

2CS701

Compiler Construction

Prof Monika Shah

Nirma University

Introduction

2CS701 Compiler Construction

Prof Monika Shah

Nirma University

Outline

- Course overview
- Objective & Course Outcomes
- Mapping of COs with PO and PSO
- What is Compiler and Interpreter
- Why study Compiler construction?
- Teaching Scheme and Examination Scheme

Course Overview

2CS701 Compiler Construction

- Introduction :

The course will discuss how a program written in H.L.L.(higher level language) is systematically translated into L.L.L(low level language)

It also help you to understand various programming constructs and their semantics
- Prerequisites:
 - C/C++ programming skill,
 - Data structure
- Course website : <https://lms.nirmauni.ac.in/course/view.php?id=4849>

for : Syllabus, LP/LOP, Handouts, References, Assignment, Forum
- Textbook: “*Compilers: Principles, Techniques, and Tools*” by Aho, Sethi, and Ullman

Objective & Course Learning Outcomes

Objective

To make student understand programming language constructs, and give them hands-on experience with crafting a simple compiler using modern software tools.

Course Learning Outcomes

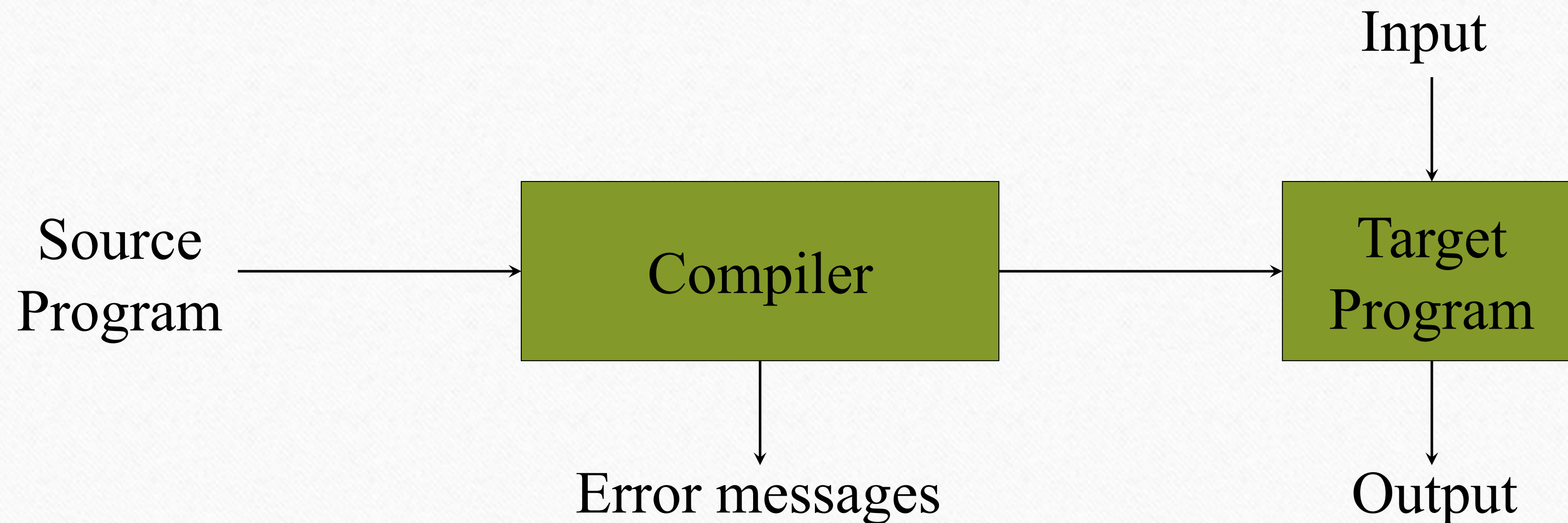
- Summarize the functionalities of various phases of compiler
- Apply language theory concepts to various phases of compiler design
- Identify appropriate optimization technique for compilation process
- Develop a miniature compiler using appropriate compiler design tool

