Architecture of 8086 µp.

The 8086 µp is devided into two independant functional units These are as follows.

1. Bus ~~interf~~ Bus interface unit
2. Execution unit.

Following diagram shows internal Architecture of 8086 µp.

Draw Here diagram.

BIU :- Bus Interface unit

Bus interface unit is basically responsible for

→ Fetch the instruction or data from memory.
→ Write data to memory.
→ Write data to port.
→ Read data from port.

The BIU contains Bus interface logic, segment registers, Address compute engine, and instruction Stream byte queue. BIU transfer data and address on the bus for execution unit. In other word BIU fetches data and addresses from memory and sends it for execution unit for further processing. The seveal blocks of Bus interface unit are as follows.

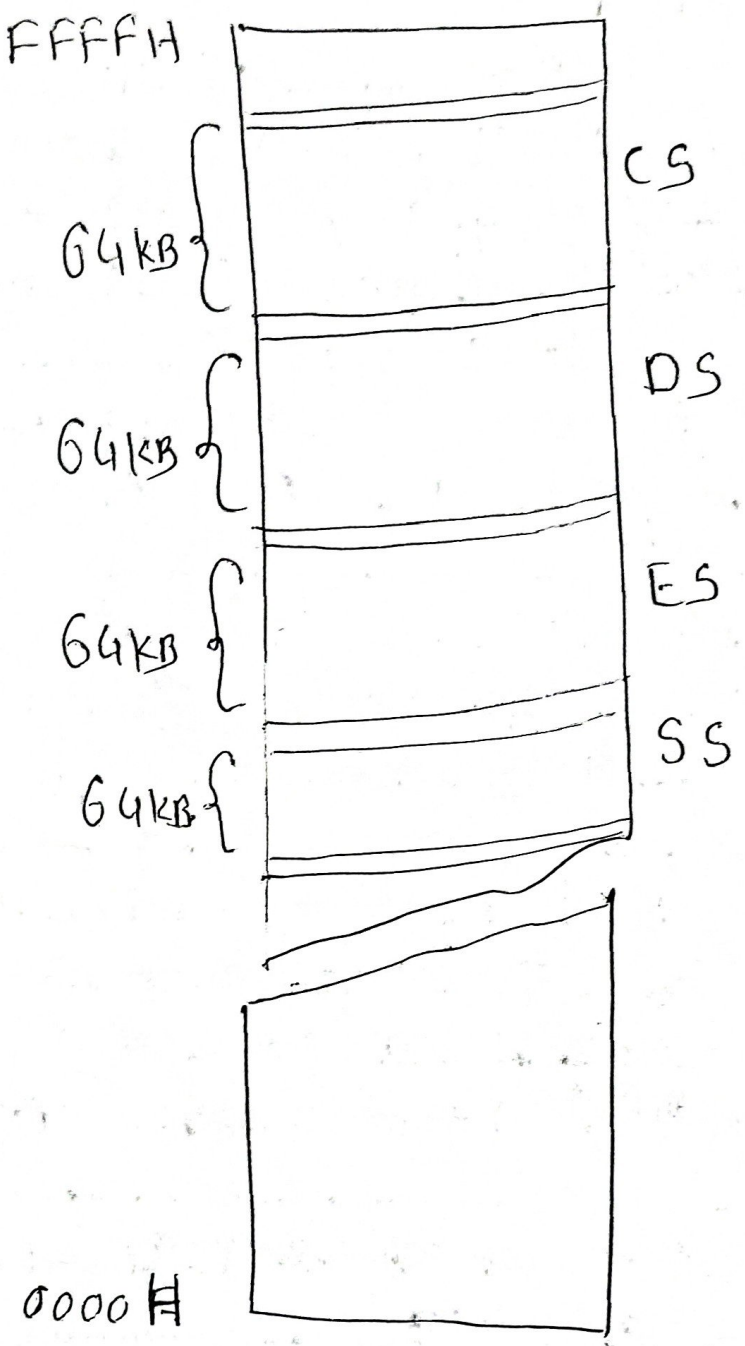


Fig:- Segmented Memory

Segment registers :- The 8086 mp use 1MB memory this memory is devided in to 4 parts of 64KB these parts of memory or partitions of memory is known as segment register. there are 4 seg. registers and each segment register of size 64KB and having specific purpose. These segments can be located any where within 1 MB memory. It means user can decide the location for segment register. The segment registers are as follows. These are 16-bit registers.

