UNIVERSIDAD CARLOS III DE MADRID
DEPARTAMENTO DE INFORMÁTICA
GRADO EN INGENIERÍA INFORMÁTICA. ESTRUCTURA DE COMPUTADORES

Para la realización del presente examen se dispondrá de **1 hora y media.**
NO se podrán utilizar libros, apuntes **ni** calculadoras de ningún tipo.

Ejercicio 1. Represente en el estándar IEEE 754 de simple precisión el valor **-36**.

Ejercicio 2. Dado el siguiente fragmento de programa en ensamblador.

```
.text
                .globl main
 main:
                li   $a0,  5
                jal  funcion
                move $a0, $v0
                li   $v0, 1
                syscall

                li   $v0, 10
                syscall

funcion:        li $t0, 10
                bgt $a0, $t0, et1
                li $t0, 10
                add $v0, $t0, $a0
                b et2
      et1:      li $t0, 8
                add $v0, $t0, $a0
      et2:      jr $ra
```

Se pide :

a) Indicar de forma razonada el valor que se imprime por pantalla (primera llamada al sistema del código anterior).

Ejercicio 3. Considere una función denominada `SumarValor`. A esta función se le pasan tres parámetros:

- El primer parámetro es la dirección de inicio de un vector de números enteros.
- El segundo parámetro es un valor entero que indica el número de componentes del vector.
- El tercer parámetro es un valor entero.

La función `SumarValor` modifica el vector, sumando el valor pasado como tercer parámetro a todas las componentes del vector. Se pide:

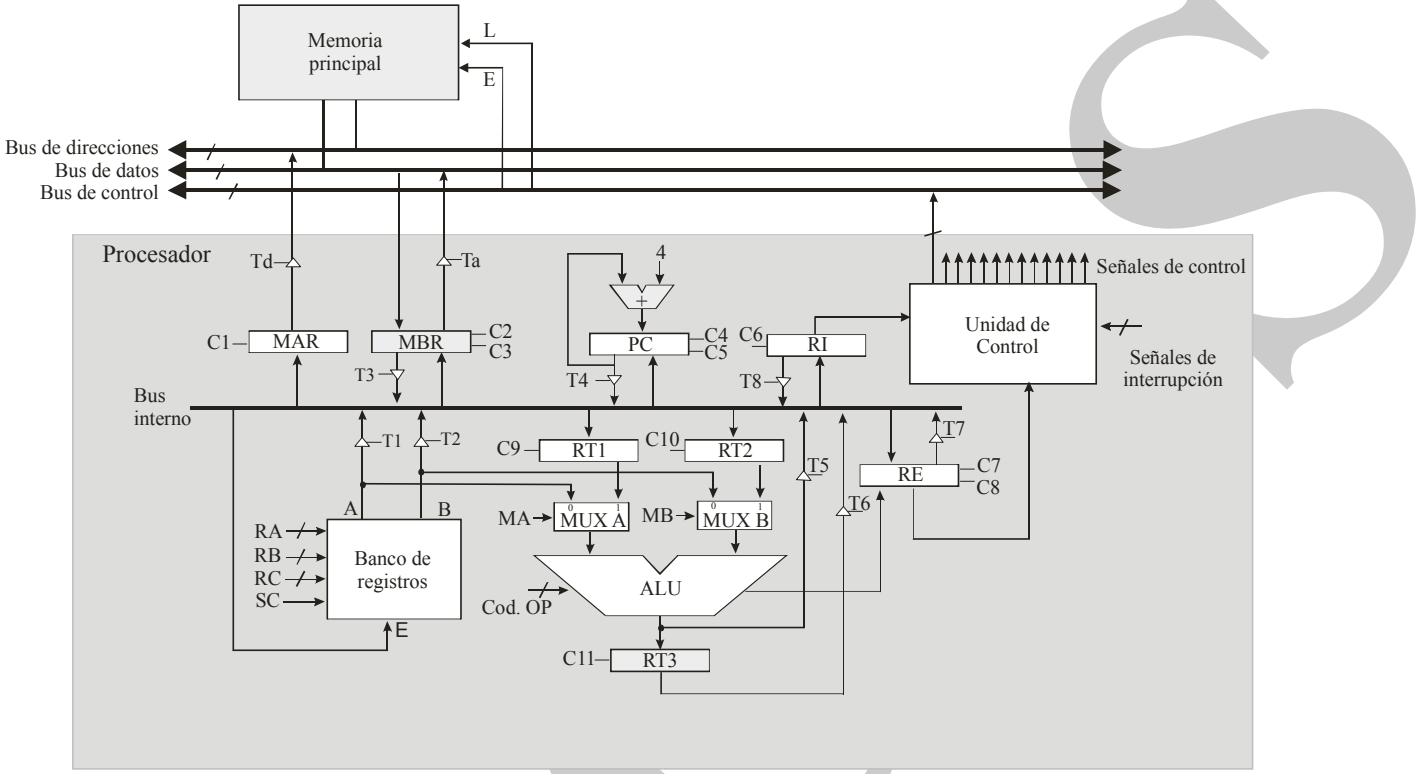
a) Indique en qué registros se han de pasar cada uno de los parámetros a la función.
b) Programar utilizando el ensamblador del MIPS32 el código de la función `SumarValor`.
c) Dado el siguiente fragmento de programa:

```
.data
        v:  .word    7, 8, 3, 4, 5, 6
.text
            .globl main

 main:
```

incluya en el `main` anterior, las sentencias en ensamblador necesarias para poder invocar a la función `SumarValor` implementada en el apartado b) de forma que sume a las componentes del vector `v` definido

en la sección de datos, el número 5. Implemente a continuación de la llamada a la función, las sentencias en ensamblador que permitan imprimir todas las componentes del vector.

Ejercicio 4. Considere el siguiente esquema de procesador de 32 bits. El banco de registros incluye 32 registros. Considere que el computador utiliza un ciclo de reloj para realizar la decodificación de la instrucción y que se conecta a una memoria que permite realizar una operación de lectura y de escritura en un ciclo.



Este computador dispone del juego de instrucciones del MIPS32. Indique las operaciones elementales y las señales de control (incluyendo el *fetch*) necesarias para ejecutar la instrucción `addi $t0, $t1, 10`.

Ejercicio 5. Sea un computador de 32 bits con una memoria caché de 256 KB, líneas de 64 bytes y un tiempo de acceso de 5 ns. La caché es asociativa por conjuntos de 4 vías y se emplea la política de reemplazo LRU. Se pide:

a) Indique el número de líneas y de conjuntos de la memoria caché del enunciado.
b) ¿Cuál es el tamaño de los bloques que se transfieren entre la memoria caché y la memoria principal?
c) Si el tiempo para transferir un bloque de memoria principal a caché es de 200 ns, indique la tasa de aciertos a caché necesaria, de forma que el tiempo medio de acceso al sistema de memoria sea de 20 ns.

Soluciones:

Solución del ejercicio 1

El valor 36 en binario es 100100. 100100 = 1.00100 x 2^5. Por tanto:

- El bit de signo es 1, porque el número es negativo.
- El exponente es 5, por tanto el exponente que se almacena es 5 + 127 = 132, que en binario es 10000100
- La mantisa es 001000000 …. 00000

Por tanto el número -36 se representa como 11000010000100000000000000000000

Solución del ejercicio 2

A la función se le pasa como argumento en el registro `$a0` el valor 5. Si este valor es mayor que (`bgt`) 10 se salta a la etiqueta `et1`. Como no es el caso, se suma 10 (`$t0`) a 5 y el resultado se almacena en `$v0`, que es el valor que se imprime, es decir, 15.

Solución del ejercicio 3

a) La dirección de inicio se pasa en $a0, el número de elementos en `$a1` y el valor a sumar en `$a2`.

b)

```
SumarValor:     li   $t0, 0
                move $t1, $a0

      bucle:    bgt  $t0, $a1, fin
                lw   $t2, ($t1)
                add  $t2, $t2, $a2
                sw   $t2, ($t1)
                addi $t0, $t0, 1
                addi $t1, $t1, 4
                b bucle
      fin:      jr $ra
```

b) El cuerpo e la función `main` es:

```
.data
        v: .word  7, 8, 3, 4, 5, 6
.text
        .globl main

 main:     sub  $sp, $sp, 24
           sw   $ra, 20($sp)
           sw   $a0, 4($sp)
           sw   $a1, 8($sp)
           sw   $a2, 12($sp)

           la   $a0, v
           li   $a1, 6
           li   $a2, 5
```

```
            jal   SumarValor


            li    $v0, 1
            li    $t0, 0
            move $t1, $a0

bucle:      bgt   $t0, $a1, fin
            lw    $a0, ($t1)
            syscall
            addi $t0, $t0, 1
            addi $t1, $t1, 4
            b bucle


      fin:  lw    $ra, 20($sp)
            lw    $a0, 4($sp)
            lw    $a1, 8($sp)
            lw    $a2, 12($sp)
            addi  $sp, $sp, 20

            li   $v0, 10
            syscall
            jr $ra
```

Solución del ejercicio 4

En la siguiente tabla se muestran las operaciones elementales y las señales de control.

Ciclo	Operación elemental	Señales de control activadas
C1	MAR ← PC	T4, C1
C2	MBR ← MP, PC ← PC + 4	L, Td, C2, C4
C3	RI ← MBR	T3, C6
C4	Decodificación	
C5	TR2 ← RI(10)	T8, C10
C6	$t0 ← $t1 + TR2	RA = <dir de $t1> MA = 0 MB = 1 Cod op = SUMAR T5 RC = <dir de $t0> SC

Solución del ejercicio 5

a) La caché tiene un tamaño de 256 KB = 2^{18} bytes. como cada línea tiene 2^6 bytes, el número de líneas es 2^{18} bytes / 2^6 bytes = 2^{12} líneas = 4096 líneas. Como la caché es asociativa por conjuntos de 4 vías, cada conjunto tiene cuatro líneas, por tanto el número de conjuntos es 4096 / 4 = 1024 conjuntos.
b) El tamaño del bloque que se transfiere entre memoria principal y caché coincide con el tamaño de la línea, es decir, con 64 bytes.

c) El tiempo medio de acceso al sistema viene dado por la siguiente expresión:

$tm = tc \cdot Pa + (1-Pa) \cdot tf$

donde tc = 5 ns, tm = 20 ns y tf = 205 (200 + 5). Por tanto:

$20 = 5 \cdot Pa + (1-Pa) \cdot 205$

Despejando, se obtiene que Pa = 185 / 200 = 0,92. Es decir, la tasa de aciertos debe ser del 92 %.

UNIVERSIDAD CARLOS III DE MADRID
DEPARTAMENTO DE INFORMÁTICA
GRADO EN INGENIERÍA INFORMÁTICA. ESTRUCTURA DE COMPUTADORES

Para la realización del presente examen se dispondrá de **2 horas. NO** se podrán utilizar libros, apuntes **ni** calculadoras de ningún tipo.

Ejercicio 1. Dado el siguiente fragmento de programa en ensamblador.

```
.text
                .globl main
 main:
                li   $a0,  5
                jal  funcion
                move $a0, $v0
                li   $v0, 1
                syscall

                li   $v0, 10
                syscall
funcion:
                move $t0, $a0
                li   $t1, 0
          bucle: beq  $t0, 0, fin
                add  $t1, $t1, $t0
                sub, $t0, $t0, 1
                b    bucle
           fin:  move $v0, $t1
                jr $ra
```

Se pide :

b) Indicar de forma razonada el valor que se imprime por pantalla (primera llamada al sistema del código anterior).

c) Si en el registro `$a0`, que se utiliza para el paso de parámetros a la función, el valor que se almacena se representa en complemento a uno, ¿qué rango de números podrían pasarse a la función?

Ejercicio 2. Considere una función denominada `Vocales`. A esta función se le pasa como parámetro la dirección de inicio de una cadena de caracteres. La función calcula el número de veces que aparece el carácter 'a' (en minúscula) en la cadena. En caso de pasar la cadena nula la función devuelve el valor -1. En caso de que la cadena no tenga ninguna 'a', la función devuelve 0. Se pide:

d) Programar utilizando el ensamblador del MIPS32 el código de la función `Vocales`.

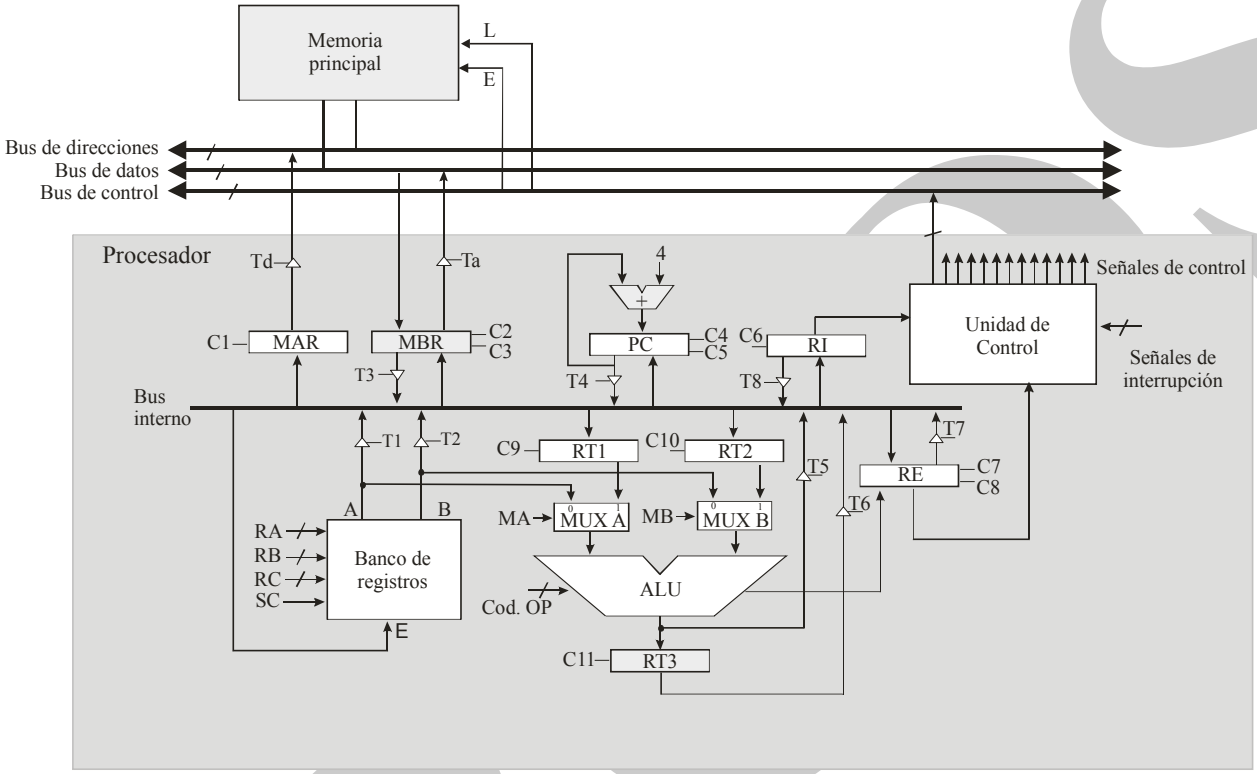
e) Indique en qué registro se ha de pasar el argumento a la función y en qué registro se debe recoger el resultado.

f) Dado el siguiente fragmento de programa:

```
.data
        cadena:  .asciiz    "Hola"
.text
        .globl main

 main:
```

incluya en el `main` anterior, las sentencias en ensamblador necesarias para poder invocar a la función `Vocales` implementada en el apartado a) e imprimir por pantalla el valor que devuelve la función. El objetivo es imprimir el número de veces que aparece el carácter 'a' en la cadena "Hola".

