DOCUMENTATION

ASSEMBLER

AUTHORS:

Amandeep Kaur (2018014)

Deepi Garg (2018389)

TABLE OF CONTENTS

1) About the Assembler

A basic implementation of a two-pass assembler, used to convert assembly language code to object code.

1.1) Key features

- The assembler supports the use of labels, handles forward and backward referencing of labels.
- The assembler handles variables and their memory allocation.
- Handles MACRO definitions and calls in the program

1.2) Machine requirements

- 12-bit accumulator architecture (Word length = 12 bits).
- Has at least 2^8 bit memory

1.3) Working

First Pass:

The assembler in the first pass initialises a location counter and traverses the Assembly code line by line, incrementing for each instruction.
It checks if the statement is a valid memory referencing instruction or a MACRO definition. It reports an error in case it is none of these.
It creates a Label table to store the addresses of all Labels used in the program.
It creates a Symbol table to store addresses of variables used.
A Macro table is created to save the macro definitions.
The Macro calls are expanded during the first pass of the assembler, where the virtual parameters are replaced by the actual parameters.
Multiple definitions for the same label are caught during the first pass and reported as an error.
References to undefined labels are caught after the first pass.
On encountering an error, the first pass continues and reaches the end, thus reporting all errors in the program in one go. Following this, the execution is stopped and the user needs to correct the code and run the assembler again.

Second Pass:

The assembler reads the instructions one by one and converts opcodes and addresses to binary. The label calls are replaced by the starting address of label definitions and variables are replaced by their storage addresses, both provided by the location counter during the first pass.

2) Instruction Set

The instruction set is as follows:

2.1) *Opcode Table*

Assembly Opcode	Machine Opcode	Function	Number of Operands
CLA	'0000'	Clear accumulator	0
LAC	'0001'	Load into accumulator from address	1
SAC	'0010'	Store accumulator contents into address	1
ADD	'0011'	Add address contents to accumulator contents	1
SUB	'0100’	Subtract address contents from accumulator contents	1
BRZ	'0101'	Branch to address if accumulator contains zero	1
BRN	'0110'	Branch to address if accumulator contains negative value	1
BRP	'0111'	Branch to address if accumulator contains positive value	1
INP	'1000'	Read from terminal and put in address	1

DSP	'1001'	Display value in address on terminal	1
MUL	'1010'	Multiply accumulator and address contents	1
DIV	'1011'	Divide accumulator contents by address content. Quotient in R1 and remainder in R2	1
STP	1100	Stop Execution	0

2.2) Assembler Directives

START statement
Optional
First line of the Program
Specifies a specific memory address to load the program at (StartAddress)
Generates error if used in any other line other than line 1
Syntax: START StartAddress

END statement
Optional
Indicates the End of the program
Lines After this statement are not translated.
Syntax: END

3) General instruction format

3.1) Syntax

<LabelName:> Opcode <Operand> <Comment>

3.2) Label name

Optional

Should only start with a letter

Cannot be an opcode

Cannot contain spaces

Label name should be followed by a colon “:”

3.3) Opcode

Should be in the same case (Uppercase) as mentioned in the table

Should be followed by the correct number of operands as mentioned in the table

3.4) Operands

3.4.1) Variable Names

Variable Names:-Should begin with a letter only

Cannot consist of spaces

Cannot be Opcodes

Cannot be reserved words (namely R1, R2)

3.4.2) Memory addresses

Value can range from 0 to 255

3.5) Comments

Optional

Only single line comments supported

Comment should begin with “//”

4) MACRO definition

Syntax: MacroName MACRO <Parameter1 Parameter2 ..>
... <MACRO body> ...
MEND

Parameters (Space separated) are variables and follow the same convention as other variables
The MACRO body should not contain Labels or references to labels
MACRO Call-
Syntax: MacroName <Parameter1 Parameter2..>

5) Providing the input file

A TEXT FILE consisting of Assembly language program saved in the same directory as the assembler program with the name given below.
File Name: "input.txt"

6) Output file format

A Text File containing the Machine Language program corresponding to the Input Assembly Program is created in the same directory as the Assembler program.
File Name: "output.txt"

The first 4 bits of each instruction constitute the opcode and the rest 8 bits define the address of the operand.

7) Errors and Warnings

Errors:
Errors abort the execution of the program after the first pass is complete. The tables are displayed but the output file is not created. The user is required to rectify the error in order to translate their program successfully.

Warnings:
Warnings are displayed at points where the assembler detects a possible error in the user's logic like a missing CLA (clear accumulator) instruction in the input code. It should not be seen as an error as output code file is successfully created.