Examen 2015/16-1

Assignatura	Codi	Data	Hora inici
Estructura de computadors	05.573	23/01/2016	15:30

⊂05.573ℜ23ℜ01ℜ16ℜEΞ2∈
05.573 23 01 16 EX

Enganxeu en aquest espai una etiqueta identificativa
amb el vostre codi personal
Examen

Aquest enunciat correspon també a les assignatures següents:

- 05.096 - Ampliació d'estructura i tecnologia de computadors

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 (35%); Pregunta 3 (35%); Pregunta 4 (10%)
- **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 2015/16-1

Assignatura	Codi	Data	Hora inici
Estructura de computadors	05.573	23/01/2016	15:30

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 2015/16-1

Assignatura	Codi	Data	Hora inici
Estructura de computadors	05.573	23/01/2016	15:30

Pregunta 1

1.1 Pràctica – Part obligatòria

Escriure un fragment de codi assemblador de la subrutina calcIndexP1, per a calcular l'índex per a accedir a una matriu 4x4 de tipus WORD. (No s'ha d'escriure el codi de tota la subrutina)

```
;;;;;
; Calcular el valor de l'índex per a accedir a una matriu (4x4) que
; guardarem a la variable (indexMat) a partir de la fila indicada
; per la variable (row) i la columna indicada per la variable (col).
; indexMat=((row*DimMatrix)+(col))*2
; multipliquem per 2 perquè és una matriu de tipus short (WORD) 2 bytes.
; Aquesta subrutina no té una funció en C equivalent.
; Variables utilitzades: row : fila per a accedir a la matriu m.
;                        col : columna per a accedir a la matriu m.
;                        indexMat: índex per a accedir a la matriu m.
; Paràmetres d'entrada : Cap
; Paràmetres de sortida: Cap
;;;;;
calcIndexP1:
    push rbp
    mov  rbp, rsp
    ...
    mov rax, 0
    mov rbx, 0
    mov rdx, 0

      mov eax, DWORD[row]   ;
      mov ebx, DimMatrix    ;
      mul ebx               ;multipliquem per 4 (EDX:EAX = EAX; font)
      add eax, DWORD[col]   ;eax = ([row]*DimMatrix)+([col])
      shl eax, 1            ;eax = ([row]*DimMatrix)+([col])*2

    mov DWORD[indexMat], eax   ;indexMat=([row]*DimMatrix)+([col])*2
    calcIndexP1_End:
    ...
    mov rsp, rbp
    pop rbp
    ret
```

Examen 2015/16-1

Assignatura	Codi	Data	Hora inici
Estructura de computadors	05.573	23/01/2016	15:30

