Examen 2018/19-1

Assignatura	Codi	Data	Hora inici
Estructura de computadors	05.573	09/01/2019	18:30

⊂05.573ℜ09ℜ01ℜ19ℜEΞ⊃∈
05.573 09 01 19 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 2018/19-1

Assignatura	Codi	Data	Hora inici
Estructura de computadors	05.573	09/01/2019	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 2018/19-1

Assignatura	Codi	Data	Hora inici
Estructura de computadors	05.573	09/01/2019	18:30

Pregunta 1

1.1 Pràctica – 1a Part

Escriure el fragment de codi assemblador de la subrutina showDigitsP1, per a converteix un valor en dos caràcters ASCII que representin aquest valor. (No s'ha d'escriure el codi de tota la subrutina).

```
;;;;;
; Converteix un valor (value) de tipus int(4 bytes) (entre 0 i 99) en
; dos caràcters ASCII que representin aquest valor. (27 -> '2' '7').
; S'ha de dividir el valor entre 10, el quocient representarà les
; desenes i el residu les unitats, i després s'han de convertir a ASCII
; sumant '0' o 48(codi ASCII de '0') a les unitats i a les desenes.
; Mostra els dígits (caràcter ASCII) a partir de la fila indicada
; per la variable (rowScreen) i a la columna indicada per la variable
; (colScreen).
; Per a posicionar el cursor es cridar a la subrutina gotoxyP1 i per a
; mostrar els caràcters a la subrutina printchP1.
;
; Variables globals utilitzades:
; rowScreen: Fila de la pantalla on posicionem el cursor.
; colScreen: Columna de la pantalla on posicionem el cursor.
; charac   : Caràcter que llegim de teclat.
; value    : Valor que volem mostrar.
;;;;;
showDigitsP1:
   push rbp
   mov  rbp, rsp

   push rax
   push rbx
   push rdx

   mov eax, DWORD[value] ;Valor que volem mostrar
   mov edx, 0

   mov ebx,10
   div ebx          ;EAX=EDX:EAX/EBX, EDX=EDX:EAX mod EBX
                    ;d = val / 10;    u = val % 10;
   add al,'0'       ;d = d + '0';
   add dl,'0'       ;u = u + '0';

   call gotoxyP1              ;gotoxyP1_C();
   mov  BYTE[charac], al
   call printchP1             ;printchP1_C();
   inc  DWORD[colScreen]      ;colScreen++;
   call gotoxyP1              ;gotoxyP1_C();
   mov  BYTE[charac], dl
   call printchP1             ;printchP1_C();

   showDigitsP1_End:
   pop rdx
   pop rbx
   pop rax

   mov rsp, rbp
   pop rbp
   ret
```

Examen 2018/19-1

Assignatura	Codi	Data	Hora inici
Estructura de computadors	05.573	09/01/2019	18:30

1.2 Pràctica – 2a part

Completar el codi de la subrutina updateBoardP2. (Només completar els espais marcats, no es poden afegir, ni modificar altres instruccions).

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; Aquesta subrutina es dóna feta. NO LA PODEU MODIFICAR.
; Mostrar un caràcter (dil) a la pantalla, rebut com a paràmetre,
; en la posició on està el cursor cridant la funció printchP2_C.
; Variables globals utilitzades:
; Cap
; Paràmetres d'entrada :
; rdi(dil): Caràcter que volem mostrar
; Paràmetres de sortida:
; Cap
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
printchP2:

;;;;;
; Converteix un valor (val) de tipus int(4 bytes) (entre 0 i 99) en
; dos caràcters ASCII que representin aquest valor. (27 -> '2' '7').
; S'ha de dividir el valor entre 10, el quocient representarà les
; desenes i el residu les unitats, i després s'han de convertir a ASCII
; sumant '0' o 48(codi ASCII de '0') a les unitats i a les desenes.
; Mostra els dígits (caràcter ASCII) a partir de la fila indicada
; per la variable (rScreen) i a la columna indicada per la variable
; (cScreen).
; Per a posicionar el cursor s'ha de cridar a la subrutina gotoxyP2 i
; per a mostrar els caràcters a la subrutina printchP2,
; implementant correctament el pas de paràmetres.
; Variables globals utilitzades:
; Cap.
; Paràmetres d'entrada :
; rScreen: rdi(edi): Fila de la pantalla on posicionem el cursor.
; cScreen: rsi(esi): Columna de la pantalla on posicionem el cursor.
; val:     rdx(edx): Valor que volem mostrar.
; Paràmetres de sortida:
; Cap.
;;;;;
showDigitsP2:

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; Mostrar els valors de la matriu (mOpenCards) dins el tauler,
; a les posicions corresponents, els moviments i les parelles fetes.
; S'ha de recórrer tota la matriu (mOpenCards), cada posició és de
; tipus char(BYTE)1byte, i per a cada element de la matriu fer:
; Posicionar el cursor en el tauler en funció de les variables
; (rScreen) fila i (cScreen) columan cridant la subrutina gotoxyP2,
; implementant correctament el pas de paràmetres.
; Les variables (rScreen) i (cScreen) s'inicialitzaran, a 10 i 14,
; respectivament i que és la posició a pantalla de la casella [0][0].
; Mostrar els caràcters de cada posició de la matriu (mOpenCards)
; cridant la subrutina printchP2, implementant correctament el pas de
; paràmetres.
; Després, mostrar els moviments (moves) de tipus int(DWORD)4bytes,
; a partir de la posició [19,15] de la pantalla i mostrar les parelles
; fetes (pairs) de tipus int(DWORD)4bytes, a partir de la
; posició [19,24] de la pantalla cridant la subrutina showDigitsP2,
; implementant correctament el pas de paràmetres.
;
; Variables globals utilitzades:
; mOpenCards   : Matriu on guardem les targetes del joc.
;
; Paràmetres d'entrada :
; moves: rdi(edi): Parelles que s'han intentat fer amb èxit o sense.
; pairs: rsi(esi): Parelles que s'han fet.
;
; Paràmetres de sortida:
; Cap.
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
updateBoardP2:
   push rbp
```

Examen 2018/19-1

Assignatura	Codi	Data	Hora inici
Estructura de computadors	05.573	09/01/2019	18:30

```
mov  rbp, rsp

push rbx
push rcx
push rdx
push rsi
push rdi
push r10
push r11

mov edx, edi                        ;edx: moves
mov ebx, esi                        ;ebx: pairs
mov ecx, 0                          ;ecx: indexMat
mov edi, 10                         ;rScreen=10;
mov r10d, 0                         ;i=0
updateBoardP2_bucle_Row:
cmp r10d, ROWDIM                    ;i<ROWDIM
jge updateBoardP2_show

   mov esi, 12                      ;cScreen=12;
   mov r11d, 0                      ;j=0;
   updateBoardP2_bucle_Col:
   cmp r11d, COLDIM                 ;j<COLDIM
   jge updateBoardP2_Col_end
      call gotoxyP2                 ;gotoxyP2_C(rScreen, cScreen);
      push rdi
      mov  dil, _BYTE[mOpenCards+ecx]_;c = mOpenCards[i][j];
      call printchP2                ;printchP2_C(c);
      _pop_  rdi
      inc  rcx
      inc  r11d                     ;j++;
      add  esi, 4                   ;cScreen = cScreen + 4;
   jmp updateBoardP2_bucle_Col

   updateBoardP2_Col_end:
   inc  r10d                        ;i++;
   add  edi, 2                      ;rScreen = rScreen + 2;
jmp updateBoardP2_bucle_Row

updateBoardP2_show:
mov _edi_, 19
mov _esi_, 15
call showDigitsP2                   showDigitsP2_C(19, 15, moves);
mov _esi_, 24
mov _edx_, ebx
call showDigitsP2                   showDigitsP2_C(19, 24, pairs);

updateBoardP2_End:
pop r11
pop r10
pop rdi
pop rsi
pop rdx
pop rcx
pop rbx

mov rsp, rbp
pop rbp
ret
```

Examen 2018/19-1

Assignatura	Codi	Data	Hora inici
Estructura de computadors	05.573	09/01/2019	18:30

Pregunta 2

2.1

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

R2 = 00000050h R4 = 00000010h R8 = 00000200h	M(00000200h) = 0810077Eh M(00000050h) = F0000810h M(00000250h) = 00000810h	Z = 0, C = 0, S = 0, V = 0

Completeu l'estat del computador després d'executar cada codi (indicqueu els valors dels registres en hexadecimal). Suposeu que l'adreça simbòlica A val 200h.

a)

```
XOR R4,R4
MOV R4,[A]
ADD R4,[R2]
```

R4 = 0
R4 = 0810077Eh
R4 = 0810077Eh+F0000810h
= F8100F8Eh

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

b)

```
ADD R8, [A]
SAR R8, 2
```

R8 = 0810097Eh
R8 = 0204025Fh

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

Examen 2018/19-1

Assignatura	Codi	Data	Hora inici
Estructura de computadors	05.573	09/01/2019	18:30

2.2

Donat el següent codi d'alt nivell:

```
do {
     C= C * A;
     B= B / A;
}
while B > C
```

Es proposa la següent traducció a CISCA on hem deixat 6 espais per què els ompliu.

