Examen 2019/2020

Estructura de computadors (Universitat Oberta de Catalunya)

Examen 2019/20-2

Assignatura	Codi	Data	Hora inici
Estructura de computadors	05.573	10/06/2020	18:30

Fitxa tècnica de l'examen

- Comprova que el codi i el nom de l'assignatura corresponen a l'assignatura matriculada.
- Temps total: **2 hores** Valor de cada pregunta: **S'indica a l'enunciat**
- En cas que els estudiants no puguin consultar algun material durant l'examen, quins són?
  CAP
- Es pot utilitzar calculadora? **NO** 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 2019/20-2

Assignatura	Codi	Data	Hora inici
Estructura de computadors	05.573	10/06/2020	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 2019/20-2

Assignatura	Codi	Data	Hora inici
Estructura de computadors	05.573	10/06/2020	18:30

Pregunta 1

1.1 Pràctica – 1a Part

Escriure un fragment de codi assemblador de la subrutina addPairsRP1 que mira si es troba una parella, dos caselles consecutives amb el mateix número, ajuntem la parella posant la suma de la parella a la casella de la dreta, un 0 a la casella de l'esquerra i acumularem aquesta suma (punts que es guanyen).
(No s'ha d'escriure el codi de tota la subrutina).

```
; Aparellar nombres iguals des de la dreta de la matriu (m) i acumular els punts al marcador
; sumant el punts de les parelles que s'hagin fet.
; Recórrer la matriu per files de dreta a esquerra i de baix a dalt.
; Quan es trobi una parella, dos caselles consecutives amb el mateix número, ajuntem la
; parella posant la suma de la parella a la casella de la dreta, un 0 a la casella de
; l'esquerra i acumularem aquesta suma (punts que es guanyen).
; Si una fila de la matriu és:[8,4,4,2] i state='1', quedarà [8,0,8,2], p=p+(4+4) i state='2'.
; Si al final fet alguna parella (punts>0), posarem (state) a '2' per a indicar que s'ha mogut
; algun nombre i actualitzarem la variable (score) amb els punts obtinguts de fer les parelles.
; Per recorrer la matriu en assemblador, en aquest cas, l'index va de la posició 30 (posició [3][3])
; a la 0 (posició [0][0]) amb increments de 2 perquè les dades son de tipus short(WORD) 2 bytes.
; Per a accedir a una posició concreta de la matriu des d'assemblador
; cal tindre en compte que l'índex és:(index=(fila*DimMatrix+columna)*2),
; multipliquem per 2 perquè les dades son de tipus short(WORD) 2 bytes.
; Els canvis s'han de fer sobre la mateixa matriu. No s'ha de mostrar la matriu.
; Variables globals utilitzades: m        : Matriu 4x4 on hi han el números del tauler de joc.
;                                score    : Punts acumulats fins el moment.
;                                state    : Estat del joc. ('2': S'han fet moviments).
;;;;;
addPairsRP1:
   push rbp
   mov  rbp, rsp
   ...
   mov r10, 0                       ;p = 0
   mov rax, (SizeMatrix-1)*2        ;index [i][j]
   mov rbx, 0
   mov rcx, (SizeMatrix-1)*2        ;index [i][j-1]
   mov rdx, 0
   mov r8, DimMatrix                ;i = DimMatrix
   dec r8                           ;i = DimMatrix-1
   addPairsRP1_Rows:
   mov r9, DimMatrix                ;j = DimMatrix
   dec r9                           ;j = DimMatrix-1
      addPairsRP1_Cols:
      mov bx, WORD[m+rax]                ;bx=m[i][j]
      cmp bx, 0                    ;if (m[i][j]!=0)
      je  addPairsRP1_EndIf                                    ; && (m[i][j]==m[i][j-1])) {
                                                               ;    m[i][j]  = m[i][j]*2;
                                                               ;    m[i][j-1]= 0;
                                                               ;    p = p + m[i][j];
                                                               ; }
      mov rcx, rax
        sub rcx, 2
        cmp bx, WORD[m+rcx]            ;m[i][j]==m[i][j-1]
        jne addPairsRP1_EndIf
           shl bx,1                    ;m[i][j]*2
           mov WORD[m+rax], bx         ;m[i][j]  = m[i][j]*2
           mov WORD[m+rcx], 0          ;m[i][j-1]= 0
           add r10w, bx                ;p = p + m[i][j]
   addPairsRP1_EndIf:
```

