Trabalho Prático 2
Software Básico
Prof. Bruno Macchiavello

1 Introdução

O trabalho consiste em implementar em C/C++ e IA-32 um realizar tradutor

O trabalho pode (e recomenda-se) ser feito em grupo de 2 alunos. Deve ser entregue somente o código e um arquivo README. O arquivo README deve indicar os nomes dos alunos, SO utilizado, como compilar o programa e como rodar (a princípio seguir a indicação da especificação). Sendo que se for LINUX deve-se utilizar o GCC, se for Windows deve ser o CODEBLOCKS. Apple OS será tratado como programa em LINUX. Recomenda-se rodar em 2 computadores diferentes antes de enviar o trabalho.

Não é permitido o uso de bibliotecas ADICIONAIS. Pode ser utilizado qualquer padrão de C ou C++. Somente UM dos alunos da dupla deve enviar o trabalho pelo APRENDER.

NÃO É PERMITIDO O USA DA IO.MAC NA PARTE DE IA-32.

2 Especificação

O trabalho deve traduzir um código assembly inventado visto em sala de aula para o assembly IA-32 padrão NASM. O arquivo de entrada é um arquivo texto com extensão ASM. O arquivo de saída deve manter o mesmo nome e trocar a extensão por .S

O arquivo de entrada pode ter EQU e IF. Para isso use a parte de pre-processamento do trabalho 1. Não tera macros, nem erros nos arquivos. O formato do arquivo ASM deve ser o mesmo do trabalho 1, ou seja dividido em seções.

O executável deve ser chamado de TRADUTOR.

O tradutor deve manter as seguintes características do trabalho anterior:

- Aceitar Maiúsculas e Minúsculas (não ser sensível ao CASO)
- A diretiva CONST deve aceitar positivos, negativos e hexa no formato 0x (ex: 0x12). .
- O comando COPY deve separar os argumentos por “,” SEM espaço
- Desconsiderar todos os espaços, tabulações ou enter desnecessários.
- Pode dar rótulo seguido de dois pontos e ENTER. O rótulo é considerado como da linha seguinte
- SPACE pode aceitar argumentos. Logo é possível fazer rótulos como X+2 (sem espaços)

- Aceitar comentário em qualquer parte do código iniciado por ; (o comentário deve ser removido no pré-processamento de EQU e IF)
- Assumir que a seção DATA sempre esta no final.
  .

O tradutor deve gerar um código equivalente em IA-32. Sendo que NÃO é necessário manter a ideia de UM único registrador acumulador. Também assumir que tudo no assembly inventado ocupa 32 bits. Ou seja: SPACE 2, reserva 2 DOUBLE WORDS ou 8 bytes no IA-32.

SPACE deve ser traduzido dentro da seção BSS e CONST dentro da seção DATA. Note que as variáveis da seção data só podem ser UM número, já que não existe como declarar array na diretiva CONST no assembly inventado. Porém, agora deve permitir TAMBÈM declarar um número de UM byte usando:

Letra: CONST ‘A’

Note que ao trabalhar com caracteres e strings usamos somente UM byte. Estes casos representam a exceção de assumir que tudo é 32 bits.

Além disso além das instruções INPUT e OUTPUT, agora o assembly inventado tem INPUT_C, OUTPUT_C, INPUT_S, e OUTPUT_S. Input e Ouput são usados para números decimais positivos ou negativos de 32 bits.

Output_C e Input_c serve para mostrar no monitor e ler um byte em formato ASCII (um único caracter). O modo de uso é similar ao Input e output normal, recebem UM argumento que é um endereço de memória.

Output_C e Input_c serve para mostrar no monitor e ler um array de bytes em formato ASCII (string). O modo de uso é: recebem DOIS argumentos separados por vírgula (similar ao COPY) sendo que o primeiro é um endereço de memória e o segundo e a quantidade de bytes a ser lidos/escritos. Ex.: INPUT Label,5.

As instruções de INPUT devem ser traduzidas como chamadas a funções em IA-32. As funções devem receber parametros pela pilha e retornar saída pelo EAX. As funções devem usar as chamadas ao sistema vistas em sala de aula para realizar a entrada e saída de dados. O valor de retorno de todas as funções é a quantidade de bytes lidos ou escritos. A função ao finalizar antes de retornar ao programa principal deve mostrar a mensagem: Quantidade de Bytes lidos/escritos = X.

Note que as funções devem ser programadas de antemão. O tradutor vai só copiar elas no inicio do seu programa em IA-32. Pode copiar as funções mesmo não sendo usadas.

2 Teste

Os testes serão feitos compilando e ligando o arquivo .S com NAMS e LD em LINUX.