```
/* Considere la siguiente representacion en punto flotante de 9 bits, basada en el
 * formato de la IEEE:
 *
 *    - El bit mas significativo es un bit de signo (s),
 *    - le siguen 3 bits de exponente (e), y
 *    - los ultimos 5 bits son la fraccion (f).
 *
 * Se cumplen las mismas reglas que en el estandar IEEE para definir numeros
 * normalizados, denormalizados, infinitos, NaN y representacion del 0. Complete
 * la siguiente tabla, donde:
 *
 *    - *binario* es la representacion en 9 bits
 *    - *M* es el significando. Puede ser un numero x o x/y, donde x e y son
 *      enteros. Ejemplo: 0, 3/256.
 *    - *E* es el exponente (Â¡distinto de e!).
 *
 * Si tiene que redondear, hagalo segun las reglas de redondeo al par mas cercano
 * (Round-To-Even).
 *
 * |   VALOR   |    binario     |        M        |       E         |
 * |-----------|----------------|-----------------|-----------------|
 * |      -3.0 |                |                 |                 |
 * |-----------|----------------|-----------------|-----------------|
 * |         0 |                |                 |                 |
 * |-----------|----------------|-----------------|-----------------|
 * |    12.012 |                |                 |                 |
 * |-----------|----------------|-----------------|-----------------|
 * |    1.3333 |                |                 |                 |
 * |-----------|----------------|-----------------|-----------------|
 * |  0.000125 |                |                 |                 |
 * |-----------|----------------|-----------------|-----------------|
 *
 *
 * Se tiene una cache de 2048 bytes con mapeo directo y bloques de 32 bytes.
 * Dadas las siguientes definiciones:
 *
 * extern unsigned long casos_diarios[512];
 *
 * y asumiendo:
 *
 *    - El sizeof corresponde a una arquitectura x86-64 (64 bits).
 *    - `casos_diarios` comienza en la direccion 0.
 *    - La memoria cache esta inicialmente vacia.
 *
 * a) Indique con que codigo assembly se corresponde esta funcion
 * b) Haga un analisis e indique el porcentaje de escrituras en la cache
 *    que va a fallar el codigo.
 */
unsigned long ma(unsigned long * v, unsigned long l, unsigned long n) {
    unsigned long i, j;
    for (i = n-1; i < l; i++) {
        unsigned long s = 0;
        for (j = 0; j < n; j++)
            s += v[i - j];
        s /= n;
        v[i - j] = s;
    }
    return l - n;
}
```

```
ma:
        movq    %rdx, %r9
        leaq    -1(%rdx), %r8
        jmp .L2
.L3:
        addq    (%rdi,%rcx,8), %rax
        subq    $1, %rcx
        jmp .L4
.L5:
        movq    %r9, %rcx
        movl    $0, %eax
.L4:
        testq   %rcx, %rcx
        jne .L3
        movl    $0, %edx
        divq    %r9
        movq    %rax, (%rdi,%r8,8)
        addq    $1, %r8
.L2:
        cmpq    %rsi, %r8
        jb  .L5
        movq    %rsi, %rax
        subq    %r9, %rax
        ret
```

```
 1: ma:
 2:          movq    %rdx, %r8
 3:          leaq    -1(%rdx), %r9
 4:          jmp .L2
 5: .L3:
 6:          addq    (%rdi,%rcx,8), %rax
 7:          addq    $1, %rcx
 8:          jmp .L4
 9: .L5:
10:          movl    $0, %eax
11:          movl    $0, %ecx
12: .L4:
13:          cmpq    %r8, %rcx
14:          jb  .L3
15:          movl    $0, %edx
16:          divq    %r8
17:          movq    %rax, (%rdi,%r9,8)
18:          addq    $1, %r9
19: .L2:
20:          cmpq    %rsi, %r9
21:          jb  .L5
22:          movq    %rsi, %rax
23:          subq    %r8, %rax
24:          ret
```

```
    1: ma:
    2:          movq    %rdx, %r8
    3:          leaq    -1(%rdx), %r10
    4:          jmp .L2
    5: .L3:
    6:          movq    %r10, %rdx
    7:          subq    %rcx, %rdx
    8:          movq    (%rdi,%rdx,8), %rax
    9:          movl    $0, %edx
   10:          divq    %r8
   11:          addq    %rax, %r9
   12:          addq    $1, %rcx
   13:          jmp .L4
   14: .L5:
   15:          movl    $0, %r9d
   16:          movl    $0, %ecx
   17: .L4:
   18:          cmpq    %r8, %rcx
   19:          jb  .L3
   20:          movq    %r10, %rdx
   21:          subq    %rcx, %rdx
   22:          movq    %r9, (%rdi,%rdx,8)
   23:          addq    $1, %r10
   24: .L2:
   25:          cmpq    %rsi, %r10
   26:          jb  .L5
   27:          movq    %rsi, %rax
   28:          subq    %r8, %rax
   29:          ret
```

```
    1: ma:
    2:          movq    %rdx, %r9
    3:          leaq    -1(%rdx), %r8
    4:          jmp .L2
    5: .L3:
    6:          movq    %r8, %rdx
    7:          subq    %rcx, %rdx
    8:          addq    (%rdi,%rdx,8), %rax
    9:          addq    $1, %rcx
   10:          jmp .L4
   11: .L5:
   12:          movl    $0, %eax
   13:          movl    $0, %ecx
   14: .L4:
   15:          cmpq    %r9, %rcx
   16:          jb  .L3
   17:          movl    $0, %edx
   18:          divq    %r9
   19:          movq    %r8, %rdx
   20:          subq    %rcx, %rdx
   21:          movq    %rax, (%rdi,%rdx,8)
   22:          addq    $1, %r8
   23: .L2:
   24:          cmpq    %rsi, %r8
   25:          jb  .L5
   26:          rep ret
```

```
 1: ma:
 2:          movq    %rdx, %r9
 3:          movl    $0, %r8d
 4:          jmp .L2
 5: .L3:
 6:          leaq    (%r8,%rcx), %rdx
 7:          addq    (%rdi,%rdx,8), %rax
 8:          addq    $1, %rcx
 9:          jmp .L4
10: .L5:
11:          movl    $0, %eax
12:          movl    $0, %ecx
13: .L4:
14:          cmpq    %r9, %rcx
15:          jb  .L3
16:          movl    $0, %edx
17:          divq    %r9
18:          movq    %rax, (%rdi,%r8,8)
19:          addq    $1, %r8
20: .L2:
21:          movq    %rsi, %rax
22:          subq    %r9, %rax
23:          cmpq    %rax, %r8
24:          jb  .L5
25:          rep ret
```

```
    1: ma:
    2:          movq    %rdx, %r9
    3:          leaq    -1(%rdx), %r8
    4:          jmp .L2
    5: .L3:
    6:          movq    %r8, %rdx
    7:          subq    %rcx, %rdx
    8:          addq    (%rdi,%rdx,8), %rax
    9:          addq    $1, %rcx
   10:          jmp .L4
   11: .L5:
   12:          movl    $0, %eax
   13:          movl    $0, %ecx
   14: .L4:
   15:          cmpq    %r9, %rcx
   16:          jb  .L3
   17:          movl    $0, %edx
   18:          divq    %r9
   19:          movq    %r8, %rdx
   20:          subq    %rcx, %rdx
   21:          movq    %rax, (%rdi,%rdx,8)
   22:          addq    $1, %r8
   23: .L2:
   24:          cmpq    %rsi, %r8
   25:          jb  .L5
   26:          movq    %rsi, %rax
   27:          subq    %r9, %rax
   28:          ret
```