Examen 2020/21-2

Assignatura	**Codi**	**Data**	**Hora inici**
Estructura de computadors	05.573	9/6/2021	12:30

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 2020/21-2

Assignatura	Codi	Data	Hora inici
Estructura de computadors	05.573	9/6/2021	12:30

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 2020/21-2

Assignatura	Codi	Data	Hora inici
Estructura de computadors	05.573	9/6/2021	12:30

Pregunta 1

1.1 Pràctica – 1a Part

Modifiqueu la subrutina openCardP1 perquè verifiqui si la segona targeta que obrim, un cop està girada, és igual a la que hem obert primer, si són iguals decrementar la variable 'pairs' (mida 2 bytes) que indica les parelles que queden per fer.
Després de girar la targeta i incrementar 'state', comprovem que estem obrint la segona targeta mirant si 'state' val 2. A continuació comparem la primera targeta que hem obert, que està guardada a la matriu mOpenCard en la posició indicada per l'element 0 del vector 'vPos' amb la segona targeta oberta, si són iguals decrementem 'pairs'. (No s'ha d'escriure el codi de tota la subrutina, només cal modificar (afegir, eliminar o canviar) el codi per fer el què es demana).

```
;;;;;
; Obrir la targeta de la matriu (mCards) de la posició indicada pel
; cursor dins la matriu indicada per la variable (pos).
; Guardar la posició de la targeta que estem obrint i que tenim a la
; variable (pos) de tipus int(DWORD)4bytes al vector (vPos), de tipus
; int(DWORD)4bytes, on a la posició [0] és per a guardar la posició de
; la 1a targeta que girem (quan state=0) i a la posició [1] és per
; guardar la posició de la 2a targeta que girem (quan state=1).
; vPos[0]:[vPos+0]: Posició de la 1a targeta.
; vPos[1]:[vPos+4]: Posició de la 2a targeta.
; Per accedir a la matriu en C cal calcular la fila i la columna:
; fila    = pos / COLDIM, que pot prendre valors [0 : (ROWDIM-1)].
; columna = pos % COLDIM, que pot prendre valors [0 : (COLDIM-1)].
; En assemblador no és necessari.
; Si la targeta no està girada (!='x') posar-la a la matriu
; (mOpenCards) per a que es mostri.
; Marcar-la amb una 'x'(minúscula) a la mateixa posició de la matriu
; (mCards) per a saber que està girada.
; Passar al següent estat (state++).
; NO s'ha de mostrar la matriu amb els canvis, es fa a updateBoardP1().
; Variables globals utilitzades:
; (mCards)    : Matriu on guardem les targetes del joc.
; (mOpenCards): Matriu on tenim les targetes obertes del joc.
; (pos)       : Posició del cursor dins la matriu.
; (state)     : Estat del joc.
; (vPos)      : Adreça del vector amb les posicions de les targetes obertes.
;;;;;
openCardP1:
   push rbp
   mov  rbp, rsp
   push rbx
   push rdx
   push rdi

   mov ebx, DWORD[pos]
   mov edi, DWORD[state]
   shl edi, 2                      ;state*4
   mov DWORD[vPos+edi], ebx        ;vPos[state] = pos;
```

Examen 2020/21-2

Assignatura	**Codi**	**Data**	**Hora inici**
Estructura de computadors	05.573	9/6/2021	12:30

```
mov dl, BYTE[mCards+ebx]
cmp dl, 'x'                     ;if (mCards[i][j] != 'x') {
je  openCardP1_End
   mov BYTE[mOpenCards+ebx], dl ;mOpenCards[i][j] = mCards[i][j];
   mov BYTE[mCards+ebx], 'x'    ;mCards[i][j] = 'x';
   inc DWORD[state]             ;state++;

   cmp DWORD[State],2
   jl  openCardP1_End
      mov ecx, DWORD[vPos+0]
      cmp dl,mOpenCard[ecx]
      jne openCardP1_End
         dec WORD[pairs]

openCardP1_End:
pop rdi
pop rdx
pop rbx
mov rsp, rbp
pop rbp
ret
```

Examen 2020/21-2

Assignatura	Codi	Data	Hora inici
Estructura de computadors	05.573	9/6/2021	12:30

1.2 Pràctica – 2a part

Feu els canvis necessaris al codi assemblador d'aquesta subrutina considerant que la variable pos 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 : (rowScreen): rdi(edi): Fila
;                        (colScreen): rsi(esi): Columna;
; Paràmetres de sortida: Cap
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
gotoxyP2:

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; Posicionar el cursor a la pantalla, dins el tauler, en funció de la
; posició del cursor dins la matriu, indicada per la variable
; (pos), rebuda com a paràmetre, de tipus int(DWORD)4bytes, a partir
; de la posició [10,12] de la pantalla.
; Per a calcular la posició del cursor a pantalla (rScreen) i
; (cScreen) utilitzar aquestes fórmules:
; rScreen=10+(pos/COLDIM)*2)
; cScreen=12+(pos%COLDIM)*4)
; Per a posicionar el cursor a la pantalla s'ha de cridar a la subrutina
; gotoxyP2.
; Variables globals utilitzades: Cap
; Paràmetres d'entrada : (pos): rdi(edi): Posició del cursor dins la matriu.
; Paràmetres de sortida: Cap
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
posCurScreenP2:
   push rbp
   mov  rbp, rsp
   ...
   mov rax, 0
   mov rdx, 0
   mov ax, di
   mov bx, COLDIM
   div bx            ;eax=pos/COLDIM, edx:pos%COLDIM

   shl eax, 1        ;((pos/COLDIM)*2)
   add eax, 10       ;rScreen=10+((pos/COLDIM)*2);
   shl edx, 2        ;((pos/COLDIM)*4)
   add edx, 12       ;cScreen=12+((pos%COLDIM)*4);

   mov edi, eax
   mov esi, edx
   call gotoxyP2     ;gotoxyP2_C(rScreen, cScreen);

   posCurScreenP2_End:
   ...
   mov rsp, rbp
   pop rbp
   ret
```

Examen 2020/21-2

Assignatura	Codi	Data	Hora inici
Estructura de computadors	05.573	9/6/2021	12:30

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:

R1 = 00000100h R2 = 00000200h R3 = 00000300h	M(00000100h) = 00000300h M(00000200h) = 00000200h M(00000300h) = 00000100h M(00000400h) = 00000500h	Z = 0, C = 0, S = 0, V = 0

Completeu l'estat del computador (registres i bits d'estat) després d'executar cada codi (indiqueu els valors dels registres en hexadecimal).

Suposeu que l'adreça simbòlica A val 400h.

a)

```
MOV R1,[R1]
XOR R1, R2
SUB R1,[R2]
```

R1 = 00000300h
R1 = 00000100h
R1 = FFFFFF00h

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

b)

```
ADD R2, [A]
SAL R2, 14h
```

R2 = 00000700h
R2 = 70000000h

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

Examen 2020/21-2

Assignatura	**Codi**	**Data**	**Hora inici**
Estructura de computadors	05.573	9/6/2021	12:30

2.2

Suposem que tenim el vector V de 5 elements de 32 bits. Completar la traducció del programa en assemblador CISCA perquè executi l'algorisme d'alt nivell mostrat. (Hem deixat 8 espais per omplir)

```
A = 16;
for (i = 4; i >= 0; i--) {
        V[i]= A + i;
        A = A * 2;
}
```

El registre R1 representa la variable **A** i R0 representa la variable **i**

