20202 75573 130621 1 E(Solución)

Estructura de Computadores (Universitat Oberta de Catalunya)

Examen 2020/21-2

Asignatura	**Código**	**Fecha**	**Hora inicio**
Estructura de computadores	75.573	13/6/2021	19:00

Ficha técnica del examen

- Comprueba que el código y el nombre de la asignatura corresponden a la asignatura de la que te has matriculado.
- Tiempo total: **2 horas** Valor de cada pregunta: **Se indica en el enunciado**
- ¿Puede consultarse algún material durante el examen? NO ¿Qué materiales están permitidos? **NINGUNO**
- ¿Puede utilizarse calculadora? NO ¿De qué tipo? NINGUNO
- Si hay preguntas tipo test, ¿descuentan las respuestas erróneas? NO ¿Cuánto?
- Indicaciones específicas para la realización de este examen:

Examen 2020/21-2

Asignatura	Código	Fecha	Hora inicio
Estructura de computadores	75.573	13/6/2021	19:00

Enunciados

Enunciado: El enunciado del examen estará en formato PDF.
En el ordenador desde donde se realizará el examen debéis tener instalado algún software para poder leer documentos en formato PDF. Por ejemplo, se puede utilizar el software gratuito Adobe Acrobat Reader DC, pero podéis utilizar cualquier otro software.

Identificación del estudiante: No es necesario identificarse, de esta forma se garantiza que el examen será tratado de forma anónima.

Respuestas:
Se deberá identificar cada respuesta dentro del examen. Es ***obligatorio indicar el número de pregunta y el apartado***, opcionalmente también se puede añadir todo o parte del enunciado si esto os ayuda en la resolución de la pregunta.
Si no se identifica correctamente a qué pregunta hace referencia la respuesta no se evaluará.

En caso de ser necesario aplicar un procedimiento para resolver alguna pregunta, mostrad claramente y argumentad el procedimiento aplicado, no solo el resultado. En caso de duda, si no se pueden resolver por los mecanismos establecidos o por falta de tiempo, haced los supuestos que consideréis oportunos y argumentadlos.

Elaboración documento a entregar:
Utilizad cualquier editor de texto para crear el documento con las respuestas, siempre que después os permita exportar el documento a formato PDF para hacer la entrega.

Entrega: Es obligatorio entregar las respuestas del examen en un único documento ***en formato PDF***.
No se aceptarán otros formatos.
Es responsabilidad del estudiante que la información que contenga el documento PDF que se entregue refleje correctamente las respuestas dadas en el examen. Recomendamos que abráis el fichero PDF generado y reviséis atentamente las respuestas para evitar que se haya podido perder, cambiar o modificar alguna información al generar el documento en formato PDF.
La entrega se puede hacer tantas veces como se quiera, se corregirá la última entrega que se haga dentro del horario especificado para realizar el examen.

COMPROMISO DE AUTORESPONSABILIDAD: este examen se tiene que resolver de forma individual bajo vuestra responsabilidad y siguiendo las indicaciones de la ficha técnica (sin utilizar ningún material, ni calculadora).
En caso de que no sea así, el examen se evaluará con un cero. Por otro lado, y siempre a criterio de los Estudios, el incumplimiento de este compromiso puede suponer la apertura de un expediente disciplinario con posibles sanciones.

Examen 2020/21-2

Asignatura	**Código**	**Fecha**	**Hora inicio**
Estructura de computadores	75.573	13/6/2021	19:00

No se puede utilizar calculadora. Hay que saber interpretar un valor en binario, decimal o hexadecimal para realizar la operación que se pida. Y el resultado se tiene que expresar en el formato correspondiente.

Valoración de las preguntas del examen

Pregunta 1 (20%)

Pregunta sobre la práctica.
Hay que completar las instrucciones marcadas o añadir el código ensamblador que se pide.
Los puntos suspensivos indican que hay más código, pero no se tiene que completar.
NOTA: Si el código propuesto en cada pregunta no se corresponde con la forma en que vosotros plantearíais la respuesta, podéis reescribir el código o parte del código según vuestro planteamiento.

