EAX - Accumulator (is the most general register)

it is used by most of the instructions

$a[7] = *(a+7)$

EBX - The Base register

ECX - Counter register

EDX - Data register + EAX / EDX:EAX

it's used usually together with EAX in computation when results are exiding DD

ESP - Stack Pointer

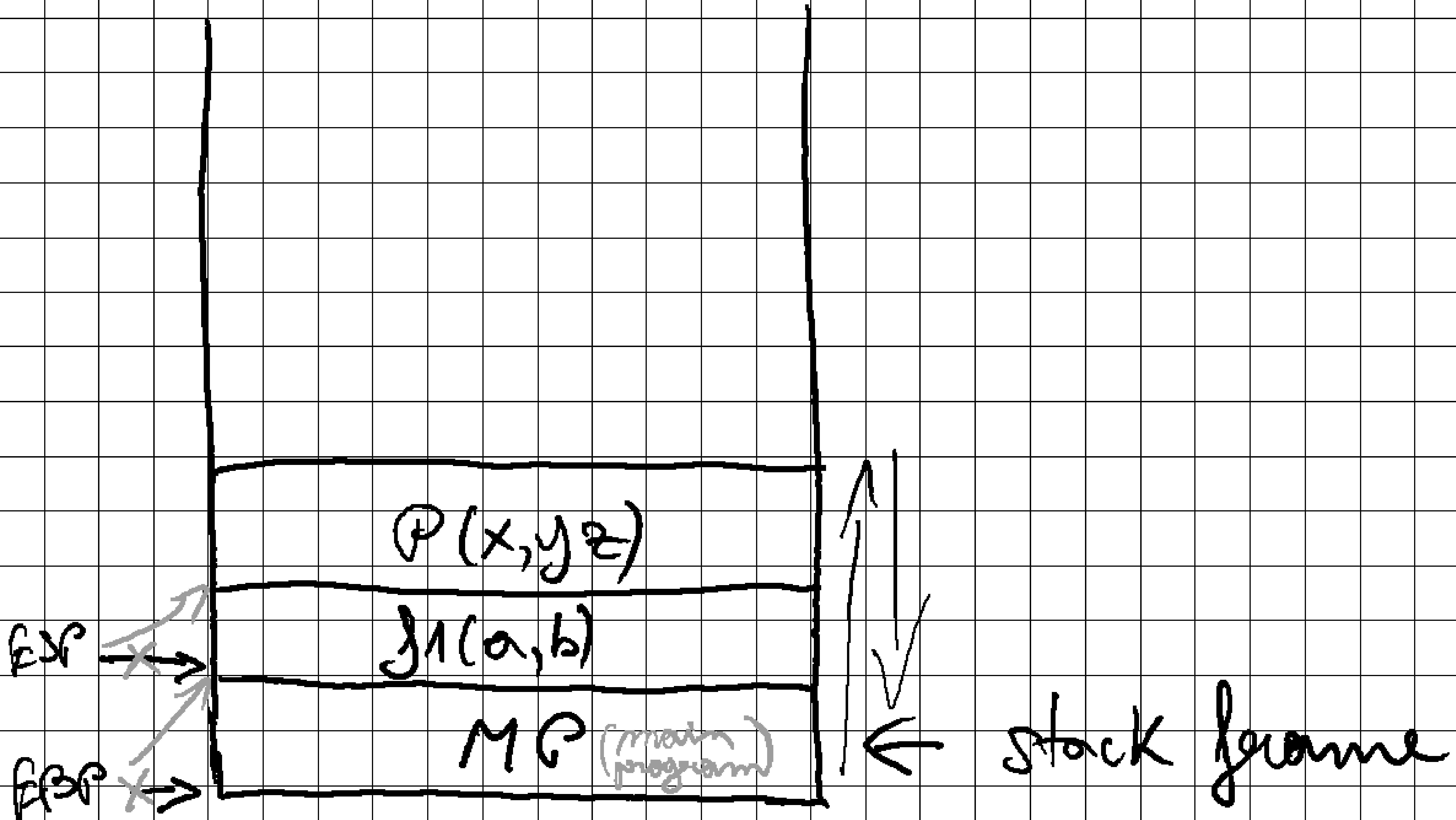
this register contains a pointer to the element located to the top of the stack

