Examen 19 Junio 2019, preguntas y respuestas

Estructura de computadors (Universitat Oberta de Catalunya)

Espai grapa

Examen 2018/19-2

Asignatura	Código	Fecha	Hora inicio
Estructura de computadores	75.573	19/06/2019	15:30

⊂75.573ℜ19ℜ06ℜ19ℜEΞι∈
75.573 19 06 19 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 2018/19-2

Asignatura	Código	Fecha	Hora inicio
Estructura de computadores	75.573	19/06/2019	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 2018/19-2

Asignatura	Código	Fecha	Hora inicio
Estructura de computadores	75.573	19/06/2019	15:30

Pregunta 1

1.1 Práctica – 1a Parte

Escribir un fragmento de código ensamblador de la subrutina printSecretP1, para mostrar la combinación secreta en la parte superior del tablero cuando finaliza el juego.
(No se tiene que escribir el código de toda la subrutina).

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; Mostrar la combinación secreta en la parte superior del tablero
; cuando finaliza el juego.
; Para mostrar os valores de la combinación secreta se tiene que llamar
; a la función gotoxyP1 para posicionar el curso y printchP1 para
; mostrar los caracteres.
;
; 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.
; vSecret  : vector donde guardamos la combinación secreta.
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
printSecretP1:
   push rbp
   mov  rbp, rsp
   ...

   mov DWORD[rowScreen], 3     ;rowScreen = 3;
   mov DWORD[colScreen], 8     ;colScreen = 8;
   mov esi, 0
   ps_fori:              ;for (i=0; i<DimVector; i++){

   cmp esi, DimVector
   jge ps_fori_end
      call gotoxyP1              ;gotoxyP1_C();
      mov al, BYTE[vSecret+esi];charac = vSecret[i];
      mov BYTE[charac], al
      call printchP1             ;printchP1_C();
      add DWORD[colScreen], 2    ;colScreen = colScreen + 2;
      inc esi
      jmp ps_fori


   ps_fori_end:

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

1.2 Práctica – 2a parte

Completar el código de la subrutina checkSecretP2. (Sólo completar los espacios marcados, no se pueden añadir o modificar otras instrucciones).

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; Verifica que la combinación secreta (vSecret) no tenga espacios.
; Para cada elemento del vector (vSecret) mirar que no haya un espacio.
; Si la combinación secreta es correcta, poner (state=1) para indicar
; que la combinación secreta es correcta y que vamos a leer jugadas.
; Si la combinación secreta es incorrecta, poner (state=3) para volverla
; a pedir sin inicializarla.
;
; Variables globales utilizadas:
; vSecret  : vector donde guardamos la combinación secreta
; state    : estado del juego.
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
checkSecretP2:
   push rbp
   mov  rbp, rsp
   ...
   mov  rbx, rdi

   mov ecx, 0                 ;int secretError = 0;
   mov rsi, 0
   cs_fori:                   ;for (i=0;i<DimVector;i++) {
     cmp rsi, DimVector
     jge cs_fori_end
       cs_if1:                ;if (vS[i]==' ') {
       cmp __BYTE[rbx+rsi]__, ' '
       jne cs_if1_end
         mov ecx, 1           ;  secretError=1;
       cs_if1_end:

     mov rdi, rsi
     inc rdi
     cs_forj:                 ;for (j=i+1;j<DimVector;j++) {
       cmp edi, DimVector
       __jge__ cs_forj_end
         cs_if2:              ;  if (vS[i]==vS[j])
         mov al, __BYTE[rbx+rdi]__
         cmp __BYTE[rbx+rsi], al__
         jne cs_if2_end
           mov ecx, 1         ;secretError=1;
       cs_if2_end:
          inc __rdi__
          jmp cs_forj
     cs_forj_end:
     inc __rsi__
     jmp cs_fori
   cs_fori_end:

   cs_if3:           ;if (secretError==1) state = 3;
   cmp ecx, 1
   jne cs_if3_else
     mov eax, 3
     jmp cs_if3_end
   cs_if3_else:     ;else state=1;
     mov eax, 1
   cs_if3_end:

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

Examen 2018/19-2

Asignatura	Código	Fecha	Hora inicio
Estructura de computadores	75.573	19/06/2019	15:30

Pregunta 2

2.1

L'estat inicial del computador CISCA just abans de començar l'execució de cada fragment de codi (a cada apartat) és el següent:

R2 = 00000250h R4 = 00000500h R8 = 00000750h	M(00000500h) = 0000FFFFh M(000009A0h) = 0F0F0F0Fh M(00000250h) = FFFF0000h	Z = 0, C = 0, S = 0, V = 0