Mapping of COs with POs and PSO

CO	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12	PSO1	PSO2
2CS701.1	3					2				2		1	3	2
2CS701.2		1	2	1		1			1	1		1	2	1
2CS701.3			2	2	1				1	1		1	2	1
2CS701.4			3	3	2				1	1		1	3	1

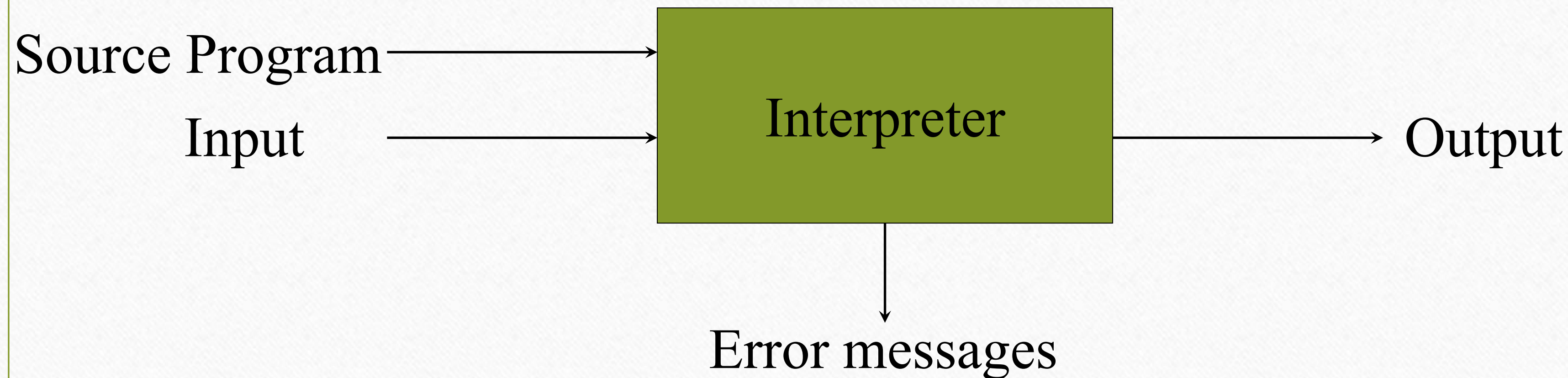
Compilers and Interpreters

- “*Compilation*”
 - Translation of a program written in a source language into a semantically equivalent program written in a target language



Compilers and Interpreters

- “*Interpreter*”
 - Translation of an instruction of a program written in a source language into a semantically equivalent instruction written in a target (machine) language
 - Execute translated line with given input



Why study compiler construction?

Importance of Compilers

Robert Hundt : Leading Compiler engineer at Google

- “At the scale of datacenters, every **single performance percent matters!** Just take a look at Google’s (and other’s) publicly available numbers on expenditure on data centers. We are talking about billions of dollars from more program features or improved utilizations.”
- “In order to deploy software at Google scale, engineers will touch a good dozen of programming and configuration languages, all with their own interesting optimization problems (from a compiler writers’ point of view). Fundamental knowledge in language, compiler, and runtime implementation will help make better engineering decisions, everywhere.”

Course Assessment Scheme Scheme

Course Assessment Scheme

Assessment scheme	CE				LPW		SEE
Component weightage	0.4				0.2		0.4
	Quiz –I 0.15	Quiz -II 0.15	Sessional Exam 0.4	Term Paper 0.30	Continuous Evaluation 75 marks	Viva Voce 25 marks	100 marks

Why study compiler construction?

- Compilers are important
 - Hundreds of programming language
 - Responsible for system performance
- We can become better programmer once we know compilation technology
 - We can write efficient code
 - Compilers usually have different options
- Very good Job Opportunities in field of compiler
 - “we do not need many compiler guys, but those that we need, we need them badly”
- Hundreds of programming languages and hundreds of different hardware
- Apply compiler design techniques to build other programming tools
 - Example : text editors, Form validation, Information retrieval system, Query Processing system, NLP systems

Why study compiler construction? contd

- Ties lots of things you learn together
 - Theory
 - Data structures
 - Software Engineering (modularization)
 - Utilization of software tools


Compiler construction shows us a microcosmic view of computer science.


