Universidad de Valladolid
Departamento de Informática

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

Ordenadores

2º I. T. Telecomunicación

Examen extraordinario
11 de julio de 2.002
Soluciones

Problema 1

Apartado 1: número 12,33

Para representar el número 12,33 puede utilizarse el algoritmo general, aunque resulta más sencillo pasar el número a binario y a continuación almacenarlo en el formato IEEE 754.

$$12,33_d = 1100,01010100011110101110_b \quad (1)$$

Una vez pasado a binario, hay que normalizar el número, que queda como

$$1,10001010100011110101110_b.2^3 \quad (2)$$

El campo de mantisa será igual a los dígitos que se encuentran a la derecha de la coma; el campo de signo será 0, ya que el número es positivo; y el campo de exponente será igual a $127 + 3 = 130$, que se codifica como 10000010. Por lo tanto, el resultado será el siguiente (los puntos indican la división en campos):

`0.10000010.10001010100011110101110 = 0x414547ae`

Apartado 2: número $-188,95.10^{25}$

El número es negativo: el campo de signo de la mantisa será 1, y a partir de ahora consideraremos el número como positivo.

$$N = 188,95.10^{25} = 2^X \rightarrow$$
$$X = \frac{ln(188,95.10^{25})}{ln2} = 90,61006307 \quad (3)$$

Una vez obtenido X, se separan la parte entera del exponente de la parte fraccionaria:

$$N = 2^{90,61006307} = 2^{90}.2^{0,61006307} = 1,526325934.2^{90} \quad (4)$$

Pasamos la mantisa a binario:

$$1,526325934_d = 1.10000110101111010100110_b \quad (5)$$

Dado que la mantisa está normalizada, sólo nos queda calcular el campo de exponente, que será 90 más el exceso. Dado que $90+127 = 217$, y que $217_d = 11011001_b$, el número en punto flotante será el siguiente (los puntos indican la división en campos):

`1.11011001.10000110101111010100110 = 0xecc35ea6`

Apartado 3: número $-0.0018.10^{-39}$

El número es negativo: como antes, se pone el bit de signo de la mantisa a 1 y se considera al número como positivo. Siguiendo el procedimiento usado en el número anterior, tenemos que

$$N = 0,0018.10^{-39} = 2^X \rightarrow$$
$$\rightarrow X = \frac{ln(0,0018.10^{-39})}{ln2} = -135.351055 \quad (6)$$

Una vez obtenido X, se separan la parte entera del exponente de la parte fraccionaria. *Cuidado:* Dado que $2^{e+f} = 2^e.2^f$, tenemos que:

$$2^{-135.351055} = 2^{-136}.2^{0.648945} \quad (7)$$

ya que la parte fraccionaria ha de ser siempre positiva. Puede verificarse que $-136 + 0.648945 = -135.351055$. Pasando la parte fraccionaria a binario, tenemos:

$$2^{0.351055} = 1,568021129_d = 1,10010001011010011101010_b \quad (8)$$

Dado que el exponente es -136, el número deberá representarse en formato no normalizado. En este formato, el campo de exponente valdrá 0000 0000, representando el exponente -126, y el número deberá normalizarse a dicho exponente. En consecuencia, tenemos que:

$1,10010001011010011101010_b.2^{-136}$ es igual a

$0.00000000011001000101101_b.2^{-126}$

Por lo tanto, el número en punto flotante será el siguiente (los puntos indican la división en campos):

`1.00000000.00000000011001000101101= 0x8000322d`

Apartado 4: número $1,2.10^{143}$

Dado que el número más grande representable en formato IEEE 754 de simple precisión es del orden de $2^{127} \sim 10^{38}$, este número es demasiado grande para poder ser representado, por lo que se almacena como "infinito". Por lo tanto, la respuesta será la siguiente (los puntos indican la división en campos):

`0.11111111.00000000000000000000000 = 0x7f800000`

Universidad de Valladolid
Departamento de Informática

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

Problema 2

Apartado 1

La cache de primer nivel (L1) es una cache asociativa por conjuntos de 4 vias. Por lo tanto, cada conjunto tendrá 4 marcos de bloque. Si la cache tiene 65536 bytes, y cada marco de bloque tiene 32 bytes, hay un total de $65536/32 = 2048$ marcos de bloque, y $2048/4 = 512$ conjuntos.

La cache de segundo nivel (L2) es una cache con correspondencia directa, por lo que no utiliza conjuntos. Dado que tiene 8Mbytes, y cada bloque tiene 64 bytes, la cache tendrá $8.388.608/64 = 131.072$ marcos de bloque.

Apartado 2

Dado que la cache de primer nivel (L1) es asociativa por conjuntos de 4 vias, esta cache dividirá las direcciones en tres campos. De derecha a izquierda, dichos campos son:

- Campo de byte dentro del bloque. Dado que cada bloque tiene 32 bytes, este campo tendrá 5 bits ($2^5 = 32$).
- Campo de conjunto. La cache L1 tiene 2048 marcos de bloque de 32 bytes. Como es asociativa por conjuntos de 4 vias, habrá un total de $2048/4 = 512$ conjuntos. Por lo tanto, el campo de conjunto tendrá 9 bits ($2^9 = 512$).
- Campo de etiqueta. Está formado por los restantes bits de la dirección: $36 - 9 - 5 = 22$.

