TYPES OF REGISTERS

BSCS

Memory locations are locations of external main memory (RAM). While, register are the fastest location inside the microprocessor (ax, bx, cx, dx, ss, cs, si, and di etc)

Mnemonic is a symbolic representation e.g. AX, BX, ADD, MOV, DB and DW etc. While, opcode is a mnemonic that performs the operation e.g. MOV, ADD, SUB, PUSH etc

Types Of Registers

14 Types Of Register

Types of Registers A/c to Task.

1). Accumulator

Used for I/O operations.

(Give it I/p and used it anytime for operations.)

Save the value we can use it for any operations.

∴ 16, 32, 64 bits of System (Basically size of Register)

∴ 16 bit means, how many digits are at a time are in register.

1 digit ⟶ 1 bit Unit

∴ Press A from keyboard, character signal (code) Binary 16 Digit at a time move for 16 bit register, same as 32, 64

- 1971, CPU by Intel 4004 first register size is 8 bits name as A for accumulator.
- For 16 bit signal.

10101010	01010101
High	Low

- First low part load in accumulator, it procened then high part load in accumulator because of 8 bit size.
- Then size of register increased from 8 bits to 16 bits.
- Its name changes to ax (16 bit size).
- Problem → If only low part is send to 16 bits because size is 16 bits, how it is load.

- In which direction it is load.
  ax divided into ah (high) and al (low).

ah	al

- To increase speed of CPU, 32 bit system a/c to Accumulator register Eax.
- For 64 bit system, named Rax.

a = 8 bits
ax = 16 bits.

X = Extended to 16 bits
E = extended to 32 bits
R = Rich register to 64 bits.

Rax.	Eax	ah	al

Used for I/O operations.
It save the value we use for any operations.

2 Base Register

- Hold address of data which is in RAM.
- In programing we need data that we used later, Used Base register for that.
- For process big data some is in RAM and some in CPU, It hold address of data which is in RAM.

b, bx, ebx, rbx.

3 Counter Register

c, cx, ecx, rcx.

Counts, used in loops.

loop, how many times program run, how long it run, Counting of program run time.

4. Data Register d, dx, edx, rdx.

- Important work, final register it hold data to be print on screen.
- It hold data at last, accumulator take data from it & print ~~[illegible]~~ ~~[illegible]~~
- Hold Output Data.

These four are general types of Register, used for different purposes.

1) Accumulator :- • take I/P & give O/P, • take part in Operations.

2) Base Register :- • Hold address of data.
• Add value to it

- Divide h and l for high and low, used complete if 16 bits by using x.
- These 4 registers can be divided.

→ Other register work is to hold address or point, these registers are not divided, the work is not general.

5. Code Segment Not divided, not general purpose.

CS.

When we write program, One part is code where we write code, second is data, 3rd part is management how we access memory.

We write program, Code is written in segment it is in RAM, its addres hold by code segment.

- Hold address of code segment.

CS

6. Data Segment :- Hold address of data segment.

- We not used it, it only hold address which are in RAM.

ds

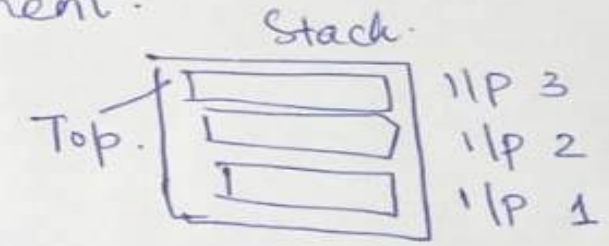
- In program, part of data which is in Road in RAM, it hold address of it.

7. Stack Segment :-

SS

- Management of data to be extract earlier or later.

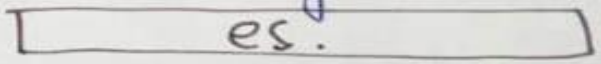
- Hold address of stack segment.
- Stack segment management in Memory. ⟶



Top is extract first, this portion is in RAM, its address hold by stack.

(8) Extra Segment (es) :-

If code segment is full, we take extra space in RAM to write code, its address hold by extra segment.



These four are called Segment Registers.

(9) Source Index Points the source operand.

ADD dl, bl → dl and bl both register
D S bl will add in dl

ADD 3, bl → 3 constant, bl register
S D 3 add in bl.

ADD dl, 3 → "
D S

Source Index tell the CPU what is Source Value.

(10) Destination Index :- Points the destination Operand.

These two are called Index Registers

(11). Instruction pointer :- Hold the next instruction.

This is running

Hold the address of next instruction to be run.

Add dl, bl
Add 3, bl
Add bl, 3.

(12). Stack pointer :- Points current top of stack.

There two are called Special purpose register

(13). Flag Register :- Hold current status of program.

Add two number.

