Examen Junio 2013

Estructura de Computadores (Universitat Oberta de Catalunya)

grapa

Examen 2012/13-2

Asignatura	Código	Fecha	Hora inicio
Estructura de computadores	75.573	15/06/2013	18:30

⊂75.573ℜ15ℜ06ℜ13ℜEΞP∈
75.573 15 06 13 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 de la cual estás matriculado.**
- **Debes pegar una sola etiqueta de estudiante en el espacio de esta hoja destinado a ello.**
- **No se puede añadir hojas adicionales.**
- **No se puede realizar las pruebas a lápiz o rotulador.**
- **Tiempo total** 2 **horas**
  - **En el caso de que los estudiantes puedan consultar algún material durante el examen, ¿cuál o cuáles pueden consultar?:** No se puede utilizar calculadora ni material auxiliar
  - **Valor de cada pregunta:** Pregunta 1 (20%); Pregunta 2 (40%); Pregunta 3 (40%)
- **En el caso de que haya preguntas tipo test: ¿descuentan las respuestas erróneas?** NO **¿Cuánto?**
- **Indicaciones específicas para la realización de este examen**

Enunciados

Examen 2012/13-2

Asignatura	Código	Fecha	Hora inicio
Estructura de computadores	75.573	15/06/2013	18:30

Pregunta 1: (20%)

Pregunta sobre la práctica.

Hay que completar las instrucciones marcadas o añadir el código 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 (40%)

2.1: 15%

2.2: 15%

2.3: 10%

Pregunta 3 (40%)

3.1: 20%
3.1.1: 10%
3.1.2: 10%

3.2: 20%
3.2.1: 10%
3.2.2: 10%

Examen 2012/13-2

Asignatura	Código	Fecha	Hora inicio
Estructura de computadores	75.573	15/06/2013	18:30

Pregunta 1

1.1

```
;;;;;
; Leer la combinación secreta.
; Primero limpiar el espacio donde se lee la combinación con espacios en blanco,
; llamando la función clearArea_C pasando como parámetro row y col.
; Inicializar a ceros el vector secret.
; Leer la combinación secreta, llamando a la subrutina GetCode, pasando como parámetro
; la dirección del vector secret a través del registro ebx, poner showChar=0, para
; indicar que GetCode no muestre los caracteres leídos y muestre *.
; ...
;
; Variables utilizadas:
; secret : vector que guarda la combinación secreta
; row : fila de la pantalla donde leeremos la combinación
; col : columna inicial de la pantalla donde leeremos la combinación.
; showChar: 0: mostrar un * para el carácter leído, 1: mostrar el carácter leído.
;
; Parámetros de entrada:
; Ninguno
;
; Parámetros de salida :
; eax: código de salida: 6 si se ha pulsado ESC, 0 en caso contrario
;;;;;
GetSecretCode:
push rbp
mov rbp, rsp

...

; escribir el código que considereis oportuno
; en el siguiente recuadro para
; inicializar a 0 las 5
; posiciones de tipo char del vector secret.

mov esi,0
GSC_iniVec:
mov byte [secret+esi],0
inc esi
cmp esi, 5
jl GSC_iniVec


...

mov rsp, rbp
pop rbp
ret
```

Examen 2012/13-2

Asignatura	Código	Fecha	Hora inicio
Estructura de computadores	75.573	15/06/2013	18:30

1.2

```
;;;;;
; ...
; Leer la combinación secreta, llamando a la subrutina GetCode, pasando como parámetro
; la dirección del vector secret a través del registro ebx, poner showChar=0, para
; indicar que GetCode no muestre los caracteres leídos y muestre *.
; Si GetCode retorna un 6,se ha pulsado ESC, imprimir el mensaje que corresponda
; llamando a la función printMessage_C y salir.
; en caso contrario llamar a la subrutina CheckSecret.
; Si la combinación secreta contiene caracteres inválidos o caracteres repetidos,
; mostrar un mensaje indicándolo llamando la función printMessage_C y
; repetir la lectura hasta que la combinación sea correcta.
; ...

;
; Variables utilizadas:
; secret : vector que guarda la combinación secreta
; row : fila de la pantalla donde leeremos la combinación
; col : columna inicial de la pantalla donde leeremos la combinación.
; showChar: 0: mostrar un * para el carácter leído, 1: mostrar el carácter leído.
;
; Parámetros de entrada:
; Ninguno
;
; Parámetros de salida :
; eax: código de salida: 6 si se ha pulsado ESC, 0 en caso contrario
;;;;;
GetSecretCode:
push rbp
mov rbp, rsp

...

mov ebx, __secret__
mov dword __[showChar]__, 0
call GetCode
cmp eax, 6
je GSC_finish

call __CheckSecret__
cmp eax, 0
je GSC_finish
mov __edi__, eax
__call__ printMessage_C
mov dword[col], 20
jmp GSC_readSecret
GSC_finish:


...

mov rsp, rbp
```

