Examen Estructura de Computadores 2019/20

Estructura de Computadores (Universitat Oberta de Catalunya)

Examen 2019/20-1

Asignatura	Código	Fecha	Hora inicio
Estructura de computadores	75.573	22/01/2020	15:30

⊂75.573ℜ22ℜ01ℜ20ℜEΞK∈
75.573 22 01 20 EX

Espacio para la etiqueta identificativa con el código personal del **estudiante**.
Examen

Ficha técnica del examen

- Comprueba que el código y el nombre de la asignatura corresponden a la asignatura matriculada.
- Debes pegar una sola etiqueta de estudiante en el espacio correspondiente de esta hoja.
- No se puede añadir hojas adicionales, ni realizar el examen en lápiz o rotulador grueso.
- Tiempo total: **2 horas** Valor de cada pregunta: **Se indica en el enunciado**
- En el caso de que los estudiantes puedan consultar algún material durante el examen, ¿cuáles son?:

  NINGUNO
- En el caso de poder usar calculadora, de que tipo? NINGUNA
- En el caso de que haya preguntas tipo test: ¿descuentan las respuestas erróneas? NO ¿Cuánto?

- Indicaciones específicas para la realización de este examen

Examen 2019/20-1

Asignatura	Código	Fecha	Hora inicio
Estructura de computadores	75.573	22/01/2020	15:30

Enunciados

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 2019/20-1

Asignatura	**Código**	**Fecha**	**Hora inicio**
Estructura de computadores	75.573	22/01/2020	15:30

Pregunta 1

Examen 2019/20-1

Asignatura	**Código**	**Fecha**	**Hora inicio**
Estructura de computadores	75.573	22/01/2020	15:30

Pregunta 1

Examen 2019/20-1

Asignatura	Código	Fecha	Hora inicio
Estructura de computadores	75.573	22/01/2020	15:30

Pregunta 1

1.1 Pràctica – 1a Part

Escribir un fragmento de código ensamblador de la subrutina showMinesP1 que obtenga el valor de las unidades y de las decenas de la variable numMines y los deje en los registros al y dl respectivamente. (No se tiene que escribir el código de toda la subrutina).

```
;;;;;
; Convierte el valor del Número de minas que quedan por marcar (numMines)
; (entre 0 y 99) a dos caracteres ASCII.
; Se tiene que dividir el valor (numMines) entre 10, el cociente
; representará las decenas y el residuo las unidades, y después se
; tienen que convertir a ASCII sumando 48, carácter '0'.
; Mostrar los dígitos (carácter ASCII) de les decenas en la fila 27,
; columna 24 de la pantalla y las unidades en la fila 27, columna 26.
; (la posición se indica através de las variables rowScreen y colScreen).
; Para posicionar el cursor se llama a la subrutina gotoxyP1 y para
; mostrar los caracteres a la subrutina printchP1.
; Variables globales utilizadas:
; rowScreen: Fila de la pantalla donde posicionamos el cursor.
; colScreen: Columna de la pantalla donde posicionamos el cursor.
; numMines : Número de minas que quedan por marcar.
; charac   : Carácter a escribir en pantalla
;;;;;
showMinesP1:
      push rbp
      mov  rbp, rsp
      push rax
      push rbx
      push rdx

      mov rax, 0
      mov eax, DWORD[numMines]
      mov edx, 0

      ;calcular unidades y decenas
      mov ebx,10
      div ebx      ;EAX=EDX:EAX/EBX EDX=EDX:EAX%EBX

      add al,'0'  ;convertimos los valor a carácter ASCII
      add dl,'0'  ;charac = charac + '0';

      ;Posicionar el cursor y mostrar dígitos
      ; ESTA PARTE DEL CÓDIGO NO SE TIENE QUE IMPLEMENTAR
      showMinesP1_End:
      pop rdx
      pop rbx
      pop rax
      mov rsp, rbp
      pop rbp
```

Asignatura	Código	Fecha	Hora inicio
Estructura de computadores	75.573	22/01/2020	15:30

```
	ret
```