Ejercicio 3. Considere el siguiente esquema de procesador de 32 bits. El banco de registros incluye 32 registros. Considere que el computador utiliza un ciclo de reloj para realizar la decodificación de la instrucción y que se conecta a una memoria que permite realizar una operación de lectura y de escritura en un ciclo.



Indique las operaciones elementales y las señales de control (incluyendo el *fetch*) necesarias para ejecutar la instrucción `lw $t1, ($t2).`

Ejercicio 4. Sea un computador de 32 bits con una memoria caché para datos de 32 KB y líneas de 64 bytes. La caché es asociativa por conjuntos de 2 vías y se emplea la política de reemplazo LRU. Dado el siguiente fragmento de código:

```
int v[262144]

for (i = 0; i < 262144; i = i + 2)
    v[i] = 9;
```

se pide:

d) Indique el número de líneas y de conjuntos de la memoria caché de datos del enunciado.
e) Considerando exclusivamente la caché de datos y los accesos al vector, calcule de forma razonada la tasa de fallos que se obtiene en la ejecución del bucle anterior.

Ejercicio 5. Sea un computador de 20 bits con memoria virtual paginada con páginas de 1 KB y un total de memoria física de 256 KB. Se pide, de forma razonada y breve:

a) ¿Cuál es el formato de la dirección virtual? Indique los campos y el número de bits de los mismos.

b) ¿Cuál es el número máximo de entradas de la tabla de páginas (de un nivel)?
c) ¿Cuántos marcos de página tiene la memoria principal?
d) ¿Cuáles son los campos que se incluyen en una entrada de la tabla de páginas? Indique también para qué se utiliza cada uno de los campos.

Soluciones:

Solución del ejercicio 1

a) La función realiza la suma de los números 5, 4, 3, 2,1 y devuelve el resultado en el registro $v0, cuyo valor por tanto es 15, que es el resultado que se imprime por pantalla.
b) Para una palabra de n bits, el rango de representación de números en complemento a 1 es $[-2^{n-1}+1, 2^{n-1}-1]$. En el caso del MIPS 32, los registros son de 32 bits, por tanto n = 32 y el rango de representación sería $[-2^{31}+1, 2^{31}-1]$.

Solución del ejercicio 2

a) Se asume que la dirección de la cadena se pasa en el registro $a0 y el resultado se devuelve en el registro $v0. El código de la función vocales es el siguiente

```
vocales:   li    $t0, -1      // contador del número de a
           move  $t1, $a0
           beqz  $t1, fin
           li    $t0, 0
           li    $t2, 'a'
   bucle:  lbu   $t3, ($t1)
           beqz  $t3, fin
           bneq  $t3, $t2, noA
           addi  $t0, $t0, 1
      noA: addi  $t1, $t1, 1
           b bucle

      fin: move  $v0, $t0
           jr    $ra
```

c) Los argumentos se pasan en los registros `$aX` y los resultados en los registros `$vX`. En este caso la dirección de inicio de la cadena se pasa en `$a0` y el resultado se recoge en `$v0`.
d) El cuerpo e la función `main` es:

```
.data
       cadena:  .asciiz    "Hola"
.text
       .globl main

 main:     sub   $sp, $sp, 24
           sw    $ra, 20($sp)
           sw    $a0, 4($sp)

           la    $a0, cadena
           jal   vocales

           move  $a0, $v0
           li    $v0, 1
           syscall

           lw    $ra, 20($sp)
           lw    $a0, 4($sp)
           addi  $sp, $sp, 24

           li   $v0, 10
           syscall
```

```
jr $ra
```

Solución del ejercicio 3

En la siguiente tabla se muestran las operaciones elementales y las señales de control.

Ciclo	Operación elemental	Señales de control activadas
C1	MAR ← PC	T4, C1
C2	MBR ← MP, PC ← PC + 4	L, Td, C2, C4
C3	RI ← MBR	T3, C6
C4	Decodificación	
C5	MAR ← $t2,	RA = <dir de $t2> T1, C1
C6	MBR ← MP	Td, L, C2
C7	$t1 ← MBR	T3, SC, RA = <dir de $t1>

Solución del ejercicio 4

d) La caché tiene un tamaño de 32 KB = 2^{15} bytes. como cada línea tiene 2^6 bytes, el número de líneas es 2^{15} bytes / 2^6 bytes = 2^9 líneas = 512 líneas. Como la caché es asociativa por conjuntos de 2 vías, cada conjunto tiene dos líneas, por tanto el número de conjuntos es 512 / 2 = 256 conjuntos.

e) El patrón de accesos del vector al bucle es el siguiente:

```
v[0], v[2], v[4], v[6], v[8] …
```

es decir, se accede cada 2 elementos. Como la caché está estructurada en líneas de 64 bytes y cada dato de tipo entero (`int`) ocupa 4 bytes, en cada línea caben 16 elementos del vector. Como el vector se recorre de forma secuencial y de estos 16 solo se acceden realmente a 8, la tasa de fallos es 1/8.

Solución del ejercicio 5

a) Las páginas ocupan 1 KB = 2^{10} bytes. Como la dirección virtual ocupa 20 bits, se emplean 20 - 10 = 10 bits para el número de página. Por tanto, el formato emplea los 10 bits superiores de la dirección para representar el número de página y los 10 bits inferiores para representar el desplazamiento dentro de la página.
b) El número máximo de entradas de la tabla de páginas coincide con el número máximo de páginas, es decir 2^{10} = 1024 entradas.
c) El número de marcos de página viene dado por 256 KB / 1 KB = 256 marcos.
d) En cada entrada de la tabla de página se incluye, entre otros:
    a. Bit de presencia
    b. Bit de modificado
    c. Bit de validez
    d. Bits de permisos
    e. Campo en el que se almacena el marco.

UNIVERSIDAD CARLOS III DE MADRID
DEPARTAMENTO DE INFORMÁTICA
GRADO EN INGENIERÍA INFORMÁTICA. ESTRUCTURA DE COMPUTADORES

Para la realización del presente examen se dispondrá de **1 hora. NO** se podrán utilizar libros, apuntes **ni** calculadoras de ningún tipo. **Responda a los ejercicios 1, 2 y 3 en el espacio reservado**

Alumno:_____.
Grupo: _____ .

Ejercicio 1. Indique la representación de los siguientes números:

a) -64 en complemento a uno con 7 bits

b) -64 en complemento a dos con 7 bits

c) 12 en signo magnitud con 6 bits

d) 18 en complemento a dos con 5 bits

Ejercicio 2. Indique el valor decimal del siguiente número hexadecimal 0x00600000 que representa un número en coma flotante según IEEE 754 (precisión simple)

Ejercicio 3. Indique una instrucción del MIPS que incluya el modo de direccionamiento relativo a registro base. ¿En qué consiste este direccionamiento?

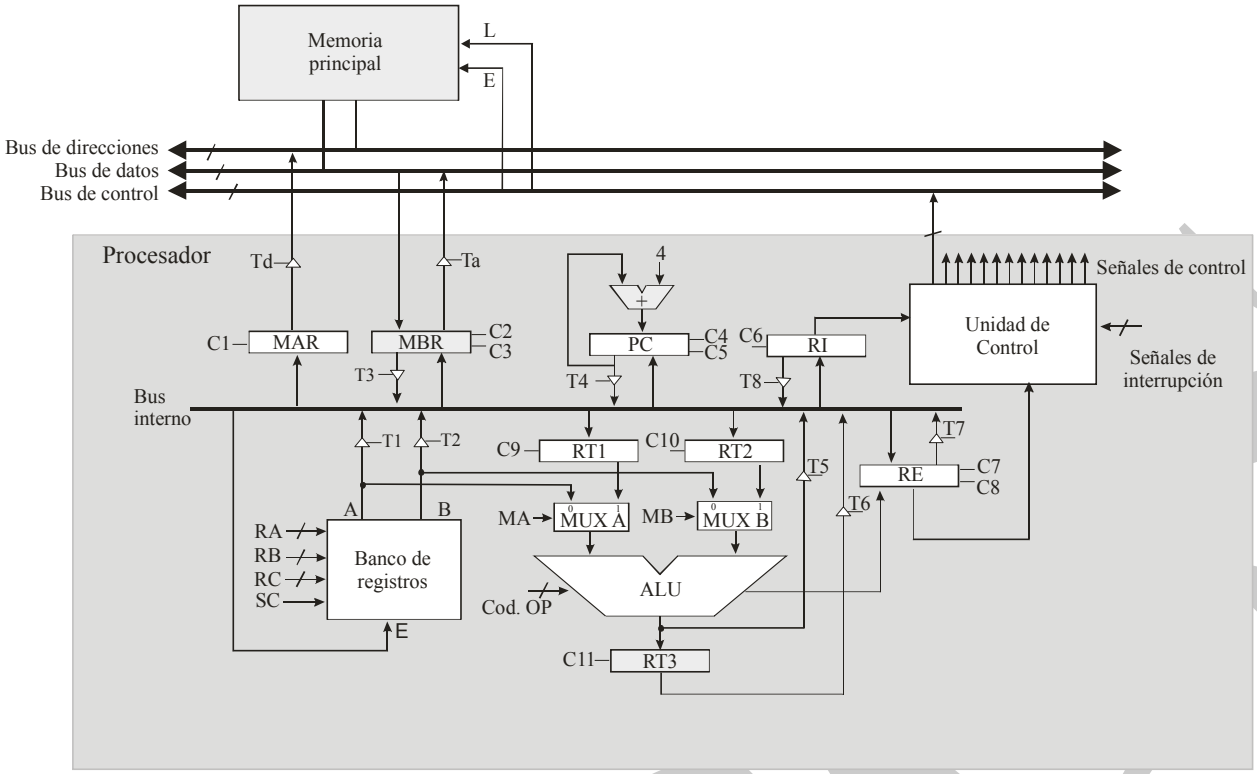
Ejercicio 4. Sea un computador de 32 bits con 48 registros y 200 instrucciones máquina. Indique el formato de la instrucción hipotética `beqz $t1, $t2, dirección` donde `$t1` y `$t2` son registros y `dirección` representa una dirección de memoria.

Ejercicio 5. Considere una función denominada `func` que recibe tres parámetros de tipo entero y devuelve un resultado de tipo entero, y considere el siguiente fragmento del segmento de datos:

```
.data
     a: .word 5
     b: .word 7
     c: .word 9

.text
```

Indique el código necesario para poder llamar a la función anterior pasando como parámetros los valores de las posiciones de memoria a, b y c. Una vez llamada a la función deberá imprimirse el valor que devuelve la función.

Ejercicio 6. Considere el siguiente esquema de procesador de 32 bits. El banco de registros incluye 32 registros. Considere que el computador utiliza un ciclo de reloj para realizar la decodificación de la instrucción y que se conecta a una memoria que permite realizar una operación de lectura y de escritura en un ciclo.



Este computador dispone del juego de instrucciones del MIPS32. Se pide:

a) Indique las señales de control necesarias para poder realizar la operación elemental PC ← R7, siendo R7 el registro del banco de registros cuyo número es el 7.

b) Si durante un ciclo de reloj se activan las señales T3 y C10, indique qué operación elemental se está realizando.

c) Indique las operaciones elementales y las señales de control (incluyendo el *fetch*) necesarias para ejecutar la instrucción `addi $t0, $t1, 10`.

Ejercicio 7. Dada la siguiente definición de un vector de enteros:

```
array: .word  10, 20, 30, 40, 6, 5, 9, 8, 10, 35, 45, 56, 7
```

Escriba un fragmento de programa que permita imprimir el valor de todos los números del array que sean menores de 30.

Soluciones:

Solución del ejercicio 1

a) -64 en complemento a uno con 7 bits
El rango de representación es [-63, 63], por tanto el número no es representable

b) -64 en complemento a dos con 7 bits
64 en binario con 7 bits es 1000000. Se complementa 0111111 y se suma 1, siendo el resultado 100000

c) 12 en signo magnitud con 6 bits
001100

d) 18 en complemento a dos con 5 bits
El rango de representación es [-16, 15], por tanto el número no es representable.

Solución del ejercicio 2

0x00600000 -> 00000000011000000000000000000000

Signo = 0, número positivo
Exponente = 00000000
Mantisa = 1100000…0000

Se trata de un número no normalizado cuyo valor es $0{,}11 \times 2^{-126} = 0{,}75 \times 2^{-126}$

Solución del ejercicio 3

lw $t1, 20($t2)

El campo con este modo de direccionamiento es 20($t2), que representa la dirección de memoria que se obtiene de sumar 20 con la dirección almacenada en el registro $t2.

