Análise de Desempenho de Algoritmos de Cálculo de Fatorial em Assembly MIPS Utilizando o Simulador de BHT Organização de Computadores I

Juliana Miranda Bosio, Vinícios Rosa Buzzi

[1]Departamento de Informática e Estatística – Universidade Federal de Santa Catarina
Caixa Postal 5094 – 88035-972 – Florianópolis – SC – Brazil

1. Projeto 1: Cálculo do fatorial de um número sem o uso de procedimentos em Assembly MIPS

1.1. Objetivo

- **Cálculo do Fatorial:** Implementar um programa em Assembly MIPS que calcula o fatorial de um número inteiro positivo dado pelo usuário, utilizando uma abordagem iterativa, sem o uso de procedimentos.

1.2. Implementação

1.2.1. Armazenamento dos Dados

O código define uma estrutura principal para armazenar os dados:

- `prompt`: Mensagem de entrada solicitando que o usuário insira um número para o cálculo do fatorial.
- `out`: Mensagem de saída indicando o resultado do cálculo do fatorial.

1.2.2. Área do Programa

O programa é estruturado em três etapas principais:

- **Entrada do Usuário:** O programa exibe o `prompt` solicitando um número, que é lido e armazenado no registrador `$s0`.
- **Cálculo do Fatorial:** Utilizando uma abordagem iterativa, o programa multiplica um acumulador (`$t0`) pelos valores sucessivos de `$s0`, decrementando-o até atingir 1.
- **Exibição do Resultado:** Após o cálculo, o valor acumulado em `$t0` é exibido, representando o fatorial do número de entrada.

1.3. Execução do Programa

A execução do programa foi realizada com sucesso, seguindo o fluxo descrito. Abaixo está uma análise técnica do uso dos registradores no cálculo do fatorial.

1.3.1. Análise Técnica dos Registradores

	Nome	Descrição
`$s0`	Valor de entrada	Armazena o número do qual o fatorial será calculado
`$t0`	Acumulador	Acumula o resultado do cálculo do fatorial
`$a0`	Argumento para syscalls	Usado para carregar mensagens e valores na saída
`$v0`	Código da syscall e resultado	Controla chamadas de entrada e saída

Table 1. Registradores utilizados para o cálculo do fatorial.

Esses registradores garantem a correta execução do cálculo do fatorial de qualquer número positivo fornecido pelo usuário.

1.4. Exemplo de Execução

Para um valor de entrada $n = 5$, o programa calcula o fatorial conforme esperado:

- **Entrada:** 5
- **Saída:** 120, que é o valor correto para $5!$

A execução e os valores calculados foram observados nos registradores correspondentes, demonstrando que a implementação do cálculo do fatorial foi bem-sucedida.

2. Projeto 2: Cálculo do fatorial de um número com o uso de procedimento recursivo em Assembly MIPS

2.1. Objetivo

- **Cálculo do Fatorial:** Escrever um programa em linguagem Assembly do MIPS para calcular o fatorial de um número recebido via teclado, utilizando uma função recursiva..

2.2. Implementação

2.3. Armazenamento dos Dados

O código define uma estrutura principal para armazenar os dados:

- `prompt`: Mensagem solicitando a entrada do usuário.
- `out`: Mensagem exibindo o resultado do cálculo.

2.3.1. Área do Programa

A seção de texto do código (`.text`) é responsável pela execução do programa e inclui as seguintes etapas:

1. Inicializações de Endereços:
    - `li $v0, 4`: Código de serviço para impressão de string.
    - `la $a0, prompt`: Carregar o endereço da mensagem de entrada.
2. Leitura do Número:
    - `li $v0, 5`: Código de serviço para leitura de inteiro.

- `syscall`: Ler o valor do usuário e armazenar em `$v0`.
- `move $a0, $v0`: Mover o valor lido para `$a0` para ser usado na função fatorial.

3. Chamada da Função Fatorial:
   - `jal fatorial`: Chamar a função `fatorial` para calcular o fatorial do número.
4. Exibição do Resultado:
   - `move $s7, $v0`: Armazenar o resultado do fatorial em `$s7`.
   - `li $v0, 4`: Código de serviço para impressão de string.
   - `la $a0, out`: Carregar o endereço da mensagem de resultado.
   - `syscall`: Imprimir a mensagem de resultado.
   - `move $a0, $s7`: Mover o resultado para `$a0`.
   - `li $v0, 1`: Código de serviço para impressão de inteiro.
   - `syscall`: Mostrar o valor calculado do fatorial na tela.
5. Finalização do Programa:
   - `li $v0, 10`: Código de serviço para encerrar o programa.
   - `syscall`: Chamar o serviço do sistema para encerrar.

