DATASTRUCTURES USED BY 2 PASS ASSEMBLER

1. **OPCODE Table(OPTAB)**

OPTAB(mnemonic opcode,m/c code,class)

2. **Symbol Table(SYMTAB)**

SYMTAB(sym-id, name, addr, length)

OPTAB

Instruction (Mnemonic/ Declaration/ Assembler Directive)	Statement Class	Machine Code
STOP	IS	00
ADD	IS	01
SUB	IS	02
MULT	IS	03
MOVER	IS	04
MOVEM	IS	05
COMP	IS	06
BC	IS	07
DIV	IS	08
READ	IS	09
PRINT	IS	10
DC	DL	01
DS	DL	02
START	AD	01
END	AD	02
ORIGIN	AD	03
EQU	AD	04
LTORG	AD	05

Register Table

Reg name	M/c Code
AREG	1
BREG	2
CREG	3
DREG	4

Input to Pass 1 of 2 Pass Assembler

AL P	
1.	START 200
3.	MOVEM AREG, A
4. LOOP	MOVER AREG, A
5.	MOVER CREG, B
11. LAST	STOP
12.	ORIGIN LOOP
13.	MULT CREG, B
14. A	DS 1
15. BACK	EQU LOOP
16. B	DC 1
17 END	

I/C ODE		
	(AD,O1)	(C,200)
200)	(IS,05)	(1)(S,01)
201)	(IS,04)	(1)(S,01)
202)	(IS,04)	(3)(S,03)
210)	(IS,00)	
	(AD,03)	(S,02)
202)	(IS,03)	(3)(S,3)
203)	(DL,02)	(C,1)
	(AD,04)	(S,02)
204)	(DL,01)	(C,1)
205)	(AD,02)	--

SYMTAB			
Sym_id	Sym_name	Sym_addr	length
1	A	203	1
2	LOOP	202	1
3	B	204	1
4	BACK	202	1
5	LAST	210	1

AL P		
	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	
	MOVEM BREG, RESULT	
	PRINT RESULT	
	STOP	
N	DS 1	
RESULT	DS 1	
ONE	DC ‘1’	
TERM	DS 1	
	END	

I/C CODE			
	(AD,O1)	(C,101)	
101)	(IS,09)	(S,01)	
102)	(IS,04)	(2)	(S,02)
103)	(IS,05)	(2)	(S,03)
104)	(IS,03)	(2)	(S,03)
105)	(IS,04)	(3)	(S,03)
106)	(IS,01)	(3)	(S,02)
107)	(IS,05)	(3)	(S,03)
108)	(IS,06)	(3)	(S,01)
109)	(IS,07)	(2)	(S,04)
110)	(IS,05)	(2)	(S,05)
111)	(IS,10)		(S,05)
112)	(IS,00)		
113)	(DL,02)		(C,1)
114)	(DL,02)		(C,1)
115)	(DL,01)		(C,1)
116)	(DL,02)		(C,1)
	(AD,02)		

AL P			I/C CODE			M/C CODE			
	START 101			(AD,O1)	(C,101)				
	READ N	101)	(IS,09)	(S,01)		101) +	09		113
	MOVER BREG, ONE	102)	(IS,04)	(2)(S,02)		102) +	04	2	115
	MOVEM BREG, TERM	103)	(IS,05)	(2)(S,03)		103) +	05	2	116
AGAIN	MULT BREG, TERM	104)	(IS,03)	(2)(S,03)		104) +	03	2	116
	MOVER CREG, TERM	105)	(IS,04)	(3)(S,03)		105) +	04	3	116
	ADD CREG, ONE	106)	(IS,01)	(3)(S,02)		106) +	01	3	115
	MOVEM CREG, TERM	107)	(IS,05)	(3)(S,03)		107) +	05	3	116
	COMP CREG, N	108)	(IS,06)	(3)(S,01)		108) +	06	3	113
	BC LE, AGAIN	109)	(IS,07)	(2)(S,04)		109) +	07	2	104
	MOVEM BREG, RESULT	110)	(IS,05)	(2)(S,05)		110) +	05	2	114
	PRINT RESULT	111)	(IS,10)	(S,05)		111) +	10	0	114
	STOP	112)	(IS,00)			112)+	00	0	000
N	DS 1	113)	(DL,02)	(C,1)		113)			
RESULT	DS 1	114)	(DL,02)	(C,1)		114)			
ONE	DC ‘1’	115)	(DL,01)	(C,1)		115) +	00	0	001
TERM	DS 1	116)	(DL,02)	(C,1)		116)			
	END		(AD,02)						

Algorithm for Pass 1 of 2 pass Assembler

1. **loc_cntr : = 0; pooltab_ptr := 1; POOLTAB[1] :=1;**
   littab_ptr :=1; symtab_ptr:=1;

2. **While next stmt is not an END stmt**
   (a) If label is present then .....
   (b) If an LTORG stmt then .....
   (c) If START or ORIGIN stmt then .....
   (d) If an EQU stmt then …….
   (e) If a declaration stmt then …….
   (f) If an imperative stmt then …….

3. **(Processing of END stmt)**
   (a) Perform step 2(b)
   (b) Generate Intermediate code.
   (C) Goto Pass 2 .

Algorithm for Pass 1 of 2 pass Assembler (contd….)

