University Institute of Engineering

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Bachelor of Engineering

Subject Name: System Programming

Subject Code: CST-315

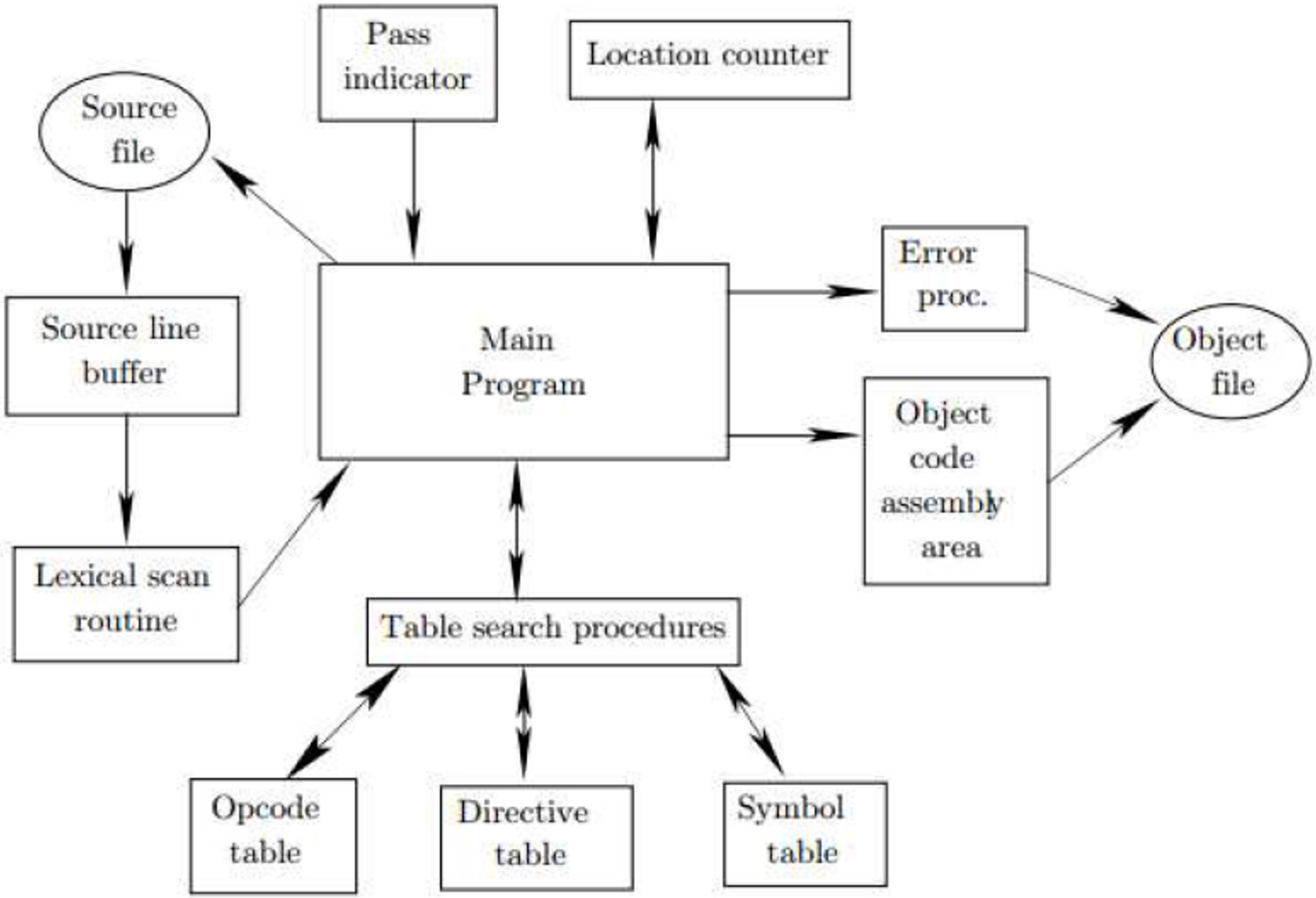
Chapter-1.2
Assemblers

Types of Assemblers

- Two-Pass Assemblers
- One-Pass Assemblers

Assembler

- The basic principles of assembler operation are simple, involving just one problem, that of unresolved references.
- This is a simple problem that has two simple solutions.
- The problem is important, however, since its two solutions introduce, in a natural way, the two main types of assemblers namely, the one-pass and the two-pass.



The main components of Assembler and operations

Assembler

Assembler divide these tasks in two passes:

- **Pass-1:**
  - Define symbols and literals and remember them in symbol table and literal table respectively.
  - Keep track of location counter
  - Process pseudo-operations
- **Pass-2:**
  - Generate object code by converting symbolic op-code into respective numeric op-code
  - Generate data for literals and look for values of symbols

One-pass assembler

- The operation of a one-pass assembler is different.
- As its name implies, this assembler reads the source file once.
- During that single pass, the assembler handles both label definitions and assembly.
- The only problem is future symbols.

