Examen 15 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	15/06/2019	12:00

⊂75.573ℜ15ℜ06ℜ19ℜEΞE∈
75.573 15 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	15/06/2019	12:00

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	15/06/2019	12:00

Pregunta 1

1.1 Práctica – 1a Parte

Escribir un fragmento de código ensamblador de la subrutina printTriesP1, mostrar los intentos que queden (tries) para acertar la combinación secreta en la parte inferior izquierda del tablero.

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; Mostrar los intentos que quedan (tries) para acertar la combinación
; secreta.
; Situar el cursor en la fila 23, columna 5 llamando a la subrutina gotoxyP1.
; Mostrar el carácter asociado al valor de la variable (tries)
; llamando a la subrutina printchP1.
; Para obtener el carácter asociado a los intentos, código ASCII del número,
; hay que sumar al valor numérico de los intentos (tries) 48 (código
; ASCII de '0'). (charac=tries+'0' o charac=tries+48).
;
; 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.
; tries    : número de intentos que quedan.
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
printTriesP1:
   push rbp
   mov  rbp, rsp

   push rax

    mov  DWORD[rowScreen], 23   ;rowScreen = 3;
    mov  DWORD[colScreen], 5    ;colScreen = 8;

    call gotoxyP1               ;gotoxyP1_C();
    mov  al, BYTE[tries]        ;charac = tries + '0';
    add  al, '0'
    mov  BYTE[charac], al
    call printchP1              ;printchP1_C();

   pt_end:
   pop rax

   mov rsp, rbp
   pop rbp
   ret
```

1.2 Práctica – 2a parte

Completar el código de la subrutina checkPlayP2. (Sólo completar los espacios marcados, no se pueden añadir o modificar otras instrucciones).

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; Situar el cursor en la fila (9+(DimVector-tries)*2) y la columna (24)
; (parte derecha del tablero) para mostrar los aciertos en el tablero
; de juego.
; Primero se muestran los aciertos en el sitio (hX), tantas 'X' como
; aciertos en el sitio haya y tantas 'O' como aciertos fuera de sitio
; (hO) haya, recibidos como parámetro.
; Para mostrar los aciertos se tiene que llamar a la función gotoxyP2_C
; para posicionar el cursor y printchP2_C para mostrar los caracteres.
; Cada vez que se muestra un acierto se tiene que incrementar en 2 la
; columna.
; NOTA: (hX + hO tiene que ser siempre menor o igual a 6).
;
; Variables globales utilizadas:
; Ninguna
;
; Parámetros de entrada:
; hX    : rdi(dil):Aciertos en el sitio.
; hO    : rsi(sil):Aciertos fuera de sitio.
; tr    : rdx(edx):Número de intentos que quedan.
;
; Parámetros de salida :
; Ninguno
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
printHitsP2:

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; Cuenta los aciertos en el sitio y fuera de sitio de la jugada (vP)
; respecto de la combinación secreta (vS), recibidos como parámetro.
; Para hacer la comparación hay que hacer lo siguiente:
; Para cada elemento de la combinación secreta (vS)
; (vector de referencia porque no tiene valores repetidos).
; Comparar-lo con todos los elementos de la jugada (vP).
; Si un elemento de la combinación secreta (vS[i]) es igual
; al elemento de la misma posición de la jugada(vP[i]): será un
; acierto en el sitio 'X' y se tiene que incrementar la variable donde
; guardamos los aciertos el sitio (hX++), si no, será un acierto fuera
; de sitio 'O', si un elemento de la combinación secreta (vS[i]) es
; igual a un elemento la jugada (vP[j]), pero ocupan posiciones distintas
; en los vectores (i!=j), se tiene que incrementar la variable donde
; guardamos los aciertos fuera de sitio (hO++) y dejar de recorrer la
; jugada (vector vP) para contar cada acierto una sola vez.
; Si todas las posiciones de la combinación secreta (vS) y de la jugada
; (vP) són iguales (hX=5) hemos ganado y se tiene que modificar el
; estado del juego para indicarlo (status=5).
; Mostrar los aciertos en el sitio y fuera de sitio en el tablero de
; juego llamando a la función printHitsP2_C.
; Retornar el estado actual del juego.
;
; Variables globales utilizadas:
; Ninguna
;
; Parámetros de entrada:
; vS     : rdi(edi):dirección del vector donde guardamos la combinación secreta.
; vP     : rsi(esi):dirección del vector donde guardamos cada jugada.
; tr     : rdx(edx):número de intentos que quedan.
; status : rcx(ecx):estado del juego.
;
; Parámetros de salida :
; status : rax(eax): estado del juego.
;
; Parámetros de salida :
; rax(eax): estado del juego.
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
checkPlayP2:
   push rbp
   mov  rbp, rsp
```

