Apellidos, Nombre: _____ **DNI:**_____ **Gupo:**_____

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

Examen final de Fundamentos de Computadores (Convocatoria de enero)

21 de enero de 2016

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

Parte II: cuestiones teórico-prácticas *(32%; puntuación indicada en cada apartado)*

C1. *(0,8 puntos)* Representación en IEEE 754:

a) Determinar el número real que se corresponde con la ristra hexadecimal de 64 bits 40B0E1C000000000, en formato IEEE 754 de doble precisión (representado en *big-endian*).

b) Indicar los 4 bytes (expresados en hexadecimal) que habría que almacenar para escribir el número +4321,75 en coma flotante en formato IEEE 754 de simple precisión, siguiendo un convenio de almacenamiento *big-endian*.

C2. *(0,8 puntos)* Dada la función expresada en su forma normal canónica ***f(A,B,C,D) =*** $\sum$ ***m(0, 2, 3, 6, 7, 8, 9, 13, 15) + d (10)****, se pide:*

a) Representarla en el mapa de Karnaugh correspondiente.

b) Localizar todos los implicantes primos esenciales, indicando porqué lo son.

c) Localizar todos los implicantes primos no esenciales usados en la cobertura, indicando porqué son no esenciales.

d) Realizar una cobertura mínima en el mapa de Karnaugh para implementar la función como suma de productos.

Nota: + *d (10)* significa que el minitérmino **10** es indeterminado (salida no importa).

C3. *(0,8 puntos)* Repertorio de instrucciones del x86-64:

a) Operandos de instrucciones del ensamblador del x86-64:
- ¿Qué tipos de operandos hemos estudiado durante el curso que manejen las instrucciones del x86-64?
- Haciendo uso de la instrucción "mov" de ensamblador, pon un ejemplo para cada tipo de operando.

b) Repertorio de instrucciones del x86-64:
- ¿En qué tres grandes grupos hemos clasificado el juego de instrucciones del x86-64? ¿En qué otros grupos se subdividen éstos?
- Pon un ejemplo de una instrucción que pertenezca a cada subgrupo.

C4. *(0,8 puntos)* Rellene el siguiente texto sobre redes de computadores:

"Las especificaciones de conexión Ethernet (cable) y Wifi (inalámbrica) se corresponden con los estándares de IEEE ___**(a)**___ *y* ___**(b)**___*, respectivamente. Ambas especificaciones están ubicadas en la capa de* ___**(c)**___ *de la arquitectura de capas de Internet. El protocolo encargado del enrutamiento de paquetes, y por tanto responsable del direccionamiento de los hosts a nivel global, se conoce con el nombre de protocolo* ___**(d)**___ *y se ubica en la capa inmediatamente superior, o sea, la capa de* ___**(e)**___*. En la capa de transporte, los dos protocolos más ampliamente utilizados son* ____**(f)**___ *y* ___**(g)**___*, el primero de los cuales es orientado a conexión y fiable, mientras que el segundo no, ya que suele usarse más para aplicaciones de tipo streaming de audio y vídeo. Finalmente, tres de los protocolos más utilizados en el nivel de aplicación son los de envío de correo (protocolo* ___**(h)**___*), transferencia de archivos (protocolo* ___**(i)**___*) y transferencia de páginas web (protocolo* ___**(j)**___*)".*

Nota: *Para todos los protocolos cuyas siglas se han mencionado en el apartado anterior (es decir, apartados **d**, **f**, **g**, **h**, **i** y **j**), hacer constar explícitamente su significado en inglés.*

Parte III: ejercicios boletines *(30,5%; puntuación indicada en cada apartado)*

P1. *(0,7 puntos)* Implementar un sumador de 4 bits usando sumadores individuales completos de un bit (cada uno con sus dos entradas de datos A y B, y otra de acarreo de entrada Cin, y sus dos salidas de resultado S y acarreo de salida Cout). Para ello, seguir los siguientes pasos:

a) Implementar el bit de salida S del sumador de 1 bit usando un multiplexor de 4 a 1, y los bits A y B como bits de control.

b) Implementar el bit de salida Cout del sumador de 1 bit usando únicamente puertas NAND.

c) Usando cuatro sumadores completos de 1 bit, construir un sumador de 4 bits (**Nota:** dibujar aquí ya cada sumador de 1 bit usando el bloque lógico correspondiente en el que se detallan sólo sus entradas y salidas, sin necesidad de dibujar su implementación interna, ya desarrollada en los apartados a y b).