Solución del ejercicio 4

El número de bits necesarios para codificar 48 registros es de 6 bits. Para codificar 200 instrucciones se necesitan 8 bits. Para representar una dirección en un computador de 32 bits se necesitan 32 bits. Por tanto son necesarias dos palabras:

Solución del ejercicio 5

```
lw   $a0, a
```

```
lw   $a1, b
lw   $a2, c
jal  func
move $a0, $v0
li   $a0, 1
syscall
```

Solución del ejercicio 6

a) Han de activarse las señales C5, T1 y RA = 00111 (registro 7)
b) Se trata de la operación RT2 ← MBR
c) Ciclo 0: MAR ← PC — Señales: T4, C3
   Ciclo 1: MBR ← MP — Señales: Td, L, C2
   PC ← PC + 4 — C4
   Ciclo 2: RI ← MBR — Señales: T3, C6
   Ciclo 3: Decodificación
   Ciclo 4: RT1 ← RI(10) — Señales: T8, C9
   Ciclo 5: $t0 ← RT1 + $t1 — Señales: MA = 1, Cod. OP = suma, RB = <$t1>, T5, RC = <$t0>, SC, C8

Solución del ejercicio 7

```
          la   $t0, array
          li   $t1, 0              ; índice
          li   $t2, 13             ; número de elementos del array
          li   $t3, 30             ; valor de referencia
          li   $v0, 1
bucle:    bge  $t1,  $t2, fin
          lw   $t4, ($t0)
          bge  $t4, $t3, mayor     ; si es mayor no se imprime
          move $a0, $t4       ; se imprime
          syscall
mayor:    addi $t0, $t0, 4
          addi $t1, $t1, 1
          b    bucle
fin:
```

UNIVERSIDAD CARLOS III DE MADRID
DEPARTAMENTO DE INFORMÁTICA
GRADO EN INGENIERÍA INFORMÁTICA. ESTRUCTURA DE COMPUTADORES

Para la realización del presente examen se dispondrá de **1:30 horas.**
NO se podrán utilizar libros, apuntes **ni** calculadoras de ningún tipo.

Ejercicio 1. Se desea desarrollar un controlador para un horno microondas. El controlador dispone de un procesador de 32 bits, mapa de E/S común y juego de instrucciones del MIPS 32. A este procesador se le conectan un módulo de E/S, que controla el funcionamiento del horno. El módulo dispone de los cinco siguientes registros de 32 bits:

- Registro con dirección 1000. En este registro se carga el valor correspondiente a la cuenta atrás en segundos.
- Registro con dirección 1004. En este registro se carga un 1 cuando se quiere comenzar la cuenta atrás. Mientras haya un 0 no se inicia la cuenta atrás.
- Registro con dirección 1008. Cuando la cuenta atrás llega a 0, en este registro el controlador almacena un 1. Mientras se está realizando la cuenta atrás el valor que se encuentra almacenado en este registro es 0.
- Registro con dirección 1012. En este registro se carga el valor correspondiente a la potencia deseada. El valor es un entero de 32 bits en complemento a dos.
- Registro con dirección 1016. Cuando se carga en este registro el valor 1, el microondas se pone en funcionamiento con la potencia almacenada en el registro anterior. Cuando en este registro se carga un 0, el microondas se para independientemente de que se haya llegado al final de la cuenta atrás.

Se desea escribir una función en ensamblador, denominada `Controlador_Horno` que permita controlar el funcionamiento del horno. Esta función recibirá dos parámetros de entrada: un valor entero indicando el número de segundos que ha de permanecer encendido el horno y un valor entero que representa la potencia del horno. La función se encargará de programar el controlador anterior con la potencia y segundos pasados como parámetros, encargándose de poner en funcionamiento el horno y apagarlo cuando haya transcurrido el tiempo necesario. La función devolverá los siguientes posibles valores:

- Un valor igual a 0 que representa que no ha habido ningún error y que se devolverá cuando la función haya parado el horno.
- Un valor igual a -1, cuando la potencia pasada como parámetro sea menor de 100 y mayor de 1000. En este caso no se ha de programar el controlador del horno y la función debe retornar inmediatamente el valor -1.

Se pide:

a) Indique en qué registros se han de pasar los parámetros y en qué registro se debe devolver el resultado.
b) Invoque a la función anterior para un valor de potencia de 800 W y una duración de 90 segundos. Escriba a continuación el código que permita imprimir por pantalla el resultado que devuelve la función.
c) Escriba el código correspondiente a la función `Controlador_Horno`.

Ejercicio 2. Se desea representar números enteros dentro del rango -8191…8191. Indicar de forma razonada:

a) ¿Cuál es el número de bits que se necesita si se quiere utilizar una representación en complemento a uno?
b) ¿Cuál es el número de bits que se necesita si se quiere utilizar una representación en signo-magnitud?

Ejercicio 3. Represente el número -24,50 utilizando el estándar de coma flotante de simple precisión IEEE 754. Exprese dicha representación en binario y en hexadecimal.

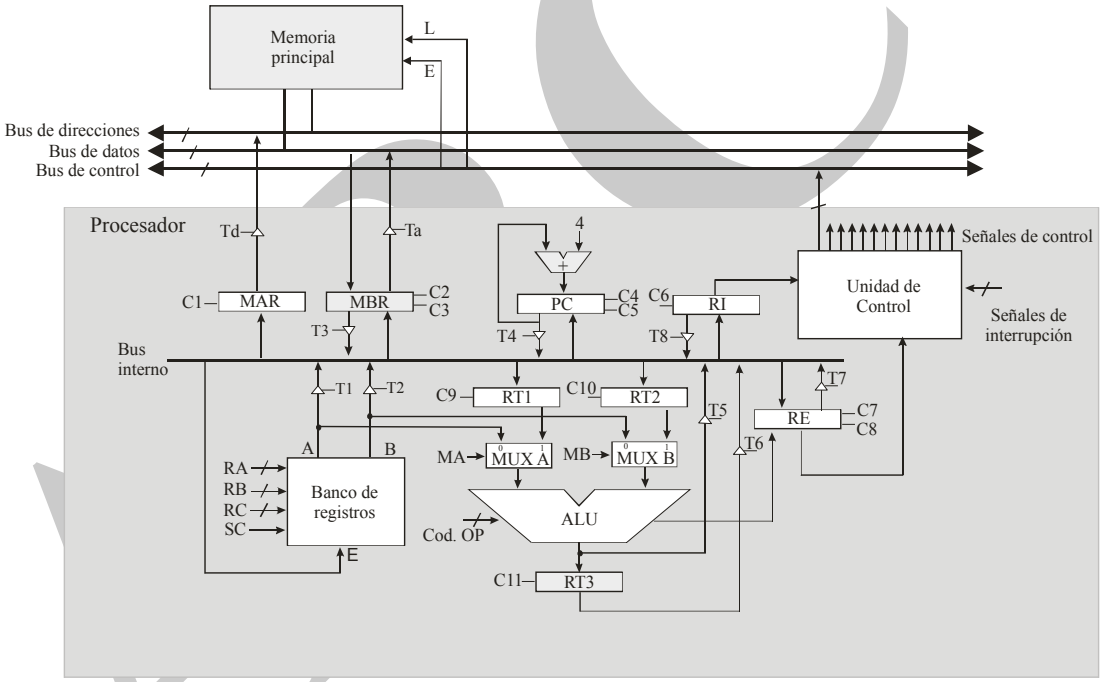
Ejercicio 4. Sea un computador de 32 bits que dispone de una memoria caché de 512 KB, asociativa por conjuntos de 4 vías y líneas de 64 bits. Sobre este computador se desea ejecutar el siguiente fragmento de código:

```
int v[200];
int i;
int s;

for (i=0; i < 200; i++)
     s = s + v[i];
```

Se pide:

a) Indique de forma razonada el número de líneas y conjuntos de de la caché.
b) Si se considera que la caché se encuentra vacía y que los valores de las variables `i` y `s` del código anterior se almacenan en registros, indique, considerando solo los accesos al vector `v`, la tasa de aciertos que se obtiene al ejecutar el fragmento de código anterior.

Ejercicio 5. Considere el siguiente esquema de procesador de 32 bits. El banco de registros incluye 32 registros. Considere que el computador utiliza un ciclo de reloj para realizar la decodificación de la instrucción y que se conecta a una memoria que permite realizar una operación de lectura y de escritura en un ciclo.



Este computador dispone del juego de instrucciones del MIPS32. Se pide:

d) Indique las señales de control necesarias para poder realizar la operación elemental RT2 ← PC.
e) Indique las operaciones elementales y las señales de control (incluyendo el *fetch*) necesarias para ejecutar la instrucción `lw $t1, ($t2).` Explique en primer lugar qué hace esta instrucción.

Soluciones

Ejercicio 1.

a) Los parámetros de entrada se pasan en los registro $a0 y $a1 y el resultado se devuelve en $v0.

b)

```
li   $a0,  800
li   $a1, 90
jal  Controlador_Horno
move $a0, $v0
li    $a1, 1
syscall
```

c)

```
controlador_Horno:
                    li   $t0, 100
                    blt  $a0, $t0, fin_error
                    li   $t0, 1000
                    bgt  $a0, $t0, fin_error
                    sw   $a1, 1000      ; cuenta atrás
                    sw   $a0, 1012      ; potencia
                    li   $t0, 1
                    sw   $t0, 1016      ; arranque motor
                    sw   $t0, 1004      ; iniciar cuenta atrás
          bucle:    lw   $t1, 1008      ; leer estado cuenta
                    beq  $t1, $0, bucle
                    sw   $0, 1016       ; parar el motor
                    move $v0, $0
                    jr $ra
      fin_error:    li $v0, -1
                    jr $ra
```

Ejercicio 2

Se necesitan en los dos casos 14 bits. Con 14 bits el rango de representación en ambos casos es de $-(2^{13}-1) \ldots 2^{13}-1 = -8191 \ldots 8191$

Ejercicio 3

$24{,}5_{(10} = 11000.1_{(2} = 1{,}10001 \times 2^4$

Signo = 1, número negativo
Exponente = 4 + 127 = 131 = 10000011
Mantisa = 1000100000 .. 00000

En binario => 11000000111000100000 …….00000
En Hexadecimal => 0xC1C40000

Ejercicio 4

a) La memoria caché tiene un tamaño de 512 Kb = 2^{19} bytes. Cada línea tiene 64 bits = 8 bytes = 2^3 bytes. El número de línea viene dado por 2^{19} bytes / 2^3 bytes = 2^{16} líneas. Como cada conjunto tiene 4 líneas, el número de conjuntos = 2^{16} / 2^2 = 2^{14} conjuntos

b) Como cada elemento del vector es un entero de 4 bytes, en cada línea caben dos enteros. Cada vez que se accede a un elemento se produce un fallo, y se traen a caché el elemento que ha producido el fallo y el siguiente, que da lugar a un acierto en la siguiente vuelta del bucle. Por tanto, hay un fallo cada dos accesos y la tasa de aciertos es del 50 %.

Ejercicio 5.

a) Hay que activar las señales T4 y C10

b)

Ciclo 0: MAR ← PC	Señales: T4, C3
Ciclo 1: MBR ← MP	Señales: Td, L, C2
PC ← PC + 4	C4
Ciclo 2: RI ← MBR	Señales: T3, C6
Ciclo 3: Decodificación	
Ciclo 4: MAR ← $t2	Señales: RA= identificador de $t2, T1, C1
Ciclo 5: MBR ← MP	Señales: Td, L, C2
Ciclo 6: $t1 ← MBR	T3, RC = identificador de $t1, SC

UNIVERSIDAD CARLOS III DE MADRID
DEPARTAMENTO DE INFORMÁTICA
GRADO EN INGENIERÍA INFORMÁTICA. ESTRUCTURA DE COMPUTADORES

Para la realización del presente examen se dispondrá de **1:30 horas.**
NO se podrán utilizar libros, apuntes **ni** calculadoras de ningún tipo.

Ejercicio 1. Considere el siguiente fragmento en ensamblador:

```
.data
     A1: .word 5, 8, 7, 9, 2, 4, 5, 9
     A2: .word 1, 4, 3, -8, 5, 6, 5, 9
     .align 2
     A3: .space 32

.text
```

Se pide:

a) ¿Qué representa A1? ¿Cuántos bytes ocupa la estructura de datos A1 (justifique su respuesta)?
b) Se desea implementar una función cuyo prototipo en un lenguaje de alto nivel es la siguiente:

   ```
   void Mezclar(int  a[], int b[], int c[], int N)
   ```

   Esta función recibe 4 parámetros, los tres primeros son vectores de números enteros y el cuarto indica el número de componentes de cada uno de estos vectores. La función se encarga de almacenar en cada componente `i` de `c`, el siguiente valor: `c[i] = max (a[i], b[i])`.

   Escriba, utilizando el ensamblador del MIPS32, el código correspondiente a esta función. Utilice para ello la convención de paso de parámetros que se ha descrito a lo largo del curso.

c) Escriba el código necesario para llamar a la función desarrollada en el apartado anterior, para los vectores A1, A2 y A3 definidos en la sección de datos anterior. Asuma que A3 es el vector donde se deben dejar los elementos máximos.

Ejercicio 2. ¿Cuál es el número positivo normalizado más pequeño que se puede representar utilizando el estándar de simple precisión IEEE 754. Justifique su respuesta. Indique también el número positivo no normalizado más pequeño que se puede representar. Justifique de igual forma su respuesta.

Ejercicio 3. Considere un computador de 32 bits que dispone de un sistema de memoria virtual que emplea páginas de 16 KB y tiene instalada una memoria principal de 1 GB. Indique de forma razonada:

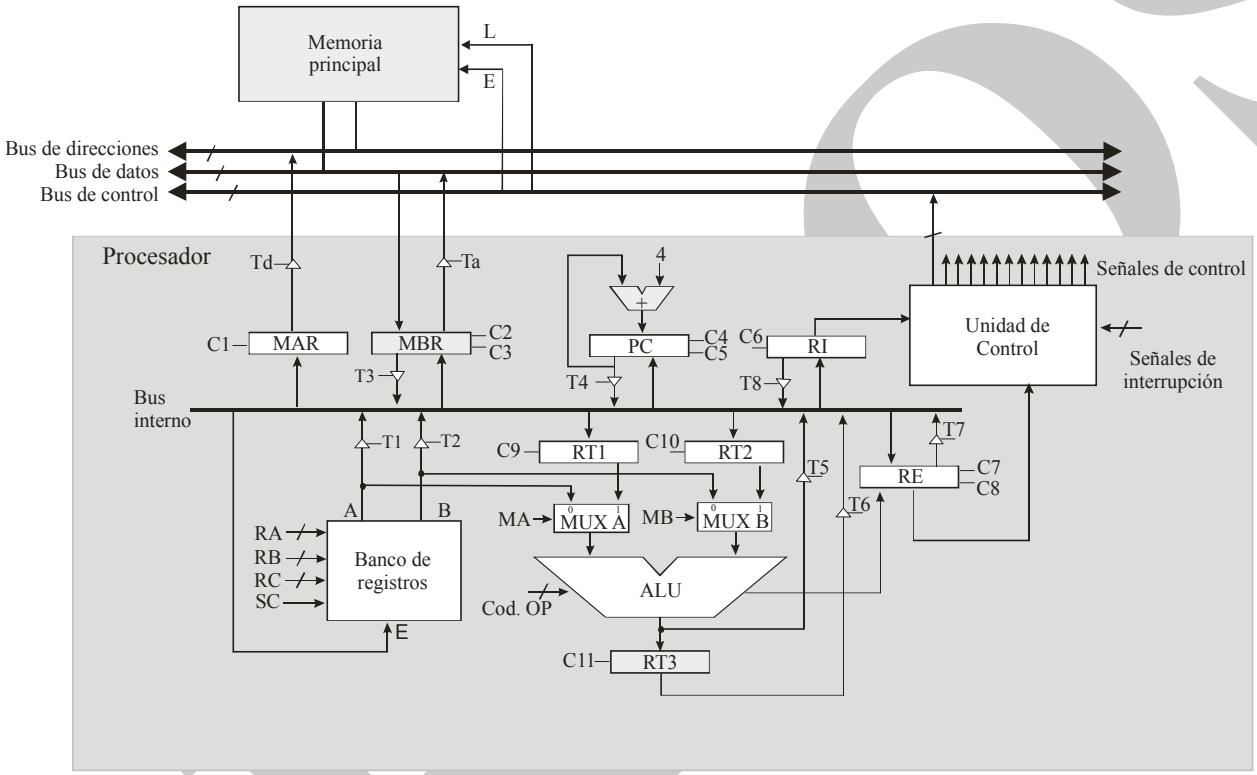
a) El formato de la dirección virtual.
b) El número máximo de páginas en este computador.
c) El número de marcos de página de este computador.
d) El tamaño del bloque que se transfiere entre disco y memoria principal cuando ocurre un fallo de página
e) El elemento del computador que genera el fallo de página y quién lo trata.

