KTU NOTES

The learning companion.

KTU STUDY MATERIALS | SYLLABUS | LIVE NOTIFICATIONS | SOLVED QUESTION PAPERS

- To make a complete microcomputer system, only a microprocessor is not sufficient. To make a complete microcomputer system, adding other peripherals such as ROM, RAM, decoders, drivers, and a number of I/O devices is necessary. In addition, special-purpose devices, such as interrupt controllers, programmable timers, programmable I/O devices, DMA controllers may be added to improve the capability and performance, and flexibility of a microcomputer system.
- On the other hand, the microcontroller has built-in ROM, RAM, parallel I/O, serial I/O, counters, and a clock circuit. It has on-chip peripheral devices which makes it possible to have a single microcomputer system.
- A microcontroller does not require any additional interfacing ICs for operation and it functions as a standalone system. The operation of a microcontroller is multipurpose.

Features of 8051

1. 4KB on-chip program memory (ROM/EPROM).
2. 128 bytes on-chip data memory.
3. Four register banks.
4. 64KB for each external ROM and external RAM addressability.
5. One-microsecond instruction cycle with 12MHz crystal.
6. 32 bidirectional I/O lines organized as four 8-bit ports.
7. Multiple modes, high-speed programmable serial port (UART).
8. 16-bit Timers/Counters.
9. Direct byte and bit addressability.

ARCHITECTURE OF 8051

Accumulator: The Accumulator, as its name suggests, is used as a general register to accumulate the results of a large number of instructions. It can hold an 8-bit (1-byte) value.

'B' Register: The "B" register is very similar to the Accumulator in the sense that it may hold an 8-bit (1- byte) value. The "B" register is only used by two 8051 instructions: MUL AB and DIV AB. Aside from the MUL and DIV instructions, the "B" register is often used as yet another temporary storage register.

Program Status Word: The PSW register contains program status information. It is an 8-bit flag register, out of 8-bits 6 bits are used and 2 bits are reserved. Out of 6 bits 4 bits are conditional bits and 2 bits are used for selecting the register bank.

