COGNOMS:															
NOM:								[NI/N	IIE:					

IMPORTANTE leer atentamente antes de empezar el examen: Escriba los apellidos, el nombre y el DNI/NIE antes de empezar el examen. Escriba un solo carácter por recuadro, en mayúsculas y lo más claramente posible. Es importante que no haya tachones ni borrones y que cada carácter quede enmarcado dentro de su recuadro sin llegar a tocar los bordes. Use un único cuadro en blanco para separar los apellidos y nombres compuestos si es el caso. No escriba fuera de los recuadros, todo lo que haya fuera de ellos es ignorado. La identificación del alumno se realiza de forma automática, no seguir correctamente estas instrucciones puede comportar no tener nota.

Problema 1. (5 puntos)

Un programa P tiene una parte del código que es perfectamente paralelizable (speed-up = numero de CPUS).

a) Calcula el porcentaje (%) del código que deberíamos paralelizar para conseguir un speed-up de 2 en un multiprocesador con 5 CPUs.

```
Ley de amdahl:
2 = 1/(1-p + p/5) ---> p = 0.625 ---> 62.5%
```

Cada CPU funciona a un frecuencia de 2 GHz. Se ha ejecutado el programa P secuencial en un simulador con una única CPU ideal donde todos los accesos a memoria tardan un ciclo. Dicho programa ejecuta $5x10^9$ instrucciones, realiza $2x10^9$ operaciones de punto flotante y tarda $8x10^9$ ciclos.

b) Calcula el CPI ideal (CPI_{ID}) y el tiempo de ejecución en segundos (Texec) del programa P en este sistema ideal.

```
CPI<sub>IDEAL</sub> = 8x10<sup>9</sup> ciclos / 5x10<sup>9</sup> instrucciones = 1,6 ciclos / instrucción

Texe = Ciclos / F= 8x10<sup>9</sup> ciclos / 2GHz = 4 seg
```

c) Calcula los MIPS y MFLOPS en dicho sistema ideal.

```
MIPS = Instrucciones / Texe*10^6 = 5x10^9 instrucciones / 4*10^6 = 1250 MIPS

MFLOPS = OpsPF / Texe*10^6 = 2x10^9 instrucciones / 4*10^6 = 500 MFLOPS
```

La implementación de dicha CPU (CPU REAL) dispone de una cache de instrucciones (IC) y una cache de datos (DC) conectadas a memoria principal (MP). El programa P realiza $5x10^9$ accesos a instrucciones y $3x10^9$ accesos a datos, con una tasa de fallos (miss) del 10% en DC y del 5% en IC.

La DC tiene una política de escritura *Copy Back* y *Write Allocate*. En el programa P sabemos que el 33% de los accesos a datos son escrituras y que la probabilidad de que un bloque haya sido modificado en cache es del 25%.

En caso de acierto en la cache (tanto en IC como en DC) el tiempo de acceso es de 1 ciclo (igual que en el simulador). En caso de fallo, el tiempo de penalización es de 80 ciclos para reemplazar un bloque NO modificado y de 160 ciclos para reemplazar un bloque modificado.

d) Calcula el tiempo medio de acceso a memoria en ciclos para los accesos a instrucciones (Tmal)

,		
Tmal = Tsa+m*Tpf = 1 + 0,05*80 = 5 ciclos/acceso		

28 April 2021 6:57 pm 1/4

٠,	- 1 1 1	10 1				
e١	Calcula el fiemn	o medio de acceso	a memoria en ciclos	nara los accesos	a datos (l	mal)
\sim	calcala ci dellip	o micalo ac acceso	a michiona chi dicios	para ios accesos	a aatos (i	mub

TmaD = Tsa+m*(Pm*Tpfm + Pn*Tpfn) = 1 + 0,10*(0,25*160+0,75*80) = **11 ciclos/acceso**

f) Calcula el tiempo de ejecución del programa P en la CPU real.

```
Ciclos = Ciclos ideal + cilosIC + ciclosDC = Ciclos = 8 \times 10^9 + 5 \times 10^9 \times 0.05 \times 80 + 3 \times 10^9 \times 0.10 \times (0.25 \times 160 + 0.75 \times 80) = 58 \times 10^9 ciclos Texe = 58 \times 10^9 / 2 GHz = 29 s

Alternativas

(1) Tma medio = (5 \times 5 + 3 \times 11)/8 = 7.25 CPI = CPIid + nr*(Tma-1) = 1.6 + 1.6 \times 6.25 = 11.6 c/i

(2) CPI = CPIid + CPIi + CPId = 1.6 + 1 \times 0.05 \times 80 + 0.6 \times 0.10 \times (0.25 \times 160 + 0.75 \times 80) = 11.6 c/i

Texe = CPI*N*tc = 11.6 ciclos/instrucción * 5 \times 10^9 instrucciones * 0.5 ns = 29 s
```

El tamaño de bloque (línea) de todas las caches es de 64 bytes. Los accesos a IC son siempre de 4 bytes (el tamaño de las instrucciones), los accesos a DC son siempre de 8 bytes.

g) Calcula el número de bytes que se leen de memoria principal.

