Universidade Federal de Santa Catarina - Centro Tecnológico - Departamento de Informática e Estatística

INE 5411 - Organização de Computadores

Roteiro do Laboratório 3 - Implementação de Laços de Iteração

Pré-requisitos para a compreensão e execução do laboratório: arranjos e estruturas de laços.

1. Objetivo

O objetivo geral desta aula é estudar alternativas eficientes para a geração de código associado a laços de iteração (while, repeat...until, do...while e for) suportados em linguagens de alto nível contemporâneas. Nesse contexto, esta aula propõe um estudo de caso de um trecho de programa contendo uma construção while. Seu objetivo específico é mostrar que a escolha de linguagens de programação de alto nível, tais como C ou Java, resulta em códigos com diferentes números de instruções, o que afeta o desempenho.

2. Estudo de caso

O código abaixo descreve o laço de iteração usado para o estudo de caso. Na situação anômala em que o índice i resulte fora dos limites do arranjo save, o mau funcionamento do programa pode ser mais difícil de depurar quando o programa foi originalmente escrito em C, linguagem que não possui mecanismo algum de teste automático de limites, ao contrário de Java, a qual requer que esse teste seja feito em tempo de execução. Em outras palavras, a pessoa programadora C precisa incluir explicitamente (manualmente) o teste de limites em seu código (se quiser que seja efetuado), enquanto a pessoa programadora Java não precisa se preocupar em fazê-lo (mas também não pode evitá-lo), pois as instruções de teste serão geradas automaticamente pelo compilador. Em um caso, gera-se um código potencialmente mais rápido (assumindo um risco), enquanto que, em outro, um código mais produtivo é gerado (aceitando uma degradação de desempenho).

```
i = 0;
while ( save[i] == k )
        i += 1;
```

Convenções para os exercícios

- Adote a seguinte alocação de registradores: **(i,k) -> ($s3,$s5)**.
- Assuma que o **endereço da base do arranjo save** está armazenado no registrador **$s6**.
- Atribua labels Loop e Exit às posições de memória contendo, respectivamente, a primeira instrução executada dentro do laço e a primeira instrução executada à saída do laço.
- **Use somente instruções nativas na inicialização e no corpo do laço** (exceto pela pseudoinstrução **`la`** necessária para inicializar o endereço-base do arranjo e **`lw`** para o valor de k).

3. Geração de código para um laço de iteração

Procedimento de teste

Para facilitar o teste, os estímulos serão organizados na área de dados globais da memória. As primeiras palavras dessa área armazenarão os elementos do arranjo de inteiros save. A palavra seguinte armazenará o valor da variável k. Aplique cada um dos procedimentos abaixo, não esquecendo de montar novamente o programa para cada novo estímulo.

Procedimento de teste 1.1: Estímulos: save = [...] e k = 6; resultado esperado: Resultado 1.1 (no relatório)

```
.data
_save: .word Estímulo 1.1 (disponível no relatório)
_k: .word 6
```

Procedimento de teste 1.2: Estímulos: save = [...] e k = 6; resultado esperado: Resultado 1.2 (no relatório)

```
.data
_save: .word Estímulo 1.2 (disponível no relatório)
_k: .word 6
```

Para facilitar a observação dos resultados esperados, acrescente ao final de seu programa, iniciando na posição associada ao label Exit, a seguinte seqüência de instruções (não se preocupe em entendê-la agora; assuma que esta sequência simplesmente imprime no console o valor de $s3 à saída do laço):

```
Exit:
addi $v0, $zero, 1
add $a0, $zero, $s3
syscall
```

Exercício 1:

Implemente um código em linguagem de montagem do MIPS para o trecho de programa escrito em linguagem C, mostrado na Seção 2.

a) Crie um arquivo exercicio1-1.txt com a sua implementação, instrumentado-o com os estímulos listados para o Procedimento de teste 1.1, conforme a estrutura do código abaixo. Monte-o e execute-o.

```
.data
_save: .word ...
_k: .word 6
.text
.globl main
main: # inicialização
...
Loop: # corpo do laço
sll $t1, $s3, 2
add $t1, $t1, $s6
lw $t0, 0($t1)
bne $t0, $s5, Exit
addi $s3, $s3, 1
j Loop

Exit: # rotina para imprimir inteiro no console
addi $v0, $zero, 1
add $a0, $zero, $s3
syscall
```

b) Responda à Questão 1.1 do relatório de aula.

c) Crie um arquivo exercicio1-2.txt com o programa acima, agora instrumentado-o com os estímulos do Procedimento de teste 1.2. Monte-o e execute-o.

d) Responda às Questões 1.2 e 1.3 do relatório de aula.

