Microprocessors & Microcontrollers

8051 Assembly Language Programming/ Instruction Sets

8051 Assembly Language Programming/Instruction Sets

The microcontroller 8051 instructions set includes 110 instructions, 49 of which are single byte instructions, 45 are two bytes instructions and 17 are three bytes instructions. The instructions format consists of a function mnemonic followed by destination and source field.

All the instructions of microcontroller 8051 may be classified based on the functional aspect are given below

- Data transfer group.
- Arithmetic group.
- Logical group.
- Bit manipulation group.
- Branching or Control transfer group.

ADDRESSING MODES

The instructions of 8051 may be classified based on the source or destination type

- Register addressing.
- Direct addressing.
- Register Indirect addressing.
- Immediate addressing.
- Base register + Index register.

Immediate Addressing Modes

When the 8051 executes an immediate data move, the program counter is automatically incremented to point to the byte(s) following the opcode byte in the program memory. Whatever, data is found there is copied to the destination address. The mnemonic for immediate data is the pound sign (#).

MOV A, # 20H: Load 20H into A

MOV R2, # 42H: Load the decimal value 42H into R2

MOV RO, # 24H: Load the decimal value 24H into RO

MOV DPTR, #4000H: DPTR=4000H

Register Addressing Modes

Register addressing modes involves the use of register to hold the data to be manipulated. Example of register addressing mode follow.

```
MOV A, R1      ; Copy the contents of Rl into A
MOV R2, A      ; Copy contents of A into R2
ADD A, R5      ; Add the contents of R5 to contents
MOV R5, A      ; save accumulator of R5
```

Direct Addressing Modes

There are 128 bytes of RAM in the 8051, The Internal RAM uses addressing from 00H to 7FH to address each byte.

- RAM locations 00-1FH are assigned to the from bands of eight working register R0 to R7
- RAM locations 20-2FH are set aside as bit addressable space to save single bit data
- RAM locations 30-7FH are available as a place to save byte sized data.

Example of Direct addressing mode

```
MOV Rl, 20H        ; same content of RAM location 20H in Rl
MOV 42H, A         ; same content of A is RAM location 42H
```

The SFR addresses between 80H to FFH, since the addresses 00H to 7FH are addresses of RAM memory inside the 8051. All the address spaces of 80 to FF are not used bytes the SFR. The unused locations 80H to FFH are reserved and must not be used by the 8051 programmer.

Indirect Addressing Mode

In the register indirect addressing mode, a register is used the contents of R0 and R1 often called a data pointer, as a pointer to location in 256 bytes block and the 256 bytes of internal RAM or the lower, 256 bytes of external data memory execution of PUSH and POP is also uses indirect addressing.

Example:

```
MOV @ R0, # n ; copy immediate byte A to the address in R0
MOV @ R0, A; copy A to RAM location R0 pointers
MOV A, # Rp ; copy the contents of the address in Rp to A.
```

Note:

- Number in register Rp must be RAM address
- R0 or Rl register for indirect addressing

Table opcode using immediate, register direct & indirect addressing mode

Mnemonic	Operations
MOV A, @ R0	Copy the content of the address in R0 to the A register
MOV @ Rl, #35H	Copy the number 35H to the address in Rl

MOV ADD,@R0	Copy the content of the address in R0 to Add
MOV @ Rl & A	Copy the content of the address in Rl to A register
MOV @ R0, 80	Copy the content 80 port 0 pins to the address into R0

Example of Indirect Addressing mode

It is widely used is accessing data elements of from program memory an indirect move from the location whose address in the sum of a base register (DPTR or PC) and index register accumulator. These mode facilities lookup accesses.

This instruction used for the purpose is "MOV A, @A + DPTR "

MOV DPTR, # 4200	Copy the number of 4200 to DPTR
MOV A, #25	Copy the number 25 to A
MOV A, @A + DPTR	Copy the content of 4225 to A

Note:

4200 + 25 = > 4225

Another type of indexed addressing is used is the "case jump" instruction. In this case, the destination address of a jump instruction is completed as the sum of the base pointer and the accumulator data.

INSTRUCTION SET OF 8051 MICROCONTROLLER

All members of the 8051 family execute the same instructions set. The 8051 instructions set is optimized for 8-bit content application. The Intel 8051 has excellent and most powerful instructions set offers possibilities in control area, serial Input/Output, arithmetic, byte and bit manipulation.

It has 111 instructions they are

- 49 single byte instructions
- 45 two bytes instructions
- 17 three bytes instructions

The- instructions set is divided into four groups, they are

- Data transfer instructions
- Arithmetic instructions
- Logical instructions
- Call and Jump instructions

DATA TRANSFER INSTRUCTIONS

MOV <dest-byte>, <src-byte>: (Move byte variable)

This instruction copies the contents of the source location to the destination location. The contents of the source location are unchanged.