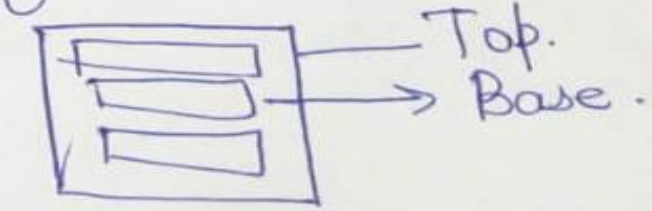
$$\begin{array}{r} (1)\ 1 \\ 1 \\ \hline 0 \end{array}$$ Sum

Carry → It goes to flag register (current status what is working).

→ Parity, interrupt call, these different status control by flag register.

(14) Base Register → Base of the top of stack.

It points



Addressing Modes:-

Perform any task in programming, first it load on RAM, then go to register, from where CPU run it.

- How CPU access it (How CPU access address of that data).
- Ways or Models to Access data:
- Data present in Reg & Memory How to Access?
- Any address through which we access it.

e.g. 2+2 (Number to Add, where to place these two numbers to perform Add).

Operands

(Then CPU access its position and action to perform.

→ Opcode (operational code).

In Assembly language where to kept them & Process them?

If we place both 2 in different register, suppose,

Opcode Reg1, Reg2

$\boxed{\text{Add Dl, Al}}$

- Reg are General purpose. Dl and Al (Data & Acc.).
- One 2 in Data & other is in Acc Register & we Perform Addition

~~Two~~ Both operands are registers (2 different registers).

This Addressing is called Register Addressing

Add Dl, 2

One value is register, Second value is direct 2.

In assembly we give direct value b/c we directly Access CPU

One register used, One Constant Value Used.
This addressing is called <u>Immediate Addressing</u>.
One operand is constant.

Opcode Reg, Value

- When we not given direct value, second value is in memory in RAM in static Memory.
- Data in static Memory is called static Data
- One value is in Data Register & other is in RAM (Static data).

Add DL, [address]

Access static data directly is called <u>Memory Addressing</u>

Opcode Reg, [Address]

These 3 way of addressing. Used this addressing to Access data, CPU access like this, In assembly language we write code and give instruction to be acceptable. Method of data access, placed used in assembly. In instruction one Reg is must present, destination where value comes.

How to write Instruction A/c to ease and requirement.

Q. How 2 is placed in any location, how it go to any location. how data transfer instruction write to transfer 2 to data register.

We used Data transfer Instruction.

Mov ⟶

Move 2 to data register when data is in register then add is performed. data is sent to register by data transfer Instruction.

Mov DL, 2 (2 is sent to Data Reg).

Want to print the value, the register used is Accumulator (used for I/O).

Mov DL, 2
Mov Ah, 2

Acc print the value present in Data Reg DL.

Mov Ah, 2 → 2 Means print, 2 is given as a function in Acc what to do, It is called Service Routine

Service Routine, for given input, we used 1.

e.g. Press A in keyboard and show A in screen.

1 = input a character with echo.

2 = Output /print a single character

we can see (echo)

Without echo, keyboard press and not appear in screen.

8 = Input a character w/o echo.

9 = Print collection of characters 'abcd'

4ch = Exit (to exit from register)

For Input

```
Mov ah, 1
INT 21h
```

For Output

```
Mov ah, 2
INT 21h.
```

→ Accumulator reg. understand 1 and 2 as I/O, understand it as a function.

→ What appear on screen, How CPU access hardware (Screen) to show number, We want all function of CPU stop and show this number, means we Interrupt CPU show this character, In assembly. we write Interrupt.

Interrupt → CPU stop, H/W access.

"Stop the current program & allow microprocessor to access h/w to take I/P or give O/P."

In assembly we used interrupt after these commands.

CPU busy in other work we interrupt it to point.

INT 21H = Interrupt for Text Handling.

INT 20H = " " Video/Graphics Handling.

21H is number to call interrupt, stop give input or output.

Q. What Appears on Screen? O/P?

Smiley face is show in place of 2.

why, It give ASCII code.

ASCII is an character encoding scheme.

CPU is of diff companies, International Standard code for Keyboard, Every character has number.

Each character has number, signal mean 'a' first ~~age~~ binary signal generate 0101.
Every key has different code, Every PC work on that code, so Internationally we access them.
In place of 2, ASCII code should given, Every key has ASCII code.
For print A, ASCII code is 65 → In decimal.
By default assembly used decimal.
B = 66, Z = 90.

Capital A = 65 ..... Z = 90.
Small a = 91, ... z = 122.
Number 0 = 48, ... 9 = 57
} Only start of there are remember

To print 2, we give its code 50. 1 means ASCII code

Mov DI, 'A'.

A not more, A more as a 65, it print as A'.

- Ascii code used for I/P and O/P.
- ~~Next feed~~ Next line Feed = 10. , move to next line (It comes to same position in next line)

For complete Enter key.

We used both, Next line Feed = 10
Carriage Return = 13] For Complete Enter key.