### Arithmetic Instructions

Using Arithmetic Instructions, you can perform addition, subtraction, multiplication and division. The arithmetic instructions also include increment by one, decrement by one and a special instruction called Decimal Adjust Accumulator.

The Mnemonics associated with the Arithmetic Instructions of the 8051 Microcontroller Instruction Set are:

* ADD
* ADDC
* SUBB
* INC
* DEC
* MUL
* DIV
* DA A

|  |  |
| --- | --- |
| **Mnemonic** | **Description** |
| ADD | Addition without Carry |
| ADDC | Addition with Carry |
| SUBB | Subtract with Carry |
| INC | Increment by 1 |
| DEC | Decrement by 1 |
| MUL | Multiply |
| DIV | Divide |
| DA A | Decimal Adjust the Accumulator (A Register) |

The arithmetic instructions have no knowledge about the data format i.e., signed, unsigned, ASCII, BCD, etc. Also, the operations performed by the arithmetic instructions affect flags like carry, overflow, zero, etc. in the PSW Register.

All the possible Mnemonics associated with Arithmetic Instructions are mentioned in the following table.



### Logical Instructions

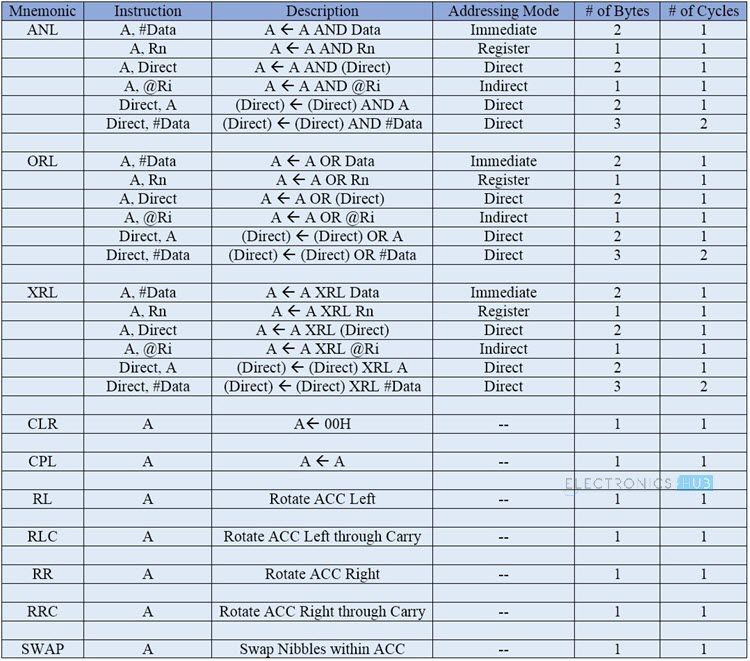
The next group of instructions are the Logical Instructions, which perform logical operations like AND, OR, XOR, NOT, Rotate, Clear and Swap. Logical Instruction are performed on Bytes of data on a bit-by-bit basis.

Mnemonics associated with Logical Instructions are as follows:

* ANL
* ORL
* XRL
* CLR
* CPL
* RL
* RLC
* RR
* RRC
* SWAP

|  |  |
| --- | --- |
| **Mnemonic** | **Description** |
| ANL | Logical AND |
| ORL | Logical OR |
| XRL | Ex-OR |
| CLR | Clear Register |
| CPL | Complement the Register |
| RL | Rotate a Byte to Left |
| RLC | Rotate a Byte and Carry Bit to Left |
| RR | Rotate a Byte to Right |
| RRC | Rotate a Byte and Carry Bit to Right |
| SWAP | Exchange lower and higher nibbles in a Byte |

The following table shows all the possible Mnemonics of the Logical Instructions.



