COGNOMS:

NOM: DNI/NIE:

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_{IDEAL} = $8x10^9$ ciclos / $5x10^9$ instrucciones = **1,6 ciclos / instrucción**

Texe = Ciclos / F= $8x10^9$ 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 (TmaI)

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

e) **Calcula** el tiempo medio de acceso a memoria en ciclos para los accesos a datos (TmaD)

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 = $8x10^9$ + $5x10^9$*0,05*80 + $3x10^9$*0,10*(0,25*160+0,75*80) = $58x10^9$ ciclos
Texe = $58x10^9$ / 2 GHz = **29 s**

Alternativas

(1) Tma medio = (5*5+ 3*11)/8 = 7,25
CPI = CPIid + nr*(Tma-1) = 1,6 + 1,6*6,25 = 11,6 c/i

(2) CPI = CPIid + CPIi + CPId = 1,6 + 1*0,05*80 + 0,6*0,10*(0,25*160+0,75*80) = 11,6 c/i

Texe = CPI*N*tc = 11,6 ciclos/instrucción * $5x10^9$ 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 * ($5x10^9$ * 0,05 + $3x10^9$ * 0,10) = **35,2 $x10^9$ bytes**

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

Bytes = 64 * $3x10^9$ * 0,10 *0,25 = **4,8 $x10^9$ 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**

COGNOMS:

NOM: DNI/NIE:

Problema 2. (5 puntos)

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

```
typedef struct {
  int ID;
  char aprobado;
  int final;
  int notas[4];
} alumnos;

int actualiza(alumnos grupo[20], int *y);
```

```
int calificar(int x, int total)
{ int i;
  char estado;
  int aux;
  alumnos lista[20];
   ...
c) lista[i].final=actualiza(lista,&aux);
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	id	+0
1	aprobado	+4
3	---	
4	final	+8
	nota[0]	+12
16	---	
	nota[3]	
		+28

b) **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.

calificar:

4	i	-572
1	estado	-568
3	---	
4	aux	-564
28*20	lista	-560
	%ebp	<-- %ebp
	@ retorno	
4	x	+8
4	total	+12

actualiza:

	%ebp	<-- %ebp
	@ retorno	
4	*grupo	+8
4	*y	+12

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';
```

```
if:     CMPL $5,%eax                ; if(final < 5)
        JL else
then:   MOVB $'A', -568(%ebp)       ; 'A'->estado
        JMP endif
else:   MOVB $'S', -568(%ebp)       ; 'S'->estado
endif:
```