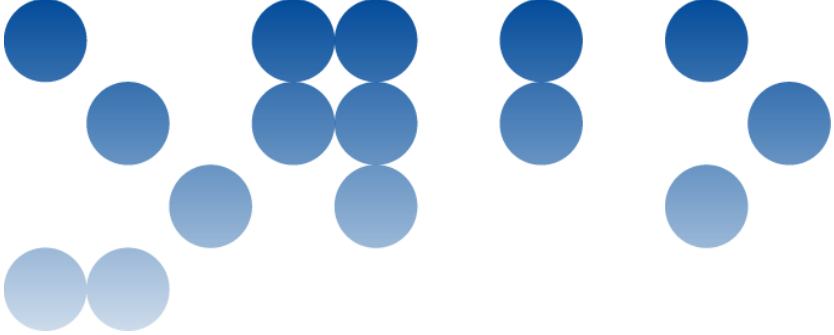

PEC2

Estructura de computadores

2019 s2

Estudios de informática, multimedia y comunicación

EIMT.UOC.EDU

UOC
Universitat Oberta de Catalunya

Presentación

La presente PEC2 contiene 2 preguntas y representa el 50% de la nota de la evaluación continua.

Como podréis ver, los ejercicios son muy parecidos a los que habéis hecho durante estos días, en los que además habéis podido dar las soluciones, comentarlas y plantear dudas en el foro. Esta PEC es **individual**, **evaluable** y por tanto no puede comentarse.

Competencias

Las competencias específicas que persigue la PEC2 son:

- [13] Capacidad para identificar los elementos de la estructura y los principios de funcionamiento de un ordenador.
- [14] Capacidad para analizar la arquitectura y organización de los sistemas y aplicaciones informáticos en red.
- [15] Conocer las tecnologías de comunicaciones actuales y emergentes y saberlas aplicar convenientemente para diseñar y desarrollar soluciones basadas en sistemas y tecnologías de la información.

Objetivos

Los objetivos de la siguiente PEC son:

- Conocer la organización del sistema de memoria de un computador.
- Conocer el funcionamiento de la memoria cache, así como los algoritmos de correspondencia y reemplazamiento.
- Conocer la organización del sistema de entrada/salida.
- Comprender las técnicas de entrada/salida (entrada/salida programada, Interrupciones y DMA).

Enunciado

Responder cada pregunta o apartado en el recuadro correspondiente.

Recursos

Podéis consultar los recursos disponibles en el aula, pero no hacer uso del foro.

Criterios de valoración

La **puntuación** y los **criterios de evaluación** los encontraréis en cada pregunta.

Formato y fecha de entrega

La PEC1 podéis entregarla en el apartado de **entrega de actividades** con el nombre **apellido1_apellido2_nombre_PEC2 (pdf / odt / doc / docx).**

La fecha **límite** de entrega es el **01/05/2020.**

Enunciado

Pregunta 1 (4 puntos)

Tenemos un sistema de memoria en el que todos los accesos se hacen a palabra (no nos importa cuál es el tamaño de una palabra). Supondremos que el espacio de direcciones de memoria se descompone en bloques de 8 palabras. Cada bloque comienza 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.

Una fórmula para calcular el identificador numérico del bloque es la siguiente:

Bloque = dirección de memoria (dirección palabra) DIV 8 (tamaño del bloque en palabras)

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 trae todo el bloque correspondiente desde la memoria principal a una línea de la memoria cache (así si hacemos referencia a dirección 2 de memoria principal traeremos el bloque formado por las palabras 0, 1, 2, 3, 4, 5, 6 y 7).

Apartado 1.1 (2 puntos) 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 a una línea determinada de la memoria cache. En este caso, el identificador del bloque determina la línea específica donde se puede guardar utilizando la siguiente fórmula (similar a la fórmula para determinar el bloque):

Línea = identificador de bloque MOD 4 (tamaño de la cache en líneas)

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

28, 29, 30, 2, 3, 15, 16, 1, 44, 45, 46, 12, 33, 8, 3, 17, 39, 60, 4, 35

1.1.a) (1,2 puntos)

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 hay que llenar una columna indicando si se trata de un acierto o de un fallo.

Si es un acierto escribiremos E en la línea correspondiente delante de la direcciones del bloque, si es un fallo escribiremos F y en ese caso se indicará el nuevo bloque que se trae a la memoria cache a la línea que corresponda, expresado de la forma b:e(a_0 – a_7), donde b: número de bloque, e: etiqueta y (a_0 – a_7) son las direcciones del bloque, donde a_0 es la primera dirección i a_7 es la octava (última) dirección del bloque.

