20202 75573 190621 1 E(Solución)

Estructura de Computadores (Universitat Oberta de Catalunya)

Examen 2020/21-2

Asignatura	**Código**	**Fecha**	**Hora inicio**
Estructura de computadores	75.573	19/6/2021	16:00

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**
- **asdf**
- ¿Puede consultarse algún material durante el examen? ¿Qué materiales están permitidos? **NINGUNO**
- ¿Puede utilizarse calculadora? ¿De qué tipo?
- Si hay preguntas tipo test, ¿descuentan las respuestas erróneas? ¿Cuánto?
- Indicaciones específicas para la realización de este examen:

Examen 2020/21-2

Asignatura	Código	Fecha	Hora inicio
Estructura de computadores	75.573	19/6/2021	16:00

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	19/6/2021	16:00

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	19/6/2021	16:00

Pregunta 1

1.1 Práctica – 1a Parte

Modificad la subrutina showDigitsP1 para que compruebe que el valor que queremos mostrar está entre 0 y 99 (tiene dos dígitos). Si es más pequeño que 0 poner el valor a mostrar a 0, si es más grande que 99 poner el valor a mostrar a 99. (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).

```
;;;;;
; Convierte un valor (value) de tipo int(4 bytes) (entre 0 y 99) en
; dos caracteres ASCII que representan este valor. (27 -> '2' '7').
; Se tiene que dividir el valor entre 10, el cociente representará las
; decenas y el resto las unidades, y después se tiene que convertir a ASCII
; sumando '0' o 48(código ASCII de '0') a las unidades y a les decenas.
; Muestra los dígitos (carácter ASCII) a partir de la fila indicada
; para la variable (rowScreen) y a la columna indicada para la variable
; (colScreen).
; Para posicionar el cursor se llama a la función gotoxyP1 y para
; mostrar los caracteres ala función printchP1.
; Variables globales utilizadas:
; (rowScreen): Fila de la pantalla donde posicionamos el cursor.
; (colScreen): Columna de la pantalla donde posicionamos el cursor.
; (charac)   : Carácter que leemos de teclado.
; (value)    : Valor que queremos mostrar.
;;;;;
showDigitsP1:
   push rbp
   mov  rbp, rsp
   ...
   mov rax, 0
   mov rdx, 0
   mov ax, WORD[value] ;Valor que queremos mostrar
   cmp ax, 0
   jl  showDigits_es zero
   cmp ax, 99
   jle  showDigits_div
   mov ax, 99
   jmp showDigits_div
   showDigits_es_zero:
   mov ax, 0
   showDigits_div:
   mov bx,10
   div bx           ;AX=DX:AX/BX, DX=DX:EAX mod BX
                    ;eax = d = value / 10; edx = u = value % 10;
   add al,'0'       ;d = d + '0';
   add dl,'0'       ;u = u + '0';
   call gotoxyP1              ;gotoxyP1_C();
   mov  BYTE[charac], al
   call printchP1             ;printchP1_C();
   inc  DWORD[colScreen]      ;colScreen++;
   call gotoxyP1              ;gotoxyP1_C();
   mov  BYTE[charac], dl
   call printchP1             ;printchP1_C();

   showDigitsP1_End:
   ...
   mov rsp, rbp
   pop rbp
   ret
```

Examen 2020/21-2

Asignatura	**Código**	**Fecha**	**Hora inicio**
Estructura de computadores	75.573	19/6/2021	16:00

1.2 Práctica – 2a parte

Haced los cambios necesarios al código ensamblador de esta subrutina considerando que las variables moves y pairs se han declarado de tipo long int (8 bytes), no se pueden añadir instrucciones, sólo modificar las instrucciones que sea necesario.