Examples

Instruction format	Algorithm	Example	Function
MOVA, Rn	A = Rn	MOV A, R1	Move byte form register Rn to accumulator
MOV A, direct	A = direct	MOV A, 40H	Move byte from direct address to accumulator
MOV A, @ Ri	A=[[Ri]]	MOVA,@ R0	Move the content of memory location to accumulator
MOV A, # data	A = # data	MOV A, #31 H	Move immediate data to accumulator
MOV Rn, A	Rn = A	MOV R5, A	Move data from accumulator to register Rn
MOV Rn, direct	Rn = direct	MOV R3, 30H	Move data from direct address to register Rn.
MOV Rn, # data	Rn = # data	MOV R7, #20H	Move immediate data to register Rn.
MOV direct, A	direct =A	MOV 80H, A	Move data from accumulator to direct address
MOV direct, Rn	direct = Rn	MOV 30H, R5	Move data from register to direct address
MOV direct, direct	direct = direct	MOV 20H, 30H	Move data form source direct address to the destination direct address.
MOV direct, @Ri	direct = [[Ri]]	MOV 20H, @R1	Move data from address specified in register Ri to direct address.
MOV direct, #data	direct = #data	MOV 10H, #10H	Move immediate data to direct address
MOV @Ri, A	[[Ri]] = A	MOV @ R0,A	Move data form accumulator to memory location pointed by Ri

MOV @Ri, direct	[[Ri]]=direct	MOV @R0, 30H	Move data from direct address to the memory location pointed by Ri.
MOV @Ri, #data	[[Ri]]= #data	MOV @Ro, 30H	Move immediate data to memory location pointed by the Ri register of selected register bank.

MOV DPTR, # data 16:

- Load data pointer with a 16 bit constant
- Operation: (DPTR) ← $\#data_{15-0}$

  (DPH) ← $\#data_{15-8}$ (

  DPL) ← $\#data_{7-0}$
- This instruction will load the data pointer with the 16-bit constant indicated.
- Example : MOV DPTR, #2476 H : This instruction will load the immediate data 2476 H into the Data Pointer. DPH will hold 24H while DPL will hold 76H.

MOV A, @ A + <base register>

- Operation: MOV A, @ A + DPTR : [A] ← [[A] + [DPTR]] or MOV A, @ A + PC : [PC] ← [PC] +1, [A] ← [A]+[PC]
- This instruction will load the accumulator with a code byte or constant from the program memory.

MOVX, <dest-byte>, <src-byte>

- The MOVX instructions transfer data between the accumulator and a byte of external data memory.
- Depending on whether the indirect address provided to the external data RAM is eight bit or 16 bit, there are two types of instructions.
- In the first type, the contents of register R0 or Rl of current register bank provide an 8-bit address that is multiplexed with data on port 0. Eight bits are sufficient for external I/O expansion decoding or for a relatively small RAM array.
- In the second type, the Data Pointer is used for generating 16-bit address. This is done for larger RAM array. Port 2 outputs the high-order address bits (the contents of DPH), while Port 0 output the low-order address bits (contents of DPL) multiplexed with data.

Instruction	Description
MOVX A, @Ri	This instruction will copy the data from 8 bit address pointed by register Ri of the selected register bank to the accumulator.
MOVX A, @DPTR	This instruction will copy the contents of external data memory location, pointed by DPTR to the accumulator.
MOVX @Ri, A	This instruction will copy the contents of the accumulator to the external data memory location pointed by register Ri of selected register bank.
MOVX @ DPTR, A	This instruction will copy the contents of the accumulator to the 16-bit address, pointed by DPTR.

PUSH <direct>

This instruction copies the data from the source address onto the stack.

Operation

$[SP] \leftarrow [SP] + 1, [[SP]] \leftarrow \text{direct}$

Addressing mode

Direct addressing

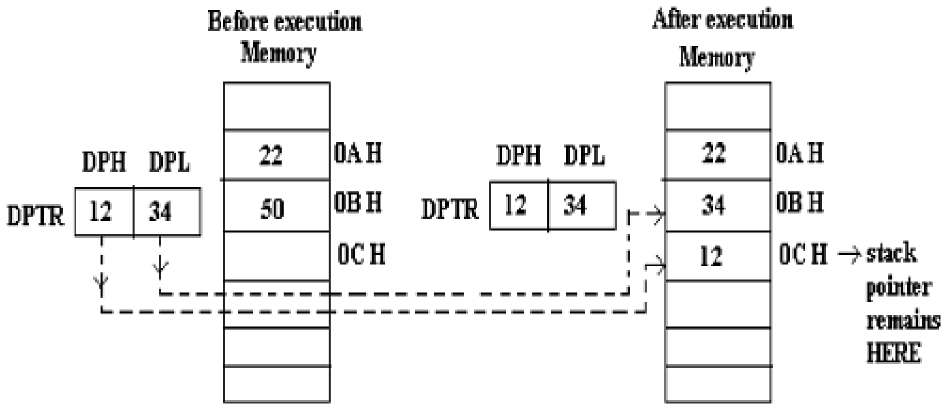
Example

Let SP =0AH and data pointer = 1234H

PUSH DPL

PUSH DPH

The first instruction PUSH DPL will set the SP = 0BH and store 34H in internal RAM location 0BH. The second instruction PUSH DPH will set the SP = 0CH and store 12H in internal RAM location 0BH. The stack pointer will remain at 0CH.