Examen 2012/13-2

Asignatura	Código	Fecha	Hora inicio
Estructura de computadores	75.573	15/06/2013	18:30

```
pop rbp
ret
```

Examen 2012/13-2

Asignatura	Código	Fecha	Hora inicio
Estructura de computadores	75.573	15/06/2013	18:30

Pregunta 2

2.1

Suponed el siguiente valor inicial de la CISCA (antes de cada apartado):

- Registros: Ri = 8 * i para i = 0, 1, …,15.
- Memoria: M(i)=(i+16) para i = 0, 4, 8,…, 2^{32}-4.

Completad el estado del computador después de ejecutar cada código (indicad los valores de los registros en hexadecimal).

Suponed que la dirección simbólica v vale 400h y la dirección simbólica B vale 800h

a)

```
    ADD  R1, 10h
    XOR  R10,[R2+E0h]
    MOV  [B+R2],R10
```

R1:= 8h+ 10h= 18h

R2+E0=10h+E0h=F0h

[R2+E0]=F0h+10h=100h

R10:= 50h XOR 100h = 150h

[B+R2]= [810H]= 150h

R1= 18h
R10 = 150h
[810H]= 150h

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

b)

```
    XOR   [V], R10
    SAR   [V], 1h
    JMP   F
S:  SUB   R1, R1
F:
```

[V]= 410h XOR 50H = 440h

[400]= 220h

[400]= 220h

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

2.2

En la memoria de un computador CISCA tenemos almacenados dos vectores A y B de 10 y 20 elementos respectivamente. Cada elemento es un número entero codificado en complemento a 2 con 32 bits.

Completad los espacios vacíos del fragmento de código CISCA para realizar el equivalente a la siguiente sentencia en C:

A[i]= A[i] + B[j]

Los vectores están almacenados en posiciones consecutivas de memoria, como es habitual cuando se traduce código en C. Por ejemplo, los elementos A[0], A[1], A[2] y M[7] se encuentran almacenados en las direcciones de memoria A, A+4, A+8 y A+28 respectivamente. Lo mismo para el vector B.

Examen 2012/13-2

Asignatura	Código	Fecha	Hora inicio
Estructura de computadores	75.573	15/06/2013	18:30

Se sabe que en R1 se encuentra almacenado el valor de la variable "i", y en R2 el de la "j" y que después de ejecutarse el fragmento de código todos los registros tienen que mantener los valores originales.

```
PUSH R2
PUSH R1
MUL  R1, 4
SAL  R2, 2
MOV  R3, [B+R2]
ADD  [A+R1], R3
POP  R1
POP  R2
```

2.3

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

```
MOV   R2, [V]
ADD   R1, 6
SUB   R1, R2
```

Traducidlo a lenguaje máquina y expresadlo en la siguiente tabla. Suponed que la primera instrucción del código se encuentra a partir de la dirección 00FF0010h (que es el valor del PC antes de empezar la ejecución del fragmento de código). Suponed también que la dirección simbólica V vale 00001D80h. En la tabla de resultados 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 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
10h	MOV
20h	ADD
21h	SUB

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

Campo modo Bk<7..4>	modo
0h	Inmediato
1h	Registro
2h	Memoria

Examen 2012/13-2

Asignatura	**Código**	**Fecha**	**Hora inicio**
Estructura de computadores	75.573	15/06/2013	18:30

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
00FF0010	`MOV R2, [V]`	10	12	20	80	1D	00	00				
00FF0017	`ADD R1, 6`	20	11	00	06	00	00	00				
00FF001E	`SUB R1, R2`	21	11	12								
00FF0021												

Examen 2012/13-2

Asignatura	Código	Fecha	Hora inicio
Estructura de computadores	75.573	15/06/2013	18:30

Pregunta 3

3.1

Tenemos un sistema de memoria en el que todos los accesos se realizan a palabra (no nos influye el tamaño de la palabra en sí). Supondremos que el espacio de direcciones de memoria se descompone en bloques de 8 palabras. Cada bloc empieza en una dirección múltiplo de 8. Así, el bloque 0 contiene las direcciones 0, 1, 2, 3, 4, 5, 6 y 7; el bloque 1, las direcciones 8, 9, 10, 11, 12, 13, 14 y 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 y 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 referencia a una dirección de memoria principal, si esta dirección no se encuentra en la memoria cache se lleva 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 llevaremos el bloque formado por las palabras 0, 1, 2, 3, 4, 5, 6 y 7).

