UNIVERSIDADE ESTADUAL DE MARINGÁ
CENTRO DE TECNOLOGIA - DEPARTAMENTO DE INFORMÁTICA
GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Relatório do Resolutor de Sistemas Lineares de N Variáveis em GNU Assembly

6894 - Programação para Interfaceamento de Hardware e Software
Ronaldo Augusto de Lara Gonçalves

Ricardo Henrique Brunetto RA: 94182

Maringá
2017

Introdução

O presente documento contém relatório do primeiro trabalho da matéria ministrada pelo Ronaldo Augusto de Lara Gonçalves na disciplina de Programação para Interfaceamento de Hardware e Software para a turma de Bacharelado em Ciência da Computação de 2015. Tal documento apresenta uma estrutura de forma a expor o funcionamento dos principais módulos do programa, bem como salientar as limitações e possíveis exceções.

O conteúdo aqui citado é advindo, além das anotações em aula e materiais disponibilizados pelo professor, da gama de referências bibliográficas por ele recomendadas e encontradas. Além disso, a fundamentação teórica é baseada em um contexto não abrangido pelo escopo da disciplina.

1 Fundamentação Teórica

Tal seção busca apresentar a teoria na qual a implementação do Resolutor de Sistemas Lineares de N variáveis se baseia. Não serão abordados aspectos referentes à linguagem de programação, tendo foco específico na técnica utilizada para resolver o problema em questão. Neste ínterim, utilizou-se o **Teorema de Cramer** para a resolução do sistema linear de N variáveis e o **Teorema de Laplace** para os cálculos dos determinantes.

1.1 Teorema de Cramer

Existem diversos métodos de encontrar a solução de um sistema linear de n variáveis, caso exista, ou mostrar sua inexistência. Um dos métodos mais eficientes é a **Regra de Cramer**, mas só pode ser utilizado quando o número de equações e de incógnitas são iguais.

Assim, dadas n variáveis, deve-se ter n equações para utilizar o método. Será apresentado o funcionamento do método, mas sua prova formal será omitida por questões de complexidade.

Dado um sistema de equações lineares de n variáveis com n equações, o seguinte é valido:

$$\begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} z_1 \\ z_2 \\ \vdots \\ z_n \end{bmatrix}$$

onde cada a_ij é um coeficiente e forma matriz dos coeficientes C, x_k é uma variável e forma a matriz das variáveis X, e z_m é um termo independente e forma a matriz dos termos independentes Z. Assim, tem-se:

$$CX = Z$$

Denomina-se *matriz ampliada* a matriz $A = CZ$, ou seja, a matriz concatenada de C e Z, sendo $n + 1 \times n$.

O **Teorema de Cramer** diz que

$$x_i = \frac{det(C_i)}{det(C)}$$

onde C_i é uma matriz obtida através da substituição da coluna i pela matriz dos termos independentes Z.

1.2 Teorema de Laplace

O Teorema (ou Regra) de Laplace consiste de uma técnica que define uma fórmula recursiva para o cálculo de determinantes de matrizes de ordem n através do conhecimento imediato dos determinantes de suas **submatrizes**.

Em termos matemáticos, o Teorema de Laplace pode ser enunciado da seguinte maneira:

Dada uma matriz A de ordem n, fixa-se uma linha i de A e o determinante de A é dado por:

$$det(A) = \sum_{j=1}^{n} a_{ij}(-1)^{i+j} det(A_{ij})$$

onde A_{ij} é a **submatriz**, de ordem $n - 1$, obtida removendo-se a linha i e coluna j de A.

