Espai grapa

Examen 2017/18-2

Assignatura	Codi	Data	Hora inici
Estructura de computadors	05.573	09/06/2018	18:30

⊂05.573ℜ09ℜ06ℜ18ℜEΞK∈
05.573 09 06 18 EX

Enganxeu en aquest espai una etiqueta identificativa amb el vostre codi personal
Examen

Fitxa tècnica de l'examen

- Comprova que el codi i el nom de l'assignatura corresponen a l'assignatura matriculada.
- Només has d'enganxar una etiqueta d'estudiant a l'espai corresponent d'aquest full.
- No es poden adjuntar fulls addicionals, ni realitzar l'examen en llapis o retolador gruixut.
- Temps total: **2 hores** Valor de cada pregunta: **S'indica a l'enunciat**
- En cas que els estudiants puguin consultar algun material durant l'examen, quins són?

  CAP En cas de poder fer servir calculadora, de quin tipus? CAP
- Si hi ha preguntes tipus test: Descompten les respostes errònies? NO Quant?
- Indicacions específiques per a la realització d'aquest examen:

Examen 2017/18-2

Assignatura	Codi	Data	Hora inici
Estructura de computadors	05.573	09/06/2018	18:30

Enunciats

No es pot utilitzar calculadora. Cal saber interpretar un valor en binari, decimal o hexadecimal per a realitzar l'operació que es demani. I el resultat s'ha d'expressar en el format corresponent.

Valoració de les preguntes de l'examen

Pregunta 1 (20%)

Pregunta sobre la pràctica.
Cal completar les instruccions marcades o afegir el codi que es demana.
Els punts suspensius indiquen que hi ha més codi però no l'heu de completar.
NOTA: En cas que el codi proposat en cada pregunta no es correspongui amb la forma que vosaltres plantejaríeu la resposta, podeu reescriure el codi o part del codi segons el vostre plantejament.

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

Assignatura	Codi	Data	Hora inici
Estructura de computadors	05.573	09/06/2018	18:30

Pregunta 1

1.1 Pràctica – 1a Part

Escriure el fragment de codi assemblador de la subrutina calcIndexP1, per a calcular l'índex per a accedir a la matriu. (No s'ha d'escriure el codi de tota la subrutina).

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; Calcular l'índex (indexMat) per a accedir a la matriu
; (tiles) de DimMatrix * Dimatrix posicions de tipus char (1 byte)
; a partir dels valors de la fila i la columna en el tauler (pantalla)
; indicats per les variables (rowScreen) fila i (colScreen) columna.
; indexMat = ((rowScreen-11)/2)*DimMatrix+((colScreen-11)/4)
; Restem 11 perquè és la posició de la casella [0][0].
; Dividim per 2 les files perquè les files estan separades per una línia.
; Dividim per 4 les columnes perquè les columnes estan separades per 4 espais.
; Multipliquem per DimMatrix, perquè cada fila de la matriu té DimMatrix elements.
; La posició en el tauler (fila:13,columna:19) seria [tiles+6].
;
; Hi ha una funció en C equivalent 'calcIndexMatP1_C', però ATENCIÓ!!!
; té una funcionalitat una mica diferent perquè l'accés
; a matrius en C es fa amb 2 índexs i en assemblador es fa amb un únic índex.
;
; Variables globals utilitzades:
; rowScreen: Fila on volem posicionar el cursor a la pantalla.
; colScreen: Columna on volem posicionar el cursor a la pantalla.
; indexRow : Fila per a accedir a la matriu tiles [0..(dimMatrix-1)]
; indexCol : Columna per a accedir a la matriu 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
```

Assignatura	Codi	Data	Hora inici
Estructura de computadors	05.573	09/06/2018	18:30

1.2 Pràctica – 2a part

Completar el codi de la subrutina checkEndP2. (Només completar els espais marcats, no es poden afegir, ni modificar altres instruccions).

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; Comprovar si l'espai en blanc està a la darrera posició del
; tauler (inferior-dreta) i si totes les peces de la matriu (t)
; rebuda com a paràmetre, estan ordenades (S'ha de recórrer tota la
; matriu (t), de tipus char (1 byte cada posició) sense considerar
; la darrera posició de la matriu que és on hauria d'estar l'espai.
; Si les peces no estan ordenades retornem l'estat rebut com a paràmetre.
; Si les peces sí estan ordenades retornem l'estat "2" per a sortir.
;
; Variables globals utilitzades: Cap
; Paràmetres d'entrada:
; rdi      : Adreça de la matriu que volem validar on guardem els nombres del joc (t).
; rsi(sil) : Estat del joc. 0: Sortir, hem premut la tecla 'ESC' per a sortir.
;                           1: Continuem jugant.
;                           2: Guanya, totes les peces estan ordenades.
;                           3: Peces del tauler incorrectes.
; Paràmetres de sortida:
; rax(al)   : Estat del joc. 0: Sortir, hem premut la tecla 'ESC' per a sortir.
;                            1: Continuem jugant.
;                            2: Guanya, totes les peces estan ordenades.
;                            3: Peces del tauler incorrectes.
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
checkEndP2:
   push rbp
   mov  rbp, rsp
   ...
   mov  rbx, __rdi__        ;adreça matriu.
   mov  r8b, __sil__        ;estat del joc

   ;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. Matriu 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)
```

