Nombre:	
DNI:	**Grupo:**

Examen de Problemas (3.0 p)

1. **Ensamblador a C** (0.6 puntos). Dada la siguiente función en ensamblador x86-64:

```
loop:
# a la entrada: a en %rdi, n en %esi
    movl    $0, %r8d
    movl    $0, %ecx
    testl   %esi, %esi
    jle .L3
.L6:
    movl    (%rdi,%rcx,4), %edx
    leal    3(%rdx), %eax
    testl   %edx, %edx
    cmovns  %edx, %eax
    sarl    $2, %eax
    addl    %eax, %r8d
    addq    $1, %rcx
    cmpl    %ecx, %esi
    jg  .L6
.L3:
    movl    %r8d, %eax
    ret
```

Rellenar los huecos en el código C correspondiente.
Se debe usar sintaxis de indexación en arrays para acceder a los elementos de **a**.
Naturalmente, no se deben usar nombres de registros x86-64 en código C.
Si se duda sobre la equivalencia aritmética de la operación de desplazamiento, expresarla también como desplazamiento en lenguaje C.

```
int loop(int a[], int n)
{
    int i, sum;
    sum = _____;
    for (i = _________; _________; _________) {
        sum += _______________;
    }
    return _____________;
}
```

2. **Acceso a estructuras** (0.5 puntos). Sean las siguientes declaraciones de estructuras en C:

```
struct data {
    long x;
    char str[16];
    };
```

```
struct node {
    struct data d;
    struct node *next;
};
```

Abajo se ofrecen cinco funciones C y cinco bloques de código x86-64.
Anotar en cada bloque x86-64 el nombre de la función C que implementa.

```
int alpha(struct node *ptr){          _____   movsbl  15(%rdi),%eax
    return ptr->d.x;                          ret
}

char *beta(struct node *ptr){         _____   movq    (%rdi), %rax
    ptr = ptr->next;                          ret
    return ptr->d.str;
}

char gamma(struct node *ptr){         _____   movq    24(%rdi), %rax
    return ptr->d.str[7];                     addq    $8, %rax
}                                             ret

long *delta(struct node *ptr){        _____   movq    %rdi, %rax
    struct data *dp =                         ret
   (struct data *) ptr;
    return &dp->x;
}

char *epsilon(struct node *ptr){      _____   leaq    10(%rdi), %rax
    return &ptr->d.str[2];                    ret
}
```

3. **Acceso a matrices**. (0.5 puntos). Hemos escrito un archivo **f.c** que contiene una función **f** escrita en lenguaje C, a la que se le pasa la dirección de una matriz cuadrada de **N×N** números. La función **f** suma los elementos de la diagonal de la matriz.

```
#define N ...
typedef ... number;
int f (number v[N][N]) {
   ...
}
```

Compilamos dicha función mediante la siguiente orden:
`gcc f.c -m32 -O1 -S -fno-omit-frame-pointer`
obteniendo un archivo **f.s** con el siguiente contenido:

```
f:
   pushl     %ebp
   movl      %esp, %ebp
   pushl     %ebx
   movl      8(%ebp), %ebx
   movl      %ebx, %edx
   addl      $10100, %ebx
   movl      $0, %eax
```

```
.L2:
   movsbl    (%edx), %ecx
   addl      %ecx, %eax
   addl      $101, %edx
   cmpl      %ebx, %edx
   jne       .L2
   popl      %ebx
   popl      %ebp
   ret
```

a) ¿Qué registros son modificados en el cuerpo de la función **f**?

b) ¿Qué registros modificados se guardan en la pila y cuáles no, y por qué?

