**DAYANANDA SAGAR COLLEGE OF ENGINEERING COMPUTER SCIENCE & ENGINEERING**

Minor Project- Report

Apr 2023-Jul 2023

Course Faculty:

Course Name & Code: SYSTEM SOFTWARE LAB WITH MINI-PROJECT (19CS6DCSSW)

Semester: VI Date: 14/06/2023

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| TITLE OF THE PROJECT | **TWO PASS ASSEMBLER** | | | |
|  |  | | | |
| STUDENT NAME | PRUTHA VASISHT | PUNITH KUMAR  P R | SUHAS REDDY | RACHANA K |
| USN | 1DS20CS157 | 1DS20CS158 | 1DS20CS159 | 1DS20CS160 |
| INDIVIDUAL CONTRIBUTION | Pass 2 Code in the Assembler | Pass 1 Code in the Assembler | Pass 1 Code in the Assembler | Pass 2 Code in the Assembler |
| GUIDE | Prof. Aparna | | | |
|  |  | | | |
| PROJECT ABSTRACT: | A two-pass assembler works by performing the following steps in two passes:   1. In Pass 1, addresses are assigned to all the statements in the program. Then the values assigned to the labels and symbols are saved for use in Pass 2 using SYMTAB and OPTAB. It also processes pseudo-operations. An intermediate file is generated. 2. In Pass 2, instructions are assembled using values in SYMTAB and OPTAB. Data values defined by BYTE and WORD are generated and the assembler directives which were not processed in Pass 1 are processed. The object program and assembly listing are written.   We will be focusing on generating machine codes from a set of assembly language codes. | | | |
| PLATFORM USED (H/W & S/W TOOLS TO BE USED) | Development Environment: Visual Studio Code, Windows OS Programming Language: C++ | | | |
|  |  | | | |

**DAYANANDA SAGAR COLLEGE OF ENGINEERING COMPUTER SCIENCE & ENGINEERING**

|  |  |
| --- | --- |
| INTRODUCTION | An assembler is a program for converting instructions written in low-level assembly code into relocatable machine code and generating along information for the loader. If the assembler does all this work in one pass, then it is called a single pass assembler and otherwise if it does it in multiple passes then it is called a multiple pass assembler. When it does it in two passes, it is called a 2 Pass Assembler. The work done in both passes can be described as follows -  Pass - 1:   * Define symbols and literals and remember them in symbol table (SYMTAB) and literal table (OPTAB) respectively. * Keep track of the location counter. * Process pseudo-operations like macros and directives.   Pass - 2:   * Generate object code by converting symbolic opcode into respective numeric opcode. * Generate data for literals and look for values of symbols in OPTAB and   SYMTAB. |
|  |  |
| DESIGN | The architectural design of a two-pass assembler can be shown as follows - |



|  |  |
| --- | --- |
|  |  |
| PROJECT SOURCE CODE LINK (GITHUB/ GOOGLE  DRIVE) |  |
|  |  |
| CONCLUSION /FUTURE ENHANCEMENT | The program implements the working of a 2 Pass Assembler for SIC machine. Further, the program could be written for SIC/XE machine where the instructions are written in 4 different formats. |
|  |  |



|  |  |
| --- | --- |
| UI SCREENSHOTS | INPUT:   1. The below text file (opcodes.txt) contains opcodes for all mnemonics.      1. This text file (startaddress.txt) contains the starting address of the program. |



|  |  |
| --- | --- |
|  | 3. This text file (input.txt) contains the source code that needs to be converted into object code.    **OUTPUT:**  The final output program and object program is displayed. |



|  |  |
| --- | --- |
|  | 1. The following text file (symtab.txt) will contain the symbols and their addresses from the source program.      1. This text file (optab.txt) contains the opcodes and mnemonics from the source program. |

