ACKNOWLEDGEMENT

Any achievement, be it scholastic or otherwise does not depend solely on the individual efforts but on the guidance, encouragement and cooperation of intellectuals, elders and friends. A number of personalities, in their own capacities have helped me in carrying out this project work. I would like to take this opportunity to thank them all.

I would like to thank my System Software lab teachers, **Prof. Vanishree K** and **Prof. Nagaraj G Cholli** from the Department of ISE, R.V.C.E for their invaluable advice and the guidance extended towards the completion of this project.

I deeply express my sincere gratitude to **Prof. Srinivas B K, Assistant Professor,** Department of ISE, R.V.C.E, Bengaluru, for his able guidance, regular source of encouragement and assistance throughout this project

I would like to thank **Dr. N. K. Cauvery**, Head of Department, Information Science & Engineering, R.V.C.E, Bengaluru, for her valuable suggestions and expert advice.

I would also like to thank **Dr. K. N. Subramanya**, Principal, R.V.C.E, Bengaluru, for his moral support towards completing my project work.

I thank my Parents, and all the Faculty members of Department of Information Science & Engineering for their constant support and encouragement.

Last, but not the least, I would like to thank my peers and friends who provided me with valuable suggestions to improve my project.

ABSTRACT

The computer cannot understand our C program or any high level language. The HLL can be converted to an assembly language program, but that too is unreadable for the computer – which can read, to say naively, nothing but a stream of bits.

Assembler is a program that translates programs from assembly language to machine language. The assembler takes as input a stream of assembly commands, and generates as output a stream of equivalent binary instructions. The resulting code can be loaded as-is into the computer's memory, and then executed by the hardware.

The main purpose of this project is to design and implement a two pass assembler for 8086 microprocessor. This project is a small step towards writing such an assembler in Lex and Yacc and C language which is supposed to perform fundamental functions like translating mnemonic operation codes to their machine language equivalents and assigning machine addresses to symbolic labels and creation of an object file which can be linked. We make use of two data structures: - Operation Code table (optab), Symbol Table (symtab) both taking advantage of hash table properties.

The output of first pass is a symbol table which is constructed by assembler using location counter. It stores addresses of labels, so that it can be used during pass-2. Second pass uses the offset of symbols, provided by symbol table to generate operand address. It performs processing of assembler directives. It provides the assembly listing and object program. Thus second pass, solves the problem of forward references.

Table of Contents

List of Figures