Problema 9. (1) Structs

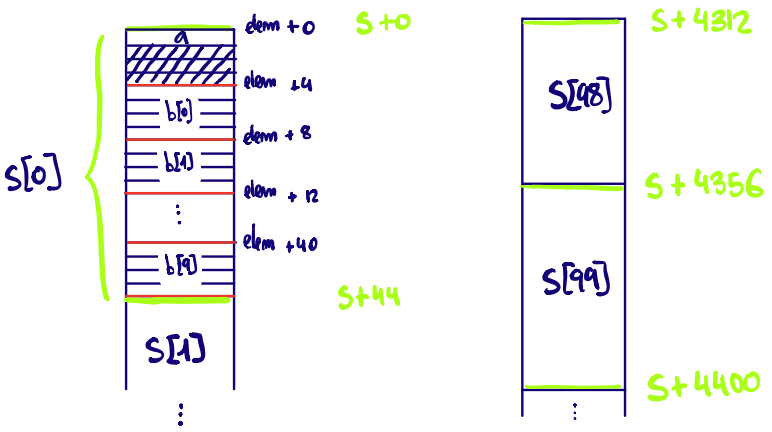
Dada la siguiente declaración de variables:

```
typedef struct{
        char a;   → 1
        int b[10]; → 4
} elem;
elem s[100];
```

} alineamiento 4

Sabemos que la dirección de s se encuentra en %ebx y las variables i, j y x se encuentran respectivamente en los registros %esi,%edi y %dl

a) Dibujad el struct elem.



b) Determinad cómo se calcula la dirección de memoria donde se almacena el dato

```
s[i].b[j]
```

$$@s[i].b[j] = @S + 44i + 4 + 4j = \boxed{@S + 4(11i + j + 1)}$$

S[i] — Cada uno mide 44 bytes

+4 por el char alineado

b[j] — Cada uno mide 4 bytes

c) Escribid la secuencia de instrucciones necesaria para codificar la siguiente sentencia

```
x = s[s[i].b[j]].a
```

(s[i].b[j] = α)

```
IMUL  $11, %esi, %eax   # eax = 11i
ADDL  %edi, %eax        # eax = 11i + j
INCL  %eax              # eax = 11i + j + 1
SHLL  2, %eax           # eax = (11i + j + 1)·2^2
ADDL  %ebx, %eax        # eax = (11i + j + 1)·2^2 + @S
MOVB  (%ebx, %eax, 44), %dl   # x = @(@S + 44·eax)  (a es el primer byte de S[α])   x = s[α]
```

(α)

%ebx = @S
%esi = i
%edi = j
%dl = x

Problema 10. (1) Subrutinas

Dada la siguiente función escrita en C:

```
int calcula(int M[10][10], int m, int n)
{  int i, suma, fila;
   suma = 0;
   fila = 0;
   for (i = m; i<n; i++)
      suma = suma + Normaliza(M[fila][i], &fila);
   return (suma + 1);
}
```

$M[i][j] = @M + NC \cdot 4 \cdot i + 4 \cdot j$

$@M + 4(i \cdot NC + j)$

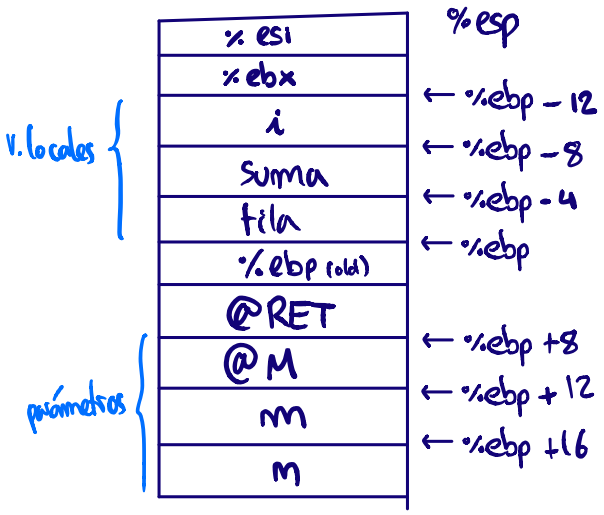
$@M + 4(10i + j)$

$@M + 4(10 \cdot fila + i)$

a) Dibujad el bloque de activación de la función.