EBP - Base Pointer

it contains a pointer to the element located at the base of the stack

SS - Stack Segment

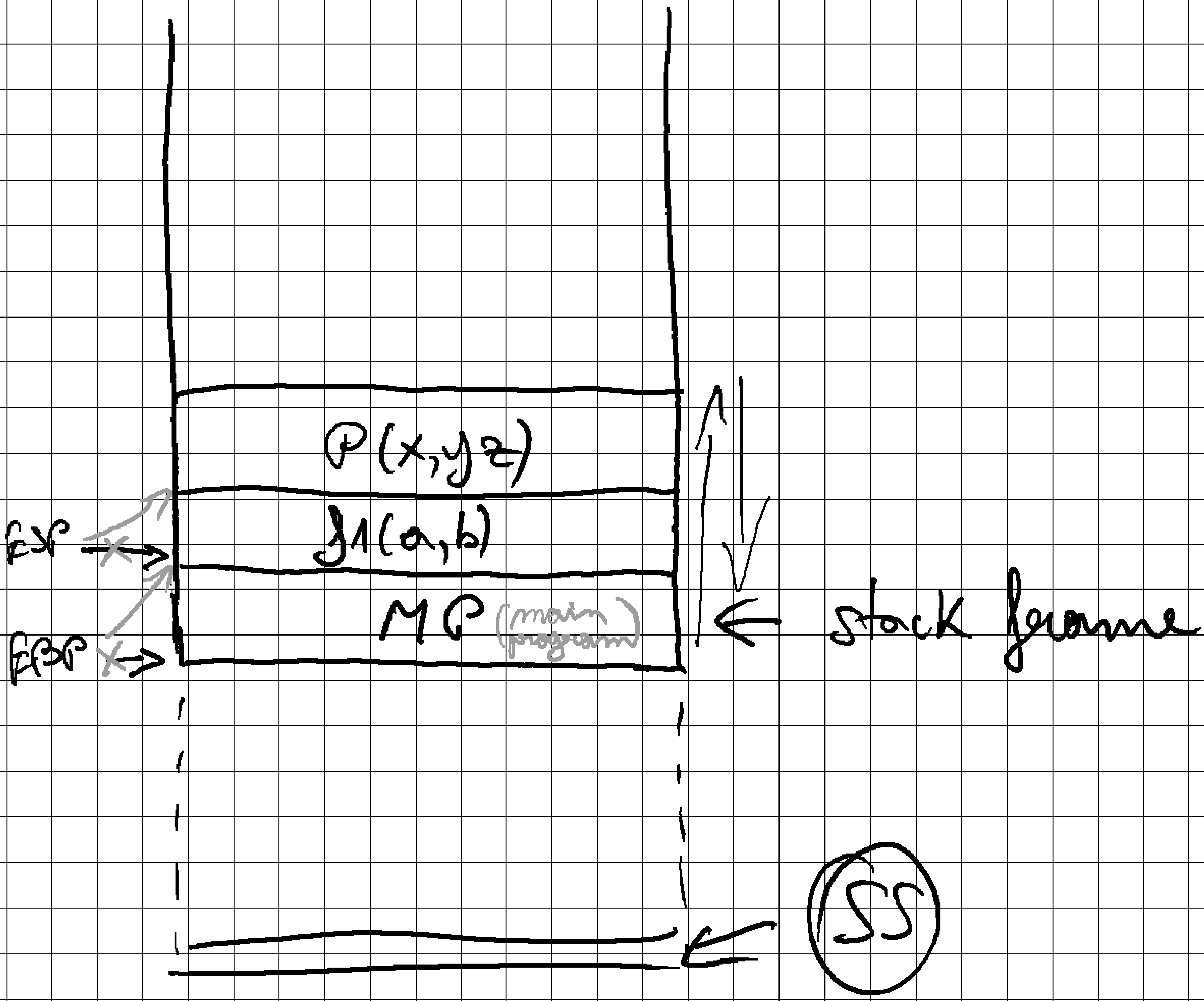
Why 3 registers of the microprocessor are dealing with the stack? (possible exam question)



