FUNDAMENTOS DE INFORMÁTICA

(14 de febrero de 2014; 2ª PARTE: ejercicios; 5 puntos)

1. Se tiene una imagen de 1600x900 píxeles de resolución, inicialmente en formato BMP (*truecolor*). Si, en su lugar, se utilizase el formato GIF con una paleta de 256 colores:

a) ¿Cuánto ocuparía la tabla de colores de la paleta?

b) ¿Cuánto ocuparía la información asociada a la propia imagen (conjunto de píxeles)?

c) ¿Cuál sería el factor de compresión del fichero BMP (*truecolor*) al representarlo en GIF?

RESPUESTAS

a) Ocupación de tabla de colores

Como la paleta tiene 256 mezclas de colores, y cada mezcla tiene 1 byte para nivel R, 1 byte para nivel G y un Byte para nivel B, la capacidad ocupada por la paleta será:

$$C_{paleta} = 3\ \frac{Bytes}{color}\ x\quad 256\ colores = 768\ Bytes = 0{,}75\ KBytes$$

b) Capacidad asociada a la propia imagen (píxeles)

Cada píxel se representa con el "código" o "índice" de cada una de las mezclas. Como hay 256 mezclas cada mezcla se representará con 8 bits (ya que 2^8=256); es decir, el color de cada píxel se representará con 1 Byte; con lo que la capacidad de la imagen propiamente dicha será:

$$C_{imagen} = (1.600\ x\ 900)\ píxeles\ x\ \ 1\frac{Byte}{píxel} = 1.440.000\ B = 1{,}37\ MBytes$$

c) Factor de compresión

La capacidad del fichero antes de comprimirlo (BMP, *truecolor*), teniendo en cuenta que por cada pixel se almacenan 1 byte para nivel R, 1byte para nivel G y 1 para nivel B, será:

$$C_a = (1.600\ x\ 900)\ píxeles\ x\ \ 3\ \frac{bytes}{píxel} = 4.320.000\ B = 4{,}12\ MBytes$$

Y la capacidad total del fichero después de comprimirlo será:

$$C_d = \ C_{paleta} + C_{píxeles}\ = 0{,}75\ KB + 1{,}37\ MB = 1{,}37\ MB$$

Con lo que el factor de compresión será:

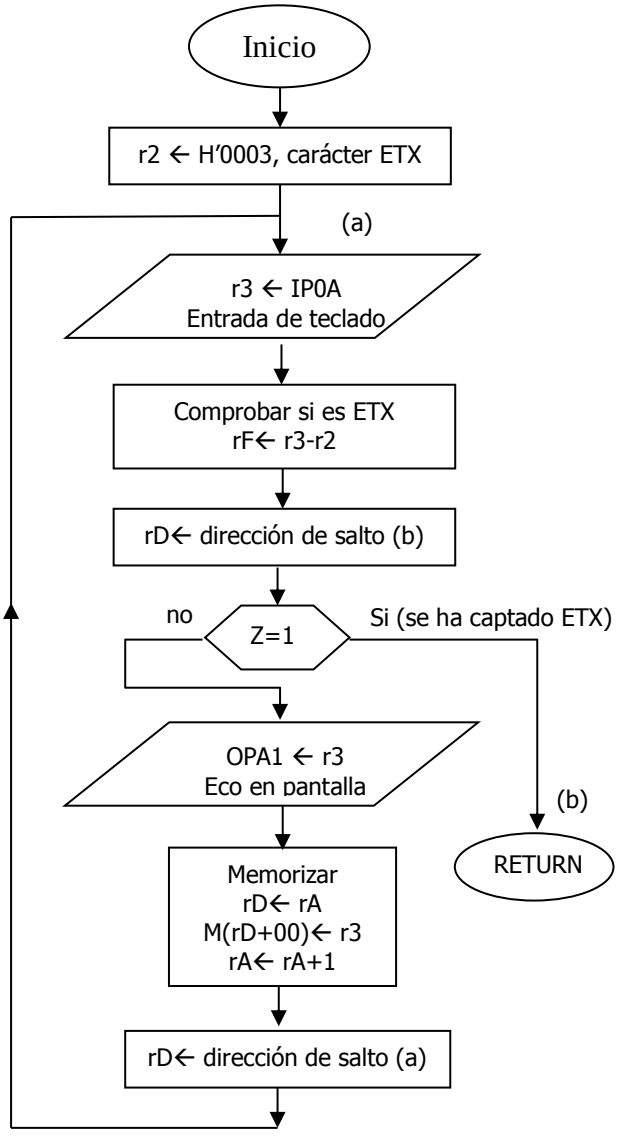
$$f_c = \frac{C_a}{C_d} = \frac{4{,}12\ MB}{1{,}37\ MB} = 3$$

Factor de compresión de 3 a 1.

