Examen 9 Junio 2018, preguntas y respuestas

Estructura de computadors (Universitat Oberta de Catalunya)

Espai grapa

Examen 2017/18-2

Asignatura	Código	Fecha	Hora inicio
Estructura de computadores	75.573	09/06/2018	18:30

⊂75.573ℜ09ℜ06ℜ18ℜEΞP∈
75.573 09 06 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 matriculada.
- Debes pegar una sola etiqueta de estudiante en el espacio correspondiente de esta hoja.
- No se puede añadir hojas adicionales, ni realizar el examen en lápiz o rotulador grueso.
- Tiempo total: **2 horas** Valor de cada pregunta: **Se indica en el enunciado**
- En el caso de que los estudiantes puedan consultar algún material durante el examen, ¿cuáles son?: **NINGUNO**
- En el caso de poder usar calculadora, de que tipo? NINGUNA
- En el caso de que haya preguntas tipo test: ¿descuentan las respuestas erróneas? NO ¿Cuánto?
- Indicaciones específicas para la realización de este examen

Examen 2017/18-2

Asignatura	Código	Fecha	Hora inicio
Estructura de computadores	75.573	09/06/2018	18:30

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.

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 2017/18-2

Asignatura	Código	Fecha	Hora inicio
Estructura de computadores	75.573	09/06/2018	18:30

Pregunta 1

1.1 Práctica – 1a Parte

Escribir el fragmento de código en ensamblador de la subrutina calcIndexP1, para calcular el índice para acceder a la matriz. (No se tiene que escribir el código de toda la subrutina).

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; Calcular el índice (indexMat) para acceder a la matriz
; (tiles) de DimMatrix * Dimatrix posiciones de tipo char (1 byte)
; a partir de los valores de la fila y la columna en el tablero (pantalla)
; indicados por las variables (rowScreen) fila y (colScreen) columna.
; indexMat = ((rowScreen-11)/2)*DimMatrix+((colScreen-11)/4)
; Restamos 11 porque es la posición de la casilla [0][0].
; Dividimos por 2 las filas porque las filas están separadas por una línea.
; Dividimos por 4 las columnas porque las columnas están separadas por 4 espacios.
; Multiplicamos por DimMatrix, porque cada fila de la matriz tiene DimMatrix elementos.
; La posición en el tablero (fila:13,columna:19) seria [tiles+6].
;
; Hay una función en C equivalente 'calcIndexMatP1_C', pero ATENCIÓN!!!
; tiene una funcionalidad un poco diferente porque el acceso
; a matrices en ensamblador se hace con un único índice.
;
; Variables globales utilizadas:
; rowScreen: Fila donde queremos posicionar el cursor en pantalla.
; colScreen: Columna donde queremos posicionar el cursor en pantalla.
; indexRow : Fila para acceder a la matriz tiles [0..(dimMatrix-1)]
; indexCol : Columna para acceder a la matriz tiles [0..(dimMatrix-1)]
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;calcIndexMatP1:
   push rbp
   mov  rbp, rsp
   ...
   mov  rax, 0
   mov  rbx, 0
   mov  rdx, 0



   mov  ebx, DWORD[rowScreen] ;ebx=((rowScreen-11)/2)
   sub  ebx, 11
   shr  ebx, 1
   mov  eax, DimMatrix
   mul  ebx                   ;eax=((rowScreen-11)/2)*DimMatrix
   mov  ebx, DWORD[colScreen] ;ebx=((colScreen-11)/4)
   sub  ebx, 11
   shr  ebx, 2
   add  eax, ebx ;eax=((rowScreen-11)/2)*DimMatrix+((colScreen-11)/4)



