TUTORIAL 4

U19CS012

1.) What are the main data structure necessary for an assembly scheme? State purpose of each of them.

1.) The various data structure used by assembler are:

(1) Symbol Table (SYMTAB)

Purpose - used to contain all Symbol used in program (variables, procedures, defined constants, labels)

:) It is generated by the analysis phase and is used by synthesis phase.

SYMTAB

SYMBOL	VALUE	LOCATION	LENGTH
N	1	21	04

(2) Literal table (LITTAB)

Purpose - contains information about all literals encountered in assembly program.

- various literal are allocated address when 'LTORG' statement is used. to ↑
- If 'LTORG' is absent, then 'END' statement is encountered

Literal	Address
='2'	212
='1'	213
='6'	220

LIT TAB

(3) Mnemonic Table (MOT)

Purpose - contain name of mnemonic, its binary opcode & its length.

- [Mnemonic key is used for searching MOT]
- MOT / static or fixed table in nature

MOT

Mnemonic	Opcode	Length
ADD	01	1
SUB	02	1

(4) Operation Code Table (OPTAB) or (POT)

Purpose: contains the fields: mnemonic opcode, class & mnemonic info.

- 'class' field indicates where opcode corresponds to an IS / DS / Assembler Directive (Declarative statement)

POT / OPTAB

Mnemonic Opcode	Class	Mnemonic Info
MOVER	IS	(04, 1)
DS	DL	R#7
START	AD	R#11

(5) Location Counter (LC)

Purpose - Keeps the track of each instruction's location
- used to perform memory allocation.

Eg: START 501

LC contains the value 501. Therefore LC = 501.

(6) Pool Table (POOLTAB)

Literal no
#1
#3
-

Purpose - used to record the details of all different literal pools.
- 'LTORG' statement

2> Write down the pass number (PASS 1 / PASS 2) of the following activities that occur in two pass assembler.

(a) Object Code Generation - PASS 2
(b) Literals added to Literal table. - PASS 1
(c) Listing printed - PASS 2
(d) Address location of local symbols. - PASS 1

The functions performed in pass 1 and pass 2 in 2 pass assember

Pass 1
- ① Assign address to all address in the program
- ② Save the values assigned to all labels for use in pass 2
- ③ Perform some processing of assembler directives.

Pass 2
- ① Assemble instructions
- ② Generate data values defined by BYTE, WORD etc.
- ③ Perform processing of assembler directives not done during pass 1.
- ④ Write the program and assembling listing

UI9CS012

3.) Explain the concept of single pass assembler with suitable example.

3.)

(1) A single pass assembler - scans the program only once and creates equivalent binary program.
- The assembler substitutes all of the symbolic instructions with machine code in one pass.

(2) When are they used?

Source program.

+ It is necessary / desirable to avoid a second pass over the
+ the external storage for intermediate file between two passes is slow / inconvenient to use.

(3) ~~Forward Referencing Problem~~

(3) It is generally faster than two pass assembler.

(4) Single pass assembler constructs symbol table, Literal table, and also uses mnemonic table & operation table.

(5) It also performs Lc processing as is done by two pass assembler.

(6) Main Problem in One Pass Assembler

"Forward Referencing" - using a variable before its definition

Eg:
```
START 100
MOVEM AREG, X
MOVEM AREG, Y
ADD   AREG, Y   <- is making forward reference.
X DC '4'
Y DC '5'
END
```

(7) Soln - Backpatching

Backpatching is a process in which the operator field of an instruction containing a forward reference is (Left Blank) initially. The address of forward reference symbol is put into this field when its definition is encountered in the program.

(8) In order to perform backpatching, single pass assembler requires additional data structure called Table of Incomplete Instructions (TII)

(9) By the time END statement is processed, the SYMTAB would contain the addresses of all symbols defined in the source program and TII would contain all info regarding forward references.

(10) Imp Aspects of one-pass assembler - FACT - It loads object code directly in memory and does not generate an object file. This makes it possible for assembler to go back and complete instructions in memory any time during assembly.

EXAMPLE :

Source Program	Target code
START 100	01 - 100
MOVER AREG, A	100) 01 01 107
PRINT B	101) 09 - 501
ADD BREG, ='9'	102) 03 02 105
SUB BREG, D	103) 04 02 503
COMP CREG, ='23'	104) 08 03 106
LTORG	105) - - 009
	106) - - 023
A DS 3	107) - - 003
LABEL : EQU A	No Code Generation
ORIGIN 500	No Code Generation
L1 : MULT CREG, ='7'	500) 05 03 504
B DC 10	501) - - 010
MOVEM CREG, ='7'	502) 02 03 504
D DC 8	503) - - 008
END	504) - - 007

U19CS012

Table of Incomplete Instructions (TII)

LC No	Incomplete Instruction
100	A
101	B
102	='9'
103	D
104	='23'
500	='7'
502	='7'

(MOT)

OP-code	Mnemonic
01	MOVER
02	MOVEM
03	ADD
04	SUB
05	MUL
06	DIV
07	BC
08	COMP
09	PRINT
10	READ

POT

OP-code	mnemonic
01	START
02	END
03	EQ
04	ORIGIN
05	LTORG

DL

op-code	mnemonic
01	DS
02	DC

SYMBOL TABLE

Sym_No	Symbol	Address
01	A	107
02	B	501
03	D	503
04	LABEL	107
05	L1	500

REGISTERS

Reg_No	Name
01	AREG
02	BREG
03	CREG
04	DREG

Literal table

Lit_No	Literal	Address
01	='9'	105
02	='23'	106
03	='7'	504

Pool table

01
03

U19CS012

4.> Show the content of the symbol table, MOT at the end of PASS 1.

