Universidade Federal de Pelotas
Centro de Desenvolvimento Tecnológico
Bacharelado em Ciência da Computação
Engenharia de Computação

Arquitetura e Organização de Computadores I

Prática

Aula 8

Tratamento de Exceções, Subrotinas

Prof. Guilherme Corrêa
gcorrea@inf.ufpel.edu.br

Prof. Bruno Zatt

Tratamento de Exceções

- Operações de entrada e saída são em geral executadas pelo código do **Sistema Operacional** e/ou por **drivers** que controlam o acesso aos dispositivos;
- Programas usuários executam essas operações através do uso de **exceções** ou **chamadas de sistema**;
- Quando ocorre uma exceção, um trecho de código correspondente do **sistema operacional** é acionado.

Tratamento de Exceções

- O simulador **MARS** não possui um sistema operacional, mas possui um pequeno **tratador de exceções** que pode ser usado para operações básicas de **entrada e saída**;
- Quando ocorre uma exceção no MARS, um trecho de código correspondente do tratador de exceções é acionado.

Tratamento de Exceções

Instrução `syscall`

- Programas assembly requisitam serviços do sistema operacional através de chamadas à instrução **`syscall`**;
- A instrução **`syscall`** **transfere** o controle para o sistema operacional, que **executa** algum serviço e em seguida **retorna** o controle ao programa;
- Sistemas operacionais diferentes usam esta instrução de forma diferente.

Tratamento de Exceções

Instrução `syscall`

- Serviços possíveis para executar com **`syscall`** :
  - Imprimir inteiro, float, double, string;
  - Ler inteiro, float, double, string;
  - Alocar memória;
  - Terminar execução do programa.

Tratamento de Exceções

Instrução `syscall`

- Serviço especificado por um **código em $v0**;
- Serviços diferentes esperam **parâmetros** em registradores diferentes e nem todos os serviços retornam valores.

```
li   $v0, codigo        # Carrega em $v0 o código de
                        # um serviço do SO.

.......                 # Coloca parâmetros para o serviço nos
.......                 # registradores $a0, $a1
                        # ou registradores $f (float)

syscall                 # Invoca o sistema operacional.
                        # Valor de retorno (se existir)
                        # em $v0 ou $f0
```

Tratamento de Exceções

Registradores para uso em `syscall`

Serviço	Código ($v0)	Argumentos	Retorno
imprime inteiro	1	$a0 (inteiro)	-
imprime float	2	$f12 (float)	-
imprime double	3	$f12, $f13 (double)	-
imprime string	4	$a0 (endereço)	-
lê inteiro	5	-	$v0 (inteiro)
lê float	6	-	$f0 (float)
lê double	7	-	$f0, $f1 (double)
lê string	8	$a0 (endereço) $a1 (tamanho)	-
aloca memória	9	$a0 (n. de bytes)	$v0 (endereço)
sai	10	-	-

Tratamento de Exceções

O que faz?

```
# hello.asm
#
        .text
        .globl  main
main:
        li      $v0,4           # código 4 == imprime string
        la      $a0,string      # $a0 == endereço da string
        syscall                 # Invoca SO.

        li      $v0,10          # código 10 == sai
        syscall                 # termina programa.

        .data
string: .asciiz     "Hello World!\n"
```

Tratamento de Exceções

O que faz?

```
li      $v0, 5          # código 5 == lê inteiro
syscall                 # Invoca o sistema operacional.
                        # Lê uma linha de caracteres ASCII.
                        # Converte-os para um inteiro de
                        # 32 bits.
                        # $v0 <-- inteiro em complem. de dois


li      $v0, 1          # código 1 == imprime inteiro
li      $a0, 123        # $a0 == 123
syscall                 # Invoca o sistema operacional.
                        # Converte o inteiro em caracteres.
                        # Imprime os caracteres no monitor.
```

