grapa

Examen 2015/16-1

Asignatura	Código	Fecha	Hora inicio
Estructura de computadores	75.573	16/01/2016	12:00

⊂75.573ℜ16ℜ01ℜ16ℜΕΞΛ∈
75.573 16 01 16 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 (35%); Pregunta 3 (35%); Pregunta 4 (10%)
- **En el caso de que haya preguntas tipo test: ¿descuentan las respuestas erróneas?** NO **¿Cuánto?**
- **Indicaciones específicas para la realización de este examen**

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.

Examen 2015/16-1

Asignatura	Código	Fecha	Hora inicio
Estructura de computadores	75.573	16/01/2016	12:00

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 2015/16-1

Asignatura	Código	Fecha	Hora inicio
Estructura de computadores	75.573	16/01/2016	12:00

Pregunta 1

1.1 Práctica – Parte obligatoria

Escribir un fragmento de código ensamblador de la subrutina copyMatrixP1 que copia la matriz (mRotated) sobre la matriz (m), que son matrices de tipo WORD. (No hay que escribir el código de toda la subrutina).

```
;;;;;
; Copiar los valores de la matriz (mRotated) a la matriz (m).
; Variables utilizadas:
; m       : matriz 4x4 donde hay los números del tablero de juego.
; mRotated: matriz 4x4 para hacer la rotación.
; Parámetros de entrada: Ninguno.
; Parámetros de salida : Ninguno.
;;;;;
copyMatrixP1:
    push rbp
    mov  rbp, rsp
      ...
    mov eax, 0
    mov r8d, 0                              ;i = r8d
    copyMatrixP1_Rows:
    mov r9d , 0                             ;j = r9d
    copyMatrixP1_Cols:

       mov bx, WORD[mRotated+eax]    ;mRotated[i][j]
       mov WORD[m+eax], bx           ;m[i][j] = mRotated[i][j]
       add eax, 2                    ;incrementamos el índice
       inc r9d                       ;
       cmp r9d, DimMatrix            ;

       jl  copyMatrixP1_Cols
    inc r8d
    cmp r8d, DimMatrix
    jl  copyMatrixP1_Rows
    copyMatrixP1_End:
      ...
    mov rsp, rbp
    pop rbp
    ret
```

Examen 2015/16-1

Asignatura	Código	Fecha	Hora inicio
Estructura de computadores	75.573	16/01/2016	12:00

1.2 Práctica – Parte opcional

Completar el código de la subrutina rotateMatrixRP2. (Sólo completar los espacios marcados, no se pueden añadir o modificar otras instrucciones).

```
;;;;;
; Calcular el valor del índice para acceder a una matriz (4x4) que guardaremos en el
; registro (eax) a partir de la fila (edi) y la columna (esi) recibidos como parámetro.
; eax=((edi*DimMatrix)+(esi))*2
; multiplicamos por 2 porque es una matriz de tipo short (WORD) 2 bytes.
; Esta subrutina no tiene una función en C equivalente.
; Variables utilizadas: Ninguna.
; Parámetros de entrada: rdi(edi) : fila para acceder a la matriz (4x4).
;                        rsi(esi) : columna para acceder a la matriz (4x4).
; Parámetros de salida : rax(eax) : índice para acceder a la matriz (4x4) de tipo WORD.
;;;;;
calcIndexP2:

;;;;;
; Rotar a la derecha la matriz recibida como parámetro (edi), sobre
; la matriz (mRotated).
; La primera fila pasa a ser la cuarta columna, la segunda fila pasa
; a ser la tercera columna, la tercera fila pasa a ser la segunda
; columna y la cuarta fila pasa a ser la primer columna.
; En el enunciado se explica con más detalle como hacer la rotación.
; NOTA: NO es lo mismo que fer la matriz traspuesta.
; La matriz recibida como parámetro no se tiene que modificar,
; los cambios se tiene que hacer en la matriz (mRotated).
; Para acceder a matrices desde ensamblador hay que calcular el índice
; a partir de la fila y la columna llamando a la subrutina calcIndexP1.
; m[row][col], en C, es equivalente a WORD[m+eax], en ensamblador, si
; eax = ((row*DimMatrix)+(col))*2. m[1][2] és DWORD[m+12].
; No se tiene que mostrar la matriz.
; Variables utilizadas:  Ninguna.
; Parámetros de entrada: rdi(edi): Dirección de la matriz que queremos rotar.
; Parámetros de salida : Ninguno.
;;;;;
rotateMatrixRP2:
    ...
    mov edx, edi                        ;Dirección de la matriz
    mov r8d, 0                          ;i = r8d
    rotateMatrixRP2_Rows:
        mov r9d , 0                     ;j = r9d
        rotateMatrixRP2_Cols:
        mov edi, r8d
        mov esi, r9d
        call calcIndexP2        ;índice para acceder a la matriz[i][j];
        mov bx, __WORD__[__edx+eax__]
        mov r10d, DimMatrix
        dec r10d
        sub r10d, __r8d__           ;DimMatrix-1-i = r10d
        mov edi, __r9d__
        mov esi, __r10d__
        call calcIndexP2
        ;mRotated[j][DimMatrix-1-i] = mRotate[i][j]
        mov __WORD__[__mRotated+eax__], bx
        inc r9d
        cmp r9d, DimMatrix
        jl  rotateMatrixRP2_Cols
    inc r8d
    cmp r8d, DimMatrix
    jl  rotateMatrixRP2_Rows
```