Ejercicio 4. ¿Qué es una interrupción y que sucede cuando se genera?

Ejercicio 5. Se dispone de un sistema con una memoria caché de 2 niveles. En la ejecución de una determinada aplicación, la tasa de aciertos de la caché de nivel 1 es del 90% y la tasa de aciertos de la caché de nivel 2 es del 95%. La aplicación genera durante su ejecución un millón de accesos a memoria. Indique de forma razonada:

a) El número de accesos que se genera a la caché de nivel 1.
b) El número de accesos que se genera a la caché de nivel 2.
c) El número de accesos que se genera a memoria principal.

Ejercicio 6. Considere el siguiente esquema de procesador de 32 bits. El banco de registros incluye 32 registros. Considere que el computador utiliza un ciclo de reloj para realizar la decodificación de la instrucción y que se conecta a una memoria que permite realizar una operación de lectura y de escritura en un ciclo.



Este computador dispone del juego de instrucciones del MIPS32. Se pide:

f) Indique las señales de control necesarias para poder realizar la operación elemental RT2 ← R2, donde R2 es el registro número 2 del banco de registros.
g) Indique las operaciones elementales y las señales de control (incluyendo el *fetch*) necesarias para ejecutar la instrucción `sw $t1, 120($t2)`. Explique en primer lugar qué hace esta instrucción.

Soluciones

Ejercicio 1

a) A1 representa un vector de ocho números enteros
Si son 8 enteros, y siendo una máquina de 32 bits cada entero se representa 4 bytes, supone un total de 8*4 = 32 bytes.

b) Y c)

```
.data

 v1: .word 1, 2, 3, 4, 5
 v2: .word 5, 4, 3, 2, 1
 v3: .word 0, 0, 0, 0, 0



.text
.globl main

maximo:
          move $t0 $a0
          li   $t4 0

          move $t1 $a1
          move $t2 $a2
          move $t3 $a3

 bucle1: bge $t4 $t0 fin1
          lw $t5 ($t1)
          lw $t6 ($t2)

          bgt $t5 $t6 es5
          sw $t6 ($t3)
          b next1
    es5: sw $t5 ($t3)

  next1: add $t1 $t1 4
          add $t2 $t2 4
          add $t3 $t3 4
          add $t4 $t4 1

          b bucle1

   fin1: jr $ra
```

```
main:
        li $a0 5

        la $a1 v1
        la $a2 v2
        la $a3 v3

        jal maximo


        li $t0 0
bucle2: bge $t0 5 fin2

        lw $a0 ($a3)
        li $v0 1
        syscall

        add $a3 $a3 4
        add $t0 $t0 1
        b bucle2

fin2:   li $v0 10
        syscall
```

Ejercicio 2

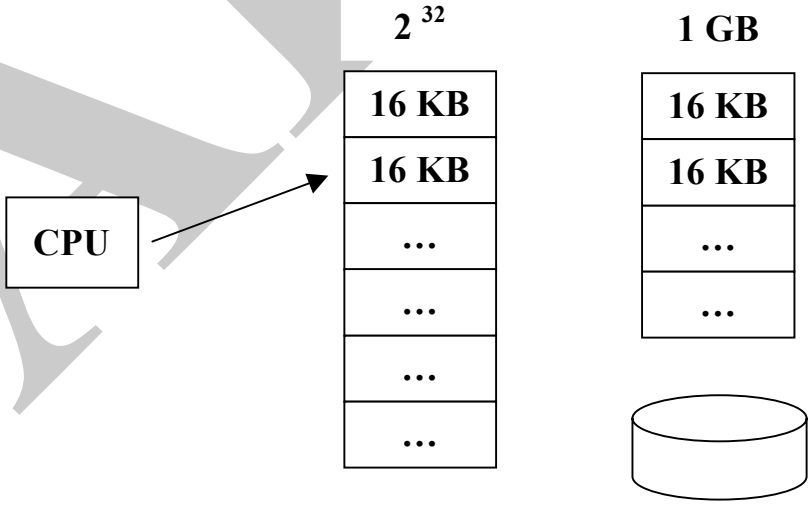
a) El número positivo normalizado más pequeño representable en el estándar IEEE 754 (32 bits) es:
0 00000001 00000000000000000000000
positivo normalizado más pequeño

Cuyo valor es $1.0 * 2^{1-127} = \mathbf{2^{-126}}$

b) El número positivo no normalizado más pequeño representable en el estándar es:
0 00000000 00000000000000000000001
positivo no normalizado más pequeño

Cuyo valor es $2^{-23} * 2^{-126} = \mathbf{2^{-149}}$

Ejercicio 3



a)

Id. página	Desplazamiento
32 – 14	**2^{14} = 16 KB**
18 bits	**14 bits**

b) $2^{32} / 2^{14} = 2^{32-14} =$ **2^{18} marcos**

c) $2^{30} / 2^{14} = 2^{30-14} =$ **2^{16} páginas**

d) El tamaño de una página es **16 KB**

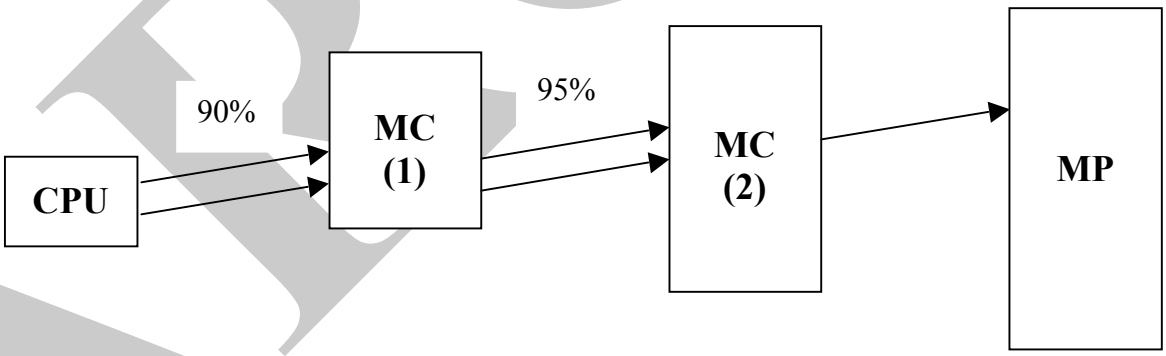
e) La **MMU genera la excepción** y
la rutina de fallo de página del sistema operativo **trata la excepción**

Ejercicio 4

a) Una interrupción es una excepción asíncrona.

Se guarda el estado y se ejecuta la rutina de interrupción asociada.

Ejercicio 5



a) El número de accesos con acierto en MC(1) es: **10^6**
b) El número de accesos con acierto en MC(2) es: **10^6 * 0,1**
c) El número de accesos con acceso a MP por fallos de MC1 y MC2: **10^6 * 0,1 * 0,05**

Ejercicio 6

a) RA=R2, T1, C10

b) Las O.E. y las S.C. asociadas a ***sw $t1 120($t2)*** podrían ser:

MAR <- PC MBR <- M[MAR] RI <- MBR, PC <- PC + 4 Decodificar	T4, C1 Td, L, C2 T3, C6, C4 Decodificar
RT2 <- RI(120) RT1 <- BR[$t2] MAR <- RT1 + RT2 MBR <- BR[$t1] M[MAR] <- MBR Salto a fetch	T8, C10 RA=$t2, T1, C9 MA=1, MB=1, Cod. OP=+, T5, C1 RA=$t1, C3 Ta, Td, E Salto a fetch

UNIVERSIDAD CARLOS III DE MADRID. DEPARTAMENTO DE INFORMÁTICA
GRADO EN INGENIERÍA INFORMÁTICA. ESTRUCTURA DE COMPUTADORES

Para la realización del presente examen se dispondrá de **2 horas.** **NO** se podrán utilizar libros, apuntes **ni** calculadoras de ningún tipo.

Ejercicio 1 (2 puntos). Represente el número `-27,25` utilizando el estándar de coma flotante de simple precisión IEEE 754. Exprese dicha representación en binario y en hexadecimal.

Ejercicio 2 (2 puntos). Dado el siguiente fragmento de código escrito en C, escriba utilizando el ensamblador del MIPS 32 el código de la función equivalente.

```
int máximo(intA, int B)
{
      if (A > B)
            return A;
      else
            return B;

}
```

Utilizando la función en ensamblador anterior implemente el código de la siguiente función utilizando el ensamblador del MIPS 32.

```
void maximoV (int v1, int v2, int v3, int N)
{
      int i;

      for (i = 0; i < N; i++)
            v3[i] = maximo(v1[i], v2[i]);
      return;
}
```

Para el desarrollo de este ejercicio ha de seguirse la convención de paso de parámetros vista en el temario de la asignatura.

Ejercicio 3 (2 puntos). Se desea desarrollar un controlador para un dispositivo que permite medir la tensión arterial de una persona. El módulo de E/S del medidor de tensión se conecta a un computador que dispone de un procesador de 32 bits, mapa de E/S común y juego de instrucciones del MIPS 32. El módulo dispone de los siguientes registros de 32 bits:

- Registro con dirección 1000. Cuando se carga en este registro el valor 1, se le indica al módulo que se quiere comenzar una medida de la tensión.
- Registro con dirección 1004. Cuando la medición ha acabado en este registro se almacena un 1, mientras el dispositivo se encuentra realizando la medición, su valor es 0.
- Registro con dirección 1008. Cuando el dispositivo finaliza la medición de la tensión, en este registro se almacena el valor correspondiente a la tensión sistólica.
- Registro con dirección 1012. Cuando el dispositivo finaliza la medición de la tensión, en este registro se almacena el valor correspondiente a la tensión diastólica.

Escriba una función en ensamblador, denominada `Controlador_tension` que permita controlar el funcionamiento de este equipo. Esta función no recibirá ningún parámetro de entrada. La función se encargará de realizar una medición de la tensión y devolverá como parámetros de salida el valor de la tensión sistólica y el valor de la tensión diastólica.

Para el desarrollo de este ejercicio ha de seguirse la convención de paso de parámetros vista en el temario de la asignatura.

Ejercicio 4 (2 puntos). Sea un computador de 32 bits que dispone de una memoria caché de 512 KB, asociativa por conjuntos de 4 vías y líneas de 128 bytes. Sobre este computador se desea ejecutar el siguiente fragmento de código:

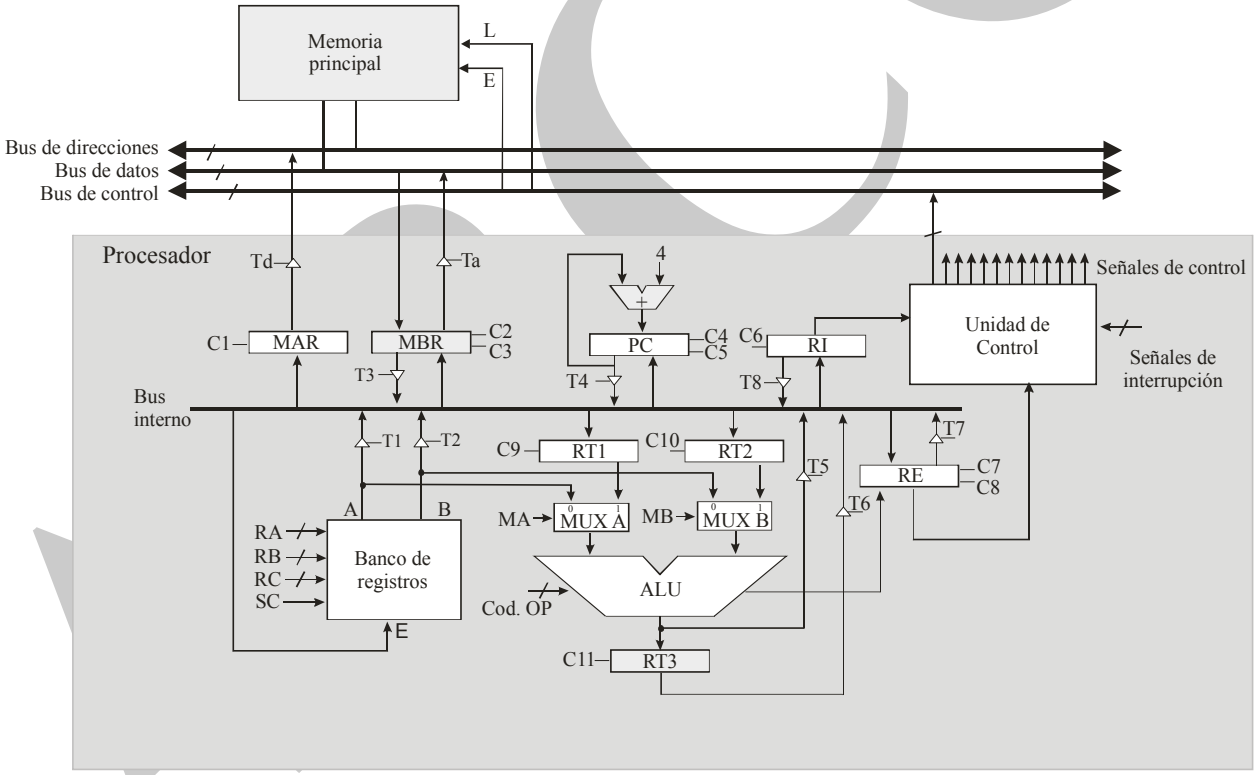
```
int a1[200];
int a2[200]:
int i;
int s;

for (i=0; i < 200; i++)
     s = s + a1[i] + a2[i];
```

Se pide:

c) Indique de forma razonada el número de líneas y conjuntos de de la caché.

d) Si se considera que la caché se encuentra vacía y que los valores de las variables `i` y `s` del código anterior se almacenan en registros, indique, considerando solo los accesos al vector `a1 y a2`, la tasa de aciertos que se obtiene al ejecutar el fragmento de código anterior.