**Programming using Arithmetic and Logical instructions of 8051**

**Aim:**

To write ALP for performing Arithmetic , Logical , Bit manipulation operations in 8051

**1. Arithematic operations:**

Program:

 org 9000h

**;addition without carry**

mov a, #06h

mov b, #09h

add a, b

mov r0, a

**;Substraction with carry**

mov a, #08h

subbc a, #03h

mov r1, a

;multiplication

mov a, #03h

mov b, #06h

mul ab

mov r2, a

;division

mov a, #08h

mov b, #03h

div ab

mov r3, a

mov r4, b

**;increment operation**

mov a, #03h

inc a

mov r5, a

**;decrement operation**

mov r6, #07h

dec r6

lcall 0003h

end

**2. Logical operations:**

Program 1:

 org 8000h

mov r0, #0fh

mov r1, #foh

mov r2, #66h

; And operation

mov a, #ffh

anl a, r0

mov r3, a

; Or operation

mov a, #ffh

orl a, r1

mov r4, a

; Xor operation

mov a, 03h

mov a, #ffh

xrl a, r2

mov r5, a

lcall 0003h

end

Program 2:

org 9000h

; clear register A

mov a, #0fh

clr a

mov r0, a

; swap nibbles of register A

mov a, #56h

swap a

mov r1, a

; Complement the bit of register A

mov a, #66h

cpl a

mov r2, a

; Rotate the register contents towards right

mov a, #63h

rr a

xrl a, r

mov r3, a

; Rotate the register contents towards left

mov a, #43h

rl a

xrl a, r

mov r4, a

lcall 0003h

end

## Sum of 8-bit Numbers Stored in Memory

ORG 00H

MOV R0, #50H ;get memory location in memory pointer R0

MOV R1, #51H ;get memory location on memory pointer register R1

MOV A,@R0 ; get content of memory location 50H to accumulator

ADD A,@R1 ; add content of A with content of memory location 51H and store result in A

MOV R0, #52H ;get 52H to memory pointer R0

MOV@R0,A ; copy content of A to memory location 52H

END

**ADD 16-bit Numbers**

Program

ORG 00H

MOV DPTR, #2040H; get 2040H into DPTR

MOV A, #2BH ; get **lower byte** of second 16-bit number on accumulator

MOV R0, #20H ; get **higher byte** of second 16-bit number on accumulator

ADD A, 82H ; [A]+[DPL]

MOV 82H, A ; save result of lower byte addition

MOV A, R0 ; get higher byte of second number in A

ADDC A, 83H ; [A] + [DPH]

