

COGNOMS: NOM:

1er Control Arquitectura de Computadors

Curs 2016-2017 Q1

- Temps: 13:30 a 15:00
- Poseu clarament amb LLETRES MAJÚSCULES a cada full els cognoms i el nom

Problema 1. (3 puntos)

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

```
typedef struct {
    char c;
    char d;
    short e[2];
    char f[4];
    float *g;
    int h;
    short i;
} s1;

typedef struct {
    short u;
    s1 v[100];
    short w;
} s2;
```

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

c <- 0	----- <- 0
d <- 1	-- u
e[0] d c e <- 2	----- <- 4
----- <- 4	v[0]
f[1] f[0] e[1] f <- 6	----- <- 28
----- <- 8
---- f[3] f[2]	-----
----- <- 12	v[99]
*g	----- <- 2404
----- <- 16	--- w
h	----- <- 2408
----- <- 20	
---- i	

Tamaño s1: 24 bytes	Tamaño s2: 2408 bytes

- b) **Escribe** UNA ÚNICA INSTRUCCIÓN que permita mover **x.v[10].d** al registro **%al**, siendo **x** una variable de tipo **s2** cuya dirección está almacenada en el registro **%ecx**. Indica la expresión aritmética utilizada para el cálculo de la dirección.

La expresión aritmética para calcular la dirección del operando es: **@x+4+24*10+1**, por lo tanto **%ecx+245**
movb 245(%ecx), %al

- c) **Escribe** UN CONJUNTO DE 2 INSTRUCCIONES que permita mover **x.v[y.h].f[2]** al registro **%al**, siendo **x** una variable de tipo **s2** cuya dirección está almacenada en el registro **%ecx** e **y** una variable de tipo **s1** cuya dirección está almacenada en el registro **%ebx**. Indica la expresión aritmética utilizada para el cálculo de la dirección.

La expresión aritmética para calcular la dirección del operando es: **@x+4+M[@y+16]*24+8**
imul \$24, 16(%ebx), %eax
movb 12(%ecx, %eax), %al

COGNOMS: NOM:

1er Control Arquitectura de Computadors

Curs 2016-2017 Q1

- Poseu clarament amb LLETRES MAJÚSCULES a cada full els cognoms i el nom

Problema 2. (3 puntos)

Dado el siguiente código escrito en C:

```
int examen(int a[5], int b[5], int x) {  
    int c[5], y;  
    y=1;  
  
    ...  
}
```

- a) **Dibuja** el bloque de activación de la rutina examen, indicando claramente los desplazamientos respecto a **%ebp** y el tamaño de todos los campos.

examen

20	c[]	-24
4	y	-4
4	ebp viejo	<- %ebp
4	dir. retorno	
4	@a	+8
4	@b	+12
4	x	+16

- b) **Traduce** a ensamblador x86 el inicio de la rutina examen (tres primeras líneas en C hasta la sentencia `y=1;`) sabiendo que usa los registros **%eax**, **%ebx**, **%ecx** y **%edx**.

```
examen: pushl %ebp  
        movl %esp, %ebp  
        subl $24, %esp;  
        pushl %ebx  
        movl $1, -4(%ebp)
```

- c) **Traduce** a ensamblador x86 la siguiente sentencia en C que se encuentra en el cuerpo de la rutina examen.
`y = examen(b, c, a[3]);`

```
        movl 8(%ebp), %ebx  
        pushl 12(%ebx)  
        leal -24(%ebp), %eax  
        pushl %eax  
        pushl 12(%ebp)  
        call examen  
        addl $12, %esp  
        movl %eax, -4(%ebp)
```

COGNOMS: NOM:

1er Control Arquitectura de Computadors

Curs 2016-2017 Q1

- Poseu clarament amb LLETRES MAJÚSCULES a cada full els cognoms i el nom

Problema 3. (4 puntos)

En un ordenador en el que tenemos instalado el entorno usado en el laboratorio de AC, hemos medido que un programa de 1000 instrucciones ensamblador se ha ejecutado en 2 segundos usando 6×10^9 ciclos y ha ejecutado $4,8 \times 10^9$ instrucciones y $2,4 \times 10^8$ operaciones de coma flotante que debido a la falta de hardware específico ejecutan 10 instrucciones cada una.

