

In this Lecture

- Overview
- General physical & operational features
- Block diagram
- Pin assignments
- Logic symbol
- Hardware description
- Pin description
- Read-modify-write port instructions

Overview of the 8051 ✓

- Made by Intel in 1981 ✓
- An 8-bit, single-chip microcontroller optimized for control applications ✓
- ✓ 128 bytes RAM, 4096 bytes (4KB) ROM, 2 timers, 1 serial port, 4 I/O ports ✓
- 40 pins in a dual in-line package (DIP) layout ✓

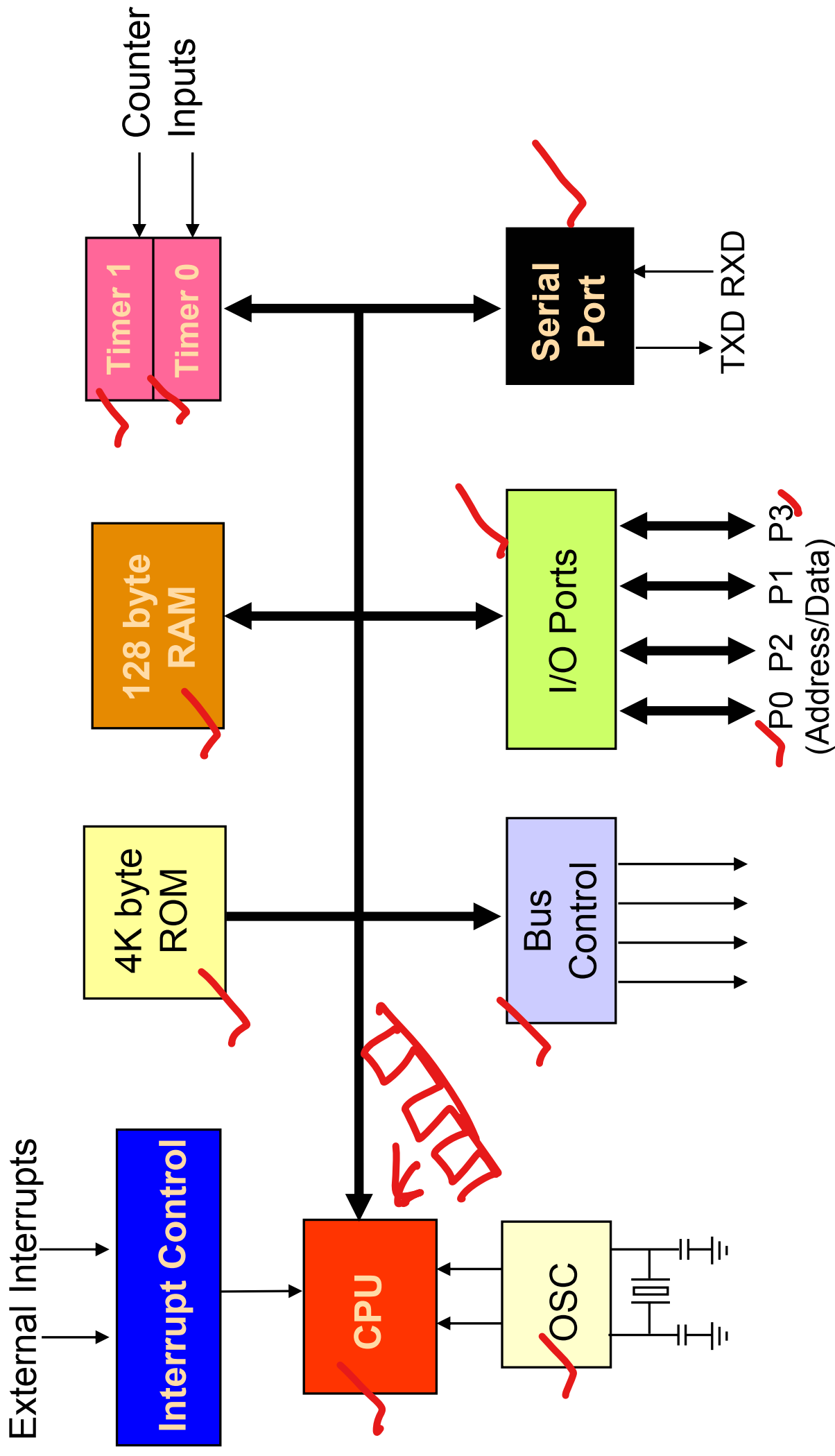
General Physical Features

- ◆ 4KB ROM
- ◆ 128 bytes internal RAM
 - ⊕ 4 register banks of 8 bytes each (R0-R7)
 - ⊕ 16 bytes of bit-addressable area
 - ⊕ 80 bytes of general purpose memory
- ◆ Four 8-bit I/O ports (P0-P3)
- ◆ Two 16-bit timers (Timer0 & Timer1)
- ◆ One serial receiver-transmitter interface
- ◆ Five interrupt sources (2 external & 3 internal)
- ◆ One oscillator (generates clock signal)

General Operational Features

- ◆ Memory of 8051 can be increased externally:
 - ⊕ Increase memory space for codes (programs) by 64K
 - ⊕ Increase memory space for data by 64K
- ◆ Boolean instructions work with 1 bit at a time
- ◆ Assume clock frequency = 12MHz, it takes about $4\text{ }\mu\text{s}$ (i.e. $4 \times 10^{-6}\text{s}$) to carry out a 8-bit multiplication instruction