<i>artificial intelligence</i>	greedy algorithms learning algorithms
<i>algorithms</i>	graph algorithms union-find network flows dynamic programming
<i>theory</i>	<i>dfa's</i> for scanning parser generators lattice theory for analysis
<i>systems</i>	allocation and naming locality synchronization
<i>architecture</i>	pipeline management memory hierarchy management instruction set use

Inside a compiler, all these things come together.

As a result, compiler construction is challenging and fun.

Why study compiler construction?

 **Compilers Help us to be Better Programmers**

 Compilers usually have different optimization options. The iconic `gcc -O1`, for instance, runs these optimizations.

What does each of these things do?

We can even run these optimizations individually:

```
$> gcc -fdefer-pop -o test test.c
```

We can enable a few of them, and disable others, e.g.:

```
$> gcc -O1 -fno-defer-pop -o test test.c
```

- `fauto-inc-dec`
- `fcompare-elim`
- `fcprop-registers`
- `fdce`
- `fdefer-pop`
- `fdelayed-branch`
- `fdse`
- `fguess-branch-probability`
- `fif-conversion2`
- `fif-conversion`
- `fipa-pure-const`
- `fipa-profile`
- `fipa-reference`
- `fmerge-constants`
- `fsplit-wide-types`
- `ftree-bit-ccp`
- `ftree-builtin-call-dce`
- `ftree-ccp`
- `ftree-ch`
- `ftree-copyrename`
- `ftree-dce`
- `ftree-dominator-opts`
- `ftree-dse`
- `ftree-forwprop`
- `ftree-fre`
- `ftree-phi-prop`
- `ftree-slr`
- `ftree-sra`
- `ftree-pta`
- `funit-at-a-time`

Why study compiler construction?

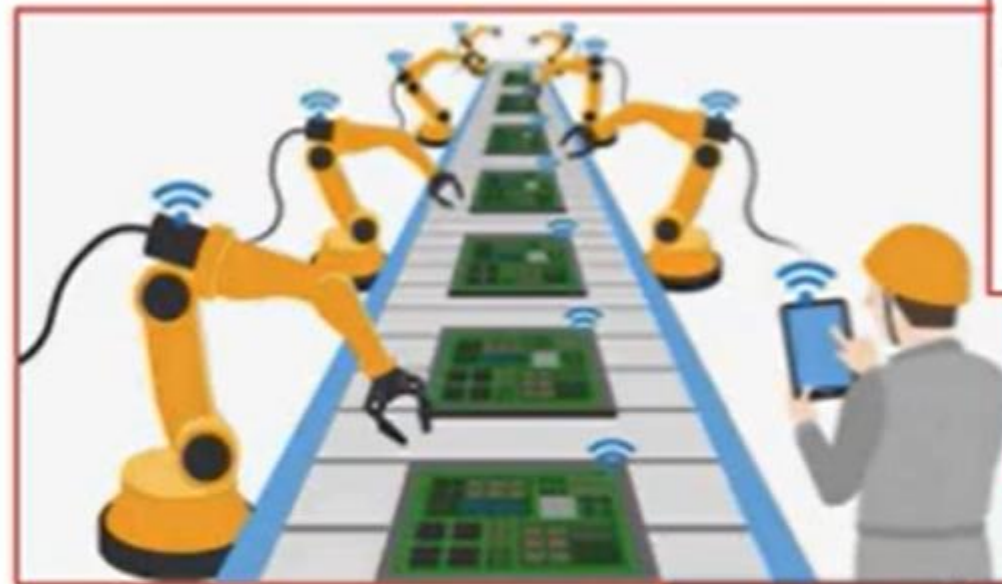
Lots of Job Opportunities

- Expert compiler writers find jobs in many large companies: Oracle, NVIDIA, Microsoft, Intel, Google, IBM, AMD, Mozilla, Apple etc.
- These Jobs usually asks for C/C++ skills, good knowledge of computer science
- Hardware companies need compilation technology
 - Intel : icc
 - Apple : LLVM
 - NVIDIA : nvcc
 - Microsoft : visual studio, .NET
 - STMicroelectronics : open 64
 - Google : ART, V8
 - Mozilla : JaegerMonkey, IonMonkey, TraceMonkey

Why study compiler construction?