$$
A = \begin{pmatrix}
a_{11} & a_{12} & a_{13} & \cdots & a_{1j} & \cdots & a_{1n} \\
a_{21} & a_{22} & a_{23} & \cdots & a_{2j} & \cdots & a_{2n} \\
\vdots & \vdots & \vdots & \cdots & \vdots & \vdots & \vdots \\
\vdots & \vdots & \vdots & \cdots & \vdots & \vdots & \vdots \\
a_{i1} & a_{i2} & a_{i3} & \cdots & a_{ij} & \cdots & a_{in} \\
\vdots & \vdots & \vdots & \cdots & \vdots & \vdots & \vdots \\
a_{n1} & a_{n2} & a_{n3} & \cdots & a_{nj} & \cdots & a_{nn}
\end{pmatrix}
$$

$$
\det A = a_{1j}.\boldsymbol{A_{1j}} + a_{2j}.\boldsymbol{A_{2j}} + \cdots + a_{ij}.\boldsymbol{A_{ij}} + \cdots + a_{nj}.\boldsymbol{A_{nj}}
$$

$$
\boldsymbol{ou}
$$

$$
\det A = a_{i1}.\boldsymbol{A_{i1}} + a_{i2}.\boldsymbol{A_{i2}} + \cdots + a_{ij}.\boldsymbol{A_{ij}} + \cdots + a_{in}.\boldsymbol{A_{in}}
$$

Figura 1: Representação da aplicação da Regra de Laplace

Sua prova formal será omitida, pois foge do escopo deste trabalho e demanda conhecimentos mais avançados no âmbito matemático.

Assim, define-se a **Regra de Laplace**, através do cálculo recursivo dos determinantes. Nesse sentido, o caso base do Teorema de Laplace é quando $n = 1$ e $det(A) = |a_{11}| = a_{11}$. Contudo, há uma fórmula fechada[1] para se calcular o determinante de matrizes de ordem 2, também baseada no Teorema de Laplace.

2 Estrutura e Funcionamento

Para que se pudesse trabalhar com melhor aproveitamento de código, desempenho e legibilidade, foram criadas funcionalidades específicas para que fossem chamadas durante a execução do programa. Cada uma das funcionalidades possui uma descrição baseada em:

- **Propósito**, que descreve o que, de fato, a funcionalidade faz.
- **Pré-Condição**, que descreve o estado do programa que a funcionalidade espera para que cumpra seu propósito.
- **Pós-Condição**, que descreve o estado do programa após a execução da funcionalidade.
- **Registradores alterados**, que lista quais sofreram alterações, para que se torne mais fácil administrar o fluxo de execução.

Evidentemente, existem outras *labels* no código. Contudo, as listadas abaixo representam

[1]Entende-se por fechada a ausência da recursão.

funcionalidades específicas que são chamadas no decorrer da execução. As seguintes funcionalidades são implementadas, a saber:

proximo_campo

- **Propósito**: Avançar um campo de tamanho fixo em um endereço de memória.
- **Pré-Condição**: Endereço de memória no topo da pilha
- **Pós-Condição**: Endereço de memória retorna à pilha deslocado em 4 *bytes*.
- **Registradores alterados**: `%edi`

pular

- **Propósito**: Pular uma quantidade de campos de tamanho fixo em um endereço de memória.
- **Pré-Condição**:

  Quantidade de *bytes* em `%ebx`

  Endereço de memória no topo da pilha
- **Pós-Condição**: Avança `%eax` *bytes* no endereço de memória e o empilha.
- **Registradores alterados**: `%edi` e `%eax`

ler_n

- **Propósito**: Ler as a quantidade de equações do sistema (dimensão da matriz principal).
- **Pré-Condição**: -
- **Pós-Condição**: A quantidade de queações do sistema na variável `N`.
- **Registradores alterados**: -

inserir_fim_str

- **Propósito**: Inserir o caractere de fim de string.
- **Pré-Condição**: O endereço da string está no topo da pilha.
- **Pós-Condição**: A string contém o caractere que sinaliza seu fim.