Línea	Estado Inicial	28	29	30	2	3
0	0:0 (0 - 7)	0:0 (0 - 7)	0:0 (0 - 7)	0:0 (0 - 7)	E 0:0 (0 - 7)	E 0:0 (0 - 7)
1	1:0 (8 - 15)	1:0 (8 - 15)	1:0 (8 - 15)	1:0 (8 - 15)	1:0 (8 - 15)	1:0 (8 - 15)
2	2:0 (16 - 23)	2:0 (16 - 23)	2:0 (16 - 23)	2:0 (16 - 23)	2:0 (16 - 23)	2:0 (16 - 23)
3	3:0 (24 - 31)	E 3:0 (24 - 31)	E 3:0 (24 - 31)	E 3:0 (24 - 31)	3:0 (24 - 31)	3:0 (24 - 31)

Línea	15	16	1	44	45	46
0	0:0 (0 - 7)	0:0 (0 - 7)	E 0:0 (0 - 7)	0:0 (0 - 7)	0:0 (0 - 7)	0:0 (0 - 7)
1	E 1:0 (8 - 15)	1:0 (8 - 15)	1:0 (8 - 15)	F 5:1 (40 - 47)	E 5:1 (40 - 47)	E 5:1 (40 - 47)
2	2:0 (16 - 23)	E 2:0 (16 - 23)	2:0 (16 - 23)	2:0 (16 - 23)	2:0 (16 - 23)	2:0 (16 - 23)
3	3:0 (24 - 31)	3:0 (24 - 31)	3:0 (24 - 31)	3:0 (24 - 31)	3:0 (24 - 31)	3:0 (24 - 31)

Línea	12	33	8	3	17	39
0	0:0 (0 - 7)	F 4:1 (32 - 39)	4:1 (32 - 39)	F 0:0 (0 - 7)	0:0 (0 - 7)	F 4:1 (32 - 39)
1	F 1:0 (8 - 15)	1:0 (8 - 15)	E 1:0 (8 - 15)	1:0 (8 - 15)	1:0 (8 - 15)	1:0 (8 - 15)
2	2:0 (16 - 23)	2:0 (16 - 23)	2:0 (16 - 23)	2:0 (16 - 23)	E 2:0 (16 - 23)	2:0 (16 - 23)
3	3:0 (24 - 31)	3:0 (24 - 31)	3:0 (24 - 31)	3:0 (24 - 31)	3:0 (24 - 31)	3:0 (24 - 31)

Línea	60	4	35			
0	4:1 (32 - 39)	F 0:0 (0 - 7)	F 4:1 (32 - 39)			
1	1:0 (8 - 15)	1:0 (8 - 15)	1:0 (8 - 15)			
2	2:0 (16 - 23)	2:0 (16 - 23)	2:0 (16 - 23)			
3	F 7:1 (56 - 63)	7:1 (56 - 63)	7:1 (56 - 63)			

1.1.b) (0,4 puntos)

¿Cuál es la tasa de fallos (T_f)?

T_f = 8 fallos / 20 accesos = 0,4

1.1.c) (0,4 puntos)

Suponemos que el tiempo de acceso a la memoria cache, o tiempo de acceso en caso de acierto (t_a), es de 4 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 medio de acceso a memoria (t_m)?

$t_m = T_f \times t_f + (1-T_f) \times t_e = 0,4 * 20 \text{ ns} + 0,6 * 4 \text{ ns} = 8 \text{ ns} + 2,4 \text{ ns} = 10,4 \text{ ns}$

Apartado 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 misma lista de lecturas a memoria:

28, 29, 30, 2, 3, 15, 16, 1, 44, 45, 46, 12, 33, 8, 3, 17, 39, 60, 4, 35

1.2.a) (1,2 puntos)

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 hay que llenar una columna indicando si se trata de un acierto o de un fallo.

Si es un acierto escribiremos E en la línea correspondiente delante de las direcciones del bloque, si es un fallo escribiremos F y se indicará el nuevo bloque que se traerá a la memoria cache a la línea que le corresponda, expresado 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 y a_7 es la octava (última) dirección del bloque.