POP <direct>

This instruction copies data from the stack to the destination location.

Operation

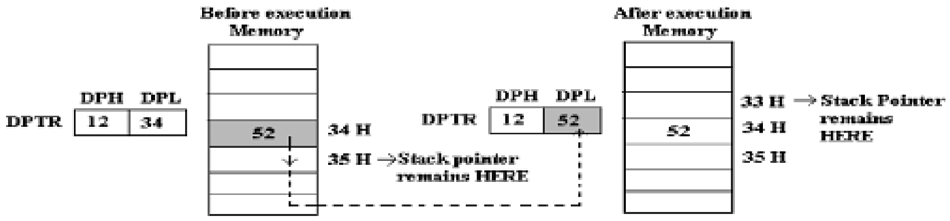
(direct) ← [[SP]], [SP] ← [SP] -1

Addressing mode

Direct addressing.

Example

Let SP = 35 H and data at internal RAM locations 34H be 52H, then instruction POP DPL will level the stack pointer to value 34H and DPL = 52H.



XCH A, <Byte variable>

Function

Exchange accumulator with byte variable.

Operation

[A] ← → [byte variable]

This instruction will load the accumulator with the contents of memory location pointed by byte variable. At the same time the original accumulator contents arc written to the memory location pointed by byte variable.

The source/destination operand can use register, direct, or register indirect addressing.

Mnemonic	Example	Description
XCH A, Rn	XCIH A, R1	This instruction will load the contents of register Rl of selected register bank in the accumulator and at the same time the contents of original accumulator will be copied in register Rl
XCH A, direct	XCH A, 10H	This instruction will load the contents of memory location whose address is 1011 to the accumulator and at the same time the contents of accumulator are transferred to the memory location whose address is 10H

XCH A, @Ri	XCH A,@RO	This instruction will load the contents of memory location pointed by register R0 of selected register bank to the accumulator and at the same time the contents of accumulator arc copied lo the memory location pointed by R0 register of the selected register bank.
XCI TO A, @Ri	XCHDA, @R0	This instruction exchange the lower nibble of accumulator (bit 3-0) with the lower nibble of the memory location indirectly addressed by the specified register R0/R1.

Data Transfer Instruction (summary)

These instructions move the content of one register to another one (the register which content is moved is not altered). If they have the suffix "X" (MOVX), the data is exchanged with external memory.

Data Transfer Instruction			
Mnemonic	Description	Byte Number	Cycle Number
MOV A, Rn	Move R register to Accumulator	1	1
MOV A, Rx	Move directly addressed Rx register to Accumulator	2	2
MOV A, @Ri	Move indirectly addressed register to Accumulator	1	1
MOV A, #X	Store number X in accumulator	2	2
MOV Rn, A	Move Accumulator to R register	1	1
MOV Rn, Rx	Move directly addressed Rx register to R register	2	2
MOV Rn, #X	Move number X to R register	1	2
MOV Rx, A	Move Accumulator to directly addressed Rx register	2	2
MOV Rx, Rn	Move R register to directly addressed Rx register	2	1
MOV Rx, Ry	Move directly addressed register Ry to directly addressed Rx register	3	3
MOV Rx, @Ri	Move indirectly addressed register to directly addressed Rx register	2	2

MOV Rx, #X	Store number X in directly addressed Rx register	3	3
MOV @Ri, A	Move Accumulator to indirectly addressed register	1	1
MOV @Ri, Rx	Move directly addressed Rx register to indirectly addressed register	2	2
MOV Rx, #X	Store number X in directly addressed Rx register	3	3
MOV @Ri, A	Move Accumulator to indirectly addressed register	1	1
MOV @Ri, Rx	Move directly addressed Rx register to indirectly addressed register	2	2
MOV @Ri, #X	Store number X in indirectly addressed register	2	2
MOV DPTR, #X	Store number X in Data Pointer	3	3
MOVC A, @A+PC	Move register from Program Memory (address = A + PC) to Accumulator	1	3
MOVX A, @Ri	Move data from external memory (8-bit address) to Accumulator	1	2
MOVXA,@DPTR	Move data from external memory (16-bit address) to Accumulator	1	2
MOVX@ Ri, A	Move Accumulator to external memory register (8-bit address)	1	2
MOVX @DPTR, A	Move Accumulator to external memory register (16-bit address)	1	2
PUSH Rx	Push directly addressed Rx register in the part of RAM provided for Stack	2	2
POP Rx	Pop data from Stack part of RAM. Store it in directly addressed Rx register	2	2
XCH A, Rn	Exchange Accumulator with R register	1	1
XCH A, Rx	Exchange Accumulator with directly addressed Rx register	2	2

XCH A, @Ri	Exchange accumulator with indirectly addressed register	1	1
XCHD A, @Ri	Exchange 4 lower bits in accumulator with indirectly addressed register	1	1

ARITHMETIC INSTRUCTIONS:

ADD A, <src-byte>

Operation

[A] ← [A] + <src-byte>

- This instruction adds the byte variable indicated to the accumulator. The result is contained in the accumulator.
- All the addressing modes can be used for source: an immediate number, a register, direct address, and indirect address.