START 102	LC
READ X	102
READ Y	103
MOVER AREG, X	104
ADD AREG, Y	105
~~MOVEM AREG, X~~	~~106~~
MOVEM AREG, RESULT	106
PRINT RESULT	107
STOP	108
X DS 1	109
Y DS 1	110
RESULT DS 1	111
END	

[* Sample Program to find X+Y]

SYMBOL TABLE

Symbol	Address	length
X	109	1
Y	110	1
RESULT	111	1

MOT :- (at the End of Pass 1)

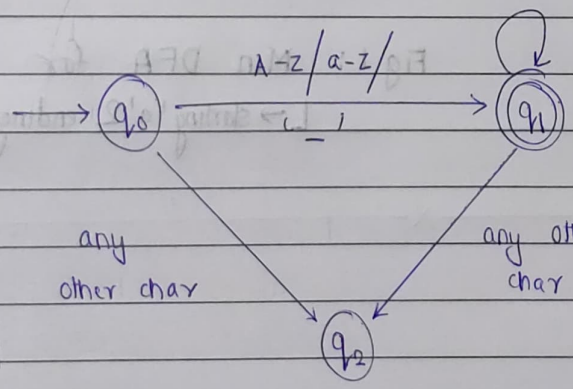
LC	Mnemonics	Type	Opcode	Length
-	START	AD	01	-
102	READ	IS	09	01
103	READ	IS	09	01
104	MOVER	IS	04	01
105	ADD	IS	01	01

LC	Mnemonics	Type	opcode	Length
106	MOVEM	IS	05	01
107	PRINT	IS	10	01
108	STOP	IS	00	01
109	DS	DL	02	-
110	DS	DL	02	-
111	DS	DL	02	-
-	END	AD	02	-

5.) Design an automata for valid identifier, integer & constant.

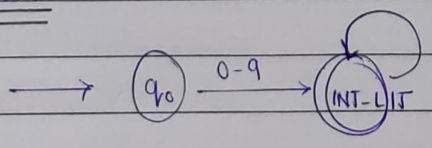
5.)

Valid Identifier

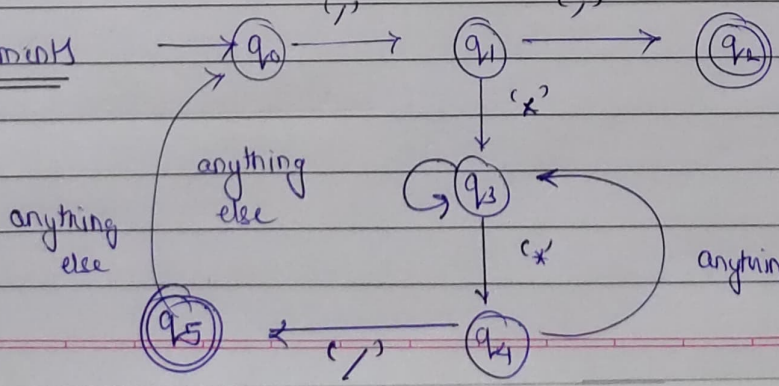


* special keyword's like "int, struct, class, ..." are not included in above DFA.

Integer Constant



Comments



U19CS012

6.) Design an automata which accepts a language of all strings starting with 'a' and ending with 'b'.

6.)

Considering $\Sigma = \{a, b\}$

$L = \{ ab, aab, aaab, abb, \ldots \}$

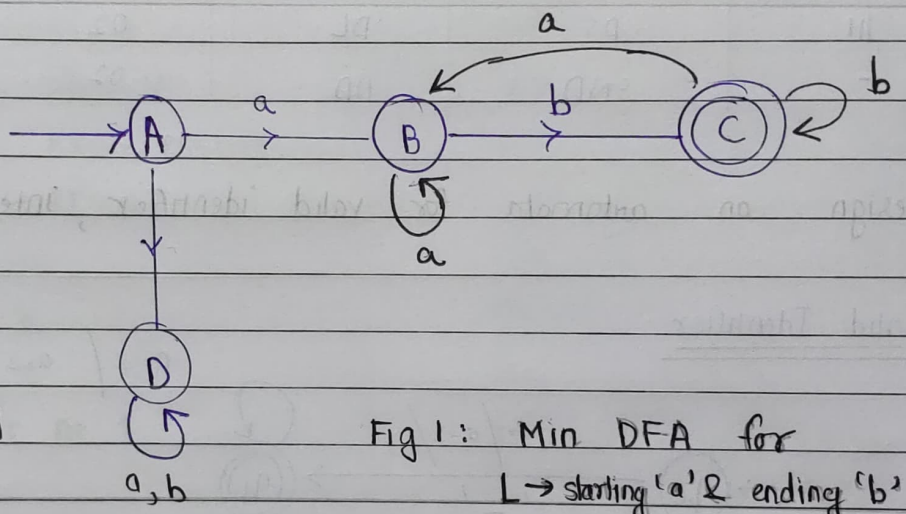


Fig 1: Min DFA for L → starting 'a' & ending 'b'