

1º de Grado en Ingeniería Informática
Final de junio de Fundamentos de Computadores - 17 de julio de 2019

Apellidos: _____

Grupo: _____

Nombre: _____

DNI: _____

Instrucciones para realizar el examen

- El tiempo total disponible es de 3 horas (desde el inicio del examen).
- Las preguntas se contestarán en folios.
- No olvide poner los apellidos y el nombre tanto en la hoja de examen como en los folios entregados.
- Entregad tanto el enunciado del examen como los folios utilizados o no al acabar el examen.

Parte II: cuestiones teórico-prácticas (30%; puntuación indicada en cada apartado)

C1. (1 punto) Responda a las siguientes cuestiones:

- a) Realiza la operación $(-18)+(-7)$ (ambos números expresados en decimal) empleando la representación en complemento a 2 (C2). Para ello, determina primero el **mínimo número de bits** que han de utilizarse para dicha representación sin tener problemas de desbordamiento, comprobando la corrección del resultado (interpretando la ristra de bits del resultado también en C2).
- b) Pone una ristra de ejemplo para cada uno de los tres siguientes casos especiales
- b1) Más infinito.
 - b2) Menos infinito.
 - b3) *Not a Number*.
 - b4) Un número desnormalizado cualquiera.

C2. (1 punto) Responda a las siguientes preguntas:


- a) Define los conceptos de interrupción, excepción y llamada al sistema, señalando sus diferencias y poniendo al menos un ejemplo de cada una de ellas.
- b) Explica brevemente los diferentes pasos que ocurren cuando un proceso de usuario solicita leer datos de un dispositivo de E/S.

C3. (1 punto) Responda a las siguientes cuestiones:

- a) Indica las capas en las que se divide la típica arquitectura de internet y pon un ejemplo de protocolo usado en cada una de ellas.
- b) Indica brevemente la diferencia entre los siguientes protocolos de envío y recepción de correo.
- b1) SMTP
 - b2) POP3
 - b3) IMAP
 - b4) HTTP

Parte III: ejercicios boletines (40%; puntuación indicada en cada apartado)

P1. (1 punto) Dado el siguiente pantallazo de *okteta*:



Dicho fichero contiene, por este orden, los siguientes datos, codificados según el esquema **big-endian**:

- a) un entero sin signo de 16 bits
- b) un entero con signo de 16 bits
- c) un entero con signo de 32 bits
- d) un número real representado mediante IEEE 754 de simple precisión
- e) una cadena de 4 caracteres

Para los apartados a) al d), se pide: el desplazamiento donde empieza, los bytes que lo codifican y el valor numérico representado (en decimal). No se considerará válido dar únicamente el resultado sino que es necesario mostrar todos los pasos del proceso de decodificación a decimal.

P2. (1 punto) Dada la siguiente función expresada en su forma canónica (en la que el término $d(2, 10, 14)$ significa que no importa la salida de la función para los minterminos 2, 10 ni 14):

$$F(A, B, C, D) = \sum \min(0, 5, 7, 8, 13, 15) + d(2, 10, 14)$$

- a) Implementa el circuito usando solo puertas NAND con el mínimo número de éstas (considera que se dispone de las variables de entrada ya negadas).
- b) Implementa la función anterior usando un multiplexor 8 a 1 (8 entradas de datos, tres de control y una salida para el dato seleccionado) y las puertas lógicas adicionales necesarias (se dispone también de las variables A, B, C y D negadas). Deja bien indicado el subíndice de cada una de las entradas de datos y de control del multiplexor.

P3. (0.75 puntos) Considera la siguiente sesión de terminal en una máquina Linux:

```
user@host:~$ ifconfig
eth0 Link encap:Ethernet  direcciónHW 00:1d:7d:ab:0d:3e
Direc. inet:155.54.205.34  Difus.:[...] Másc:255.255.254.0
[...]
user@host:~$ nmcli
[...]
DNS configuration:
servers: 155.54.1.10 155.54.1.1 155.54.1.2
domains: inf.um.es
interface: enol
user@host:~$ host www.cocacola.es
www.cocacola.es has address 82.144.108.198
```

- a) ¿Cuántas tarjetas de red pueden conectarse como máximo a la subred a la que pertenece este host? Indica la identificación exacta de dicha subred, en formato X.Y.Z.W/V. ¿Cuál es la dirección de *broadcast* de dicha red?
- b) ¿Qué comando deberíamos utilizar para averiguar la IP de nuestro router? ¿Sería la IP 155.54.205.0 una dirección válida para dicho router? Razona en una sola frase la respuesta.
- c) ¿Qué protocolo utiliza el último comando (*host*)? Da las siglas correspondientes, su significado, y una breve explicación de para qué sirve ¿Qué dirección IP exacta tiene la máquina servidora a la que se conecta dicho comando, en este caso? ¿Qué indica exactamente en este caso la respuesta del comando?