- a) **Calcula** el CPI del programa y la frecuencia de la CPU (usa el prefijo del sistema internacional más adecuado).

$$\text{CPI} = 6 \times 10^9 \text{ ciclos} / 4,8 \times 10^9 \text{ instrucciones} = 1,25 \text{ c/i}$$

$$\text{Frec} = 6 \times 10^9 \text{ ciclos} / 2 \text{ segundos} = 3 \times 10^9 \text{ ciclos/segundo} = 3 \text{ GHz}$$

- b) **Calcula** los MIPS y MFLOPS a los que se ejecuta el programa.

$$\text{MIPS} = 4,8 \times 10^9 \text{ instrucciones} / 2 \text{ segundos} / 10^6 = 2400 \text{ MIPS}$$

$$\text{MFLOPS} = 2,4 \times 10^8 \text{ operaciones} / 2 \text{ segundos} / 10^6 = 120 \text{ MFLOPS}$$

El tiempo de ejecución usado en el primer apartado se corresponde al tiempo de CPU (usuario + sistema). Usando el comando “time” de linux hemos obtenido que el tiempo de CPU representa solo el 20% del tiempo total del programa (wall time). El 80% restante es tiempo de entrada/salida (accesos al disco duro concretamente). Cada acceso al disco duro del sistema tarda 8 milisegundos, mientras que si los datos estuviesen en un disco SSD cada acceso tardaría 10 microsegundos.

- c) **Calcula** la ganancia de la parte de entrada/salida si los datos del programa estuviesen en un SSD en lugar de en un disco duro.

$$\text{Ganancia} = 8 \times 10^{-3} \text{ segundos} / \text{acceso} / 10 \times 10^{-6} \text{ segundos} / \text{acceso} = 800$$

- d) **Calcula** la ganancia total en el programa que se obtendría con el cambio de tipo de disco.

$$\text{Ganancia} = 1 / ((1 - f_m) + f_m / g_m) = 1 / ((1 - 0,8) + 0,8 / 800) = 4,975$$

A pleno rendimiento, la CPU funciona a una frecuencia de 3 GHz y está alimentada a 1,6 V. En modo bajo consumo la CPU funciona a una frecuencia de 0,8 GHz y está alimentada a 1 V. Hemos medido que el consumo de la CPU en alto rendimiento es de 120W y en modo bajo consumo es de 25 W. En estos datos solo se considera la potencia debida a conmutación y la debida a fugas. Tanto la corriente de fugas (I) como la carga capacitiva equivalente (C) son las mismas en ambos modos.

- e) **Calcula** la corriente de fugas (I) y la carga capacitiva equivalente (C) de la CPU
(usar prefijo más adecuado del SI) .

$(1,6 \text{ V})^2 \cdot 3 \times 10^9 \text{ Hz} \cdot C + 1,6 \text{ V} \cdot I = 120 \text{ W}$ (alto rendimiento)
 $(1 \text{ V})^2 \cdot 0,8 \times 10^9 \text{ Hz} \cdot C + 1 \text{ V} \cdot I = 25 \text{ W}$ (bajo consumo)
resolvemos sistema de 2 ecuaciones lineales con 2 incógnitas
 $C = 12,5 \text{ nF}$ $I = 15 \text{ A}$

- f) **Calcula** la ganancia en energía que tendría el sistema si ejecutara el programa en el modo de bajo rendimiento en vez de en el modo de alto rendimiento suponiendo que el CPI medio no varía.

$E_{AR} = 120 \cdot 2 = 240 \text{ J}$ $E_{BR} = 25 \cdot 2 \cdot 3 / 0,8 = 187,5 \text{ J}$ $G = 240 / 187,5 = 1,28$