Respecto de la cache de segundo nivel (L2), se trata de una cache con correspondencia directa. Sus campos son:

- Campo de byte dentro del bloque. Dado que cada bloque tiene 64 bytes, este campo tendrá 6 bits ($2^6 = 64$).
- Campo de bloque. La cache L2 tiene 8Mbytes. Como cada bloque tiene 64 bytes, la cache tendrá $8.388.608/64 = 131.072$ bloques, esto es, 2^{17} bloques, por lo que este campo tendrá 17 bits.
- Campo de etiqueta. Está formado por los restantes bits de la dirección: $36 - 17 - 6 = 13$.

Apartado 3

Suponiendo que ambas caches están frias (es decir, que todos los marcos de bloque que las componen son inválidos), el procesador pide el dato de 32 bits que está en la dirección `0x001234567`. La evolución de ambas caches es la siguiente:

1. La cache L1 recibe la solicitud. Deberá devolver al procesador los datos contenidos a partir de la dirección `0x001234567`. Dividiendo esta dirección en campos, tenemos que:

   `0x001234567 =`
   `00000000000100100011010000101011001111 =`
   `= 00000000000100100011010 000101011 00111`

2. La cache se dirige al conjunto 0 0010 1011_b (`0x02b`) en busca del bloque. Como todos los marcos de bloque de la cache L1 son inválidos, se produce un fallo cache en L1. En consecuencia, la cache L1 solicita a la cache de segundo nivel (L2) el bloque completo que contiene los datos buscados. Para ello, solicita a L2 el bloque de 32 bytes que comienza en la dirección

   `00000000000100100011010 000101011 00000`$_b$ =
   `0x001234560`

3. La cache L2 recibe la solicitud y divide la dirección en campos:

   `0x001234560` =
   `0000000000010 01000110100010101 100000`$_b$

4. La cache L2 se dirige a su marco de bloque número 0 1000 1101 0001 0101_b (`0x08d15`). Como el marco de bloque es inválido, se produce un nuevo fallo cache. En consecuencia, la cache L2 solicita a la memoria principal la transferencia del bloque completo.

5. El bloque de 64 bytes que comienza en la dirección

   `0000000000010 01000110100010101 000000`$_b$ =
   `0x001234540`

   se trae de la memoria principal, y se almacena en el marco de bloque `0x08d15` de la cache L2.

6. Dado que la solicitud de L1 indicaba que se deseaban 32 bytes de ese bloque, a partir de la posición `100000` (ver campo de "byte dentro del bloque" en el paso 3), la cache L2 transfiere a la cache L1 los bytes `100000` a `111111` de ese bloque, ambos inclusive. (Dicho en otras palabras, dado que los bloques de L2 son de 64 bytes y los de L1 son de 32 bytes, la solicitud de L1 a L2 provoca la transferencia de 32 bytes, lo que equivale a medio bloque de L2. En nuestro caso, se transfiere la segunda mitad, la que va de los bytes 32 hasta 63, ambos inclusive.)

7. La cache L1 recibe los 32 bytes y los almacena en uno de los marcos de bloque del conjunto `0x02b` (ver paso 2).

8. La cache L1 devuelve al procesador los cuatro bytes solicitados, que serán los correspondientes a las palabras `00111`$_b$, `01000`$_b$, `01001`$_b$ y `01010`$_b$ del bloque recibido desde L2.

Cuestión 1

La selección de un dispositivo de E/S concreto depende en gran medida de la arquitectura del computador. Existen dos soluciones habituales.

La primera es el uso de un **Bus específico de E/S**. Consiste en utilizar un bus para efectuar las operaciones de E/S (figura 1). Este bus es independiente del bus de acceso a la memoria. El bus tiene tres conjuntos de líneas: direcciones, datos y control. Las lineas de dirección lleva la dirección del dispositivo (para seleccionarlo) y las lineas

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

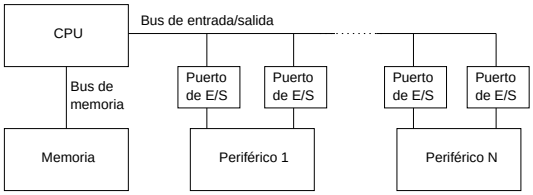

Figura 1: Organización de E/S con bus específico.

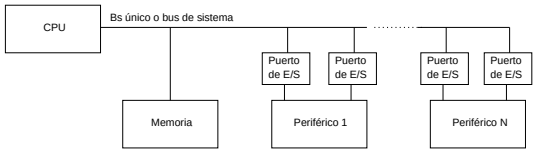

Figura 2: Organización de E/S con bus común.

de datos trasportan los datos. Las lineas de control se utilizan para gestionar el funcionamiento del bus. Muchas veces los dispositivos se conectan al bus a través de dos direcciones diferentes, denominadas *puertos de entrada/salida*. Se utiliza uno de esos puertos para enviarle al dispositivo las instrucciones y otro para enviarle los datos.

