Assembler Intel 8086

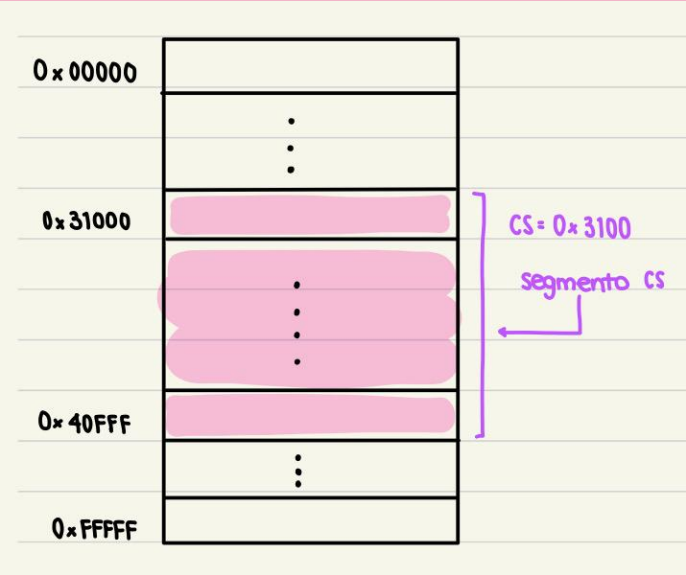
Arquitectura de Computadoras - Práctico 7

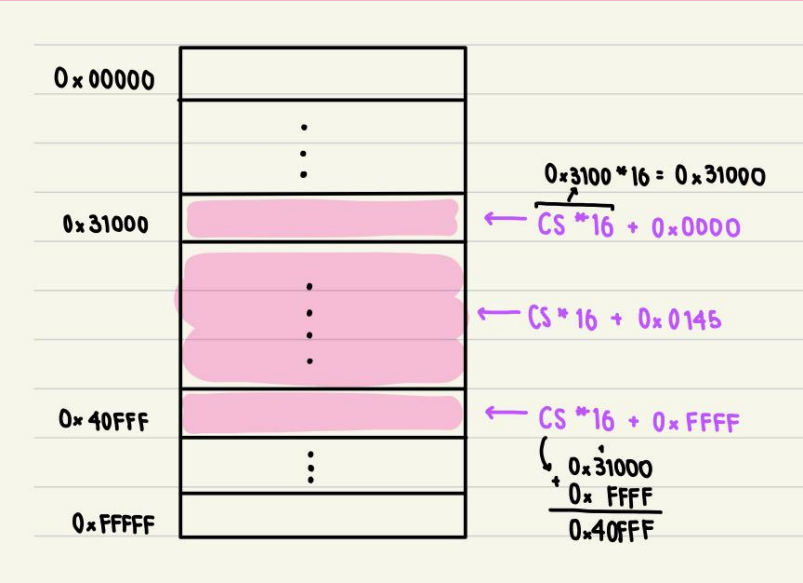
Características principales

- Procesador de 16 bits: que implica?
- Memoria segmentada
- 12 registros visibles, con personalidad
- Little Endian
- Diseño CISC
- Stack implementado por hardware sobre memoria
- Memoria direccionada de a byte

Memoria segmentada

direccion_fisica[20] = registro_segmento*16 + desplazamiento





ejercicio: Presente tres formas de lograr la dirección 0xCAFE en el modo segmentado

