Examen 2018/19-1

Assignatura	Codi	Data	Hora inici
Estructura de computadors	05.573	12/01/2019	12:00

⊂05.573ℜ12ℜ01ℜ19ℜEΞN∈
05.573 12 01 19 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 matriculada.
- Només has d'enganxar una etiqueta d'estudiant a l'espai corresponent d'aquest full.
- No es poden adjuntar fulls addicionals, ni realitzar l'examen en llapis o retolador gruixut.
- Temps total: **2 hores** Valor de cada pregunta: **S'indica a l'enunciat**
- En cas que els estudiants puguin consultar algun material durant l'examen, quins són?

  CAP En cas de poder fer servir calculadora, 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 2018/19-1

Assignatura	Codi	Data	Hora inici
Estructura de computadors	05.573	12/01/2019	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 : 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 2018/19-1

Assignatura	Codi	Data	Hora inici
Estructura de computadors	05.573	12/01/2019	12:00

Pregunta 1

1.1 Pràctica – 1a Part

Escriure el fragment de codi assemblador que falta a la subrutina openCardP1 per a obrir la targeta de la matriu (mCards) de la posició indicada pel cursor dins la matriu i que tenim al vector (rowcol). (No s'ha d'escriure el codi de tota la subrutina).

```
;;;;;
; Obrir la targeta de la matriu (mCards) de la posició indicada pel
; cursor dins la matriu i que tenim al vector (rowcol).
; rowcol[0] fila i rowcol[1] columna.
; 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 la matriu
; (mCards) per a saber que està girada.
; Passar al següent estat (state++).
; Per accedir a les matrius (mOpenCards) i (mCards) en assemblador
; s'ha de calcular l'índex cridant la subrutina calcIndexP1.
; No s'ha de mostrar la matriu amb els canvis, es fa a updateBoardP1.
; Variables globals utilitzades:
; rowcol     : Vector on tenim la posició del cursor dins la matriu.
; row        : Fila que volem accedir de la matriu (4x5).
; col        : Columna que volem accedir de la matriu (4x5).
; indexMat   : Índex per a accedir a la matriu (4x5).
; mCards     : Matriu on guardem les targetes del joc.
; mOpenCards : Matriu on tenim les targetes obertes del joc.
; state      : Estat del joc.
;;;;;
openCardP1:
   push rbp
   mov  rbp, rsp

   push rbx
   push rdx
   push rsi
   push rdi
   push r10

   mov rax, 0
   mov rbx, 0
   mov rcx, 0

   mov bx, WORD[rowcol+0]          ;int i = rowcol[0];
   mov cx, WORD[rowcol+2]          ;int j = rowcol[1];

   mov  DWORD[row], ebx
   mov  DWORD[col], ecx
   call calcIndexP1
   mov  eax, DWORD[indexMat]

   mov r10b, BYTE[mCards+eax]
   cmp r10b, 'x'                   ;if (mCards[i][j] != 'x') {
   je  openCardP1_End
      mov BYTE[mOpenCards+eax], r10b ;mOpenCards[i][j] = mCards[i][j];
      mov BYTE[mCards+eax], 'x'      ;mCards[i][j] = 'x';
      inc DWORD[state]               ;state++;

   openCardP1_End:
   pop r10
   pop rdi
   pop rsi
   pop rdx
   pop rbx

   mov rsp, rbp
   pop rbp
```

Assignatura	Codi	Data	Hora inici
Estructura de computadors	05.573	12/01/2019	12:00

```
   ret
```