La segunda solución es el uso de un **Bus único o bus del sistema**. Consiste en utilizar el mismo bus que se utiliza para acceder a la memoria (figura 2). Con esta organización pueden usarse dos técnicas de selección de dispositivo:

1. E/S aislada: Existe una línea auxiliar de control (!IO/M), que indica si se trata de una operación de acceso a memoria o de acceso a un periférico de E/S. Si el acceso es a un dispositivo, se utilizan las líneas de orden más bajo del bus de direcciones para seleccionar el dispositivo. Requiere usar instrucciones específicas para acceder a los dispositivos. Ejemplos: Z-80, 8086.
2. E/S mapeada o E/S con memoria asignada: Esta técnica utiliza parte del espacio direccionable para acceder a la memoria y el resto para acceder a periféricos. Por lo tanto, los periféricos ocupan una parte del espacio direccionable, contrariamente a la E/S aislada. Ejemplos: Motorola 68000, VAX, PDP-11. Pueden usarse las mismas instrucciones que para acceder a memoria.

Ventajas e inconvenientes de ambos sistemas: La E/S aislada facilita la identificación de las instrucciones de E/S en un programa, y permite distinguir claramente entre accesos a memoria y a la E/S: los puertos de E/S son normalmente mucho más lentos, unidireccionales y muchas veces incapaces de retener el contenido enviado. Por otra parte, la E/S por mapa permite al programador utilizar el juego de instrucciones de acceso a memoria (es mucho más ortogonal), facilitando el flujo de información entre la memoria y la E/S. Además, el espacio de E/S es mas fácil de manejar al requerir menos señales de control.

Cuestión 2

Los requisitos mínimos que la arquitectura de una máquina debe cumplir para poder ejecutar subrutinas fueron descritos por Eckert en 1946. Son los siguientes:

1. **Debe existir una instrucción que permita regresar al programa principal**. Para ello deberá almacenarse en algún sitio la dirección de la instrucción inmediatamente siguiente a la llamada de la subrutina. Esta dirección puede almacenarse en un registro de propósito específico, en una posición fija de memoria o en la pila.
2. **Debe existir un mecanismo de paso de parámetros a la subrutina**. Se soluciona almacenando los parámetros de entrada en una posición de memoria determinada, o bien en la pila.
3. **Debe existir un mecanismo de devolución de resultados**. También se suele usar la pila.

La familia MIPS está diseñada de modo de acelerar al máximo las invocaciones a las subrutinas. Para ello, utiliza registros para almacenar la dirección de retorno, los parámetros y el resultado devuelto por la rutina. Los registros usados son los siguientes:

- Dirección de retorno: se almacena en un registro llamado `$ra`.
- Parámetros de entrada a la rutina: se almacenan en los registros `$a0`, `$a1`, `$a2` y `$a3`.
- Resultados: se almacenan en los registros `$v0` y `$v1`.

La función de invocación a la rutina es `jal etiqueta`, y se encarga de almacenar la dirección de salto en el registro `$ra` y de saltar. Para volver de la rutina, se ejecutará `jp $ra`

Si tenemos más parámetros de entrada o resultados de los que podemos manejar con registros, se usa la pila. Del mismo modo, si tenemos que saltar desde un procedimiento a otro, volcamos el contenido del registro `$ra` a la pila y ejecutamos la instrucción `jal`. Al volver, habrá que recuperar la dirección de retorno original.

Dado que en MIPS existe una instrucción de retorno al programa principal, un mecanismo de paso de parámetros a la subrutina y un mecanismo de devolución de resultados, esta arquitectura cumple con las condiciones arriba mencionadas.

Cuestión 3

Las memorias asociativas, también llamadas “memorias direccionables por contenido”, se caracterizan por identificar la información por uno de los campos de su propio contenido (campo clave).

Su funcionamiento es el siguiente: se pone el valor de la clave buscada en el registro de entrada y se compara simultáneamente con todos los valores de las claves. Si hay alguna coincidente se copia en el registro de salida. Si hay varias, el circuito de selección decidirá cuál usar: sacar una, sacar todas en un orden determinado, etcétera

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

En algunas memorias los bits correspondientes al campo clave no son fijos, sino que una *máscara* de bits, almacenada en un registro, indica en cada momento cuales son los bits que actúan como campo clave.

La principal aplicación de las memorias asociativas es la construcción de memorias capaces de almacenar las etiquetas utilizadas en las memorias cache.

Cuestión 4

Como se indica en el enunciado, el valor devuelto por `fcom` que codifica "desordenado" significa que los números no pueden compararse. Esta circunstancia puede darse cuando uno de ellos (o los dos) valgan `NaN`, ya que esa combinación no representa ningún número válido.