P2. *(0,8 puntos)* Indique las órdenes Linux utilizadas para las siguientes tareas (se pide solamente los nombres de los respectivos comandos en sí, sin opciones ni parámetros):

a) Mostrar el contenido de un fichero de texto por el terminal sin posibilidad de edición.
b) Mostrar todos los procesos en ejecución (actualizándose en tiempo real).
c) Matar un proceso.
d) Crear un archivo nuevo vacío.
e) Obtener espacio físico real ocupado por un determinado directorio o fichero.
f) Saber la ruta absoluta del directorio en el que nos encontramos.
g) Saber qué usuario soy.
h) Llevarnos a nuestro directorio de usuario (nuestro home).
i) Crear un enlace duro.
j) Borrar un directorio vacío.
k) Cambiar permisos a archivos y directorios.
l) Mostrar (de forma estática) los procesos del usuario.
m) Traer a primer plano un proceso parado o en segundo plano.
n) Encontrar la ubicación de ficheros o directorios.
o) Copiar ficheros o directorios.
p) Crear directorios.

(Sigue detrás)

P3. *(1 punto)* 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):

```
user@host:~/$ gdb ./main
[...]
(gdb) list
1 int array[4] = {-1,0,+2,+4};
2 int main() {
3 int i;
4 for(i=1;i<16;i++)
[...]
(gdb) disassemble main
Dump of assembler code for function main:
[...]
0x4005ac <main+47>:  movl $0x01,-0x04(%rbp)
0x4005b0 <main+51>:  jmp 0x400622 <main+149>
0x4005b2 <main+53>:  [...]
0x400622 <main+149>: addl $0x01,-0x04(%rbp)
0x400626 <main+153>: cmpl $0x10,-0x04(%rbp)
0x40062a <main+157>: jl 0x4005b2 <main+53>
[...]
(gdb) x/10bx 0x400622
0x400622 <main+149>: 0x83 0x45 0xfc 0x01 0x83 0x7d 0xfc 0x10
(gdb) x/12bx array
0x502050 <array>:    0xXX 0xff 0xff 0xYY 0x00 0x00 0x00 0x00
0x502058 <array+8>: 0xZZ 0x00 0x00 0x00
```

En base a dicha sesión, rellenar todos los huecos del texto que va a continuación:

"El código depurado en la sesión manipula una tabla de ___(a)___ *elementos de tipo entero de 32 bits. Dicha tabla ocupa exactamente* ___(b)___ *bytes en memoria, y comienza en la dirección exacta de memoria* ___(c)___*. Justo al inicio de la ejecución del programa, el valor del byte que aparece sustituido en negrita con* `0xXX` *será, en realidad, el valor* ___(d)___ *(suponer que el convenio de almacenamiento utilizado es little endian y ponerlo en hexadecimal), el valor de* `0xYY` *será en realidad* ___(e)___**,** *mientras que el valor de* `0xZZ` *será en realidad* ___(f)___ *(ponerlos también en hexadecimal). El código que aparece en la figura se corresponde con un código de alto nivel que inicializa la variable* ___(g)___ *del bucle* `for` *a* ___(h)___*. En ensamblador se corresponde con la instrucción* ___(i)___ *que se encuentra en la dirección* ___(j)___*. Después de ejecutar las instrucciones dentro del bucle, el bucle incrementa la variable i, comprueba si es* ___(k)___ *que dieciséis, y en caso afirmativo vuelve a realizar otra iteración del bucle. En particular, sabiendo que en algún lugar previo del programa se puso* `%rbp` *apuntando al marco de pila de la función* `main`*, la instrucción* `cmpl $0x10,-0x04(%rbp)` *se encarga de* ___(l)___ *el valor de la variable* `i` *(que está en la pila) con el valor* ___(m)___*. Podemos también afirmar que dicha instrucción* `cmpl`*, una vez ubicada en memoria, ocupa exactamente* ___(n)___ *bytes, cuyos valores concretos son* ___(ñ)___ *(expresar aquí los bytes correspondientes tal y como aparecen en el listado separados por un espacio en blanco), que están comprendidos entre la dirección* ___(o)___ *y la* ___(p)___*, ambas inclusive. La instrucción ubicada en la dirección* `0x4005b0` *es de tipo* ___(q)___*, mientras que la ubicada en la dirección* `0x400622` *es de tipo* ___(r)___*. Por último, el valor alojado en la posición de memoria* `0x502054` *se corresponde con un contenido del segmento de* ___(s)___*."*

