ESTRUCTURA Y TECNOLOGÍA DE COMPUTADORES

1º Ingeniero en Informática

EJERCICIOS DE LA PRÁCTICA 1: REPRESENTACIÓN DE LA INFORMACIÓN (TIPOS DE DATOS)

Noviembre de 2008

EJERCICIO 1

Modificar el segmento de datos del programa "codifica.s" (ver anexo) para que contenga los siguientes datos:

a) Las variables **cad1** y **cad2** deben contener los nombres y apellidos respectivos de los dos componentes del grupo. La primera cadena terminará con un espacio en blanco y la segunda con el retorno de carro. Al igual que en el programa original, la primera cadena debe ser de tipo **.ascii** y la segunda de tipo **.asciiz**.

b) Sustituir los valores de los bytes a partir de la etiqueta **ent8** con los siguientes valores: Día de nacimiento del primer componente del grupo, el mismo valor pero negativo, día de nacimiento del segundo componente del grupo, el mismo día pero negativo, y el valor decimal 199. Por ejemplo, si el primer alumno nación un 21/02/1982 y el segundo un 12/10/1981, habrá que colocar la secuencia **21, -21, 12, -12, 199**.

c) Sustituir los valores de tipo **.half** con la siguiente secuencia de valores: Repetir los cinco valores anteriores, y a continuación poner esos mismos valores al cuadrado (pero conservando el signo negativo en cada caso). Para el mismo ejemplo anterior habría que colocar los valores **21, -21, 12, -12, 199, 441, -441, 144, -144, 39601**.

d) Sustituir los valores de tipo **.word** por la misma secuencia de valores que para el tipo **.half**, pero añadiendo al final dos nuevos valores: El producto del primer día de nacimiento por el segundo y por mil, y el mismo valor pero negativo. Para el caso anterior, sería la secuencia **21, -21, 12, -12, 441, -441, 144, -144, 39601, 252000, -252000**.

e) Sustituir los valores de tipo **.float** con el último valor obtenido en el apartado anterior (pero positivo), dividido entre el año de nacimiento del primer alumno (en el caso anterior, sería 252000/1982 = 127.1442987). Después agregar el mismo valor pero negativo, y, por último, añadir el 0. Es decir, la secuencia quedaría, para el ejemplo de antes, así: **127.1442987, -127.1442987, 0.0**.

f) Sustituir los valores de tipo **.double** por los mismos valores de la etiqueta **real32**.

EJERCICIO 2

Cargar el programa en PCSpim y ejecutarlo. Después observar y describir detalladamente el segmento de datos, explicando cómo y dónde se han codificado todos los datos en él contenidos.

Las explicaciones dadas deben ajustarse al siguiente modelo de tabla:

Dato	**Tipo**	**Dirección / Rango Dir.**	**Contenido**	**Comentario**
"Alumno 1 "	ascii	0x10010000	0x41	Código ascii de la "A" ($0x41=65_{10}$)
		0x10010001	0x6C	Código ascii de la "l" ($0x6C=108_{10}$)
		…	…	(avanzamos 7 posiciones más, ya que la cadena "Alumno 1 " consume 9 bytes (1 carácter = 1 byte)
		0x10010008	0x20	Código ascii el espacio en blanco (0x20=32)
(…)	(…)	(…)	(…)	(…)

NOTA: No es necesario comentar los datos de la etiqueta real64.

ANEXO: codifica.s

```
            .data

cad1:       .ascii  "Primer mensaje... "
cad2:       .asciiz "Segundo mensaje.\n"
ent8:       .byte   70,250,-5,0,-128
ent16:      .half   -4000,4000,70,250,-5,128
ent32:      .word   10000000,-10000000,-4000,4000,70,250,-5,0,128
real32:     .float  3.14159,-1234.5678,123.4e-7
real64:     .double 3.14159,-1234.5678,123.4e-7
cr:         .asciiz "\n"

            .text

main:       li $v0,4
            la $a0,cad1
            syscall

            # Impresión del primer byte, el primer half y el
            # primer word, separados por un retorno de carro.

            li $v0,1
            lb $a0,ent8
            syscall
            jal imprime_cr      # El flujo de ejecución se desvía a la dirección
                                # "imprime_cr". Antes, se guarda en $ra el valor actual
                                # del contador de programa (registro PC) más 4 (PC
                                # apuntará a la dirección de la siguiente instrucción).
            li $v0,1
            lh $a0,ent16
            syscall
            jal imprime_cr      # Estamos REUTILIZANDO el código de "imprime_cr"
                                # (motivo de la existencia de los procedimientos).
            li $v0,1
            lw $a0,ent32
            syscall
            jal imprime_cr

            # Impresión de todos los float.

            la $t0,real32       # $t0 <- Dirección de inicio de los float.
            addi $t2,$t0,12     # $t2 <- Dirección final de los float, más 1.
buclefloat: li $v0,2            # Imprimir el dato apuntado por $t0.
            l.s $f12,0($t0)
            syscall
            jal imprime_cr      # Imprimir retorno de carro.
            addi $t0,$t0,4      # $t0 <- $t0 + 4
            bne $t0,$t2,buclefloat

            # Impresión de todos los double.

            la $t0,real64       # $t0 <- Dirección de inicio de los double.
            addi $t2,$t0,24     # $t2 <- Dirección final de los double, más 1.
bucledouble:li $v0,3            # Imprimir el dato apuntado por $t0.
            l.d $f12,0($t0)
            syscall
            jal imprime_cr      # Imprimir retorno de carro.
            addi $t0,$t0,8      # $t0 <- $t0 + 8
            bne $t0,$t2,bucledouble

            j final             # Salto incondicional a la dirección "final", para
                                # así eludir la ejecución de "imprime_cr".

imprime_cr: li $v0,4
            la $a0,cr
            syscall
            jr $ra              # Saltar a la dirección guardada en el registro $ra
                                # (en $ra está la dirección de la instrucción que
                                # sigue al "jal imprime_cr" que desvió la ejecución
                                # a este procedimiento).
final:      li $v0,10
            syscall
```