Mnemonic	Example	Description
ADD A, Rn	ADD A, R0	This instruction will add the byte in register Rn of the selected register bank with the byte in accumulator. The result is contained in the accumulator
ADD A, direct	ADD A, 20H	This instruction will add the contents of the memory location whose direct address is specified in the instruction with the accumulator contents. The result of addition will he stored in the accumulator.
ADD A, @ Ri	ADD A, @ R0	This instruction will add the contents of memory location whose address is pointed by register Ri of the selected register bank with contents of the accumulator. The result of addition is stored in the accumulator
ADD A, # data	ADD A, # 30H	This instruction will add the immediate 8 bit data with data in the accumulator. The result of addition is stored in the accumulator.

ADDC A, <src-byte>

Operation

[A] ← [A] + < src-byte> + carry

- This instruction will add the byte variable indicated, the carry flag and the accumulator contents. The result of addition is stored in the accumulator.

- All the addressing modes can be used for source : an immediate number, a register, direct address, indirect address.

Mnemonics	Addressing mode	Example	Description
ADDC A, Rn	Register addressing	ADDC A, Rl	This instruction will add the contents of accumulator with the contents of register Rn of the selected register bank and carry flag. The result of addition is stored in accumulator.
ADDC A, direct	Direct addressing	ADDC A, 10H	This instruction will add the contents of memory location whose direct address is specified in the instruction with the contents of accumulator and carry. The result of addition is stored in the accumulator.
ADDC A, @Ri	Register Indirect	ADDC A, @R0	This instruction will add the contents of memory location pointed by register Ri of selected register bank with the accumulator and earn' flag. The result is stored in accumulator.
ADDC A, #data	Immediate addressing	ADDC A, #40H	This instruction will add the contents of accumulator with immediate data specified in the instruction along with carry.

SUBB A, < src-byte>

Operation

[A] ← [A] - <src-byte> - CY

- This instruction subtracts the indicated byte variable and the carry flag contents together, from the accumulator. The result is stored in accumulator.
- All the addressing modes can be used as source: an immediate number, a register, direct address, and indirect address.

Mnemonics	Addressing mode	Example	Description
SUBB A, Rn	Register addressing	SUBB A, Rl	This instruction will subtract the contents of register Rn of the current register bank and carry flag together, from the accumulator. The result is stored in accumulator.
SUBB A, direct	Direct addressing	SUBB A, 10H	This instruction will subtract the contents of memory location whose direct address is specified in the instruction and carry flag together from the contents of accumulator. The result is stored in accumulator.
SUBB A, @Ri	Register Indirect	SUBB A, @R0	This instruction will subtract the contents of memory location pointed by register Ri and contents of carry flag from the accumulator. The result is stored in accumulator.
SUBB A, #data	Immediate addressing	SUBB A. #40H	This instruction will subtract the contents of data specified m the instruction and contents of carry flag from the contents of accumulator. The result is stored in accumulator.

INC <byte>

Operation

byte ← byte +1

- This instruction will increment the indicated variable by 1.
- If the byte value is FFH and if it is incremented, then the result will overflow to 00H.
- It supports three addressing modes. Register, direct and register-indirect.

Mnemonics	Addressing mode	Example	Description
INC Rn	Register addressing	INC R5	This instruction will increment the contents of register Rn of selected register bank by 1.

INC direct	Direct addressing	INC 10H	This instruction will increment the contents of memory location whose address is specified in the instruction by 1
INC @Ri	Register Indirect	INC, @R0	This instruction will increment the contents of memory location that is pointed by register Ri by 1.

INC DPTR

Operation

[DPTR] ← [DPTR] + 1

- Addressing mode: Register addressing mode.
- This instruction will increment the contents of Data Pointer by 1.
- A 16-bit increment is performed. An overflow of the low-order byte of the data pointer (DPL) from 0FFH to 00H will increment the high order byte (DPH).
- DPTR is the only 16-bit register that is incremented.
- Example: let the contents of DPH = 12 H and the contents of DPL = FFH. The instruction ING DPTR will cause DPL = 13 H and DPL = 00 H.

DEC <byte>

Operation

<byte> ← <byte> -1

- This instruction will decrement the indicated variable by 1.
- An original value of 00H will underflow FFH.
- Three operand-addressing modes are allowed: register, direct and register indirect.

o

Mnemonics	Addressing mode	Example	Description
DEC Rn	Register addressing	DEC R5	This instruction will decrement the contents of register Rn of selected register bank by 1-
DEC direct	Direct addressing	DEC 10H	This instruction will decrement the contents of memory location whose direct address is specified in the instruction by 1.

