Departamento de Informática
Universidad de Valladolid

Escuela Técnica Superior de Ingeniería Informática
Camino del Cementerio s/n. 47011 Valladolid

11 de febrero de 2010 16h.

EXAMEN ORDINARIO DE ORGANIZACIÓN DE COMPUTADORES

NOTA: *Los alumnos con las prácticas pendientes deben sacar una nota mínima de 2 en el primer problema para superar la parte práctica de la asignatura.*

1 (3 p.) Sea una matriz bidimensional de números enteros de n filas por m columnas que está almacenada por filas a partir de la dirección `A` de la memoria de un computador con arquitectura SPARC.

a) Escribir los directivos para reservar el espacio de memoria necesario para almacenar la citada matriz suponiendo que $n = 8$ y $m = 6$.

Solución:

El número de componentes de la matriz será $nm = 8 \times 6 = 48$, el tamaño mejor para representar un entero en esta arquitectura son 4 bytes, con lo que el espacio necesario para almacenar la matriz, en bytes, será $48 \times 4 = 192$. Dado que se precisa alineación, los directivos necesarios para reservar espacio para la matriz A son:

```
   .align 4
A: .skip 192
```

⊗

b) Escribir una función en lenguaje ensamblador de SPARC que admita los siguientes parámetros (en ese mismo orden):

1. Dirección de la matriz.
2. Número de filas.
3. Número de columnas.

La función debe devolver, en un cuarto parámetro, la suma de todas las componentes de la matriz cuyo índice de fila sea par, a este efecto debe suponerse que los índices comienzan en 1.

Solución:

```
suma:          save %sp,-64,%sp
               clr %i3            ! Inicialización de la salida
               mulx %i2,4,%l0     ! Memoria ocupada por cada fila
               mov %i1,%l2        !%l2: contador de fila
Buclefila:     add %i0,%l0,%i0    ! Saltamos las filas impares
               mov %i2,%l1        !%l1: contador de columna
Buclecolumna:  ld [%i0],%l4       ! Carga de la componente
               add %l4,%i3,%i3
               add %i0,4,%i0
               deccc %l1
               bgt Buclecolumna
               nop
               deccc 2,%l2
               bgt Buclefila
               nop
               ret
               restore
```

⊗

```
       MOV #C, R2
       MOV N, R3
Bucle: CMPB (R2), #97.
       BLT Fin
       CMPB (R2), #122.
       BGT Fin
       BICB #32., (R2)
Fin:   INC R2
       SOB R3, Bucle
```

(a)

Dirección	Contenido
025166	010140
025170	066101
025172	060546
025174	042105
025176	000000
025200	021527
025202	000005
025204	177777

(b)

Figura 1.

2 (2.5 p.) En la memoria de un PDP-11 se encuentra el fragmento de programa mostrado en la figura 1 (a):

a) Escribir dicho programa en código máquina suponiendo que los símbolos `C`, `N` y `Bucle` representan, respectivamente, a las direcciones $025170_{(8}$, $025202_{(8}$, y $037674_{(8}$.

Solución:

Dirección	*Contenido*	*Comentarios*
037664	*012702*	*MOV #C, R2*
037666	*025170*	*C*
037670	*016703*	*MOV N, R3*
037672	*165306*	*Desplazamiento para N*
037674	*121227*	*CMPB (R2), #97.*
037676	*000141*	$141_{(8}=97_{(10}$
037700	*002405*	*BLT Fin (desplazamiento = 5)*
037702	*121227*	*CMPB (R2), #122.*
037704	*000172*	$172_{(8}=122_{(10}$
037706	*003002*	*BGT Fin (desplazamiento = 2)*
037710	*142712*	*BICB #32., (R2)*
037712	*000040*	$40_{(8}=32_{(10}$
037714	*005202*	*INC R2*
037716	*077312*	*SOB R3, Bucle (desplazamiento = 12)*

⊗

b) Explicar qué cambios produciría ese programa sobre los registros y la memoria, si ésta se encuentra inicialmente en la situación mostrada en la figura 1 (b).