The 8051 Block Diagram

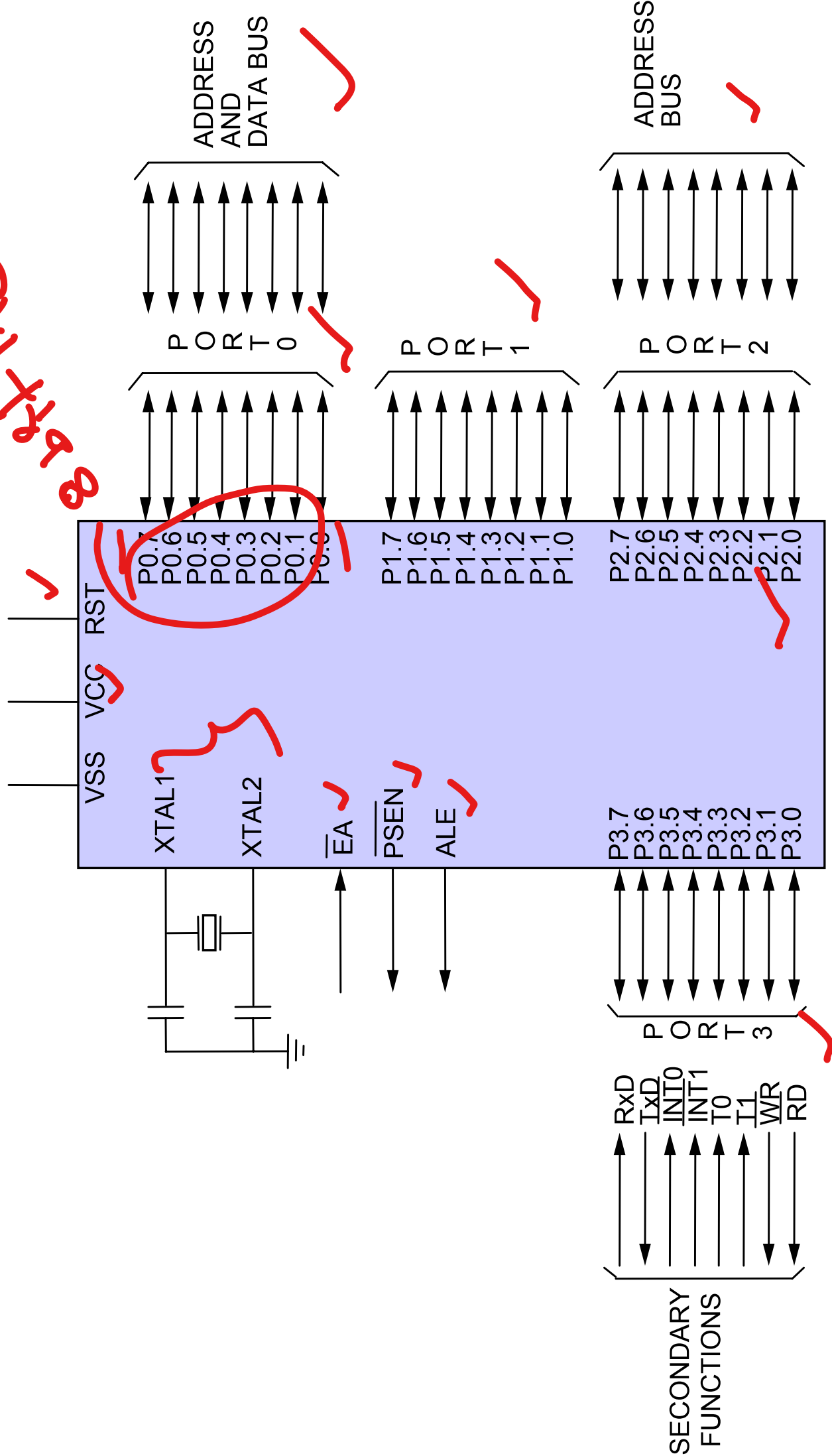


The 8051 Pin Assignments



The 8051 Logic Symbol

8051



Hardware Description ✓

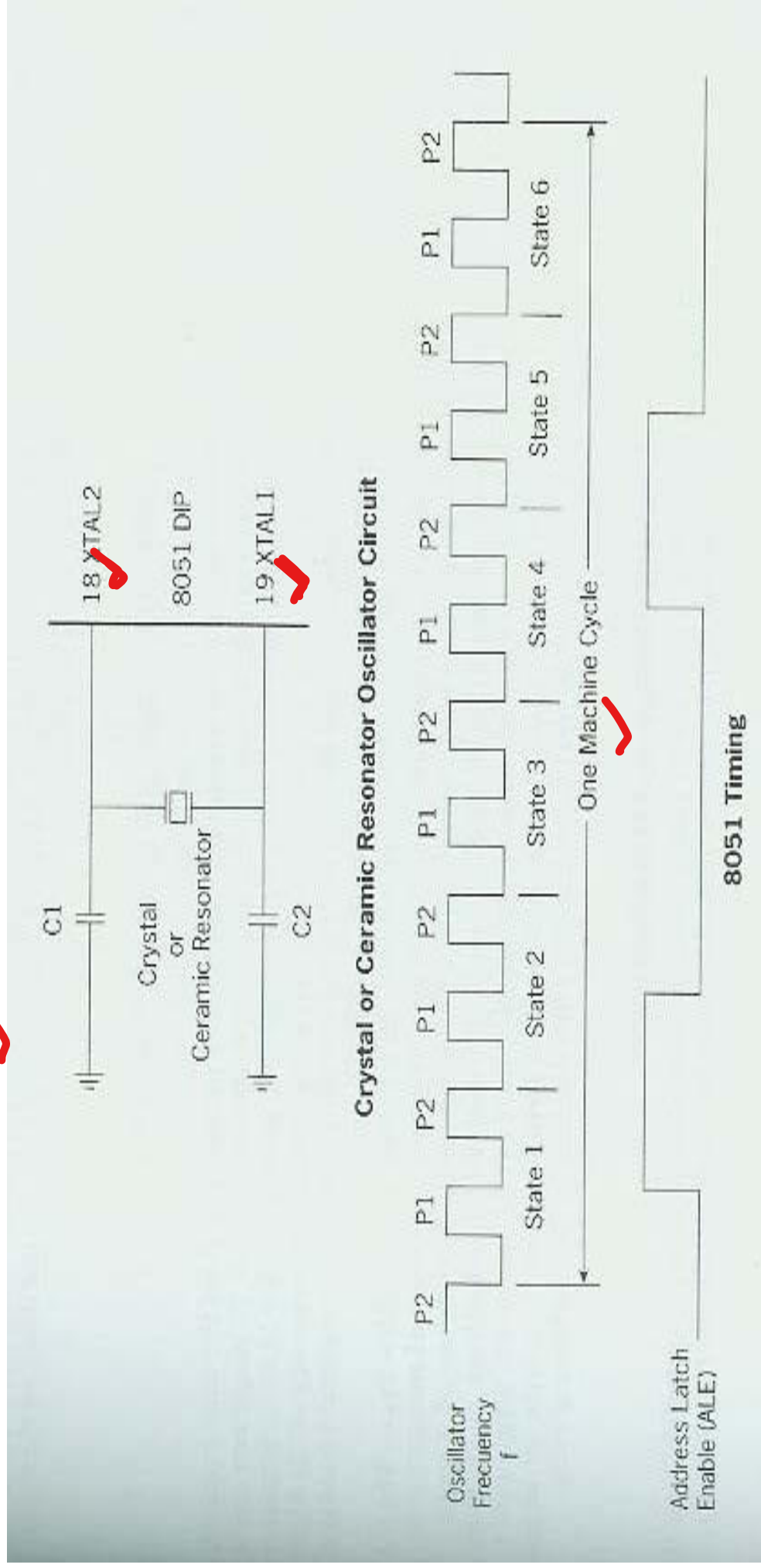
1. Oscillator circuit
2. Program counter (PC)
3. Data pointer (DPTR)
4. Accumulator ("A") register
5. B register
6. Flags
7. Program status word (PSW)
8. Internal memory (ROM, RAM, additional memory)
9. Stack & stack pointer (SP)
10. Special function register (SFR)

Handwritten notes in red:

- PC (Program Counter)
- DPTR (Data Pointer Register)
- A (Accumulator)
- B (B Register)
- PSW (Program Status Word)
- SP (Stack Pointer)
- SFR (Special Function Register)

Oscillator Circuit

- The heart of the 8051
- Produces clock pulses
- Synchronize all 8051's internal operations



Machine Cycle

- Machine cycle is the basic repetitive process that the CPU performs once it is powered on. A machine cycle consists of a fixed number of clock cycles (pulses). It is different for different kinds of CPU.
- The 8051 family needs 12 clock cycles for a machine cycle.
- The CPU takes one or more machine cycles to complete an instruction. More complex instructions require more number of machine cycles to complete the instruction. The number of machine cycles of the 8051 instructions are ranging from 1 to 4.

