FUNDAMENTOS DE COMPUTADORES (20 de mayo de 2015)

1ª (2p)	2ª (2p)	3ª (2.5p)	4ª (2.5p)	5ª (1p)	*Nota (10p)*

NOMBRE: _____

1. Si tenemos un sistema que utiliza IEEE 754 de precisión simple y redondeo al más próximo, indica lo que imprimirá el siguiente código C (`uint8_t` indica un entero sin signo de 8 bits):

```
#include<stdio.h>
#include<stdint.h>
void main() {
      float f = -1023.4;
      uint8_t * u = (uint8_t *) &f;
      printf("%x-%x", *u, *(u+1));
}
```

   Suponiendo:

   a) Un sistema big-endian.

   b) Un sistema little-endian.

Solución: pasamos a binario:

1023 = `1111111111`

0.4 * 2 = **0**.8; 0.8*2=**1**.6; 0.6*2=**1**.2; 0.2*2=**0**.4; 0.4 * 2 = **0**.8; 0.8*2=**1**.6; 0.6*2=**1**.2; 0.2*2=**0**.4;

Es decir:

0.4 = `0.0110011001100110011001100110011…`

Por lo tanto:

1023.4 = `1111111111.0110011001100110011…` =>

`1023.4 = 1.11111111101100110011001`*`100110011`*`...*2`^{9}

Redondeando para dejar 23 cifras decimales (redondeo al más próximo):

`1023.4 = 1.11111111101100110011010 * 2`^{9}

El exponente en exceso = 127+9 = 136 = `10001000`

Por lo tanto, el número IEEE 754 SP será:

`1 10001000 11111111101100110011010`

En hexadecimal:

`1100 0100 0111 1111 1101 1001 1001 1010 = C4 7F D9 9A`

Por lo tanto, en big-endian escribirá C4-7F y en little endian 9A-D9

2. Conseguimos cambiar el código de un programa de forma que paralelizamos el 85% de las instrucciones, consiguiendo que se ejecuten (las instrucciones paralelizadas) N veces más rápido en un sistema multicore con N cores. Indica cuántos cores necesitaremos para conseguir que el código completo sea 4 veces más rápido.

Solución: aplicar la ley de Amdahl. Si el tiempo original es T, el tiempo mejorado es T'=T/4 =>

T'/T = ¼ = F/M + (1-F)

F es la fracción mejorada = 0,85, y M es lo que la mejoramos, que en este caso es igual al número de cores:

0,25 = 0,85/M + (1-0,85) => 0,1 = 0,85/M => M = 8,5 = 9 cores

3. Dado el siguiente código MIPS:

Dirección	Código instrucción	Instrucción
⋮	⋮	⋮
[0x00400010]	0x00094020	add ...
[0x00400014]	0x1128ffff	beq ...
⋮	⋮	⋮

Responde a las siguientes preguntas:

a) ¿Es posible saber si el salto se toma o no?

b) En el caso de que sea posible, ¿a qué dirección está saltando? ¿qué se ejecuta a continuación?

Solución: La instrucción add es de tipo R y tiene el siguiente formato:

000000 00000 01001 01000 00000 100000

por lo tanto rs=$0, rt=$9 y rd=$8, es decir, es la instrucción add $8, $0, $9. Como $0==0, después de la suma $8==$9

La instrucción beq es de tipo I y tiene el siguiente formato:

000100 01001 01000 1111111111111111

es decir, es beq $9, $8, destino, por lo que se salta.

La dirección destino del salto se obtiene a partir del dato, con el signo extendido y multiplicado * 4:

11111111111111111111111111111100 = -4

Por lo tanto, la dirección destino de salto es [PC+4] – 4 = PC, con lo que se ejecuta de nuevo el beq => entramos en un bucle infinito.

4. Supón un computador de 32 bits con una memoria cache de 64 KiB, asociativa por conjuntos de 8 vías. El tamaño de la línea es de 16 palabras y usa direccionamiento a nivel de byte. Indica:

a) El formato de una dirección desde el punto de vista de la caché. ¿Cuántos conjuntos tiene la cache?

b) El tamaño en bits del directorio caché, incluyendo el bit de validez. ¿Cuántos comparadores necesitaremos para determinar si hay acierto o fallo?

c) Considerando la siguiente secuencia de direcciones 0x0007B042, 0x0007D042, 0x0007D074, indica cuales ocasionan un fallo caché y por qué, suponiendo la cache inicialmente vacía. Indica también a qué conjunto va cada uno de los datos y a qué línea en el conjunto.

Solución:

a) N.º conjuntos = Tamaño cache/tamaño conjuntos

1 conjunto = 8 vías = 8 líneas. 1 línea = 16 palabras * 4 bytes =64 bytes => 1 conjunto = 8 * 64 = 512 bytes

N.º conjuntos = 64*1024/512 = 128 conjuntos = 2^7 conjuntos

Por lo tanto la dirección será

19	7	4	2
Etiqueta	Conjunto	Palabra/ línea	Byte/ palabra

b) Para calcular el tamaño del directorio necesitamos el número total de líneas:

N Líneas = N Ctos * lineas_cto = 128 * 8 = 1024 líneas

Por cada línea se guardan 20 bits (19 etiqueta + 1 validez) =>

Tamaño directorio = 1024 líneas * 20 bits = 20480 bits

Se necesita 1 comparador por vía => 8 comparadores

c)

0x0007B042 = `00000000000000111101 1000001 000010` => fallo, va al conjunto 65, a cualquier línea

0x0007D042 = `00000000000000111110 1000001 000010` => fallo, va al conjunto 65, a cualquier línea libre

0x0007D074 = `00000000000000111110 1000001 110100` => acierto, está en el conjunto 65

5. Responde brevemente a las siguientes preguntas

a) Describe los componentes del tiempo de acceso a un disco duro magnético.

b) Ventajas e inconvenientes de la entrada/salida por interrupciones y por DMA.