- **Registradores alterados**: -

ler_elemento

- **Propósito**: Ler um valor inteiro do arquivo de entrada.
- **Pré-Condição**: A variável `file_descriptor` deve ter sido inicializada com o descritor do arquivo.
- **Pós-Condição**: O valor lido estará na variável `valor_lido`.
- **Registradores alterados**: -

ler_dados

- **Propósito**: Ler as entradas do sistema e preencher a matriz ampliada através de um arquivo de entrada.
- **Pré-Condição**: Endereço da matriz principal já alocado.
- **Pós-Condição**: A matriz ampliada preenchida (coeficientes e termos independentes).
- **Registradores alterados**: `%edi`, `%ecx` e `%edx`

alocar_matriz

- **Propósito**: Aloca um bloco de memória com base em valores matriciais.
- **Pré-Condição**:

  Quantidade de linhas da matriz em `%eax`

  Quantidade de colunas da matriz em `%ebx`
- **Pós-Condição**: Endereço do primeiro elemento da matriz de inteiros alocada está em `%edi`
- **Registradores alterados**: `%edi`, `%eax` e `%ecx`

mostrar_sistema

- **Propósito**: Exibe o sistema linear na tela.
- **Pré-Condição**: Endereço do primeiro elemento da matriz ampliada em `%edi`

&submatriz
det_valor (det parcial)
%ecx (coluna fixa)
%ebx (ordem da matriz)
%edi (&matriz)
&retorno

Tabela 1: Frame da chamada do procedimento Determinante

- **Pós-Condição**: -
- **Registradores alterados**: `%edi` e `%ecx`

determinante

- **Propósito**: Calcula o determinante de uma matriz através da Regra de Laplace.
- **Pré-Condição**: Endereço do primeiro elemento da matriz em `%edi` e ordem da matriz em `%ebx`
- **Pós-Condição**: A variável `det_valor` possui o determinante da matriz
- **Registradores alterados**: `%edi`, `%eax`, `%ebx`, `%ecx`, `%edx` e `esi`

Neste ponto, aborda-se uma questão inerente ao cálculo do determinante, que se baseia na Regra de Laplace (vide 1.2). Cada uma das *labels* que compõem a funcionalidade representam o cálculo de um determinante baseado na Regra de Laplace. O cálculo do determinante faz uso de uma **recursão** para decompor matrizes principais de ordens maiores ou iguais a tr?s em matrizes principais de ordem dois e, dessa forma, fazer uso da Regra de Laplace para chegar ao resultado final.

Dessa forma, é necessário construir um *frame*, ou seja, um conjunto de informações que cada chamada específica terá. Neste caso, define-se o frame da chamada como ilustra a Tabela 2.

Além disso, a cada iteração do procedimento, alocar-se-á uma submatriz de ordem $ordem(M)-1$, onde $ordem(M)$ é a ordem da matriz que fora informado por `%ebx`. Nesse contexto, tal matriz será formada pela matriz original com exceção da primeira linha (fixa para resolução do problema) e de alguma coluna (representada por `indice_fcol`).

Assim, construir-se-á N submatrizes de ordem $N-1$. Cada uma dessas submatrizes terá seu determinante calculado recursivamente. Após, fazer-se-á as N multiplicações de $-1^{i+j} \times a_{ij}$

onde i será sempre 1 (fixa-se a primeira linha) e j será algum valor entre 1 e N, dependendo da iteração em que o procedimento estiver. Por fim, a_{ij} é um elemento da primeira linha cuja coluna foi removida na formação da submatriz. Tal produto fornece o **Determinante pela Regra de Laplace**.

sinal_cofator

- **Propósito**: Retorna o sinal do cofator (-1^{i+j})
- **Pré-Condição**: Coluna fixa está em $indice_fcol$.
- **Pós-Condição**: A constante que multiplica o cofator $(-1)^(linha + coluna)$ está em `%eax`.
- **Registradores alterados**: `%eax`

submatriz

- **Propósito**: Retorna a submatriz sem a primeira linha e sem a coluna $indice_fcol$ da matriz original.
- **Pré-Condição**:

  Ordem da matriz principal está em `%ebx`

  Coluna fixa está no topo da pilha

  Endereço da matriz auxiliar está em `%esi`

  Endereço da matriz principal está em `%edi`
- **Pós-Condição**: A submatriz (desconsiderando a coluna `%ebx` e a primeira linha) de `%edi` está em `%esi`.
- **Registradores alterados**: `%eax` `%ecx` `%edx` `%edi` (`%esi` e `%ebx` são recalculados - não alteram)

gerar_matriz_sem_z

- **Propósito**: Gera a matriz sem a última coluna. No caso, sem a matriz dos termos independentes. Em outras palavras, isola a matriz dos coeficientes (matriz quadrada) da matriz ampliada.
- **Pré-Condição**: Endereço de memória da matriz auxiliar alocado e em `%esi`

