Solución castellano 16 enero 2021

Estructura de Computadores (Universitat Oberta de Catalunya)

Examen 2020/21-1

Asignatura	**Código**	**Fecha**	**Hora inicio**
Estructura de computadores	75.573	16/1/2021	18:30

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**
- ¿Se puede consultar 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 2020/21-1

Asignatura	Código	Fecha	Hora inicio
Estructura de computadores	75.573	16/1/2021	18:30

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 2020/21-1

Asignatura	**Código**	**Fecha**	**Hora inicio**
Estructura de computadores	75.573	16/1/2021	18:30

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 2020/21-1

Asignatura	**Código**	**Fecha**	**Hora inicio**
Estructura de computadores	75.573	16/1/2021	18:30

Pregunta 1

1.1 Práctica – 1a parte

Modificad la subrutina copyMatrixP1 para que cuente los espacios en blanco que tiene la matriz Tiles. Si no hay 1 único espacio poner la variable state a '5' y salir. (No se tiene que escribir el código de toda la subrutina, sólo hay que modificar el código para hacer lo que pide el enunciado).

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; Copiar la matriz (tilesIni), sobre la matriz (tiles).
; Recorrer toda la matriz por filas de izquierda a derecha y de arriba a abajo.
; Para recorrer la matriz en ensamblador el índice va de 0 (posición [0][0])
; a 8 (posición [2][2]) con incrementos de 1 porque los datos son de
; tipo char(BYTE) 1 byte.
;
; Variables globales utilizadas:
; (tilesIni): Matriz con las fichas iniciales del juego
; (tiles)   : Matriz donde guardamos las fichas del juego.
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;; copyMatrixP1:
   push rbp
   mov  rbp, rsp

   push rax
   push rbx                        ;i
   push rcx                        ;j
   push rdx                        ;índice matriz

   mov r10, 0
   mov rdx, 0
   mov ebx, 0                      ;i=0
   copyMatrixP1_Bucle_Row:
   cmp ebx, DimMatrix              ;i<DimMatrix
   jge copyMatrixP1_Row_End

     mov ecx, 0                    ;j=0
     copyMatrixP1_Bucle_Col:
     cmp ecx, DimMatrix            ;j<DimMatrix
     jge copyMatrixP1_Col_End
       mov al, BYTE[tilesIni+rdx] ;al = tilesIni[i][j]
       mov BYTE[tiles+rdx], al    ;t[i][j] = al
       inc rdx
       inc ecx                   ;i++
       inc r10
     jmp copyMatrixP1_Bucle_Col
     copyMatrixP1_Col_End:         ;}
     inc  ebx                      ;j++
   jmp copyMatrixP1_Bucle_Row

   copyMatrixP1_Row_End:
   cmp r10, 1
   jne copyMatrixP1_Row_EndIf
     mov BYTE[State], '5'
   copyMatrixP1_Row_EndIf:

   pop rdx
   pop rcx
   pop rbx
   pop rax

   mov rsp, rbp
```

```
   pop rbp
   ret
```

1.2 Práctica – 2a parte

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

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; Mover la ficha en la dirección indicada por el carácter (charac),
; ('i':arriba, 'k':abajo, 'j':izquierda o 'l':derecha) recibido como
; parámetro en la posición donde hay el espacio dentro de la matriz indicada
; por la variable (spacePos) recibida como parámetro, controlando los casos
; donde no se puede hacer el movimiento.
; fila    (i = spacePos / DimMatrix) y columna (j = spacePos % DimMatrix)
; Si la casilla donde hay el espacio está en los extremos de la matriz
; no se podrá hacer el movimiento desde ese lado.
; Si se puede hacer el movimiento:
; Mover la ficha a la posición donde está el espacio de la matriz (tiles)
; y poner el espacio en la posición donde estaba la ficha movida.
; Para recorrer la matriz en ensamblador el índice va de 0 (posición [0][0])
; a 8 (posición [2][2]) con incrementos de 1 porque los datos son de tipo char(BYTE) 1 byte.
; No se tiene que mostrar la matriz con los cambios, se hace en UpdateBoardP2.
;
; Variables globales utilizadas: (tiles): Matriz donde guardamos la fichas del juego
; Parámetros de entrada: rdi (dil): (charac)  : Carácter leído de teclado.
;                        rsi (edi): (spacePos): Posición del espacio en la matriz (tiles).
; Parámetros de salida : rax (eax): (newSpacePos): Nueva posición del espacio en la matriz (tiles).
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
moveTileP2:
   push rbp
   mov  rbp, rsp
   ...
   mov r8, rsi             ;spacePos
   mov rax, rsi
   mov edx, 0
   mov ebx, DimMatrix
   div ebx
   mov r10d, eax           ;int i = spacePos / DimMatrix;
   mov r11d, edx           ;int j = spacePos % DimMatrix;
   mov r9, r8              ;int newPosSpace = spacePos;
                                     ;switch(charac){
   moveTileP2_Switchi:
   cmp dil, 'i'                      ;case 'i':
   jne moveTileP2_Switchk
      cmp r10d, DimMatrix-1          ;if (i < (DimMatrix-1)) {
      jge moveTileP2_SwitchEnd
         mov r9d, r8d
         add r9d, DimMatrix          ;newspacePos = spacePos+DimMatrix;
         mov dl, BYTE[tiles+r9]      ;tiles[i+1][j];
         mov BYTE[tiles+r8, dl       ;tiles[i][j]= tiles[i+1][j];
         mov BYTE[tiles+r9], ' '     ;tiles[i+1][j] = ' ';
         jmp moveTileP2_SwitchEnd    ;break
      moveTileP2_Switchk:
   ...
   moveTileP2_Switchj:
   ...
   moveTileP2_Switchl:
   ...
   moveTileP2_SwitchEnd:
   mov rax, r9                       ;return newspacePos;

   moveTileP2_End:
```

Examen 2020/21-1

Asignatura	**Código**	**Fecha**	**Hora inicio**
Estructura de computadores	75.573	16/1/2021	18:30

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

Examen 2020/21-1

Asignatura	Código	Fecha	Hora inicio
Estructura de computadores	75.573	16/1/2021	18:30

Pregunta 2

2.1

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

R2 = 00000050h R4 = 00000010h R8 = 00000200h	M(00000200h) = 0810077Eh M(00000050h) = F0000810h M(00000250h) = 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 A vale 200h.

a)

```
XOR R4,R4
MOV R4,[A]
ADD R4,[A]
```

R4 = 0
R4 = 0810077Eh
R4 = 0810077Eh + 0810077Eh
= 10200EFCh

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

b)

```
SUB R8, [A]
NOT R8
```

R8 = 200h – 0810097Eh = F7EFF882h
R8 = 0810077Dh

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

Examen 2020/21-1

Asignatura	Código	Fecha	Hora inicio
Estructura de computadores	75.573	16/1/2021	18:30

2.2

Suponemos que tenemos el vector V de 10 elementos de 32 bits. Completar la traducción del programa en ensamblador CISCA para que ejecute el algoritmo de alto nivel mostrado. (Hemos dejado 7 espacios para llenar)

```
i= 9;
while i!=0 do  {
     if (V [i] <= 10)  V[i] = V[i]-1;
     else              V[i] = 0;
     i= i-1;
};
```

```
      MOV  R1, 36
COND: CMP  R1  , 0
      JE   END
      CMP  [V+R1], 10
      JG   ELSE
      SUB  [V+R1], 1
      JMP  NEXT
ELSE: MOV  [V+R1], 0
NEXT: SUB  R1, 4
      JMP  COND
END:
```

Examen 2020/21-1

Asignatura	Código	Fecha	Hora inicio
Estructura de computadores	75.573	16/1/2021	18:30

2.3

Dado el siguiente fragmento de código de un programa en lenguaje ensamblador de CISCA:

```
            CMP  R11, [B]
            JNE  Label1
            INC  R11
Label1:     SUB  4, [B+R11]
```