P4 *(0,55 puntos)* Responda a las siguientes preguntas sobre redes de ordenadores:

a) Dado un equipo con IP y máscara `155.54.14.254/23,` indique:

- **a1)** Su dirección de red.
- **a2)** Su dirección de broadcast.

b) Para la red anterior indíquese la primera IP válida para su router por defecto **(b)**.

c) Rellene los huecos de cada apartado (c1), (c2), (c3), (c4), (c5),(c6), (c7) y (c8) considerando que ambos comandos Linux (c1 y c5) se han ejecutado en dicho equipo:

```
user@host:~$ ___(c1)____
eth0  Link encap: Ethernet     dirección HW  00:1A:BC:2E:00:00
      Direc. inet:__(c2)___    Difus.:__(c3)___ Másc: ___(c4)___
```

```
user@host:~$__(c5)__
Destino     Pasarela     Genmask     Indic  Métric   Ref    Uso    Interfaz
__(c6)__      *         __(c7)__      U       0       0      0      eth0
Default     __(c8)__    0.0.0.0       UG      0       0      0      eth0
```

d) Si dispongo de una red `155.54.0.0/16` tendré un total (incluyendo router en su caso) de hasta ___(d1)___ interfaces de red válidas; si configurara todos los equipos de la misma con la máscara `255.255.240.0`, obtendría ___(d2)___ subredes de ___(d3)___ interfaces de red válidos (incluyendo los posibles respectivos routers) cada una.

Apellidos, Nombre: _____ **DNI:**_____ **Gupo:**_____

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

Examen final de Fundamentos de Computadores (Convocatoria de enero)

21 de enero de 2016

SÓLO SEGUNDO PARCIAL (TEMAS 4, 5 y 6)

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

C1. *(1 punto)* SO que permiten la Multitarea (multiprogramación):

a) ¿Qué son los sistemas operativos *multiprogramación* (o *multitarea*)?
b) ¿Qué mecanismos se usan fundamentalmente para conseguir la multitarea? Describid someramente en qué consiste cada uno de ellos.

C2. *(1 punto)* Repertorio de instrucciones del x86-64:

a) Operandos de instrucciones del ensamblador del x86-64:
- ¿Qué tipos de operandos hemos estudiado durante el curso que manejen las instrucciones del x86-64?
- Haciendo uso de la instrucción "mov" de ensamblador, póngase un ejemplo para cada tipo de direccionamiento.

b) Repertorio de instrucciones del x86-64:
- ¿En qué tres grandes grupos hemos clasificado el juego de instrucciones del x86-64? ¿En qué otros grupos se subdividen éstos?
- Ponga un ejemplo de una instrucción que pertenezca a cada subgrupo.

C3. *(1 punto)* Rellene el siguiente texto sobre redes de computadores:

"Las especificaciones de conexión Ethernet (cable) y Wifi (inalámbrica) se corresponden con los estándares de IEEE ___**(a)**___ *y* ___**(b)**___ *, respectivamente. Ambas especificaciones están ubicadas en la capa de* ___**(c)**___ *de la arquitectura de capas de Internet. El protocolo encargado del enrutamiento de paquetes, y por tanto responsable del direccionamiento de los hosts a nivel global, se conoce con el nombre de protocolo* ___**(d)**___ *y se ubica en la capa inmediatamente superior, o sea, la capa de* ___**(e)**___*. En la capa de transporte, los dos protocolos más ampliamente utilizados son* ___**(f)**___ *y* ___**(g)**___*, el primero de los cuales es orientado a conexión y fiable, mientras que el segundo no, ya que suele usarse más para aplicaciones de tipo streaming de audio y vídeo. Finalmente, tres de los protocolos más utilizados en el nivel de aplicación son los de envío de correo (protocolo* ___**(h)**___*), transferencia de archivos (protocolo* ___**(i)**___*) y transferencia de páginas web (protocolo* ___**(j)**___*)".*

