Examen 2019/2020

Estructura de computadors (Universitat Oberta de Catalunya)

Examen 2019/20-2

Assignatura	Codi	Data	Hora inici
Estructura de computadors	05.573	20/06/2020	15:30

Fitxa tècnica de l'examen

- Comprova que el codi i el nom de l'assignatura corresponen a l'assignatura matriculada.
- Temps total: **2 hores** Valor de cada pregunta: **S'indica a l'enunciat**
- En cas que els estudiants no puguin consultar algun material durant l'examen, quins són?
  CAP
- Es pot utilitzar 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:

Examen 2019/20-2

Assignatura	Codi	Data	Hora inici
Estructura de computadors	05.573	20/06/2020	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 : 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 2019/20-2

Assignatura	Codi	Data	Hora inici
Estructura de computadors	05.573	20/06/2020	15:30

Pregunta 1

1.1 Pràctica – 1a Part

Escriure un fragment de codi assemblador de la subrutina copyMatrixP1 que fa el bucle interior per a recorre les columnes de la matriu mentre es còpia la matriu m sobre mRotated. (No s'ha d'escriure el codi de tota la subrutina)

```
; Copiar els valors de la matriu (mRotated) a la matriu (m).
; La matriu (mRotated) no s'ha de modificar,
; els canvis s'han de fer a la matriu (m).
; Per recorrer la matriu en assemblador l'índex va de 0 (posició [0][0])
; a 30 (posició [3][3]) amb increments de 2 perquè les dades son de
; tipus short(WORD) 2 bytes.
; No es mostrar la matriu.
; Variables globals utilitzades:
; m        : matriu 4x4 on hi han el números del tauler de joc.
; mRotated : matriu 4x4 per a fer la rotació.
;;;;;
copyMatrixP1:
   push rbp
   mov  rbp, rsp
   ...
   mov rax, 0       ;index [i][j] matriu m i mRotated
   mov r8, 0                       ;i=0
   copyMatrixP1_fori:
   cmp r8, DimMatrix               ;i<DimMatrix
   jge copyMatrixP1_endfori
                                   ; for (j=0; j<DimMatrix; j++) {
                                   ;    m[i][j] = mRotated[i][j];
                                   ; }
```

```
      mov r9, 0                          ;    j=0
      copyMatrixP1_forj:
      cmp r9, DimMatrix                  ;j<DimMatrix
      jge copyMatrixP1_endforj

      mov bx, WORD[mRotated+rax]         ;mRotated[i][j]
      mov WORD[m+rax], bx                ;m[i][j] = mRotated[i][j];

      add rax, 2                         ;index+2
      inc r9                             ;j++
```

```
      jmp  copyMatrixP1_forj
      copyMatrixP1_endforj:
      inc r8                          ;i++
      jmp copyMatrixP1_fori
   copyMatrixP1_endfori:
   ...
   mov rsp, rbp
   pop rbp
   ret
```

Examen 2019/20-2

Assignatura	Codi	Data	Hora inici
Estructura de computadors	05.573	20/06/2020	15:30