Ejercicio 5 (2 puntos). Considere el siguiente esquema de procesador de 32 bits. El banco de registros incluye 32 registros. Considere que el computador utiliza un ciclo de reloj para realizar la decodificación de la instrucción y que se conecta a una memoria que permite realizar una operación de lectura y de escritura en un ciclo.



Este computador dispone del juego de instrucciones del MIPS32. Se pide:

h) Indique las señales de control necesarias para poder realizar la operación elemental RT1 ← PC.

i) Indique las operaciones elementales y las señales de control (incluyendo el *fetch*) necesarias para ejecutar la instrucción `sw $t1, ($t2).` Explique en primer lugar qué hace esta instrucción.

Soluciones

Ejercicio 1.

$-27{,}25_{(10} = 11011{,}01_{(2} = 1{,}101101 \times 2^4$

Signo = 1, número negativo
Exponente = 4 + 127 = 131 = 10000011
Mantisa = 1011010000…..0000000000

En binario => 11000001110110100000 0000
En binario => 1100 0001 1101 1010 0000 0000 0000 0000
En Hexadecimal => 0xC1DA0000

Ejercicio 2

```
maximo: bgt   $a0, $a1, then
        move  $v0, $a1
        jr    $ra

  then: move  $v0, $a0
        jr    $ra


maximoV:subu  $sp, sp, 20
        sw    $a0, ($sp)
        sw    $a1, 4($sp)
        sw    $a2, 8($sp)
        sw    $a3, 12($sp)
        sw    $ra, 16($sp)

        move  $t0, $a0
        move  $t1, $a1
        move  $t2, $a2
        li    $t3, 0
        move  $t4, $a3

 bucle: bge   $t3, $t4, finBucle
        lw    $a0, ($t0)
        lw    $a1, ($t1)
        jal   maximo
        sw    $v0, (t2)

        addi  $t0, $t0, 4
        addi  $t1, $t1, 4
        addi  $t2, $t2, 4
        addi  $t3, $t3, 1
        b     bucle

finBucle: lw  $ra, 16($sp)
        lw    $a3, 12($sp)
        lw    $a2, 8($sp)
        lw    $a1, 4($sp)
        lw    $a0, ($sp)
        add   $sp, $sp, 20
```

```
                jr    $ra
```

Ejercicio 3

```
ControladorTension:   li    $t0, 1
                      sw    $t0, 1000

            Bucle:    lw    $t0, 1004
                      beqz  $t0, bucle

                      lw    $v0, 1008
                      lw    $v1, 1012

                      jr    $ra
```

Ejercicio 4

b) La memoria caché tiene un tamaño de 512 Kb = 2^{19} bytes. Cada línea tiene 128 bytes = 2^7 bytes. El número de líneas viene dado por 2^{19} bytes / 2^7 bytes = 2^{12} líneas = 4096 líneas. Como cada conjunto tiene 4 líneas, el número de conjuntos = 2^{12} / 2^2 = 2^{10} conjuntos = 1024 conjuntos

c) Como cada elemento del vector es un entero de 4 bytes, en cada línea caben 128 / 4 = 32 enteros. Cuando se accede por primera vez a `a1[0]` se produce un fallo y se trae a caché una línea que incluye a los elementos: `a1[0]`, `a1[1]`, … , `a1[31]`. Lo mismo ocurre para el vector `a2`. Cuando se acceden a los elementos `a1[1]`, `a2[1]`, … , `a1[31]`, `a2[31]`, se producen aciertos.
Cada 32 accesos hay un fallo. Este proceso se repite 6 veces, hasta llegar al elemento 192. Para los últimos 8 elementos del vector, hay un fallo y 7 aciertos. Por tanto el número de accesos por cada vector es de 200 y el número de fallos es de 7, luego la tasa de aciertos es 193 /200 =96,5%

Ejercicio 5

b) Hay que activar las señales T4 y C9

b)

Ciclo 0: MAR ← PC	Señales: T4, C1
Ciclo 1: MBR ← MP	Señales: Td, L, C2
PC ← PC + 4	C4
Ciclo 2: RI ← MBR	Señales: T3, C6
Ciclo 3: Decodificación	
Ciclo 4: MAR ← $t2	Señales: RA= identificador de $t2, T1, C1
Ciclo 5: MBR ← $t1	Señales: RA= identificador de $t1, T1, C3
Ciclo 6: MP ← MBR	Td, E,Ta

UNIVERSIDAD CARLOS III DE MADRID. DEPARTAMENTO DE INFORMÁTICA
GRADO EN INGENIERÍA INFORMÁTICA. ESTRUCTURA DE COMPUTADORES

Para la realización del presente examen se dispondrá de **2 horas.** **NO** se podrán utilizar libros, apuntes **ni** calculadoras de ningún tipo.

Ejercicio 1 (2 puntos). Considere la siguiente definición de función:

```
int  sustituir (String cadena, char c1)
```

La función sustituye cada ocurrencia de carácter `c1` que aparece en la cadena de caracteres `cadena`, por el último carácter de `cadena`. La función devuelve también la posición del último carácter cambiado en la cadena.

Ejemplo: Si `cadena` = "`Hola mundo`", y c1 = '`a`'. La función debe modificar `cadena` para que su nuevo valor sea "`Holo mundo`". La función devolvería para este caso el valor 3.

Se pide:

a) Implemente el código de la función anterior utilizando el ensamblador del MIPS 32.
b) Dado el siguiente fragmento del segmento de datos:

```
.data
   Cad: .ascizz "Esto es una cadena de prueba"
```

Indique el código necesario para invocar a la función `sustituir` pasando como parámetro la cadena `Cad` y el carácter '`a`'. Imprima el resultado que devuelve la función por pantalla. (Tiene que seguirse la convención en el paso de parámetros que se han visto en la asignatura.)

Ejercicio 2 (2 puntos). Un computador posee un sistema de memoria virtual implementada mediante paginación que utiliza páginas de 8 KB. El computador proporciona un espacio de memoria virtual de 2^{32} bytes y tiene 2^{23} bytes de memoria física. Si la tabla de páginas correspondiente a un programa en ejecución es la siguiente:

Bit de presencia	Bit de modificado	Marco de página/ Bloque de swap
1	0	1
0	0	7
1	1	9
1	0	14
1	0	8
1	1	3
0	0	25
0	1	16
0	0	23
1	0	78

Se pide:

a) Indique el formato de la dirección virtual.
b) Indique la dirección física correspondiente a la dirección virtual 0x0000608A.
c) ¿Cuál es el tamaño que ocupa el espacio de direcciones virtual de este programa?
d) Exprese en MB el tamaño de la memoria principal.

Ejercicio3 (2 puntos). Considere un computador de 32 bits con un juego de 140 instrucciones máquina y un banco con 64 registros.
Considere la siguiente instrucción máquina: `sumar R1, R2, dirección`. La instrucción suma el contenido del registro `R1`, con el contenido del registro `R2` y deja el resultado en la posición de memoria dada por z`dirección`.

Se pide:

a) Indique de forma razonada un posible formato para la instrucción anterior.

c) Utilizando el ensamblador del MIPS 32, indique un fragmento de código que sea equivalente a la instrucción `sumar` anterior.

Ejercicio 4 (2 puntos). Sea un computador de 64 bits que dispone de una memoria caché de 512 KB, asociativa por conjuntos de 4 vías y líneas de 128 bytes. El tiempo de acceso a la memoria caché es de 2 ns y el tiempo de penalización de fallo es de 120 ns. Sobre este computador se desea ejecutar el siguiente fragmento de código:

```
int m[32][32];     // matriz de enteros de 32 x 32 elementos.
                   // los elementos de m se almacenan por filas.


for (i=0; i < 32; i = i + 1)
     for (j = 0; j < 32; j = j + 2)
          s = s + m[i][j];
```

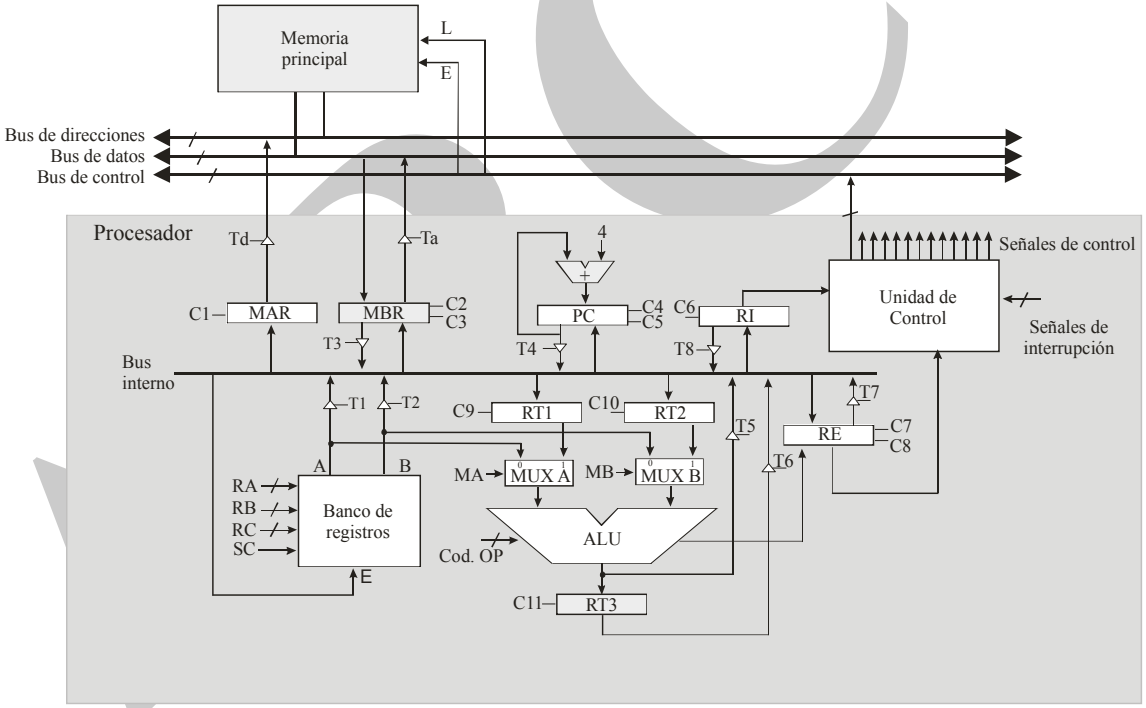
Se pide:

e) Indique de forma razonada el número de líneas y conjuntos de de la caché.

f) Si se considera que la caché se encuentra vacía y que los valores de las variables `i, j` y `s` del código anterior se almacenan en registros, indique, considerando solo los accesos a la matriz `m,` la tasa de aciertos que se obtiene al ejecutar el bucle anterior.

g) Calcule el tiempo total invertido en el acceso a los datos del bucle anterior.

Ejercicio 5 (2 puntos). Considere el siguiente esquema de procesador de 32 bits. El banco de registros incluye 32 registros. Considere que el computador utiliza un ciclo de reloj para realizar la decodificación de la instrucción y que se conecta a una memoria que permite realizar una operación de lectura y de escritura en un ciclo.



Este computador dispone del juego de instrucciones del MIPS32. Se pide:

j) Indique las señales de control necesarias para poder realizar la operación elemental RT1 ← $t2, siendo $t2 el registro número 2 del banco de registros.

k) Indique las operaciones elementales y las señales de control (incluyendo el *fetch*) necesarias para ejecutar la instrucción hipotética `addm $t1, ($t2).` Esta función suma el contenido del registro `$t1` con el contenido de la posición de memoria, que se encuentra almacena en `$t2`. El resultado se almacena en el registro `$t1`. Es decir: `$t1 ← $t1 + MP[$t2]`

Soluciones

Ejercicio 1.

a)

```
sustituir:
          move  $t0, $a0
          li    $v0, 0      // posición del último caracteres cambiado
          lbu   $t1, ($t0)
          bneqz $t1, bucle1
          jr    $ra

  bucle1: beqz  $t1, fin1
          lbu   $t2, ($t0)
          addi  $t0, 1
          lbu   $t1, ($t0)
          b bucle1

// en $t2 está el ultimo caracter de la cadena

    fin1: move  $t0, $a0
          lbu   $t1,  ($t0)
  bucle2: beqz  $t1, fin2
          bneq  $t1, $t2, seguir
          sw    $a1,  ($t0)
          addi  $v0, $v0, 1
  seguir: addi  $t0, 1
          lbu   $t1, ($t0)
          b     bucle2

    fin2: jr    $ra
```

b)

```
la    $a0, Cad
li    $a1, 'a'
jr    sustituir
move  $a0, $v0
li    $v0, 1
syscall
```

Ejercicio 2.

a) El computador tiene páginas de 8 KB = 2^{13} bytes. Como la memoria virtual es de de 2^{32} bytes, se emplean los 19 bits superiores de la dirección para la página y los 13 inferiores para el desplazamiento dentro de la página.

b) La dirección 0x0000608A == 0000 0000 0000 0000 0110 0000 1000 1010
Los 19 bits superiores son 0000 0000 0000 0000 011 = 3. La dirección hace referencia a la página 3, que se encuentra en memoria, en el marco 14 = 1110
La dirección física es 0000 0000 0000 0001 1100 0000 1000 1010 == 0x 0001C08A

c) Este programa ocupa 10 páginas, luego el espacio de direcciones que ocupa es de 10 x 8 = 80 KB.

d) La memoria principal tiene 2^{23} bytes = 2^{13} KB = 2^{3} MB = 8 MB.

