

1º Trabalho Prático
CIC 116432 – Software Básico
Prof. Bruno Macchiavello
2º Semestre de 2018

1 Introdução

O trabalho consiste em implementar em C/C++ um método de tradução de uma linguagem de montagem simples para uma representação de código objeto. O tradutor a ser implementado será um Assembler da linguagem hipotética vista em sala de aula.

2 Objetivo

Fixar o funcionamento de um processo de tradução. Especificamente as etapas de análise léxica, sintática e semântica, a etapa de geração de código objeto e ligação.

3 Especificação

3.1 Montador

A linguagem de montagem utilizada será a linguagem simbólica hipotética apresentada em sala. Esta linguagem é formada por um conjunto de apenas 14 instruções. Uma diferença com o formato visto em sala de aula é que os programas devem ser divididos em três seções: uma de código (TEXT), dados declarados (DATA) e dados não declarados (BSS).

Para cada instrução da máquina hipotética, a Tabela 1 abaixo contém o mnemônico, quantidade de operandos, código de operação utilizado na montagem, tamanho em palavras da instrução montada e uma breve descrição da sua utilidade. As linhas finais da tabela definem as diretivas.

Os identificadores de variáveis e rótulos são limitados em 50 caracteres e seguem as regras comuns da linguagem C, sendo compostos por letras, números ou o caractere *_* (*underscore*) e com a restrição de que o primeiro caractere não pode ser um número.

Para eliminar ambiguidade, as seções de código e dados devem ser devidamente marcadas com as diretivas correspondentes, como ilustra o exemplo abaixo:

```

SECTION TEXT
ROT: INPUT N1
      COPY N1, N4 ;comentario qualquer
ROT2:
      COPY N2, N3
      COPY N3, N3 + 1
      OUTPUT N3 + 1
      STOP
;comentario qualuer
SECTIO DATA
N2:  CONST -5
SECTION BSS
N1:  SPACE
N3:  SPACE 2
N4:  SPACE

```

O montador deve ser capaz de:

- NÃO ser sensível ao caso, podendo aceitar instruções/diretivas/rótulos em maiúsculas e minúsculas.
- NÃO ter ordem fixa entre as seções DATA e BSS.
- A seção de TEXTO deve sempre vir antes das seções DATA e BSS
- Gerar um arquivo de de saída em formato TEXTO (mais detalhes serão descritos a seguir).
- Desconsiderar tabulações, quebras de linhas e espaços desnecessários em qualquer lugar do código.
- Aceitar um Rotulo definido em linhas anteriores (ver exemplo acima de ROT2)
- A diretiva CONST deve aceitar números positivos e negativos (inteiros e hexadecimal);
- Deve ser possível trabalhar com vetores (SPACE com operando, e usar operações do tipo: LABEL + Número)
- Capacidade de aceitar comentários indicados pelo símbolo “;”
- O comando COPY deve utilizar uma vírgula e um espaço entre os operandos (COPY A, B)
- Ser capaz de aceitar MACROS (mais detalhes serão descritos a seguir).
- Poder criar um rótulo, dar quebra de linha e continuar a linha depois (o rótulo seria equivalente a linha seguinte)

- Identificar erros durante a montagem. Montado sempre o programa inteiro e mostrando na tela a(s) LINHA(S) e TIPO DOS ERROS (segundo a relação a seguir e indicar se é LÉXICO, SINTÁTICO OU SEMANTICO). A linha deve ser em relação ao programa original, e iniciar com a linha 1. O programa deve pelo menos detetar os seguintes tipos de erro:
 - declarações e rótulos ausentes;
 - declarações e rótulos repetidos;
 - pulo para rótulos inválidos;
 - pulo para seção errada;
 - diretivas inválidas;
 - instruções inválidas;
 - diretivas ou instruções na seção errada;
 - divisão por zero (para constante);
 - instruções com a quantidade de operando inválida;
 - tokens inválidos;
 - dois rótulos na mesma linha;
 - seção TEXT faltante;
 - seção inválida;
 - tipo de argumento inválido;
 - modificação de um valor constante;

O programa de tradução deve ser capaz de realizar as fases de análise e síntese, mantendo informação intermediária armazenada em estruturas de dados. A escolha apropriada de estruturas de dados faz parte do escopo do trabalho. Não é obrigatório o uso de Hashing, nem ordenação de tabelas. O grupo pode escolher utilizar o algoritmo de duas passagens ou passagem única. O montador deve ser capaz de avaliar as diretivas EQU e IF, numa passagem inicial. Assumir que a diretiva EQU será sempre definida no início do program ANTES da seção de TEXTO.

