Examen junio 2020

Estructura de Computadores (Universitat Oberta de Catalunya)

Espacio grapa

Examen 2019/20-2

Asignatura	Código	Fecha	Hora inicio
Estructura de computadores	75.573	13/06/2020	12:00

Ficha técnica del examen

- Comprueba que el código y el nombre de la asignatura corresponden a la asignatura matriculada.
- Tiempo total: **2 horas** Valor de cada pregunta: **Se indica en el enunciado**
- En el caso de que los estudiantes no puedan consultar algún material durante el examen, ¿cuáles son?: **NINGUNO**
- Se puede utilitzar calculadora? **NO** De que tipo? **NINGUNO**
- En el caso de que haya preguntas tipo test: ¿descuentan las respuestas erróneas? **NO** ¿Cuánto?

- Indicaciones específicas para la realización de este examen

Examen 2019/20-2

Asignatura	Código	Fecha	Hora inicio
Estructura de computadores	75.573	13/06/2020	12:00

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 2019/20-2

Asignatura	Código	Fecha	Hora inicio
Estructura de computadores	75.573	13/06/2020	12:00

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 rescribir el código o parte del código según vuestro planteamiento.

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 2019/20-2

Asignatura	Código	Fecha	Hora inicio
Estructura de computadores	75.573	13/06/2020	12:00

Pregunta 1

1.1 Práctica – 1a Parte

Escribir un fragmento de código ensamblador de la subrutina shiftNumbersRP1 que después de mirar si hay un número diferente de cero en la misma fila (k!=-1) para ponerlo en aquella posición y pondremos la variable (state) a '2' para indicar que se ha hecho el movimiento.(No se tiene que escribir el código de toda la subrutina)

```
;;;;;
; Desplazar a la derecha los números de cada fila de la matriz (m),
; manteniendo el orden de los números y poniendo los ceros a la izquierda.
; Recorrer la matriz por filas de derecha a izquierda y de abajo hacia arriba.
; Si se desplaza un número (NO LOS CEROS) pondremos la variable (state) a '2'.
; Si una fila de la matriz es: [0,2,0,4] y state = '1', quedará [0,0,2,4] y state= '2'.
; En cada fila, si encuentra un 0, mira si hay un número distinto de cero,
; en la misma fila para ponerlo en aquella posición.
; Para recorrer la matriz en ensamblador, el índice va de la posición 30 (posición [3][3]) a la 0
; (posición [0][0]) con decrementos de 2 porque los datos son de tipo short(WORD) 2 bytes.
; Per a acceder a una posición concreta de la matriz desde ensamblador hay que tener en cuenta que
; el índice es:(index=(fila*DimMatrix+columna)*2), multiplicamos por 2 porque los datos son de
; tipo short(WORD) 2 bytes. Los cambios se tienen que hacer sobre la misma matriz.
; No se tiene que mostrar la matriz.
; Variables globales utilizadas:
; state    : Estado del juego. (2: Se han hecho movimientos).
; m        : Matriz 4x4 donde hay los números del tablero de juego.
; rowcol   : Vector con la fila y la columna que queremos acceder de la matriz.
; indexMat : Índice para acceder a la matriz m.
;;;;;
shiftNumbersRP1:
   push rbp
   mov  rbp, rsp
   ...
   mov rax, (SizeMatrix-1)*2       ;index [i][j]
   mov rcx, 0                      ;index [i][k]
   mov r8, DimMatrix               ;i = DimMatrix
   dec r8                          ;i = DimMatrix-1
   shiftNumbersRP1_Rows:
      mov r9 , DimMatrix           ;j = DimMatrix
      dec r9                       ;j = DimMatrix-1
      shiftNumbersRP1_Cols:
      mov rax, r8                  ;rax = i
      shl rax, (DimMatrix/2)       ;rax = i*DimMatrix    (DimMatrix=4)
      add rax, r9                  ;rax = i*DimMatrix+j
      shl rax, 1                   ;rax = (i*DimMatrix+j)*2
      cmp WORD[m+rax], 0           ;if (m[i][j] == 0)
      jne shiftNumbersRP1_IsZero
         mov r10, r9
         dec r10                   ;k = j-1;
         mov rcx, r8               ;rcx = i
         shl rcx, (DimMatrix/2)    ;rcx = i*DimMatrix    (DimMatrix=4)
         add rcx, r10              ;rcx = i*DimMatrix+k
         shl rcx, 1                ;rcx = (i*DimMatrix+k)*2
         shiftNumbersRP1_While:
         cmp r10, 0                ;k>=0
         jl  shiftNumbersRP1_EndWhile
            cmp WORD[m+rcx], 0     ;m[i][k]==0
            jne shiftNumbersRP1_EndWhile
               sub rcx, 2
               dec r10             ;k--;
            jmp shiftNumbersRP1_While


         shiftNumbersRP1_EndWhile:
```

Examen 2019/20-2

Asignatura	Código	Fecha	Hora inicio
Estructura de computadores	75.573	13/06/2020	12:00

```
;if (k!=-1) {
;  m[i][j]=m[i][k];
;   m[i][k]= 0;
;  state='2';
;}
```

```
      shiftNumbersRP1_IsZero:
      dec r9
      cmp r9, 0                    ;j>0
      jg  shiftNumbersRP1_Cols
   shiftNumbersRP1_Next:
   dec r8
   cmp r8, 0                       ;i>=0
   jge  shiftNumbersRP1_Rows

shiftNumbersRP1_End:
   ...
   mov rsp, rbp
   pop rbp
   ret
```

1.2 Práctica – 2a parte

Completar el código de la subrutina showNumberP2. (Sólo completar los espacios marcados, no se pueden añadir o modificar otras instrucciones).

