Introduction

Rupesh Nasre.

CS3300 Compiler Design
IIT Madras
Aug 2015

Languages

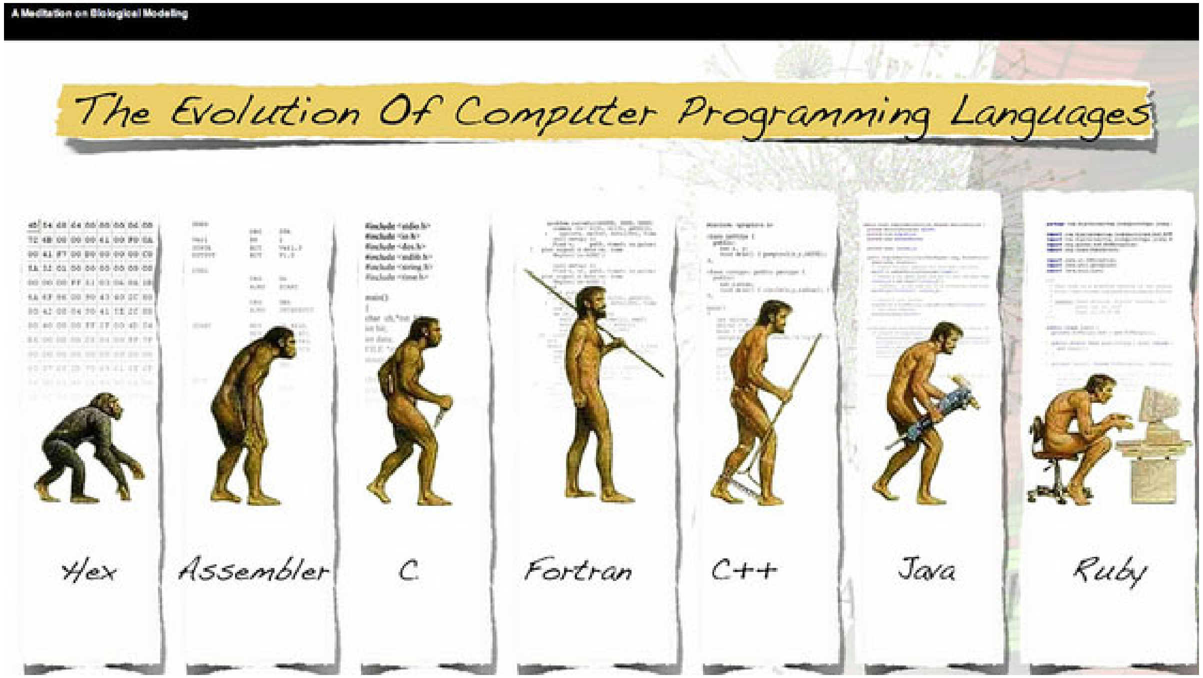
Wall of Love, Paris
Source: google images

Languages

- Why do we need languages?
  - Humans communicate
    - sign language, body language, braille
  - Birds communicate
    - mark territories, attract for mating, warn danger
  - Animals communicate
    - mark territories, convey need, preparation for attack
  - Aliens?

Programming Languages

- Why do we need programming languages?
- And why so many?
  - What is your first language?
  - Tamil. Yours?
  - C.


A Meditation on Biological Modeling
The Evolution Of Computer Programming Languages
Hex
Assembler
C
Fortran
C++
Java
Ruby

Source: google images

Programming Languages

- There are some special purpose languages
  - HTML for webpages
  - LaTeX for document formatting
  - ps for postscript files; sql, VHDL
  - Shell scripts, awk, grep, sed
  - Makefile has a language; smtp
  - How about google search?
    - filetype:pdf, link:www.cse.iitm.ac.in
  - Gmail: in:unread, in:starred
  - vi: :se ai, :wq, :se ft=c
  - What about ls -l, ls -Ri, ls --color, ls -1 dir1 dir2 ?