3.1.1 Memoria Cache de Acceso Directo

Suponemos que el sistema utiliza una **política de asignación directa**, de manera que cada bloque de la memoria principal sólo se puede llevar línea determinada de la memoria cache.

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

24, 16, 8, 25, 2, 48, 22, 53, 2, 12, 21, 32, 22, 23, 13

3.1.1.a) 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 fallo en la cache hay que llenar una nueva columna indicando cual ha sido la referencia a memoria ha provocado el fallo y el cambio que se produce en el estado de la memoria cache (la línea que se modifica).

	Estado Inicial	Fallo: 48
Línea 0	0, 1, 2, 3, 4, 5, 6, 7	0, 1, 2, 3, 4, 5, 6, 7
Línea 1	8, 9, 10, 11, 12, 13, 14, 15	8, 9, 10, 11, 12, 13, 14, 15
Línea 2	16, 17, 18, 19, 20, 21, 22, 23	48, 49, 50, 51, 52, 53, 54, 55
Línea 3	24, 25, 26, 27, 28, 29, 30, 31	24, 25, 26, 27, 28, 29, 30, 31

	Fallo: 22	Fallo: 53
Línea 0	0, 1, 2, 3, 4, 5, 6, 7	0, 1, 2, 3, 4, 5, 6, 7
Línea 1	8, 9, 10, 11, 12, 13, 14, 15	8, 9, 10, 11, 12, 13, 14, 15
Línea 2	16, 17, 18, 19, 20, 21, 22, 23	48, 49, 50, 51, 52, 53, 54, 55
Línea 3	24, 25, 26, 27, 28, 29, 30, 31	24, 25, 26, 27, 28, 29, 30, 31

Examen 2012/13-2

Asignatura	**Código**	**Fecha**	**Hora inicio**
Estructura de computadores	75.573	15/06/2013	18:30

	Fallo: 22	Fallo: 32
Línea 0	0, 1, 2, 3, 4, 5, 6, 7	32, 33, 34, 35, 36, 37, 38, 39
Línea 1	8, 9, 10, 11, 12, 13, 14, 15	8, 9, 10, 11, 12, 13, 14, 15
Línea 2	16, 17, 18, 19, 20, 21, 22, 23	16, 17, 18, 19, 20, 21, 22, 23
Línea 3	24, 25, 26, 27, 28, 29, 30, 31	24, 25, 26, 27, 28, 29, 30, 31

3.1.1.b) ¿Cuál es la tasa de fallos (T_f) ?

T_f = 5 fallos / 15 accesos = 0,33

3.1.1.c) Suponemos que el tiempo de acceso a la memoria cache, o tiempo de acceso en caso de acierto (t_e), es de 2 ns y el tiempo total de acceso en caso de fallo (t_f) es de 20 ns. Considerando la tasa de fallos obtenida en la pregunta anterior, ¿cuál es el tiempo promedio de acceso a memoria (t_m) ?

$$t_m = T_f \times t_f + (1\text{-}T_f) \times t_e = 0{,}33 * 20 \text{ ns} + 0{,}66 * 2 \text{ ns} = 6{,}6 \text{ ns} + 1{,}33 \text{ ns} = 8 \text{ ns}$$

3.1.2 Memoria Cache de Acceso Completamente Asociativo

Ahora suponemos que el mismo 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 cache ya está llena, se utiliza un algoritmo de reemplazamiento LRU, de manera que sacaremos de la memoria cache aquel bloque que hace más tiempo que no se referencia.

Consideremos la mismo lista de lecturas a memoria:

24, 16, 8, 25, 2, 48, 22, 53, 2, 12, 21, 32, 22, 23, 13

3.1.2.a) 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.

	Estado Inicial	Fallo: 48
Línea 0	0, 1, 2, 3, 4, 5, 6, 7	0, 1, 2, 3, 4, 5, 6, 7
Línea 1	8, 9, 10, 11, 12, 13, 14, 15	8, 9, 10, 11, 12, 13, 14, 15
Línea 2	16, 17, 18, 19, 20, 21, 22, 23	48, 49, 50, 51, 52, 53, 54, 55
Línea 3	24, 25, 26, 27, 28, 29, 30, 31	24, 25, 26, 27, 28, 29, 30, 31

	Fallo: 22	Fallo: 12

Examen 2012/13-2

Asignatura	**Código**	**Fecha**	**Hora inicio**
Estructura de computadores	75.573	15/06/2013	18:30

Línea 0	0, 1, 2, 3, 4, 5, 6, 7	0, 1, 2, 3, 4, 5, 6, 7
Línea 1	16, 17, 18, 19, 20, 21, 22, 23	16, 17, 18, 19, 20, 21, 22, 23
Línea 2	48, 49, 50, 51, 52, 53, 54, 55	48, 49, 50, 51, 52, 53, 54, 55
Línea 3	24, 25, 26, 27, 28, 29, 30, 31	8, 9, 10, 11, 12, 13, 14, 15

	Fallo: 32	Fallo:
Línea 0	0, 1, 2, 3, 4, 5, 6, 7	
Línea 1	16, 17, 18, 19, 20, 21, 22, 23	
Línea 2	32, 33, 34, 35, 36, 37, 38, 39	
Línea 3	8, 9, 10, 11, 12, 13, 14, 15	

3.1.2.b) ¿Cuál es la tasa de fallos (T_f) ?

