

Introducción a la Arquitectura de Sistemas

Parcial 04/06/2008

- 1) **a.** Como tengo 10 platos, y 2 caras por plato, el numero de pistas por cilindro es **20**.
Cada pista tiene 24 sectores de 260 bytes, entonces la capacidad por pista es de $24 \times 260 = \mathbf{6240 \text{ bytes}}$.
La capacidad total del disco es: 20 caras x 411 pistas/caras x 24 sectores/pista x 260 bytes/sector = **51292800 bytes**.
Tiempo = tiempo de espera (latencia) + tiempo de lectura/escritura
Tiempo de espera = $1/(2 \times (3600/60)\text{revoluciones/segundo}) = 0,008\hat{3}$ segundos
velocidad de transferencia = capacidad/pista x revoluciones/segundo
velocidad de transferencia = $6240 \times 120 = 748800 \text{ bytes/segundo}$
3 sectores son 780 bytes.
Entonces si transfiero 748800 bytes en 1 segundo, para transferir 780 bytes necesito $0,001041\hat{6}$ segundos.
El tiempo total para escribir 3 sectores es: $0,008\hat{3}$ segundos + $0,001041\hat{6}$ segundos = **0,009375 segundos**.
- b.** Para implementar un raid de nivel 5, se necesitaría al menos 3 discos.
La capacidad total del arreglo, sería $51292800 \text{ bytes} \times 3 \approx 146,749877 \text{ Mb}$.
Aunque como un disco sería de paridad, la capacidad utilizable del arreglo es de $51292800 \text{ bytes} \times 2 \approx \mathbf{97,83325195 \text{ Mb}}$.
La redundancia es 1 disco de 3, osea un **33,33333%**.
- 2) **a.** IEEE754 corta usa mantisa SVA(2,24), exponente CD(2,8) con frontera no equilibrada $((b^d)/2 - 1) = 127$, y normalización 1,X. Entonces :
 $0x018CCAB5_h = \underline{0000\ 0001\ 1000\ 1100\ 1100\ 1010\ 1011\ 0101}_b$.
 $0x0168144A_h = \underline{0000\ 0001\ 0110\ 1000\ 0001\ 0100\ 0100\ 1010}_b$.
Como los exponentes no son iguales, tenemos que hacer un corrimiento a derecha de la mantisa del segundo numero (corrimiento a izquierda de la coma). Recordando que la mantisa esta normalizada 1,X Entonces nos quedarían
 $0x018CCAB5_h = \underline{0000\ 0001\ 1000\ 1100\ 1100\ 1010\ 1011\ 0101}_b$.
 $0x0168144A_h = \underline{0000\ 0001\ 1111\ 0100\ 0000\ 1010\ 0010\ 0101}_b$.
Ahora como los exponentes están iguales podemos restar los números.
 $1,000\ 1100\ 1100\ 1010\ 1011\ 0101_b$.
 $- 0,111\ 0100\ 0000\ 1010\ 0010\ 0101_b$.
 $= 0,001\ 1000\ 1100\ 0000\ 1001\ 0000_b$.
Ahora el numero en IEEE754, recordando que la normalización es 1,X debemos hacer un corrimiento (de la coma) a derecha de 3 lugares y restarlos al exponente. Quedando:
 $\underline{0000\ 0000\ 0100\ 0110\ 0000\ 0100\ 1000\ 0000}_b$.
- b.** Frontera de CB(16,4) = $b^d = 65536 = 10000_h$
 $116 = 74_h$ entonces -116 en CB(16,4) es $10000_h - 0074_h = \mathbf{FF8C_h}$.
 $68 = 44_h = \mathbf{0044_h}$ en CB(16,4).

Ahora como CB no tiene factor de corrección $FF8C_h + 0044_h = \mathbf{FFD0}_h$.

c. Frontera de $CD(2,8) = ((b^d)/2) = 128 = 10000000$ (como no dice nada, la supongo equilibrada).

$12 = 1100_b$ entonces -12 en $CD(2,8)$ es $10000000_b - 1100_b = \mathbf{01110100}_b$.

$15 = 1111_b = \mathbf{00001111}_b$.

Como sabemos el rango de representación de $CD(2,8) = [-f, b^d - f - 1] = [-128, 127]$ y $-12 \times 15 = 180$, por lo que la multiplicación daría un overflow y no se podría representar.

- 3) a. El código busca al mayor valor de una lista o arreglo, dejándolo en r3. Para esto pregunta siempre que el valor sea distinto de 0. Esto indicaría el final de la lista.

Pseudo-código: utilizo un arreglo en vez de una lista porque es más fácil de entender.

```
rD=1301;
```

```
r1=arreglo[rD];
```

```
r3=arreglo[rD]
```

```
r2=arreglo[rD+1]
```

```
while (r2!=0){
```

```
    rD=r2;
```

```
    r4=arreglo[rD];
```

```
    r2=arreglo[rD+1];
```

```
    if ((r3-r4) < 0)
```

```
        r3=r4;
```

```
}
```

b. No es posible usar el rE en el lugar del r3, ya que es usado para guardar la dirección de retorno del call – ret.

c. Las microinstrucciones siempre van acompañadas del fetching.

FETCHING:

$$RI \leftarrow M(PC)$$
$$PC \leftarrow PC + 1$$

SHRA:

$$C \leftarrow R(RI[11..8])[0]$$
$$R(RI[11..8])[14..0] \leftarrow R(RI[11..8])[15..1]$$

- 4) a. Al ser vectorizada, no cambia el tamaño del archivo, ya que este depende de la cantidad de vectores de la imagen, y esta cantidad no cambiará al agrandar o achicar la imagen.

b. Los S.O. que poseen esta característica asigna los recursos disponibles (CPU, memoria, periféricos...) de forma alternada a los procesos que los solicitan, de manera que el usuario percibe que todos funcionan a la vez.

c. 1. Falso, es un protocolo de nivel 2, ethernet utiliza el protocolo CSMA/CD para transmitir, en el caso de que se esté transmitiendo espera, y sino transmite.

2. Falso, se utiliza para determinar errores de transmisión.