ASSEMBLY LANGUAGE PROGRAMMING OF 8051

Assembly Language Programming of 8051-25 mark

- ❖ Addressing Modes
- ❖ Instruction Set
- ❖ Development tools
- ❖ Assembler Directives
- ❖ Programming based on
  - ✓ Arithmetic & Logical operations
  - ✓ I/O parallel and serial ports
  - ✓ Timers & Counters, and ISR

8051 Addressing Modes

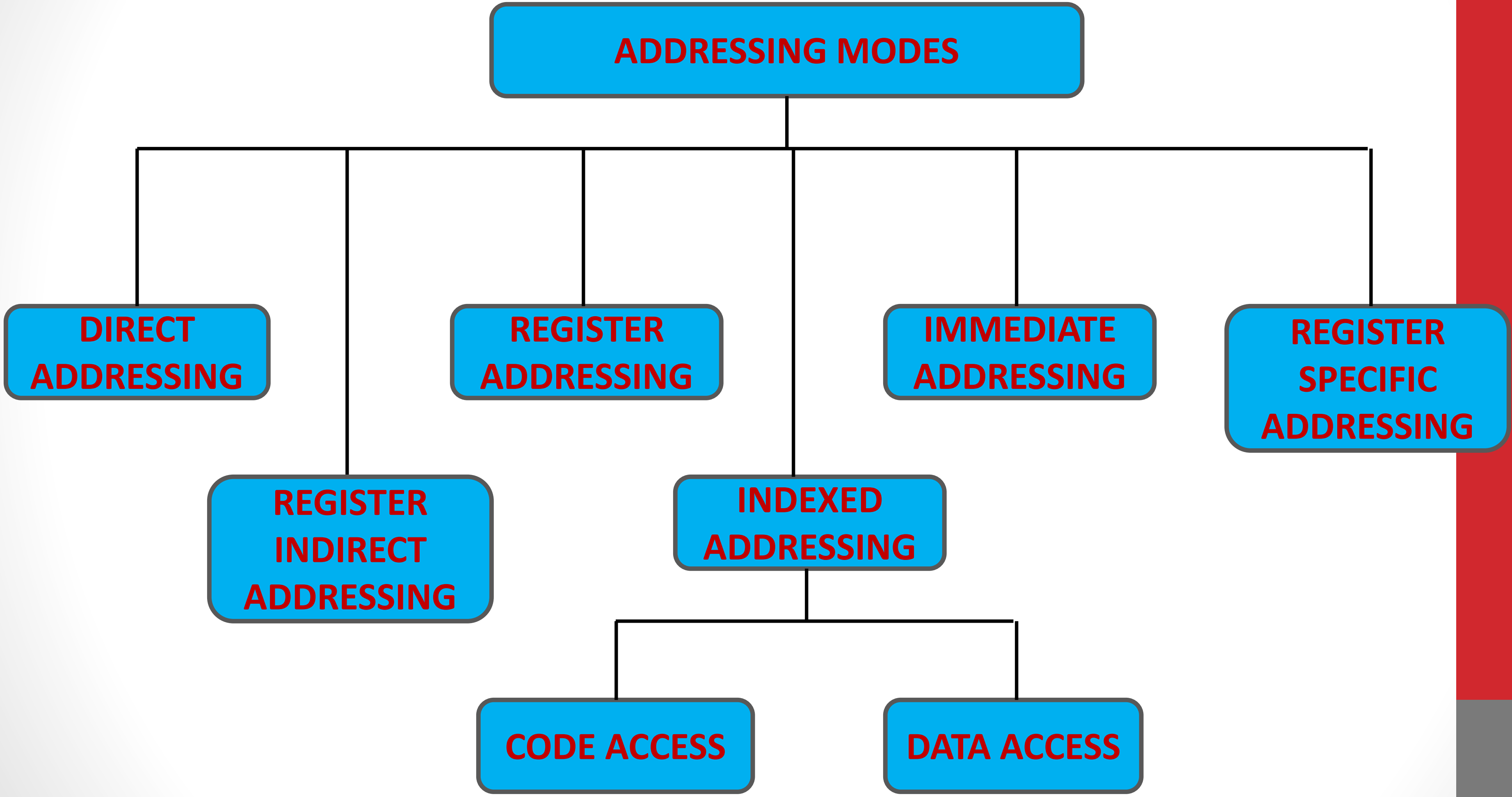
- When MC executes an instruction, it performs specific function on data

- Data is stored at some source location i.e. it could be in a register, memory or be provided as an immediate value

- Data must be moved or copied from source to destination location

- Ways by which these address locations are specified are called as **addressing modes**

8051 Addressing Modes



General Format of Assembly Language

- **LABEL:** $\underbrace{\textbf{OPCODE DEST, SRC}}_{\textbf{MNEMONIC}}$ **; COMMENTS**

- **MOV R1, A ; load contents of accumulator into R1**

- Here,
  - MOV → Opcode
  - R1 → Destination register
  - A → Source register

8051 Addressing Modes

❖Direct Addressing Mode

- RAM has been assigned addresses from 00H to 7FH → 128 bytes
  - RAM locations 00 – 1FH → Register banks and stack
  - RAM locations 20 – 2FH → Bit addressable space
  - RAM locations 30 – 7FH → General purpose RAM
- Special function registers use addresses from 80H – FFH
- Entire 128 bytes of RAM and SFRs can be accessed using single byte address assigned to each RAM location and each SFR

8051 Addressing Modes

❖ **Direct Addressing Mode**

- Operand is specified by an 8 bit address field in instruction
- MSB in the address indicates whether the location is within the on chip internal RAM or in SFR (Special Function Register)
- If MSB = 0 → location is within on chip internal RAM
- If MSB = 1 → location is in special function register

8051 Addressing Modes

❖**Direct Addressing Mode**

- **Example:**
- MOV A, 35H ; Copy contents of memory location 35H into register A
- MOV R1, 25H; Copy contents of memory location 25H into register R1 of selected register bank

8051 Addressing Modes

❖Register Indirect Addressing Mode

- Register is used as a pointer to the data
- Instruction specifies a register which contains address of an operand i.e. the register holds the actual address that will be used in the data movement operation
- Address may be 8 bit or 16 bit

8051 Addressing Modes

❖Register Indirect Addressing Mode

- Registers R0 and R1 of each register bank can be used as a pointer register to point to the contents in the RAM
- Registers R2 – R7 cannot be used to hold the address of an operand located in RAM
- @ sign indicates that the register acts as a pointer to memory location

8051 Addressing Modes

❖**Register Indirect Addressing Mode**

- **Advantage:**

Register indirect addressing mode makes accessing of data dynamic rather than static as in case of direct addressing mode

- **Limitation:**

R0 and R1 are the only registers that can be used as pointers in register indirect addressing mode. Since R0 and R1 are 8 bits, their use is limited to accessing any information only in internal RAM and SFR

8051 Addressing Modes

❖**Register Indirect Addressing Mode**

- **Example:**
- MOV A, @R0 ; Move contents of RAM location whose address is held by R0 into A
- MOV @R1, B; Move contents of B into RAM location whose address is held by R1
- MOV @R2, A ; Invalid instruction

8051 Addressing Modes

❖**Register Addressing Mode**

- Each register bank consists of registers from R0 – R7 which can be accessed using register addressing mode
- Registers are used to hold the data to be manipulated
- Source and destination registers must match in size. Permitted register are A and R0-R7
- Eg: MOV DPTR, A will give an error, since the source is an 8 bit register and destination is 16 bit register

8051 Addressing Modes

- **Register Addressing Mode**

- **Example:**
- MOV A, R0 ; Move contents of R0 into A
- MOV R2, A ; Move contents of A into R2
- ADD A, R5 ; Add contents of R5 to contents of A
- MOV R6, A ; Move contents of A into R6

8051 Addressing Modes

❖**Register Addressing Mode**

- Can move data between the accumulator and Rn (for n = 0 to 7)
- Movement of data between Rn registers is not allowed
- Eg: MOV R4, R7 ; Invalid instruction
- MOV R7, DPL ; Move contents of DPL register to R7 register
- MOV R4, DPH ; Move contents of DPH register to R4 register

8051 Addressing Modes

❖**Immediate Addressing Mode**

- Simplest method to get the data
- Source operand is a constant rather than a variable
- As the name implies, when the instruction is assembled, the operand comes immediately after the opcode
- Immediate data must be preceded by “#” sign

8051 Addressing Modes

❖ **Immediate Addressing Mode**

- **Example:**
- MOV A, #25H ; Move 25H into A
- MOV R4, #62 ; Move decimal value 62 into R4
- MOV B, #40H ; Move 40H into B
- MOV DPTR, #4510H ; Move 4510H into DPTR

  i.e. DPTR = 4510H

8051 Addressing Modes

❖ **Immediate Addressing Mode**

- Although DPTR is 16 bit register, it can be accessed as two 8 bit registers, DPH and DPL
- DPH → higher byte of DPTR
- DPL → lower byte of DPTR
- MOV DPTR, #2550H is same as:
  MOV DPL, #50H and MOV DPH, #25H

8051 Addressing Modes

❖**Immediate Addressing Mode**

- Can also be used to send data to 8051 ports

- **Example:**

  MOV P1, #55H ; Move 55H to Port P1

8051 Addressing Modes

❖**Register Specific Addressing Mode**

- Instructions refer to a specific register such as accumulator or data pointer DPTR

- **Example:**

- DA A ; Decimal Adjust Accumulator
- RR A ; Rotate contents of accumulator to right
- SWAP A ; Swap the nibbles within the accumulator

8051 Addressing Modes

❖**External Addressing Mode (Indexed Addressing Mode)**

- Code Access (External ROM access)
- Used in accessing data elements of look-up table entries located in ROM space of 8051
- Since the data elements are stored on the program (code) space ROM of 8051, **MOVC** is used instead of MOV
- C stands for code in the instruction MOVC

8051 Addressing Modes

❖**External Addressing Mode (Indexed Addressing Mode)**

- Code Access (External ROM access)
- **Requires accumulator to be one of the operand. Second operand can be DPTR or PC (Program Counter)**

- Eg: MOVC A, @A + DPTR ; loads the accumulator with the byte from program memory. The byte from program memory is fetched from the memory location obtained by the sum of unsigned 8 bit accumulator contents and contents of DPTR

8051 Addressing Modes

❖**External Addressing Mode (Indexed Addressing Mode)**

▪ Data Access (External RAM access)

- 8051 has 64KB of memory space set aside exclusively for data storage

- This memory is referred to as external memory and it is accessed only by **MOVX** instruction

- **Requires accumulator to be one of the operand**

8051 Addressing Modes

❖**External Addressing Mode (Indexed Addressing Mode)**

- Data Access (External RAM access)

- Example: MOVX @R0, A

- The above instruction will copy the data from accumulator to the external memory location whose address is given by register R0

- Using R0 or R1 registers, programmer can access external data memory from location 00H to FFH. To access memory beyond this DPTR is used

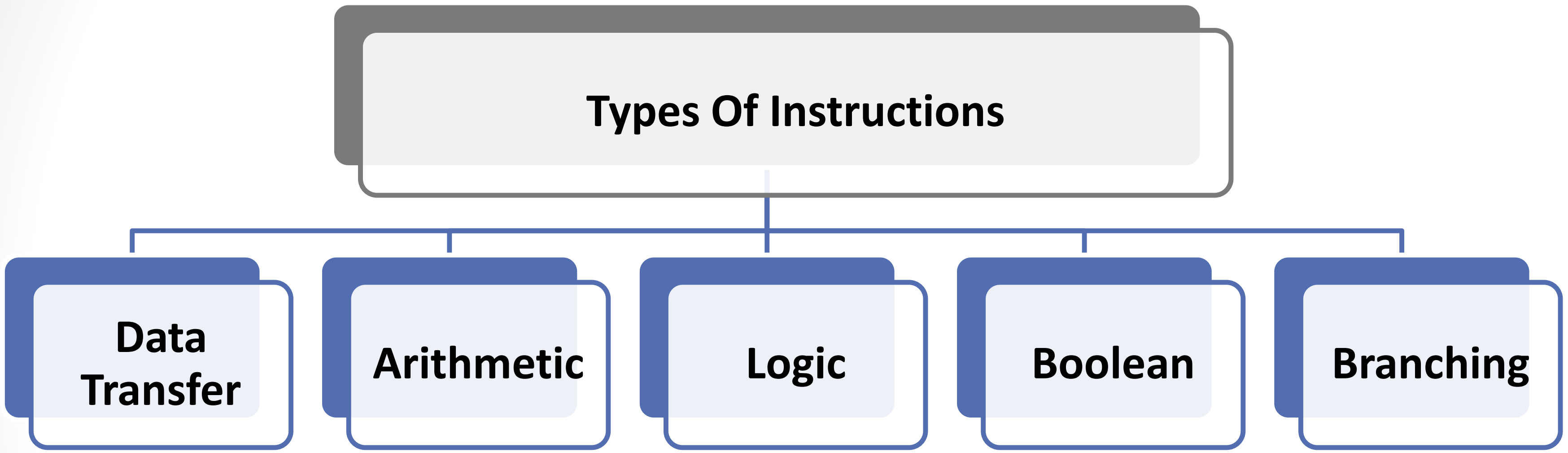
8051 Instruction Set

❖8051 has 255 instructions

❖Every 8-bit opcode from 00 to FF is used except for A5

❖A5 is a reserved opcode that performs the same function as NOP (No Operation)