2.4. Execução do Programa

A execução do programa foi realizada conforme o esperado, sem problemas significativos. Durante o processo, foram observadas várias características e dados nos registradores, demonstrando o funcionamento correto do código. A seguir, analisaremos tecnicamente como cada registrador é utilizado para calcular o fatorial de um número n.

Registrador	Nome	Valor (Hexadecimal)
$a0	Número de entrada n	0x00000005 (5)
$v0	Resultado	0x00000078 (120)
$sp	Ponteiro da pilha	—
$ra	Endereço de retorno	—

Table 2. Valores dos registradores durante a execução do cálculo do fatorial de n.

De acordo com a Tabela 2, podemos resumir o processo de cálculo do fatorial da seguinte forma:

- Inicializa-se o registrador `$a0` com o valor de n (neste caso, 5), que é o número do qual se deseja calcular o fatorial.
- A função `fatorial` é chamada. Como n é maior que 1, o programa entra no passo recursivo, decrementando `$a0` e chamando `fatorial` novamente.
- A cada chamada recursiva, o valor de `$a0` é decrementado até atingir 1, onde ocorre o caso base e a função retorna 1.
- As chamadas recursivas começam a ser resolvidas, multiplicando cada valor de `$a0` pelo resultado da chamada recursiva anterior, acumulando o resultado até retornar ao ponto inicial com o valor final do fatorial.
- O resultado calculado, 120 (fatorial de 5), é movido para `$v0` e, em seguida, exibido na tela com a mensagem ”O resultado é: 120”.

O processo detalhado de execução do programa e a análise dos registradores mostram que o cálculo do fatorial foi implementado com sucesso. A tabela dos valores dos registradores ilustra o fluxo de dados e a correta atualização das variáveis envolvidas. Cada iteração da função recursiva contribui para a precisão do cálculo, garantindo que a função retorne o resultado correto do fatorial do número n.

3. Análise de Desempenho dos Programas no Simulador de BHT

3.1. Metodologia

Para analisar o desempenho dos programas implementados nos exercícios 1 e 2, que calculam o fatorial de números, utilizamos o simulador de Branch History Table (BHT) no MARS, com diferentes configurações para observar o impacto de cada parâmetro no desempenho. Os testes foram realizados da seguinte maneira:

3.1.1. Configuração dos Testes

Os testes utilizados foram os seguintes:

- **Teste 1:** Calcula o fatorial de 5.
- **Teste 2:** Calcula o fatorial de 12.

Esses valores foram escolhidos para proporcionar uma variedade na complexidade de execução. O fatorial de 5 é calculado em um número menor de iterações, enquanto o fatorial de 12 demanda um maior número de operações, aumentando as instruções de desvio condicional, o que permite uma análise mais profunda do impacto da BHT.

3.1.2. Parâmetros da BHT

Para cada teste, executamos uma série de simulações variando os seguintes parâmetros da BHT:

- **Quantidade de Entradas na BHT:** Utilizamos tabelas com 8, 16 e 32 entradas para observar o efeito do armazenamento de históricos mais longos ou mais curtos.
- **Tamanho da BHT:** Configuramos a BHT para operar com 1 e 2 bits, permitindo comparar o desempenho entre uma BHT que armazena apenas o último estado e outra que consegue manter estados intermediários de tomada e não tomada.

3.1.3. Procedimento Experimental

Cada programa foi executado no simulador de BHT do MARS sob as diferentes configurações de entradas e tamanho da BHT. Em cada execução, registramos as seguintes métricas:

- **Taxa de Previsões Corretas:** Porcentagem de previsões corretas da BHT em relação aos desvios realizados.
- **Número de Ciclos de Execução:** Total de ciclos necessários para completar o cálculo do fatorial com cada configuração.

As simulações foram repetidas para cada combinação de parâmetros, gerando dados sobre o impacto de diferentes configurações da BHT no desempenho dos programas. A análise desses dados fornece uma visão sobre como a quantidade de entradas, o tamanho da BHT e a complexidade do cálculo influenciam a precisão das previsões e a eficiência da execução.