Nota: *Para todos los protocolos cuyas siglas se han mencionado en el apartado anterior (es decir, apartados **d**, **f**, **g**, **h**, **i** y **j**), hacer constar explícitamente su significado en inglés.*

Parte III: ejercicios boletines *(30%; puntuación indicada en cada apartado)*

P1. *(0,9 puntos)* Indique las órdenes Linux utilizadas para las siguientes tareas (se pide solamente los nombres de los respectivos comandos en sí, sin opciones ni parámetros):

a) Mostrar el contenido de un fichero de texto por el terminal sin posibilidad de edición.
b) Mostrar todos los procesos en ejecución (actualizándose en tiempo real).
c) Matar un proceso.
d) Crear un archivo nuevo vacío.
e) Obtener espacio físico real ocupado por un determinado directorio o fichero.
f) Saber la ruta absoluta del directorio en el que nos encontramos.
g) Saber qué usuario soy.
h) Llevarnos a nuestro directorio de usuario (nuestro home).
i) Creación de un enlace duro.
j) Borrar un directorio vacío.
k) Cambiar permisos a archivos y directorios.
l) Mostrar (de forma estática) los procesos del usuario.
m) Traer a primer plano un proceso parado o en segundo plano.
n) Traer a segundo plano un proceso parado o dormido.
o) Encontrar la ubicación de ficheros o directorios.
p) Copiar ficheros o directorios.
q) Mover ficheros o directorios.
r) Crear directorios.

P2. *(1 punto)* 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):

```
user@host:~/$ gdb ./main
[...]
(gdb) list
1 int array[4] = {-1,0,+2,+4};
2 int main() {
3 int i;
4 for(i=1;i<16;i++)
[...]
(gdb) disassemble main
Dump of assembler code for function main:
[...]
0x4005ac <main+47>:  movl $0x01,-0x04(%rbp)
0x4005b0 <main+51>:  jmp 0x400622 <main+149>
0x4005b2 <main+53>:  [...]
0x400622 <main+149>: addl $0x01,-0x04(%rbp)
0x400626 <main+153>: cmpl $0x10,-0x04(%rbp)
0x40062a <main+157>: jl 0x4005b2 <main+53>
[...]
(gdb) x/10bx 0x400622
0x400622 <main+149>: 0x83 0x45 0xfc 0x01 0x83 0x7d 0xfc 0x10
(gdb) x/12bx array
0x502050 <array>:   0xXX 0xff 0xff 0xYY 0x00 0x00 0x00 0x00
0x502058 <array+8>: 0xZZ 0x00 0x00 0x00
```

(Sigue detrás)

En base a dicha sesión, rellénense todos los huecos del texto que van a continuación:

"El código depurado en la sesión manipula una tabla de ___**(a)**___ *elementos de tipo entero de 32 bits. Dicha tabla ocupa exactamente* ___**(b)**___ *bytes en memoria, y comienza en la dirección exacta de memoria* ___**(c)**___*. Justo al inicio de la ejecución del programa, el valor del byte que aparece sustituido en negrita con* **`0xXX`** *será, en realidad, el valor* ___**(d)**___ *(suponer que el convenio de almacenamiento utilizado es little endian y ponerlo en hexadecimal), el valor de* **`0xYY`** *será en realidad* ___**(e)**___**,** *mientras que el valor de* **`0xZZ`** *será en realidad* ___**(f)**___ *(ponerlos también en hexadecimal). El código que aparece en la figura se corresponde con un código de alto nivel que inicializa la variable* ___**(g)**___ *del bucle* `for` *a* ___**(h)**___*. En ensamblador se corresponde con la instrucción* ___**(i)**___ *que se encuentra en la dirección* ___**(j)**___*. Después de ejecutar las instrucciones dentro del bucle, el bucle incrementa la variable i, comprueba si es* ___**(k)**___ *que dieciséis, y en caso afirmativo vuelve a realizar otra iteración del bucle. En particular, sabiendo que en algún lugar previo del programa se puso* `%rbp` *apuntando al marco de pila de la función* `main`**,** *la instrucción* `cmpl $0x10,-0x04(%rbp)` *se encarga de* ___**(l)**___ *el valor de la variable* `i` *(que está en la pila) con el valor* ___**(m)**___*. Podemos también afirmar que dicha instrucción* `cmpl`*, una vez ubicada en memoria, ocupa exactamente* ___**(n)**___ *bytes, cuyos valores concretos son* ___**(ñ)**___ *(expresar aquí los bytes correspondientes tal y como aparecen en el listado separados por un espacio en blanco), que están comprendidos entre la dirección* ___**(o)**___ *y la* ___**(p)**___ *, ambas inclusive. La instrucción ubicada en la dirección* `0x4005b0` *es de tipo* ___**(q)**___ *, mientras que la ubicada en la dirección* `0x400622` *es de tipo* ___**(r)**___*. Por último, el valor alojado en la posición de memoria* `0x502054` *se corresponde con un contenido del segmento de* ___**(s)**___*."*