2 (a) If label is present then

this_label = symbol in label field

Enter(this_label, loc_cntr) in SYMTAB

(b) If an LTORG stmt then

i. Process literals LITTAB[POOLTAB[pooltab_ptr]]……LITTAB[littab_ptr-1]

to allocate memory and put the address field. Update loc_cntr.

ii. Pooltab_ptr:=pooltab_ptr+1; iii. POOLTAB[pooltab_ptr] := littab_ptr;

(c) If START or ORIGIN stmt then

loc_cntr := value in operand field

(d) If an EQU stmt then

this_addr=value of <address spec> , update the SYMTAB entry

(e) If a declaration stmt then

code=code of declaration stmt, size=size req by DC/DS.

update the SYMTAB entry

loc_cntr=loc_cntr+size, generate Intermediate code

(f) If an imperative stmt then

code=m/c code from MOT, loc_cntr=loc_cntr+length of instr

If operand is literal then

this_lit=literal in operand field

LITTAB[littab_ptr]=this_lit

littab_ptr=littab_ptr+1

else this_entry=SYMTAB entry no of operand

symtab_ptr=symtab_ptr+1

Algorithm for Pass 2 of 2 pass Assembler

1. code_area_addr=addr of code_area, pooltab_ptr=1, loc_cntr=0
2. While next stmt is not an END stmt

(a) clear machine_code_buffer

(b) If an LTORG stmt

(i) Process literals in LITTAB[POOLTAB[pooltab_ptr]]……..LITTAB[POOLTAB[pooltab_ptr+1]-1] assemble literals in machine_code_buffer.

(ii) size=size of memory req for literals (iii)pooltab_ptr=pooltab_ptr+1

(c) If a START or ORIGIN stmt then

(i) loc_cntr=value specified in operand field (ii) size=0

(d) If a declaration stmt

(i) If a DC stmt then

Assemble the const in machine_code_buffer.

(ii) size=size of memory required by DC/DS stmt

(e) If an imperative stmt

(i) Get operand address from SYMTAB or LITTAB

(ii) Assemble instr in machine_code_buffer. (iii) size=size of instr

(f) If size <> 0 then

(i) Move contents of machine_code_buffer to the address code_area_addr+loc_cntr

(ii) loc_cntr=loc_cntr+size

3.Processing of END stmt

(a) Perform step 2(b) and 2(f)

(b) Write code_area into output file.

Input to Pass 1 of 2 Pass Assembler
Batch 1

AL P		
1.	START 100	
2.	MOVER AREG, A	
3. L1	ADD BREG, A	
4.	MOVER BREG, B	
5.	ORIGIN L1	
6.	MOVER BREG,A	
7. A	DS 5	
8. B	DC 5	
9.	END	

I/C CODE	
LC (C,100)	(AD,O1)
100)	(IS,04) (1)(S,01)
101)	(IS,01) (1)(S,01)
102)	(IS,04) (2)(S,03)
(S,02)	(AD,03)
101)	(IS,04) (2)(S,01)
102)	(DL,02)

M/C CODE		
100) 102	04	1
101 102	01	1
102) 107	04	2
101) 102	04	2
102) 5	02	
107) 5	01	
108)		

SYMTAB

Sym_id	Sym_name	Sym_ad	
1	A	102	5
2	L1	101	1
3	B	107	1

Input to Pass 1 of 2 Pass Assembler
Batch 2

AL P
1. START 400
2. MOVER AREG,A1
3. LOOP SUB BREG, A1
4. MOVER BREG, B1
5. ORIGIN 300
6. MOVER BREG,A1
7. A1 DS 3
8. B1 DC 3
9. END

I/C CODE	
LC (C,400)	(AD,O1)
400)	(IS,04) (1)(S,01)
401)	(IS,02) (2)(S,01)
402)	(IS,04) (2)(S,03)
(C,300)	(AD,03)
300)	(IS,04) (2)(S,01)
301)	(DL,02)

M/C CODE		
400) 301	04	1
401) 301	02	1
402) 304	04	2
300) 301	04	2
301) 3	02	
304) 3	01	
305)		

SYMTAB			
Sym_id	Sym_name	Sym_ad	
1	A1	301	3
2	LOOP	401	1
3	B1	304	1

Input to Pass 1 of 2 Pass Assembler
Batch 3

AL P
1. START 250
2. MOVEM AREG,A
3. LOOP MULT BREG, A
4. MOVEM BREG, B
5. TERM EQU LOOP
6 MOVEM BREG,A
7. A DS 3
8. B DC 3
9. END

I/C CODE	
LC (C,250)	(AD,O1)
250)	(IS,05) (1)(S,01)
251)	(IS,03) (2)(S,01)
252)	(IS,05) (2)(S,03)
(S,02)	(AD,04)
253)	(IS,05) (2)(S,01)
254)	(DL,02)

M/C CODE		
250) 254	05	1
251) 254	03	2
252) 257	05	2
253) 254	05	2
254) 3	02	
257) 3	01	
258)		

SYMTAB

Sym_id	Sym_name	Sym_ad	
1	A	254	3
2	LOOP	251	1
3	B	257	1
4	TERM	251	1

Input to Pass 1 of 2 Pass Assembler
Batch 4

AL P
1. START
2. MOVEM AREG,S1
3. L1 DIV BREG, S2
4. MOVEM BREG, S1
5. L2 EQU L1
6. MOVEM BREG,S1
7. S1 DC 4
8. S2 DS 3
9. END

I/C CODE	
LC (C,00)	(AD,O1)
00)	(IS,05) (1)(S,01)
1)	(IS,08) (2)(S,03)
2)	(IS,05) (2)(S,01)
(S,02)	(AD,04)
3)	(IS,05) (2)(S,01)
4)	(DL,01)

M/C CODE		
0) 4	05	1
1) 5	08	2
2) 4	05	2
3) 4	05	2
4) 4	02	
5) 3	01	
8)		

SYMTAB

Sym_id	Sym_name	Sym_ad	
1	S1	4	1
2	L1	1	1
3	S2	5	3
4	L2	1	1