University Institute of Engineering

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Bachelor of Engineering

Subject Name: System Programming

Subject Code: CST-315

Assemblers

Chapter-1.2
Assemblers

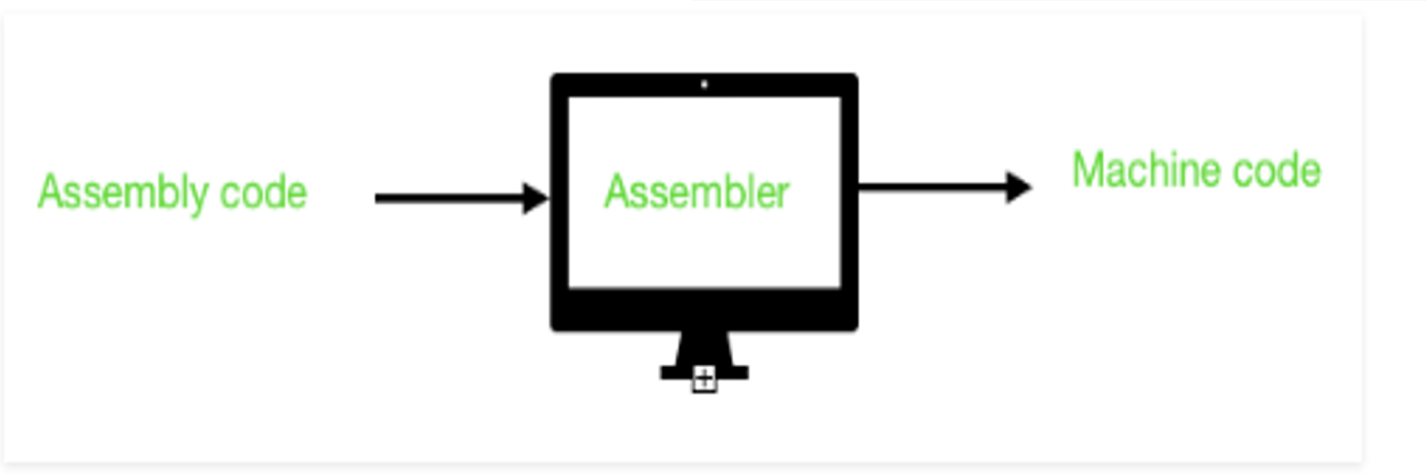
- Elements of Assembly Language Programming
- Design of the Assembler
- Assembler Design Criteria

Assembler

- An assembler is a program that converts assembly language into machine code.
- It takes the basic commands and operations from assembly code and converts them into binary code that can be recognized by a specific type of processor.
- Assemblers are similar to compilers in that they produce executable code.
- However, assemblers are more simplistic since they only convert low-level code (assembly language) to machine code.
- Since each assembly language is designed for a specific processor, assembling a program is performed using a simple one-to-one mapping from assembly code to machine code.

Assembler

- It generates instructions by evaluating the mnemonics (symbols) in operation field and find the value of symbol and literals to produce machine code.
- Now, if assembler do all this work in one scan then it is called single pass assembler, otherwise if it does in multiple scans then called multiple pass assembler.



Elements of Assembly Language

Machine language is very difficult to program in directly.
Understanding the meanings of the numerical-coded instructions is tedious for humans.
For example, the instruction that says to add the EAX and EBX registers together and store the result back into EAX is encoded by the following hexcodes: 03 C3

Assembly language is basically like any other language, which means that it has its words, rules and syntax. The basic elements of assembly language are:
Labels;
Orders;
Directives; and
Comments.

Elements of Assembly Language

Syntax of Assembly language

- When writing a program in assembly language it is necessary to observe specific rules in order to enable the process of compiling into executable “HEX-code” to run without errors. These compulsory rules are called syntax and there are only several of them: Every program line may consist of a maximum of 255 characters;
- Every program line to be compiled, must start with a symbol, label, mnemonics or directive;
- Text following the mark “;” in a program line represents a comment ignored (not compiled) by the assembler; and
- All the elements of one program line (labels, instructions etc.) must be separated by at least one space character. For the sake of better clearness, a push button TAB on a keyboard is commonly used instead of it, so that it is easy to delimit columns with labels, directives etc. in a program.

