Grupo 1: Montador NASM

1 - Exemplifique como debugar um programa em NASM usando um software de debug

Integrantes:

- 180076272 - Jonas de Souza Fagundes
- 190031891 - Kayran Vieira de Oliveira
- 200043722 - Thais Fernanda de Castro Garcia
- 190034084 - Marcus Vinicius Oliveira de Abrantes
- 160125260 - Ingrid Lorraine Rodrigues da Silva
- 190126892 - Thiago Elias dos Reis

Introdução NASM

É um montador e desmontador que suporta as arquiteturas IA-32 e x86-64. O NASM permite o desenvolvimento de linguagem de baixo nível(Assembly) em diversas arquiteturas de sistemas operacionais, sendo mais popular para o Linux.

Vantagens de Assembly:

- Os programas são menores;
- Assembly permite criar ações de alta complexidade;
- O conhecimento em Assembly possibilita a programação nos outros tipos de linguagem;
- Boa performace.

Macros em NASM

:

- O termo "macro" refere-se a uma sequência de comandos ou instruções que são gravados ou definidos uma vez e podem ser executados repetidamente para realizar uma determinada tarefa.

- As macros no NASM são uma ferramenta poderosa para simplificar e reutilizar código assembly. Elas permitem criar blocos de instruções complexas que podem ser chamados várias vezes, evitando repetições desnecessárias e facilitando a manutenção do código. Além disso, as macros podem ajudar a tornar o código assembly mais legível e compreensível, encapsulando funcionalidades em blocos nomeados e bem definidos.

:

- Macros de uma única linha
  - %define
- Macros multilinhas:
  - %macro
  - %endmacro

:

- O que o processador de macros do NASM é capaz de suportar?
  - Montagem condicional
  - Múltiplos níveis de inclusão de arquivos
  - Mecanismo de pilha de contexto para controlar macros

Exemplificação de como debugar um programa em NASM usando o software de debug GDB

GDB (GNU Debugger)

O GDB é um programa que roda outros programas e permite que o usuário analise estes programas para resolver possíveis problemas no código
O GDB permite:

- Inspecionar o código usando de ‘breakpoints’ para que o código pare em certa partes e seja possível analisar o comportamento do código.
- Ver o que tem em uma variável durante um certo momento da execução.
- Execução passo a passo, para evitar vários ‘prints’ entre linhas do código e saber, exatamente, o que está ocorrendo em cada momento do código.
- Mudanças em tempo real para entender o comportamento do código.

GDB (GNU Debugger)

Comandos mais utilizados:

- **break** - permite que escolhamos em que linha o código será parado. No assembly, precisa ser em uma label.
- **run** - roda o programa depois da definição dos breakpoints.
- **stepi/nexti** - permite ir para a próxima instrução do assembly. Por si só, o next pulará linhas de acordo com o C e uma linha de C pode significar múltiplas de assembly.
- **continue** - permite avançar até o próximo breakpoint.
- **finish** - termina a função atual.