DEC @Ri	Register Indirect	DEC @R0	This instruction will decrement the contents of memory location that is pointed by register Ri by 1.

MUL AB

Operation

$[A]_{7-0} \leftarrow [A] \times [B] . [B]_{15-8}$

- This instruction multiplies an eight bit unsigned integer in the Accumulator and the 13 register. The low-order byte of the sixteen-bit product is left in the accumulator, and the high-order byte in B.
- Addressing mode : Register addressing mode.
- Example: Let A = 50 H, B = A0 H after execution of MUL AB the content of register B = 32 H and register A = 00 H (as (50H) X (A0 M) = (3200H)).

DIV AB

Operation

[A] (Quotient) <- A + B.   [BJ (Remainder)

- This instruction divides the unsigned number in accumulator with the unsigned number in register B.
- Accumulator contents the quotient of the result and register B contains the remainder.
- The contents of A and B, when division by 0 is attempted, are undefined.
- Addressing mode. Register addressing mode.
- Example; Let [A] = FB H and [B] = 12 H then DIV AB will result [A] = 0DH (quotient), [B] = 11 H (remainder)

DAA

Decimal Adjust Accumulator for addition.

Arithmetic Instructions (Summary)

Mnemonic	Description	Byte Number	Oscillator Period
ADO A, Rn	Add Rn Register to Accumulator	1	1
ADD A, Rx	Add directly addressed Rx Register to Accumulator	2	2
ADD A, @Ri	Add indirectly addressed Register to Accumulator	1	1
ADD A, #X	Add number X lo Accumulator	2	2

ADDC A, Rn	Add R Register with Carry bit to Accumulator	1	1
ADDC A, Rx	Add directly addressed Rx Register with Carry bit to Accumulator	2	2
ADDC A, @Ri	Add indirectly addressed Register with Carry bit lo Accumulator	1	1
ADDC A, #X	Add number X with Carry bit to Accumulator	2	2
SUBB A, Rn	Subtract R Register with borrow from Accumulator	1	1
SUBB A, Rx	Subtract directly addressed Rx Register with borrow from Accumulator	2	2
SUBB A, @Ri	Subtract indirectly addressed Register with borrow from .Accumulator	1	1
SUBB A, #X	Subtract number X with borrow from Accumulator	2	2
INC Rn	Increment R Register by 1	1	1
INC Rx	Increment directly addressed Rx Register by 1	2	2
INC @Ri	Increment indirectly addressed Register by 1	1	1
DEC A	Decrement Accumulator by 1	1	1
DEC Rn	Decrement R Register by 1	1	1
DEC Rx	Decrement directly addressed Rx Register by 1	2	2
DEC @Ri	Decrement indirectly addressed Register by 1	1	1
INC DPTR	Increment Data Pointer by 1	1	3
MUL AB	Multiply number in Accumulator by B register	1	5
DIY AB	Divide number in Accumulator by B register	1	5
DA A	Decimal adjust Accumulator according to BCD code	1	1

LOGICAL INSTRUCTIONS

ANL <dest-byte>, <src-byte>

- This instruction performs bit wise logical AND operation between the destination and the source byte. The result is stored at the destination byte.
- The source and destination support four addressing modes: register, direct, register-indirect, and immediate addressing modes.

Mnemonics	**Addressing mode**	**Example**	**Description**
ANL A, Rn	Register addressing	ANL A, R5	This instruction will perform bit wise logical AND operation between the contents of accumulator and register Rn of the selected register bank. The result will be stored in the accumulator
ANL A, direct	Direct addressing	ANL A, 70H	This instruction will bit wise logically AND the contents of accumulator with the contents of memory location whose direct address is specified in the instruction. The result will be stored in the accumulator
ANL direct, A	Direct addressing	AXL 30H, A	This instruction will bit wise logically AND the contents of memory location whose direct address is specified in the instruction with the contents of accumulator. The result will be stored in the memory location whose direct address is specified in the instruction.
ANL A, ®Ri	Register Indirect Addressing	ANL A, @R1	This instruction will bit wise logically AND the contents of accumulator with the contents of memory location pointed by register Ri of the selected register bank. The result will be stored
ANL A, #data	Immediate addressing	ANL A, #57H	This instruction will bit wise logically AND the contents of accumulator with the immediate data specified in the instruction. The result will be stored in the accumulator.
ANL direct, #data	Immediate addressing	ANL 54H, #33H	This instruction will bit wise logically AND the contents of memory location whose direct address is specified in the instruction with the contents of with the immediate data specified in the instruction. The result will be stored in the memory location whose direct address is specified in the instruction.

ORL <dest-byte>, <src-byte>

- This instruction performs bit wise logical OR operation between the destination and the source byte. The result is stored at the destination byte.
- The source and destination support four addressing modes: register, direct, register-indirect, and immediate addressing modes.

