Cognoms: . . . . . Nom: . . . . . .

1er Control Arquitectura de Computadors **Curs 2012-2013 Q2**

Problema 1. (4 puntos)

Disponemos de un computador con un procesador a 2.2 Ghz. Un programa de prueba P realiza 1800 millones de operaciones de punto flotante y ejecuta 1200 millones de instrucciones que se distribuyen de la siguiente forma

	punto flotante	**enteras**	**memoria**
Nunero de instrucciones	500 millones	200 millones	500 millones
CPI	5	3	7

a) **Calcula** el tiempo de ejecución del programa P.

b) **Calcula** el CPI del programa P.

c) **Calcula** el rendimiento en MIPS y MFLOPS de P.

d) **Explica** cómo es posible ejecutar 1800 millones de operaciones de punto flotante con solo 500 millones de instrucciones de punto flotante.

e) **Define** el concepto de ganancia aplicado a la energia consumida por una aplicación en dos procesadores A y B. ¿Qué significa que el procesador B tiene una ganacia de 2 respecto al procesador A? ¿Qué significa que el procesador B tiene una ganacia del 50% respecto al procesador A?

En este procesador y con el programa P, las instrucciones de memòria representan el 75% de la energia consumida debida a conmutación. El fabricante del procesador está estudiando una modificación en el mismo que supondría una ganancia de 3 en la energia consumida por dichas instrucciones (el resto consumirian lo mismo) que no afectaria ni al CPI ni a la frecuencia.

f) **Calcula** la ganancia en **energia debida a conmutación** en la aplicación P con dicha mejora.

Una vez aplicada dicha mejora, se ha medido que cada instrucción de punto flotante consume 114 nJ (nano Joules) de energía debida a conmutación, cada instrucción entera consume 75 nJ y cada instrucción de memória consume 144 nJ. Además, esta CPU tiene una corriente de fugas de 10 A y funciona a un voltaje de 1,2 V.

g) **Calcula** la eficiencia energetica en Mflops/W para el programa P.

Este computador está formado por los componentes mostrados en la tabla siguiente. La tabla también muestra el número de componentes de cada tipo y el tiempo medio hasta fallo (MTTF) de cada componente.

Componente	Fuente alimentación	CPU	Ventilador CPU	Placa base	DIMMs	Discos duros	Tarjetas graficas
Nº	1	1	1	1	4	2	2
MTTF (horas)	100.000	1.000.000	100.000	200.000	1.000.000	125.000	500.000

El tiempo medio para reemplazar un componente que ha fallado (MTTR) es de 5 horas y la probabilidad de fallo sigue una distribución exponencial.

h) **Calcula** el tiempo medio hasta fallos del hardware (MTTF), el tiempo medio entre fallos (MTBF) y la disponibilidad del sistema.

Cognoms: . . . . Nom: . . . .

1er Control Arquitectura de Computadors **Curs 2012-2013 Q2**

Problema 2. (3 puntos)

Dado el siguiente código escrito en C:

```
typedef struct {
  char a;
  int b;
  short c;
  char d;
  int e[4];
} s1;
```

```
typedef struct {
  s1 f[100];
  int d;
  } s2;
```

```
int examen(s1 j, s2 *k, char m, short n){
  short u;
  char v;
  int w;
    ...
}
```

a) **Dibuja** como quedarían almacenadas en memoria las estructuras s1 y s2, indicando claramente los deplazamientos respecto al inicio y el tamaño de todos los campos.

b) **Dibuja** el bloque de activación de la función examen, indicando claramente los desplazamientos relativos al registro EBP necesarios para acceder a los parámetros y a las variables locales.

c) **Escribe** la expresión aritmética que permite calcular la dirección del elemento x.f[y.e[i]].d, siendo x una variable de tipo s2 e y una variable de tipo s1:

d) **Traduce** la siguiente sentencia a ensamblador del x86, suponiendo que está dentro de la función examen. Se valorará la optimización en el código. Escribe claramente la expresión aritmética a traducir.

```
j.e[w]=0;
```

e) **Define en C** una estructura equivalente a s1, reordenando sus campos de forma que se optimice el espacio ocupado en memoria. Indica cuántos bytes de memoria se ahorran al almacenar s2. :;

Cognoms: . . . . . Nom: . . . . . .

1er Control Arquitectura de Computadors **Curs 2012-2013 Q2**

Problema 3. (3 puntos)

Dado el siguiente código escrito en C:

```
void Subr1 (int *a, int *b, int c);

int Subr2(int *d) {
  int i;
  int local=0;

  for (i=0;i<100;i+=2)
    Subr1(&local, d, *d);

  return local;
}
```

a) Dada una rutina que en su interior llama a otra, indica qué registros debe salvar la rutina que realiza la llamada y qué registros debe salvar la rutina a la que llama..

b) En el caso de la rutina Subr2 anterior, explica qué diferencias puede haber entre utilizar el registro %ebx o el registro %ecx para almacenar la variable de control del bucle i.

c) Traduce a ensamblador del x86 la rutina Subr2.