Línea	Estado Inicial	28	29	30	2	3
0	0 (0 - 7)	0 (0 - 7)	0 (0 - 7)	0 (0 - 7)	E 0 (0 - 7)	E 0 (0 - 7)
1	1 (8 - 15)	1 (8 - 15)	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)	2 (16 - 23)
3	3 (24 - 31)	E 3 (24 - 31)	E 3 (24 - 31)	E 3 (24 - 31)	3 (24 - 31)	3 (24 - 31)

Línea	15	16	1	44	45	46
0	0 (0 - 7)	0 (0 - 7)	E 0 (0 - 7)	0 (0 - 7)	0 (0 - 7)	0 (0 - 7)
1	E 1 (8 - 15)	1 (8 - 15)	1 (8 - 15)	1 (8 - 15)	1 (8 - 15)	1 (8 - 15)
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)	F 5 (40 - 47)	E 5 (40 - 47)	E 5 (40 - 47)

Línea	12	33	8	3	17	39
0	0 (0 - 7)	0 (0 - 7)	0 (0 - 7)	E 0 (0 - 7)	0 (0 - 7)	0 (0 - 7)
1	E 1 (8 - 15)	1 (8 - 15)	E 1 (8 - 15)	1 (8 - 15)	1 (8 - 15)	1 (8 - 15)
2	2 (16 - 23)	F 4 (32 - 39)	4 (32 - 39)	4 (32 - 39)	4 (32 - 39)	E 4 (32 - 39)
3	5 (40 - 47)	5 (40 - 47)	5 (40 - 47)	5 (40 - 47)	F 2 (16 - 23)	2 (16 - 23)

Línea	60		4		35											
0		0 (0 - 7)	E	0 (0 - 7)		0 (0 - 7)										
1	F	7 (56 - 63)		7 (56 - 63)		7 (56 - 63)										
2		4 (32 - 39)		4 (32 - 39)	E	4 (32 - 39)										
3		2 (16 - 23)		2 (16 - 23)		2 (16 - 23)										

1.2.b) (0,4 puntos)
¿Cuál es la tasa de fallos (T_f)?

T_f = 4 fallos / 20 accesos = 0,2

1.2.b) (0,4 puntos)
Suponemos que el tiempo de acceso a la memoria cache, o tiempo de acceso en caso de acierto (t_a), es de 4 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 medio de acceso a memoria (t_m)?

$t_m = T_f \times t_f + (1-T_f) \times t_e = 0{,}2 * 20 \text{ ns} + 0{,}8 * 4 \text{ ns} = 4 \text{ ns} + 3{,}2 \text{ ns} = 7{,}2 \text{ ns}$

Criterios de valoración. Para los apartados 1.1.a y 1.2.a cada error en los fallos o aciertos de la memoria cache o en la colocación de un bloque en la cache resta 0,6. Los apartados restantes se puntuarán con los 0,4 puntos cada uno de ellos si la solución es correcta y coherente con vuestra respuesta a los apartados a) correspondientes.

Pregunta 2 (5 puntos)

Se desea analizar el rendimiento de la comunicación de datos entre la memoria de un procesador y un port USB con un dispositivo conectado, que tienen las siguientes características:

- Velocidad de transferencia del dispositivo de E/S ($\mathbf{v}_{transf}$)= 8MB/s
- Tiempo de latencia promedio del dispositivo ($t_{latencia}$) = 0
- Direcciones de los **registros de datos y de estado** del controlador de E/S: 0400h y 0404h
- 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)
- Procesador con una frecuencia de reloj de 1 GHz, y el procesador puede ejecutar 1 instrucción cada dos ciclos de reloj (t_{instr}= 2*t_{ciclo})
- Tiempo de programación y finalización de la transferencia de 1000 ns (t_{prog} + t_{final})
- Transferencia de **escritura** desde memoria al port de E/S
- En cada escritura de un dato se transfieren 4 Bytes
- Transferencia de N_{datos}= 800.000 datos, es decir, 800.000 × 4 Bytes = 3.200.000 Bytes
- Dirección inicial de memoria donde residen los datos: 80000000h

Apartado 2.1 (2 puntos) E/S programada

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

```
1.              MOV   R3, VALOR1
2.              MOV   R2, VALOR2
3. Bucle:       IN    R0, [VALOR3]      ; leer 4 bytes
4.              AND   R0, VALOR4
5.              JE    Bucle
6.              MOV   R0,[R2]           ; leer 4 bytes
7.              ADD   R2, VALOR5
8.              OUT   [VALOR6], R0      ; escribir 4 bytes
9.              SUB   R3, 1
10.             JNE   Bucle
```

2.1.a) (0,5 puntos)

Substituir por los valores apropiados:

VALOR1= 800000 **VALOR2**= 80000000h

VALOR3= 0404h **VALOR4**= 00001000b – 08h = 8d

VALOR5= 4 **VALOR6**= 0400h

2.1.b) (0,5 punto)

¿Cuánto tiempo dura la transferencia?
¿Qué porcentaje de este tiempo dedica la CPU a la transferencia?