- **Pós-Condição**: Copiada a matriz quadrada dos coeficientes em `%esi`
- **Registradores alterados**: `%edi`, `%esi` e `%ecx`

copiar_ultima_coluna

- **Propósito**: Substitui uma coluna da matriz pela última coluna de outra.
- **Pré-Condição**:

  Endereço da matriz principal (em geral, a matriz ampliada) em `%edi`

  Endereço da matriz auxiliar (que receberá a coluna) em `%esi`

  Índice da coluna da matriz auxiliar que será substituída em `%ebx`
- **Pós-Condição**: Avança `%eax` *bytes* no endereço de memória e o empilha.
- **Registradores alterados**: -

resolver_sistema

- **Propósito**: Resolver o sistema linear através da aplicação da Regra de Cramer.
- **Pré-Condição**:

  Endereço da matriz principal (matriz ampliada) já preenchida em `%edi`

  Determinante da matriz dos coeficientes já deve ter sido calculado e armazenado em `det_D`
- **Pós-Condição**: Resolve o sistema e exibe os resultados à medida que são calculados.
- **Registradores alterados**: `%eax`, `%ebx`, `%ecx`, `%edx`, `%edi` e `%esi`

Neste ponto, tal funcionalidade é responsável por:

- Alocar uma matriz auxiliar
- Gerar, nessa matriz recém-alocada, a matriz dos coeficientes (matriz ampliada sem a matriz dos termos independentes)
- Substituir uma das colunas (iterativo, começa substituindo a primeira e se segue até a última) pela matriz dos termos independentes (a última coluna da matriz ampliada)
- Calcular o determinante de tal matriz

- Dividir pelo determinante da matriz principal dos coeficientes (`det_valor`)
- Exibir o resultado

Dessa forma, a funcionalidade `resolver_sistema` resolve corretamente o sistema linear através da Regra de Cramer.

inicio_resolucao

- **Propósito**: Prepara o que é necessário para a execução do programa.
- **Pré-Condição**: -
- **Pós-Condição**:

  Matriz ampliada alocada e preenchida

  Determinante da matriz principal dos coeficientes calculado e armazenado em `det_D`
- **Registradores alterados**: *N/A*

Neste ponto que, caso o determinante principal (`det_D`) seja 0, o programa interrompe a execução classificando o sistema como **impossível** ou **possível e indeterminado**.

3 Limitações e Exceções

Tal programa possui as seguintes limitações:

- Calcula **unicamente** soluções para sistemas de n variáveis com n equações.
- Opera **apenas** sobre números inteiros e não faz verificações das entradas.
- Realiza divisões produzindo resultados **inteiros**.

  Dessa forma, os determinantes são arredondados, **considerando apenas suas partes inteiras**.
- Os ponteiros das matrizes auxiliares são alocados e desalocados a cada iteração da Regra de Cramer e a matriz principal é desalocada no final da resolução do sistema. Contudo, resíduos podem acabar sendo deixados na pilha a cada execução, o que, a longo prazo (bem longo), pode ocasionar estouro de memória.

- **Não faz operações com ponto flutuante**.
- **A entrada é dada por um arquivo** no seguinte formato:

$$\begin{bmatrix} c_1x_1 & c_1x_2 & \dots & c_1x_n & res_{eq1} \\ c_2x_1 & c_2x_2 & \dots & c_2x_n & res_{eq2} \\ c_3x_1 & c_3x_2 & \dots & c_3x_n & res_{eq3} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ c_nx_1 & c_nx_2 & \dots & c_nx_n & res_{eqn} \end{bmatrix}$$

onde cada c_ix_k é um coeficiente para x_k. Assim, a entrada para o sistema é uma matriz $nxn + 1$, com os n primeiros valores como coeficientes e o $n + 1$-ésimo a solução daquela equação.