Ejercicio 3

a) Se necesitan 8 bits para representar el código de operación y 6 bits para codificar a un registro. Como el computador es de 32 bits, sus direcciones son de 32 bits. Un posible formato sería:

b)

```
Add     R0, R1, R2
Sw      R0,  direccion
```

Ejercicio 4

c) La memoria caché tiene un tamaño de 512 Kb = 2^{19} bytes. Cada línea tiene 128 bytes = 2^7 bytes. El número de líneas viene dado por 2^{19} bytes / 2^7 bytes = 2^{12} líneas = 4096 líneas. Como cada conjunto tiene 4 líneas, el número de conjuntos = 2^{12} / 2^2 = 2^{10} conjuntos = 1024 conjuntos

d) Como cada elemento de la matriz es un entero de 8 bytes (puesto que el computador es de 64 bits), en cada línea caben 128 / 8 = 16 enteros. Cuando se accede por primera vez a `m[0][0]` se produce un fallo y se trae a caché una línea que incluye a los elementos: `m[0][0]`, `m[0][1]`, … , `m[0][15]`. Es decir que cada 8 accesos se produce 1 fallo. Este proceso se repite (32 x 16) / 8 = 64 veces. Por tanto el número total de fallos es de 64 fallos y el de aciertos es 512 – 64 = 448.

e) El tiempo de acceso es = 64 * 120 + 448 * 2 = 8576 ns.

Ejercicio 5.

c) Hay que activar las señales C9, T1, RA = 00010

b)

```
Ciclo 0: MAR ← PC          Señales: T4, C3
Ciclo 1: MBR ← MP          Señales: Td, L, C2
         PC ← PC + 4                C4
Ciclo 2: RI ← MBR          Señales: T3, C6
Ciclo 3: Decodificación
Ciclo 4: MAR ← $t2         Señales: RA= identificador de $t2, T1, C1
Ciclo 5: MP ← MBR          L
Ciclo 6: RT1 ← MBR         T3, c9
Ciclo 7: $t1 ← RT1 + $t1   MA = 1, RB = id. Del registro $t1, Cod. Op = sumar, T5, C8,
                           RC = id. Del registro $t1, SC
```

UNIVERSIDAD CARLOS III DE MADRID
DEPARTAMENTO DE INFORMÁTICA
GRADO EN INGENIERÍA INFORMÁTICA. ESTRUCTURA DE COMPUTADORES

Para la realización del presente examen se dispondrá de **2 horas y media.** **NO** se podrán utilizar libros, apuntes **ni** calculadoras de ningún tipo.

Ejercicio 1. Responda a las siguientes preguntas:

a) ¿Cómo se detecta un desbordamiento en Complemento a 2 cuando se hace una operación de suma?
b) Represente en el estándar de coma flotante IEEE 754 de 32 bits los valores **10,25** y **6,75** . Exprese el resultado final en hexadecimal.
c) Realice la suma de los números anteriores representados en IEEE 754, indicando los pasos que va realizando en cada momento.
d) ¿Cuál es la principal funcionalidad de la memoria caché en un computador?
e) Indique qué diferencias hay entre un computador CISC y uno RISC.
f) En un dispositivo de Entrada/Salida. ¿De qué se encarga el bus de datos? ¿Y el bus de control?
g) Indique qué acciones realiza la unidad de control cuando se produce una interrupción.

Ejercicio 2. Se dispone de un computador con direcciones de memoria de 32 bits, que direcciona la memoria por bytes. El computador dispone de una memoria caché asociativa por conjuntos de 4 vías, con un tamaño de línea de 4 palabras. Dicha caché tiene un tamaño de 64 KB. El tiempo de acceso a la memoria caché es de 2 ns y el tiempo necesario para tratar un fallo de caché es de 80 ns. Indique de forma razonada:

a) Tamaño en MB de la memoria que se puede direccionar en este computador.
b) Número de palabras que se pueden almacenar en la memoria caché.
c) Número de líneas que se pueden almacenar en el mismo conjunto.
d) Número de líneas de la caché.
e) Número de conjuntos de la caché.
f) Indique la tasa de aciertos necesaria para que el tiempo medio de acceso al sistema de memoria de este computador sea de 10 ns.

Ejercicio 3. Se desea desarrollar utilizando el ensamblador del MIPS 32 el código de la siguiente función:

```
void vuelta(char[] cadena_origen, char[] cadena_final);
```

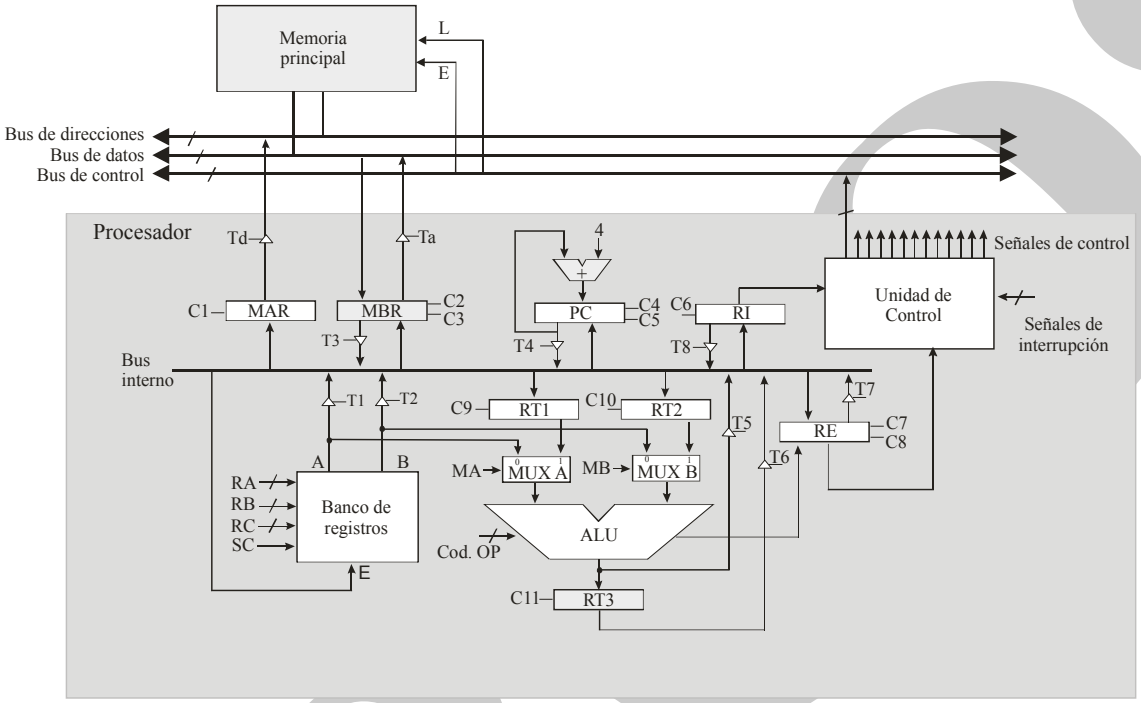
que toma como parámetros dos cadenas de texto. Esta función invierte la primera cadena y guarda el resultado en la segunda. Así por ejemplo, la llamada a esta función con "`Hola Mundo`" almacenará en `cadena_final` la cadena "`odnuM aloH`". Considere que la cadena final tiene al menos el mismo espacio reservado que la cadena origen. Considere además que el final de la cadena se indica con '`\0`'. Se pide:

a) Indique el algoritmo que va a utilizar para implementar la rutina anterior.
b) Desarrolle, de acuerdo al algoritmo descrito en el apartado a, el contenido de la subrutina utilizando el ensamblador del MIPS 32. Debe seguirse estrictamente el convenio de paso de parámetros descrito en la asignatura.
c) Desarrolle la función `void imprimir(char[] cadena)` en ensamblador MIPS 32. Dicha función toma como parámetro una cadena de caracteres y la imprime por pantalla. Debe seguirse estrictamente el convenio de paso de parámetros descrito en la asignatura.
d) Suponga que cuenta con el siguiente segmento de datos:

```
.data

    cadena1:  .asciiz "Hola Mundo"

    .align 2

    cadena2:  .space 11
```

desarrolle, utilizando las rutinas anteriores, una rutina main que primero de la vuelta a `cadena1` y la almacene en `cadena2`, y posteriormente imprima el contenido de `cadena2` por pantalla. Para ello, tenga en cuenta el convenio de paso de parámetros.

Ejercicio 4. Considere un computador de **32 bits**, que dispone de un banco de **32 registros**, que se encuentra conectado a una memoria, que se **direcciona por bytes** y que requiere **un ciclo para las operaciones de lectura y escritura**. El computador dispone de un juego de instrucciones con **130 instrucciones máquina**. La unidad aritmético-lógica es capaz de realizar operaciones aritméticas y lógicas (sumar, restar, multiplicar, dividir, incrementar, disminuir en uno, etc.). El procesador emplea un ciclo para la decodificación de la instrucción. Su estructura es la siguiente



Considere las dos siguientes instrucciones:

`moveM dir1, dir2` Esta instrucción mueve el contenido de la posición de memoria `dir2` a la posición de memoria `dir1`.

`moveR R1, R2` Esta instrucción mueve el contenido del registro R2 al registro R1 (R1 y R2 son los registros 1 y 2 del banco de registros).

Se pide:

a) Indique, de forma razonada, el formato de las instrucciones anteriores.

b) Indique, en función del formato definido en el apartado a, las operaciones elementales y las señales de control necesarias para la ejecución de las instrucciones anteriores. Incluya también el ciclo de fetch.

Ejercicio 5. Se desea adquirir un disco duro para un equipo doméstico que dé la mejor relación calidad/precio. Dado que el disco se va a utilizar de manera intensiva para la grabación y reproducción de películas, se considera más importante el tiempo medio de acceso y la capacidad de almacenamiento que el coste por MB del disco.

Se dispone de dos posibles discos candidatos con los siguientes parámetros:

Datos	Disco A		Disco B	
Platos	12	platos/disco	24	platos/disco
Superficies por plato	2	superficie/plato	2	superficie/plato
Pistas	100000	pista/cilindro	50000	pista/cilindro
Tamaño Pista	64	sector/pista	64	sector/pista
Tamaño Sector	512	bytes/sector	512	bytes/sector
Velocidad Rotación	5400	r.p.m.	7200	r.p.m.
$T_{busqueda}$(Disco) es proporcional a la distancia	0,001	seg/pista	0,003	seg/pista
Coste (precio final)	24,26	€uros	33,49	€uros

El disco A está dotado de una tecnología más antigua que el disco B, como se puede apreciar por la velocidad de rotación del disco.

Se pide:

a) Calcular la capacidad de cada disco si se formatea con cuatro zonas de igual tamaño con densidades de 64, 128, 256 y 512 sectores por pista.

b) Calcular el tiempo medio de acceso de cada disco con dicho formato por zonas.

c) Calcular el coste por MB de cada disco y justifique, en función de todos los datos calculados, cuál es el mejor disco para comprar.

Ejercicio 6 En relación a la práctica 3 responda a las siguientes preguntas:

a) Describa y critique (positiva y negativamente) brevemente la aportación realizada por su compañero de prácticas en la realización de la práctica 3.
b) Valore de 1 (muy malo) a 10 (muy bueno) la aportación realizada por su compañero en la realización de la práctica 3.

Observación: Responda solo si ha realizada la práctica en grupo.

Soluciones:

Ejercicio 1:

a) Se detecta cuando los dos operandos tienen el mismo signo y el resultado tiene signo distinto al de los operandos.

b) **$10,25_{(10}$** = $1010,01_{(2}$ = 1.01001 x 2^$3_{(2}$
Signo = 0
Exponente = 127+3 = $130_{(10}$ = $10000010_{(2}$
Mantisa = 010010000…00

En Binario: 0100 0001 0010 0100 00…00
En Hexadecimal: **0x41240000**

$6,75_{(10}$ = $110,11_{(2}$ = 1.1011 x 2^$2_{(2}$
Signo = 0
Exponente = 127+2 = $129_{(10}$ = $10000001_{(2}$
Mantisa = 10110000…00

En Binario: 0100 0000 1101 1000 00…00
En Hexadecimal: **0x40D80000**

c) Para sumar ambos números en IEEE754 , lo primero que tenemos que hacer es igualar exponentes, y tener en cuenta a la hora de sumar el bit implícito de la mantisa, después realizar la suma, y si el resultado no está normalizado, normalizarlo.

10,25 = 0 10000010 1.01001000000000000000000
6,75 = 0 10000010 0.11011000000000000000000
Suma = 0 10000010 10.00100000000000000000000
Resultado normalizado = 0 10000011 00010000000000000000000
En Hexadecimal: **0x41880000**

d) La principal funcionalidad de la memoria caché en un procesador es reducir el tiempo de acceso a los datos o instrucciones que se encuentran en memoria principal. Para ello la memoria caché realiza copias de datos o instrucciones de la memoria principal, al ser más rápida que la memoria principal cuando se repite el acceso a un dato se mejora el rendimiento.

e) CISC: Complex Instruction Set Architecture.
- *Muchas instrucciones.*
- *Complejidad variable.*
- *Diseño irregular.*

RISC: Reduced Instruction Set Code
- *Instrucciones bastante simples.*
- *Número reducido de instrucciones.*
- *Instrucciones de tamaño fijo.*
- *La mayoría de las instrucciones usan registros.*
- *Paso de parámetros a través del banco de registros.*
- *Arquitecturas segmentadas.*

f) El bus de datos es una línea entre el dispositivo y la CPU que se encarga de transmitir los datos que se intercambiarán. El bus de control intercambiará entre el dispositivo y la CPU las señales de control y estado de los intercomunicadores.

g) La unidad de control:
- *Salva el contenido del contador de programa y del registro de estado, por ejemplo, en la pila.*
- *Cambia el modo de ejecución de usuario o núcleo*

- *Obtiene la dirección de la rutina de tratamiento de la interrupción y la almacena en el PC.*

Ejercicio 2

a) Si el computador tiene direcciones de memoria de 32 bits, el tamaño de memoria principal será 2^{32} bytes = 2^{12} MB
b) Tamaño de la cache = 64 KB
   Como el computador es de 32 bits, las palabras son de 32 bits, 4 bytes, por tanto el número de palabras que puede almacenar la cache es de 64 KB / 4 bytes = 16 * 1024 palabras.
c) En cada conjunto se pueden almacenar 4 líneas.
d) El tamaño de la línea es de 4 palabras = 4 *4 = 16 bytes. El número de líneas = 64 KB / 16 bytes = 2^{16} / 2^{4} = 2^{12} = 4096 líneas
e) El número de conjuntos de la caché será el número de líneas caché dividido por el número de vías de cada conjunto = 4096 líneas / 4 vías = 1024 conjuntos
f) Tm = Ta * T_cache + (1-Ta) * T_fallo => 10 = Ta*2 + (1-Ta) * 80 => despejando Ta = 70 / 78, es decir, una tasa de aciertos del 89 %

Ejercicio 3

a) SOLUCIÓN:

```
Apuntar puntero 1 al comienzo de cadena origen
Longitud de cadena origen = 0
Mientras no se lea un 0
        Leer carácter
        Aumentar longitud en 1
        Mover puntero de cadena origen a siguiente posición
Apuntar puntero 1 al comienzo de cadena final
Retroceder una posición puntero de cadena origen
Mientras longitud sea mayor o igual a 0
        Leer carácter en cadena origen
        Escribir carácter en cadena final
        Mover puntero de cadena origen a posición anterior
        Mover puntero de cadena final a siguiente posición
        Disminuir longitud en 1
Escribir final de cadena en cadena final
```

b) Desarrolle, de acuerdo al algoritmo descrito en el apartado a, el contenido de la subrutina utilizando el ensamblador del MIPS 32. Debe seguirse estrictamente el convenio de paso de parámetros descrito en la asignatura.