Traducidlo a lenguaje máquina y expresadlo en la siguiente 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). Suponed que la dirección simbólica B vale 00000080h. En la siguiente tabla usáis una fila para codificar cada instrucción. Si suponemos que la instrucción empieza en la dirección @, el valor Bk de cada uno de los bytes de la instrucción con direcciones @+k para k=0, 1,… se tiene que indicar en la tabla en hexadecimal en la columna correspondiente (recordad que los campos que codifican un desplazamiento en 2 bytes o un inmediato o una dirección en 4 bytes lo hacen en formato little endian, esto hay que tenerlo en cuenta escribiendo los bytes de menor peso, de dirección más pequeña, a la izquierda y los de mayor peso, dirección mayor, a la derecha). Completad también la columna @ que indica para cada fila la dirección de memoria del byte B0 de la instrucción que se codifica en esta fila de la tabla.

A continuación os damos como ayuda las tablas de códigos:

Tabla de códigos de instrucción

B0	Instrucción
26h	CMP
42h	JNE
24h	INC
21h	SUB

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

Camp modo Bk<7..4>	Mode
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 para k=0..10										
@	Ensamblador	0	1	2	3	4	5	6	7	8	9	10
00023C00h	CMP R11, [B]	26	1B	20	80	00	00	00				
00023C07h	JNE Label1	42	60	02	00							
00023C0Bh	INC R11	24	1B									
00023C0Dh	SUB 4, [B+R11]	21	00	04	00	00	00	5B	80	00	00	00
00023C18h												

Examen 2020/21-1

Asignatura	**Código**	**Fecha**	**Hora inicio**
Estructura de computadores	75.573	16/1/2021	18:30

Pregunta 3

3.1. Memoria cache

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 memoria cache ya está llena, se utiliza un **algoritmo de reemplazamiento LRU.**

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

12, 4, 14, 28, 5, 20, 6, 44, 28, 7, 8, 30, 45, 55, 10, 43, 0, 56, 46, 1

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 acceso se debe rellenar una columna indicando si se trata de un acierto o un fallo.

Si es un acierto escribiremos E en la línea correspondiente delante de las direcciones del bloque, si es un fallo escribiremos F y se indicará el nuevo bloque que se trae a la memoria cache en 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 del bloque y a_7 es la octava (última) dirección del bloque.

Examen 2020/21-1

Asignatura	Código	Fecha	Hora inicio
Estructura de computadores	75.573	16/1/2021	18:30

Línea	Estado Inicial	12	4	14	28	5
0	0 (0 - 7)	0 (0 - 7)	E 0 (0 - 7)	0 (0 - 7)	0 (0 - 7)	E 0 (0 - 7)
1	1 (8 - 15)	E 1 (8 - 15)	1 (8 - 15)	E 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)

Línea	20	6	44	28	7	8
0	0 (0 - 7)	E 0 (0 - 7)	0 (0 - 7)	0 (0 - 7)	E 0 (0 - 7)	0 (0 - 7)
1	1 (8 - 15)	1 (8 - 15)	F 5 (40 – 47)	5 (40 – 47)	5 (40 – 47)	5 (40 – 47)
2	E 2 (16 - 23)	2 (16 - 23)	2 (16 - 23)	2 (16 - 23)	2 (16 - 23)	F 1 (8 - 15)
3	3 (24 - 31)	3 (24 - 31)	3 (24 - 31)	E 3 (24 - 31)	3 (24 - 31)	3 (24 - 31)

Línea	30	45	55	10	43	0
0	0 (0 - 7)	0 (0 - 7)	F 6 (48 - 55)	6 (48 - 55)	6 (48 - 55)	6 (48 - 55)
1	5 (40 – 47)	E 5 (40 – 47)	5 (40 – 47)	5 (40 – 47)	E 5 (40 – 47)	5 (40 – 47)
2	1 (8 - 15)	1 (8 - 15)	1 (8 - 15)	E 1 (8 - 15)	1 (8 - 15)	1 (8 - 15)
3	E 3 (24 - 31)	3 (24 - 31)	3 (24 - 31)	3 (24 - 31)	3 (24 - 31)	F 0 (0 – 7)

Línea	56	46	1			
0	F 7 (56 - 63)	7 (56 - 63)	7 (56 - 63)			
1	5 (40 – 47)	E 5 (40 – 47)	5 (40 – 47)			
2	1 (8 - 15)	1 (8 - 15)	1 (8 - 15)			
3	0 (0 – 7)	0 (0 – 7)	E 0 (0 – 7)			

3.1.2 a)

¿Cuál es la tasa de aciertos (T_a)?

