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

1er Control Arquitectura de Computadors **Curs 2011-2012 Q2**

Problema 1. (4 puntos)

Disponemos de una estación de trabajo con un procesador a 3.0 Ghz. Un programa de prueba P realiza 300 millones de operaciones de punto flotante. La ejecución de P tarda 1,5 segundos, y su CPI es 6 ciclos/instrucción.

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

MIPS = $f/(CPI*10^6) = 3x10^9/ (6*10^6) = 500$ MIPS

MFLOPS = $300x10^6/ (1,5*10^6) = 200$ MFLOPS

b) Calcula el número de instrucciones ejecutadas por P.

Texe = N * CPI * Tc ----> N = Texe * f / CPI

N(P) = 1,5 * $3x10^9$ / 6 = $750x10^6$ instrucciones

Podemos considerar que en nuestro procesador sólo existen 2 tipos de instrucciones, las de coma flotante y las de enteros. Sabemos que las de enteros tienen un CPI de 3 c/i y que para realizar los 300 millones de operaciones de punto flotante se han ejecutado 500 millones de instrucciones de punto flotante.

c) **Calcula** el CPI de las instrucciones de coma flotante en P.

6 c/i = (3 c/i * $250x10^6$ + CPIF * $500x10^6$)/$750x10^6$ -> 6*$750x10^6$ - 3 c/i * $250x10^6$ = CPIF * $500x10^6$

CPIF = 7,5 c/i

El fabricante del procesador está estudiando una modificación en el mismo que supondría una ganancia de 1,2 en las instrucciones de coma flotante y una ganancia de 0,75 en las instrucciones de enteros.

d) ¿Es beneficiosa esta mejora cuando ejecutamos P? Justificad la respuesta.

ciclos CF = $500x10^6$ * 7,5 = $3750x10^6$ ciclos --> 83,3%

ciclos ENT = $250x10^6$ * 3 = $750x10^6$ ciclos --> 16,7%

Ganancia = 1/((0,833/1,2)+(0,167/0,75)) = 1,09

El nuevo procesador es ligeramente más rápido.

Esta CPU, tiene una carga capacitiva equivalente de 12,5 nF (nanofaradios), y una corriente de fugas de 10 A y funciona a un voltaje de 1,2 V.

e) **Calcula** la potencia media debida a fugas, la debida a conmutación y la total para el programa P.

P fugas = I*V = 10 A * 1,2 V = 12W
Pconmutacion = $C*V^2*f = 12,5x10^{-9}$ F * $(1,2\ V)^2$ * $3x10^9$ Hz = 54 W
Ptotal = 12 W + 54 W = **66 W**

Este servidor está formado por los componentes 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	Ventilador CPU	Placa base	DIMMs	Disco duro
Nº	1	1	1	1	4	1
MTTF (horas)	100.000	1.000.000	100.000	200.000	1.000.000	100.000

El tiempo medio para reemplazar un componente que ha fallado (*mean time to repair*) es de 10 horas y la probabilidad de fallo sigue una distribución exponencial.

f) **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+1/100000) = **25000 horas**

MTBF = MTTF + MTTR = **25010 horas**

disponibilidad = 25000 h / 25010 h * 100 = **99,96%**

En los cálculos anteriores se ha supuesto que solo puede haber fallos de hardware, sin embargo hay otros aspectos que pueden influir en la fiabilidad de un sistema, cómo la estabilidad del software o la propia red eléctrica. Sabemos que el sistema operativo usado en nuestro servidor tienen un tiempo medio entre fallos de 15000 horas y la inestabilidad de la alimentación eléctrica (microcortes, caídas de tensión, etc) provoca un fallo cada 745 horas en media. Los fallos provocados tanto por el sistema operativo como la red eléctrica siguen también una distribución exponencial.

g) **Calcula** el tiempo medio hasta fallos del sistema (MTTF) teniendo en cuenta la combinación del hardware, el SO y la red eléctrica. ¿crees que valdría la pena gastar mucho más en un computador cuyos componentes hardware tienen el doble de MTTF (respecto la tabla anterior)? Justifica la respuesta.

MTTF = 1/(1/25000+1/15000+1/745 = **690 horas**

No vale la pena, el elemento mas influyente en el MTTF es la red electrica.

