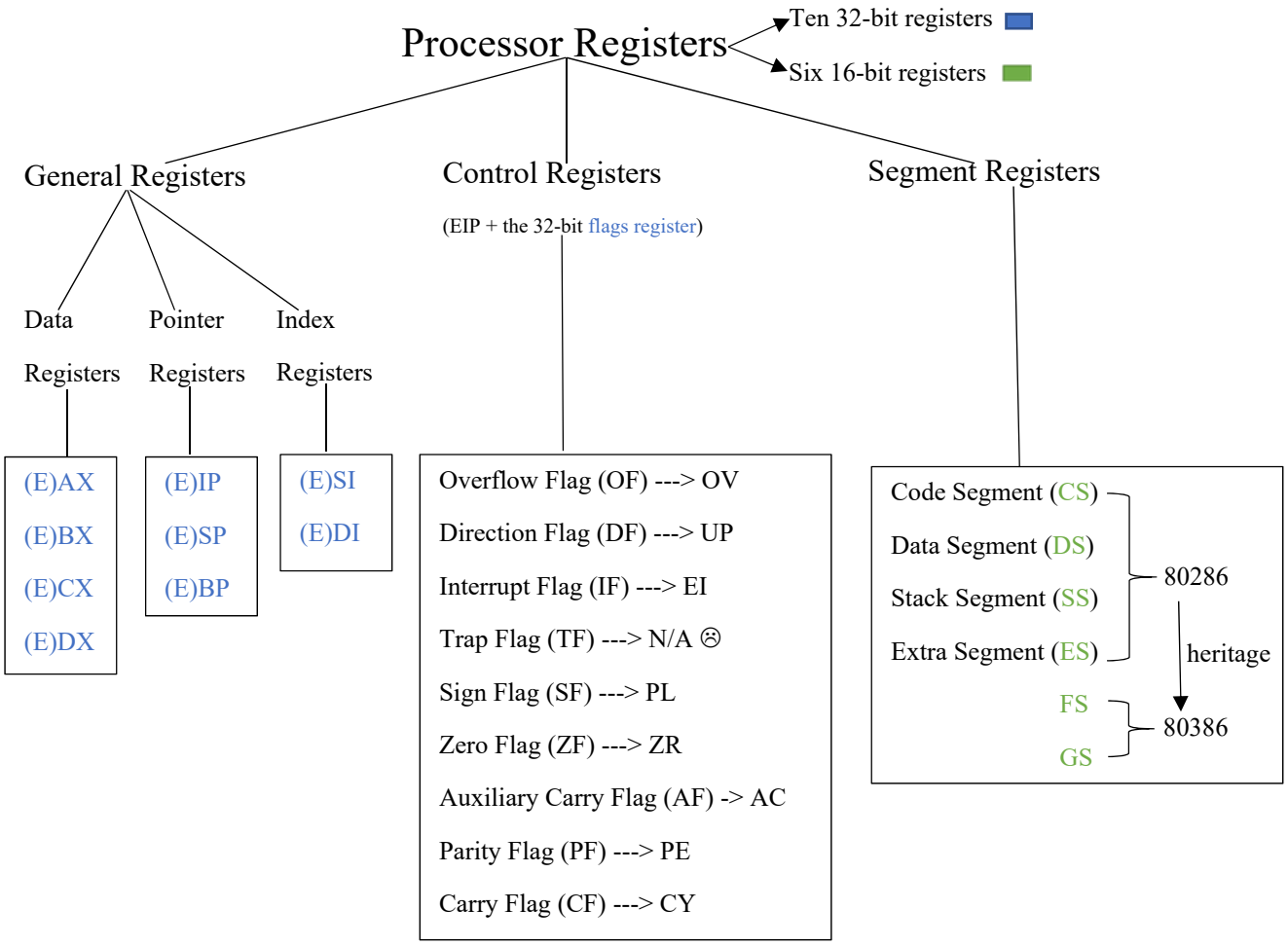
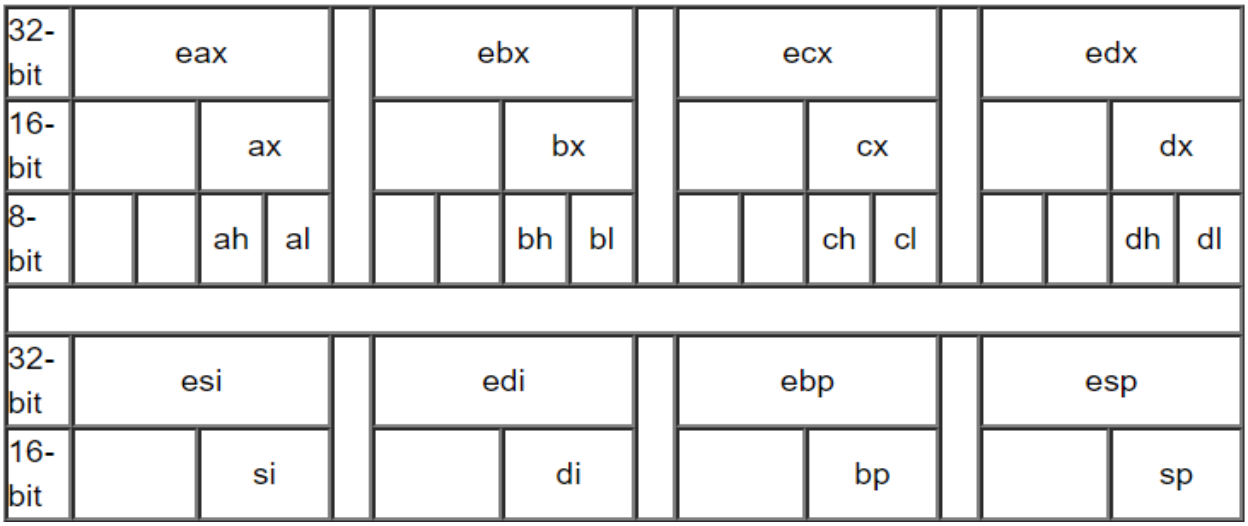

