INSTRUCTION SET

ARITHMETIC INSTRUCTIONS

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

ADD A, source byte

Mnemonic	Description	Byte	Cycle
ADD A, #data	Add immediate data to accumulator	2	1
ADD A,Rn	Add register to accumulator	1	1
ADD A,direct	Add direct byte to accumulator	2	1
ADD A,@Ri	Add indirect RAM to .accumulator	1	1

Mnemonic : ADD A, <source-byte>
Function : Add
Flags : OV, AC, CY

- This instruction adds the source byte to the accumulator (A), and place the result in A. Since register A is one byte in size, the source operands must also be one byte.
- The ADD instruction is used for both signed and unsigned numbers.

Examples:
ADD A @ RO ; add to A data pointed to by RO.
ADD A @ RI ; add to A data pointed to by RI.
SUBB A, source byte

Mnemonic	Description	Byte	Cycle
SUBB A, #data	Subtract immediate data from A with borrow.	2	1
SUBB A, Rn	Subtract register from A with borrow.	1	1
SUBB A, direct	Subtract direct byte from A with borrow.	2	1
SUBB A,@Ri	Subtract indirect RAM from A with borrow	1	1

Mnemonic : SUBB A, <src-byte>
Function : Subtract with borrow.
Flags : OV, AC, CY

- This instruction subtracts the source byte and the carry flag from the accumulator and puts the result in the accumulator.

The steps for subtraction performed by the internal hardware of the CPU are as follows:

(i) Take the 2's complement of the source byte.
(ii) Add this to register A.
(iii) Invert the carry.

Example:
MOV A, #45H
CLR C
SUBB A,#23H ;45H-23H-0 = 22H

MUL AB

Mnemonic	Description	Byte	Cycle
MUL AB	Multiply A and B	1	4

Mnemonics : MULAB
Function : Multiply A x B
Flags : OV, CY si

Operation : (A) $_{7-10}$ (A) x (B)
(B)$_{15-8}$
This multiplies an unsigned byte in A by an unsigned byte in register B. The result is placed in A and B where A has the lower byte and B has the higher " byte.
This instruction always clears the CY flag and OV is changed according to the product.

Example:
MOV A, #5
MOV B, #7
MUL AB ; A=35=23H,B=00

DIV

Mnemonic	Description	Byte	Cycle
DIV AB	Divide A by B	1	4

Mnemonics : DIV AB
Function : Divide
Flags : CY and OV
Operation : A (quotient) ←A ←t←B
B (remainder)

- This instruction divides a byte accumulator by the byte in register B. It is assumed that both registers A and B contains an unsigned byte.
- After the division, the quotient will be in register A and the remainder in register B.

Example
MOV A, #35
MOV B, #10
DIV AB ;A = 3andB = 5

LOGICAL INSTRUCTIONS

Logical instructions can perform logical operations upon corresponding bit of two registers like AND, OR, XOR, NOT, Rotate, Clear and Swap. They performed on bytes of data on a bit-by-bit basis. After execution, the result if stored in the first operand.

ANL dest-byte, source-byte

Mnemonic : ANL <dest-byte>, <src-byte>
Function : Logical AND for byte variables
Flags : None affected
This instruction performs a logical AND on the operands, bit by bit and storing the result in the destination. Both the source and destination values are byte-size only.

Example:

```
MOV   A, #39H      ; A = 39H
ANL   A, #09H      ; A = 39H ANDed with 09
      39   0011     1001
      09   0000     1001
      ---------------------
      09   0000     1001
```

ORL dest-byte, source-byte

Mnemonic	Description	Byte	Cycle
ORL A, #data	OR immediate data to accumulator	2	2
ORL A, Rn	OR Register to accumulator	1	1
ORL A, direct	OR direct byte to accumulator	2	1
ORL A, @Ri	OR indirect RAM to accumulator	1	1
ORL direct, # data	OR immediate data to direct byte	3	1
ORL direct A	OR accumulator to direct byte	2	1

Mnemonic : ORL <dest-byte>, <src-byte>
Function : Logical OR for byte variable.
Flags : None
Operation : <dest-byte> ← <dest-byte> ← <src-byte>

Example

```
MOV   A, #32H     ; A = 32H
MOV   R4, #50H    ; R4 = 50H
ORL   A, R4       ; (A = 72H)
```

32H	0011	0010
50H	0101	0000
72H	0111	0010

CLR

Mnemonic : CLR A
Function : Clear accumulator
Flags : None are affected
Operation : A <- 0
This instruction clears register A and all bits of the accumulator are set to 0.

CPL

This, complements the contents of register A (accumulator). The result is the l's complement of the accumulator. That is: Os become Is and Is become Os,

Rotate and Swap Instructions

Mnemonic	Description	Byte	Cycle
RL A	Rotate accumulator left	1	1
RLC A	Rotate accumulator left through carry.	1	1
RR A	Rotate accumulator right	1	1
RRC A	Rotate accumulator right through Carry.	1	1
SWAP A	Swap nibbles within the accumulator	1	1

This instruction will rotate the eight bits in the accumulator by one bit to the left. Bit 7 is rotated into the bit 0 position.