Examen 2015/16-1

Asignatura	**Código**	**Fecha**	**Hora inicio**
Estructura de computadores	75.573	16/01/2016	12:00

```
rotateMatrixRP2_End:
...
ret
```

Examen 2015/16-1

Asignatura	Código	Fecha	Hora inicio
Estructura de computadores	75.573	16/01/2016	12:00

Pregunta 2

2.1

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

R1 = 00000008h R5 = 00000028h R10 = 00000050h	M(00007E40h) = 00007E50h M(00000800h) = 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 V vale 800h.

a)

```
MUL R5, 10h
MUL R10, 16
SUB R10, R5
```

```
R5= 00000280h
R10= 00000500h
R10= 00000280h

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

b)

```
MOV R1, [V]
XOR R1, [00007E40h]
DEC R1
```

```
R1= 00000810h
R1= 00007640h
R1= 0000763Fh

R1= 0000763Fh
C=0, V=0, S=0, Z=0
```

Examen 2015/16-1

Asignatura	Código	Fecha	Hora inicio
Estructura de computadores	75.573	16/01/2016	12:00

2.2

Dado el siguiente código de alto nivel:

```
if (A<B) {
      if (C>=B) A=C;
      else B= C;
}
else A=B;
```

Se propone la siguiente traducción a CISCA donde hemos dejado 6 espacios para llenar.

```
INI: MOV R0, [A]
     MOV R1, [B]
     MOV R2, [C]
     CMP R0, R1
     JGE  LSE1
     CMP R2, R1
     JL   LSE2
     MOV R0, R2
     JMP  END
LSE2: MOV R1, R2
     JMP END
LSE1:MOV R0, R1
END: MOV [A], R0
     MOV [B], R1
     MOV [C], R2
```

Examen 2015/16-1

Asignatura	Código	Fecha	Hora inicio
Estructura de computadores	75.573	16/01/2016	12:00

2.3

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

```
MOV R1,R10
MUL R12, 4
SUB [A+R12], 2048
```

Traducidlo a lenguaje máquina y expresadlo en la siguiente tabla. Suponed que la primera instrucción del código se ensambla a partir de la dirección 00004600h (que es el valor del PC antes de empezar la ejecución del fragmento de código). Suponed que la dirección simbólica A vale 02A00600h. 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
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
Nº registro	Si el modo tiene que especificar un registro
0	No se especifica registro.

Examen 2015/16-1

Asignatura	**Código**	**Fecha**	**Hora inicio**
Estructura de computadores	75.573	16/01/2016	12:00

		Bk para k=0..10										
@	**Ensamblador**	0	1	2	3	4	5	6	7	8	9	10
00004600h	`MOV R1,R10`	10	11	1A								
00004603h	`MUL R12, 4`	22	1C	00	04	00	00	00				
0000460Ah	`SUB [A+R12], 2048`	21	5C	00	06	A0	02	00	00	08	00	00
00004615h												

Examen 2015/16-1

Asignatura	Código	Fecha	Hora inicio
Estructura de computadores	75.573	16/01/2016	12:00

Pregunta 3

3.1. Memoria cache

Memoria cache completamente asociativa (FIFO)

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 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 FIFO.**

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

0, 1, 2, 15, 23, 16, 55, 56, 17, 18, 28, 30, 40, 5, 63, 25, 43, 56, 42, 50

Inicialmente la memoria cache está vacía y se llena secuencialmente comenzando por la línea 0.

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	15	23
0	-	F 0 (0 - 7)	E 0 (0 - 7)	E 0 (0 - 7)	0 (0 - 7)	0 (0 - 7)
1	-	-	-	-	F 1 (8 - 15)	1 (8 - 15)
2	-	-	-	-	-	F 2 (16 - 23)
3	-	-	-	-	-	-

Línea	16	55	56	17	18	28
0	0 (0 - 7)	0 (0 - 7)	F 7 (56 - 63)	7 (56 - 63)	7 (56 - 63)	7 (56 - 63)
1	1 (8 - 15)	1 (8 - 15)	1 (8 - 15)	1 (8 - 15)	1 (8 - 15)	F 3 (24 - 31)
2	E 2 (16 - 23)	2 (16 - 23)	2 (16 - 23)	E 2 (16 - 23)	E 2 (16 - 23)	2 (16 - 23)
3	-	F 6 (48 - 55)	6 (48 - 55)	6 (48 - 55)	6 (48 - 55)	6 (48 - 55)

Examen 2015/16-1

Asignatura	**Código**	**Fecha**	**Hora inicio**
Estructura de computadores	75.573	16/01/2016	12:00

Línea	30	40	5	63	25	43
0	7 (56 - 63)	7 (56 - 63)	7 (56 - 63)	E 7 (56 - 63)	7 (56 - 63)	7 (56 - 63)
1	E 3 (24 - 31)	3 (24 - 31)	3 (24 - 31)	3 (24 - 31)	E 3 (24 - 31)	3 (24 - 31)
2	2 (16 - 23)	F 5 (40 - 47)	5 (40 - 47)	5 (40 - 47)	5 (40 - 47)	E 5 (40 - 47)
3	6 (48 - 55)	6 (48 - 55)	F 0 (0 - 7)	0 (0 - 7)	0 (0 - 7)	0 (0 - 7)

Línea	56	42	50			
0	E 7 (56 - 63)	7 (56 - 63)	F 6 (48 - 55)			
1	3 (24 - 31)	3 (24 - 31)	3 (24 - 31)			
2	5 (40 - 47)	E 5 (40 - 47)	5 (40 - 47)			
3	0 (0 - 7)	0 (0 - 7)	0 (0 - 7)			

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

T_e = 11 aciertos / 20 accesos = 0,55

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 2 ns y el tiempo total de acceso en caso de fallo (t_f) es de 30 ns. Considerando la tasa de aciertos obtenida en la pregunta anterior, cuál es el tiempo medio de acceso a memoria (t_m) ?

$t_m = T_e \times t_e + (1-T_e) \times t_f$ = 0,55 * 2 ns + 0,45 * 30 ns = 1,1 ns + 13,5 ns = 14,6 ns

Examen 2015/16-1

Asignatura	Código	Fecha	Hora inicio
Estructura de computadores	75.573	16/01/2016	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}= 400000 datos
- El tamaño de un dato es m_{dato} = 4 bytes
- Dirección inicial de memoria donde residen los datos: A0000000h
- El tiempo para atender a la interrupción (t_{rec_int}) es de 2 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 _R0_
3.  PUSH R1
4.  _IN_ R0, [0BF4h]
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_
```