Two pass Assembler

- Such an assembler performs two passes over the source file.
- In the first pass it reads the entire source file, looking only for label definitions.
- All labels are collected, assigned values, and placed in the symbol table in this pass.
- No instructions are assembled and, at the end of the pass, the symbol table should contain all the labels defined in the program.
- In the second pass, the instructions are again read and are assembled, using the symbol table

Example

- Firstly, We will take a small assembly language program to understand the working in their respective passes.

- Assembly language statement format:

```
[Label] [Opcode] [operand]
```

Example: M ADD R1, ='3'

where, M - Label; ADD - symbolic opcode;

R1 - symbolic register operand; (='3') - Literal

Assembly Program:

Label	Op-code	operand	LC value(Location counter)
JOHN	START	200	
	MOVER	R1, ='3'	200
	MOVEM	R1, X	201
L1	MOVER	R2, ='2'	202
	LTORG		203
X	DS	1	204
	END		205

Example

- **START:** This instruction starts the execution of program from location 200 and label with START provides name for the program.(JOHN is name for program)
- **MOVER:** It moves the content of literal(=’3’) into register operand R1.
- **MOVEM:** It moves the content of register into memory operand(X).
- **MOVER:** It again moves the content of literal(=’2’) into register operand R2 and its label is specified as L1.
- **LTORG:** It assigns address to literals(current LC value).
- **DS(Data Space):** It assigns a data space of 1 to Symbol X.
- **END:** It finishes the program execution.

Example

- **Working of Pass-1:** Define Symbol and literal table with their addresses.
  Note: Literal address is specified by LTORG or END.
- **Step-1: START 200** (here no symbol or literal is found so both table would be empty)
- **Step-2: MOVER R1, =’3’ 200** (=’3’ is a literal so literal table is made)

Example

- **Step-3: MOVEM R1, X 201**
  X is a symbol referred prior to its declaration so it is stored in symbol table with blank address field.
- **Step-4: L1 MOVER R2, =’2’ 202**
  L1 is a label and =’2’ is a literal so store them in respective tables

Symbol	Address
X	- - -
L1	202

Literal	Address
='3'	- - -
='2'	- - -

Example

- **Step-5: LTORG 203**
  Assign address to first literal specified by LC value, i.e., 203
- **Step-6: X DS 1 204**
  It is a data declaration statement i.e X is assigned data space of 1. But X is a symbol which was referred earlier in step 3 and defined in step 6.
- This condition is called Forward Reference Problem where variable is referred prior to its declaration and can be solved by back-patching. So now assembler will assign X the address specified by LC value of current step.

Example

- **Step-7: END 205**

  Program finishes execution and remaining literal will get address specified by LC value of END instruction. Here is the complete symbol and literal table made by pass 1 of assembler.

Symbol	Address
X	204
L1	202

Literal	Address
='3'	203
='2'	205

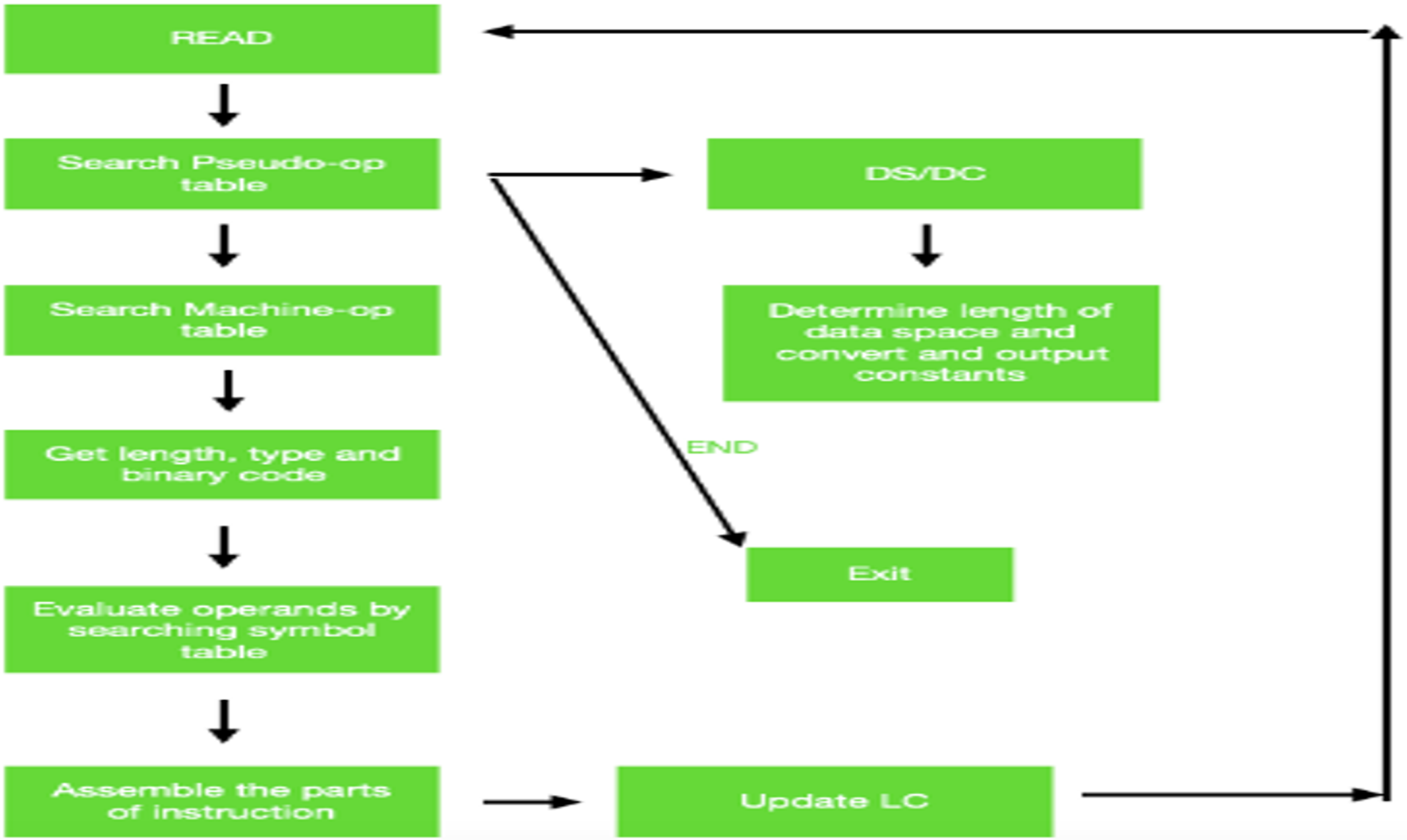
Working of Pass-2:

- Pass-2 of assembler generates machine code by converting symbolic machine-opcodes into their respective bit configuration(machine understandable form).
- It stores all machine-opcodes in MOT table (op-code table) with symbolic code, their length and their bit configuration.
- It will also process pseudo-ops and will store them in POT table(pseudo-op table).

Working of Pass-2:

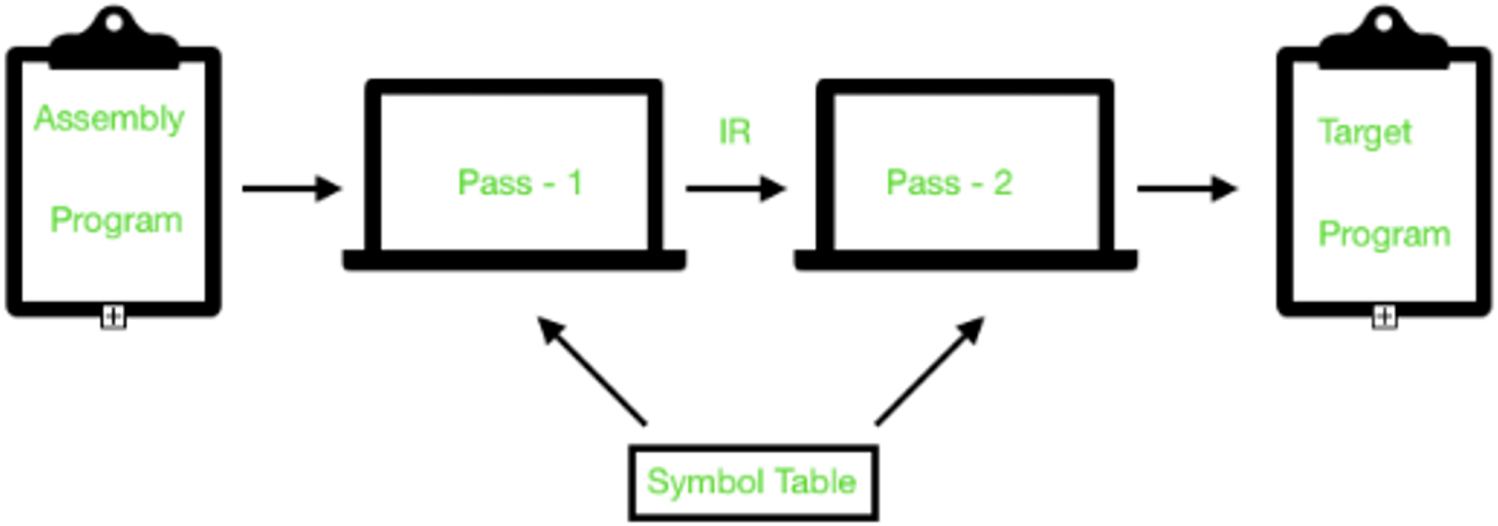
- Various Data bases required by pass-2:
- 1. MOT table(machine opcode table)
- 2. POT table(pseudo opcode table)
- 3. Base table(storing value of base register)
- 4. LC (location counter)

Pass Description



Assembler Working Diagram

- As a whole assembler works as:
-



References

- [PDF] Systems Programming and Operating Systems by Dhamdhere - Free Download PDF (dlscrib.com)
- [PDF] Principles of Compiler Design By Alfred V. Aho & J.D.Ullman Free Download – Learnengineering.in

THANK YOU