T_f = 4 fallos / 15 accesos = 0,266

3.1.2.c) Suponemos que el tiempo de acceso a la memoria cache, o tiempo de acceso en caso de acierto (t_e), es de 2 ns y el tiempo total de acceso en caso de fallo (t_f) es de 20 ns. Considerando la tasa de fallos obtenida en la pregunta anterior, ¿cuál es el tiempo promedio de acceso a memoria (t_m) ?

$$t_m = T_f \times t_f + (1\text{-}T_f) \times t_e = 0{,}266 * 20\ ns + 0{,}733 * 2\ ns = 5{,}66\ ns + 1{,}433\ ns = 6{,}8\ ns$$

3.2

Se quiere realizar la siguiente comunicación de datos entre la memoria de un computador y un port USB, que tienen las siguientes características:

- La CPU funciona con un reloj de 1GHz de frecuencia y ejecuta 1 instrucción por cada ciclo de reloj
- Direcciones de los **registros de datos** y **de estado** del controlador de E/S: 0B0h y 0B4h
- El bit del **registro de estado** que indica que el controlador del port de E/S está disponible es el bit 3, o el cuarto bit menos significativo (cuando vale 1 indica que está disponible)
- Transferencia de **escritura** desde memoria al port de E/S
- Transferencia de $\boldsymbol{N}_{datos}$=400.000 datos, es decir, 400.000 **×** 4 Bytes = 1.600.000 Bytes
- Dirección inicial de memoria donde residen los datos: 20000000h
- La velocidad de transferencia del port es de 10.000 Bytes por segundo

3.2.1 E/S programada

Examen 2012/13-2

Asignatura	Código	Fecha	Hora inicio
Estructura de computadores	75.573	15/06/2013	18:30

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

```
1.          MOV   R3, __400000__
2.          MOV   R2, 20000000h
3. Bucle:   IN    R0, [0B4h]          ; leer 4 bytes
4.          _AND_ R0, 00001000b
5.          _JE___Bucle
6.          MOV   R0,[_R2_]           ; leer 4 bytes
7.          ADD   R2, __4____
8.          _OUT_ 0B0h, R0            ; escribir 4 bytes
9.          SUB   R3, __1__
10.         _JNE_ Bucle
```

¿Cuál es el porcentaje de tiempo que dedica la CPU a la tarea de Entrada/Salida?

100%

3.2.2 E/S por Interrupciones

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, el mismo número de datos que antes con E/S programada, pero ahora mediante la técnica de E/S por interrupciones. Suponer:

- Se utiliza una variable global que se representa con la etiqueta **Dir,** y que al principio del programa contiene la dirección inicial de memoria donde residen los datos a transferir

```
1.          _CLI___
2.          PUSH      __R0___
3.          PUSH      R1
4.          _MOV_     R1, [Dir]
5.          MOV       R0, _[R1]___
6.          OUT       _[0B0h]_,R0  ; escribir 4 bytes
7.          _ADD___   R1, 4
8.          MOV       _[Dir]__, R1
9.          POP   __R1__
10.         POP   R0
11.         STI
12.         __IRET___
```

¿Cuál es el porcentaje de tiempo que dedica la CPU a la tarea de Entrada/Salida?

1.600.000 Bytes a transferir. 10.000 Bytes por segundo. Esto implica que el tiempo total de la transferencia sea de 160 segundos.

Cada ciclo de reloj es de 1ns. Por tanto, cada instrucción tarda 1 ns.

Una interrupción necssita 12 instrucciones, por tanto son 12 ns.

Hay 400.000 interrupciones, por tanto son 4.800.000 ns. o 4,8 ms.

Examen 2012/13-2

Asignatura	**Código**	**Fecha**	**Hora inicio**
Estructura de computadores	75.573	15/06/2013	18:30

Esto representa un 0,003% del tiempo total de la transferencia.

Examen 2012/13-2

Asignatura	**Código**	**Fecha**	**Hora inicio**
Estructura de computadores	75.573	15/06/2013	18:30

Examen 2012/13-2

Asignatura	**Código**	**Fecha**	**Hora inicio**
Estructura de computadores	75.573	15/06/2013	18:30