Arquitectura de computadores
Práctica 3: Excepciones

3.1. OBJETIVO.

En esta práctica veremos el procesamiento de excepciones. El sistema de excepciones está pensado, básicamente, para facilitar tres funciones:

1. la atención a circunstancias externas que pueden surgir en instantes interdeterminados;
2. el manejo de situaciones incorrectas dentro de la ejecución normal de programas;
3. la posibilidad de establecer distintos contextos de programación.

3.2. PRÁCTICA

Las excepciones del microprocesador 68000 pueden clasificarse en tres grupos según su modo de procesamiento:

Grupo 0: 0.0 Reset
0.1 Errores de alineamiento de dirección (Address Error)
0.2 Errores de bus

Grupo 1: 1.0 Traza
1.1 Interrupciones
1.2 Instrucciones ilegales
1.3 Violaciones de privilegio

Grupo 2: 2.0 Instrucciones TRAP y TRAPV
2.1 Instruccion CHK
2.2 División por cero

Para el grupo 0 el procesamiento de la excepción se realiza de modo inmediato (dentro de los dos ciclos de reloj posteriores al momento de producirse la causa). Para el segundo grupo el procesamiento se produce antes de que se ejecute la siguiente instrucción a la causante de la excepción. El tercer grupo costa de las excepciones asociadas a la ejecución normal de las instrucciones por lo que su procesamiento se produce según lo requiere la instrucción.

A cada excepción se asocia un número (por ejemplo, para el *address error* es el nº 3). Este número tiene por objeto servir de índice en una tabla de direcciones de memoria que apuntan a los lugares donde se encuentran las rutinas de atención a las excepciones. Estos apuntadores se denominan vectores y por ello el número de orden se conoce como *número de vector*.

La tabla de vectores se encuentra en el primer kbyte del espacio de direccionamiento (000000-0003FF). A continuación se presenta el mapa de los vectores que nos interesan en esta práctica:

Número de vector	Dirección Decimal	Hexadec.	Asignación
2	8	000008	*Bus error*
3	12	00000C	*Address error*
4	16	000010	*Illegal instruction*
5	20	000014	*Zero divide*
6	24	000018	*CHK instruction*
7	28	00001C	*TRAPV instruction*
8	32	000020	*Privilege violation*
9	36	000024	*Trace*
10	40	000028	*Line 1010 emulation*
11	44	00002C	*Line 1111 emulation*
32-47	128 : 188	000080 : 0000BC	*Trap instruction vectors*

Como práctica:

Pueden verse los vectores de excepción del 68fil volcando a la pantalla la memoria en la dirección 0. Al hacer esto (68fil: V 0), se observará algo así como lo que sigue:

```
68fil: V 0
<000000> 4669 6C6F 536F 6674 00FF C100 00FF C112 FiloSoft........
<000010> 00FF C1E8 00FF C31E 00FF C324 00FF C32A ...........$...*
<000020> 00FF C330 00FF C23C 00FF C228 00FF C232 ...0...<...(...2
    ⋮                        ⋮                              ⋮
<0000F0> 00FF C336 00FF C336 00FF C336 00FF C336 ...6...6...6...6
```

De aquí obtenemos las direcciones de las rutinas asociadas con cada excepción; por ejemplo:

Error de bus	00FFC100
Error de dieccionamiento	00FFC112
Instrucción ilegal	00FFC1E8
División por cero	00FFC31E
Instrucción CHK	00FFC324
Instrucción TRAPV	00FFC32A
etc.	

Nota: los dos primeros vectores presentan una notable anomalía que quedará justifidada en la siguiente práctica ya que corresponden a la excepción externa "reset".

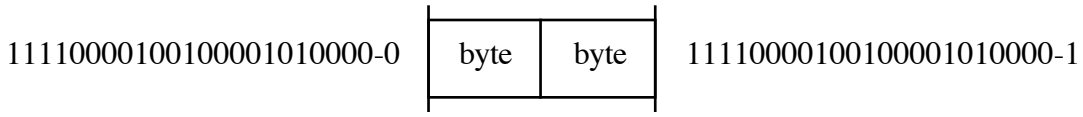
Otra nota: cabe la posibilidad de que los valores observados no sean exactamente los indicados aquí, ya que pueden depender de la versión de software que lleve el entrenador 68fil (los valores mostrados son los de la versión 1.2-195).

