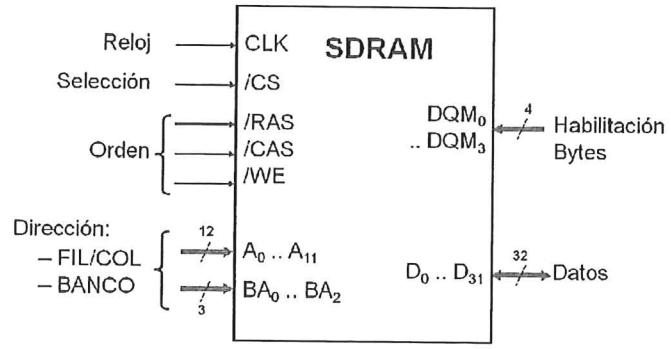
ESTRUCTURA DE COMPUTADORES

Recuperacion Primer Parcial **14-Mayo-2012**

Apellidos y Nombre	DNI	Grupo

1. La figura adjunta muestra el patillaje de un chip de memorias SDRAM. El ancho de palabra del chip es de 32 bits y las direcciones de fila son de 12 bits. La lectura/escritura se realiza por bloques de longitud programable, siendo el máximo tamaño de bloque de 256 palabras, correspondientes a una fila completa de las matrices de bits. **(2 puntos)**



a) Calcule la capacidad del chip en bytes. Indique también el número de bancos, y filas y columnas de cada banco.

$2^3 \times 2^{12} \times 2^8 = 2^{23}$ palabras $\times$ 4B = 32MB

bancos Filas Columnas

b) Suponiendo un tamaño de bloque de 8 palabras (32 bytes), indique el número total de bloques del chip y de cada banco.

$\frac{32MB}{32B} = 1M$ bloques → 128K bloques/banco

c) En el supuesto anterior, indique en qué banco, fila y columna comienza el bloque número 0x68204. Asuma un direccionamiento lineal del chip (banco-fila-columna).

0110 1000 0010 0000 0110 0000

Banco (0x3) Fila (0x410) Columna (0x20)

d) Si el chip se conecta a un bus con reloj de 200 MHz, ¿cuál será el ancho de banda máximo de este chip?

frec x tamaño_bus = 200 MHz x 4B = 800 MB/s

e) Se desean conectar dos de estos chips a un procesador de 32 bits con espacio de direccionamiento de 256 MB. Para ello se ha diseñado un descodificador de direcciones con las funciones de selección siguientes:

F_{SDRAM1} * = $A_{27}+A_{26}+A_{25}$ F_{SDRAM2} * = $A_{27}+A_{26}+A_{25}$*

Indique los rangos de direcciones que ocupan cada uno de los dos chips SDRAM. Indique también, qué direcciones del mapa quedan libres.

SDRAM1 → 0x00000000 – 0x1FFFFFFF
SDRAM2 → 0x20000000 – 0x3FFFFFFF ⇒ 64MB

Libre → 0x40000000 – 0xFFFFFFFF ⇒ 192MB

2. Considérese el programa que se muestra a continuación:

```
                .data 0x10005000
v:              .word 0,1,2,3,4,5,6,7

                .text 0x00400000
                lui $t0, 0x1000
                ori $t0, $t0, 0x5000
                addi $t1, $zero, 8
bucle:          lw $t2, 0($t0)
                bgez $t2, salta
                sw $zero, 0($t0)
salta:          addiu $t0, $t0, 4
                addi $t1, $t1, -1
                bgtz $t1, bucle
```

Conteste las cuestiones siguientes justificando la respuesta: **(1.5 puntos)**

a) ¿Cuántos bytes ocupan los segmentos de datos y código, respectivamente?

Código → 9 x 4B = 36B
Datos → 8 x 4B = 32B

b) Indique que instrucción se encuentra en la dirección de memoria 0x00400018.

addiu $t0, $t0, 4

c) ¿En que dirección de memoria se encuentra el elemento **v[7]**?

0x1000501C

d) ¿Cuántas instrucciones se ejecutan?

3 + 5 x 8 = 43 instrucciones

e) ¿Cuántos accesos y de que tipo (lectura o escritura) se hacen al segmento de datos y código, respectivamente?

datos → 8 lecturas + ~~[illegible]~~
código → 43 lecturas

3. Se dispone de un procesador MIPS R2000 con una memoria cache de datos de 32 KB de capacidad, con tamaño de línea de 16 bytes y correspondencia asociativa por conjuntos de 2 vías, algoritmo de reemplazo LRU, política de ubicación en escritura y escritura posterior. El procesador ejecuta el siguiente código:
(2.5 puntos)

```
        . data 0x10000000
A:      .space 16384
        .data 0x10040000
B:      .space 16384
        .data 0x10080000
N:      .byte 4096

        .text 0x00400000
        .globl __start
```

```
__start: lui $t1, 0x1000
         lui $t2, 0x1004
         lui $t3, 0x1008
         lbu $t3, 0($t3)
bucle:   lw $t0, 0($t1)
         slt $t4, $t0, $zero
         beq $t4, $0, positivo
         sw $zero, 0($t2)
         j sigue
positivo: sw $t0, 0($t2)
sigue:   addiu $t1, $t1, 4
         addiu $t2, $t2, 4
         addi $t3, $t3, -1
         bne $t3, $0, bucle
         li $v0, 2
         syscall
         .end
```

a) Describe brevemente que acción realiza el código sobre los vectores A y B.

