ESTRUCTURA DE COMPUTADORES

GRADO EN INGENIERÍA INFORMÁTICA

UNIVERSIDAD CARLOS III DE MADRID

Grupo de Arquitectura de Computadores

Práctica 1

Introducción a la programación en C y lenguaje ensamblador

Curso 2016/2017

Contenido

Objetivos de la práctica 3

Ejercicio 1 4

Ejercicio 2 6

Aspectos importantes a tener en cuenta 7

- **Normas generales** 7
- **Códigos de la práctica** 7
- **Memoria de la práctica** 8

Procedimiento de entrega de la práctica 9

Evaluación de la práctica 10

Descripción del simulador QTSPIM 11

Objetivos de la práctica

El objetivo de la práctica consiste en entender el formato de representación de los números en coma flotante en el computador. Para ello se propone el desarrollo de una serie de programas en lenguaje C y ensamblador del MIPS32.

Para el desarrollo de la práctica es necesario que el alumno repase:

- Los principales aspectos del lenguaje de programación C:

  Material disponible en aula global.

- Los diferentes operadores que ofrece C para trabajar con bits. En el siguiente enlace se puede obtener información sobre los diferentes operadores de bit de C.

  http://en.wikipedia.org/wiki/Bitwise_operations_in_C

- La representación de números en coma flotante según el estándar IEEE754 de precisión simple.
- Los principales aspectos del ensamblador de MIPS 32 y el simulador QTSPIM utilizado para el desarrollo de la parte de ensamblador.

Ejercicio 1

Escriba en lenguaje C un programa lea dos matrices cuadradas A y B de números de tipo *float*, y genere a partir de ellas otras dos matrices cuadradas (C y D) de la misma dimensión cuyo contenido se describirá a continuación. El programa, denominado sumarMin, se invocará de la siguiente manera desde la línea de mandatos:

```
sumarMin  N   ficheroA  ficheroB  ficheroC  ficheroD
```

Donde `N` es un número que representa la dimensión de las cuatro matrices y `ficheroA`, `ficheroB`, `ficheroC` y `ficheroD` son ficheros de texto que almacenan cada matriz y tienen el siguiente formato:

```
dato1  dato2  dato3 …….  datoNxN
```

Donde `datoi` almacena un número de tipo *float*. Los valores que puede tener cada dato pueden ser los siguientes:

- 0, 0.0, -0, -0.0 para representar el valor 0.
- NaN, para representar el valor Not a Number.
- +Inf, para representar + infinito.
- -Inf, para representar –infinito.
- Cualquier número de tipo *float* representable (ejemplos 2.3, 2.34e23, etc.). Se asumirá que los números almacenados son representables en una variable de tipo *float*.

Las matrices se almacenan en los ficheros por filas, un dato seguido de otro. En el material de apoyo se proporciona un ejemplo de matriz.

El objetivo es generar dos matrices C y D, de forma que C[i][j] se obtiene de la siguiente manera:

1. Si A[i][j] == ±0 y B[i][j]== ±0, entonces C[i][j] = +0
2. Si A[i][j] == NaN o B[i][j]== NaN, entonces C[i][j] = NaN
3. Si A[i][j] == ±Inf o B[i][j]== ±Inf, entonces C[i][j] = NaN
4. Si A[i][j] almacena un valor no normalizado (se considera que el valor 0 es normalizado) y B[i][j] almacena también un valor no normalizado, entonces C[i][j]=0
5. Si A[i][j] almacena un valor no normalizado (se considera que el valor 0 es normalizado) y B[i][j] almacena un valor normalizado, entonces C[i][j]=B[i][j]
6. Si A[i][j] almacena un valor normalizado (se considera que el valor 0 es normalizado) y B[i][j] almacena un valor no normalizado, entonces C[i][j]=A[i][j]
7. En caso contrario, si A[i][j] almacena un valor normalizado y B[i][j] también (se considera que el 0 está normalizado), C[i][j]=A[i][j] + B[i][j]

La matriz D se obtiene de la siguiente forma: D[i][j]= minimo (A[i][j], B[i][j]), donde mínimo es una función que obtiene el valor mínimo de dos valores de tipo *float*. En caso de que alguno de los dos valores sea NaN, el resultado será también NaN. Si un valor es –Inf y el otro +Inf, se entiende que el valor mínimo es –Inf.

