Práctico 3 - Procesador con Excepciones

Ejercicio 4

Analizar el comportamiento del procesador con manejo de excepciones implementado y responder con V (verdadero) o F (falso) a las siguientes afirmaciones. En caso de ser falsa, elaborar la respuesta correcta:

a) La ocurrencia de una excepción en un procesador causa que la ejecución secuencial del código se vea interrumpida necesariamente.
b) La diferencia entre un evento de interrupción y de excepción es que la primera es causada por un recurso del procesador, mientras que la segunda se trata de un evento proveniente de un controlador de E/S externo.
c) Los registros de excepción son utilizados típicamente en la ISR o ESR (Interrupt /Exception Service Routine) para poder procesar debidamente el evento ocurrido.
d) La dirección donde se aloja la ISR (vector de interrupciones) es fija sólo si el sistema posee solo un módulo de E/S que genera interrupción.
e) El registro ESR (Exception Syndrome Register) contiene una referencia a la dirección de memoria de la instrucción en ejecución al momento de la ocurrencia de la excepción.
f) Si un procesador no reconoce una instrucción se genera un evento de excepción.
g) Que una fuente de interrupción externa sea "enmascarable" significa que el procesador puede retrasar su ejecución si tiene eventos de mayor prioridad pendientes.
h) La instrucción ERET (Exception Return) retorna a la posición del PC (Program Counter) al momento de la excepción más cuatro (PC excepción + 4).

a. **Verdadero**, aunque luego de manejar la excepción, se puede reanudar la ejecución del código.
b. **Falso**, las excepciones provienen del CPU, como por ejemplo un opcode invalido. Mientras que las interrupciones son causadas por los controladores de E/S.
c. **Verdadero**, los registros de excepción, a veces llamados registros de manejo de excepciones o registros de estado de excepción, son componentes importantes en la arquitectura de un sistema de cómputo. Estos registros son utilizados típicamente en las ISR (Interrupt Service Routines) o ESR (Exception Service Routines) para gestionar adecuadamente eventos inesperados o excepcionales que ocurren en un sistema informático.
d. **Falso**, no necesariamente debe ser fija.
e. **Falso**, el registro que almacena la dirección de memoria en la que se encuentra la instrucción que fue interrumpida, es el ELR (Exception Link Register). Mientras que, el ESR almacena la causa de la excepción.
f. **Verdadero**, por opcode invalido.
g. **Verdadero**, incluso el programador puede modificar bits, causando que el procesador ignore estas señales.
h. **Falso**, la instrucción ERET tiene la finalidad de retornar la dirección de la instrucción almacenada en ERR (no necesariamente es PC + 4).

Ejercicio 5

Considere que la siguiente sección de código está presente en el vector de excepciones del procesador implementado.

```
1>    exc_vector: MRS X9, S2_0_C2_C0_0
2>                CMP X9, 0x01
3>                B.EQ trap
4>                MRS X9, S2_0_C0_C0_0
5>                BR X9
6>    trap:       B trap
```

Seleccionar las respuestas correctas de las siguientes afirmaciones:

01. "Ante la ocurrencia de una excepción, el código queda atrapado en un lazo infinito..."
    a. ... solo si se trata de una excepción de OpCode invalido.
    b. ... solo si se trata de una excepción por interrupción externa.
    c. ... siempre, independientemente del tipo de excepción.
    d. Ninguna de las anteriores es correcta
02. "Si suponemos que la instrucción de la línea 2> está corrompida en memoria de forma permanente (generando un OpCode invalido), considerando la implementación particular de nuestro procesador, el mismo...":
    a. ... no realiza ninguna acción porque ya está en el vector de excepciones y retorna normalmente.
    b. ... queda atrapado en el bucle infinito del label "trap".
    c. ... genera un comportamiento impredecible, porque este caso no está contemplado en la lógica de excepciones.
    d. Ninguno de los anteriores es correcta
03. "Este código retorna a la dirección de memoria donde se encuentra ...":
    a. ... la instrucción que generó la excepción por OpCode invalido
    b. ... la siguiente instrucción que debía ejecutarse de no haberse producido la excepción por OpCode invalido.
    c. ... la instrucción que estaba en ejecución al generarse una excepción por interrupción externa.
    d. ... la siguiente instrucción que debía ejecutarse de no haberse producido una excepción por interrupción externa.

Analizemos un poco el codigo para entender mas detalladamente su funcionamiento :

```
1>    exc_vector :   MRS X9, S2_0_C2_C0_0
2>                   CMP X9, #0x01
3>                   B.EQ trap
4>                   MRS X9, C2_0_C0_C0_0
5>                   BR X9
6>    trap :         B trap
```

- La primera línea, copia el contenido del registro de sistema (ESR = 0010) a X9. Recordemos que el ESR indica quien generó la excepción.

- La segunda línea del código, básicamente está ahí para chequear si es una excepción externa (externa IRQ = 0001).

Aquí se abren dos posibles caminos :
→O bien lo que se cargó en X9, es una external IRQ, por lo que nos quedamos encerrados en Trap.
→ O por el otro lado, se carga en X9 la dirección que tiene el registro ERR (0000) quien indica cual es la dirección de la siguiente instrucción que se debería haber ejecutado sino hubiera saltado la excepción.