Ahora que ya sabemos cómo localizar las direcciones de las rutinas de atención a las excepciones podremos cambiar éstas por otras que nos permitan estudiar lo que sucede en el procesamiento de cada excepción.

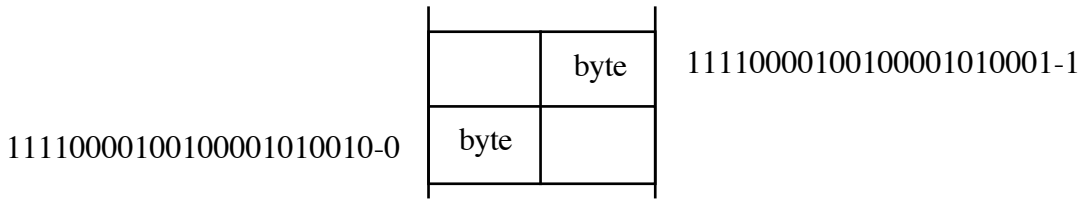
3.2.1. Excepciones del grupo 0: *Address Error*.

Dentro del grupo 0 de excepciones solo es objeto de esta práctica la asociada a errores en el alineamiento de las direcciones (*address error* o error de direccionamiento).

El 68000 accede a la memoria mediante un bus de 16 bits, lo que haría lógico que cada dirección hiciese referencia a una posición con un contenido de dicho tamaño. No obstante esto no es así ya que se conserva el direccionamiento por bytes (el byte es un tamaño de operando válido para el 68000). Esto tiene como consecuencia que el bus de direcciones sea de 23 bits en lugar de 24 ya que el bit más bajo (A0) no existe. Cada vez que el 68000 mueve a/desde memoria una word debe hacerlo con dirección par de manera que la word a la que se accede esté formada por dos bytes cuya dirección expresada en binario solo difiera en el inexistente bit A0.



Direccionamiento correcto. Los 32 bits son iguales



Direccionamiento incorrecto. Los 32 bits no son iguales

Como práctica:

Para comprobar lo anterior bastará con introducir en una posición de memoria la instrucción `CLR.W (A0)` y ejecutarla mediante el mando `T` para diferentes valores de `A0`.

Pruébese esto para una dirección válida, como por ejemplo 1002, y se observará que el funcionamiento de la instrucción es correcto. A continuación si se pone el registro `A0` a 1003 se obtendrá el siguiente resultado:

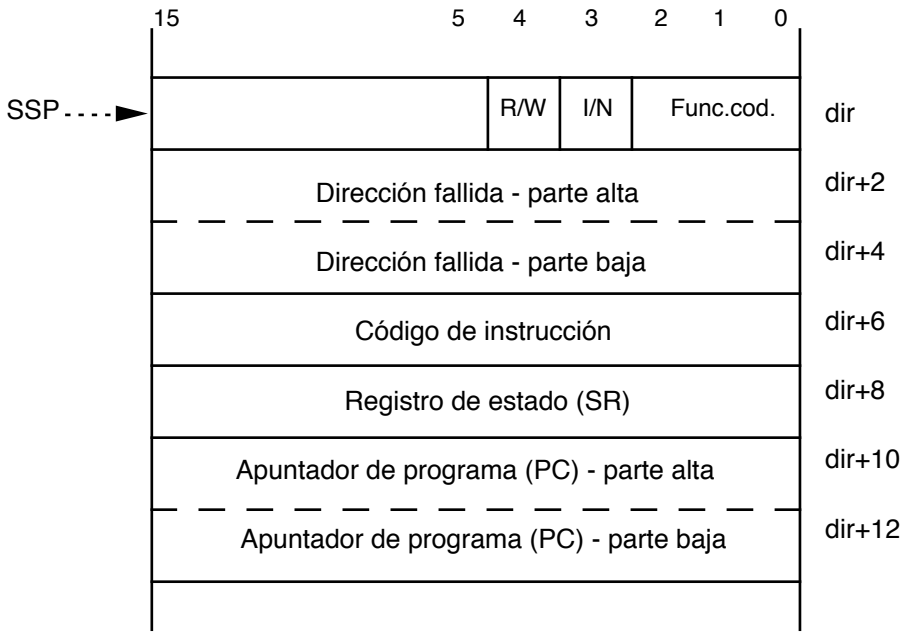
```
68fil: t 1000
Excepcion! Address error: acceso en lectura a la direccion 001003
Instruccion causante de la excepcion:
<001000> 4250                                    CLR.W    (A0)
```

