2017 1 75573 EC-Ibe 2001 18 1 E (Solución)

Estructura de computadors (Universitat Oberta de Catalunya)

grapa

Examen 2017/18-1

Asignatura	Código	Fecha	Hora inicio
Estructura de computadores	75.573	20/01/2018	15:30

⊂75.573ℜ20ℜ01ℜ18ℜEΞ<∈
75.573 20 01 18 EX

Espacio para la etiqueta identificativa con el código personal del **estudiante**.
Examen

Ficha técnica del examen

- **Comprueba que el código y el nombre de la asignatura corresponden a la asignatura de la cual estás matriculado.**
- **Debes pegar una sola etiqueta de estudiante en el espacio de esta hoja destinado a ello.**
- **No se puede añadir hojas adicionales.**
- **No se puede realizar las pruebas a lápiz o rotulador.**
- **Tiempo total** 2 **horas**
  - **En el caso de que los estudiantes puedan consultar algún material durante el examen, ¿cuál o cuáles pueden consultar?:** No se puede utilizar calculadora ni material auxiliar
  - **Valor de cada pregunta:** Pregunta 1 (20%); Pregunta 2 (35%); Pregunta 3 (35%); Pregunta 4 (10%)
- **En el caso de que haya preguntas tipo test: ¿descuentan las respuestas erróneas?** NO **¿Cuánto?**
- **Indicaciones específicas para la realización de este examen**

Enunciados

No se puede utilizar calculadora. Hay que saber interpretar un valor en binario, decimal o hexadecimal para realizar la operación que se pida. Y el resultado se tiene que expresar en el formato correspondiente.

Examen 2017/18-1

Asignatura	Código	Fecha	Hora inicio
Estructura de computadores	75.573	20/01/2018	15:30

Valoración de las preguntas del examen

Pregunta 1 (20%)

Pregunta sobre la práctica.

Hay que completar las instrucciones marcadas o añadir el código ensamblador que se pide. Los puntos suspensivos indican que hay más código pero no se tiene que completar.

NOTA: Si el código propuesto en cada pregunta no se corresponde con la forma en que vosotros plantearíais la respuesta, podéis rescribir el código o parte del código según vuestro planteamiento.

1.1: 10%
1.2: 10%

Pregunta 2 (35%)
2.1: 10%
2.2: 15%
2.3: 10%

Pregunta 3 (35%)
3.1: 15%
3.1.1: 10%
3.1.2: 5%

3.2: 20%
3.2.1: 10%
3.2.2: 5%
3.2.3: 5%

Pregunta 4 (10%)
4.1: 5%
4.2: 5%

Examen 2017/18-1

Asignatura	Código	Fecha	Hora inicio
Estructura de computadores	75.573	20/01/2018	15:30

Pregunta 1

1.1 Práctica – 1ª Parte

Escribir un fragmento de código ensamblador de la subrutina calcIndexP1, para calcular el índice para acceder a una matriz 4x4 de tipo WORD. (No se tiene que escribir el código de toda la subrutina).

```
;;;;;
; Calcular el valor del índice (indexMat) para acceder, en ensamblador,
; a la matriz (m) y (mRotated) de DimMatrix * Dimatrix posiciones de
; tipo int (4 bytes) a partir de los valores de la fila y columna
; indicados por rl vector (rowcol).
; (rowcol: vector con la fila (rowcol[0]), en ensamblador (DWORD[rowcol+0])
; y la columna (rowcol[1]), en ensamblador (DWORD[rowcol+4]) de
; la posición de la matriz a la que queremos acceder.)
; indexMat = ( ([rowcol+0]*DimMatrix)+([rowcol+4]) )*4
; multiplicamos por 4 porque es una matriz de tipo int (DWORD) 4 bytes.
; m[rowcol[0]][rowcol[1]] en C, es equivalente a DWORD[m+indexMat] en
; ensamblador, si indexMat = ((rowcol[0]*DimMatrix)+(rowcol[1]))*4.
; m[1][2] en C, es DWORD[m+24] en ensamblador.
;
; Esta subrutina no tiene una función en C equivalente.
;
; Variables globales utilizadas:
; rowcol   : Vector con la fila y la columna que queremos acceder de la matriz.
; indexMat : Índice para acceder a la matriz (4x4) de tipo DWORD.
;;;;;
calcIndexP1:
   push rbp
   mov  rbp, rsp

   push rax
   push rbx
   push rdx

   mov rax, 0
   mov rbx, 0
   mov rdx, 0

   indexMat = ((rowcol[0]*DimMatrix)+(rowcol[1]))*4
   mov eax, DWORD[rowcol+0]
   mov ebx, DimMatrix
   mul ebx      ;multiplicamos por DimMatrix (EDX:EAX = EAX * DimMatrix)
   add eax, DWORD[rowcol+4]
   shl eax, 2  ;eax =(rowcol[0]*DimMatrix)+(rowcol[1])*4
   mov DWORD[indexMat], eax



calcIndexP1_End:
   pop rdx
   pop rbx
   pop rax

   mov rsp, rbp
   pop rbp
   ret
```

