Práctica.
Ensamblador de MIPS e introducción al simulador SPIM.

El objetivo de esta práctica es que el alumno se familiarice con la programación en ensamblador y la representación de distintos tipos de datos utilizando el ensamblador del MIPS32. Para el desarrollo de esta práctica se usará el emulador QtSpim disponible en:

http://spimsimulator.sourceforge.net

QtSpim es un simulador auto-contenido que ejecuta programas escritos en el lenguaje ensamblador del MIPS32. QtSpim proporciona además un depurador y un conjunto mínimo de servicios del sistema operativo.

1. Ejercicios a desarrollar

A continuación se detallan los ejercicios a desarrollar.

1.1. Entendiendo el simulador QtSpim

Considere el siguiente conjunto de registros y segmento de datos extraído del simulador.

```
       PC = 004000e0 EPC = 00000000 Cause = 00000000 BadVAddr= 00000000
       Status = 3000ff10 HI = 00000000 LO = 00000000

                                    General Registers
R0 (r0) = 00000000 R8 (t0) = 0000006e R16 (s0) = 00000000 R24 (t8) = 00000000
R1 (at) = 10010000 R9 (t1) = 10010008 R17 (s1) = 00000000 R25 (t9) = 00000000
R2 (v0) = 00000004 R10 (t2) = 00000000 R18 (s2) = 00000000 R26 (k0) = 00000000
R3 (v1) = 00000000 R11 (t3) = 00000000 R19 (s3) = 00000000 R27 (k1) = 00000000
R4 (a0) = 10010036 R12 (t4) = 00000000 R20 (s4) = 00000000 R28 (gp) = 10008000
R5 (a1) = 7fffef74 R13 (t5) = 00000000 R21 (s5) = 00000000 R29 (sp) = 7fffef70
R6 (a2) = 7fffef78 R14 (t6) = 00000000 R22 (s6) = 00000000 R30 (s8) = 00000000
R7 (a3) = 00000000 R15 (t7) = 00000000 R23 (s7) = 00000000 R31 (ra) = 00400018

DATA
[0x10000000]...[0x10010000]  0x00000000
[0x10010000]                 0x00000064 0x0000000a 0xffffffff 0x0a0b0c0d
[0x10010010]                 0x00000001 0x00000002 0x00000003 0x0000000b
[0x10010020]                 0x0000000c 0x0000000d 0x00000015 0x00000016
[0x10010030]                 0x00000017 0x69535a41 0x616c756d 0x20726f64
[0x10010040]                 0x4d495053 0x69644100 0x0a00736f 0x20734500
[0x10010050]                 0x20616e75 0x65646163 0x7320616e 0x65206e69
[0x10010060]                 0x6163206c 0x74636172 0x6e207265 0x206f6c75
[0x10010070]                 0x66206c61 0x6c616e69 0x00000000 0x00000000
[0x10010080]...[0x10040000]  0x00000000
```

Responda las siguientes cuestiones, y justifique su respuesta:

a) Identifique el valor del contador de programa y el contenido de la instrucción a la que hace referencia.

b) El registro $**a0** contiene la dirección de memoria de comienzo de una cadena de caracteres. Identifique esta cadena de caracteres en el segmento de datos, indicando cada byte de la cadena y su contenido. ¿Cuál es la ordenación de los bytes utilizado por SPIM? Justifique su respuesta.

c) Localice la dirección de memoria **0x10010000**. Describa el segmento de datos que es necesario escribir en un fichero de ensamblador para lograr la representación en memoria indicada por tal dirección, hasta la dirección **0x10010010**. ¿Cuál es el formato de representación de los números negativos?

d) Indique el valor máximo y mínimo representable en hexadecimal y decimal para cada uno de los siguientes tipos de datos:
   a. .word.
   b. .half.
   c. .byte.

1.2. Detección de errores

El siguiente programa escrito en el ensamblador de MIPS32, tiene como objetivo calcular el valor máximo de un conjunto de datos de tipo entero de longitud 32 bits (.word), es decir, calcular el máximo elemento de un array de números enteros.

```
        .data
a:      .word 36, 20, 27, 15, 1, 62, 41
n:      .word 7
max:    .word 0

        .text
        .globl main
main:
        li $t0, 0           # i in $t0
        li $s0, 0           # max in $s0
        lw $s1, n           # n in $s1

m1:     bge $t0, $s1, m3
        mul $t1, $t0, 1     # scale i
        lw $t2, a($t1)      # load a[i] into $t2
        ble $t2, $s0, m2    # skip "then part" if a[i] <= max
        move $s0, $t2       # "then part": max = a[i]
m2:     addi $t0, $t0, 1    # i++
        b m1
m3:     move $a0, $s0       # end of loop
        li $v0, 1
        syscall
        li $v0, 10
        syscall
```

El código presenta un **único** error que hace que el programa no funcione correctamente. Se pide:

a) Detecte y corrija dicho error para que el programa imprima correctamente el valor máximo del array. Describa el problema y la solución propuesta.

b) Indique para qué se utiliza cada uno de los registros usados en el programa.

c) Al final de cada iteración del bucle identificado por la etiqueta m1, indique el valor de cada uno de los siguientes registros usados. Complete una tabla como la siguiente:

	Iteración 1	Iteración 2	…	Iteración n
R8 (t0) **R9 (t1)** **R10 (t2)** **R16 (s0)** **R17 (s1)**				

1.3. Vectores de números enteros

El siguiente segmento de datos define tres vectores cuyo contenido son datos enteros de distinta longitud: .byte, .word y .half

```
.data

array1: .byte 60, 61, 62, 63   # array1 es una secuencia de bytes
array2: .word -1,-2,-3,-4      # array2 es una secuencia de palabras
array3: .half 10,20,30,40      # array3 es una secuencia de medias palabras
```

Considerando esta descripción de datos, se pide desarrollar los siguientes tres programas:

a) **arrayBytes** recorre el vector referenciado en `array1`, visualiza cada uno de sus elementos como caracteres, y a continuación sobrescribe cada carácter con su mayúscula (si el carácter estuviera en minúscula) o con su minúscula (si estuviera en mayúscula). El programa deberá imprimir el número de caracteres convertidos a mayúsculas y el número de caracteres convertidos a minúsculas.

b) **arrayPalabras** recorre el vector referenciado en `array2`, visualiza cada uno de sus elementos, y a continuación sobrescribe su valor con el valor absoluto de cada elemento. El programa deberá imprimir el número de elementos negativos encontrados en dicho vector.

c) **arrayMediasPalabras** recorre el vector referenciado en `array3`, visualiza cada uno de sus elementos, y a continuación sobrescribe su valor multiplicando cada elemento por la posición que ocupa dentro del array. El programa deberá imprimir el número de medias palabras que podrían haber sido almacenadas como bytes, es decir, el contenido no excede la capacidad de un byte.