

Test de Teoría (3.0p)

| | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 |
| b | b | d | c | b | b | b | b | b | d | a | c | c | c | b | c | d | a | b | b | d | b | b | b | b | c | d | a | a | c |

Test de Prácticas (4.0p)

| | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|----|----|----|----|----|----|----|----|----|----|----|
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 |
| a | a | a | a | c | d | c | a | b | d | d | a | d | c | d | b | c | a | a | d |

Examen de Problemas (3.0p)

1. Ensamblador (0.8 puntos).

Nota completa **0.80p** si el resultado es correcto. Las soluciones ocupan unas 15 líneas incluyendo etiquetas, por eso en modo “rescate” se puntúa **0.05p** por cada línea “acertada” (total **0.05 x 15 = 0.75p**).

Algunas alternativas posibles:

Solución de gcc 9.2 con -Og:

```
big2little:
    movq %rdi, -8(%rsp)      ; src.n = n;
    movl $0, %eax           ; for (i=0; ...
    jmp .L2                 ; jump-to-middle

.L3:
    movl $7, %edx           ; NUM_SIZ-1 ...
    subq %rax, %rdx         ; ... - i
    movzbl -8(%rsp,%rdx), %edx ; src.b[ N-1-i]
    movb %dl, -16(%rsp,%rax) ; dst.b[i] = ...
    addq $1, %rax           ; for (... i++)

.L2:
    cmpq $7, %rax           ; for (... i<8; ...
    jbe .L3                 ; unsigned i
    movq -16(%rsp), %rax    ; return dst.n;
    ret
```

Solución de gcc 9.2 con -Os:

```
big2little:
    movq %rdi, -16(%rsp)
    xorl %eax, %eax
    leaq -9(%rsp), %rcx

.L2:
    movq %rcx, %rdx
    subq %rax, %rdx
    movb (%rdx), %dl
    movb %dl, -8(%rsp,%rax)
    incq %rax
    cmpq $8, %rax
    jne .L2
    movq -8(%rsp), %rax
    ret
```

Solución de gcc 9.2 con -O1:

```
big2little:
    movq %rdi, -8(%rsp)
    leaq -1(%rsp), %rax
    leaq -16(%rsp), %rdx
    leaq -8(%rsp), %rsi

.L2:
    movzbl (%rax), %ecx
    movb %cl, (%rdx)
    movq %rax, %rcx
    subq $1, %rax
    addq $1, %rdx
    cmpq %rsi, %rcx
    jne .L2
    movq -16(%rsp), %rax
    ret
```

Solución de gcc 9.2 con -O2:

```
big2little:
    movq %rdi, -16(%rsp)
    leaq -16(%rsp), %rsi
    leaq -9(%rsp), %rax
    leaq -8(%rsp), %rdx

.L2:
    movzbl (%rax), %ecx
    addq $1, %rdx
    movb %cl, -1(%rdx)
    movq %rax, %rcx
    subq $1, %rax
    cmpq %rsi, %rcx
    jne .L2
    movq -8(%rsp), %rax
    ret
```

Solución de Tacho:

```
big2little:
    mov $8, %ecx

loop:
    shl $8, %rax
    movb %dil, %al
    shr $8, %rdi
    # loop loop
    dec %ecx
    jnz loop
    ret
```

Solución de gcc 9.2 con -O3:

```
big2little:
    movq %rdi, %rax
    bswap %rax
    ret
```

⇐ (no se esperaba que nadie conociera bswap)

↑ (no se esperaba que nadie conociera loop)
Versión más eficiente, no usa memoria
Solución encontrada por Ahmed Brek y Atanasio Rubio

