1º de Grado en Ingeniería Informática (Grupos 1, 2, 3 y 4)

Examen final de Fundamentos de Computadores (convocatoria de enero)

29 de enero de 2015

EXAMEN COMPLETO (TEMAS 1, 2, 3, 4, 5 y 6)

Apellidos, Nombre:_____ **DNI**:_____ **Grupo(1/2/3/4)**:__

Parte II: cuestiones teórico-prácticas *(35.0%; puntuación indicada en cada apartado)*

C1. *(1.00 puntos)* El formato que vamos a usar en la siguientes cuestión es un formato en coma flotante normalizado IEEE 754 de 8 bits, con 1 bit para el signo, 4 bits para el exponente (sesgo S=7) y 3 bits para la mantisa. Es idéntico al formato de 32 y 64 bits visto en clase en cuanto al significado de los campos y codificación de números especiales. Asume que usamos redondeo al número par más cercano especificado en el estándar IEEE utilizando dos bits. Se pide:

a) La codificación del número $+0{,}011011)_2$ en el formato en coma flotante especificado.

b) La codificación del número $+15{,}0)_{10}$ en el formato en coma flotante especificado.

c) La codificación del número $-1{,}0)_2$ en el formato en coma flotante especificado.

d) Decodifica la ristra de 8 bits 11010101 del formato IEEE 754 de 8 bits especificado a su valor decimal correspondiente.

C2. *(1.00 puntos)* Diseña un circuito combinacional que realice la comparación de dos números binarios A y B, de dos bits cada uno (A_1A_0 y B_1B_0). La salida toma el valor lógico 1 cuando se cumple que A > B, y el valor lógico 0 en caso contrario. Se pide:

a) Tabla de verdad.

b) Función lógica simplificada como suma de productos.

c) Circuito simplificado utilizando un nivel de puertas AND y otro de puertas OR, más la posible negación con puertas NOT en las entradas.

C3. *(0.75 puntos)* Completa los huecos que aparecen en la siguiente tabla de distintas subredes (se puede contestar, si se quiere, en la misma tabla):

IP	*Máscara*	*Subred*	*Broadcast*	Dir *Host* 1º	Dir *Host* último
155.54.1.130	255.255.255.128	155.54.1.128	155.54.1.255	155.54.1.129	155.54.1.254
216.58.210.15	255.255.255.0	216.58.210.0	216.58.210.255		
192.168.4.61	255.255.255.224				192.168.4.62
155.54.60.99	255.255.0.0				

C4. *(0.75 puntos)* Indicar brevemente las similitudes y diferencias que hay entre estos tres posibles comandos de Linux (**Nota.-** No se trata de una secuencia de comandos, se trata de comandos distintos para realizar una tarea deseada de formas alternativas):

a) `cp video.avi video2.avi`

b) `ln video.avi video2.avi`

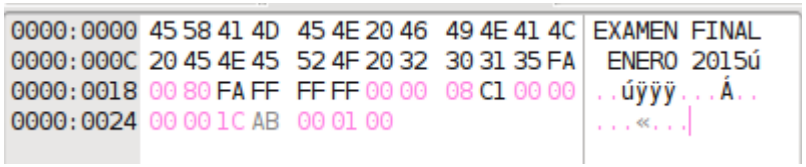
c) `ln -s video.avi video2.avi`.

Indicar también, para cada uno de los tres casos por separado, lo que ocurrirá exactamente si tras ejecutar el comando correspondiente, se ejecuta el comando `rm video.avi`. Suponer que el tamaño del archivo `video.avi` es relativamente grande, p.e. alrededor de un gigabyte, y especificar, para la ejecución particular de cada uno de los tres comandos, qué ocurrirá con el espacio usado en el disco.

Parte III: ejercicios boletines *(35.0%; puntuación indicada en cada apartado)*

P1. *(1.00 puntos)*

a) Dado el siguiente pantallazo de *okteta,* y considerando codificación ***little-endian***, rellenar los huecos del texto que va a continuación:



"En la posición 0000:0016 se sitúa el byte (en hexadecimal) __(a1)__. Considerándolo como un carácter ASCII se corresponderá con el carácter __(a2)__, si lo consideramos como un valor de tipo entero CON signo en C2 será el valor numérico decimal __(a3)__. En la posición 0000:0018 se sitúa un valor entero de 16 bits, cuyo valor en hexadecimal es exactamente ____(a4)____ . Considerándolo como un valor entero SIN signo correspondería al valor decimal __(a5)__, y considerándolo CON signo en C2 al valor decimal __(a6)__. Este número va seguido de otro valor entero considerado CON signo de 32 bits cuyo valor en hexadecimal es __(a7)__, y en decimal __(a8)__. Finalmente, en la posición 0000:001E se sitúa un número real representado mediante *IEEE754 de simple precisión*, siendo su valor **en hexadecimal** __(a9)__. Este número va seguido de otro número representado mediante *IEEE754 de doble precisión*, cuyo **valor hexadecimal** es (a10)____ (nótese que para contestar (a9) y (a10) **no hace falta** convertir a los correspondientes valores reales".

b) Dado un fichero PPM abierto en *okteta*, conociendo que su metainformación de cabecera indica que la anchura es 15 y la altura 100, sabiendo que el último byte de la cabecera acaba en la dirección 0000:004F (inclusive), y suponiendo que las filas y las columnas de la imagen comienzan en la posición (0,0), indique:

b1) La dirección de memoria de comienzo del píxel situado en las coordenadas (fila,columna) = (2,5). (**Nota.-** Suponer aquí que el primer píxel de la imagen tiene coordenadas (0,0)).

b2) Las coordenadas del píxel correspondiente a la dirección 0000:00CE.

P2. *(0.75 puntos)* Rellenar el siguiente texto (indicando en el examen la correspondiente referencia al hueco para cada respuesta):

"Una vez arrancado el ordenador y lanzado el terminal de comandos (`bash`) aparece un *prompt* como el siguiente: `Carlos@linux_casa:~$`. La virgulilla ('~') me indica que estoy en el directorio __(a1)__, pero si quiero asegurarme de cuál es mi directorio actual usaré el comando __(a2)__. Si deseo moverme al directorio `apuntes` que cuelga del directorio actual usaré el comando __(a3)__, y para saber los ficheros y directorios que están en éste, así como sus permisos, tamaño, fecha de modificación etc., usaré el

comando __(a4)__, aunque si lo que me interesa es saber la fecha de creación usaré el comando __(a5)__. De ese mismo directorio cuelga otro directorio que se llama `Fundamentos` y de él cuelgan otros 6 directorios llamados `tema_1`, `tema_2`, etc. Si quiero encontrar, sin moverme de donde estoy, todos los ficheros en formato `pdf` que se llamen `tema1-boletin-practicas.pdf`, `tema2-boletin-practicas.pdf`, etc. hasta `tema4-boletin-practicas.pdf`, sin saber exactamente en qué subdirectorio están (pero bajo `apuntes/Fundamentos`) usaré el comando __(a6)__. Una vez localizados los anteriores ficheros deseo copiarlos en un directorio que aún he de crear que se llamará `examen` (colgará del directorio `apuntes`) y para crearlo utilizo el comando __(a7)__). Si deseo copiar con un sólo comando todo el directorio `Fundamentos` y todo lo que cuelgue de él al recién creado directorio `examen`, entonces utilizaré el comando __(a8)__. Finalmente deseo tener en un fichero de texto llamado `todo.txt` todos los nombres de ficheros (sólo ficheros) que cuelgan desde `Fundamentos`, para ello hago uso del comando con redireccionamiento:__(a9)__."

