# MES Wadia College of Engineering Pune-01

## Department of Computer Engineering

| | |
|---|---|
| **Name of Student:** | **Class:** |
| **Semester/Year:** | **Roll No:** |
| **Date of Performance:** | **Date of Submission:** |
| **Examined By:** | **Experiment No: Group A-01** |

**Group A: ASSIGNMENT NO: 01**

**AIM: Design suitable Data structures and implement Pass-I and Pass-II of a two-pass assembler for pseudo-machine. Implementation should consist of a few instructions from each category and few assembler directives. The output of Pass-I (intermediate code file and symbol table) should be input for Pass-II.**

**OBJECTIVES:**

- To implement basic language translator by using various needed data structures.
- To implement basic Assembler Pass I and Pass II

**PRE-REQUISITES:**

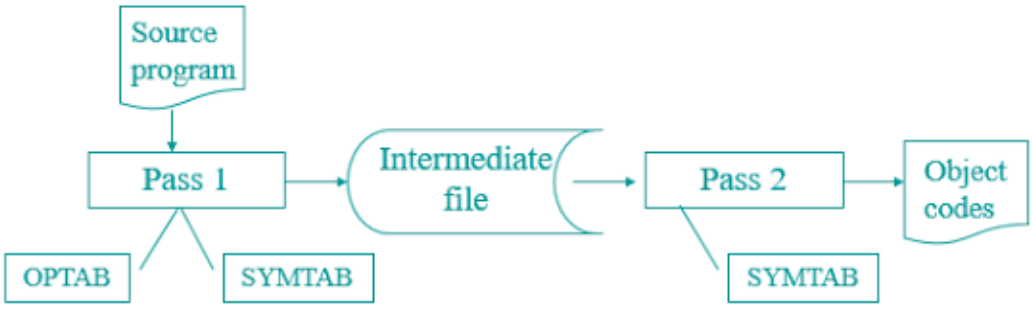
1.Eclipse java.

**2.**Basics of Language processors.

**APPARATUS:**

**THEORY:**

Design of two pass assembler:




**Algorithm (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
      (1) Process literals LITTAB [POOLTAB [pooltab_ptr]]. . .LITTAB[littab_ptr -1]
          to allocate memory ant put the address in the address field. Update loc_cntr accordingly.
      (2) pooltab_ptr := pooltab_ptr  + 1;
      (3) 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
      (1) this_addr := value of < address spec>;
      (2) Correct the symtab entry for this_label to (this_label , this_addr).
  (e)  If a declaration statement then
      (1) code := code of the declaration statement;
      (2) size := size of memory  area required by DC/DS.
      (3) loc_cntr := loc_cntr + size;
      (4) Generate IC '(DL,code). . .' .
  (f)  If an imperative statement then
      (1) code := machine opcode from OPTAB;
      (2) loc_cntr := loc_cntr + instruction length from OPTAB;
      (3) 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.

e.g.

```
1          START   200
2          MOVER   AREG, ='5'      200)  +04 1 211
3          MOVEM   AREG, A         201)  +05 1 217
4   LOOP   MOVER   AREG, A         202)  +04 1 217
5          MOVER   CREG, B         203)  +05 3 218
6          ADD     CREG, ='1'      204)  +01 3 212
7          ...

12         BC      ANY, NEXT       210)  +07 6 214
13         LTORG
                   ='5'            211)  +00 0 005
                   ='1'            212)  +00 0 001
14         ...
15  NEXT   SUB     AREG, ='1'      214)  +02 1 219
16         BC      LT, BACK        215)  +07 1 202
17  LAST   STOP                    216)  +00 0 000
18         ORIGIN  LOOP+2
19         MULT    CREG, B         204)  +03 3 218
20         ORIGIN  LAST+1
21  A      DS      1               217)
22  BACK   EQU     LOOP
23  B      DS      1               218)
24         END
25                 ='1'            219)  +00 0 001
```

*Pass I Use following Data Structures*

- OPTAB

| *mnemonic opcode* | *class* | *mnemonic info* |
|---|---|---|
| MOVER | IS | (04,1) |
| DS | DL | R#7 |
| START | AD | R#11 |
| | : | |

OPTAB

- SYM TAB

| symbol | address | length |
|---|---|---|
| LOOP | 202 | 1 |
| NEXT | 214 | 1 |
| LAST | 216 | 1 |
| A | 217 | 1 |
| BACK | 202 | 1 |
| B | 218 | 1 |

SYMTAB

- LITTAB

| | value | address |
|---|---|---|
| 1 | ='5' | |
| 2 | ='1' | |
| 3 | ='1' | |

LITTAB

- POOLTAB

| | first | # literals |
|---|---|---|
| 1 | 1 | 2 |
| 2 | 3 | 1 |

**Algorithm for pass II** assumes that the intermediate code is stored in the file. Target code will be assembled in the area named code area.

**Algorithm (Assembler Second Pass):**

1. *code_area_address*:= address of code_area;
   *pooltab_ptr* :=1;
   *loc_cntr :*=0;

2.While next statement is not an END statement

(a) Clear *machine_code_buffer ;*
(b) If an LTORG statement

(i) Process literals in LITTAB[POOLTAB[*poottab_ptr*]]... LTAB [POOLTAB[*pooltab_ptr*+1]]-1 similar to processing of constants in a DC statement , i.e.
assemble the literals in *machine_code_buffer*;
(ii) *size*:= size of memory area required for literals;
(iii) *pooltab_ptr* := *pooltab_ptr*+1;

(c) If START or ORIGIN statement then
(i) *loc_ctr* := value specified in operand feild;
(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 memory area required by DC/DS;

(e) if an IMPERATIVE STSATEMENT
(i) Get operand address from **SYMTAB** or **LITTAB**.
(ii) Assemble Instruction in *macheine_code_buffer*.
(iii) *size*:=size of instruction.

(f) IF ***size* != 0** then
(i) Move content of *machine_code_buffer* to the address *code_area_address +loc_cntr*;
(ii) *loc_cntr*:= *loc_cntr+size*;

3. (Processing of END statement)
(a) Perform step 2(b) and 2(f).
(b) Write *code_area* into output file.

**CONCLUSION:**

**QUESTIONS:**

1) What is forward reference? How it is handled in 2 pass assembler?
2) What is ORIGIN statement?
3) Explain EQU statement with example.
4) Explain variants of intermediate code?
5) Which data structures are used in pass I?
6) Which data structures are used in Pass II?
7) Give Example of LTORG statement.