NATIONAL INSTITUTE OF TECHNOLOGY GOA

SYSTEM PROGRAMMING ASSIGNMENT

SIC/XE ASSEMBLER THAT SUPPORTS PROGRAM RELOCATION AND SYMBOL DEFINING

BY:
VISHAL KUMAR
17CSE1027
SANGRAM PATIL
17CSE1030

email id:
vishalvia1228@gmail.com
sangram.mahadeopatil15@gmail.com

April 19, 2019

Introduction

BASIC SIC/XE

The SIC/XE loader is a 128 character SIC/XE program designed to fit in the 128 character boot file space. It is composed of fully relocatable code. The loader is designed to input a load module of hex characters from a text file, converting them to true numeric form, and storing them beginning from a specified load point. At the end of the load, control is passed to a specified starting address.

Load module file format

A text file of hex characters as follows:

1. 1st 6 characters = load point

2. Next 6 characters = address to receive control when load is finished ("000000" to re-run the loader and input the next module - see below)

3. Rest of file = code to be stored in memory beginning from the load point

4. Module delimiter = !

Restrictions

1.The loader input is via device F1 (default file name DEVF1)

2.The loader is stored in the file "loader". The SIC simulator will boot from this file under the START command if the default file name for device 00 (DEV00) is replaced by the file name "loader".

3.The load point must be greater than X'00007F'

4.The loader does no error checking for validity of input.

Each load module normally consists only of actual code. In particular, the storage directives RESW and RESB have no corresponding code and so typically provide a natural point at which to terminate a load module (alternately, garbage could be inserted into the code to reserve the needed amount of storage). If more than one load module is employed for a program, then the loader must be restarted for each load module. This is done by putting the loader start address (000000) as the start address component for each load module except the last one. The last load module has the start address for the full program in its start address component.

Register Numbers :
A 0
X 1
L 2
B 3
S 4
T 5
F 6

The operand value for 3 and 4 byte instructions is at the target address determined according to the addressing mode:

direct: - operand value is at the target address determined directly using the displacement/address information
indirect - operand value is at the target address stored at the address given by the direct address computation immediate
- operand value is the value of the direct address computation (target address is the instruction register in the CPU)

HOW TO COMPILE

1.Enter command g++ name.cpp in the terminal to compile.

2.Enter ./a.out to run the program.

FILES GENERATED

0.1 AFTER PASS ONE:

-INTERMEDIATE_FILE: contains address of each instruction generated after pass one.

-SYMTAB: contains all the symbols defined in the instructions.

-SYMDEF: contains all the symbolic values used .(as a MACRO)

0.2 AFTER PASS TWO:

-OBJECT_PROGRAM: contains a file which reflects all the object codes generated in a sequential order.

-ASSEMBLER_LISTING: contains the intermediate file with the object code generated for each instruction.