8051 Instruction Set



8051 Instruction Set – Data Transfer Instructions

- Used to move the contents from one register to other
- Data can be transferred to stack with help of PUSH & POP instructions
- Includes the following instructions:
  - ✓MOV
  - ✓MOVX
  - ✓MOVC
  - ✓PUSH
  - ✓POP
  - ✓XCH
  - ✓XCHD

8051 Instruction Set – Data Transfer Instructions

- **Syntax:** MOV destination, source

- Moves data bytes between the two specified operands i.e. source and destination

- Byte specified by the second operand is copied to the location specified by the first operand. Source data byte is not affected

- Allows 15 combinations of source and destination transfers

- Depending on the type of transfer, the number of bytes required may be 1, 2 or 3 and number of cycles required are 1 or 2

8051 Instruction Set – Data Transfer Instructions

Mnemonic	**MOV A, Rn**
Function	Move a byte from register (R0 – R7) to accumulator
Example	MOV A, R1 ; Move the contents of register R1 of selected register bank to accumulator
Machine Cycles	1
Clock Pulses	12
Bytes	1
Addressing Mode	Register Addressing Mode
Flags	No flags are affected

8051 Instruction Set – Data Transfer Instructions

Mnemonic	**MOV A, direct**
Function	Move a byte from the direct address specified to accumulator
Example	MOV A, 50H ; Move the contents from location 50H to accumulator
Machine Cycles	1
Clock Pulses	12
Bytes	2
Addressing Mode	Direct Addressing Mode
Flags	No flags are affected

8051 Instruction Set – Data Transfer Instructions

Mnemonic	**MOV A, @Ri**
Function	Move a byte from the memory location pointed by Ri (R0 or R1) to the accumulator
Example	MOV A, @R0 ; Move the contents of memory location whose address is specified in R) register of selected register bank to accumulator Let R0=50H, contents of 50H = 20H, then A=20H
Machine Cycles	1
Clock Pulses	12
Bytes	1
Addressing Mode	Register Indirect Addressing Mode
Flags	No flags are affected

8051 Instruction Set – Data Transfer Instructions

Mnemonic	**MOV A, #data**
Function	Move the immediate data to accumulator
Example	MOV A, #50H ; Moves the data 50H to accumulator
Machine Cycles	1
Clock Pulses	12
Bytes	2
Addressing Mode	Immediate Addressing Mode
Flags	No flags are affected

8051 Instruction Set – Data Transfer Instructions

Mnemonic	**MOV Rn, A**
Function	Move a byte from accumulator to register (R0 – R7)
Example	MOV R4, A ; Moves the contents of accumulator to register R4 of selected register bank
Machine Cycles	1
Clock Pulses	12
Bytes	1
Addressing Mode	Register Addressing Mode
Flags	No flags are affected

8051 Instruction Set – Data Transfer Instructions

Mnemonic	**MOV Rn, direct**
Function	Move a byte from direct address specified to register Rn of selected register bank (R0 – R7)
Example	MOV R4, 50H ; Moves the contents from memory location 50H to register R4 of selected register bank
Machine Cycles	2
Clock Pulses	24
Bytes	2
Addressing Mode	Direct Addressing Mode
Flags	No flags are affected

8051 Instruction Set – Data Transfer Instructions

Mnemonic	**MOV Rn, #data**
Function	Move the immediate data specified to register Rn of selected register bank (R0 – R7)
Example	MOV R4, #40H ; Moves the immediate data 40H to register R4 of selected register bank
Machine Cycles	1
Clock Pulses	12
Bytes	2
Addressing Mode	Immediate Addressing Mode
Flags	No flags are affected

8051 Instruction Set – Data Transfer Instructions

Mnemonic	**MOV direct, A**
Function	Move the contents of accumulator to the direct address specified
Example	MOV 40H, A ; Moves the contents of accumulator to the address 40H
Machine Cycles	1
Clock Pulses	12
Bytes	2
Addressing Mode	Direct Addressing Mode
Flags	No flags are affected

8051 Instruction Set – Data Transfer Instructions

Mnemonic	**MOV direct, Rn**
Function	Move the contents of register Rn (R0 to R7) to the direct address specified
Example	MOV 40H, R2 ; Moves the contents of register R2 of selected register bank to the direct address specified
Machine Cycles	2
Clock Pulses	24
Bytes	2
Addressing Mode	Direct Addressing Mode
Flags	No flags are affected

8051 Instruction Set – Data Transfer Instructions

Mnemonic	**MOV direct, direct**
Function	Move the contents of source direct address to destination direct address specified
Example	MOV 40H, 50H ; Moves the contents of memory location 50H to memory location 40H
Machine Cycles	2
Clock Pulses	24
Bytes	3
Addressing Mode	Direct Addressing Mode
Flags	No flags are affected

8051 Instruction Set – Data Transfer Instructions

Mnemonic	**MOV direct, @Ri**
Function	Move the contents from memory location whose address is specified in register Ri (R0 or R1) of selected register bank to the direct address specified
Example	MOV 40H, @R1 ; Moves the contents of memory location whose address is specified by register R1 to memory location 40H
Machine Cycles	2
Clock Pulses	24
Bytes	2
Addressing Mode	Register Indirect Addressing Mode
Flags	No flags are affected

8051 Instruction Set – Data Transfer Instructions

Mnemonic	**MOV direct, #data**
Function	Move the immediate data to the direct address specified
Example	MOV 40H, #20H ; Moves the immediate data 20H to the memory location 40H
Machine Cycles	2
Clock Pulses	24
Bytes	3
Addressing Mode	Immediate Addressing Mode
Flags	No flags are affected

8051 Instruction Set – Data Transfer Instructions

Mnemonic	**MOV @Ri, A**
Function	Move the contents of accumulator to the memory location whose address is specified in register Ri (R0 or R1) of selected register bank
Example	MOV @R1, A ; Moves the contents of accumulator to memory location whose address is specified in register R1
Machine Cycles	1
Clock Pulses	12
Bytes	1
Addressing Mode	Register Indirect Addressing Mode
Flags	No flags are affected

8051 Instruction Set – Data Transfer Instructions

Mnemonic	**MOV @Ri, direct**
Function	Move the contents of direct address specified to the memory location specified in register Ri (R0 or R1) of selected register bank
Example	MOV @R1, 40H ; Moves the contents of memory location 40H to the memory location specified by register R1
Machine Cycles	2
Clock Pulses	24
Bytes	2
Addressing Mode	Register Indirect Addressing Mode
Flags	No flags are affected

8051 Instruction Set – Data Transfer Instructions

Mnemonic	**MOV @Ri, #data**
Function	Move the immediate data to the memory location specified in register Ri (R0 or R1) of selected register bank
Example	MOV @R1, #40H ; Moves the immediate data 40H to the memory location specified by register R1
Machine Cycles	1
Clock Pulses	12
Bytes	2
Addressing Mode	Immediate Addressing Mode
Flags	No flags are affected

8051 Instruction Set – Data Transfer Instructions

Mnemonic	**MOV DPTR, #immediate (16 bit data)**
Function	Move the 16 bit constant to the data pointer
Example	MOV DPTR, #1234H ; Moves the immediate data 1234H into DPTR. High order byte i.e. 12H will be stored in DPH and low order byte i.e. 34H will be stored in DPL.
Machine Cycles	2
Clock Pulses	24
Bytes	3
Addressing Mode	Immediate Addressing Mode
Flags	No flags are affected
This is the only instruction which moves 16 bits of data at a time	

8051 Instruction Set – Data Transfer Instructions

Mnemonic	**MOVC A, @A+ <base register>**
Function	Used to access the ROM (code) Loads the accumulator with a code byte or constant from program memory Hence, it is essential to generate a 16 bit address, so that data can be fetched from that address The address of the byte to be fetched is equal to the sum of the unsigned contents of 8 bit accumulator and 16 bit base register Base register can be DPTR or PC

8051 Instruction Set – Data Transfer Instructions

Mnemonic	**MOVC A, @A+ <base register>**
Function	If the base register used is PC (Program Counter) then the PC is first incremented to the address of the next instruction and then the contents of the accumulator are added to it
Example 1:	MOVC A, @A + DPTR Let A = 30H, DPTR = 2000H, 2030H = 15H Then contents of memory location (30 + 2000) i.e. 2030H which is 15H will be moved to the accumulator

8051 Instruction Set – Data Transfer Instructions

Mnemonic	**MOVC A, @A+ <base register>**
Example 2:	MOVC A, @A + PC Let A = 30H, PC = 3000H, 3031H = 55H The 16 bit address is calculated as follows: PC = PC + 1 = 3000 H+ 1H = 3001H A + PC = 30H + 3001H = 3031H This will move the contents at memory location 3031H i.e. 55H to the accumulator

8051 Instruction Set – Data Transfer Instructions

Mnemonic	**MOVC A, @A+ <base register>**
Machine Cycles	2
Clock Pulses	24
Bytes	1
Addressing Mode	Indirect Addressing Mode
Flags	No flags are affected

8051 Instruction Set – Data Transfer Instructions

Mnemonic	**MOVX <dest-byte>, <src-byte>**
Function	Used to access external RAM (data) memory Transfers data between the accumulator and a byte of external data memory Depending on whether the indirect address provided of the external data RAM is 8 bit or 16 bit, there are 2 types of instructions

8051 Instruction Set – Data Transfer Instructions

Mnemonic	**MOVX <dest-byte>, <src-byte>**
Function	If the address is 16 bit, DPTR is used for generating it Done for larger RAM array Port 2 outputs the higher order address bits (contents of DPH) and Port 0 outputs the lower order address bits (contents of DPL) multiplexed with data

8051 Instruction Set – Data Transfer Instructions

Mnemonic	**MOVX <dest-byte>, <src-byte>**
Example	MOVX A, @R0 ; Move the data from 8 bit address specified by register R0 of selected register bank to the accumulator Here, R0 register points to external data memory

8051 Instruction Set – Data Transfer Instructions

MOVX A, @DPTR

Move the contents of external data memory location pointed by DPTR to the accumulator

MOVX @Ri, A

Move the contents of accumulator to the external data memory location pointed by register Ri (R0 or R1) of selected register bank

MOVX @DPTR, A

Move the contents of accumulator to the 16 bit external data memory location pointed by DPTR

8051 Instruction Set – Data Transfer Instructions

Mnemonic	**PUSH <direct>**
Function	Copies the data from the source address specified onto the stack Stack Pointer (SP) is incremented by 1 before the data is copied to the internal RAM location addressed by stack pointer Stack will grow up in memory as data is pushed onto the stack If the stack exceeds 7FH (top of internal RAM) then it results in error

8051 Instruction Set – Data Transfer Instructions

Mnemonic	PUSH <direct>
Example	PUSH DPL PUSH DPH Let SP = 0AH and DPTR = 1234H 1st instruction i.e. PUSH DPL will set SP = 0BH and store 34H in internal RAM location 0BH 2nd instruction i.e. PUSH DPH will set SP = 0CH and store 12H in internal RAM location 0CH Stack pointer will remain at 0CH

8051 Instruction Set – Data Transfer Instructions

Mnemonic	**PUSH <direct>**
Machine Cycles	2
Clock Pulses	24
Bytes	2
Addressing Mode	Direct Addressing Mode
Flags	No flags are affected

