UNIVERSIDAD CARLOS III DE MADRID. DEPARTAMENTO DE INFORMÁTICA
GRADO EN INGENIERÍA INFORMÁTICA. DOBLE GRADO ADE E INGENIERÍA INFORMÁTICA
ESTRUCTURA DE COMPUTADORES
21 de junio de 2018 **Examen final. Convocatoria extraordinaria**

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

Ejercicio 1 (1 punto). Calcule el error que se comete al almacenar el valor entero $2^{30}+7$ en una variable de tipo `float` y en una de tipo `double`.

Ejercicio 2 (3 puntos). Considere la rutina `Convertir`. Esta rutina acepta tres parámetros de entrada:

- La dirección de una matriz (se almacena por filas) de números enteros de dimensión `MxN`.
- El número de filas de la matriz (`M`).
- El número de columnas de la matriz (`N`).

La función convierte todos los valores negativos de la matriz en positivos. La función devuelve dos valores:

- El número de elementos cambiados, es decir, el número de elementos negativos en la matriz original.
- El número de elementos con valor igual a 0.

Se pide:

a) Codifique correctamente en el ensamblador del MIPS32 la rutina `Convertir` anteriormente descrita.

b) Dada la siguiente definición de matriz de dimensión 3x4:

```
.data
  Matriz:   .word       0, 1, -2, 4
                        1, 1, 2, -3
                        2, 2, -2, 0
```

Codifique el fragmento de código que permite invocar correctamente a la función `Convertir` para la matriz definida en el segmento de datos.

Ejercicio 3 (2 puntos). Considere el siguiente fragmento de código:

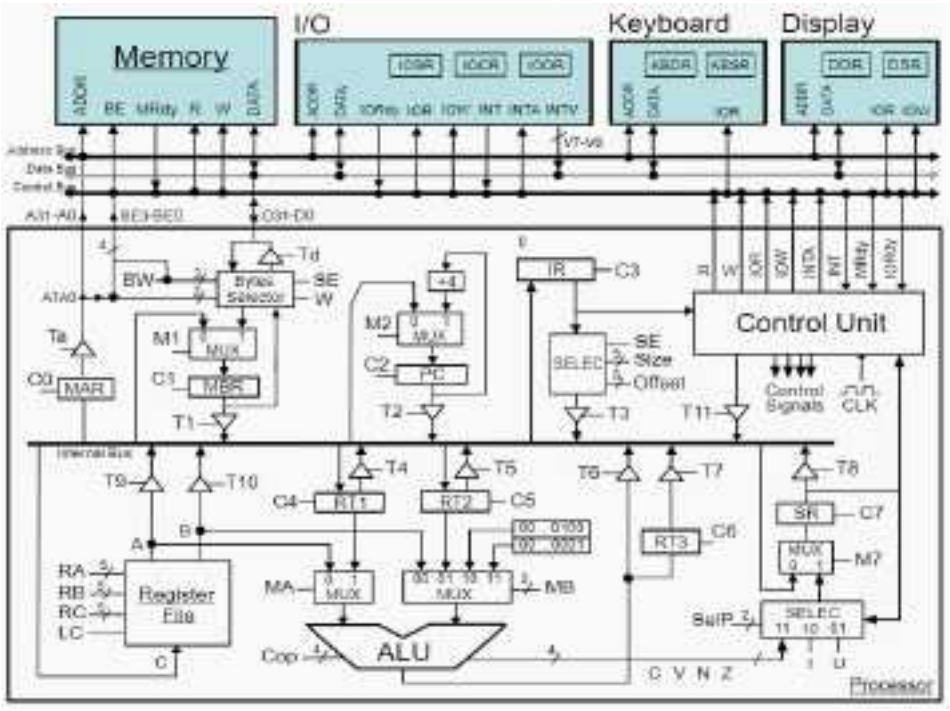
```
double  A[2048];
double  B[2048];
double  C[2048];

for (i=0; i<2048; i++) {
     C[i] = B[i] + A[i] + A[i];
}
```