Completeu l'estat del computador després d'executar cada codi (indiqueu els valors dels registres en hexadecimal). Suposeu que l'adreça simbòlica A val 250h.

a)

```
ADD R8, R2
SAL [R8],4
MOV R8, R2
```

R8 = 000009A0h
[R8] = [000009A0h] = F0F0F0F0h
R8 = 00000250h

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

b)

```
MOV R4,[A+R2]
NOT R4
XOR R4, [R2]
```

R4 = 0000FFFFh
R4 = FFFF0000h
R4 = 00000000h

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

Asignatura	Código	Fecha	Hora inicio
Estructura de computadores	75.573	19/06/2019	15:30

2.2

Dado el siguiente código de alto nivel:

```
i= -8;
do {
        A[8+i]= 0;
        i= i + 1;
} while (i<0);
```

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 [I], F8h
        MOV R0, [I]
PLUS:
        MOV R1, R0
        ADD R1, 8
        SAL R1, 2
        MOV [A+R1], 0
        ADD R0, 1
        CMP R0,0
        JL PLUS
        MOV [I], R0
        END:
```

Asignatura	Código	Fecha	Hora inicio
Estructura de computadores	75.573	19/06/2019	15:30

2.3

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

```
        SUB R0, [X]
        JG BEGIN
        JMP END
BEGIN: CMP [X+R2], 100h
END:
```

Traducidlo a lenguaje máquina y expresadlo en la siguiente tabla. Suponed que la primera instrucción del código se asemeja a partir de la dirección 0FF00FF0h (que es el valor del PC antes de empezar la ejecución del fragmento de código). Suponed que la dirección simbólica X vale 00000440h. 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
45h	JG
26h	CMP
40h	JMP
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

Examen 2018/19-2

Asignatura	**Código**	**Fecha**	**Hora inicio**
Estructura de computadores	75.573	19/06/2019	15:30

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
0FF00FF0h	SUB R0,[X]	21	10	20	40	04	00	00				
0FF00FF7h	JG BEGIN	45	60	06	00							
0FF00FFBh	JMP END	40	00	0C	10	F0	0F					
0FF01001h	CMP [X+R2], 100h	26	52	40	04	00	00	00	00	01	00	00
0FF0100Ch												

Examen 2018/19-2

Asignatura	Código	Fecha	Hora inicio
Estructura de computadores	75.573	19/06/2019	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 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, 13, 25, 26, 17, 8, 22, 3, 24, 62, 25, 63, 64, 17, 18, 19, 57, 58, 20, 25

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 bloc.

Línea	Estado Inicial	12		13		25		26		17	
0	0 (0 - 7)		0 (0 - 7)		0 (0 - 7)		0 (0 - 7)		0 (0 - 7)		0 (0 - 7)
1	1 (8 - 15)	E	1 (8 - 15)	E	1 (8 - 15)		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)	E	2 (16 - 23)
3	3 (24 - 31)		3 (24 - 31)		3 (24 - 31)	E	3 (24 - 31)	E	3 (24 - 31)		3 (24 - 31)

Examen 2018/19-2

Asignatura	Código	Fecha	Hora inicio
Estructura de computadores	75.573	19/06/2019	15:30

Línea	8	22	3	24	62	5
0	0 (0 - 7)	0 (0 - 7)	E 0 (0 - 7)	0 (0 - 7)	0 (0 - 7)	E 0 (0 - 7)
1	E 1 (8 - 15)	1 (8 - 15)	1 (8 - 15)	1 (8 - 15)	F 7 (56 - 63)	7 (56 - 63)
2	2 (16 - 23)	E 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)	E 3 (24 - 31)	3 (24 - 31)	3 (24 - 31)

Línea	63	64	17	18	19	57
0	0 (0 - 7)	0 (0 - 7)	0 (0 - 7)	0 (0 - 7)	0 (0 - 7)	0 (0 - 7)
1	E 7 (56 - 63)	7 (56 - 63)	7 (56 - 63)	7 (56 - 63)	7 (56 - 63)	E 7 (56 - 63)
2	2 (16 - 23)	F 8 (64 - 71)	8 (64 - 71)	8 (64 - 71)	8 (64 - 71)	8 (64 - 71)
3	3 (24 - 31)	3 (24 - 31)	F 2 (16 - 23)	E 2 (16 - 23)	E 2 (16 - 23)	2 (16 - 23)

Línea	58	20	25			
0	0 (0 - 7)	0 (0 - 7)	F 3 (24 – 31)			
1	E 7 (56 - 63)	7 (56 - 63)	7 (56 - 63)			
2	8 (64 - 71)	8 (64 - 71)	8 (64 - 71)			
3	2 (16 - 23)	E 2 (16 - 23)	2 (16 - 23)			

3.1.2 a)

¿Cuál es la tasa de aciertos (T_a)?

T_a = 16 aciertos / 20 accesos = 0,8

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 40 ns. Considerando la tasa de aciertos obtenida en la pregunta anterior, ¿cuál es el tiempo medio de acceso a memoria (t_m)?

$t_m = T_e \times t_e + (1-T_e) \times t_f$ = 0,8 * 5 ns + 0,2 * 40 ns = 4 ns + 8 ns = 12 ns

Examen 2018/19-2

Asignatura	Código	Fecha	Hora inicio
Estructura de computadores	75.573	19/06/2019	15:30

3.2 Sistema d'E/S

E/S programada
Si 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 d'E/S v_{transf} = 2 MBytes/s = 2000 Kbytes/s
- Tiempo de latencia medio del dispositivo $t_{latencia}$ = 0
- Direcciones de los **registros de estado** y **datos** del controlador de E/S: 0A00h y 0A04h
- El bit del **registro de estado** que indica que el controlador del puerto de E/S está disponible es el bit 2, o el tercer 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 **escritura** desde memoria al puerto de E/S
- Transferencia de N_{datos}= 160000 datos
- El tamaño de un dato es m_{dato} = 4 bytes
- Dirección inicial de memoria donde residen los datos: C0000000h

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, _160000_
2.       MOV R2, C0000000h
3.Bucle: _IN_     R0, [0A00h]  ; leer 4 bytes
4.       AND R0, _00000100b_
5.       _JE_Bucle
6.       MOV R0,_[R2]_         ; leer 4 bytes
7.       ADD R2, 4
8.       OUT _[0A04h]_, R0     ; escribir 4 bytes
9.       SUB R3, 1
10.      JNE Bucle
```

