Universidade Federal de Santa Catarina - Centro Tecnológico - Departamento de Informática e Estatística
INE 5411 - Organização de Computadores
Roteiro do Laboratório 5 - Estudo de caso de ordenação
Pré-requisitos para a compreensão e execução do laboratório: arranjos, laços e procedimentos.

1. Objetivo

O objetivo desta aula é revisar a convenção de chamada de procedimentos do MIPS e fazer uso prático dela através de um estudo de caso. O estudo de caso proposto é a implementação de um algoritmo de ordenação de números inteiros. O algoritmo escolhido faz uso de dois procedimentos:

- `swap`: troca de posição dois elementos de um arranjo de inteiros;
- `sort`: ordena um arranjo de inteiros no estilo do método "Bubble Sort".

O procedimento `sort` invoca `swap`, que é um procedimento-folha. Como veremos, essa diferença de invocação tem consequências na alocação de registradores e, consequentemente, no uso da convenção de chamada. Para a geração de código de cada um desses procedimentos, você deve observar os seguintes passos:

1. Alocar registradores para as variáveis;
2. Produzir o código para o corpo do procedimento;
3. Preservar registradores através da invocação do procedimento.

Os Passos 1 e 3 precisam obedecer à convenção de chamada. A alocação correspondente ao Passo 1 deve levar em conta a declaração de variáveis do código-fonte e o fato de um procedimento ser folha ou não para decidir se uma variável deve ser alocada em registrador temporário (`$t0, $t1, ...`) ou em registrador salvo (`$s0, $s1, ...`). O Passo 3 deve levar em conta se um registrador foi usado no corpo do procedimento (Passo 2) e se ele é temporário ou salvo para decidir se tal registrador precisa ser preservado ou não.

2. Revisão da convenção de chamada de procedimentos

Alocação de valores em registradores

- **Registradores de argumento** (`$a0-$a3`): devem armazenar os quatro primeiros argumentos de um procedimento (os demais são armazenados na pilha).
- **Registrador**(es) **de valor** (`$v0-$v1`): devem armazenar os valores de retorno de funções.
- **Registradores temporários** (`$t0-$t9`): devem armazenar preferencialmente valores temporários com curto tempo de vida, que geralmente não são necessários após uma chamada de procedimento, podendo ser usados no procedimento invocado sem a necessidade de preservação por este último.
- **Registradores salvos** (`$s0-$s7`): devem armazenar valores com longos tempos de vida que geralmente são necessários após uma chamada de procedimento, devendo por isso ser preservados através dela.

Dica: O livro-texto possui uma lista de todos registradores temporários e registradores salvos.

Divisão de responsabilidades

- Procedimento **chamador**: é responsável por preservar os registradores `$a0-$a1` e `$t0-$t9`. Devem ser preservados apenas os registradores cujos valores serão usados no corpo do procedimento chamador depois da chamada.
- Procedimento **chamado:** é responsável por preservar os registradores `$s0-$s7`. Devem ser preservados apenas os registradores cujos valores serão usados no corpo do procedimento chamado. Deve também preservar o registrador `$ra`, exceto se for um procedimento-folha.

Salvamento e restauração de valores preservados

- Procedimento **chamador**: os conteúdos de registradores `$a0-$a1` ou `$t0-$t9` a serem preservados são normalmente copiados em registradores salvos (`$s0-$s7`), pois estes são preservados através de chamadas (em geral isso é feito apenas para os registradores `$a0-$a1`, pois se houvesse a necessidade de preservar registradores `$t0-$t9` seria mais eficiente ter alocado os conteúdos diretamente em registradores salvos). O salvamento deve ser feito antes de se efetuar a chamada. Alternativamente, tais conteúdos podem ser salvos na pilha.
- Procedimento **chamado**: o conteúdo dos registradores `$s0-$s7` ou `$ra` a serem preservados são armazenados na pilha. O salvamento deve ser feito antes de se iniciar o corpo do procedimento (ou seja, antes dos valores serem alterados). A restauração deve ser feita antes de retornar ao procedimento chamador.

3. Procedimento 1: swap

O código abaixo descreve o procedimento, escrito em linguagem C. O primeiro argumento é o endereço-base do arranjo e o segundo é o valor do índice do elemento a ser trocado com seu sucessor.

```
void swap ( int v[], int k )
{
        int temp;
        temp = v[k];
        v[k] = v[k+1];
        v[k+1] = temp;
}
```

Procedimento de teste

O arranjo `v[]` será armazenado nas 10 primeiras palavras da área de dados globais da memória. A palavra seguinte armazenará o valor do parâmetro `k`. Aplique o procedimento abaixo para verificar o funcionamento de `swap`.

```
# Estímulos: v[] = [9, 8, 7, 6, 5, 4, 3, 2, 1,-1] e k = 2
.data
_v: .word 9,8,7,6,5,4,3,2,1,-1
_k: .word 2
```

Resultado esperado: v[] = [9, 8, 6, 7, 5, 4, 3, 2, 1, -1]

Exercício 1: implementação e teste do procedimento swap

Obedecendo às convenções de chamada de procedimentos, implemente o corpo do código do procedimento `swap` em linguagem de montagem do processador MIPS, sob as seguintes condições:

- Aloque somente registradores temporários para a implementação do corpo do procedimento.
- **NÃO** use os registradores `$a0` e `$a1` como destino de instrução alguma.
- Assuma que a variável `k` está sempre dentro dos limites do arranjo (você **NÃO** deve implementar o teste de limites do arranjo).

