

# Assembler & Assembly Language Programming)

## Systems Programming Lab (CS-244)



**Dr. Samit Bhattacharya**  
**CSE, IIT Guwahati**

# Things to know

- Compiler
- Interpreter
- Assembler

# COMPILER VS INTERPRETER VS ASSEMBLER

Software that converts programs written in a high level language into machine language

Converts the whole high level language program to machine language at a time

Used by C, C++

Software that translates a high level language program into machine language

Converts the high level language program to machine language line by line

Used by Ruby, Perl, Python, PHP

Software that converts programs written in assembly language into machine language

Converts assembly language program to machine language

Used by assembly language

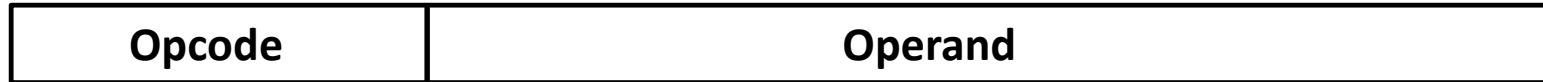
# Assembler Design

- Single pass
  - Entire conversion done with a single scan (reading) of the code
  - Not easy
- Two pass
  - Two readings of the code for conversion
  - Easier to do

# Things to know

- Instruction format
- Addressing models

# Instruction Format



e.g. ADD, MUL ...

**How to get the operands (arguments) – Addressing modes**

## Addressing Modes

- The way in which the operand of an instruction is specified
- Specifies a rule for interpreting or modifying the address field of the instruction before the operand is actually executed

### Example (8086 processor)

- 1) Immediate addressing mode  
MOV AX, 2000
- 2) Register addressing mode  
MOV AX, BX
- 3) Direct addressing mode  
MOV AX, [0500]

# Two Pass Assembler

## Pass 1 (Define symbol and literals)

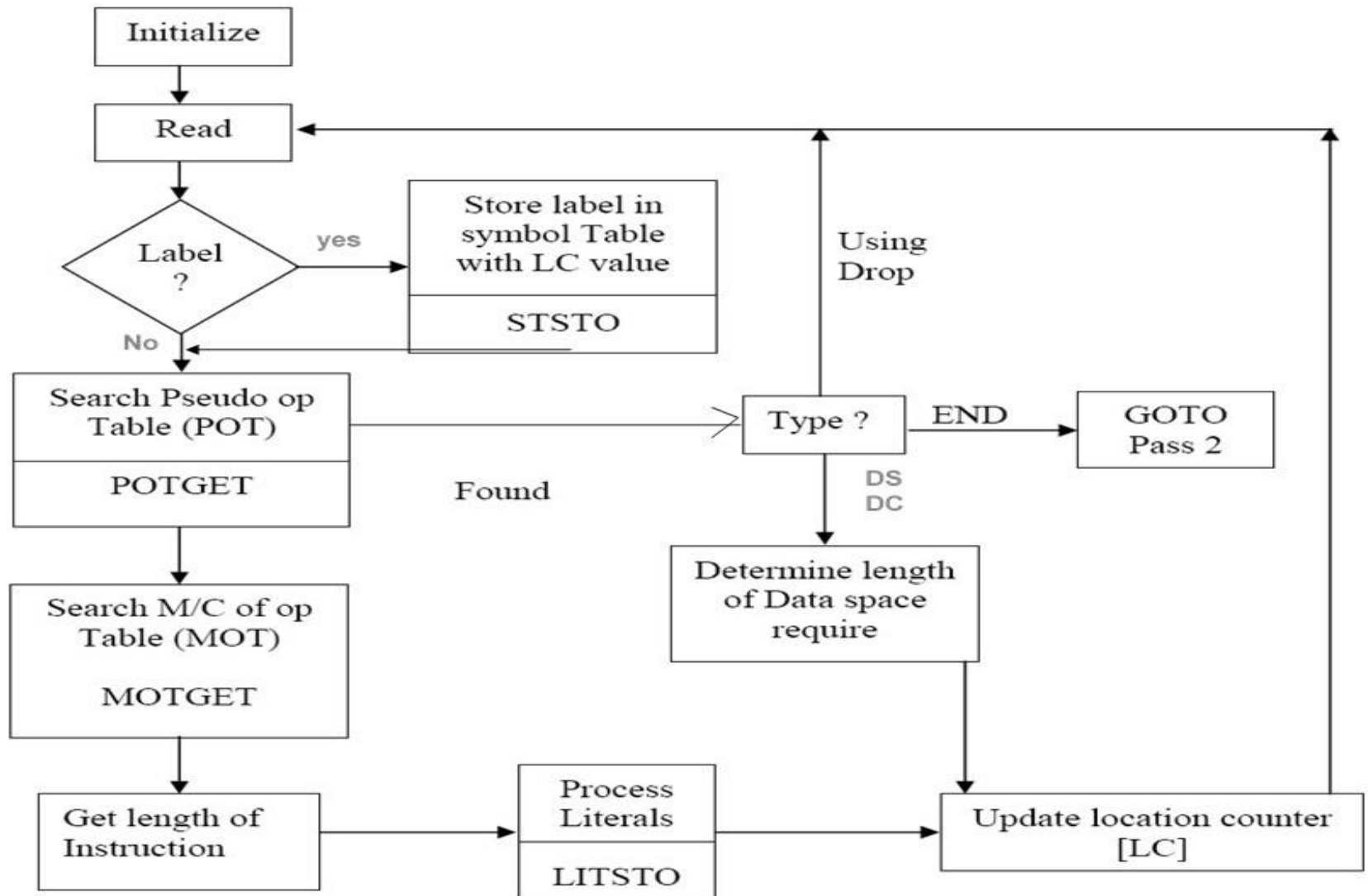
- Determine length of machine instructions (MOTGET1)
- Keep track of location counter (LC)
- Remember values of symbol until pass 2 (STSTO)
- Process some pseudo-ops ...EQU, DS
- Remember literals (LITSTO)