Examen 2017/18-1

Asignatura	Código	Fecha	Hora inicio
Estructura de computadores	75.573	20/01/2018	15:30

1.2 Práctica – 2ª Parte

Completar el código de la subrutina chackEndP2, para verificar si se ha llegado a 2048 o si no se pueden hacer movimientos. (Sólo completar los espacios marcados, no se pueden añadir o modificar otras instrucciones).

```
;;;;;
; Verificar si se ha llegado a 2048 o si no se puede hacer algún movimiento.
; Si hay el número 2048 en la matriz (m), cambiar el estado a 3
; (status=3) para indicar que se ha ganado (WIN!).
; Si no hemos ganado, mirar si se puede hacer algun movimento,
; Si no se puede hacer ningún movimiento cambiar el estado a 4
; (status=4) para indicar que se ha perdido (GAME OVER!!!).
; Recorrer la matriz (m) por filas de derecha a izquierda y de a bajo a
; arriba contando las casillas vacías y mirando si hay el número 2048.
; Si hay el número 2048 poner el estado en 3 (status=3) y acabar.
; Si no hay el número 2048 y no hay casillas vacías, mirar si se puede
; hacer alguna pareja en horizontal o en vertical. Par hacerlo
; hay que copiar la matriz (m) sobre la matriz (mAux) llamando a la
; subrutina (copyStatusP2), hacer parejas en la matriz (mAux) para mirar
; si se pueden hacer parejas en horizontal llamando a la subrutina
; (addPairsLP2) y guardar los puntos obtenidos, rotar la matriz (mAux)
; llamando a la subrutina (rotateMatrixRP2) y copiar la matriz rotada
; (mRotated) en la matriz (mAux) llamando a la subrutina (copyMatrixP2),
; volver a hacer parejas en la matriz (mAux) para mirar si se pueden
; hacer parejas en vertical llamando a la subrutina (addPairsLP2)
; y acumular los puntos obtenidos con los puntos obtenidos antes.
; Si los puntos acumulados son cero, quiere decir que no se pueden hacer
; parejas y se tiene que poner el estado del juego en 4 (status=4).
; Para acceder a matrices desde ensamblador hay que calcular el índice
; a partir de la fila y la columna llamando a la subrutina calcIndexP2.
; No se puede modificar ni la matriz (m), ni la matriz (mUndo).
;
; Variables globales utilizadas:
; m       : Dirección de la matriz que queremos verificar.
; mAux    : Dirección de la matriz donde copiaremos los datos para hacer verificaciones.
; mRotated:  Dirección de la matriz donde se hace la rotación.
; Parámetros de entrada:
; rdi(edi) :  Estado del juego.
; Parámetros de salida :
; rax(eax) :  Estado del juego.
;;;;;
checkEndP2:   push rbp
   mov  rbp, rsp
   ...
   mov edx, _edi_          ;guardamos el estado del juego en edx
   mov ebx, 0              ;zeros=0;
   mov ecx, 0              ;pairs=0;

   mov r8d, DimMatrix      ;i = r8d
   checkEndP2_Rows:
      dec r8d
      mov r9d , DimMatrix  ;j = r9d
      checkEndP2_Cols:
         dec r9d
         mov edi, r8d
         mov esi, r9d
         call _calcIndexP2_
         cmp DWORD[m+eax], 0    ;m[i][j] == 0
         jne checkEndP2_Win
            inc ebx             ;zeros++;
         checkEndP2_Win:
         cmp _DWORD_[_m+eax_], 2048 ;m[i][j] == 2048
          jne checkEndP2_Next
```

