Espai grapa

Examen 2017/18-2

Assignatura	**Codi**	**Data**	**Hora inici**
Estructura de computadors	05.573	20/06/2018	15:30

⊂05.573ℜ20ℜ06ℜ18ℜEΞθ∈
05.573 20 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	20/06/2018	15:30

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	20/06/2018	15:30

Pregunta 1

1.1 Pràctica – 1a Part

Escriure un fragment de codi assemblador de la subrutina moveTileP1, per moure la fitxa de la posició on està el cursor en la matriu (tiles) a la posició on està l'espai de la matriu (tiles). (No s'ha d'escriure el codi de tota la subrutina)

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; Moure la fitxa de la casella on està el cursor indicada pel vector
; (rowcolCurScreen) a la casella on hi ha l'espai indicada pel vector
; (rowcolSpaceScreen) si la casella on hi ha l'espai està al costat de
; la casella on està el cursor (dalt, baix, esquerra o dreta).
; (rowcolCurScreen: vector amb la fila rowcolCurScreen[0], en assemblador
; DWORD[rowcolCurScreen+0] i la columna rowcolCurScreen[1], en assemblador
; DWORD[rowcolCurScreen+4] de la posició del cursor).
; (rowcolSpaceScreen: vector amb la fila rowcolSpaceScreen[0], en assemblador
; DWORD[rowcolSpaceScreen+0] i la columna rowcolSpaceScreen[1], en assemblador
; DWORD[rowcolSpaceScreen+4] de la posició de l'espai).
; NOTA: Per a calcular l'índex per a accedir a la matriu tiles a partir
; dels valors de la fila i la columna, del cursor i de l'espai, a la
; pantalla, s'ha de cridar a la subrutina calcIndexMatP1.
; Si la casella on hi ha l'espai no està al costat de la casella on
; hi ha el cursor en la matriu (tiles), no fer el moviment.
; Si la casella on està l'espai sí està al costat de la casella on
; està el cursor:
;   - Moure la fitxa de la posició on està el cursor en la matriu (tiles)
;     a la posició on està l'espai de la matriu (tiles) i posar l'espai
;     en la posició en està el cursor de la matriu (tiles).
;   - Posar els valors del vector (rowcolCurScreen) al vector
;     (rowcolSpaceScreen) per a indicar la posició correcta de l'espai.
;
; No s'ha de mostrar la matriu amb els canvis,
; es fa a la subrutina updateBoardP1.
;
; Variables globals utilitzades:
; tiles            : Matriu on guardem els nombres del joc.
; rowcolSpaceScreen: Vector on tenim la posició de l'espai a la pantalla.
; rowcolCurScreen  : Vector on tenim la posició del cursor a la pantalla.
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
moveTileP1:
   push rbp
   mov  rbp, rsp
   ...
   mov  r10d,DWORD[rowcolCurScreen+0]
   mov  r11d,DWORD[rowcolCurScreen+4]
   mov  r12d,DWORD[rowcolSpaceScreen+0]
   mov  r13d,DWORD[rowcolSpaceScreen+4]
   moveTileP1_If_SameRow:
   cmp  r10d, r12d      ;Mateixa fila
                        ;rowcolCurScreen[0]==rowcolSpaceScreen[0])
   jne  moveTileP1_If_SameCol
      sub r13d, 4       ;(rowcolSpaceScreen[1]-4)
      cmp r11d,r13d
      je  moveTileP1_OK ;rowcolCurScreen[1]==(rowcolSpaceScreen[1]-2)
      add r13d, 8       ;(rowcolSpaceScreen[1]+2)
      cmp r11d,r13d
      je  moveTileP1_OK ;rowcolCurScreen[1]==(rowcolSpaceScreen[1]+4)
      jmp moveTileP1_End
   moveTileP1_If_SameCol:
   cmp  r11d, r13d      ;Mateixa columna
                        ;(rowcolCurScreen[1]==rowcolSpaceScreen[1])
   jne  moveTileP1_End
```

Examen 2017/18-2

Assignatura	**Codi**	**Data**	**Hora inici**
Estructura de computadors	05.573	20/06/2018	15:30

```
    sub r12d, 2       ;(rowcolSpaceScreen[0]-4)
    cmp r10d,r12d
    je  moveTileP1_OK ;rowcolCurScreen[0]==(rowcolSpaceScreen[0]-2)
    add r12d, 4       ;(rowcolSpaceScreen[0]+4)
    cmp r10d,r12d
    je  moveTileP1_OK ;rowcolCurScreen[0]==(rowcolSpaceScreen[0]+2)
    jmp moveTileP1_End
moveTileP1_OK:
    mov  r12d,DWORD[rowcolSpaceScreen+0]
    mov  r13d,DWORD[rowcolSpaceScreen+4]

