Memorial Descritivo de Soluções da Disciplina de Arquitetura e Organização de Computadores 1
Laboratório de Programação Assembly 2

Cauã de Paula Dias, Ester Camilly Simplício de Freitas, Marcos Gabriel Moreira Fonseca, Victor Gava Senema
Faculdade de Computação
Universidade Federal de Uberlândia (UFU) - Uberlândia, MG - Brasil
cauadepauladias@ufu.br, esterzolas@ufu.br, gabriel.kconde@ufu.br, victor.senema@ufu.br.br

Abstract: *This document describes and provides examples of the main topics in the descriptive report related to implementing and analyzing algorithms in the MIPS architecture. This architecture, based on registers, requires a detailed understanding of the behavior of each programmed algorithm, implementation strategies, auxiliary functions, and the performance tests carried out to evaluate their usage limits.*

Resumo: *Este documento descreve e apresenta exemplos sobre os principais tópicos no memorial descritivo relacionados à implementação e análise de algoritmos na arquitetura MIPS. Esta arquitetura, baseada em registradores, requer uma compreensão detalhada do comportamento de cada algoritmo programado, estratégias de implementação, funções auxiliares e os testes de funcionamento realizados para avaliar seus limites de uso.*

1. Introdução

A proposta deste trabalho é apresentar de forma didática e simplificada alguns dos principais tópicos em programação assembly para a arquitetura MIPS. Sendo assim, cada tópico poderá ter a explicação teórica pura, exemplos práticos, analogias para facilitar o entendimento e figuras. Serão abordados os temas: manipulação de dados, controle de fluxo e sub-rotinas.

2. Calculando o Comprimento de uma String

A função string_length() desempenha um papel importante ao calcular o comprimento de uma string em linguagem C, contando o número de caracteres antes do terminador nulo. E se traduzissemos o código para MIPS assembly, onde cada instrução é crucial Em MIPS, cada linha representa uma operação fundamental, desde o movimento de dados até o uso eficiente de registradores e controle de fluxo. A função string_length() em MIPS assembly ilustra como a arquitetura MIPS lida com operações básicas de manipulação de strings.

2.1. Tarefa

Converta o fragmento de código abaixo, escrito em linguagem C, para a linguagem do MIPS assembly.

```c
/** Returns the number of bytes in S before, but not counting, the null
terminator. */
size_t string_length(char *s) {
    char *s2 = s;
    while(*s2++);
    return s2 - s - 1;
}
```

Figura 1: Fragmento de Algoritmo em linguagem C. Fonte: Google Imagens

2.1.1. Análise do Algoritmo de Cálculo de Comprimento de String em MIPS Assembly

Como requisitado, o algoritmo feito implementa a função string_length, que calcula o comprimento de uma string, contando o número de bytes antes do terminador nulo. Ele percorre a string byte a byte até encontrar o terminador nulo, incrementando um contador a cada iteração. Quando o terminador nulo é encontrado, o algoritmo calcula o comprimento da string subtraindo o endereço inicial do endereço final. São utilizadas funções auxiliares neste algoritmo. Ele é genérico, pois pode calcular o comprimento de qualquer string terminada por nulo. Quanto ao desempenho, o algoritmo é eficiente, com tempo de execução proporcional ao tamanho da string, tornando-o adequado para uma variedade de cenários de aplicativos.

3. Conversão de Strings para Minúsculas

A função string_to_lowercase em C converte uma string para minúsculas, percorrendo cada caractere da string. Se um caractere estiver em maiúsculo, a função o converte para minúsculo. Em MIPS Assembly, é possível implementar operações básicas de manipulação de strings, como comparações, manipulação de dados e controle de fluxo, para realizar tarefas específicas em nível de caractere.

3.1. Tarefa

Converta o fragmento de código abaixo, escrito em linguagem C, para a linguagem do MIPS assembly.

```c
/** Converts the string S to lowercase */
void string_to_lowercase(char *s) {
    for(char c = *s; (c=*s) != '\0'; s++) {
        if(c >= 'A' && c <= 'Z') {
            *s += 'a' - 'A';
        }
    }
}
```

Figura 2: Fragmento de Algoritmo em linguagem C. Fonte: Google Imagens

3.1.1. Análise do Desempenho do Algoritmo de Conversão para Minúsculas em MIPS Assembly

O algoritmo em MIPS Assembly converte uma string para minúsculas. Para isso, o algoritmo lê a string fornecida pelo usuário e, em seguida, percorre cada caractere usando um loop. Para cada caractere, verifica se é uma letra maiúscula e, se for, converte-o para minúscula. O algoritmo é genérico e pode lidar com qualquer string fornecida pelo usuário, independentemente do conteúdo ou tamanho. Ele não depende de informações específicas sobre a string além do seu comprimento. Em termos de desempenho, o algoritmo é diretamente proporcional ao tamanho da string de entrada. Em outras palavras, para uma string de tamanho n, o tempo de execução será proporcional a n, já que o algoritmo percorre cada caractere da string em um loop. A complexidade da conversão de maiúsculas para minúsculas é constante para cada caractere, o que significa que não há impacto adicional no tempo de execução à medida que a string cresce.