```
        MOV [A], 10h
        MOV R1,[A]
        MOV R0, 4
        MOV R2, 16
COMP:CMP R0, 0
        JL  END
        MOV [V+R2], R1
        ADD [V+R2], R0
        SAL R1, 1
        SUB R0, 1
        SUB R2, 4
        JMP  COMP
END:    MOV [A], R1
        MOV [I], R0
```

Examen 2020/21-2

Assignatura	**Codi**	**Data**	**Hora inici**
Estructura de computadors	05.573	9/6/2021	12:30

2.3

Traduïu a llenguatge màquina el fragment de codi en llenguatge assemblador que us proposem a la 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). L'etiqueta A representa l'adreça **0000FFF4h**.

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
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 k=0..10										
Adreça	**Eti**	**Assemblador**	0	1	2	3	4	5	6	7	8	9	10
00006880h	E1:	MOV R10, [A+R4]	10	1A	54	F4	FF	00	00				
00006887h		SUB [R10], 10h	21	3A	00	10	00	00	00				
0000688Eh		CMP R10, R4	26	1A	14								
00006891h		JNE E2	42	60	06	00							
00006895h		JMP E1	40	00	80	68	00	00					
0000689Bh	E2:												

Examen 2020/21-2

Assignatura	Codi	Data	Hora inici
Estructura de computadors	05.573	9/6/2021	12:30

Pregunta 3

3.1. Memòria cau

Memòria cau d'assignació directa

Tenim un sistema de memòria en el que tots els accessos es realitzen a paraules (no ens importa la mida de la paraula). Suposem 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, és a dir, 8 paraules). Aquestes línies s'identifiquen com línies 0, 1, 2 i 3. Quan es fa una 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 utilitza una **política d'assignació directa**, de manera que cada bloc de la memòria principal sols es pot portar a una línia determinada de la memòria cache.

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

7, 8, 9, 12, 4, 20, 18, 29, 5, 45, 51, 13, 73, 14, 52, 42, 28, 58, 20, 21

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 cache durant l'execució del programa. Indicar únicament aquells accessos que provoquen una fallada de cau. Per a cada fallada ompliu una columna indicant el nou bloc que es porta a la memòria cau en la línia que 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, sent a_0 la primera adreça del bloc i a_7 la vuitena (última) adreça del bloc.

Línia	Estat Inicial	Fallada: 45	Fallada: 51	Fallada: 13	Fallada: 73	Fallada: 14
0	0:0 (0 - 7)					
1	1:0 (8 - 15)	5:1 (40 - 47)		1:0 (8 - 15)	9:2 (72 - 79)	1:0 (8 - 15)
2	2:0 (16 - 23)		6:1 (48 - 55)			
3	3:0 (24 - 31)					

Examen 2020/21-2

Assignatura	Codi	Data	Hora inici
Estructura de computadors	05.573	9/6/2021	12:30

Línia	Fallada: 42	Fallada: 58	Fallada: 20	Fallada:	Fallada:	Fallada:
0						
1	5:1 (40 - 47)					
2			2:0 (16 - 23)			
3		7:1 (56 - 63)				

Línia	Fallada:	Fallada:	Fallada:	Fallada:	Fallada:	Fallada:
0						
1						
2						
3						

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

T_e = 12 encerts / 20 accessos = 0,6

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 en la pregunta anterior, quin és el temps mig d'accés a memòria (t_m)?

$t_m = T_e \times t_e + (1-T_e) \times t_f = 0,6 * 5 \text{ ns} + 0,4 * 25 \text{ ns} = 3 \text{ ns} + 10 \text{ ns} = 13 \text{ ns}$

Examen 2020/21-2

Assignatura	Codi	Data	Hora inici
Estructura de computadors	05.573	9/6/2021	12:30

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} = 40 MBytes/s = 40.000 Kbytes/s.
- Temps de latència mig del dispositiu $t_{latència}$ = 0.
- Adreces dels **registres de dades** i **estat** del controlador d'E/S: 0F00h y 0F04h.
- El bit del **registre d'estat** que indica que el controlador del port d'E/S està disponible és el bit 4, o el cinquè 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 1 instrucció per cicle de rellotge.
- Transferència d'**escriptura** des de la memòria al port d'E/S.
- Transferència de N_{dades}= 800.000 dades
- La mida d'una dada és m_{dada} = 4 bytes
- Adreça inicial de memòria on es troben les dades: 40000000h

3.2.1

El següent codi realitzat amb el repertori d'instruccions CISCA realitza la transferència descrita a dalt mitjançant la tècnica d'E/S programada. Completar el codi.

```
1.        MOV     R3, 800000
2.        MOV     R2, 40000000h
3.Bucle:  _IN_    R0, [0F04h]       ; llegir 4 bytes
4.        AND     R0, _00010000b_
5.        _JE_    Bucle
6.        MOV     R0,_[R2]_         ; llegir 4 bytes
7.        ADD     _R2_, 4
8.        OUT     _[0F00h]_, R0     ; escriure 4 bytes
9.        SUB     R3, __1__
10.       _JNE_   Bucle
```

Assignatura	Codi	Data	Hora inici
Estructura de computadors	05.573	9/6/2021	12:30

3.2.2

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

$t_{transf\_bloc} = t_{latència} + (N_{dades} * t_{transf\_dada})$
$t_{latencia} = 0$
$N_{dades} = 800000$
$t_{transf\_dada} = m_{dada} / v_{transf} = 4$ Bytes / 40000 Kbytes/s = 0,0001 ms = 0,1 us
$t_{transf\_bloc} = 0 + (800000 * 0{,}0001$ ms) = 80 ms

3.2.3

Si volguéssim fer servir el mateix processador i el mateix programa, però amb un dispositiu més ràpid d'E/S, quina seria la taxa o velocitat màxima de transferència del nou dispositiu que es podria suportar sense que el dispositiu hagués d'esperar?

$t_{cicle} = 0{,}5$ ns (nanosegons)

$t_{instr} = 0{,}5$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 $8 * t_{instr} = 8 * 0{,}5$ ns = 4 ns

Per tant, el temps mínim per a transferir una dada és: 4 ns

Es poden transferir 4 bytes cada 4 ns, és a dir: $4 / 4 * 10^{-9} =$ 1000 Mbyte/s = 1 Gbytes/s

Examen 2020/21-2

Assignatura	**Codi**	**Data**	**Hora inici**
Estructura de computadors	05.573	9/6/2021	12:30

Enunciats