1. Code segment (CS) :- It is 16-bit register containing 64KB segment address. CS is used to store executable program code.

2. Data segment :- The DS contains most data used by program. Data are accessed in the Data segment by an offset address or the content of other register that hold effective offset address.

3. Stack segment :- Stack segment is used for the stack related operations.

4. Extra segment :- ES is additional data segment that is used by some of the string to hold the destination data.

2. Instruction pointer :- It is 16-bit register. It contains the offset address of the next sequential instruction to be executed. Thus IP can't change directly. In other words instruction pointer points next instruction to be executed.

3. Instruction stream Byte Queue :-

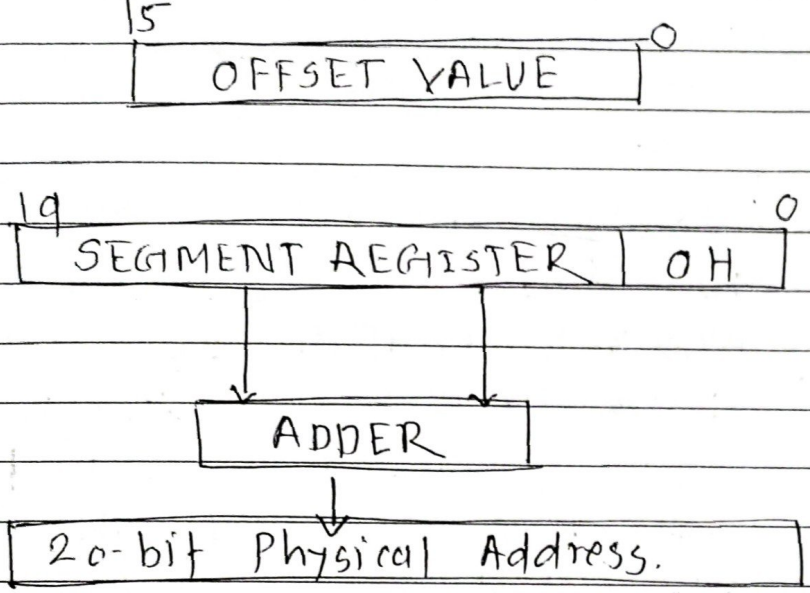
To increase the speed, BIU fetches as many as six instruction bytes ahead to time i.e. in advance from the memory and stores in to 6 byte register called queue in first ~~in first~~ out manner. and provide to execution unit. This concept is called as "pipelining". This prefetching operation may be in parallel with execution operation of execution unit. which improves speed of execution.

4. Address compute engine :- Address compute engine is responsible to genrate physical address using logical address hold by segment registers. to expose to outside world.

Formation of physical address.

The 8086 addresses a segmented memory. The complete physical address which is 20-bits long. is genrated using segment and offset address and content of an offset register also

called as offset address. To get the total physical address, put the lower nibble 0H to segment address and add offset address. Following fig. shows formation of 20-bit physical address.



Execution unit :-

The functions of execution unit are.

→ To tell BIU where to fetch the instructions or data from.

→ To decode the instructions.

→ To executes the instructions.

In other words execution unit is responsible for execution of instruction fetched by BIU. Execution unit consist of following blocks.

1. Control system.
2. Arithmatic logical unit.
3. Genral purpose registers.
4. Special purpose register
5. Flag register

1. Control system :- It is responsible for various internal operations. A decoder in EU decodes the instruction fetched memory to genrate different internal or external control signals required to perform the operation.

2. Arithmatic and Logical unit :- It is responsible to carry out all arithmatic and logical operations such as addition, substract, multiplican, division and OR, AND, NOT respectively.

3. Genral purpose registers :-

These genral purpose registers are used to store 8-bit or as ~~an~~ 16-bit registers by adding two 8-bit registers with each other. such as Ax, Bx, Cx, Dx, i.e each of regi. is the combination of two 8-bit registers such as AH+AL, BH+BL CH+CL, DH+DL respectively.

(1) Ax :- This is 16-bit regi. combination of AH & AL 8-bit regi. This is also known as Accumalater that stores operand for arithmatic ~~regt~~ operation and can hold result by default in some instructions.

(2) Bx :- (BH+BL) - 16-bit regi. This is used mainly as base register. It holds the starting base

location of memory region within data segment.

(3) CX :- (CH + CL) - 16-bit. It is defined as counter. It is primarily used in loop instruction to store loop counter.