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; Esta subrutina se da hecha. NO LA PODEIS MODIFICAR.
; Mostrar un carácter (dil) en pantalla, recibido como parámetro,
; en la posición donde está el cursor llamando a la función printchP2_C.
; Variables globales utilizadas: Ninguna
; Parámetros de entrada:         rdi(dil): carácter que queremos mostrar
; Parámetros de salida :         Ninguno
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
printchP2:

; Convertir el valor recibido como parámetro (edx) de tipo int (DWORD)de 6 dígitos (num<=999999)
; a los caracteres ASCII que representen su valor.
; Hay que dividir el valor entre 10, de forma iterativa, hasta obtener 6 dígitos.
; En cada iteración, el residuo de la división que es un valor entre (0-9) indica el valor del
; dígito que tenemos que convertir a ASCII ('0'-'9') sumando '0' (48 decimal) para poderlo mostrar.
; Cuando el cociente sea 0 mostraremos espacios en la parte no significativa.
; Por ejemplo, si number=103 mostraremos "   103" y no "000103".
; Se tienen que mostrar los dígitos (carácter ASCII) desde la posición indicada por la fila (edi) y
; la columna (esi) recibidos como parámetro, posición de les unidades, hacia la izquierda.
; Como el primer dígito que obtenemos son las unidades, después las decenas, ..., para mostrar el
; valor se tiene que desplazar el cursor una posición a la izquierda en cada iteración.
; Para posicionar el cursor llamar a la subrutina gotoxyP2 y para mostrar los caracteres a la
; subrutina printchP2 implementando correctamente el paso de parámetros.
;
; Variables globales utilizadas:
; Ninguna.
```

Examen 2019/20-2

Asignatura	Código	Fecha	Hora inicio
Estructura de computadores	75.573	13/06/2020	12:00

```
; Parámetros de entrada:
; rdi (edi): Fila donde lo queremos mostrar en pantalla.
; rsi (esi): Columna donde lo queremos mostrar en pantalla.
; rdx (edx): Valor que queremos mostrar en pantalla.
; Parámetros de salida :
; Ninguno.
;;;;;
showNumberP2:
   push rbp
   mov  rbp, rsp
   ...
   mov eax, edx ;Valor que queremos mostrar
   cmp eax, 999999           ;if (n > 999999)
   jle showNumberP2_IniFor
        mov eax, 999999      ;n = 999999;
   showNumberP2_IniFor:
   mov ecx, 0                ;i=0
   showNumberP2_For:
   cmp ecx, 6                ;i<6
   _____ showNumberP2_End
     mov dl, ' '             ;charac = ' ';
     cmp eax, 0              ;(n > 0)
     jle showNumberP2_Show
       mov edx, 0
       mov ebx,10                ;n%10; n/10;
       div __________            ;EAX=EDX:EAX/EBX, EDX=EDX:EAX mod EBX
       add dl, __________        ;charac = charac + '0';

      showNumberP2_Show:
      call gotoxyP2 ;gotoxyP2_C(rScreen, cScreen);
      push rdi
      mov __________, dl
      call printchP2;printchP2_C(charac);
      _____ rdi
      dec __________           ;cScreen--;
      inc  ecx                 ;i++;
      jmp  showNumberP2_For
   showNumberP2_End:
   ...
   mov rsp, rbp
   pop rbp
   ret
```

Examen 2019/20-2

Asignatura	Código	Fecha	Hora inicio
Estructura de computadores	75.573	13/06/2020	12:00

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:

R0 = 00000A10h R1 = 00000B20h R2 = 00000C30h	M(00000A10h) = 0000F00Fh M(00000B20h) = 0000F000h M(00000C30h) = 00000FF0h M(00000FF0h) = 00000001h M(0000F00Fh) = 0000000Ah	Z = 0, C = 0, S = 0, V = 0

¿Cuál será el estado del computador después de ejecutar cada fragmento de código? (sólo modificaciones, excluyendo el PC).

a)

```
MOV R0, [R0]
ADD R0, [R2]
SAL  R0, Fh
NOT R0
```

Z = ___ , S = ___ , C = ___ , V = ___

b)

```
XOR  R1, R1
ADD  R1, [R0]
SUB  [R1], R0
```

Z = ___ , S = ___ , C = ___ , V = ___

Asignatura	Código	Fecha	Hora inicio
Estructura de computadores	75.573	13/06/2020	12:00

2.2

Dado el siguiente código de alto nivel:

Si (j>i) AND (A[j]==A[i]) i= j;

A es un vector de 8 elementos de 4 bytes cada uno. Se propone la siguiente traducción a CISCA donde hemos dejado 6 espacios para llenar.