1.2 Práctica – 2a parte

Completar el código de la subrutina checkEnd, para verificar si hemos marcado todas las minas y hemos abierto todas las otras casillas marcar. (Sólo completar los espacios marcados, no se pueden añadir o modificar otras instrucciones).

```
;;;;;
; Verificar si hemos marcado todas las minas (numMines=0), que se reciben
; como parámetro y hemos abierto o marcado con una mina todas las otras
; casillas y no hay ningún espacio en blanco (' ') en la matriz (marks),
; si es así, cambiar el estado (state) que se recibe como parámetro, a
; 2 "Gana la partida". Retornar el estado del juego actualizado (status).
;
; Variables globales utilizadas:
; marks  : Matriz con las minas marcadas y las minas de las abiertas.
; Parámetros de entrada : rdi(edi) : Minas que quedan por marcar.
;                         rsi(esi) : Estado del juego.
; Parámetros de salida:   rax(eax) : Estado del juego.
;;;;;
checkEndP2:
	push rbp
	mov  rbp, rsp
	push rsi
	push rdi
	;Guardamos estado del juego para retornarlo
	mov  eax, __esi_
	;Miramos si hemos marcado todas las minas.
	cmp  __edi__, 0

	__jg__ checkEndP2_End

	mov  rsi, 0     ;índice para acceder a la matriz marks.
	;Iniciamos el bucle para mirar si hay espacios en blanco.
	checkEndP2_Loop:
	cmp __BYTE[marks+rsi]__, ' '
	je checkEndP2_End ;Si es espacio en blanco no hemos terminado.

	;incrementamos el índice para acceder a la matriz.
	inc rsi
	cmp rsi, __SizeMatrix__  ;DimMatrix*DimMatrix

	jl checkEndP2_Loop
	mov __eax__, 2  ;si hemos mirado todas las posiciones
	                ;y no hay ningún espacio, quiere decir
	                ;que hemos marcado todas las minas y
	                ;abierto el resto de posiciones.

	checkEndP2_End:
	pop rdi
	pop rsi
	mov rsp, rbp
	pop rbp
```

Examen 2019/20-1

Asignatura	Código	Fecha	Hora inicio
Estructura de computadores	75.573	22/01/2020	15:30

```
        ret
```

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= 100h R1= 200h R2= 300h R3= 400h	M(00000200h)=FFFFF600h M(00000300h)=00000600h M(00000400h)=0000FFFFh M(00000500h)=80000000h	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)

```
        ADD R3, R1
        SUB R3,[A+R1]
        MOV R2, 0
        JNE END
        DEC R1
END:
```

R3:= 400h + 200h = 600h
[00000400h]=0000FFFFh
R3= 00000600h – 0000FFFh= FFFF0601h
R2:= 0

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

b)

```
    MOV R2,[500]
    CMP R2,[R1]
    JNE EXIT
    ...
EXIT:
```

R2 = [00000500h]=80000000h

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

Examen 2019/20-1

Asignatura	**Código**	**Fecha**	**Hora inicio**
Estructura de computadores	75.573	22/01/2020	15:30

2.2

Suponemos que tenemos el vector V de 10 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)

```
i = 0;
do  {
        if (V [i] < i )  V[i] = 0;
        else V[i] = V[i]*2;
        i= i +1;
}
while (i <=9)
```

R2 representa el índice **i**

```
            MOV R1,0
            MOV R2,0
NEXT: CMP [V+R1], R2
            JL  CERO
            MUL [V+R1], 2
            JMP CONT
CERO:    MOV [V+R1], 0
CONT: ADD R1, 4
            ADD R2, 1
            CMP R2, 9
            JLE NEXT
END:
```

Examen 2019/20-1

Asignatura	Código	Fecha	Hora inicio
Estructura de computadores	75.573	22/01/2020	15:30

2.3

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

ADD [00023C00h+R1], 100h

CMP [R10], R2

MOV R1, [R2+A]