P3 *(1,1 puntos)* Responda a las siguientes preguntas sobre redes de ordenadores:

a) Dado un equipo con IP y máscara `155.54.14.254/23`, indique:

- **`a1)`** Su dirección de red:
- **`a2)`** Su dirección de Broadcast:

b) Para la red anterior indíquese la primera IP válida para su router por defecto **`(b)`**.

c) Rellene los huecos de cada apartado (c1), (c2), (c3), (c4), (c5),(c6), (c7) y (c8) considerando que ambos comandos Linux (c1 y c5) se han ejecutado en dicho equipo:

```
user@host:~$ ___(c1)___
eth0  Link encap: Ethernet     dirección HW  00:1A:BC:2E:00:00
      Direc. inet:__(c2)__     Difus.:__(c3)__       Másc: __(c4)__

user@host:~$___(c5)___
Destino    Pasarela    Genmask     Indic  Métric   Ref   Uso   Interfaz
__(c6)__      *        __(c7)__      U       0       0     0     eth0
Default    __(c8)__     0.0.0.0      UG      0       0     0     eth0
```

d) Si dispongo de una red `155.54.0.0/16` tendré un total (incluyendo router en su caso) de hasta __(d1)__ interfaces de red válidas; si configurara todos los equipos de la misma con la máscara `255.255.240.0`, obtendría __(c2)__ subredes de __(d3)__ interfaces de red válidos (incluyendo los posibles respectivos routers) cada una.

Soluciones de TODO:

C1.

a)

4 0 B 0 E 1 C $000000000_{)16}$

0100 0000 1011 0000 1110 0001 1100 0..... $….0_{)2}$

S = 0 = +;

E = $100\ 0000\ 1011_{)2}$ = 1024 + 8 + 2 + 1 = 1035; e = 1035 – 1023 = 12.

M = 1,m = 1, 0000 1110 0001 1100 0 ..... $….0_{)2}$; Nº = 1,0000 1110 0001 1100 0..... $….0_{)2}$ * 2^{12} = 1 0000 1110 0001,1100 0 .... $….0_{)2}$ * 2^0 = 4321,75

b)

4321,75 = $1\ 0000\ 1110\ 0001,11_{)2}$ = 1,0000 1110 0001 1100... ...0 * $2^{12;}$

s = + = 0;

E = 12 => e = 12+ 127 = 139 = $1000\ 1011_{)2}$

M = 1,m => m = 0000 1110 0001 1100... $...0_{)2}$

IEEE = 0 100 0101 1000 0111 0000 1110 0... $...0_{)2}$ =

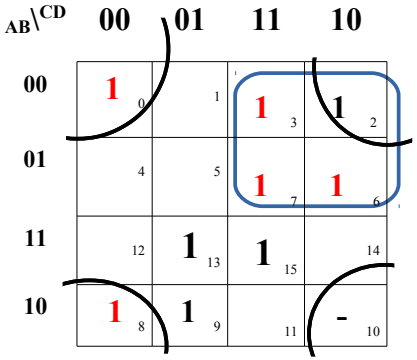
4 5 8 7 0 E 0 $0_{)16}$

C2. Nota: Fijaos que en los mapas de Karnaugh en este ejercicio se emplea una numeración por filas en lugar de por columnas, al poner las variables A y B a la izquierda y CD a la derecha de la tabla. Por lo demás, ésto por supuesto no afecta a las soluciones.

a)