Elements of Assembly Language

Numbers

If octal number system, otherwise considered as obsolete, is disregarded, assembly language allows numbers to be used in one out of three number systems:

- Decimal Numbers
- Hexadecimal Numbers
- Binary Numbers

Elements of Assembly Language

- **Operators**

Some of the assembly-used commands use logical and mathematical expressions instead of symbols having specific values.

Addition	Subtraction	Multiplication
Division	&Bitwise logical AND	Bitwise logical OR
>>Shift right	<<Shift left	%Remainder
Bitwise logical AND	NOT	Bitwise logical XOR

Elements of Assembly Language

Symbols

- Every register, constant, address or subroutine can be assigned a specific symbol in assembly language, which considerably facilitates the process of writing a program.
- For the purpose of writing symbols in assembly language, all letters from alphabet (A-Z, a-z), decimal numbers (0-9) and two special characters ("?" and "_") can be used. Assembly language is not case sensitive.
- For example, the following symbols will be considered identical:Serial_Port_Buffer SERIAL_PORT_BUFFER

Elements of Assembly Language

Symbols (Contd..)

- In order to distinguish symbols from constants (numbers), every symbol starts with a letter or one of two special characters (? or _).
- The symbol may consist of maximum of 255 characters, but only first 32 are taken into account.
- Some of the symbols cannot be used when writing a program in assembly language because they are already part of instructions or assembly directives.

Elements of Assembly Language

Label

- A label is a special type of symbols used to represent a textual version of an address in ROM or RAM memory. They are always placed at the beginning of a program line. It is very complicated to call a subroutine or execute some of the jump or branch instructions without them. They are easily used:A symbol (label) with some easily recognizable name should be written at the beginning of a program line from which a subroutine starts or where jump should be executed.
- It is sufficient to enter the name of label instead of address in the form of 16-bit number in instructions calling a subroutine or jump.
- During the process of compiling, the assembler automatically replaces such symbols with appropriate addresses.

Elements of Assembly Language

- **Directives**

  Unlike instructions being compiled and written to chip program memory, directives are commands of assembly language itself and have no influence on the operation of the microcontroller. Some of them are obligatory part of every program while some are used only to facilitate or speed up the operation. Directives are written in the column reserved for instructions. There is a rule allowing only one directive per program line.

Eg. EQU directive and SET directive

- The EQU directive is used to replace a number by a symbol.
- The SET directive is also used to replace a number by a symbol

Design of Assembler

General design procedure

- specify the problem

1.specify the data structure

2.define format of data structure

3.specify algorithm

4.look for modularity

repeat 1 ~ 5 on modules

Design of Assembler

- **Objectives**
- 1. Generate instructions Evaluate the mnemonic in the operation field to produce its machine code

1. Find the value of each symbol, process literals

2. Process pseudo ops

- 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

Design of Assembler

Pass 1:define symbols & literals

- Determine length of each instruction Keep track of LC Remember values of symbols until pass
- Process some pseudo ops Remember literals

Pass 2: generate object program Look up values of symbols

- Generate instructions
- Generate data Process pseudo ops

Design of Assembler

Pass 1 data bases

- Input source program A
- LC
  -LC
- A MOT (Machine Operation Table)
- A POT (Pseudo operation Table)
- A ST (Symbol Table)
- A LT (Literal Table)
- A copy of the input to be used by pass 2

Pass 2 databases

-Copy of source program input to pass 1

-MOT

-POT

-ST

-BT (Base table)

Assembler Design Criteria

Design Specification of an assembler

Four step approach to develop a design specification

1) Identify the information necessary to perform a task
2) Design a suitable data structure to record the information
3) Determine the processing necessary to obtain and maintain the information.
4) Determine the processing necessary to perform the task

References

- https://techterms.com/definition/assembler#:~:text=An%20assembler%20is%20a%20program%20that%20converts%20assembly%20language%20into%20machine%20code.&text=Most%20programs%20are%20written%20in,perform%20in%20a%20specific%20way.
- https://www.geeksforgeeks.org/introduction-of-assembler/
- https://www.techopedia.com/definition/3971/assembler
- https://techterms.com/definition/assembly_language#:~:text=An%20assembly%20language%20is%20a,machine%20code%20using%20an%20assembler.

THANK YOU