Language is for Communication

- Using mobile buttons
- Using ipad touch
- Using a calculator
- Using a fan switch
- Using a remote for projector / laser

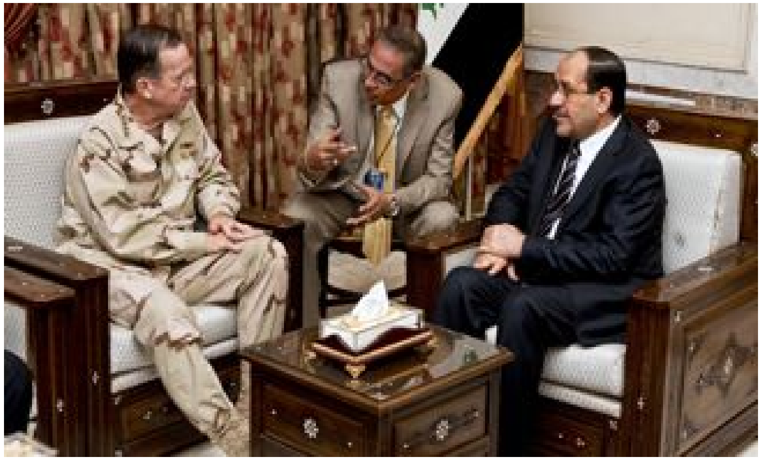
... some of these are not programmable today.

They have a limited abstraction.

We will work with programming languages.

Compiler

- When do we need a compiler?
  - நான் தமிழ் தெரியுமா
  - मुझे हिंदी आता है
  - నేను తెలుగు తెలుసు
  - I know English



Source: google images

Jobs of a Compiler

- Translate: input language, output language
- Maintain correctness
  - पिताजी अजमेर गए।
  - Father died today.
- Be efficient
  - Why are you laughing?
  - I understood yesterday's joke.
- Generate a good language
  - I got books but more than that I got your letter.
  - मैं किताबें, लेकिन मैं अपने पत्र मिला है कि अधिक से अधिक

Good Language

I got books but more than that I got your letter.

मैं किताबें, लेकिन मैं अपने पत्र मिला है कि अधिक से अधिक मिला है।

I have books, but I got your letter got more than that.

मैं किताबें हैं, लेकिन मैं अपने पत्र है कि अधिक से अधिक मिला।

I have books, but that's more than I got your letter.

मैं किताबें हैं, लेकिन लगता है कि मैं अपने पत्र मिला है की तुलना में अधिक है।

I have books, but I have received your letter is more than.

मैं किताबें हैं, लेकिन मैं अपने पत्र की तुलना में अधिक है प्राप्त हुआ है।

Compilers work with Strings

- Characters, words / tokens, sentences, programs
- Fun with strings
  - quick brown fox jumps over the lazy dog
  - stewardesses
  - typewriter
  - skepticisms
  - quine

Programs as Data

```
char*f="char*f=%c%s%c;main(){printf(f,34,f,34,10);}%c";main(){printf(f,34,f,34,10);}
```

Why should we Design a language?

- Language matters!
  - A: Would you accept a gamble that offers a 10% chance to win $95 and a 90% chance to lose $5?
  - B: Would you pay $5 to participate in a lottery that offers a 10% chance to win $100 and a 90% chance to win nothing.
- Outcomes of a treatment for lung cancer. Two descriptions were:
  - C: The one-month survival rate is 90%.
  - D: There are 10% deaths in the first month.
- B fetched many more positives. 84% physicians chose option C.

Source: Thinking Fast and Slow, Daniel Kahneman

Why should we Design a Language?

Asian disease problem.

An asian disease is expected to kill 600 people. Two alternative programs are proposed.

- If program A is adopted, 200 people will be saved.
- If program B is adopted, there is a one-third probability that 600 people will be saved and a two-thirds probability that no people will be saved.

A substantial majority of respondents choose program A, due to preference to a sure option than gamble.

Now change the description.

- If program A' is adopted, 400 people will die.
- If program B' is adopted, there is a one-third probability that nobody will die and a two-thirds probability that 600 people will die.

A large majority chose B'.

Source: Thinking Fast and Slow, Daniel Kahneman

Why should we Design a Language?

The KMPL Fallacy

Virat switches from a car of 12 km average (per liter) to a 14 km.