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

Asignatura	Código	Fecha	Hora inicio
Estructura de computadores	75.573	13/6/2021	19:00

Pregunta 1

1.1 Práctica – 1a Parte

Modificad la subrutina moveCursorP1 par que funcione si en vez de disponer de la variable 'pos' (Posición del cursor dentro de la matriz), tenemos las variables 'row' y 'col', también de tipo int, que indican la fila y la columna del cursor dentro de la matriz, i para hacer el movimiento actualizamos las variables 'row' y 'col'.
; Para cambiar de fila sumamos o restamos 1 a (row) y para cambiar de
; columna sumamos o restamos 1 a (col) porque cada posición de la matriz
; es de tipo char(BYTE)1byte y tiene ROWDIM filas y COLDIM columnas.
(No se tiene que escribir el código de tota la subrutina, sólo hay que modificar (añadir, eliminar o cambiar) el código para hacer lo que se pide).

```
;;;;;
; Actualizar la posición del cursor dentro la matriz actualizando la
; variable(pos), de tipo int(DWORD)4bytes, en función de la tecla
; pulsada que tenemos a la variable (charac) de tipo char(BYTE)1byte,
; (i: arriba, j:izquierda, k:abajo, l:derecha).
; Comprobar que no salimos de la matriz, (pos) sólo pued tomar de
; de posiciones de dentro de la matriz [0 : (ROWDIM*COLDIM)-1].
; Para comprobarlo hay que calcular la fila y columna dentro de la matriz:
; fila    = pos / COLDIM, que puede tomar valores [0 : (ROWDIM-1)].
; columna = pos % COLDIM, que puede tomar valores [0 : (COLDIM-1)].
; Para cambiar de fila sumamos o restamos COLDIM a (pos) y para cambiar de
; columna sumamos o restamos 1 a (pos) porque cada posición de la matriz
; es de tipo char(BYTE)1byte y tiene ROWDIM filas y COLDIM columnas.
; Si el movimiento sale de la matriz, no hacer el movimiento.
; NO se tiene que posicionar el cursor en pantalla llamando posCurScreenP1.
; Variables globales utilizadas:
: (charac): Carácter que leemos del teclado.
; (pos)   : Posición del cursor dentro de la matriz.
;;;;;
moveCursorP1:
   push rbp
   mov  rbp, rsp

   push rax
   push rbx
   push rcx
   push rdx

   mov eax, DWORD[pos]
   mov edx, 0
   mov ebx, COLDIM
   div ebx      ;(eax) i = pos / COLDIM; j(edx) = pos % COLDIM;

   mov  ebx, DWORD[pos]
   mov eax, DWORD[row]
   mov edx, DWORD[col]
   mov  cl , BYTE[charac]
   moveCursorP1_i:
      cmp cl,'i'              ;case 'i':
      jne moveCursorP1_k
      cmp eax, 0              ;i > 0
      jle moveCursorP1_End
      sub ebx, COLDIM         ;pos=pos-COLDIM
```

Examen 2020/21-2

Asignatura	**Código**	**Fecha**	**Hora inicio**
Estructura de computadores	75.573	13/6/2021	19:00

```
      sub eax, 1
      jmp moveCursorP1_End
   moveCursorP1_k:
      cmp cl,'k'              ;case 'k':
      jne moveCursorP1_j
      cmp eax, (ROWDIM-1)     ;i < (ROWDIM-1)
      jge moveCursorP1_End
      add bx, COLDIM          ;pos=pos+COLDIM;
      add eax,1
      jmp moveCursorP1_End
   moveCursorP1_j:
      cmp cl,'j'              ;case 'j':
      jne moveCursorP1_l
      cmp edx, 0              ;j > 0
      jle moveCursorP1_End
      sub cbx, 1              ;pos=pos-1;
      sub edx, 1
      jmp moveCursorP1_End
   moveCursorP1_l:
      cmp cl,'l'              ;case 'l':
      jne moveCursorP1_End
      cmp edx, (COLDIM-1)     ;j < (COLDIM-1)
      jge moveCursorP1_End
      add cbx, 1              ;pos=pos+1;
      add edx,1
   moveCursorP1_End:
   mov  DWORD[pos], cbx
   mov DWORD[row], eax
   mov DWORD[col], edx

   pop rdx
   pop rcx
   pop rbx
   pop rax

   mov rsp, rbp
   pop rbp
   ret
```

Examen 2020/21-2

Asignatura	**Código**	**Fecha**	**Hora inicio**
Estructura de computadores	75.573	13/6/2021	19:00