Examen 2018/19-2

Asignatura	Código	Fecha	Hora inicio
Estructura de computadores	75.573	15/06/2019	12:00

```
    ...
    mov  r8, rdi
    mov  r9, rsi
    mov al, 0         ;hX=0;
    mov bl, 0         ;hO=0
    mov rsi, 0


cp_fori:          ;for (i=0;i<DimVector;i++) {
    cmp rsi, DimVector
    jge cp_fori_end
      cp_if1:                     ;if (vS[i]==vP[i]) {
      mov r10b, BYTE[r9+rsi]
      cmp BYTE[r8+rsi], r10b
      jne cp_if1_else
        inc al        ;hX++;
        jmp cp_if1_end
      cp_if1_else:
        mov rdi, 0
        cp_forj:                   ;for (j=0;j<DimVector;j++) {
        cmp rdi, DimVector
        jge cp_forj_end
          cp_if2:                  ;if (vS[i]==vP[j]) {
          mov r10b, __BYTE[r9+rdi]__
          cmp __BYTE[r8+rsi]__, r10b
          jne cp_if2_end
            inc bl             ;hO++;
            mov rdi, DimVector ;j=DimVector;
          cp_if2_end:
          inc rdi
          jmp cp_forj
        cp_forj_end:
      cp_if1_end:
      inc rsi
      jmp cp_fori
    cp_fori_end:

    cp_if3:           ;if (hX == 6 ) {
    cmp al, 6
    jne cp_if3_end
      mov ecx, 5      ;status = 5;
    cp_if3_end:

    mov  __dil, __al
    mov  __sil, __bl
    call __printHitsP2__   ;printHitsP2_C(hX, hO, tr);
    mov __eax__, ecx       ;return status;

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

Examen 2018/19-2

Asignatura	Código	Fecha	Hora inicio
Estructura de computadores	75.573	15/06/2019	12:00

Pregunta 2

2.1

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

R2 = 00000010h R4 = 00000020h R8 = 00000030h	M(00000020h) = 80808080h M(000000FFh) = 11111111h M(00000100h) = 44444444h	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 100h.

a)

```
    MOV R2, [A]
    CMP  [000000FFh], R2
    JG X
    MOV R2, 0
X:
```

R2 = 44444444h
CMP 11111111h, 44444444h
R2 = 00000000h

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

b)

```
MOV R4, [R4]
SAL R4, R2
```

R4= 80808080h
R4= 80800000h

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

Asignatura	Código	Fecha	Hora inicio
Estructura de computadores	75.573	15/06/2019	12:00

2.2

Dado el siguiente código de alto nivel:

Si (A[j]>A[i]) A[i]= A[i]+A[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]
      MUL R0, 4
      MOV R2, [A+R0]
      MOV R1, [i]
      MUL R1, 4
      CMP R2, [A+R1]
   JLE END
      ADD [A+R1], R2
END:
```

Asignatura	Código	Fecha	Hora inicio
Estructura de computadores	75.573	15/06/2019	12:00

2.3

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

```
        XOR R0, R0
eti1:   SUB R0, [R2+100H]
        ADD R2, 4
        JNE eti1
```

Traducidlo a lenguaje máquina y expresadlo en la siguiente tabla. Suponed que la primera instrucción del código se asemeja a partir de la dirección 00000FF0h (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 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
32h	XOR
21h	SUB
20h	ADD
42h	JNE

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	15/06/2019	12:00

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
00000FF0h	XOR R0, R0	32	10	10								
00000FF3h	SUB R0,[R2+100H]	21	10	42	00	01						
00000FF8h	ADD R2, 4	20	12	00	04	00	00	00				
00000FFFh	JNE eti1	42	60	F0	FF							

Examen 2018/19-2

Asignatura	Código	Fecha	Hora inicio
Estructura de computadores	75.573	15/06/2019	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 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 FIFO.**