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 A vale 00004000h. En la siguiente tabla usad 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 pesp, 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
20h	ADD
26h	CMP
10h	MOV

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 para k=0..10										
@	Ensamblador	0	1	2	3	4	5	6	7	8	9	10
00023C00h	ADD [00023C00h+R1], 100h	20	61	00	3C	02	00	00	00	01	00	00
00023C0Bh	CMP [R10], R2	26	3A	12								

00023C0Eh.	MOV R1, [R2+A]	10	11	52	00	40	00	00				
00023C16h												

Examen 2019/20-1

Asignatura	Código	Fecha	Hora inicio
Estructura de computadores	75.573	22/01/2020	15:30

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 bolque 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, 63, 25, 64, 17, 18, 19, 2, 4, 6, 65, 66, 20, 56, 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. 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)	E 0:0 (0 - 7)	E 0:0 (0 - 7)	E 0:0 (0 - 7)	0:0 (0 - 7)	0:0 (0 - 7)
1	1:0 (8 - 15)	1:0 (8 - 15)	1:0 (8 - 15)	1:0 (8 - 15)	E 1:0 (8 - 15)	1:0 (8 - 15)
2	2:0 (16 - 23)	2:0 (16 - 23)	2:0 (16 - 23)	2:0 (16 - 23)	2:0 (16 - 23)	2:0 (16 - 23)
3	3:0 (24 - 31)	3:0 (24 - 31)	3:0 (24 - 31)	3:0 (24 - 31)	3:0 (24 - 31)	F 3:1 (56-63)

Línea	63	25	64	17	18	19
0	0:0 (0 - 7)	0:0 (0 - 7)	F 0:2 (64-71)	0:2 (64-71)	0:2 (64-71)	0:2 (64-71)
1	1:0 (8 - 15)	1:0 (8 - 15)	1:0 (8 - 15)	1:0 (8 - 15)	1:0 (8 - 15)	1:0 (8 - 15)

Examen 2019/20-1

Asignatura	Código	Fecha	Hora inicio
Estructura de computadores	75.573	22/01/2020	15:30

2		2:0 (16 - 23)		2:0 (16 - 23)		2:0 (16 - 23)	E	2:0 (16 - 23)	E	2:0 (16 - 23)	E	2:0 (16 - 23)
3	E	3:1 (56-63)	F	3:0 (24 - 31)		3:0 (24 - 31)		3:0 (24 - 31)		3:0 (24 - 31)		3:0 (24 - 31)

Línea	2		4		6		65		66		20	
0	F	0:0 (0-7)	E	0:0 (0-7)	E	0:0 (0-7)	F	0:2 (64-71)	E	0:2 (64-71)		0:2 (64-71)
1		1:0 (8 - 15)		1:0 (8 - 15)		1:0 (8 - 15)		1:0 (8 - 15)		1:0 (8 - 15)		1:0 (8 - 15)
2		2:0 (16 - 23)		2:0 (16 - 23)		2:0 (16 - 23)		2:0 (16 - 23)		2:0 (16 - 23)	E	2:0 (16 - 23)
3		3:0 (24 - 31)		3:0 (24 - 31)		3:0 (24 - 31)		3:0 (24 - 31)		3:0 (24 - 31)		3:0 (24 - 31)

Línea	56		42		50							
0		0:2 (64-71)		0:2 (64-71)		0:2 (64-71)						
1		1:0 (8 - 15)	F	1:1 (40 - 47)		1:1 (40 - 47)						
2		2:0 (16 - 23)		2:0 (16 - 23)	F	2:1 (48 - 55)						
3	F	3:1 (56-63)		3:1 (56-63)		3:1 (56-63)						

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 4 ns y el tiempo total de acceso en caso de fallo (t_f) es de 20 ns. Considerando la tasa de aciertos obtenida en la pregunta anterior, cuál es el tiempo medio de acceso a memoria (t_m) ?