    ;tiles[rowcolSpaceScreen[0]][rowcolSpaceScreen[1]]=
    ;tiles[ rowcolCurScreen[0] ][ rowcolCurScreen[1] ]




    mov  DWORD[rowScreen], r10d
    mov  DWORD[colScreen], r11d
    call calcIndexMatP1
    mov  ecx, DWORD[indexMat]
    mov  r8b, BYTE[tiles+ecx]

    mov  DWORD[rowScreen], r12d
    mov  DWORD[colScreen], r13d
    call calcIndexMatP1
    mov  edx, DWORD[indexMat]
    mov  BYTE[tiles+edx], r8b




    ;tiles[ rowcolCurScreen[0] ][ rowcolCurScreen[1] ]=' '
    mov  BYTE[tiles+ecx], ' '

    ;rowcolSpaceScreen[0] = rowcolCurScreen[0];
    mov  DWORD[rowcolSpaceScreen+0], r10d
    ;rowcolSpaceScreen[1] = rowcolCurScreen[1]
    mov  DWORD[rowcolSpaceScreen+4], r11d
moveTileP1_End:
...
mov rsp, rbp
pop rbp
ret
```

1.2 Pràctica – 2a part

Completar el codi de la subrutina moveCursorP2. (Només completar els espais marcats, no es poden afegir, ni modificar altres instruccions).

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; Actualitzar la posició del cursor actualitzant el vector (rcCurScreen), rebut com a paràmetre,
; vector amb la fila (DWORD[rowcolCur+0]) i la columna (DWORD[rowcolCurScren+4]) de la posició
; del cursor a la pantalla, en funció de la tecla premuda, rebuda com a paràmetre:
; (i: amunt,j:esquerra, k:avall, l:dreta).
; Comprovar que no sortim del tauler, el vector (rcCurScreen) només pot prendre els valors de
; les posicions del cursor dins del tauler. Si s'ha de sortir del tauler, no fer el moviment.
; No s'ha de posicionar el cursor a la pantalla, es fa a updateBoardP2.
; Variables globals utilitzades: Cap
; Paràmetres d'entrada : rdi(dil): Caràcter llegit de teclat
;                                  'i': amunt, 'j':esquerra, 'k':avall, 'l':dreta
;                        rsi     : Vector on tenim la posició de l'espai a la pantalla.
; Paràmetres de sortida: Cap
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
moveCursorP2:
   push rbp
   mov  rbp, rsp
   ...
    mov   al,   __dil__          ;caràcter llegit de teclat
    mov   ebx,  __DWORD[rsi+0]__
    mov   ecx,  __DWORD[rsi+4]__
  moveCursorP2_j:
      cmp __al__,'i'
      jne moveCursorP2_l
      cmp ebx, 11              ;(rcCurScreen[0]>11)
      jle moveCursorP2_end
      sub ebx,2                ;rcCurScreen[0]=rcCurScreen[0]-2;
      jmp moveCursorP2_end
  moveCursorP2_l:
      cmp al,'k'
      jne moveCursorP2_i
      cmp ebx, 17              ;(rcCurScreen[0]<17)
      jge moveCursorP2_end
      add ebx,2                ;rcCurScreen[0]=rrclCurScreen[0]+2;
      jmp moveCursorP2_end
  moveCursorP2_i:
      cmp al,'j'
      jne moveCursorP2_k
      cmp ecx, 11              ;(rcCurScreen[1]=11)
      jle moveCursorP2_end
      sub ecx,4                ;rcCurScreen[1]=rcCurScreen[1]-4;
      jmp moveCursorP2_end
  moveCursorP2_k:
      cmp al,'l'
      jne moveCursorP2_end
      cmp ecx, 23              ;(rcCurScreen[1]<23)
      jge moveCursorP2_end
      add ecx, 4               ;rcCurScreen[1]=rcCurScreen[1]+4;
  moveCursorP2_end:
    mov   __DWORD[rsi+0]__, ebx
    mov   __DWORD[rsi+4]__, ecx
   ...
   mov rsp, rbp
   pop rbp
   ret
```

Examen 2017/18-2

Assignatura	Codi	Data	Hora inici
Estructura de computadors	05.573	20/06/2018	15:30

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 = 0, S = 0, V = 0

Quin serà l'estat del computador després d'executar cada fragment de codi? (només modificacions, excloent-hi el PC).