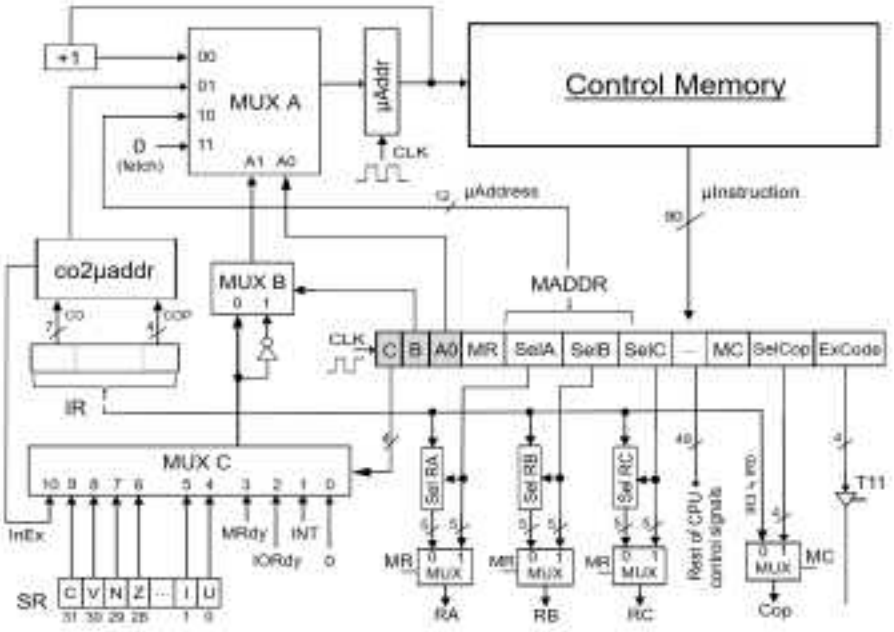
Dicho código se ejecuta en una arquitectura con un ancho de palabra de 32 bits, que incluye una memoria caché de datos totalmente asociativa de 64 KB, con política de escritura retrasada y líneas de 64 bytes.
Se pide:

a) Indique el número de líneas de esta memoria.
b) Indique la tasa de aciertos que se produce en la ejecución del fragmento de código anterior teniendo en cuenta solo accesos a los vectores A, B y C.

Ejercicio 4 (3 puntos). Dado el procesador WepSIM con la siguiente estructura:



Este procesador dispone de una Unidad de control representada por la siguiente figura:



Se pide:

a) Especifique las operaciones elementales y señales de control necesarias para ejecutar la instrucción máquina `strnchr Ra Rb Rc`. Esta instrucción recibe en el registro indicado por `Rb` la dirección no nula de una cadena de caracteres terminada en el valor cero y en el registro indicado por `Rc` el código ASCII de un carácter. La instrucción deberá devolver en `Ra` el número de ocurrencias de dicho carácter en la cadena. El único registro que la instrucción modifica es el indicado por `Ra`.

b) Especifique las operaciones elementales y señales de control necesarias para el *fetch*.

NOTA: Asuma que R29 actúa como puntero de pila, que el puntero de pila apunta a cima de pila y que la pila crece hacia direcciones decrecientes de memoria.

Ejercicio 5 (1 punto). La empresa para la que trabaja está desarrollado un nuevo proyecto de bio-sensor "inteligente" para una pulsera que mide el número de latidos del corazón. Hay un prototipo de dispositivo y el módulo de E/S asociado que emplea una arquitectura MIPS 32 con mapa de entrada y salida conjunto y técnica de entrada/salida programada.

El módulo de entrada/salida dispone de tres registros de 32 bits:

- **Registro de datos** (con dirección 0x204). Almacena el número de latidos leídos.
- **Registro de control** (con dirección 0x208). Cuando se escribe en el registro el valor 0x123 se enciende el sensor. Cuando se escribe el valor 0x321 se apaga. Cuando se escribe el valor 0x111, se solicita que el sensor realiza una medición.
- **Registro de estado** (con dirección 0x200). Si el valor del registro es 0, no se ha completado ninguna medición. Si el valor es 1, se ha completado una medición y el controlador dispone del valor en el registro de datos. Si el valor es -1, se ha producido un error y ha de volver a realizarse la medida.

Dicha empresa le pide que desarrolle una rutina en ensamblador del MIPS 32 que se encargue de encender el sensor y realizar de forma indefinida la lectura del número de latidos. Si la lectura es correcta entonces se imprime el valor por pantalla (usando el servicio syscall con código 1). En caso contrario, se continúa realizando la siguiente medida.

Guía rápida del ensamblador MIPS32

Instrucciones de bifurcación y salto

b etiqueta	Bif. incondicional a la instrucción que está en etiqueta.
beq Rsrc1, Src2, etiqueta	Bif. condicional si Rsrc1 es igual a Src2.
beqz Rsrc, etiqueta	Bif. condicional si el registro Rsrc es igual a 0.
bge Rsrc1, Src2, etiqueta	Bif. condicional si el registro Rsrc1 es mayor o igual a Src2 (con signo).
bgt Rsrc1, Src2, etiqueta	Bif. condicional si el registro Rsrc1 es mayor que Src2 (con signo).
ble Rsrc1, Src2, etiqueta	Bif. condicional si Rsrc1 es menor o igual a Src2 (con signo).
blt Rsrc1, Src2, etiqueta	Bif. condicional si Rsrc1 es menor que Src2 (con signo).
bne Rsrc1, Src2, etiqueta	Bif. condicional si Rsrc1 no es igual a Src2.
bnez Rsrc, etiqueta	Bif. condicional si Rsrc no es igual a 0.