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; Mostrar los valores de la matriz (mOpenCards) dentro del tablero,
; en las posiciones correspondientes, los intentos que quedan (moves)
; y las parejas que faltan por hacer (pairs), recibidos como parámetro.
; Se tiene que recorrer toda la matriz (mOpenCards), de izquierda a
; derecha y de arriba a bajo, cada posición es de tipo char(BYTE)1byte,
; y para cada elemento de la matriz hacer:
; Posicionar el cursor en el tablero en función de las variables
; (rScreen) fila y (cScreen) columna llamando a la subrutina gotoxyP2.
; Les variables (rScreen) y (cScreen) se inicializan a 10 y 14
; respectivamente, que es la posición en pantalla de la casilla [0][0].
; Mostrar los caracteres de cada posición de la matriz (mOpenCards)
; llamando a la subrutina printchP2.
; Después, mostrar los intentos que quedan (moves) de tipo short(WORD)2bytes,
; a partir de la posición [19,15] de la pantalla y mostrar les parejas
; que faltan por hacer (pairs) de tipo short(WORD)2bytes, a partir de la
; posición [19,24] de la pantalla llamando a la subrutina showDigitsP2.
; Variables globales utilizadas: (mOpenCards): Matriz donde guardamos las tarjetas.
; Parámetros de entrada: (moves): rdi(di): Intentos que quedan.
;                        (pairs): rsi(si): Parejas que faltan por hacer.
; Parámetros de salida : Ninguno.
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
updateBoardP2:
   push rbp
   mov  rbp, rsp
   ...
   mov rdx, rdi                         ;edx: moves
   mov rbx, rsi                         ;ebx: pairs
   mov rcx, 0                           ;ecx: indexMat
   mov edi, 10                          ;rScreen=10;
   mov r10d, 0                          ;i=0
   updateBoardP2_bucle_Row:
   cmp r10d, ROWDIM                     ;i<ROWDIM
   jge updateBoardP2_show
      mov esi, 12                       ;cScreen=12;
      mov r11d, 0                       ;j=0;
      updateBoardP2_bucle_Col:
      cmp r11d, COLDIM                  ;j<COLDIM
      jge updateBoardP2_Col_end
         call gotoxyP2                  ;gotoxyP2_C(rScreen, cScreen);
         push rdi
         mov  dil, BYTE[mOpenCards+rcx];c = mOpenCards[i][j];
         call printchP2                 ;printchP2_C(c);
         pop  rdi
         inc  rcx
         inc  r11d                      ;j++;
         add  esi, 4                    ;cScreen = cScreen + 4;
      jmp updateBoardP2_bucle_Col

      updateBoardP2_Col_end:
      inc  r10d                         ;i++;
      add  edi, 2                       ;rScreen = rScreen + 2;
   jmp updateBoardP2_bucle_Row

   updateBoardP2_show:
   mov edi, 19
   mov esi, 15
   mov rdx, rdx
```

Examen 2020/21-2

Asignatura	**Código**	**Fecha**	**Hora inicio**
Estructura de computadores	75.573	19/6/2021	16:00

```
call showDigitsP2                   ;showDigitsP2_C(moves, 19, 15);
mov esi, 24
mov rdx, rbx
call showDigitsP2                   ;showDigitsP2_C(pairs, 19, 24);

updateBoardP2_End:
...
mov rsp, rbp
pop rbp
ret
```

Examen 2020/21-2

Asignatura	Código	Fecha	Hora inicio
Estructura de computadores	75.573	19/6/2021	16:00

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 = 00000A10h R1 = 00000B20h R2 = 00000C30h R3 = 00000D40h R4= 00000E50h	M(00000A10h) = 0000F00Fh M(00000B20h) = 0000F000h M(00000C30h) = 00000001h M(00000D40h) = 11111111h M(00000E50h) = 77777777h	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).

a)	b)
SUB R0, R1 XOR R1, [R1]	MOV R1, [R4] SAL R1, [R2]
R0= FFFFFEF0 R1= 0000FB20h Z = 0 , S = 0 , C = 0 , V = 0	R1= 77777777h R1= EEEEEEEEh Z = 0 , S = 1 , C = 0 , V = 1

Examen 2020/21-2

Asignatura	**Código**	**Fecha**	**Hora inicio**
Estructura de computadores	75.573	19/6/2021	16:00

2.2

Dado el siguiente código de alto nivel:

```
i = 10;
while (i>=MIN) {
A[i] = i+5;
i = i - 1;
};
```

Se propone la siguiente traducción a CISCA dónde hemos dejado 8 espacios para completar.