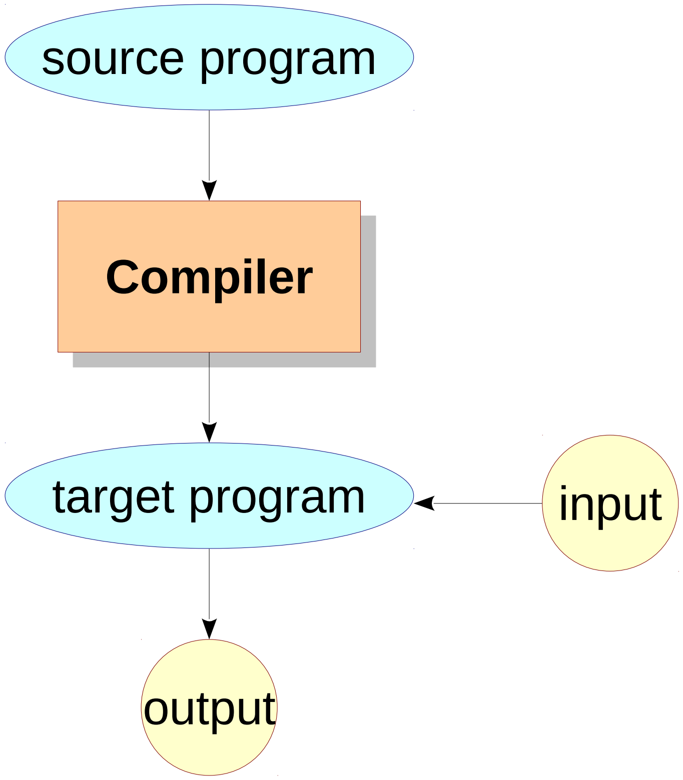
Dhoni switches from a car of 30 km average to a 40 km.

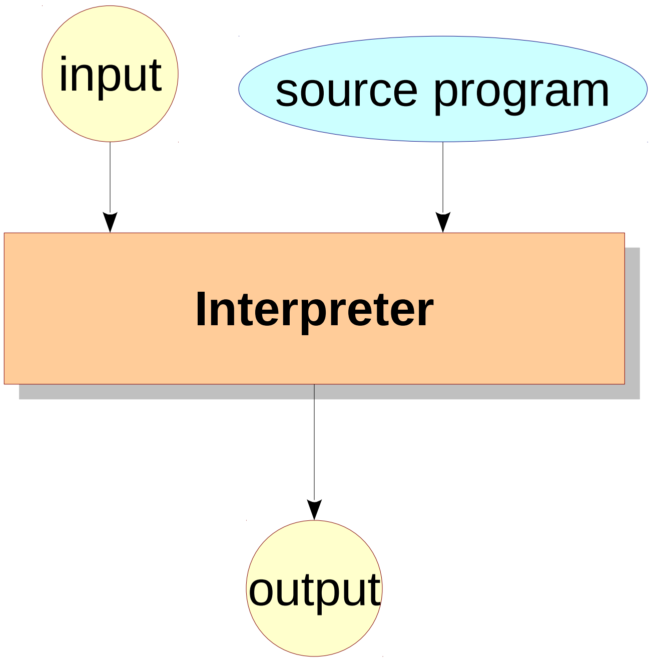
If both drivers travel the same distance over a year, who saves more fuel by switching?

One may "feel" that Dhoni saves more, but calculations say the opposite. Say they both travel 10K km in a year. Virat reduces his consumption from 833 liters to 714 liters, saving 119 liters. Dhoni's consmption reduces from 333 to 250 saving 83 liters.

Instead of the average in km/liter, the fuel efficiency should be in liter/km or liter/100km.