donde vemos que se ha producido un error de direccionamiento

Cuando el microprocesador toma una instrucción y reconoce que se requiere un acceso a memoria mal alineado inicia un proceso de excepción que permite tomar las medidas oportunas. En el caso del programa monitor de la placa 68fil vemos que la acción que se realiza es la de mostrar un mensaje informando del problema. Esta información contiene datos como el tipo de acceso (lectura/escritura), dirección errónea e instrucción causante, que, como veremos enseguida, son proporcionados por el microprocesador.

Como práctica:

Vamos a comprobar a continuación la información que se almacena en la pila cuando se produce esta excepción. A continuación se muestra la estructura de la información que debe encontrarse.



R/W=0) ciclo de escritura (R/W=1) ciclo de lectura
(IN=0) instrucción (I/N=1) no instrucción.

Para descubrir esto haremos lo siguiente:

1. Reiniciar el entrenador mediante el mando `X` (o el pusador RESET).

2. Preparar una rutina de atención a la excepción que devuelva el control al monitor:

    2.1. Introducir en 2000 el siguiente código:

```
<002000> 4E4F 0000                  SERMON FPROG
```

    2.2. Establecer el vector 3 apuntando a 2000 mediante el mando `M`:

```
68fil: M C
 <00000C> 000F 0
 <00000E> C112 2000
 <000010> 00FF [ESC]
```

3. Ejecutar una instrucción que produzca un error de direccionamiento:

    3.1. Introducir en 1000 un `NOP` (`4E71`) y a continuación `MOVE.W (A0),D0` (`3010`).

    3.2. Introducir en el registro `A0` un valor impar.

    3.3.-Ejecutar a partir de la dirección 1000 con el mando `E`.

4. Obtener el valor del apuntador de la pila de supervisor (SSP):

```
68fil: R
 D0=00000000 D1=00000000 D2=00000000 D3=00000000 ! PC=00002004
 D4=00000000 D5=00000000 D6=00000000 D7=00000000 !USP=0000F800
 A0=00000000 A1=00000000 A2=00000000 A3=00000000 !SSP=0000F9F2
 A4=00000000 A5=00000000 A6=00000000     SR=2000  [.S0.....]
```

5. Mirar la pila mediante un volcado de memoria:

```
68fil: V F9F2 F9F2
<00F9F2> 3011 0000 0001 3010 0000 0000 1004 FFFF 0.....0.........
```

Queda como trabajo para el alumno comprobar el contenido de la pila de acuerdo con la estructura que se ha mostrado más arriba.

Cuestión:

En la práctica que se acaba de realizar se ha comprobado que en la pila, entre otras cosas se encuentra la dirección de retorno al programa para continuar la ejecución a partir de la siguiente instrucción a la que se provocó el "address error", pero ¿este valor apuntará siempre a la siguiente instrucción?. Compruébese lo que sucede cuando la instrucción causante de la excepción tiene words de extensión, como es el caso de:

```
MOVE.W   (A0),2(A1)
```

3.2.2. Error de bus (*Bus Error*).

La excepción de error de bus es causada por la activación de la línea BERR del 68000. En el entrenador 68fil esto ocurre cuando ha transcurrido demasiado tiempo desde el inicio de un ciclo de bus por parte del µP sin que éste haya finalizado.

Si un ciclo de bus no termina, la razón ha de ser forzosamente la falta de respuesta del dispositivo direccionado en forma de activación de la señal DTACK. El 68fil pone en marcha un contador de 8 bits (a una frecuencia de 8 MHz) cada vez que el 68000 selecciona un dispositivo durante un ciclo de bus; se supone que la selección se produce en el momento en que se activa cualquiera de las señales UDS o LDS. Si una vez transcurridos 32 µseg. no ha habido respuesta por parte de ningún dispositivo, se activa el bit más alto del contador, lo que a su vez provoca la activación de BERR.

La secuencia de procesamiento de la excepción de Bus Error es igual a la del error de direccionamiento (Address Error), salvo en el vector utilizado (el nº 2). La información que se almacena en la pila posee el mismo formato). Por las misma razones no es posible recuperar con seguridad el contexto de ejecución previo a la excepción.