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
CY	AC	F0	RS_1	RS_0	OV	—	P

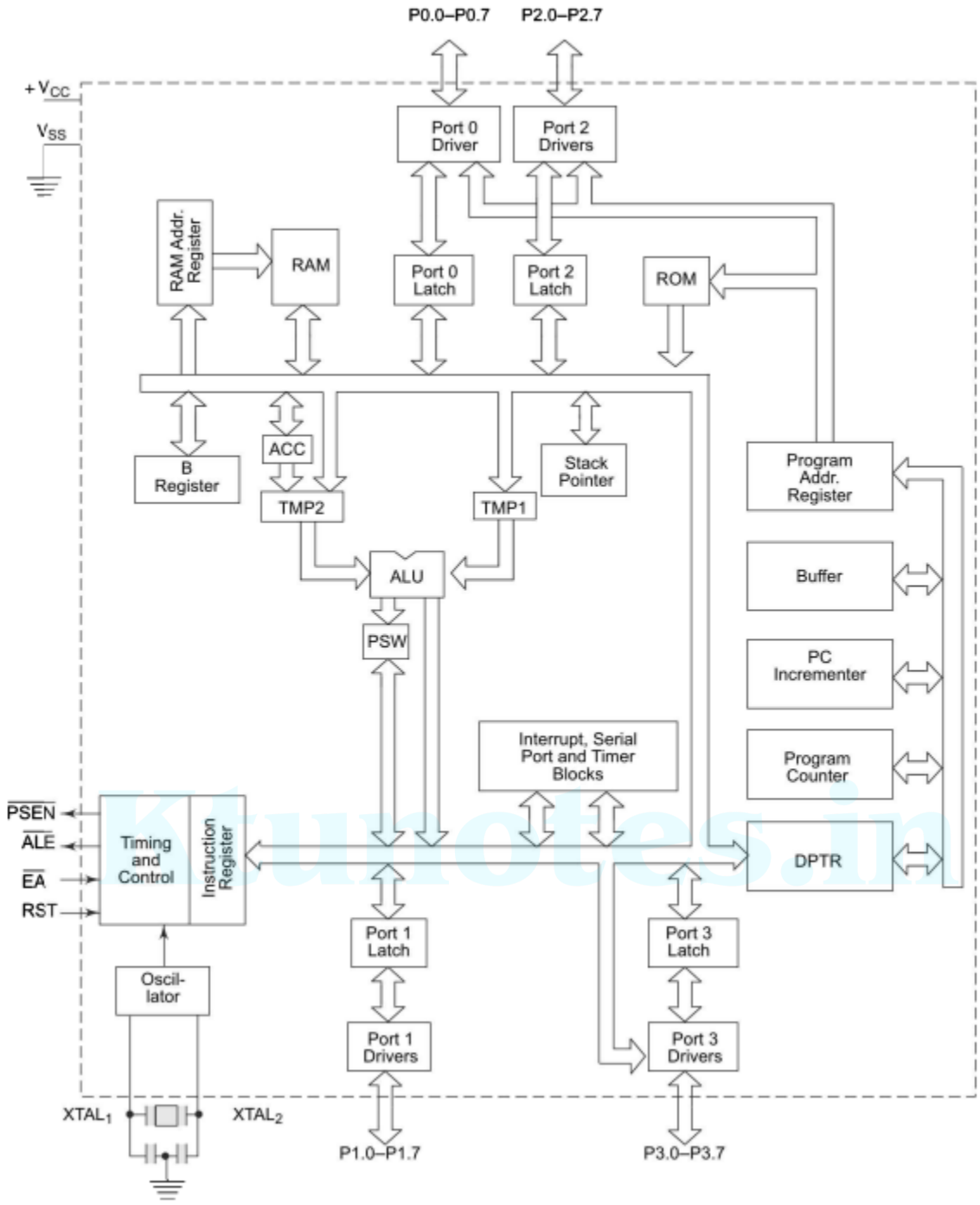
CY	D_7	Carry Flag.
AC	D_6	Auxiliary carry Flag.
F0	D_5	Flag 0 is available to the user for general purpose.
RS_1	D_4	Register Bank selector bit 1.
RS_0	D_3	Register Bank selector bit 0.

The value presented by RS_0 and RS_1 bits select the corresponding register bank as shown below.

RS_1	RS_0	Register Bank	Address
0	0	0	00H-07H
0	1	1	08H-0FH
1	0	2	10H-17H
1	1	3	18H-1FH

OV	D_2	Overflow Flag.
—	D_1	User definable flags (Reserved for future use)
P	D_0	Parity flag is set/cleared by hardware in each instruction cycle to indicate an odd/even number of '1' bits in the accumulator

Format of PSW



Stack Pointer: The Stack Pointer register is 8 bits wide. It is incremented before data is stored during PUSH and CALL executions. While the stack may reside anywhere in on-chip RAM, the Stack Pointer is initialized to 07H after a reset. This causes the stack to begin at location 08H.

Data Pointer: The Data Pointer (DPTR) consists of a high byte (DPH) and a low byte (DPL). Its intended function is to hold a 16-bit address. It may be manipulated as a 16-bit register or as two independent 8-bit registers.

Program Counter: The Program Counter (PC) is a 2-byte address (16 - bit) that tells the 8051 where the next instruction to execute is found in memory. When the 8051 is initialized PC always starts at 0000h and is incremented each time an instruction is executed.

Ports 0 to 3: P0, P1, P2, and P3 are the SFR latches of Ports 0, 1, 2, and 3, respectively. Writing a one to a bit of a port SFR (P0, P1, P2, or P3) causes the corresponding port output pin to switch high. Writing a zero causes the port output pin to switch low.

Serial Data Buffer: The Serial Buffer is actually two separate registers, a transmit buffer and a receive buffer. When data is moved to SBUF, it goes to the transmit buffer and is held for serial transmission. For serial reception, the data is received in receive buffer.

Timer Registers: Register pairs (TH0, TL0), and (TH1, TL1) are the 16-bit Counting registers for Timer/Counters 0 and 1, respectively.

