POP opcode copies the data from the stack address to the register address.**

Mnemonic:

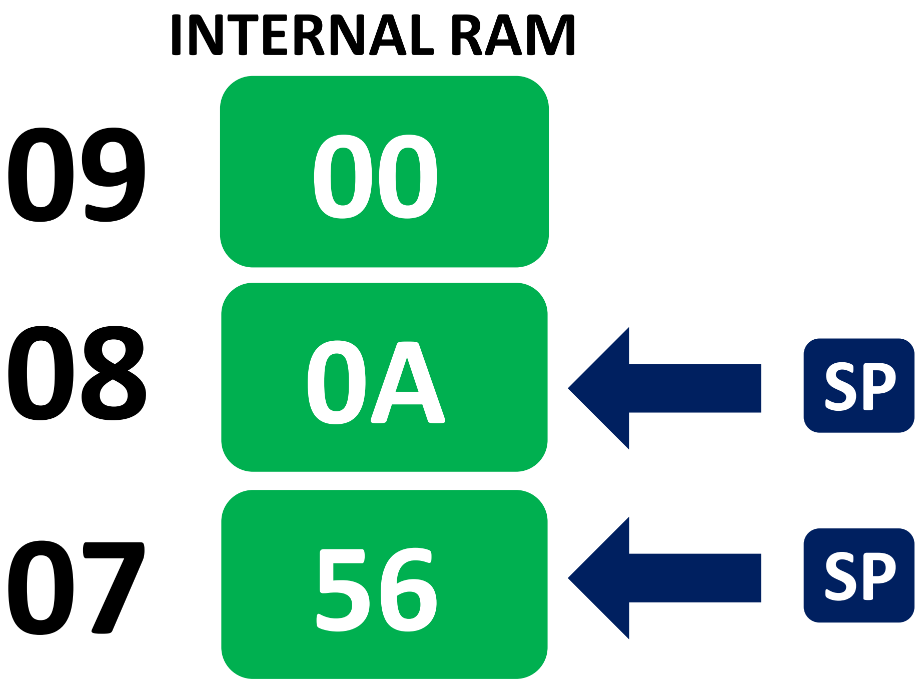
POP add

Copy the data from the internal RAM address contained in SP to address;
Decrement the SP

Pop Operation

Example:

```
MOV R0, #0A H
PUSH 00 H
POP  01 H
```



INTERNAL RAM

09	00
08	0A
07	56

R0 0A SP 08

Before execution

R1 0A SP 07

After execution

01 H is the address of R1 of Register bank 0.

Push and pop Operation

NOTES TO REMEMBER

1. **When the SP reaches FF H; it rolls over to 00 H.**
2. **RAM ends at address 7F H; PUSH above 7F H result in to errors.**
3. **SP is usually set at addresses above the resister banks.**
4. **Direct addresses must be used for stack operation.**
5. **No register names are used in PUSH and POP instructions.**

Xchange Instruction

- Used to exchange the data between source and destination operand.
- Register A is always involved in the exchange operation.

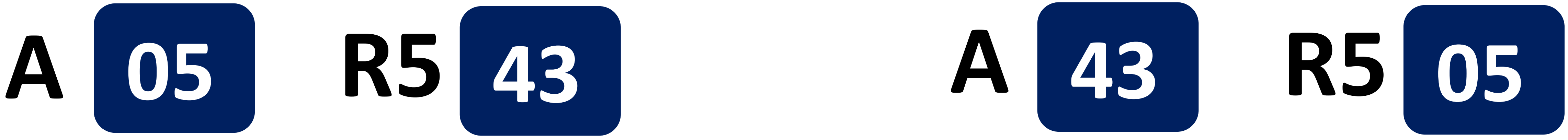
Mnemonic:

XCH A, Rr	**Exchange data bytes between register Rr and A.**
XCH A, add	**Exchange data bytes between address and A.**
XCH A, @Rp	**Exchange data bytes between A and address in Rp.**
XCHD A, @Rp	**Exchange lower nibble between A and address in Rp.**

Xchange Instruction

Example:

XCH A, R5 Exchange data bytes between register R5 and A.