SOLUCIÓN:

```
vuelta:
        li $t0 0                    #Contador de longitud
        li $t1 0                    #Para almacenar caracteres
        move $t2 $a0

        for1:                       #Medición de la longitud
                lb $t1 ($t2)        #Carga de un caracter
                beqz $t1 seguir     #Localizar '\0'
                add $t0 $t0 1       #Incremento de longitud
```

```
        add $t2 $t2 1           #Incremento de puntero de cadena1
        j for1

seguir:
        sub $t2 $t2 1           #Se vuelve a la última posición de cadena1

for2:                           #Bucle para invertir la cadena
        bgt $a0 $t2 fin         #Terminar de leer cadena1
        lb $t1 ($t2)            #Carga de un carácter
        sb $t1 ($a1)            #Almacenamiento de un carácter
        sub $t2 $t2     1       #Decremento puntero de cadena1
        add $a1 $a1     1       #Incremento puntero de cadena2
        j for2

fin:                            #Se cierra la cadena2
        li $t1 0
        sb $t1 ($a1)

        jr $ra
```

c) Desarrolle la función `void imprimir(char[] cadena)` en ensamblador MIPS 32. Dicha función toma como parámetro una cadena de caracteres y la imprime por pantalla. Debe seguirse estrictamente el convenio de paso de parámetros descrito en la asignatura.

SOLUCIÓN:

```
imprimir:
        li $v0 4
        syscall

        jr $ra
```

d) Suponga que cuenta con el siguiente segmento de datos:

```
.data

    cadena1:  .asciiz "Hola Mundo"

    .align 2

    cadena2:  .space 11
```

desarrolle, utilizando las rutinas anteriores, una rutina main que primero de la vuelta a cadena1 y la almacene en cadena2, y posteriormente imprima el contenido de cadena2 por pantalla. Para ello, tenga en cuenta el convenio de paso de parámetros.

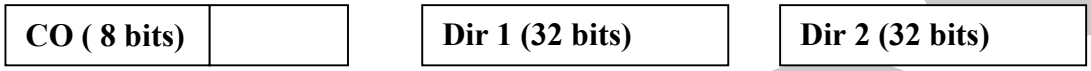
SOLUCIÓN:

```
main:
        sw $ra ($sp)
        sub $sp $sp 4

        la $a0 cadena1
        la $a1 cadena2
        jal vuelta

        la $a0 cadena2
        jal imprimir
```

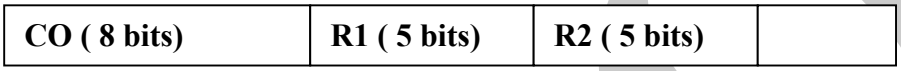
```
lw $ra 4($sp)
add $sp $sp 4

jr $ra
```

Ejercicio 4.

a) Como el computador dispone de 130 instrucciones, se necesitan 8 bits para codificar una instrucción. El computador es de 32 bits, por tanto se necesitan 32 bits para codificar una dirección de memoria. Esta instrucción, por tanto, necesita tres palabras para su codificación. Su contenido es el siguiente:



El formato de la segunda instrucción sería:



b)

Primera instrucción

ciclo	**Operaciones elementales**	**Señales de control**
C0	MAR ← PC	T4, C1
C1	MBR ← MP, PC ← PC + 4	Td, L, C2, C4
C2	RI ← MBR	T3, C6
C3	Decodificación	
C4	MAR ← PC	T4, C1
C5	MBR ← MP, PC ← PC +4	Td, L, C2, C4
C6	RT1 ← MBR, RT2 ← MBR	T3, C9, C10
C7	MAR ← PC,	T4, C1
C8	MBR ← MP, PC ← PC+ 4	Td, L, C2, C4
C9	MAR ← MBR	T3, C1
C10	MBR ← MP	Td, L, C2
C11	MAR ← RT1 OR RT2	MA, MB, T5, C1, Cod. OP = OR
C12	MP ← MBR	Td, Ta, E

Segunda instrucción:

Ciclo	Operaciones elementales	Señales de control
C0	MAR ← PC	T4, C1
C1	MBR ← MP, PC ← PC + 4	Td, L, C2, C4
C2	RI ← MBR	T3, C6
C3	Decodificación	
C4’	R1 ← R2	RA = 00001, RC = 00010, T1, SC

Ejercicio 5.

a) Calcular la capacidad de cada disco si se formatea con cuatro zonas de igual tamaño con densidades de 64, 128, 256 y 512 sectores por pista.

Hay que tener en cuenta que la fórmula a utilizar es:

$$\text{Nº Platos} * \text{Nº Superficies} * \left(\frac{\text{Nº Pistas}}{4}\right) * (64 * \text{Tam(Sector)} + 128 * \text{Tam(Sector)} + 256 * \text{Tam(Sector)} + 512 * \text{Tam(Sector)})$$

Apdo. c	C_{mz}(Disco A)		C_{mz}(Disco B)	
C_{mz}() en bytes	294.912.000.000,00	bytes	294.912.000.000,00	bytes
C_{mz}() en KB	288.000.000,00	KB	288.000.000,00	KB
C_{mz}() en MB	281.250,00	MB	281.250,00	MB
C_{mz}() en GB	274,66	GB	274,66	GB

b) Calcular el tiempo medio de acceso de cada disco con dicho formato por zonas.

Apdo. b: Cálculos previos				
$T_{busqueda}()$	$T_{busqueda}$(Disco A)		$T_{busqueda}$(Disco B)	
Peor Caso: de la pista 0 a la pista 99999	100000	pistas	5000	pistas
Mejor Caso: se está en la pista	0	pistas	0	pistas
Caso medio: se está a la mitad de pistas entre ambas	50000	pistas	2500	pistas
$T_{busquedapromedio}$(Disco A)=Sumatorio (Caso*su tiempo)/3casos	50	segundos	7,5	segundos
$T_{busquedapromedio}$(Disco A)=Sumatorio (Caso*su tiempo)/3casos	50000	mseg.	7500	mseg.

$T_{rotacion}()$	$T_{rotacion}$(Disco A)		$T_{rotacion}$(Disco B)	
Velocidad rotación en segundos(v_{rot})	90	r.p.seg.	120	r.p.seg.
Tiempo una rotación ($1/v_{rot}$) en segundos	0,011111111	segundos	0,008333333	segundos
Tiempo una rotación ($1/v_{rot}$) en mseg	11,11111111	mseg.	8,333333333	mseg.

Calculando el $T_{transferencia}$				
$T_{transferencia}()$ con zonas de igual número de pistas y diferente densidad: 64, 128, 256 y 512 sectores/pista				
$T_{transferencia}()$ con zona de **64 sectores/pista**	Disco A		Disco B	
Tiempo recorrer todos los sectores de una pista	11,11111111	mseg.	8,333333333	mseg.
Número de sectores por pista	64	sector/pista	64	sector/pista
$T_{transferencia}()$ promedio con zona de **64 sectores/pista**	0,173611111	mseg.	0,130208333	mseg.

$T_{transferencia}()$ con zona de **128 sectores/pista**	Disco A		Disco B	
Tiempo recorrer todos los sectores de una pista	11,11111111	mseg.	8,333333333	mseg.
Número de sectores por pista	128	sector/pista	128	sector/pista
$T_{transferencia}()$ promedio con zona de **128 sectores/pista**	0,086805556	mseg.	0,065104167	mseg.

$T_{transferencia}()$ con zona de **256 sectores/pista**	Disco A		Disco B	
Tiempo recorrer todos los sectores de una pista	11,11111111	mseg.	8,333333333	mseg.
Número de sectores por pista	256	sector/pista	256	sector/pista
$T_{transferencia}()$ promedio con zona de **256 sectores/pista**	0,043402778	mseg.	0,032552083	mseg.

$T_{transferencia}()$ con zona de **512 sectores/pista**	Disco A		Disco B	
Tiempo recorrer todos los sectores de una pista	11,11111111	mseg.	8,333333333	mseg.
Número de sectores por pista	512	sector/pista	512	sector/pista
$T_{transferencia}()$ promedio con zona de **512 sectores/pista**	0,021701389	mseg.	0,016276042	mseg.

	$T_{transferencia}$(Disco A)		$T_{transferencia}$(Disco B)	
$T_{transferencia}()$ con zonas (promediando)	0,081380208	mseg.	0,061035156	mseg.

Calculando T_{acceso} requerido en los apartados b y d	Disco A		Disco B	
$T_{busqueda}()$	50000	mseg.	7500	mseg.
$T_{rotacion}()$	11,11111111	mseg.	8,333333333	mseg.
$T_{transferencia}()$ con zonas (promediando)	0,081380208	mseg.	0,061035156	mseg.
$T_{acceso}()$ con Zonas	50011,19249	mseg.	7508,394368	mseg.

c) Calcular el coste por MB de cada disco y justifique, en función de todos los datos calculados, cuál es el mejor disco para comprar.

El coste por GB de cada disco es:

Calculando el coste por almacenamiento de cada disco	Disco A		Disco B	
Tamaño con CAV	73,24	GB	73,24	GB
Tamaño con pistas de tamaño variable	274,66	GB	274,66	GB
Coste por GB con CAV	0,33	€uros/GB	0,46	€uros/GB
Coste por GB con pistas de tamaño variable	0,09	€uros/GB	0,12	€uros/GB

Dado que ambos discos tendrían el mismo tamaño, ya sea como CAV o con distribución variable de sectores por pista, el tamaño del disco no es un dato significativo para la decisión.

Fijándose que el tiempo de acceso del disco A es mayor que el tiempo de acceso del disco B pero el coste por GB es menor en el disco A, se tendrá que tomar la decisión entre qué característica es más deseable: si es la velocidad entonces habrá que adquirir el disco B, si es la economía entonces habrá que adquirir el disco A.

UNIVERSIDAD CARLOS III DE MADRID
DEPARTAMENTO DE INFORMÁTICA
GRADO EN INGENIERÍA INFORMÁTICA. ESTRUCTURA DE COMPUTADORES

Para la realización del presente examen se dispondrá de **2:30 horas.** **NO** se podrán utilizar libros, apuntes **ni** calculadoras de ningún tipo.

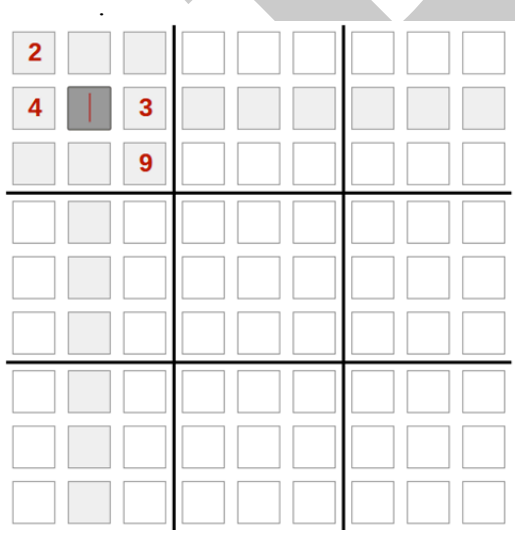
Ejercicio 1. Responda a las siguientes preguntas:

h) ¿Cuál sería la representación en Complemento a 2 del número -31 utilizando 7 bits?
i) Indique el valor decimal del siguiente número representado en el estándar IEEE 754 de simple precisión: 0xBF400000.
j) ¿Qué es la MMU y cuál es su principal función?
k) Indique brevemente en qué consiste el ciclo de fetch.
l) En un dispositivo de Entrada/Salida. ¿De qué se encarga el bus de direcciones?
m) Indique qué acciones realiza la unidad de control cuando se produce una interrupción.

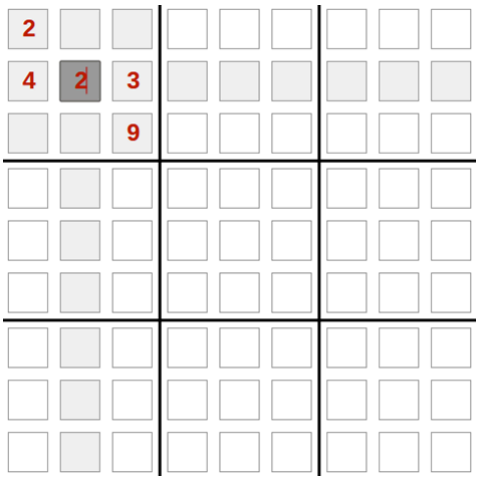
Ejercicio 2. Se dispone de un computador (que direcciona la memoria por bytes) con un sistema de memoria virtual que emplea direcciones virtuales de 16 bits y páginas de 2 KB. El computador dispone de una memoria física instalada de 8 KB. Se pide:

a) ¿Cuál es el tamaño máximo, en KB, de la memoria virtual que se puede direccionar.
b) Indique el número de páginas máximo que puede tener un programa que ejecuta en este computador.
c) Indique el formato de la dirección virtual empleada en este computador.
d) Indique el tamaño del marco de página.
e) Indique el número de marcos de página de la memoria física.
f) Indique el formato de la dirección física de este computador.
g) ¿Cuál es el número máximo de entradas que puede tener la tabla de páginas asociada a un programa que ejecuta en este computador, asumiendo que se trata de una tabla de páginas de un único nivel.
h) Indique al menos dos campos de cada entrada en la tabla de página y diga para que se utilizan.

