SOFTWARE BÁSICO – TRABALHO 2

TRADUÇÃO PARA IA32

VICTOR HUGO MARQUES VIEIRA – 150047649

Resumo:

Neste trabalho é desenvolvido um software de tradução da pseudo-linguagem assembly estudada em sala de aula para IA32 Nasm, com acréscimo de funções básica de entrada e saída de strings.

Desenvolvimento:

O tradutor foi desenvolvido em C/C++ em ambiente Windows utilizando o compilador do CodeBlocks versão 17.12.

O programa recebe um arquivo objeto (.obj) da pseudo-linguagem vista em sala de aula, e retorna uma arquivo de mesmo nome com extensão alterada (.s):

Exemplo:

>>.*tradutoIA32.exe soma.obj*

Cria-se na mesma pasta "soma.s" com a tradução para IA32 Nasm.

Adaptações:

Como a linguagem origem na tradução é mais simples, possui somente um acumulador, optou-se utilizar o EDX como acumulador. Isso facilita sempre saber os estão os resultados, somente precisando realizar realocação em funções que precisam do edx, como multiplicação e divisão.

Como requisitado, são implementadas as funções de entrada (sinput) e saída (soutput) de string em IA32. O tradutor ao encontrar os códigos destas ações, 15 e 16 respectivamente, chama a função correspondente em IA32. Estas funções são adicionadas ao arquivo de saída independentemente de elas terem sido chamadas ou não no programa.

As funções implementadas recebem argumento via pilha, e devolvem via registrador EAX.

Para distinguir os espaços alocados no código objeto pela função sinput, de outros alocamentos no mesmo programa, alterou-se a anotação de alocamento para esta função. A expressão:

sinput var1, 5

(...)

Var1: space 5

Converte-se para:

15 [end Var1], 5

(...)

-1 5 ;alocação na memória

Ou seja, o -1 indica que o próximo número é a quantidade de bytes a serem alocado para aquele endereço.

Desta forma os arquivos objeto construídos com o trabalho 1 não funcionam para este tradutor. Entretanto os mesmos testes utilizados no primeiro trabalho foram implementados novamente, e testados utilizando o *simulador.exe* fornecido. Todos os testes foram bem sucedidos. Para dúvida podem ser executados os testes chamando o simulador. Exemplo:

>>.*simulador.exe fatorial.obj*

Um problema encontrado na proposta da função sinput e soutput, é que elas leem uma quantidade fixa de caracteres, e não até encontrar um linefeed. Desta forma ao escrever pode ser que a variável recebe itens não pretendidos.

Alterando isso estas funções ficariam bem similar ao scanf e printf em sua forma mais básica, porém não foram implementadas estas mudanças pois não estava na proposta do trabalho.

Não foi implementado o handle para overflow na multiplicação. Isto pois mesmo foquei mais tempo na implementação dos testes, o que foi bom já que estes foram bem sucedidos.

Para cada processo de escrita ou leitura do teclado é impressa uma mensagem

Funções implementadas:

As funções implementadas em IA32 estão em uma arquivo "baseInputOutput.txt" que precisa estar no mesmo diretório que o tradutor no momento da tradução.

As funções implementas contidas neste arquivo são:

- input: esta função não recebe. Faz a leitura de um valor até encontrar enter, retornado a quantidade de bytes lidos em EAX e o valor lido em EBX. Caso o primeiro caractere lido seja um '-' o valor será negativo. Não testa entradas incorretas pelo usuário.
- Output: recebe na pilha o valor a ser impresso, e retorna em EAX a quantidade de bytes escritos. Desempilha os argumentos recebidos.
- input: recebe na pilha respectivamente o endereço para escrever e a quantidade de bytes a serem lidos. Desempilha os argumentos recebidos.
- output: recebe na pilha respectivamente o endereço para ler e a quantidade de bytes a serem escritos. Desempilha os argumentos recebidos.
- impQtdBytes: faz parte da tradução das funções anteriores, sendo chamado logo após delas , enquanto ainda de sem no EAX a quantidade de bytes. Esta função recebe em EAX esta quantidade e imprimi utilizando a função output. Não recebe argumentos na pilha.
- lineFeedD: imprimi uma quebra de linha quando necessário. Não recebe argumentos na pilha.

Testes:

Para facilitar a exportação dos testes para outra máquina, os testes realizados foram rodados em um simulador online.

https://www.jdoodle.com/compile-assembler-nasm-online/

Optou-se, pois, este simulador simula um compilador nasm. Para não ter problemas na transcrição das funções de entrada e saída de dados está foi a solução escolhida. Para replicar os testes apresentados nas pastas vão junto a este arquivo.

Foram abordados diversos testes diferentes, cada qual com inputs diferentes. A pasta para cada teste que acompanha esse readMe, com o log manual do resultado dos testes:

- 1_TESTE_SOMA: recebe dois números e apresenta a soma.
- 2_TESTE_OPERACOES: recebe dois números e apresenta o resultado para as quatro operações entre eles, soma, subtração, multiplicação e divisão.
- 3_TESTE_AREA_TRIAGULO: recebe a base e altura de um triângulo, e retorna sua área. (bxh)/2.
- 4_TESTE_SINPUT_SOUTPUT: recebe duas strings de 5 caracteres e depois imprime as duas subsequentemente.
- 5_TESTE_FATORIAL: recebe um número e apresenta seu fatorial. Atenção ao overflow da multiplicação.
- 6_TESTE_FIBONACI: recebe um número e apresenta a série de fibonaci até o último menos que ele.

Estrutura do código:

O código foi desenvolvido em três módulos:

- tradutoIA32: módulo principal que deve ser chamado, lidando com os arquivos de entrada e saída.
- functions: módulo responsável por selecionar as funções no assembly inventado. Adiciona também as funções em desenvolvidas em IA32, e as seções de .data e .bss do arquivo final.
- functionsSelec: módulo que implementa a substituição das funções originais pelos seus equivalentes.

Imagem da relação dos módulos.