Date

Assignment - A1

Title :- Pass I of two pass assembler.

Problemstatement :-

Design suitable data structures & implement pass-1 of two pass assembler for pseudo-machine in Java using object oriented feature. Implementation should consist of a few instructions from each category & few assembler directive.

Objective :-

i) Understand the internal of language ~~[illegible]~~ translators.
ii) Handle tools like LEX & YACC.
iii) Understand the operating system internals & functionalities with implementation point of view.
~~iv)~~

Outcome :-

student should be able to
i) Understand internal of language translator
ii) Handle tools like LEX & YACC.
iii) Understand operating system internals & functionalities with implementation point of view.

Date

S/W requirements :- 64 bit OS , Eclipse IDE, Java.

Theory :-

Assembler :-

Assembler is program which converts assembly language instructions into machine language form. A two pass assembler takes two scans of source code to produce the machine code from assembly language program.

It consists of :

i) Convert mnemonics to their machine language opcode equivalents.

ii) Convert symbolic (ie. variables, jump lables) operands to their machine addresses.

iii) Translate data constants into internal machine representations.

iv) Output the object program & provide other information required for linker & loader.

Pass I tasks :

i) Assign addresses to all statements in program.

ii) save addresses assigned to all lables (including label & variable names) for use in pass II

iii) perform processing of assembler directives

Date

Description using set theory :-

let 's' be set which represents system

$S = \{ I, O, T, D, succ, fail \}$

I = Input , O = Output , T = Type (variant I or II)

D = Data Structure

$I = \{sf, mf\}$ sf = source code file, mf = Mnemonic Table.

$O = \{ St, Lt, I_c \}$ St = symbol, Lt = Literal, I_c = Intermediate code file.

$St = \{N, A\}$ N = Name of Symbol , A = Address of Symbol

$Lt = \{N, A\}$ N = Name of literal , A = Address of literal

T = Variant II

$D = \{ Ar, Fl, Sr \}$ Ar = Array, Fl = file , Sr = Structure

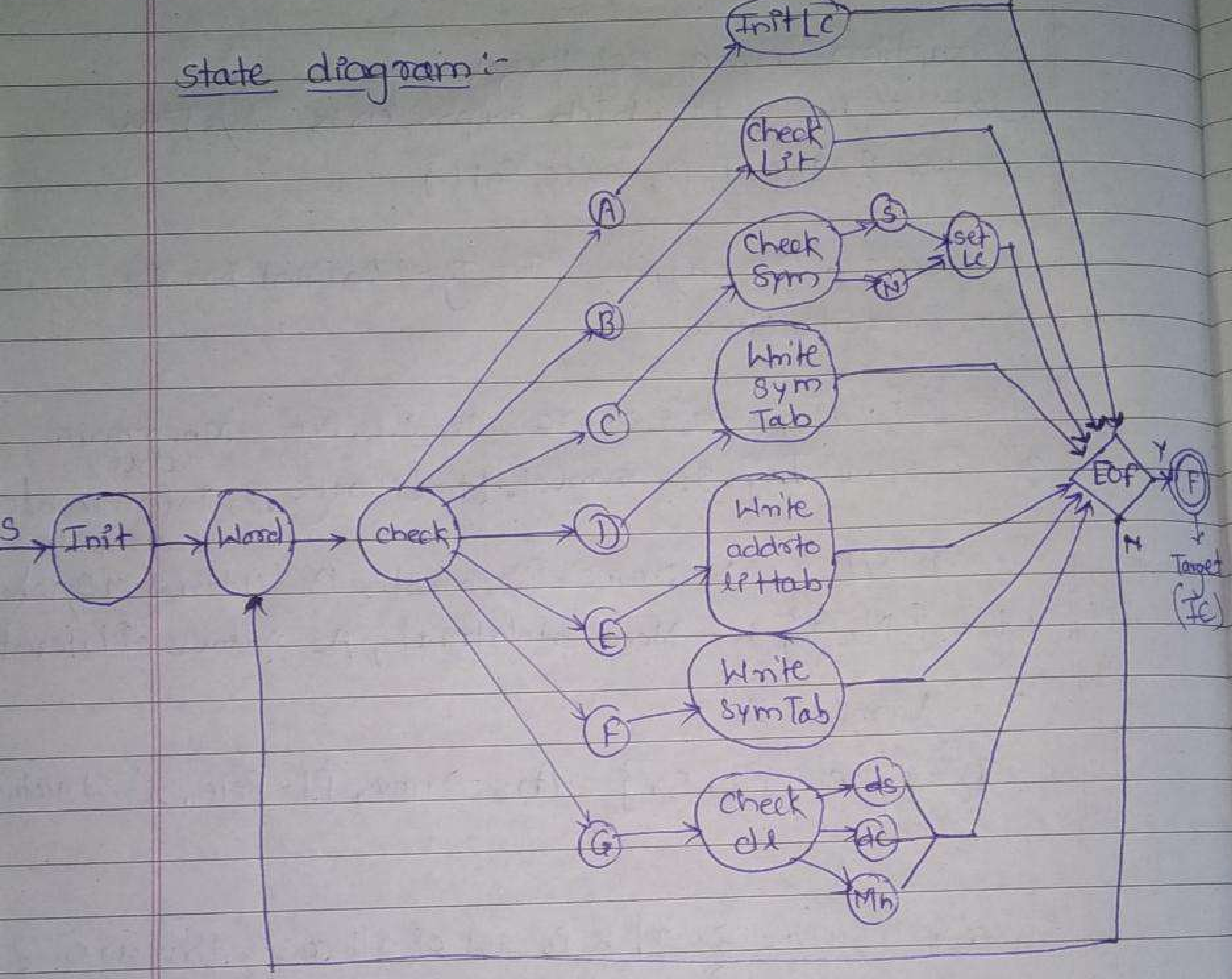
Success succ = { x | x is set of all cases that are handled in program }

succ = { undefined symbol (also label) , Duplicate symbol , Undefined symbol in assembler directives. }

Failures fail = { x | x is set of all cases that are not handled in program }

fail = { multiple statements in line }

state diagram :-



Pass 1 of 2 pass Assembler

Algorithm :-

1] Create MOT
2] Read .asm file & tockenize it.
3] Create Symbol & literals tables
4] Generate intermediate code file.

Date

Test Cases :-

	Input	Expected Output	Result
1]	Input all valid mnemonics	Replace mnemonics with correct opcodes	success
2]	Input the Instructions & operands in valid format	Generate valid Intermediate Code format	success.

Conclusion :-

We have learnt & successfully implemented pass I assembler.