Ejercicio 3. Como parte de un programa de resolución de sudokus, se desea desarrollar una subrutina en ensamblador del MIPS32 que compruebe si el valor de una celda cumple con la regla de los cuadrados 3x3, es decir, un número no se puede repetir dentro de un mismo cuadrado de tamaño 3x3. Un sudoku se divide en 9 cuadrados de 3 x 3. Dentro de cada uno de estos cuadrados de 3x3 no se puede repetir un número (del 1 al 9), independientemente de si están o no en la misma fila o columna. El siguiente sudoku cumple con esta regla:



Sin embargo, el siguiente sudoku no cumple la regla, ya que en el primer cuadrado 3x3 se repite el número 2.



Suponga que el sudoku se almacena como una matriz de 9 elementos por 9 elementos. Dicha matriz se almacena por filas consecutivas en la sección de datos y cada elemento de la matriz es de 1 byte. La rutina a desarrollar recibirá como parámetros:

- La dirección de inicio del sudoku
- El valor que se quiere introducir en el sudoku (un número de 0 a 9)
- El índice de la fila y columna donde se quiere introducir el valor anterior. El índice de la fila y la columna irán de 0 a 8.

La función devolverá un 0 si el contenido que se quiere introducir en el sudoku cumple con la regla y 1 si es incorrecto (no cumple con la regla)

.

Se pide:

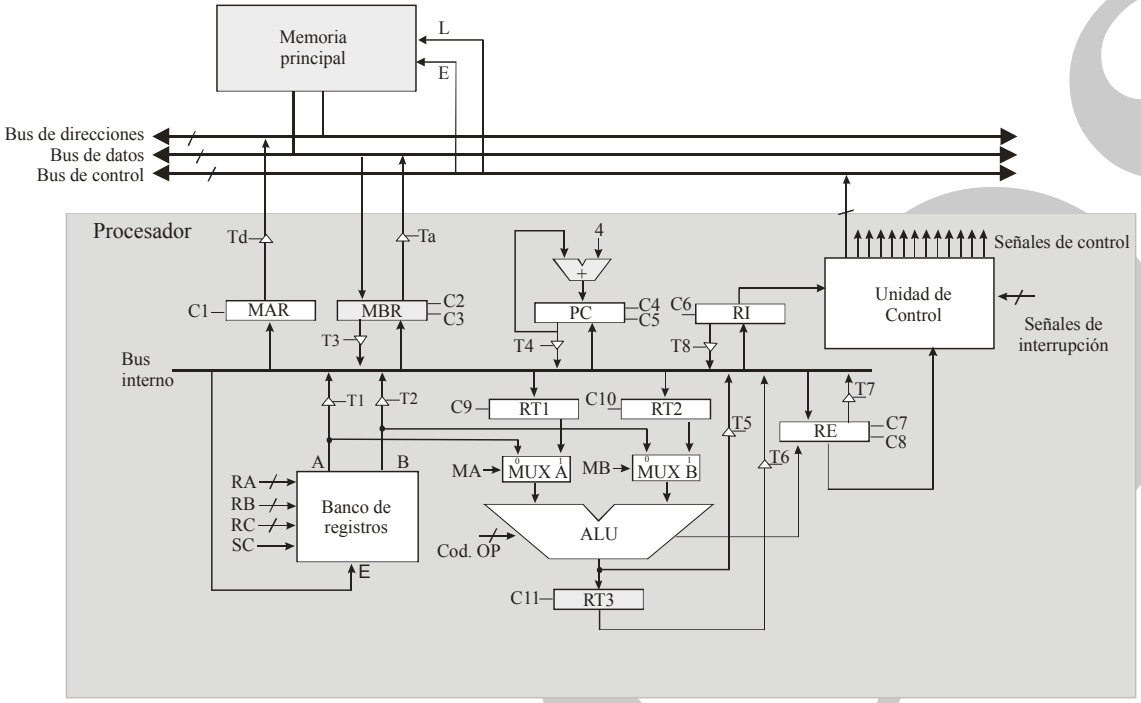
a) Indique un posible pseudocódigo de la función anterior.
b) Desarrollo el código en ensamblador que implemente el algoritmo anterior.
c) Se desea desarrollar una rutina main que llame a la subrutina anterior. Suponga que cuenta con el siguiente segmento de datos:

```
.data
valor:      .word 8
fila:       .word 1
columna:    .word 2
tablero:    .byte     0, 6, 0, 1, 0, 4, 0, 5, 0
            .byte     0, 0, 8, 3, 0, 5, 6, 0, 0
            .byte     2, 0, 0, 0, 0, 0, 0, 0, 1
            .byte     8, 0, 0, 4, 0, 7, 0, 0, 6
            .byte     0, 0, 6, 0, 0, 0, 3, 0, 0
            .byte     7, 0, 0, 9, 0, 1, 0, 0, 4
            .byte     5, 0, 0, 0, 0, 0, 0, 0, 2
            .byte     0, 0, 7, 2, 0, 6, 9, 0, 0
            .byte     0, 4, 0, 5, 0, 8, 0, 7, 0
```

Ejercicio 4. Se dispone de un computador con una frecuencia de reloj de 2 GHz. El número medio de ciclos por instrucción máquina es de 20. A este computador se le conecta un dispositivo que genera 100.000 interrupciones por segundo. La rutina de tratamiento de la interrupción ejecuta 500 instrucciones. Se pide:

a) Calcule el porcentaje de tiempo que este computador dedica al tratamiento de este dispositivo.
b) ¿Podría este computador atender a todas las interrupciones generadas por este dispositivo, si la rutina de tratamiento de la interrupción tuviera 2000 instrucciones máquina? Razone su respuesta.

Ejercicio 5. Considere un computador de **32 bits**, que dispone de un banco de **32 registros**, que se encuentra conectado a una memoria, que se **direcciona por bytes** y que requiere **un ciclo para las operaciones de lectura y escritura**. El computador dispone de un juego de instrucciones con **110 instrucciones máquina**. La unidad aritmético-lógica es capaz de realizar operaciones aritméticas y lógicas (sumar, restar, multiplicar, dividir, incrementar, disminuir en uno, etc.). El procesador emplea un ciclo para la decodificación de la instrucción. Su estructura es la siguiente



Considere las dos siguientes instrucciones:

`PUTS R1, R2`	Esta instrucción inserta dos palabras en la cima de la pila, primero el contenido del registro R1 y encima el contenido del registro R2
`ADDS`	Esta instrucción extrae de la pila el contenido de las dos primeras palabras apiladas. Realiza la suma de estos dos valores e inserta el resultado en la cima de la pila
`POP R1`	Esta función extrae un elemento (palabra de la pila) y lo almacena en R1

Considerando que el registro puntero de pila es el registro R30 y que la unidad aritmética lógica dispone entre otros de una operación que permite sumar 4 al dato que entra en la ALU por el MUX A y otra que permite restar 4 al dato que entra a la ALU por el MUX A. Se pide:

c) Indique, de forma razonada, el formato de las instrucciones anteriores.

d) Indique, en función del formato definido en el apartado a, las operaciones elementales y las señales de control necesarias para la ejecución de la instrucción PUTS. Incluya también el ciclo de *fetch*.

e) Utilizando las tres instrucciones anteriores, escriba un fragmento de programa cuyo comportamiento sea similar al que realiza la instrucción de MIPS `add R1, R2, R3`.

Soluciones:

Ejercicio1.

a) El número -31 utilizando 7 bits es: 31(10 = 0011111(2 luego -31= 1100000(C1 + 1 = 1100001(C2
b) El valor decimal de 0xBF400000 es el siguiente:

En Binario: 1011 1111 0100 0000…00

Signo= Negativo
Exponente = 01111110(2 = 126(10 = Exponente = -1
Mantisa = 1

Número en binario: -1,1 x 2^-1(2 = -0,11(2 = -0,75(10

c) La MMU (*Memory Management Unit*), es una unidad hardware que permite la traducción de las direcciones virtuales generadas por el procesador en direcciones físicas que pueden referenciar marcos de páginas en la memoria principal (si acierto) o bien permite detectar que una página no se encuentra en la memoria principal, generando el correspondiente fallo de página.
d) El ciclo de *fetch* consiste en la captación de la instrucción a ejecutar por parte del procesador, de tal modo que se lee de memoria la instrucción que se encuentra en la posición de memoria almacenada en el contador de programa. La instrucción leída se almacena en el registro de instrucción.
e) El bus de direcciones es una línea entre el dispositivo y la CPU que se encarga de especificar cuál es la posición de memoria o dispositivo a comunicar
f) La unidad de control:
   - Salva el contenido del contador de programa y del registro de estado, por ejemplo, en la pila.
   - Cambia el modo de ejecución de usuario o núcleo
   - Obtiene la dirección de la rutina de tratamiento de la interrupción y la almacena en el PC.

Ejercicio 2

a) Se pueden direccionar 2^{16} direcciones de memoria. Como el computador direcciona la memoria por byte, en cada dirección se almacena un byte y por tanto el tamaño de la memoria virtual será de 64KB.
b) Como cada página es de 2KB, como mucho habrá 64KB/2KB = 32 páginas.
c) El formato de la dirección virtual es el siguiente:

Nº página = 5 bits	**Desplazamiento = 11 bits**

d) El tamaño del marco de página es igual al tamaño de la página, o sea, 2 KB.
e) Si la memoria física es de 8KB. Habrá 8 KB/2 KB = 4 marcos.
f) El formato de la dirección física es el siguiente:

Nº Marco = 2 bits	**Desplazamiento = 11 bits**

g) Como hay 32 páginas y la tabla de páginas es de un único nivel, tendrá 32 entradas como mucho: una para cada posible página de la memoria virtual.
h) Posibles campos:
P/A = Presente/Ausente, indica si una página se encuentra en Memoria Principal o no.
M = Modificado, indica si la página que se encuentra en Memoria Principal ha sido modificada o no, de manera que si se ha modificado cuando se produzca un fallo de página antes de traer una nueva página a Memoria Principal habrá que grabarla en el dispositivo de almacenamiento utilizado para dar soporte a la memoria virtual.
Nº de Marco = indica el marco en el que se encuentra una página en caso de estar presente en Memoria Principal

Ejercicio 3.

a) El pseudocódigo es el siguiente:

Guardar cociente de fila/3 en Y
Repetir 3 veces:
    Guardar cociente de columna/3 en X
    Repetir 3 veces
        Situarse en (X, Y)
        Si la posición no es la que se quiere comprobar y el valor es igual
            No se cumple la regla
            Terminar
        Si no
            X + 1
Y + 1

b) Código:

```
comprobar_cuadrado:
          div $t0, $a1, 3           #$t0 = fila / 3
          mul $t0, $t0, 27          #Desplazamiento de la fila
          div $t1, $a2, 3           #$t1 = col / 3
          mul $t1, $t1, 3           #Desplazamiento de la columna
          add $t1, $t0, $t1         #Desplazamiento de la primera celda de la caja
          li $t0, 3                 #Contador de filas
          li $t3, 3                 #Contador de columas
          li $v0 1            #Carga el valor de acierto (1)

     for:
          add $t4 $t1 $a3           #Índice de la celda
          lb $t2, ($t4)                   #Valor de la celda
          beq $a0, $t2, incorrecto  #Número repetido en la caja
          sub $t3, $t3, 1                 #Disminuye el contador de columnas
          beqz $t3, fin             #Condición de parada
          addi $t1, $t1, 1                      #Incrementa el desplazamiento
          j for


     incorrecto:
          li $v0, 0               #0 en caso de fallo

     fin:
          jr $ra
```

c) .

```
    main:
          lw $a0 valor
          lw $a1 fila
          lw $a2 columna
          la $a3 tablero

          sw $ra ($sp)
          sub $sp $sp 4
          jal comprobar_cuadrado
```

```
add $sp $sp 4
lw $ra ($sp)
jr $ra
```

Ejercicio 4.

b) Como el computador dispone de 110 instrucciones, se necesitan 8 bits para codificar una instrucción. El computador es de 32 bits, por tanto se necesitan 32 bits para codificar una dirección de memoria. Esta instrucción, por tanto, necesita tres palabras para su codificación. Su contenido es el siguiente:

El formato de la instrucción PUTS es:

CO (8 bits)	**R1 (5 bits)**	**R2 (5 bits)**	Sin usar (14 bits)

El formato de la segunda instrucción sería:

CO (8 bits)	Sin usar (24 bits)

El formato de la tercera instrucción es:

CO (8 bits)	**R1 (5 bits)**	Sin usar (19 bits)

c)

Primera instrucción PUTS

ciclo	**Operaciones elementales**	**Señales de control**
C0	MAR ← PC	T4, C1
C1	MBR ← MP, PC ← PC + 4	Td, L, C2, C4
C2	RI ← MBR	T3, C6
C3	Decodificación	
C4	R30 ← R30 – 4	RA = 11110 (30 en binario) MA = 0 Cod. Op = restar 4 T5, RC = 11110, SC
C5	MAR ← R30	RA = 11110 (30 en binario) T1 C1
C6	MBR ← R1	RA = 00001, T1, C3
C7	MP ← MBR	Td, Ta, E
C7	R30 ← R30 – 4	RA = 11110 (30 en binario) MA = 0 Cod. Op = restar 4 T5, RC = 11110, SC
C8	MAR ← R30	RA = 11110 (30 en binario) T1 C1

C9	MBR ← R2	RA = 00010, T1, C3
C10	MP ← MBR	Td, Ta, E

d) El fragment necesario sería

```
PUTS R1, R2

ADDS

POP R1
```

Ejercicio 5.

a) El computador dispone de una frecuencia de reloj de 2 GHz = 2 x 10^9 ciclos/segundo.
La rutina de tratamiento ejecuta 500 instrucciones máquina, lo que suponen: 500 x 20 = 10000 = 10^4 ciclos de reloj consumidos por cada interrupción generada. Como el dispositivo genera 100000 interrupciones por segundo, el número de ciclos por segundo dedicados a tratar a este dispositivo es 10^5 x 10^4 = 10^9 ciclos por segundo.

Como la frecuencia de reloj es 2 x 10^9 ciclos/segundo, el porcentaje de tiempo dedicado a tratar la rutina sería: 10^9 / (2 x 10^9) x 100 = 50%.

b) Si la rutina tuviera 2000 instrucciones máquina, requeriría 2000 x 20 = 4 x 10^4 ciclos de reloj. Si se generan 100000 interrupciones por segundo, serían necesarios 10^5 x 4 x 10^4 = 4 x 10^9 ciclos. Como el computador solo dispone de = 2 x 10^9 ciclos/segundo, no tiene capacidad, por tanto para poder atender todas las interrupciones que genera este dispositivo.