Example 4-1

Find the elapse time of the machine cycle for:

(a) XTAL = 11.0592 MHz

(b) XTAL = 16 MHz

(c) XTAL = 20 MHz

Solution:

(a) $11.0592 \text{ MHz} / 12 = 921.6 \text{ kHz}$

Machine cycle = $1 / 921.6 \text{ kHz} = 1.085 \mu\text{s}$

(b) $16 \text{ MHz} / 12 = 1.333 \text{ MHz}$

Machine cycle = $1 / 1.333 \text{ MHz} = 0.75 \mu\text{s}$

(c) $20 \text{ MHz} / 12 = 1.667 \text{ MHz}$

Machine cycle = $1 / 1.667 \text{ MHz} = 0.60 \mu\text{s}$

$$f = \frac{1}{T}$$

$11.0592 \text{ MHz} = 11,059,200 \text{ cycles/sec}$

Program Counter (PC)

- PC is a 16-bit register
- PC is the only register that does not have an internal address
- Holds the address of the memory location to fetch the program instruction
- Program ROM may be on the chip at addresses 0000H to 0FFFFH (4Kbytes), external to the chip for addresses that exceed 0FFFFH
- Program ROM may be totally external for all addresses from 0000H to FFFFFH
- PC is automatically incremented (+1) after every instruction byte is fetched

Data Pointer (DPTR)

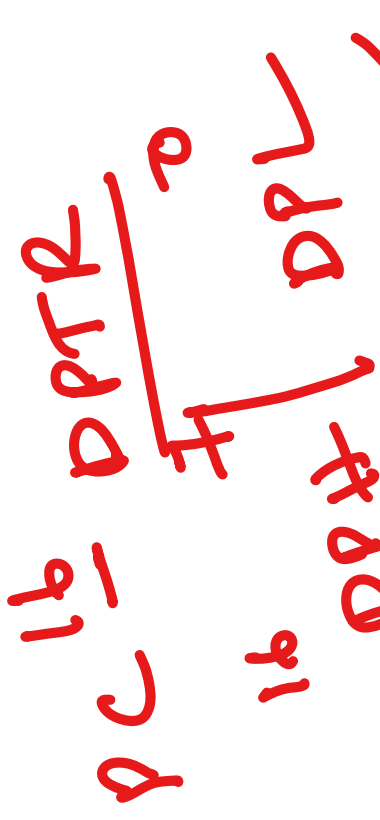
- **DPTR** is a 16-bit register ✓

- **DPTR** is made up of two 8-bit registers: **DPH** and **DPL**

- **DPTR** holds the memory addresses for internal and external code access and external data access