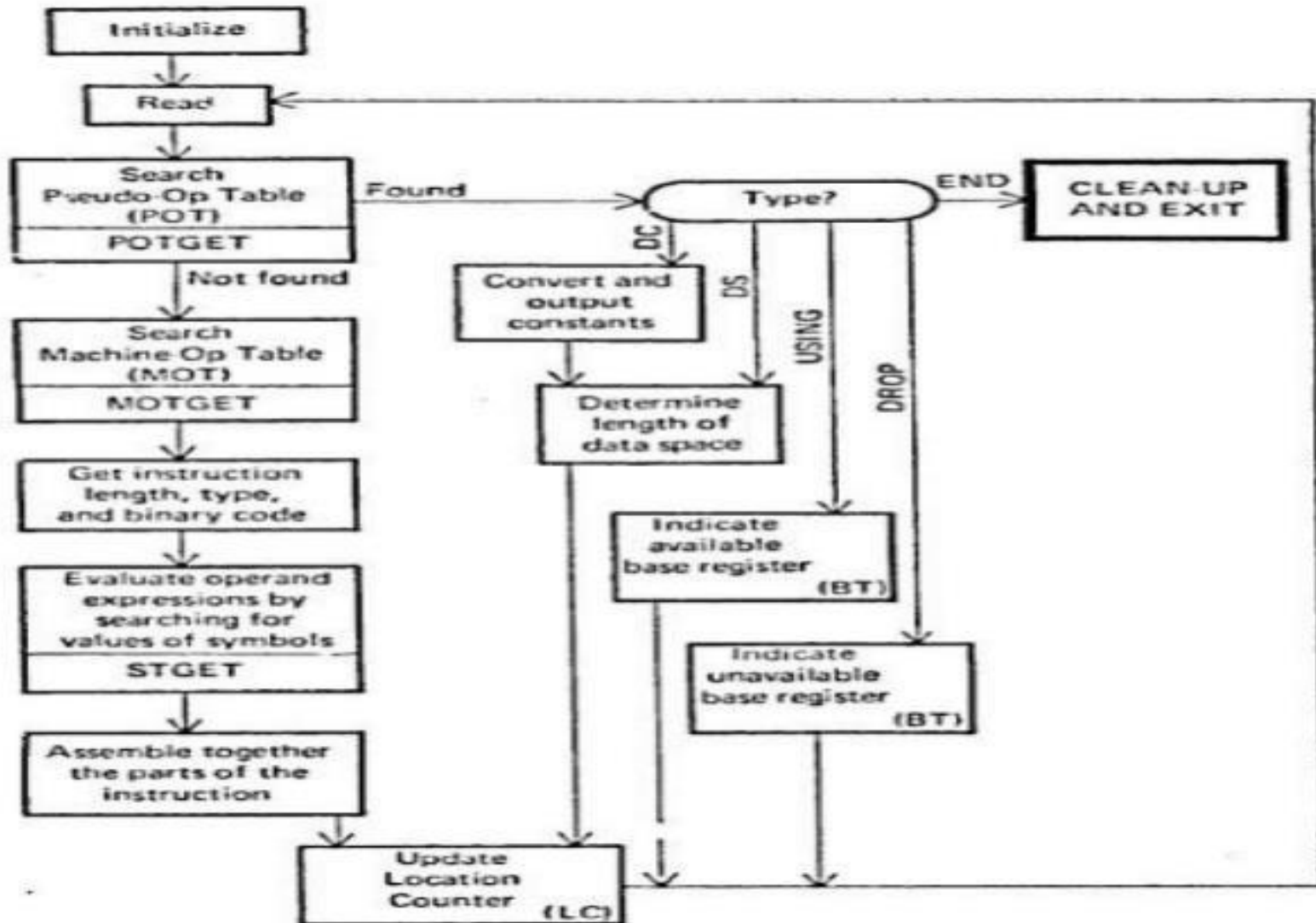
## **Pass 2 (Generate object program)**

- Look up value of symbols (STGET)
- Generate Instructions (MOTGET2)
- Generate data for DS, DC and literals
- Process pseudo-ops (POTGET2)

## Pass 1 Flowchart:



## Pass 2 Flowchart:



- We will be dealing with 8086 microprocessor for this lab
- Other tutorials will cover the simpler instructions (and assignments) !!