1.2 Pràctica – 2a part

Completar el codi de la subrutina updateBoardP2. (Només completar els espais marcats, no es poden afegir o modificar altres instruccions).

```
;;;;;
; Convertir el número rebut com a paràmetre (edx) de tipus int(DWORD) de 6 dígits
; (número <= 999999), a caràcters ASCII que representen el seu valor.
; Si el número és més gran que 999999 canviarem el valor a 999999.
; S'ha de dividir el valor entre 10, de forma iterativa, fins obtenir els 6 dígits.
; A cada iteració, el residu de la divisió que és un valor entre (0-9) indica el valor del dígit
; que s'ha de convertir a ASCII ('0' - '9') sumant '0' (48 decimal) per a poder-lo mostrar.
; Quan el quocient sigui 0 mostrarem espais a la part no significativa.
; Per exemple, si number=103 mostrarem "   103" i no "000103".
; S'han de mostrar els dígits (caràcters ASCII) des de la posició indicada per la fila (edi)
; i la columna (esi) rebuts com a paràmetre, posició de les unitats, cap a l'esquerra.
; Com el primer dígit que obtenim són les unitats, després les desenes, ..., per a mostrar el valor
; s'ha de desplaçar el cursor una posició a l'esquerra a cada iteració.
; Per a posicionar el cursor cridar a la subrutina gotoxyP2 i per a mostrar els caràcters a
; la subrutina printchP2 implementant correctament el pas de paràmetres.
; Variables globals utilitzades: Cap.
; Paràmetres d'entrada : rdi (edi): Fila on el volem mostrar a la pantalla.
;                        rsi (esi): Columna on el volem mostrar a la pantalla.
;                        rdx (edx): Número que volem mostrar a la pantalla.
; Paràmetres de sortida: Cap.
;;;;;
showNumberP2:

;;;;;
; Actualitzar el contingut del Tauler de Joc amb les dades de la matriu (m) de 4x4 valors de
; tipus short (WORD) i els punts del marcador (score) que s'han fet.
; S'ha de recórrer tota la matriu (m), i per a cada element de la matriu posicionar el cursor
; a la pantalla i mostrar el nombre d'aquella posició de la matriu.
; Després, mostrar el marcador que rebem com a parametre (edi) a la part inferior del tauler,
; fila 18, columna 26 cridant la subrutina showNumberP2.
; Finalment posicionar el cursor a la fila 18, columna 28 cridant la subrutina gotoxyP2.
; Variables globals utilitzades: m : matriu 4x4 on hi han els nombres del tauler de joc.
; Paràmetres d'entrada :  rdi (edi): punts acumulats fins al moment.
; Paràmetres de sortida:  Cap.
;;;;;
updateBoardP2:
   push rbp
   mov  rbp, rsp
   ...
   mov r8d, edi     ;src;
   mov eax, 0       ;i
   mov ebx, 0       ;j
   mov edi, 0       ;rScreen
   mov esi, 0       ;cScreen
   mov rcx, 0       ;índex per a accedir a la matriu m. (0-15)
                    ;índex=(fila*DimMatrix)+(columna)*4

   ;Iniciem el bucle per a mostrar la matriu.
   mov edi, 10      ;rScreen = 10;
   mov eax, 0       ;i=0;
   updateBoardP2_fori:
   cmp eax, DimMatrix    ;i<DimMatrix;
   jge updateBoardP2_endfori

      mov esi, 17        ;cScreen = 17;
```

Examen 2019/20-2

Assignatura	Codi	Data	Hora inici
Estructura de computadors	05.573	20/06/2020	15:30

```
    mov ebx, 0          ;j=0;
    updateBoardP2_forj:
    cmp ebx, DimMatrix ;j<DimMatrix;
    _jge_ updateBoardP2_endforj
       mov   edx, 0
       mov    dx, _WORD[m+rcx]_  ;m[i][j];
       call showNumberP2    ;showNumberP2_C(rScreen,cScreen, m[i][j]);

     add esi, 9         ;cScreen = cScreen + 9;
     add rcx, _2_          ;index
     inc ebx            ;j++.

    jmp  updateBoardP2_forj

    updateBoardP2_endforj:

    add edi, 2          ;rScreen = rScreen + 2;
    inc eax             ;i++.

    jmp updateBoardP2_fori

updateBoardP2_endfori:

mov   _edi_, 18
mov   _esi_, 26
mov  edx, r8d          ;src;
call showNumberP2     ;showNumberP2_C(18, 26, scr);
mov  esi, 28
_call_ gotoxyP2          ;gotoxyP1_C(18,28);
...
mov rsp, rbp
pop rbp
ret
```

Examen 2019/20-2

Assignatura	Codi	Data	Hora inici
Estructura de computadors	05.573	20/06/2020	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:

R0 = 00000000h R1 = 00000A10h R2 = 00000B20h R3 = 00000C30h R4 = 00000D40h	M(00000A10h) = F0000000h M(00000B20h) = 80000000h M(00000C30h) = 0000FFFFh M(000000F0h) = 00000001h M(00000D40h) = 00000010h	Z = 0, C = 0, S = 0, V = 0

Quin serà l'estat del computador després d'executar cada fragment de codi? (només modificacions, excloent el PC).

a)

```
MOV R0, [R1]
ADD R0, [R2]
```

R0= F00000000h
R0= 700000000h

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

b)

```
CMP [R1], R2
JG FIN
SAR R3, [R4]
FIN:
```

JG no salta
R3= 00000000h

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

Examen 2019/20-2

Assignatura	Codi	Data	Hora inici
Estructura de computadors	05.573	20/06/2020	15:30

2.2

Donat el següent codi d'alt nivell:

Si (A[i+1]==A[i]) A[i]= A[i+1] * 2;

A és un vector de 8 elements de 4 bytes cadascun. Es proposa la següent traducció a CISCA on hem deixat 6 espais per omplir.

```
      MOV R0, [i]
      MUL R0, 4
      ADD R0, 4
      MOV R2, [A+R0]
      DEC R0, 4
      CMP R2,[A+R0]
      JNE END
      MOV [A+R0], R2
      SAL [A+R0], 1
END:
```

Examen 2019/20-2

Assignatura	Codi	Data	Hora inici
Estructura de computadors	05.573	20/06/2020	15:30