Además, en el caso del inciso “b”, cabe aclarar que nuestro micro está preparado para manejar excepciones por opcode invalido. Por lo tanto, ante la llegada de este tipo de excepción, el programa volverá al exc_vector advirtiendo de este evento. (se quedaría atrapado entre las líneas 1 y 2).

Ejercicio 6

Considere que la siguiente sección de código está presente en el vector de excepciones del procesador implementado.

```
1>   exc_vector:   MRS X9, S2_0_C2_C0_0
2>                 CMP x9, 0x01
3>                 B.NE end
4>                 MRS X10, S2_0_C1_C0_0
5>                 MOVZ X9, #0x8B1F, LSL #16
6>                 MOVK X9, #0x03FF, LSL #0
7>                 STURW W9, [X10, #0]
8>   end:          ERET
```

Seleccionar las respuestas correctas de las siguientes afirmaciones:

01. “Ante la ocurrencia de una excepción por OpCode invalido, este código... ”
    a. ... siempre retorna a la dirección de memoria de la instrucción que generó la excepción + 4.
    b. ... reemplaza la instrucción corrupta que generó la excepción por una instrucción válida.
    c. Ninguna es correcta
    d. Ambas son correctas.
02. “Si suponemos que la instrucción de la línea 3> está corrompida en memoria de forma permanente (generando un OpCode invalido), considerando la implementación particular de nuestro procesador, el mismo...”
    a. ... genera un comportamiento impredecible, porque este caso no está contemplado en la lógica de excepciones.
    b. ... queda atrapado en el bucle infinito.
    c. ... no realiza ninguna acción porque ya está en el vector de excepciones y retorna normalmente.
    d. Ninguna de los anteriores es correcta.
03. “Ante la ocurrencia de una excepción por interrupción externa, este código... ”
    a. ... no realiza ninguna acción.
    b. ... siempre retorna a la dirección de memoria de la instrucción que generó la excepción + 4.
    c. Ambas son correctas.
    d. Ninguna es correcta

Ejercicio 7

Considere que la siguiente sección de código está presente en el vector de excepciones del procesador implementado. En el caso de una excepción por OpCode invalido, este código deberá ejecutar un procedimiento alojado en la dirección 0x0400, usando X0 como argumento que contenga la dirección del OpCode invalido.

Completar el código con los argumentos faltantes:

```
esr_address:
      mrs x9,S2_0_C2_C0_0
      subis xzr, x9, #0x010
      b.ne esr_end
      mrs X0, s2_0_c1_c0_0
      add x10, x0, xzr
      movz x9, #0x0400, lsl #0
      br x9
esr_end: eret
```

Ejercicio 8

Considere que la siguiente sección de código está presente en el vector de excepciones de un microprocesador LEGv8 (ISA completa), con el mismo tratamiento de excepciones utilizado hasta el momento.

- En el caso de una excepción por **Interrupción externa (IRQ)**, este código deberá ejecutar la **ISR** alojada en la dirección con etiqueta "***isr_proc***". Una vez que se retorne de la ISR, se debe retomar el flujo original del programa previo a la ocurrencia de la interrupción.
- En caso de un **OpCode invalido** se debe reemplazar el contenido de la memoria que contiene la instrucción corrompida con el valor **0x8B1F03FF**. Luego se debe forzar la ejecución de este nuevo OpCode.
- Cualquier otra fuente de excepción el procesador debe quedar atrapado en un lazo infinito dentro del vector de excepciones.

```
exc_vector:     mrs x9, S2_0_C2_C0_0
                subis xzr, x9,#0x01
                b.ne jmp1
question1:      bl isr_proc
                eret
jmp1:           subis xzr, x9, #0x02
                b.ne exc_trap
                movz x9, 0x8b1f, lsl #16
                movk x9, #0x03FF, lsl #0
                mrs x10, s2_0_c1_c0_0
                sturw w9,[x10, #0]
                br x10
exc_trap:       b exc_trap
```

2. Seleccionar todas las respuestas correctas de las siguientes afirmaciones:
    a. Si suponemos que la posición de la etiqueta "*question1*" está corrompida en memoria de forma permanente (generando un OpCode invalido), considerando la implementación particular de nuestro procesador:
        i. Ante una excepción por IRQ queda atrapado en un bucle infinito.
        ii. Ante una excepción por OpCode Invalido queda atrapado en un bucle infinito.
        iii. Se genera un comportamiento impredecible ante cualquier excepción, porque este caso no está contemplado en la lógica de excepciones.
        iv. No se realiza ninguna acción porque ya está en el vector de excepciones y retorna normalmente.
        v. En cualquier caso se reemplaza el OpCode Invalido y se retorna al flujo original del programa, previo a la ocurrencia de la primera interrupción.
        vi. Ninguna es correcta, ya que la lógica dependerá del tipo de excepción.
    b. Ante la ocurrencia de una excepción por OpCode invalido, este código...
        i. no es posible para el procesador determinar la dirección de retorno para este contexto.
        ii. siempre queda atrapado en el lazo "*exc_trap*".
        iii. siempre retorna a la dirección de memoria de la instrucción que generó la excepción + 4.
        iv. siempre retorna a la dirección de memoria de la instrucción que generó la excepción.
        v. Ninguna es correcta

Respuestas :

a. La opción correcta es la "i".
b. La opción correcta es la "v".