1.2 Pràctica – 2a part

Completar el codi de la subrutina moveCursorP2. (Només completar els espais marcats, no es poden afegir, ni modificar altres instruccions).

```
;;;;;
; Actualitzar la posició del cursor dins la matriu actualitzant el
; vector (rc): vector de tipus short(WORD)2bytes, amb la fila
; (rowcol[0]) i la columna rowcol[1] de la posició del cursor dins
; la matriu, en funció de la tecla premuda que tenim a la
; variable (c), de tipus char(BYTE)1byte,
; (i: amunt, j:esquerra, k:avall, l:dreta).
; Comprovar que no sortim de la matriu, el vector (rc) només
; pot prendre els valors de les posicions dins de la matriu.
; Les files i les columnes s'incrementen de 1 en 1 perquè cada
; posició de la matriu és de tipus char(BYTE)1byte.
; Si s'ha de sortir de la matriu, no fer el moviment.
; NO S'ha de posicionar el cursor a la pantalla cridant UpdateBoardP1_C.
;
; Variables globals utilitzades:
; Cap.
; Paràmetres d'entrada :
; c : rdi(dil): caràcter llegit de teclat
; rc: rsi(rsi): Vector per a indicar la posició del cursor dins la matriu.
; Paràmetres de sortida:
; Cap.
;;;;;
moveCursorP2:
   push rbp
   mov  rbp, rsp

   push rax
   push rbx
   push rcx
   push rsi
   push rdi

   mov rax, 0
   mov rbx, 0
   mov rcx, 0

   _mov_  al,  dil          ;(dil):caràcter llegit de teclat
   mov  bx, _WORD[rsi+0]_
   mov  cx, _WORD[rsi+2]_
   moveCursorP2_j:
      cmp _al_,'i'          ;case 'i':
      jne moveCursorP2_l
      cmp bx, 0             ;rc[0]>0
      jle moveCursorP2_end
      sub bx,1              ;rc[0]=rc[0]-1;;
      jmp moveCursorP2_end
   moveCursorP2_l:
      cmp al,'k'             ;case 'k':
      jne moveCursorP2_i
      cmp bx, (ROWDIM-1)     ;c[0]<(ROWDIM-1)
      jge moveCursorP2_end
      add bx,1               ;rc[0]=rc[0]+1;
      jmp moveCursorP2_end
   moveCursorP2_i:
      cmp al,'j'             ;case 'j':
      jne moveCursorP2_k
      cmp cx, 0              ;rc[1]>0
      jle moveCursorP2_end
      sub cx,1               ;rc[1]=rc[1]-1
```

Examen 2018/19-1

Assignatura	Codi	Data	Hora inici
Estructura de computadors	05.573	12/01/2019	12:00

```
    jmp moveCursorP2_end


moveCursorP2_k:
    cmp al,'l'              ;case 'l':
    jne moveCursorP2_end
    cmp cx, (COLDIM-1)      ;rc[1]<(COLDIM-1)
    jge moveCursorP2_end
    add cx, 1               ;rc[1]=rc[1]+1


moveCursorP2_end:
mov   _WORD[rsi+0]_, bx
mov   _WORD[rsi+2]_, cx

moveCursorP2_End:
pop rdi
pop rsi
pop rcx
pop rbx
pop rax

mov rsp, rbp
pop rbp
ret
```

Examen 2018/19-1

Assignatura	Codi	Data	Hora inici
Estructura de computadors	05.573	12/01/2019	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 = 00000010h R4 = 00000020h R8 = 00000030h	M(00000020h) = 810077E0h M(00000030h) = 00000810h M(00000200h) = FFFFF7F0h	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
MOV [R4], R2
```

R2 = 810077E0h
R2 = 0200EFC0h
[R4] = [00000020h] = 0200EFC0h

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

b)

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

R2 = 00000030h
R8 = FFFFF7F0h
R8 = 0

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

Examen 2018/19-1

Assignatura	Codi	Data	Hora inici
Estructura de computadors	05.573	12/01/2019	12:00

2.2

Suposem que tenim el vector V de 10 elements de 32 bits. Completeu la traducció del programa en assemblador CISCA perquè executi l'algorisme d'alt nivell mostrat, omplint els 6 espais que hem deixat.