- **DPTR** is under the control of program instructions and can be specified by its 16-bit name, or by each individual byte name, **DPH** and **DPL**

- **DPTR** does not have a single internal address; **DPH** and **DPL** are each assigned an address (83H and 82H)



③ ④

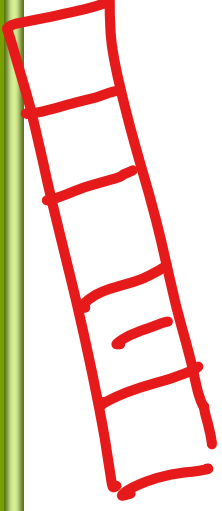
- [illegible]

B Register

- **B** register is used with the **A** register for multiplication and division operations
- No other special function other than as a location where data may be stored

Flags

Q50



ADD

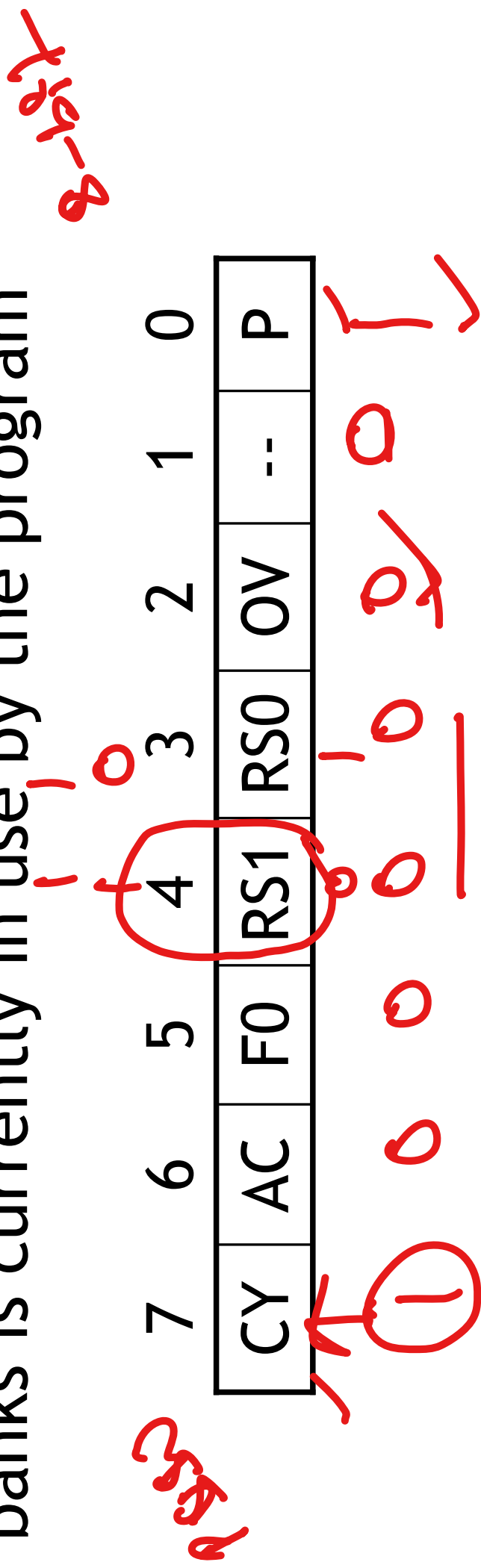
©

- Flags are **1-bit registers** provided to store the results of certain program instructions
- Other instructions can test the condition of the flags and make decisions based on the flag states
- Flags are grouped inside the **program status word (PSW)** and the **power control (PCON)** registers for convenient addressing
 - Math flags:** respond automatically to the outcomes of math operations (**CY, AC, OV, P**)
 - User flags:** general-purpose flags that may be used by the programmer to record some event in the program (**FO, GF0, GF1**)

word x 10 int

Program Status Word (PSW)

PSW contains the math flags, user program flag **F0**, and the register select bits that identify which of the four general-purpose register banks is currently in use by the program

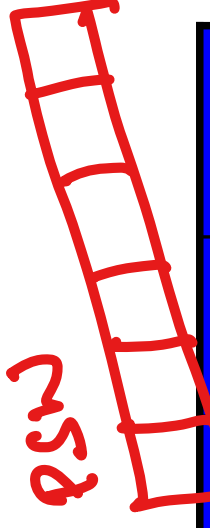


Program Status Word (PSW)

4 bits

Bit	Symbol	Function
7 ✓	CY	Carry Flag; used in arithmetic and BOOLEAN instruction
6 ✓	AC	Auxiliary carry flag; used for BCD arithmetic
5 ✓	F0	User flag 0
4	RS1	Register bank select bit 1
3	RS0	Register bank select bit 0
2	OV ✓	Overflow flag; used in arithmetic instructions
1	--	Reserved for future use
0	P ✓	Parity flag; shows parity of register A: 1 = Odd Parity

