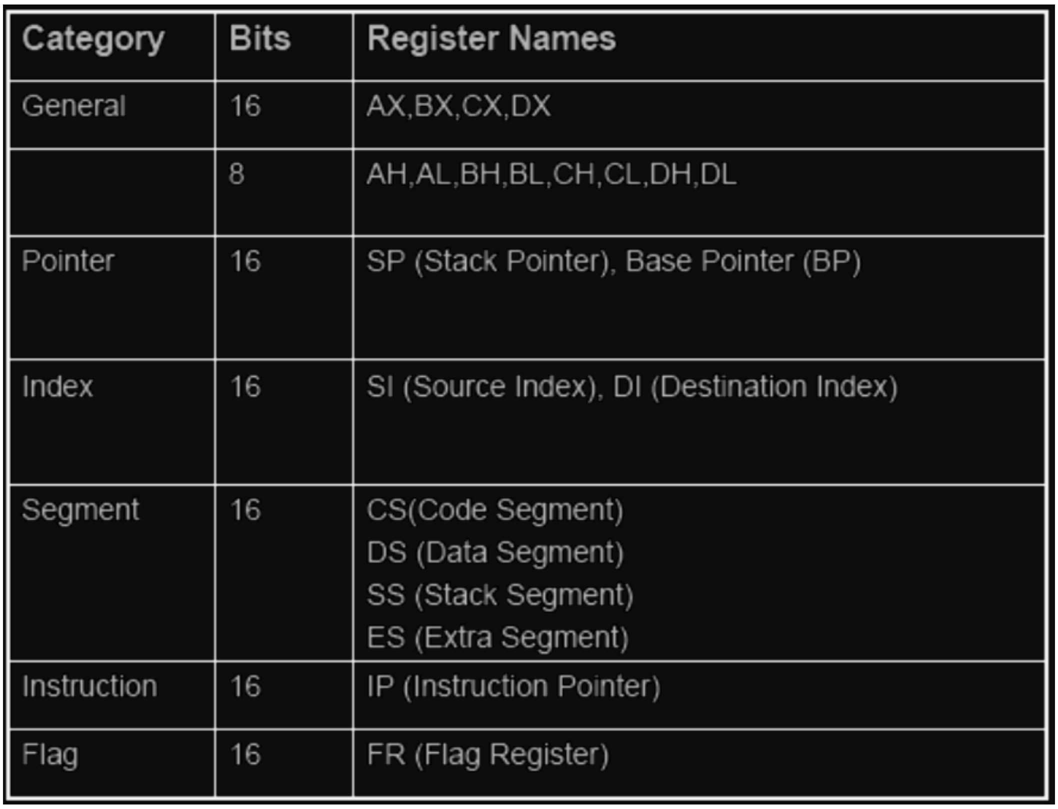
REGISTERS IN x86

Control & Status Registers

- Program Counter (PC)
- Instruction Register(IR)
- Memory Address Register(MAR)
- Memory Buffer Register(MBR)



Category	Bits	Register Names
General	16	AX,BX,CX,DX
	8	AH,AL,BH,BL,CH,CL,DH,DL
Pointer	16	SP (Stack Pointer), Base Pointer (BP)
Index	16	SI (Source Index), DI (Destination Index)
Segment	16	CS(Code Segment) DS (Data Segment) SS (Stack Segment) ES (Extra Segment)
Instruction	16	IP (Instruction Pointer)
Flag	16	FR (Flag Register)

*IP is also part of Pointer

What are general purpose registers , advantages, different ways that they can be used?

- **AX is the primary accumulator**; it is used in input/output and most arithmetic instructions. For example, in multiplication operation, one operand is stored in EAX or AX or AL register according to the size of the operand.

- **BX is known as the base register**, as it could be used in indexed addressing.

- **CX is known as the count register**, as the ECX, CX registers store the loop count in iterative operations.

- **DX is known as the data register**. It is also used in input/output operations. It is also used with AX register along with DX for multiply and divide operations involving large values.

