Trabalho Prâtico 1
Software Bâsico
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 codigo e um arquivo README. O arquivo README deve indicar os nomes dos alunos, SO utilizado, como compilar o programa e como rodar em linha de comando (mesmo se for igual a indicação desta especificação). Se foi feito em LINUX o compilador DEVE ser GCC, se for Windows deve ser o CODEBLOCKS (já que usa o GCC do mingw). Apple OS será tratado como programa em LINUX. Recomenda-se rodar em 2 computadores diferentes antes de enviar o trabalho. Se for usar o VS CODE modificar a configuração dele de tabulação que normalmente ele modifica por espaços ao abrir o arquivo, isso vai dar problema nos testes.

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

2 MONTADOR

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 caso não procisar de ligação e uma UNICA linha com o código objeto. Para a diretiva SPACE deve ser colocado 0 (zero) ou 00, NAO 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

0 executável deve ser chamado de MONTADOR. E pode criar arquivos adicionais ao de sapida (como por exemplo um arquivo pre-processado, que eu fortemente recomendo).

Deve ser possível chamar por linha de comando da seguinte forma:
./montador <arquivo>

O arquivo de entrada deve ser indicado SEM extensão e o arquivo de saída deve MANTER o mesmo nome e mudar a extensão de ASM para EXC.

NÃO precisa ser capaz de processar diretivas como:
- EQU e IF
- ORG
- MACROS

O montador deve ter as seguintes características:

- Aceitar Maiúsculas e Minúsculas (não ser sensível ao CASO, dica transforme tudo para maisculas num arquivo pre-processado e depois use ele)
- utilizar passagem UNICA (inclussive para arquivos que precisem ser ligados)

- A diretiva CONST deve aceitar positivos, negativos e hexa no formato 0x (ex: 0x12). No arquivo de saída EXC tudo deve estar em decimal (ou seja se eu colocar CONST 0x30, deve parecer o número 48 no arquivo EXC).
- O comando COPY deve separar os argumentos por “,” COM espaços antes e depois da “,”(ex: COPY A1 , A2)
- Desconsiderar todos os espaços, tabulações ou enter desnecessários em qualquer parte do código (dica tirar no pre-processamento, isto é importante porque vários testes possuem esse tipo de coisas, para verificar se o scanner esta funcionando corretamente)
- SEMPRE tera UM espaço no mínimo entre TOKENS, com exceção do “:” (ex. R1: INPUT N1)
- Pode dar rotulo seguido de dois pontos e ENTER. O rotulo é considerado como da linha seguinte (dica coloca na mesma linha no arquivo pre-processado)
- SPACE pode aceitar argumento. Logo é possível fazer rótulos como X + 2 (com espaços)
- Aceitar comentário em qualquer parte do código iniciado por ;.

O montador deve verificar os seguintes erros (mesmo tendo erros recomenda-se criar o arquivo EXC)

- Dois rotulos na mesma linha
- Rótulo não definido na seção de TEXT
- Dado não definido na seção DATA
- Seção TEXT faltante
- EXTERN e PUBLIC em arquivo que NAO tenha BEGIN e END
- Erros léxicos (caracteres especiais fora o "_" (underscore) ou "numero inicial nos rótulos)

Deve indicar a linha do erro é se é LÉXICO, SINTÁTICO ou SEMÂNTICO. (se você gerar arquivo pre-processado, pode ser a linha do arquivo pre-processado. Indicar isso no README) O código vai estar separado em TEXT e DATA. Utilizando a diretiva SECTION (conforme exemplo no TEAMS. A seção DATA pode vir antes ou depois no ASM, mas no OBJ sempre os dados devem ir no FINAL, ou seja o código deve começão com o OPCODE da primeira instrução

4. LIGADOR

Primeiro para ligação o MONTADOR deve aceitar na linha de comando até 4 arquivos ASM sem extenção. (ex. ./montador teste1 teste2 teste3 teste4)

Os arquivos agora vao ter BEGIN, END e EXTERN e PUBLIC. As direticas EXTERN e PUBLIC sempre estarão no inicio antes de qualuer uma das seções de DATA ou TEXT. (ver exemplo no teams)

O arquivo de saída objeto do Montador para cada um dos arquivos ASM deve mantar o nome e trocar a extensão por .OBJ (notar que é diferente ao caso SEM ligação). Ver exemplo do formato no teams (o arquivo neste único caso não representa um código real, é só para ver o formato). Na parte USO, deve ser coloca os endereços da tabela de uso, na DEF a tabela de definições, na REF os endereços de todos os valores relativos e na CODE o código traduzido antes de ligar.

O ligador, deve chamar LINKER. Deve receber por linha de comando os TODOS os arquivos objeto. E dar como saída o arquivo EXC com o nome do primeiro arquivo OBJ. O formato do EXC deve ser igual ao já indicado.

4. Teste e exemplos

No Teams na aba de arquivos tem uma pasta trabalho 1 onde tem os exemplos e o SIMULADOR. Para rodar o SIMULADOR deve-se somente colocar o arquivo de entrada (COM extensão por linha de comando, ex: ./simulador entrada.exc). A entrada deve ser os arquivos de texto EXC. 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 esta correto. Simulador não funciona com arquivos ainda NAO ligados.