Examen Parcial de FCO – Temas 5 y 6

17 de Enero 2018

APELLIDOS: _____ NOMBRE: _____

FIRMA:

DNI: _____

Normativa:

- La duración del examen es de 2:00hrs.
- Por favor, escriba su nombre y apellidos en letras **MAYÚSCULAS** y **firme** en **TODAS** las hojas.
- DEBE responder en el espacio asignado.
- No se permiten calculadoras ni apuntes.
- Debe permanecer en silencio durante la realización del examen.
- No se puede abandonar el examen hasta que el profesor lo indique.
- Debe tener una identificación en la mesa a la vista del profesor (DNI, carnet UPV, tarjeta residente, etc.)

1. (1,5 puntos) Complete la tabla siguiente representando los 3 valores numéricos que aparecen, en las representaciones que faltan. Si no puede representar alguno de los valores, escriba FdR. Indique también (en decimal) el rango de representación de cada convenio:

Decimal	**Signo y Magnitud 8 bits**	**Complemento a 2 8 bits**	**Exceso 127 8 bits**
+25	00011001	00011001	10011000
-31	10011111	11100001	01100000
+128	Fuera de rango (FdR)	Fuera de rango(FdR)	11111111
Rango	[-127 , +127]	[-128 , +127]	[-127 , +128]

2. Dados los números enteros A = 01110011_{Ca2} y B = 10110001_{Ca2} representados en complemento a 2 con 8 bits, realice las operaciones siguientes sin cambiar de representación. Indique claramente y justifique si hay o no hay desbordamiento. Muestre el detalle de su solución.

a. (0,4 puntos) A + B

Como se trata de una suma, se suman y se comprueban los dos bits últimos:

```
---11110011
    01110011
  + 10110001
    00100100 ; Suma sin desbordamiento, pues los dos bits últimos de acarreo son iguales
```

b. (0,6 puntos) A - B

Como se trata de una resta, se cambia el signo al sustraendo haciéndole el complemento a dos, y se convierte la resta en una suma:
Ca2(B) = Ca2(10110001) = 01001110 + 1 = 01001111 = -B

Ahora realizamos la suma A + (-B)

```
01111111
 01110011
+01001111
 11000010
```

Los dos últimos bits de acarreo, indicados por el recuadro, son distintos, luego hay desbordamiento.

3. (1 punto) Dado el número real cuya representación en IEEE-754 de simple precisión es la secuencia de bits 0xC26D0000, obtenga su valor en decimal.

1100 0010 0110 1101 0000 0000 0000 0000
Signo =1 (Negativo)
Exp: $10000100_{(exc\ 127)}$-01111111= $00000101_{(2}$ = $5_{(10}$
Mant:0.11011010000000000000000
Bit implicito: **1.**11011010000000000000000x2^5=
111011.01000000000000000 x2^0=

Parte entera: **111011**$_{(2}$= $59_{(10}$
Parte fraccionaria: 0.01000000000000000

0.01=0x2^{-1}+1x2^{-2}= 0.25 Resultado = **-59,25**

4. A partir del siguiente código escrito en lenguaje ensamblador del MIPS R2000:

```
                .data 0x10000000
paswd:          .half 5
secret:         .asciiz "<kmjq\o"
llarg:          .word 8
clar:           .space 8
                .globl __start
                .text 0x00400000
__start:        la $4, secret
                la $5 paswd
                la $6 clar
                lh $7, 0($5)
next:           lb $8, 0($4)
                beq $8, $0, fi
                add $8, $8, $7
                sb $8, 0($6)
                addi $4, $4, 1
                addi $6, $6, 1
                j next
fi:     .end
```

a. (0,5 puntos). Indique el contenido del segmento de datos antes de ejecutarse el programa, teniendo en cuenta que los datos se almacenan en formato "little endian". El contenido debe especificarse por cada byte en hexadecimal para los datos numéricos, y con los caracteres correspondientes entre comillas en el caso de las cadenas. Indique claramente las zonas de memoria de contenido desconocido mediante un interrogante o guión.

31 … 24	23 … 16	15 … 8	7 … 0	Dirección
"k"	"<"	00	05	0x10000000
"\"	"q"	"j"	"m"	0x10000004
¿?	¿?	00(NULL)	"o"	0x10000008
00	00	00	08	0x1000000C
00	00	00	00	0x10000010
00	00	00	00	0x10000014

b. (0,5 puntos). Indique el contenido de los siguientes registros justo después de la primera ejecución de la instrucción `beq $8, $0, fi`. Exprese el contenido en hexadecimal para los registros $4 a $7 y como caracter para el $8.

