Universidad de Valladolid
Departamento de Informática
Escuela Universitaria Politécnica c/Francisco Mendizábal,1 47014 Valladolid Tel.:(983) 423501-02 Fax:(983) 423490

Ordenadores - I. T. Telecomunicación

Examen ordinario - 1 de febrero de 2.001
Soluciones

Problema 1

Apartado 1

Este programa almacena en `cx` el valor 9, lo guarda en la pila y llama a una rutina que incrementa dicho valor en uno. Al finalizar la rutina el valor se extrae de la pila, y finaliza el programa.

Apartado 2

La cache tiene 2 Kb y 32 bytes por bloque. Por lo tanto, tiene $2048/32 = 64$ bloques. Como es asociativa por conjuntos de cuatro vías, esto hace un total de $64/4 = 16$ conjuntos.

Desde el punto de vista de la cache, las direcciones de 20 bits se dividirán en los siguientes campos:

- Campo de palabra: 5 bits (ya que $2^5 = 32$).
- Campo de conjunto: 4 bits (ya que hay que direccionar 16 conjuntos diferentes).
- Campo de etiqueta: $20 - 4 - 5 = 11$ bits.

Apartado 3

Dado que el 8086 utiliza un mecanismo de segmentación, las direcciones efectivas se calculan multiplicando el valor del registro de segmento correspondiente por 16 (lo que equivale a desplazarlo 4 bits a la izquierda) y sumando al resultado el valor del desplazamiento dentro del segmento. En el caso del acceso a código, el registro de segmento es `cs` y el desplazamiento es el indicado por `ip`. En el caso de los accesos a datos (que no aparecen en este programa), el segmento de datos a utilizar será `ds`. En el caso de la pila, el segmento de pila viene dado por el registro `ss`, y el desplazamiento es el indicado por el puntero de pila `sp`.

Los accesos serán los que aparecen en la tabla 1. Los accesos en orden creciente a partir de la posición 0x4c680 son a código; los accesos en orden *decreciente* a partir de la posición 0x4c67e inclusive son a la pila.

La ejecución de este programa genera dos faltas frias en los conjuntos 4 y 3, la primera debido al acceso de un bloque de código y la segunda debido al acceso al bloque de pila. Cabe destacar que mientras que el bloque de código se recorre de atrás hacia adelante, los accesos a la pila en escritura se realizan desde el final hacia el inicio del bloque, mientras que en lectura el orden es el contrario.

Problema 2

Apartado 1

El algoritmo de suma en punto flotante es el siguiente:

1. *Alinear las mantisas: Tomar el número con menor exponente y desplazar su mantisa hacia la derecha (o, lo que es lo mismo, su coma a la izquierda) hasta que el exponente resultante sea el mismo.*

   En nuestro caso, A es un número positivo con exponente igual a 63 (ya que el campo de exponente vale 190 y está en exceso a 127) y mantisa igual a 1,4068604 aproximadamente. B es un número negativo con exponente igual a 0 (ya que el campo de exponente vale 127 y está expresado en exceso a 127) y mantisa igual a 1,75. Si alineamos las mantisas según el algoritmo, tenemos que desplazar la mantisa $1,11_b$ sesenta y tres posiciones a la derecha, lo que la situa fuera del campo de mantisa, que tiene 23 bits.

2. *Sumar o restar las mantisas, colocando el bit de signo al resultado.*

   Restando ambos números, obtenemos el resultado que puede verse en la figura 1. Como puede comprobarse en la figura, al restar la mantisa de B de la de A (ya que B es negativo), obtenemos un resultado que es prácticamente igual a A, salvo en los bits 22 y 23 de la mantisa.

3. *Normalizar el resultado, comprobando que no se produzca desbordamiento o subdesbordamiento.*

   En nuestro caso, el resultado está normalizado.

4. *Redondear la mantisa al número apropiado de bits.*

   El número a normalizar es el siguiente:

   Si tomamos 25 bits de la mantisa (los 23 más 2 de guarda y redondeo) del número a normalizar nos queda:

Mantisa	Guarda
1,010 1010 0000 0000 1011 1101	11

   Si aplicamos redondeo utilizando estos bits, el resultado es el siguiente:

Mantisa
1,010 1010 0000 0000 1011 1110

   es decir, el valor de A.

5. *Si el número obtenido no está normalizado, volver al paso 3.*

   El número obtenido está normalizado.

Puede verse que en este caso el resultado de A+B=A, debido a la falta de precisión del formato.