   mov DWORD[indexMat], eax
   calcIndexP1_End:
   ...
   mov rsp, rbp
   pop rbp
   ret
```

Asignatura	Código	Fecha	Hora inicio
Estructura de computadores	75.573	09/06/2018	18:30

1.2 Práctica – 2a parte

Completar el código de la subrutina checkEndP2. (Sólo completar los espacios marcados, no se puede añadir código, ni modificar otras instrucciones).

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; Comprobar si el espacio en blanco está en la última posición del
; tablero (inferior-derecha) y si todas las pieza de la matriz (t),
; recibida como parámetro, están ordenadas (Se tiene que recorrer toda
; la matriz (t), de tipo char (1 byte cada posición) sin considerar la
; última posición de la matriz que es donde debería estar el espacio).
; Si las piezas no están ordenadas retornamos el estado recibido como parámetro.
; Si las piezas si están ordenadas retornamos el estado "2" para salir.
;
; Variables globales utilizadas: Ninguna
; Parámetros de entrada :
; rdi      : Matriz donde guardamos los números del juego.
; rsi(sil) : Estado del juego.
;            0: Salir, hemos pulsado la tecla 'ESC' para salir.
;            1: Continuemos jugando.
;            2: Gana, todas las fichas están ordenadas
; Parámetros de salida:
; rax(al) : Estado del juego.
;           0: Salir, hemos pulsado la tecla 'ESC' para salir.
;           1: Continuemos jugando.
;           2: Gana, todas las fichas están ordenadas
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
checkEndP2:
   push rbp
   mov  rbp, rsp
   ...
   mov  rbx, __rdi__          ;dirección matriz.
   mov  r8b, __sil__          ;estado del juego

   ;cl=t[DimMatrix-1][DimMatrix-1]
   mov  cl, __BYTE[rbx+SizeMatrix-1]__
   mov  ch, cl                  ;aux = t[DimMatrix-1][DimMatrix-1]
   cmp  ch, ' '                 ;(aux==' ')
   jne  checkEndP2_end
   mov rax, 0
   mov dl , 1                   ;sorted=1. Matriz ordenada
   mov edi, 0                   ;i=0
   checkEndP2_while_i:
      cmp  edi, DimMatrix       ;i<DimMatrix
      jge  checkEndP2_endWhile_i
      cmp  dl, 1                ;sorted==1
      jne  checkEndP2_endWhile_i
      mov  esi, 0               ;j=0
      checkEndP2_while_j:
         cmp esi, DimMatrix     ;j<DimMatrix)
         jge checkEndP2_endWhile_j
         cmp  dl, 1             ;sorted==1
         jne  checkEndP2_endWhile_j
            mov  cl, __BYTE[rbx+rax]__ ;cl=t[i][j]
            cmp  ch, cl         ;aux>t[i][j]
            jle  checkEndP2_Sorted
            cmp  rax, (SizeMatrix-1);(i*DimMatrix+j)<(SizeMatrix-1))
            jge  checkEndP2_Sorted
               mov  dl, 0       ;sorted=0; (unsorted)
         checkEndP2_Sorted:
```

Examen 2017/18-2

Asignatura	Código	Fecha	Hora inicio
Estructura de computadores	75.573	09/06/2018	18:30

```
        mov ch, cl                ;aux=t[i][j];
        inc rax
        inc esi                   ;j++
        jmp checkEndP2_while_j
    checkEndP2_endWhile_j:
    inc edi                       ;i++
    jmp checkEndP2_while_i
checkEndP2_endWhile_i:
cmp dl, 1                         ;(sorted==1)
jne checkEndP2_end
    mov __r8b__, 2                ;status = 2
checkEndP2_end:
mov rax, 0
mov __al__, r8b                   ;return status;
...
mov rsp, rbp
pop rbp
ret
```

Examen 2017/18-2

Asignatura	Código	Fecha	Hora inicio
Estructura de computadores	75.573	09/06/2018	18:30

Pregunta 2

2.1

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

R0 = 00000A10h R1 = 00000B20h R2 = 00000C30h	M(00000A10h) = 0000F00Fh M(00000B20h) = 0000F000h M(00000C30h) = 00000FF0h M(000000F0h) = 00000001h M(000FF0A0h) = 0000000Ah	Z = 0, C = 1, S = 1, V = 0

La dirección simbólica W vale 000FF0A0h. ¿Cuál será el estado del computador después de ejecutar cada fragmento de código? (sólo modificaciones, excluyendo el PC).

a)

```
SUB    R0, R1
ADD    [W], R2
```

R0= A10h – B20h = FFFFFEF0h
M(000FF0A0h)= 0000000Ah + 00000C30h = 00000C3Ah

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

b)

```
MOV    R2, [W]
MOV    [R1],R0
```

R2= 0000000Ah
M(00000B20)= 00000A10h

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

Examen 2017/18-2

Asignatura	Código	Fecha	Hora inicio
Estructura de computadores	75.573	09/06/2018	18:30

2.2

En la memoria de un computador CISCA tenemos almacenados dos vectores A y B de 10 y 20 elementos respectivamente. Cada elemento es un número entero codificado en complemento a 2 con 32 bits.