1.2 Práctica – 2a parte

Haced los cambios necesarios al código ensamblador de esta subrutina considerando que el vector vPos es de tipo short(2 bytes), no se pueden añadir instrucciones, sólo modificar las instrucciones que sea necesario.

```
;;;;;
; Comprovar si les dues targetes obertes són iguals.
; Si les targetes són iguals canviar a l'estat de 'hi ha parella' (status=3).
; Si no són iguals tornar a girar-les. Per a fer-ho cal tornar a posar
; els valors de les targetes que ara tenim a la matriu (mOpenCards),
; a la matriu (mCards) i a la matriu (mOpenCards) posar-hi una 'X'
; (majúscula) per a indicar que estan tapades. Canviar a l'estat
; de 'no hi ha parella' (status=4). Retornar l'estat actualitzat.
; El vector (vPos) de tipus int(DWORD)4bytes, rebut com a paràmetre,
; conté les posicions de les targetes obertes.
; vPos[0]:[vPos+0]: Posició de la 1a targeta.
; vPos[1]:[vPos+4]: Posició de la 2a targeta.
; Variables globals utilitzades:
; (mCards)    : Matriu on guardem les targetes del joc.
; (mOpenCards): Matriu on tenim les targetes obertes del joc.
; Paràmetres d'entrada:
; (vPos)  : rdi(rdi): Adreça del vector amb les posicions de les targetes obertes.
; Paràmetres de sortida:
; (status): rax(eax): Estat del joc.
;;;;;
checkPairsP2:
   push rbp
   mov  rbp, rsp
   push rbx
   push rcx
   push rdi

   mov rbx, 0
   mov rcx, 0
   mov bx, WORD[rdi+0]         ;vPos[0]
   mov cx, WORD[rdi+2]         ;vPos[1]

   mov al, BYTE[mOpenCards+rbx] ;mOpenCards[i0][j0]
   cmp al, BYTE[mOpenCards+rcx] ;if ( mOpenCards[i0][j0] == mOpenCards[i1][j1] )
   jne checkPairsP2_Else
      mov eax, 3               ;status = 3; //Hi ha parella
      jmp checkPairs_End
   checkPairsP2_Else:
      mov BYTE[mCards+rbx], al      ;mCards[i0][j0]   = mOpenCards[i0][j0];
      mov BYTE[mOpenCards+rbx], 'X' ;mOpenCards[i0][j0] = 'X';
      mov al, BYTE[mOpenCards+rcx]
      mov BYTE[mCards+rcx], al      ;mCards[i1][j1]   = mOpenCards[i1][j1];
      mov BYTE[mOpenCards+rcx], 'X' ;mOpenCards[i1][j1] = 'X';
      mov eax, 4              ;status = 4; //No hi ha parella

   checkPairs_End:
   pop rdi
   pop rcx
   pop rbx
   mov rsp, rbp
   pop rbp
   ret
```

Examen 2020/21-2

Asignatura	Código	Fecha	Hora inicio
Estructura de computadores	75.573	13/6/2021	19:00

Pregunta 2

2.1

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

R0= 100h R1= 200h R2= 300h R3= 400h	M(00000100h)=00000100h M(00000200h)=FFFFF600h M(00000300h)=00000600h M(00000400h)=0000FFFFh M(00000500h)=60000000h	Z = 0, C = 0, S = 0, V = 0

Completad el estado del computador (registros y bits de estado) después de ejecutar cada código (indicad los valores de los registros en hexadecimal).

Suponed que la dirección simbólica A vale 100h.

a)

```
MOV R2 , FFFFFFF0h
ADD R2 , [A+ R3]
```

```
R2 := FFFFFFF0h
R2 :=
FFFFFFF0h+[00000500h]=
FFFFFFF0h+60000000h =
5FFFFFF0h

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

b)

```
MOV R2, [A+100h]
SUB R2, R3
NOT  R2
```

```
R2 = [200]= FFFFF600h
R2 = FFFFF200h
R2 = 00000DFFh

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

Examen 2020/21-2

Asignatura	**Código**	**Fecha**	**Hora inicio**
Estructura de computadores	75.573	13/6/2021	19:00

2.2

Dado el siguiente código de alto nivel:

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

A es un vector de 8 elementos de 4 bytes cada uno. Se propone la siguiente traducción a CISCA dónde hemos dejado 8 espacios para llenar.

```
        MOV R0, [j]
        MUL R0, 4
        MOV R2, [A+R0]
        MOV R1, [i]
        MUL R1, 4
        MOV R3, [A+R1]
        SAL R3, 1
        CMP R2, R3
        JLE END
        ADD R2,[A+R1]
        MOV [A+R1], R2
END:
```

Examen 2020/21-2

Asignatura	Código	Fecha	Hora inicio
Estructura de computadores	75.573	13/6/2021	19:00

2.3

Traducid a lenguaje máquina el fragmento de código en lenguaje ensamblador que os proponemos a la tabla.

Suponed que la primera instrucción del código se ensambla a partir de la dirección **00023C00h** (que es el valor del PC antes de empezar la ejecución del fragmento de código). Lo etiqueta B representa la dirección **00880000h**.