Transfiere vector A sobre vector B reemplazando los valores negativos por "zero"
Si A[i] ≥ 0 entonces B[i] = A[i], sino B[i] = 0

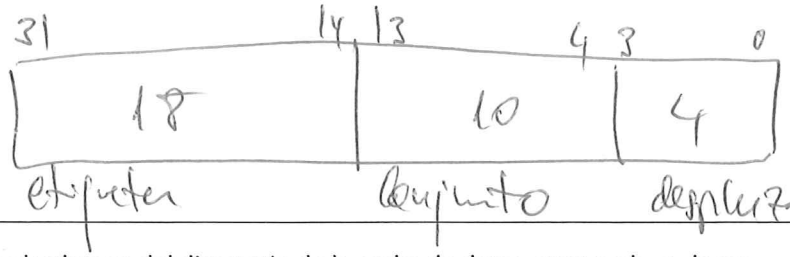
b) Indica el número de accesos a la cache de datos por parte del procesador.

1 + 2×4096 = 8.193 accesos.

c) Indica el número de bloques accedidos por el procesador al segmento de datos.

A → 1024 bloques (lecturas)
B → 1024 " (escrituras)
N → 1 bloque (lectura)

d) Interpreta las direcciones en el acceso a la cache de datos según sus campos



e) Calcula el volumen del directorio de la cache de datos, expresado en bytes.

etiqueta (18) + V(1) + M(1) + LRU(1) = 21 bits.

2048 bloques x 21 bits =

f) Calcula la tasa de aciertos de la cache de datos.

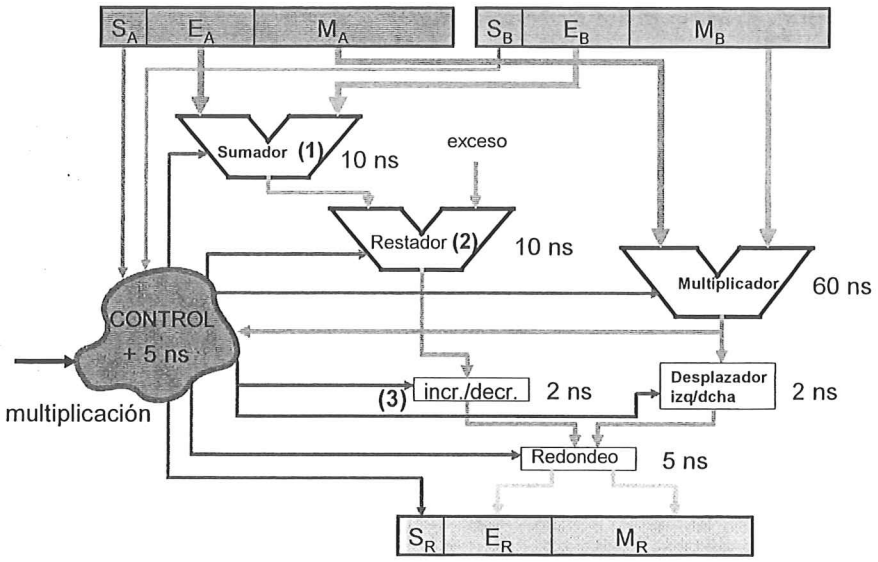
Fallos = 1 + 2x1024 = 2.049

Ta = $1 - \frac{2.049}{8.193}$ =

g) ¿Se obtendría para este programa una mayor tasa de aciertos en la cache de datos si la correspondencia fuera directa? Razona la respuesta

NO. La Ta sería = 0 porque los vectores A y B colisionarían en el mismo conjunto (1 vía) al que mapean.

4. El circuito de la figura es un multiplicador en coma flotante para números expresados según el formato definido por la norma IEEE 754. Los retardos de cada uno de los elementos que lo conforman se muestran también en la figura. Se debe tener en cuenta que la circuitería de control tarda 5 ns desde que tiene disponible la información de entrada hasta que genera las señales de control correspondientes. **(1 punto)**



a) Indique la función que desempeñan los elementos marcados como (1), (2) y (3) a la hora de realizar la operación A*B, donde A y B son dos números enteros en coma flotante (IEEE 754)

(1) Suma los exponentes $E_A + E_B = (exp_A + 2^7) + (exp_B + 2^7)$

(2) Compensa el exceso adicional introducido al sumar los exponentes $E_R = E_A + E_B - 2^7 = exp_A + exp_B + 2^7$

(3) Incrementa/decrementa exponente resultado. para compensar el desplazamiento de la mantisa para su normalización

b) ¿Cuál es el tiempo necesario para realizar una operación de multiplicación en coma flotante? Justifique los retardos en función de los pasos que se realizan para llevar a cabo la operación.