Examen 2017/18-2

Assignatura	Codi	Data	Hora inici
Estructura de computadors	05.573	09/06/2018	18:30

```
      checkEndP2_Sorted:
      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

Assignatura	Codi	Data	Hora inici
Estructura de computadors	05.573	09/06/2018	18:30

Pregunta 2

2.1

L'estat inicial del computador CISCA just abans de començar l'execució de cada fragment de codi (en cada apartat) és el següent:

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

L'adreça simbòlica W val 000FF0A0h. Quin serà l'estat del computador després d'executar cada fragment de codi? (només modificacions, excloent-hi 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

Assignatura	**Codi**	**Data**	**Hora inici**
Estructura de computadors	05.573	09/06/2018	18:30

2.2

En la memòria d'un computador CISCA tenim emmagatzemades dos vectors A i B de 10 i 20 elements respectivament. Cada element és un nombre enter codificat en complement a 2 amb 32 bits.

Completeu els buits del fragment de codi CISCA per a que implementi la següent sentència en C:

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

Els vectors estan emmagatzemats en posicions consecutives de memòria, com és habitual quan es tradueix codi en C. Per exemple, els elements A[0], A[1], A[2] i A[7] es troben emmagatzemats en les adreces de memòria A, A+4, A+8 i A+28 respectivament. El mateix per al vector B.

Se sap que en R1 es troba emmagatzemat el valor de la variable “i”, i en R2 el de la “j” i que després d'executar-se el fragment de codi tots els registres han de mantenir els valors originals.

```
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

Assignatura	Codi	Data	Hora inici
Estructura de computadors	05.573	09/06/2018	18:30

2.3

Donat el següent fragment de codi de un programa en llenguatge assemblador del CISCA:

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

Traduïu-ho a llenguatge màquina i expresseu-ho en la següent taula. Suposeu que la primera instrucció del codi s'assembla a partir de l'adreça **003FC000h** (que és el valor del PC abans de començar l'execució del fragment de codi). Suposeu que l'adreça simbòlica Q val **00003A00h**. En la següent taula utilitzeu una fila per codificar cada instrucció. Si suposem que la instrucció comença en l'adreça @, el valor Bk de cadascun dels bytes de la instrucció amb adreces @+k per a k=0, 1,… s'ha d'indicar en la taula en hexadecimal en la columna corresponent (recordeu que els camps que codifiquen un desplaçament en 2 bytes o un immediat o una adreça en 4 bytes ho fan en format little endian, això cal tenir-ho en compte escrivint els bytes de menor pes, d'adreça més petita, a l'esquerra i els de major pes, adreça major, a la dreta). Completeu també la columna 'Adreça' que indica per a cada fila l'adreça de memòria del byte B0 de la instrucció que es codifica en aquesta fila de la taula.

A continuació us donem com ajuda les taules de codis:

Taula de codis d'instrucció

B0	Instrucció
22h	MUL
20h	ADD
36h	SAL

Taula de modes d'adreçament (Bk<7..4>)

Camp mode Bk<7..4>	Mode
0h	Immediat
1h	Registre
2h	Memòria
3h	Indirecte
4h	Relatiu
5h	Indexat
6h	Relatiu a PC

Taula de modes d'adreçament (Bk<3..0>)

Camp mode Bk<3..0>	Significat
Nº registre	Si el mode ha d'especificar un registre
0	No s'especifica registre.