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

El tiempo de un ciclo, t_{ciclo} = 0,5 ns (nanosegundos)

Tiempo para atender la interrupción, t_{rec_int}: 2 cicles * 0,5 ns = 1 ns

Tiempo de ejecución de una instrucción, t_{instr}: t_{ciclo} = 0,50 ns

Tiempo de ejecución RSI, t_{rsi}: N_{rsi} * t_{instr} = 12 instr. * 0,50 ns = 6 ns

Examen 2015/16-1

Asignatura	**Código**	**Fecha**	**Hora inicio**
Estructura de computadores	75.573	16/01/2016	12:00

Tiempo consumido por CPU en cada interrupción, t_{transf_dada}:

$t_{transf\_dada} = t_{rec\_int} + t_{rsi} = 1 + 6 = 7$ ns

Número de interrupcions producidas (número total de datos, N_{datos}): 400000 interrupciones

Tiempo consumido en total en TODAS las interrupciones:

$t_{transf\_bloque} = t_{transf\_dato} * N_{datos} = 7$ ns * 400000 interrupciones = 2800000 ns = 2,8 ms (milisegundos)

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}

$t_{bloque} = t_{latencia} + (N_{datos} * t_{dato})$
$t_{latencia} = 0$
$N_{datos} = 400000$
$t_{dato} = m_{dato} / v_{transf} = 4 / 1000$ Kbytes/s = 0,004 ms
$t_{bloque} = 0 + (400000 * 0{,}004)$ ms = 160 ms
% ocupación = $(t_{transf\_bloque} * 100 / t_{bloque}) = (2{,}8 * 100) / 160 = 1{,}75\%$

Examen 2015/16-1

Asignatura	Código	Fecha	Hora inicio
Estructura de computadores	75.573	16/01/2016	12:00

Pregunta 4

4.1

Cuando se diseña una arquitectura, uno de los aspectos importantes es el tamaño de las instrucciones. ¿Qué dos alternativas hay y cuáles son sus ventajas e inconvenientes?

• Instrucciones de tamaño fijo: todas las instrucciones ocuparán el mismo número de bits. Esta alternativa simplifica el diseño del procesador y la ejecución de las instrucciones puede ser más rápida.
• Instrucciones de tamaño variable: el tamaño de las instrucciones dependerá del número de bits necesario para cada una. Esta alternativa permite diseñar un conjunto amplio de códigos de operación, el direccionamiento puede ser más flexible y permite poner referencias a registros y memoria. Como contrapartida, aumenta la complejidad del procesador.

4.2

a) En la memoria cache, ¿Qué políticas de asignación se definen? Describirlas brevemente.

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 ne cualquier posición. La memoria cache asociativa por conjuntos es una combinación.

b) ¿Cuáles son los pasos básicos para la gestión de una interrupción en un sistema con una única línea de interrupción y un único módulo de E/S?

1.- Petición del módulo de Entrada/Salida
2.- Ciclo de reconocimiento de la interrupción
    2.a.- Reconocimiento de la interrupción
    2.b.- Salvaguarda del estado del procesador

Examen 2015/16-1

Asignatura	Código	Fecha	Hora inicio
Estructura de computadores	75.573	16/01/2016	12:00