Ha de hacerse un adecuado tratamiento de errores, por ejemplo, que los datos almacenados en el fichero no se correspondan con el formato adecuado, que haya menos o más datos de los indicados según la dimensión de la matriz, etc. El tratamiento consistirá en indicar por pantalla el error detectado y abortar la ejecución del programa.

NOTAS:

- el programa `sumarMin.c` solo podrá hacer uso de las funciones de biblioteca definidas en `stdio.h` y `stdlib.h`.
- A la hora de escribir las matrices en los ficheros C y D, basta con imprimirlo con formato de tipo *float* en cualquier caso, independientemente del valor almacenado en el *float*.
- La lectura y escritura de *floats* en fichero se realiza con las funciones de biblioteca *fscanf* y *fprintf*, cuyo funcionamiento es similar a *scanf* y *printf*.
- La escritura de los valores de tipo *float* al archivo utilizando *fprintf* debe hacerse con el especificador de formato %g.

Ejercicio 2

El objetivo de este ejercicio es realizar el mismo ejercicio anterior pero en este caso utilizando el ensamblador del MIPS32 y el simulador QTSPIM. En este caso el programa no leerá las matrices de ficheros, sino que su contenido estará indicado en el segmento de datos del programa. Por ejemplo:

```
.data:

   .align 2

   dimension:  .word 3
   matrizA:
             .float 0.0, -2.3, 4.5
             .float N
aN, +Inf, -Inf
             .float 3.4, 6.7, 9.9

   matrizB:
             .float 0.0, -2.3, 4.5
             .float NaN, +Inf, -Inf
             .float 3.4, 6.7, 9.9

   matrizC: .space 36

   matrizD: .space 36

.text:
```

Para realizar este ejercicio tendrá que desarrollarse obligatoriamente, al menos, las dos siguientes subrutinas aparte del `main`:

- `sumarMin`: aceptará como parámetros de entrada la dimensión de las matrices y las direcciones de memoria donde se almacena cada una de las matrices. La función devolverá 0 si no ha habido ningún error en el procesamiento de las matrices y -1 en caso contrario.
- `minimoFloat`: acepta como parámetros de entrada dos valores de tipo *float* y devuelve como resultado un valor de tipo *float*. La función calcula el mínimo de dos valores de acuerdo a la definición de mínimo que se indicó en el ejercicio 1.

Deberá seguirse estrictamente el convenio de paso de parámetros. En caso de que la práctica no incluya al menos estas 2 subrutinas y no se siga el convenio de paso de parámetros, la práctica estará suspensa. Se valorará muy positivamente que la práctica haga uso de más funciones auxiliares.

NOTA: Tenga en cuenta que en el simulador QTSPIM no se pueden utilizar las constantes NaN, +Inf y –Inf, tal y como se han usado en el segmento de datos anterior. En el segmento de datos a utilizar en realidad en la práctica debe cambiarse estas constantes por valores hexadecimal que representen a un valor NaN, +Inf y –Inf.

Aspectos importantes a tener en cuenta

Normas generales

1) La entrega de la práctica se realizará a través de los entregadores habilitados. No se permite la entrega a través de correo electrónico.
2) La entrega se realizará en el plazo dado por los entregadores. Es posible que para un entregador de Aula Global el fin del plazo para una entrega a las 24:00 termine 10 minutos antes. Revise el soporte de Aula Global.
3) Se prestará especial atención a detectar funcionalidades copiadas entre dos prácticas. En caso de encontrar implementaciones comunes en dos prácticas (o contenidos similares en la memoria), ambas obtendrán una calificación de 0.