a)

```
        XOR   R0, R0
PLUS:   CMP   R0, R2
        JLE   END
        ADD   R0, R1
        JMP   PLUS
END:
```

R0 = 0h

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

b)

```
        SAL  [R0],10h
        MOV R1, [R0]
```

M(00000A10h) = F00F0000h
R1 = F00F0000h

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

Examen 2017/18-2

Assignatura	Codi	Data	Hora inici
Estructura de computadors	05.573	20/06/2018	15:30

2.2

Donat el següent codi d'alt nivell:

```
for (i=0,j=9; j>=0; i++, j--)
        {
                V1[i]= V2[j];
        }
```

V1 i V2 són vectors de 10 elements de 4 bytes cadascun. Es proposa la següent traducció a CISCA on hem deixat 5 llocs per omplir:

```
INI:    MOV R1, 36
        MOV R3, V1
REP:    MOV R4, [V2+R1]
        MOV [R3], R4
        ADD R3, 4
        SUB R1, 4
        CMP R1, 0
        JGE REP
```

Examen 2017/18-2

Assignatura	**Codi**	**Data**	**Hora inici**
Estructura de computadors	05.573	20/06/2018	15:30

2.3

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

```
        CMP [R6+16h], R1
        JNE END
        MOV [A], R10
END:
```

Traduïu-lo a llenguatge màquina i expresseu-lo en la següent taula. Suposeu que la primera instrucció del codi s'assembla a partir de l'adreça 00000900h (que és el valor del PC abans de començar l'execució del fragment de codi). Suposeu que l'adreça simbòlica A val 00000400h. 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 codifiquen un desplaçament en 2 bytes o un immediat o una adreça en 4 bytes ho fan en format little endian, això cal tenir-lo 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 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
42h	JNE
10h	MOV
26h	CMP

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										
@	**Assemblador**	0	1	2	3	4	5	6	7	8	9	10
00000900h	CMP [R6+16h], R1	26	46	16	00	11						
00000905h	JNE END	42	60	07	00							
00000909h	MOV [A], R10	10	20	00	04	00	00	1A				
00000910h												

Examen 2017/18-2

Assignatura	Codi	Data	Hora inici
Estructura de computadors	05.573	20/06/2018	15:30

Pregunta 3

3.1 Memòria cau

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'emplaçament completament associativa**, de manera que qualsevol bloc de la memòria principal es pot portar a qualsevol bloc de la memòria cau. Si trobem que la cau ja està plena, es fa servir un **algorisme de reemplaçament LRU.**

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

7, 8, 5, 24, 3, 18, 55, 6, 17, 18, 32, 40, 4, 6, 63, 40, 48, 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 s'indicarà el nou bloc que es porta a la memòria cau en la línia que li correspongui, expressat de la forma b (a_0 - a_7) on b:número de bloc, 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	7		8		5		24		3	
0	0 (0 - 7)	E	0 (0 - 7)		0 (0 - 7)	E	0 (0 - 7)		0 (0 - 7)	E	0 (0 - 7)
1	1 (8 - 15)		1 (8 - 15)	E	1 (8 - 15)		1 (8 - 15)		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	3 (24 - 31)		3 (24 - 31)		3 (24 - 31)		3 (24 - 31)	E	3 (24 - 31)		3 (24 - 31)

Línia	18		55		6		17		18		32	
0		0 (0 - 7)		0 (0 - 7)	E	0 (0 - 7)		0 (0 - 7)		0 (0 - 7)		0 (0 - 7)
1		1 (8 - 15)	F	6 (48 - 55)		6 (48 - 55)		6 (48 - 55)		6 (48 - 55)		6 (48 - 55)
2	E	2 (16 - 23)		2 (16 - 23)		2 (16 - 23)	E	2 (16 - 23)	E	2 (16 - 23)		2 (16 - 23)
3		3 (24 - 31)		3 (24 - 31)		3 (24 - 31)		7 (56 - 63)		7 (56 - 63)	F	4 (32 - 39)

Examen 2017/18-2

Assignatura	**Codi**	**Data**	**Hora inici**
Estructura de computadors	05.573	20/06/2018	15:30

Línia	40	4	6	63	40	48
0	0 (0 - 7)	E 0 (0 - 7)	E 0 (0 - 7)	0 (0 - 7)	0 (0 - 7)	0 (0 - 7)
1	F 5 (40 - 47)	5 (40 - 47)	5 (40 - 47)	5 (40 - 47)	E 5 (40 - 47)	5 (40 - 47)
2	2 (16 - 23)	2 (16 - 23)	2 (16 - 23)	F 7 (56 - 63)	7 (56 - 63)	7 (56 - 63)
3	4 (32 - 39)	4 (32 - 39)	4 (32 - 39)	4 (32 - 39)	4 (32 - 39)	F 6 (48 - 55)

Línia	56	42	50			
0	0 (0 - 7)	0 (0 - 7)	0 (0 - 7)			
1	5 (40 - 47)	E 5 (40 - 47)	5 (40 - 47)			
2	E 7 (56 - 63)	7 (56 - 63)	7 (56 - 63)			
3	6 (48 - 55)	6 (48 - 55)	E 6 (48 - 55)			

3.1.2 a) Quina és la taxa d'encerts (T_e) ?