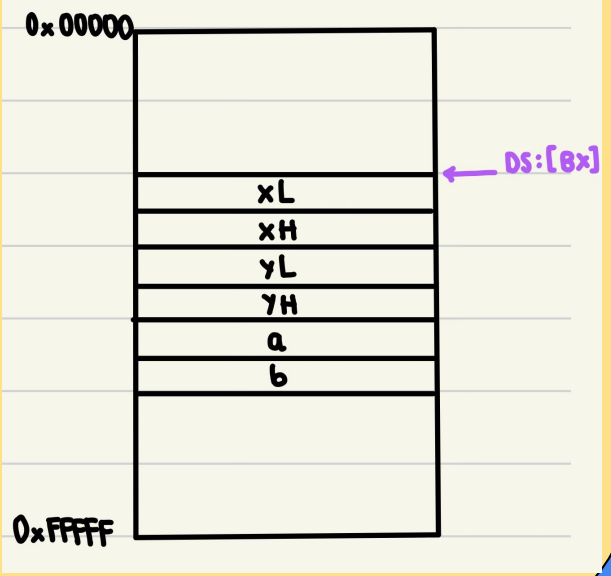
Variables

variable: es un espacio en memoria con un nombre asignado

tipo: se refleja con el tamaño que ocupan en memoria

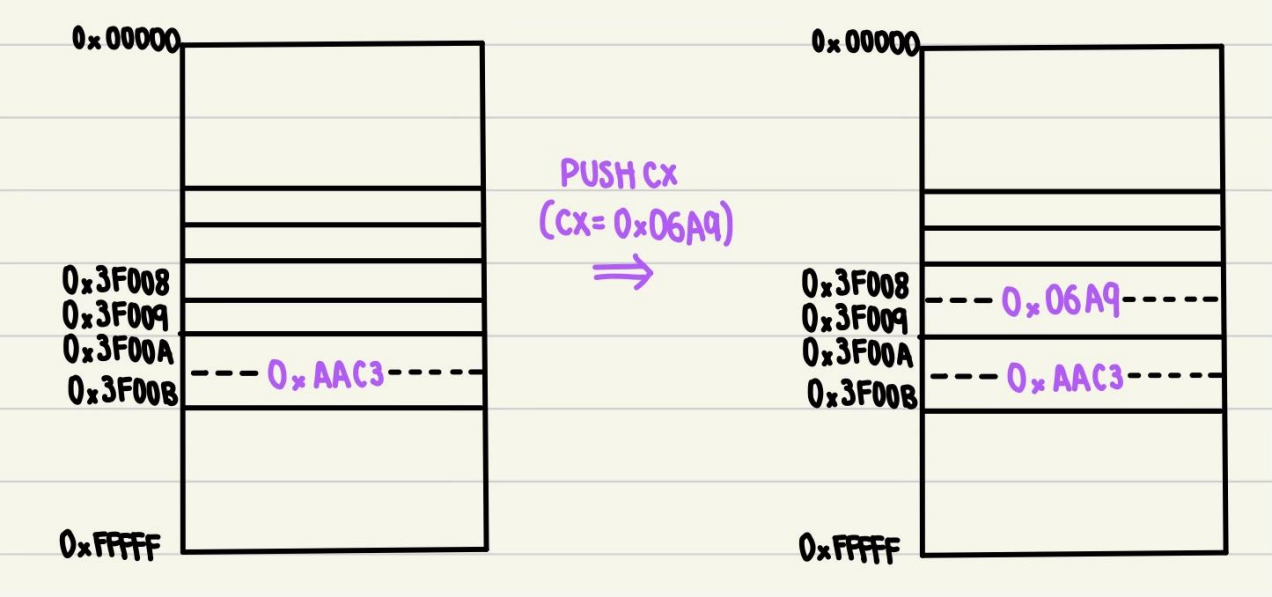
```
struct ejemplo {
    short x,y;
    char a,b;
}; // 6 bytes
```

```
mov Ax, [Bx]      // Ax = x
mov Ax, [Bx+2]    // Ax = y
mov Ax, [Bx+4]    // Ax = a
mov Ax, [Bx+5]    // Ax = b
```



Stack

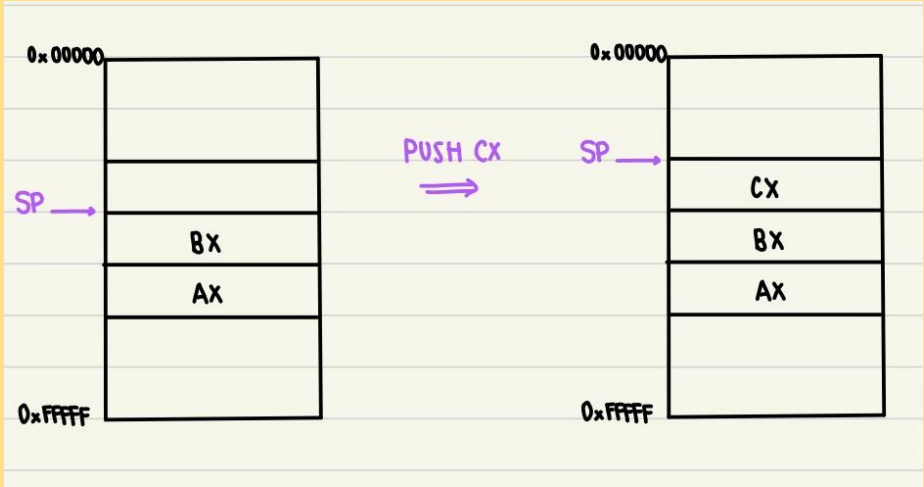
- posiciones ocupan 2 bytes **SIEMPRE**
- crece hacia las posiciones más bajas de memoria



Stack (2)

el stack se implementa mediante dos punteros:

- stack pointer (SP): última posición del stack
- base pointer (BP): para direccionar en el stack



Punteros

¿Cómo se representan los punteros?

