Examen 16 Junio 2018, preguntas y respuestas

Estructura de computadors (Universitat Oberta de Catalunya)

Espai grapa

Examen 2017/18-2

Asignatura	Código	Fecha	Hora inicio
Estructura de computadores	75.573	16/06/2018	12:00

⊂75.573ℜ16ℜ06ℜ18ℜEΞ?∈
75.573 16 06 18 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 2017/18-2

Asignatura	Código	Fecha	Hora inicio
Estructura de computadores	75.573	16/06/2018	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 2017/18-2

Asignatura	Código	Fecha	Hora inicio
Estructura de computadores	75.573	16/06/2018	12:00

Pregunta 1

1.1 Práctica – 1a Parte

Escribir el fragmento de código ensamblador que falta a la subrutina copyMatrixP1 para copiar la matriz (tilesIni) sobre la matriz (tiles), que son matrices de tipo BYTE. (No se tiene que escribir el código de toda la subrutina).

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; Copiar la matriz (tilesIni) en la matriz (tiles), de tipo char
; (1 byte cada posición).
; Para acceder a las matrices (tilesIni) y (tiles) se tiene que acceder
; con un índice que inicialmente valdrá 0 y se irá incrementando hasta 15.
; 0,1,2,3 corresponde a la fila 1, 4,5,6,7 corresponde a la fila 2, ..., hasta al 15.
; Después, actualizar la posición del espacio en pantalla en l vector
; (rowcolSpaceScreen) (fila (DWORD[rowcolSpaceScreen+0])=15 y la columna
; (DWORD[rowcolSpaceScreen+4])=19).
;
; Variables globales utilizadas:
; tilesIni         : Matriz con los números iniciales del juego.
; tiles            : Matriz donde guardamos los números del juego.
; charac           : Carácter que queremos mostrar.
; rowcolSpaceScreen: Vector donde tenemos la posición del espacio en pantalla.
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
copyMatrixP1:
   push rbp
   mov  rbp, rsp
   ...
   mov edx, 0                       ;índice matriz
   mov ebx, 0                       ;i=0
   copyMatrixP1_bucle_Row:
   cmp ebx, DimMatrix               ;i<DimMatrix
   jge copyMatrixP1_Row_end
     mov ecx, 0                     ;j=0
     copyMatrixP1_bucle_Col:
     cmp ecx, DimMatrix             ;j<DimMatrix


     jge copyMatrixP1_Col_end
       mov  al, BYTE[tilesIni+edx] ;al = tilesIni[i][j]
       mov  BYTE[tiles+edx], al    ;tiles[i][j] = al
       inc  edx
       inc  ecx                    ;i++
     jmp copyMatrixP1_bucle_Col

     copyMatrixP1_Col_end:
     inc  ebx                      ;j++
   jmp copyMatrixP1_bucle_Row



