Assignment No. 1

Part I: Systems Programming and Operating Systems, Group A, 1

Problem Statement: Design suitable data structures and implement pass1 and pass2 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 pass1 (intermediate code file and symbol table) should be input for pass2

Objectives:

1. To learn systems programming tools
2. Implement language translator

Software Requirement:

Operating System recommended :- 64-bit Open source Linux or itsderivative

Programming tools recommended: - Eclipse IDE

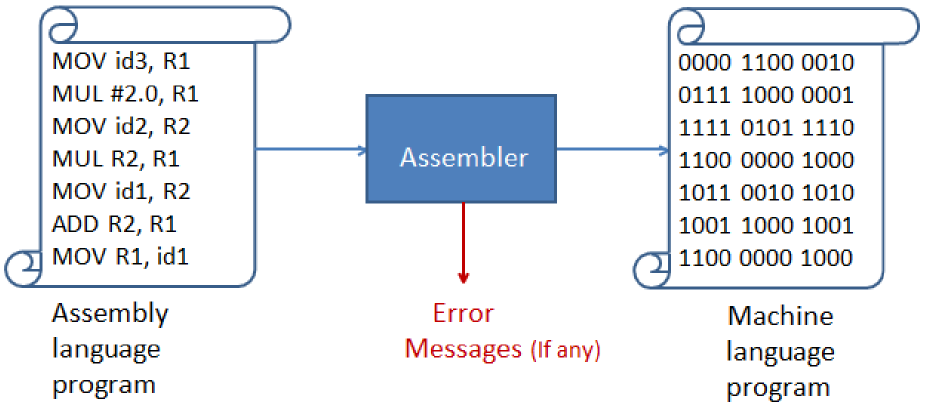
Hardware Requirement:I3 and I5 machines

Theory:

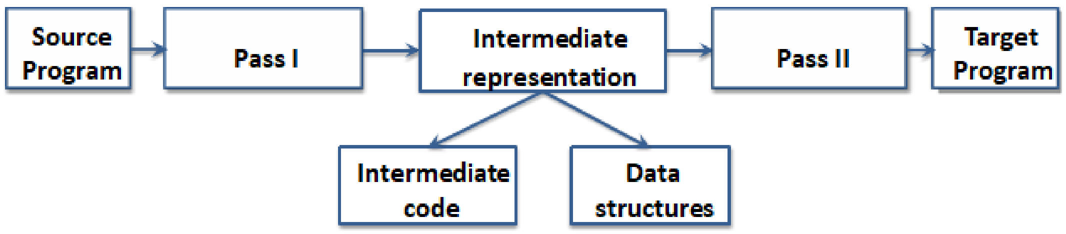
1. Assembler is a low level translator which translates source code to machine code
2. It works in two phases : analysis phase and synthesis phase
3. In analysis phase, source assembly code is analysed to generate some intermediate data structures
4. In synthesis phase, machine code is generated
5. There are well defined system level standard algorithms to design the assembler as a two pass assembly, namely PassI and PassII algorithms of assembler
6. PassI takes the source code in assembly language as input and generates intermediate data structures.
7. PassII takes the intermediate data structures generated by PassI as input and generates machine code.

Assembler

- Assembler is a language processor that converts assembly language program to machine language program.

