Page No. 27
Date 08 02 22

TUTORIAL 5

U19CS012

1.) What are the uses of OPTAB (Mnemonic Operation Table) and SYMTAB (symbol Table) during assembling process? Specify the use of each during pass 1 and pass 2 of a two pass assembler.

1.) OPTAB [static table] ↗ content never change

(a) USES - Used to look up mnemonic operation codes and translate them to machine language equivalents.

(b) During PASS 1, OPTAB is used to look up and validate operation coded in source program and to find the instruction length for incrementing LOCCTR.

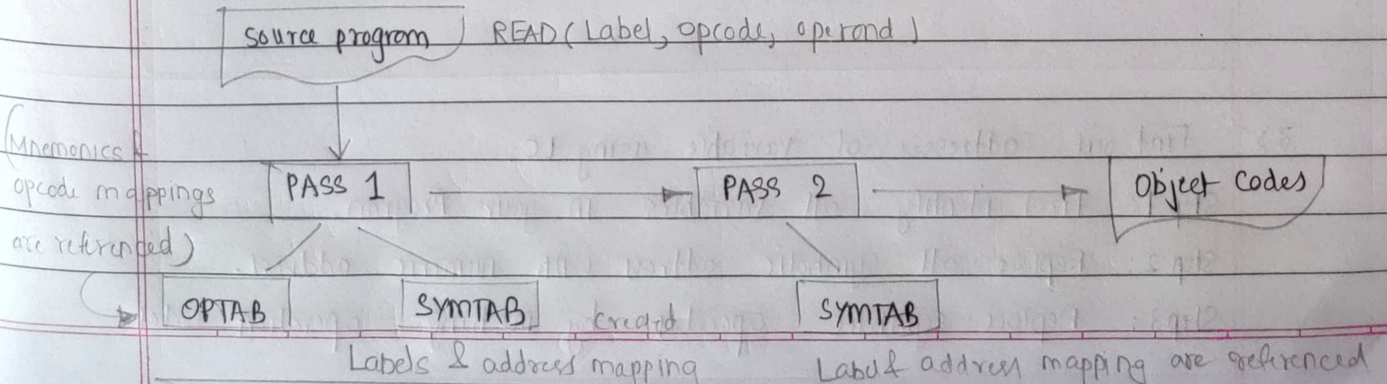
(c) In PASS 2, it is used to translate the operation code to machine language.

SYMTAB [Dynamic table] ↗(symbol inserted/deleted/searched)

(a) USES - SYMTAB includes the name and value (address) for each label in the source program, toghether with flags to indicate error conditions (eg: symbol defined at two different places).

(b) During Pass 1 of the assembler, labels are entered into SYMTAB as they are encountered in the source program, along with their assigned addresses (form LOCCTR)

(c) During Pass 2, symbols used as operands are looked up in SYMTAB to obtain the addresses to be inserted in the assembled instruction



Source program
READ(Label, opcode, operand)
(Mnemonics & opcode mappings are referenced)
PASS 1
PASS 2
Object Codes
OPTAB
SYMTAB
Created
SYMTAB
Labels & address mapping
Label & address mapping are referenced

U19CS012

2.> What are assembler directives? List any three assembler directives

2.>

Assembler Directives are instructions that direct the assembler to do something.

Directives do many things:

① Some tell the assembler to set aside space for variables

② other's tell the assembler to include additional source files,

③ ~~and~~ and other establish the start address for your program.

[Note - Assembler Directives can't generate machine code]

Example of assembler directives

① START <address-constant> - indicates the first word of target program will start on ROM memory location with address <address-constant>

START 400 - Rom Location would be 200 where first machine code will recide.

② END - This directive indicates the end of source program.

③ EQU ⇒ <symbol> EQU <address spec>

Eg: A EQU 100 :- A is assigned to address space 100.

Other Advanced Assembler Directives

④ ORIGIN ⑥ DROP

⑤ USING ⑦ LTORG.

3.> Find out addresses of variable using LC.

Step 1: First identify all variables in your Program.