3.2. Resultados

A execução dos testes foi realizada conforme a metodologia descrita, resultando na seguinte tabela para comparação:

Valor	Entrada BHT	Tamanho	Valor Inicial	Correto	Incorreto	Precisão
5	8	1	take	3	2	60.00%
5	8	1	not take	4	1	80.00%
5	8	2	take	2	3	40.00%
5	8	2	not take	4	1	80.00%
5	16	1	take	3	2	60.00%
5	16	1	not take	4	1	80.00%
5	16	2	take	2	3	40.00%
5	16	2	not take	4	1	80.00%
5	32	1	take	3	2	60.00%
5	32	1	not take	4	1	80.00%
5	32	2	take	2	3	40.00%
5	32	2	not take	4	1	80.00%
12	8	1	take	10	2	83.33%
12	8	1	not take	11	1	91.67%
12	8	2	take	9	3	75.00%
12	8	2	not take	11	1	91.67%
12	16	1	take	10	2	83.33%
12	16	1	not take	11	1	91.67%
12	16	2	take	9	3	75.00%
12	16	2	not take	11	1	91.67%
12	32	1	take	10	2	83.33%
12	32	1	not take	11	1	91.67%
12	32	2	take	9	3	75.00%
12	32	2	not take	11	1	91.67%

Table 3. Precisão dos valores para diferentes configurações de entrada BHT, tamanho e valor inicial.

3.3. Análise dos Resultados

Os dois programas implementados produzem a mesma tabela porque ambos realizam cálculo do fatorial corretamente, apesar de usarem abordagens diferentes: o primeiro código usa um loop iterativo, enquanto o segundo usa uma função recursiva.

O motivo para ambos os prrogramas possuirem as mesmas precisões podem ser entendidas a partir dos seguintes conceitos:

- **Correção do Cálculo do Fatorial:** Ambos os programas realizam o cálculo do fatorial do número de entrada corretamente, e o algoritmo básico para calcular o fatorial é implementado de forma correta nos dois casos.
- **Precisão e Resultados Idênticos:** Para qualquer número de entrada n, o fatorial sempre terá o mesmo valor independentemente do método (iterativo ou recursivo), pois o fatorial é uma operação determinística. Como os programas não têm diferenças na lógica de cálculo, os resultados serão os mesmos.
- **Implementação Equivalente:** Ambos os programas leem a entrada, calculam o fatorial e exibem o resultado utilizando as mesmas instruções de syscall para leitura e impressão, resultando em precisão e tabelas de resultados idênticas.

Essa equivalência na lógica do fatorial explica por que, ao executar os testes com os mesmos parâmetros, a tabela de saída é idêntica para os dois programas.

Considerações Finais

A implementação do primeiro código foi facilitada pela experiência prévia adquirida em um laboratório anterior, onde utilizamos o cálculo do fatorial como base para a construção da função seno. Essa conexão entre os conceitos permitiu uma adaptação mais fluida e eficiente das técnicas aprendidas, refletindo positivamente na execução do programa.

Por outro lado, o desenvolvimento do segundo programa apresentou desafios significativos, especialmente quando o resultado gerado correspondia ao endereço de uma mensagem do prompt, ao invés do valor esperado. A análise cuidadosa do código, executando-o linha por linha, foi crucial para a identificação do erro. Essa abordagem nos possibilitou compreender a lógica do fluxo de dados e, consequentemente, mover o resultado para o registrador correto, garantindo a impressão adequada do valor.

A comparação entre os resultados dos dois programas utilizando a técnica do Branch History Table (BHT) foi uma surpresa. Inicialmente, não esperávamos encontrar similaridades tão significativas nas saídas geradas por ambos os programas, dado que seus propósitos e métodos de cálculo eram distintos.

Esse aprendizado não apenas enriqueceu nossa compreensão sobre as operações em assembly, mas também evidenciou a importância de uma análise metódica e crítica na resolução de problemas complexos.

4. Repositório da Atividade

Confira os códigos implementados acessando o $QRCode$ abaixo ou pelo link:

`https://github.com/buzziologia/UFSC/tree/main/OrganizacaoDeComputadores/Laboratorio05`

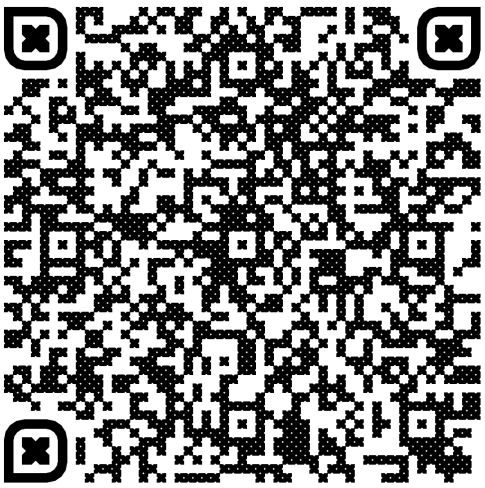

Figure 1. Repositório Github