Instruction that Affect Flag Bits



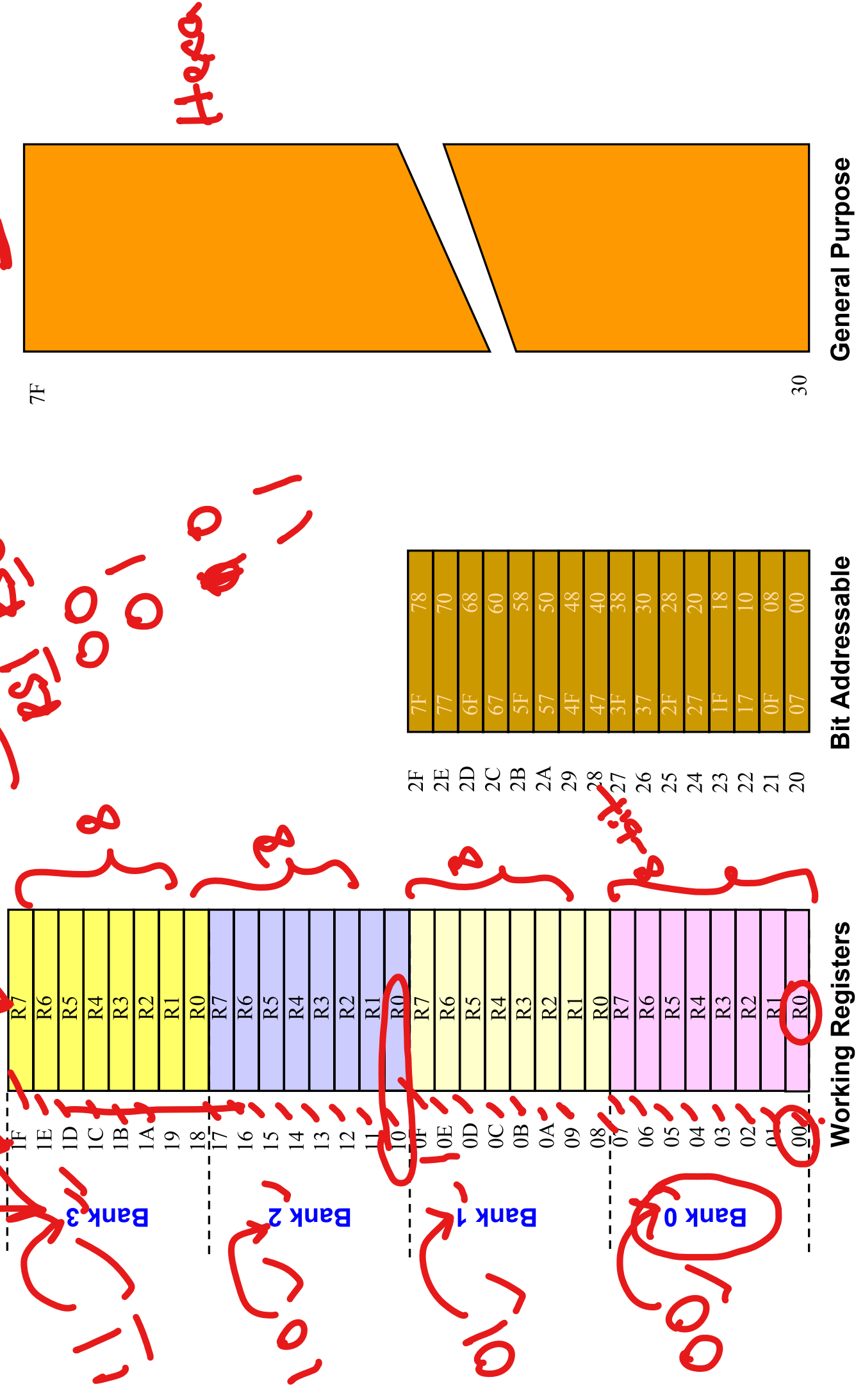
Instruction	CY	OV	AC	Instruction	CY	OV	AC
ADD	X	X	X	SETB C	1		
ADDC	X	X	X	CLR C	0		
SUBB	X	X	X	CPL C	X		
MUL	0	X		ANL C, bit	X		
DIV	0	X		ANL C, /bit	X		
DA	X			ORL C, bit	X		
RRC	X			ORL C, /bit	X		
RLC	X			CJNE	X		
MOV C, bit	X			Note: X can be 0 or 1			

Internal Memory ✓

MC 2PM
2021
The Oscillator
Clock

- A functioning computer must have memory for program code bytes, commonly in ROM, and RAM memory for variable data that can be altered as the program runs 2) 8502
- 8051 has internal RAM (128 bytes) and ROM (4Kbytes) 3) 8051 4) 8051 5)
- 8051 uses the same address but in different memories for code and data
- Internal circuitry access the correct memory based on the nature of the operation in progress
- Can add memory externally if needed

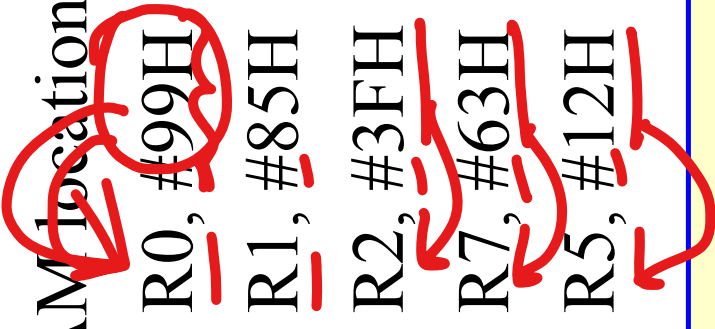
8051 Internal RAM Organisation



Example 2-5

State the contents of RAM locations after the following program:

```
✓ MOV  R0, #99H  
MOV  R1, #85H  
MOV  R2, #3FH  
MOV  R7, #63H  
MOV  R5, #12H
```



After the execution of the above program we have the following:

RAM location 0 has value 99H RAM location 1 has value 85H
RAM location 2 has value 3FH RAM location 7 has value 63H
RAM location 5 has value 12H

Example 2-6

Repeat Example 2-5 using RAM addresses instead of register names.

