Page No.
Date: / /

Assignment A1

Title: Pass I of a 2 Pass Assembler

Date of Completion: 09/01/2020

Problem Statement:
Design suitable data structures & implement pass I of a 2-Pass Assembler for a pseudo-machine in Java using object oriented feature. Implementation should consist of a few instructions from each category & few assembler directives.

Objectives:
Implement Pass I of a 2 Pass Assembler
Use Object-Oriented features for implementing pass I
Implement instructions from each category.
Implement assembler directives as well

Outcome:
Student should be able to:
Understand & implement pass I of a 2 pass assembler
Implement pass I for instructions from each category

Software & Hardware Requirements:
Fedora 64 bit OS
Eclipse IDE for Java
i5 processor
4GB RAM
500 GB HDD

Implement algorithmic techniques by virtue ofadvance programming skill set and Free and Open Source Software (FOSS) tools.

Page No.
Date:

Theory :

Assembler :

An assembler is a program that accepts an assembly language as the input & produces its machine code.

An assembler has to perform the following major functions

i. Replace Symbolic Address by Numeric Address
ii. Replace Symbolic Operating Codes by Machine Operation Codes
iii. Reserve Storage for instruction & data
iv. Handle literals

Two Pass Assembler :

~~Pass I~~ Many Assemblers are designed in 2 pass

Pass I :-

i. Separate the labels, mnemonics, opcode & operand fie
ii. Determine the storage requirement for every assembly language statement & update the location counter
iii. Build the symbol table

Pass II :-

Generate the machine code.

2-Pass Assembler can handle literals

Page No.
Date: / /

Pass I uses the following data structures :

1. Machine Opcode Table (MOT)
2. Symbol Table (ST)
3. Literal Table (LT)
4. Pool Table (PT)
5. Table for storing intermediate code in internal format (I

1. Machine Opcode Table

Type of Mnemonic Symbol	Value of Class field
Imperative Statement (IS)	1
Declarative Statement (DL)	2
Assembler Directive (AD)	3
CPU Register (RG)	4
Condition Code (CC)	5

2. Symbol Table :
A symbol table entry contains :-
   - label
   - address
   - size

3. Literal Table :
A literal table entry contains :-
   - value
   - address

4. Pool Table :
A Pool Table contains the literal number of the starting literal of each literal pool.

Page No.
Date:

5 Intermediate Code :

IC Table entry contains :-

- Type I : Gives the type of mnemonic symbol
- Code I : Gives the index of mnemonic symbol

Algorithm :

1. LC = 0 ;
2. iPT = 0 ;
3. PT [iPT] = 0 ;
4. iLT = 0 ;
5. Open the source file ;
6. While (! end of the source file) {
   a) Read the next line
      Store the line in character array nextline
      after substituting special character with blan
   b) if (nextline is an END statement) then
      break;
   c) if nextline contains a label then
      store label in Symbol Table
   d) Separate opcode & operands
   e) if nextline is declaration statement then
      i. Type I = type of declaration statement
      ii. Code I = code (index in MOT)
      iii. size = size of memory area req by D
      iv. Generate intermediate code & store in
      v. LC = LC + size
   f) if nextline is imperative statement then
      i. Type I = type of imperative statement
      ii. Code I = code (index in MOT)
      iii. size = size of instruction
      iv. If operand is literal then
          literal is stored in literal table

7.

v. Generate intermediate code & store in IC table
vi. LC = LC + size

g) If nextline is LTORG statement then
   i. Literals of current pool are assigned addresses
   ii. iPT = iPT + 1
   iii. PT[iPT] = iLT
   iv. LC modified accordingly.

h) If nextline is START or ORIGIN then
   i. LC = address specified in statement
   iv. Generate intermediate code & store the same in IC table

i) If nextline is EQU statement then
   i. Address of variable is corrected in symbol table
      New address is extracted from statement
   ii. Generate intermediate code & store the same in IC table.

}

7. Processing of END statement
   i. Perform G(g)
   ii). Go to pass II.

Implement algorithmic techniques by virtue of advance programming skill set and Free and Open Source Software (FOSS) tools.

Page:
Date: 1

Test Cases

	Description	Input	Output	Conclusion
1	General IS	MOVER AREG,A	(IS,04) 1 (S,01)	Success
2	START statement	START 200	LC set to 200	Success
3	LTORG statement	LTORG	Address assigned to literals	Success
4	EQU statement	BACK: EQU LOOP	Back is given address of ~~loop~~ LOOP	Success
5	END statement	END	Address assigned to literals & Program moves to Pass II	Success

Conclusion:
Succesful Implementation of Pass I of two pass assembler using object oriented features & proper data structures.