```
      MOV R0, [j]
      MOV R1, [i]
      CMP R0, _________
      ____END
      MUL R0, _________
      MOV R2, _________
      MUL R1, 4
      CMP R2, _________
      ____END
OK:   MOV R0, [j]
      MOV [i], R0
END:
```

Asignatura	Código	Fecha	Hora inicio
Estructura de computadores	75.573	13/06/2020	12:00

2.3

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

```
        SUB R1, [R10]
INI:    ADD R1, 4
        MUL [A+R10], 100h
        JMP INI
```

Traducidlo a lenguaje máquina y expresadlo en la siguiente 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). Suponed que la dirección simbólica A vale 00004000h. En la siguiente tabla usad una fila para codificar cada instrucción. Si suponemos que la instrucción comienza en la dirección @, el valor Bk de cada uno de los bytes de la instrucción con direcciones @+k para k=0, 1,… se debe 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 esa fila de la tabla.

A continuación os damos como ayuda las tablas de códigos:

Tabla de códigos de instrucción

B0	Instrucción
21h	SUB
22h	MUL
20h	ADD
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>)

Campo modo Bk<3..0>	Significado
Nº 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
	SUB R1, [R10]											
	ADD R1, 4											
	MUL [A+R10], 100h											
	JMP INI											

Examen 2019/20-2

Asignatura	Código	Fecha	Hora inicio
Estructura de computadores	75.573	13/06/2020	12:00

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 utilizar 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:

0, 1, 2, 12, 62, 25, 63, 64, 17, 18, 19, 57, 58, 20, 21, 3, 4, 5, 65, 66

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 i se indicará el nuevo bloque que es trae a la memoria cache en la línea que le corresponda, expresando 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.

Línea	Estado Inicial	0	1	2	12	62
0	0:0 (0 - 7)					
1	1:0 (8 - 15)					
2	2:0 (16 - 23)					
3	3:0 (24 - 31)					

Línea	25	63	64	17	18	19
0						
1						
2						
3						

Examen 2019/20-2

Asignatura	**Código**	**Fecha**	**Hora inicio**
Estructura de computadores	75.573	13/06/2020	12:00

Línea	57	58	20	21	3	4
0						
1						
2						
3						

Línea	5	65	66			
0						
1						
2						
3						

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

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)?

Asignatura	Código	Fecha	Hora inicio
Estructura de computadores	75.573	13/06/2020	12:00

3.2 Sistema de E/S

3.2.1 E/S por interrupciones

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

- Velocidad de transferencia del dispositiv d'E/S v_{transf} = 10 MBytes/s = 10000 Kbytes/s
- Tiempo de latencia medio del dispositivo $t_{latencia}$ = 0
- Direcciones de los **registros de estado** y **datos** del controlador de E/S: 0BF0h y 0BF4h
- 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 2 instrucciones por ciclo de reloj
- Transferencia de **lectura** desde memoria al puerto de E/S
- Transferencia de N_{datos}= 100000 datos
- El tamaño de un dato es m_{dato} = 4 bytes
- El tiempo para atender a la interrupción (t_{rec_int}) es de 8 ciclos de reloj

a) 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____
3.   PUSHR1
4.   ____R0, [0BF4h]
5.   MOV R1, [Addr]
6.   MOV ____,R0
7.   ADD ____,4
8.   MOV ____,R1
9.   POP R1
10.  POP R0
11.  STI
12.  ____
```

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

Asignatura	Código	Fecha	Hora inicio
Estructura de computadores	75.573	13/06/2020	12:00

c) ¿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}

Examen 2019/20-2

Asignatura	Código	Fecha	Hora inicio
Estructura de computadores	75.573	13/06/2020	12:00

Pregunta 4

4.1

¿Cuál es la función y dónde se encuentran los “vectores de interrupción” en una máquina que disponga de E/S por interrupciones?

4.2

4.2.1

Uno de los factores básicos que hacen que el esquema de jerarquía de memorias funcione satisfactoriamente es la proximidad referencial. ¿Qué tipos de proximidad referencial podemos distinguir? Explicar brevemente en que consiste cada una de ellas.

4.2.2

Una manera de optimizar las operaciones de E/S por DMA consiste en reducir el número de cesiones y recuperaciones del bus, mediante una modalidad de transferencia denominada modo ráfaga. ¿Cuál es el funcionamiento del DMA en este modo en el caso de una transferencia del dispositivo a la memoria?

Examen 2019/20-2

Asignatura	**Código**	**Fecha**	**Hora inicio**
Estructura de computadores	75.573	13/06/2020	12:00