- **Instruction Pointer (IP)** − The 16-bit IP register stores the offset address of the next instruction to be executed. IP in association with the CS register (as CS:IP) gives the complete address of the current instruction in the code segment.

- **Stack Pointer (SP)** − The 16-bit SP register provides the offset value within the program stack. SP in association with the SS register (SS:SP) refers to be current position of data or address within the program stack.

- **Base Pointer (BP)** − The 16-bit BP register mainly helps in referencing the parameter variables passed to a subroutine. The address in SS register is combined with the offset in BP to get the location of the parameter. BP can also be combined with DI and SI as base register for special addressing.

- **Source Index (SI)** − It is used as source index for string operations.

- **Destination Index (DI)** − It is used as destination index for string operations.

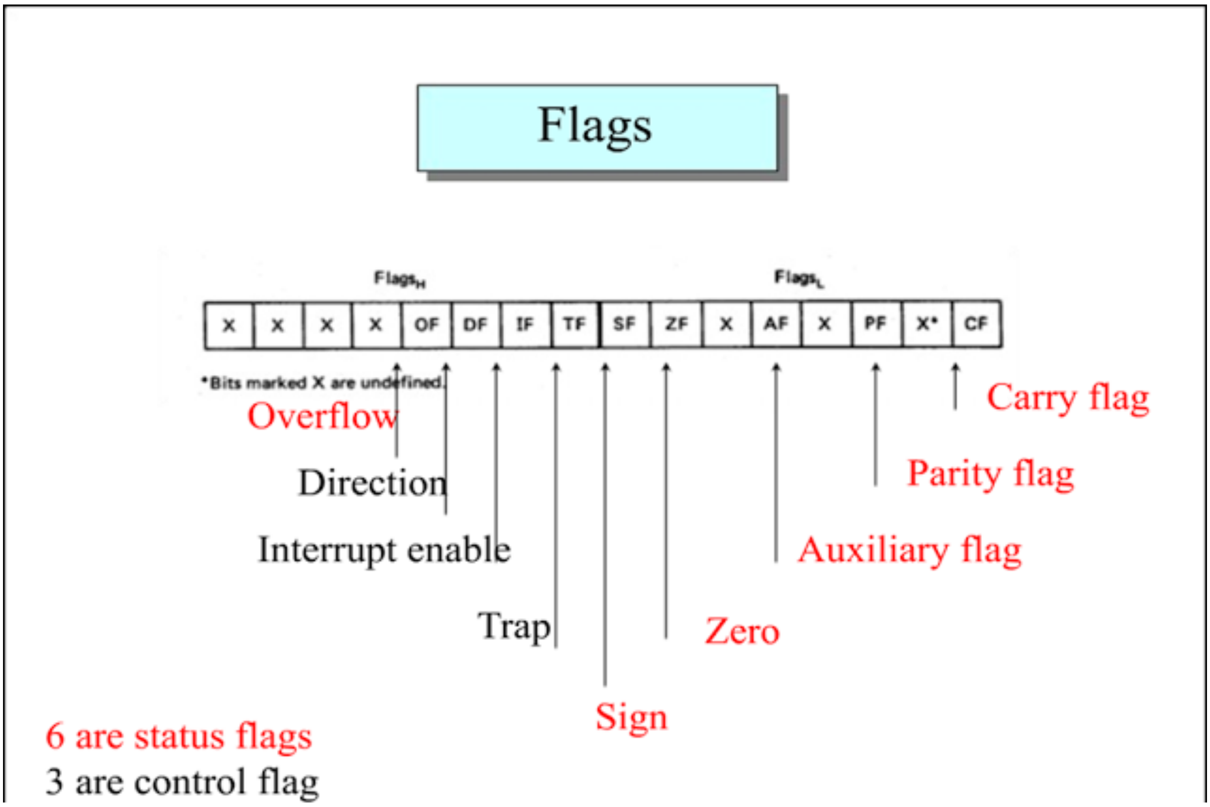
- **Flag Register-** is a 16-bit register in the Intel 8086 microprocessor that contains information about the state of the processor after executing an instruction. It is sometimes referred to as the status register because it contains various status flags that reflect the outcome of the last operation executed by the processor.
  The flag register is an important component of the 8086 microprocessor because it is used to determine the behaviour of many conditional jump and branch instructions.