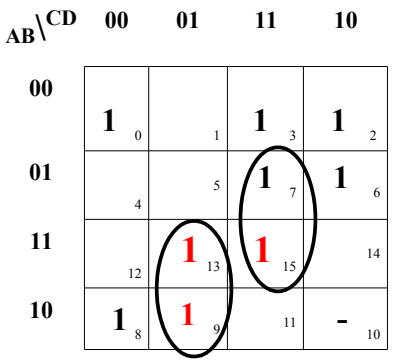
AB\CD	00	01	11	10
00	1 0	1	1 3	1 2
01	4	5	1 7	1 6
11	12	1 13	1 15	14
10	1 8	1 9	11	- 10

b)



Los implicantes 3, 6 y 7 sólo pueden ser cubiertos por un único implicante primo de orden 2 (2, 3, 6, 7).
Los implicantes 0 y 8 sólo pueden ser cubiertos por un único implicante primo de orden 2 (0, 2, 8, 10).
El resto puede ser cubierto por 2 implicantes del mismo orden (orden 1). Luego no son esenciales.

c)



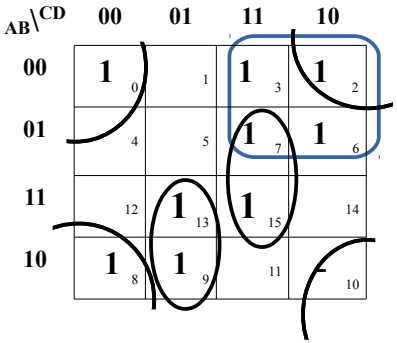
El minitérmino 9 puede ser cubierto por el implicante primo (orden 1) (9, 13) ó también (8, 9)

El minitérmino 13 puede ser cubierto por el implicante primo (orden 1) (9, 13) ó también (13, 15)

El minitérmino 15 puede ser cubierto por el implicante primo (orden 1) (7, 15) ó también (13, 15)

Luego son implicantes primos “no esenciales (no pueden ser cubiertos por implicantes de orden 2 y sí hay más de uno de orden 1 que los cubre).

d) Hay varias coberturas, pero todas ellas necesitan, obligatoriamente, contener los implicantes primos esenciales (2, 3, 6, 7) y (0, 2, 8, 10), por ejemplo:



C3.

a)

Básicamente 3 tipos:
- operandos en memoria
- operandos en registros del procesador
- operandos inmediatos (en la propia instrucción)

-Haciendo uso de la instrucción “mov” de ensamblador, pon un ejemplo para cada tipo de operando:
- mov array(,%rax,4),%ecx (primer operando en memoria)
- mov %rbx, %rax (primer y segundo operandos son registros)
- mov $3100, %ebx (primer operando valor inmediato.)

b)

1. Instrucciones aritmético lógicas:
   - Instrucciones aritméticas. (add %rbx, %rax)
   - Instrucciones lógicas. (xor %bh, %al)
   - Instrucciones de incremento y decremento. (inc %eax)
   - Instrucciones de desplazamiento. (shl $3, %eax)

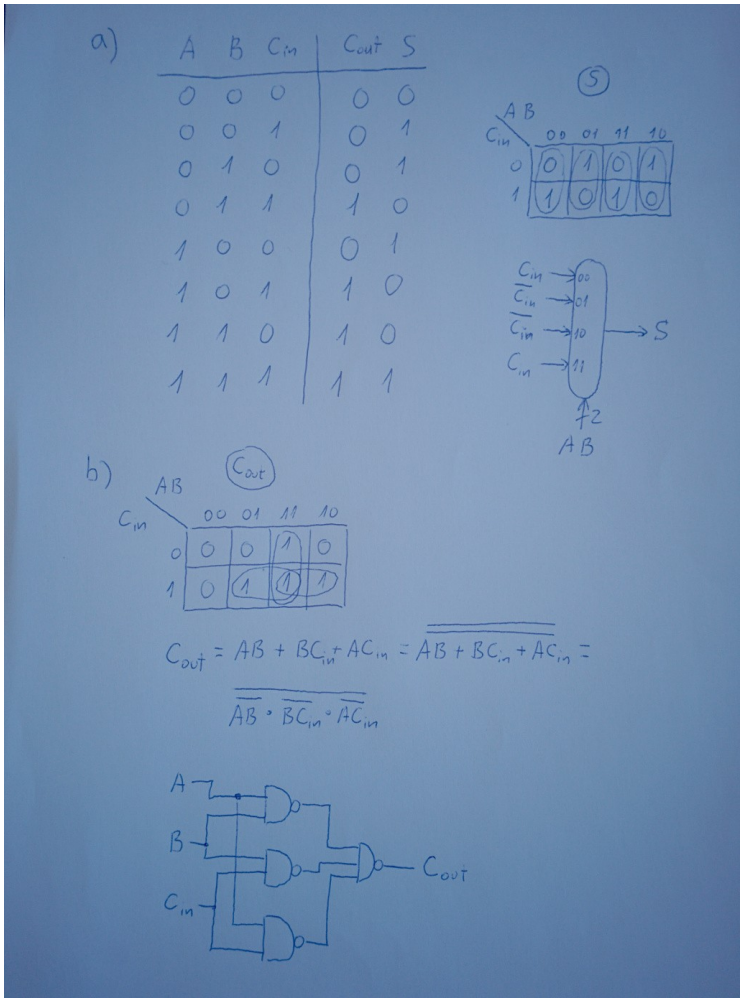
2. Instrucciones de movimiento de datos:
   Valen aquí los ejemplos del apartado a) anterior