Control Registers: Special Function Registers IP (Interrupt priority), IE (Interrupt enable), TMOD (Timer Mode), TCON (Timer Control), SCON (serial Control) and PCON (power Control) contain control and status bits for the interrupt system, the Timer/Counters, and the serial port.

Register Banks: The 8051 uses 8 "R" registers which are used in many of its instructions. These "R" registers are numbered from 0 through 7 (R0, R1, R2, R3, R4, R5, R6, and R7). These registers are generally used to assist in manipulating values and moving data from one memory location to another.

Instruction register: This register decodes the opcode of an instruction to be executed and gives information to the timing and control unit to generate necessary control signals for the execution of the instruction.

Timing and Control Unit: This unit derives all timing and control signals required for the internal operation of the chip. It also derives the control signals required for controlling the external system bus

PSEN (Program Store Enable): It is a control signal that enables external program (code) memory. It usually connects to an EPROM's Output Enable (OE) pin to permit the reading of program bytes.

ALE (Address Latch Enable): The 8051 uses ALE for demultiplexing the address and data bus. ALE enables address lines.

EA (External Access): The EA input signal is generally tied high (+5 V) or low (ground). If high, the 8051 executes programs from internal ROM. If low, programs execute from external memory only (and PSEN pulses low accordingly).

RST (Reset): The RST input is the master reset for the 8051. When this signal is brought high for at least two machine cycles, the 8051 internal registers are loaded with appropriate values for an orderly system start-up.

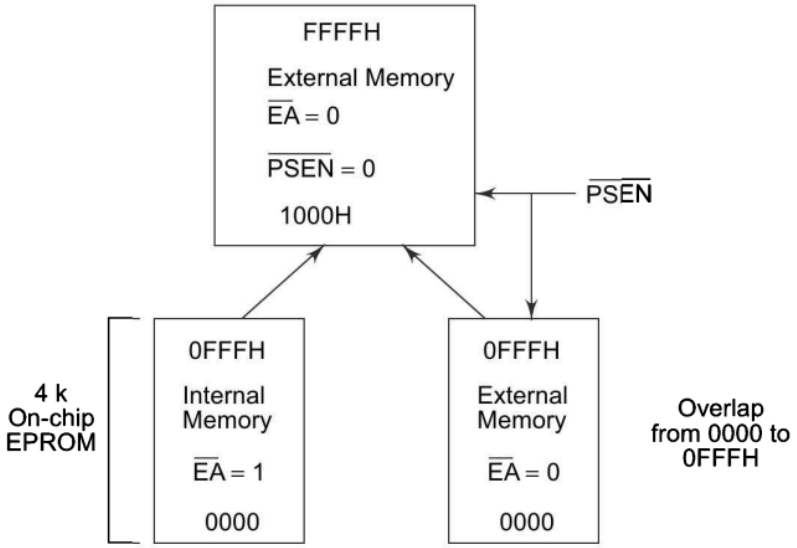
On-chip Oscillator Inputs: The 8051 features an on-chip oscillator. The nominal crystal frequency is 12 MHz.

MEMORY ORGANIZATION OF 8051

- The 8051 has two types of memory and these are Program Memory and Data Memory. Program Memory (ROM) is used to permanently save the program being executed, while Data Memory (RAM) is used for temporarily storing data and intermediate results created and used during the operation of the microcontroller.
- All 8051 microcontrollers have a 16-bit addressing bus and are capable of addressing 64 kb memory.

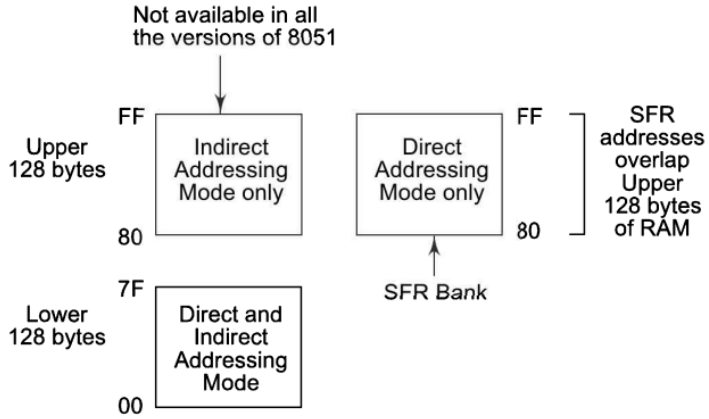
1. **PROGRAM MEMORY:**
    - Program Memory (ROM) is used for the permanent saving of the program (CODE) being executed. The memory is read-only. It is further divided into internal (on-chip) and external ROM.