A 05 R5 43

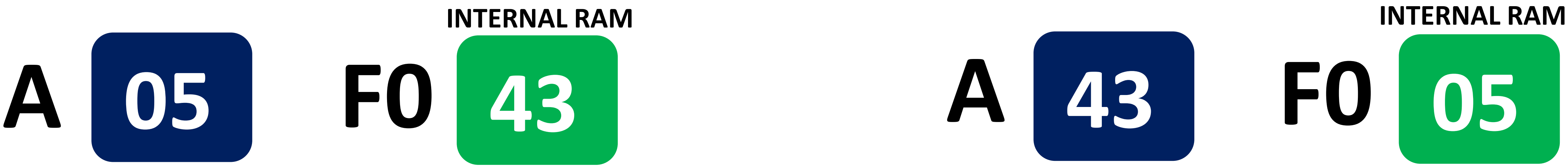
Before execution

A 43 R5 05

After execution

Example:

XCH A, 0F0 H Exchange data bytes between address F0 H and A.



A 05 F0 43 INTERNAL RAM

Before execution

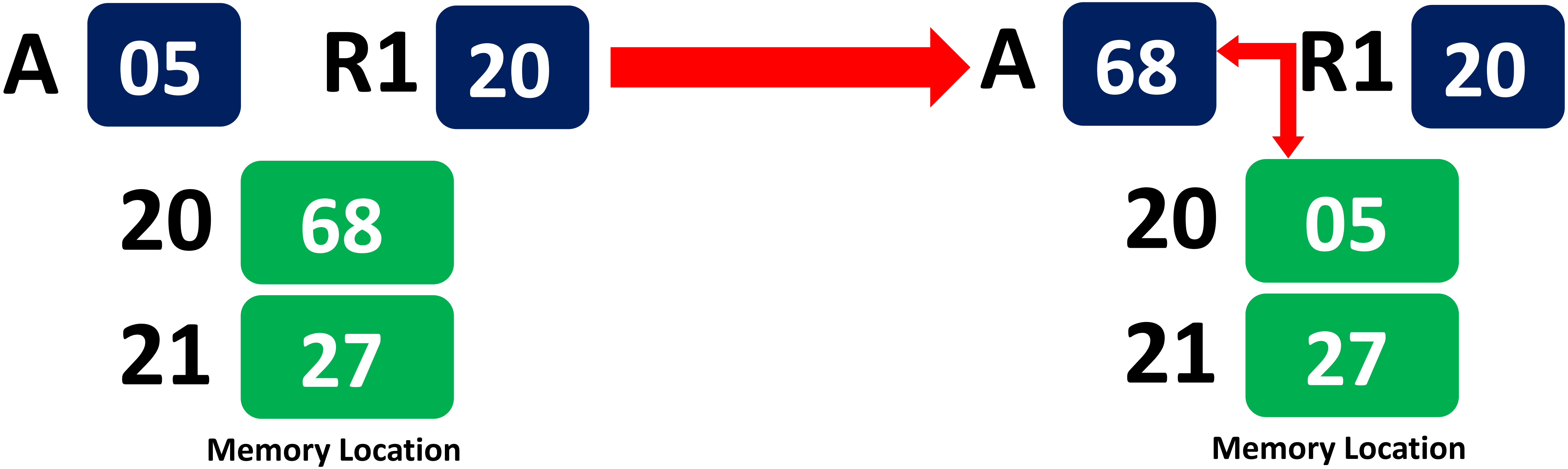
A 43 F0 05 INTERNAL RAM

After execution

Xchange Instruction

Example:

XCH A, @R1 (or R0) **Exchange data bytes between A and address in R0 or R1.**

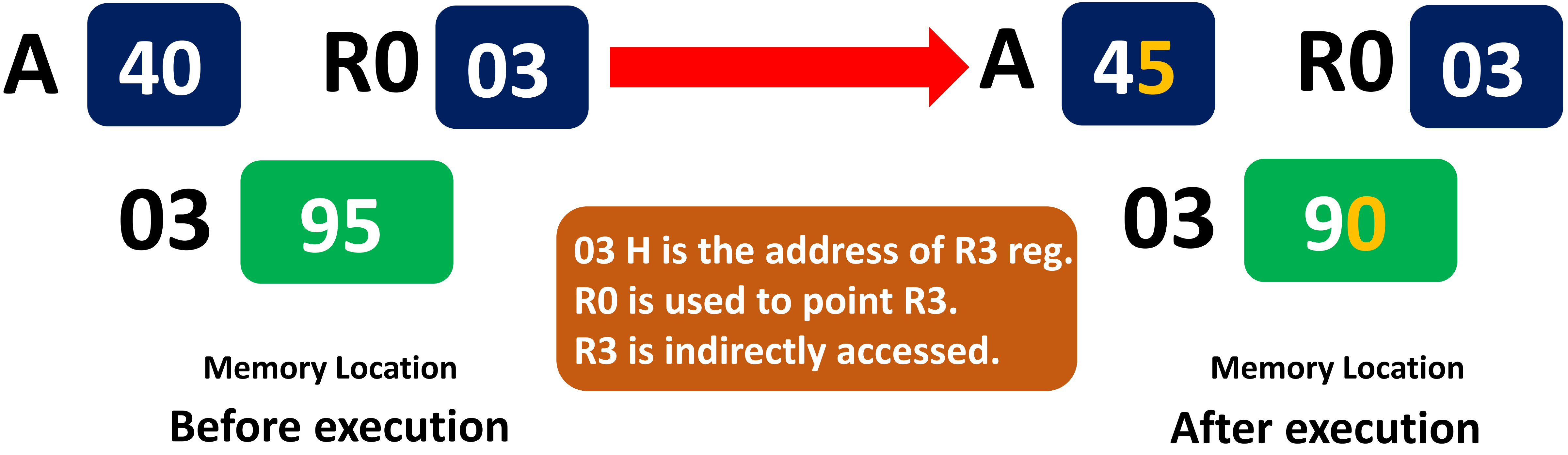


Before execution **After execution**

Xchange Instruction

Example:

XCHD A, @R0 Exchange lower nibble (lower four bits) of A reg. and lower nibbles of contents of address stored either in R0 or R1.



03 H is the address of R3 reg.
R0 is used to point R3.
R3 is indirectly accessed.

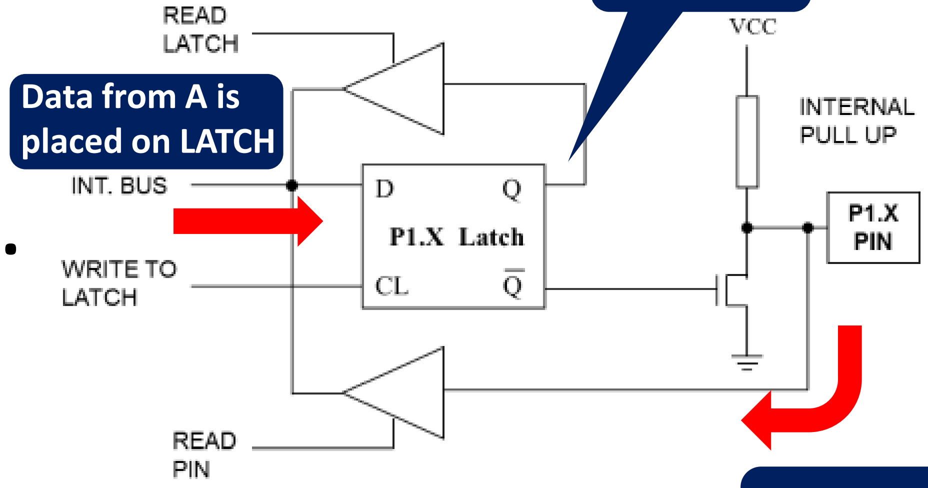
Xchange Instruction

NOTES TO REMEMBER

1. All exchanges are internal to the 8051.

2. All exchanges use register A.

3. Exchange between register A and any port location copy the data on the port pin to A, while the data in A is copied to the port latch.