8051 Instruction Set – Data Transfer Instructions

Mnemonic	**POP <direct>**
Function	Copies the data from the stack to the destination location Stack Pointer (SP) is decremented by 1 after the data is copied from the stack RAM address to the destination location
Example	POP DPL Let SP = 45H and data at internal RAM location 45H be 60H This instruction will decrement SP to 44H and DPL = 60H

8051 Instruction Set – Data Transfer Instructions

Mnemonic	**POP <direct>**
Machine Cycles	2
Clock Pulses	24
Bytes	2
Addressing Mode	Direct Addressing Mode
Flags	No flags are affected
PUSH and POP operation can push / pop a single byte only	

8051 Instruction Set – Data Transfer Instructions

Mnemonic	**XCH A, <byte variable>**
Function	Loads the accumulator with the contents of byte variable At the same time, the original accumulator contents are written to the byte variable Source / destination operand can use register, direct or register indirect addressing

8051 Instruction Set – Data Transfer Instructions

Mnemonic	**XCH A, Rn**
Function	Exchanges the contents of accumulator with the register Rn (R0 to R7) of the selected register bank
Example	XCH A, R1 ; Load the contents of register R1 of selected register bank in the accumulator and at the same time the original contents of accumulator will be copied in register R1
Machine Cycles	1
Clock Pulses	12
Bytes	1
Addressing Mode	Register Addressing Mode
Flags	No flags are affected

8051 Instruction Set – Data Transfer Instructions

Mnemonic	**XCH A, direct**
Function	Exchanges the contents of accumulator with the contents of the direct address specified
Example	XCH A, 20H ; Load the contents of memory location specified i.e. 20H into the accumulator and at the same time move the original contents of accumulator to memory location 20H
Machine Cycles	1
Clock Pulses	12
Bytes	2
Addressing Mode	Direct Addressing Mode
Flags	No flags are affected

8051 Instruction Set – Data Transfer Instructions

Mnemonic	**XCH A, @Ri**
Function	Exchanges the contents of accumulator with the contents of memory location pointed by register Ri (R0 or R1)
Example	XCH A, @R1 ; Load the contents of memory location pointed by register R1 to the accumulator and at the same time the original contents of accumulator are copied to memory location pointed by R1 register
Machine Cycles	1
Clock Pulses	12
Bytes	1
Addressing Mode	Register Indirect Addressing Mode
Flags	No flags are affected

8051 Instruction Set – Data Transfer Instructions

Mnemonic	**XCHD A, @Ri**
Function	Exchanges the lower nibble of the accumulator (bits 3-0) with the lower nibble of the memory location pointed by register Ri (R0 or R1) of specified register bank Upper nibble (bits 7-4) of each register remains unchanged
Example	XCHD A, @R1 ; If R1 contains address 45H and A = 24H and internal RAM location 45H = 15H, then this instruction will leave RAM location 45H with value14H and accumulator with value 25H

8051 Instruction Set – Data Transfer Instructions

Mnemonic	**XCHD A, @Ri**
Machine Cycles	1
Clock Pulses	12
Bytes	1
Addressing Mode	Register Indirect Addressing Mode
Flags	No flags are affected

8051 Instruction Set – Arithmetic Instructions

- Includes the following instructions:
  - ✓ADD
  - ✓ADDC
  - ✓SUBB
  - ✓INC
  - ✓DEC
  - ✓MUL
  - ✓DIV
  - ✓DA A

8051 Instruction Set – Arithmetic Instructions

- **Syntax:** ADD A, <src-byte>
- Adds the byte variable indicated by the source operand to the contents of the accumulator
- After addition the result is stored back to the accumulator
- All the addressing modes can be used for source such as: immediate, register, direct address or indirect address

8051 Instruction Set – Arithmetic Instructions

Mnemonic	**ADD A, Rn**
Function	Adds the byte in register Rn (R0 – R7) of the selected register bank with the byte in the accumulator Result is stored in the accumulator
Machine Cycles	1
Clock Pulses	12
Bytes	1
Addressing Mode	Register Addressing Mode
Flags	Flags are affected (Carry flag, Auxiliary Carry flag and Overflow flags)

8051 Instruction Set – Arithmetic Instructions

Mnemonic	**ADD A, Rn**
Example: ADD A, R0	Let A = 42H → (0100 0010 B) and R0 = 91H → (1001 0001 B) A + R0 → (1101 0011 B) Then, ADD A, R0 will leave A = D3H with carry flag, auxiliary carry flag and overflow flag cleared Carry flag is set when there is carry out from MSB (D7 bit) after an addition AC flag is set when there is a carry out of the lower nibble into the higher nibble (D3 to D4 bit)

8051 Instruction Set – Arithmetic Instructions

Mnemonic	**ADD A, Rn**
Example: ADD A, R0	Overflow flag is set if either of the conditions occur: 1) There is carry from D6 to D7 but no carry out of D7 (CY=0) 2) There is carry from D7 out (CY=1) but no carry from D6 to D7

8051 Instruction Set – Arithmetic Instructions

Mnemonic	**ADD A, direct**
Function	Adds the contents of the memory location whose direct address is specified with the contents of the accumulator
Example 1	ADD A, 25H Let A = 77H → (0111 0111 B) and Contents at 25H = 45H → (0100 0101 B) ADD A, 25H → (1011 1100 B) A = BCH with carry and auxiliary carry flags cleared and overflow and parity flags are set. PSW = 05H

8051 Instruction Set – Arithmetic Instructions

Mnemonic	**ADD A, direct**
Example 2	ADD A, 25H Let A = 0F7H → (1111 0111 B) and Contents at 25H = 85H → (1000 0101 B) ADD A, 25H → (0111 1100 B) A = 7CH with carry, overflow and parity flags are set. PSW = 85H

8051 Instruction Set – Arithmetic Instructions

Example 3 - ADD A, direct

MOV A, #-128;   A = 1000 0000 (A=80H)
MOV R4, #-2;   R4 = 1111 1110 (R4=FEH)
ADD A, R4 ;   A = 0111 1110 (A=7EH= +126 → Invalid)

-128 + (-2) = -130

PSW = 84H

Carry and overflow flags are set

8051 Instruction Set – Arithmetic Instructions

Example 3 - ADD A, direct

MOV A, #-2;   A = 1111 1110 (A=FEH)
MOV R4, #-5;   R4 = 1111 1011 (R4=FBH)
ADD A, R4 ;   A = 1111 1001 (A=F9H= -7 → Valid)

-2+ (-5) = -7

PSW = C0H

Carry & auxiliary carry = 1 and overflow = 0

8051 Instruction Set – Arithmetic Instructions

Mnemonic	**ADD A, @Ri**
Function	Adds the contents of memory location whose address is pointed by register Ri (R0 or R1) of selected register bank with the byte in the accumulator Result is stored in the accumulator
Machine Cycles	1
Clock Pulses	12
Bytes	1
Addressing Mode	Register Indirect Addressing Mode
Flags	Flags are affected (Carry flag, Auxiliary Carry flag and Overflow flags)

8051 Instruction Set – Arithmetic Instructions

Mnemonic	**ADD A, @Ri**
Example: ADD A, @R0	Let R0 = 35H and data at 35H = 89H 89H → (1000 1001 B) A = 95H → (1001 0101 B) ADD A, @R0 → (0001 1110 B) = 1EH Then, ADD A, R0 will leave A = 1EH with auxiliary carry flag cleared and sets the carry flag and overflow flag

8051 Instruction Set – Arithmetic Instructions

Mnemonic	**ADD A, #data**
Function	Adds the immediate 8 bit data with the byte in the accumulator Result is stored in the accumulator
Machine Cycles	1
Clock Pulses	12
Bytes	2
Addressing Mode	Register Indirect Addressing Mode
Flags	Flags are affected (Carry flag, Auxiliary Carry flag and Overflow flags)

8051 Instruction Set – Arithmetic Instructions

Mnemonic	**ADD A, #data**
Example: ADD A, #40H	Let A = 29H = (0010 1001 B) A = 29H → (0010 1001 B) 40H → (0100 0000 B) ADD A, #40H → (0110 1001 B) = 69H Then, ADD A, #40H will add data 40H with the contents of accumulator i.e. 29H. Thus A = 69H with auxiliary carry, carry and overflow flag cleared

8051 Instruction Set – Arithmetic Instructions

Mnemonic	**ADDC A, <src-byte>**
Function	Adds the byte variable indicated in the instruction with the contents of carry flag and accumulator (A) ← (A) + <src-byte> + carry Result is stored in the accumulator All addressing modes can be used for the source: immediate, register, direct and indirect address

8051 Instruction Set – Arithmetic Instructions

Mnemonic	**ADDC A, Rn**
Function	Adds the contents of the accumulator with the contents of register Rn (R0 – R7) of selected register bank and the carry flag Result is stored in accumulator
Machine Cycles	1
Clock Pulses	12
Bytes	1
Addressing Mode	Register Addressing Mode
Flags	Flags are affected (Carry flag, Auxiliary Carry flag and Overflow flags)

8051 Instruction Set – Arithmetic Instructions

Mnemonic	ADDC A, Rn
Example: ADDC A, R1	Let R1 = 54H, A = 99H and CF = 1 Then ADDC A, R1 will add the contents of accumulator with the contents of register R1 of selected register bank and carry flag 54H → (0101 0100 B) 99H → (1001 1001 B) 01H → (0000 0001 B) A → (1110 1110 B) = EEH Clears auxiliary carry, carry and overflow flag

8051 Instruction Set – Arithmetic Instructions

Mnemonic	**ADDC A, direct**
Function	Adds the contents of the accumulator with the contents of memory location whose direct address is specified and the carry flag Result is stored in accumulator
Machine Cycles	1
Clock Pulses	12
Bytes	2
Addressing Mode	Direct Addressing Mode
Flags	Flags are affected (Carry flag, Auxiliary Carry flag and Overflow flags)

8051 Instruction Set – Arithmetic Instructions

Mnemonic	**ADDC A, direct**
Example: ADDC A, 10H	Let the contents of memory location whose address is 10H be 9AH A = 98H and CF = 0 Then ADDC A, 10H will add the contents of accumulator with the contents of memory location 10H i.e. 9AH and carry flag 98H → (1001 1000 B) 9AH →(1001 1010 B) 00H → (0000 0000 B) A → (0011 0010 B) → 32H Auxiliary carry, overflow and carry flag is set

8051 Instruction Set – Arithmetic Instructions

Mnemonic	**ADDC A, @Ri**
Function	Adds the contents of the accumulator with the contents of memory location whose address is specified by register Ri (R0 or R1) of selected register bank and the carry flag Result is stored in accumulator
Machine Cycles	1
Clock Pulses	12
Bytes	1
Addressing Mode	Register Indirect Addressing Mode
Flags	Flags are affected (Carry flag, Auxiliary Carry flag and Overflow flags)

8051 Instruction Set – Arithmetic Instructions

Mnemonic	**ADDC A, @Ri**
Example: ADDC A, @R0	Let R0 hold 30H and the contents of memory location 30H be 89H A = 45H and CY = 1 Then ADDC A, @R0 will add the contents of accumulator with the contents of memory location pointed by R0 i.e. 89H and carry flag 45H → (0100 0101 B) 89H → (1000 1001 B) 01H → (0000 0001 B) A → (1100 1111 B) → CFH Auxiliary carry, overflow and carry flag are cleared

8051 Instruction Set – Arithmetic Instructions

Mnemonic	**ADDC A, #data**
Function	Adds the contents of the accumulator with the immediate 8 bit data specified and the carry flag Result is stored in accumulator
Machine Cycles	1
Clock Pulses	12
Bytes	2
Addressing Mode	Immediate Addressing Mode
Flags	Flags are affected (Carry flag, Auxiliary Carry flag and Overflow flags)

8051 Instruction Set – Arithmetic Instructions

Mnemonic	**ADDC A, #data**
Example: ADDC A, #40H	Let A = 50H and CY = 1 Then ADDC A, #40H will add the contents of accumulator with the immediate contents 40H and carry flag 50H → (0101 0000 B) 40H → (0100 0000 B) 01H → (0000 0001 B) A → (1001 0001 B) → 91H Auxiliary carry and carry flag are cleared and overflow flag is set

8051 Instruction Set – Arithmetic Instructions

- **Syntax:** SUBB A, <src-byte>

- Subtracts the indicated byte variable and the carry flag contents together, from the accumulator
- (A) ← (A) – (src-byte) – (CY)
- Result is stored in accumulator
- Carry flag is treated as borrow flag
- All addressing modes can be used as source: immediate data, register, direct address and indirect address

