EX SOL - CAT - Estructura de Computadors

Estructura de computadors (Universitat Oberta de Catalunya)

Examen 2020/21-1

Assignatura	**Codi**	**Data**	**Hora inici**
Estructura de computadors	05.573	20/1/2021	15:30

Fitxa tècnica de l'examen

- Comprova que el codi i el nom de l'assignatura corresponen a l'assignatura de què t'has matriculat.
- Temps total: **2 hores** Valor de cada pregunta:
-
- Es pot consultar cap material durant l'examen? NO Quins materials estan permesos? **CAP**
- Es pot fer servir 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:

Enunciat: L'enunciat de l'examen estarà en format PDF.
A l'ordinador des d'on fareu l'examen cal tindre instal·lat algun programari per a poder llegir documents en format PDF. Per exemple, es pot utilitzar el programari gratuït Adobe Acrobat Reader DC, però podeu utilitzar qualsevol altre programari.

Identificació de l'estudiant: No és necessari identificar-se, d'aquesta forma es garanteix que l'examen serà tractat de forma anònima.

Respostes:
S'ha d'identificar cada resposta dins l'examen. És ***obligatori indicar el número de pregunta i l'apartat***, opcionalment també es pot afegir tot o part de l'enunciat si això us ajuda en la resolució de la pregunta.
Si no s'identifica correctament a quina pregunta fa referència la resposta no s'avaluarà.

En cas de ser necessari aplicar un procediment per resoldre alguna pregunta, mostreu clarament i argumenteu el procediment aplicat, no només el resultat. En cas de dubte, si no es poden resoldre pels mecanismes establerts o per manca de temps, feu els supòsits que considereu oportuns i argumenteu-los.

Elaboració document a lliurar:
Utilitzar qualsevol editor de text per crear el document amb les respostes, sempre que després us permeti exportar el document a format PDF per fer el lliurament.

Lliurament: És obligatori lliurar les respostes de l'examen en un únic document ***en format PDF***.
No s'acceptaran altres formats.
És responsabilitat de l'estudiant que la informació que contingui el document PDF que es lliuri reflecteixi correctament les respostes donades a l'examen. Recomanem que obriu el fitxer PDF generat i reviseu atentament les respostes per evitar que s'hagi pogut perdre, canviar o modificar alguna informació al generar el document en format PDF.
El lliurament es pot fer tantes vegades com es vulgui, es corregirà el darrer lliurament que es faci dins l'horari especificat per realitzar l'examen.

COMPROMÍS D'AUTORESPONSABILITAT: aquest examen s'ha de resoldre de forma individual sota la vostra responsabilitat i seguint les indicacions de la fitxa tècnica (sense utilitzar cap material, ni calculadora).
En cas que no sigui així, l'examen s'avaluarà amb un zero. Per altra banda, i sempre a criteri dels Estudis, l'incompliment d'aquest compromís pot suposar l'obertura d'un expedient disciplinari amb possibles sancions.

Examen 2020/21-1

Assignatura	Codi	Data	Hora inici
Estructura de computadors	05.573	20/1/2021	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 : 15%
1.2 : 5%

Pregunta 2 (40%)

2.1 : 10%
2.2 : 15%
2.3 : 15%

Pregunta 3 (40%)

3.1: 20%
3.1.1 : 10%
3.1.2 : 10%

3.2: 20%
3.2.1 : 10%
3.2.2 : 5%
3.2.3 : 5%

Examen 2020/21-1

Assignatura	Codi	Data	Hora inici
Estructura de computadors	05.573	20/1/2021	15:30

Pregunta 1

1.1 Pràctica – 1a Part

Modifiqueu la subrutina moveTileP1 perquè comprovi que la posició indicada per la variable spacePos de la matriu Tiles hi ha un espai, si hi ha l'espai moure la fitxa en la direcció indicada (funcionament normal), si no hi ha espai posar la variable state a '5', cridar la subrutina spacePosScreenP1 i sortir. (No s'ha d'escriure el codi de tota la subrutina, només cal modificar el codi per fer el què es demana).