Source: Thinking Fast and Slow, Daniel Kahneman





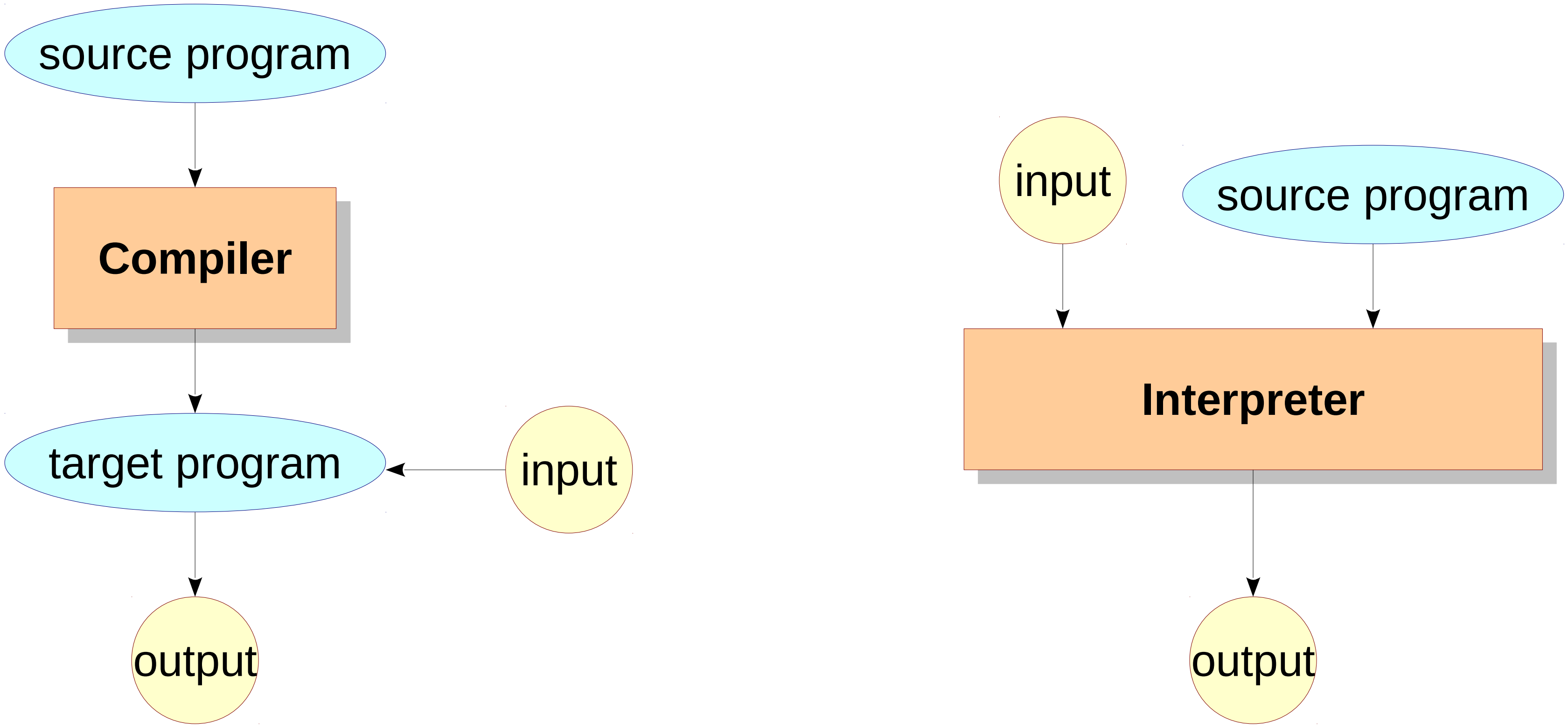
- What does this mean?
  - You may be able to do the following with interpreters.

```
$x = 0; $y = 0;
echo "Enter a variable name: ";
$line = fgets(STDIN);
$line = trim($line);
${$line} = 20;
echo "x=$x, y=$y\n";
```

How about C?

```
void main() {
        int x = 0, y = 0;
#include "/dev/stdin"
         = 10;
         printf("x = %d, y = %d\n", x, y);
}
```

Everything is fair in love, war and C.

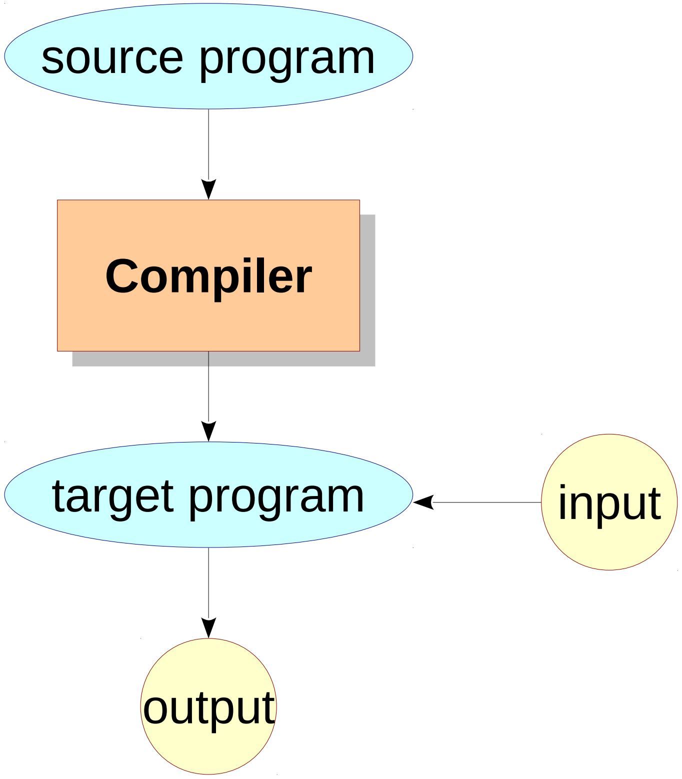


- What does this mean?
  - You may be able to do the following with compilers.

```
x += 2;
x += 2;
--x;
x += 5;
++x;
x += 9
```