$t_{trnsf\_bloque} = t_{latencia} + (N_{datos} * t_{tranf\_dato})$

$t_{latencia} = 0$

$N_{datos} = 800000$

$t_{transf\_dato} = M_{dato} / V_{transf} = 4Bytes / 8\ Mbytes/s = 0{,}0005ms$

$t_{transf\_bloque} = 0 + (800000 * 0{,}0005ms) = 400ms = 0{,}4s$

La CPU dedica el 100% del tiempo y, por lo tanto, el tiempo coincide con el tiempo dedicado por el periférico t_{bloc}.

2.1.c) (1 punto)

Si quisiéramos utilizar el mismo procesador y el mismo programa, pero con un dispositivo de E/S más rápido, ¿Cuál es la máxima tasa o velocidad de transferencia del nuevo dispositivo que se podría suportar sin que el dispositivo se tuviera que esperar?

Frecuencia de reloj = 1 GHz.

t_{ciclo} = 1 / (1 * 10^9) = 1ns.

t_{instr} = 2 * 1 = 2 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 * 2 ns = 16 ns.

Se pueden transferir 4 bytes cada 8 ns, es decir: 4 / (16 * 10^{-9}) = 250 MByte/s.

Apartado 2.2 (2 puntos) E/S por Interrupciones

Suponer que el siguiente código CISCA 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:

- El tiempo para atender la interrupción (t_{rec_int}), o tiempo adicional desde que la CPU detecta la interrupción hasta que comienza a ejecutarse la primera instrucción de la RSI es de 10 ciclos de reloj.
- 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.    IN    R0, [VALOR1] ; leer 4 bytes
5.    AND   R0, VALOR2
6.    JE    Error        ; salta a un código de tratamiento de error no
                         ; descrito, se ha producido la petición por parte
                         ; del dispositivo, pero el dato no está disponible.
7.    MOV   R1, [VALOR3]
8.    MOV   R0, [R1]
9.    OUT   [VALOR4], R0 ; escribir 4 bytes
10.   ADD   R1, VALOR5
11.   MOV   [VALOR3], R1
12.   POP   R1
13.   POP   R0
14.   STI
15.   RETI
```

2.2.a) (0,5 puntos)
Sustituir por los valores adecuados:

VALOR1= 0404h

VALOR2= 00001000b - 08h = 8d

VALOR3= Dir

VALOR4= 400h

VALOR5= 4

2.2.b) (1 punto)
¿Cuál es el tiempo total que dedica la CPU a la tarea de Entrada/Salida, t_{cpu}? ¿Cuál es el porcentaje que representa el tiempo de transferencia del bloque t_{transf_bloque} con respecto al tiempo de transferencia del bloque por parte del periférico t_{bloque}?

El tiempo de un ciclo, t_{ciclo}, es 1 nanosegundos.

Tiempo para atender la interrupción, t_{rec_int} : 10 ciclos × 1 ns (t_{ciclo}) = 10 ns.

Tiempo de ejecución de una instrucción, t_{instr} : 2 ns .

Tiempo de ejecución RSI, t_{rsi}: $N_{rsi} * t_{instr}$ = 15 instr. * 2 ns = 30 ns.

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

$t_{transf\_dato} = t_{rec\_int} + t_{rsi}$ = 10 + 30 = 40 ns.

Número de interrupciones producidas, N_{datos}): 800.000 interrupciones.

Tiempo consumido en total en todas las interrupciones:

$t_{transf\_bloque} = t_{transf\_dato} * N_{datos}$ = 40 ns × 800.000 = 32 ms

El tiempo final de ocupación de la CPU debe incluir el tiempo de programación y finalización de la transferencia:

$t_{cpu} = (t_{prog} + t_{final}) + t_{transf\_bloque}$ = 1000 ns + 20.000.000 ns = 32.001.000 ns = 32,001 ms

De los 0,4s = 400ms de tiempo total para realizar la transferencia, la CPU está dedicada a la tarea de E/S:

%ocupación = $t_{transf\_bloque} * 100 / t_{bloque}$ = 32 * 100 / 400 => 8 % del tiempo.

2.2.c) (0,5 puntos)
Si quisiéramos reducir la frecuencia de reloj del procesador para reducir su consumo energético, hasta qué frecuencia lo podríamos hacer sin reducir la velocidad de transferencia con el dispositivo de 'E/S?

En la fase de transferencia de datos, el controlador de E/S genera 800.000 interrupciones durante 0,4 segundos.

$N_{datos} * t_{dato}$ = 0,4 s = 400 ms = 400.000 us (microsegundos).

Es decir, tenemos una interrupción cada 400.000 / 800.000 = 0,5 us. Este es el tiempo máximo que debería tardar la gestión de la interrupción, incluyendo el tiempo adicional para transferir el control a la RSI, el tiempo que puede consumir la CPU en una interrupción es t_{transf_dato}.

El tiempo consumido por la CPU en cada interrupción es la suma de los tiempos de transferir el control a la RSI + ejecutar la RSI:

t_{rec_int} = 10 ciclos de reloj = 10 * t_{ciclo}

Por lo tanto:

$t_{transf\_dato} = t_{rec\_int} + t_{rsi} = t_{rec\_int} + (N_{rsi} * t_{instr}) = 10 * t_{ciclo} + (15 * t_{instr})$

El tiempo de una instrucción es: $t_{instr} = t_{ciclo} * 2$

Por lo tanto: $t_{transf\_dato} = 10 * \underline{t_{ciclo}} + (15 * (t_{ciclo} * 2)) = 40 * t_{ciclo}$

Queremos encontrar el tiempo de ciclo tal que el tiempo de transferencia de un dato sea 0.5 us:

0.5 us = 40 * t_{ciclo} => t_{ciclo} = 0.5 / 40 = 0,0125 us.
1 / 0.0125 * 10^{-6} = 80 Mhz

Apartado 2.3 (1 punto) E/S por DMA

Supondremos que el controlador de E/S puede funcionar en modo DMA (Acceso Directo a Memoria). La suma del **tiempo de cesión** del bus y del **tiempo de recuperación** del bus es de 40 ns ($t_{cesión}$ + t_{recup}= 40 ns). El **tiempo de la transferencia** por el bus es de 1 ns (t_{mem} = 1 ns).

2.3.a) (0,5 puntos)
Consideremos que en la transferencia por DMA, los datos se envían entre el controlador de DMA y la memoria, en modo ráfaga, i dispone de un buffer de medida m_{buffer} = 1600 bytes. Calcular el tiempo total de ocupación del bus por parte del controlador de DMA para llevar a cabo la transferencia que venimos analizando.

Medida de les ráfagas $N_{ráfaga}$: m_{buffer} / m_{dato} = 1600 / 4 = 400

Tiempo ocupación Bus, $t_{transf\_ráfaga}$: $t_{cesión}$ + 400 × t_{mem} + t_{recup} = 40 + 400 * 1 = 440 ns

Número de peticiones del Bus, N_{datos} / $N_{ráfaga}$: 800.000 / 400 = 2.000

Tiempo total de ocupación del Bus t_{transf_bloque} :
$t_{transf\_ráfaga}$ × N_{datos} / $N_{ráfaga}$ = 440 * 2000 = 880000 ns = 880 us = 0,88 ms

2.3.b) (0,5 puntos)
La CPU no puede hacer ninguna tarea durante todo el tiempo en que el bus está ocupado por parte del controlador de DMA. ¿Cuál es el porcentaje de tiempo que tiene disponible la CPU para ejecutar código efectivo de otros programas durante la transferencia?

t_{transf_bloque} = 0,88 ms

t_{bloque} = 400 ms

%ocupación = (t_{transf_bloque} · 100) / t_{bloque}