This is called direct addressing mode and uses the RAM address location for the destination address.

```
MOV 00, #99H  
MOV 01, #85H  
MOV 02, #3FH  
MOV 07, #63H  
MOV 05, #12H
```

Example 2-7

State the contents of RAM locations after the following program:

```

SETB  PSW.4
MOV    R0, #99H
MOV    R1, #85H
MOV    R2, #3FH
MOV    R7, #63H
MOV    R5, #12H
    
```



Bank 0

PS1 = 1

PSW = 4113

SETB PSW.4

PS0 = 1

00 R50
01 R51
10 R52
11 R53

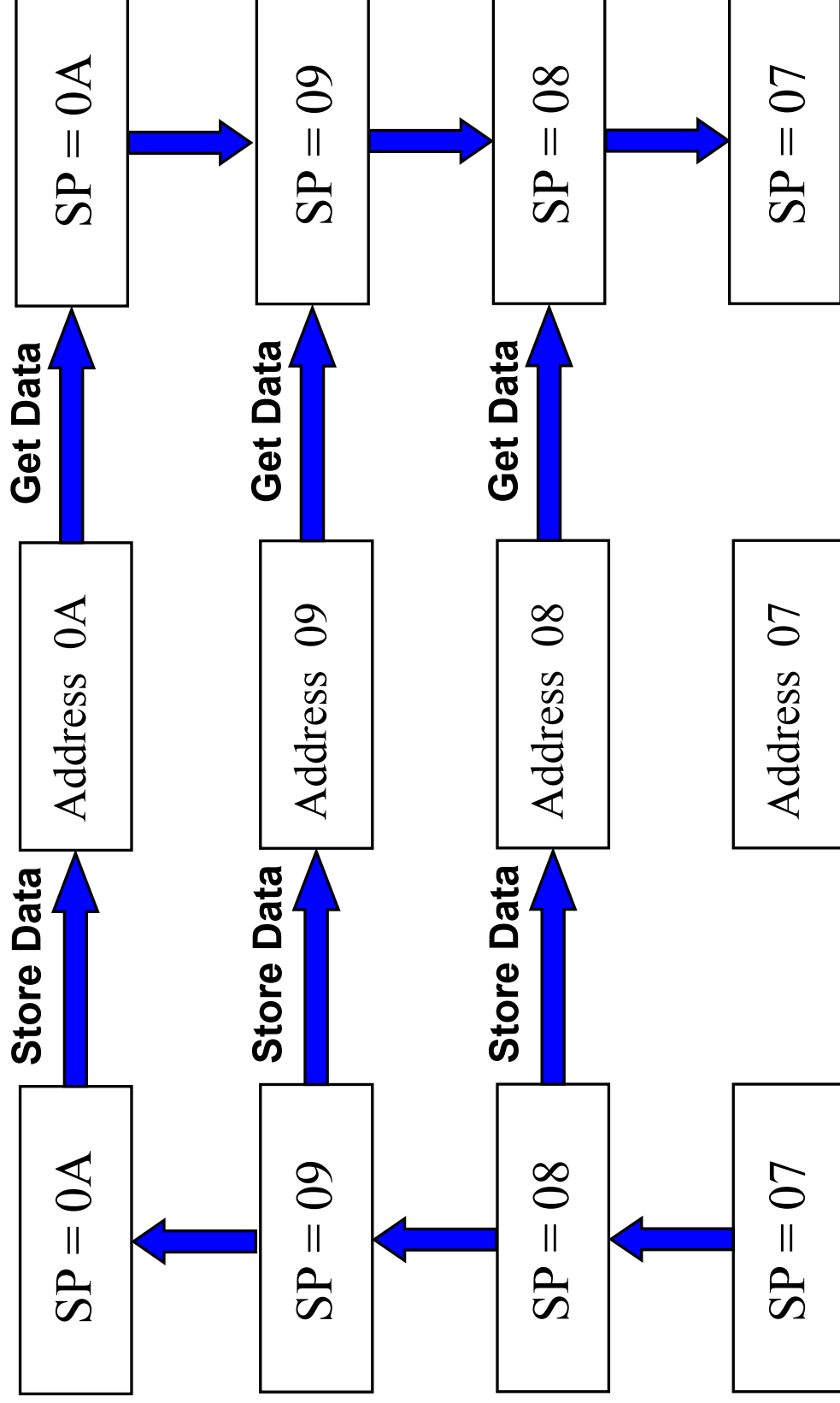
By default, PSW.3=0 and PSW.4=0; therefore, the instruction “SETB PSW.4” sets RS1=1 and RS0=0, thereby selecting register bank 2. Register bank 2 uses RAM locations 10H – 17H. After the execution of the above program we have the following

RAM location 10 has value 99H RAM location 11 has value 85H
 RAM location 12 has value 3FH RAM location 17 has value 63H
 RAM location 15 has value 12H

Stack and Stack Pointer (SP)