MOV 83H,A ;Save result of higher byte addition

END

**Write an ALP to find division of two 8 bit numbers.**

Label Opcode and Operands Comments

ORG 0000H

LJMP 8000H

ORG 8000H

MOV DPTR,#8500H ;dptr=8501h

MOVX A,@DPTR ;store the number in register a from memory location

8500h

MOV B,A ;copy the number in register B

DEC DPL ; dptr=8500h

MOVX A, @DPTR store the number in register a from memory location 8501h

DIV AB ;divide A by B

MOV DPTR, #9500H ;increment DPTR

MOVX A, @DPTR ;store the quotient in memory location in 8501h

MOV A,B ;store the reminder in register b

INC DPTR ;increment DPTR

MOVX @DPTR,A ;store the reminder in memory location 8502h

LCALL 0003H ;end of asm file

**a. Write an assembly language program to perform the addition of two16-bit numbers.**

mov r0,#34h //lower nibble of No.1

mov r1,#12h //higher nibble of No.1

mov r2,#0dch //lower nibble of No.2

mov r3,#0feh //higher nibble of No.2

clr c

mov a,r0

add a,r2

mov 22h,a

mov a,r1

addc a,r3

mov 21h,a

mov 00h,c

end

**Write an assembly language program to perform the subtraction oftwo 16-bit numbers.**

mov r0,#0dch //lower nibble of No.1

mov r1,#0feh //higher nibble of No.1

mov r2,#34h //lower nibble of No.2

mov r3,#12h //higher nibble of No.2

clr c //

mov a,r0

subb a,r2

mov 22h,a

mov a,r1

subb a,r3

mov 21h,a

mov 00h,c

end

**c. Write an assembly language program to perform the multiplication oftwo 16-bit numbers.**

mov r0,#34h // 5678\*1234

mov r1,#12h

mov r2,#78h

mov r3,#56h

mov a,r0

mov b,r2

mul ab

mov 33h,a

mov r4,b

mov a,r0

mov b,r3

mul ab

add a,r4

mov r5,a

mova,b

addc a,#00h

mov r6,a

mov a,r1

mov b,r2

mul ab

add a,r5

mov 32h,a

mova,b

addc a,r6

mov 00h,c

mov r7,a

mov a,r3

mov b,r1

mul ab

add a,r7

mov 31h,a

mova,b

addc a,20h

mov 30h,a

end

1. **mov r0,#34h**: This instruction moves the value **34h** (which is hexadecimal for 52 in decimal) into the register **r0**.
2. **mov r1,#12h**: This instruction moves the value **12h** (which is hexadecimal for 18 in decimal) into the register **r1**.
3. **mov r2,#78h**: This instruction moves the value **78h** (which is hexadecimal for 120 in decimal) into the register **r2**.
4. **mov r3,#56h**: This instruction moves the value **56h** (which is hexadecimal for 86 in decimal) into the register **r3**.
5. **mov a,r0**: This instruction moves the value from register **r0** into the accumulator **a**.
6. **mov b,r2**: This instruction moves the value from register **r2** into the register **b**.
7. **mul ab**: This instruction multiplies the values in the accumulator **a** and register **b** and stores the result in the accumulator **a**.
8. **mov 33h,a**: This instruction moves the value from the accumulator **a** into memory location **33h**.
9. **mov r4,b**: This instruction moves the value from register **b** into register **r4**.
10. **mov a,r0**: This instruction moves the value from register **r0** into the accumulator **a**.
11. **mov b,r3**: This instruction moves the value from register **r3** into register **b**.
12. **mul ab**: This instruction multiplies the values in the accumulator **a** and register **b** and stores the result in the accumulator **a**.
13. **add a,r4**: This instruction adds the value in register **r4** to the value in the accumulator **a** and stores the result in the accumulator **a**.
14. **mov r5,a**: This instruction moves the value from the accumulator **a** into register **r5**.
15. **mova,b**: This instruction moves the value from register **b** into the accumulator **a**.
16. **addc a,#00h**: This instruction adds the value **00h** to the value in the accumulator **a** with carry and stores the result in the accumulator **a**.
17. **mov r6,a**: This instruction moves the value from the accumulator **a** into register **r6**.
18. **mov a,r1**: This instruction moves the value from register **r1** into the accumulator **a**.
19. **mov b,r2**: This instruction moves the value from register **r2** into register **b**.
20. **mul ab**: This instruction multiplies the values in the accumulator **a** and register **b** and stores the result in the accumulator **a**.
21. **add a,r5**: This instruction adds the value in register **r5** to the value in the accumulator **a** and stores the result in the accumulator **a**.
22. **mov 32h,a**: This instruction moves the value from the accumulator **a** into memory location **32h**.
23. **mova,b**: This instruction moves the value from register **b** into the accumulator **a**.
24. **addc a,r6**: This instruction adds the value in register **r6** to the value in the accumulator **a** with carry and stores the result in the accumulator **a**.
25. **mov 00h,c**: This instruction moves the carry flag value into memory location **00h**.
26. **mov r7,a**: This instruction moves the value from the accumulator **a** into register **r7**.
27. **mov a,r3**: This instruction moves the value from register **r3** into the accumulator **a**.
28. **mov b,r1**: This instruction moves the value from register **r1** into register **b**.
29. **mul ab**: This instruction multiplies the values in the accumulator **a** and register **b** and stores the result in the accumulator **a**.
30. **add a,r7**: This instruction adds the value in register **r7** to the value in the accumulator **a** and stores the result in the accumulator **a**.
31. **mov 31h,a**: This instruction moves the value from the accumulator **a** into memory location **31h**.
32. **mova,b**: This instruction moves the value from register **b** into the accumulator **a**.
33. **addc a,20h**: This instruction adds the value **20h** to the value in the accumulator **a** with carry and stores the result in the accumulator **a**.
34. **mov 30h,a**: This instruction moves the value from the accumulator **a** into memory location **30h**.
35. **end**: This instruction signifies the end of the program.

To summarize the program, it performs the calculation of **5678 \* 1234** and stores the result in memory locations **33h**, **32h**, **31h**, and **30h**. The intermediate results are stored in registers **r5**, **r6**, and **r7**.