Códigos de la práctica

1) El ejercicio desarrollado en C debe compilarse con los flags `-Wall` y `-Werror` del compilador `gcc`. El último *flag* convierte todos los avisos y advertencias en error de compilación. Es decir, de la siguiente forma:

```
gcc -Wall  -Werror   -o <nombre del ejecutable>   <nombre del programa (.c)>
```

2) Todos los ejercicios deben seguir el formato de salida que se pide en cada enunciado. Debe seguirse el formato de fichero definido en el enunciado. En caso contrario la práctica estará suspensa.
3) Los ejercicios que no compilen o que no se ajusten a la funcionalidad y requisitos planteados, obtendrán una calificación de 0.
    - Esto incluye no usar los nombres de variables, funciones o ficheros pedidos.
4) Un programa no comentado, obtendrá una calificación de 0.
5) Hay que tener en cuenta que un programa en C que compile correctamente y sin advertencias (*warnings*) no es garantía de que funcione correctamente. Por ello tendrá que realizar aquellas pruebas que garanticen el correcto funcionamiento de la práctica.
6) Todos los ejercicios han de funcionar en **guernika.lab.inf.uc3m.es** (sistema operativo Linux) con el entorno de desarrollo allí disponible. Se utilizará el compilador **gcc disponible** en este sistema. La versión del lenguaje C a usar será C11 (formalmente conocido como ISO/IEC 9899:2011). Esto quiere decir que todos los alumnos deben asegurarse de que sus ejercicios compilan y ejecutan correctamente en esta máquina, independientemente de que puedan funcionar en otros entornos. Para compilar cada uno de los ejercicios tendrá utilizar el siguiente mandato:

```
gcc -Wall -Werror -std=c11 -o   sumar    sumar.c
```

que permite compilar el programa *sumar.c* y generar el fichero ejecutable de nombre *sumar*.

Memoria de la práctica

1) La memoria (un único documento) tendrá que contener al menos los siguientes apartados:
    - Portada donde figuren los autores (incluyendo nombre completo, NIA y dirección de correo electrónico).
    - Índice de contenidos.
    - Contenidos pedidos en los distintos ejercicios (una sección por ejercicio).
    - Conclusiones y problemas encontrados.
2) **La longitud de la memoria no deberá superar las 15 páginas** (portada e índice incluidos).
3) Al respecto de la posible descripción de los programas pedidos:
    - Se ha de detallar las principales funciones implementadas. La memoria debe describir el comportamiento de los programas, así como las principales decisiones de diseño (adicionalmente se pueden incluir diagramas de flujo, algoritmos, etc.).
    - Se ha de incluir la batería de pruebas utilizadas y resultados obtenidos. Se dará mayor puntuación a pruebas avanzadas, casos extremos, y en general a aquellas pruebas que garanticen el correcto funcionamiento de la práctica en todos los casos.
        - Evite pruebas duplicadas que evalúan los mismos flujos de programa. La puntuación de este apartado no se mide en función del número de pruebas, sino del grado de cobertura de las mismas. Es mejor pocas pruebas que evalúan diferentes casos a muchas pruebas que evalúan siempre el mismo caso.

<u>NOTA:</u> NO DESCUIDE LA CALIDAD DE LA MEMORIA DE SU PRÁCTICA.

Aprobar la memoria es tan imprescindible para aprobar la práctica, como el correcto funcionamiento de la misma. Si al evaluarse la memoria de su práctica, se considera que no alcanza el mínimo admisible, su práctica estará suspensa.

Procedimiento de entrega de la práctica

La entrega de la práctica 1 se realizará de forma electrónica a través de Aula Global

Se habilitarán dos entregadores distintos, uno para el código de los **ejercicios** de este cuaderno y otro para la **memoria** completa. Para ello se habilitará dos enlaces, uno por cada entregador.

La fecha límite de entrega para ambos es el día **6 de noviembre de 2015 a las 23:55 horas.**

Es posible entregar tantas veces como quiera dentro del plazo dado, la única versión registrada de su práctica es la última entregada. La valoración de la práctica es la valoración del contenido de esta última entrega. Revise siempre lo que entregue.

Entregador 1: Se deberá entregar un único archivo comprimido en formato **zip** con el nombre ec_p1_AAAAAAAAA_BBBBBBBBB.zip donde A…A y B…B son los NIA de los integrantes del grupo.