Bytes = 64 * (5x10⁹ * 0,05 + 3x10⁹ * 0,10) = **35,2 x10⁹ bytes**

h) Calcula el número de bytes que se escriben en memoria principal.

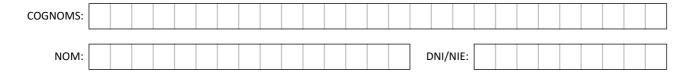
Bytes = 64 * 3x10⁹ * 0,10 *0,25 = **4,8 x10⁹ bytes**

Dicha CPU forma parte de un sistema con los siguientes componentes:

- 1 CPU, MTTF= 1.000.000 horas
- 1 Placa base, MTTF = 500.000 horas
- 1 Fuente de alimentación, MTTF = 250.000 horas
- 2 Discos duros, MTTF = 200.000 horas
- 8 DIMMs de memoria, MTTF = 1.000.000 horas
- i) **Calcula** el tiempo medio hasta fallo del sistema.

MTTF = 1/(1/1e6+1/5e5+1/2.5e5+8/1e6+2/2e5) = **40.000 Horas**

28 April 2021 6:57 pm 2/4



Problema 2. (5 puntos)

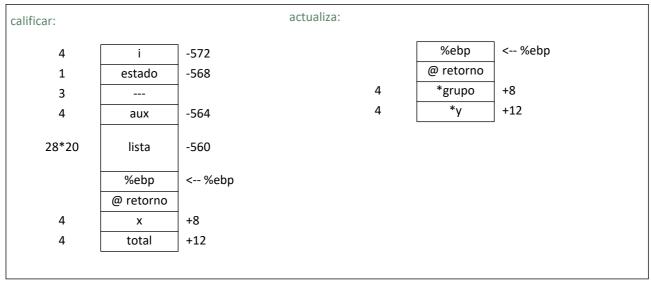
Dado el siguiente código escrito en C, que compilamos para un sistema linux de 32 bits:

```
int calificar(int x, int total)
typedef struct {
  int ID;
                                                 { int i;
  char aprobado;
                                                   char estado;
  int final;
                                                   int aux;
  int notas[4];
                                                   alumnos lista[20];
} alumnos;
                                                 c) lista[i].final=actualiza(lista, &aux);
int actualiza(alumnos grupo[20], int *y);
                                                 d) if(lista[i].final>=5)
                                                        estado='A';
                                                    else
                                                        estado='S';
                                                 }
```

a) **Dibuja** como quedaría almacenado en memoria la estructura alumnos, indicando claramente los desplazamientos respecto del inicio y el tamaño de todos los campos.

```
Tamaño: 28 bytes
          4
                                     +0
                       id
                   aprobado
                                     +4
          1
          3
                      final
                                     +8
          4
                     nota[0]
                                     +12
         16
                     nota[3]
                                     +28
```

Dibuja el bloque de activación de la función calificar y actualiza (no tiene variables locales), indicando claramente los desplazamientos relativos al EBP necesarios para acceder a los parámetros de ambas funciones y las variables locales de calificar.



28 April 2021 6:57 pm

c) **Traduce** a ensamblador el siguiente código dentro de la función calificar. Considera que la variable i se encuentra en el registro **%esi** con el valor actualizado

lista[i].final=actualiza(lista,&aux);

```
LEAL -564(%ebp), %ecx

PUSHL %ecx ; &aux

LEAL -560(%ebp), %ecx

PUSHL %ecx ; &lista

CALL actualiza

ADDL $8, %esp ; recupero puntero pila

IMULL $28, %esi, %ecx ; ecx <-i*size struct

MOVL %eax, -560+8(%ebp,%ecx) ; ret de actu -> lista[i].final
```

d) **Traduce** a ensamblador el siguiente código dentro de la función calificar, la cual se encuentra inmediatamente a continuación del apartado (c). Considera que lista[i].final se encuentra en el registro %eax.

```
if(lista[i].final>=5)
    estado='A';
else
```

```
estado='S';
```

28 April 2021 6:57 pm 4/4