# Práctica 8: Excepciones

Organización y Arquitectura de Computadoras

Hernández Ferreiro Enrique Ehécatl
López Soto Ramses Antonio

9 de mayo de 2019

## 1 Introducción

Al ejecutar los programas que hacemos, es probable que se presenten casos en los cuales el programa falle y se rompa. Para evitar que esto ocurra utilizamos las llamadas *excepiones*.

Las *excepiones*, por ejemplo en *Java*, son aquellas que nos permiten que algún método le indique al código acerca de algún error que se pueda ocasionar durante la ejecución del programa. Cuando un excepción es lanzada la ejecución normal del programa y éste se altera.

En MIPS MARS sucede algo similar, son señales que son enviadas al procesador por el hardware o el software que sucede al ocurrir un error por lo que el flujo normal de las instrucciones se ve alterado. Las excepciones en MIPS MARS se dividen de la siguiente manera:

Excepciones internas

Para comunicarse con el procesador pues los dispositivos de entrada y salida ocasionan una *interrupción*.

Excepciones externas

Cuando ocurre un error o el procesador envía un mensaje al dispositivo.

En la práctica se utilizarán las excepciones para evitar que ocurran errores inesperados.

## 2 Desarrollo

En esta práctica se desarrolló una calculadora con notación postfija, es decir, primero se encuentran los operandos y al final los operadores, lo implica que la precedencia de operadores permanece sin la necesidad de separar las operaciones haciendo uso de paréntesis.

| Notación normal (infija) | Notación postfija |
|---|---|
| 4 * 9 | 4 9 * |
| 3 * 1 + 2 | 3 1 * 2 + |