Assignatura	Codi	Data	Hora inici
Estructura de computadors	05.573	10/06/2020	18:30

```
      sub rax, 2
      dec r9                        ;j--
      cmp r9, 0                     ;j>0
      jg  addPairsRP1_Cols
   sub rax, 2
   dec r8                           ;i--
   cmp r8, 0                        ;i>=0
   jge  addPairsRP1_Rows
   cmp r10w, 0                      ;p > 0
   je  addPairsRP1_End
      mov BYTE[state], '2'         ;satate='2';
      add DWORD[score], r10d       ;score = score + p;
   addPairsRP1_End:
   ...
   mov rsp, rbp
   pop rbp
   ret
```

1.2 Pràctica – 2a part

Completar el codi de la subrutina rotateMatrixRP2. (Només completar els espais marcats, no es poden afegir o modificar altres instruccions).

```
;;;;;
; Copia la matriu, que rebem com a paràmetre (rsi), a la matriu
; destinació, que rebem com a primer paràmetre (rdi).
; La matriu origen no s'ha de modificar,
; els canvis s'han de fer a la matriu destinació.
; Això permetrà copiar dues matrius després d'una rotació
; i gestionar l'opció '(u)Undo' del joc.
; Per recorrer la matriu en assemblador l'índex va de 0 (posició [0][0])
; a 30 (posició [3][3]) amb increments de 2 perquè les dades son de
; tipus short(WORD) 2 bytes.
; No s'ha de mostrar la matriu.
; Variables globals utilitzades:
; Cap.
; Paràmetres d'entrada :
; rdi : Adreça de la matriu destinació.
; rsi : Adreça de la matriu origen.
; Paràmetres de sortida:
; Cap.
;;;;;
copyMatrixP2:

;;;;;
; Rotar a la dreta la matriu, rebuda com a paràmetre (rdi), sobre la matriu (mRotated).
; La primera fila passa a ser la quarta columna, la segona fila passa
; a ser la tercera columna, la tercera fila passa a ser la segona
; columna i la quarta fila passa a ser la primer columna.
; A l'enunciat s'explica en més detall com fer la rotació.
; NOTA: NO és el mateix que fer la matriu transposada.
; La matriu rebuda com a paràmetre no s'ha de modificar, els canvis s'han de fer a (mRotated).
; Per recorrer la matriu en assemblador l'index va de 0 (posició [0][0])a 30 (posició [3][3])
; amb increments de 2 perquè les dades son de tipus short(WORD) 2 bytes.
; Per a accedir a una posició concreta de la matriu des d'assemblador
; cal tindre en compte que l'índex és:(index=(fila*DimMatrix+columna)*2),
; multipliquem per 2 perquè les dades son de tipus short(WORD) 2 bytes.
; Un cop s'ha fet la rotació, copiar la matriu (mRotated) a la matriu
; rebuda com a paràmetre cridant la subrutina copyMatrixP2.
; No s'ha de mostrar la matriu.
; Variables globals utilitzades:
; mRotated : Matriu on guardem els nombres ja rotats
;
```

Examen 2019/20-2

Assignatura	Codi	Data	Hora inici
Estructura de computadors	05.573	10/06/2020	18:30

```
; Paràmetres d'entrada :
; rdi      : Adreça de la matriu que volem rotar.
; Paràmetres de sortida:
; Cap.
;;;;;
rotateMatrixRP2:
   push rbp
   mov  rbp, rsp
   ...
   mov rdx, rdi               ;short mToRotate[DimMatrix][DimMatrix]
   mov r10, 0                 ;[i][j]
   mov r8d, 0                 ;i=0;
   rotateMatrixRP1_fori:
   cmp r8d, DimMatrix         ;i<DimMatrix;
   jge rotateMatrixRP1_endfori
      mov r9d, 0              ;j=0;
      rotateMatrixRP1_forj:
      _cmp_ r9d, DimMatrix          ;j<DimMatrix;
      jge rotateMatrixRP1_endforj
      mov r11, r9             ;r11 = j
      shl r11, DimMatrix/2    ;r11 = j*DimMatrix      (DimMatrix=4)
      add r11, DimMatrix      ;r11 = j*DimMatrix+(DimMatrix)
      dec r11                 ;r11 = j*DimMatrix+(DimMatrix-1)
      sub r11, r8             ;r11 = j*DimMatrix+(DimMatrix-1-i)
      _shl_ r11, 1                ;r11 = j*DimMatrix+(DimMatrix-1-i)*2

      mov bx, __WORD[rdx+r10]__          ;mToRotate[i][j];
      mov __WORD[mRotated+r11]__, bx ;mRotated[j][DimMatrix-1-i]=m[i][j]
      add __r10__, 2                     ;index+2

      inc r9                  ;j++.
      jmp  rotateMatrixRP1_forj
      rotateMatrixRP1_endforj:
      inc r8                  ;i++.
      jmp rotateMatrixRP1_fori

   rotateMatrixRP1_endfori:

   mov  rdi, rdx
   mov   _rsi_, mRotated
   call copyMatrixP2          ;copyMatrixP2_C(mToRotate, mRotated);
   ...
   mov rsp, rbp
   pop rbp
   ret
```