Solución:

En la siguiente tabla puede verse las alteraciones que van provocando las instrucciones del programa sobre la memoria o los registros:

	Registros (octal)			Direcciones de memoria (bytes en hex.)				
Instrucción	*PC*	*R2*	*R3*	*025170*	*025171*	*025172*	*025173*	*025174*
Situación inicial	*037664*	*?*	*?*	*41*	*6C*	*66*	*61*	*45*
MOV #C, R2	*037670*	*025170*						
MOV N, R3	*037674*		*5*					
CMPB (R2), #97.	*037700*							
BLT Fin	*037714*							
INC R2	*037716*	*025171*						
SOB R3, Bucle	*037674*		*4*					
CMPB (R2), #97.	*037700*							
BLT Fin	*037702*							
CMPB (R2), #122.	*037706*							
BGT Fin	*037710*							
BICB #32., (R2)	*037714*				*4C*			
INC R2	*037716*	*025172*						
SOB R3, Bucle	*037674*		*3*					
CMPB (R2), #97.	*037700*							
BLT Fin	*037702*							
CMPB (R2), #122.	*037706*							
BGT Fin	*037710*							
BICB #32., (R2)	*037714*					*46*		
INC R2	*037716*	*025173*						
SOB R3, Bucle	*037674*		*2*					
CMPB (R2), #97.	*037700*							
BLT Fin	*037702*							
CMPB (R2), #122.	*037706*							
BGT Fin	*037710*							
BICB #32., (R2)	*037714*						*41*	
INC R2	*037716*	*025174*						
SOB R3, Bucle	*037674*		*1*					
CMPB (R2), #97.	*037700*							
BLT Fin	*037714*							
INC R2	*037716*	*025175*						
SOB R3, Bucle	*037674*		*0*					

⊗

c) Indicar cuál puede ser el propósito de ese código.

Solución:

El programa pasa a mayúsculas la cadena, almacenada en código ASCII, que está a partir de la dirección C con una longitud especificada en la dirección N. Para ello, analiza si cada byte (carácter ASCII) está entre 97 ('a') y 122 ('z') y, en caso de que lo esté (es decir, que sea una letra minúscula), pone a 0 el bit 5, que es el que diferencia las mayúsculas y las minúsculas.

⊗

3 (1.5 p.) Supongamos que los registros A y B de un procesador de 16 bits, que trabaja en complemento a 2, contienen, respectivamente, 73B4H y 85A3H. En esta máquina se ejecuta una instrucción que efectúa la siguiente operación de comparación: $tmp \leftarrow A - B$. Esta instrucción actúa sobre los bits de condición:

a) ¿Cómo quedará cada uno de esos bits de después de la ejecución de la citada instrucción?

Solución:

Para efectuar la operación $tmp \leftarrow A - B$, sumaremos a A el complemento a 2 de B, lo que nos da:

$$A + C_2(B) = 73B4H + C_2(85A3H) = 73B4H + 7A5DH = EE11H$$

Como resultado de esta última operación los bits de condición quedarán así:

- $Z = 0$ *(el resultado no es 0)*
- $N = 1$ *(el bit de más orden, el de signo, es 1)*
- $C = 0$ *(no hay llevada en el último bit)*
- $V = 1$ *(los dos sumandos son positivos, pero el resultado es negativo)*

⊗

b) Supongamos que después de la instrucción anterior se ejecuta la instrucción BLE (bifurcar si menor o igual) que analiza si $N \vee Z = 1$: ¿Se cumplirá la condición de bifurcación en este caso? ¿Es eso correcto? Razónese la contestación.