(4) DX :- (DH + DL) - 16-bit. It is used to contain I/O port address for I/O instruction.

4. Special purpose registers :-

(1) Stack pointer :- It is 16-bit register pointing to program stack in stack segment.

(2) Base pointer :- It is 16-bit register to data in stack segment. BP register is usually used for based indexed or register indirect addressing.

(3) Source index :- It is 16-bit register and used in string operations. to address source data.

(4) Destination index :- It is 16-bit register. used to address destination data in string manipulations.

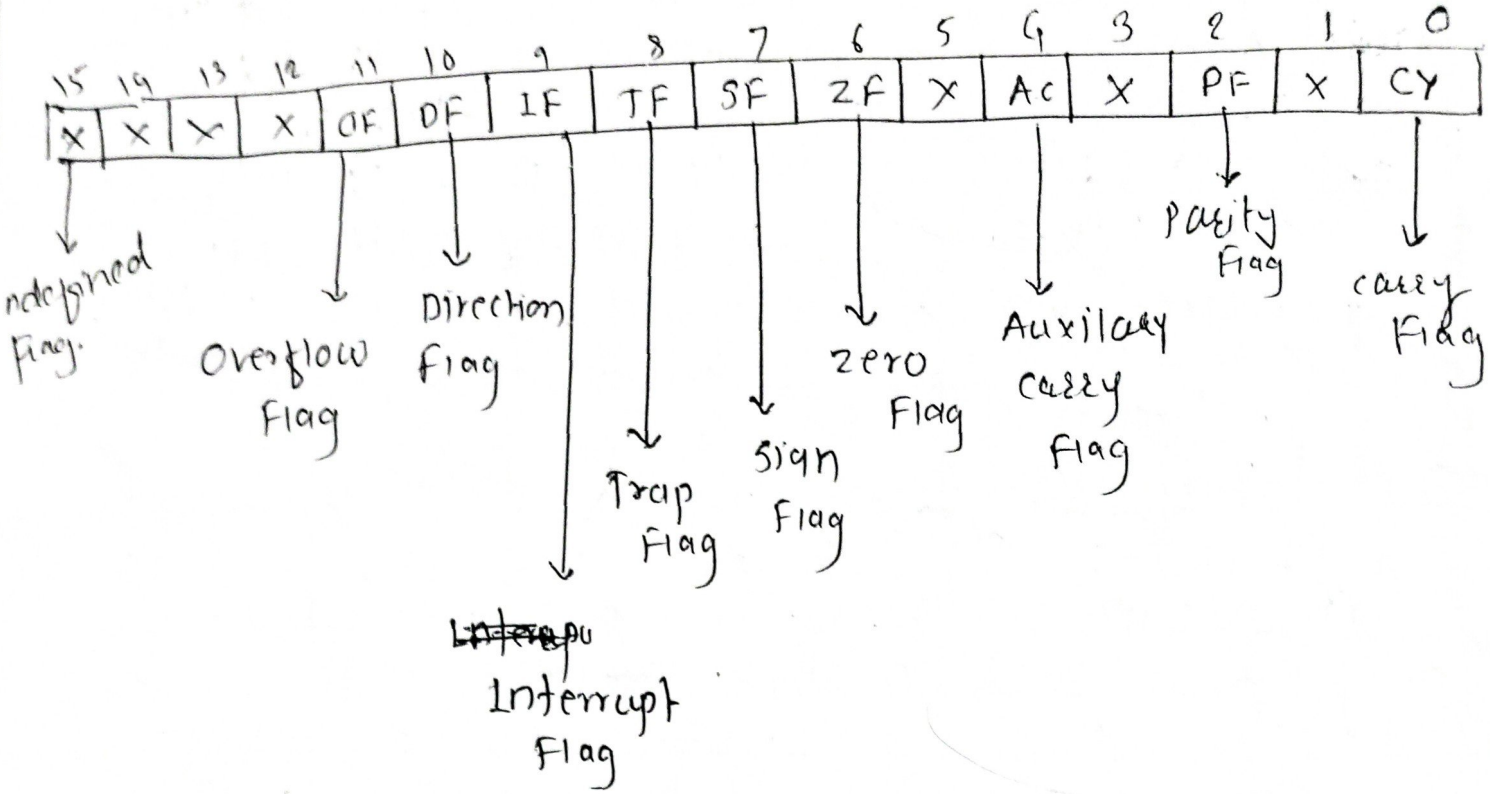


Fig:- Flag register

5. Flag register :- Flag register determines the current state of the processor. They are modified automatically by cpu after mathematical operations, this allows to determine the type of the result, and to determine conditions to transfer control other parts of the program. There are 9 Flags (Nine Flags) active and other as undefined flags. These Flags are devided in to two main category

1. Conditional Flags.
2. control Flags.

(1) Conditional flags:-

* Carry flag :- If carry is genrated in mathematical operation then the flag is seted to 1 , other wise it is at logic '0'.