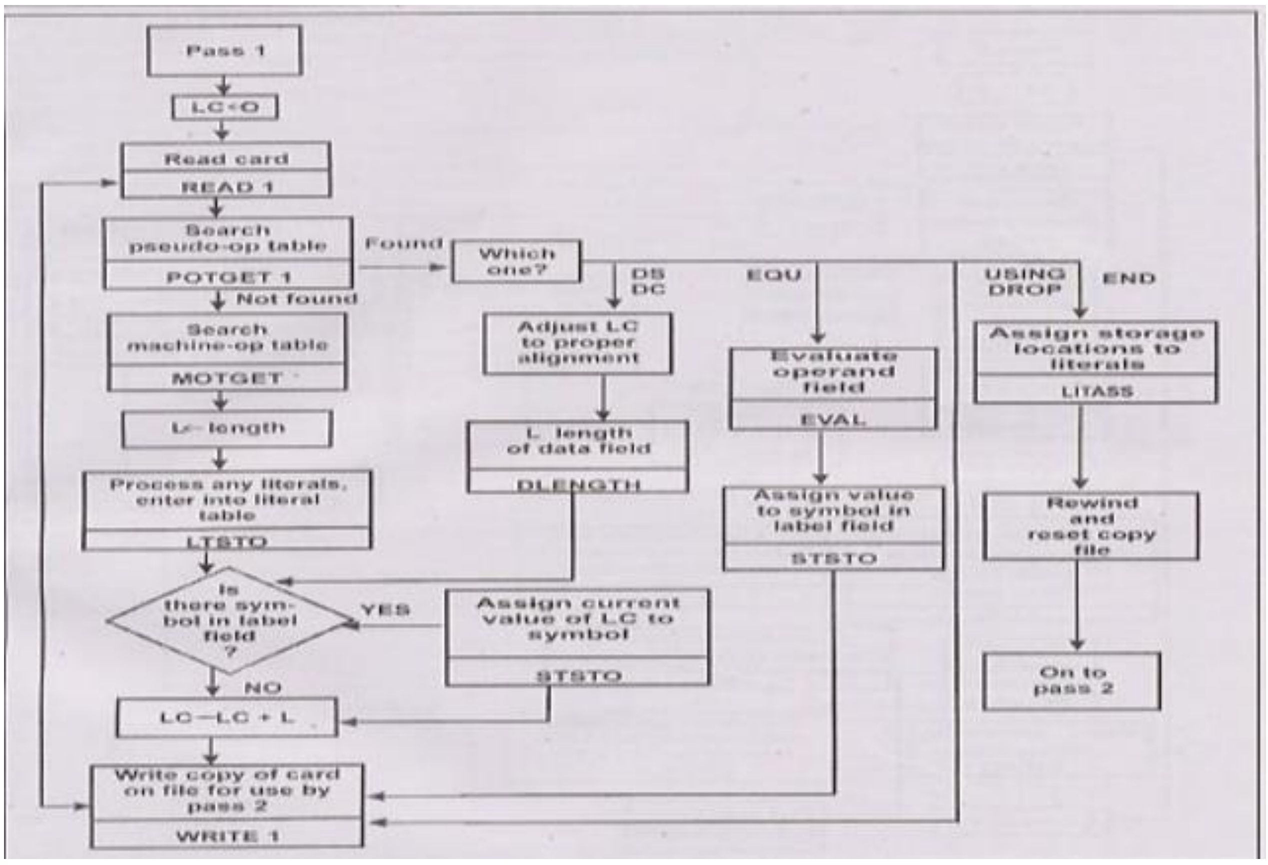


Two pass assembler (Two pass translation)



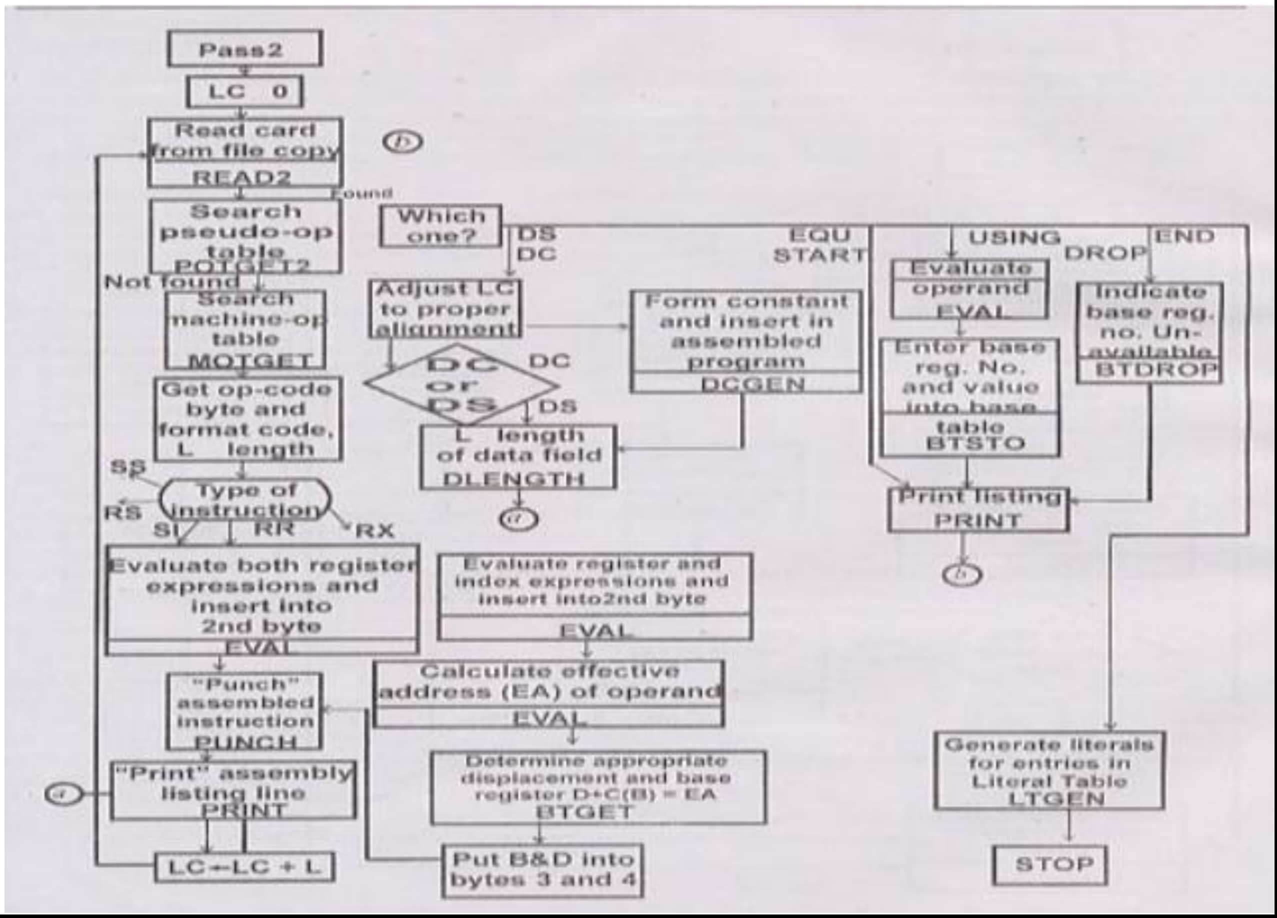
- The first pass performs analysis of the source program.
- The first pass performs Location Counter processing and records the addresses of symbols in the symbol table.
- It constructs intermediate representation of the source program.
- Intermediate representation consists of following two components:
  1. Intermediate code
  2. Data structures