Registro	Contenido
$4	0x10000002
$5	0x10000000
$6	0x10000010
$7	0x00000005
$8	"<"

c. (0,25 puntos). Convierta a hexadecimal la cadena apuntada por la etiqueta *secret*. Puede utilizar la tabla ascii que se incluye al final del examen.

```
3c 6b 6d 6a 71 5c 6f 00
00("NULL")6f("o")5c("\")71("q")6a("j")6d("m")6b("k")3c("<")
```

d. (1 punto). Indique el contenido del segmento de datos al finalizar completamente la ejecución del programa, teniendo en cuenta que los datos se almacenan en formato "little endian". El contenido debe especificarse por cada byte, en hexadecimal para **todas** las posiciones de memoria, incluso aquellas que almacenan caracteres. Indique claramente las zonas de memoria de contenido desconocido mediante un interrogante o guión.

31 … 24	23 … 16	15 … 8	7 … 0	Dirección
“k”	“<”	00	05	0x10000000
“\”	“q”	“j”	“m”	0x10000004
¿?	¿?	00(NULL)	“o”	0x10000008
00	00	00	08	0x1000000C
6f(“o”)	72(“r”)	70(“p”)	41(“A”)	0x10000010
00	74(“t”)	61(“a”)	76(“v”)	0x10000014

e. (0,25 puntos). Convierta a caracteres la cadena almacenada en la dirección apuntada por la etiqueta *clar*. Puede utilizar la tabla ascii que se incluye al final del examen.

Aprovat

41(“A”)70(“p”)72(“r”)6f(“o”)76(“v”)61(“a”)74(“t”)

f. (1 punto). Indique el contenido de los siguientes registros al finalizar completamente la ejecución del programa. Exprese el contenido en hexadecimal para todos los registros.

Registro	Contenido
$4	0x10000009
$5	0x10000000
$6	0x10000017
$7	0x00000005
$8	0x00000000 o NULL

g. (0,5 puntos). Indique la secuencia de instrucciones en que se traduce la pseudoinstrucción `la $6 clar`.

```
lui $1, 0x1000 (= lui $1, 4096)
ori $6, $1, 0x10 (= ori $6, $1, 16)
```

h. (1 punto). Codifique la instrucción `j next`. Indique el resultado en binario y hexadecimal y detalle los pasos realizados. Indique, además, la dirección en hexadecimal que representa la etiqueta `next`.

CO = 0x02;
Dirección de salto (etiqueta “next”)=**0x00400018**

Eliminamos los 4 bits de la parte alta y dos bits de la parte baja:
~~0000~~ 0000 0100 0000 0000 0000 0001 10~~00~~
Añadimos el código de operación en la parte alta 6 bits:
0000 1000 0001 0000 0000 0000 0000 0110
Y representamos en hexadecimal.
0x08100006

5. **(1,5 puntos).** Escriba las líneas de código de un programa en lenguaje ensamblador del MIPS R2000 que presente el mismo comportamiento que el pseudocódigo siguiente:

```
if (largo < 60)
 {
  entra = 1;
 }
else
 {
  entra = 0;
 }
largo++;
```

Tenga en cuenta las siguientes consideraciones:

Las variables *largo* y *entra* se definirán como enteros de 32 bits ubicados a partir de la posición de memoria 0x1001000. Los valores iniciales usados en la declaración de las variable largo y entra debe ser -1.

Tanto la variable *largo* cómo *entra* deben actualizarse en memoria al finalizar la ejecución del código.

ATENCIÓN. Existen múltiples soluciones correctas a este ejercicio.
En la forma en que se evalúa la expresión "(largo < 60)" hay tres opciones posibles.
Opción 1: slti $8,$7,60
Opción 2: ori $9, $0, 60 y luego slt $8,$7,$9
Opción 3: declarar una variable .word 60 y cargarla en $9

También pueden implementar el if/else usando beq o bne después de la comparación.

```
.data 0x10001000
largo:     .word -1
entra:     .word -1

.globl __start
.text 0x00400000
__start:
la $4,largo
la $5,entra
lw $7,0($4) # $7 <- valor de largo -1
lw $8,4($4) # $8 <- entra = -1, valor inicial
slti $8,$7,60 # compara largo<cota_largo, si=>entra="1"en $8
addi $7,$7,1 # incrementa largo=largo+1
la $4,largo # carga dirección de largo
sw $7,0($4) # guarda nuevo largo, sobre dato largo
```

```
sw $8,4($4) # guarda nuevo entra, sobre dato entra
.end
```