```
      MOV   R0, [A]
DO:   MUL   [C] , R0
      DIV   [B], R0
      MOV   R2, [B]
      CMP   [C], R2
      JLE   DO
```

Examen 2018/19-1

Assignatura	Codi	Data	Hora inici
Estructura de computadors	05.573	09/01/2019	18:30

2.3

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

```
            CMP  R3, [B]
            JGE  Label1
            INC  R3
Label1:     SUB  [B+R1], 4
```

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 00023C00h (que és el valor del PC abans de començar l'execució del fragment de codi). Suposeu que l'adreça simbòlica B val 00000800h. En la següent taula useu una fila per a 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 codifiquin un desplaçament en 2 bytes o un immediat o una adreça en 4 bytes ho fan en format little endian, això calç 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 @ que indica per a cada fila l'adreça de memòria del byte B0 de la instrucció que és codifica en aquesta fila de la taula.

A continuació us donem com a ajuda els taules de codis:

Taula de codis d'instrucció

B0	Instrucció
26h	CMP
46h	JGE
24h	INC
21h	SUB

Taula de modes d'adreçament (Bk<7..4>) i (Bk<3..0>)

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
Num. registre	Si el mode ha d'especificar un registre
0	No s'especifica registre

		Bk per a k=0..10										
@	**Assemblador**	0	1	2	3	4	5	6	7	8	9	10
00023C00h	CMP R3, [B]	26	13	20	00	08	00	00				
00023C07h	JGE Label1	46	60	02	00							
00023C0Bh	INC R3	24	13									
00023C0Dh	SUB [B+R1], 4	21	51	00	08	00	00	00	04	00	00	00
00023C18h												

Examen 2018/19-1

Assignatura	Codi	Data	Hora inici
Estructura de computadors	05.573	09/01/2019	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, 62, 25, 63, 64, 17, 18, 19, 57, 58, 20, 21, 3, 4, 5, 65, 66

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	62
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	25	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	F 3:0 (24 - 31)	F 7:1 (56-63)	7:1 (56-63)	7:1 (56-63)	7:1 (56-63)	7:1 (56-63)

Examen 2018/19-1

Assignatura	Codi	Data	Hora inici
Estructura de computadors	05.573	09/01/2019	18:30

Línia	57	58	20	21	3	4
0	8:2 (64-71)	8:2 (64-71)	8:2 (64-71)	8:2 (64-71)	F 0:0 (0-7)	E 0:0 (0-7)
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)	E 2:0 (16 - 23)	E 2:0 (16 - 23)	2:0 (16 - 23)	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)

Línia	5	65	66			
0	E 0:0 (0-7)	F 8:2 (64-71)	E 8:2 (64-71)			
1	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)			
3	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 = 14 encerts / 20 accessos = 0,70

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 5 ns i el temps total d'accés en cas de fallada (t_f) és de 25 ns. Considerant la taxa d'encerts obtinguda a la pregunta anterior, quin és el temps mitja d'accés a memòria (t_m) ?

$t_m = T_e \times t_e + (1-T_e) \times t_f = 0,70 * 5 \text{ ns} + 0,30 * 25 \text{ ns} = 3,5 \text{ ns} + 7,5 \text{ ns} = 11 \text{ ns}$

Examen 2018/19-1

Assignatura	Codi	Data	Hora inici
Estructura de computadors	05.573	09/01/2019	18:30

3.2 Sistema d'E/S

E/S programada

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} = 10 MBytes/s = 10000 Kbytes/s
- Temps de latència mitjà del dispositiu $t_{latència}$ = 0
- Adreces dels **registres d'estat** i **dades** del controlador d'E/S: 0E00h i 0E04h, respectivament
- 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 1 GHz, el temps de cicle t_{cicle} = 1 ns.
- El processador pot executar 2 instruccions per cicle de rellotge
- Transferència de **escriptura** des de memòria al port d'E/S
- Transferència de N_{dades}= 400000 dades
- La mida d'una dada és m_{dada} = 4 bytes
- Adreça inicial de memòria on resideixen les dades: 80000000h

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, _400000_
2.       MOV R2, 80000000h
3.Bucle: _IN_     R0, [0E00h]  ; llegir 4 bytes
4.       AND R0, _00010000b_
5.       _JE_Bucle
6.       MOV R0,_[R2]_        ; llegir 4 bytes
7.       ADD R2, 4
8.       OUT _[0E04h]_, R0    ; escriure 4 bytes
9.       SUB R3, 1
10.      JNE Bucle
```

Examen 2018/19-1

Assignatura	Codi	Data	Hora inici
Estructura de computadors	05.573	09/01/2019	18:30

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

$t_{transf\_bloc} = t_{latència} + (N_{dades} * t_{transf\_dada})$
$t_{latència} = 0$
$N_{dades} = 400000$
$t_{transf\_dada} = m_{dada} / v_{transf} = 4$ Bytes / 10000 Kbytes/s = 0,0004 ms = 0,4 us
$t_{transf\_bloc} = 0 + (400000 * 0{,}0004$ ms) = 160 ms

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} = 1 ns (nanosegon)
t_{instr} = 1 ns / 2 = 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

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

Examen 2018/19-1

Assignatura	Codi	Data	Hora inici
Estructura de computadors	05.573	09/01/2019	18:30

Pregunta 4

4.1

Què és el bit de signe (S) i quan s'activa? Posa un exemple d'una instrucció que consulti el bit de signe.

El bit de signe és un dels bits d'estat del computador. S'activa si el resultat d'una operació és negatiu. Instruccions de salt condicional com JL consulten el bit de signe per saber si han de saltar o no.

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 del DMA en aquest mode en el cas d'una transferència del dispositiu a la memòria?

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.