```
i= 9;
while i>=0 do  {
     if (V [i] > 10)  V[i] = V[i]-1;
     else V[i] = 0;
     i= i-1;
};
```

```
        MOV     R1, 36
COND:   CMP     R1 , 0
        JL      END
        CMP     [V+R1], 10
        JLE     ELSE
        SUB     [V+R1], 1
        JMP     NEXT
ELSE:           MOV     [V+R1], 0
NEXT:   SUB     R1, 4
        JMP     COND
        END:
```

Examen 2018/19-1

Assignatura	Codi	Data	Hora inici
Estructura de computadors	05.573	12/01/2019	12:00

2.3

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

```
CONT: CMP R2, [A]
      JNE END
      ADD R1, [A+R3]
      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 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 codifiquin un desplaçament en 2 bytes o un immediat o una adreça en 4 bytes ho fan en format little endian, això calç 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 és codifica en aquesta fila de la taula.

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

Taula de codis d'instrucció

B0	Instrucció
42h	JNE
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]	26	12	20	00	40	00	00				
003FC007h	JNE END	42	60	0D	00							
003FC00Bh	ADD R1,[A+R 3]	20	11	53	00	40	00	00				
003FC012h	JMP CONT	40	00	00	C0	3F	00					
003FC018h												

Examen 2018/19-1

Assignatura	Codi	Data	Hora inici
Estructura de computadors	05.573	12/01/2019	12:00

Pregunta 3

3.1. Memòria cau

Memòria cau completament associativa (FIFO)
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 FIFO.**

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

0, 1, 2, 15, 23, 16, 55, 56, 17, 18, 28, 30, 40, 5, 63, 25, 43, 56, 42, 50

Inicialment la memòria cau és buida i s'omple seqüencialment començant per la línia 0.

3.1.1 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	0	1	2	15	23
0	-	F 0 (0 - 7)	E 0 (0 - 7)	E 0 (0 - 7)	0 (0 - 7)	0 (0 - 7)
1	-	-	-	-	F 1 (8 - 15)	1 (8 - 15)
2	-	-	-	-	-	F 2 (16 - 23)
3	-	-	-	-	-	-

Línia	16	55	56	17	18	28
0	0 (0 - 7)	0 (0 - 7)	F 7 (56 - 63)	7 (56 - 63)	7 (56 - 63)	7 (56 - 63)
1	1 (8 - 15)	1 (8 - 15)	1 (8 - 15)	1 (8 - 15)	1 (8 - 15)	F 3 (24 - 31)
2	E 2 (16 - 23)	2 (16 - 23)	2 (16 - 23)	E 2 (16 - 23)	E 2 (16 - 23)	2 (16 - 23)
3	-	F 6 (48 - 55)	6 (48 - 55)	6 (48 - 55)	6 (48 - 55)	6 (48 - 55)

Examen 2018/19-1

Assignatura	Codi	Data	Hora inici
Estructura de computadors	05.573	12/01/2019	12:00

Línia	30	40	5	63	25	43
0	7 (56 - 63)	7 (56 - 63)	7 (56 - 63)	E 7 (56 - 63)	7 (56 - 63)	7 (56 - 63)
1	E 3 (24 - 31)	3 (24 - 31)	3 (24 - 31)	3 (24 - 31)	E 3 (24 - 31)	3 (24 - 31)
2	2 (16 - 23)	F 5 (40 - 47)	5 (40 - 47)	5 (40 - 47)	5 (40 - 47)	E 5 (40 - 47)
3	6 (48 - 55)	6 (48 - 55)	F 0 (0 - 7)	0 (0 - 7)	0 (0 - 7)	0 (0 - 7)

Línia	56	42	50			
0	E 7 (56 - 63)	7 (56 - 63)	F 6 (48 - 55)			
1	3 (24 - 31)	3 (24 - 31)	3 (24 - 31)			
2	5 (40 - 47)	E 5 (40 - 47)	5 (40 - 47)			
3	0 (0 - 7)	0 (0 - 7)	0 (0 - 7)			

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 2 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) ?

