20202 75573 090621 1 E(Solución)

Estructura de Computadores (Universitat Oberta de Catalunya)

Examen 2020/21-2

Asignatura	**Código**	**Fecha**	**Hora inicio**
Estructura de computadores	75.573	9/6/2021	12:30

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	9/6/2021	12:30

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	9/6/2021	12:30

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	9/6/2021	12:30

Pregunta 1

1.1 Práctica – 1a Parte

Modificad la subrutina openCardP1 para que verifique si la segunda tarjeta que abrimos, una vez girada, es igual a la que hemos abierto primero, si son iguales decrementar la variable 'pairs' (medida 2 bytes) que indica las parejas que quedan por hacer.
Después de girar la tarjeta y incrementar 'state', comprovamos que estamos abriendo la segunda tarjeta mirando si 'state' vale 2. A continuación comparamos la primera tarjeta que hemos abierto, que está guardada en la matriz mOpenCard en la posición indicada por el elemento 0 del vector 'vPos' con la segunda tarjeta abierta, si son iguales decrementamos 'pairs'. (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).

```
;;;;;
; Abrir la tarjeta de la matriz (mCards) de la posición indicada por el
; cursor dentro de la matriz indicada por la variable (pos).
; Guardar la posición de la tarjeta que estamos abriendo y que tenemos
; en la variable (pos) de tipo int(DWORD)4bytes en el vector (vPos), de
; tipo int(DWORD)4bytes, donde en la posición [0] es para guardar la
; posición de la 1a tarjeta que giramos (cuando state=0) y la posición
; [1] es para guardar la posición de la 2a tarjeta que giramos (cuando
; state=1).
; vPos[0]:[vPos+0]: Posición de la 1a tarjeta.
; vPos[1]:[vPos+4]: Posición de la 2a tarjeta.
; Para acceder a la matriz en C hay que calcular la fila y la columna:
; fila    = pos / COLDIM, que pot prendre valors [0 : (ROWDIM-1)].
; columna = pos % COLDIM, que pot prendre valors [0 : (COLDIM-1)].
; En ensamblador no es necesario.
; Si la targeta no está girada (!='x') ponerla en la matriz
; (mOpenCards) para que se muestre.
; Marcarla con una 'x'(minúscula) en la misma posición de la matriz
; (mCards) para saber que está volteada.
; Pasar al siguiente estado (state++).
; NO se tiene que mostrar la matriz con los cambios, es fa a updateBoardP1.
; Variables globales utilizadas:
; (mCards)    : Matriz donde guardamos las tarjetas del juego.
; (mOpenCards): Matriz donde tenemos las tarjetas abiertas del juego.
; (pos)       : Posición del cursor dentro de la matriz.
; (state)     : Estado del juego.
; (vPos)      : Dirección del vector con lass posiciones de las tarjetas abiertas.
;;;;;
openCardP1:
   push rbp
   mov  rbp, rsp
   push rbx
   push rdx
   push rdi

   mov ebx, DWORD[pos]
   mov edi, DWORD[state]
   shl edi, 2                      ;state*4
   mov DWORD[vPos+edi], ebx        ;vPos[state] = pos;

   mov dl, BYTE[mCards+ebx]
   cmp dl, 'x'                     ;if (mCards[i][j] != 'x') {
```

Examen 2020/21-2

Asignatura	**Código**	**Fecha**	**Hora inicio**
Estructura de computadores	75.573	9/6/2021	12:30

```
je  openCardP1_End
   mov BYTE[mOpenCards+ebx], dl ;mOpenCards[i][j] = mCards[i][j];
   mov BYTE[mCards+ebx], 'x'    ;mCards[i][j] = 'x';
   inc DWORD[state]             ;state++;

   cmp DWORD[State],2
   jl  openCardP1_End
      mov ecx, DWORD[vPos+0]
      cmp dl,mOpenCard[ecx]
      jne openCardP1_End
         dec WORD[pairs]

openCardP1_End:
pop rdi
pop rdx
pop rbx
mov rsp, rbp
pop rbp
ret
```

Examen 2020/21-2

Asignatura	**Código**	**Fecha**	**Hora inicio**
Estructura de computadores	75.573	9/6/2021	12:30

1.2 Práctica – 2a parte

Haced los cambios necesarios al código ensamblador de esta subrutina considerando que la variable pos se ha declarado de tipo short(2 bytes), no se pueden añadir instrucciones, sólo modificar las instrucciones que sea necesario.