is equivalent to

```
x += 18;
```



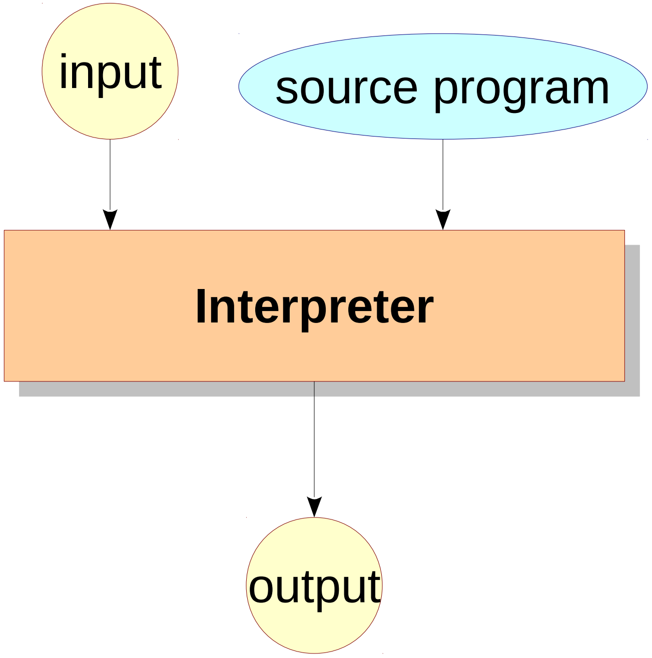
source program

Compiler

target program

input

output

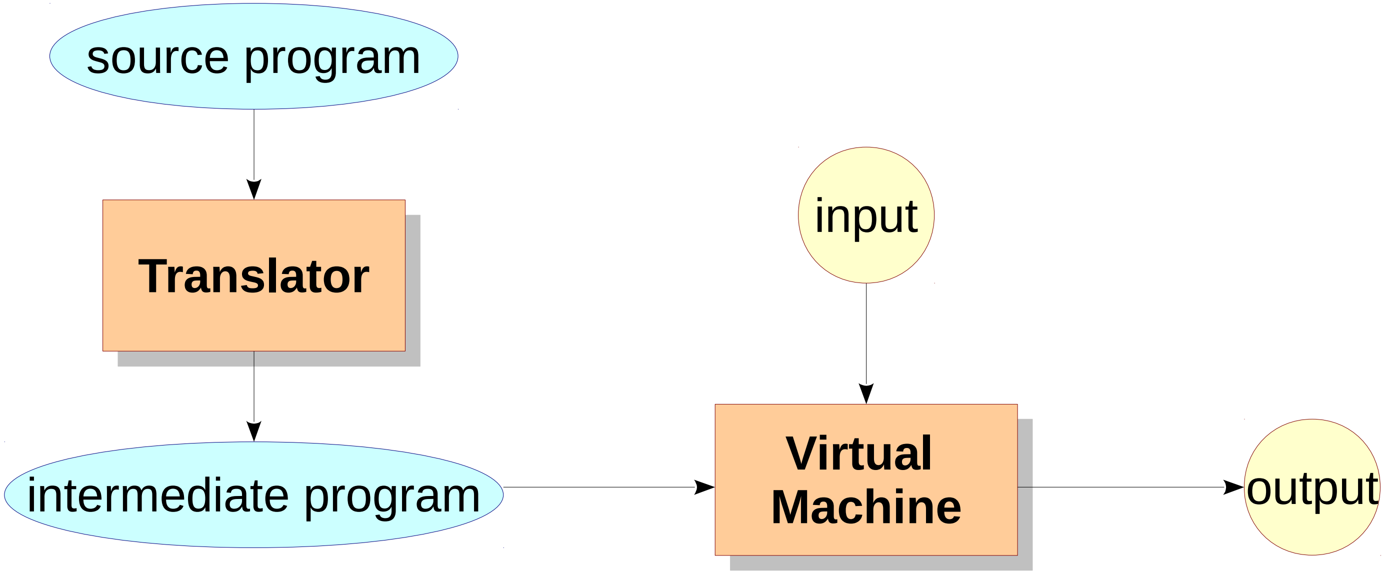


input

source program

Interpreter

output



source program

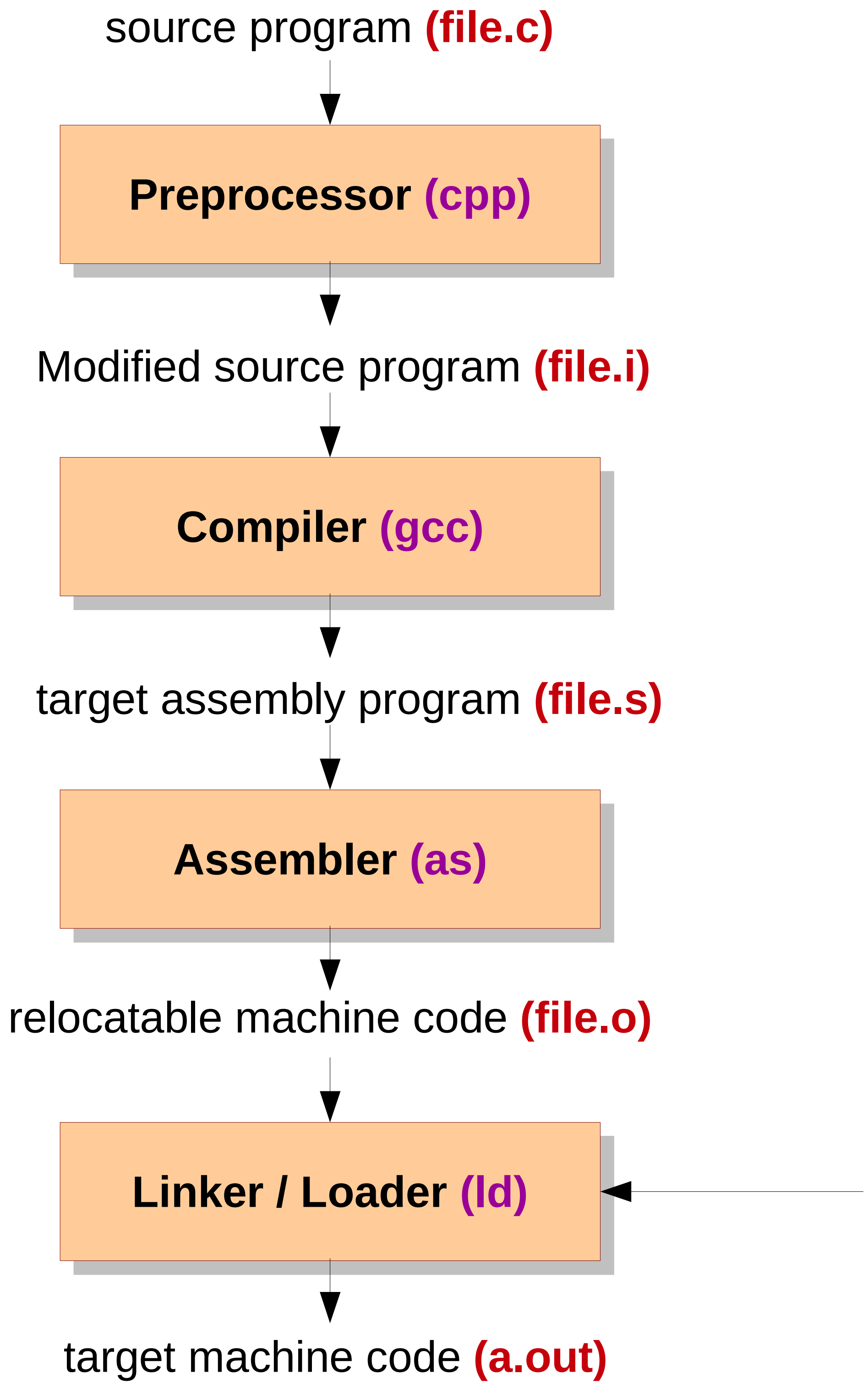
Translator

intermediate program

input

Virtual Machine

output



source program **(file.c)**

Preprocessor (cpp)

Modified source program **(file.i)**

Compiler (gcc)

target assembly program **(file.s)**

Assembler (as)

relocatable machine code **(file.o)**

Linker / Loader (ld)

target machine code **(a.out)**

library files,
relocatable object files

- cpp file.c >file.i
- gcc -S file.i
- as file.s -o file.o
- ld -o a.out file.o ...*libraries*...