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; Moure la fitxa en la direcció indicada pel caracter (charac),
; ('i':dalt, 'k':baix, 'j':esquerra o 'l':dreta) a la posició
; on hi ha l'espai dins la matriu indicada per la variable (spacePos),
; controlant el casos en els que no es pot fer el moviment.
; fila    (i = spacePos / DimMatrix)
; columna (j = spacePos % DimMatrix)
; Si la casella on hi ha l'espai està als extrems de la matriu no es
; podrà fer el moviment des d'aquell costat.
; Si es pot fer el moviment:
; Moure la fitxa a la posició on està l'espai de la matriu (tiles)
; i posar l'espai a la posició on hi havia la fitxa moguda.
; Per recorrer la matriu en assemblador l'índex va de 0 (posició [0][0])
; a 9 (posició [2][2]) amb increments de 1 perquè les dades son de
; tipus char(BYTE) 1 byte.
;
; No s'ha de mostrar la matriu amb els canvis, es fa a UpdateBoardP1().
;
; Variables globals utilitzades:
; (tiles)      : Matriu on guardem els nombres del joc.
; (charac)     : Caracter llegit de teclar.
; (spacePos)   : Posició de l'espai a la matriu (tiles).
; (newspacePos): Nova posició de l'espai a la matriu (tiles).
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
moveTileP1:
   push rbp
   mov  rbp, rsp
   ...
   mov r8d, DWORD[spacePos]   ;spacePos
   cmp BYTE[Tile+r8d], ' '
   je  moveTile_Space
     mov BYTE[state], '5'
     call spacePosScreenP1
     jmp moveTileP1_End
   moveTile_Space:
   mov eax, r8d
   mov edx, 0
   mov ebx, DimMatrix
   div ebx
   mov r10d, eax              ;int i = spacePos / DimMatrix;
   mov r11d, edx              ;int j = spacePos % DimMatrix;
   mov r9d, r8d               ;newspacePos = spacePos;
   mov dil, BYTE[charac]      ;switch(charac){
   moveTileP1_Switchi:
   cmp dil, 'i'                    ;case 'i':
   jne moveTileP1_Switchk
      cmp r10d, DimMatrix-1        ;if (i < (DimMatrix-1)) {
      jge moveTileP1_SwitchEnd
         mov r9d, r8d
```

```
        add r9d, DimMatrix        ;newspacePos = spacePos+DimMatrix;
        mov dl, BYTE[tiles+r9d]   ;tiles[i+1][j];
        mov BYTE[tiles+r8d], dl   ;tiles[i][j]= tiles[i+1][j];
        mov BYTE[tiles+r9d], ' '  ;tiles[i+1][j] = ' ';
        jmp moveTileP1_SwitchEnd  ;break
  moveTileP1_Switchk:
  cmp dil, 'k'                    ;case 'k':
  jne moveTileP1_Switchj
     cmp r10d, 0                  ;if (i > 0) {
     jle moveTileP1_SwitchEnd
        mov r9d, r8d
        sub r9d, DimMatrix        ;newspacePos = spacePos-DimMatrix;
        mov dl, BYTE[tiles+r9d]   ;tiles[i-1][j];
        mov BYTE[tiles+r8d], dl   ;tiles[i][j]= tiles[i-1][j];
        mov BYTE[tiles+r9d], ' '  ;tiles[i-1][j] = ' ';
        jmp moveTileP1_SwitchEnd  ;break
  moveTileP1_Switchj:
  cmp dil, 'j'                    ;case 'j':
  jne moveTileP1_Switchl
     cmp r11d, DimMatrix-1        ;if (j < (DimMatrix-1)) {
     jge moveTileP1_SwitchEnd
        mov r9d, r8d
        inc r9d                   ;newspacePos = spacePos+1;
        mov dl, BYTE[tiles+r9d]   ;tiles[i][j+1];
        mov BYTE[tiles+r8d], dl   ;tiles[i][j]= tiles[i][j+1];
        mov BYTE[tiles+r9d], ' '  ;tiles[i][j+1] = ' ';
        jmp moveTileP1_SwitchEnd  ;break
  moveTileP1_Switchl:
  cmp dil, 'l'                    ;case 'l':
  jne moveTileP1_SwitchEnd
     cmp r11d, 0                  ;if (j > 0) {
     jle moveTileP1_SwitchEnd
        mov r9d, r8d
        dec r9d                   ;newspacePos = spacePos-1;
        mov dl, BYTE[tiles+r9d]   ;tiles[i][j-1];
        mov BYTE[tiles+r8d], dl   ;tiles[i][j]= tiles[i][j-1];
        mov BYTE[tiles+r9d], ' '  ;tiles[i][j-1] = ' ';
  moveTileP1_SwitchEnd:
  mov DWORD[newSpacePos], r9d