$5ns + \max(10ns + 10ns, 60ns) + 5ns + 2ns + 5ns = 77ns$

- 5ns: control (órdenes a (1), (2) y multiplicador)
- max(10ns + 10ns, 60ns): control en función del resultado de la multiplicación de las mantisas (órdenes a (3) y seg. desplazamiento)
- 2ns: seg. desplazamiento
- 5ns: La acción de control sobre Redondeo no está en camino crítico

5. Se desea confeccionar, para el MIPS R2000, una rutina de multiplicación de un valor de 8 bits con signo por una constante ($-18_d \equiv 11101110_b$), cuyos parámetros de entrada y salida se muestran a continuación: **(2 puntos)**

Nombre función	Parámetros de entrada	Parámetros de salida
`Mult_18n`	`$a0= multiplicando`	`$v0= producto`

Supóngase que se han definido las siguientes variables y que el programa que realiza la multiplicación P= M × (-18), invocando a tal fin la rutina **`Mult_18n`**, es el que se muestra más abajo:

```
        .data
M:      .byte     0        # multiplicando {-128,127}
P:      .space    ¿?       # producto
W:      .word     0        # entero SIN signo

        .text
        lb $a0,M           # carga multiplicando
        jal Mult_18n       # invoca subrutina $a0 x (-18)
        ¿? $v0,P           # almacena el producto
```

Se pide:

a) ¿Cuál sería la codificación de Booth para la constante (-18)?

0 0 -1 1 0 0 -1 0

b) ¿Qué ventajas aporta el empleo de la codificación de Booth en el multiplicador respecto a la codificación en Ca2?

1) Tratamiento similar de enteros positivos y negativos y de números con signo y sin signo.
2) En determinados casos podría reducir el número de productos intermedios (menos op. suma en circuito).

c) ¿Cuál deberá ser el tamaño (número de bytes) de la variable P? ¿Qué instrucción emplearía el programa para almacenar $v0 en la variable P?

P → 2 bytes (.half).

lh $t0, P

d) Confeccionar el código de la rutina `Mult_18n` empleando la instrucción de multiplicación disponible en la arquitectura MIPS R2000.

```
Mult_18n: li $t0, -18
          mult $a0, $t0
          mflo $v0
          jr $ra
```

e) Confeccionar el código de la rutina `Mult_18n` empleando las instrucciones de suma, resta y desplazamiento disponibles en la arquitectura MIPS R2000 para que la multiplicación se realice aprovechando que el multiplicador se halla representado empleando la codificación de Booth.

$-18 = -2^5 + 2^4 - 2^1$

```
Mult_18n: sll  $t0, $a0, 4      # M x 2^4
          sll  $t1, $a0, 5      # M x 2^5
          sub  $t0, $t0, $t1    # (-2^5 + 2^4) * M
          sll  $t1, $a0, 1      # M x 2
          sub  $v0, $t0, $t1    # (-2^5 + 2^4 - 2) * M
          jr   $ra
```

f) Compara brevemente las ventajas y/o inconvenientes de las anteriores implementaciones.

La primera implementación, aunque requiere menos instrucciones, tiene un tiempo de ejecución mayor debido al mayor retardo de "mult"

g) Teniendo en cuenta el tipo de la variable P y que W es un entero de 32 bits sin signo,v indica en qué supuesto se podría producir desbordamiento en la asignación W=P.

Desbordamiento si P<0

6. El siguiente fragmento de código basado en el ensamblador del MIPS R2000 va a ejecutarse en dos tipos diferentes de procesador. El Procesador 1 tiene una ruta de datos uniciclo con una frecuencia de reloj de 100 MHz. El Procesador 2 es de tipo multiciclo y la frecuencia de su reloj es de 500 MHz. Este último, requiere 6 ciclos de reloj para ejecutar las instrucciones que conllevan acceso a la cache de datos, 3 ciclos para ejecutar las instrucciones aritmético-lógicas y solo 2 ciclos para las instrucciones de salto condicional. Dado el siguiente fragmento de código: **(1 punto)**

```
                lw $1, 100($2)
                beq $1, $2, etiqueta
                and $3, $1, $4
etiqueta:       lw $1, 50($3)
                sw $1, 100($2)
                sub $5, $1, $2
                lw $1, 10($5)
```

Nota: Suponemos que el salto no tiene lugar.

a) Indíquese el tiempo necesario para ejecutarlo en cada uno de los procesadores.

Procesador 1 $T_{clk} = 10ns$ $T_{P1} = 7 instrucc. \times 10ns = 70ns$

Si salto → (60ns)

(Si salto → 58 ns)

Procesador 2 $T_{clk} = 2ns$

$6T + 2T + 3T + 6T + 6T + 3T + 6T = 32T \rightarrow 64ns$

b) Cuál es la Productividad que se consigue en cada caso.

Procesador 1 $\chi_{P_1} = \frac{7}{70ns} = 100$ MIPs

Procesador 2 $\chi_{P_2} = \frac{7}{64ns} =$