Compilers are everywhere

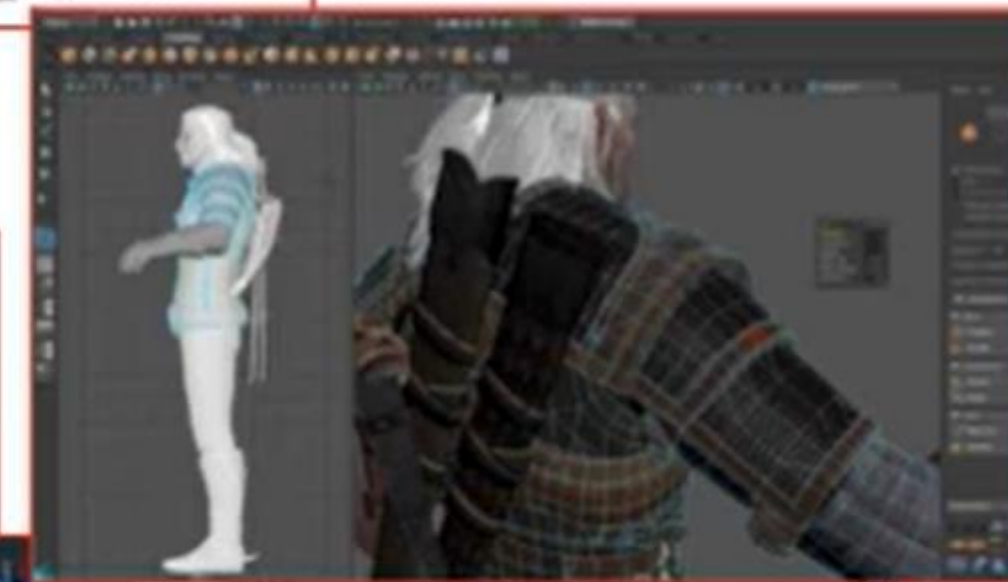
Embedded System



Car Industries



Real Game Industries



Macros, Commands



Data Centres



Examples of Specilized Companies

There are companies that sell mostly compilation technology, which can be used in several ways, e.g., to create a new back-end, to parse big data, to analyze programs for security vulnerabilities, etc.

CodePlay is a company that develops compilers for Systems on a Chip devices used in the automotive industry.



Coverity is a software vendor which develops static code analysis tools, for C, C++, Java and C#.



The Associated Compiler Experts (**ACE**) have developed compilers for over 100 industrial systems, ranging from 8-bit microcontrollers to CISC, RISC, DSP and 256-bit VLIW processor architectures.



PathScale Inc. is a company that develops a highly optimizing compiler for the x86-64 microprocessor architectures.



The Portland Group, Inc. (**PGI**) is a company that produces a set of commercially available Fortran, C and C++ compilers.



Green Hills produces compilers for C, C++, Fortran, and Ada that target the ARC, ARM, Blackfin and ColdFire platforms



Why study compiler construction? contd...

- Today's approach :
 - Programming abstraction + handcoded libraries
- Tool for evaluating computer performance
 - Deliver performance on different architecture i.e. VLIW, HPC, GPU,
 - Machine's performance measured on compiled code
- Example: Machine learning
 - Why is machine learning hard ?
 - Can we have friendly high level language?
 - Bridge gap between Neural network and HPC,GPU ...
 - Machine learning for Computer optimization