- 8051 can address 4 kb of on-chip program memory from 0000H to 0FFFH.
- It can address 64 kb of external program memory from 0000H to FFFFH.
- Address mapping from 0000H to 0FFFH overlaps for internal and external memory. They are distinguished using PSEN and EA.



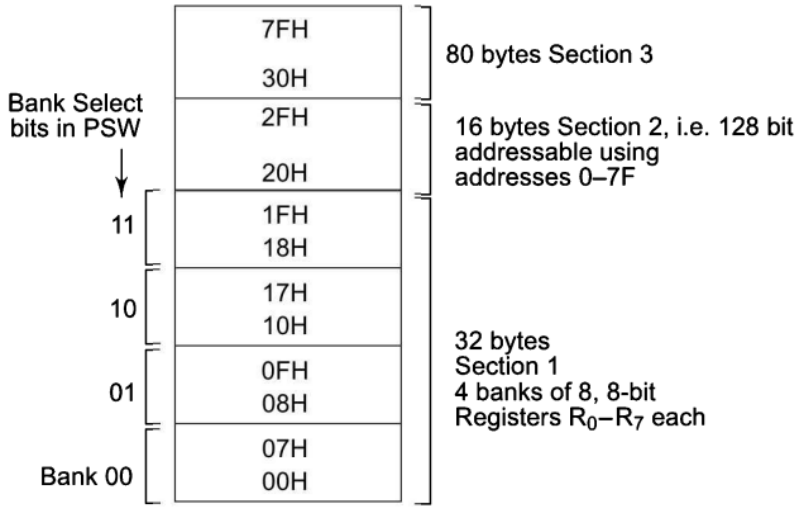
2. DATA MEMORY:

- 8051 supports 64 kb of external data memory from 0000H to FFFFH. This memory can be accessed under the control of the register DPTR.
- RD', WR', CS' signals are used while accessing external data memory.
- Internal data memory consists of two parts.
    - Lower 128 bytes from 00H to 7FH
    - Upper 128 bytes from 80H to FFH



Internal Data Memory of 8051

- **Lower 128 bytes from 00H to 7FH:**
    - Direct and indirect addressing modes possible
    - This memory space is divided into
        1. register banks (00H-1FH)
        2. bit-addressable RAM (20H-2FH)
        3. general-purpose RAM (scratch-pad) (30H-7FH).



- Default stack is register bank 1.
- Default register bank is register bank 0.
- RS1 and RS0 bits of PSW are used to select register banks.
- Bit-addressable memory space has the property of individually accessing bits through software. Bits can be set, cleared, ANDed, ORed, etc., with a single instruction.

- **Upper 128 bytes from 00H to 7FH:**
  - This address space overlaps with the SFR address space.
  - This address space can be accessed using indirect addressing modes only.
  - SFRs can be accessed using only direct addressing mode.

8051 - INSTRUCTION SET

8051 Microcontroller has a set of instructions to perform different operations. There are five groups of instructions which are listed below.

1. Arithmetic Instructions
2. Logic Instructions
3. Data Transfer Instructions
4. Data transfer Instructions (Branch instructions)
5. Boolean Instructions (Bit-oriented instructions)

1. ARITHMETIC INSTRUCTION: Arithmetic instructions perform several basic operations such as addition, subtraction, division, multiplication, etc. After execution, the result is stored in the first operand.