Examen 2019/20-2

Assignatura	Codi	Data	Hora inici
Estructura de computadors	05.573	10/06/2020	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:

R0 = 00000A10h R1 = 00000B20h R2 = 00000C30h R3 = 00000D40h R4= 00000004h	M(00000A10h) = 0000F00Fh M(00000B20h) = 0000F000h M(00000C30h) = 00000FF0h M(00000D40h) = 00000001h M(00001640h) = 00000F00h	Z = 0, C = 0, S = 0, V = 0

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

a)

```
        ADD R0, R2
        XOR [R0], R0
```

R0= 1640
[00001640h]= 00001940h

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

b)

```
CONT:  CMP R4, 0
       JE END
       SUB R4, R0
END:
```

R4= 00000004h – 00000A10h = FFFFF5F4h

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

Examen 2019/20-2

Assignatura	Codi	Data	Hora inici
Estructura de computadors	05.573	10/06/2020	18:30

2.2

Donat el següent codi d'alt nivell:

Si (A[j]<=A[i]*2) A[i]= A[i]-A[j];

A és un vector de 8 elements de 4 bytes cadascun. Es proposa la següent traducció a CISCA on hem deixat 6 espais per omplir.

```
        MOV R0, [j]
        MUL R0, 4
        MOV R2, [A+R0]
        MOV R1, [i]
        MUL R1, 4
        MOV R3, [A+R1]
        SAL R3, 1
        CMP R2, R3
        JG END
        SUB [A+R1], R2
END:
```

Examen 2019/20-2

Assignatura	Codi	Data	Hora inici
Estructura de computadors	05.573	10/06/2020	18:30

2.3

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

```
        MOV R1, [A+R4]
PLUS:   SUB [R10], 10h
        CMP R1,R4
        JNE PLUS
```

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 00006880h (que és el valor del PC abans de començar l'execució del fragment de codi). Suposeu que l'adreça simbòlica A val 00004000h. 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-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 es codifica en aquesta fila de la taula.

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

Tabla de códigos de instrucción

B0	Instrucción
10h	MOV
21h	SUB
26h	CMP
42h	JNE

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
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
00006880h	MOV R1, [A+R4]	10	11	54	00	40	00	00				
00006887h	SUB [R10], 10h	21	3A	00	10	00	00	00				
0000688Eh	CMP R1, R4	26	11	14								
00006891h	JNE PLUS	42	60	F2	FF							
00006895h												

Examen 2019/20-2

Assignatura	Codi	Data	Hora inici
Estructura de computadors	05.573	10/06/2020	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 2019/20-2

Assignatura	**Codi**	**Data**	**Hora inici**
Estructura de computadors	05.573	10/06/2020	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 2019/20-2

Assignatura	Codi	Data	Hora inici
Estructura de computadors	05.573	10/06/2020	18:30

3.2 Sistema d'E/S

3.2.1 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

a) 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
```

b) 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

Examen 2019/20-2

Assignatura	**Codi**	**Data**	**Hora inici**
Estructura de computadors	05.573	10/06/2020	18:30

c) 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 2019/20-2

Assignatura	Codi	Data	Hora inici
Estructura de computadors	05.573	10/06/2020	18:30

Pregunta 4

4.1

Totes les instruccions modifiquen els bits d'estat Z, V, S i C? Raona la resposta i posa exemples del teu raonament

Les instruccions no tenen per què modificar els bits d'estat. Poden modificar algun, tots o cap, depenent del tipus d'instrucció.

Per exemple:

JMP: no modifica els bits d'estat. Es tracta d'una instrucció de salt incondicional que simplement posa en el PC l'adreça continguda en el seu operand.

ADD: és una operació aritmètica que pot modificar tots els bits d'estat citats depenent del resultat.

NOT: es tracta d'una instrucció de negació lògica que no modifica cap dels 4 bits d'estat.

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.