3. Instrucciones de salto:
   - Saltos incondicionales. (jmp .L1)
   - Saltos condicionales. (jle .L1)
   - Saltos con retorno (llamadas a subrutinas). (call subrutina)

C4.

a) 802.3
b) 802.11
c) Enlace
d) IP (Internet Protocol)
e) Red
f) TCP (Transmission Control Protocol)
g) UDP (User Datagram Protocol)
h) SMTP (Simple Mail Transfer Protocol)
i) FTP File Transfer Protocol)
j) HTTP (Hypertext Transfer Protocol)

P1.



P2.

a) cat
b) top
c) kill
d) touch
e) du
f) pwd
g) whoami
h) cd
i) ln
j) rmdir
k) chmod
l) ps
m) fg
n) find
o) cp
p) mkdir

P3.

a) Cuatro
b) 16
c) 0x502050
d) 0xff
e) 0xff
f) 0x02
g) i
h) 1
i) movl $0x01,-0x04(%rbp)
j) 0x4005ac
k) menor
l) comparar
m) 0x10 ó 16
n) 4
ñ) 0x83 0x7d 0xfc 0x10
o) 0x400626
p) 0x400629
q) Salto incondicional
r) Aritmetico-lógica
s) Datos

P4.

a) Dirección de red: 155.54.14.0
Dirección de Broadcast: 155.54.15.255

b) 155.54.14.1

c)

- c1) ifconfig
- c2) 155.54.14.254
- c3) 155.54.15.255
- c4) 255.255.254.0
- c5) route
- c6) 155.54.14.0
- c7) 255.255.254.0
- c8) 155.54.14.1

d)

- d1) 65534
- d2) 16
- d3) 4094

Soluciones del PARCIAL:

C1.

a) Aquellos SO que permiten un uso compartido de la(s) CPU(s) entre distintos procesos, de manera que los procesos van avanzando en su ejecución con sensación de simultaneidad.

b) El uso de interrupciones (timer) que desvían el flujo de ejecución del procesador al núcleo del SO para que este haga un cambio de tarea y los procesos bloqueados en una E/S que aprovecha el SO para asignar al procesador a otra tarea.

c) 1. Periódicamente (un elevado número de veces por segundo), un reloj (el timer) interrumpe al proceso actual, y entra a ejecutarse código del kernel que cambia el contexto para ejecutar otro proceso.

2. Cuando un proceso le pide al SO una operación de E/S, el SO lo deja bloqueado, momento que aprovecha el SO para realizar la multitarea con sólo el resto de tareas activas, utilizando, por ejemplo, el método anterior, hasta que el dispositivo encargado genera una interrupción avisando de que ha acabado su transferencia. En este momento el SO reanuda la multitarea incluyendo al proceso que pidió la operación de E/S.

C2. Véase C3 de TODO.

C3. Véase C4 de TODO.

P1. Véase P2 de TODO.

P2.

a) cat
b) top
c) kill
d) touch
e) du
f) pwd
g) whoami
h) cd
i) ln
j) rmdir
k) chmod
l) ps
m) fg
n) bg
o) find
p) cp
q) mv
r) mkdir

P3. Véase P4 de TODO.