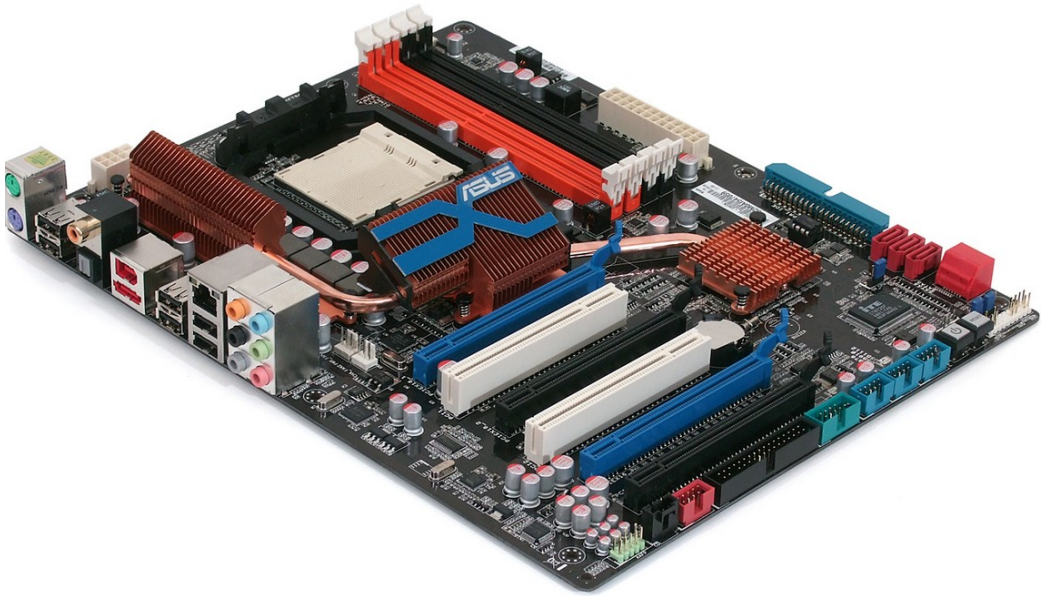
P3. *(1.0 puntos)* Considérese la siguiente sesión `gdb` (los puntos suspensivos `[...]` indican que se ha suprimido la parte de la salida que no nos interesa para el ejercicio):

```
(gdb) l
[...]
2       int array[5] = {100,200,300,400,500};
3       int main() {
4           int i;
5           for(i=4;i>=0;i--)
6               array[i] = funcion(i);
[...]
(gdb) disassemble main
[...]
0x400585 <+8>:        movl   $0x5,-0x4(%rbp)
0x40058c <+15>:       jmp    0x4005ae <main+49>
0x40058e <+17>:       mov    -0x4(%rbp),%eax
[...]
0x400598 <+27>:       callq  0x4005f0 <funcion>
[...]
0x4005aa <+45>:       subl   $0x1,-0x4(%rbp)
0x4005ae <+49>:       cmpl   $0x0,-0x4(%rbp)
0x4005b2 <+53>:       jns    0x40058e <main+17>
[...]
(gdb) x/48bx 0x400585
0x400585 <main+8>:    0xc7  0x45  0xfc  0x05  0x00  0x00  0x00  0xeb
[...]
0x4005ad <main+48>:   0x01  0x83  0x7d  0xfc  0x00  0x79  0xda  0xc7
(gdb) x/20bx array
0x601050 <array>:     0xYY  0x00  0x00  0x00  0xZZ  0x00  0x00  0x00
0x601058 <array+8>:   0x2c  0x01  0x00  0x00  0x90  0x01  0x00  0x00
0x601060 <array+16>:  0xf4  0x01  0x00  0x00
```

En base a dicha sesión, rellenar todos los huecos del texto que va a continuación (indicando en el examen la correspondiente referencia al hueco para cada respuesta):

"El código depurado en la sesión manipula una tabla de __(a1)__ elementos de tipo entero de 32 bits, que ocupa exactamente __(a2)__ bytes en memoria, y que comienza en la dirección exacta __(a3)__. La función `funcion`, por su parte, comienza exactamente en la dirección __(a4)__. Justo al inicio de la ejecución del programa, el valor de byte que aparece sustituido en negrita con **`0xYY`** será, en realidad, el valor __(a5)__ (suponer que el convenio de almacenamiento utilizado es *little endian*), mientras que el valor de **`0xZZ`** será en realidad __(a6)__. El código desensamblado que aparece en la figura se corresponde con un código de alto nivel que __(a7)__ (dar aquí una explicación concisa pero completa de lo que hace dicho código). En particular, la instrucción `movl $0x5,-0x4(%rbp)` dentro de ese código (y sabiendo que en algún lugar previo del programa se puso `%rbp` apuntando al marco de pila de la función `main`), exactamente realiza lo siguiente: __(a8)__, mientras que la instrucción `cmpl $0x0,-0x4(%rbp)` se encarga de: __(a9)__. La instrucción ubicada en la dirección `0x40058c` es de tipo __(a10)__, mientras que la ubicada en la dirección `0x4005b2` es de tipo __(a11)__. Podemos también afirmar que la instrucción `cmpl`, una vez ubicada en memoria, ocupa exactamente __(a12)__ bytes, cuyos valores concretos son __(a13)__ (expresar aquí los bytes correspondientes tal y como aparecen en el listado), que están comprendidos entre la dirección __(a14)__ y la __(a15)__, ambas inclusive."