Por ejemplo:

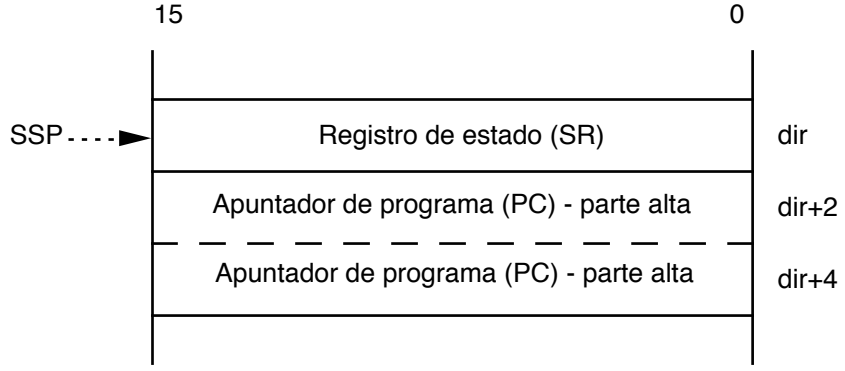
Una manera de provocar un error de bus consiste en realizar un ciclo de lectura o escritura en una dirección en la que no existe ningún dispositivo. Si consultamos el mapa de direccionamiento del 68fil (ver Práctica 0) podemos ver que en su mayor parte se encuentra sin ocupar. De modo que un acceso a cualquiera de las direcciones no asignadas (por ejemplo, la 030000_{16}) producirá una excepción de Bus Error. Ello se puede comprobar mediante la siguiente orden:

```
68fil : V 30000
```

En respuesta a este mandato veremos un mensaje indicando que se ha producido un error de bus. El mensaje es generado por la rutina de servicio proporcionada por el monitor para esta excepción. Toda la información del mensaje es extraída de los datos almacenados en la pila durante el procesamiento de la excepción de error de bus.

3.2.3. Excepciones del grupo 1: Traza, instrucción ilegal y violación de privilegio

Las excepciones del grupo 1 no se procesan en un punto medio de la ejecución de una instrucción, por lo que la información guardada en la pila es menor que la de las excepciones del grupo 0.



Hay que tener en cuenta que en el caso de instrucciones ilegales (incluyendo aquí las de emulación de línea 1010 y 1111) y las violaciones de privilegio no se puede saber donde se encuentra la siguiente instrucción por lo que el PC almacenado en la pila apunta a la misma instrucción que produce la excepción.

Como práctica:

Hágase una comprobación de la estructura almacenada en la pila como se ha hecho en el apartado anterior para el caso de una instrucción ilegal y de una violación de privilegio.

Obsérvese como la violación de privilegio no se produce cuando se establece el estado supervisor.

La excepción de traza resulta muy útil para depurar programas, ya que se produce tras la ejecución de cada instrucción. Recuérdese que durante el procesamiento de excepción se desactiva el bit de traza, impidiendo así que se continúe generando dicha excepción si antes estaba activado.En el 68fil la rutina de servicio de dicha excepción muestra por pantalla una información similar a la resultante de la orden R del monitor.

Como práctica:

Introdúzcanse en memoria un programa corto (unas pocas instrucciones) finalizado con un sermón FPROG. Mediante la orden R del monitor póngase a 1 solamente el bit T del SR, y ejecútese ese programa con la orden E, como es habitual. Obsérvese lo que aparece en pantalla.*

A continuación, escríbase una "rutina de servicio" para la excepción de traza que se limite a mostrar por pantalla el contenido del PC previo a la excepción (utilizando para ello los sermones ITOA y LNPUTS). Instálese diche rutina en el vector de la excepción de traza y ejecútese un programa cualquiera (por ejemplo, el mismo de antes).

Nota importante: es vital que la rutina de servicio de la excepción de traza no modifique ningún registro; recuérdese que el sermón ITOA modifica `A0`, y además necesita que se le pase el parámetro en `D7` (el cual, por lo tanto, es necesario modificar).

Obsérvese como la violación de privilegio no se produce cuando se establece el estado supervisor.

* No se observa nada. En versiones anteriores del monitor sí, pero en esta no.