ADD	A,Rn	Add register to accumulator
ADD	A,direct	Add direct byte to accumulator
ADD	A, @Ri	Add indirect RAM to accumulator
ADD	A,#data	Add immediate data to accumulator
ADDC	A,Rn	Add register to accumulator with carry flag
ADDC	A,direct	Add direct byte to A with carry flag
ADDC	A, @Ri	Add indirect RAM to A with carry flag
ADDC	A, #data	Add immediate data to A with carry flag
SUBB	A,Rn	Subtract register from A with borrow
SUBB	A,direct	Subtract direct byte from A with borrow
SUBB	A,@Ri	Subtract indirect RAM from A with borrow
SUBB	A,#data	Subtract immediate data from A with borrow
INC	A	Increment accumulator
INC	Rn	Increment register
INC	direct	Increment direct byte
INC	@Ri	Increment indirect RAM
DEC	A	Decrement accumulator
DEC	Rn	Decrement register
DEC	direct	Decrement direct byte
DEC	@Ri	Decrement indirect RAM
INC	DPTR	Increment data pointer
MUL	AB	Multiply A and B
DIV	AB	Divide A by B
DA	A	Decimal adjust accumulator

2. LOGICAL INSTRUCTION: Logic instructions perform logic operations upon corresponding bits of two registers. After execution, the result is stored in the first operand.

ANL	A,Rn	AND register to accumulator
ANL	A,direct	AND direct byte to accumulator
ANL	A,@Ri	AND indirect RAM to accumulator
ANL	A,#data	AND immediate data to accumulator
ANL	direct,A	AND accumulator to direct byte
ANL	direct,#data	AND immediate data to direct byte
ORL	A,Rn	OR register to accumulator
ORL	A,direct	OR direct byte to accumulator
ORL	A,@Ri	OR indirect RAM to accumulator
ORL	A,#data	OR immediate data to accumulator
ORL	direct,A	OR accumulator to direct byte
ORL	direct,#data	OR immediate data to direct byte
XRL	A,Rn	Exclusive OR register to accumulator
XRL	A direct	Exclusive OR direct byte to accumulator
XRL	A,@Ri	Exclusive OR indirect RAM to accumulator
XRL	A,#data	Exclusive OR immediate data to accumulator
XRL	direct,A	Exclusive OR accumulator to direct byte
XRL	direct,#data	Exclusive OR immediate data to direct byte
CLR	A	Clear accumulator
CPL	A	Complement accumulator
RL	A	Rotate accumulator left
RLC	A	Rotate accumulator left through carry
RR	A	Rotate accumulator right
RRC	A	Rotate accumulator right through carry
SWAP	A	Swap nibbles within the accumulator

3. DATA TRANSFER INSTRUCTION: These instructions are used to copy the content of the source operand to the destination operand.

MOV	A,Rn	Move register to accumulator
MOV	A,direct *)	Move direct byte to accumulator
MOV	A,@Ri	Move indirect RAM to accumulator
MOV	A,#data	Move immediate data to accumulator
MOV	Rn,A	Move accumulator to register
MOV	Rn,direct	Move direct byte to register
MOV	Rn,#data	Move immediate data to register

MOV	direct,A	Move accumulator to direct byte
MOV	direct,Rn	Move register to direct byte
MOV	direct,direct	Move direct byte to direct byte
MOV	direct,@Ri	Move indirect RAM to direct byte
MOV	direct,#data	Move immediate data to direct byte
MOV	@Ri,A	Move accumulator to indirect RAM
MOV	@Ri,direct	Move direct byte to indirect RAM
MOV	@Ri, #data	Move immediate data to indirect RAM
MOV	DPTR, #data16	Load data pointer with a 16-bit constant
MOVC	A,@A + DPTR	Move code byte relative to DPTR to accumulator
MOVC	A,@A + PC	Move code byte relative to PC to accumulator
MOVX	A,@Ri	Move external RAM (8-bit addr.) to A
MOVX	A,@DPTR	Move external RAM (16-bit addr.) to A
MOVX	@Ri,A	Move A to external RAM (8-bit addr.)
MOVX	@DPTR,A	Move A to external RAM (16-bit addr.)
PUSH	direct	Push direct byte onto stack
POP	direct	Pop direct byte from stack
XCH	A,Rn	Exchange register with accumulator
XCH	A,direct	Exchange direct byte with accumulator
XCH	A,@Ri	Exchange indirect RAM with accumulator
XCHD	A,@Ri	Exchange low-order nibble indir. RAM with A

4. BOOLEAN INSTRUCTIONS: Similar to logic instructions, bit-oriented instructions perform logic operations. The difference is that these are performed upon single bits.