8051 Instruction Set – Arithmetic Instructions

Mnemonic	**SUBB A, Rn**
Function	Subtract the contents of register Rn (R0 – R7) of the selected register bank and contents of carry flag together, from the contents of accumulator Result is stored in accumulator
Machine Cycles	1
Clock Pulses	12
Bytes	1
Addressing Mode	Register Addressing Mode
Flags	Flags are affected

8051 Instruction Set – Arithmetic Instructions

Mnemonic	**SUBB A, Rn**
Example: SUBB A, R3	Let A = C9H, R3 = 54H and CY = 1 Then SUBB A, R3 will subtract the contents of R3 i.e. 54H and carry flag from the contents of accumulator i.e. C9H R3 + CY = 54H + 01H = 55H C9H → (1100 1001 B) 55H → (0101 0101 B) A → (0111 0100 B) → 74H Auxiliary carry and carry flag are cleared and overflow flag is set

8051 Instruction Set – Arithmetic Instructions

Mnemonic	**SUBB A, Rn**
Example: SUBB A, R3	Carry (borrow) flag is set if a borrow is needed for bit 7 and clears CY otherwise Auxiliary carry flag is set if a borrow is needed for bit 3 otherwise it is cleared Overflow flag is set if : 1) a borrow is needed for bit 6, but not into bit 7 2) a borrow is needed into bit 7, but not into bit 6

8051 Instruction Set – Arithmetic Instructions

Mnemonic	**SUBB A, direct**
Function	Subtract the contents of memory location whose direct address is specified and contents of carry flag together, from the contents of accumulator Result is stored in accumulator
Machine Cycles	1
Clock Pulses	12
Bytes	2
Addressing Mode	Direct Addressing Mode
Flags	Flags are affected

8051 Instruction Set – Arithmetic Instructions

Mnemonic	**SUBB A, direct**
Example: SUBB A, 45H	Let A = 54H Contents at memory location 45H = C9H and CY = 1 Then SUBB A, 45H will subtract the contents of memory location 45H i.e. C9H and carry flag from the contents of accumulator i.e. C9H Data at 45H + CY = C9H + 01H = CAH 54H → (0101 0100 B) CAH → (1100 1010 B) A → (1000 1010 B) → 8AH Auxiliary carry, overflow and carry flag are set

8051 Instruction Set – Arithmetic Instructions

Mnemonic	SUBB A, @Ri
Function	Subtract the contents of memory location pointed by register Ri (R0 or R1) and contents of carry flag together, from the contents of accumulator Result is stored in accumulator
Machine Cycles	1
Clock Pulses	12
Bytes	1
Addressing Mode	Register Indirect Addressing Mode
Flags	Flags are affected

8051 Instruction Set – Arithmetic Instructions

Mnemonic	**SUBB A, @Ri**
Example: SUBB A, @R1	Let A = C9H R1 = 30H and data at 30H = 54H and CY = 1 Then SUBB A, @R1 will subtract the contents of memory location pointed by R1 i.e. 54H and carry flag from the contents of accumulator i.e. C9H Data at 30H + CY = 54H + 01H = 55H C9H → (1100 1001 B) 55H → (0101 0101 B) A → (0111 0100 B) → 74H Auxiliary carry and carry flag are cleared and overflow flag is set

8051 Instruction Set – Arithmetic Instructions

Mnemonic	**SUBB A, #data**
Function	Subtract the immediate data and contents of carry flag together, from the contents of accumulator Result is stored in accumulator
Machine Cycles	1
Clock Pulses	12
Bytes	2
Addressing Mode	Immediate Addressing Mode
Flags	Flags are affected

8051 Instruction Set – Arithmetic Instructions

Mnemonic	**SUBB A, #data**
Example: SUBB A, #40H	Let A = 50H CY = 1 Then SUBB A, #40H will subtract the immediate data 40H and carry flag from the contents of accumulator i.e. 50H 40H + CY = 40H + 01H = 41H 50H → (0101 0000 B) 41H → (0100 0001 B) A → (0000 1111 B) → 0FH Auxiliary carry flag is set and carry and overflow flag are cleared

8051 Instruction Set – Arithmetic Instructions

- **Syntax:** INC <byte>

- Increments the specified byte variable by 1
- byte ← byte + 1
- If the byte value is FFH and if it is incremented, then the result will overflow to 00H
- Supports 3 addressing modes: register, direct and register indirect

8051 Instruction Set – Arithmetic Instructions

Mnemonic	**INC Rn**
Function	Increment the contents of register Rn (R0 to R7) of selected register bank by 1 Rn ← Rn + 1
Example: INC R5	Let R5 = 0BH then this instruction will make the value of R5 to R5 = 0CH
Machine Cycles	1
Clock Pulses	12
Bytes	1
Addressing Mode	Register Addressing Mode
Flags	No flags are affected

8051 Instruction Set – Arithmetic Instructions

Mnemonic	**INC <direct>**
Function	Increment the contents of memory location whose direct address is specified in instruction by 1
Example: INC 40H	Let contents of memory location 40H = 35H, then this instruction will increment the contents of memory location 40H by 1 i.e. 40H = 36H
Machine Cycles	1
Clock Pulses	12
Bytes	2
Addressing Mode	Direct Addressing Mode
Flags	No flags are affected

8051 Instruction Set – Arithmetic Instructions

Mnemonic	**INC @Ri**
Function	Increment the contents of memory location that is pointed by register Ri (R0 or R1) by 1
Example: INC @R1	Let contents of R1 = 36H and contents at memory location 36H = 12H, then this instruction will increment the contents of memory location pointed by register R1 i.e. 36H by 1 i.e. now 36H = 13H
Machine Cycles	1
Clock Pulses	12
Bytes	1
Addressing Mode	Register Indirect Addressing Mode
Flags	No flags are affected

8051 Instruction Set – Arithmetic Instructions

Mnemonic	**INC DPTR**
Function	Increment the contents of data pointer by 1 A 16 bit increment is performed Overflow of low order byte of data pointer (DPL) from FFH to 00H will increment the higher order byte (DPH) DPTR is the only 16 bit register that is incremented

8051 Instruction Set – Arithmetic Instructions

Mnemonic	**INC DPTR**
Example: INC DPTR	Let contents of DPTR = 15FFH. So DPH = 15H and DPL = FFH. Then this instruction will cause DPH = 16H and DPL = 00H. Thus DPTR = 1600H
Machine Cycles	2
Clock Pulses	24
Bytes	1
Addressing Mode	Register Addressing Mode
Flags	No flags are affected

8051 Instruction Set – Arithmetic Instructions

- **Syntax:** DEC <byte>

- Decrements the specified byte variable by 1
- byte ← byte - 1
- If the byte value is 00H and if it is decremented, then the result will underflow to FFH
- Supports 3 addressing modes: register, direct and register indirect

8051 Instruction Set – Arithmetic Instructions

Mnemonic	**DEC Rn**
Function	Decrement the contents of register Rn (R0 to R7) of selected register bank by 1 Rn $\leftarrow$ Rn – 1
Example: DEC R5	Let R5 = 0BH then this instruction will make the value of R5 to R5 = 0AH
Machine Cycles	1
Clock Pulses	12
Bytes	1
Addressing Mode	Register Addressing Mode
Flags	No flags are affected

8051 Instruction Set – Arithmetic Instructions

Mnemonic	**DEC <direct>**
Function	Decrement the contents of memory location whose direct address is specified in instruction by 1
Example: DEC 40H	Let contents of memory location 40H = 35H, then this instruction will decrement the contents of memory location 40H by 1 i.e. 40H = 34H
Machine Cycles	1
Clock Pulses	12
Bytes	2
Addressing Mode	Direct Addressing Mode
Flags	No flags are affected

8051 Instruction Set – Arithmetic Instructions

Mnemonic	**DEC @Ri**
Function	Decrement the contents of memory location that is pointed by register Ri (R0 or R1) by 1
Example: DEC @R1	Let contents of R1 = 36H and contents at memory location 36H = 12H, then this instruction will decrement the contents of memory location pointed by register R1 i.e. 36H by 1 i.e. now 36H = 11H
Machine Cycles	1
Clock Pulses	12
Bytes	1
Addressing Mode	Register Indirect Addressing Mode
Flags	No flags are affected

8051 Instruction Set – Arithmetic Instructions

Mnemonic	**MUL AB**
Function	Multiplies an 8 bit unsigned integer in the accumulator and register B After multiplication: Low order byte of the 16 bit product → A High order byte → B Largest possible product is FE01H when A = FFH and B = FFH Then A = 01H and B = FEH after multiplication There is no comma between A and B. Only registers A and B can be used. Original contents of A and B are lost.

8051 Instruction Set – Arithmetic Instructions

Mnemonic	**MUL AB**
Example: MUL AB	Let A = 50H and B = A0H Then MUL AB = (50H) * (A0H) = (3200H) So B = 32H and A = 00H Sets the overflow flag and clears the carry flag
Machine Cycles	4
Clock Pulses	48
Bytes	1
Addressing Mode	Register Addressing Mode
Flags	Flags are affected

8051 Instruction Set – Arithmetic Instructions

Mnemonic	**MUL AB**
Overflow flag is set if the product is greater than $(255)_{10}$ i.e. FFH otherwise it is cleared Carry flag is always cleared	

8051 Instruction Set – Arithmetic Instructions

Mnemonic	DIV AB
Function	Divides the unsigned number in the accumulator and with the unsigned number in register B After division: Quotient → A Remainder → B Contents of A and B, when division by 0 is attempted, are undefined There is no comma between A and B. Only registers A and B can be used Original contents of A and B are lost

8051 Instruction Set – Arithmetic Instructions

Mnemonic	**DIV AB**

Example:
DIV AB
	Let A = FBH and B = 12H
	Then DIV AB = (FBH) / (12H) will make
	
	A = $(13)_{10}$ i.e. 0DH ← quotient
	B = $(17)_{10}$ i.e. 11H ← remainder

Machine Cycles	4
Clock Pulses	48
Bytes	1
Addressing Mode	Register Addressing Mode
Flags	Flags are affected

8051 Instruction Set – Arithmetic Instructions

Mnemonic	**DIV AB**
Overflow and carry flags are always cleared But if, A contains some number and B contains 00H and if A/B is attempted then the values of A and B are undefined Overflow flag will be set and carry flag will be cleared i.e. overflow flag is set when divide by zero is attempted	

8051 Instruction Set – Arithmetic Instructions

Mnemonic	**DA A**
Function	Adjusts the sum of two packed BCD numbers to an 8 bit value i.e. producing two four bit digits To perform addition, ADD or ADDC instruction can be used Rules of BCD addition are: a) If number is greater than 9, add 6 b) If auxiliary carry or carry is generated, add 6 This is done in order to produce a valid BCD number

8051 Instruction Set – Arithmetic Instructions

Mnemonic	**DA A**
Machine Cycles	1
Clock Pulses	12
Bytes	1
Addressing Mode	Register Specific Addressing Mode
Flags	Flags are affected

8051 Instruction Set – Arithmetic Instructions

Mnemonic	**DA A**
Example: DA A	Let A = 56H (packed BCD) R3 = 67H (packed BCD) CY = 1 Then ADDC A, R3 A → 5 6 H R3 →+ 6 7 H CY→+ 0 1 B E H DA A, as B > 9, E > 9, 6 should be added to both B E H + 6 6 H 2 4 H with CY = 1

8051 Instruction Set – Arithmetic Instructions

Mnemonic	DA A

- If (A_{3-0} > 9) or AC = 1, then 6 is added to accumulator, so that it produces a proper BCD digit in lower nibble

  This internal addition may set the AC flag if there is a carry out of bit 3, propagating into higher order bits

- If (A_{7-4} > 9) or CY = 1, then 6 is added to produce proper BCD digit in high order nibble

  If there is a carry out of high order bits then carry flag will again be set

- Carry flag is set, indicates whether the sum of 2 BCD numbers is greater than 99, allowing multiple precision decimal addition. Overflow flag is not set.