Completad los huecos del fragmento de código CISCA para que realice el equivalente a la siguiente sentencia en C:

A[i]= A[i] + B[j]

Los vectores están almacenados en posiciones consecutivas de memoria, como es habitual cuando se traduce código en C. Por ejemplo, los elementos A[0], A[1], A[2] y A[7] se encuentran almacenados en las direcciones de memoria A, A+4, A+8 y A+28 respectivamente. El mismo para el vector B.

Se sabe que en R1 se encuentra almacenado el valor de la variable "i", y en R2 el de la "j" y que después de ejecutarse el fragmento de código todos los registros tienen que mantener los valores originales.

```
PUSH  R2
PUSH  R1
MUL   R1, 4
MUL   R2, 4
MOV   R3, [B+R2]
ADD   [A+R1], R3
POP   R1
POP   R2
```

Examen 2017/18-2

Asignatura	Código	Fecha	Hora inicio
Estructura de computadores	75.573	09/06/2018	18:30

2.3

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

```
MUL R10, [Q]
ADD R2, 2
SAL [Q], 10h
```

Traducidlo a lenguaje máquina y expresadlo en la siguiente tabla. Suponed que la primera instrucción del código se ensambla a partir de la dirección **003FC000h** (que es el valor del PC antes de empezar la ejecución del fragmento de código). Suponed que la dirección simbólica Q vale **00003A00h**. En la siguiente tabla usad una fila para codificar cada instrucción. Si suponemos que la instrucción comienza en la dirección @, el valor Bk de cada uno de los bytes de la instrucción con direcciones @+k para k=0, 1,… se debe 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 'Dirección' que indica para cada fila la dirección de memoria del byte B0 de la instrucción que se codifica en esa fila de la tabla.

A continuación os damos como ayuda las tablas de códigos:

Tabla de códigos de instrucción

B0	Instrucción
22h	MUL
20h	ADD
36h	SAL

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

Campo 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>)

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

		Bk para k=0..10										
Dirección	**Ensamblador**	0	1	2	3	4	5	6	7	8	9	10
003FC000	MUL R10, [Q]	22	1A	20	00	3A	00	00				
003FC007	ADD R2, 2	20	12	00	02	00	00	00				
003FC00E	SAL [Q], 10h	36	20	00	3A	00	00	00	10	00	00	00
003FC019												

Examen 2017/18-2

Asignatura	Código	Fecha	Hora inicio
Estructura de computadores	75.573	09/06/2018	18:30

Pregunta 3

3.1 Memoria cache

Memoria cache de asignación directa
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 utilizar una **política de asignación directa**, de manera que cada bloque de la memoria principal sólo se puede traer a una línea determinada de la memoria cache.

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

0, 1, 2, 12, 61, 62, 63, 64, 17, 18, 19, 32, 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 bloque.

Linea	Estado Inicial	0	1	2	12	61
0	0:0 (0 - 7)	E 0:0 (0 - 7)	E 0:0 (0 - 7)	E 0:0 (0 - 7)	0:0 (0 - 7)	0:0 (0 - 7)
1	1:0 (8 - 15)	1:0 (8 - 15)	1:0 (8 - 15)	1:0 (8 - 15)	E 1:0 (8 - 15)	1:0 (8 - 15)
2	2:0 (16 - 23)	2:0 (16 - 23)	2:0 (16 - 23)	2:0 (16 - 23)	2:0 (16 - 23)	2:0 (16 - 23)
3	3:0 (24 - 31)	3:0 (24 - 31)	3:0 (24 - 31)	3:0 (24 - 31)	3:0 (24 - 31)	F 7:1 (56 - 63)