O programa de montagem (DEVE ser chamado 'montador.c') deve receber um argumentos em linha de comando (nessa ordem): um arquivo de entrada contendo um programa em *Assembly* em formato texto (só nome, assume-se a extensão “.asm”) na linguagem hipotética. O programa deve dar DUAS saídas. Uma saída (*.pre) com o programa pré-processado, onde devem ser retirados os comentários e avaliadas as diretivas EQU e IF. E uma outra saída (*.obj) contendo o arquivo objeto. O nome dos arquivos de saída deve manter o mesmo nome do arquivo de entrada.

O arquivo objeto deve estar em formato TEXTO. Se o programa for um módulo, o arquivo de saída deve indicar a tabela de USO, tabela de DEFINIÇÕES e os OPCODES e operandos sem quebra de linha, nem endereço indicado, separados por espaço. As

diferentes informações devem estar separados por os indicadores TABLE e CODE. Além de ter uma seção indicando quais códigos são relativos (no exemplo abaixo os códigos 1,3 e 4 que correspondem a 12,15,04 são relativos). Se não for um módulo o arquivo de saída deve ser OPCODES e operandos sem quebra de linha: nem endereço indicado, separados por espaço, SEM O INDICADOR DE SEÇÃO “CODE”, sem a indicação de absoluto e relativo, sem tabelas (ou seja, somente uma linha de números). Ao usar SPACE colocar ZERO no código máquina nos endereços reservados.

TABLE USE

ROT1 11

ROT1 15

ROT2 18

TABLE DEFINITION

ROT3 4

RELATIVE

1 3 4

CODE

14 12 12 15 04 14 12 5

3.2 Ligador

Fazer um código (ligador.c) que receba por linha de comando o nome um a quatro arquivos (sem extensão, assume-se extensão “.obj”). O programa deve fazer a ligação dos arquivos de entrada e criar um arquivo executável (*.e) com o nome do primeiro arquivo. O arquivo de saída deve ser em formato TEXTO contendo OPCODES e operandos sem quebra de linha, nem endereço indicado, separados por espaço, SEM O INDICADOR DE SEÇÃO “CODE”, sem a indicação de absoluto e relativo, sem tabelas (ou seja, somente uma linha de números).

No Moodle tem arquivos exemplos a serem utilizados. Na correção, serão utilizados vários outros programas além dos disponibilizados. No Moodle existe também um simulador para rodar os arquivos objetos.

4 Entrega

A entrega consistirá em:

- Código-fonte completo e comentado com instruções de compilação dos programas de tradução e simulação;

- Incluir um comentário indicando os dois membros do grupo, assim como o sistema operacional onde o código foi realizado e instruções de como compilar o código.

A forma de entrega é pelo Moodle. O trabalho pode ser feito individualmente ou em dupla.

Tabela 1: Instruções e diretivas.

Instruções				
Mnemônico	Operandos	Código	Tamanho	Descrição
ADD	1	1	2	ACC \leftarrow ACC + MEM[OP]
SUB	1	2	2	ACC \leftarrow ACC - MEM[OP]
MULT	1	3	2	ACC \leftarrow ACC * MEM[OP]
DIV	1	4	2	ACC \leftarrow ACC / MEM[OP]
JMP	1	5	2	PC \leftarrow OP
JMPN	1	6	2	Se ACC < 0, PC \leftarrow OP
JMPP	1	7	2	Se ACC > 0, PC \leftarrow OP
JMPZ	1	8	2	Se ACC = 0, PC \leftarrow OP
COPY	2	9	3	MEM[OP2] \leftarrow MEM[OP1]
LOAD	1	10	2	ACC \leftarrow MEM[OP]
STORE	1	11	2	MEM[OP] \leftarrow ACC
INPUT	1	12	2	MEM[OP] \leftarrow STDIN
OUTPUT	1	13	2	STDOUT \leftarrow MEM[OP]
STOP	0	14	1	Encerrar execução.
Diretivas				
SECTION	1	-	0	Marcar início de seção de código (TEXT) ou dados (DATA).
SPACE	1	-	1	Reservar 1 ou mais endereços de memória não-inicializada para armazenamento de uma palavra.
CONST	1	-	1	Reservar memória para armazenamento de uma constante inteira de 16 <i>bits</i> em base decimal ou hexadecimal.
PUBLIC	0	-	0	Indica que o rótulo é público
EQU	1	-	0	Cria um sinônimo textual para um símbolo
IF	1	-	0	Instrue o montador a incluir a linha seguinte do código somente se o valor do operando for 1
EXTERN	0	-	0	Indica que o rótulo é um símbolo externo
BEGIN	0	-	0	Marcar início de um módulo
END	0	-	0	Marcar o fim de um módulo.