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; Esta subrutina se da hecha. NO LA PODEIS MODIFICAR.
; Situar el cursor en una fila y una columna de la pantalla
; en función de la fila (edi) y de la columna (esi) recibidos como
; parámetro llamando a la función gotoxyP2_C.
; Variables globales utilizadas: Ninguna
; Parámetros de entrada: (rowScreen): rdi(edi): Fila
;                        (colScreen): rsi(esi): Columna;
; Parámetros de salida :  Ninguno
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
gotoxyP2:

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; Posicionar el cursor en pantalla, dentro del tablero, en función de
; la posición del cursor dentro de la matriz, indicada por la variable
; (pos), recibida como parámetro, de tipo int(DWORD)4bytes, a partir
; de la posición [10,12] de la pantalla.
; Para calcular la posición del cursor en pantalla (rowScreen) y
; (colScreen) utilizar estas fórmulas:
; rScreen=10+(pos/COLDIM)*2)
; cScreen=12+(pos%COLDIM)*4)
; Para posicionar el cursor en pantalla se tiene que llamar a la
; subrutina gotoxyP2.
; Variables globales utilizadas: Ninguna.
; Parámetros de entrada: (pos): rdi(edi): Posición cursor dentro la matriz.
; Parámetros de salida : Ninguno.
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
posCurScreenP2:   push rbp
   mov  rbp, rsp
   ...
   mov rax, 0
   mov rdx, 0
   mov ax, di
   mov bx, COLDIM
   div bx          ;eax=pos/COLDIM, edx:pos%COLDIM

   shl eax, 1       ;((pos/COLDIM)*2)
   add eax, 10      ;rScreen=10+((pos/COLDIM)*2);
   shl edx, 2       ;((pos/COLDIM)*4)
   add edx, 12      ;cScreen=12+((pos%COLDIM)*4);

   mov edi, eax
   mov esi, edx
   call gotoxyP2    ;gotoxyP2_C(rScreen, cScreen);

   posCurScreenP2_End:
   ...
   mov rsp, rbp
   pop rbp
   ret
```

Examen 2020/21-2

Asignatura	Código	Fecha	Hora inicio
Estructura de computadores	75.573	9/6/2021	12:30

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:

R1 = 00000100h R2 = 00000200h R3 = 00000300h	M(00000100h) = 00000300h M(00000200h) = 00000200h M(00000300h) = 00000100h M(00000400h) = 00000500h	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 400h.

a)

```
MOV R1,[R1]
XOR R1, R2
SUB R1,[R2]
```

R1 = 00000300h
R1 = 00000100h
R1 = FFFFFF00h

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

b)

```
ADD R2, [A]
SAL R2, 14h
```

R2 = 00000700h
R2 = 70000000h

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

Examen 2020/21-2

Asignatura	**Código**	**Fecha**	**Hora inicio**
Estructura de computadores	75.573	9/6/2021	12:30

2.2

Suponemos que tenemos el vector V de 5 elementos de 32 bits. Completad la traducción del programa en ensamblador CISCA para que ejecute el algoritmo de alto nivel mostrado. (Hemos dejado 8 espacios para llenar)

```
A = 16;
for (i = 4; i >= 0; i--) {
        V[i]= A + i;
        A = A * 2;
}
```

El registro R1 representa la variable **A** y R0 representa la variable i