2.3

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

```
        ADD R0, [R2+100h]
        JE END
        MUL R0, [R4]
        SAL R10, [A]
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 00006880h (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 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	Instrucción
35h	SAL
22h	MUL
20h	ADD
43h	JE

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
00006880h	ADD R0,[R2+100h]	20	10	42	00	01						
00006885h	JE END	43	60	0A	00							
00006889h	MUL R0, [R4]	22	10	34								
0000688Ch	SAL R10, [A]	35	1A	20	00	40	00	00				
00006893h												

Examen 2019/20-2

Assignatura	Codi	Data	Hora inici
Estructura de computadors	05.573	20/06/2020	15:30

3.1. Memòria cau

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:

12, 13, 25, 26, 17, 8, 22, 3, 24, 62, 5, 63, 64, 17, 18, 19, 57, 58, 20, 25

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		12		13		25		26		17
0	0 (0 - 7)		0 (0 - 7)		0 (0 - 7)		0 (0 - 7)		0 (0 - 7)		0 (0 - 7)
1	1 (8 - 15)	E	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)		2 (16 - 23)		2 (16 - 23)	E	2 (16 - 23)
3	3 (24 - 31)		3 (24 - 31)		3 (24 - 31)	E	3 (24 - 31)	E	3 (24 - 31)		3 (24 - 31)

Examen 2019/20-2

Assignatura	**Codi**	**Data**	**Hora inici**
Estructura de computadors	05.573	20/06/2020	15:30

Línia	8	22	3	24	62	5
0	0 (0 - 7)	0 (0 - 7)	E 0 (0 - 7)	0 (0 - 7)	0 (0 - 7)	E 0 (0 - 7)
1	E 1 (8 - 15)	1 (8 - 15)	1 (8 - 15)	1 (8 - 15)	F 7 (56 - 63)	7 (56 - 63)
2	2 (16 - 23)	E 2 (16 - 23)	2 (16 - 23)	2 (16 - 23)	2 (16 - 23)	2 (16 - 23)
3	3 (24 - 31)	3 (24 - 31)	3 (24 - 31)	E 3 (24 - 31)	3 (24 - 31)	3 (24 - 31)

Línia	63	64	17	18	19	57
0	0 (0 - 7)	0 (0 - 7)	0 (0 - 7)	0 (0 - 7)	0 (0 - 7)	0 (0 - 7)
1	E 7 (56 - 63)	7 (56 - 63)	7 (56 - 63)	7 (56 - 63)	7 (56 - 63)	E 7 (56 - 63)
2	2 (16 - 23)	F 8 (64 - 71)	8 (64 - 71)	8 (64 - 71)	8 (64 - 71)	8 (64 - 71)
3	3 (24 - 31)	3 (24 - 31)	F 2 (16 - 23)	E 2 (16 - 23)	E 2 (16 - 23)	2 (16 - 23)

Línia	58	20	25			
0	0 (0 - 7)	0 (0 - 7)	F 3 (24 – 31)			
1	E 7 (56 - 63)	7 (56 - 63)	7 (56 - 63)			
2	8 (64 - 71)	8 (64 - 71)	8 (64 - 71)			
3	2 (16 - 23)	E 2 (16 - 23)	2 (16 - 23)			

3.1.2 a)

Quina és la taxa d'encerts (T_e) ?

T_e = 16 encerts / 20 accessos = 0,8

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 40 ns. Considerant la taxa d'encerts obtinguda a la pregunta anterior, quin és el temps mitja d'accés a memòria (t_m) ?

$t_m = T_e \times t_e + (1-T_e) \times t_f = 0,8 * 5 \text{ ns} + 0,2 * 40 \text{ ns} = 4 \text{ ns} + 8 \text{ ns} = 12 \text{ ns}$

Examen 2019/20-2

Assignatura	Codi	Data	Hora inici
Estructura de computadors	05.573	20/06/2020	15:30

3.2 Sistema d'E/S

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} = 1 MBytes/s = 1000 Kbytes/s
- Temps de latència mitjà del dispositiu $t_{latència}$ = 0
- Adreces dels **registres d'estat** i **dades** del controlador d'E/S: 0B00h i 0B04h
- 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 1 GHz, el temps de cicle t_{cicle} = 1 ns. El processador pot executar 4 instruccions per cicle de rellotge
- Transferència de **lectura** des del port d'E/S cap a memòria
- Transferència de N_{dades}= 100000 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.   IN   R0, [0B04h]
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 2019/20-2

Assignatura	Codi	Data	Hora inici
Estructura de computadors	05.573	20/06/2020	15:30

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

El temps d'un cicle, t_{cicle} = 1 ns (nanosegons)

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

Temps d'execució de una instrucció, t_{instr}: t_{cicle} / 4 = 0,250 ns

Temps d'execució RSI, t_{rsi}: N_{rsi} * t_{instr} = 12 instr. * 0,250 ns = 3 ns

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

t_{transf_dada} = t_{rec_int} + t_{rsi} = 2 + 3 = 5 ns

Nombre d'interrupcions produïdes (nombre total de dades, N_{dades}): 100000 interrupcions

Temps consumit en total en TOTES les interrupcions:

t_{transf_bloc} = t_{transf_dada} * N_{dades} = 5 ns * 100000 interrupcions = 500000 ns = 0,5 ms (milisegons)

c) 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} = 100000
t_{dada} = m_{dada} / v_{transf} = 4 / 1000 Kbytes/s = 0,004 ms
t_{bloc} = 0 + (100000 * 0,004) ms = 400 ms
% ocupació = (t_{transf_bloc} * 100 / t_{bloc}) = (0,5 * 100) / 400 = 0,125%

Examen 2019/20-2

Assignatura	Codi	Data	Hora inici
Estructura de computadors	05.573	20/06/2020	15:30

Pregunta 4

4.1

Quina diferència hi ha en assemblador entre un salt condicional i un altre incondicional? Què operant codifiquem en la instrucció en cada cas? Posa algun exemple de tots dos tipus.

Els salts condicionals són aquells que prenen la decisió de salt consultant els bits d'estat en funció d'una condició codificada en la instrucció. Juntament amb el codi d'operació es codifica el desplaçament o nombre de bits que s'han de saltar, cap a endavant o cap a enrere. Per exemple JE, JNE, JG són instruccions de salt condicional.

Els salts incondicionals no depenen d'una condició. Es codifica l'adreça real de salt. Per exemple JMP.

4.2

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

- **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.

- **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.

- **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.

b) 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.