  moveTileP1_End:
  ...
  mov rsp, rbp
  pop rbp
  ret
```

1.2 Pràctica – 2a part

Feu els canvis necessaris al codi assemblador d'aquesta subrutina considerant que les variables moves i sorted s'ha declarat de tipus char(1 byte), no es poden afegir instruccions, només modificar les instruccions que sigui necessari.

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; Verificar l'estat del joc.
; Si s'han exhaurit els moviments (moves == 0) posar (status='3').
; Si no s'ha trobat l'espai (newSpacePos == sizeMatrix) posar (status='5').
; Si no s'ha pogut fer el moviment (spacePos == newSpacePos) posar (status='2').
; Si estem en cap d'aquest casos:
; Comprovar si la (tiles) és igual a la matriu objectiu (tilesEnd).
; Recorrer tota la matriu (tiles) per files d'esquerra a dreta i de dalt a baix
```

Examen 2020/21-1

Assignatura	**Codi**	**Data**	**Hora inici**
Estructura de computadors	05.573	20/1/2021	15:30

```
; comparant cada posició amb la mateixa posició de la matriu (tilesEnd).
; Si hi ha una posició diferent (sorted=0) i aturar la cerca.
; Si s'ha recorrecorregut tota la matriu i totes les posicions són iguals
; (sorted=1), posar (status = '4') per sortir i indicar que s'ha guanyat.
; Retornar (status).
; Si les peces no estan ordenades posar (status = '1') per continuar jugant.
; Retornar (status).
; Per recorrer la matriu en assemblador l'índex va de 0 (posició [0][0])
; a 9 (posició [2][2]) amb increments de 1 perquè les dades son de
; tipus char(BYTE) 1 byte.
;
; Variables globals utilitzades: Cap.
; Paràmetres d'entrada :
; rdi (edi): (moves)      : Moviments que queden per a ordenar les fitxes.
; rsi (esi): (spacePos)   : Posició de l'espai dins la matriu tiles.
; rdx (edx): (newSpacePos): Nova Posició de l'espai dins la matriu tiles.
; Paràmetres de sortida:  rax (al): (status): Estat del joc.
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
checkStatusP2:
   push rbp
   mov  rbp, rsp
   ...
                                  ;al - char status
   mov r9b, 1                     ;int  sorted = 1;
   mov r10d, 0                    ;i=0
   mov r11d, 0                    ;j=0
   mov r12d, 0                    ;k=0;

   cmp dil, 0                     ;if (moves == 0) {
   jne checkStatusP2_ElseIf1
      mov al, '3'                 ;status= '3';
      jmp checkStatusP2_End
   checkStatusP2_ElseIf1:
   cmp esi, edx                   ;} else if (spacePos == newSpacePos) {
   jne checkStatusP2_While
      mov al, '2'                 ;status= '2';
      jmp checkStatusP2_End
   checkStatusP2_While:           ;} else {
      cmp r12d, SizeMatrix         ;while ((k<SizeMatrix)
      jge checkStatusP2_EndWhile
        cmp r9b, 1                ;&& (sorted==1)){
        jne checkStatusP2_EndWhile
          mov dl, BYTE[tiles+r12d]
          cmp dl, BYTE[tilesEnd+r12d] ;if(tiles[i][j]!=tilesEnd[i][j])
          je  checkStatusP2_EndIf
            mov r9b, 0              ;sorted=0;
          checkStatusP2_EndIf:      ;}
          inc r12d                  ;k++;
      jmp checkStatusP2_While
   checkStatusP2_EndWhile:
   cmp r9b, 1                     ;if (sorted==1)
   jne checkStatusP2_Else
      mov al, '4'                 ;status = '4'
      jmp checkStatusP2_End
   checkStatusP2_Else:            ;} else {
      mov al, '1'                 ;status = '1'
   checkStatusP2_End:
   ...
   mov rsp, rbp
   pop rbp
   ret
```

