ARITHMETIC INSTRUCTIONS

Here you will see operations like:

- Addition
- Addition with Carry
- Subtraction with Borrow
- Increment
- Decrement
- Multiply
- Divide
- Decimal Adjustment after addition.

1) ADD A, #n

| "Add"

Example:

ADD A, #25H; A ← A + 25H

Operation:

Adds A Register with Immediate data.
Stores the result in A Register.

No of cycles required: **1**

IMPORTANT TIP FROM BHARAT ACHARYA
Please do remember to check for carry after performing addition.
Carry flag is checked by instructions like JC and JNC.

---

2) ADD A, Rr

| "Add"

Example:

ADD A, R0; A ← A + R0

Operation:

Adds A Register with the value of a RAM register.
Stores the result in A Register.

No of cycles required: **1**

---

3) ADD A, addr

| "Add"

Example:

ADD A, 25H; A ← A + [25H]

Operation:

Adds A Register with the contents of the address.
Stores the result in A Register.

No of cycles required: **1**

---

4) ADD A, @Rp

| "Add"

Example:

ADD A, @R0; A ← A + [R0]

Operation:

Adds A Register with the contents of the location pointed by the register.
Result is stored in A Register.

If R0 = 20H and Location 20H contains value 35H, then 35H will be added to A register.

No of cycles required: **1**

5) ADDC A, #n

| "Add with carry"

Example:

ADDC A, #25H; A ← A + 25H + Carry Flag

Operation:

Adds A Register with Immediate data along with the Carry of the previous addition which is present in the Carry Flag. Stores the result in A Register.

No of cycles required: **1**

IMPORTANT TIP FROM BHARAT ACHARYA

ADDC is used when we want to ADD two large numbers like 16 bit numbers.
First we add the lower bytes using ADD instruction.
Then we add the higher bytes using ADDC instruction.
If the Lower byte has produced a Carry, then CF will be 1.
This Carry will be added into the higher bytes.
Please refer to our classroom example of adding 12FFH + 0001H = 1300H.

---

6) ADDC A, Rr

| "Add with carry"

Example:

ADDC A, R0; A ← A + R0 + Carry Flag

Operation:

Adds A Register with the value of a RAM register along with the Carry of the previous addition. Stores the result in A Register.

No of cycles required: **1**

---

7) ADDC A, addr

| "Add with carry"

Example:

ADDC A, 25H; A ← A + [25H] + Carry Flag

Operation:

Adds A Register with the contents of the address along with the Carry of the previous addition. Stores the result in A Register.

No of cycles required: **1**

---

8) ADDC A, @Rp

| "Add with carry"

Example:

ADDC A, @R0; A ← A + [R0] + Carry Flag

Operation:

Adds A Register with the contents of the location pointed by the register along with the Carry of the previous addition. Result is stored in A Register.

No of cycles required: **1**

9) SUBB A, #n | "Subtract with borrow"

Example:

SUBB A, #25H; A ← A - 25H - Carry Flag

Operation:

Performs A Register – Immediate data – Carry Flag (Carry Flag holds the borrow of the previous subtraction). Stores the result in A Register.

No of cycles required: **1**

IMPORTANT TIP FROM BHARAT ACHARYA

SUBB is used when we want to Subtract two large numbers like 16 bit numbers.
First we Subtract the lower bytes.
If the Lower byte subtraction needs a borrow, then CF will be 1.
This Carry will be subtracted from the higher bytes.
It is important to realize that there is no ordinary SUB instruction.
Hence if we want to perform simple 8-bit subtraction, we still have to use SUBB instruction.
If we don't want Carry flag to interfere with the operation,
We must first clear the carry flag using CLR C instruction before using SUBB.

---

10) SUBB A, Rr | "Subtract with borrow"

Example:

SUBB A, R0; A ← A - R0 - Carry Flag

Operation:

Performs A Register – value of RAM register – Carry Flag (Carry Flag holds the borrow of the previous subtraction). Stores the result in A Register.

No of cycles required: **1**

---

11) SUBB A, addr | "Subtract with borrow"

Example:

SUBB A, 25H; A ← A - [25H] - Carry Flag

Operation:

Performs A Register – contents of memory address – Carry Flag (Carry Flag holds the borrow of the previous subtraction). Stores the result in A Register.

No of cycles required: **1**

---

12) SUBB A, @Rp | "Subtract with borrow"

Example:

SUBB A, @R0; A ← A - [R0] - Carry Flag

Operation:

Performs A Register – Contents of the memory location pointed by the register – Carry Flag. Result is stored in A Register.

No of cycles required: **1**

13) INC A | "Increment"

Example:

INC A; A ← A + 1

Operation:

Increments the value of A register. Stores the result in A Register.

No of cycles required: **1**

IMPORTANT TIP FROM BHARAT ACHARYA
During INC A, if A was FFH, it will roll over to 00H.

---

14) INC Rr | "Increment"

Example:

INC R0; R0 ← R0 + 1

Operation:

Increments the value of a RAM register. Stores the result in the same RAM Register.

No of cycles required: **1**

---

15) INC addr | "Increment"

Example:

INC 25H; [25H] ← [25H] + 1

Operation:

Increments the contents of a memory address. Stores the result back in the same location.