- **SP** is a 8-bit register used to hold an internal RAM address that is called the “**top of the stack**” ✓
- **Stack** refers to an area of internal RAM that is used in conjunction with certain opcodes to store and retrieve data quickly
- **SP** holds the internal RAM address where the last byte of data was stored by a stack operation
- When data is to be placed on the stack, the **SP** increments before storing data on the stack so that the stack **grows up** as data is stored
- As data is retrieved from the stack, the byte is read from the stack, and then the **SP** decrements to point to the next available byte of stored data

Stack Operation



Storing Data on the Stack
(Increment then store)

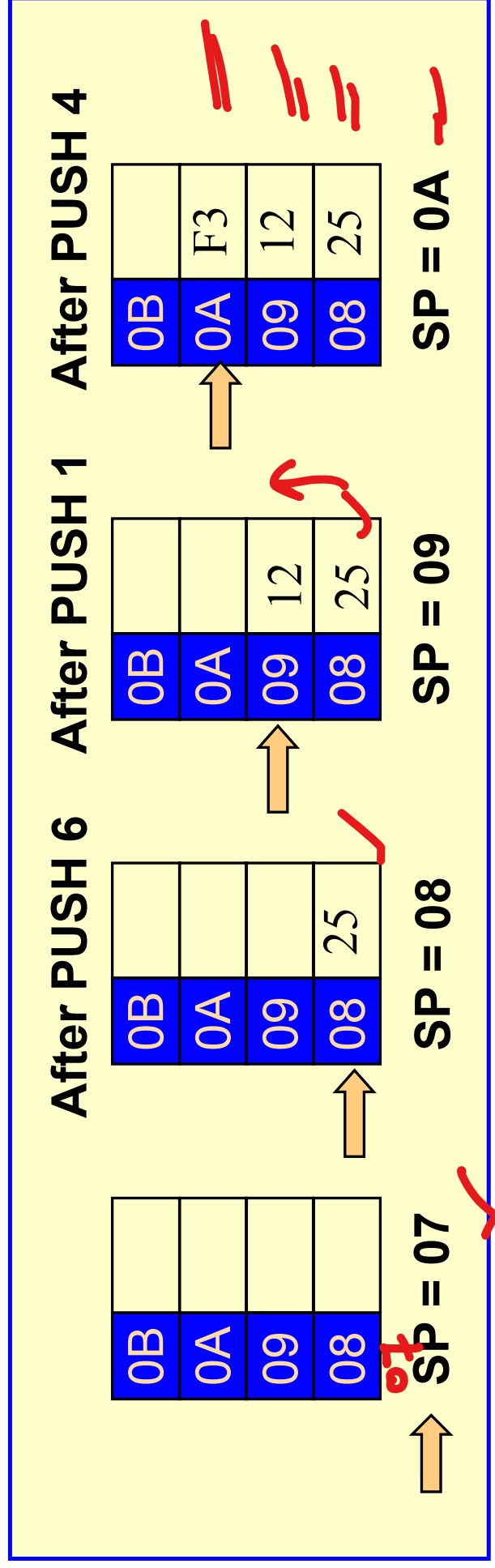
Internal RAM
(Get then decrement)

Getting Data From the Stack

Example 2-8

Show the stack and stack pointer for the following. Assume the default stack area.


MOV R6, #25H
MOV R1, #12H
MOV R4, #0F3H
~~PUSH 6~~
~~PUSH 1~~
~~PUSH 4~~



Example 2-9

Examine the stack, show the contents of the registers and **SP** after execution of the following instruction. All values are in hex.


POP 3 ;POP stack into R3
POP 5 ;POP stack into R5
POP 2 ;POP stack into R2



0B	54
0A	F9
09	76
08	6C


Start SP = 0B

After POP 3 After POP 5 After POP 2




0B	54
0A	F9
09	76
08	6C

**SP = 0A
R3 = 54**



0B	54
0A	F9
09	76
08	6C

**SP = 09
R5 = F9**



0B	54
0A	F9
09	76
08	6C

**SP = 08
R2 = 76**

Example 2-10

Show the stack and stack pointer for the following.

```
MOV SP, #5FH
MOV R2, #25H
MOV R1, #12H
MOV R4, #0F3H
PUSH 2
PUSH 1
PUSH 4
```

After PUSH 2 After PUSH 1 After PUSH 4

63	
62	
61	
60	

Start SP = 5F

63	
62	
61	
60	25

SP = 60

63	
62	
61	12
60	25

SP = 61

63	
62	F3
61	12
60	25

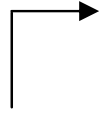
SP = 62

Special Function Registers (SFR)

- 8051 has 21 *SFRs* which occupy the addresses from 80H to FFH (128bytes)
- Not all of the addresses from 80H to FFH are used for *SFRs*
- Attempt to use the “empty” addresses may get unpredictable result

Special Function Register Map

Bit addressable

[illegible]

4F20

Internal ROM

128

1) OSC
2) PC

- Internal ROM occupies the code address space from 0000H to 0FFFFH

3)

58

4)

53

- Program addresses higher than 0FFFFH will automatically fetch code bytes from external program memory

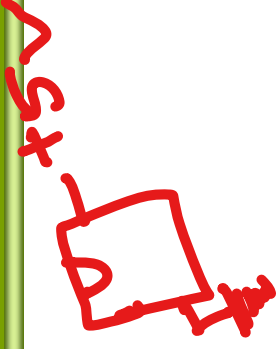
5) External

6)

20H

- Code bytes can also be fetched exclusively from an external memory by connecting the external access pin (**EA**) to ground