		Bk per a k=0..10										
Adreça	**Assemblador**	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

Assignatura	Codi	Data	Hora inici
Estructura de computadors	05.573	09/06/2018	18:30

Pregunta 3

3.1. Memòria cau

Memòria cau d'assignació directa
Tenim un sistema de memòria en el que tots els accessos es fan a paraula (no ens importa quina és la mida d'una paraula). Suposarem que l'espai d'adreces de memòria es descompon en blocs de 8 paraules. Cada bloc comença en una adreça múltiple de 8. Així, el bloc 0 conté les adreces 0, 1, 2, 3, 4, 5, 6, 7, el bloc 1, les adreces 8, 9, 10, 11, 12, 13, 14, 15, i el bloc N les adreces 8*N, 8*N+1, 8*N+2, 8*N+3, 8*N+4, 8*N+5, 8*N+6, 8*N+7.

Suposem que el sistema també disposa d'una memòria cau de 4 línies (on cada línia té la mida d'un bloc, es a dir, 8 paraules). Aquestes línies s'identifiquen com a línies 0, 1, 2 i 3. Quan es fa referència a una adreça de memòria principal, si aquesta adreça no es troba a la memòria cau, es porta tot el bloc corresponent des de la memòria principal a una línia de la memòria cau (així si fem referència a l'adreça 2 de memòria principal portarem el bloc format per les paraules 0, 1, 2, 3, 4, 5, 6, 7).

Suposem que el sistema fa servir una **política d'assignació directa**, de manera que cada bloc de la memòria principal només es pot portar a una línia determinada de la memòria cau.

L'execució d'un programa genera la següent llista de lectures a memòria:

0, 1, 2, 12, 61, 62, 63, 64, 17, 18, 19, 32, 4, 6, 65, 66, 20, 56, 42, 50

3.1.1 La següent taula mostra l'estat inicial de la cau, que conté les primeres 32 paraules de la memòria (organitzades en 4 blocs).

Completar la taula per a mostrar l'evolució de la cau durant l'execució del programa. Per a cada accés cal omplir una columna indicant si es tracta d'un encert o una fallada.

Si és un encert escriurem E en la línia corresponent davant de les adreces del bloc, si és una fallada escriurem F i en aquest cas s'indicarà el nou bloc que es porta a la memòria cau en la línia que li correspongui, expressat de la forma b:e (a_0 - a_7) on b:número de bloc, e:etiqueta i (a_0 - a_7) són les adreces del bloc, on a_0 és la primera adreça del bloc i a_7 és la vuitena (darrera) adreça del bloc.

Línia	Estat 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)

Línia	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

Assignatura	**Codi**	**Data**	**Hora inici**
Estructura de computadors	05.573	09/06/2018	18:30

Línia	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)

Línia	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) Quina és la taxa d'encerts (T_e) ?

T_e = 13 encerts / 20 accessos = 0,65

3.1.2 b) Suposem que el temps d'accés a la memòria cau, o temps d'accés en cas d'encert (t_e), és de 4 ns i el temps total d'accés en cas de fallada (t_f) és de 20 ns. Considerant la taxa d'encerts obtinguda a la pregunta anterior, quin és el temps mitja d'accés a memòria (t_m) ?

$t_m = T_e \times t_e + (1-T_e) \times t_f$ = 0,65 * 4 ns + 0,35 * 20 ns = 2,6 ns + 7 ns = 9,6 ns

Examen 2017/18-2

Assignatura	Codi	Data	Hora inici
Estructura de computadors	05.573	09/06/2018	18:30

3.2 Sistema d'E/S

Es vol analitzar el rendiment de la comunicació de dades entre la memòria d'un processador i un port USB, utilitzant E/S programada, amb les següents característiques:

- Velocitat de transferència del dispositiu d'E/S v_{transf} = 4 MBytes/s = 4000 Kbytes/s
- Temps de latència mitjà del dispositiu $t_{latència}$ = 0
- Adreces dels **registres d'estat** i **dades** del controlador d'E/S: 0F00h i 0F04h
- El bit del **registre d'estat** que indica que el controlador del port d'E/S està disponible és el bit 4, o sigui el cinqué bit menys significatiu (quan val 1 indica que està disponible)
- Processador amb una freqüència de rellotge de 2 GHz, el temps de cicle t_{cicle} = 0,5 ns.
- El processador pot executar 1 instrucció per cicle de rellotge
- Transferència de **escriptura** des de memòria al port d'E/S
- Transferència de N_{dades}= 200000 dades
- La mida d'una dada és m_{dada} = 4 bytes
- Adreça inicial de memòria on resideixen les dades: A0000000h

3.2.1 El següent codi realitzat amb el joc d'instruccions CISCA realitza la transferència descrita abans mitjançant la tècnica d'E/S programada. Completeu el codi.