   copyMatrixP1_Row_end:
   ;Indiquem la posició de l'espai.
   mov DWORD[rowcolSpaceScreen+0], 15   ;rowcolSpaceScreen[0]=15;
   mov DWORD[rowcolSpaceScreen+4], 19   ;rowcolSpaceScreen[1]=19;
   copyMatrixP1_end:
   ...
   mov rsp, rbp
   pop rbp
   ret
```

Examen 2017/18-2

Asignatura	Código	Fecha	Hora inicio
Estructura de computadores	75.573	16/06/2018	12:00

1.2 Práctica – 2a parte

Completar el codi de la subrutina updateBoardP2. (Sólo completar los espacios marcados, no se puede añadir código, ni modificar otras instrucciones).

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; Situar el cursor en una fila y una columna de la pantalla
; en función de la fila (edi) y de la columna (esi) recibidos como
; parámetro llamando a la función gotoxyP2_C.
;
; Variables globales utilizadas: Ninguna
; Parámetros de entrada: rdi(edi): Fila
;                        rsi(esi): Columna
; Parámetros de salida : Ninguno
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
gotoxyP2:

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; Mostrar un carácter (dil) en pantalla, recibido como parámetro,
; en la posición donde está el cursor llamando a la función printchP2_C.
;
; Variables globales utilizadas: Ninguna
; Parámetros de entrada: rdi(dil): carácter que queremos mostrar
; Parámetros de salida : Ninguno
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
printchP2:

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; Mostrar los valores de la matriz (t), recibida como paràmetro, en la
; pantalla, dentro del tablero, en las posiciones correspondientes.
; Se tiene que recorrer toda la matriz (t) de tipo char (1 byte cada
; posición), y para cada elemento de la matriz:
; Posicionar el cursor en el tablero llamando a la función gotoxyP2_C.
; La posición inicial del cursor es la fila 11 y la columna 11
; que es la posición en pantalla de la casilla [0][0].
; Mostrar los caracteres de cada posición de la matriz (t)
; llamando a la función printchP2_C.
; Después, posicionar el cursor en la pantalla en función del vector
; (rcCurScreen) recibido como parámetro (fila (DWORD[rcCurScreen+0]) y
; la columna (rDWORD[rcCurScreen+4]) llamando a la función gotoxyP2_C().
;
; Variables globales utilizadas:  Ninguna
;
; Parámetros de entrada: rdi : Matriz donde guardamos los números del juego.
;                        rsi : Vector donde tenemos la posición del cursor en pantalla.
; Parámetros de salida : Ninguno
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
updateBoardP2:
   push rbp
   mov  rbp, rsp
   ...
   mov rbx, rdi                          ;rbx: t
   mov rdx, rsi                          ;rdx: rcCurScreen
   mov rcx, 0                            ;rcx: indexMat
   mov edi, 11                           ;rowScreen=11;
   mov r10d, 0                           ;i=0
   updateBoardP2_bucle_Row:
   cmp r10d, DimMatrix                   ;i<DimMatrix
   jge updateBoardP2_Cur
      mov esi, 11                        ;colScreen=11;
      mov r11d, 0                        ;j=0;
      updateBoardP2_bucle_Col:
      cmp r11d, DimMatrix                ;j<DimMatrix
      jge updateBoardP2_Col_end
         call gotoxyP2          ;gotoxyP2_C(rowScreen, colScreen);
         push rdi
         mov  dil, __BYTE[rbx+rcx]__ ;charac = t[i][j];
```

Examen 2017/18-2

Asignatura	Código	Fecha	Hora inicio
Estructura de computadores	75.573	16/06/2018	12:00

```
        _call_ printchP2        ;printchP2_C(t[i][j]);
        _pop__ rdi
        inc  rcx
        inc  r11d                       ;j++;
        add   __esi__, 4        ;colScreen = colScreen + 4;
    jmp updateBoardP2_bucle_Col
    updateBoardP2_Col_end:
    inc  r10d                       ;i++;
    add  edi, 2                     ;rowScreen = rowScreen + 2;
jmp updateBoardP2_bucle_Row
updateBoardP2_Cur:
mov edi, __DWORD[rdx+0]__  ;edi = rcCurScreen[0];
mov esi, __DWORD[rdx+4]__  ;esi = rcCurScreen[1];
call gotoxyP2           ;gotoxyP2_C(rcCurScreen[0],rcCurScreen[1]);
updateBoardP2_End:
...
mov rsp, rbp
pop rbp
ret
```

Examen 2017/18-2

Asignatura	Código	Fecha	Hora inicio
Estructura de computadores	75.573	16/06/2018	12:00

Pregunta 2

2.1

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

R0 = 00000A10h R1 = 00000B20h R2 = 00000C30h	M(00000A10h) = 0000F00Fh M(00000B20h) = 0000F000h M(00000C30h) = 00000FF0h M(000000F0h) = 00000001h M(000FF0A0h) = 0000000Ah	Z = 0, C = 0, S = 0, V = 0

¿Cuál será el estado del computador después de ejecutar cada fragmento de código? (sólo modificaciones, excluyendo el PC).