Data from A appears here.

Data from A is placed on LATCH

Data is moved to A reg.

Increment Instruction

- Used to increment the data in an operand by 1.
- Operand could be register, RAM direct address, register pointer.

Mnemonic:

INC A	Add a one to the register A.
INC Rr	Add a one to the specified register Rr.
INC add	Add a one to the data given by direct address.
INC @Rp	Add a one to the data of the address given by Rp.
INC DPTR	Add a one to the 16-bit DPTR register.

Increment Instruction

Example:

INC A — Add a one to the register A.

A 05 — Before execution

A 06 — After execution

Example:

INC R7 — Add a one to the content of register R7.

R7 09 — Before execution

R7 0A — After execution

Increment Instruction

Example:

INC 70 H — Add a one to the content of memory location 70 H.

70 43

Before execution

70 44

After execution

Example:

INC @R0 — Add a one to the data of the address given by R0.

INTERNAL RAM

R0 45   45 06

Before execution

INTERNAL RAM

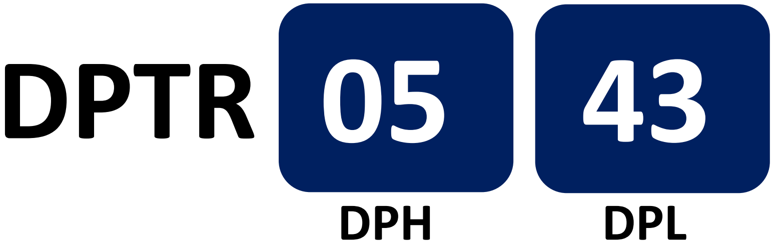
R0 45   45 07

After execution

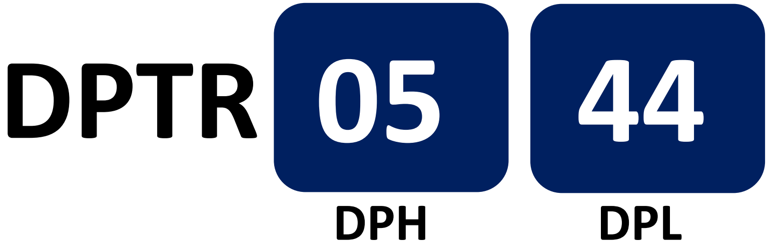
Increment Instruction

Example:

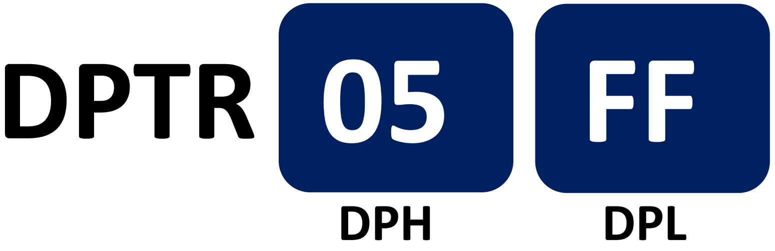
INC DPTR Add a one to the content of 16-bit DPTR register.



