Espai grapa

Examen 3

Assignatura	Codi	Data	Hora inici
Estructura de computadors	05.573		

⊂05.573ℜ24ℜ01ℜ15ℜEΞ,∈
05.573 24 01 15 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 en la qual estàs matriculat.**
- **Només has d'enganxar una etiqueta d'estudiant a l'espai corresponent d'aquest full.**
- **No es poden adjuntar fulls addicionals.**
- **No es pot realitzar la prova en llapis ni en retolador gruixut.**
- **Temps total:** 2 **h.**
- **En cas que els estudiants puguin consultar algun material durant l'examen, quin o quins materials poden consultar?**
  No es pot utilitzar calculadora, ni material auxiliar.
- **Valor de cada pregunta:** Pregunta 1 (20%); Pregunta 2 (40%); Pregunta 3 (40%)
- **En cas que hi hagi preguntes tipus test: Descompten les respostes errònies?** NO **Quant?**

- **Indicacions específiques per a la realització d'aquest examen:**

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.

Examen 3

Assignatura	Codi	Data	Hora inici
Estructura de computadors	05.573		

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 (40%)
2.1: 15%
2.2: 15%
2.3: 10%

Pregunta 3 (40%)
3.1: 15%
3.1.1: 10%
3.1.2: 5%

3.2: 25%
3.2.1: 15%
3.2.2: 10%

Examen 3

Assignatura	Codi	Data	Hora inici
Estructura de computadors	05.573		

Pregunta 1
1.1 Pràctica – Part obligatòria

Escriure un fragment de codi assemblador de la subrutina GetPlay que llegeixi els dígits de teclat i els emmagatzemi al vector play. No és necessari netejar la pantalla ni treure els dígits per pantalla.

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; GetPlay
; Llegir una jugada.
; Primer netejar l'espai on es llegeix la combinació amb espais en
; blanc, cridant la subrutina clearArea
; Inicialitzar a zeros el vector play.
; Posar showChar=1, per a indicar que Printch mostri els
; caràcters llegits.
; Llegir la jugada (5 dígits), cridant a la subrutina getch, que deixa el
; caràcter pitjat a la variable carac, i emmagatzemar-la a play
; Imprimir per pantalla la combinació llegida de teclat
;
; Variables utilitzades:
; play    : vector que emmagatzema la jugada
; showChar: 1: mostrar el caràcter llegit
; carac : variable on la subrutina getch deixa el caràcter
; pitjat
; Paràmetres d'entrada :
; Cap
; Paràmetres de sortida:
; Cap
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
GetPlay:
```

```
      mov esi, 0
   GC_readPlay:
      call getch
      mov al, [carac]
      mov [play+esi], al
      inc esi
      cmp esi, 5
      jl GC_readPlay
```

Assignatura	Codi	Data	Hora inici
Estructura de computadors	05.573		

1.2 Pràctica – Part opcional

Completar el codi de la subrutina CompareSecretPlay per a que inicialitzi el vector hits

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
CompareSecretPlay
; Compara la combinació secreta amb la jugada, guardant
; a la variable hits (5 chars) els encerts.
; Per fer la comparació cal fer el següent:
; Per cada element del vector secret compara l'element que
; estem tractant de secret amb l'element de play que ocupa
; la mateixa posició per saber si és un encert a lloc,
; Si és un encert a lloc posem un 1 a la posició corresponent de
; hits, si no és un encert a lloc hem de mirar si és un encert fora de
; lloc i ho marquem a hits amb un 2.
; Situar el cursor en la fila (row) i columna (60) correctes de
; la pantalla cridant la subrutina gotoxy, passant fila i columna
; indicades per row i col mitjançant esi i edi.
; Mostrar els encerts a la pantalla cridant la subrutina
; PrintHits.
;
; Variables utilitzades:
; secret  : vector amb la combinació secreta
; play    : vector amb la darrera jugada.
; hits    : vector que identifica els encerts
; showChar: 1: mostrar el caràcter llegit
;
; Paràmetres d'entrada : Cap
;
; Paràmetres de sortida: Cap
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
CompareSecretPlay:
            ...

            mov esi,0
GSC_inihit:
            mov byte _[hits+esi]_,0
            _inc_ esi
            cmp esi, _5_
            _jl_ GSC_inihit

            mov esi, 0
CSP_nextSecret:
            mov al, [secret+esi]
            cmp al, [play+esi]
            jne CSP_noHitPlace
            mov byte[hits+esi],1
            jmp CSP_noHits

            ...
            ret
```

Examen 3

Assignatura	Codi	Data	Hora inici
Estructura de computadors	05.573		

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
        JG    END
        ADD   R0, R1
        JMP   PLUS
END:
```

R0 = 0h
R0 = 0h + 00000B20h = 00000B20h
R0 = 00000B20h + 00000B20h = 00001640h

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

b)

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

M(00000A10h) = F00F0000h
R1 = F00F0000h

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

Examen 3

Assignatura	Codi	Data	Hora inici
Estructura de computadors	05.573		

2.2

Donat el següent codi en alt nivell:

```
if (i > j) {
        aux= A[i];
        A[i]= A[j];
        A[j]= aux;
}
```

A és un vector de 8 elements de 4 bytes cadascun. Es proposa la següent traducció a CISCA on hem deixat 5 llocs per omplir:

```
      MOV R0,[i]
      MOV R1,[j]
      CMP R0,R1
      JLE END
      SAL R0, 2
      MUL R1, 4
      MOV R2, [A+R0]
      MOV R3, [A+R1]
      MOV [A+R0], R3
      MOV [A+R1], R2
      MOV [aux],R2
END:
```

Examen 3

Assignatura	Codi	Data	Hora inici
Estructura de computadors	05.573		

2.3

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

```
        JNE   end
        SUB   R2,4
end:    MOV   R3, [R2+10]
```

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 000000C0h (que és el valor del PC abans de començar l'execució del fragment de codi). 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
21h	SUB
10h	MOV

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 té que 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
000000C0h	JNE end	42	60	07	00							
000000C4h	SUB R2,4	21	12	00	04	00	00	00				
000000CBh	MOV R3, [R2+10]	10	13	42	0A	00						
000000D0h												

Examen 3

Assignatura	Codi	Data	Hora inici
Estructura de computadors	05.573		

Pregunta 3

3.1 Memòria cau

Memòria cau completament associativa (LRU)

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, 23, 2, 31, 38, 39, 45, 46, 47, 48, 16, 18, 20, 32, 40, 48, 33, 41, 49

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	23	2	31
0	0 (0 - 7)	E 0 (0 - 7)	0 (0 - 7)	0 (0 - 7)	E 0 (0 - 7)	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)	E 2 (16 - 23)	2 (16 - 23)	2 (16 - 23)
3	3 (24 - 31)	3 (24 - 31)	3 (24 - 31)	3 (24 - 31)	3 (24 - 31)	E 3 (24 - 31)

Línia	38	39	45	46	47	48
0	0 (0 - 7)	0 (0 - 7)	0 (0 - 7)	0 (0 - 7)	0 (0 - 7)	F 6 (48 - 55)
1	F 4 (32 - 39)	E 4 (32 - 39)	4 (32 - 39)	4 (32 - 39)	4 (32 - 39)	4 (32 - 39)
2	2 (16 - 23)	2 (16 - 23)	F 5 (40 - 47)	E 5 (40 - 47)	E 5 (40 - 47)	5 (40 - 47)
3	3 (24 - 31)	3 (24 - 31)	3 (24 - 31)	3 (24 - 31)	3 (24 - 31)	3 (24 - 31)

Examen 3

Assignatura	Codi	Data	Hora inici
Estructura de computadors	05.573		

Línia	16	18	20	32	40	48
0	6 (48 - 55)	6 (48 - 55)	6 (48 - 55)	6 (48 - 55)	6 (48 - 55)	E 6 (48 - 55)
1	4 (32 - 39)	4 (32 - 39)	4 (32 - 39)	E 4 (32 - 39)	4 (32 - 39)	4 (32 - 39)
2	5 (40 - 47)	5 (40 - 47)	5 (40 - 47)	5 (40 - 47)	E 5 (40 - 47)	5 (40 - 47)
3	F 2 (16 - 23)	2 (16 - 23)	2 (16 - 23)	2 (16 - 23)	2 (16 - 23)	2 (16 - 23)

Línia	33	41	49			
0	6 (48 - 55)	6 (48 - 55)	E 6 (48 - 55)			
1	E 4 (32 - 39)	4 (32 - 39)	4 (32 - 39)			
2	5 (40 - 47)	E 5 (40 - 47)	5 (40 - 47)			
3	2 (16 - 23)	2 (16 - 23)	2 (16 - 23)			

3.1.2 a) Quina és la taxa de fallades (T_f) ?

T_f = 4 fallades / 20 accessos = 0,2

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 3 ns i el temps total d'accés en cas de fallada (t_f) és de 25 ns. Considerant la taxa de fallades obtinguda a la pregunta anterior, quin és el temps mitja d'accés a memòria (t_m) ?

$t_m = T_f \times t_f + (1-T_f) \times t_e = 0,2 * 25 \text{ ns} + 0,8 * 3 \text{ ns} = 5 \text{ ns} + 2,4 \text{ ns} = 7,4 \text{ ns}$

Examen 3

Assignatura	Codi	Data	Hora inici
Estructura de computadors	05.573		

3.2 Sistema d'E/S

3.2.1 E/S per interrupcions

Es vol analitzar el rendiment de la comunicació de dades entre la memòria d'un processador i un port USB, utilitzant E/S per interrupcions, amb les següents característiques:

- Velocitat de transferència del dispositiu d'E/S v_{transf} = 200 Kbytes/s
- Temps de latència mitjà del dispositiu $t_{latència}$ = 0
- Adreces dels **registres d'estat** i **dades** del controlador d'E/S: 0F00h i 0F04h
- El bit del **registre d'estat** que indica que el controlador del port d'E/S està disponible és el bit 2, o el tercer bit menys significatiu (quan val 1 indica que està disponible)
- Processador amb una freqüència de rellotge de 500 MHz, el temps de cicle t_{cicle} = 2 ns. El processador pot executar 2 instruccions per cicle de rellotge
- Transferència d'**escriptura** des de memòria al port d'E/S
- Transferència de N_{dades}= 10000 dades
- La mida d'una dada és m_{dada} = 4 bytes
- El temps per atendre la interrupció (t_{rec_int}) és de 2 cicles de rellotge

a) Completeu el següent codi CISCA que és una rutina de servei a les interrupcions (RSI) per a transferir a través del dispositiu d'E/S anterior, mitjançant la tècnica de E/S per interrupcions.

Es fa servir una variable global que es representa amb l'etiqueta **Addr,** i que al principi del programa conté l'adreça inicial de memòria on emmagatzemar les dades rebudes.