3.2.4. Excepciones del grupo2: `CHK`, `TRAPV`, `TRAP` y división por cero.

Las excepciones del grupo 2 son aquellas que se asocian a instrucciones que permiten introducir dentro de su ejecución una rutina escrita por el usuario. Tomemos por ejemplo el caso de las instrucciones `DIVU` y `DIVS`; su función está claramente definida siempre y cuando el divisor sea distinto de cero, pero en tal caso (divisor=0) se permite que el usuario adopte una estrategia particular mediante la escritura de una rutina de servicio para la excepción asociada. De esta forma `DIVU` y `DIVS` tienen un comportamiento correcto en todo caso de acuerdo con el criterio del programador.

Dentro de este grupo tenemos tres excepciones que se ejecutan sólo al darse determinada condición en la instrucción (`CHK`, `TRAPV`, `DIVU`, `DIVS`), y un conjunto de 15 excepciones incondicionales asociadas a las instrucciones `TRAP #0` … `TRAP #15`. Estas últimas son de gran importancia ya que facilitan un mecanismo para el cambio de contexto de trabajo. Esto puede verse con claridad estudiando lo que sucede en el entrenador 68fil con lo que se denominan *sermones* (SERvicios del MONitor) que se implementan mediante instrucciones `TRAP`.

Es normal que en un sistema informático de propósito general se disponga de un núcleo de programas (sistema operativo o monitor) que permite al programador de una aplicación específica utilizar una serie de procedimientos generales. Esto sucede con el entrenador 68fil, el cual incorpora un programa monitor que facilita un conjunto de servicios que son:

`FPROG`	Marca el final de un programa
`GETCH`	Toma un carácter del puerto serie
`PUTCH`	Pone un carácter en el puerto serie
`GETS`	Toma una cadena de caracteres del puerto serie
`PUTS`	Pone una cadena de caracteres en el puerto serie
`TOUPPER`	Convierte minúsculas en mayúsculas en una cadena
`ATOI`	Convierte una cadena en un número entero
`ITOA`	Convierte un número entero en una cadena
`STRCPY`	Copia una cadena en otra
`STRLEN`	Mide la longitud de una cadena en bytes
`MEMCPY`	Copia memoria
`LNPUTS`	Pone en el puerto serie un salto de línea y una cadena
`KBHIT`	Indica si se ha recibido un carácter por el puerto serie

Como se puede observar este sistema monitor se ha escrito considerando que los usuarios harán un uso considerable de las comunicaciones con la pantalla y de cadenas de caracteres.

El 68fil, mediante las instrucciones `TRAP`, permite llamar a estas rutinas de una manera sencilla de forma que no sean subrutinas del usuario sino parte del monitor (ejecutándose en modo supervisor). En concreto las rutinas listadas se ejecutan mediante un `TRAP #15` seguido de una constante (word) que especifica el número de rutina.

3.3. PROBLEMA PROPUESTO

Se trata de crear una nueva "instrucción" para el 68000, que denominaremos `XBTST`, y cuya sintaxis será

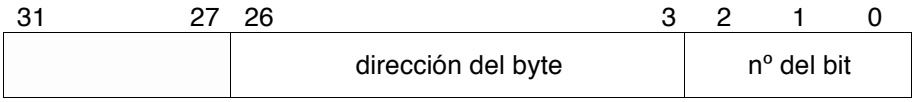
`XBTST` *dirección*

y que se codificará en memoria de la siguiente forma:

1010 XXXX XXXX XXXX
dirección [31:16]
dirección [15:0]

por lo que su ejecución provocará una excepción del tipo "emulación de línea 1010", cuyo número de vector es el 10.

El funcionamiento de `XBTST` debe ser similar al de la instrucción `BTST` del 68000, pero con la particularidad de que el operando de `XBTST` debe ser interpretado como una dirección de bit. Es decir, `XBTST` considerará que cada bit del espacio de direccionamiento se direcciona independientemente. Una dirección de bit tendrá el siguiente formato:



De esta manera, por ejemplo la "instrucción" `XBTST $81D2` será equivalente a la instrucción real del 68000 `BTST #5,$103A`. Obsérvese que, si bien desde el punto de vista de `BTST` los bits de un byte se numeran del 7 al 0 (del MSB al LSB), ocurre exactamente lo opuesto para la `XBTST`.

Recuérdese que el `PC` que se almacena en la pila durante el procesamiento de las excepciones de instrucción ilegal y emulación de línea 1010 y 1111 es el que apunta a la propia word de operación causante de la excepción.