P4. *(0.75 puntos)* Identifica **15 elementos relevantes** en la siguiente Placa Base . Indicando para cada uno de ellos su **utilidad y significado de las siglas.**



SOLUCIÓN C1:

a) 00101110
b) 01010111
c) 10111000
d) -13,0

SOLUCIÓN C2:

a) Obtenemos la siguiente tabla de verdad:

A_1	A_0	B_1	B_0	S
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	1
0	1	0	1	0
0	1	1	0	0
0	1	1	1	0

A_1	A_0	B_1	B_0	S
1	0	0	0	1
1	0	0	1	1
1	0	1	0	0
1	0	1	1	0
1	1	0	0	1
1	1	0	1	1
1	1	1	0	1
1	1	1	1	0

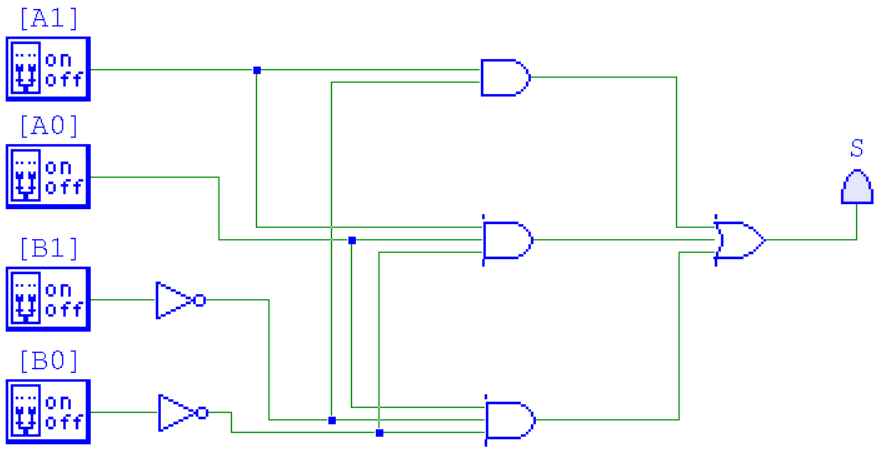
b) La función lógica sin reducir, correspondiente a su forma canónica una vez expresada como suma de productos, es la siguiente:

$$S = \bar{A}_1 A_0 \bar{B}_1 \bar{B}_0 + A_1 \bar{A}_0 \bar{B}_1 \bar{B}_0 + A_1 \bar{A}_0 \bar{B}_1 B_0 + A_1 A_0 \bar{B}_1 \bar{B}_0 + A_1 A_0 \bar{B}_1 B_0 + A_1 A_0 B_1 \bar{B}_0$$

Que por Karnaugh queda simplificada a:

$$S = A_1 \bar{B}_1 + A_1 A_0 \bar{B}_0 + A_0 \bar{B}_1 \bar{B}_0$$

y obtenemos la siguiente implementación con puertas de 2 entradas (se muestra el diagrama lógico que emplea puertas AND, OR y NOT):



SOLUCIÓN C3:

IP	*Máscara*	*Subred*	*Broadcast*	Dir *Host* 1º	Dir *Host* último
155.54.1.130	255.255.255.128	155.54.1.128	155.54.1.255	155.54.1.129	155.54.1.254
216.58.210.15	255.255.255.0	216.58.210.0	216.58.210.255	**216.58.210.1**	**216.58.210.254**
192.168.4.61	255.255.255.224	**192.168.4.32**	**192.168.4.63**	**192.168.4.33**	192.168.4.62
155.54.60.99	255.255.0.0	**155.54.0.0**	**155.54.255.255**	**155.54.0.1**	**155.54.255.254**

SOLUCIÓN C4:

a) El primer comando genera una copia en el disco duro del archivo `video.avi`, llamada `video2.avi`. Dicha copia ocupará exactamente el mismo tamaño que el vídeo original (la entrada al directorio no ocuparía más, excepto si se rebasan los 4KB del directorio donde se hace la copia (*)), duplicándose por tanto el espacio necesario (otro GB adicional). Al tratarse de dos archivos independientes, el borrado de `video.avi` con el correspondiente comando `rm` no tendrá ningún efecto sobre el archivo copiado (se borrará y liberará el espacio ocupado por `video.avi`, mientras que `video2.avi` seguirá perfectamente accesible).

b) El segundo comando, sin embargo, genera una nueva entrada de directorio llamada también `video2.avi`, igual que el comando anterior, y cuyo contenido será exactamente el mismo (es decir, podremos acceder a él exactamente igual que en el caso 1). Pero en este caso el espacio usado en el disco será mucho menor (en realidad casi siempre nada más, pues es espacio que se modifica de los 4KB del directorio, a menos que se sobrepasen esos 4KB), puesto que el contenido del vídeo en sí no estará duplicado. Se trata del denominado *enlace de tipo duro*. Si entonces se borra el archivo original `video.avi` con el comando `rm` no hay ningún problema,

puesto que el archivo `video2.avi` seguirá conteniendo los datos del vídeo original.

c) El tercer caso es similar al segundo, en cuanto a que se genera también una entrada nueva de directorio llamada `video2.avi`, con los mismos contenidos, pero sin duplicar el espacio de disco necesario. Sin embargo, en este último caso, al tratarse de un *enlace de tipo simbólico*, el efecto del comando de borrado `rm` sobre el archivo original `video.avi` será que el contenido del vídeo se borrará irremediablemente del disco duro, quedando el enlace simbólico `video2.avi` "colgando" sin apuntar a nada.

