PAGE No.
DATE / /

Assignment A1

Title: Pass I of a two pass assembler

Problem statement: Design suitable data structures and implement pass-I of a two-pass assembler for psuedo machine in Java using objet oriented feature. Implementation should consists of a few instructions from each category and few assembler diretives.

objetives:
- Understand the internals of language translators
- Handle tools like LEX and YACC.
- Understand the operating system internals and functionalities with implementation point of view.

s/w Packages and h/w Appratus used	:	64-bit open source Linux IntelJ IDE, JAVA I3 and I5 Machines.

Theory:
Assembler is a program which converts assembly language instrutions into machine language form. A two pass assembler takes two scans of source code to produce the machine code from assembly language program.
Assembly process consists of following activities
- Convert mnemonics to their machine language opcode equivalents
- Convert symbolic operands to their machine address
- Translate data constants into internal machine representations.

- output the object program and provide other information required for linker and loader.

Pass I Tasks:

- Assign addresses to all the statements in the program
- Save the Values assigned to all labels for use in pass II
- Perform processing of assembler directives

Description using set theory:

Let 'S' be set which represents a System

S = {I, O, T, D, Succ, fail}

Where, I = Input
O = Output
T = Type (variant I or II)
D = Data structures

I = {SF, MF}

where,
SF = Source Code file
MF = Manemonic Table

O = {St, Lt, Ic}

Where, St = Symbol
Lt = Literal
Ic = Intermediate Code file

St = {N, A}

Where, N = Name of Symbol
A = Address of Symbol

Lt = {N, A}

Where, N = Name of Literal
A = Address of Literal.

T = variant II

D = {Ar, Fl, Sr}

Where,

Ar = Array
fl = file
Sr = structures.

· Test Cases :

Test case	Expected output	Actual Result
Input all valid Mnemonics	Replace the mnemonics with Correct opcodes	Success
Input the instruction and operands in valid format	Generate Valid intermediate Code Format	Success

PAGE No.
DATE / /

• Conculusion :-

We were able to Parse and tokenize the assembly Source Code, Perform the LC processings, Generate the Intermediate file. Successfully.