8051 Instruction Set – Arithmetic Instructions

Mnemonic	**DA A**

- All the above conversions are done in one instruction cycle
- DA A does not apply to decimal subtraction
- DA A cannot simply convert a hexadecimal number in the accumulator to BCD notation

8051 Instruction Set – Logical Instructions

- Includes the following instructions:

  - ✓ANL
  - ✓ORL
  - ✓XRL
  - ✓CLR
  - ✓RL
  - ✓RR
  - ✓RRC
  - ✓RLC
  - ✓SWAP

8051 Instruction Set – Logical Instructions

- **Syntax:** ANL <dest-byte>, <src-byte>
- Performs bitwise logical AND operation between the destination and source byte
- <dest-byte> ← <dest-byte> ^ <src-byte>
- Result is stored in destination byte
- Source and destination supports 4 addressing modes: register, direct, register indirect and immediate

8051 Instruction Set – Logical Instructions

Mnemonic	**ANL A, Rn**
Function	Performs bitwise logical AND operation between the contents of accumulator and register Rn (R0 – R7) of selected register bank Result is stored in accumulator
Example: ANL A, R5	Let A = FFH and R5 = 15H Then ANL A, R5 will logically AND contents of accumulator with contents of register R5 i.e. 15H Therefore, A = 15H

8051 Instruction Set – Logical Instructions

Mnemonic	**ANL A, Rn**
Machine Cycles	1
Clock Pulses	12
Bytes	1
Addressing Mode	Register Addressing Mode
Flags	No flags are affected

8051 Instruction Set – Logical Instructions

Mnemonic	**ANL A, direct**
Function	Performs bitwise logical AND operation between the contents of accumulator and the contents of memory location whose direct address is specified Result is stored in accumulator
Example: ANL A, 50H	Let A = 10H and contents of memory location 50H = 80H Then ANL A, 50H will logically AND contents of accumulator with contents of memory location 50H i.e. 80H Therefore, A = 00H

8051 Instruction Set – Logical Instructions

Mnemonic	**ANL A, direct**
Machine Cycles	1
Clock Pulses	12
Bytes	2
Addressing Mode	Direct Addressing Mode
Flags	No flags are affected

8051 Instruction Set – Logical Instructions

Mnemonic	**ANL A, @Ri**
Function	Performs bitwise logical AND operation between the contents of accumulator and the contents of memory location pointed by register Ri (R0 or R1) of selected register bank Result is stored in accumulator
Example: ANL A, @R1	Let A = 40H and R1 = 0AH, contents of memory location 0AH = CAH Then ANL A, @R1 will logically AND contents of accumulator with contents of memory location pointed by R1 i.e. CAH Therefore, A = 40H

8051 Instruction Set – Logical Instructions

Mnemonic	**ANL A, @Ri**
Machine Cycles	1
Clock Pulses	12
Bytes	1
Addressing Mode	Register Indirect Addressing Mode
Flags	No flags are affected

8051 Instruction Set – Logical Instructions

Mnemonic	**ANL direct, A**
Function	Performs bitwise logical AND operation between the contents of memory location whose direct address is specified and contents of accumulator Here, result is stored in memory location whose direct address specified
Example: ANL 30H, A	Let A = 10H and contents of memory location 30H = 57H Then ANL 30H, A will logically AND contents of accumulator with contents of memory location whose direct address is specified i.e. 30H = 57H Therefore, 30H = 10H

8051 Instruction Set – Logical Instructions

Mnemonic	**ANL direct, A**
Machine Cycles	1
Clock Pulses	12
Bytes	2
Addressing Mode	Direct Addressing Mode
Flags	No flags are affected

8051 Instruction Set – Logical Instructions

Mnemonic	**ANL A, #data**
Function	Performs bitwise logical AND operation between the immediate data specified and contents of accumulator Here, result is stored in accumulator
Example: ANL A, #57H	Let A = 22H Then ANL A, #57H will logically AND contents of accumulator with immediate data i.e. 57H Therefore, A = 02H

8051 Instruction Set – Logical Instructions

Mnemonic	**ANL A, #data**
Machine Cycles	1
Clock Pulses	12
Bytes	2
Addressing Mode	Immediate Addressing Mode
Flags	No flags are affected

8051 Instruction Set – Logical Instructions

Mnemonic	**ANL direct, #data**
Function	Performs bitwise logical AND operation between the immediate data specified and contents of memory location whose direct address is specified Here, result is stored in memory location whose direct address is specified
Example: ANL 54H, #33H	Let contents of memory location 54H = 25H Then ANL 54H, #33H will logically AND contents of memory location 54H i.e. 25H with immediate data i.e. 33H Therefore result at 54H = 21H

8051 Instruction Set – Logical Instructions

Mnemonic	**ANL direct, #data**
Machine Cycles	2
Clock Pulses	24
Bytes	3
Addressing Mode	Immediate Addressing Mode
Flags	No flags are affected

8051 Instruction Set – Logical Instructions

- **Syntax:** ORL <dest-byte>, <src-byte>
- Performs bitwise logical OR operation between the destination and source byte
- <dest-byte> ← <dest-byte> v <src-byte>
- Result is stored in destination byte
- Source and destination supports 4 addressing modes: register, direct, register indirect and immediate

8051 Instruction Set – Logical Instructions

Mnemonic	**ORL A, Rn**
Function	Performs bitwise logical OR operation between the contents of accumulator and register Rn (R0 – R7) of selected register bank Result is stored in accumulator
Example: ORL A, R5	Let A = FFH and R5 = 15H Then ORL A, R5 will logically OR contents of accumulator with contents of register R5 i.e. 15H Therefore, A = FFH

8051 Instruction Set – Logical Instructions

Mnemonic	**ORL A, Rn**
Machine Cycles	1
Clock Pulses	12
Bytes	1
Addressing Mode	Register Addressing Mode
Flags	No flags are affected

8051 Instruction Set – Logical Instructions

Mnemonic	**ORL A, direct**
Function	Performs bitwise logical OR operation between the contents of accumulator and the contents of memory location whose direct address is specified Result is stored in accumulator
Example: ORL A, 50H	Let A = 10H and contents of memory location 50H = 80H Then ORL A, 50H will logically OR contents of accumulator with contents of memory location 50H i.e. 80H Therefore, A = 90H

8051 Instruction Set – Logical Instructions

Mnemonic	**ORL A, direct**
Machine Cycles	1
Clock Pulses	12
Bytes	2
Addressing Mode	Direct Addressing Mode
Flags	No flags are affected

8051 Instruction Set – Logical Instructions

Mnemonic	**ORL A, @Ri**
Function	Performs bitwise logical OR operation between the contents of accumulator and the contents of memory location pointed by register Ri (R0 or R1) of selected register bank Result is stored in accumulator
Example: ORL A, @R1	Let A = 40H and R1 = 0AH, contents of memory location 0AH = CAH Then ORL A, @R1 will logically OR the contents of accumulator with contents of memory location pointed by R1 i.e. CAH Therefore, A = CAH

8051 Instruction Set – Logical Instructions

Mnemonic	**ORL A, @Ri**
Machine Cycles	1
Clock Pulses	12
Bytes	1
Addressing Mode	Register Indirect Addressing Mode
Flags	No flags are affected

8051 Instruction Set – Logical Instructions

Mnemonic	**ORL direct, A**
Function	Performs bitwise logical OR operation between the contents of memory location whose direct address is specified and contents of accumulator Here, result is stored in memory location whose direct address specified
Example: ORL 30H, A	Let A = 64H and contents of memory location 30H = 57H Then ORL 30H, A will logically OR contents of accumulator i.e. 64H with contents of memory location whose direct address is specified i.e. 30H = 57H Therefore, 30H = 77H

8051 Instruction Set – Logical Instructions

Mnemonic	**ORL direct, A**
Machine Cycles	1
Clock Pulses	12
Bytes	2
Addressing Mode	Direct Addressing Mode
Flags	No flags are affected

8051 Instruction Set – Logical Instructions

Mnemonic	**ORL A, #data**
Function	Performs bitwise logical OR operation between the immediate data specified and contents of accumulator Here, result is stored in accumulator
Example: ORL A, #57H	Let A = 22H Then ORL A, #57H will logically OR contents of accumulator with immediate data i.e. 57H Therefore, A = 77H

8051 Instruction Set – Logical Instructions

Mnemonic	**ORL A, #data**
Machine Cycles	1
Clock Pulses	12
Bytes	2
Addressing Mode	Immediate Addressing Mode
Flags	No flags are affected

8051 Instruction Set – Logical Instructions

Mnemonic	**ORL direct, #data**
Function	Performs bitwise logical OR operation between the immediate data specified and contents of memory location whose direct address is specified Here, result is stored in memory location whose direct address is specified
Example: ORL 54H, #33H	Let contents of memory location 54H = 25H Then ORL 54H, #33H will logically OR contents of memory location 54H i.e. 25H with immediate data i.e. 33H Therefore result at 54H = 37H

8051 Instruction Set – Logical Instructions

Mnemonic	**ORL direct, #data**
Machine Cycles	2
Clock Pulses	24
Bytes	3
Addressing Mode	Immediate Addressing Mode
Flags	No flags are affected

8051 Instruction Set – Logical Instructions

- **Syntax:** XRL <dest-byte>, <src-byte>
- Performs bitwise Exclusive-OR operation between the destination and source byte
- <dest-byte> $\leftarrow$ <dest-byte> EXOR <src-byte>
- Result is stored in destination byte
- Source and destination supports 4 addressing modes: register, direct, register indirect and immediate

8051 Instruction Set – Logical Instructions

Mnemonic	**XRL A, Rn**
Function	Performs bitwise logical Exclusive-OR operation between the contents of accumulator and contents of register Rn (R0 – R7) of selected register bank Result is stored in accumulator
Example: XRL A, R5	Let A = FFH and R5 = 15H Then XRL A, R5 will logically EX-OR contents of accumulator with contents of register R5 i.e. 15H Therefore, A = EAH

8051 Instruction Set – Logical Instructions

Mnemonic	**XRL A, Rn**
Machine Cycles	1
Clock Pulses	12
Bytes	1
Addressing Mode	Register Addressing Mode
Flags	No flags are affected

8051 Instruction Set – Logical Instructions

Mnemonic	**XRL A, direct**
Function	Performs bitwise logical Exclusive-OR operation between the contents of accumulator and the contents of memory location whose direct address is specified Result is stored in accumulator
Example: XRL A, 50H	Let A = 10H and contents of memory location 50H = 80H Then XRL A, 50H will logically EX-OR contents of accumulator with contents of memory location 50H i.e. 80H Therefore, A = 90H

8051 Instruction Set – Logical Instructions

Mnemonic	**XRL A, direct**
Machine Cycles	1
Clock Pulses	12
Bytes	2
Addressing Mode	Direct Addressing Mode
Flags	No flags are affected

8051 Instruction Set – Logical Instructions

Mnemonic	**XRL A, @Ri**
Function	Performs bitwise logical Exclusive-OR operation between the contents of accumulator and the contents of memory location pointed by register Ri (R0 or R1) of selected register bank Result is stored in accumulator
Example: XRL A, @R1	Let A = 40H and R1 = 0AH, contents of memory location 0AH = CAH Then XRL A, @R1 will logically EX-OR the contents of accumulator with contents of memory location pointed by R1 i.e. CAH Therefore, A = 8AH

8051 Instruction Set – Logical Instructions

Mnemonic	**XRL A, @Ri**
Machine Cycles	1
Clock Pulses	12
Bytes	1
Addressing Mode	Register Indirect Addressing Mode
Flags	No flags are affected

8051 Instruction Set – Logical Instructions

Mnemonic	**XRL direct, A**
Function	Performs bitwise logical Exclusive-OR operation between the contents of memory location whose direct address is specified and contents of accumulator Here, result is stored in memory location whose direct address specified
Example: XRL 30H, A	Let A = 64H and contents of memory location 30H = 57H Then XRL 30H, A will logically EX-OR contents of accumulator i.e. 64H with contents of memory location whose direct address is specified i.e. 30H = 57H Therefore, 30H = 33H

8051 Instruction Set – Logical Instructions

Mnemonic	**XRL direct, A**
Machine Cycles	1
Clock Pulses	12
Bytes	2
Addressing Mode	Direct Addressing Mode
Flags	No flags are affected

8051 Instruction Set – Logical Instructions

Mnemonic	**XRL A, #data**
Function	Performs bitwise logical Exclusive-OR operation between the immediate data specified and contents of accumulator Here, result is stored in accumulator
Example: XRL A, #57H	Let A = 22H Then XRL A, #57H will logically EX-OR contents of accumulator i.e. 22H with immediate data i.e. 57H Therefore, A = 75H

8051 Instruction Set – Logical Instructions

Mnemonic	**XRL A, #data**
Machine Cycles	1
Clock Pulses	12
Bytes	2
Addressing Mode	Immediate Addressing Mode
Flags	No flags are affected