Examen 2017/18-1

Asignatura	Código	Fecha	Hora inicio
Estructura de computadores	75.573	20/01/2018	15:30

```
            mov _edx_, 3    ;status = 3
            jmp checkEndP2_End

      checkEndP2_Next:
      cmp r9d, 0                    ;j>0
      jg   checkEndP2_Cols

   cmp r8d, 0                       ;i>0
   jg   checkEndP2_Rows

   cmp ebx, 0                       ;zeros == 0
   jne checkEndP2_End
      ;cmp  edx, 3
      ;je  checkEndP2_End
      mov  edi, mAux
      mov  esi, m
      call copyMatrixP2             ;copyMatrixP2_C(mAux,m);
      call addPairsLP2              ;addPairsLP2_C(mAux);
      mov  ecx, eax                 ;pairs = addPairsLP2_C(mAux)
      call rotateMatrixRP2          ;rotateMatrixRP2_C(mAux)
      mov  edi, mAux
      mov  esi, mRotated
      call copyMatrixP2             ;copyMatrixP2_C(mAux,mRotated);
      call addPairsLP2              ;addPairsLP2_C(mAux)
      add  ecx, eax                 ;pairs = pairs + addPairsLP2_C(mAux)
      cmp  ecx, 0                   ;pairs==0
      _jne_ checkEndP2_End
         mov edx, 4                 ;state = 4

   checkEndP2_End:
   mov _eax_, edx                   ;return status;
   ...
   mov rsp, rbp
   pop rbp
   ret
```

Examen 2017/18-1

Asignatura	Código	Fecha	Hora inicio
Estructura de computadores	75.573	20/01/2018	15:30

Pregunta 2

2.1

El estado inicial del computador CISCA justo antes de empezar la ejecución de cada fragmento de código (en cada apartado) es el siguiente:

R1 = 00000010h R5 = 00000028h R10 = 00000050h	M(00000078h) = 810077E0h M(00000800h) = 00000810h	Z = 0, C = 0, S = 0, V = 0

Completad el estado del computador después de ejecutar cada código (indicad los valores de los registros en hexadecimal).
Suponed que la dirección simbólica A vale 800h.

a)

```
MOV R2, 0000FFFFh
SAL R2, R1
NOT R2
```

Solució:

R2= 0000FFFFh
R2= FFFF0000h
R2= 0000FFFFh

C=0, V=0, S=1, Z=0

b)

```
ADD R5, R10
MOV R5, [R5]
AND [V], R5
```

Solució:

R5= 00000078h
R5= 810077E0h
[A]= 00000000h

C=0, V=0, S=0, Z=1

Examen 2017/18-1

Asignatura	**Código**	**Fecha**	**Hora inicio**
Estructura de computadores	75.573	20/01/2018	15:30

2.2

Suponemos que tenemos el vector V de 100 elementos de 32 bits. Dado el siguiente código en alto nivel:

```
X2= 0;
while (X1<=V[X2] && X2<100)
{
   X2++;
   X1 = X1 * 2;
}
```

Completad la traducción del programa en ensamblador CISCA para que ejecute el algoritmo de alto nivel mostrado. (Hemos dejado 6 espacios para llenar)

```
            MOV   R1, [X1]
            MOV   R2, 0
WHILE:      CMP   R2, 400
            JE    ENDWHILE
            CMP   R1, [V+R2]
            JG    ENDWHILE
            MUL   R1, 2
            ADD   R2, 4
            JMP   WHILE
ENDWHILE:   MOV   [X1], R1
            MOV   [X2], R2
```

Examen 2017/18-1

Asignatura	**Código**	**Fecha**	**Hora inicio**
Estructura de computadores	75.573	20/01/2018	15:30

2.3

Dado el siguiente fragmento de código de un programa en lenguaje ensamblador de CISCA:

```
Loop:  ADD R0, 4
       CMP R0, 400h
       JL Loop
```

Traducidlo a lenguaje máquina y expresadlo en la siguiente tabla. Suponed que la primera instrucción del código se ensambla a partir de la dirección 00023C00h (que es el valor del PC antes de empezar la ejecución del fragmento de código). En la siguiente tabla usad una fila para codificar cada instrucción. Si suponemos que la instrucción empieza en la dirección @, el valor Bk de cada uno de los bytes de la instrucción con direcciones @+k para k=0, 1,… se tiene que indicar en la tabla en hexadecimal en la columna correspondiente (recordad que los campos que codifican un desplazamiento en 2 bytes o un inmediato o una dirección en 4 bytes lo hacen en formato little endian, esto hay que tenerlo en cuenta escribiendo los bytes de menor peso, de dirección más pequeña, a la izquierda y los de mayor peso, dirección mayor, a la derecha). Completad también la columna @ que indica para cada fila la dirección de memoria del byte B0 de la instrucción que se codifica en esta fila de la tabla.

