Gokhale Education Society's
R.H.Sapat College of Engineering, Management Studies & Research ,Nashik 422005

Department of Computer Engineering

Subject: System Programming & Operating System (310251)
Class: T.E Computer
Division: A
Semester: VI
Faculty: Vrushali Nikam

Group Members:

Roll No.	Name	PRN No.
01	JAGTAP MADHURA NARENDRA	71918080F
02	MULAY ADITI DINESH	71918146B
03	AGRAWAL DEEPAK VIJAY	71917955G
04	AHER ADITI SANJAY	71917956E
05	AHER SHRITESH SANTOSH	71837221C

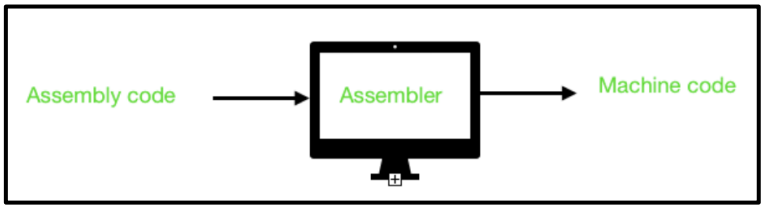
Problem Statement 1:

Generate intermediate code as well as final machine code for any Assembly program.

Theory:

Assembly Language:- An assembly language is a type of low-level programming language that is intended to communicate directly with a computer's hardware. Unlike machine language, which consists of binary and hexadecimal characters, assembly languages are designed to be readable by humans.

Assembler:- Assembler is a program for converting instructions written in low-level assembly code into relocatable machine code and generating along information for the loader.



It generates instructions by evaluating the mnemonics (symbols) in operation field and find the value of symbol and literals to produce machine code. Now, if assembler do all this work in one scan then it is called single pass assembler, otherwise if it does in multiple scans then called multiple pass assembler. Here assembler divide these tasks in two passes:

- **Pass-1:**
    1. Define symbols and literals and remember them in symbol table and literal table respectively.
    2. Keep track of location counter
    3. Process pseudo-operations

- **Pass-2:**
    1. Generate object code by converting symbolic opcode into respective numeric op-code
    2. Generate data for literals and look for values of symbols

Intermediate Code:- Intermediate code is used to translate the source code into the machine code. Intermediate code lies between the high-level language and the machine language.

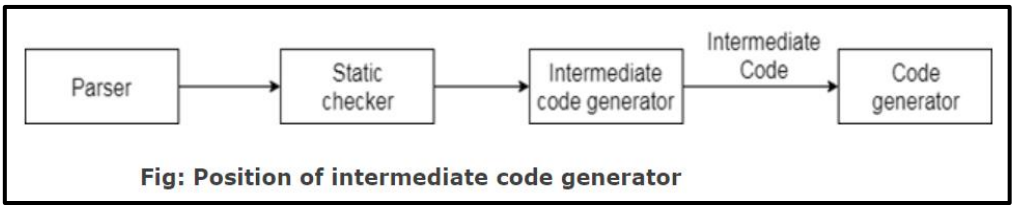


Fig: Position of intermediate code generator

Symbol Table :-Symbol table is used to store the information about the occurrence of various entities such as objects, classes, variable name, interface, function name etc. it is used by both the analysis and synthesis phases.

The symbol table used for following purposes:

- It is used to store the name of all entities in a structured form at one place.
- It is used to verify if a variable has been declared.
- It is used to determine the scope of a name.
- It is used to implement type checking by verifying assignments and expressions in the source code are semantically correct.

Literal Table:- Literal table is used for keeping track of literals that are encountered in the programs. We directly specify the value, Literal is used to give a location for the value. Literals are always encountered in the operand field of an instruction.

Pool Table:- In computer science, and specifically in compiler and assembler design, a literal pool is a lookup table used to hold literals during assembly and execution.

Solution:-

Problem statement: 1

For the following assembly language code show the contents of symbol table, literal table and pool table at the end of pass 1. Show the intermediate code generated for the program. Show the machine code generated for the program

	START	100
	MOVER	AREG, =5
	ADD	CREG, =1
A	DS	3
L1	MOVER	AREG, B
	ADD	AREG, C
	MOVEM	AREG, D
	LTORG	
D	EQU	A+1
L2	PRINT	D
	ORIGIN	A-1
	SUB	AREG, =1
	MULT	CREG, B
C	DC	'5'
	ORIGIN	L2+1
	STOP	
B	DC	19
	END	

→ Solution:

Step 1: Assembly code to Intermediate Code

		Assembly code		LC	Intermediate Code
1.		START	100		(AD,01) (C,100)
2.		MOVER	AREG,=5	100	(IS,04) (RG,01) (L,0)
3.		ADD	CREG,=1	101	(IS,01) (RG,03) (L,1)
4.	A	DS	3	102	(DL,01) (C,3)
5.	L1	MOVER	AREG,B	105	(IS,04) (RG,01) (S,2)
6.		ADD	AREG,C	106	(IS,01) (RG,01) (S,3)
7.		MOVEM	AREG,D	107	(IS,05) (RG,01) (S,4)
8.		LTORG		108	(DL,02) (C,5)
				109	(DL,02) (C,1)
9.	D	EQU	A+1		(AD,04) (C,103)
10.	L2	PRINT	D	110	(IS,10) (S,4)
11.		ORIGIN	A-1		(AD,03) (C,101)
12.		SUB	AREG,=1	101	(IS,02) (RG,01) (L,2)
13.		MULT	CREG,B	102	(IS,03) (RG,03) (S,2)
14.	C	DC	'5'	103	(DL,02) (C,5)
15.		ORIGIN	L2+1		(AD,03) (C,111)
16.		STOP		111	(IS,00)
17.	B	DC	19	112	(DL,02) (C,19)
18.		END			(AD,02)
19.		LTORG		113	(DL,02) (C,1)

	Symbol Table	
	Symbol	Address
0	A	102
1	L1	105
2	B	112
3	C	103
4	D	103

	Literal Table	
	literal	Address
0	=5	108
1	=1	109
2	=1	113

	Pool table
0	0
1	2

Step 2: Intermediate Code to Machine code.

	Intermediate Code	LC	Machine Code		
1.	(AD,01) (C,100)				
2.	(IS,04) (RG,01) (L,0)	100	04	01	108
3.	(IS,01) (RG,03) (L,1)	101	01	03	109
4.	(DL,01) (C,3)	102			
5.	(IS,04) (RG,01) (S,2)	105	04	01	112
6.	(IS,01) (RG,01) (S,3)	106	01	01	103
7.	(IS,05) (RG,01) (S,4)	107	05	01	103
8.	(DL,02) (C,5)	108	00	00	005
	(DL,02) (C,1)	109	00	00	001
9.	(AD,04) (C,103)				
10.	(IS,10) (S,4)	110	10	00	103
11.	(AD,03) (C,101)				
12.	(IS,02) (RG,01) (L,2)	101	02	01	113
13.	(IS,03) (RG,03) (S,2)	102	03	03	112
14.	(DL,02) (C,5)	103	00	00	005
15.	(AD,03) (C,111)				
16.	(IS,00)	111	00	00	000
17.	(DL,02) (C,19)	112	00	00	019
18.	(AD,02)				
19.	(DL,02) (C,1)	113	00	00	001

Conclusion:- Hence, we have successfully generated intermediate code as well as final machine code for Assembly program.