P4. (1,25 punto) Considera la siguiente sesión interactiva con el depurador *gdb* (los puntos suspensivos [...] indican simplemente que se ha suprimido la parte de la salida que no nos interesa para este ejercicio; aparecen subrayados los comandos tecleados por el usuario; el resto es la salida producida por el terminal):

```
pedroeb@borges:~/ $ gdb ./main
[...]
(gdb) l main
1      #include<stdio.h>
2      int array[10] = {1,-2,3,-4,5,-6,7,-8,9,-10};
3      int main() {
4          int i;
5          for(i=9;i>=1;i=i-2)
6              array[i] = -array[i];
7          for(i=0;i<10;i++)
8              printf("%d ",array[i]);
9          printf("\n");
10     }
(gdb) disassemble main
[...]
0x00000000040055c <+15>: jmp      0x40057e <main+49>
0x00000000040055e <+17>: mov      -0x4(%rbp),%eax
0x000000000400561 <+20>: cltq
0x000000000400563 <+22>: mov      0x601060(,%rax,4),%eax
0x00000000040056a <+29>: neg      %eax
0x00000000040056c <+31>: mov      %eax,%edx
0x00000000040056e <+33>: mov      -0x4(%rbp),%eax
0x000000000400571 <+36>: cltq
0x000000000400573 <+38>: mov      %edx,0x601060(,%rax,4)
0x00000000040057a <+45>: subl     $0x2,-0x4(%rbp)
0x00000000040057e <+49>: cmpl     $0x0,-0x4(%rbp)
0x000000000400582 <+53>: jg       0x40055e <main+17>
[...]
(gdb) x/40bx array
0x601060 <array>:      0x01 0x00 0x00 0x00 0xfe 0xff 0xff 0xff
```

```
0x601068 <array+8>:      0x03 0x00 0x00 0x00 0xfc 0xff 0xff 0xff
0x601070 <array+16>:     0x05 0x00 0x00 0x00 0xfa 0xff 0xff 0xff
0x601078 <array+24>:     0x07 0x00 0x00 0x00 0xf8 0xff 0xff 0xff
0x601080 <array+32>:     0x09 0x00 0x00 0x00 0xf6 0xff 0xff 0xff
(gdb) x/32bx 0x40055c
0x40055c <main+15>:      0xeb 0x20 0x8b 0x45 0xfc 0x48 0x98 0x8b
0x400564 <main+23>:      0x04 0x85 0x60 0x10 0x60 0x00 0xf7 0xd8
0x40056c <main+31>:      0x89 0xc2 0x8b 0x45 0xfc 0x48 0x98 0x89
0x400574 <main+39>:      0x14 0x85 0x60 0x10 0x60 0x00 0x83 0x6d
```

- a) Indica para qué sirven exactamente cada uno de los cuatro comandos subrayados.
- b) ¿Qué parte del código C implementan exactamente las instrucciones en ensamblador mostradas?
- c) ¿Qué contiene exactamente la dirección `-0x4(%rbp)`?
- d) ¿Qué crees que hará la instrucción ensamblador `neg %eax`? ¿Y qué trabajo concreto realiza la secuencia de las tres últimas instrucciones (`subl, cmpl, y jg`)?
- e) ¿Qué valor hay exactamente almacenado en los cuatro bytes de las cuatro direcciones `0x601064-0x601067`? ¿Y qué valor habrá en esos mismos bytes al terminar de ejecutarse el programa?
- f) ¿Qué significan los bytes impresos por el comando `x/32bx 0x40055c`?
- g) ¿Cuáles son las dos instrucciones cuyo código máquina es más largo (en bytes) de todas las instrucciones ensamblador del ejemplo mostrado? (Indica sus códigos en ensamblador completos, así como sus respectivas direcciones de comienzo).
- h) Indica el código máquina concreto (todos los bytes) de la primera instrucción del apartado f).