Solución:

Si la instrucción BLE analiza si $N \vee Z = 1$ nos dará verdadero, ya que N es 1; sin embargo, 73B4H no es menor que 85A3H, puesto que el primero es positivo y el segundo negativo. La bifurcación no da el resultado correcto porque no tiene en cuenta el bit de overflow (V) que nos indica que la operación no se ha realizado correctamente.

⊗

4 (1.5 p.) Sean a y b dos direcciones de memoria de un computador con arquitectura SPARC que representan sendos conjuntos A y B enmarcados en un conjunto universal de 32 elementos. Escribir las instrucciones necesarias en ensamblador de SPARC para poner un 1 en el registro $i0$ si $A \subseteq B$, y 0 en caso contrario.

Solución:

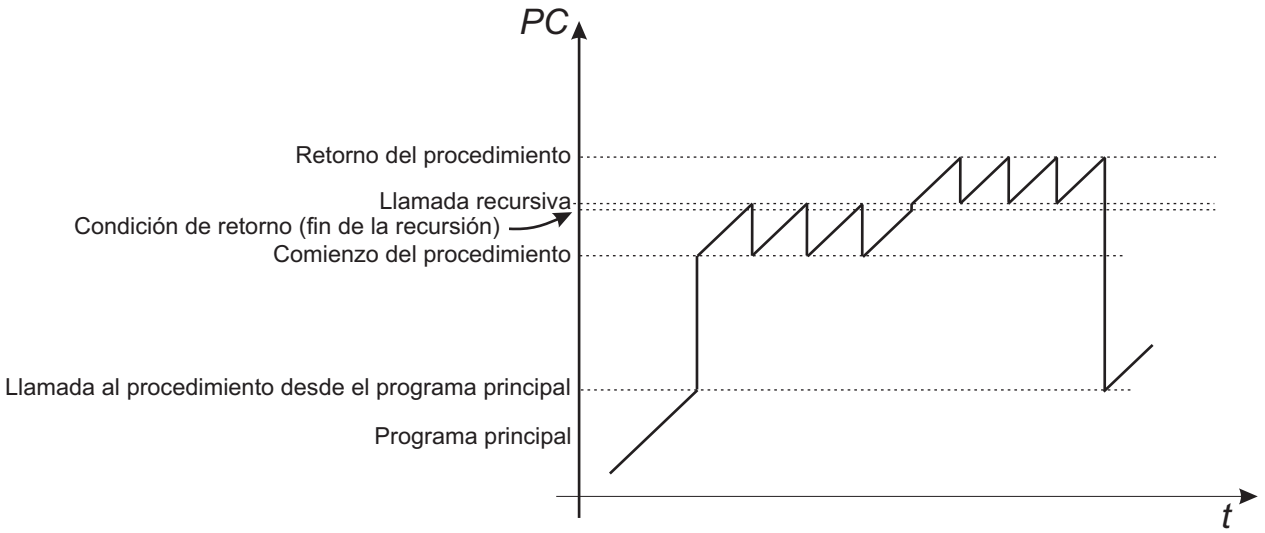
Para analizar si $A \subseteq B$ comprobaremos una de las condiciones equivamentes: $A \cap \overline{B} = \emptyset$, esto, referido a la cadena de bits, se transformará en $a \wedge \overline{b} = 0$, la instrucción que hace exactamente esto es `andn`*, por lo que el programa para analizar la condición solicitada es:*

```
        clr %i0
        set a, %l0
        set b, %l1
        ld (%l0), %l2
        ld (%l1), %l3
        andncc %l2, %l3, %g0
        bne fin
        nop
        mov 1, %i0
fin:
```

⊗

5 (1.5 p) Dibujar la gráfica de variación del contador de programa respecto al tiempo para un programa que llame a una función recursiva que se invoque a sí misma 3 veces. La llamada recursiva se produce hacia la mitad de la función, y, cuando se produce la condición de retorno, esta llamada se salta mediante una instrucción de bifurcación condicional.

Solución:



⊗