1.2 Pràctica – Part opcional

Completar el codi de la subrutina shitfNumbersLP2. (Només completar els espais marcats, no es poden afegir o modificar altres instruccions)

```
;;;;;
; Desplaça a l'esquerra els nombres de cada fila de la matriu rebuda com a paràmetre
; (edi), mantenint l'ordre dels nombres i posant els zeros a la dreta.
; Recórrer la matriu per files d'esquerra a dreta i de dalt a baix.
; Si es desplaça un nombre (NO ELS ZEROS), s'han de comptar els desplaçaments.
; Si s'ha mogut algun nombre (nombre de desplaçaments>0), posarem la variable (moved) a 1.
; Retornarem el nombre de desplaçaments fets.
; Si una fila de la matriu és: [0,2,0,4] i el número de desplaçaments es 0, quedarà [2,4,0,0] i
; el número de desplaçaments serà 2. Per a accedir a les matrius des d'assemblador cal calcular
; l'índex a partir de la fila i la columna cridant la subrutina calcIndexP1. m[row][col], en C, és
; equivalent a WORD[m+eax], en assemblador, si eax = ((row*DimMatrix)+(col))*2. m[1][2] és [m+12].
; Els canvis s'han de fer sobre la mateixa matriu. No s'ha de mostrar la matriu.
; Variables utilitzades: moved : Per indicar si s'han fet canvis a la matriu.
; Paràmetres d'entrada : rdi(edi): Adreça de la matriu que volem desplaçar els nombres.
; Paràmetres de sortida: rax(eax) : Número de desplaçaments fets.
;;;;;
shiftNumbersLP2:
    ...
    mov edx, edi                        ;Adreça de la matriu
    mov ecx, 0                          ;shifts=0;
    mov r8d, 0                          ;i = r8d
    shiftNumbersLP2_Rows:
    mov r9d , 0                         ;j = r9d
        shiftNumbersLP2_Cols:
        mov edi, r8d
        mov esi, r9d
        call calcIndexP2
        cmp WORD[edx+eax], 0            ;if (mShift[i][j] == 0)
        jne shiftNumbersLP2_IsZero
            mov r10d, r9d
            inc r10d ;k = j+1           ;Busquem un nombre diferent de zero.
            shiftNumbersLP2_While:
            cmp r10d, DimMatrix         ;k<DimMatrix
            jge shiftNumbersLP2_EndWhile
                mov edi, r8d
                mov esi, r10d
                call calcIndexP2
                cmp WORD[edx+eax], 0    ;mShift[i][k]==0
                jne shiftNumbersLP2_EndWhile
                    inc r10d            ;k++;
                jmp shiftNumbersLP2_While
            shiftNumbersLP2_EndWhile:
            cmp r10d, DimMatrix         ;k==DimMatrix
            je shiftNumbersLP2_Next
                mov bx, __WORD__[__edx+eax__]
                mov WORD[edx+eax], 0    ;mShift[i][k]= 0
                mov edi, __r8d__
                mov __esi__, r9d
                call __calcIndexP2__
                mov [edx+eax], bx       ;mShift[i][j]=mShift[i][k]
                inc __ecx__             ;shifts++
        shiftNumbersLP2_IsZero:
        inc r9d
        cmp r9d, DimMatrix-1            ;j<DimMatrix-1
        jl  shiftNumbersLP2_Cols
    shiftNumbersLP2_Next:
    inc r8d
    cmp r8d, DimMatrix                  ;i<DimMatrix
    jl  shiftNumbersLP2_Rows
    ...
shiftNumbersLP2_End:
    ...
    ret
```

Examen 2015/16-1

Assignatura	Codi	Data	Hora inici
Estructura de computadors	05.573	23/01/2016	15:30

Pregunta 2

2.1

El estado inicial del computador CISCA justo antes de comenzar la ejecución de cada fragmento de código (en cada apartado) es el siguiente:

R1 = 00000008h R5 = 00000028h R10 = 00000050h	M(00007E40h) = 00007E50h M(00000800h) = 00000810h	Z = 0, C = 0, S = 0, V = 0

Completad el estado del computador después de ejecutar cada código (indicad los valores de los registros en hexadecimal).
Suponed que la dirección simbólica V vale 800h.

a)

```
MUL [V], -1
ADD R5, [V]
SAL R5, 1
```

```
[V]= FFFFF7F0h
R5= FFFFF818h
R5= FFFFF030h

R5= FFFFF030h
Z = 0, C = 0 , S = 1  , V = 0
```

b)

```
   MOV R2, [00007E40h]
   SAR  R2, R1
   JE      F
   XOR   R1, R1
F:
```

```
R2= 00007E50h
R2= 0000007Eh
R1= 0

R2= 0000007Eh
R1= 0

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

Examen 2015/16-1

Assignatura	Codi	Data	Hora inici
Estructura de computadors	05.573	23/01/2016	15:30

2.2

Donat el següent codi d'alt nivell:

```
do {
        C= C * A;
        B= B / A;
}
while B > C
```

Es proposa la següent traducció a CISCA on hem deixat 6 espais per omplir.

```
      MOV R0, [A]
DO:   MUL  [C], R0
      DIV  [B], R0
      MOV  R2, [B]
      CMP [C], R2
      JLE  DO
```

Examen 2015/16-1

Assignatura	**Codi**	**Data**	**Hora inici**
Estructura de computadors	05.573	23/01/2016	15:30

2.3

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

