SRES's Sanjivani College of Engineering, Kopargaon
(*An Autonomous Institute*)
Department of Computer Engineering

SPOS Lab Manual

Assignment No. 02

Title: Implementation of Pass 2 of Two Pass Assembler

Aim :
Design suitable data structures and implement pass-II of a two-pass assembler for pseudo-machine in Java using object oriented feature. Implementation should consist of a few instructions from each category and few assembler directives.

Inputs :
1. Assembly language program containing few instructions from each category
2. Intermediate representation of assembly language program
3. Symbol table
4. Literal Table
5. Pool Table

Outputs :
1. Machine language program (Object Program)

Theory :

1. Pass II of assembler use the information generated by Pass I of assembler such as symbol table, literal table and pool table and process the intermediate file to generate the machine language program
2. For Assemble Directive (AD) class of statements no machine code will be generated by Pass II. Therefore if Assemble Directive (AD) is found in input file then nothing is written in machine language (output) file

   e.g (AD, 01) (C, 200) ==> ----------
3. If LTORG (AD, 05) and END (AD, 02) Assembler Directives is found in input file then literal constants in current literal pool will be assigned the locations

   e.g 211 (AD, 05) (C, 5) 211) 00 0 005
   212 (AD, 05) (C, 1) 212) 00 0 001

   Here LTORG assign the location 211 to literal 5 and location 212 to literal 1

   e.g. 219 (AD, 02) (C, 1) 219) 00 0 001

   Here END assign the location 219 to literal 1
4. For Imperative Statements (IS) Pass II will generate machine laguage code. If IS class statement is found in input file then Pass II will write its Location Counter (LC), Machine Opcode, Register Constant and Address of Sybol or Literal (taken from symbol/literal table)

e.g 200 (IS, 04) (1) (L, 01) ==> 200) 04 1 211

e.g 203 (IS, 04) (3) (S,03) ==> 203) 04 3 218

5. For Delarative Class statements (DL) no machine code will be generated

   For DC (DL, 01) only constant value will be stored in the machine language code.

   e.g. 218 (DL, 01) (C, 1) 218) 00 0 001

   For DS (DL, 02) constant value is not stored in machine language code, but only LC value is incremented by the storage size declared by DS

   e.g. 217 (DL, 02) (C, 1) 217) ----------

   218)

   Here storage size is 1 byte so LC will incremented by 1 i.e. 217+1 = 218

Sample Example Showing Inputs and Outputs of Pass II of Assembler

SYMTAB

A SYMTAB entry contains the fields symbol entry number, symbol name, address and length

symbol entry number	Symbol Name	Symbol address	Symbol length
1	A	217	1
2	LOOP	202	1
3	B	218	1
4	NEXT	214	1
5	BACK	202	1
6	LAST	216	1

LITTAB

a LITTAB entry contains the fields literal entry nymber, literal value and address

Literal entry number	Literal value	address
1	=’5’	211
2	=’1’	212
3	=’1’	219

POOLTAB

a POOLTAB entry contains the fields pool number, literal number and number of literal in pool

Pool number	First literal entry number	Number of literals in pool
1	1	2
2	3	1
3	4	0

Assembly Program			Intermediate Program		Machine Language Program			
	START	200		(AD, 01) (C, 200)				
	MOVER	AREG, ='5	200	(IS, 04) (1) (L, 01)	200)	04	1	211
	MOVEM	AREG , A	201	(IS, 05) (1) (S, 01)	201)	05	1	217
LOOP	MOVER	AREG, A	202	(IS,04) (1) (S, 01)	202)	04	1	217
	MOVER	CREG, B	203	(IS, 04) (3) (S,03)	203)	04	3	218
	ADD	CREG, ='1'	204	(IS, 01) (3) (L, 02)	204)	01	3	212
	...							
	BC	ANY, NEXT	210	(IS, 07) (6) (S, 04)	210)	07	6	214
	LTORG		211	(AD, 05) (C, 5)	211)	00	0	005
			212	(AD, 05) (C, 1)	212)	00	0	001
	...							
NEXT	SUB	AREG, ='1	214	(IS, 02) (1) (L, 03)	214)	02	1	219
	BC	LT, BACK	215	(IS, 07) (1) (S, 05)	215)	07	1	202
LAST	STOP		216	(IS, 00)	216)	00	0	000
	ORIGIN	LOOP+2		----------		----------		
	MULT	CREG, B	204	(IS, 03) (3) (S, 03)	204	03	3	218
	ORIGIN	LAST+1		----------		----------		
A	DS	1	217	(DL, 02) (C, 1)	217)			
BACK	EQU	LOOP		----------		----------		
B	DC	1	218	(DL, 01) (C, 1)	218)	00	0	001
	END		219	(AD, 02) (C, 1)	219)	00	0	001

Important data structures used by algorithm for Pass II are

SYMTAB, LITTAB, and POOLTAB

LC : Location counter

littab_ptr : Points to an entry in LITTAB

pooltab_ptr : Points to an entry in POOLTAB

machine_code_buffer : Area for constructing code for one statement

code_area : Area for assembling target program

code_area_address : Contains address of code_area

Algorithm for Pass 2 of a two-pass assembler

1. code_area_address = address of code_area;
   pooltab_ptr = 1;
   LC = 0;
2. While the next statement is not an END statement
   (a) Clear machine_code_buffer;
   (b) If an LTORG statemnet
      (i) If POOLTAB[pooltab_ptr].#literals > 0 then
         Process literals in the entries LITTAB [POOLTAB [pooltab_ptr].first ... LITTAB [POOLTAB [pooltab_ptr+1]-1] similar to processing of constants in a DC statement. It results in assembling the literals in machine_code_buffer.
      (ii) size = size of memory area required for literals;
      (iii) pooltab_ptr = pooltab_ptr + 1;
   (c) If a START or ORIGIN statement
      (i) LC =value specified in operand field;
      (ii) size = 0;
   (d) If a declaration statement
      (i) If a DC statement then
         Assemble the constant in machine_code_buffer.
      (ii) size = size of the memory area required by the declaration statement;
   (e) If an imperative statement
      (i) Get address of the operand from its entry in SYMTAB or LITTAB, as per the case
      (ii) Assemble the instruction in machine_code_buffer.
      (iii) size = size of instruction;
   (f) If size ≠ 0 then
      (i) Move contents of machine_code_buffer to the memory word with the address code_area_address + <LC>;
      (ii) LC = LC + size;
3. Procsssing of END statement
   (a) Perform actions (i) – (iii) of step 2 (b)
   (b) Perform actions (i) – (ii) of step 2 (f)
   (c) Write code_area into the output file

Conclusion: In this assignment we have implemented pass II of assembler. Information from symbol table, literal table and pool table is used to process intermediate file and to generate machine code file.

References: Systems Programming & Operating Systems by D M Dhamdhere

Prepared by
Prof.N.G.Pardeshi
Subject Teacher

Approved by
Dr. D.B.Kshirsagar
HOD