- en el mismo segmento: son 2 bytes para el desplazamiento
- en otro segmento: son 4 bytes (2 para el segmento y 2 para el desplazamiento)
- como direcciones absolutas de memoria

Ejercicio

Considere la siguiente estructura de datos para manejar inscripciones a cursos en un sistema x86:

```
struct _asignatura{
        short codA;
        char* nombre;
}
struct curso{
        short año;
        _asignatura* asig;
        short cantEstudiantes;
}
```

Y las siguientes variables globales:

```
short cantCursos;
curso[1024] cursos;
```

Donde la variable 'cantCursos' indica la cantidad de posiciones válidas del arreglo.

a. Implementar en un lenguaje de alto nivel, la función:

   short cantidadTotalEstudiantes(short año);

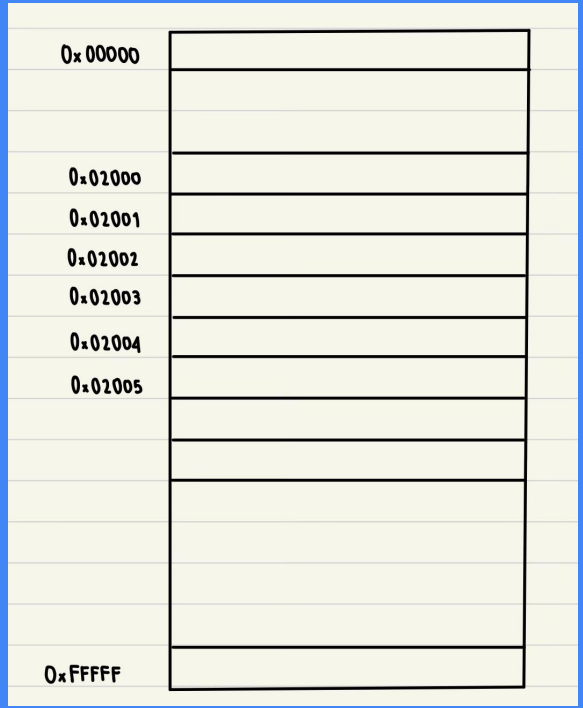
   Que dado un año, indica la cantidad total de estudiantes en cursos de dicho año.

Ejercicio - Parte A

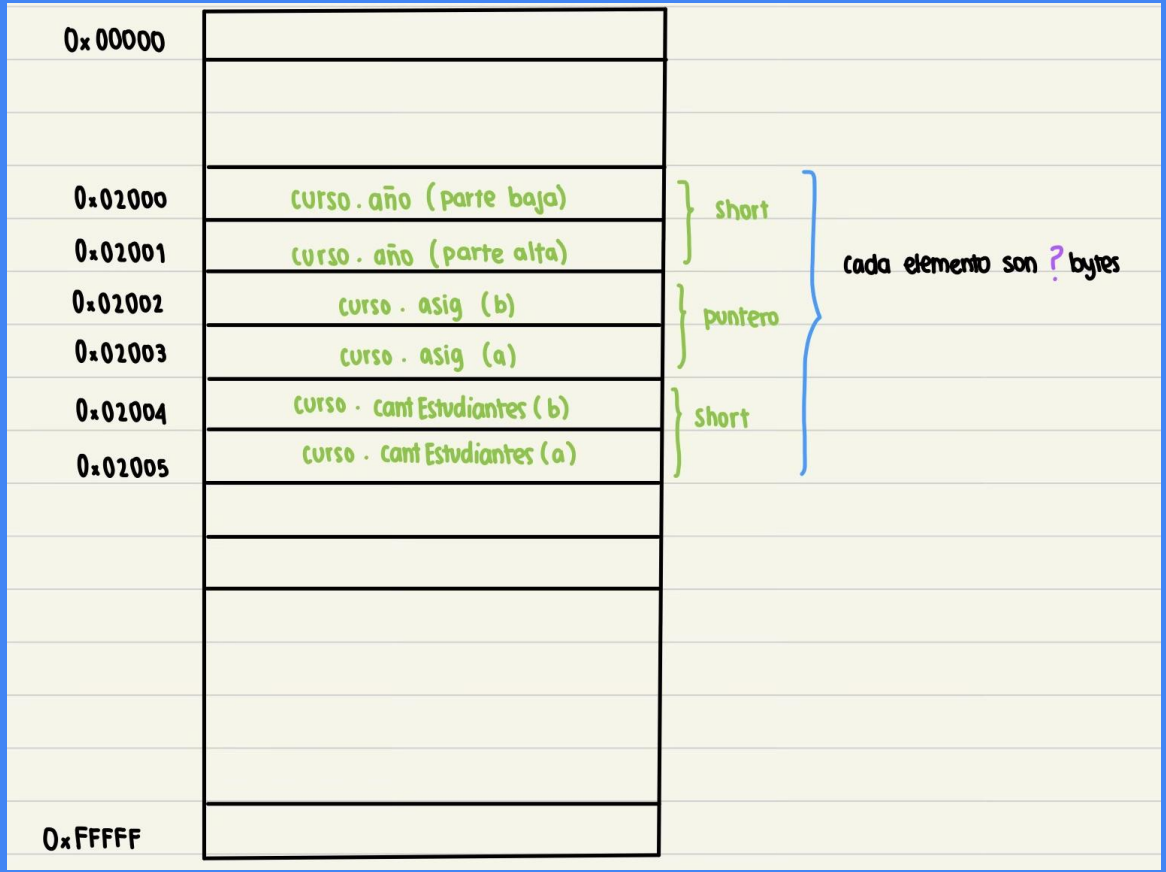
```
short cantidadTotalEstudiantes (short año){
    short cant =0;
    for ( int i=0 ; i< cantCursos ; i++) {
        if (cursos [i]. año == año)
            cant += cursos [i]. cantEstudiantes;
    }
    return cant;
};
```