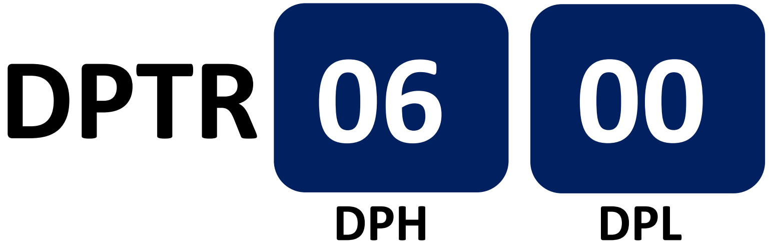
Before execution



After execution



Before execution



After execution

Decrement Instruction

- Used to decrement the data in an operand by 1.
- Operand could be register, RAM direct address, register pointer.

Mnemonic:

DCR A	Subtract a one from the register A.
DCR Rr	Subtract a one from the specified register Rr.
DCR add	Subtract a one from the data given by direct address.
DCR @Rp	Subtract a one from the data of the address given by Rp.

Decrement Instruction

Example:

DCR A Subtract a one from the register A.

A 09

Before execution

A 08

After execution

Example:

DCR R5 Subtract a one from the content of register R5.

R5 0E

Before execution

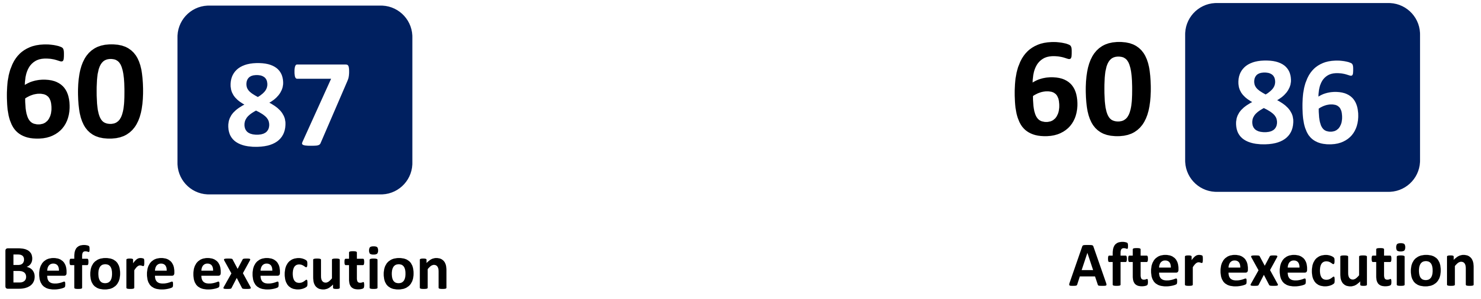
R5 0D

After execution

Decrement Instruction

Example:

DCR 60 H Subtract a 1 from the data of memory location 60 H.



60 87

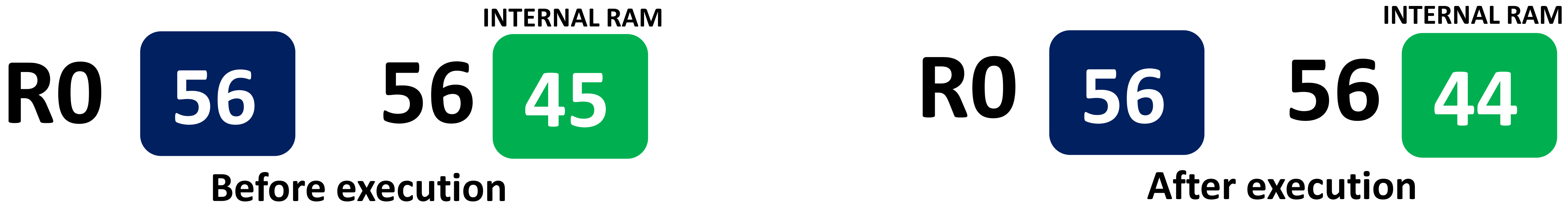
Before execution

60 86

After execution

Example:

DCR @R0 Subtract a 1 from the data of the address given by R0.



