ARITHMETIC GROUP

Special Note:
In the explanation of every instruction, I have mentioned its machine cycles and T-states. You will understand this part once you watch the two videos of timing diagrams

1) ADD R

This instruction adds the contents of register R with the accumulator, stores result in the accumulator.
Eg: ADD B ; A ← A + B

Addr. Mode	Flags Affected	Cycles	T-States
Register	All	1	**4**

2) ADD M

This instruction adds the contents of the memory location pointed by HL, with the accumulator, and stores the result in the accumulator.
Eg: ADD M ; A ← A + [[HL]]

Addr. Mode	Flags Affected	Cycles	T-States
Indirect	All	2	**7**

3) ADI 8-bit data

This instruction adds the immidiate data with the accumulator, and stores the result in A.
Eg: ADI 25 ; A ← A + 25

Addr. Mode	Flags Affected	Cycles	T-States
Immediate	All	2	**7**

4) ADC R

This instruction adds the contents of the register R with the accumulator, and also adds the carry flag, and stores the result in the accumulator. It is used **while adding large numbers**.
Refer example from our video at **www.BharatAcharyaEducation.com**

Eg: ADC B ; A ← A + B + Cy

Addr. Mode	Flags Affected	Cycles	T-States
Register	All	1	**4**

5) ADC M

This instruction adds the contents of the memory location pointed by HL, with the accumulator, and also adds the carry flag, and stores the result in the accumulator.

Eg: ADC M ; A ← A + [[HL]] + Cy

Addr. Mode	Flags Affected	Cycles	T-States
Indirect	All	2	**7**

6) ACI 8-bit data

This instruction adds the immidiate data with the accumulator, and also adds the carry flag, and stores the result in the accumulator.

Eg: ACI 25 ; A ← A + 25 +Cy

Addr. Mode	Flags Affected	Cycles	T-States
Immediate	All	2	**7**

Simillarly subtraction is also done as above.

7) SUB R
8) SUB M
9) SUI 8-bit data
10) SBB R
11) SBB M
12) SBI 8-bit data

Special Note:
During subtraction if a borrow is taken, then carry flag CY becomes 1.

Special Note:
SBB is used to subtract two large numbers like 16-bit numbers.
First, we subtract the lower bytes using SUB
Then, we subtract the higher bytes using SBB to include the borrow taken by the lower byte.

INR R

This instruction increments the contents of the specified register.
The incremented value is stored back in the same register.
Eg: INR B ; B ← B + 1

Addr. Mode	Flags Affected	Cycles	T-States
Register	All except carry	1	4

14) INR M

This instruction increments the contents of memory location pointed by HL pair.
The incremented value is stored back at the same location.
Eg: INR M ; M ← M + 1 i.e. [[HL]] ← [[HL]] +1

Addr. Mode	Flags Affected	Cycles	T-States
Indirect	All except carry	3	10

15) INX Rp

This instruction increments the contents of the specified register **pair**.
The incremented value is stored back in the same register **pair**.
Eg: INX BC ; BC ← BC + 1 i.e. if [BC]=3000 then [BC] becomes 3001.

Addr. Mode	Flags Affected	Cycles	T-States
Register	**NONE**	1	**6**

16) DCR R

This instruction decrements the contents of the specified register.
The decremented value is stored back in the same register.
Eg: DCR B ; B ← B + 1

Addr. Mode	Flags Affected	Cycles	T-States
Register	All except carry	1	4

17) DCR M

This instruction decrements the contents of memory location pointed by HL pair.
The decremented value is stored back at the same location.
Eg: DCR M ; M ← M - 1 i.e. [[HL]] ← [[HL]] - 1

Addr. Mode	Flags Affected	Cycles	T-States
Indirect	All except carry	3	10

18) DCX Rp

This instruction decrements the contents of the specified register **pair**.
The decremented value is stored back in the same register **pair**.
Eg: DCX BC ; BC ← BC - 1 i.e. if [BC]=3001 then [BC] becomes 3000.

Addr. Mode	Flags Affected	Cycles	T-States
Register	**NONE**	1	**6**

19) DAD Rp

This instruction adds the contents of the given register pair with HL pair.
The result is stored in the HL pair.
Eg: DAD B ; HL ← HL + BC

Addr. Mode	Flags Affected	Cycles	T-States
Register	**Only Carry**	3	**10**

20) DAA

This instruction is used to get the answer in BCD form.
It adjusts the result of an addition operation to make the addition work like a decimal addition.
It is implied addressing and works strictly on A register.
It checks the nibbles of A as follows
If LN > 9 or AC = 1 then add 06H, If HN > 9 or CY = 1 then add 60H
#For examples, Please refer Bharat Sir's video fro more on this ...

Addr. Mode	Flags Affected	Cycles	T-States
Implied	**ALL**	1	**4**