* Auxilary carry flag :- If carry is genrated from D3 to D4 bit the auxilary carry flag seted to 1 and if it is not then it is at logic 0'.

* Parity flag :- if lower order 8-bit of result is even then parity flag is seted to one 1 or if no. is odd then it is at logic '0'.

* Zero flag :- If result is zero then it is seted to one 1. if not it is at logic '0'.

* Sign flag :- If results MSB bit is 1 then the no. is -ve then sign flag is seted otherwise it is at logic 0 means no. is +ve.

2 * Control Flags :-

* control flags are seted and reseted to control the operation of execution unit.

* Trap Flag :- It allows to execute one instruction at a time for debugging. When it set program run in single step.

* Interrupt Flag :- If it is set maskable interrupt is enabled If reset interrupts are disabled.

* Direction Flag :- It is used in string operation. If it is set string byte are accessed from higher memory address to lower If reseted lower to higher.

Direct program memory Addressing :-

Direct program memory Addressing is used in many early microprocessor for Jump & CALL instructions. It is also used in high level language such as 'c', c++, Basic, etc. (GOTO statement). But relative and indirect program memory addressing modes are not often used.

The instruction with this addressing mode stores the address with op-code.

e.g. If a program jumps to memory location 10000H then the address 10000H is stores with op-code. jumps are of two types intersegment jump & intrasegment jump. intersegment means within segment & intrasegment means outside the segment. Consider intersegment jump instruction this instruction will required 4 bytes to store. direct jump ~~some times called~~ often called far jump because on any memory location it can be jump.

eg:-
Consider Jmp 10000H
jmp opcode require one-byte and 10000H required 4-byte of memory to store. The jump instruction loads cs with 1000H and instruction pointer to 0000H to jump to memory location 10000H for the next instruction.

Relative program Addressing mode:-

It is not avilable in every 8p. relative means relative to instruction pointer (IP), If jmp instruction skips two bytes of memory, the address in instruction pointer is 2 that add to the IP. It means that in relation to instruction pointer 2 is added in address of IP.

Indirect program memory addressing mode:-

The 8p allows several forms of program indirect memory addressing mode for jmp & call instruction.
e.g. jmp AX jump to current code seg. locan addressed by AX.
This can used any 16-bit regi. or any relative regi. BP, [BX] [SI], [DI] & and any relative with displacement.

In 80386 Extended also used.

e.g. Jump [BX].

