1. **Introdução**

Durante a disciplina de Laboratório de Linguagem de Montagem, compreendemos as vantagens e a importância de utilizar assembly no desenvolvimento de projetos. Dentre uma série de pontos, podemos destacar a alta eficiência combinada com o baixo custo de memória que um programa trabalhado com esta linguagem possui (quando comparado a linguagens de alto nível), além da possibilidade de conversar diretamente com o processador, sem intermédios. Tudo isso faz com que o assembly, embora seja uma linguagem pouco intuitiva, necessite ser conhecida por todo programador.

1. **Objetivo**

O objetivo dessa atividade é implementar um programa simulando um editor de texto. O usuário deverá passar como parâmetro um arquivo “.txt”, e o programa deverá verificar se este existe. Se sim, o programa será responsável por carregar o arquivo na memória. Se não, deverá criar o documento. Além disso, existirão as opções “/slvr”, que permitirá ao usuário salvar qualquer alteração digitada no documento, “/sair” para encerrar o processo e “/cpal” para contar o número de palavras no arquivo de texto.

1. **Metodologia**

A linguagem de programação *assembly* foi utilizada para a realização do projeto.

Foram utilizados os registradores e as diretivas de reserva “res” para reserva de memória para variáveis e diretivas de definição “d” e para definição de constantes. Para a implementação do programa foi utilizado o editor de texto Pluma (pluma editor) e o assembler NASM (Netwide Assembler), cuja função é converter códigos em *assembly* para linguagem de máquina executável.

A implementação seguiu os seguintes passos:

1 – Declaração na *section .data* do apêndice “(cópia).txt”, bem como da mensagem que seria exibida para o usuário, informando para o mesmo entrar com o nome do arquivo a ser copiado;

2 – Declaração na *section .bss* das variáveis *arq\_in* e *arq\_copy*, com 30 bytes cada, utilizadas para manipular o nome do arquivo; *arq\_cpy\_L id\_out e id\_in,* com 4 bytes cada, utilizadas para manipular as posições de memória onde os dados do arquivo se iniciam/terminam; *info,* com 400 bytes,responsável por armazenar o conteúdo do arquivo;

3 – Declaração da *section .text*;

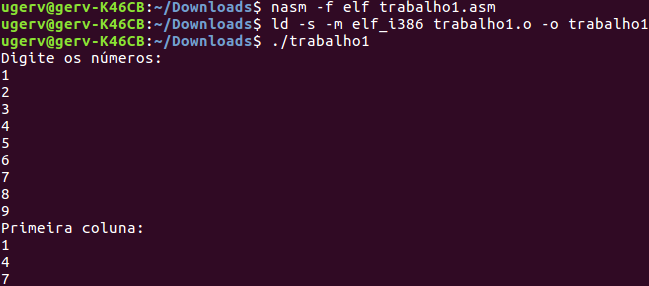
4 - Declaração da sub-rotina *coloca\_apendice*, utilizada para renomear o arquivo copiado com o sufixo “(cópia)”;

5 – Declaração da sub-rotina *coloca\_fim*, utilizada para localizar o fim do nome do arquivo (byte 10) e substituir pelo byte 0;

6 – Declaração da *\_start*; na qual vai apresentar a mensagem ao usuário, pedindo o nome do arquivo a ser copiado; em seguida, vai ler e armazenar o nome do arquivo, chamando a sub-rotina *coloca\_fim* e logo após a sub-rotina *coloca\_apendice*; logo após, ela irá: abrir o arquivo existente, ler e armazenar o conteúdo, fechar o arquivo; criar um novo, abri-lo, transcrever os dados armazenados para este arquivo e fechar; sair.

1. **Resultados**

Após terminada a implementação do programa foi obtido o executável, que foi submetido a um teste para analisar se o programa correspondia ao que foi proposto em sala. A *figura 2.1* mostra o teste realizado.

**** Figura 2.1 – Teste programa matrizes

1. **Discussão**

GERV

1. **Conclusão**

Com a execução deste projeto, pudemos colocar em prática todos os conceitos aprendidos durante a disciplina, bem como compreender como algumas funções básicas do sistema operacional funcionam. A partir disso, compreendemos ainda mais por que o uso do assembly é imprescindível nos dias de hoje, sendo de muito valor o conhecimento profundo nesta linguagem.