XRL <dest-byte>, <src-byte>

- This instruction performs bit wise logical Exclusive-OR operation between the destination and the source byte. The result is stored at the destination byte.
- The source and destination support four addressing modes: register, direct, register-indirect, and immediate addressing modes.

CLRA

- This instruction will clear all the bits of accumulator to zero.

CPLA

- This instruction will complements all the bits (l's complement) of the accumulator.

RLA

- This instruction will rotate the eight bits in the accumulator by one bit to the left.
- Addressing mode: Register Specific addressing mode.

RLCA

- This instruction will rotate the eight bits in the accumulator and the carry flag together by one bit to the left
- Addressing mode: Register Specific addressing mode.

RRA

- This instruction will rotate the eight bits in the accumulator by one bit to the right.
- Addressing mode: Register Specific addressing mode.

RRCA

- This instruction will rotate the eight bits in the accumulator and the carry flag together by one bit to the right.
- Addressing mode: Register Specific addressing mode.

SWAP A

This instruction interchanges the low order and high order nibbles of the accumulator.

Operation:

$$[A_{3-0}] \leftarrow \rightarrow [A_{7-4}]$$

- Addressing mode: Register Specific addressing mode.

TIMER/COUNTER PROGRAMS

To Generate a Square Wave On The Port 1.

```
            MOV SP, #7H         ; Initialize stack pointer since we are using
                                ; subroutine program
BACK:       MOV P1, #00H        ; Send 00H on port 1 to generate low level
                                ; of square wave
            ACALL DELAY         ; Wait for sometime
            MOV P1, #0FFH       ; Send FFH on port 1 high level of square wave
            ACALL DELAY         ; Wait for sometime
            SJMP BACK           ; Repeat the sequence
DELAY:      MOV R1, #0FFH       ; Load Count
AGAIN:      DJNZ R1, AGAIN      ; Decrement count and repeat the process
                                ; until count is zero
            RET                 ; Return to main page
```

To Generate a Square Wave On the Port Pin P1.0

```
            MOV SP, #7H         ; Initialize stack pointer since we are
                                ; using subroutine program
            CLR P1.0            ; Send 0 on port 1.0 to generate low level of
                                ; square wave
            ACALL DELAY         ; Wait for sometime
            SETB P1.0           ; Send 1 on port 1.0 to generate high level of
                                ; square wave
            ACALL DELAY         ; Wait for sometime
            SJMP BACK           ; Repeat the sequence
DELAY:      MOV R1, 0FFH        ; Load Count
AGAIN:      DJNZ R1, AGAIN      ; Decrement count and repeat the process
                                ; until count is zero.
            RET                 ; Return to main program.
```

Program using Timer 0 to create a 10 kHz square wave on P1.0

```
            ORG 8100H
            MOV TMOD, #02H          ; 8 bit auto reload mode
            MOV TH0, #-50           ; -50 reload value in TH0
```

```
                SETB TR0                ; start timer
      LOOP:     JNB TF0, LOOP           ; wait for overflow
                CLR TF0                 ; clear timer overflow flag
                CPL P1.0                ; toggle port bit
                SJMP LOOP               ; repeat
                END
```

The program above creates a square wave on P1.0 with a high-time of 50 μs and a low-time of 50 μs. Since the interval is less than 256 μs, timer mode 2 can be used. An overflow every 50 μs requires a TH0 reload value of 50 counts less than 00H, or -50.

The program uses a complement bit instruction (CPL) rather than SETB and CLR. Between each complement operation, a delay of 1/2 the desired period (50 μs) is programmed using Timer 0 in 8-bit auto-reload mode. The reload value is specified using decimal notation as -50, rather than using hexadecimal notation. The assembler performs the necessary conversion. Note that the timer overflow flag (TF0) is explicitly cleared in software after each overflow.

Program using Timer 0 to create a 1kHz square wave on P1.0

```
8100                        ORG   8100H
8100   758901               MOV   TMOD,#01H   ;16-bit timer mode
8103   75BCFE     LOOP:     MOV   TH0,#0FEH   ;-500 (high byte)
8106   758A0C               MOV   TL0 #0CH    ;-500 (low byte)
8109   D28C                 SETB  TR0         ;start timer
810B   308DFD     WAIT:     JNB   TF0,WAIT    ;wait for overflow
810E   C28C                 CLR   TR0         ;stop timer
8110   C28D                 CLR   TF0         ;clear timer overflow
                                              flag
8112   B290                 CPL   P1.0        ;toggle port bit
8114   80ED                 SJMP  LOOP        ;repeat
                            END
```

```
                ORG 8100H
                MOV TMOD, #01H          ; 16 bit timer mode
      LOOP:     MOV TH0, #0FEH          ; -500 (high byte)
                MOV TL0, #0CH           ; -500 (low byte)
                SETB TR0                ; start timer
      WAIT:     JNB TF0, WAIT           ; wait for overflow flag
                CLR P1.0                ; toggle port bit
                SJMP LOOP               ; repeat
                END
```

A 1 kHz square wave requires a high-time of 500 μs and a low-time of 500 μs. Since the interval is longer than 256 μs, mode 2 cannot be used. Full 16-bit timer mode, mode 1, is required. The main difference in the software is that the timer registers, TL0 and TH0, are reinitialized after each overflow.

