

1º Trabalho Prático
CIC 116432 – Software Básico
Prof. Bruno Macchiavello
1º Semestre de 2015

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 Macro-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 e a etapa de geração de código objeto.

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 duas seções: uma de código (TEXT) e outra de dados (DATA).

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 100 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
      COPY N2, N3
      COPY N3, N3+1
OUTPUT N3+1
      STOP
```

SECTION DATA

```
N1: SPACE
N2: CONST -0x10
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.
- A seção de dados deve vir depois da seção de códigos.
- Gerar um arquivo de saída em formato TEXTO (mais detalhes serão descritos a seguir).
- Desconsiderar tabulações e espaços desnecessários em qualquer lugar do código.
- A diretiva CONST deve aceitar declaração em hexadecimal também;
- Deve ser possível trabalhar com vetores
- Capacidade de aceitar comentários indicados pelo símbolo “;”
- O comando COPY deve utilizar uma vírgula entre os operandos (COPY A, B)
- Identificar erros durante a montagem. Montado sempre o programa inteiro e mostrando na tela aa LINHAS e TIPO DOS ERROS encontrados (léxico, sintático, semântico). O programa deve pelo menos detetar os seguintes tipos de erro:
 - declarações ausentes;
 - declarações repetidas;
 - pulo para rótulos inválidos;
 - diretivas inválidas;
 - instruções inválidas;
 - diretivas ou instruções na seção errada;

- divisão por zero;
- instruções com a quantidade de operando inválida;
- tokens inválidos;
- dois rótulos na mesma linha;
- rótulos repetidos;
- seção (TEXT ou DATA) faltante;
- seção inválida;
- tipo de argumento inválido;
- endereço de memória não reservado;
- 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 programa de montagem (chamado 'montador.c') deve receber dois 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 e um arquivo de saída (só o nome sem extensão assume-se extensão “.o”).

O arquivo objeto deve estar em formato TEXTO. Se foram usadas as diretivas PUBLIC e/ou EXTERN 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 informação devem estar separados por os indicadores TABLE e CODE. Se não foram utilizadas as diretivas PUBLIC ou EXTERN 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”).

TABLE USE

ROT1 11

ROT1 15

ROT2 18

TABLE DEFINITION

ROT3 4

CODE

14 12 12 15 04 12 5

3.2 Ligador

Fazer um código (ligador.c) que receba por linha de comando o nome de três arquivos (sem extensão, assume-se extensão “.o” para os dois primeiros e extensão “.e” para o terceiro). O programa deve fazer a ligação entre os dois primeiros módulos gerando o arquivo ligado de saída. 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”).

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

4 Avaliação

O prazo de entrega do trabalho é 27 de Maio de 2015. A entrega consistirá em:

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

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
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.