Step 2: Replace all symbolic address with numeric address.

Step 3: Replace symbolic opcodes by maching operation codes.

Step ② Assembly Instruction			Step ① LC	Step ③ Machine code		
	START	101				
	READ	N 113	101)	09	0	113
	MOVER	BREG, ONE 115	102)	04	2	115
	MOVEM	BREG, TERM 116	103)	05	2	116
AGAIN	MULT	BREG, TERM 116	104)	03	2	116
	MOVER	CREG, TERM 116	105)	04	3	116
	ADD	CREG, ONE 115	106)	01	3	115
	MOVEM	CREG, TERM 116	107)	05	3	116
	COMP	CREG, N 113	108)	06	3	113
	BC	LE, AGAIN	109)	07	2	104
	MOVEM	BREG, RESULT 114	110)	05	2	114
	PRINT	RESULT 114	111)	10	0	114
	STOP		112)	00	0	000
N	DS	1	113)			
RESULT	DS	1	114)			
ONE	DC	'1'	115)			
TERM	DS	1	116)			
	END					

memory reserved by no code generated

SrNo	Variable Name	Address
1	N	113
2	RESULT	114
3	ONE	115
4	TERM	116

4.> Design an automata for set of all strings of length 5.

Assuming $\Sigma = \{0, 1\}$

$|w| = 5$

L = { 00000, 00001, 00010, 00011, ...
.... 11111 }

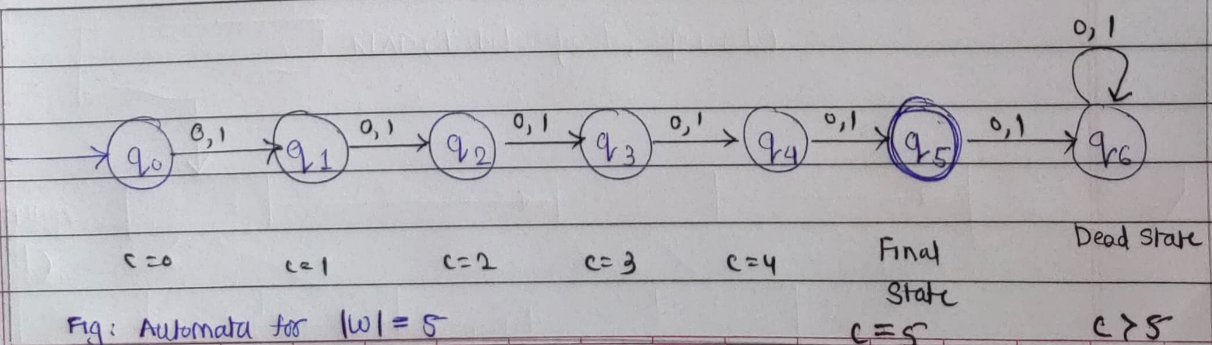


Fig: Automata for |w| = 5

D19CS012

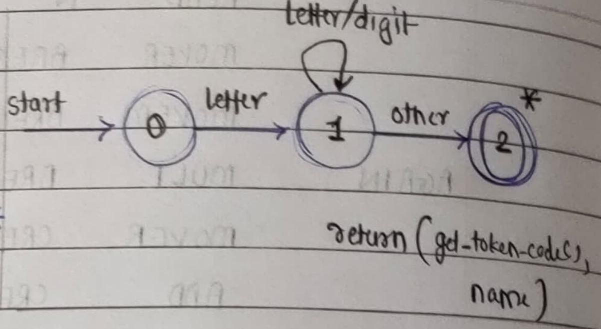
5.> Design an automata for identifying constants and keywords.

Identifiers and Reserved words

letter = [a-z A-Z]

digit = [0-9]

Identifier = letter (letter/digit)*



① • $\boxed{\text{get-token-code()}}$ searches a tables to check if the name is reserved word or constant, and returns its integer code if so.

② otherwise, it returns the integer code of the IDENTIFIER token, with $\boxed{\text{name}}$ containing the string of characters forming the token.

{ name, is not relevant for reserved words }