Ejercicio - Parte B

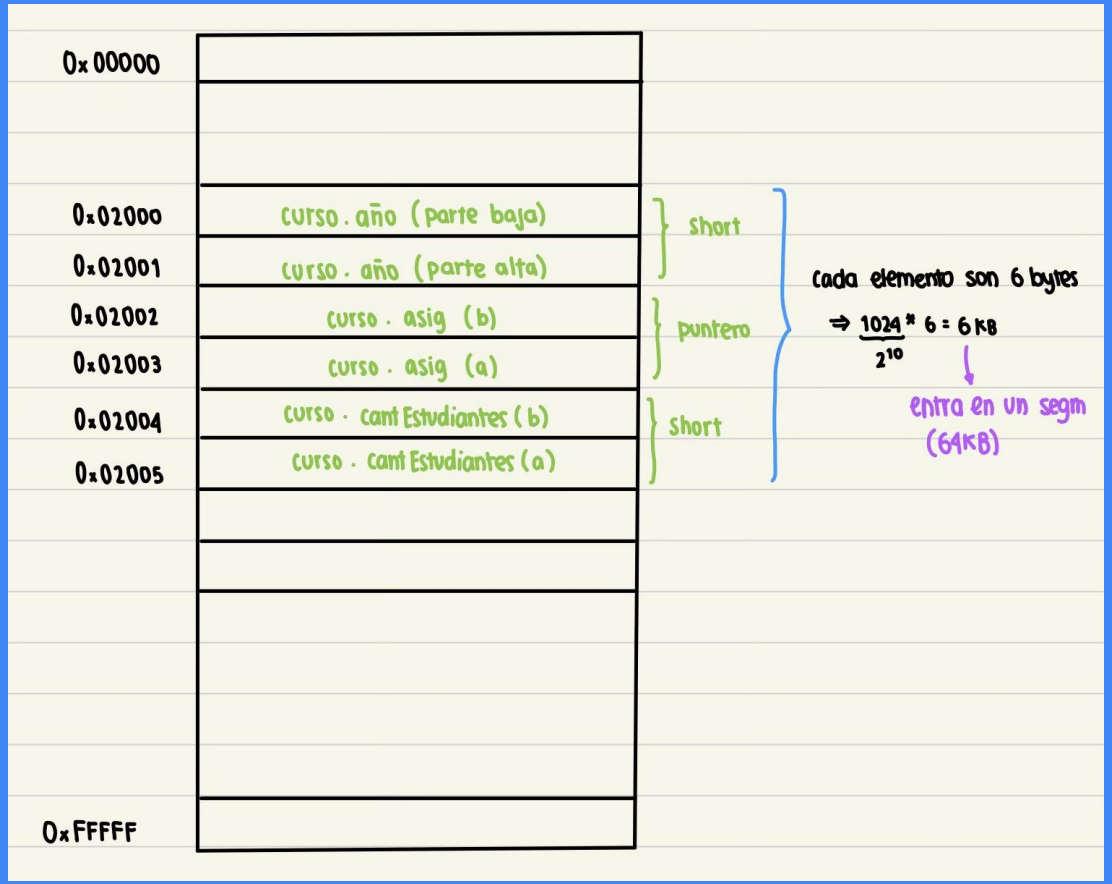
b. Representar gráficamente un elemento del struct 'curso' en memoria y calcular la cantidad de bytes necesaria para almacenar el arreglo de cursos. Suponga que los punteros se guardan como desplazamientos con respecto al segmento ES.



