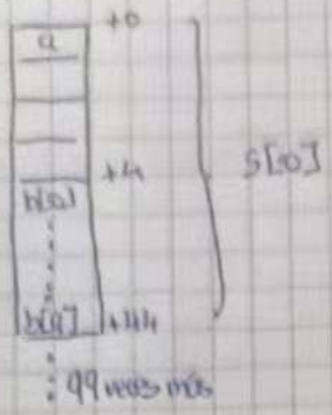
PROBLEMA 9

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

s → %ebx
i → %esi
j → %edi
x → %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] = 44i + 4j + 4 + @s[]

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

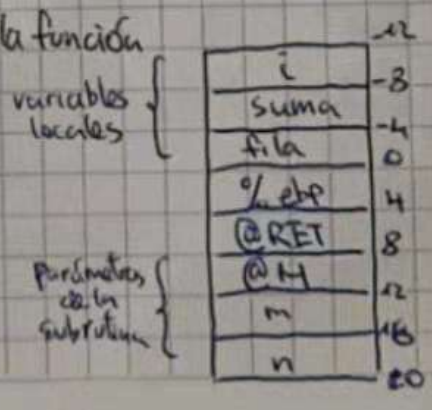
x = s[s[i].b[j]].a

```
Imull $44, %esi, %eax      || %eax ← 44i
addl %ebx, %eax            || %ebx ← 44i + @s[]
Imull $44, 4(%ebx,%eax,4), %eax || %eax ← 44(44i + @s[] + 4j + 4) || s[s[i].b[j]]
movb (%ebx, %eax), %dl     || x = s[s[i].b[j]].a
```

PROBLEMA 10

```
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);
}
```

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.

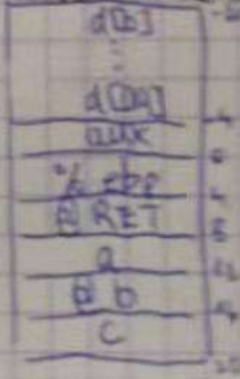
H[i,k][j] = @H + (i·NumCol + j)·4

```
CALCULA:  pushl %ebp
          movl %esp, %ebp
          subl $12, %esp
          pushl %ebx
          movl $0, -8(%ebp)      // suma = 0
          movl $0, -4(%ebp)      // i,k = 0
          movl 12(%ebp), %ebx    // %ebx ← M

for:      cmpl 16(%ebp), %ebx
          jge fifor              // with si o mayor o igual
          leal -4(%ebp), %eax    // %eax ← @i,k
          pushl %eax
          movl -4(%ebp), %edx    // %edx ← i,k
          imull $10, %edx        // %edx ← i,k * 10
          addl %ebx, %edx        // %edx ← i,k*10 + j
          movl 8(%ebp), %ecx     // %ecx ← @H
          movl (%ecx, %edx, 4), %edx
          pushl %edx
          call Normaliza
          addl %eax, -8(%ebp)    // suma = suma + Normaliza( )
          incl %ebx              // %ebx + 1
          jmp for

fifor:    movl 8(%ebp), %eax     // %eax ← suma
          incl %eax
          popl %ebx
          movl %ebp %esp
          popl %ebp
          ret
```

PROBLEMA 14

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

a) Dibuja el bloque de activación de la función



b) Traducid a ensamblador IA32 la siguiente sentencia suponiendo que se encuentra en el cuerpo de la rutina:

```
examen(0, d, baux);

    leal -4(%ebp), %eax
    pushl %eax
    leal -404(%ebp), %ecx
    pushl %ecx
    push $0
    call examen
```

c) Traducid a ensamblador IA32 la siguiente sentencia suponiendo que se encuentra en el cuerpo de la rutina:

```
for (aux=0; aux < 100, aux++){
   b[aux] = d[aux];
}
    movl $0, %ecx                    // aux = 0
for: cmpl $100, -4(%ebp)
    jge ffor
    leal -404(%ebp), %eax            // %eax ← d
    movl (%eax,%ecx,4), %eax         // %eax ← d[aux]
    movl 12(%ebp), %edx              // %edx ← b
    movl %eax, (%edx,%ecx,4)         // %edx ← %eax ( b[aux] = d[aux] )
    incl %ecx
    jmp for
ffor:
```

d) Traducid a ensamblador IA32 la siguiente sentencia suponiendo que se encuentra en el cuerpo de la rutina:

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

movl 16(%ebp), %eax
push %eax
movl 12(%ebp), %ecx
push %ecx
movl 8(%ebp), %edx
push %edx
call examen
```