No of cycles required: **1**

---

16) INC @Rp | "Increment"

Example:

INC @R0; [@R0] ← [@R0] + 1

Operation:

Increments the contents of a memory location pointed by R0. Will store the result back at the same location.

No of cycles required: **1**

---

17) INC DPTR | "Increment"

Example:

INC DPTR; DPTR ← DPTR + 1

Operation:

Increments the 16-bit value of DPTR register. Stores the result in DPTR Register.

No of cycles required: **2**

18) DEC A

| "Decrement"

Example:

DEC A; A ← A – 1

Operation:

Decrements the value of A register.
Stores the result in A Register.

No of cycles required: **1**

IMPORTANT TIP FROM BHARAT ACHARYA
During DEC A, if A was 00H, it will roll back to FFH.
That's because 00H – 1 = – (01H) which is FFH in 2's complement form.
Please refer to classroom explanation for more clarity on this.
Also, do remember, **DEC DPTR does not exists.**
If you need to decrement DPTR, you can do it by individually decrementing DPL and DPH.
First decrement DPL. If it rolls back from 00H to FFH, then also decrement DPH.
E.g.:: 1300H – 1 = 12FFH.

---

19) DEC Rr

| "Decrement"

Example:

DEC R0; R0 ← R0 – 1

Operation:

Decrements the value of a RAM register.
Stores the result in the same RAM Register.

No of cycles required: **1**

---

20) DEC addr

| "Decrement"

Example:

DEC 25H; [25H] ← [25H] – 1

Operation:

Decrements the contents of a memory address.
Stores the result back in the same location.

No of cycles required: **1**

---

21) DEC @Rp

| "Decrement"

Example:

DEC @R0; [@R0] ← [@R0] – 1

Operation:

Decrements the contents of a memory location pointed by R0.
Will store the result back at the same location.

No of cycles required: **1**

22) MUL AB

| "Multiply A and B"

Example:

MUL AB; B (Higher) . A (Lower) ← A x B

Operation:

Multiples the 8-bit values of A register and B register.
Stores the 16 bit result in B and A Registers.
B register gets the Higher Byte, A register gets the Lower Byte.

No of cycles required: **4**

---

23) DIV AB

| "Divide A by B"

Example:

DIV AB; B (Remainder) . A (Quotient) ← A ÷ B

Operation:

Divides the 8-bit value of A register by the 8-bit value of B register.
Stores the result in B and A Registers.
B register gets the Remainder, A register gets the Quotient.

No of cycles required: **4**

IMPORTANT TIP FROM BHARAT ACHARYA
During DIV AB, if B register is 00H, then the instruction will be aborted.
A and B registers will contain garbage values after the operation.
This is indicated to the programmer by the **OVERFLOW FLAG**.
If Overflow Flag becomes "1" after the operation, it simply means division by 0 was attempted.

---

24) DA A

| "Decimal Adjust after Addition"

Example:

DA A;

Operation:

It is used when we want to add two decimal numbers (BCD numbers).
We first enter the decimal numbers, as if they are Hexadecimal.
We add them by normal ADD instruction.
The answer is then adjusted using DA A instruction.

DA A always works on A Register only.

It first checks the Lower nibble of A Register.
If Lower nibble is > 9 or Aux Carry is 1, then ADD 06H

It then checks the Higher nibble of A Register.
If Higher nibble is > 9 or Carry Flag is 1, then ADD 60H

The final answer will be stored in A and Carry flag.

Please refer numerous examples discussed in the class. For doubts call #BharatSir @9820408217

Assume we want to add 25d + 25d, the result must obviously be 50d.

We enter the numbers as if they are hexadecimal, and add them by normal ADD instruction.

```
MOV A, #25H   ; A ← 25
ADD A, #25H   ; A ← 4A
```

Now we perform the adjustment using DAA instruction.

```
DA A          ; A ← 50
```

	24H + 25H		26H + 26H		28H + 28H		50H + 50H		80H + 80H		99H + 99H
After ADD	= 49H	After ADD	= 4CH	After ADD	= 50H	After ADD	= A0H	After ADD	= 100H	After ADD	= 132H
After DA A	→ 49	After DA A	→ 52	After DA A	→ 56	After DA A	→ 100	After DA A	→ 160	After DA A	→ 198

No of cycles required: **1**

IMPORTANT TIP FROM BHARAT ACHARYA

Please get this clear, "DA A does not convert any number from Hexadecimal to Decimal".
It simply makes the addition work like decimal addition.

DA A can only be used "After performing an Addition" operation.

VIVA question: Put 25H in A register and show the working of DA A.
Reply: Invalid question! We must first perform addition. Simply putting 25H in A and doing DA A is absurd.

VIVA question: 29H and 3CH and show the working of DA A.
Reply: Invalid question! DA A is used to Add decimal numbers. 3C is not decimal!

Also, do remember, DA A is an adjustment for Addition. So there is always a chance of a carry.
If the answer exceeds 99, the lower two digits will be in A and the highest digit will be Carry Flag.
If the answer is 160, A will contain 60 and Carry Flag will get 1.