Ejercicio - Parte B



Ejercicio - Parte B



Ejercicio - Parte C

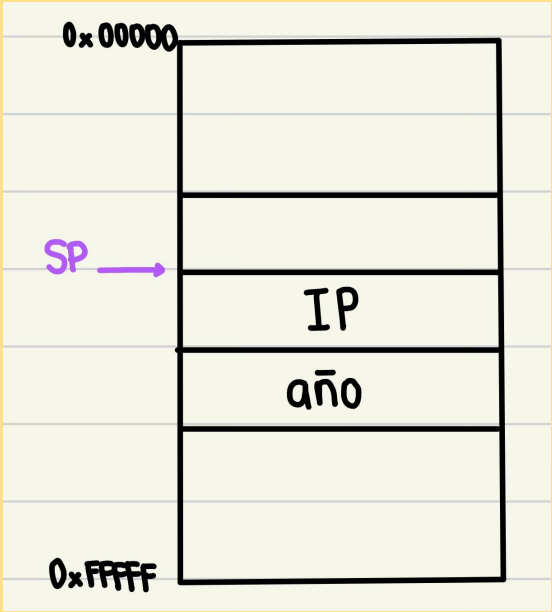
c. Compilar en assembler 8086 sabiendo que las variables globales se encuentran almacenadas a partir de la dirección 0 del ES. La variable año se recibe en el stack y el resultado se devuelve en el stack. Se debe preservar el valor de todos los registros.

cómo se reciben los parámetros?

Ejercicio - Parte C

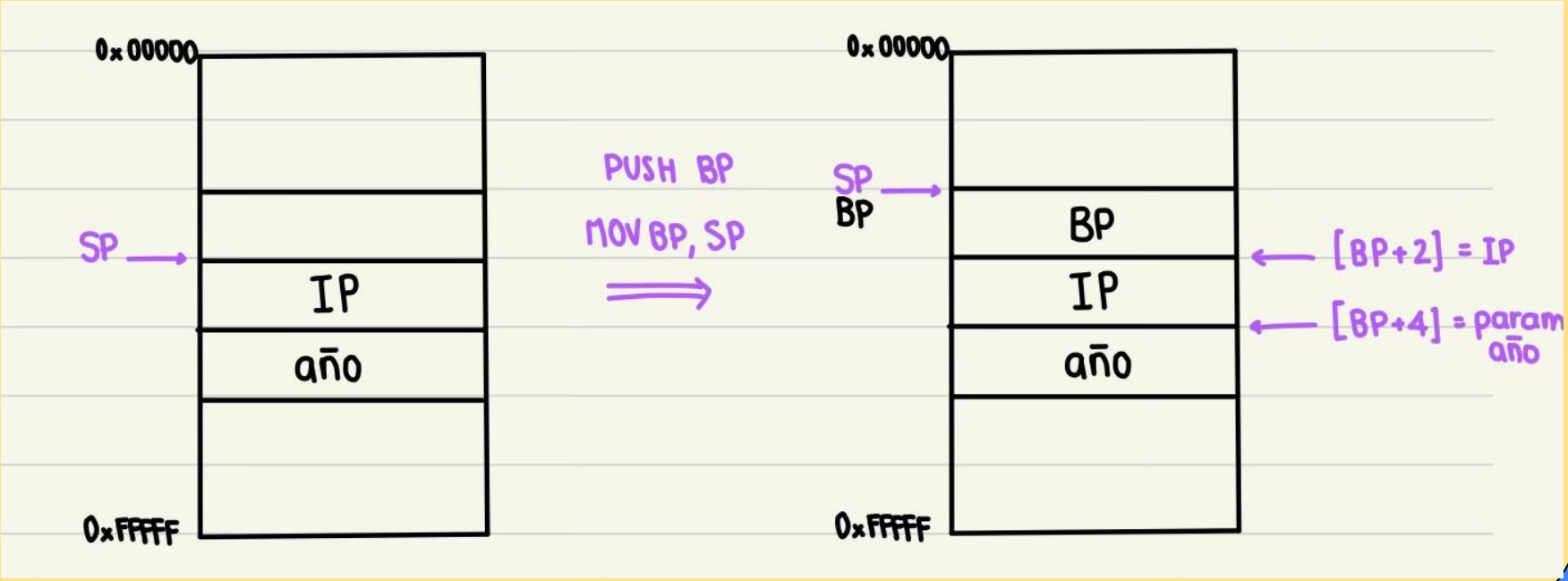
c. Compilar en assembler 8086 sabiendo que las variables globales se encuentran almacenadas a partir de la dirección 0 del ES. La variable año se recibe en el stack y el resultado se devuelve en el stack. Se debe preservar el valor de todos los registros.

