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	10/1/2021	12:00

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: **S'indica a l'enunciat**
- 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	10/1/2021	12:00

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	10/1/2021	12:00

Pregunta 1

1.1 Pràctica – 1a Part

Modifiqueu la subrutina spacePosScreenP1 perquè compti els espais en blanc que té la matriu Tiles. Si hi ha 1 únic espai posicionar el cursor al tauler (funcionament normal), sinó posar la variable state a '5' (No s'ha d'escriure el codi de tota la subrutina, només cal modificar el codi per fer el què es demana)

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; Busca on està l'espai en blanc dins la matriu (tiles), posicionar
; el cursor en el lloc on hi ha l'espai
; (rowScreen = (pos / DimMatrix) * 2 + 12)
; (colScreen = (pos % DimMatrix) * 4 + 8)
; i actualitza la nova posició (newSpacePos) de l'espai dins la matriu.
; Recorrer tota la matriu per files d'esquerra a dreta i de dalt a baix.
; Si no hi ha espai (pos = sizeMatrix).
; Per recorrer la matriu en assemblador l'índex va de 0 (posició [0][0])
; a 8 (posició [2][2]) amb increments de 1 perquè les dades son de
; tipus char(BYTE) 1 byte.
;
; Variables globals utilitzades:
; (tiles)      : Matriu on guardem les fitxes del joc.
; (newSpacePos): Posició de l'espai dins la matriu.
; (rowScreen)  : Fila de la pantalla on posicionem el cursor.
; (colScreen)  : Columna de la pantalla on posicionem el cursor.
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
spacePosScreenP1:
   push rbp
   mov  rbp, rsp
   ...
   mov esi, 0                  ;i=0
   mov edi, 0                  ;j=0,
   mov r8d, 0                  ;pos=0;
   mov r10, 0
   pospaceScreenP1_While:      ;while ((pos<SizeMatrix) && (tiles[i][j]!=' ')){
   cmp r8d, SizeMatrix
   jge pospaceScreenP1_If
   cmp BYTE[tiles+r8d], ' '
   jne pospaceScreenP1_NoSpace
      inc r10
   pospaceScreenP1_NoSpace:
      inc r8d                  ;pos++;
      mov eax, r8d
      mov edx, 0
      mov ebx, DimMatrix       ;i = pos / DimMatrix;
      div ebx                  ;j = pos % DimMatrix;
      mov edi, eax
      mov esi, edx
      jmp pospaceScreenP1_While

   pospaceScreenP1_If:
   cmp r8d, SizeMatrix         ;if (pos < SizeMatrix){
   jge pospaceScreenP1_End
   cmp r10, 1
   jne pospaceScreenP1_else:
      shl edi, 1               ;rowScreen = i*2;
      add edi, 12              ;rowScreen = i*2 + 12;
      mov DWORD[rowScreen],edi
      shl esi, 2               ;colScreen = j*4;
      add esi, 8               ;colScreen = j*4 + 8;
      mov DWORD[colScreen],esi
```

```
      call gotoxyP1           ;gotoxyP1_C();
      jmp pospaceScreenP1_End
   pospaceScreenP1_else:
      mov BYTE[State], '5'

   pospaceScreenP1_End:
   mov DWORD[newSpacePos], r8d;newSpacePos=pos;
   ...
   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 rowScreen, colScreen i moves s'ha declarat de tipus short(2 bytes), no es poden afegir instruccions, només modificar les instruccions que sigui necessari.

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; Aquesta subrutina es dóna feta. NO LA PODEU MODIFICAR.
; Situar el cursor en una fila i una columna de la pantalla
; en funció de la fila (edi) i de la columna (esi) rebuts com
; a paràmetre cridant a la funció gotoxyP2_C.
; Variables globals utilitzades: Cap
; Paràmetres d'entrada : rdi(edi): (rowScreen) Fila
;                        rsi(esi): (colScreen) Columna;
; Paràmetres de sortida: Cap
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
gotoxyp2:

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; Mostrar els valors de la matriu (t), rebuda com a paràmetre, a la
; pantalla, dins el tauler a les posicions corresponents.
; S'ha de recórrer tota la matriu (t), de tipus char (1 byte cada
; posició), i per a cada element de la matriu:
; Posicionar el cursor en el tauler cridant la subrutina gotoxyP2_C.
; La posició inicial del cursor és la fila 12 de la pantalla (fila 0
; de la matriu), columna 8 de la pantalla (columna 0 de la matriu).
; Mostrar els caràcters de cada posició de la matriu (t)
; cridant la subrutina printchP2_C.
; Actualitzar la columna (colScreen) de 4 en 4 i al canviar de fila
; (rowScreen) de 2 en 2.
; Per recorrer la matriu en assemblador l'índex va de 0 (posició [0][0])
; a 8 (posició [2][2]) amb increments de 1 perquè les dades son de
; tipus char(BYTE) 1 byte.
; Mostrar el moviments que queden per fer (moves) dins el tauler
; a la fila 8, columna 22 de la pantalla.
;
; Variables globals utilitzades:  Cap
; Paràmetres d'entrada : rdi(rdi): (t)    : Matriu on guardem els nombres del joc.
;                        rsi(esi): (moves): Moviments que queden per ordenar les fitxes.
; Paràmetres de sortida: Cap
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
updateBoardP2:
   push rbp
   mov  rbp, rsp
   ...
   mov rbx, rdi                      ;rbx: t
   mov ax, si                       ;eax: moves
   mov di, 12                        ;rowScreen=12;
   mov r10d, 0                       ;i=0
   mov rcx, 0                        ;indexMat
```

Examen 2020/21-1

Assignatura	**Codi**	**Data**	**Hora inici**
Estructura de computadors	05.573	10/1/2021	12:00

```
updateBoardP2_Bucle_Row:
cmp r10d, DimMatrix                ;i<DimMatrix
jge updateBoardP2_Moves

   mov si, 8                       ;colScreen=8;
   mov r11d, 0                      ;j=0;
   updateBoardP2_Bucle_Col:
   cmp r11d, DimMatrix              ;j<DimMatrix
   jge updateBoardP2_Col_End
      call gotoxyP2                 ;gotoxyP2_C(rowScreen, colScreen);
      push rdi
      mov  dil, BYTE[rbx+rcx]       ;charac = t[i][j];
      call printchP2                ;printchP2_C(t[i][j]);
      pop  rdi
      inc  rcx
      inc  r11d                     ;j++;
      add  si, 4                    ;colScreen = colScreen + 4;
   jmp updateBoardP2_Bucle_Col

   updateBoardP2_Col_End:
   inc  r10d                        ;i++;
   add  di, 2                       ;rowScreen = rowScreen + 2;
jmp updateBoardP2_Bucle_Row

updateBoardP2_Moves:
mov di, 8
mov si, 20
call gotoxyP2                       ;gotoxyP2_C(8, 20);
add  al, '0'                        ;moves = moves + '0';
mov  dil, al
call printchP2                      ;printchP2_C(moves);

updateBoardP2_End:
...
mov rsp, rbp
pop rbp
ret
```

Examen 2020/21-1

Assignatura	Codi	Data	Hora inici
Estructura de computadors	05.573	10/1/2021	12:00

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 = 00000020h R4 = 00000040h R8 = 00000080h	M(00000290h) = 77007700h M(00000060h) = 00000810h M(00000250h) = 00001810h	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 250h.

a)

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

R2 = 00000060h
R8 = 00000810h
R8 = FFFFF000h

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

b)

```
      MOV R2,[A+R4]
      CMP [A],R2
      JNE EXIT1
      JMP EXIT2
EXIT1: ADD R2,R2
EXIT2:
```

R2 = 77007700h
R2= EE00EE00h

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

Examen 2020/21-1

Assignatura	Codi	Data	Hora inici
Estructura de computadors	05.573	10/1/2021	12:00

2.2

Donat el següent codi d'alt nivell:

```
if (A>=B) {
     if (C=B) A=C;
     else C= B;
}
else A=B;
```

Es proposa la següent traducció a CISCA on hem deixat 7 espais per omplir.

```
INI:    MOV  R0, [A]
        MOV  R1, [B]
        MOV  R2, [C]
        CMP  R0, R1
        JL   LSE1
        CMP  R2, R1
        JNE  LSE2
        MOV  R0, R2
        JMP  END
LSE2:   MOV  R2, R1
        JMP  END
LSE1:   MOV  R0, R1
END:    MOV  [A], R0
        MOV  [B], R1
        MOV  [C], R2
```

Examen 2020/21-1

Assignatura	Codi	Data	Hora inici
Estructura de computadors	05.573	10/1/2021	12:00

2.3

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