b) Traducid la función a ensamblador del IA32. Suponed que la función **Normaliza** ya está programada.

```
pushl %ebp                  # guardar ebp
movl  %esp, %ebp            # establecer nuevo ebp
subl  $12, %esp             # guardar espacio para i, suma, fila
pushl %ebx
pushl %esi

movl  $0, -8(%ebp)          # suma = 0
movl  $0, -4(%ebp)          # fila = 0
movl  12(%ebp), %esi        # i = m

for:
cmpl  %esi, 16(%ebp)
jmple endfor                # acaba si n <= i

leal  -4(%ebp), %eax              # @fila
pushl %eax                  # paso parámetro &fila

imull $10, -4(%ebp), %eax   # 10*fila
addl  %esi, %eax            # 10*fila + i
movl  8(%ebp), %ebx         # @M
movl  (%ebx,%eax,4), %eax   # @M + 4*(10*fila + i)
pushl %eax                  # paso parámetro M[fila][i]

call Normaliza              # llamada a Normaliza (no hace falta guardar nada)
addl $8, %esp               # borramos de la pila el espacio ocupado por los parametros

addl  %eax, -8(%ebp)        # suma = suma + retorno
incl  %esi                  # ++i
jmp   for

endfor:
movl  -8(%ebp), %eax
incl  %eax
popl  %esi
popl  %ebx
movl  %ebp, %esp
popl  %ebp
ret
```



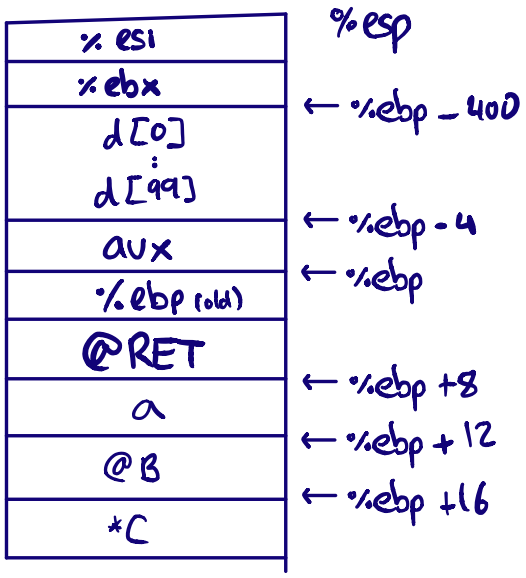
Problema 14.

Dada la siguiente función escrita en C:

```
void examen (int a, int b[100], int *c)
{  int d[100];
   int aux;
   ...
}
```

a) Dibujad el bloque de activación de la función.

Traducid a ensamblador del IA32 las siguientes sentencias suponiendo que se encuentran en el cuerpo de la rutina `examen`:




```
examen(0, d, &aux);
```

```
leal   -4(%ebp), %eax
pushl  %eax
leal   -400(%ebp), %eax
pushl  %eax
pushl  $0
call   examen
```

```
for (aux = 0; aux < 100; aux++)
        b[aux] = d[aux];
```

```
  movl   $0, %esi
  for:
  cmpl   $100, %esi
  jge    endfor
  movl   -400(%ebp, %esi, 4), %ebx
  movl   %ebx, 12(%ebp, %esi, 4)
  incl   %esi
  jmp    for
endfor:
```

```
examen(a, b, c);
```

```
movl   16(%ebp), %ebx
pushl  %ebx
movl   12(%ebp), %ebx
pushl  %ebx
movl   8(%ebp), %ebx
pushl  %ebx
```