Instrucciones aritméticas del coprocesador de coma flotante

add.s fd, fs, ft	Suma los registros fs y ft y almacena el resultado en fd (float)
add.d fd, fs, ft	Suma los registros fs y ft y almacena el resultado en fd (double)
div.s fd, fs, ft	Divide fs entre ft y deja el resultado en fd (float)
div.d fd, fs, ft	Divide fs entre ft y deja el resultado en fd (double)
mul.s fd, fs, ft	Multiplica los registros fs y ft y deja su resultado en fd. (float)
mul.d fd, fs, ft	Multiplica los registros fs y ft y deja su resultado en fd. (double)

Instrucciones de transferencia de datos entre registros

move Rdest, Rsrc	Mueve el contenido del registro Rsrc al registro Rdest (del procesador general)
mfc1 Rdest, CPsrc	Mueve el contenido del registro CPsrc del coprocesador en coma flotante al registro de la CPU Rdest.
mtc1 Rsrc, CPdest	Mueve el contenido del registro Rsrc de la CPU al registro Cpdest del coprocesador en coma flotante.
mov.s fd, fs	Mueve el contenido del registro fs al registro fd. (float)

Instrucciones de comparación

En todas las instrucciones siguientes, Src2 puede ser un registro o un valor inmediato (de 16 bits).

seq Rdest, Rsrc1, Src2	Pone Rdest a 1 si Rsrc1 y Src2 son iguales, en otro caso pone 0.
sge Rdest, Rsrc1, Src2	Pone Rdest a 1 si Rsrc1 es mayor o igual a Src2, y 0 en otro caso (para números con signo).
sgeu Rdest, Rsrc1, Src2	Pone Rdest a 1 si Rsrc1 es mayor o igual a Src2, y 0 en otro caso (para números sin signo).
sgt Rdest, Rsrc1, Src2	Pone Rdest a 1 si Rsrc1 es mayor que Src2, y 0 en otro caso (para números con signo).
sgtu Rdest, Rsrc1, Src2	Pone Rdest a 1 si Rsrc1 es mayor que Src2, y 0 en otro caso (para números sin signo).
sle Rdest, Rsrc1, Src2	Pone Rdest a 1 si Rsrc1 es menor o igual a Src2, en otro caso pone 0 (para números con signo).
sleu Rdest, Rsrc1, Src2	Pone Rdest a 1 si Rsrc1 es menor o igual a Src2, en otro caso pone 0 (para números sin signo).
slt Rdest, Rsrc1, Src2	Pone Rdest a 1 si Rsrc1 es menor a Src2, en otro caso pone 0 (para números con signo).
sltu Rdest, Rsrc1, Src2	Pone Rdest a 1 si Rsrc1 es menor a Src2, en otro caso pone 0 (para números sin signo).
sne Rdest, Rsrc1, Src2	Pone Rdest to 1 si el registro Rsrc1 no es igual a Src2 y 0 en otro caso.

Soluciones

Ejercicio 1.

El valor entero $2^{30}+7$ en binario es 10000000000000000000000000000111. Para almacenar este valor entero en variables de tipo `float` (estándar IEEE 754 de simple precisión) y `double` (doble precisión) es necesario, en primer lugar, proceder a su normalización.

$10000000000000000000000000000111 = 1.00000000000000000000000000000111 \times 2^{30}$

A continuación, hay que eliminar el bit implícito (situado a la izquierda de la coma) y almacenar la parte fraccionaria en la mantisa de la variable de tipo `float` y `double`. Una variable de tipo `float` (simple precisión) tiene una mantisa de 23 dígitos, por tanto, solo podrá almacenar los 23 dígitos más representativos de la parte fraccionaria. Es decir, la mantisa almacenada será 00000000000000000000. El error que se comete al almacenar el valor $2^{30}+7$ en una variable de tipo `float` es 7. En el caso de una variable de tipo `double`, la mantisa es de 52 dígitos y no se perderá ningún dígito de la parte fraccionaria del número. En este caso el error será 0.