Tratamento de Exceções

O que faz?

```
li      $v0,8                   # código 8 == lê string
la      $a0,buffer              # $a0 == endereço do buffer
li      $a1,16                  # $a1 == tamanho do buffer
syscall                         # Invoca SO.

 . . . .

.data
buffer: .space 16               # reserva 16 bytes
```

Subrotinas

Instrução `jal`

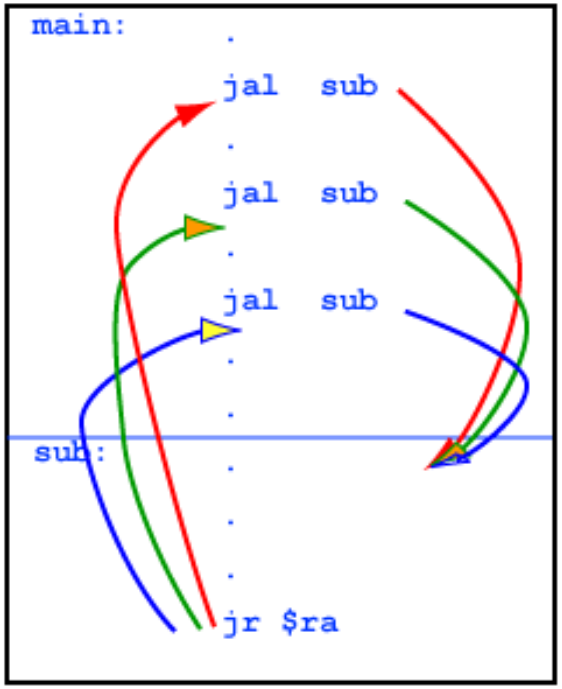
- As **chamadas** para subrotinas (funções) no MIPS são feitas através da instrução **`jal`** (*jump and link*);
- O registrador especial **`$ra`** (**`$31`**) recebe o endereço de **retorno** da subrotina;
- O endereço de retorno equivale à instrução **após o *delay slot***.

```
jal sub         # $ra ($31) <- PC+8
                # (endereço a 8 bytes da instrução jal)
                # PC  <- sub   PC recebe o endereço de
                # entrada da subrotina
                # a instrução precisa de um delay slot
```

Subrotinas

Instrução `jal`

- A cada chamada a subrotina (`jal`), `$ra` recebe o endereço de retorno apropriado;
- `jr $ra` (retorno da subrotina) salta para endereço após o ***delay slot*** do `jal` correspondente



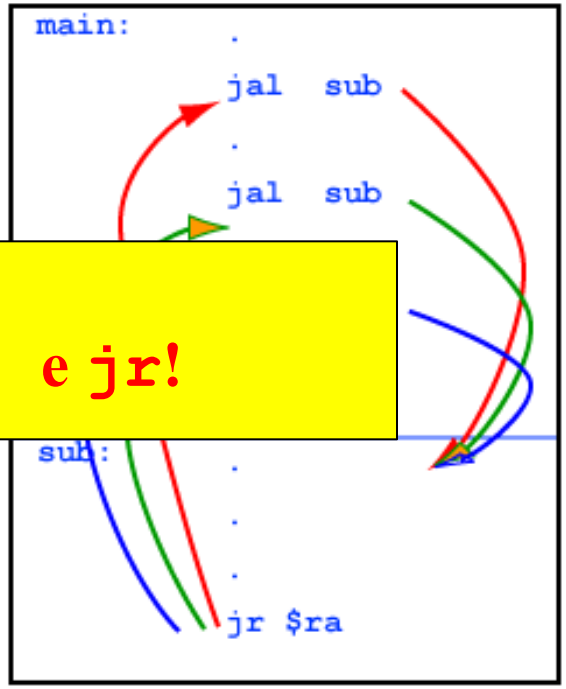
Correct Subroutine Linkage

Subrotinas

Instrução `jal`

- A cada chamada a subrotina (`jal`), `$ra` recebe o endereço de retorno apropriado;
- `j`
  su
  após o ***delay slot*** do `jal` correspondente

ATENÇÃO!!
Sempre usar `nop` após `jal` e `jr`!