## Código

El código que se usó para la implementación de la calculadora es el siguiente:

```
.data
        x12:            .float  12.
        operacion:      .space  101
        nombre:         .space 101
        nombreN:        .space 101
        buffer:         .space  16
        res:            .space  16
        pregunta:       .asciiz "\n                                    Calculadora
        solicitud:      .asciiz "Inserte operacion en notacion posfija: \n"
        archivo:        .asciiz "Nombre del archivo: "
        operL:          .asciiz "La operacion leida es: "
        nombreArch:     .asciiz "Inserte nombre del archivo: "
        resultado:      .asciiz "El resultado es: "
        sistema:        .asciiz "Se ha creado un archivo con el resultado\n"
        adios:          .asciiz "Adios"
        error:          .asciiz "Opcion invalida \n"

.kdata
        divZero:        .asciiz "No puedes dividir entre cero\n"
        expInvd:        .asciiz "Expresion invalida\n"

.ktext  0x80000180
        mfc0    $k0 $13
        andi    $k0 0x7D
        srl     $k0 $k0 9
        bne     $k0 $13 fina
        li      $v0 4
        la      $a0 divZero
        syscall

        #mfc0   $k0 $13
        #andi   $k0 0x7D
        #srl    $k0 $k0 13
        #bne    $k0 $13 fin
        #li     $v0 4
```

```
        #la     $a0 expInvd
        #syscall

fina:   mfc0    $k0 $14
        addi    $k0 $k0 4
        mtc0    $k0 $14
        eret

.macro meter_a_pila(%r)
        subi    $sp $sp 1               #Apilo el numero en la pila
        sb      $t0 ($sp)
        addi    $a0 $a0 1               #Mover el puntero en la cadena
        add     $t9 $t9 1#
        j       calculadora
.end_macro

.macro preparar_numeros(%r)
        #Desapilo el primer numero de la pila
        lb      $t0 ($sp)
        move    $t5 $t0
        addi    $sp $sp 1
        sub     $t9 $t9 1#
        #Desapilo el segundo numero de la pila
        lb      $t0 ($sp)
        move    $t6 $t0
        addi    $sp $sp 1
        sub     $t9 $t9 1#
.end_macro

.macro terminar_operacion(%r)
        move    $t0 $t7
        subi    $sp $sp 1               #Apilo el numero en la pila
        sb      $t0 ($sp)
        addi    $a0 $a0 1               #Mover el puntero en la cadena
        add     $t9 $t9 1#
        j       calculadora
.end_macro

.macro remover_n(%r)
while:  lb      $t1 ($t0)
        beq     $t1 '\n' quita
        beq     $t1 '\0' end
        addi    $t0 $t0 1
```

```
        j       while

quita:  sb      $zero ($t0)

end:    nop
.end_macro

.text

pedir_op:
        #Imprimir pedido de operacion
        la      $a0 pregunta
        li      $v0 4
        syscall
        #Leer opcion
        li $v0 5
        syscall

        beq     $v0 1 escribir
        beq     $v0 2 leer
        beq     $v0 0 fin

        #Error
        li $v0 4
        la $a0 error
        syscall
        j pedir_op

escribir:
        #Imprimir pedido de operacion
        la      $a0 solicitud
        li      $v0 4
        syscall
        #Leer operacion
        li      $v0 8
        la      $a0 operacion
        li      $a1 100
        syscall
        jal     calculadora
        #Remover salto de linea
        move    $t0 $a0
        remover_n($t0)
        j calculadora
```

```
leer:
        add     $t8 $t8 1       #bandera para saber si debo devolver archivo
        #Pedir nombre
        la      $a0 archivo
        li      $v0 4
        syscall
        #Recivir nombre de archivo
        li      $v0 8
        la      $a0 nombre
        li      $a1 100
        syscall
        move    $t0 $a0
        remover_n($t0)
        #abrir archivo
        la      $a0 nombre
        li      $a1 0   #no importa mucho
        li      $a2 0   #para leer
        li      $v0 13
        syscall
        #leer archivo
        move    $a0 $v0         #poner en a0 el descriptor
        la      $a1 buffer      #poner en a1 el buffer
        li      $a2 16          #poner el tama o en a2
        li      $v0 14
        syscall
        #Imprimir operacion leida
        la      $a0 operL
        li      $v0 4
        syscall
        li      $v0 4
        move    $a0 $a1
        syscall
        move    $t0 $a0
        remover_n($t0)
        j calculadora
fin:
        #Terminar el programa
        la      $a0 adios
        li      $v0 4
        syscall
        li      $v0 10
        syscall
```

```
#################################################
calculadora:
        lb      $t0 ($a0)               #Cargar un el caracter en t0 para lec
        beqz    $t0 tresult             #Comprobar si se acab de leer la c
        ##################################
        beq     $t0 48 meter0ToPila
        beq     $t0 49 meter1ToPila
        beq     $t0 50 meter2ToPila
        beq     $t0 51 meter3ToPila
        beq     $t0 52 meter4ToPila     # Si es numero lo mete a la pila...
        beq     $t0 53 meter5ToPila
        beq     $t0 54 meter6ToPila
        beq     $t0 55 meter7ToPila
        beq     $t0 56 meter8ToPila
        beq     $t0 57 meter9ToPila
        ##################################
        beq     $t0 43 sumar
        beq     $t0 45 restar
        beq     $t0 42 multiplicar
        beq     $t0 47 dividir
        beq     $t0 32 espacio
        ##################################
        addi    $a0 $a0 1               #Mover el puntero en la cadena
        j calculadora
#################################################
espacio:
        addi    $a0 $a0 1               #Mover el puntero en la cadena
        j calculadora
#################################################
sumar:
        preparar_numeros($t0)
        add     $t7 $t5 $t6             #Suma
        terminar_operacion($t7)

restar:
        preparar_numeros($t0)
        sub     $t7 $t5 $t6             #Resta
        terminar_operacion($t7)

multiplicar:
        preparar_numeros($t0)
        mul     $t7 $t5 $t6             #Multiplica
```

```
        terminar_operacion($t7)

dividir:
        preparar_numeros($t0)
        div     $t7 $t6 $t5          #Divide
        blt     $t7 0 flotante
        terminar_operacion($t7)

######################## Meter numero a la pila ###################################
meter0ToPila:
        li      $t0 0                   #Convierto el ascii en entero
        meter_a_pila($t0)
meter1ToPila:
        li      $t0 1                   #Convierto el ascii en entero
        meter_a_pila($t0)
meter2ToPila:
        li      $t0 2                   #Convierto el ascii en entero
        meter_a_pila($t0)
meter3ToPila:
        li      $t0 3                   #Convierto el ascii en entero
        meter_a_pila($t0)
meter4ToPila:
        li      $t0 4                   #Convierto el ascii en entero
        meter_a_pila($t0)
meter5ToPila:
        li      $t0 5                   #Convierto el ascii en entero
        meter_a_pila($t0)
meter6ToPila:
        li      $t0 6                   #Convierto el ascii en entero
        meter_a_pila($t0)
meter7ToPila:
        li      $t0 7                   #Convierto el ascii en entero
        meter_a_pila($t0)
meter8ToPila:
        li      $t0 8                   #Convierto el ascii en entero
        meter_a_pila($t0)
meter9ToPila:
        li      $t0 9                   #Convierto el ascii en entero
        meter_a_pila($t0)
###############################################################################
flotante:
        mtc0    $t5 $1
        mtc0    $t6 $2
```

```
        div.s   $f3 $f6 $f5
        lwc1    $f12 x12
        syscall

tresult:beq     $t8 1 resultArch
result:
        #Imprimir secuencia de resultado
        la      $a0 resultado
        li      $v0 4
        syscall
        #Imprimir resultado
        move    $a0 $t7
        li      $v0 1
        syscall
        j pedir_op

resultArch:

        #Pedir nombre
        la      $a0 nombreArch
        li      $v0 4
        syscall
        #Recivir nombre de archivo
        li      $v0 8
        la      $a0 nombreN
        li      $a1 100
        syscall
        move    $t0 $a0
        remover_n($t0)
        #crear archivo copia
        li      $v0 13
        la      $a0 nombreN
        li      $a1 9
        li      $a2 0
        syscall

        sw      $t7 res

        #escribimos en la copia
        move    $a0 $v0
        la      $a1 res
        li      $a2 16
        li      $v0 15
```

```
syscall
#Imprimir secuencia de guardado
la      $a0 sistema
li      $v0 4
syscall
#Imprimir secuencia de resultado
la      $a0 resultado
li      $v0 4
syscall
#Imprimir resultado
move    $a0 $t7
li      $v0 1
syscall
sub     $t8 $t8 1
j pedir_op
```

# 3 Conclusiones

En esta práctica aprendimos el uso y funcionamiento básicos en MIPS MARS y también nos divertimos un poco.

## Preguntas

1. En un procesador, ¿qué es el modo supervisor? ¿Qué funciones tiene? ¿Cómo se implementa?

   R. El modo supervisor es el que está permitido la ejecución de cualquier instrucción de máquina como: la autorización de interrupciones, el acceso a los registros utilizados por el hardware u operaciones de entrada y de salida.

2. ¿Cuál es la relación entre una llamada al sistema y una excepción?

   R. Ambas se ejecutan en el modo supervisor.

3. ¿Qué es un vector de interrupciones?

   R. Es un vector en el que se encuentran los valores que apuntan a la dirección en memoria del gestor de una interrupción. Es decir, gestiona correctamente las interrupciones que se solicitan al microprocesador.