Ejercicio 2

a) Un posible pseudocódigo para la rutina convertir podría ser el siguiente:

```
v0 = 0;  // número de elementos cambiados
v1 = 0;  // número de elementos con valor 0
for (i = 0; i < M; i ++) {
     for (j=0; j <M; j++) {
          b = Mat[i][j];
          if ( b < -1) {
               b = b * -1;
               Mat[i][j] = b;
               v0++;
          }
          if  (b == 0)
               v1++;
     }
}

Convertir:
          li   $v0, 0
          li   $v1, 0
          li   $t0, 0    // i
          li   $t2, -1
bucle1:   bge  $t0, $a1, fin1
          li   $t1, 0    // j
bucle2:   bge  $t1, $a2, fin2
          lw   $t3, ($a0)
          bneqz$t3, no0
          addi $v1, $v1, 1
no0:      bgt  $t3, $zero, positivo
          addi $v0, $v0, 1
          mul  $t3, $t3, $t2
          sw   $t3, ($a0)
positivo: addi a0, $a0, 4
          addi $t1, $t1, 1
          b    bucle2
fin2:     addi $t0, $t0, 1
          b    bucle1
fin1:     jr   $ra
```

b)

```
la   $a0, Matriz
li   $a1, 3
li   $a2, 4
jal  Matriz
```

Ejercicio 3

a) La caché tiene un tamaño de 64 KB = 2^{16} bytes. Las líneas son de 64 bytes, por tanto, el número de líneas es de $2^{16} / 2^6 = 2^{10} = 1024$.

b) En cada línea de caché de 64 bytes se pueden almacenar 64/8 = 8 elementos de cada vector (elementos de tipo `double` ocupan 8 bytes).

Se va a describir el patrón de accesos:

Iteración i = 0;

- Lectura de A[0] — Fallo — se trae a cache A[0]… A[7]
- Lectura de A[0] — Acierto
- Lectura de B[0] — Fallo — se trae a cache B[0]… B[7]
- Escritura en C[0] — Fallo — se trae a cache C[0]… C[7]

Iteración i = 1;

- Lectura de A[0] — Acierto
- Lectura de A[0] — Acierto
- Lectura de B[0] — Acierto
- Escritura en C[0] — Acierto

Cada 8 iteraciones hay 32 accesos a los vectores (2 lecturas de A, 1 lectura de B y una escritura en C). Solo se producen 3 fallos en la primera iteración de cada 8. Por tanto el número de accesos es 32 y el número de aciertos es 29. La tasa de aciertos es 29/32 = 0,90625. Es decir, la tasa de aciertos es del 90,625%

Ejercicio 4

a)

Operaciones elementales	Señales de control
RT2 <- SR	T8, C5
RT3 <- RT2	MR=1, SelA=<R0>, MA=0, MB=01, MC=1, SelCOP=<+>, C6
R1 <- R0	MR=0, SelA=<R1>, MA=0, SelB=<R1>, MB=0, MC=1, SelCOP=<->, T6, SelC=<R1>, LC=1
{MAR, RT1} <- R2	MR=0, SelA=<R2>, T9, C0, C4
buc1: MBR <- Mem[RT1]	Ta, BW=0, R, M1, C1
RT2 <- MBR	T1, C5
RT2 + 0 -> flags	MR=1, SelA=<R0>, MA=0, MB=01, MC=1, SelCOP=<+>, SelP=11, M7, C7
bZ fin1	C=6, B=0, A0=0, MADDR=fin1
R3 – RT2 -> flags	MR=0, SelA=<R3>, MA=0, MB=01, MC=1, SelCOP=<->, SelP=11, M7, C7
{MAR, RT1} <- RT1 + 1	MA=1, MB=11, MC=1, SelCOP=<+>, T6, C0, C4
bnZ buc1	C=6, B=1, A0=0, MADDR=buc1
R1 <- R1 + 1	MR=0, SelA=<R1>, MA=0, MB=11, MC=1, SelCOP=<+>, LC, T6, SelC=<R1>
b buc1	C=0, B=1, A0=0, MADDR=buc1
fin1: SR <- RT3	T7, M7=0, C7
Salto a fetch	C=0, B=1, A0=0, MADDR=fetch

b)

Operaciones elementales	Señales de control
MAR <- PC	T2, C0
M[MAR] -> MBR, PC <- PC+4	Ta, R, BW=11, M1=1, C1, M2=1, C2
MBR -> RI	T1, C3
Decodificar y salto a c.o.	C=0, B=0, A0=1

Ejercicio 5

Una posible solución sería:

```
sensor:
              # encender el hardware
   encender: li    $t0, 0x123
             sw    $t0, 0x208
      wait1: lw    $t0, 0x200
             beqz  $t0, wait1

            # pedir medida
    medida:  li   $t1, 0x111
             sw   $t1, 0x208
     wait2:  lw   $t0, 0x200
             beqz $t0, wait2

             # si error…
            li    $t2, -1
             beq  $t0, $t2, medida

             # imprime el valor leído y volver a leer
             lw   $a0, 0x204
             li  $v0  1
            syscall

            b medida
```