Correct Subroutine Linkage

Subrotinas

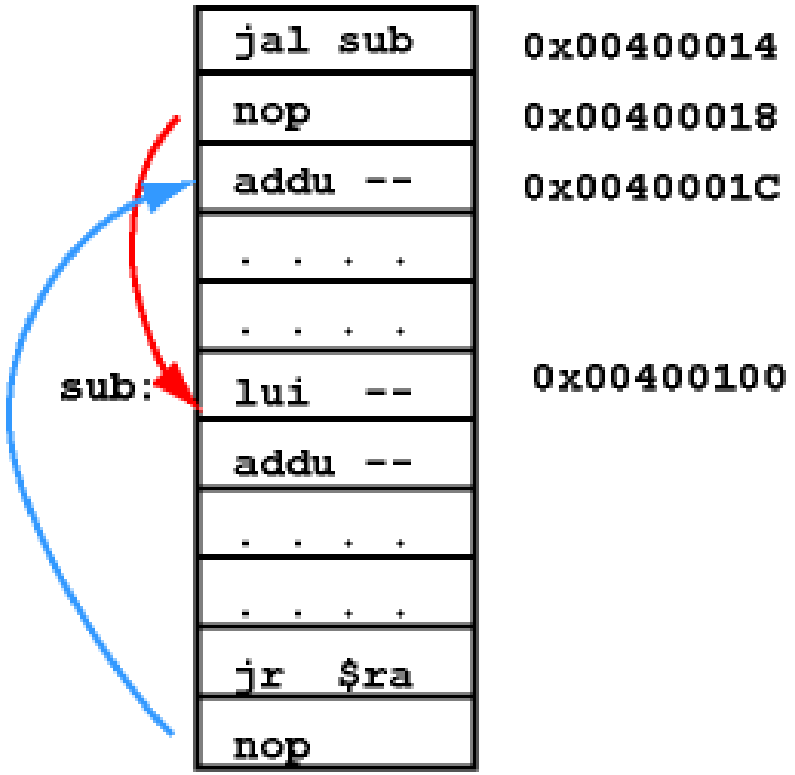
Exemplo:

- Chamada à subrotina com a instrução **`jal:`**

```
jal sub
    # $ra <- 0x0040001C
    # PC <- 0x00400100
```

- Retorno da subrotina com a instrução **`jr:`**

```
jr $ra
    # PC <- $ra
```



Subrotinas

Convenções (uso):

- Subrotinas sempre chamadas através da instrução **`jal`**;
- Uma subrotina não pode chamar outra subrotina;
- Subrotina termina e retorna para o chamador usando **`jr $ra`**;
- Programa termina usando **`syscall`** código **10**.

Subrotinas

Convenções (registradores):

- **`$t0-$t9`**: temporários, alterados livremente pela subrotina;
- **`$s0-$s7`**: salvos, subrotina **não deve** alterá-los;
- **`$a0-$a3`**: **argumentos** da subrotina, podem ser alterados por ela;
- **`$v0-$v1`**: valores de **retorno** da subrotina, **devem ser alterados** (se a subrotina retorna algum valor).

Subrotinas

Utilização de Registradores

Registrador	Nome	Uso (convenção)
$0	$zero	Zero
$1	$at	*Assembler Temporary*
$2, $3	**$v0, $v1**	**Valor de retorno de subrotina**
$4 – $7	**$a0 – $a3**	**Argumentos de subrotina**
$8 – $15	**$t0 – $t7**	**Temporários (locais à função)**
$16 – $23	**$s0 – $s7**	**Salvos (não alterados na função)**
$24, $25	**$t8, $t9**	**Temporários**
$26, $27	$k0, $k1	Kernel (reservado para SO)
$28	$gp	*Global Pointer*
$29	$sp	*Stack Pointer*
$30	$fp	*Frame Pointer*
$31	**$ra**	**Endereço de Retorno**