INTERNAL RAM

R0 56 56 45

Before execution

INTERNAL RAM

R0 56 56 44

After execution

Rotate and swap Instruction

- Used to rotate the accumulator right or left.
- Rotate and swap instructions are limited to Accumulator.
- Swap instruction interchanges the nibbles of accumulator.

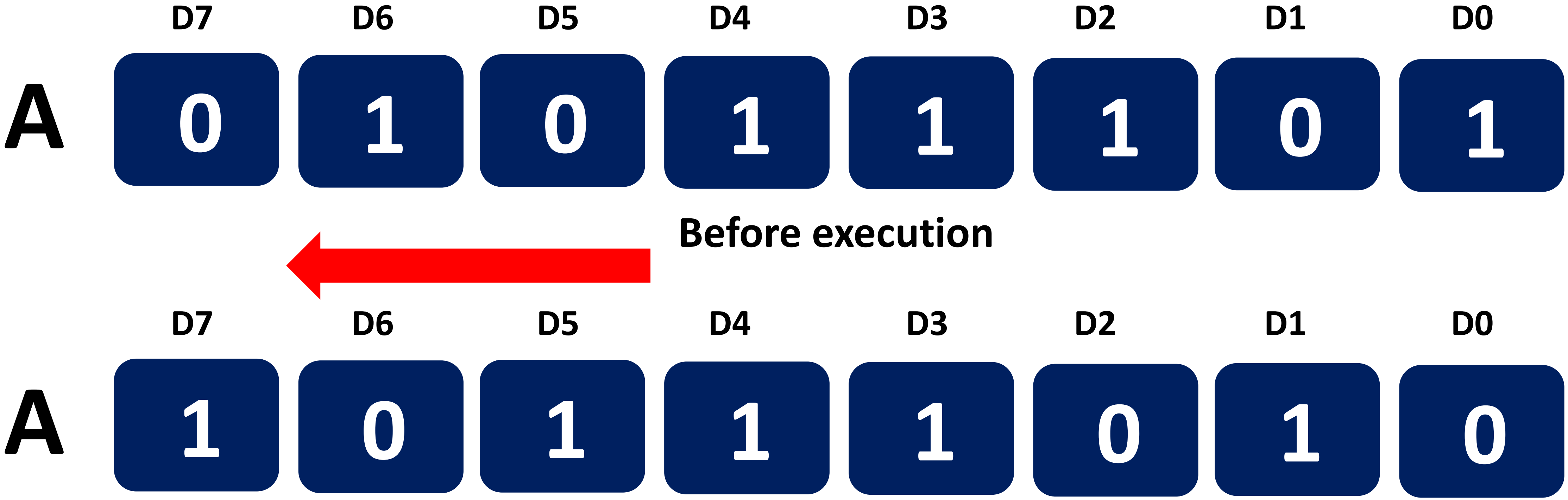
Mnemonic:

RR A	Rotate the A register one bit to the right.
RL A	Rotate the A register one bit to the left.
RLC A	Rotate the A register and carry one bit to the left.
RRC A	Rotate the A register and carry one bit to the right.
SWAP A	Interchange the nibbles of register A.

Rotate Instruction

Example:

RL A The 8 bits of the accumulator are rotated left one bit, and Bit D7 exits from the MSB and enters into LSB, D0



	D7	D6	D5	D4	D3	D2	D1	D0
A	0	1	0	1	1	1	0	1

Before execution

	D7	D6	D5	D4	D3	D2	D1	D0
A	1	0	1	1	1	0	1	0

After execution

Rotate Instruction

Example:

RR A The 8 bits of the accumulator are rotated right one bit, and Bit D0 exits from the LSB and enters into MSB, D7