cómo se reciben los parámetros:



Ejercicio - Parte C

referencia al stack:



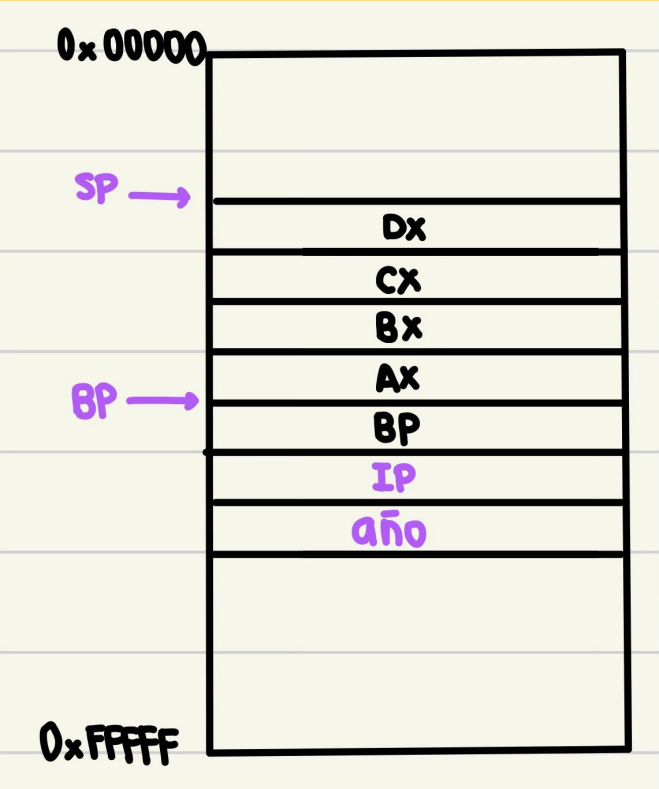
Ejercicio - Parte C

```
cantidadTotalEstudiantes PROC

    PUSH BP;

    MOV BP, SP;
```

Ejercicio - Parte C

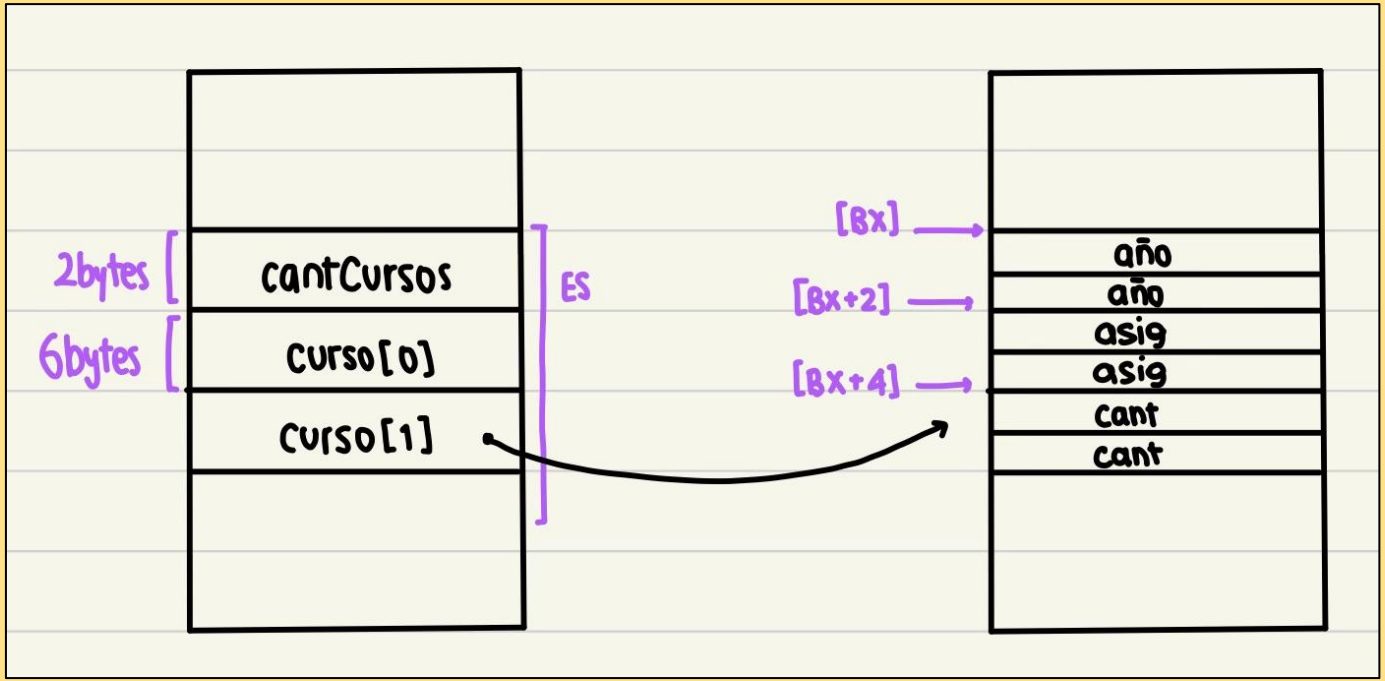
preservar los registros:

```
cantidadTotalEstudiantes PROC
    PUSH BP;
    MOV BP, SP;
    PUSH AX;
    PUSH Bx;
    PUSH CX;
    PUSH DX;
```