Try the following:

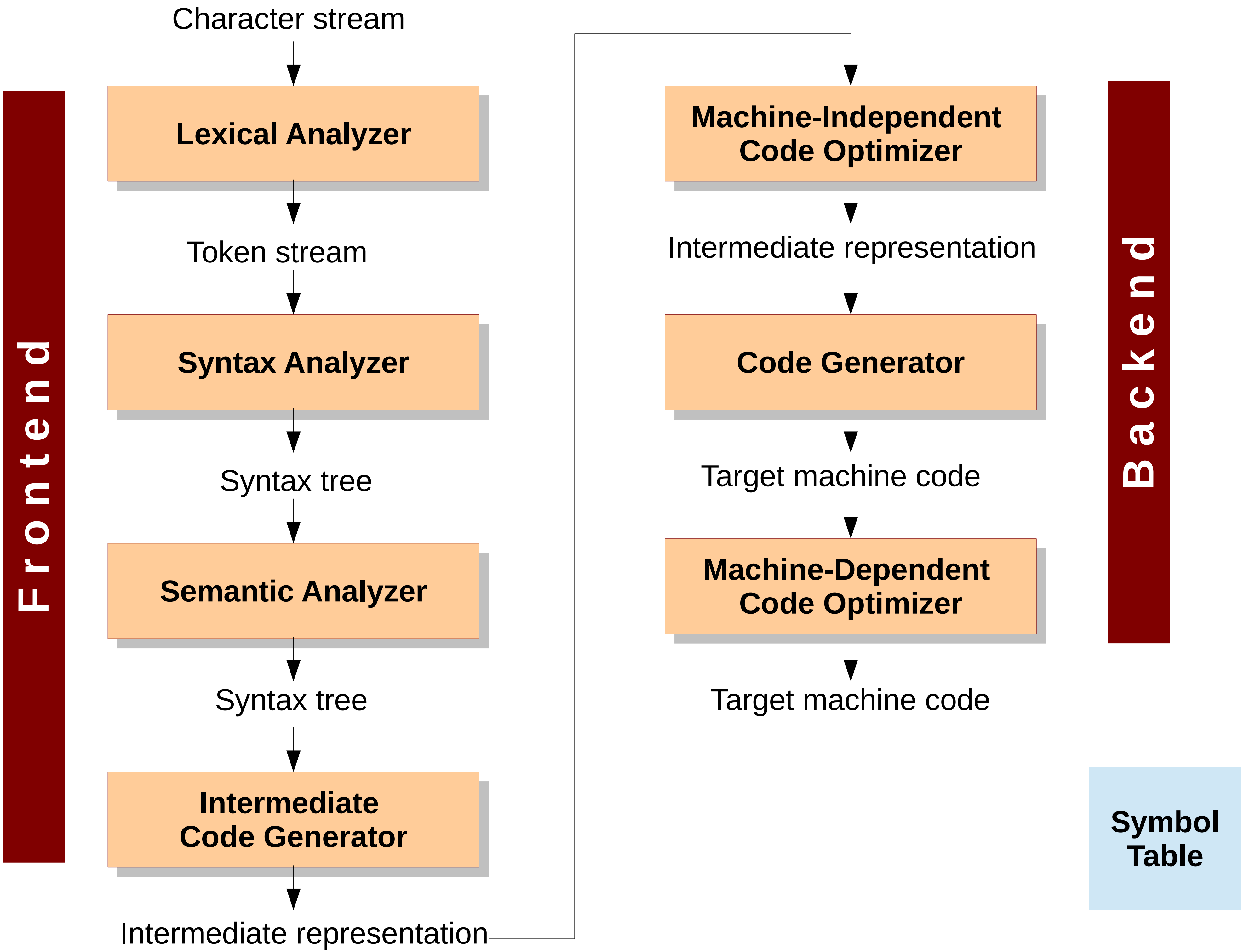
- gcc **-v** file.c
- gcc **-save-temps** file.c

Language Processors

- **Preprocessor**: collects source programs, expands macros.
- **Compiler**: Translates source program into a low-level assembly.
- **Assembler**: Produces (relocatable) machine code.
- **Linker**: Resolves external references **statically**, combines multiple machine codes.
- **Loader**: Loads executable codes into memory, resolves external references **dynamically**.

Homework

- Exercises 1.1.1-5 from ALSU.