Linea	62	63	64	17	18	19
0	0:0 (0 - 7)	0:0 (0 - 7)	F 8:2 (64 - 71)	8:2 (64 - 71)	8:2 (64 - 71)	8:2 (64 - 71)
1	1:0 (8 - 15)	1:0 (8 - 15)	1:0 (8 - 15)	1:0 (8 - 15)	1:0 (8 - 15)	1:0 (8 - 15)
2	2:0 (16 - 23)	2:0 (16 - 23)	2:0 (16 - 23)	E 2:0 (16 - 23)	E 2:0 (16 - 23)	E 2:0 (16 - 23)
3	E 7:1 (56 - 63)	E 7:1 (56 - 63)	7:1 (56 - 63)	7:1 (56 - 63)	7:1 (56 - 63)	7:1 (56 - 63)

Examen 2017/18-2

Asignatura	Código	Fecha	Hora inicio
Estructura de computadores	75.573	09/06/2018	18:30

Linea	32		4		6		65		66		20	
0	F	4:1 (32 - 39)	F	0:0 (0 - 7)	E	0:0 (0 - 7)	F	8:2 (64 - 71)	E	8:2 (64 - 71)		8:2 (64 - 71)
1		1:0 (8 - 15)		1:0 (8 - 15)		1:0 (8 - 15)		1:0 (8 - 15)		1:0 (8 - 15)		1:0 (8 - 15)
2		2:0 (16 - 23)		2:0 (16 - 23)		2:0 (16 - 23)		2:0 (16 - 23)		2:0 (16 - 23)	E	2:0 (16 - 23)
3		7:1 (56 - 63)		7:1 (56 - 63)		7:1 (56 - 63)		7:1 (56 - 63)		7:1 (56 - 63)		7:1 (56 - 63)

Linea	56		42		50							
0		8:2 (64 - 71)		8:2 (64 - 71)		8:2 (64 - 71)						
1		1:0 (8 - 15)	F	5:1 (40 - 47)		5:1 (40 - 47)						
2		2:0 (16 - 23)		2:0 (16 - 23)	F	6:1 (48 - 55)						
3	E	7:1 (56 - 63)		7:1 (56 - 63)		7:1 (56 - 63)						

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

T_a = 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_a), es de 4 ns y el tiempo total de acceso en caso de fallo (t_f) es de 20 ns. Considerando la tasa de aciertos obtenida en la pregunta anterior, ¿cuál es el tiempo medio de acceso a memoria (t_m)?

$t_m = T_a \times t_a + (1-T_a) \times t_f = 0,65 * 4 \text{ ns} + 0,35 * 20 \text{ ns} = 2,6 \text{ ns} + 7 \text{ ns} = 9,6 \text{ ns}$

Examen 2017/18-2

Asignatura	Código	Fecha	Hora inicio
Estructura de computadores	75.573	09/06/2018	18:30

3.2 Sistema d'E/S

Se quiere analizar el rendimiento de la comunicación de datos entre la mamoria de un procesador y un puerto USB, utilizando E/S programada con las siguientes características:

- Velocidad de transferencia del dispositivo d'E/S v_{transf} = 4 MBytes/s = 4000 Kbytes/s
- Tiempo de latencia medio del dispositivo $t_{latencia}$ = 0
- Direcciones de los **registros de estado** y **datos** del controlador de E/S: 0F00h y 0F04h
- El bit del **registro de estado** que indica que el controlador del puerto de E/S está disponible es el bit 4, o el quinto bit menos significativo (cuando vale 1 indica que está disponible)
- Procesador con una frecuencia de reloj de 2 GHz, el tiempo de ciclo t_{ciclo} = 0,5 ns. El procesador puede ejecutar 1 instrucción por ciclo de reloj.
- Transferencia de **escritura** desde memoria al puerto de E/S
- Transferencia de N_{datos}= 200000 datos
- El tamaño de una dato es m_{dato} = 4 bytes
- Dirección inicial de memoria donde residen los datos: A0000000h

3.2.1 El siguiente código realizado con el repertorio de instrucciones CISCA realiza la transferencia descrita antes mediante la técnica de E/S programada. Completar el código.

```
1.        MOV R3, _200000_
2.        MOV R2, A0000000h
3.Bucle:  _IN_     R0, [0F00h]  ; leer 4 bytes
4.        AND R0, _00010000b_
5.        _JE_Bucle
6.        MOV R0,_[R2]_         ; leer 4 bytes
7.        ADD R2, 4
8.        OUT _[0F04h]_, R0     ; escribir 4 bytes
9.        SUB R3, 1
10.       JNE Bucle
```