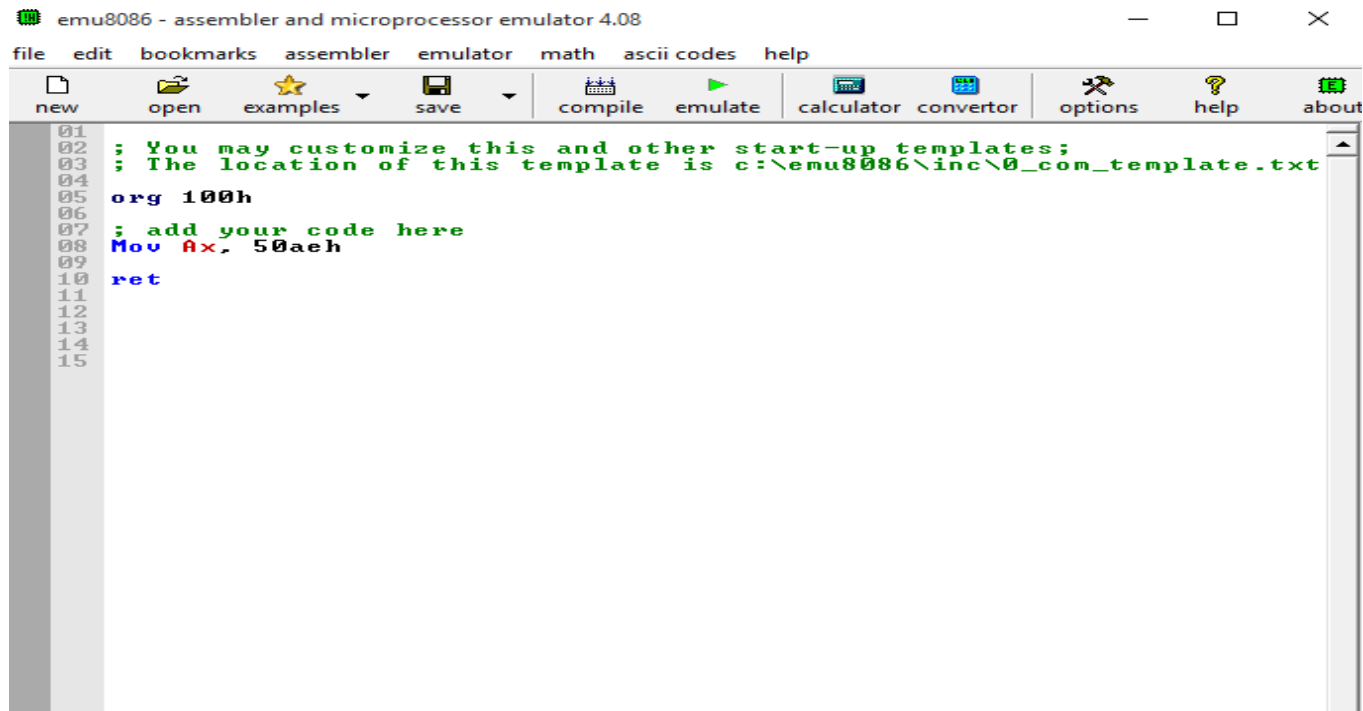
- Book: J J Donovan, Systems Programming (very old book but useful)
- Book: L L Beck & D Manjula, Systems Software.

On 8086 instruction set:

- [http://www.gabrielececchetti.it/Teaching/CalcolatoriElettronici/Docs/i8086\\_instruction\\_set.pdf](http://www.gabrielececchetti.it/Teaching/CalcolatoriElettronici/Docs/i8086_instruction_set.pdf)
- [https://www.tutorialspoint.com/microprocessor/microprocessor\\_8086\\_instruction\\_sets.htm](https://www.tutorialspoint.com/microprocessor/microprocessor_8086_instruction_sets.htm)

## Emulator:

- We will be using **emu8086** Emulator in lab sessions for assembly language programming.



The screenshot shows the emu8086 - assembler and microprocessor emulator 4.08 window. The interface includes a menu bar (file, edit, bookmarks, assembler, emulator, math, ascii codes, help) and a toolbar with icons for new, open, examples, save, compile, emulate, calculator, convertor, options, help, and about. The main text area displays assembly code with line numbers 01 through 15 on the left. The code is as follows:

```
01 ; You may customize this and other start-up templates;  
02 ; The location of this template is c:\emu8086\inc\0_com_template.txt  
03  
04  
05 org 100h  
06  
07 ; add your code here  
08 Mov Ax, 50aeh  
09  
10 ret  
11  
12  
13  
14  
15
```

