Assembler

Lexer (lexer.c)

O Lexer processa o arquivo de entrada caractere por caractere. Ele identifica elementos como:
Palavras-chave (.DATA, .CODE)
Instruções (LDA, STA, ADD, etc.)
Números (valores decimais e hexadecimais)
Espaços em branco e comentários (que são ignorados)

Quando o Lexer encontra um componente significativo, ele cria um token que contém:
O tipo do componente (instrução, número, diretiva)
O valor do componente (como uma string)
O número da linha onde foi encontrado (para mensagens de erro)

O Lexer tem funções para identificar números hexadecimais, comentários e delimitadores.

Parser (parser.c)

O Parser recebe os tokens do Lexer e os interpreta dentro do contexto do programa. O Parser mantém:
Um array de memória que representa a memória do computador Neander
Um endereço atual para a seção de código
Uma flag indicando se está na seção .DATA ou .CODE
Um contador para o maior endereço utilizado

Quando o Parser identifica uma instrução, ele busca em uma tabela o opcode correspondente ao mnemônico. Essa tabela é definida no início do programa com todos os mnemônicos e seus respectivos opcodes.

Instruções

NOP: Não faz nada, apenas segue para a próxima instrução.
STA: Recebe um endereço de memória e armazena nele o valor atual do acumulador.
LDA: Recebe um endereço de memória e carrega seu valor para o acumulador.
ADD: Recebe um endereço de memória e adiciona seu valor ao acumulador.
OR: Recebe um endereço de memória e faz um OR bit a bit entre seu valor e o acumulador.
AND: Recebe um endereço de memória e faz um AND bit a bit entre seu valor e o acumulador.
NOT: Inverte todos os bits no acumulador (muda 0s para 1s e 1s para 0s).
JMP: Recebe um endereço de memória e pula para aquela localização no programa.
JN: Recebe um endereço de memória e pula para lá apenas se o acumulador for negativo.
JZ: Recebe um endereço de memória e pula para lá apenas se o acumulador for zero.
HLT: Para a execução do programa.

Geração de Saída

Quando o parser termina, ele:

1. Adiciona um cabeçalho especial ao arquivo de saída
2. Escreve todos os valores de memória em um arquivo binário
3. Cria um arquivo de texto separado com um dump legível dos conteúdos da memória

O arquivo de saída binário contém o código de máquina que pode ser executado pela máquina virtual Neander.