```
        JMP PLUS
        INC R0

        MOV [A], 01101111b
PLUS:
```

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 00000600h (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 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
40h	JMP
24h	Inc
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 registro	Si el mode ha d'especificar un registre
0	No s'especifica registre.

Examen 2015/16-1

Assignatura	**Codi**	**Data**	**Hora inici**
Estructura de computadors	05.573	23/01/2016	15:30

		Bk per a k=0..10										
@	**Assemblador**	0	1	2	3	4	5	6	7	8	9	10
00000600h	JMP PLUS	40	00	13	06	00	00					
00000606h	INC R0	24	10									
00000608h	MOV [A], 01101111b	10	20	00	40	00	00	00	6F	00	00	00
00000613h												

Examen 2015/16-1

Assignatura	Codi	Data	Hora inici
Estructura de computadors	05.573	23/01/2016	15:30

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, 5, 24, 3, 18, 55, 6, 17, 18, 32, 40, 4, 6, 63, 40, 48, 56, 42, 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 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		5		24		3	
0	0 (0 - 7)	E	0 (0 - 7)		0 (0 - 7)	E	0 (0 - 7)		0 (0 - 7)	E	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)		2 (16 - 23)		2 (16 - 23)		2 (16 - 23)
3	3 (24 - 31)		3 (24 - 31)		3 (24 - 31)		3 (24 - 31)	E	3 (24 - 31)		3 (24 - 31)

Línia	18		55		6		17		18		32	
0		0 (0 - 7)		0 (0 - 7)	E	0 (0 - 7)		0 (0 - 7)		0 (0 - 7)		0 (0 - 7)
1		1 (8 - 15)	F	6 (48 - 55)		6 (48 - 55)		6 (48 - 55)		6 (48 - 55)		6 (48 - 55)
2	E	2 (16 - 23)		2 (16 - 23)		2 (16 - 23)	E	2 (16 - 23)	E	2 (16 - 23)		2 (16 - 23)
3		3 (24 - 31)		3 (24 - 31)		3 (24 - 31)		7 (56 - 63)		7 (56 - 63)	F	4 (32 - 39)

Examen 2015/16-1

Assignatura	Codi	Data	Hora inici
Estructura de computadors	05.573	23/01/2016	15:30

Línia	40	4	6	63	40	48
0	0 (0 - 7)	E 0 (0 - 7)	E 0 (0 - 7)	0 (0 - 7)	0 (0 - 7)	0 (0 - 7)
1	F 5 (40 - 47)	5 (40 - 47)	5 (40 - 47)	5 (40 - 47)	E 5 (40 - 47)	5 (40 - 47)
2	2 (16 - 23)	2 (16 - 23)	2 (16 - 23)	F 7 (56 - 63)	7 (56 - 63)	7 (56 - 63)
3	4 (32 - 39)	4 (32 - 39)	4 (32 - 39)	4 (32 - 39)	4 (32 - 39)	F 6 (48 - 55)

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

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

T_e = 15 encerts / 20 accessos = 0,75

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 de fallades obtinguda a la pregunta anterior, quin és el temps mitja d'accés a memòria (t_m) ?

$t_m = T_e \times t_e + (1-T_e) \times t_f$ = 0,75 * 5 ns + 0,25 * 40 ns = 3,75 ns + 10 ns = 13,75 ns

Assignatura	Codi	Data	Hora inici
Estructura de computadors	05.573	23/01/2016	15:30

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} = 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

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
```

Assignatura	Codi	Data	Hora inici
Estructura de computadors	05.573	23/01/2016	15:30

3.2.2) 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)

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}

$t_{bloc} = t_{latència} + (N_{dades} * t_{dada})$
$t_{latència} = 0$
$N_{dades} = 100000$
$t_{dada} = m_{dada} / v_{transf}$ = 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 2015/16-1

Assignatura	Codi	Data	Hora inici
Estructura de computadors	05.573	23/01/2016	15:30

Pregunta 4

4.1

Un dels camps importants en el disseny de les instruccions és el “codi d'operació”. El normal és que la grandària sigui fixa. Si es desitja codificar 50 operacions diferents, de quina grandària en bits ha de ser el codi d'operació? Ens quedaria algun codi disponible sense usar? Si és així, indica quants.

Suposant que volem codificar 50 operacions diferents, necessitem 6 bits com a mínim. Amb 6 bits podem codificar 2^6 operacions (64) i per tant tindríem lliures 14 codis per a operacions addicionals.

4.2

a) Quines són les tres polítiques d’assignació per a emmagatzemar dades dins d’una memòria cau? En que consisteixen?

1) Política d'assignació directa: un bloc de la memòria principal només potser 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 memòria 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 memòria principal pot ser en un subconjunt de les línies de la memòria cau, però dins del subconjunt pot trobar-se en qualsevol posició. La memòria cau associativa per conjunts és una combinació

b) En un sistema d’E/S gestionat per DMA. Explica quan i perquè es produeix una interrupció. Serveix per indicar l’inici o el final d’una transferència? Qui la genera?

Finalització de l'operació d'E/S: quan s'ha acabat la transferència del bloc
el controlador de DMA envia una petició d'interrupció al processador per informar que s'ha acabat la transferència de dades.