2. Una implementación de un computador con CODE-2 tiene asociado el puerto *IPA1* a un teclado y el puerto *OPA1* a una pantalla. Hacer una subrutina en código máquina para CODE-2, a ubicar a partir de la dirección de memoria *A000*, que capte los caracteres que se vayan tecleando, los almacene a partir de la posición de memoria que se especifica en el registro *rA* y vaya haciendo un eco en la pantalla de los mismos. La subrutina debe concluir cuando se reciba del teclado el carácter de control *ETX* (fin de texto).

RESPUESTA

Al ser una subrutina suponemos que el programa principal (que la llama) se ha encargado de almacenar en r0 el valor H'0000 en r1 el valor H'0001, y en rA la dirección a partir de la cual hay que cargar la información de entrada. En r2 almacenaremos el código correspondiente al carácter de control ETX (consultada la tabla ASCII es: ETX → H'03). Un organigrama de lo que debe hacer el programa será



3. Suponga que el tiempo de ejecución de un programa es directamente proporcional al tiempo de acceso a sus instrucciones, y que el tiempo de acceso a una instrucción en la caché, t_{aC}, es diez veces menor que el tiempo de acceso a una instrucción en la memoria principal, t_{aMP}. Admitiendo que una instrucción requerida por el procesador se encuentra en la caché con una probabilidad de $\tau_{aciertos,caché} = 0.8$, y que si un dato o una instrucción requerida por el procesador no se encuentra en la caché, a continuación de ser llevada de la memoria principal a la caché debe ser llevada de la caché al procesador, calcular:

a) La eficiencia lograda por la caché, considerado ésta como la relación entre el tiempo de ejecución sin la caché y el tiempo de ejecución con la caché.

b) Teniendo en cuenta que al duplicar el tamaño de la caché, la probabilidad de no encontrar en ella una instrucción buscada se reduce a la mitad, dibujar una gráfica que represente la eficiencia (tal como se ha definido en el apartado anterior) en función del tamaño de la caché, indicando cual es la eficiencia mínima (sin caché) y máxima (tamaño caché → ∞).

RESPUESTAS:

a) Eficiencia lograda por la caché.

Del enunciado del problema sabemos que el tiempo de acceso a memoria principal (t_{aMP}) en función del tiempo de acceso a caché (t_{aC}) es:

$$t_{aMP} = 10 \cdot t_{aC}$$

Por otra parte sabemos que la tasa (probabilidad) de aciertos más la tasa de fallos en un nivel determinado debe ser 1 (o hay acierto o hay fallo); es decir:

$$\tau_{aC} + \tau_{fC} = 1$$

Es decir:

$$\tau_{fC} = 1 - \tau_{aC} = 1 - 0{,}8 = 0{,}2$$

Del enunciado del problema, también deducimos que el tiempo de acceso efectivo a la caché será:

$$t_{eaC} = \tau_{aC} \cdot t_{aC} + \tau_{fC} \cdot (t_{aMP} + t_{aC}) = 0{,}8 \cdot t_{aC} + 0{,}2 \cdot (10 \cdot t_{aC} + t_{aC}) = 2{,}2 \cdot t_{aC}$$

Con lo que la eficiencia será:

$$\gamma = \frac{t_{aMP}}{t_{eaC}} = \frac{10 t_{aC}}{2{,}2 t_{aC}} = 4{,}55$$

b) Teniendo en cuenta que al duplicar el tamaño de la caché, la probabilidad de no encontrar en ella una instrucción buscada se reduce a la mitad, dibujar una gráfica que represente la eficiencia (tal como se ha definido en el apartado anterior) en función del tamaño de la caché, indicando cual es la eficiencia mínima (sin caché) y máxima (tamaño → ∞).

Debemos encontrar la eficiencia en función de la tasa de fallos en la caché, y obtener aquella para los siguientes valores: τ_{fC} = 1 (no hay caché), 0,1; 0,05; 0 (caché infinita, no se producen fallos).