a)

```
        XOR   R0, R0
        JE    END
        ADD   R0, [R2]
END:    MOV   R1, [R2]
```

R0 = 0
R1 = 00000FF0h

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

b)

```
        DEC  [000000F0h]
        SUB  [000000F0h], R2
```

M(000000F0h) = 00000000h
M(000000F0h) = FFFFF3D0h

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

Examen 2017/18-2

Asignatura	Código	Fecha	Hora inicio
Estructura de computadores	75.573	16/06/2018	12:00

2.2

Dado el siguiente lenguaje de alto nivel:

```
Desde i=0 hasta 9  hacer
        Si (V [i] > 10) entonces V[i]= V[i]*2
        sino V[i] = 10;
fdesde;
```

Donde V se trata de un vector de 10 enteros de 32 bits. Se propone el siguiente código CISCA en el que tenéis que rellenar los 5 huecos para que sea operativo:

```
        MOV  R1, 0
cont:   CMP  R1, 40
        JGE  final
        CMP  [V+R1], 10
        JLE  m
        SAL  [V+R1], 1
        JMP  p
m:      MOV  [V+R1], 10
p:      ADD  R1, 4
        JMP  cont
final:
```

Examen 2017/18-2

Asignatura	Código	Fecha	Hora inicio
Estructura de computadores	75.573	16/06/2018	12:00

2.3

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

```
FOREVER:   MUL R10, [Q]
           MOV R1,[M+R2]
           JMP FOREVER
```

Traducidlo a lenguaje máquina y expresadlo en la siguiente tabla. Suponed que la primera instrucción del código se ensambla a partir de la dirección **003FC000h** (que es el valor del PC antes de empezar la ejecución del fragmento de código). Suponed que las direcciones simbólicas Q y M valen **00003A00h** y **00003C00h** respectivamente. 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 'Dirección' 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
22h	MUL
10h	MOV
40h	JMP

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 2017/18-2

Asignatura	**Código**	**Fecha**	**Hora inicio**
Estructura de computadores	75.573	16/06/2018	12:00

		Bk para k=0..10										
Dirección	**Ensamblador**	0	1	2	3	4	5	6	7	8	9	10
003FC000	MUL R10, [Q]	22	1A	20	00	3A	00	00				
003FC007	MOV R1,[M+R2]	10	11	52	00	3C	00	00				
003FC00E	JMP FOREVER	40	00	00	C0	3F	00					
003FC014												

Examen 2017/18-2

Asignatura	Código	Fecha	Hora inicio
Estructura de computadores	75.573	16/06/2018	12:00

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 utiliza una **política de emplazamiento completamente asociativa**, de manera que cualquier bloque de la memoria principal se puede llevar a cualquier bloque de la memòria 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:

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 bloc.

Línea	Estado Inicial	0		1		2		12		62	
0	0 (0 - 7)	E	0 (0 - 7)	E	0 (0 - 7)	E	0 (0 - 7)		0 (0 - 7)		0 (0 - 7)
1	1 (8 - 15)		1 (8 - 15)		1 (8 - 15)		1 (8 - 15)	E	1 (8 - 15)		1 (8 - 15)
2	2 (16 - 23)		2 (16 - 23)		2 (16 - 23)		2 (16 - 23)		2 (16 - 23)		2 (16 - 23)
3	7 (56 - 63)		7 (56 - 63)		7 (56 - 63)		7 (56 - 63)		7 (56 - 63)	E	7 (56 - 63)

Línea	63	25	64	17	18	19

Examen 2017/18-2

Asignatura	Código	Fecha	Hora inicio
Estructura de computadores	75.573	16/06/2018	12:00

0		0 (0 - 7)		0 (0 - 7)	F	8 (64 – 71)		8 (64 – 71)		8 (64 – 71)		8 (64 – 71)
1		1 (8 - 15)		1 (8 - 15)		1 (8 - 15)	F	2 (16 - 23)	E	2 (16 - 23)	E	2 (16 - 23)
2		2 (16 - 23)	F	3 (24 - 31)		3 (24 - 31)		3 (24 - 31)		3 (24 - 31)		3 (24 - 31)
3	E	7 (56 - 63)		7 (56 - 63)		7 (56 - 63)		7 (56 - 63)		7 (56 - 63)		7 (56 - 63)

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

Línea	56		42		50							
0		8 (64 – 71)		8 (64 – 71)	F	6 (48 – 55)						
1		2 (16 - 23)		2 (16 - 23)		2 (16 - 23)						
2	F	7 (56 - 63)		7 (56 - 63)		7 (56 - 63)						
3		0 (0 - 7)	F	5 (40 - 47)		5 (40 - 47)						

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

T_a = 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_a), 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)?