```
1.   CLI
2.   PUSH R0
3.   PUSH R1
4.   MOV  R1, [Addr]
5.   MOV  R0, [R1]
6.   OUT  [0F04h],R0    ; escriure 4 bytes
7.   ADD  R1, 4
8.   MOV  [Addr], R1
9.   POP  R1
10.  POP  R0
11.  STI
12.  IRET
```

Examen 3

Assignatura	Codi	Data	Hora inici
Estructura de computadors	05.573		

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

El temps d'un cicle, t_{cicle} = 2 ns (nanosegons)

Temps per atendre la interrupció, t_{rec_int}: 2 cicles * 2 ns = 4 ns

Temps d'execució de una instrucció, t_{instr}: t_{cicle} / 2 = 1 ns

Temps d'execució RSI, t_{rsi}: N_{rsi} * t_{instr} = 12 instr. * 1 ns = 12 ns

Temps consumit per CPU en cada interrupció, t_{transf_dada}:

$t_{transf\_dada} = t_{rec\_int} + t_{rsi} = 4 + 12 = 16$ ns

Nombre d'interrupcions produïdes (nombre total de dades, N_{dades}): 10000 interrupcions

Temps consumit en total en TOTES les interrupcions:

$t_{transf\_bloc} = t_{transf\_dada} * N_{dades}$ = 16 ns * 10000 interrupcions = 160000 ns = 0,16 ms (milisegons)

c) Quin és el percentatge d'ocupació del processador? Percentatge que representa el tems de transferència del bloc t_{transf_bloc} respecte al temps de transferència del bloc per part del perifèric t_{bloc}

$t_{bloc} = t_{latència} + (N_{dades} * t_{dada})$
$t_{latència} = 0$
$N_{dades} = 10000$
$t_{dada} = m_{dada} / v_{transf}$ = 4 / 200 Kbytes/s = 0,02 ms
t_{bloc} = 0 + (10000 * 0,02) ms = 200 ms
% ocupació = (t_{transf_bloc} * 100 / t_{bloc}) = (0,16 * 100) / 200 = 0,08%

3.2.2 Qüestions sobre E/S

a) Indica com queden repartides les responsabilitats en una transferència d'E/S segons la tècnica d'E/S que utilitzem:

	Programació	Sincronització	Intercanvi	Finalització
E/S Programada	Processador	Processador	Processador	Processador
E/S per Interrupcions	Processador	Mòdul d'E/S	Processador	Processador
E/S per DMA	Processador	DMA	DMA	Processador
Canals d'E/S	Proc. / Canal	Canal d'E/S	Canal d'E/S	Processador

Examen 3

Assignatura	Codi	Data	Hora inici
Estructura de computadors	05.573		

b) Quina política de prioritats és més flexible, la utilitzada en E/S programada o la utilitzada en un sistema amb Daisy Chain? Explica breument la resposta.

El principal desavantatge del Daisy Chain és que és un sistema de prioritats fix i inalterable –canviar l'ordre de prioritats implica modificar el maquinari– i que, com que té una única línia de petició d'interrupcions, no permet que altres perifèrics més prioritaris siguin atesos si ja atenem una petició d'interrupció sigui quina sigui la prioritat que té.

En E/S programa l'enquesta es fa amb codi i per tant, és molt fàcil modificar-la.

c) En un sistema d'E/S gestionat per interrupcions cal inhibir les interrupcions? Quan? Perquè?

Si les interrupcions estan habilitades, el processador accepta la petició (quan hi ha més d'un dispositiu s'ha de determinar si és prou prioritari per a ser atès, situació que analitzarem més endavant ja que té altres implicacions en el procés). Aleshores cal que el processador avisi el mòdul d'E/S perquè sàpiga que l'està atenent i perquè desactivi la INT, i també cal que el processador inhibeixi les interrupcions per evitar noves peticions.

Si el processador no inhibeix les interrupcions, en executar la primera instrucció de l'RSI, en la fase de comprovació d'interrupció tornarà a acceptar la petició i podria entrar en un bucle infinit executant aquesta primera instrucció indefinidament, ja que tornarà a començar un cicle de reconeixement de la mateixa interrupció.