Trabalho Prático 1
Software Básico
Prof. Bruno Macchiavello

1 Introdução

O trabalho consiste em implementar em C/C++ um realizar um macro-montador

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 príncipio 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

2 Especificação

O trabalho deve traduzir um código assembly. No assembly criado em sala de aula. O arquivo de entrada é um arquivo texto com extensão ASM. O arquivo de saída é um arquivo texto com uma única linha com o código objeto. Para a diretiva SPACE deve ser colocado 0 (zero), NÃO colocar XX. Exemplo:

12 29 10 29 4 28 11 30 3 28 11 31 10 29 2 31 11 31 13 31 9 30 29 10 29 7 4 14 2 0 0 0

O executável deve ser chamado de MONTADOR. E deve ter 3 modos de uso. Deve ser possível chamar por linha de comando da seguinte forma: ./montador -<op> <arquivo>

O arquivo de entrada deve ser indicado SEM extensão e o arquivo de saída deve MANTER o mesmo nome e mudar a extensão. Deve existir 3 formas de operação:

-p: A extensão da saída deve ser .PRE e somente deve se processar EQU e IF
-m: A extensão da saída deve ser .MCR e somente deve se processar MACROS. A entrada vai ser a saída pré-processada
-o: A extensão da saída deve ser OBJ e deve seguir o formato indicado anteriormente. A entrada vai ser a saída das Macros.

O montador deve ter as seguintes características:

- Aceitar Maiúsculas e Minúsculas (não ser sensível ao CASO)

- A diretiva CONST deve aceitar positivos, negativos e hexa no formato 0x (ex: 0x12). No arquivo de saída OBJ tudo deve estar em decimal.
- 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 argumento. 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).

O montador deve verificar os seguintes erros, somente ao criar o OBJ. Assumir que nunca vai ter erro no uso de INPUT, IF e nas diretivas de MACROS

- Dois rótulos na mesma linha
- Rótulo não definido
- Dado não definido
- Quantidade de argumentos errada
- Seção TEXT faltante
- Instrução ou diretiva inexistente
- Erros léxicos (caracteres especiais ou ´número inicial nos rótulos)

Deve indicar a linha do erro é se é LÉXICO, SINTÁTICO ou SEMÂNTICO. O código vai estar separado em TEXT e DATA. Utilizando a diretiva SECTION (conforme exemplo no aprender). A seção DATA sempre depois.

EQU e IF: EQU vai estar fora da seção TEXT no início do arquivo. Pode ter até 2 inputs. O IF sempre precisa de um rótulo definido como EQU. EQU pode ser utilizado para argumentos do SPACE E CONST. Pode ter vários IFs no código. Não tem IFs nas Macros.

MACROS: Deve aceitar argumentos utilizando o caracter “&” conforme padrão dos slides de aula. Até 3 argumentos como máximo. No máximo tem 2 macros no código, uma macro pode chamar a outra.

2 Teste

No aprender tem o SIMULADOR para rodar somente colocar o arquivo de entrada (com extensão por linha de comando, ex: ./simulador entrada.obj). A entrada deve ser os arquivos de texto OBJ. O simulador espera que o usuário saiba rodar o programa, ou seja qualquer entrada de INPUT o cursor vai esperar o usuário digitar o número, sem mostrar nenhuma mensagem. O simulador ajuda a verificar se o arquivo objeto está correto.