8051 Instruction Set – Logical Instructions

Mnemonic	**XRL direct, #data**
Function	Performs bitwise logical Exclusive-OR operation between the immediate data specified and contents of memory location whose direct address is specified Here, result is stored in memory location whose direct address is specified
Example: XRL 54H, #33H	Let contents of memory location 54H = 25H Then XRL 54H, #33H will logically EX-OR contents of memory location 54H i.e. 25H with immediate data i.e. 33H Therefore result at 54H = 16H

8051 Instruction Set – Logical Instructions

Mnemonic	**XRL direct, #data**
Machine Cycles	2
Clock Pulses	24
Bytes	3
Addressing Mode	Immediate Addressing Mode
Flags	No flags are affected

8051 Instruction Set – Logical Instructions

Mnemonic	**CLR A**
Function	Clears all the bits of the accumulator to zero
Example: CLR A	Let A = 77H. Then CLR A will leave the contents of accumulator to 00H
Machine Cycles	1
Clock Pulses	12
Bytes	1
Addressing Mode	Register Specific Addressing Mode
Flags	No flags are affected

8051 Instruction Set – Logical Instructions

Mnemonic	**CPL A**
Function	Complements all the bits of accumulator i.e. 1’s complement of the number in accumulator. 1’s are changed to 0’s and vice versa
Example: CPL A	Let A = 57H = (0101 0111 B). Then CPL A will complement all bits in accumulator . So accumulator will now contain, A = 1010 1000 = A8H
Machine Cycles	1
Clock Pulses	12
Bytes	1
Addressing Mode	Register Specific Addressing Mode
Flags	No flags are affected

8051 Instruction Set – Logical Instructions

Mnemonic	**RL A**
Function	Rotates the 8 bits in the accumulator by one bit to the left Bit 7 is rotated into bit 0 position
Example: RL A	Let A = 58H = (0101 1000 B). Then RL A will rotate the bits in accumulator by one bit to the left. So now A = (1011 0000 B) = B0H
Machine Cycles	1
Clock Pulses	12
Bytes	1
Addressing Mode	Register Specific Addressing Mode
Flags	No flags are affected

8051 Instruction Set – Logical Instructions

Mnemonic	**RLC A**
Function	Rotates the 8 bits in the accumulator as well as the contents of carry flag together by one bit to the left Bit 7 will move into the carry flag and original carry will move into bit 0 position
Example: RLC A	Let A = 72H = (0111 0010 B) and CY = 1. Then RLC A will move the contents of accumulator along with the carry to the left by 1 bit So now A = 1110 0101 = E5H and CY = 0

8051 Instruction Set – Logical Instructions

Mnemonic	**RLC A**
Machine Cycles	1
Clock Pulses	12
Bytes	1
Addressing Mode	Register Specific Addressing Mode
Flags	No flags are affected

8051 Instruction Set – Logical Instructions

Mnemonic	**RR A**
Function	Rotates the 8 bits in the accumulator by one bit to the right Bit 0 is rotated into bit 7 position
Example: RR A	Let A = 75H = (0111 0101 B). Then RR A will rotate the bits in accumulator by one bit to the right. So now A = (1011 1010 B) = BAH
Machine Cycles	1
Clock Pulses	12
Bytes	1
Addressing Mode	Register Specific Addressing Mode
Flags	No flags are affected

8051 Instruction Set – Logical Instructions

Mnemonic	**RRC A**
Function	Rotates the 8 bits in the accumulator as well as the contents of carry flag together by one bit to the right Bit 0 will move into the carry flag and original carry will move into bit 7 position
Example: RRC A	Let A = 85H = (1000 0101 B) and CY = 1. Then RRC A will move the contents of accumulator along with the carry to the right by 1 bit So now A = 1100 0010 = C2H and CY = 1

8051 Instruction Set – Logical Instructions

Mnemonic	**RRC A**
Machine Cycles	1
Clock Pulses	12
Bytes	1
Addressing Mode	Register Specific Addressing Mode
Flags	No flags are affected

8051 Instruction Set – Logical Instructions

Mnemonic	**SWAP A**
Function	Interchanges the low order and high order nibbles of the accumulator Can be used as a 4 bit rotate operation Belongs to register specific addressing mode
Example: SWAP A	Let A = 87H = (1000 0111 B) Then SWAP A will swap the lower order byte with the higher order byte in the accumulator So now A = 0111 1000 = 78H

8051 Instruction Set – Bit Level (Boolean Instructions

- Includes the following instructions:
  - ✓CLR
  - ✓SETB
  - ✓CPL
  - ✓ANL
  - ✓ORL
  - ✓MOV

8051 Instruction Set – Boolean Instructions

Mnemonic	**CLR bit**
Function	Clears the indicated bit CLR can operate on carry flag or any directly addressable bit
Example: CLR P2.3	Let Port 2 be previously been written with ADH = (1010 1101 B) Then CLR P2.3 will clear the 3rd bit of Port 2 leaving Port 2 = (1010 0101 B) = A5H

8051 Instruction Set – Boolean Instructions

Mnemonic	**CLR bit**
Machine Cycles	1
Clock Pulses	12
Bytes	1 → if carry specific 2 → if directly addressable bit
Addressing Mode	If operated on carry flag → Register Addressing Mode Otherwise → Direct addressing mode
Flags	If carry specific → Carry flag is affected If direct →No flags are affected

8051 Instruction Set – Boolean Instructions

Mnemonic	**SETB bit**
Function	Sets the indicated bit SETB can operate on carry flag or any directly addressable bit
Example: SETB P2.3	Let Port 2 be previously been written with A5H = (1010 0101 B) Then SETB P2.3 will set the 3rd bit of Port 2 leaving Port 2 = (1010 1101 B) = ADH

8051 Instruction Set – Boolean Instructions

Mnemonic	SETB bit
Machine Cycles	1
Clock Pulses	12
Bytes	1 → if carry specific 2 → if directly addressable bit
Addressing Mode	If operated on carry flag → Register Addressing Mode Otherwise → Direct addressing mode
Flags	If carry specific → Carry flag is affected If direct →No flags are affected

8051 Instruction Set – Boolean Instructions

Mnemonic	**CPL bit**
Function	Complements the bit variable specified CPL can operate on carry flag or any directly addressable bit
Example: CPL P2.5	Let Port 2 be previously been written with A5H = (1010 0101 B) Then CPL P2.5 will complement the 5th bit of Port 2 leaving Port 2 = (1000 1101 B) = 85H

8051 Instruction Set – Boolean Instructions

Mnemonic	**CPL bit**
Machine Cycles	1
Clock Pulses	12
Bytes	1 → if carry specific 2 → if directly addressable bit
Addressing Mode	If operated on carry flag → Register Addressing Mode Otherwise → Direct addressing mode
Flags	If carry specific → Carry flag is affected If direct →No flags are affected

8051 Instruction Set – Boolean Instructions

Mnemonic	**ANL C, <src-bit>**
Function	Logically AND the specified bit with the carry bit Result is stored in the carry bit If the boolean value of the source bit is 0, then this instruction will clear the carry flag If the boolean value of the source bit is 1, then it will leave the carry flag in its current state i.e. contents of carry flag will be preserved

8051 Instruction Set – Boolean Instructions

Mnemonic	**ANL C, <src-bit>**
Example: ANL C, ACC.4	Let C = 1, A = 11H = (0001 0001 B) then ANL C, ACC.4 will logically AND the contents of carry with bit 4 in the accumulator Then C = 1
ANL C, /ACC.4 Let C = 1 and A = 11H = (0001 0001 B)	Slash (/) preceding the operand indicates that the logical complement of the addressed bit is used as the source value, but the source bit itself is unaffected Here, ANL C, /ACC.4 will logically AND the contents of carry i.e. 1 with the complement of the 4th bit i.e. 0 in accumulator So now C = 0

8051 Instruction Set – Boolean Instructions

Mnemonic	**ANL C, <src-bit>**
Machine Cycles	2
Clock Pulses	24
Bytes	2
Addressing Mode	Direct addressing mode
Flags	Except carry, no other flags are affected

8051 Instruction Set – Boolean Instructions

Mnemonic	**ORL C, <src-bit>**
Function	Logically OR the specified bit with the carry bit Result is stored in the carry bit If the boolean value of the source bit is 1, then this instruction will set the carry flag If the boolean value of the source bit is 0, then it will leave the carry flag in its current state i.e. contents of carry flag will be preserved

8051 Instruction Set – Boolean Instructions

Mnemonic	**ORL C, <src-bit>**
Example: ORL C, ACC.4	Let C = 1, A = 11H = (0001 0001 B) then ORL C, ACC.4 will logically OR the contents of carry with bit 4 in the accumulator Then C = 1
ORL C, /ACC.4 Let C = 1 and A = 11H = (0001 0001 B)	Slash (/) preceding the operand indicates that the logical complement of the addressed bit is used as the source value, but the source bit itself is unaffected Here, ORL C, /ACC.4 will logically OR the contents of carry i.e. 1 with the complement of the 4th bit i.e. 0 in accumulator So now C = 1

8051 Instruction Set – Boolean Instructions

Mnemonic	**ORL C, <src-bit>**
Machine Cycles	2
Clock Pulses	24
Bytes	2
Addressing Mode	Direct addressing mode
Flags	No flags are affected

8051 Instruction Set – Boolean Instructions

- **Syntax:** MOV <dest-bit>, <src-bit>
- Copy the source bit to the destination bit
- One of the operands must be carry flag and the other operand may be any directly addressable bit

8051 Instruction Set – Boolean Instructions

Mnemonic	**MOV bit, C**
Function	Copy the carry flag contents into the boolean variable whose address is specified
Example: MOV ACC.3, C	Let A = 11H = (0001 0001 B) and C = 1. Then MOV ACC.3, C will copy the contents of carry to 3rd bit of accumulator So now A = (0001 1001 B) = 19H
Machine Cycles	2
Clock Pulses	24
Bytes	2
Addressing Mode	Direct Addressing Mode
Flags	No flags are affected

8051 Instruction Set – Boolean Instructions

Mnemonic	**MOV C, bit**
Function	Copy the data from boolean variable whose address is specified to the carry flag
Example: MOV C, ACC.4	Let A = 01H = (0000 0001 B) and C = 1. Then MOV C, ACC.4 will copy the contents of 3rd bit of accumulator to carry flag So now C = 0
Machine Cycles	1
Clock Pulses	12
Bytes	2
Addressing Mode	Direct Addressing Mode
Flags	Except carry, no other flags are affected

8051 Instruction Set – Branching Instructions

- Includes the following instructions:
  - ✓ ACALL
  - ✓ LCALL
  - ✓ RET
  - ✓ RETI
  - ✓ AJMP
  - ✓ LJMP
  - ✓ SJMP
  - ✓ JMP
  - ✓ JZ
  - ✓ JNZ
  - ✓ JC
  - ✓ JNC
  - ✓ JB
  - ✓ JNB
  - ✓ JBC
  - ✓ CJNE
  - ✓ DJNZ
  - ✓ NOP

8051 Instruction Set – Branching Instructions

JUMP instructions	CALL instructions
Program control is transferred to a memory location which is in the main program	Program Control is transferred to a memory location which is not a part of main program
Initialisation of SP(Stack Pointer) is not mandatory	Initialisation of SP(Stack Pointer) is mandatory
Value of Program Counter(PC) is not transferred to stack	Value of Program Counter(PC) is transferred to stack
After JUMP, there is no RET instruction	After CALL, there is a RET instruction
Value of SP does not changes	Value of SP is decremented by 2
Does not store the status of PSW, A and other registers onto the stack before executing JUMP instruction	Does not store the status of PSW, A and other registers onto the stack before executing CALL instruction

8051 Instruction Set – Branching Instructions

- **Syntax:** ACALL addr11
- 2 byte instruction, that unconditionally calls a subroutine at the indicated address
- At the end of subroutine the program will resume operation at the opcode address following the call instruction
- Return address of the next instruction after the call instruction is in the program counter
- Subroutine that is called must be located in the same 2KByte block of program memory
- It can be called a number of times in the same program

8051 Instruction Set – Branching Instructions

❖ **Steps followed while executing ACALL instruction:**

Step 1:

✓PC increments twice in order to obtain the address of the next instruction after CALL

Step 2:

✓It then pushes the 16 bit result onto the stack.