O trecho de código abaixo ilustra esquematicamente a estrutura do arquivo de programa em linguagem de montagem. Armazene seu código em um arquivo exercicio1.txt, instrumentando-o com os estímulos do procedimento de teste.

```
# Estímulos: v[] = [9, 8, 7, 6, 5, 4, 3, 2, 1,-1] e k = 2
.data
_v: .word 9,8,7,6,5,4,3,2,1,-1
_k: .word 2

.text
.globl main
main:
# Inicialização do parâmetros
la $a0, _v
lw $a1, _k
jal swap       # Chamada do procedimento
li $v0, 10     # Exit syscall
syscall

# Corpo do procedimento
swap:
...
jr $ra # retorno ao programa principal
```

a) Carregue e execute o arquivo exercicio1.txt. Retrabalhe o código até que o resultado esperado seja alcançado. Certifique-se de que o código está correto, pois ele será usado ao longo dos próximos experimentos (se o código contiver um erro, tal erro prejudicará todos os experimentos seguintes).

b) Responda à Questão 1.1 do relatório de aula.

4. Procedimento 2: sort

O código seguinte descreve o procedimento escrito em linguagem C. O primeiro argumento é o endereço-base do arranjo e o segundo é o seu número total de elementos.

```
void sort(int v[], int n)
{
        int i, j;
        for ( i = 0 ; i < n ; i = i + 1 )
        {
                for ( j = i – 1 ; j >= 0 && v[j] > v[j+1] ; j = j - 1 )
                {
                        swap(v,j);
                }
        }
}
```

Procedimento de teste

O arranjo `v[]` será armazenado nas 10 primeiras palavras da área de dados globais da memória. A décima primeira palavra é o tamanho `n` do arranjo. Aplique o procedimento abaixo para verificar o funcionamento de `sort`.

```
# Estímulos: v[] = [9, 8, 7, 6, 5, 4, 3, 2, 1,-1] e n = 10
.data
_v: .word 9,8,7,6,5,4,3,2,1,-1
_n: .word 10
```

Resultado esperado: v[] = [-1, 1, 2, 3, 4, 5, 6, 7, 8, 9]

Exercício 2: implementação e teste do procedimento `sort`

Este exercício induz propositadamente alguns erros típicos na codificação de procedimentos. Partiremos de um código que contém erros e vamos eliminá-los gradativamente até que o resultado esperado seja alcançado. Assim, faça **apenas** as alterações solicitadas a cada passo e interprete os resultados anômalos. **Não faça qualquer outra modificação no código**.

a) Faça o download do arquivo no Moodle**.** Salve o arquivo como exercicio2-1.txt e edite-o acrescentando o código do procedimento swap do Exercício 1. Estude a correspondência entre o código baixado para o procedimento sort e o respectivo código-fonte, inserindo comentários no arquivo para facilitar sua interpretação.

b) Carregue o arquivo exercicio2-1.txt no simulador. Execute passo a passo o programa modificado.

c) Responda às Questões 2.1 e 2.2 do relatório de aula.
Dica: Use os breakpoints do simulador.

d) Copie o arquivo exercicio2-1.txt para um novo arquivo exercicio2-2.txt, cujo código será retrabalhado para se tentar eliminar o comportamento anômalo observado para o código anterior. Neste arquivo, inclua uma instrução `move $a1, $s1` imediatamente antes de **MARCA 2**. Execute passo a passo o programa retrabalhado.

e) Responda às Questões 2.3 e 2.4 do relatório de aula.
Dica: Use a descrição em linguagem de alto nível como suporte para entender o que acontece no código.

f) Copie o arquivo exercicio2-2.txt para um novo arquivo exercicio2-3.txt, cujo código será retrabalhado para se tentar eliminar completamente o comportamento anômalo ainda remanescente no código anterior. Neste arquivo, inclua uma instrução `move $s3, $a1` imediatamente antes da instrução que inicializa o valor de `i`. Agora modifique a instrução `slt` que é sucessora imediata da **MARCA 1** para `slt $t0, $s0, $s3`. Execute o programa retrabalhado, que agora deve funcionar corretamente.
Dica: Se o código não produziu o resultado esperado, provavelmente há um erro na codificação na função `swap`, ou você não obedeceu alguma das especificações; neste caso, revise o código e repita todos os experimentos anteriores.

g) O código do exercicio2-3.txt, embora esteja funcionando corretamente, ainda não obedece plenamente à convenção de chamadas de procedimentos. Suponha que você tivesse codificado seu programa `swap` de forma que o registrador `$a0` fosse usado como destino de uma de suas instruções. Assim, você deveria acrescentar uma instrução `move $s2, $a0` imediatamente antes da instrução `move $s3, $a1`, que já havia sido incluída para fazer o código funcionar corretamente (obviamente, você deveria também alterar a(s) instrução(ões) que referenciavam `$a0` para agora usar `$s2`). Suponha ainda que o procedimento `main` (que invoca `sort`) utilize em seu corpo os registradores `$s2` e `$s3` (que você usou para preservar `$a0` e `$a1` em `sort`). Crie uma cópia do arquivo exercicio2-3.txt, renomeando-a para exercicio2-4.txt. Neste arquivo, insira as instruções necessárias para prevenir as anomalias decorrentes das hipóteses acima descritas.

h) Responda à Questão 2.5 do relatório de aula.