Universidad de Valladolid
Departamento de Informática
Escuela Universitaria Politécnica c/Francisco Mendizábal,1 47014 Valladolid Tel.:(983) 423501-02 Fax:(983) 423490

Dirección	Etiqueta	Conjunto	Palabra	¿R o W?	¿Fallo?	Acción
0x4c680	0x263	0x4	0x0	R	Si	`mov ax, datos`
0x4c681	0x263	0x4	0x1	R	No	
0x4c682	0x263	0x4	0x2	R	No	
0x4c683	0x263	0x4	0x3	R	No	`mov ds, ax`
0x4c684	0x263	0x4	0x4	R	No	
0x4c685	0x263	0x4	0x5	R	No	`mov cx,0x9`
0x4c686	0x263	0x4	0x6	R	No	
0x4c687	0x263	0x4	0x7	R	No	
0x4c688	0x263	0x4	0x8	R	No	`push cx`
0x4c67e	0x263	0x3	0x1e	W	Si	Escritura
0x4c67f	0x263	0x3	0x1f	W	No	a la pila
0x4c689	0x263	0x4	0x9	R	No	`call rutina`
0x4c68a	0x263	0x4	0xa	R	No	
0x4c67c	0x263	0x3	0x1c	W	No	Escritura
0x4c67d	0x263	0x3	0x1d	W	No	a la pila
0x4c692	0x263	0x4	0x12	R	No	`pop bx`
0x4c67c	0x263	0x3	0x1c	R	No	Lectura
0x4c67d	0x263	0x3	0x1d	R	No	de la pila
0x4c693	0x263	0x4	0x13	R	No	`pop cx`
0x4c67e	0x263	0x3	0x1e	R	No	Lectura
0x4c67f	0x263	0x3	0x1f	R	No	de la pila
0x4c694	0x263	0x4	0x14	R	No	`inc cx`
0x4c695	0x263	0x4	0x15	R	No	`push cx`
0x4c67e	0x263	0x3	0x1e	W	No	Escritura
0x4c67f	0x263	0x3	0x1f	W	No	a la pila
0x4c696	0x263	0x4	0x16	R	No	`push bx`
0x4c67c	0x263	0x3	0x1c	W	No	Escritura
0x4c67d	0x263	0x3	0x1d	W	No	a la pila
0x4c697	0x263	0x4	0x17	R	No	`ret`
0x4c67c	0x263	0x3	0x1c	R	No	Lectura
0x4c67d	0x263	0x3	0x1d	R	No	de la pila
0x4c68c	0x263	0x4	0xc	R	No	`pop cx`
0x4c67e	0x263	0x3	0x1e	R	No	Lectura
0x4c67f	0x263	0x3	0x1f	R	No	de la pila
0x4c68d	0x263	0x4	0xd	R	No	`mov ax,0x4c00`

Cuadro 1: Tabla de accesos del problema 1.

Mantisa simple precisión	Resto de bits
1,010 1010 0000 0000 1011 1110	0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 00
- 0,000 0000 0000 0000 0000 0000	0000 0000 0000 0000 0000 0000 0000 0000 0000 0001 11
1,010 1010 0000 0000 1011 1101	1111 1111 1111 1111 1111 1111 1111 1111 1111 1110 01

Figura 1: Resta de números en punto flotante del apartado 1, problema 2.

Universidad de Valladolid
Departamento de Informática
Escuela Universitaria Politécnica c/Francisco Mendizábal,1 47014 Valladolid Tel.:(983) 423501-02 Fax:(983) 423490

Apartado 2

Los números a dividir son infinito. Por lo tanto, el resultado de la división es NaN, que se representa con exponente igual a 127 y la mantisa distinta de 0: típicamente se elige el valor `0x7fc00000`.

Cuestión 1

Si no se inicializaran los bits de validez en los dos niveles de cache, se corre el riesgo de que algunas de las primeras solicitudes que la CPU realice a la memoria provoquen falsos aciertos caché, tanto en el acceso a datos como a código, y tanto para lectura o escritura. Si sólo se inicializaran los bits de la cache de nivel 1, ésta funcionará correctamente, pero un eventual fallo caché en L1 podría resolverse en L2 de forma incorrecta.

Cuestión 2

Apartado 1

El algoritmo de Booth se basa en la llamada "propiedad de la cadena". Supongamos que tenemos un número formado por una secuencia de unos entre las posiciones i y j. Este número será equivalente a un número formado por ceros y con un 1 en la posición $i+1$ menos un número formado por ceros y con un 1 en la posición j.

El algoritmo de Booth permite utilizar esta propiedad para acelerar el proceso de multiplicación. La idea es la siguiente: cuando nos encontramos al inicio de una cadena de unos (es decir, a la derecha de la cadena de unos), restamos el multiplicando del multiplicador. Si estamos al final de la cadena de unos, es decir, en el cero siguiente a la cadena, se suma el multiplicando al multiplicador. En otros casos no se hace nada.

Apartado 2

Completar