El archivo **zip** debe contener solo los siguientes archivos:

- **sumar.c**
- **sumar.s**
- **memoria.pdf**

Evaluación de la práctica

La evaluación de la práctica se va a dividir en dos partes:

- **Código (7 puntos)**
- **Memoria (3 puntos)**

La puntuación de cada ejercicio será:

- Ejercicio 1 (3.5 *puntos*)
- Ejercicio 2 (3.5 *puntos*)

Si un ejercicio no se entrega su puntuación será 0. Tenga en cuenta que para seguir el proceso de evaluación continua, la nota mínima obtenida en cada práctica debe ser de 2 y la media de las dos prácticas 4.

NOTAS:

1. **Si se detecta un error de concepto grave en la práctica (en cualquier apartado de cualquier ejercicio), la valoración global de toda la práctica será de cero puntos (0 puntos).**
2. **En caso de encontrar implementaciones comunes en dos prácticas (o contenidos similares en la memoria), ambas obtendrán una calificación de 0.**
3. **En caso de encontrarse fragmentos de código obtenidos directamente de Internet, la práctica tendrá una calificación de 0.**

Descripción del simulador QTSPIM

La versión QTSPIM (en su versión para Windows) posee una interfaz similar a la que aparece en la siguiente figura. Hay dos paneles horizontales. El panel de la izquierda contiene los valores de los registros, y el panel de la derecha contiene el código y los datos declarados.

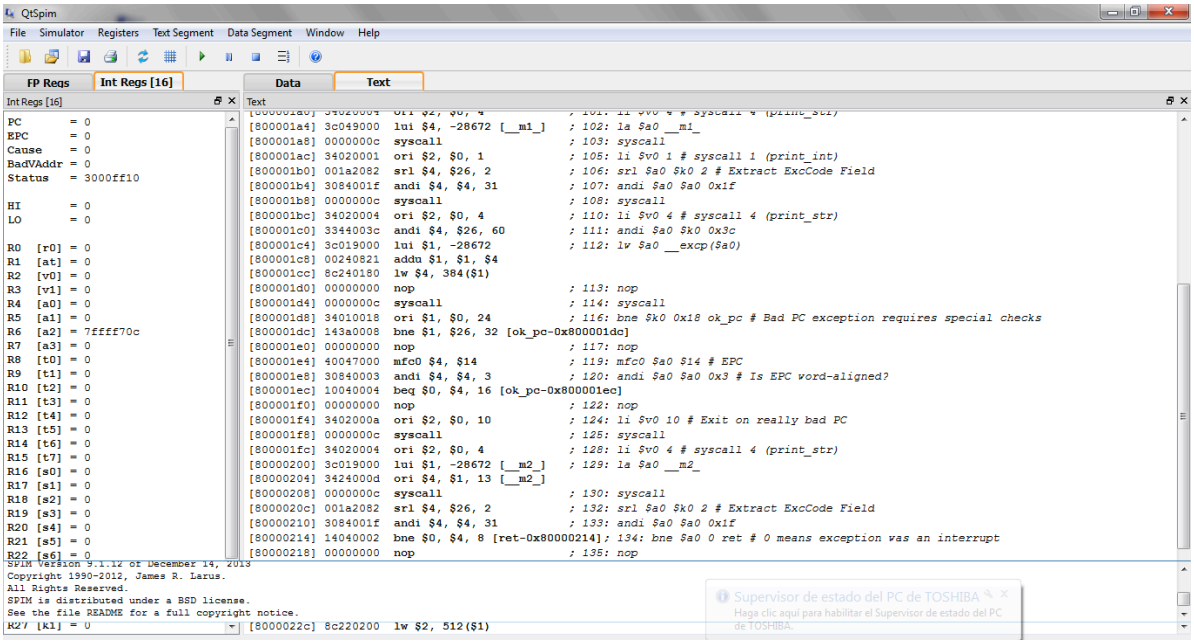
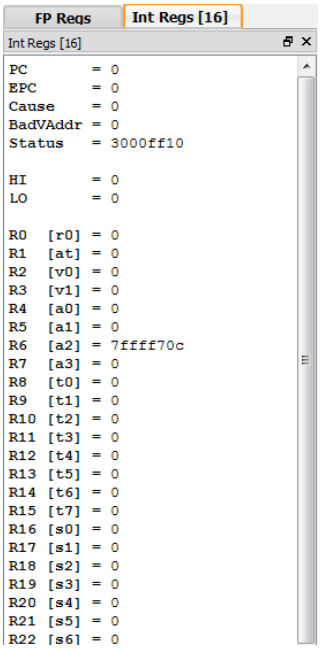


Figura 1. Interfaz del simulador QTSPIM

PANEL 1. Registros de la máquina MIPS.