```
        MOV R0, Ah
        MOV R1, R0
        MUL R1, 4
PLUS:   CMP R0,[MIN]
        JL END
        MOV [A+R1], R0
        ADD [A+R1], 5
        SUB R1, 4
        DEC R0
        JMP PLUS
END:    MOV [i], R0
```

Examen 2020/21-2

Asignatura	Código	Fecha	Hora inicio
Estructura de computadores	75.573	19/6/2021	16:00

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 **00044A00h** (que es el valor del PC antes de empezar la ejecución del fragmento de código). Lo etiqueta A representa la dirección **00000400h**.

A continuación os damos como ayuda las tablas de códigos:

Tabla de códigos de instrucción

B0	Instrucción
10h	MOV
26h	SUB
41h	JE
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
00044A00h	E1:	SUB R6,[A+R4]	26	16	54	00	04	00	00				
00044A07h		MOV R3,[100h]	10	13	20	00	01	00	00				
00044A0Eh		JE PLUS	41	60	EE	FF							
00044A12h		SAL [R6], 2h	35	36	00	02	00	00	00				
00044A19h													

Examen 2020/21-2

Asignatura	Código	Fecha	Hora inicio
Estructura de computadores	75.573	19/6/2021	16:00

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, 5, 24, 3, 18, 55, 6, 17, 18, 32, 40, 4, 6, 63, 40, 18, 53, 42, 50

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: 55	Fallo: 17	Fallo: 32	Fallo: 40	Fallo: 4
0	0:0 (0 - 7)			4:1 (32 - 39)		0:0 (0 – 7)
1	1:0 (8 - 15)				5:1 (40 - 47)	
2	2:0 (16 - 23)	6:1 (48 - 55)	2:0 (16 - 23)			
3	3:0 (24 - 31)					

Examen 2020/21-2

Asignatura	Código	Fecha	Hora inicio
Estructura de computadores	75.573	19/6/2021	16:00

Línea	Fallo: 63	Fallo: 53	Fallo:	Fallo:	Fallo:	Fallo:
0						
1						
2		6:1 (48 - 55)				
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 = 13 aciertos / 20 accesos = 0,65

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)?

$t_m = T_e \times t_e + (1-T_e) \times t_f$ = 0,65 * 5 ns + 0,35 * 25 ns = 3,25 ns + 8,75 ns = 12 ns

Examen 2020/21-2

Asignatura	Código	Fecha	Hora inicio
Estructura de computadores	75.573	19/6/2021	16:00

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} = 2 MBytes/s = 2.000 Kbytes/s.
- Tiempo de latencia medio del dispositivo $t_{latencia}$ = 0.
- Direcciones de los **registros de estado** y **de datos** del controlador de E/S: 0200h y 0204h.
- El bit del **registro de estado** que indica que el controlador del puerto de E/S está disponible es el bit 6, o el séptimo bit menos significativo (cuando vale 1 indica que está disponible).
- Procesador con una frecuencia de reloj de 4 GHz, el tiempo de ciclo t_{ciclo} = 0,25 ns. El procesador puede ejecutar 1 instrucción cada 2 ciclos de reloj.
- Transferencia de **escritura** desde memoria al puerto de E/S
- Transferencia de N_{datos}= 1.600.000 datos
- El tamaño de un dato es m_{dato} = 4 bytes
- Dirección inicial de memoria donde residen los datos: 20000000h

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, _1600000_
2.        MOV     R2, 20000000h
3.Bucle:  IN      R0, _[0200h]_      ; leer 4 bytes
4.        AND     R0, _0100000b_
5.        _JE_    Bucle
6.        MOV     R0,_[R2]_          ; leer 4 bytes
7.        _OUT_   [0204h], R0        ; escribir 4 bytes
8.        ADD     R2, _4_
9.        DEC     R3
10.       _JNE_   Bucle
```

Asignatura	Código	Fecha	Hora inicio
Estructura de computadores	75.573	19/6/2021	16:00

3.2.2

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

$t_{transf\_bloque} = t_{latència} + (N_{datos} * t_{transf\_dada})$
$t_{latencia} = 0$
$N_{datos} = 1600000$
$t_{transf\_dada} = m_{dato} / v_{transf} = 4$ Bytes / 2000 Kbytes/s = 0,002 ms = 2 us
$t_{transf\_bloque} = 0 + (1.600.000 * 0,002$ ms) = 3.200 ms =3,2 s

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?

$t_{ciclo} = 0,25$ ns (nanosegundos)

$t_{instr} = 0,25$ns * 2 = 0,5ns

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 i 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

Examen 2020/21-2

Asignatura	**Código**	**Fecha**	**Hora inicio**
Estructura de computadores	75.573	19/6/2021	16:00