	D7	D6	D5	D4	D3	D2	D1	D0
A	0	1	0	1	1	1	0	1

Before execution

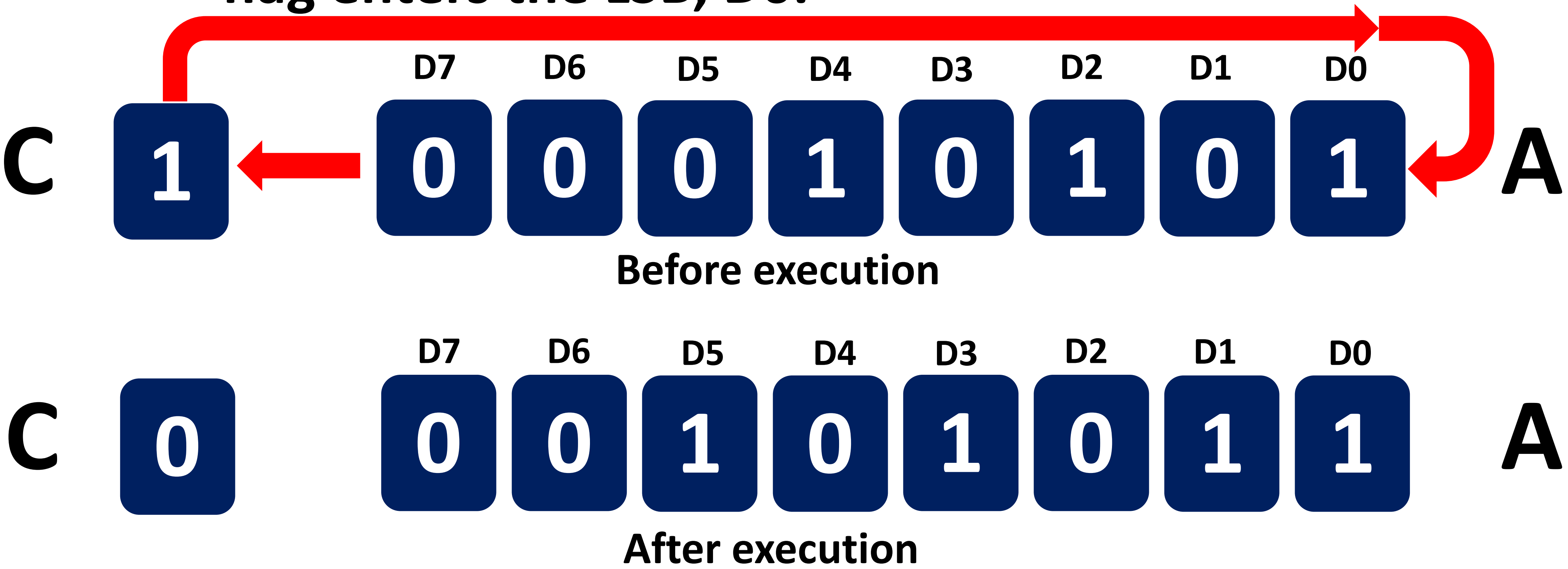
	D7	D6	D5	D4	D3	D2	D1	D0
A	1	0	1	0	1	1	1	0

After execution

Rotate Instruction

Example:

RLC A The 8 bits of the accumulator are rotated to left they exit the MSB, D7 and enter the carry flag, and the carry flag enters the LSB, D0.



		D7	D6	D5	D4	D3	D2	D1	D0	
C	1	0	0	0	1	0	1	0	1	A

Before execution

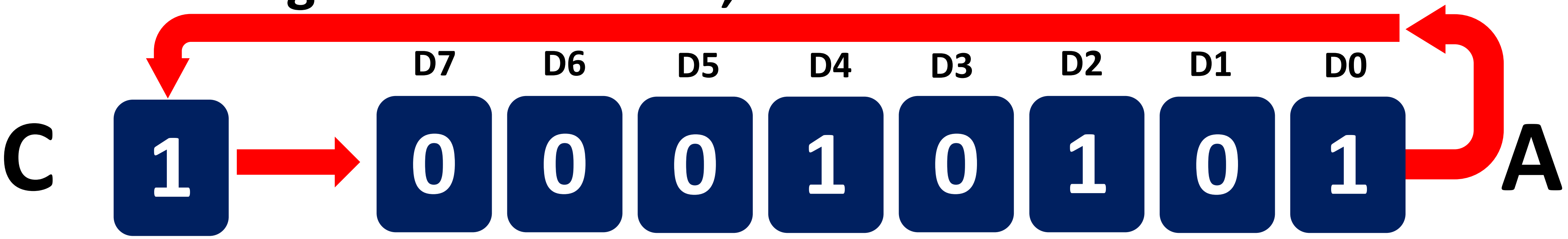
		D7	D6	D5	D4	D3	D2	D1	D0	
C	0	0	0	1	0	1	0	1	1	A

