TDE A - Assembly dos Compiladores

Objetivo

O trabalho tem como finalidade **analisar o código assembly gerado pelo compilador GCC em diferentes níveis de otimização**, identificando como variáveis, parâmetros e estruturas de controle do programa em C são traduzidas para a linguagem de baixo nível.

Formação dos Grupos

- O trabalho pode ser desenvolvido individualmente ou em grupos de até **3 integrantes**.
- Em caso de grupo, somente **um membro** deve submeter a resposta.

Programa de Referência (em C)

O programa abaixo implementa a soma dos elementos de um vetor por meio de uma função:

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

#define TAM_VETOR 5

int somaVetor(int vetor[], int elementos){
    int j, acum = 0;
    for (j = 0; j < elementos; j++)
        acum += vetor[j];
    return acum ;
}

int main() {
    int i, soma, numeros[TAM_VETOR];
    srand(time(NULL));
    for (i = 0; i < TAM_VETOR; i++)
        numeros[i] = rand();
    soma = somaVetor(numeros, TAM_VETOR);
    printf("%d", soma);
    return 0;
}
```

Orientações Gerais para a Geração do Assembly

1. **Ambiente de desenvolvimento**
    - Pode ser utilizada qualquer IDE (Dev C++, Code::Blocks, etc.) que suporte o compilador GCC. No caso do sistema operacional Linux, é possível rodar diretamente na linha de

comando.

2. **Configuração de compilação**
    - Compile em **arquitetura 32 bits** e em **versão release** (sem depuração).
    - Isso garante que o assembly utilizará os registradores da arquitetura (EAX, EBX, ECX, EDX, ESP, EBP, EDI e ESI).
3. **Limpeza do código**
    - Remova linhas iniciadas por ponto `"."` (ex.: `.text`, `.globl`, `.ident`).
4. **Padrão de sintaxe**
    - O assembly pode ser gerado em dois padrões de sintaxe:
        - **Intel**: destino à esquerda. Por exemplo: `mov eax, 2`
        - **AT&T**: destino à direita, registradores com `%` e constantes com `$`. Por exemplo: `mov $2, %eax`
    - É obrigatório que o código seja gerado no **padrão Intel**.

Atividade Código Sem Otimização

1. **Geração do Código Assembly sem Otimização:**
    - Se você estiver utilizando uma IDE (como Dev C++, Code::Blocks, etc.), adicione a seguinte opção de compilação. Ao compilar, o arquivo gerado conterá o código assembly correspondente ao seu programa.
        - `-S -masm=intel -O0 -m32`
    - Se você está usando Linux ou MinGW, use o comando:
        - `gcc -m32 -S -O0 -masm=intel programa.c -o programa_O0.s`
    - As opções representam:
        - `-S` pede somente o assembly (sem linkagem).
        - `-O0` sem otimização.
        - `-masm=intel` força sintaxe Intel (destino à esquerda).
        - `-m32` para gerar código com registradores da arquitetura 32 bits (EAX/EBX/ECX/EDX/EBP/ESP/ESI/EDI)

2. **Limpeza do Código Assembly**

   Ao abrir o arquivo `programa_O0.s` (ou `programa.exe`), você encontrará várias diretivas e informações extras, por exemplo:

   ```
   .file "programa.c"
   .intel_syntax noprefix
   .text
   .globl main
   .type main, @function
   ```

   Essas linhas que **começam com ponto (`.`)** são metadados de montagem, e **não devem ser incluídas na análise**.

3. **Enumeração das Linhas do Código**

   Para facilitar a análise do código assembly, todas as linhas do código limpo devem ser **enumeradas**. Isso ajuda na referência das instruções durante a explicação e na comparação com o código C. Para enumerar as linhas, você pode utilizar o serviço online gratuito disponível em:

   https://www.browserling.com/tools/number-lines