4. Cálculo do Seno em MIPS Assembly

Tarefa: Escreva em MIPS assembly uma função denominada senox, que calcula o seno de x. A função seno de x pode ser calculado utilizando a expansão da Série de Taylor:

$$x = x - (x\text{^}3 / 3!) + (x\text{^}5 / 5!) - (x\text{^}7 / 7!) + \ldots$$

O elemento x da Série de Taylor deve ser em radianos. Portanto, a função senox deve ler o valor do ângulo em graus e converter para radianos. Para realizar a conversão utilize a regra de três, considerando que PI radianos é igual a 180 e que a constante PI assume valor de 3.141592. A função deve somar os termos da série até aparecer um termo cujo valor absoluto seja menor que 0.00001 (precisão). Isto é, o termo $|x\text{^}k / k!|$ tende a zero quando k tende a +∞. Repetir o cálculo dos termos até um k tal que $|x\text{^}k / k!| < 0.00001$. Dica: Use a convenção de registrador de ponto flutuante MIPS para passar o parâmetro x e retornar o resultado da função. Evite calcular o valor do fatorial, calcule um termo da série usando o termo anterior.

4.1. Análise do Algoritmo de Cálculo do Seno em MIPS Assembly

O algoritmo em MIPS assembly apresentado calcula o seno de um ângulo em graus usando a série de Taylor. Ele solicita ao usuário o ângulo em graus, converte-o para radianos e, em seguida, calcula o seno usando a série de Taylor. O código apresenta funções auxiliares separadas. Foram feitos diferentes testes com valores de ângulos para verificar se os resultados estavam dentro do esperado. O algoritmo é genérico e pode calcular o seno de qualquer ângulo dentro dos limites do hardware. Em relação ao desempenho, o tempo de execução depende da precisão desejada e do número de iterações necessárias para alcançá-la. Em geral, o desempenho é razoável para ângulos moderados, mas pode ser afetado por ângulos extremos.

5. Cálculo do Cosseno em MIPS Assembly

Tarefa: Escreva em MIPS assembly uma função denominada cosex, que calcula o cosseno de x. A função cosseno de x pode ser calculado utilizando a expansão da série de Taylor:

$$x = 1 - (x\text{^}2 / 2!) + (x\text{^}4 / 4!) - (x\text{^}6 / 6!) + \ldots$$

O elemento x da série de Taylor deve ser em radianos. Portanto, a função cosex deve ler o valor do ângulo em graus e converter para radianos. Para realizar a conversão utilize a regra de três, considerando que PI radianos é igual a 180 e que a constante PI assume valor de 3.141592. A função deve somar os termos da série até aparecer um termo cujo valor absoluto seja menor que 0.00001 (precisão). Isto é, o termo $|x\text{^}k / k!|$ tende a zero quando k tende a +∞. Repetir o cálculo dos termos até um k tal que $|x\text{^}k / k!| < 0.00001$. Dica: Use a convenção de registrador de ponto flutuante MIPS para passar o parâmetro x e retornar o resultado da função. Evite calcular o valor do fatorial, calcule um termo da série usando o termo anterior.

5.1. Análise do Algoritmo de Cálculo do Cosseno em MIPS Assembly

O algoritmo fornecido em MIPS assembly calcula o cosseno de um ângulo em graus usando a série de Taylor. Ele realiza a conversão do ângulo para radianos e executa os cálculos da série de Taylor, somando os termos até que um termo atinja uma precisão menor que a especificada. São utilizadas funções auxiliares separadas para melhor organização. Os testes envolveram diferentes valores de ângulos para garantir que os resultados estivessem dentro das expectativas. O algoritmo é capaz de calcular o cosseno de qualquer ângulo dentro dos limites do hardware MIPS. Quanto ao desempenho, o tempo de execução depende do número de iterações necessárias para alcançar a precisão desejada. Em geral, o desempenho é adequado para ângulos moderados, embora possa ser afetado por ângulos extremos.

6. Algoritmo Quicksort em MIPS Assembly

Tarefa: Escreva em MIPS assembly uma função denominada quicksort, que classifica um array recursivamente. Traduza a função a seguir, em código MIPS Assembly. Escreva uma função principal para chamar e testar a função quicksort.

```
void quicksort(int* array, int low, int high)
{
    int i = low, j = high;                    // low and high index
    int pivot = array[(low+high)/2];          // pivot = middle value
    while (i <= j)
    {
        while (array[i] < pivot) i++;
        while (array[j] > pivot) j--;
        if (i <= j)
        {
            int temp=array[i];
            array[i]=array[j];                // swap array[i]
            array[j]=temp;                    // with array[j]
            i++;
            j--;
        }
    }
    if (low < j) quick_sort(array, low, j);     // Recursive call 1
    if (i < high) quick_sort(array, i, high);   // Recursive call 2
}
```

Figura 3: Fragmento de Algoritmo em linguagem C. Fonte: Google Imagens

6.1. Análise do Algoritmo Quicksort em MIPS Assembly

O algoritmo feito implementa o algoritmo de ordenação quicksort para ordenar um array de inteiros. A implementação segue o algoritmo clássico do quicksort, onde um pivô é escolhido e o array é particionado em subarrays menores com elementos menores e maiores que o pivô. Em seguida, a função é chamada recursivamente nos subarrays menores até que todo o array esteja ordenado.

A função principal chama a função quicksort com os argumentos adequados e imprime o array ordenado após a ordenação. Os testes de funcionamento incluíram diferentes arrays de entrada, tanto com elementos em ordem quanto fora de ordem, para verificar se o algoritmo ordenava corretamente o array em todos os casos. Os limites de uso deste algoritmo estão relacionados ao tamanho máximo do array e à capacidade de memória do hardware MIPS.

O algoritmo é genérico e pode ordenar arrays de inteiros de qualquer tamanho dentro dos limites do hardware MIPS. Em relação ao desempenho, o algoritmo é genérico e pode ordenar arrays de qualquer tamanho, mas o desempenho pode variar dependendo da distribuição dos elementos.