2. Ensamblador(0.2 puntos).

Se puntúa **0.1p** por cada apartado (total **0.1 x 2 = 0.20p**),

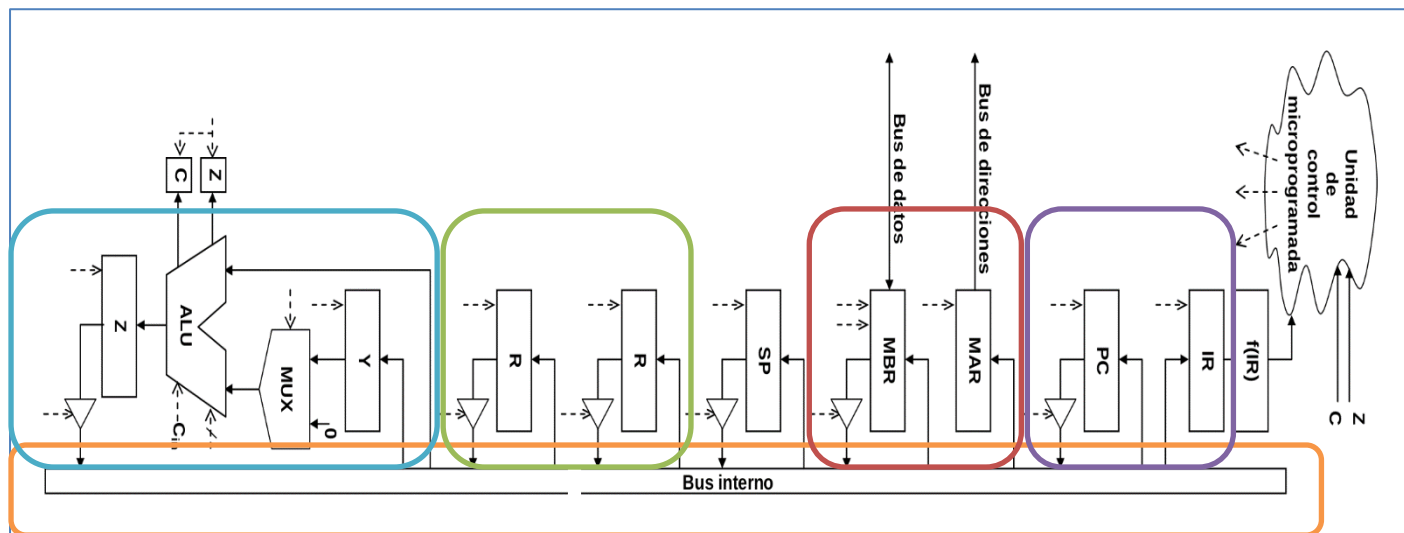
a) signed long, por imulq y el factor de escala 8

b) viene especificado por el registro rdx, es decir, se pasa como tercer parámetro de la función

3. Unidad de control (0.5 puntos).

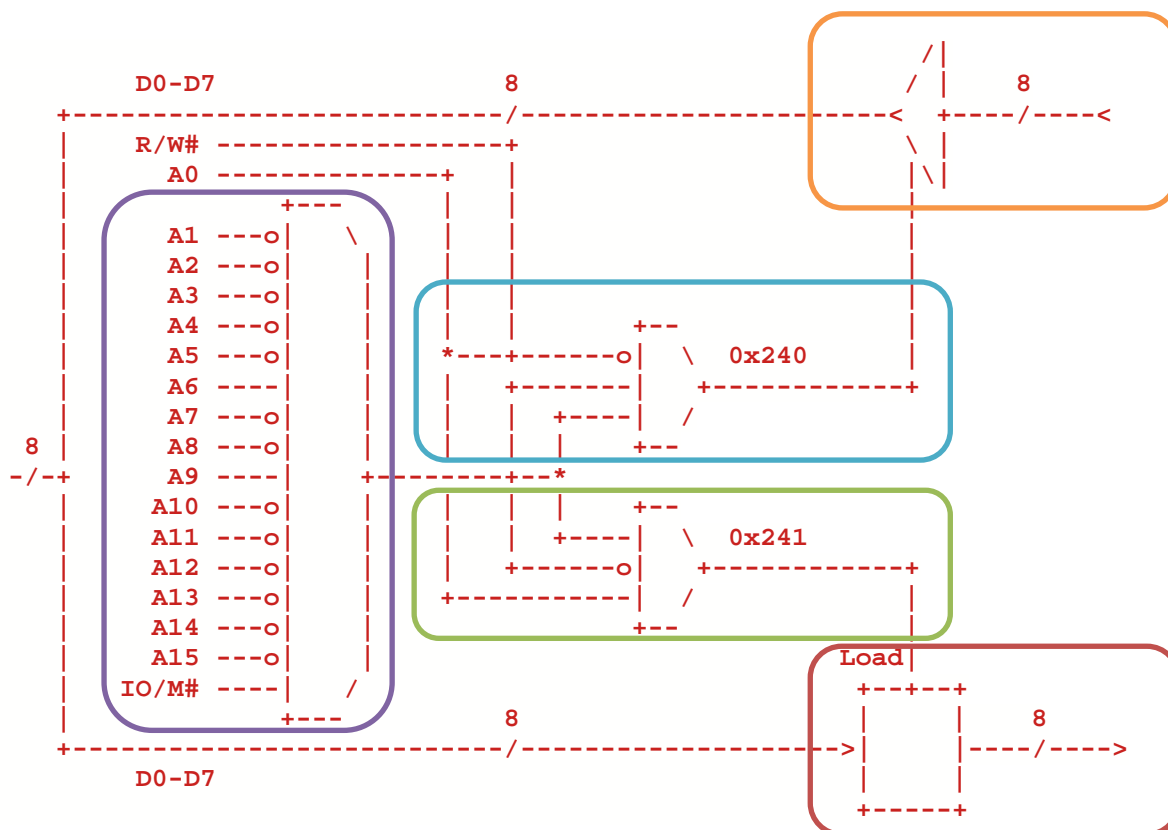
Se puntúa **0.1p** por cada zona del dibujo (total **0.1 x 5 = 0.5p**).