```
CONT: CMP R2, [A+R4]
      JE END
      ADD R10, [A]
      JMP CONT
END:
```

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 A val 00000040h. 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
20h	ADD
40h	JMP

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
003FC000h	CMP R2, [A+R4]	26	12	54	40	00	00	00				
003FC007h	JE END	43	60	0D	00							
003FC00Bh	ADD R10, [A]	20	1A	20	40	00	00	00				
003FC012h	JMP CONT	40	00	00	C0	3F	00					
003FC018h												

Examen 2020/21-1

Assignatura	Codi	Data	Hora inici
Estructura de computadors	05.573	10/1/2021	12:00

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, 18, 2, 22, 15, 23, 24, 50, 17, 3, 18, 19, 32, 4, 6, 65, 51, 20, 56, 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	18	2	22	15
0	0:0 (0 - 7)	E 0:0 (0 - 7)	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)	1:0 (8 - 15)	E 1:0 (8 - 15)
2	2:0 (16 - 23)	2:0 (16 - 23)	E 2:0 (16 - 23)	2:0 (16 - 23)	E 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)

Línia	23	24	50	17	3	18
0	0:0 (0 - 7)	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)	1:0 (8 - 15)	1:0 (8 - 15)	1:0 (8 - 15)	1:0 (8 - 15)
2	E 2:0 (16 - 23)	2:0 (16 - 23)	F 6:1 (48 - 55)	F 2:0 (16 - 23)	2:0 (16 - 23)	E 2:0 (16 - 23)
3	3:0 (24 - 31)	E 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	10/1/2021	12:00

Línia	19	32	4	6	65	51
0	0:0 (0 - 7)	F 4:1 (32 - 39)	F 0:0 (0 - 7)	E 0:0 (0 - 7)	F 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	E 2:0 (16 - 23)	2:0 (16 - 23)	2:0 (16 - 23)	2:0 (16 - 23)	2:0 (16 - 23)	F 6:1 (48 - 55)
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)

Línia	20	56	50
0	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)
2	F 2:0 (16 - 23)	2:0 (16 - 23)	F 6:1 (48 - 55)
3	3:0 (24 - 31)	F 7:1 (56 - 63)	7:1 (56 - 63)

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

T_e = 11 encerts / 20 accessos = 0,55

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) ?

$t_m = T_e \times t_e + (1-T_e) \times t_f$ = 0,55 * 5 ns + 0,45 * 25 ns = 2,75 ns + 11,25 ns = 14 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 v_{transf} = 20 MBytes/s = 20000 Kbytes/s
- Temps de latència mitjà del dispositiu $t_{latència}$ = 0
- Adreces dels **registres d'estat** i **dades** del controlador d'E/S: 0A10h i 0A14h, 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 2 GHz, el temps de cicle t_{cicle} = 0,5 ns.
- El processador executa una instrucció cada dos cicles de rellotge
- Transferència de **escriptura** des de memòria al port d'E/S
- Transferència de N_{dades}= 2.500.000 dades
- La mida d'una dada és m_{dada} = 4 bytes

Examen 2020/21-1

Assignatura	Codi	Data	Hora inici
Estructura de computadors	05.573	10/1/2021	12:00

- 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, 2500000
2.       MOV R2, _A0000000h_
3.Bucle: IN  R0, _[0A10h]_     ; llegir 4 bytes
4.       AND R0, _00010000b_
5.       _JE_  Bucle
6.       MOV R0,_[R2]_         ; llegir 4 bytes
7.       ADD R2, 4
8.       _OUT_ [0A14h], R0     ; escriure 4 bytes
9.       _DEC_ R3
10.      JNE Bucle
```

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

$t_{transf\_bloc} = t_{latència} + (N_{dades} * t_{transf\_dada})$
$t_{latència} = 0$
$N_{dades} = 2500000$
$t_{transf\_dada} = m_{dada} / v_{transf} = 4$ Bytes / 20000 KBytes/s = 0,0002 ms = 0,2 us
$t_{transf\_bloc} = 0 + (2500000 * 0{,}0002$ ms) = 500 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?

$t_{cicle} = 0{,}5$ ns (nanosegon)
$t_{instr} = 0{,}5$ ns *2 = 1 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 $8 * t_{instr} = 8 * 1$ ns = 8 ns

Per tant, el temps mínim per a transferir una dada és: 8 ns

Es poden transferir 4 bytes cada 8 ns, es a dir: $4/ 8 * 10^{-9} =$ 500 MByte/s

Examen 2020/21-1

Assignatura	**Codi**	**Data**	**Hora inici**
Estructura de computadors	05.573	10/1/2021	12:00