Asignatura	Código	Fecha	Hora inicio
Estructura de computadores	75.573	19/06/2019	15:30

3.2.2

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

$t_{transf\_bloque} = t_{latència} + (N_{datos} * t_{transf\_dada})$
$t_{latencia} = 0$
$N_{datos} = 160000$
$t_{transf\_dada} = m_{dato} / v_{transf} = 4$ Bytes / 2000 Kbytes/s = 0,002 ms
$t_{transf\_bloque} = 0 + (160000 * 0{,}002$ ms) = 320 ms = 0,32 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?

$t_{ciclo} = 0{,}5$ ns (nanosegundos)

$t_{instr} = 0{,}5$ ns / 2 = 0,250 ns

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 * 0,250 ns = 2 ns

Por tanto, el tiempo mínimo para transferir un dato es: 2 ns

Se pueden transferir 4 bytes cada 2 ns, es decir: $4 / 2 * 10^{-9}$ = 2000 Mbyte/s = 2 Gbytes/s

Examen 2018/19-2

Asignatura	Código	Fecha	Hora inicio
Estructura de computadores	75.573	19/06/2019	15:30

Pregunta 4

4.1

¿Qué es el registro SP y cuál es su función?

El registro SP también llamado Stack Pointer es el registro que apunta a la cima de la Pila de datos en ensamblador. Es incrementado y decrementado automáticamente por las instrucciones PUSH y POP cuando trabajan con la pila añadiendo o quitando datos.

4.2

4.2.1

¿Cuáles son las tres políticas de asignación para almacenar datos dentro de una memoria cache? ¿En qué consisten?

1) Política de asignación directa: un bloque de la memoria principal sólo puede estar en una única línea de la memoria cache. La memoria cache de asignación directa es la que tiene la tasa de fallos más alta, pero se utiliza mucho porqué es la más barata y fácil de gestionar

2) Política de asignación completamente asociativa: un bloque de la memoria principal puede almacenarse en cualquier línea de la memoria cache. La memoria cache completamente asociativa es la que tiene la tasa de fallos más baja. A pesar de eso, no se suele utilizar porqué es la más cara y compleja de gestionar.

3) Política de asignación asociativa por conjuntos: un bloque de la memoria principal puede almacenarse en un subconjunto de las líneas de la memoria cache, pero dentro del subconjunto puede encontrarse en cualquier posición. La memoria cache asociativa por conjuntos es una combinación.

4.2.2

En un sistema de E/S gestionado por DMA. Explica, cuándo i por qué se produce una interrupción. ¿Sirve para indicar el inicio o el final de una transferencia? ¿Quién la genera?

Finalización de la operación de E/S: Cuando ha finalizado la transferencia del bloque el controlador de DMA envía una petición de interrupción al procesador para informar que ha acabado la transferencia de datos.

Examen 2018/19-2

Asignatura	**Código**	**Fecha**	**Hora inicio**
Estructura de computadores	75.573	19/06/2019	15:30