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

12, 13, 25, 26, 7, 8, 22, 23, 24, 62, 25, 63, 64, 17, 18, 19, 57, 58, 20, 25

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

3.1.1

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	12		13		25		26		7	
0	-	F	1 (8 – 15)	E	1 (8 – 15)		1 (8 – 15)		1 (8 – 15)		1 (8 – 15)
1	-					F	3 (24 -31)	E	3 (24 -31)		3 (24 -31)
2	-									F	0 (0 - 7)
3	-										

Examen 2018/19-2

Asignatura	Código	Fecha	Hora inicio
Estructura de computadores	75.573	15/06/2019	12:00

Línea	8	22	23	24	62	25
0	E 1 (8 – 15)	1 (8 – 15)	1 (8 – 15)	1 (8 – 15)	F 7 (56 – 63)	7 (56 – 63)
1	3 (24 -31)	3 (24 -31)	3 (24 -31)	E 3 (24 -31)	3 (24 -31)	E 3 (24 -31)
2	0 (0 - 7)	0 (0 - 7)	0 (0 - 7)	0 (0 - 7)	0 (0 - 7)	0 (0 - 7)
3		F 2 (16 – 23)	E 2 (16 – 23)	2 (16 – 23)	2 (16 – 23)	2 (16 – 23)

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

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

3.1.2 a)

¿Cuál es la tasa de aciertos (T_e)?

T_e = 13 aciertos / 20 accesos = 0,65

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)?

$t_m = T_e \times t_e + (1-T_e) \times t_f = 0,65 * 2 \text{ ns} + 0,35 * 30 \text{ ns} = 1,3 \text{ ns} + 10,5 \text{ ns} = 11,8 \text{ ns}$

Asignatura	Código	Fecha	Hora inicio
Estructura de computadores	75.573	15/06/2019	12:00

3.2 Sistema d'E/S

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 dispositivo 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: 0AB0h y 0AB4h
- 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 500 MHz, el tiempo de ciclo t_{ciclo} = 2 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}= 32000 datos
- El tamaño de un dato es m_{dato} = 4 bytes
- Dirección inicial de memoria donde residen los datos: FF000000h
- El tiempo para atender a la interrupción (t_{rec_int}) es de 5 ciclos de reloj

3.2.1

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, [0AB4h]
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_
```

Asignatura	Código	Fecha	Hora inicio
Estructura de computadores	75.573	15/06/2019	12:00

3.2.2

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

El tiempo de un ciclo, t_{ciclo} = 2 ns (nanosegundos)

Tiempo para atender la interrupción, t_{rec_int}: 5 cicles * 2 ns = 10 ns

Tiempo de ejecución de una instrucción, t_{instr}: t_{ciclo} /1= 2 ns

Tiempo de ejecución RSI, t_{rsi}: N_{rsi} * t_{instr} = 12 instr. * 2 ns = 24 ns

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

t_{transf_dada} = t_{rec_int} + t_{rsi} = 10 + 24 = 34 ns

Número de interrupcions producidas (número total de datos, N_{datos}): 32000 interrupciones

Tiempo consumido en total en TODAS las interrupciones:

t_{transf_bloque} = t_{transf_dato} * N_{datos} = 34 ns * 32000 interrupciones = 1088000 ns = 1,088 ms (milisegundos)

3.2.3

¿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} = 32000
t_{dato} = m_{dato} / v_{transf} = 4 / 10000 Kbytes/s = 0,0004 ms
t_{bloque} = 0 + (32000 * 0,0004) ms = 12,8 ms
% ocupación = (t_{transf_bloque} * 100 / t_{bloque}) = (1,088 * 100) / 12,8 = 8,5%

Examen 2018/19-2

Asignatura	Código	Fecha	Hora inicio
Estructura de computadores	75.573	15/06/2019	12:00

Pregunta 4

4.1

¿En qué consiste el modo de direccionamiento relativo a registro base?. Además de explicarlo pon una instrucción de ejemplo en la que se utilice.

El modo de direccionamiento relativo a registro base especifica un registro sumado a un número entre corchetes [ ]. El registro contendrá una dirección de memoria que actuará como dirección base y el número como un desplazamiento respecto a esta dirección.
Un ejemplo es «mov rax, [rbx+4]", donde RBX funciona como registro base y 4 es el desplazamiento.

4.2

4.2.1

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.

4.2.2

¿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
    2.c.- Llamada a la RSI
3.- Ejecución de la rutina de servicio de interrupción
    3.a. -Inicio de la ejecución de la RSI
    3.b.- Intercambio del dato
    3.c Finalización de la ejecución de la RSI
    3.d Retorno de interrupción: Restaurar el estado del procesador