CLR	C	Clear carry flag
CLR	bit	Clear direct bit
SETB	C	Set carry flag
SETB	bit	Set direct bit
CPL	C	Complement carry flag
CPL	bit	Complement direct bit
ANL	C,bit	AND direct bit to carry flag
ANL	C,/bit	AND complement of direct bit to carry
ORL	C,bit	OR direct bit to carry flag
ORL	C,/bit	OR complement of direct bit to carry
MOV	C,bit	Move direct bit to carry flag
MOV	bit,C	Move carry flag to direct bit

5. BRANCH INSTRUCTIONS: There are two types of branch instructions:

1. **Unconditional jump instructions:** upon their execution, a jump to a new location from where the program continues execution is executed.

ACALL	addr11	Absolute subroutine call
LCALL	addr16	Long subroutine call
RET		Return from subroutine
RETI		Return from interrupt
AJMP	addr11	Absolute jump
LJMP	addr16	Long iump
SJMP	rel	Short jump (relative addr.)
JMP	@A + DPTR	Jump indirect relative to the DPTR

2. **Conditional jump instructions:** a jump to a new program location is executed only if a specified condition is met. Otherwise, the program proceeds with the next instruction.

JZ	rel	Jump if accumulator is zero
JNZ	rel	Jump if accumulator is not zero
JC	rel	Jump if carry flag is set
JNC	rel	Jump if carry flag is not set
JB	bit,rel	Jump if direct bit is set
JNB	bit,rel	Jump if direct bit is not set
JBC	bit,rel	Jump if direct bit is set and clear bit
CJNE	A,direct,rel	Compare direct byte to A and jump if not equal
CJNE	A,#data,rel	Compare immediate to A and jump if not equal
CJNE	Rn,#data rel	Compare immed. to reg. and jump if not equal
CJNE	@Ri,#data,rel	Compare immed. to ind. and jump if not equal
DJNZ	Rn,rel	Decrement register and jump if not zero
DJNZ	direct,rel	Decrement direct byte and jump if not zero
NOP		No operation

8051 INTERRUPTS

There are **five interrupt sources** for the 8051, which means that they can recognize 5 different events that can interrupt regular program execution. Each interrupt can be enabled or disabled by setting bits of the IE (Interrupt enable) register. Likewise, the whole interrupt system can be disabled by clearing the EA (global interrupt enable) bit of the same register.

The five interrupt sources are:

1. External interrupt 0 (IE0)
2. Timer 0 (TF0)
3. External interrupt 1 (IE1)
4. Timer 1 (TF1)
5. Serial port (R1 = T1)

Interrupt registers in 8051: 8051 μC has 2 8-bit interrupt registers.

1. Interrupt enable register (IE): it is an 8-bit register. It is bit/byte addressable. It is used to enable and disable the function of interrupt.

2. Interrupt priority register (IP): It is an 8-bit register. It is bit/byte addressable. It is used to select the low or high-level priority of individual interrupts.

Interrupt Priorities

There is a priority list instructing the controller how to handle multiple interrupts.

The priority list offers 3 levels of interrupt priority:

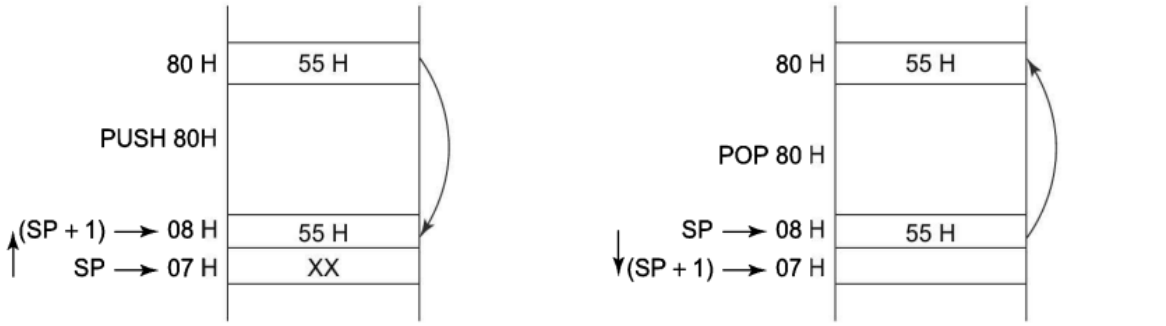
1. Reset - When a reset request arrives, everything is stopped and the microcontroller restarts.
2. Interrupt priority 1 - can be disabled by Reset only.
3. Interrupt priority 0 - can be disabled by both Reset and interrupt priority 1.