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

3.2.2 Quant temps dura la transferència del bloc de dades t_{transf_bloc}?

t_{transf_bloc} = $t_{latència}$ + (N_{dades} * t_{transf_dada})
$t_{latència}$ = 0
N_{dades} = 200000
t_{transf_dada} = m_{dada} / v_{transf} = 4 Bytes / 4000 Kbytes/s = 0,001 ms
t_{transf_bloc} = 0 + (200000 * 0,001 ms) = 200 ms = 0,2 s

3.2.3 Si volguéssim fer servir el mateix processador i el mateix programa però amb un dispositiu d'E/S més ràpid, quina és la màxima taxa o velocitat de transferència del nou dispositiu que es podria suportar sense que el dispositiu s'hagués d'esperar?

t_{cicle} = 0,5 ns (nanosegons)
t_{instr} = 0,5 ns / 1 = 0,50 ns

El mínim nombre d'instruccions que ha d'executar el programa per a cada dada transferida són les 8 instruccions: 3, 4, 5, 6, 7, 8, 9 i 10. Executar les 8 instruccions requereix 8 * t_{instr} = 8 * 0,50 ns = 4 ns

Examen 2017/18-2

Assignatura	Codi	Data	Hora inici
Estructura de computadors	05.573	09/06/2018	18:30

Per tant, el temps mínim per a transferir una dada és: 4 ns

Es poden transferir 4 bytes cada 4 ns, es a dir: $4 / 4 * 10^{-9} =$ 1000 Mbyte/s = 1 Gbytes/s

Pregunta 4

4.1

Què entenem per arquitectura d'un computador i quins elements s'associen a una arquitectura?

L'arquitectura del computador fa referència al conjunt d'elements del computador que són visibles des del punt de vista del programador d'assemblador. Els elements habituals associats a l'arquitectura del computador són els següents:

- Joc d'instruccions i modes d'adreçament del computador.
- Tipus i formats dels operands.
- Mapa de memòria i d'E/S.
- Models d'execució.

4.2

4.2.1 Un dels factors bàsics que fan que l'esquema de jerarquia de memòries funcioni satisfactòriament és la proximitat referencial. Quins tipus de proximitat referencial podem distingir? Explicar breument en que consisteix cadascun d'ells.

Distingim dos tipus de proximitat referencial:

1) **Proximitat temporal**. És quan, en un interval de temps determinat, la probabilitat que un programa accedeixi de manera repetida a les mateixes posicions de memòria és molt gran.
La proximitat temporal és deguda principalment a les estructures iteratives; un bucle executa les mateixes instruccions repetidament, de la mateixa manera que les crides repetitives a subrutines.

2) **Proximitat espacial**. És quan, en un interval de temps determinat, la probabilitat que un programa accedeixi a posicions de memòria properes és molt gran.
La proximitat espacial és deguda principalment al fet que l'execució dels programes és seqüencial –s'executa una instrucció darrere l'altra llevat de les bifurcacions–, i també a la utilització d'estructures de dades que estan emmagatzemades en posicions de memòria contigües.

4.2.2 Una manera d'optimitzar les operacions d'E/S per DMA consisteix a reduir el nombre de cessions i recuperacions del bus, mitjançant una modalitat de transferència anomenada mode ràfega. Quin és el funcionament en el cas d'una transferència del dispositiu a la memòria?

Examen 2017/18-2

Assignatura	**Codi**	**Data**	**Hora inici**
Estructura de computadors	05.573	09/06/2018	18:30

Cada cop que el mòdul d'E/S té una dada disponible el controlador de DMA l'emmagatzema en la memòria intermèdia i decrementa el registre comptador. Quan la memòria intermèdia és plena o el comptador ha arribat a zero, sol·licita el bus. Un cop el processador li cedeix el bus, escriu a memòria tot el conjunt de dades emmagatzemades en la memòria intermèdia, i fa tants accessos a memòria com dades tenim i actualitza el registre d'adreces de memòria en cada accés. En acabar la transferència del conjunt de dades allibera el bus.
Un cop acabada una ràfega, si el registre comptador no ha arribat a zero, comença la transferència d'una nova ràfega.