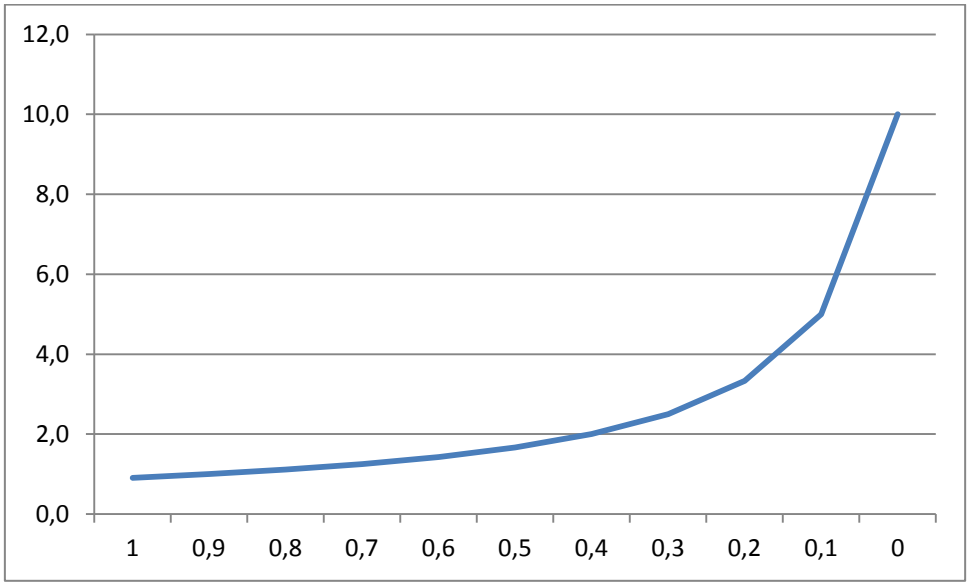
Siguiendo los razonamientos del apartado a) se tiene lo siguiente:

$$t_{eaC} = \tau_{aC} \cdot t_{aC} + \tau_{fC} \cdot (t_{aMP} + t_{aC}) = (1 - \tau_{fC}) \cdot t_{aC} + \tau_{fC} \cdot (10 \cdot t_{aC} + t_{aC})$$
$$= (1 - \tau_{fC}) \cdot t_{aC} + 10 \cdot \tau_{fC} \cdot t_{aC} = (1 + 10 \cdot \tau_{fC}) \cdot t_{aC}$$

Con lo que la eficiencia será:

$$\gamma = \frac{t_{aMP}}{t_{eaC}} = \frac{10 \cdot t_{aC}}{(1 + 10 \cdot \tau_{fC}) \cdot t_{aC}} = \frac{10}{1 + 10 \cdot \tau_{fC}}$$

- Sin caché (eficiencia mínima): τ_{fC} = 1; Υ = 0,91
- Caché del enunciado: τ_{fC} = 0,2; Υ = 3,33
- Duplicamos la capacidad de la caché: τ_{fC} = 0,1; Υ = 5
- Caché muy grande (no se produce ningún fallo): τ_{fC} = 0; Υ = 10



4. **Un sistema operativo gestiona la memoria principal de un computador utilizando la técnica de memoria virtual bajo demanda de páginas. Las direcciones virtuales en hexadecimal están formadas mediante la yuxtaposición de tres cifras de página y cuatro de desplazamiento. La memoria principal del computador está dividida en 256 marcos de página. En un momento dado, el contenido de la tabla de marcos de página afectados por un determinado proceso se muestra al margen.**

Tabla marcos de página		
Marco	***Contenido***	***Estado***
00 ... 07	S.O	**-**
A8	P1, DF7	**0**
A9	P1,1B9	**0**
AA	P1,1BA	**0**
AB	Libre	**1**
AC	P1, 0CB	**0**
AD	P1, 0CC	**0**
AE	P1, FF0	**0**
AF	P3, 0CC	0
….	…..	….

PREGUNTAS Y SOLUCIONES

a) Determinar la dirección física (en hexadecimal) a la que se accede cuando el proceso P1 referencia a la dirección 0CC905A.

Según el enunciado del problema las direcciones virtuales se forman con tres cifras HEX para especificar la página y cuatro cifras HEX para especificar el desplazamiento dentro de esa página con lo que se tiene:

dv= =0CC905A → pág. 0CC , offset: 905A

En la tabla de páginas de indica que la página 0CC del proceso P1 se encuentra en el marco de página AD, con lo que la dirección física será:

df= marco de página&offset → **AD905A**

b) Determinar la dirección virtual correspondiente a la dirección física A979AB

df = A979AB → marco de página A9 offset 79AB

En la tabla vemos que en el marco de página A9 se encuentra la página 1B9 del proceso P1, con lo que la dirección virtual a que corresponde esa dirección física es del proceso P1, y

dv = página & offset → **1B979AB**

c) Suponiendo que el sistema operativo es de multiprogramación y que como máximo de cada proceso se cargan en memoria 4 páginas ¿Cuántos procesos como mínimo pueden garantizarse que se ejecuten concurrentemente si la memoria principal tuviese su máxima capacidad?

Como el código de marco de página se da con 2 cifras HEX (es decir, 8 bits), el número de marcos de página, con la memoria completa, es 256. También el enunciado del problema indica explícitamente que el número de marcos es 256. Entonces, si como máximo un proceso ocupa 4 marcos de página, el número mínimo de procesos en ejecución concurrente será:

$$N = \frac{256}{4} = \mathbf{64}$$