Examen 2020/21-1

Assignatura	Codi	Data	Hora inici
Estructura de computadors	05.573	20/1/2021	15:30

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 = 00000010h R4 = 00000020h R8 = 00000030h	M(00000020h) = 11223344h M(00000030h) = 77665544h M(00000200h) = 8899AABCh	Z = 0, C = 0, S = 0, V = 0

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

a)

```
MOV R2, [R4]
SAL R2, 1
ADD [R8], R2
```

R2 = 11223344h
R2 = 22446688h
[R8] = 77665544h + 22446688h = 99AABBCCh

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

b)

```
ADD R2, R4
MOV R8, [A]
ADD R8, [R2]
```

R2 = 00000030h
R8 = 8899AABCh
R8 = 8899AABCh + 77665544h= 0h

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

Examen 2020/21-1

Assignatura	Codi	Data	Hora inici
Estructura de computadors	05.573	20/1/2021	15:30

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 7 espais per omplir.

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

Examen 2020/21-1

Assignatura	Codi	Data	Hora inici
Estructura de computadors	05.573	20/1/2021	15:30

2.3

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

```
        TEST   R0,[M+R4]
        JE     BEGIN
        INC    [M]
BEGIN:  CMP    [R2], 1h
```

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 M val 00000400h. En la següent taula useu 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 @ 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:

Taula de codis d'instrucció

B0	Instrucció
43h	JE
26h	CMP
33h	TEST
24h	INC

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
00023C00h	TEST R0,[M+R4]	33	10	54	00	04	00	00				
00023C07h	JE BEGIN	43	60	06	00							
00023C0Bh	INC [M]	24	20	00	04	00	00					
00023C11h	CMP [R2], 1h	26	32	00	01	00	00	00				
00023C18h												

Examen 2020/21-1

Assignatura	Codi	Data	Hora inici
Estructura de computadors	05.573	20/1/2021	15:30

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:

12, 4, 14, 28, 5, 20, 6, 44, 22, 7, 8, 30, 45, 55, 10, 43, 0, 56, 46, 1

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	12		4		14		28		5	
0	0:0 (0 - 7)		0:0 (0 - 7)	E	0:0 (0 - 7)		0:0 (0 - 7)		0:0 (0 - 7)	E	0:0 (0 - 7)
1	1:0 (8 - 15)	E	1:0 (8 - 15)		1:0 (8 - 15)	E	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)		2:0 (16 - 23)
3	3:0 (24 - 31)		3:0 (24 - 31)		3:0 (24 - 31)		3:0 (24 - 31)	E	3:0 (24 - 31)		3:0 (24 - 31)

Línia	20		6		44		22		7		8	
0		0:0 (0 - 7)	E	0:0 (0 - 7)		0:0 (0 - 7)		0:0 (0 - 7)	E	0:0 (0 - 7)		0:0 (0 - 7)
1		1:0 (8 - 15)		1:0 (8 - 15)	F	5:1 (40 - 47)		5:1 (40 - 47)		5:1 (40 - 47)	F	1:0 (8 - 15)
2	E	2:0 (16 - 23)		2:0 (16 - 23)		2:0 (16 - 23)	E	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)		3:0 (24 - 31)