A continuación os damos como ayuda los tablas de códigos:

Tabla de códigos de instrucción

B0	Instrucción
26h	CMP
43h	JE
20h	ADD
35h	SAL

Tabla de modos de direccionamiento (Bk<7..4>)

Campo modo Bk<7..4>	Modo
0h	Inmediato
1h	Registro
2h	Memoria
3h	Indirecto
4h	Relativo
5h	Indexado
6h	Relativo a PC

Tabla de modos de direccionamiento (Bk<3..0>)

Campo modo Bk<3..0>	Significado
Num. registro	Si el modo tiene que especificar un registro
0	No se especifica registro.

			Bk por k=0..10										
Dirección	**Eti**	**Ensamblador**	0	1	2	3	4	5	6	7	8	9	10
00023C00h		CMP R1, 10h	26	11	00	10	00	00	00				
00023C07h		JE E1	43	60	12	00							
00023C0Bh		ADD [B+R10], 200h	20	5A	00	00	88	00	00	00	02	00	00
00023C16h		SAL [R10], 4h	35	3A	00	04	00	00	00				
00023C1Dh	E1:												

Examen 2020/21-2

Asignatura	Código	Fecha	Hora inicio
Estructura de computadores	75.573	13/6/2021	19:00

Pregunta 3

3.1. Memoria cache

Memoria cache de acceso completamente asociativo
Tenemos un sistema de memoria en el que todos los accesos se realizan a palabra (no nos importa cuál es el tamaño de la palabra). Supondremos que el espacio de direcciones de memoria se descompone en bloques de 8 palabras. Cada bloque empieza en una dirección múltiplo de 8. Así, el bloque 0 contiene las direcciones 0, 1, 2, 3, 4, 5, 6, 7, el bloque 1, las direcciones 8, 9, 10, 11, 12, 13, 14, 15, y el bloque N las direcciones 8*N, 8*N+1, 8*N+2, 8*N+3, 8*N+4, 8*N+5, 8*N+6, 8*N+7.

Suponemos que el sistema también dispone de una memoria cache de 4 líneas (donde cada línea tiene el tamaño de un bloque, es decir, 8 palabras). Estas líneas se identifican como líneas 0, 1, 2 y 3. Cuando se hace una referencia a una dirección de memoria principal, si esta dirección no se encuentra en la memoria cache, se trae todo el bloque correspondiente desde la memoria principal a una línea de la memoria cache (así si hacemos referencia a la dirección 2 de memoria principal traeremos el bloque formado por las palabras 0, 1, 2, 3, 4, 5, 6, 7).

Suponemos que el sistema utiliza una política de **emplazamiento completamente asociativa**, de manera que cualquier bloque de la memoria principal se puede llevar a cualquier bloque de la memoria cache.

Si encontramos que la cache ya está llena, se utiliza un **algoritmo de reemplazamiento LRU**, de manera que sacaremos de la memoria cache aquel bloque que hace más tiempo que no se referencia.

La ejecución de un programa genera la siguiente lista de lecturas a memoria:

7, 8, 9, 12, 4, 24, 18, 29, 15, 45, 51, 13, 73, 14, 52, 42, 28, 58, 20, 21

3.1.1

La siguiente tabla muestra el estado inicial de la cache, que contiene las primeras 32 palabras de la memoria (organizadas en 4 bloques).

Completar la tabla para mostrar la evolución de la cache durante la ejecución del programa. Para cada fallo hay que llenar una columna indicando el nuevo bloque que se traerá a la memoria cache a la línea que le corresponda, expresado de la forma b (a_0 – a_7), donde b: número de bloque, y (a_0 - a_7) son las direcciones del bloque, donde a_0 es la primera dirección y a_7 es la octava (última) dirección del bloque.

Línea	Estado Inicial	Fallo: 45	Fallo: 51	Fallo: 73	Fallo: 28	Fallo: 58
0	0 (0 - 7)	5 (40 – 47)				
1	1 (8 - 15)					7 (56 – 63)
2	2 (16 - 23)		6 (48 – 55)			
3	3 (24 - 31)			9 (72 – 79)	3 (24 – 31)	

Examen 2020/21-2

Asignatura	Código	Fecha	Hora inicio
Estructura de computadores	75.573	13/6/2021	19:00

Línea	Fallo: 20	Fallo:	Fallo:	Fallo:	Fallo:	Fallo:
0						
1						
2	2 (16 - 23)					
3						

Línea	Fallo:	Fallo:	Fallo:	Fallo:	Fallo:	Fallo:
0						
1						
2						
3						

3.1.2 a)
¿Cuál es la tasa de aciertos (T_e)?

