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.

Ciclos = $500x10^6$ i * 5 c/i + $200x10^6$ i * 3 c/i + $500x10^6$ c * 7c/i = $6,6x10^9$ ciclos

Texe = $6,6x10^9$ ciclos / $2,2x10^9$ ciclos/s = 3 segundos

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

CPI = $6,6x10^9$ ciclos / $1,2x10^9$ i = 5,5 c/i

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

MIPS = $1200x10^6$ instr / (3 s *10^6) = 400 MIPS

MFLOPS = $1800x10^6$ ops / (3 s *10^6) = 600 MFLOPS

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

Instrucciones SIMD

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?

Ganancia de B respecto A es la relación entre las energias consumidas: G = E_A/E_B

B tiene ganancia de 2 respecto A significa que: A consume el doble (o que B consume la mitad)

B tiene ganancia de 50% respecto A significa que: A consume un 50% mas que B (Ganancia de 1,5).

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.

Amdhal

G = 1 / (1-fm + fm/Gm) = 1/(0,25+0,75/3) = 2

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.

P fugas = I*V = 10 A * 1,2 V = 12W

Energia conmutación = $500x10^6$ i * 114 nJ/i + $200x10^6$ i * 75 nJ/i + $500x10^6$ c * 144 nJ/i = 144 Joules
P conmutación = 144 Joules / 3 s = 48 W

P total = 12W + 48 W = 60W
eficiencia = 600 Mflops / 60 W = 10 Mflops/W

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.

MTTF = 1/(1/100000+1/1000000+1/100000+1/200000+4/1000000+2/125000+2/500000) = **20000 horas**

MTBF = MTTF + MTTR = **20005 horas**

disponibilidad = 20000 h / 20005 h * 100 = **99,975%**

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.

s1 (28 bytes)

1	a	+0
3	---	+1
4	b	+4
2	c	+8
1	d	+10
1	---	+11
4	e[0]	+12
4	e[1]	+16
4	e[2]	+20
4	e[3]	+24

s2 (2804 bytes)

28	f[0]	+0
28	f[1]	+28
	...	
28	f[i]	+28*i
	...	
28	f[99]	+2772
4	d	2800

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.

2	u	-8
1	v	-6
1	--	-5
4	w	-4
4	ebp old	<--%ebp
4	ret	+4
28	j	+8
4	*k	+36
1	m	+40
3	---	+41
2	n	+44
2	---	+46

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:

```
@x(s2) + 28*M[@y(s1) + 12 + 4*i] + 10
```

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;`

```
M[8+%ebp + 12 + w*4] <- 0 o bien, más óptimo:
M[%ebp + 20 + w*4] <- 0

opción 1:
movl -4(%ebp), %eax; w
leal 8(%ebp, %eax, 4), %eax ; %eax <- 8+%ebp+w*4
movl $0, 12(%eax)

opción 2:optimizado
movl -4(%ebp), %eax; w
movl $0, 12+8(%ebp, %eax, 4), %eax
```

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. :;

```
char a;
char d;
short c;
int b;
int e[4]

o también

short c;
char a;
char d;
int b;
int e[4]; Nos ahorramos 4 bytes en s1 y 400 bytes en 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..

La que llama: eax, ecx y edx

La llamada: ebx, esi y edi

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.

En el caso de utilizar %ecx hay que hacer un push y un pop en cada iteración del bucle, si se usa ebx el push y el pop se realizan una vez al principio y final de la rutina.

c) Traduce a ensamblador del x86 la rutina Subr2.

```
Subr2:    pushl %ebp
          movl %esp, %ebp
          subl %esp, 8
          movl $0, -4(%ebp)
          movl $0, -8(%ebp)
bucle:    cmpl $100, -8(%ebp)
          jl finbucle
          movl 8(%ebp), %eax
          pushl (%eax)
          pushl %eax
          leal -4(%ebp), %eax
          pushl %eax
          call Subr1
          addl $12, %esp
          addl $2, -8(%ebp)
          jmp bucle
fbucle:   movl -4(%ebp), %eax
          movl %ebp, %esp
          popl %ebp
          ret
```