  (a) Status Flags – There are 6 flag registers in 8086 microprocessor which become set(1) or reset(0) depending upon condition after either 8-bit or 16-bit operation. These flags are conditional/status flags.

  The 6 status flags are:
  Sign Flag (S)
  Zero Flag (Z)
  Auxiliary Carry Flag (AC)
  Parity Flag (P)
  Carry Flag (CY)
  Overflow Flag (O)

  (b) Control Flags – The control flags enable or disable certain operations of the microprocessor. There are 3 control flags in 8086 microprocessor and these are

  Directional Flag (D)
  Interrupt Flag (I)
  Trap Flag (T)



Flags

Flags_H Flags_L

X	X	X	X	OF	DF	IF	TF	SF	ZF	X	AF	X	PF	X*	CF

*Bits marked X are undefined.

Overflow

Direction

Interrupt enable

Trap

Sign

Zero

Auxiliary flag

Parity flag

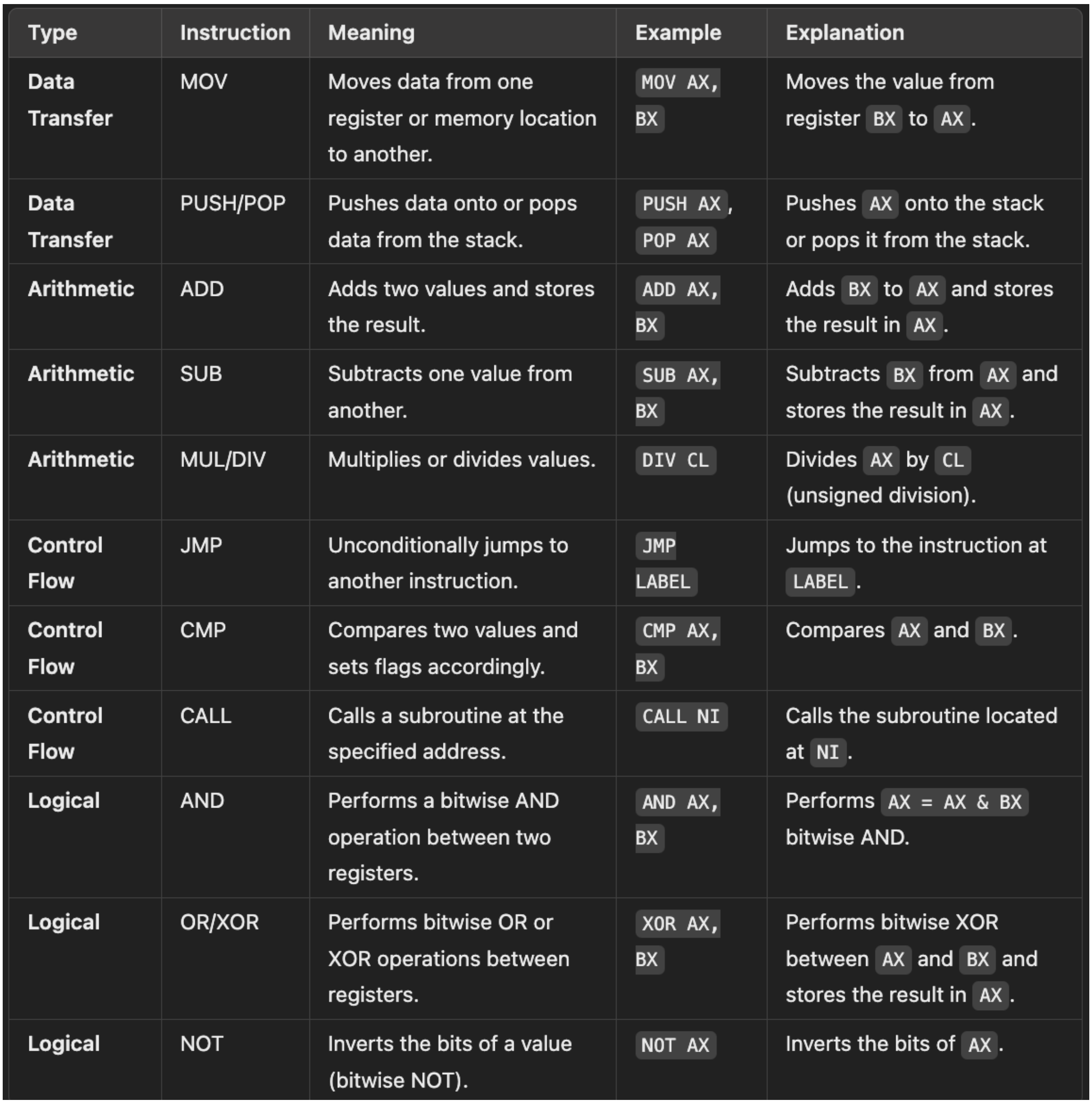
Carry flag

6 are status flags

3 are control flag

***Read each of these flags in detail from ppt

ASSEMBLY LANGUAGE / MEANING OF A PARTICULAR INSTRUCTIONS



Type	Instruction	Meaning	Example	Explanation
Data Transfer	MOV	Moves data from one register or memory location to another.	`MOV AX, BX`	Moves the value from register `BX` to `AX`.
Data Transfer	PUSH/POP	Pushes data onto or pops data from the stack.	`PUSH AX`, `POP AX`	Pushes `AX` onto the stack or pops it from the stack.
Arithmetic	ADD	Adds two values and stores the result.	`ADD AX, BX`	Adds `BX` to `AX` and stores the result in `AX`.
Arithmetic	SUB	Subtracts one value from another.	`SUB AX, BX`	Subtracts `BX` from `AX` and stores the result in `AX`.
Arithmetic	MUL/DIV	Multiplies or divides values.	`DIV CL`	Divides `AX` by `CL` (unsigned division).
Control Flow	JMP	Unconditionally jumps to another instruction.	`JMP LABEL`	Jumps to the instruction at `LABEL`.