Apartado para nota: Cuando se produce un fallo (sea causado por el hardware, el SO o por un fallo eléctrico), se pierden los datos de la aplicación que se estaba ejecutando en ese momento y hay que volverla a ejecutar una vez el computador vuelve a estar en funcionamiento. En nuestro servidor queremos ejecutar una aplicación que tarda 1000 horas en ejecutarse, si durante su ejecución hay una fallo se deberá ejecutar de nuevo desde el principio. Por simplicidad supondremos que el tiempo de poner en marcha de nuevo el computador y la aplicación es negligible (**MTTR = 0**)

h) **Calcula** la probabilidad de que el computador acabe la ejecución de la aplicación sin que se produzca ningún fallo. **Calcula** cuantas veces (en media) habrá que ejecutar la aplicación hasta que acabe completamente.

P acabar = 1 - P fallo = $1 - (1 - e^{-t/MTTF}) = e^{-t/MTTF} = e^{-1000/690} = 0.235$

veces = 1/p = 4,26

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

1er Control Arquitectura de Computadors **Curs 2011-2012 Q1**

Problema 2. (3 puntos)

Dada el siguiente código escrito en C:

```
typedef struct {
  int a;
  char ca;
  unsigned short us[5];
  char *p;
  char cb;
} ss;

int Test(ss X1, ss *ps) {
  int i;
  ss aux;
  ...
}
```

a) **Dibujad** como quedaría almacenada la estructura de datos en linux, indicando claramente los desplazamientos respecto al inicio de la estructura y el tamaño de ésta. Dibujad el bloque de activación de la rutina Test.

ss (24 bytes)

4	a	+0
1	ca	+4
1	---	
10	us[5]	+6
4	*p	+16
1	cb	+20
3	--	

Test

4	i	-28
24	ss	-24
4	%ebp	<--%ebp
4	@ret	
24	X1	+8
4	*ps	+32

En las siguientes preguntas, suponed que el código en C está dentro de la rutina Test.

b) **Escribid** una secuencia de 3 instrucciones x86 que realice la siguiente asignación:

ps->a = i;

```
movl -28(%ebp), %eax
movl 32(%ebp), %ebx
movl %eax, (%ebx)
```

c) **Escribid** una secuencia de 2 instrucciones x86 que realicen la siguiente asignación:

aux.p = &X1.cb;

```
leal 28(%ebx), %eax
movl %eax, -8(%ebx)
```

d) **Escribid** una secuencia de 3 instrucciones x86 que realicen la siguiente asignación:

e) `ps->us[i] = ps->us[i] + 1;`

```
movl 32(%ebp), %edx
movl -28(%ebp), %eax
incl 6(%edx, %eax, 2)
```

f) **Traducid** a x86 la siguiente asignación:

`i = Test(aux, &X1);`

```
leal 8(%ebp), %eax
pushl %eax
pushl -4(%ebp)
pushl -8(%ebp)
pushl -12(%ebp)
pushl -16(%ebp)
pushl -20(%ebp)
pushl -24(%ebp)
call Test
addl $28, %esp
movl %eax, -28(%ebp)
```

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

1er Control Arquitectura de Computadors **Curs 2011-2012 Q1**

Problema 3. (3 puntos)

Dada el siguiente código escrito en C:

```
int Unknown(int v[], char *x);
int Examen(int v[], char c, int *N){
  int i;

  if (c<'a' || c>'z')
    i = *N;
  else
    i = 0;
  return (i+Unknown(&v[0], &c));
}
```

a) **Dibujad** el bloque de activación de la subrutina examen.

Examen

4	i	-4
4	ebp old	<--%ebp
4	ret	
4	&v	+8
1	c	+12
3	---	
4	&N	+16

b) **Traducid** la subrutina examen a ensamblador del x86:

```
Examen: pushl %ebp
        movl %esp, %ebp
        subl $4, %esp
        cmpb $'a',12(%ebp)
        jb then
        cmpb $'z',12(%ebp)
        ja then
        jmp else
then:   movl 16(%ebp),%eax
        movl (%eax), %eax
        movl %eax, -4(ebp)
        jmp finif
else:   movl $0, -4(%ebp)
finif:  leal 12(%ebp), %eax
        pushl %eax
        pushl 8(%ebp)
        call Unknown
        addl $8, %esp
        addl -4(%ebp), %eax
        movl %ebp, %esp
        popl %ebp
        ret
```