After execution

Rotate Instruction

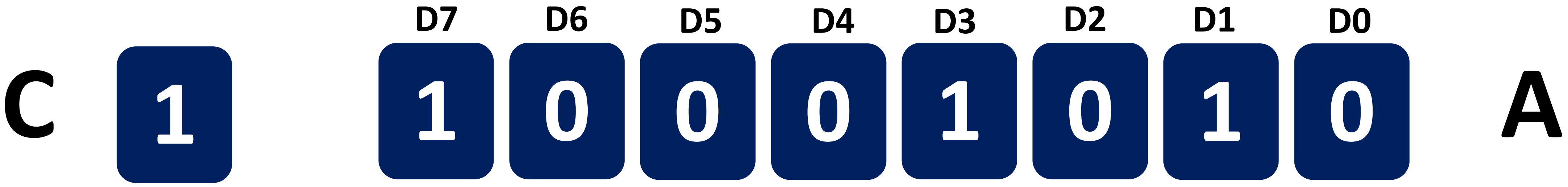
Example:

RRC A **The 8 bits of the accumulator are rotated to right they exit the LSB, D0 and enter the carry flag, and the carry flag enters the MSB, D7.**



	C	D7	D6	D5	D4	D3	D2	D1	D0	
C	1	0	0	0	1	0	1	0	1	A

Before execution



	C	D7	D6	D5	D4	D3	D2	D1	D0	
C	1	1	0	0	0	1	0	1	0	A

After execution

Swap Instruction

Example:

SWAP A It exchanges the low-order and high-order nibbles within the accumulator. No flags are affected.

	D7	D6	D5	D4	D3	D2	D1	D0
A	0	1	0	1	1	1	0	1

Before execution

	D7	D6	D5	D4	D3	D2	D1	D0
A	1	1	0	1	0	1	0	1

After execution

Multiplication Instruction

- Performs 8-bit multiplication.
- A and B both are source and destination register.
- A is multiplied by the reg B.

Mnemonic:

MUL AB	Multiply A by B. Put the low order byte of the product in A, put the high –order byte in B.

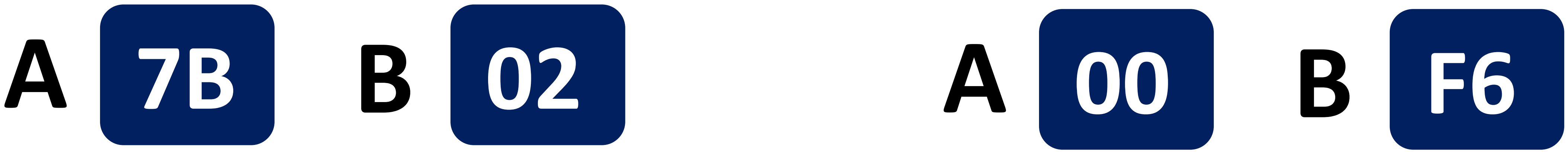
Multiplication Instruction

- The OV flag will be set if A X B > FF H.
- If OV Flag is means that number is larger than eight bits and B register has to be inspected for high order byte.
- Carry flag is always set to 0.

Example:

MUL AB

Multiply A by B.



A 7B B 02

A 00 B F6

Before execution

After execution