A continuación os damos como ayuda las tablas de códigos:

tabla de códigos de instrucción

B0	Instrucción
43h	JL
26h	CMP
20h	ADD

tabla de modos de direccionamiento (Bk<7..4>)

Camp modo Bk<7..4>	Modo
0h	Inmediato
1h	Registro
2h	Memoria
3h	Indirecto
4h	Relativo
5h	Indexado
6h	Relativo a PC

tabla de modos de direccionamiento (Bk<3..0>)

Camp modo Bk<3..0>	Significado
Num. registro	Si el modo tiene que especificar un registro
0	No se especifica registro.

		Bk para k=0..10										
@	**Ensamblador**	0	1	2	3	4	5	6	7	8	9	10
00023C00h	ADD R0, 4	20	10	00	04	00	00	00				
00023C07h	CMP R0, 400h	26	10	00	00	04	00	00				
00023C0Eh	JL Loop	43	60	EE	FF							
0003FC12h												

Examen 2017/18-1

Asignatura	Código	Fecha	Hora inicio
Estructura de computadores	75.573	20/01/2018	15:30

Pregunta 3

3.1. Memoria cache

Memoria cache completamente asociativa (LRU)

Tenemos un sistema de memoria en el que todos los accesos se realizan a palabra (no nos importa cuál es el tamaño de la palabra). Supondremos que el espacio de direcciones de memoria se descompone en bloques de 8 palabras. Cada bloque empieza en una dirección múltiplo de 8. Así, el bloque 0 contiene las direcciones 0, 1, 2, 3, 4, 5, 6, 7, el bloque 1, las direcciones 8, 9, 10, 11, 12, 13, 14, 15, y el bolque N las direcciones 8*N, 8*N+1, 8*N+2, 8*N+3, 8*N+4, 8*N+5, 8*N+6, 8*N+7.

Suponemos que el sistema también dispone de una memoria cache de 4 líneas (donde cada línea tiene el tamaño de un bloque, es decir, 8 palabras). Estas líneas se identifican como líneas 0, 1, 2 y 3. Cuando se hace una referencia a una dirección de memoria principal, si esta dirección no se encuentra en la memoria cache, se trae todo el bloque correspondiente desde la memoria principal a una línea de la memoria cache (así si hacemos referencia a la dirección 2 de memoria principal traeremos el bloque formado por las palabras 0, 1, 2, 3, 4, 5, 6, 7).

Suponemos que el sistema utiliza una política de emplazamiento completamente asociativa, de manera que cualquier bloque de la memoria principal se puede llevar a cualquier bloque de la memòria cache. Si encontramos que la memoria cache ya está llena, se utiliza un **algoritmo de reemplazamiento LRU.**

La ejecución de un programa genera la siguiente lista de lecturas a memoria:

0, 1, 2, 12, 62, 63, 25, 64, 17, 18, 19, 2, 4, 6, 65, 66, 20, 56, 42, 50

3.1.1. La siguiente tabla muestra el estado inicial de la cache, que contiene las primeras 32 palabras de la memoria (organizadas en 4 bloques).

Completar la tabla para mostrar la evolución de la cache durante la ejecución del programa. Para cada acceso se debe rellenar una columna indicando si se trata de un acierto o un fallo.

Si es un acierto escribiremos E en la línea correspondiente delante de las direcciones del bloque, si es un fallo escribiremos F i se indicará el nuevo bloque que es trae a la memoria cache en la línea que le corresponda, expresando de la forma b (a_0 - a_7) donde b:número de bloque, y (a_0 - a_7) son las direcciones del bloque, donde a_0 es la primera dirección del bloque y a_7 es la octava (última) dirección del bloc.