$t_m = T_e \times t_e + (1-T_e) \times t_f$ = 0,55 * 2 ns + 0,45 * 30 ns = 1,1 ns + 13,5 ns = 14,6 ns

Examen 2018/19-1

Assignatura	Codi	Data	Hora inici
Estructura de computadors	05.573	12/01/2019	12:00

3.2 Sistema d'E/S

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} = 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: 0BF0h i 0BF4h
- 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 de **lectura** 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: A0000000h
- El temps per atendre la interrupció (t_{rec_int}) és de 2 cicles de rellotge

3.2.1

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.   PUSHR1
4.   _IN_R0, [0BF4h]
5.   MOV R1, [Addr]
6.   MOV _[R1]_,R0
7.   ADD _R1_,4
8.   MOV _[Addr]_,R1
9.   POP R1
10.  POP R0
11.  STI
12.  _IRET_
```

Examen 2018/19-1

Assignatura	Codi	Data	Hora inici
Estructura de computadors	05.573	12/01/2019	12:00

3.2.2
Quant temps dedica la CPU a la transferència del bloc de dades t_{transf_bloc}?

El temps d'un cicle, t_{cicle} = 0,5 ns (nanosegons)

Temps per atendre la interrupció, t_{rec_int}: 2 cicles * 0,5 ns = 1 ns

Temps d'execució de una instrucció, t_{instr}: t_{cicle}= 0,5 ns

Temps d'execució RSI, t_{rsi}: N_{rsi} * t_{instr} = 12 instr. * 0,5 ns = 6 ns

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

t_{transf_dada} = t_{rec_int} + t_{rsi} = 1 + 6 = 7 ns

Nombre d'interrupcions produïdes (nombre total de dades, N_{dades}): 400000 interrupcions

Temps consumit en total en TOTES les interrupcions:

t_{transf_bloc} = t_{transf_dada} * N_{dades} = 7 ns * 400000 interrupcions = 2800000 ns = 2,8 ms (milisegons)

3.2.3
Quin és el percentatge d'ocupació del processador? Percentatge que representa el temps 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} = 400000
t_{dada} = m_{dada} / v_{transf} = 4 / 10000 Kbytes/s = 0,0004 ms
t_{bloc} = 0 + (400000 * 0,0004) ms = 160 ms
% ocupació = (t_{transf_bloc} * 100 / t_{bloc}) = (2,8 * 100) / 160 = 1,75%

Examen 2018/19-1

Assignatura	Codi	Data	Hora inici
Estructura de computadors	05.573	12/01/2019	12:00

Pregunta 4

4.1

En què consisteix la manera d'adreçament relatiu a registre índex?. A més d'explicar-ho posa una instrucció d'exemple en la qual s'utilitzi.

L'adreça de memòria es troba explícitament en la instrucció i el desplaçament en un registre que cridarem índex.
L'adreça final de l'operand es calcula sumant l'adreça explícita amb el valor del registre. Un exemple en CISCA és: ADD R1, [vector+R3]

4.2

4.2.1
En la memòria cau, quines polítiques d'assignació es defineixen? Descriure-les breument.

1) **Política d'assignació directa**: un bloc de la memòria principal només pot ser 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 memoria 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 memoria principal pot ser en un subconjunt de les línies de la memòria cau, però dins del subconjunt pot trobar-se en qualsevol posició.

4.2.2
Quins són els passos bàsics per a la gestió d'una interrupció en un sistema amb una única línia d'interrupció i un únic mòdul d'E/S?

1.- Petició del mòdul d'Entrada/Sortida
2.- Cicle de reconeixement de la interrupció
    2.a.- Reconeixement de la interrupció
    2.b.- Salvaguarda de l'estat del processador
    2.c.- Crida a la RSI
3.- Execució de la rutina de servei d'interrupció
    3.a.-Inici de l'execució de la RSI
    3.b.- Intercanvi de la dada
    3.c Finalització de l'execució de la RSI
    3.d Retorn d'interrupció: Restaurar l'estat del processador.

Examen 2018/19-1

Assignatura	**Codi**	**Data**	**Hora inici**
Estructura de computadors	05.573	12/01/2019	12:00