Trabalho de Programação 3

Processador Intel (80x86)

1. Descrição Geral

Neste trabalho você deverá desenvolver um programa capaz de ler um arquivo (arquivo de entrada) e calcular um código de verificação ou verificar se o arquivo corresponde a um código de verificação fornecido.

O nome do arquivo de entrada será fornecido na linha de comando assim como as opções de calcular o código de verificação ou verificar se o código fornecido corresponde ao arquivo.

O arquivo de entrada pode ter qualquer nome válido no MS-DOS. Além disso, o arquivo deverá ser tratado como uma sequência de bytes.

Seu programa deverá ser desenvolvido em linguagem simbólica de montagem do processador 8086 da Intel e executado no ambiente DosBox. Para montagem de seu programa fonte será usado o montador MASM 6.11.

2. Código de Verificação

O código de verificação que seu programa deverá operar será calculado através do seguinte algoritmo:

1) Definir “X” como uma variável de 64 bits sem sinal.
2) Definir “V[]” como o vetor de bytes que representa o arquivo. O elemento V[0] é o primeiro byte do arquivo.
3) Definir “N” como o número de bytes existentes no arquivo.
4) Fazer X = 0
5) Fazer FOR I = 0 até N-1: X = X + V[I]
6) O código de verificação estará na variável “X”

3. Linha de Comando

Seu programa será chamado na linha de comando usando duas de três opções possíveis. As opções são as seguintes:

- “-a” seguida do nome de um arquivo, para informar o nome do arquivo de entrada. Essa opção sempre estará presente na linha de comando;
- “-g”, para informar que o programa deve calcular o código de verificação. Essa opção só estará presente se o programa deve calcular o código de verificação;
- “-v” seguida de um código de verificação, para informar que o programa deve verificar se o arquivo de entrada tem o mesmo código de verificação informado na linha de comando. Essa opção só estará presente se o programa deve verificar a validade de um código de verificação. O código de verificação deve ser fornecido em hexadecimal.

As opções podem aparecer em qualquer ordem na linha de comando. Supondo que o programa tenha o nome de “app” e que o arquivo de entrada tem o nome “file.txt”, são exemplos de linhas de comando as seguintes:

- app -a file.txt -g
- app -v A94F3218C4766A90 -a file.txt
- app -g -a file.txt
- app -a file.txt -v a94f3218C4B66a90

Ver a seção 6 para informações de como obter a linha de comando em um programa escrito em assembler.

4. Entregáveis: o que deve ser entregue?

Deverá ser entregue, via Moodle da disciplina, **APENAS** o arquivo fonte com a solução do problema apresentado, escrito *na linguagem simbólica de montagem* dos processadores 80X86 da Intel (arquivo .ASM). Além disso, esse programa fonte deverá conter comentários descritivos da implementação.

Para a correção, o programa será montado usando o montador **MASM 6.11** no ambiente **DosBox 0.74** e executado com diferentes arquivos e frases de entrada. A nota final do trabalho será proporcional às funcionalidades que forem atendidas pelo programa.

O trabalho deverá ser entregue até a data prevista, conforme programado no MOODLE. **Não será aceita a entrega de trabalhos após a data estabelecida**.

5. Observações

Recomenda-se a troca de ideias entre os alunos. Entretanto, a identificação de cópias de trabalhos acarretará na aplicação do Código Disciplinar Discente e a tomada das medidas cabíveis para essa situação (**tanto o trabalho original quanto os copiados receberão nota zero**).

O professor da disciplina reserva-se o direito, caso necessário, de solicitar uma demonstração do programa, onde o aluno será arguido sobre o trabalho como um todo. Nesse caso, a nota final do trabalho levará em consideração o resultado da demonstração.

6. Como obter a linha de comando?

O string escrito na chamada "linha de comando" pode ser lido por um programa escrito em assembler. Esse string está no PSP – Program Segment Prefix, que se encontra em um segmento específico da memória. Nesse segmento, o string pode ser encontrado a partir do offset 81H. O final do string é identificado pelo byte CR (0DH).

Ainda, no offset 80H pode-se encontrar o tamanho do string digitado na linha de comando, em bytes.

O segmento onde se encontra o PSP está presente nos registradores DS e ES, logo no início da execução do programa.

Entretanto, quando se usa o modo simplificado do MASM (com as diretivas "ponto"), o DS será carregado com o segmento de dados do programa. Assim, a informação do PSP só estará presente no registrador ES.

Portanto, se for desejado usar o ES sem perder as informação do PSP, é necessário salvar a informação do ES, antes de realizar qualquer alteração no ES.

Exemplo: No exemplo abaixo está sendo considerado que o programa será montado pelo MASM no modelo simplificado. Esse trecho de programa deve ser colocado no início do programa pois ele utiliza os valores fornecidos pelo Sistema Operacional nos registradores de segmento DS e ES. A instrução "rep movsb" é responsável por copiar a linha de comando do PSP para uma variável de nome "VAR" no segmento de dados do programa.

```
        push    ds          ; salva as informações de segmentos
        push    es

        mov     ax,ds       ; troca DS <-> ES, para poder usa o MOVSB
        mov     bx,es
        mov     ds,bx
        mov     es,ax

        mov     si,80h      ; obtém o tamanho do string e coloca em CX
        mov     ch,0
        mov     cl,[si]

        mov     si,81h      ; inicializa o ponteiro de origem

        lea     di,VAR      ; inicializa o ponteiro de destino

        rep     movsb

        pop     es          ; retorna as informações dos registradores de segmentos
        pop     ds
```