```
        MOV [A], 10h
        MOV R1,[A]
        MOV R0, 4
        MOV R2, 16
COMP:CMP R0, 0
        JL  END
        MOV [V+R2], R1
        ADD [V+R2], R0
        SAL R1, 1
        SUB R0, 1
        SUB R2, 4
        JMP  COMP
END: MOV [A], R1
        MOV [i], R0
```

Examen 2020/21-2

Asignatura	**Código**	**Fecha**	**Hora inicio**
Estructura de computadores	75.573	9/6/2021	12:30

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 **00006880h** (que es el valor del PC antes de empezar la ejecución del fragmento de código). El etiqueta A representa la dirección **0000FFF4h**.

A continuación os damos como ayuda las tablas de códigos:

Tabla de códigos de instrucción

B0	Instrucción
10h	MOV
21h	SUB
26h	CMP
42h	JNE
40h	JMP

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>)

Camp 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
00006880h	E1:	MOV R10, [A+R4]	10	1A	54	F4	FF	00	00				
00006887h		SUB [R10], 10h	21	3A	00	10	00	00	00				
0000688Eh		CMP R10, R4	26	1A	14								
00006891h		JNE E2	42	60	06	00							
00006895h		JMP E1	40	00	80	68	00	00					
0000689Bh	E2:												

Examen 2020/21-2

Asignatura	Código	Fecha	Hora inicio
Estructura de computadores	75.573	9/6/2021	12:30

Pregunta 3

3.1. Memoria cache

Memoria cache de asignación directa

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 asignación directa**, de manera que cada bloque de la memoria principal sólo se puede traer a una línea determinada de la memoria cache.

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

7, 8, 9, 12, 4, 20, 18, 29, 5, 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. Indicar únicamente aquellos accesos que provocan un fallo de cache. Para cada fallo rellenar una columna indicando el nuevo bloque que se trae a la memoria cache en la línea que le corresponda, expresado de la forma b:e (a_0 - a_7) donde b: número de bloque, e:etiqueta y (a_0 - a_7) son las direcciones del bloque, siendo a_0 la primera dirección del bloque y a_7 la octava (última) dirección del bloque.

Línea	Estado Inicial	Fallo: 45	Fallo: 51	Fallo: 13	Fallo: 73	Fallo: 14
0	0:0 (0 - 7)					
1	1:0 (8 - 15)	5:1 (40 - 47)		1:0 (8 - 15)	9:2 (72 - 79)	1:0 (8 - 15)
2	2:0 (16 - 23)		6:1 (48 - 55)			
3	3:0 (24 - 31)					

Examen 2020/21-2

Asignatura	Código	Fecha	Hora inicio
Estructura de computadores	75.573	9/6/2021	12:30

Línea	Fallo: 42	Fallo: 58	Fallo: 20	Fallo:	Fallo:	Fallo:
0						
1	5:1 (40 - 47)					
2			2:0 (16 - 23)			
3		7:1 (56 - 63)				

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 = 12 aciertos / 20 accesos = 0,6

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)?

$t_m = T_e \times t_e + (1-T_e) \times t_f = 0,6 * 5 \text{ ns} + 0,4 * 25 \text{ ns} = 3 \text{ ns} + 10 \text{ ns} = 13 \text{ ns}$

Examen 2020/21-2

Asignatura	Código	Fecha	Hora inicio
Estructura de computadores	75.573	9/6/2021	12:30

3.2 Sistema de E/S

E/S programada

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

- Velocidad de transferencia del dispositivo de E/S v_{transf} = 40 MBytes/s = 40.000 Kbytes/s.
- Tiempo de latencia medio del dispositivo $t_{latencia}$ = 0.
- Direcciones de los **registros de datos** y **estado** del controlador de E/S: 0F00h y 0F04h.
- 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 ciclo de reloj.
- Transferencia de **escritura** desde memoria al puerto de E/S.
- Transferencia de N_{datos}= 800.000 datos.
- El tamaño de un dato es m_{dato} = 4 bytes.
- Dirección inicial de memoria donde residen los datos: 40000000h.

3.2.1

El siguiente código realizado con el repertorio de instrucciones CISCA realiza la transferencia descrita antes mediante la técnica de E/S programada. Completar el código.

```
1.        MOV     R3, 800000
2.        MOV     R2, 40000000h
3.Bucle:  _IN_    R0, [0F04h]       ; leer 4 bytes
4.        AND     R0, _00010000b_
5.        _JE_    Bucle
6.        MOV     R0,_[R2]_         ; leer 4 bytes
7.        ADD     _R2_, 4
8.        OUT     _[0F00h]_, R0     ; escribir 4 bytes
9.        SUB     R3, __1__
10.       _JNE_   Bucle
```

Asignatura	Código	Fecha	Hora inicio
Estructura de computadores	75.573	9/6/2021	12:30

3.2.2

¿Cuánto tiempo dura la transferencia del bloque de datos t_{transf_bloque}?

$t_{transf\_bloque} = t_{latència} + (N_{datos} * t_{transf\_dato})$
$t_{latencia} = 0$
$N_{datos} = 800000$
$t_{transf\_dato} = m_{dato} / v_{transf} = 4$ Bytes / 40000 Kbytes/s = 0,0001 ms = 0,1 us
$t_{transf\_bloque} = 0 + (800000 * 0{,}0001$ ms) = 80 ms

3.2.3

Si quisiéramos utilizar el mismo procesador y el mismo programa, pero con un dispositivo más rápido de E/S, ¿Cuál es la tasa o velocidad máxima de transferencia del nuevo dispositivo que se podría soportar sin que el dispositivo tuviera que esperar?

$t_{ciclo} = 0{,}5$ ns (nanosegundos)

$t_{instr} = 0{,}5$ns

El mínimo número de instrucciones que ha de ejecutar el programa para cada dato transferido son las 8 instrucciones: 3, 4, 5, 6, 7, 8, 9 y 10. Ejecutar las 8 instrucciones requiere $8 * t_{instr} = 8 * 0{,}5$ ns = 4 ns

Por tanto, el tiempo mínimo para transferir un dato es: 4 ns

Se pueden transferir 4 bytes cada 4 ns, es decir: $4 / 4 * 10^{-9} =$ 1000 Mbyte/s = 1 Gbytes/s