The IP Register (Interrupt Priority Register) specifies which one of the existing interrupt sources has higher and which one has lower priority. Interrupt priority is usually specified at the beginning of the program. According to that, there are several possibilities:

1. If an interrupt of higher priority arrives while an interrupt is in progress, it will be immediately stopped and the higher priority interrupt will be executed first.
2. If two interrupt requests, at different priority levels, arrive at the same time then the higher priority interrupt is serviced first.
3. If both interrupt requests, at the same priority level, occur one after another, the one which came later has to wait until the routine being in progress ends.
4. If two interrupt requests of equal priority arrive at the same time then the interrupt to be serviced is selected according to the following priority list:
    1. External interrupt INT0
    2. Timer 0 interrupt
    3. External Interrupt INT1
    4. Timer 1 interrupt
    5. Serial Communication Interrupt

8051 STACK

8051 stack operations are 8-bit wide i.e. in an operation using PUSH or POP instruction one byte of data is stored on to stack or retrieved from the stack. In case of internal 16 bit address push or pop to/from the stack, the operation is implemented byte by byte i.e. lower byte first followed by higher byte. The SP register is an 8-bit register and is initialized to internal RAM address 07H after reset. Obviously, the capabilities of stack in 8051 are limited compared to microprocessors. Fig. shows operation of PUSH instruction. The SP registers points to stack top. The stack top is always assumed to be preoccupied. So the SP is incremented first. Then 8 bit content of the 8-bit address provided as operand is pushed on to the stack memory address available in SP.



(a) Storing into Stack Memory **(b)** *Retrieving from Stack Memory*

Thus the PUSH instruction has following two steps.

1. Increment stack by 1.
2. Store 8-bit content of the 8-bit address specified in the instruction to the address pointed to by SP.

Complementarily, POP operation has the following two steps as shown in Fig.

1. Store the content of top of stack pointed to by SP register to the 8 bit memory specified in the instruction.
2. Decrement SP by 1.

8051 Addressing modes

8051 supports 6 addressing modes:

1. **Immediate Addressing:** Data is immediately available in the instruction.

   For example -
   ADD A, #77 Adds 77 (decimal) to A and stores in A
   MOV DPTR, #1000H Moves 1000 (hexadecimal) to the data pointer

2. **Register Addressing**: This way of addressing accesses the bytes in the current register bank. Data is available in the register specified in the instruction. The register bank is decided by 2 bits of Processor Status Word (PSW).

   For example -
   ADD A, R0 Adds content of R0 to A and stores in A

3. **Direct Addressing:** The address of the data is available in the instruction.

   For example -
   MOV A, 088H Moves the content of SFR TCON to A (088H is the address of special function register TCON in 8051)

4. **Register Indirect Addressing:** The address of data is available in the R0 or R1 registers as specified in the instruction.

   For example -

MOV A, @R0	moves the content of the address pointed by R0 to A

5. **Register-specific Addressing:** The operand is implicitly specified using one of the registers ie, such instructions operate only on a specific register.

   For example -

RLA	This instruction rotates accumulator left

6. **Indexed Addressing:** Only program memory can be accessed using this mode. It is mainly used for look-up table manipulations.

   For example -

MOVC A, @A+DPTR	DPTR has the base address of LUT and A has the relative (offset) address. The LUT content at the effective address obtained by adding the base address and relative address is moved to ACC (accumulator)