Design diagrams (if any): Flow chart for Pass-1



Pass 1
LC←0
Read card
READ 1
Search pseudo-op table
POTGET 1
Found
Not found
Which one?
DS
DC
EQU
USING
DROP
END
Search machine-op table
MOTGET
L← length
Process any literals, enter into literal table
LTSTO
Adjust LC to proper alignment
L length of data field
DLENGTH
Evaluate operand field
EVAL
Assign value to symbol in label field
STSTO
Assign storage locations to literals
LITASS
Rewind and reset copy file
On to pass 2
Is there symbol in label field?
YES
NO
Assign current value of LC to symbol
STSTO
LC←LC + L
Write copy of card on file for use by pass 2
WRITE 1

Flow chart for Pass-2



Pass2
LC 0
Read card from file copy
READ2
Found
Search pseudo-op table
POTGET2
Not found
Which one?
DS
DC
EQU
START
USING
DROP
END
Search machine-op table
MOTGET
Adjust LC to proper alignment
Form constant and insert in assembled program
DCGEN
Evaluate operand
EVAL
Indicate base reg. no. Unavailable
BTDROP
Get op-code byte and format code, L length
DC or DS
DC
DS
L length of data field
DLENGTH
Enter base reg. No. and value into base table
BTSTO
Print listing
PRINT
SS
RS
Type of instruction
SI
RR
RX
Evaluate both register expressions and insert into 2nd byte
EVAL
Evaluate register and index expressions and insert into 2nd byte
EVAL
"Punch" assembled instruction
PUNCH
Calculate effective address (EA) of operand
EVAL
"Print" assembly listing line
PRINT
Determine appropriate displacement and base register D+C(B) = EA
BTGET
LC←LC + L
Put B&D into bytes 3 and 4
Generate literals for entries in Literal Table
LTGEN
STOP

Input for PassI of assembler:
1. Assembly language program in hypothetical language as per the Author Dhamdhere
2. OPTB
3. Condition code table
4. Register table

Input for PassII of assembler:
1. IC
2. SYMTAB
3. LITTAB
4. POOLTAB
5. OPTAB

Output of PassI of assembler:
1. IC
2. SYMTAB
3. LITTAB
4. POOLTAB

Output of PassII of assembler:
Machine code

Conclusion:
1. Input assembly language program is processed by applying PassI algorithm of assembler and intermediate data structures, SYMTAB, LITTAB, POOLTAB, IC, are generated.
2. The intermediate data structures generated in PassI of assembler are given as input to the PassIIof assembler, processed by applying PassII algorithm of assembler and machine code is generated

Frequently Asked Questions:
1. What is two pass assembler?
2. What is the significance of symbol table?
3. Explain the assembler directives EQU, ORIGIN.
4. Explain the assembler directives START, END, LTORG.
5. What is the use of POOLTAB and LITTAB?
6. How literals are handled in pass I?
7. What are the tasks done in Pass I?
8. How error handling is done in pass I?
9. Which variant is used in implementation? Why?
10. Which intermediate data structures are designed and implemented in PassI?
11. What is the format of a machine code generated in PassII?
12. What is forward reference? How it is resolved by assembler?
13. How error handling is done in pass II?
14. What is the difference between IS, DL and AD?
15. What are the tasks done in Pass II?