T_e = 14 aciertos / 20 accesos = 0,7

3.1.2 b)
Suponemos que el tiempo de acceso a la memoria cache, o tiempo de acceso en caso de acierto (t_e), es de 5 ns y el tiempo total de acceso en caso de fallo (t_f) es de 25 ns. Considerando la tasa de aciertos obtenida en la pregunta anterior, ¿cuál es el tiempo medio de acceso a memoria (t_m)?

$t_m = T_e \times t_e + (1-T_e) \times t_f = 0,7 * 5 \text{ ns} + 0,3 * 25 \text{ ns} = 3,5 \text{ ns} + 7,5 \text{ ns} = 11 \text{ ns}$

Examen 2020/21-2

Asignatura	Código	Fecha	Hora inicio
Estructura de computadores	75.573	13/6/2021	19:00

3.2 Sistema de E/S

E/S por interrupciones
Se quiere analizar el rendimiento de la comunicación de datos entre la memoria de un procesador y un puerto USB, utilizando E/S por interrupciones, con las siguientes características:

- Velocidad de transferencia del dispositivo de E/S v_{transf} = 10 MBytes/s = 10.000 Kbytes/s.
- Tiempo de latencia medio del dispositivo $t_{latencia}$ = 0.
- Direcciones de los **registros de estado** y **datos** del controlador de E/S: 0B00h y 0B04h.
- El bit del **registro de estado** que indica que el controlador del puerto de E/S está disponible es el bit 4, o el quinto bit menos significativo (cuando vale 1 indica que está disponible).
- Procesador con una frecuencia de reloj de 2 GHz, el tiempo de ciclo t_{ciclo} = 0,5 ns. El procesador puede ejecutar 1 instrucción por cada 2 ciclos de reloj.
- Transferencia de **lectura** desde el puerto de E/S a la memoria.
- Transferencia de N_{datos}= 400.000 datos.
- El tamaño de un dato es m_{dato} = 4 bytes
- Dirección inicial de memoria donde residen los datos: A0000000h.
- El tiempo para atender a la interrupción (t_{rec_int}) es de 2 ciclos de reloj.

3.2.1

Completar el siguiente código CISCA que es una rutina de servicio a las interrupciones (RSI) para transferir datos desde el dispositivo de E/S anterior, mediante la técnica de E/S por interrupciones.

Se utiliza una variable global que se representa con la etiqueta **Addr,** y que al principio del programa contiene la dirección inicial de memoria donde se almacenan los datos recibidos.

```
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 2020/21-2

Asignatura	Código	Fecha	Hora inicio
Estructura de computadores	75.573	13/6/2021	19:00

3.2.2

¿Cuánto tiempo dedica la CPU a la transferencia del bloque de datos t_{transf_bloque}?

El tiempo de un ciclo, t_{ciclo} = 0,5 ns (nanosegundos)

Tiempo para atender la interrupción, t_{rec_int}: 2 ciclos * 0,5 ns = 1 ns

Tiempo de ejecución de una instrucción, t_{instr}: t_{ciclo} *2= 1 ns

Tiempo de ejecución RSI, t_{rsi}: N_{rsi} * t_{instr} = 12 instr. * 1 ns = 12 ns

Tiempo consumido por la CPU en cada interrupción, t_{transf_dato}:

t_{transf_dato} = t_{rec_int} + t_{rsi} = 1 + 12 = 13 ns

Número de interrupciones producidas (número total de datos, N_{datos}): 400.000 interrupciones

Tiempo consumido en total en TODAS las interrupciones:

t_{transf_bloque} = t_{transf_dato} * N_{datos} = 13 ns * 400.000 interrupciones = 5.200.000 ns = 5,2 ms (milisegundos)

3.2.3

¿Cuál es el porcentaje de ocupación del procesador? Porcentaje que representa el tiempo de transferencia del bloque t_{transf_bloque} respecto al tiempo de transferencia del bloque por parte del periférico t_{bloque}

t_{bloque} = $t_{latencia}$ + (N_{datos} * t_{dato})
$t_{latencia}$ = 0
N_{datos} = 400.000
t_{dato} = m_{dato} / v_{transf} = 4 / 1000 Kbytes/s = 0,004 ms
t_{bloque} = 0 + (400.000 * 0,004) ms = 160 ms
% ocupación = (t_{transf_bloque} * 100 / t_{bloque}) = (5,2 * 100) / 160 = 3,25%