Porcentaje de tiempo disponible:
100 – % ocupación = 100 - (t_{transf_bloque} * 100) / t_{bloque} =
100 – (0,88 * 100) / 400 = 100 - 0,22 = 99, 78%

Criterios de valoración. En los apartados 2.1.a y 2.2.a cada valor erróneo resta 0,25. El resto de los apartados están bien o están mal. No hay gradación.

Pregunta 3 (1 punto)

Preguntas teóricas

3.a) (0,25 puntos)
¿Por qué el sistema de memoria se organiza en una estructura jerárquica? ¿Cuáles son los niveles de esta estructura?

Se organiza de esta manera porque no hay ninguna tecnología que sea lo suficientemente barata y rápida para obtener un tiempo de acceso a memoria reducido ya que, a mayor es la velocidad de acceso, mayor es su coste. Por este motivo se ha decidido organizarlo de forma jerárquica, para obtener un mayor rendimiento a un precio más reducido.

Los niveles de este tipo de estructura son, de más rápido a más lento y de más pequeña a más grande: registros del procesador, memoria cache, memoria principal y memoria secundaria. Los niveles más rápidos de memoria se encuentran más cerca del procesador y los más lentos se encuentran más alejados.

El procesador solo es capaz de leer los datos desde los registros del procesador y obtiene los datos de memoria desde la memoria cache, que funciona como una memoria intermedia entre los registros del procesador y la memoria principal y secundaria del computador. Si el dato no está disponible en la memoria cache, el procesador pedirá el dato a la memoria en la que esté disponible. Este dato se almacenará en la memoria cache para después poder ser guardada en la memoria cache y más adelante en los registros del procesador.

3.b) (0,25 puntos)
En la memoria caché, ¿Cuándo debe utilizarse un algoritmo de reemplazo? ¿Cuáles son los principales algoritmos de reemplazo?

Los algoritmos de reemplazo se utilizan cunado debemos guardar un bloque de memoria en la memoria cache. La labor de estos algoritmos es decidir qué bloque de memoria va a ser reemplazado por el nuevo bloque de memoria que queremos guardar. Esta acción se produce cuando se ha producido un fallo de acceso a la memoria cache.

Actualmente disponemos de cuatro tipos de algoritmos de reemplazo: FIFO, LFU LRU y de tipo aleatorio.

- El algoritmo de reemplazo de tipo FIFO (*first in first out*) se utiliza una cola para que el bloque que lleva más tiempo sin ser utilizado sea el que sea reemplazado. Este algoritmo no garantiza un buen rendimiento porque el que más tiempo lleva sin ser utilizado no tiene por que ser el menos necesario.
- El algoritmo LFU (*least frequently used*) reemplaza el bloque de memoria que menos se ha utilizado. Para ello, se puede utilizar un contador.

- El algoritmo LRU (*least recently used*) reemplaza la línea que más tiempo lleva sin ser utilizada. Es el algoritmo con mejor rendimiento, pero el más difícil de implementar.

- El algoritmo aleatorio reemplaza una línea de la memoria cache al azar. Es un algoritmo más fácil de implementar, pero con un rendimiento un poco inferior a los anteriores

3.c) (0,25 puntos)
¿Cuáles son las tres partes básicas de un módulo de E/S? ¿Qué tipos de registros incluye un módulo de E/S?

Las partes que componen un módulo de E/S son:

- Una interfaz interna que nos da acceso al banco de registros del procesador mediante el bus de sistema.
- Una interfaz externa, que controla el periférico que se desea conectar con el computador y que, normalmente, se conecta mediante un sistema de conexión normalizado de E/S.
- La lógica necesaria para la gestión del módulo de E/S y que se ocupa del paso de información entrada de la interfaz interna y la externa.

Los módulos de E/S disponen de tres tipos de registros:

- Los registros de control, que guardan las señales de control que se utilizan para dar órdenes a los módulos de E/S. Estas señales se reciben desde las líneas de control o desde las líneas de datos del bus del sistema.
- Los registros de estado, que guarda las señales de estado del módulo de E/S y se actualizan mediante la lógica del módulo de E/S.
- Los registros de datos que guardan la información que se desea intercambiar entre el módulo de E/S y procesador.

3.d) (0,25 puntos)
En un sistema de E/S gestionada por DMA ¿Qué diferencia hay entre el funcionamiento normal y el modo ráfaga, qué ventajas tiene un modo con respecto al otro?

En el modo de ráfaga, se solicita y libera el bus de datos para transferir un conjunto de datos de forma consecutiva en vez de solicitar y liberar el bus para cada dato que se desea transferir. De esta manera, se reduce el número de cesiones y recuperaciones del bus.

Para poder realizar la transferencia del conjunto de datos, se utiliza una memoria intermedia (*buffer*) para que la velocidad de transferencia sea la máxima que permita la memoria para que la velocidad del periférico no limite la transferencia. Con este modo de funcionamiento se modifica la transferencia de datos, pero no la se modifica la programación ni la finalización de la operación de E/S.