Some Important Pins



- ✓ VCC (pin 40 - provides supply voltage of +5V) ✓
- GND (pin 20) ✓
- ✓ XTAL1 & XTAL2 (pins 19 & 18 - to crystal and then caps)
- ✓ RST (pin 9- reset)
- ✓ EA (pin 31 - external access)
- ✓ PSEN (pin 29 - program store enable)
- ✓ ALE (pin 30 - address latch enable)
- ✓ Ports 0-3

I/O Ports (P0 - P3)

One of the most useful features of the 8051 is that it consists of 4 I/O ports (P0 - P3)

- All ports are bidirectional (they can take input and to provide output)
- All ports have multiple functions (except P1)
- All ports are bit addressable
- On **RESET** all the ports are configured as *output*
- When a bit latch is to be used as an *input*, a “1” *must be* written to the corresponding latch by the program to configure it as input

Port 0

- Occupies a total of 8 pins (Pins 32-39)
- Can be used for :
 - ✦ Input only ✓
 - ✦ Output only ✓
 - ✦ Input and output at the same time (i.e. some pins for input and the others for output)
- Can be used to handle both address and data
- Need pull-up resistors

Port 0 as an Output Port

The following code will continuously send out to port 0 the alternating values 55H and AAH

```
BACK:  MOV     A, #55H
        MOV     P0, A
        ACALL   DELAY
        CPL     A
        SJMP    BACK
```

Port 0 as an Input Port

In the following code, port 0 is configured first as an input port by writing 1s to it, and then data is received from that port and sent to P1

```
BACK:  MOV    A, #0FFH
        MOV    P0, A
        MOV    A, P0
        MOV    P1, A
        SJMP   BACK
```

Dual Role of Port 0

- When connecting an 8051 to an external memory, port 0 provides both address and data (AD0 – AD7)
- When ALE = 0, it provides data D0 – D7
- When ALE = 1, it provides data A0 – A7
- ALE is used for demultiplexing address and data with the help of a 74LS373 latch

Port 1

- Occupies a total of 8 pins (Pins 1-8)
- Can be used as input or output
- Does not need any pull-up resistors
- Upon reset, port 1 is configured as an output port
- No alternative functions

Port 1 as an Output Port

The following code will continuously send out to port 1 the alternating values 55H and AAH

```
BACK:    MOV     A, #55H
          MOV     P1, A
          ACALL   DELAY
          CPL     A
          SJMP    BACK
```


Port 1 as an Input Port

In the following code, port 1 is configured first as an input port by writing 1s to it, and then data is received from that port and saved in R7, R6, and R5

MOV	A, #0FFH
MOV	P1, A
MOV	A, P1
MOV	R7, A
ACALL	DELAY
MOV	A, P1
MOV	R6, A
ACALL	DELAY
MOV	A, P1
MOV	R5, A

Port 2

- Occupies a total of 8 pins (Pins 21-28)
- Similar function as Port 1
- Can be used as input or output
- Does not need any pull-up resistors
- Upon reset, port 1 is configured as an output port

Port 2 as an Output Port

The following code will continuously send out to port 2 the alternating values 55H and AAH

```
BACK:      MOV     A, #55H
           MOV     P2, A
           ACALL   DELAY
           CPL     A
           SJMP    BACK
```

Port 2 as an Input Port

In the following code, port 2 is configured first as an input port by writing 1s to it, and then data is received from that port and sent to P1

```
MOV    A, #0FFH
MOV    P2, A
MOV    A, P2
MOV    P1, A
SJMP   BACK
```

Dual Role of Port 2

- When connecting an 8051 to an external memory, port 2 provides both address (A8 – A15)
- It is used along with P0 to provide the 16-bit address
- When P2 is used for the upper 8 bits of the 16-bit address, it cannot be used for I/O

Port 3

- Occupies a total of 8 pins (Pins 10-17)
- Similar function as Port 1 and Port 2
- Can be used as input or output
- Does not need any pull-up resistors
- Upon reset, port 1 is configured as an output port
- Pins can be individually programmable for other uses
- Most commonly be used to provide some important signals (e.g. interrupts)

Port 3 Alternate Functions

P3 Bit	Function	Pin
P3.0	RxD	10
P3.1	TxD	11
P3.2	$\overline{\text{INT0}}$	12
P3.3	$\overline{\text{INT1}}$	13
P3.4	T0	14
P3.5	T1	15
P3.6	$\overline{\text{WR}}$	16
P3.7	$\overline{\text{RD}}$	17

Read-Modify-Write Feature

- A method used to access the 8051 ports
- Combining all 3 actions in a single instructions :
 - ✦ Read the data at the port
 - ✦ Modify (do operation on) the data at the port
 - ✦ Write the results to the port

AGAIN:	MOV	P1, #55H
	XRL	P1, #0FFH
	ACALL	DELAY
	SJMP	AGAIN

Single-bit Addressability of Ports

- One of the most powerful features of the 8051
- Access only one or several bits of the port instead of the entire 8 bits

BACK:	CPL	P1.2
	ACALL	DELAY
	SJMP	BACK

Example 4-2

Write a program to perform the following:

- (a) Keep monitoring the P1.2 bit until it becomes high;
- (b) When P1.2 becomes high, write value 45H to port 0; and
- (c) Send a high-to-low (H-to-L) pulse to P2.3

SETB	P1.2
MOV	A, #45H
JNB	P1.2, AGAIN
MOV	P0,A
SETB	P2.3
CLR	P2.3

AGAIN:

Summary

- General physical & operational features
- 8051 hardware description
- 8051 pin description
- Read-modify-write port instructions

- 8051 hardware description
- 8051 pin description
- Read-modify-write port instructions