Always the MP starts and ends the execution

Why do we need 3 registers to hold the stack? (exam question)

the role of ESP, EBP is to always define the current executing stack frame

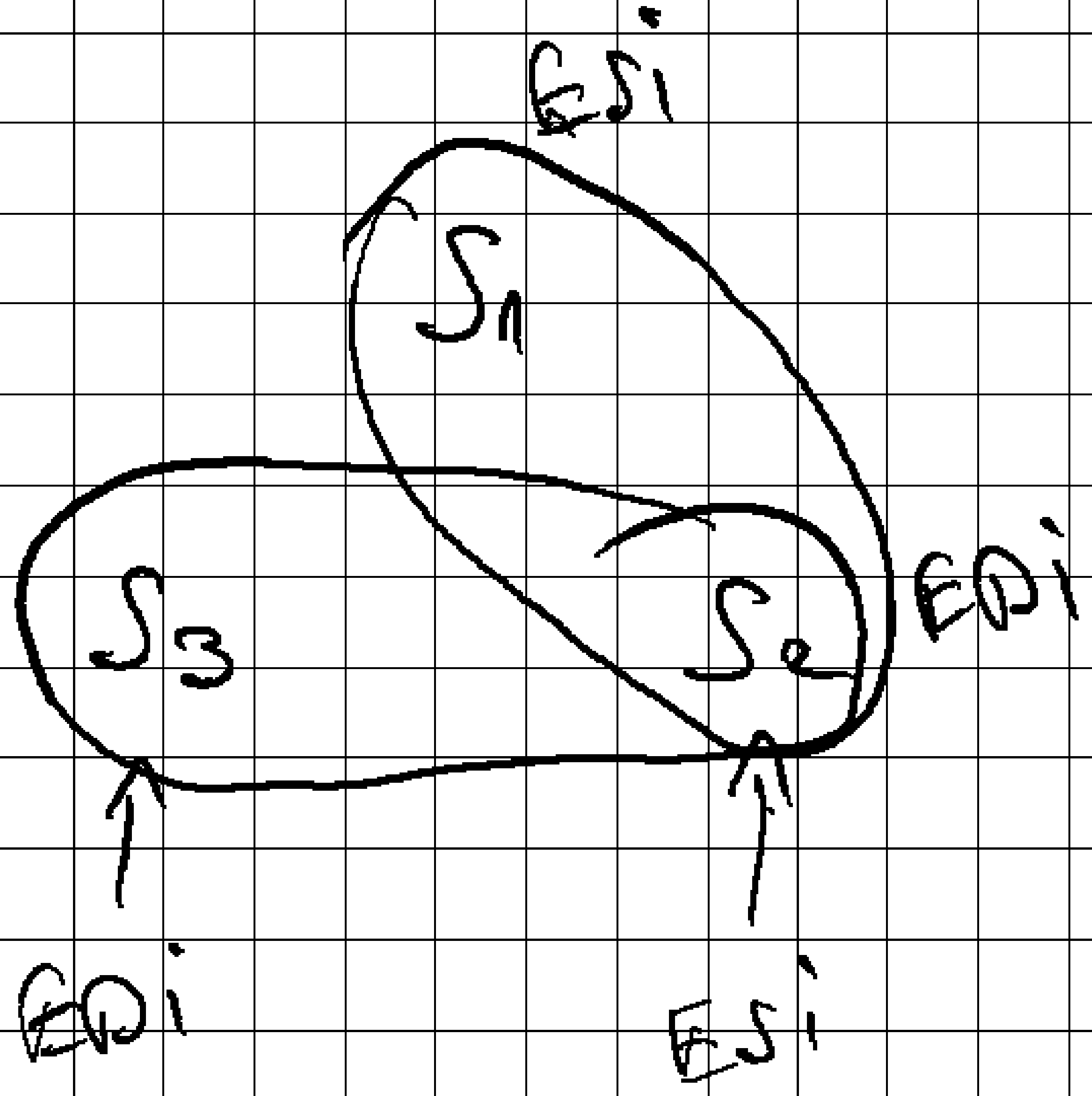

P(x,y,z)
f1(a,b)
MP (main program)
ESP
EBP
stack frame
SS

There are 4 types of segments:

- $\overbrace{\text{code segments}}^{CS}$, $\overbrace{\text{data segments}}^{DS}$, $\overbrace{\text{stack segments}}^{SS}$ and $\overbrace{\text{extra segments}}^{ES}$

EDi – Destination index

ESi – Source index



push ESi

push EDi

Flags:

A flag is an indicator represented on 1 bit

CF - carry flag (transport flag)

LPO - last performed operation

it will be set to 1 if we have a carry outside the representation and 0 otherwise

Why they designed it so for addition and substraction the result must be on the same size (byte + byte = byte)?

Why do you think in the assembly language it doesn't allow you to have instructions with more than 2 opperands (3, 4 etc)?

```
mov  ah,93h
mov  al,73h
add  ah,al     ; AH=06h, CF=1
mov  al,ah
mov  ah,0
adc  ah,0
```

↓
add
with
carry