Roll No : 33231
Name : Prajakta Keer

SP classmate
Date
Page

ASSIGNMENT 1

Title : Pass I of two pass assembler

Problem Statement : Write a program to implement Pass-I of two pass assembler for symbols and literal processing considering following cases :

1) Forward referencing
2) DC and DS statements
3) START, EQU, LTORG, END
4) Error Handling

Objective : To study basic translation process of assembly language to machine learning using hypothetical instruction set.

Theory : A language translator bridges an execution gap to machine language of company system. An assembler is a language translator whose source language is assembly language.

Design of Pass I of two pass assembler

Tasks performed are as follows :

Pass-I

- Seperate the symbol, mnemonic, opcode and operand fields
- Determine the storage required for every assembly language statement and update the location counter
- Build the symbol table and literal table.
- Construct the intermediate code for every assembly language statement.

Data Structure Required for Pass I

1) Soure file containing assembly program
2) OPTAB : It has following fields

classmate
Date
Page

- Mnemonic : such as ADD, END, DC
- Type : IS for imperative, DL of declarative, AD for assembler directive
- Opcode : Operation code indicating the operation to be performed.
- Length : Length of instruction required for location counter processing
- No of operands : max no. of operands the instruction needs

Algorithm :

1) Open the source file in read mode
2) If end of file of source file go to step 8
3) Read the next line of source program
4) Seperate the lines into words. These words would be stored in an array of strings.
5) Search for first word in mnemonic opcode table, if not present it is a label, add this as a symbol in symbol table with current LC. And then search for second word in mnemonic opcode table
6) If instruction is found
   Case 1 : IS , Case 2 : DL , Case 3 : AD
7) Goto step 2
8) Close source file and open intermediate code file
9) If end of file go to step 13
10) Read next line from intermediate code file
11) Write opcode, register code and address of memory onto target file. This is to be done only for imperative statement
12) Go to step 9
13) Close all files
14) Display symbol table, literal table and target file.

33231

classmate
Date
Page

Imperative statement Case :

1) If opcode >=1 && opcode <= 8
set type as IS, get opcode, get register code and make entry into symbol / literal table as the case may be. In case of symbol, used as operand, LC field isn't known so LC could be -1. Perform LC processing LC++.

2) If opcode is 00
set all fields of intermediate calls as 00LC++.

3) else register operand not required
Same case 1, only register code is not required so set it to zero. Here again update the symbol table LC++.

Conclusion : We learnt Pass-I assembler and implemented it and also handled various errors.