Processor Registers
Ten 32-bit registers
Six 16-bit registers
General Registers
Control Registers
(EIP + the 32-bit flags register)
Segment Registers
Data Registers
Pointer Registers
Index Registers
(E)AX
(E)BX
(E)CX
(E)DX
(E)IP
(E)SP
(E)BP
(E)SI
(E)DI
Overflow Flag (OF) ---> OV
Direction Flag (DF) ---> UP
Interrupt Flag (IF) ---> EI
Trap Flag (TF) ---> N/A ☹
Sign Flag (SF) ---> PL
Zero Flag (ZF) ---> ZR
Auxiliary Carry Flag (AF) -> AC
Parity Flag (PF) ---> PE
Carry Flag (CF) ---> CY
Code Segment (CS)
Data Segment (DS)
Stack Segment (SS)
Extra Segment (ES)
80286
heritage
FS
GS
80386


32-bit
eax
ebx
ecx
edx
16-bit
ax
bx
cx
dx
8-bit
ah
al
bh
bl
ch
cl
dh
dl
32-bit
esi
edi
ebp
esp
16-bit
si
di
bp
sp

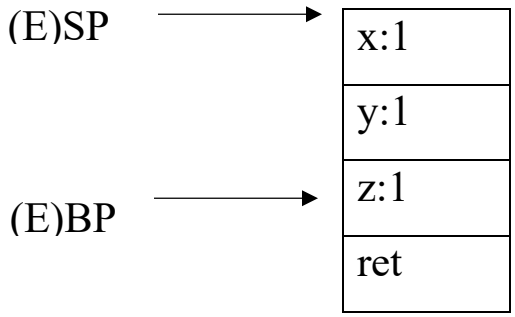
1. **Data Registers** (32-bit/16-bit/8-bit)

- (E)AX: (Extended) Accumulator register: addition/multiplication
  add eax,-2 // value in eax + (-2)D
- (E)BX: (Extended) Base address register: memory address
- (E)CX: (Extended) Count register: used in "LOOP"/ "REP"
- (E)DX: (Extended) Data register: remainder storage in division

2. **Pointer Registers** (32-bit/16-bit)

- (E)BP: (Extended) Base (Frame) pointer--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
- (E)SP: (Extended) Stack pointer--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
- (E)IP: (Extended) Instruction Pointer (IP)--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

```
int demo() {
    int x=1;
    int y=1;
    int z=1;
    return 0;
}
```



3. **Index Registers** (32-bit/16-bit)

- (E)SI: (Extended) Source index: string operation
- (E)DI: (Extended) Destination index: string operation

4. **Segment Registers** (16-bit)

- CS: Code Segment − contains all the instructions to be executed. A 16-bit Code Segment register or CS register stores the starting address of the code segment.
- DS: Data Segment − contains data, constants and work areas. A 16-bit Data Segment register or DS register stores the starting address of the data segment.
- SS: Stack Segment − contains data and return addresses of procedures or subroutines. It is implemented as a 'stack' data structure. The Stack Segment register or SS register stores the starting address of the stack.
- ES: Extra Segment
- FS
- GS

5. **Control Registers**

- **Overflow Flag (OF)** − It indicates the overflow of a high-order bit (leftmost bit) of data after a signed arithmetic operation.
- **Direction Flag (DF)** − It determines left or right direction for moving or comparing string data. When the DF value is 0, the string operation takes left-to-right direction and when the value is set to 1, the string operation takes right-to-left direction.
- **Interrupt Flag (IF)** − It determines whether the external interrupts like keyboard entry, etc., are to be ignored or processed. It disables the external interrupt when the value is 0 and enables interrupts when set to 1.
- **Trap Flag (TF)** − It allows setting the operation of the processor in single-step mode. The DEBUG program we used sets the trap flag, so we could step through the execution one instruction at a time.
- **Sign Flag (SF)** − It shows the sign of the result of an arithmetic operation. This flag is set according to the sign of a data item following the arithmetic operation. The sign is indicated by the high-order of leftmost bit. A positive result clears the value of SF to 0 and negative result sets it to 1.
- **Zero Flag (ZF)** − It indicates the result of an arithmetic or comparison operation. A nonzero result clears the zero flag to 0, and a zero result sets it to 1.
- **Auxiliary Carry Flag (AF)** − It contains the carry from bit 3 to bit 4 following an arithmetic operation; used for specialized arithmetic. The AF is set when a 1-byte arithmetic operation causes a carry from bit 3 into bit 4.
- **Parity Flag (PF)** − It indicates the total number of 1-bits in the result obtained from an arithmetic operation. An even number of 1-bits clears the parity flag to 0 and an odd number of 1-bits sets the parity flag to 1.
- **Carry Flag (CF)** − It contains the carry of 0 or 1 from a high-order bit (leftmost) after an arithmetic operation. It also stores the contents of last bit of a *shift* or *rotate* operation

Flag:					O	D	I	T	S	Z		A		P		C
Bit No.:	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

References:

1. *https://docs.microsoft.com/en-us/previous-versions/dotnet/netframework-3.0/kwydd1t7(v=vs.85)*
2. *https://www.tutorialspoint.com/assembly_programming/assembly_registers.htm*