T_e = 15 encerts / 20 accessos = 0,75

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 40 ns. Considerant la taxa d'encerts obtinguda a la pregunta anterior, quin és el temps mitja d'accés a memòria (t_m) ?

$t_m = T_e \times t_e + (1-T_e) \times t_f$ = 0,75 * 5 ns + 0,25 * 40 ns = 3,75 ns + 10 ns = 13,75 ns

Examen 2017/18-2

Assignatura	Codi	Data	Hora inici
Estructura de computadors	05.573	20/06/2018	15:30

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} = 2 MBytes/s = 2000 Kbytes/s
- Temps de latència mitjà del dispositiu $t_{latència}$ = 0
- Adreces dels **registres d'estat** i **dades** del controlador d'E/S: 0A00h i 0A04h
- El bit del **registre d'estat** que indica que el controlador del port d'E/S està disponible és el bit 2, o sigui el tercer 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 2 instruccions per cicle de rellotge
- Transferència de **escriptura** des de memòria al port d'E/S
- Transferència de N_{dades}= 160000 dades
- La mida d'una dada és m_{dada} = 4 bytes
- Adreça inicial de memòria on resideixen les dades: C0000000h

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, _160000_
2.        MOV R2, C0000000h
3.Bucle:  _IN_     R0, [0A00h]  ; llegir 4 bytes
4.        AND R0, _00000100b_
5.        _JE_Bucle
6.        MOV R0,_[R2]_         ; llegir 4 bytes
7.        ADD R2, 4
8.        OUT _[0A04h]_, 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}?

$t_{transf\_bloc} = t_{latència} + (N_{dades} * t_{transf\_dada})$
$t_{latència} = 0$
$N_{dades} = 160000$
$t_{transf\_dada} = m_{dada} / v_{transf} = 4$ Bytes / 2000 Kbytes/s = 0,002 ms
$t_{transf\_bloc} = 0 + (160000 * 0{,}002$ ms) = 320 ms = 0,32 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 / 2 = 0,250 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 * 0,250 ns = 2 ns

Examen 2017/18-2

Assignatura	Codi	Data	Hora inici
Estructura de computadors	05.573	20/06/2018	15:30

Per tant, el temps mínim per a transferir una dada és: 2 ns

Es poden transferir 4 bytes cada 2 ns, es a dir: $4 / 2 * 10^{-9} =$ 2000 Mbyte/s = 2 Gbytes/s

Pregunta 4

4.1

Dins del joc d'instruccions d'un processador, per a què serveixen les «instruccions lògiques» i quines són les més habituals.

Les instruccions que fan operacions lògiques permeten manipular de manera individual els bits d'un operand. Les operacions lògiques més habituals són AND, OR, XOR, NOT.

4.2

4.2.1 Quines són les tres polítiques d'assignació per a emmagatzemar dades dins d'una memòria cau? En que consisteixen?

1) Política d'assignació directa: un bloc de la memòria principal només potser en una única línia de la memòria cau. La memòria cau d'assignació directa és la que té la taxa de fallades més alta, però s'utilitza molt perquè és la més barata i fàcil de gestionar.

2) Política d'assignació completament associativa: un bloc de la memòria principal pot ser en qualsevol línia de la memòria cau. La memòria cau completament associativa és la que té la taxa de fallades més baixa. No obstant això, no se sol utilitzar perquè és la més cara i complexa de gestionar.

3) Política d'assignació associativa per conjunts: un bloc de la memòria principal pot ser en un subconjunt de les línies de la memòria cau, però dins del subconjunt pot trobar-se en qualsevol posició. La memòria cau associativa per conjunts és una combinació

4.2.2 En un sistema d'E/S gestionat per DMA. Explica quan i perquè es produeix una interrupció. Serveix per indicar l'inici o el final d'una transferència? Qui la genera?

Finalització de l'operació d'E/S: quan s'ha acabat la transferència del bloc
el controlador de DMA envia una petició d'interrupció al processador per informar que s'ha acabat la transferència de dades.