4. Inserção de teste de limites de um arranjo

Informações adicionais

Quando um programa é escrito em uma linguagem orientada a objetos, o compilador encapsula cada objeto na forma de uma estrutura de dados, onde o primeiro elemento contém o endereço de uma tabela que permite invocar métodos do objeto. Por isso, em Java arrays, a primeira palavra armazenada na estrutura de dados é reservada para armazenar o endereço da correspondente tabela. Além disso, a segunda palavra da estrutura de dados é reservada para armazenar um atributo: o número total de elementos do arranjo. Portanto, ao resolver o Exercício 2, lembre que os elementos propriamente ditos do arranjo save são armazenados a partir da terceira palavra.

Lembre também que, se o arranjo contém N elementos, índices válidos estão no intervalo [0, N).

Procedimento de teste

Os procedimentos de teste a serem usados nesta seção são em tudo similares aos usados na Seção 3, exceto pelo fato de que as duas primeiras palavras armazenadas em memória são utilizadas para armazenar atributos do arranjo (para aderir à representação Java). Em particular, a segunda palavra contém o tamanho do arranjo (a primeira é irrelevante para nosso código e o valor 9999 lhe será atribuído só para distingui-la das demais). **Os resultados esperados abaixo referem-se ao comportamento desejado para o código-fonte original em linguagem de alto nível**.

Procedimento de teste 2.1: Estímulos: save = [...] e k = 6; resultado esperado: Resultado 2.1 (no relatório)

```
.data
_save: .word Estímulo 2.1 (disponível no relatório)
_k: .word 6
_error: .asciiz "Index Out of Bounds Exception"
```

Procedimento de teste 2.2: Estímulos: save = [...] e k = 6; resultado esperado: Resultado 2.2 (no relatório)

```
.data
_save: .word Estímulo 2.2 (disponível no relatório)
_k: .word 6
_error: .asciiz "Index Out of Bounds Exception"
```

Para facilitar a observação de erro em tempo de execução, modifique o final de seu programa, iniciando na posição associada ao label Exit, com a seguinte sequência de instruções (não se preocupe em entendê-la agora; assuma que esta sequência simplesmente imprime no console o valor de $s3 à saída do laço, caso o valor seja válido e, em caso contrário, imprime uma mensagem de erro):

```
Exit:
addi $v0, $zero, 1
add $a0, $zero, $s3
syscall
j End

IndexOutOfBounds:
addi $v0, $zero, 4
la $a0, _error
syscall
End:
```

Convenções adicionais

- Faça com que a última instrução executada antes de se entrar no laço carregue em $t2 o número de elementos do arranjo (tamanho).
- Assuma que, quando o índice não está dentro dos limites do arranjo, o programa desvia para um endereço representado pelo label IndexOutOfBounds, onde há uma chamada para uma rotina de sinalização de erro (rotina que já foi dada nos procedimentos de teste).

Exercício 2:

Assuma agora que o código original do trecho de programa tenha sido escrito em Java. Isso requer um refinamento do programa em linguagem de montagem. Partindo do código original (Exercício 1), modifique-o e teste seu funcionamento da seguinte forma:

a) Reescreva o código original incluindo as instruções necessárias para a verificação dos **limites inferior e superior** do arranjo.
Sugestão: use o deslocamento ("offset") da instrução lw para compensar o efeito dos atributos armazenados nas duas primeiras palavras reservadas da estrutura de dados.

b) Crie um arquivo exercicio2-1.txt com o programa refinado, instrumentado-o com os estímulos do Procedimento de teste 2.1, conforme a estrutura abaixo. Carregue-o e execute-o. Se você não tiver obtido o resultado esperado, revise seu programa para que isso ocorra antes de prosseguir no roteiro.

```
.data
_save: .word ...
_k: .word 6
_error: .asciiz "Index Out of Bounds Exception"
.text
.globl main
main: # inicialização
...
Loop: # verificação de limites do arranjo
...
# corpo do laço
...
Exit: # rotina para imprimir inteiro no console
addi $v0, $zero, 1
add $a0, $zero, $s3
syscall
j End

IndexOutOfBounds: # rotina para imprimir mensagem de erro no console
addi $v0, $zero, 4
la $a0, _error
syscall
End:
```

c) Crie um arquivo exercicio2-2.txt com o programa refinado, instrumentado-o com os estímulos do Procedimento de teste 2.2. Carregue-o e execute-o.

d) Responda às Questões 2.1 e 2.2 do relatório de aula.