There is a slight discrepancy in the output frequency in the program above. This results from the extra instructions inserted after the timer overflow to reinitialize the timer. If exactly 1 kHz is required, the reload value for TL0/TH0 must be adjusted somewhat. Such errors do not occur in auto-reload mode, since the timer is never stopped - it overflows at a consistent rate set by the reload value in TH0.

PROGRAMMING EXAMPLES

Example (AMIE W12, 10 marks)

List the programming steps needed to receive data serially using 8051 microcontroller and explain.

Solution

The following steps are taken to program 8051 to receive data serially.

1. The TMOD register is loaded with value 20 H indicating timer 1 is used mode 2 (8 bit auto reload mode). This timer is used to set band rate.
2. TH1 is loaded with a suitable value to set proper band rate. If crystal frequency is 11.0592 MHz and if we load TH1 with FD we get a band rate of 9600.
3. The SCON register is loaded with the value 50 H, indicating serial mode 1, where 8 bit data is framed with start and stop bits and receive enable is turned on.
4. TR1 flag is set to 1 to start Timer
5. R1 is cleared with the CLR R1 instruction.
6. The R1 flag bit is monitored with the use of the instruction JNB R1, XX to see if an entire character has been received yet.
7. When Ri is raised SBUF has the serial data byte move the
8. contents to desired memory location.
9. To receive the next character go to step 5.

Example

Write program to receive bytes of data serially and put them in P1. Set the baud rate at 2400, 8-bit data, and 1 stop bit. Assuming crystal frequency 11.0592 MHz.

Solution

```
        MOV TMOD, #20H        ; timer l, mode 2 (auto reload)
        MOV TH1, #-12         ; 2400 baud
        MOV SCON, #50H        ; 8-bit. 1 stop, REN enabled
```

```
            SETB TR1            ; start timer 1
NEXT:       JNB R1, NEXT        ; wait for char to come in
            MOV A, SBUF         ; save incoming byte in A
            MOV P1, A           ; send to port 1
            CLR R1              ; get ready to receive next byte
            SJMP NEXT           ; keep getting data
```

Example

Write program to send 44H to ports P1 and P2, using (a) their addresses (b) their names.

Solution

```
(a)         MOVA, #44H
            M0V P1, A
            MOV   P2, A
(b)         MOVA, #44H
            MOV 80, A
            MOV 0A0H, A
```

Example

Write program to copy a block of 8 bytes of data to RAM locations starting at 50H from RAM locations 30H.

Solution

```
            MOV R0, #30H        ; source pointer
            MOV R1, #50H        ; destination pointer
            MOV R3, #8          ; counter
RETURN:     MOV A, @R0          ; get a byte from source
            MOV @Rl, A          ; copy it to destination
            INC R0              ; increment source pointer
            INC R1              ; increment destination pointer
            DJNZ R3, RETURN     ; keep doing it for all eight bytes
```

Example

Write a program for 8051 to transfer serially letter N at 9600 baud, continuously. Assuming crystal frequency 11.0592 MHz.

Solution

```
            MOV TMOD, #20H    ; timer 1, mode 2 (auto reload)
            MOV TH1, #-3      ; 9600 baud rate at 11.0592 MHz
            MOV SCON, #50H    ; 8-bit, 1 stop, REN enabled
            SETB TR1          ; start timer 1
REPEAT:     MOV SBUF, #"N"    ; letter "N" to be transferred
NEXT:       JNB TI, NEXT      ; wait for the last bit
            CLR TI            ; clear TI for next char
            SJMP REPEAT       ; keep sending N
```

Example

Add the contents of RAM locations 60H, 61H and 62H. Store the result in RAM locations 41H (MSB) and 40H (LSB).

Solution

1. Add first two numbers.
2. Store carry and result. Result is in accumulator. Carry is indicated by CY flag.
3. Add the third byte to the result. If there is a carry add it to the previous carry.

We now code this algorithm.

```
MOV   41H, #00H     ; Clear the RAM location where MSB has to be stored.
MOV   A, 60H        ; Move first number to accumulator.
ADD   A, 61H        ; Add the second number.
MOV   R0, A         ; Store result in R0.
MOV   A, 00H        ; Clear accumulator.
ADDC A, 41H         ; Add A + CY + Contents of 41.
                    ; Now A = 00H; 41H contains 00H.
                    ; Therefore we essentially get the status of
                    ; CY flag into accumulator.
MOV   41H, A        ; Move it to location 41H.
MOV   A, R0         ; Get back result of first addition into accumulator.
ADD   A, 62H        ; Add third byte. Result (LSB) is in accumulator.
MOV 40H, A          ; Move result (LSB) to location 40H
MOV   A, 00H        ; Clear accumulator.
ADDC A, 41H         ; Add present carry with previous carry in 41H.
MOV   41H, A        ; Move MSB to 4lH.
```

ASSIGNMENT

Q.1. (AMIE S14, 10 marks): Explain the instruction sets of 8051 in detail.