Línea	Estado Inicial		0		1		2		12		62
0	0 (0 - 7)	E	0 (0 - 7)	E	0 (0 - 7)	E	0 (0 - 7)		0 (0 - 7)		0 (0 - 7)
1	1 (8 - 15)		1 (8 - 15)		1 (8 - 15)		1 (8 - 15)	E	1 (8 - 15)		1 (8 - 15)
2	2 (16 - 23)		2 (16 - 23)		2 (16 - 23)		2 (16 - 23)		2 (16 - 23)		2 (16 - 23)
3	7 (56 - 63)		7 (56 - 63)		7 (56 - 63)		7 (56 - 63)		7 (56 - 63)	E	7 (56 - 63)

Línea	63		25		64		17		18		19
0		0 (0 - 7)		0 (0 - 7)	F	8 (64 – 71)		8 (64 – 71)		8 (64 – 71)		8 (64 – 71)
1		1 (8 - 15)		1 (8 - 15)		1 (8 - 15)	F	2 (16 - 23)	E	2 (16 - 23)	E	2 (16 - 23)
2		2 (16 - 23)	F	3 (24 - 31)		3 (24 - 31)		3 (24 - 31)		3 (24 - 31)		3 (24 - 31)

Examen 2017/18-1

Asignatura	**Código**	**Fecha**	**Hora inicio**
Estructura de computadores	75.573	20/01/2018	15:30

3	E	7 (56 - 63)		7 (56 - 63)		7 (56 - 63)		7 (56 - 63)		7 (56 - 63)		7 (56 - 63)

Línea	2		4		6		65		66		20	
0		8 (64 – 71)		8 (64 – 71)		8 (64 – 71)	E	8 (64 – 71)	E	8 (64 – 71)		8 (64 – 71)
1		2 (16 - 23)		2 (16 - 23)		2 (16 - 23)		2 (16 - 23)		2 (16 - 23)	E	2 (16 - 23)
2		3 (24 - 31)		3 (24 - 31)		3 (24 - 31)		3 (24 - 31)		3 (24 - 31)		3 (24 - 31)
3	F	0 (0 - 7)	E	0 (0 - 7)	E	0 (0 - 7)		0 (0 - 7)		0 (0 - 7)		0 (0 - 7)

Línea	56		42		50							
0		8 (64 – 71)		8 (64 – 71)	F	6 (48 – 55)						
1		2 (16 - 23)		2 (16 - 23)		2 (16 - 23)						
2	F	7 (56 - 63)		7 (56 - 63)		7 (56 - 63)						
3		0 (0 - 7)	F	5 (40 - 47)		5 (40 - 47)						

3.1.2 a) ¿Cuál es la tasa de aciertos (T_e) ?

T_e = 13 aciertos / 20 accesos = 0,65

3.1.2 b) Suponemos que el tiempo de acceso a la memoria cache, o tiempo de acceso en caso de acierto (t_e), es de 5 ns y el tiempo total de acceso en caso de fallo (t_f) es de 40 ns. Considerando la tasa de aciertos obtenida en la pregunta anterior, cuál es el tiempo medio de acceso a memoria (t_m) ?

$t_m = T_e \times t_e + (1-T_e) \times t_f$ = 0,65 * 5 ns + 0,35 * 40 ns = 3,25 ns + 14 ns = 17,25 ns

Examen 2017/18-1

Asignatura	Código	Fecha	Hora inicio
Estructura de computadores	75.573	20/01/2018	15:30

3.2 Sistema de E/S

3.2.1 E/S por interrupcions

Se quiere analizar el rendimiento de la comunicación de datos entre una memoria de un procesador y un puerto USB, que tienen las siguientes características, utilizando E/S por interrupciones:

- Velocidad de transferencia del dispositivo de E/S (**v** $_{transf}$)= 1 MBytes/s = 1000 Kbytes/s
- Tiempo de latencia media del dispositivo ($t_{latencia}$) = 0
- Direcciones de los **registros de datos** y **de estado** del controlador de E/S: 0B00h i 0B04h
- El bit del **registro de estado** que indica que el controlador del puerto de E/S está disponible es el bit 2, o el tercer bit menos significativo (cuando vale 1 indica que está disponible)
- Procesador con una frecuencia de reloj de 1 GHz, el procesador puede ejecutar 2 instrucciones por ciclo de reloj.
- Transferencia de **lectura** desde puerto de E/S hacia la memoria
- Transferencia de N_{datos}=100.000 datos
- El tiempo para atender la interrupción (t_{rec_int}) es de 4 ciclos de reloj

a) Completar el siguiente código CISCA que es una rutina de servicio a las interrupciones (RSI) para transferir a través del dispositivo de E/S anterior, mediante la técnica de E/S por interrupciones.