$t_m = T_e \times t_e + (1-T_e) \times t_f = 0,6 * 4 \text{ ns} + 0,4 * 20 \text{ ns} = 2,4 \text{ ns} + 8 \text{ ns} = 10,4 \text{ ns}$

Examen 2019/20-1

Asignatura	Código	Fecha	Hora inicio
Estructura de computadores	75.573	22/01/2020	15:30

3.2 Sistema d'E/S

3.2.1 E/S programada

Si quiere analizar el rendimiento de la comunicación de datos entre la mamoria de un procesador y un puerto USB, utilizando E/S programada con las siguientes características:
Velocidad de transferencia del dispositivo d'E/S v_{transf} = 4 MBytes/s = 4000 Kbytes/s
Tiempo de latencia medio del dispositivo $t_{latencia}$ = 0
Direcciones de los **registros de estado** y **datos** 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}= 200000 datos
El tamaño de una dato es m_{dato} = 4 bytes
Dirección inicial de memoria donde residen los datos: A0000000h

a) 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, _200000_
2.        MOV  R2, A0000000h
3.Bucle:  _IN_      R0, [0F00h]   ; leer 4 bytes
4.        AND  R0, _00010000b_
5.        _JE_ Bucle
6.        MOV  R0,_[R2]_          ; leer 4 bytes
7.        ADD  R2, 4
8.        OUT  _[0F04h]_, R0      ; escribir 4 bytes
9.        SUB  R3, 1
10.       JNE  Bucle
```

b) 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} = 160000
t_{transf_dada} = m_{dato} / v_{transf} = 4 Bytes / 4000 Kbytes/s = 0,001 ms
t_{transf_bloque} = 0 + (200000 * 0,001 ms) = 200 ms = 0,2 s

c) Si quisiéramos utilizar el mismo procesaodr 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 / 1 = 0,50 ns

Examen 2019/20-1

Asignatura	**Código**	**Fecha**	**Hora inicio**
Estructura de computadores	75.573	22/01/2020	15:30

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 * 0,50 ns = 4 ns

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

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

Examen 2019/20-1

Asignatura	Código	Fecha	Hora inicio
Estructura de computadores	75.573	22/01/2020	15:30

Pregunta 4

4.1

¿Qué es la señal de reloj en un procesador? ¿En qué unidad se mide?

La señal de reloj del procesador es una señal de sincronización que marca el ritmo de ejecución de operaciones del procesador. Se mide en ciclos por segundos o hertzs y es uno de los elementos que determina la velocidad de un procesador.

4.2

a) 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.

Distinguimos dos tipos de proximidad referencial:

1) **Proximidad temporal**. Es cuando, en un intervalo de tiempo determinado, la probabilidad que un programa acceda de manera repetida a les mismas posiciones de memoria es muy grande.
   La proximidad temporal es debida principalmente a las estructuras iterativas; un bucle ejecuta las mismas instrucciones repetidamente, de la misma manera que las llamadas repetitivas a subrutinas.

2) **Proximidad espacial**. Es cuando, en un intervalo de tiempo determinado, la probabilidad que un programa acceda a posiciones de memoria próximas es muy grande.
   La proximidad espacial es debida principalmente al hecho que la ejecución de los programas es secuencial –se ejecuta una instrucción detrás de otra a excepción de las bifurcaciones–, y también a la utilización de estructuras de datos que están almacenadas en posiciones de memoria contiguas.

b) 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?

Cada vez que el módulo de E/S tiene un dato disponible el controlador de DMA lo almacena en la memoria intermedia y decrementa el registro contador. Cuando la memoria intermedia está llena o el contador ha llegado a cero, solicita el bus. Una vez el procesador le cede el bus, escribe en memoria todo el conjunto de datos almacenados en la memoria intermedia, y hace tantos accesos a memoria como datos tenemos y actualiza el registro de direcciones de memoria en cada acceso. Al acabar la transferencia del conjunto de datos libera el bus.
Una vez acabada una ráfaga, si el registre contador no ha llegado a cero, comienza la transferencia de una nueva ráfaga.