Examen 2020/21-1

Assignatura	Codi	Data	Hora inici
Estructura de computadors	05.573	20/1/2021	15:30

Línia	30	45	55	10	43	0
0	0:0 (0 - 7)	0:0 (0 - 7)	0:0 (0 - 7)	0:0 (0 - 7)	0:0 (0 - 7)	E 0:0 (0 - 7)
1	1:0 (8 - 15)	F 5:1 (40 - 47)	5:1 (40 - 47)	F 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)	6:1 (48 - 55)	6:1 (48 - 55)	6:1 (48 - 55)
3	E 3:0 (24 - 31)	3:0 (24 - 31)	3:0 (24 - 31)	3:0 (24 - 31)	3:0 (24 - 31)	3:0 (24 - 31)

Línia	56	46	1			
0	0:0 (0 - 7)	0:0 (0 - 7)	E 0:0 (0 - 7)			
1	5:1 (40 - 47)	E 5:1 (40 - 47)	5:1 (40 - 47)			
2	6:1 (48 - 55)	6:1 (48 - 55)	6:1 (48 - 55)			
3	F 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 30 ns. Considerant la taxa d'encerts obtinguda a la pregunta anterior, quin és el temps mitja d'accés a memòria (t_m) ?

$t_m = T_e \times t_e + (1-T_e) \times t_f = 0,65 * 4 \text{ ns} + 0,35 * 30 \text{ ns} = 2,6 \text{ ns} + 10,5 \text{ ns} = 13,1 \text{ ns}$

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 $\boldsymbol{v}_{transf}$ = 4 MBytes/s = 4000 Kbytes/s
- Temps de latència mitjà del dispositiu $\boldsymbol{t}_{latència}$ = 0
- Adreces dels **registres d'estat** i **dades** del controlador d'E/S: 1F00h i 1F04h
- El bit del **registre d'estat** que indica que el controlador del port d'E/S està disponible és el bit 5, o el sisè bit menys significatiu (quan val 1 indica que està disponible)
- Processador amb una freqüència de rellotge de 4 GHz, el temps de cicle $\boldsymbol{t}_{cicle}$ = 0,25 ns. El processador pot executar una instrucció en 8 cicles de rellotge
- Transferència de **lectura** des del port d'E/S cap a memòria
- Transferència de $\boldsymbol{N}_{dades}$= 800.000 dades
- La mida d'una dada és $\boldsymbol{m}_{dada}$ = 4 bytes
- Adreça inicial de memòria on resideixen les dades: A0000000h

Examen 2020/21-1

Assignatura	Codi	Data	Hora inici
Estructura de computadors	05.573	20/1/2021	15:30

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, 800000
2.       MOV R2, _A0000000h_
3.Bucle: _IN_ R0, [1F00h]      ; llegir 4 bytes
4.       _AND_ R0, 00100000b
5.       _JE_  Bucle
6.       MOV R0,_[R2]_         ; llegir 4 bytes
7.       ADD R2,_4_
8.       _OUT_ [1F04h], 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}?

$t_{transf\_bloc} = t_{latència} + (N_{dades} * t_{transf\_dada})$
$t_{latència} = 0$
$N_{dades} = 800.000$
$t_{transf\_dada} = m_{dada} / v_{transf} = 4$ Bytes / 4000 KBytes/s = 0,001 ms = 1 us
$t_{transf\_bloc} = 0 + (800.000 * 0{,}001$ ms) = 800 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} = 0,25 ns (nanosegon)
t_{instr} = 0,25 ns *8 = 2 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 * 2 ns = 16 ns

Per tant, el temps mínim per a transferir una dada és: 16 ns

Es poden transferir 4 bytes cada 16 ns, es a dir: $4/ 16 * 10^{-9} =$ 250 MByte/s

Examen 2020/21-1

Assignatura	**Codi**	**Data**	**Hora inici**
Estructura de computadors	05.573	20/1/2021	15:30