Problema 12. (1) fiabilidad, disponibilidad

Tenemos un sistema compuesto por los elementos mostrados en la tabla siguiente. La tabla también muestra el numero de componentes de cada tipo y el tiempo medio hasta fallo (MTTF) de cada componente.

Componente	Fuente alimentación	CPU	Placa base	DIMM	GPU	Disco duro
Nº	1	1	1	4	1	8
MTTF (horas)	125.000	1.000.000	200.000	1.000.000	500.000	100.000

a) **Calcula** el tiempo medio hasta fallos del sistema

$$\frac{1}{MTTF_{total}} = \frac{1}{125\cdot 10^3} + \frac{1}{10^6} + \frac{1}{200\cdot 10^3} + \frac{4}{10^6} + \frac{1}{500\cdot 10^3} + \frac{8}{100\cdot 10^3}$$

MTFF = **10.000 horas**

b) El tiempo medio para reemplazar un componente que ha fallado (MTTR) es de 20 horas. Calcula el tiempo medio entre fallos (MTBF).

$$MTBF = MTTF + MTTR = 10.000 + 20 = 10.020\ h$$

c) ¿Cual es la disponibilidad del sistema?

$$Disponibilidad = \frac{MTTF}{MTBF} = \frac{10000}{10020} = 0'9980 \rightarrow 99'80\%$$

x = 0110 0110 ~x = 1001 1001 !x = 0
y = 1001 0011 ~y = 0110 1100 !y = 0

Problema 1. (1) Operadores lógicos

Suponed que `x` e `y`, variables de tamaño 1 byte, tienen los valores `0x66` y `0x93` respectivamente. Rellenad la siguiente tabla, indicando los valores resultantes de aplicar las siguientes expresiones en C:

Expresión	valor binario	valor hex	Expresión	valor binario	valor hex
`x & y`	0000 0010	02	`x && y`	0000 0001	01
`x | y`	1111 0111	F7	`x || y`	0000 0001	01
`~x | ~y`	1111 1101	FD	`!x || !y`	0000 0000	00
`x & !y`	0000 0000	00	`x && ~y`	0000 0001	01

Problema 2. (1) Desplazamientos

x		x << 4		x >> 3 (lógico)		x >> 3 (aritmético)	
hex	**binario**	**hex**	**binario**	**hex**	**binario**	**hex**	**binario**
0xF0	1111 0000	00	0000 0000	1E	0001 1110	FE	1111 1110
0x0F	0000 1111	F0	1111 0000	01	00000001	01	00000001
0xCC	1100 1100	C0	1100 0000	19	0001 1001	F9	1111 1001
0x55	0101 0101	50	0101 0000	0A	0000 1010	0A	0000 1010
0x80	1000 0000	00	0000 0000	10	0001 0000	F0	1111 0000
0x02	0000 0010	20	0010 0000	00	0000 0000	00	0000 0000

Problema 5. (2) Traducción

Dada la siguiente definición de variables globales:

```
char A[256];
char tabla[256];
```

Traducid a ensamblador de IA32 el siguiente fragmento de código:

```
for (i=0; i<256; i++)
  A[i] = tabla[A[i]];
```

```
movl    $0,%esi              # eax = i
movl    $A, %ebx             # ebx = A
movl    $tabla, %ecx         # ecx = tabla

for:
cmpl    $256, %esi
je      end

movzbl  (%ebx, %esi),%edx    # dl = A[i]
movb    (%ecx, %edx), %dl
movb    %dl, (%ebx, %esi)    # A[i] = tabla[A[i]]
incl    %esi                 # ++i
jmp     for

end:
```

Problema 6. (2) Traducción

Dado el siguiente código escrito en C:

```
int *sorpresa (int i,  int *x)
{
        if (i>-10 && i<10)
           *x = i;
        else
           x = &i;
        return x;
}
```

Teniendo en cuenta que los parámetros de la función son accesibles en las siguientes direcciones: `i` en `8(%ebp)`, dirección de `x` en `12(%ebp)`, traducid a ensamblador del IA32 el cuerpo de la subrutina.

```
# @x = 12(%ebp)
# i = 8(%ebp)

pushl       %ebp
movl        %esp, %ebp
subl        $4, %ebp

————————————————————————

if:
cmpl        -10, 8(%ebp)
jle         else
cmpl        10, 8(%ebp)
jge         else

then:
movl        12(%ebp), %eax
movl        8(%ebp), %ecx
movl        %ecx, (%eax)      # *x = i
jmp         end

else:
leal        8(%ebp), 12(%ebp)

end:
movl 12(%ebp), %eax
ret
```