Control Flow	CMP	Compares two values and sets flags accordingly.	`CMP AX, BX`	Compares `AX` and `BX`.
Control Flow	CALL	Calls a subroutine at the specified address.	`CALL NI`	Calls the subroutine located at `NI`.
Logical	AND	Performs a bitwise AND operation between two registers.	`AND AX, BX`	Performs `AX = AX & BX` bitwise AND.
Logical	OR/XOR	Performs bitwise OR or XOR operations between registers.	`XOR AX, BX`	Performs bitwise XOR between `AX` and `BX` and stores the result in `AX`.
Logical	NOT	Inverts the bits of a value (bitwise NOT).	`NOT AX`	Inverts the bits of `AX`.

TYPE OF ADDRESSING MODE	EXAMPLE
Immediate Addressing	Mov CL,#30
Register Addressing	MOV AX,BX ADD AX,BX
Memory: Direct Addressing	ADD AX,[1234h]
Memory: Register Indirect Addressing	MOV [SI],AL ADD AL, [SI] *Memory cell pointed to by address field contains the address of (pointer to) the operand
Memory: Based Addressing Mode	MOV AL,[BX+5] *Same as register indirect ,difference is an 8 or 16 bit displacement may be included in the operand field.
Memory: Indexed Addressing Mode	MOV AL,[SI + 5] *Like based addressing ,indexed addressing allows the use of signed displacement. Index registers SI or DI must be used in the operand field.
Memory: Based Indexed Addressing Mode	ADD AX,[BX][SI] *Contains the featurres of based and indexed addressing but **does not** allow use of displacement.
Memory: Based Indexed with displacement Addressing Mode	ADD AX,[BX][SI + 20] *The EA of the operand is found by adding the contents of the base register and index register with 8 or 16 bit displacement.
String Addressing	MOVSB

***Read about each instruction meaning from ppt**