Q.2. (AMIE S10, 20 marks): Explain the following instructions of 8051 microcontroller:

(i) DJNZ R2, THERE

(ii) CJNE A, DIRECT, REL

(iii) PUSH 0

(iv) MOVX A, @DPTR

(v) MOVCA, @ A+DPTR

(vi) LJMP 4100H

(vii) MOV @ R0, # data

(viii) ORL A, @R1

(ix) XCH A, @R0

(x) SETB C

Q.3. (AMIE W10, 10 marks): Explain the following instructions of 8051 microcontrollers: (i) INCA (ii) INC add (iii) ADD @ RP (iv) ADD CA, #n (v) MOL AB

Q.4. (AMIE S11, 8 marks): Explain the following instructions:

(i) MOV A, # 56H

(ii) MOVC A, @A+DPTR

(iii) DEC @R1

(iv) DJNZ R0 BACK

Q.5. (AMIE W12, 10 marks): Explain the operation of the following instructions of 8051 microcontroller: (i) SWAPA (ii) MOV A @R1 (iii) MOV$_x$ A#R$_1$ (iv) DJNZ R$_0$ TABLE (v) SETB P1.3

Q.6. (AMIE W13, 10 marks): Explain the operation of the following instructions with examples: (i) MOVC A @A + DPTR (ii) XCHD A, @RO (iii) MOV C, P3.1 (iv) SWAPA (vv) RR A

Q.7. (AMIE W14, 10 marks): Explain the operations of following instructions: (i) CJNE A, # data, LABEL 1 (ii) DJNZ R$_0$, LABEL 2 (iii) RL A (iv) DA A (v) MOVX @ DPTR, A

Q.8. (AMIE S15, 6 marks): What are the significance of DPTR and EA pin?

Q.9. (AMIE S15, 6 marks): Explain why SJMP instruction is used in place of HLT in 8051.

Q.10. (AMIE S15, 6 marks): Explain the operation of CJNE and DJNZ instructions.

Q.11. (AMIE S10, 5 marks): List and explain the logical group of instructions of 8051 microcontroller with examples.

Q.12. (AMIE W11, 5 marks): Explain any five arithmetic instructions of 8051.

Q.13. (AMIE S12, 16 marks): Write four instructions for each data transfer group, arithmetic group, logical group and branch group.

Q.14. (AMIE S10, 12, 5 marks): Write a delay program using registers of 8051 microcontroller.

Q.15. (AMIE W10, 10 marks): Write a program to double the number in register R1 and store the result in R2 and R3. The microcontroller used is 8051.

Q.16. (AMIE W10, 10 marks): Illustrate the logical OR operation in 8051 using a program.

Q.17. (AMIE S15, 8 marks): Explain addressing modes of 8051 microcontroller.

Q.18. (AMIE W10, 10 marks): Explain how data is stored and received in 8051 using cell and stack instructions.

Q.19. (AMIE S11, 12, W12, 8 marks): Write a program to generate a square wave of frequency 2 kHz through port P1.0 by timer 0 of 8051 microcontroller.

Q.20. (AMIE W13, 10 marks): Write an assembly language program using 8051 microcontroller instructions to generate a 50 Hz square wave at port 0, pin 6 (i.e. p0.6).

Q.21. (AMIE W11, 10 marks): Write an assembly language program using 8051 to generate a sawtooth and square waveform using the general purpose ports of 8051.

Q.22. (AMIE S11, 4 marks): Explain the difference between forward jump and backward jump.

Q.23. (AMIE W11, 4 marks): Explain the instructions to access external RAM and external ROM.

Q.24. (AMIE W11, 10 marks): Write an assembly language program using 8051 to access the 7 segment code of a number which is stored in ROM. Store 7 segment codes of 0-9.

Q.25. (AMIE S13, 10 marks): Explain different types of instructions groups with the help of at least two assembly language instructions for 8051 microcontroller.

Q.26. (AMIE W13, 10 marks): An array of 10 numbers is stored in the internal data RAM starting from location 30H. Write an assembly language program to move the array starting from location 40H.
Q.26. (AMIE S15, 8 marks): Write a program using 8051 assembly language to change the data 55H stored in the lower byte of the data pointer register to AAH using rotate instruction.

Q.27. (AMIE W14, 10 marks): Write an assembly language program of 8051 for producing binary image of a 8 bit binary data.

Q.28. (AMIE S15, 10 marks): Justify why the crystal oscillator frequency in 8051 is chosen as 11.0592 MHz.

(For online support such as eBooks, video lectures, audio lectures, unsolved papers, quiz, test series and course updates, visit www.amiestudycircle.com)