T_a = 15 aciertos / 20 accesos = 0,75

3.1.2 b)

Suponemos que el tiempo de acceso a la memoria cache, o tiempo de acceso en caso de acierto (t_a), es de 2 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)?

$t_m = T_a \times t_a + (1-T_a) \times t_f$ = 0,75 * 2 ns + 0,25 * 25 ns = 1,5 ns + 6,25 ns = 7,75 ns

Examen 2020/21-1

Asignatura	Código	Fecha	Hora inicio
Estructura de computadores	75.573	16/1/2021	18:30

3.2 Sistema d'E/S

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

- Velocidad de transferencia del dispositivo de E/S ($\mathbf{v}_{transf}$)= 4 MBytes/s = 4000 Kbytes/s
- Tiempo de latencia media del dispositivo ($t_{latencia}$) = 0
- Direcciones de los **registros de datos** y **de estado** del controlador de E/S: 1F00h y 1F04h
- El bit del **registro de estado** que indica que el controlador del puerto de E/S está disponible es el bit 5, o el sexto bit menos significativo (cuando vale 1 indica que está disponible)
- Procesador con una frecuencia de reloj de 4 GHz, el tiempo de ciclo es de 0,25 ns. El procesador puede ejecutar una instrucción en 8 ciclos de reloj.
- Transferencia de **lectura** desde el puerto de E/S hacia la memoria
- Transferencia de N_{datos}= 800.000 datos
- La medida de un dato es m_{dato} = 4 Bytes
- El tiempo para atender la interrupción (t_{rec_int}) es de 20 ciclos de reloj

3.2.1 Completar el siguiente código CISCA, que es una rutina de servicio a las interrupciones (RSI) para transferir a través del 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 almacenar los datos recibidos.

```
1.   _CLI_
2.   PUSH R0
3.   PUSH R1
4.   IN   R0,_[1F00h]_
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_
```

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

El tiempo de un ciclo, t_{ciclo} = 0,25 ns (nanosegundos)

Tiempo para atender la interrupción, t_{rec_int}: 20 cicles * 0,25 ns = 5 ns

Tiempo de ejecución de una instrucción, t_{instr}: t_{ciclo} * 8 = 2 ns

Tiempo de ejecución RSI, t_{rsi}: N_{rsi} * t_{instr} = 12 instr. * 2 ns = 24 ns

Tiempo consumido por la CPU en cada interrupción, t_{transf_dato}: $t_{transf\_dato} = t_{rec\_int} + t_{rsi}$ = 5 + 24 = 29 ns

Número de interrupciones producidas (número total de datos, N_{datos}): 800000 interrupciones

Tiempo consumido en total por TODAS las interrupciones:

Asignatura	Código	Fecha	Hora inicio
Estructura de computadores	75.573	16/1/2021	18:30

$t_{transf\_bloque} = t_{transf\_dato} * N_{datos}$ = 29 ns * 800000 interrupciones = 23.200.000 ns = 23,2 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}

$t_{bloque} = t_{latencia} + (N_{datos} * t_{dato})$
$t_{latencia} = 0$
$N_{datos} = 800000$
$t_{dato} = m_{dato} / v_{transf}$ = 4 / 4000 Kbytes/s = 0,001 ms
t_{bloque} = 0 + (800000 * 0,001) ms = 800 ms
% ocupación = (t_{transf_bloque} * 100 / t_{bloque}) = (23,2 * 100) / 800 = 2,90%