3.2.2 Cuánto tiempo dura la transferencia del bloque de datos t_{transf_bloque}?

t_{transf_bloque} = $t_{latència}$ + (N_{datos} * t_{transf_dada})
$t_{latencia}$ = 0
N_{datos} = 160000
t_{transf_dada} = m_{dato} / v_{transf} = 4 Bytes / 4000 Kbytes/s = 0,001 ms
t_{transf_bloque} = 0 + (200000 * 0,001 ms) = 200 ms = 0,2 s

3.2.3 Si quisiéramos utilizar el mismo procesaodr y el mismo programa, pero con un dispositivo más rápido de E/S, ¿cuál es la tasa o velocidad máxima de transferencia del nuevo dispositivo que se podría soportar sin que el dispositivo tuviera que esperar?

t_{ciclo} = 0,5 ns (nanosegundos)

t_{instr} = 0,5 ns / 1 = 0,50 ns

Examen 2017/18-2

Asignatura	Código	Fecha	Hora inicio
Estructura de computadores	75.573	09/06/2018	18:30

El mínimo número de instrucciones que ha de ejecutar el programa para cada dato transferido son las 8 instrucciones: 3, 4, 5, 6, 7, 8, 9 i 10. Ejecutar las 8 instrucciones requiere 8 * t_{instr} = 8 * 0,50 ns = 4 ns

Por tanto, el tiempo mínimo para transferir un dato es: 4 ns

Se pueden tranferir 4 bytes cada 4 ns, es decir: 4 / 4 * 10^{-9} = 1000 Mbyte/s = 1 Gbytes/s

Pregunta 4

4.1

¿Qué entendemos por arquitectura de un computador y qué elementos se asocian a una arquitectura?

La arquitectura del computador hace referencia al conjunto de elementos del computador que son visibles desde el punto de vista del

programador de ensamblador. Los elementos habituales asociados a la arquitectura del computador son los siguientes:

• Juego de instrucciones y modos de direccionamiento del computador.

• Tipos y formatos de los operandos.

• Mapa de memoria y de E/S.

• Modelos de ejecución.

4.2

4.2.1 Uno de los factores básicos que hacen que el esquema de jerarquía de memorias funcione satisfactoriamente es la proximidad referencial. ¿Qué tipos de proximidad referencial podemos distinguir? Explicar brevemente en que consiste cada una de ellas.

Distinguimos dos tipos de proximidad referencial:

1) **Proximidad temporal**. Es cuando, en un intervalo de tiempo determinado, la probabilidad que un programa acceda de manera repetida a les mismas posiciones de memoria es muy grande.
La proximidad temporal es debida principalmente a las estructuras iterativas; un bucle ejecuta las mismas instrucciones repetidamente, de la misma manera que las llamadas repetitivas a subrutinas.

2) **Proximidad espacial**. Es cuando, en un intervalo de tiempo determinado, la probabilidad que un programa acceda a posiciones de memoria próximas es muy grande.
La proximidad espacial es debida principalmente al hecho que la ejecución de los programas es secuencial –se ejecuta una instrucción detrás de otra a excepción de las bifurcaciones–, y también a la utilización de estructuras de datos que están almacenadas en posiciones de memoria contiguas.

Examen 2017/18-2

Asignatura	Código	Fecha	Hora inicio
Estructura de computadores	75.573	09/06/2018	18:30

4.2.2 Una manera de optimizar las operaciones de E/S por DMA consiste en reducir el número de cesiones y recuperaciones del bus, mediante una modalidad de transferencia denominada modo ráfaga. ¿Cuál es el funcionamiento del DMA en este modo en el caso de una transferencia del dispositivo a la memoria?

Cada vez que el módulo de E/S tiene un dato disponible el controlador de DMA lo almacena en la memoria intermedia y decrementa el registro contador. Cuando la memoria intermedia está llena o el contador ha llegado a cero, solicita el bus. Una vez el procesador le cede el bus, escribe en memoria todo el conjunto de datos almacenados en la memoria intermedia, y hace tantos accesos a memoria como datos tenemos y actualiza el registro de direcciones de memoria en cada acceso. Al acabar la transferencia del conjunto de datos libera el bus.
Una vez acabada una ráfaga, si el registro contador no ha llegado a cero, comienza la transferencia de una nueva ráfaga.