Trabalho Prático CIC 116432
Software Básico
Prof. Bruno Macchiavello
1 o Semestre de 2021

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 e a etapa de geração de código objeto. 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 seções de código e dados. Para cada instrução da máquina hipotética, a tabela vistas nos slides das aulas 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 para alocação de memória no segmento de dados. 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 (seção de dados pode vir ANTES ou DEPOIS da seção de texto):

```
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 3
N3: SPACE 2
```

N4: SPACE

O montador deve ser capaz de:

• NAO ser sensível ao caso, podendo aceitar instruções/diretivas/rótulos em maísculas e minusculas.
• Gerar um arquivo de saída em formato TEXTO (mais detalhes serão descritos a seguir). • Desconsiderar tabulação, quebras de linhas e espaços desnecessários em qualquer lugar do código.
• A diretiva CONST deve aceitar números positivos
• Deve ser possível trabalhar com vetores (SPACE com operando, e usar operaçõs do tipo: LABEL+Numero, sem espaço entre o '+')
• Capacidade de aceitar comentários indicados pelo símbolo ";" em qualquer lugar do código
• O comando COPY deve utilizar uma vírgula e um SEM espaço entre os operandos (COPY A,B)
• Utilizar o algoritmo de passagem ÚNICA
• Poder criar um rótulo, dar quebra de linha e continuar a linha depois
• 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 LEXICO, SINTÁTICO OU SEMÂNTICO):

– declarações de rótulos ausentes;
– declarações de rótulos repetidos;
– diretivas inválidas;
– instruções inválidas;
– instruções com a quantidade de operando errada;
– tokens inválidos;
– dois rótulos na mesma linha;

Todos os arquivos de saída devem estar em formato TEXTO. No caso do arquivo objeto, o arquivo de saída deve ser somente os OPCODES e operandos SEM quebra de linha, nem endereço indicado, mas separados por espaço (numa única linha). No caso da diretiva SPACE deve ser colocado o valor 00 (zero,zero) no arquivo objeto (não colocar XX).

O programa deve chamar "montador" e receber o arquivo de entrada (com extensão) por argumento na linha de comando (ex: ./montador myprogram.asm). O programa deve dar como saída o arquivo com o mesmo nome MUDANDO a extensão para .OBJ. É obrigatório que o funcionamento do programa seja tal como descrito, caso contrário o trabalho recebe automaticamente a nota ZERO.

3.1 Simulador

A segunda parte é fazer um simulador. O simulador deve chamar "simulador" e deve receber o arquivo de entrada (com extensão) por argumento na linha de comando (ex: ./simulador myprogram.obj). A entrada do simulador é a saída do montador. O próprio simulador do aluno

será utilizado na correção do trabalho para verificar se os programas foram traduzidos corretamente. Porém, caso não seja entregue esta parte, será utilizado um simulador padrão que sempre espera que o código referente a parte de dados esteja depois da parte de texto no arquivo objeto.

O simulador deve executar o programa OBJETO. Mostrando na tela depois de cada instrução (não diretiva). A linha que está sendo executada, depois o valor do acumulador, o valor do contador de programa, e qualquer informação que seja mostrada pela instrução OUTPUT. Após OUTPUT deve esperar o usuário digitar ENTER para continuar a execução.