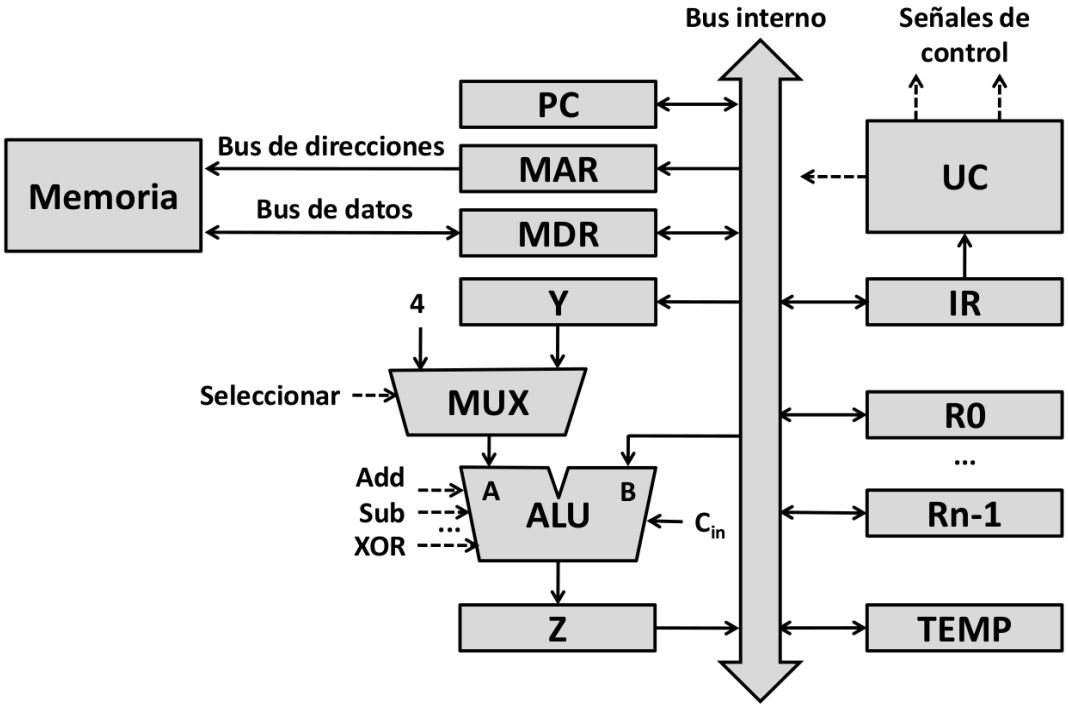
c) ¿Cuánto vale **N**?

d) ¿De qué tipo son los elementos de la matriz?

e) ¿Dónde devuelve el resultado la función?

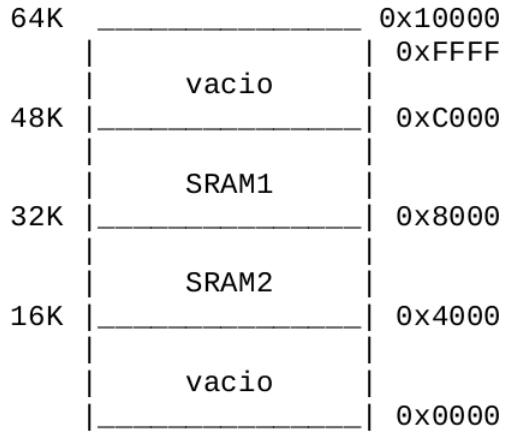
Escriba la función f completa en C (contenido del archivo **f.c**)

4. **Unidad de control** (0.5 puntos). Exprese en lenguaje RTL (transferencia entre registros) o pseudocódigo las operaciones elementales que se producen en cada uno de los ciclos de la ejecución de la instrucción de una palabra `BRANCH desplazamiento`, incluyendo la fase de captura de instrucción (*fetch*), para la siguiente microarquitectura:



La lectura de memoria se realiza en un solo ciclo. El bus interno y todos los registros tienen un tamaño de *n* bits. El desplazamiento es un número con signo relativo al contador de programa, ocupará los *m* bits menos significativos ($m < n$) del registro `IR` cuando la instrucción pase a dicho registro, y la salida del registro `IR` hacia el bus interno es ese campo desplazamiento con el signo extendido a *n* bits.

5. **Diseño de memoria** (0.5 puntos). Un sistema basado en un pequeño microprocesador (con un bus de datos de 8 bits, un bus de direcciones de 16 bits y una patilla R/W#) dispone del siguiente mapa de memoria:



Dibuje la decodificación y conexionado de los dos módulos de memoria SRAM1 y SRAM2 al microprocesador.

6. **Memoria cache** (0.4 puntos). Sea un computador de 32 bits con una memoria cache de datos asociativa por conjuntos de 2 vías con líneas de 64 bytes y un tamaño total de 32 KB, y que emplea la política de reemplazo LRU.

a) Indique el número de líneas y de conjuntos de la cache.

b) Dado el fragmento de código:

```
int v[1048576];
for (i = 0; i < 1048576; i += 4)
   v[i] = i;
```

y considerando exclusivamente los accesos al vector, calcule y razone la tasa de fallos que se obtiene en dicha cache de datos en la ejecución del bucle anterior.