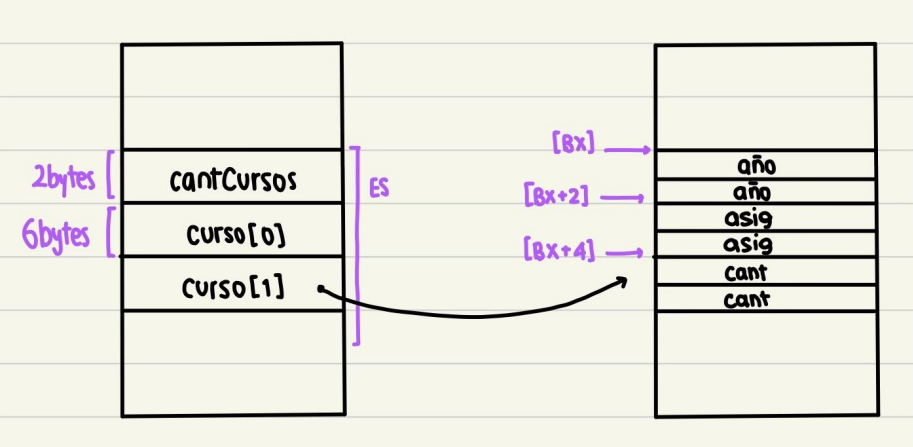


Ejercicio - Parte C

variables globales en memoria:

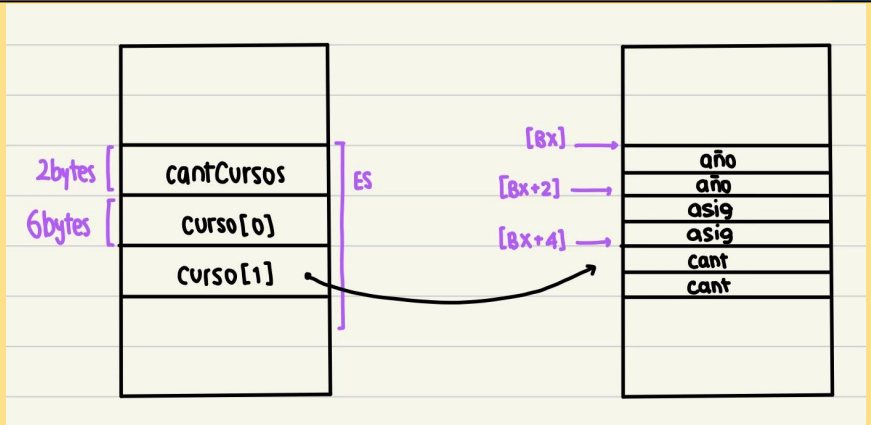


Ejercicio - Parte C



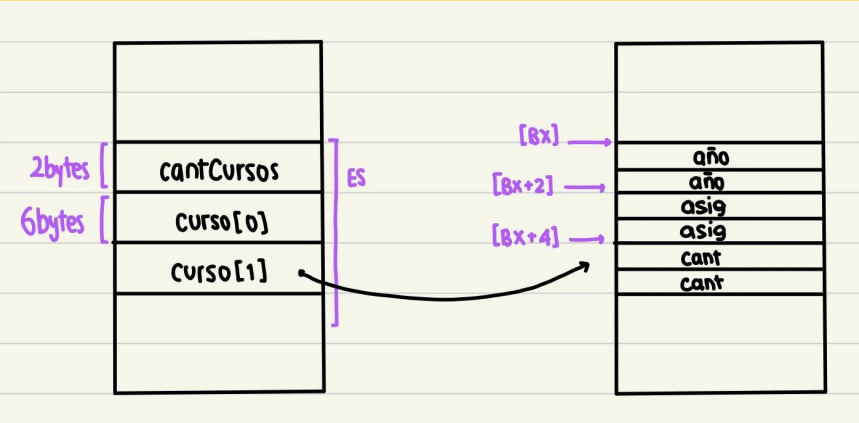

```
XOR AX, AX ;  AX = cant
XOR DX, DX ;  DX = i

MOV BX, 2

MOV CX, [BP+4]  ; CX = año

FOR:
```

Ejercicio - Parte C




```
FOR:
    CMP DX , ES:[0] ; cantCursos (short)
    JNB FINFOR; salta si  DX ≥ ES:[0] (i ≥ cantCursos)
    CMP CX, ES:[BX] ;  cursos[i].año == año
    JNE FINIF
    ADD AX, ES:[BX+4] ; cant += cursos[i].cantEstudiantes
```

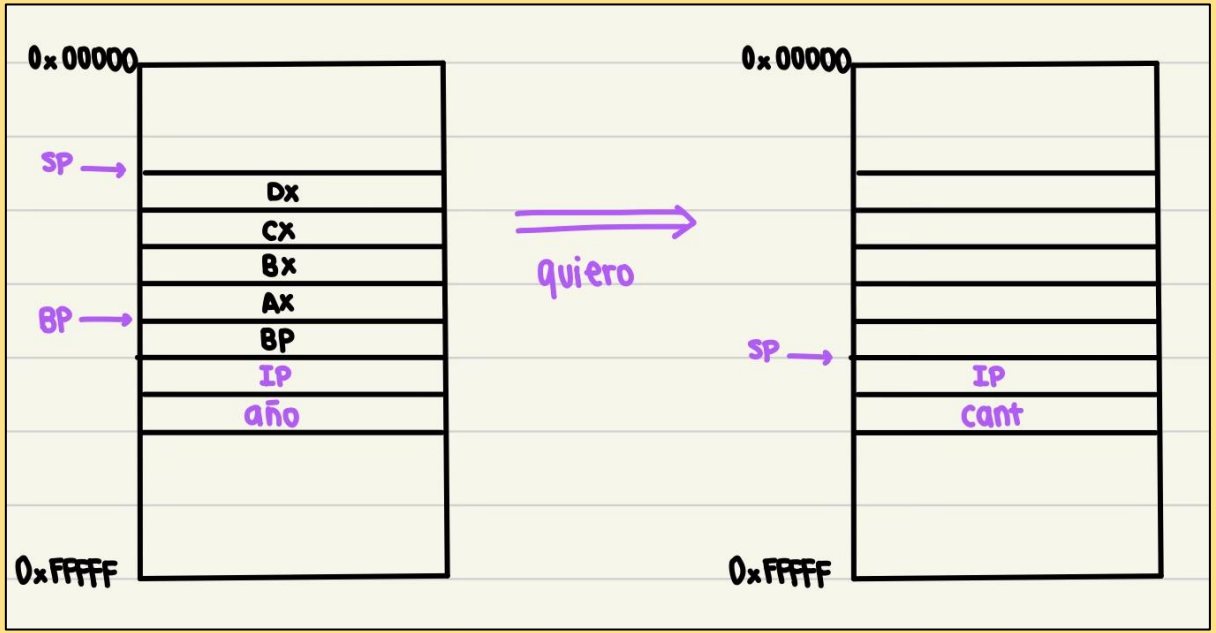
Ejercicio - Parte C




```
FINIF :

        ADD Bx, 6 ; apunto al siguiente curso

        INC DX ; i ++

        JMP FOR
```

Ejercicio - Parte C

retorno de la función:



Ejercicio - Parte C

```
FINFOR:
        MOV [BP+4], AX ; guardo el res donde vino el parámetro
        POP DX
        POP CX
        POP BX
        POP AX
        POP BP ; dejo IP en el tope
        RET
```