—o—

1. Direct :- Jmp 10000H
2. Relative :- JMP[2].
3. Jmp Indirect : JMP [BX].

↑

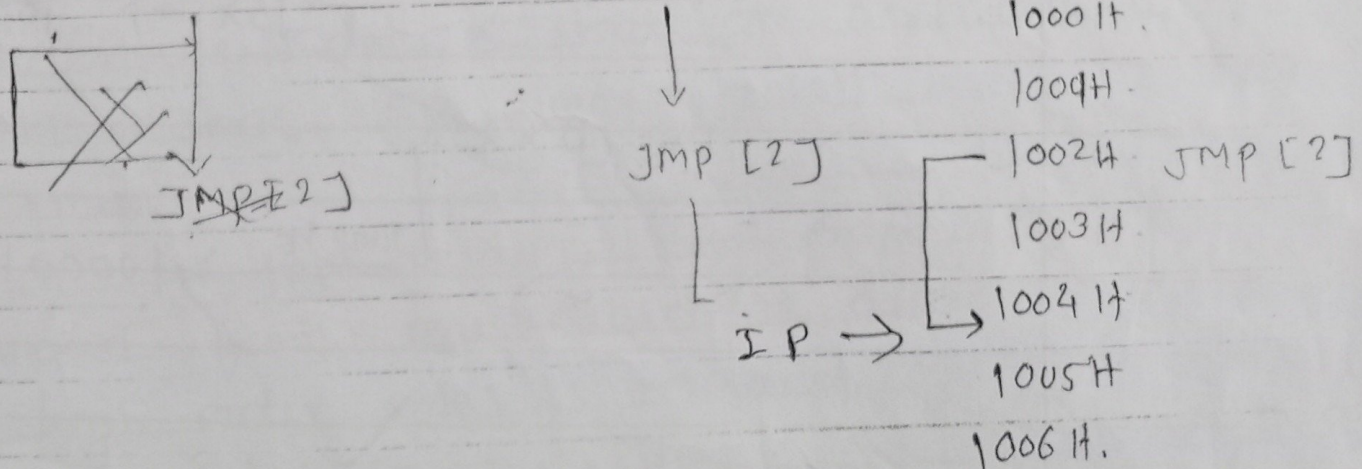
—o—

refer this :- ~~rather than previous one.~~ Instead above

Relative Program Addressing mode :-

This is another kind of program memory addressing mode. It is not avilable in earlier ep. Relative mean relative to instruction pointer, If consider e.g.

If jump instruction will skips two bytes of memory 2 is added in address of instruction pointer for the next instruction. This shown in following ~~diagram~~ e.g.



This type of instruction uses relative program memory addressing mode.

4 Q. No. Q. No.

Q. No.

Indirect Program memory Addressing modes:-

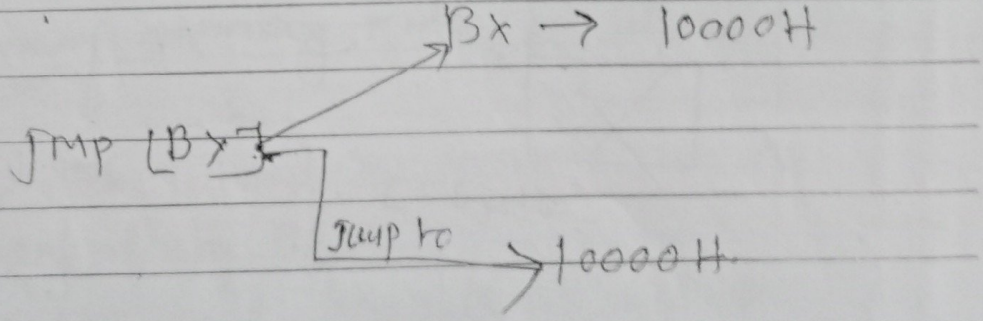
The 8µp allows several forms of Jump and call instructions with indirect addressing modes.

e.g. JMP AX.

This instruction will transfer the program control on memory location which is specified by AX, regi.

In this type of program addressing mode we can use any 16-bit regi. such as AX, BX, CX, DX and relative regi such as SI, DI, BP, BX, etc. In 80386 it uses extended registers. Here, we are creating first reference to memory location by the use of regi. and then it is used with jump instruction. So, This is called indirect, program memory addressing modes.

e.g.



Stack memory Addressing Mode :-

Stack plays an important role in 8ep. It holds data temporarily and stores the return address used by procedures. The stack memory is based on LIFO which describe the way by which th. on elements insertion and deletion from stack. A way by which we can insert an element in stack is known as push and a way by which we can remove an element from stack is known as 'pop'. In 8up we have push and pop both instruction.

The CALL instruction also uses stack to hold return address and RET instruction uses stack to remove return address from the stack. The stack memory is maintained by two register 1. stack pointer and 2. Stack segment register