Se utiliza una variable global que se representa con la etiqueta **Addr,** y que al principio del programa contiene la dirección inicial de memoria donde almacenar los datos recibidos.

```
1.   CLI
2.   PUSH R0
3.   PUSH R1
4.   IN   R0, [0B04h]
5.   MOV  R1, [Addr]
6.   MOV  [R1],R0
7.   ADD  R1,4
8.   MOV  [Addr],R1
9.   POP  R1
10.  POP  R0
11.  STI
12.  IRET
```

b) ¿Cuánto tiempo dura la transferencia del bloque de datos t_{transf_bloc}?

El tiempo de un ciclo, t_{ciclo} = 1 ns (nanosegundos)

Tiempo para atender la interrupción, t_{rec_int}: 2 cicles * 1 ns = 2 ns

Tiempo de ejecución de una instrucción, t_{instr}: t_{ciclo} / 4 = 0,250 ns

Tiempo de ejecución RSI, t_{rsi}: N_{rsi} * t_{instr} = 12 instr. * 0,250 ns = 3 ns

Tiempo consumido por la CPU en cada interrupción, t_{transf_dada}:

t_{transf_dada} = t_{rec_int} + t_{rsi} = 2 + 3 = 5 ns

Número de interrupciones producidas (número total de datos, N_{datos}): 100000 interrupciones

Examen 2017/18-1

Asignatura	Código	Fecha	Hora inicio
Estructura de computadores	75.573	20/01/2018	15:30

Tiempo consumido en total por TODAS las interrupciones:

$t_{transf\_bloc} = t_{transf\_dada} * N_{dades}$ = 10 ns * 100000 interrupciones = 1000000 ns = 1 ms (milisegundos)

c) ¿Cuál es el porcentaje de ocupación del procesador? Porcentaje que representa el tiempo de transferencia del bloc t_{transf_bloc} respecto al tiempo de transferencia del bloque por parte del periférico t_{bloc}

$t_{bloc} = t_{latencia} + (N_{datos} * t_{dato})$
$t_{latencia} = 0$
$N_{datos} = 100000$
$t_{dato} = m_{dato} / v_{transf}$ = 4 / 1000 Kbytes/s = 0,004 ms
t_{bloc} = 0 + (100000 * 0,004) ms = 400 ms
% ocupación = (t_{transf_bloc} * 100 / t_{bloc}) = (1 * 100) / 400 = 0,25%

Examen 2017/18-1

Asignatura	Código	Fecha	Hora inicio
Estructura de computadores	75.573	20/01/2018	15:30

Pregunta 4

4.1

¿Qué entendemos por «segmentación de las instrucciones»?

La segmentación de las instrucciones (pipeline) consiste al dividir el ciclo de ejecución de las instrucciones en un conjunto de etapas. Estas etapas pueden coindicir, o no, con las fases del ciclo de ejecución de las instrucciones.

4.2

a) ¿Cuáles son las tres políticas de asignación para almacenar datos dentro de una memoria cache? ¿En qué consisten?

1) Política de asignación directa: un bloque de la memoria principal sólo puede estar en una única línea de la memoria cache. La memoria cache de asignación directa es la que tiene la tasa de fallos más alta, pero se utiliza mucho porqué es la más barata y fácil de gestionar

2) Política de asignación completamente asociativa: un bloque de la memoria principal puede almacenarse en cualquier línea de la memoria cache. La memoria cache completamente asociativa es la que tiene la tasa de fallos más baja. A pesar de eso, no se suele utilizar porqué es la más cara y compleja de gestionar.

3) Política de asignación asociativa por conjuntos: un bloque de la memoria principal puede almacenarse en un subconjunto de las líneas de la memoria cache, pero dentro del subconjunto puede encontrarse ne cualquier posición. La memoria cache asociativa por conjuntos es una combinación.

b) En un sistema de E/S gestionado por DMA. Explica, cuándo i por qué se produce una interrupción. ¿Sirve para indicar el inicio o el final de una transferencia? ¿Quién la genera?

Finalización de la operación de E/S: Cuando ha finalizado la transferencia del bloque el controlador de DMA envía una petición de interrupción al procesador para informar que ha acabado la transferencia de datos.