Incluye el banco de registros del procesador MIPS. En primer lugar aparece el contador de programa (**PC**) y otros registros especiales como **HI** y **LO** utilizados para almacenar los resultados de la multiplicación y división.
En las siguientes líneas aparecerán los 32 registros enteros de propósito general (**R0** a **R31**), junto a su correspondiente etiqueta simbólica.



En la pestaña FP Regs aparecen los 32 registros usados para representar números en coma flotante, tanto en precisión simple, como en precisión doble (**FP0** a **FP31**).

PANEL 2. Segmento de Texto.

Incluye el código del programa ensamblador a ejecutar, identificado mediante la directiva *.text*. Cada instrucción ocupa una única línea, identificada por una dirección de memoria que aparece en la parte izquierda de la línea.

```
Data   Text
Text
[800001a4] 3c049000  lui $4, -28672 [__m1_]   ; 102: la $a0 __m1_
[800001a8] 0000000c  syscall                  ; 103: syscall
[800001ac] 34020001  ori $2, $0, 1            ; 105: li $v0 1 # syscall 1 (print_int)
[800001b0] 001a2082  srl $4, $26, 2           ; 106: srl $a0 $k0 2 # Extract ExcCode Field
[800001b4] 3084001f  andi $4, $4, 31          ; 107: andi $a0 $a0 0x1f
[800001b8] 0000000c  syscall                  ; 108: syscall
[800001bc] 34020004  ori $2, $0, 4            ; 110: li $v0 4 # syscall 4 (print_str)
[800001c0] 3344003c  andi $4, $26, 60         ; 111: andi $a0 $k0 0x3c
[800001c4] 3c019000  lui $1, -28672           ; 112: lw $a0 __excp($a0)
[800001c8] 00240821  addu $1, $1, $4
[800001cc] 8c240180  lw $4, 384($1)
[800001d0] 00000000  nop                      ; 113: nop
[800001d4] 0000000c  syscall                  ; 114: syscall
[800001d8] 34010018  ori $1, $0, 24           ; 116: bne $k0 0x18 ok_pc # Bad PC exception requires special checks
[800001dc] 143a0008  bne $1, $26, 32 [ok_pc-0x800001dc]
[800001e0] 00000000  nop                      ; 117: nop
[800001e4] 40047000  mfc0 $4, $14             ; 119: mfc0 $a0 $14 # EPC
[800001e8] 30840003  andi $4, $4, 3           ; 120: andi $a0 $a0 0x3 # Is EPC word-aligned?
[800001ec] 10040004  beq $0, $4, 16 [ok_pc-0x800001ec]
[800001f0] 00000000  nop                      ; 122: nop
[800001f4] 3402000a  ori $2, $0, 10           ; 124: li $v0 10 # Exit on really bad PC
[800001f8] 0000000c  syscall                  ; 125: syscall
[800001fc] 34020004  ori $2, $0, 4            ; 128: li $v0 4 # syscall 4 (print_str)
[80000200] 3c019000  lui $1, -28672 [__m2_]   ; 129: la $a0 __m2_
[80000204] 3424000d  ori $4, $1, 13 [__m2_]
[80000208] 0000000c  syscall                  ; 130: syscall
[8000020c] 001a2082  srl $4, $26, 2           ; 132: srl $a0 $k0 2 # Extract ExcCode Field
[80000210] 3084001f  andi $4, $4, 31          ; 133: andi $a0 $a0 0x1f
[80000214] 14040002  bne $0, $4, 8 [ret-0x80000214]; 134: bne $a0 0 ret # 0 means exception was an interrupt
[80000218] 00000000  nop                      ; 135: nop
```

La primera dirección de memoria utilizada para almacenar el texto de un programa es 0x00400000.