(*) Aunque ésto último en realidad depende de los detalles de implementación de cada sistema de archivos concreto que se pueda estar utilizando (ext3, ext4, FAT, NTFS, etc.).

SOLUCIÓN P1:

a1) 0x35
a2) '5'
a3) +53
a4) 0x8000
a5) 32768
a6) -32768
a7) 0xFFFFFFFA
a8) -6
a9) 0xC1080000
a10) 0x0100AB1C00000000

b1) 0000:00B9
b2) (2,12)

SOLUCIÓN P2:

a1) */home/Carlos*
a2) `pwd`
a3) `cd apuntes`
a4) `ls -l`
a5) `ls -lU`
a6) `find Fundamentos -name "tema[1-4]-boleti*.pdf" (p. ejem.)`
a7) `mkdir examen`
a8) `cp -R Fundamentos examen`
a9) `find /home/Carlos/apuntes/Fundamentos -type f > todo.txt` o
`find ./Fundamentos -type f > todo.txt`

SOLUCIÓN P3:

a1) 5.
a2) 20.
a3) `0x601050`.
a4) `0x4005f0`.
a5) `0x64` (100 en decimal).
a6) `0xc8` (200 en decimal).
a7) Inicia la variable `i` del bucle `for` a 5, en el cuerpo del bucle llama a la función `funcion`, y al final del bucle decrementa la variable `i`, comprueba si es menor que cero, y en caso contrario cierra el bucle.
a8) Inicializa la variable `i` (que está en la pila) con el valor 5.
a9) Compara el valor de la variable `i` (que está en la pila) con el valor 0.
a10) Salto incondicional.
a11) Salto condicional.
a12) 4 bytes.
a13) `0x83 0x7d 0xfc 0x00`.
a14) `0x4005ae`.
a15) `0x4005b1`.

SOLUCIÓN P4:

1.- Puerto PS2 /mini DIN. Conexión de Ratón.
2.- Disipador para los reguladores de tensión de la placa/procesador.
3.-Zócalo para colocar el microprocesador o CPU (Unidad Central de Proceso).
4.- Disipador sobre (NorthBridge/ Puente Norte). Chipset para control de accesos a memoria y tarjeta gráfica/periféricos PCI express .
5.- Zócalos ubicación memoria RAM (módulos DIMM). Al ser en dos colores permite memorias en Dual Channel .
6.- Conector fuente alimentación.
7.- Disipador sobre SouthBridge /Puente Sur .- Control de periféricos de Entrada/salida .
8.- Módulos PCI (Peripheral Component Interconnect).-Conexión periféricos antiguos .
9.- Conexión dispositivos SATA (Serial ATA) para discos duros o unidades ópticas .
10.- Pines de conexión leds frontales.
11.- Botones de encendido y reseteo .
12.- Conectores para USB (Universal Serial Bus) frontales. El color Rojo es para un conector USB 3.0 .
13.- Conector IDE (Integrate Drive Electronics) /ATA (Advanced Technology Attachment). Para discos duros o unidades ópticas antiguos .
14.- Pines para sonido frontal .
15.- PCI Express 16x principalmente para tarjetas gráficas. Al llevar esta placa varios se podrían conectar dos tarjetas en SLI/Crossfire .
16.- Sonido análogico .
17.- Conector RJ45 (LAN Ethernet) .

18.-Puertos USB Universal Serial Bus (no se puede precisar si son 2.0 ó 3.0 porque no se ve el color negro/azul respectivamente) .
19.- Puerto E-SATA (External SATA) para conectar dispositivos SATA externos .
20.-Puerto Firewire (IEEE 1394) Conexión de elevado ancho de banda para dispositivos extenos .
21.- S/PDIF fibra óptica- oslink(Interfaz audio digital) .
22.- S/PDIF coaxial-RCA (Interfaz audio digital) .
23.- Puerto PS2/mini DIN. Conexión teclado.