SOLUCIONES: Parte II: cuestiones teórico-prácticas

C1:

a) 6 bits.

101110 (-18)

111001 (-7)

± 100111 (-25)

b1) 0 11111111 000000000000000000000000

b2) 1 11111111 000000000000000000000000

b3) 1 11111111 10101010101010101010101

b4) 1 00000000 000000000000000000000001

C2:

a) Transparencias 55-56 del T4.

b) Transparencia 30 del T4.

C3:

a) Transparencia 16. Tema 6.

b) Transparencias 39-41. Tema 6.

SOLUCIONES: Parte III: ejercicios boletines

P1:

a) FF FF (unsigned short int) → 65535

b) 80 00 (short int) → -32768

c) FF FF 00 00 (int) → -65536

d) 3F C0 00 00 (float) → 1,5

s: 0 (positivo)

e: (8 bits) = 0111 1111 = 127 → E = 127 - 127 = 0

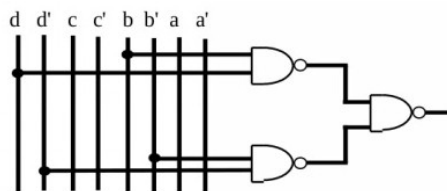
m: 1000 0000... → M = 1.1

N = +1.1₂ · 2⁰ = 1,5

P2: a)

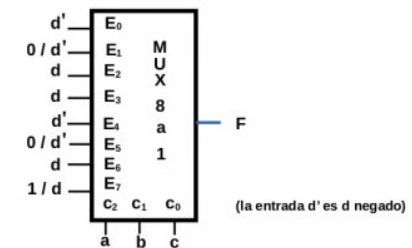
$$F(A,B,C,D) = ((B \cdot D)' \cdot (B' \cdot D'))'$$

		AB			
		00	01	11	10
CD	00	1	0	0	1
	01	0	1	1	0
	11	0	1	1	0
	10	-	0	-	-



P2: b)

		AB			
		00	01	11	10
CD	00	1	0	0	1
	01	0	1	1	0
	11	0	1	1	0
	10	-	0	-	-



P3:

a) 510 / 155.54.204.0 / 155.54.205.255

b) route -n (o el propio nm-tool) / Sí (ya que está dentro de la subred, y no es la de red ni la de broadcast).

c) DNS (Domain Name Server): servicio que traduce IPs a nombres de dominio y viceversa / 155.54.1.1 / La respuesta indica que el host cuyo nombre de dominio es www.cocacola.es tiene la dirección IP 82.144.108.198.

P4:

a)

l_main lista el código de la función main del código de alto nivel escrito en C original.

disassemble main muestra el código ensamblador generado para dicha función.

x/40bx array muestra el contenido de los primeros 40 bytes del array de datos array.

Cada grupo de 4 bytes va representando uno de los valores iniciales 1, -2, 3, -4, etc.

x/24x 0x40055c muestra el código máquina de las mismas instrucciones mostradas por el disassemble main (a partir de la dirección de código 0x40055c).

b) Se trata del código correspondiente al primer bucle (líneas 5 y 6 del ensamblador).

c) La variable local i del programa en C.

d) La instrucción `neg` cambia el signo del valor contenido en el registro `%eax`. La secuencia de tres instrucciones finales realiza el decremento de la variable de control del bucle (`i`) en dos unidades, comprueba si sigue siendo mayor o igual que 1 (o lo que es lo mismo, estrictamente mayor que cero), y en caso afirmativo vuelve a ejecutar el siguiente paso del bucle.

e) Son los bytes `0xfe 0xff 0xff 0xff`, que interpretado como un entero de 32 bits almacenado en *little endian* se corresponden con el valor -2 (segunda posición del array, esto es, `array[1]`). Al terminar el programa, se encontrará en ellos el valor +2, esto es, los bytes `0x02 0x00 0x00 0x00`.

f) Se corresponden al código máquina de la secuencia de instrucciones en ensamblador mostrada (a partir de la dirección de código `0x40055c`).

g) Son estas dos instrucciones (ambas ocupando 7 bytes de código máquina):

```
0x400563: mov 0x601060(,%rax,4),%eax
0x400573: mov %edx,0x601060(,%rax,4)
```

h) El código máquina de la primera instrucción anterior es la siguiente secuencia de 7 bytes: `0x8b 0x04 0x85 0x60 0x10 0x60 0x00`.