PANEL 3. Segmento de Datos y pila.

Incluye el conjunto de datos definidos en el segmento de datos *.data* del programa a ejecutar. Estos datos se almacenan de manera secuencial en memoria, a partir de la dirección **0x10000000.** A continuación se muestra el contenido de la pila (*stack*) que comienza en la dirección **0x7FFFEFFF**.

```
Data   Text
Data
User data segment [10000000]..[10040000]
[10000000]..[1003ffff]  00000000


User Stack [7ffff704]..[80000000]
[7ffff704]    00000000  00000000  7fffffe1            . . . . . . . . . . . .
[7ffff710]    7fffffb0  7fffff79  7fffff3d  7fffff0c  . . . . y . . . = . . . . . . .
[7ffff720]    7ffffef3  7ffffecf  7ffffebb  7ffffeae  . . . . . . . . . . . . . . . .
[7ffff730]    7ffffe8e  7ffffe5a  7ffffe40  7ffffe29  . . . . Z . . . @ . . . ) . . .
[7ffff740]    7ffffe1b  7ffffb14  7ffffad6  7ffffabb  . . . . . . . . . . . . . . . .
[7ffff750]    7ffffa9e  7ffffa56  7ffffa44  7ffffa2c  . . . . V . . . D . . . , . . .
[7ffff760]    7ffffa11  7ffff9ed  7ffff9c4  7ffff9a6  . . . . . . . . . . . . . . . .
[7ffff770]    7ffff965  7ffff94e  7ffff93a  7ffff92b  e . . . N . . . : . . . + . . .
[7ffff780]    7ffff915  7ffff8eb  7ffff8c2  7ffff8ab  . . . . . . . . . . . . . . . .
[7ffff790]    7ffff892  7ffff86d  7ffff81d  7ffff80b  . . . . m . . . . . . . . . . .
[7ffff7a0]    7ffff7f3  7ffff7ad  00000000  6e697700  . . . . . . . . . . . . w i n
[7ffff7b0]    73776f64  6172745f  676e6963  676f6c5f  d o w s _ t r a c i n g _ l o g
[7ffff7c0]    656c6966  5c3a433d  42545642  545c6e69  f i l e = C : \ B V T B i n \ T
[7ffff7d0]    73747365  736e695c  6c6c6174  6b636170  e s t s \ i n s t a l l p a c k
[7ffff7e0]    5c656761  6c697363  6966676f  6c2e656c  a g e \ c s i l o g f i l e . l
[7ffff7f0]    7700676f  6f646e69  745f7377  69636172  o g . w i n d o w s _ t r a c i
[7ffff800]    665f676e  7367616c  7700333d  69646e69  n g _ f l a g s = 3 . w i n d i
[7ffff810]    3a433d72  6e69775c  73776f64  39535600  r = C : \ w i n d o w s . V S 9
[7ffff820]    4d4f4330  4f4f544e  633d534c  72505c3a  0 C O M N T O O L S = c : \ P r
[7ffff830]    6172676f  6946206d  2073656c  36387828  o g r a m   F i l e s   ( x 8 6
[7ffff840]    694d5c29  736f7263  2074666f  75736956  ) \ M i c r o s o f t   V i s u
[7ffff850]    53206c61  69647574  2e39206f  6f435c30  a l   S t u d i o   9 . 0 \ C o
[7ffff860]    6e6f6d6d  6f545c37  5c736c6f  45535500  m m o n 7 \ T o o l s \ . U S E
[7ffff870]    4f525052  454c4946  5c3a433d  72657355  R P R O F I L E = C : \ U s e r
[7ffff880]    72415c73  2d736f63  65747345  696e6166  s \ A r c o s - E s t e f a n i
[7ffff890]    53550061  414e5245  413d454d  736f6372  a . U S E R N A M E = A r c o s
[7ffff8a0]    7473452d  6e616665  55006169  44524553  - E s t e f a n i a . U S E R D
```

PANEL 4. Pantalla para depuración de programas

El último panel es el depurador (situado en la parte inferior de la Figura 1), destinado a visualizar mensajes de control o error del simulador QTSPIM.