El MUX no es imprescindible: ALU/Y/N/Buffer puntúan 0.1p aunque falte el MUX.



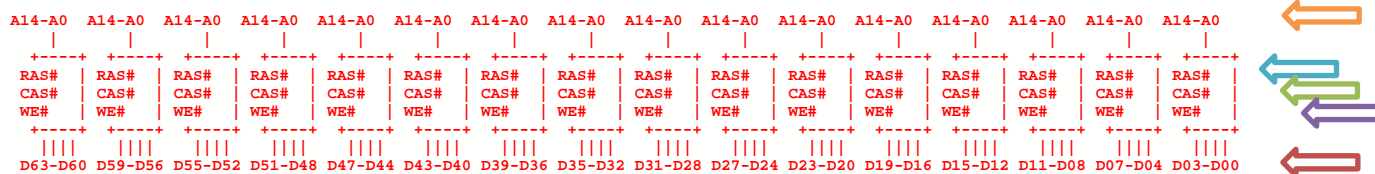
4. Entrada/Salida (0.5 puntos).

Se puntúa **0.1p** por cada zona del dibujo (total **0.1 x 5 = 0.5p**).



5. Configuración de memoria (0.5 puntos).

Se puntúa **0.05p** por cada zona del dibujo (total **0.05 x 10 = 0.50p**).



6. Cache (0.5 puntos).

Se puntúa **0.1p** por cada zona del dibujo con todos los (números/flechas) correct@s (total **0.1 x 5 = 0.5p**).

$$64 \text{ GB} = 2^{36} \text{ B}$$

$$2^{36} \text{ B} / 2^6 \text{ B/block} = 2^{30} \text{ blocks} \quad (1\ 073\ 741\ 824)$$

$$12 \text{ MB} = 3 \times 2^{22} \text{ B}$$

$$3 \times 2^{22} \text{ B} / 2^6 \text{ B/line} = 3 \times 2^{16} \text{ lines} \quad (196\ 608) \quad (-12 = 196\ 596)$$

$$12\text{-way} = 3 \times 2^2 \text{ lines/set}$$

$$3 \times 2^{16} \text{ lines} / (3 \times 2^2 \text{ lines/set}) = 2^{14} \text{ sets} \quad (16\ 384)$$

De los conjuntos (0/1/último=31) se dibujan las dos líneas (0/1, 2/3, 62/63)

Pasan a ser conjuntos (0/1/16383), primera/última líneas (0/11, 12/23, 196 596/196 607)

De memoria se dibujan los bloques que van a conj. 0/1 con etiqueta Tag=0/2,

es decir, bloques (0/1) (Tag=0), bloques (64/65) (Tag=64/32=2),

pasan a ser bloques (0/1) Tag=0, bloques (32 768/32 769)

...

un bloque (256) que va al conj. 0 con Tag=256/32=2^3=8

pasa a ser bloque (131 072), conj.0 Tag=8

...

y también el anterior y el último (255/4095), que van al conj.31

pasan a ser (131 071/1 073 741 823)

