Algorithm 4.1 (Assembler First Pass)

1. *loc_cntr* := 0; (default value)
   pooltab_ptr := 1; POOLTAB [1] := 1;
   littab_ptr := 1;
2. While next statement is not an END statement
   (a) If label is present then
      this_label := symbol in label field;
      Enter (*this_label*, *loc_cntr*) in SYMTAB.
   (b) If an LTORG statement then
      (i) Process literals LITTAB [POOLTAB [*pooltab_ptr*]] ... LITTAB [*littab_ptr*−1] to allocate memory and put the address in the address field. Update *loc_cntr* accordingly.
      (ii) *pooltab_ptr* := *pooltab_ptr* + 1;
      (iii) POOLTAB [*pooltab_ptr*] := *littab_ptr*;
   (c) If a START or ORIGIN statement then
      loc_cntr := value specified in operand field;
   (d) If an EQU statement then
      (i) *this_addr* := value of <*address spec*>;
      (ii) Correct the symtab entry for *this_label* to (*this_label*, *this_addr*).
   (e) If a declaration statement then
      (i) *code* := code of the declaration statement;
      (ii) *size* := size of memory area required by DC/DS.
      (iii) *loc_cntr* := *loc_cntr* + *size*;
      (iv) Generate IC '(DL, *code*) ...'.
   (f) If an imperative statement then
      (i) *code* := machine opcode from OPTAB;
      (ii) *loc_cntr* := *loc_cntr* + instruction length from OPTAB;
      (iii) If operand is a literal then
         this_literal := literal in operand field;
         LITTAB [*littab_ptr*] := *this_literal*;
         littab_ptr := *littab_ptr* + 1;
      else (i.e. operand is a symbol)
         this_entry := SYMTAB entry number of operand;
         Generate IC '(IS, *code*)(S, *this_entry*)';
3. (Processing of END statement)
   (a) Perform step 2(b).
   (b) Generate IC '(AD,02)'.
   (c) Go to Pass II.

Input to Pass I Assembler is the Source program and all the tables below.

Output will be Intermediate code, SYMTAB and/or POOLTAB, and LITTAB

	Source program	Intermediate Code	LC
1	START 200	(AD, 01) (C, 200)	
2	MOVER AREG, ='5'	(IS, 04) (1) (L, 01)	200
3	MOVEM AREG, A	(IS, 05) (1) (S, 01)	201
4 LOOP	MOVER AREG, A	(IS 04) (1) (S, 01)	202
5	MOVER CREG, B	(IS, 04) (3) (S, 03)	203
6	ADD CREG, ='1'	(IS, 01) (3) (L, 02)	204
	[Put some 5 Imperative statements]	Literal processing after LTORG	
		Write IC	
12 B	BC ANY, NEXT	(IS, 07) (6) (S, 04)	210
13	LTORG	='5' (DL, 01) (C, 5)	211
14	[Put any 1 Imperative Statement]	='1' (DL, 01) (C, 1)	212
		→ Write IC	213
15 NEXT	SUB AREG, ='4'	(IS, 02) (1) (L, 03)	214
16	BC LT, BACK.	(IS, 07) (1) (S, 05)	215
17 LAST	STOP.	(IS, 00)	216
18	ORIGIN LOOP+2	(AD, 03) (S, 02) +2	~~217~~
19	MULT CREG, B	(IS, 03) (3) (S, 03)	204
20	ORIGIN LAST+1	(AD, 03) (S, 06) +1	~~217~~
21 A	DS 1	(DL, 02) (C, 1)	217
22 BACK	EQU LOOP	← NO IC →	
23 B	DS 1	(DL, 02) (C, 1) 218	
24.	END	(AD, 02)	

Processing literal after END → (DL, 01) (C, 1) 219

C: Constant L: Literal S: Symbol

AD (Assembler Directives)			CC (Conditional Codes)	OPTAB (IS)
		START 01	LT - 1	00 - STOP
DL		END 02	LE - 2	01 - ADD
DC	01	ORIGIN 03	EQ - 3	02 - SUB
DS	02	EQU 04	GT - 4	03 - MULT
		LTORG 05	GE - 5	04 - MOVER
			ANY - 6	05 - MOVEM
				06 - COMP.
				07 - BC
				08 DIV
				09 READ
				10 - PRINT

Registers		
	AREG - 1	CREG - 3
	BREG - 2	DREG - 4.

SYMTAB

Symbol	Address
A	217
LOOP	202
B	218
NEXT	214
BACK	202
LAST	216

LITTAB

Literal	Address
5	211
1	212
1	219

POOLTAB

1	1
2	3

Note 1:-

In the OPTAB table please include "length of Instruction" as one of the coloumn. This will be used for LC processing.

NOTE 2:-

Do Error handling also:
(Refer Dhamdhere Pg 108).

Rainbo

PAGE:
DATE: /

One more test Case

```
START 101
READ N
MOVER BREG, ONE
MOVEM BREG, TERM
AGAIN MULT BREG, TERM
MOVER CREG, TERM
ADD CREG, ONE
MOVEM CREG, TERM
COMP CREG, N
BC LE, AGAIN
DIV BREG, TWO
MOVEM BREG, RESULT
PRINT RESULT
STOP
N DS 1
RESULT DS 1
ONE DC '1'
TERM DS 1
TWO DC '2'
END
```