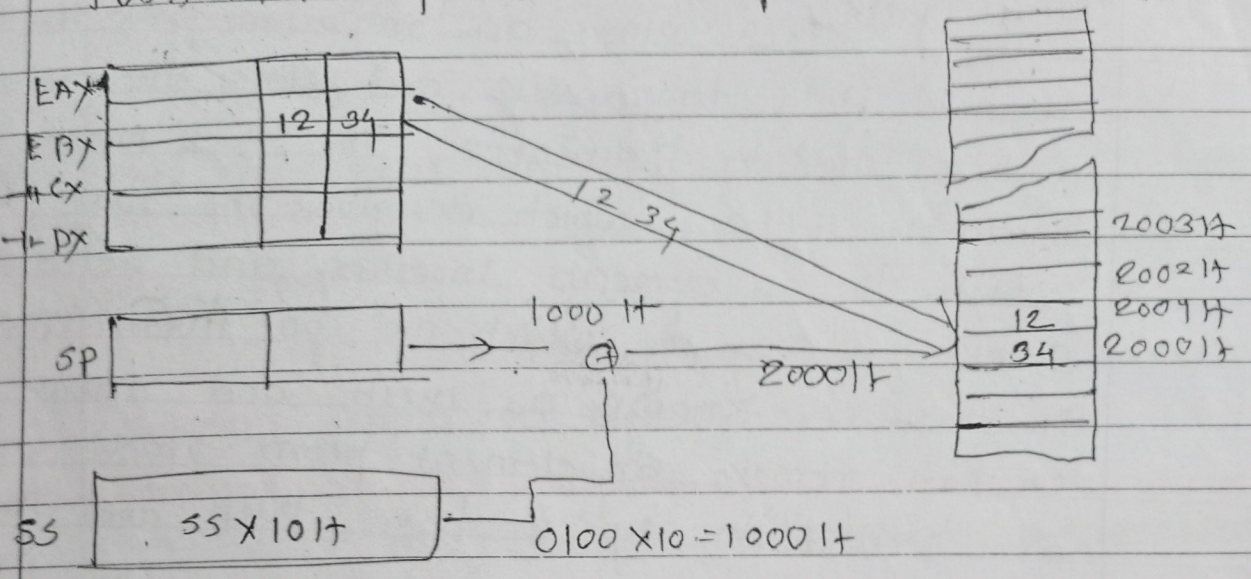
Whenever element push on stack (word) then high order 8-bit placed in location addressed by sp-1 & loco order 8-bit are place in location sp-2. Then sp is decramented by 2 sothat next data is stored on next avilable m.l on stack. sp always points location in stack segment.

In pop operation loco order 8-bit data is removed from m.l which is addressed by sp. & high order 8-bit is removed from location addressed by sp+1. The sp is incrament by 2

6 Q. No. | Q. No.

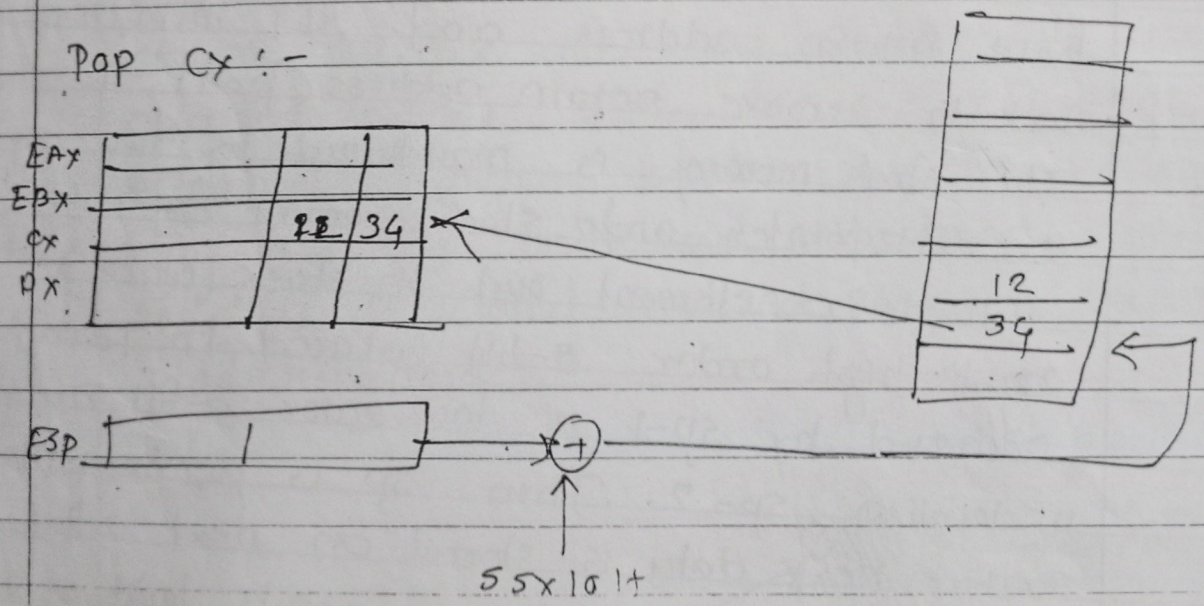
प्र. क्र. Q. No.

PUSH BX :- place content of BX on to stack.



— o —

Pop CX :-



— o —