✓Initially SP increments by 1 and then the low order byte of PC is pushed onto the stack

✓Then SP again increments by 1 and now the high order byte of PC is pushed onto the stack

8051 Instruction Set – Branching Instructions

- **Steps followed while executing ACALL instruction:**

Step 3:

- ✓ Destination address i.e. address of subroutine is computed by concatenating the high order 5 bits of the incremented PC i.e. PC $_{15\text{-}8}$, 3 bits (A_{10}, A_9, A_8) from the first byte of instruction and the 8 bits (A_7 to A_0) from second byte of instruction

8051 Instruction Set – Branching Instructions

Mnemonic	**ACALL addr11 (absolute call)**
Example: ACALL add	Let SP = 0AH and PC = 0239H Label "add" is at program memory location 0435H After execution of instruction, ACALL add at location 0237H: i) SP will contain 0CH ii) Internal RAM location 0BH → 39H iii) Location 0CH → 02H iv) PC → 0435H

8051 Instruction Set – Branching Instructions

Mnemonic	**ACALL add**
Calculation of destination address: PC = 0239H = 0000 0010 0011 1001 Memory location of label “add” → 0435H i.e. 0000 0100 0011 0101 High order 5 bits of PC → 00000 3 bits from 1st byte of instruction i.e. from memory location of label “add” (A_{10}, A_9, A_8) → 100 8 bits from 2nd byte of instruction i.e. from memory location of label “add” (A_7 to A_0) →0011 0101 Destination address = PC = 0000 0100 0011 0101 = 0435H	

8051 Instruction Set – Branching Instructions

Mnemonic	ACALL addr11 (absolute call)
Machine Cycles	2
Clock Pulses	24
Bytes	2

8051 Instruction Set – Branching Instructions

- **Syntax:** LCALL addr16 (Long Call)
- 3 byte instruction
- Calls a subroutine at the specified address
- Allows to jump to a subroutine anywhere in the 64KB code space.
- **Steps followed while executing LCALL instruction are:**

Step 1:

✓PC increments thrice in order to generate the address of the next instruction

8051 Instruction Set – Branching Instructions

Step 2:

- ✓It then pushes the 16 bit result onto the stack.
- ✓Initially SP increments by 1 and then the low order byte of PC is pushed onto the stack
- ✓Then SP again increments by 1 and now the high order byte of PC is pushed onto the stack

Step 3:

- ✓PC (location of subroutine) will be loaded with the second and third bytes of the instruction i.e. 16 bits so it can access upto 64KB of code space
- ✓Program execution is transferred to the subroutine at this address

8051 Instruction Set – Branching Instructions

Mnemonic	**LCALL addr16**
Example: LCALL add	Let SP = 07H and PC = 023AH Label “add” is at program memory location 0435H After execution of instruction, LCALL add at location 0237H: i) SP will contain 09H ii) Internal RAM location 08H → 3AH iii) Location 09H → 02H iv) PC → 0435H

8051 Instruction Set – Branching Instructions

Mnemonic	**LCALL addr16**
Machine Cycles	2
Clock Pulses	24
Bytes	3

8051 Instruction Set – Branching Instructions

- **Syntax:** RET (Return from subroutine)
- This instruction informs the MC to return back to the program, from where the subroutine was called
- This instruction pops the high and low order bytes of PC (initially stored on stack) successively from the stack. Stack pointer is decremented by 2
- Program execution will resume from the resulting address, generally the address of the next instruction that follows the LCALL or CALL instruction

8051 Instruction Set – Branching Instructions

Mnemonic	**RET (Return from subroutine)**
Example: RET	Let SP = 09H Internal RAM locations 08H and 09H contain 53H and 27H Instruction RET will leave the stack pointer equal to the value 07H Program execution will continue at location 2753H
Machine Cycles	2
Clock Pulses	24
Bytes	1

8051 Instruction Set – Branching Instructions

- **Syntax:** RETI (Return from Interrupt)
- Used to end the Interrupt Service Routine
- This instruction pops the high and low order bytes of PC successively from the stack and restores the interrupt logic i.e. TF0, TF1, IE0 and IE1 to accept additional interrupts at the same priority levels as the one just processed. Stack pointer is decremented by 2

8051 Instruction Set – Branching Instructions

- Program execution will resume from the instruction that follows the point at which the interrupt request was detected

- If another interrupt was pending when the **RETI** instruction is executed, one instruction at the return address is executed before the pending interrupt is processed

- Does not restore the contents of PSW register to its value before the interrupt

- ISR must save and restore the PSW

8051 Instruction Set – Branching Instructions

Mnemonic	**RETI (Return from interrupt)**
Example: RETI	Let SP = 09H Assume RETI is detected at location 0322H Internal RAM locations (i.e. SP) 08H and 09H contain 57H and 12H Instruction RET will leave the stack pointer equal to the value 07H and returns the program execution to location 1257H i.e. PC = 1257H
Machine Cycles	2
Clock Pulses	24
Bytes	1

8051 Instruction Set – Branching Instructions

- **Syntax:** AJMP addr11 (Absolute Jump)
- 3 byte instruction
- Transfers the program execution to the indicated address i.e. AJMP label
- Limitation is that the label must be within the same 2KB block of program memory
- Destination (jump) address is calculated by concatenating the high order 5 bits of PC i.e. PC $_{15-11}$ after the PC is incremented twice, 3 higher order bits (A_{10}, A_9, A_8) from the first byte of instruction and the 8 bits (A_7 to A_0) from second byte of instruction

8051 Instruction Set – Branching Instructions

Mnemonic	**AJMP addr11 (Absolute Jump)**
Example: AJMP L7	Let label “L7” be at program memory location 0537H Instruction AJMP L7 is at location 0671H and will load the PC with 0537H
Machine Cycles	2
Clock Pulses	24
Bytes	2

8051 Instruction Set – Branching Instructions

Mnemonic	**AJMP L7**

Calculation of destination address: AJMP L7 is at location 0671H
PC = 0673H = 0000 0110 0111 0011

Memory location of label "L7" → 0537H = 0000 0101 0011 0111

High order 5 bits of PC → 00000

3 bits from 1st byte of instruction i.e. from memory location of label "L7" (A_{10}, A_9, A_8) → 101

8 bits from 2nd byte of instruction i.e. from memory location of label "L7" (A_7 to A_0) →0011 0111

Destination address = PC = 0000 0101 0011 0111 = 0537H

8051 Instruction Set – Branching Instructions

- **Syntax:** LJMP addr16 (Long Jump)

- Causes an unconditional branch to the indicated address, by loading the high order and low order bytes of PC respectively, with the second and third instruction bytes

- Destination can thus be anywhere in the full 64KB program memory address space, because it uses full 16 bits of two bytes i.e. 2nd and 3rd instruction bytes

8051 Instruction Set – Branching Instructions

Mnemonic	**LJMP addr16 (Long Jump)**
Example: AJMP L7	Let label “L7” be at program memory location 5678H Instruction LJMP L7 is at location 0537H and will load the PC with 5678H
Machine Cycles	2
Clock Pulses	24
Bytes	3

8051 Instruction Set – Branching Instructions

- **Syntax:** SJMP rel (Short Jump)
- Program control branches unconditionally to the specified address
- Branch destination is computed by adding the signed i.e. relative displacement in the second instruction byte to the PC, after incrementing PC twice
- Only limitation is that the jump range is limited from -128 to +127 bytes
- Destination range allowed is from 128 bytes preceding this instruction to 127 bytes following it

8051 Instruction Set – Branching Instructions

Mnemonic	**SJMP rel (Short Jump)**
Example: SJMP L7	Let label "L7" be assigned to an instruction whose program memory location is 0478H Instruction SJMP L7 assembles into location 0401H . After execution of this instruction the PC will contain 0478H
Machine Cycles	2
Clock Pulses	24
Bytes	2

8051 Instruction Set – Branching Instructions

- **Syntax:** JMP @A + DPTR (Jump Indirect)
- Adds the 8 bit unsigned contents of the accumulator with the contents of 16 bit data pointer
- Result of addition is loaded into Program Counter
- This is the address from where the microcontroller will begin execution
- Neither the DPTR nor the accumulator contents are altered

8051 Instruction Set – Branching Instructions

Mnemonic	**JMP @A + DPTR**
Example: SJMP L7	Let A = 30H, DPTR = 1000H and PC = 0500H JMP @A + DPTR will make PC = 1030H and the execution will begin from location 1030H instead of location 0500H
Machine Cycles	2
Clock Pulses	24
Bytes	1

8051 Instruction Set – Branching Instructions

- **Syntax:** JZ rel (Jump if accumulator is zero)

- Jumps to the indicated address if all the bits in the accumulator are 0, otherwise it will continue with the next instruction

- Branch destination is calculated by adding the signed relative displacement in the second instruction byte to the PC, after incrementing PC by 2

- JZ rel

  PC ← PC + 2

  if A = 0 then PC ← PC + rel

- Accumulator remains unchanged

8051 Instruction Set – Branching Instructions

Mnemonic	**JZ rel**
Example: JZ label	Let A = 00H JZ label will cause the program execution to continue at the instruction identified by label
Machine Cycles	2
Clock Pulses	24
Bytes	2

8051 Instruction Set – Branching Instructions

- **Syntax:** JNZ rel (Jump if accumulator is not zero)

- Jumps to the indicated address if the bits in the accumulator are not 0, otherwise it will continue with the next instruction

- Branch destination is calculated by adding the signed relative displacement in the second instruction byte to the PC, after incrementing PC by 2

- JNZ rel

  PC $\leftarrow$ PC + 2

  if A $\neq$ 0 then PC $\leftarrow$ PC + rel

- Accumulator remains unchanged

8051 Instruction Set – Branching Instructions

Mnemonic	**JNZ rel**
Example: JNZ label	Let A = 05H JNZ label will cause the program execution to continue at the instruction identified by label
Machine Cycles	2
Clock Pulses	24
Bytes	2

8051 Instruction Set – Branching Instructions

- **Syntax:** JC rel (Jump if carry is set)
- Jumps to the indicated address if the carry flag is set, otherwise it will continue with the next instruction
- Branch destination is calculated by adding the signed relative displacement in the second instruction byte to the PC, after incrementing PC by 2
- JC rel

  PC $\leftarrow$ PC + 2

  if CY = 1 then PC $\leftarrow$ PC + rel

8051 Instruction Set – Branching Instructions

Mnemonic	**JC rel**
Example: JC label	Let CY = 1 JC label will cause the program execution to continue at the instruction identified by label
Machine Cycles	2
Clock Pulses	24
Bytes	2

8051 Instruction Set – Branching Instructions

- **Syntax:** JNC rel (Jump if carry is not set)

- Jumps to the indicated address if the carry flag is not set, otherwise it will continue with the next instruction

- Branch destination is calculated by adding the signed relative displacement in the second instruction byte to the PC, after incrementing PC by 2

- JNC rel

  PC $\leftarrow$ PC + 2

  if CY $\neq$ 1 i.e. CY = 0, then PC $\leftarrow$ PC + rel

8051 Instruction Set – Branching Instructions

Mnemonic	**JNC rel**
Example: JNC label	Let CY = 1 Then the instruction sequence JNC L1 CPL C JNC L2 Will clear the carry flag and program execution will resume from the instruction identified by label L2
Machine Cycles	2
Clock Pulses	24
Bytes	2

8051 Instruction Set – Branching Instructions

- **Syntax:** JB bit, rel (Jump if bit is set)
- Jumps to the indicated address, if the indicated bit in the instruction is 1, otherwise program will continue with the next instruction
- Branch destination is calculated by adding the signed relative displacement in the third instruction byte to the PC, after the PC is incremented to the first byte of next instruction
- Bit indicated is unchanged.
- JB bit, rel

  PC $\leftarrow$ PC + 3

  if bit = 1 then PC $\leftarrow$ PC + rel

8051 Instruction Set – Branching Instructions

Mnemonic	**JB bit, rel**
Example: JB P2.1, L1	Let Port 2 = 73H = (0111 0011 B) The instruction JB P2.1, L1 will cause the program execution to jump to the instruction at label L1
Machine Cycles	2
Clock Pulses	24
Bytes	3

8051 Instruction Set – Branching Instructions

- **Syntax:** JNB bit, rel (Jump if bit is not set)

- Jumps to the indicated address, if the indicated bit in the instruction is 0, otherwise program will continue with the next instruction

- Branch destination is calculated by adding the signed relative displacement in the third instruction byte to the PC, after the PC is incremented to the first byte of next instruction

- Bit indicated is unchanged.