$t_m = T_a \times t_a + (1-T_a) \times t_f$ = 0,65 * 5 ns + 0,35 * 40 ns = 3,25 ns + 14 ns = 17,25 ns

Examen 2017/18-2

Asignatura	Código	Fecha	Hora inicio
Estructura de computadores	75.573	16/06/2018	12:00

3.2 Sistema d'E/S

Se quiere analizar el rendimiento de la comunicación de datos entre una memoria de un procesador y un puerto USB, que tienen las siguientes características, utilizando E/S por interrupciones:

- Velocidad de transferencia del dispositivo de E/S ($\mathbf{v}_{transf}$)= 1 MBytes/s = 1000 Kbytes/s
- Tiempo de latencia media del dispositivo ($t_{latencia}$) = 0
- Direcciones de los **registros de datos** y **de estado** del controlador de E/S: 0B00h i 0B04h
- 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 1 GHz, el procesador puede ejecutar 4 instrucciones por ciclo de reloj.
- Transferencia de **lecture** desde puerto de E/S hacia la memoria
- Transferencia de N_{datos}=100.000 datos
- La medida de un dato es m_{dada} = 4 bytes
- El tiempo para atender la interrupción (t_{rec_int}) es de 2 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, [0B04h]
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
```

3.2.2 ¿Cuánto tiempo dura la transferencia del bloque de datos t_{transf_bloc}?

El tiempo de un ciclo, t_{ciclo} = 1 ns (nanosegundos)

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

Tiempo de ejecución de una instrucción, t_{instr}: t_{ciclo} / 4 = 0,250 ns

Tiempo de ejecución RSI, t_{rsi}: N_{rsi} * t_{instr} = 12 instr. * 0,250 ns = 3 ns

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

$t_{transf\_dada} = t_{rec\_int} + t_{rsi}$ = 2 + 3 = 5 ns

Examen 2017/18-2

Asignatura	Código	Fecha	Hora inicio
Estructura de computadores	75.573	16/06/2018	12:00

Número de interrupciones producidas (número total de datos, N_{datos}): 100000 interrupciones

Tiempo consumido en total por TODAS las interrupciones:

$t_{transf\_bloc} = t_{transf\_dada} * N_{dades}$ = 5 ns * 100000 interrupciones = 500000 ns = 0,5 ms (milisegundos)

3.2.3 ¿Cuál es el porcentaje de ocupación del procesador? Porcentaje que representa el tiempo de transferencia del bloc t_{transf_bloc} respecto al tiempo de transferencia del bloque por parte del periférico t_{bloc}

$t_{bloc} = t_{latencia} + (N_{datos} * t_{dato})$
$t_{latencia} = 0$
$N_{datos} = 100000$
$t_{dato} = m_{dato} / v_{transf}$ = 4 / 1000 Kbytes/s = 0,004 ms
t_{bloc} = 0 + (100000 * 0,004) ms = 400 ms
% ocupación = (t_{transf_bloc} * 100 / t_{bloc}) = (0,5 * 100) / 400 = 0,125%

Pregunta 4

4.1

¿Qué es una interrupción y cómo se activa?

Las interrupciones son el mecanismo mediante el cual un dispositivo externo al procesador puede interrumpir el programa que está ejecutando el procesador con el fin de ejecutar otro programa (una rutina de servicio a la interrupción o RSI) para dar servicio al dispositivo que ha producido la interrupción.
La petición de interrupción se efectúa activando alguna de las líneas de petición de las que dispone el procesador.

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.

Examen 2017/18-2

Asignatura	Código	Fecha	Hora inicio
Estructura de computadores	75.573	16/06/2018	12:00

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