- JB bit, rel

  PC $\leftarrow$ PC + 3

  if bit = 0 then PC $\leftarrow$ PC + rel

8051 Instruction Set – Branching Instructions

Mnemonic	**JNB bit, rel**
Example: JNB P1.3, L3	Let Port 2 = 73H = (0111 0011 B) The instruction JNB P1.3, L3 will cause the program execution to jump to the instruction at label L3
Machine Cycles	2
Clock Pulses	24
Bytes	3

8051 Instruction Set – Branching Instructions

- **Syntax:** JBC bit, rel (Jump if bit is set and clear bit)
- Jumps to the indicated address, if the indicated bit in the instruction is 1, otherwise program will continue with the next instruction
- Branch destination is calculated by adding the signed relative displacement in the third instruction byte to the PC, after the PC is incremented to the first byte of next instruction
- Content of indicated bit are changed to 0 if the bit is set
- JBC bit, rel

  PC $\leftarrow$ PC + 3

  if bit = 1 then bit = 0 and PC $\leftarrow$ PC + rel

8051 Instruction Set – Branching Instructions

Mnemonic	**JBC bit, rel**
Example: JBC P2.0, L1	Let Port 2 = 75H = (0111 0101 B) The instruction JBC P2.0, L1 will cause the program execution to jump to the instruction at label L1 and port 2 = 74H = (0111 0100 B)
Machine Cycles	2
Clock Pulses	24
Bytes	3

8051 Instruction Set – Branching Instructions

- **Syntax:** CJNE <dest-byte>, <src-byte>, rel
- Compare and jump if not equal
- Compares the magnitudes of source bytes and destination byte
- If their values are unequal then it jumps to the indicated address otherwise program execution continues from the next instruction
- Source and destination bytes are not altered
- Destination is calculated by adding the signed relative displacement in the third instruction byte to PC after the PC is incremented to point to first byte of next instruction

8051 Instruction Set – Branching Instructions

- **Syntax:** CJNE <dest-byte>, <src-byte>, rel

- Carry flag is set if destination byte is less than the source byte, otherwise carry flag is cleared

- Algorithm:

PC ← PC + 3

If (dest-byte) ≠ (src-byte)

Then PC ← PC + rel

If (dest-byte) < (src-byte)

Then C = 1 else C = 0

8051 Instruction Set – Branching Instructions

- **Syntax:** CJNE <dest-byte>, <src-byte>, rel

- Possible combinations:

  - ✓CJNE @Rn, #immediate, offset
  - ✓CJNE A, #immediate, offset
  - ✓CJNE A, direct, offset
  - ✓CJNE Rn, #immediate, offset

8051 Instruction Set – Branching Instructions

Mnemonic	**CJNE <dest-byte>, <src-byte>, rel**
Example: CJNE A, 60H, L5	Let A = 75H and contents of memory location 60H = 44H Instruction CJNE A, 60H, L5 will clear the carry flag as contents of A > contents of memory location 60H and jump to the instruction at label L5
Machine Cycles	2
Clock Pulses	24
Bytes	3

8051 Instruction Set – Branching Instructions

- **Syntax:** DJNZ <byte>, <rel-add>

- Decrements the specified register or memory location by 1 and branches to the address indicated by the second operand if the resulting value is non zero

- Destination is calculated by adding the signed relative displacement value in the last instruction byte to the PC, after incrementing PC to the first byte of next instruction

- Original value of 00H will underflow to FFH if decremented

8051 Instruction Set – Branching Instructions

- **Syntax:** DJNZ <byte>, <ret-addr>

- Algorithm:

$PC \leftarrow PC + 2$

$byte = byte - 1$

If $(byte) \neq 0$

Then $PC \leftarrow PC + rel$

8051 Instruction Set – Branching Instructions

Mnemonic	**DJNZ <byte>, <ret-addr>**
Example: DJNZ R5, SUB	Let R5 = 50H DJNZ R5, SUB will cause the control to jump to the instruction at label SUB and R5 = 4FH
Machine Cycles	2
Clock Pulses	24
Bytes	2 if register addressing is used 3 if direct addressing is used

8051 Instruction Set – Branching Instructions

- **Syntax:** NOP
- No operation is performed
- Only the PC is affected
- Contents are incremented by 1
- Mainly used to insert delay in program

8051 Instruction Set – Branching Instructions

Mnemonic	**NOP**
Machine Cycles	1
Clock Pulses	12
Bytes	1

- WAP to generate frequencies of 2KHz and 10KHz on pins P0.0 and P0.1 respectively. Assume crystal frequency as 12MHz. Use interrupt logic and timers in mode 2.

- Solution: Timer clock frequency = 1MHz

For wave of 2KHz,

On time = 0.25 msec, Off time = 0.25msec

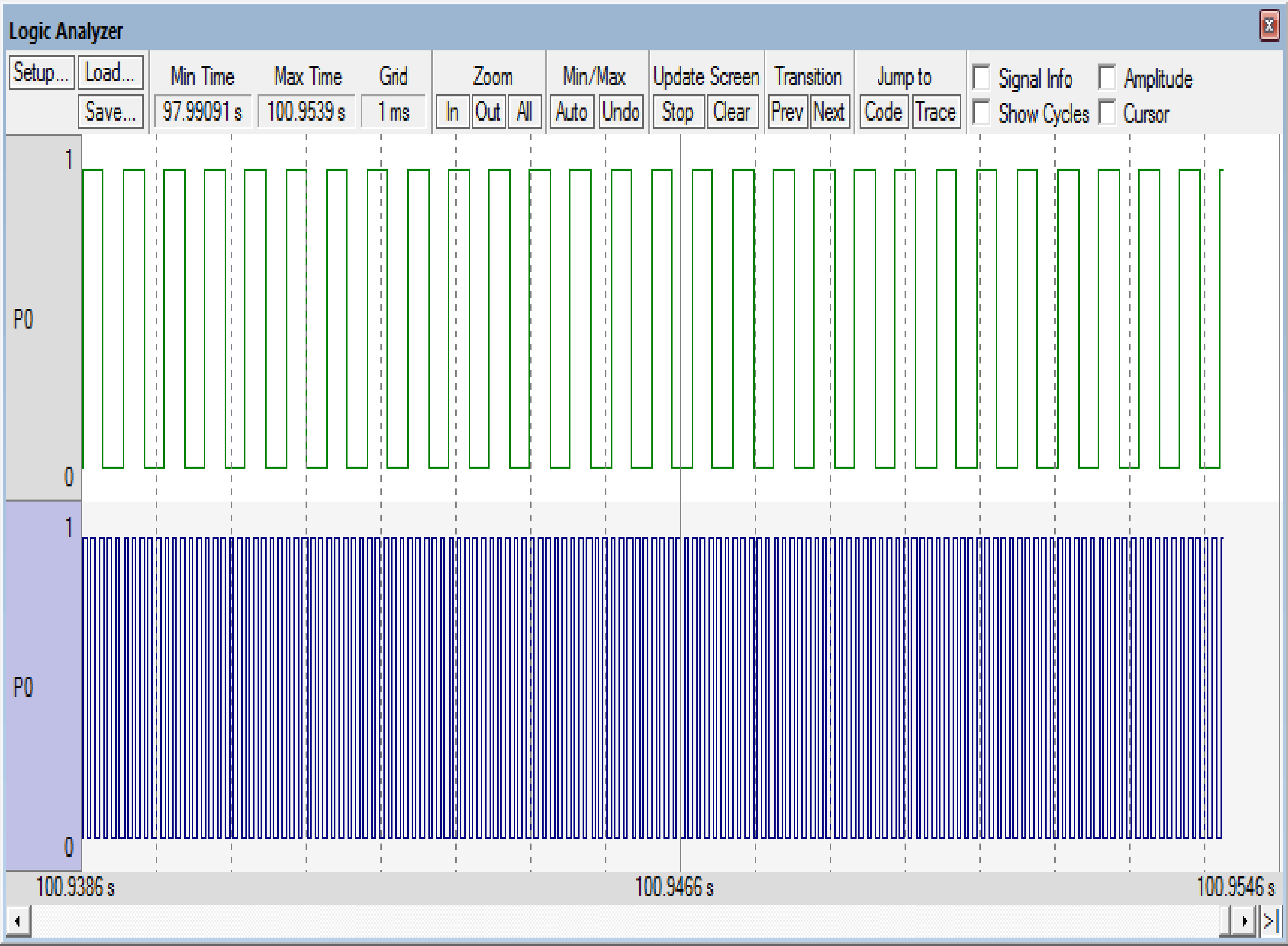
TH0 = 06H

For wave of 10KHz,

On time = 0.05 msec, Off time = 0.05msec

TH1 = 0CEH

```
ORG 0000H
LJMP MAIN
ORG 000BH //ISR for TF0
CPL P0.0
RETI
ORG 001BH //ISR for TF1
CPL P0.1
RETI
ORG 0030H
MAIN: MOV TMOD, #22H
MOV TH0, #06H
MOV TH1, #0CEH
MOV IE, #8AH
SETB TR0
SETB TR1
H: SJMP H
```


Logic Analyzer
Setup...
Load...
Save...
Min Time
97.99091 s
Max Time
100.9539 s
Grid
1 ms
Zoom
In
Out
All
Min/Max
Auto
Undo
Update Screen
Stop
Clear
Transition
Prev
Next
Jump to
Code
Trace
Signal Info
Show Cycles
Amplitude
Cursor
1
P0
0
1
P0
0
100.9386 s
100.9466 s
100.9546 s

Assembler Directives of 8051

- **ORG - Originate**

✓ Allows to set the beginning address of the program or subroutine

✓ Number after ORG can be in hex or decimal

✓ If the number is in decimal, the assembler automatically converts it to hex

✓ Syntax: **ORG address**

✓ Eg: ORG 0000H ; start program from location 0000H in ROM

Assembler Directives of 8051

- **DB – Define Byte**

✓Define byte type variable

✓Numbers following DB can be in decimal, binary, hex or ASCII formats

✓Whatever be the type of the byte, assembler converts the byte to hex

✓For ASCII, the characters are enclosed in quotation marks (single or double quotes)

Assembler Directives of 8051

- **DB – Define Byte**

✓DB is the only directive that can be used to define ASCII strings having more than 2 characters

✓Syntax:

DB 46 ; Decimal (D after no. is optional)

DB 10110110 B ; Binary

DB “123” ; ASCII

DB ‘HELLO’ ; ASCII

DB 27H ; Hexadecimal

PRICE DB 20H, 30H, 40H ; Array of 3 bytes, named as price and initialized

✓Used to allocate memory in byte sized chunks

Assembler Directives of 8051

- **EQU - Equate**

✓Use to give a name to a value or symbol in the program so that the value can be referred to by that name at all places within the program

✓Defines a constant without occupying a memory location

✓Each time the assembler finds that name in the program, it replaces that name with the value assigned to that name

✓Syntax: **[name] EQU 15H**

✓Eg: XYZ EQU 12H

Assembler Directives of 8051

- **EQU – Equate**

✓This statement needs to be written at the beginning of the program

✓Whenever XYZ appears in an instruction or another directive, the assembler substitutes the value as 12H

✓Symbols may be defined in this way only once in the program

✓Advantage: If XYZ is used several times in a program and the value has to be changed, all that has to be done is change the value in EQU statement instead of changing the value at multiple places in the program

Assembler Directives of 8051

- **END**

✓Placed after the last statement of a program

✓Tells the assembler that this is the end of program module

✓Assembler ignores any statement after END directive

✓Syntax: **END**

Assembler Directives of 8051

- **Public and Extern**

✓Public and extern directives are written at the beginning of the program

✓Public directive declares the variables defined in a specific file that can be used in other source files

✓Extern directive declares the variables that are used in the present source file but are defined in some other source file

✓Syntax: PUBLIC DIVISOR, DIVIDEND ;these two variables are public so these are available to all modules.

EXTERN NUMBER ; variable NUMBER is to used from some other module

Assembler Directives of 8051

- **SET**

✓Used to replace a number by a symbol

✓Significant difference compared to EQU directive is that the SET directive can be used an unlimited number of times

✓Eg:

SPEED SET 45

SPEED SET 46

SPEED SET 57

Assembler Directives of 8051

- **BIT**

✓Used to replace a bit address by a symbol

✓Bit address must be in range of 0 to 255

✓Eg:

TRANSMIT BIT PSW.7 ; 7th bit in PSW register is assigned the name as TRANSMIT

OUTPUT BIT 6 ; bit at address 06H is assigned the name as OUTPUT