Frontend

Character stream

Lexical Analyzer

Token stream

Syntax Analyzer

Syntax tree

Semantic Analyzer

Syntax tree

Intermediate Code Generator

Intermediate representation

Backend

Machine-Independent Code Optimizer

Intermediate representation

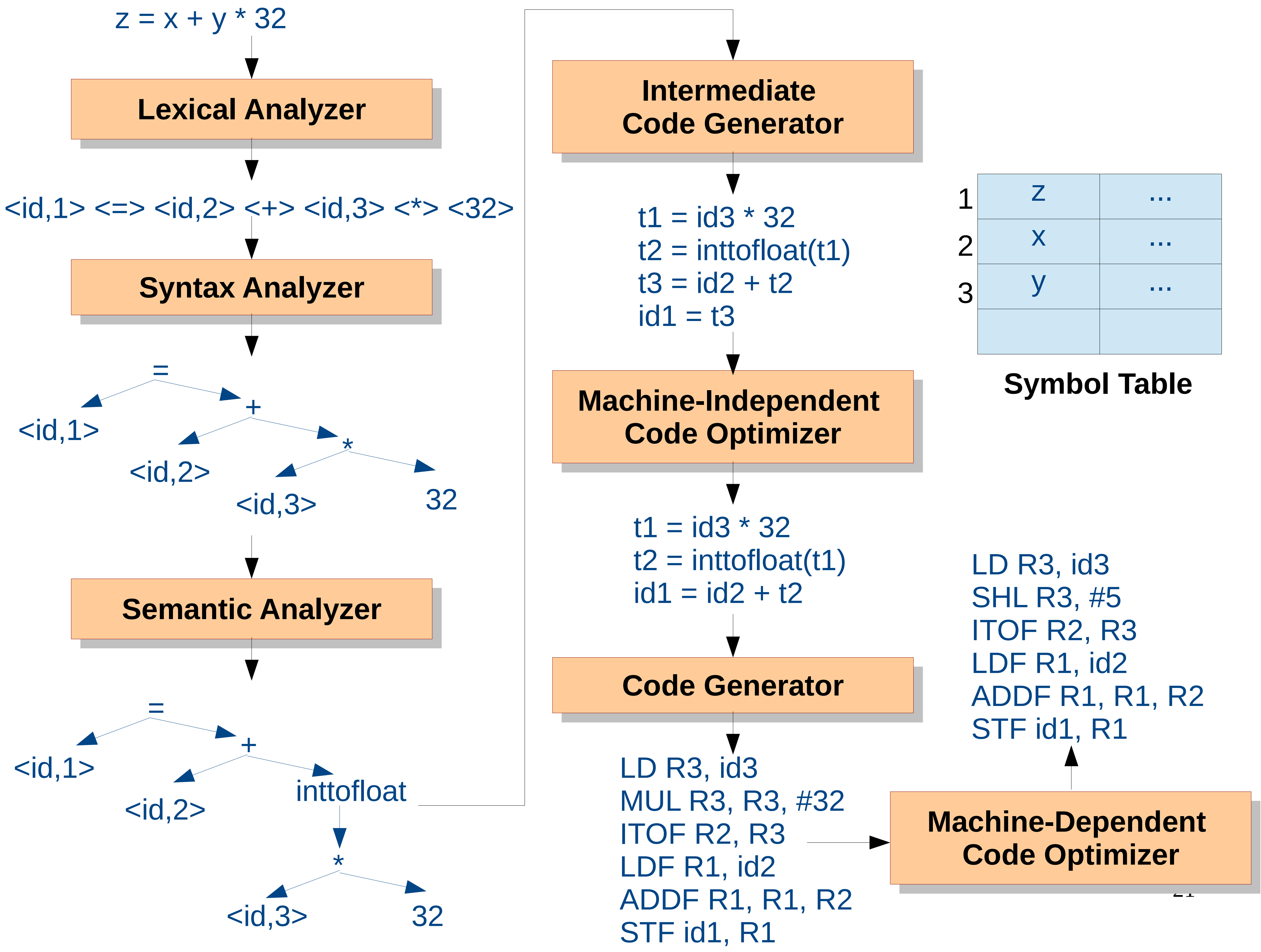
Code Generator

Target machine code

Machine-Dependent Code Optimizer

Target machine code

Symbol Table



z = x + y * 32

Lexical Analyzer

<id,1> <=> <id,2> <+> <id,3> <*> <32>

Syntax Analyzer

Semantic Analyzer

Intermediate Code Generator

```
t1 = id3 * 32
t2 = inttofloat(t1)
t3 = id2 + t2
id1 = t3
```

Machine-Independent Code Optimizer

```
t1 = id3 * 32
t2 = inttofloat(t1)
id1 = id2 + t2
```

Code Generator

```
LD R3, id3
MUL R3, R3, #32
ITOF R2, R3
LDF R1, id2
ADDF R1, R1, R2
STF id1, R1
```

Machine-Dependent Code Optimizer

```
LD R3, id3
SHL R3, #5
ITOF R2, R3
LDF R1, id2
ADDF R1, R1, R2
STF id1, R1
```

1	z	...
2	x	...
3	y	...

Symbol Table

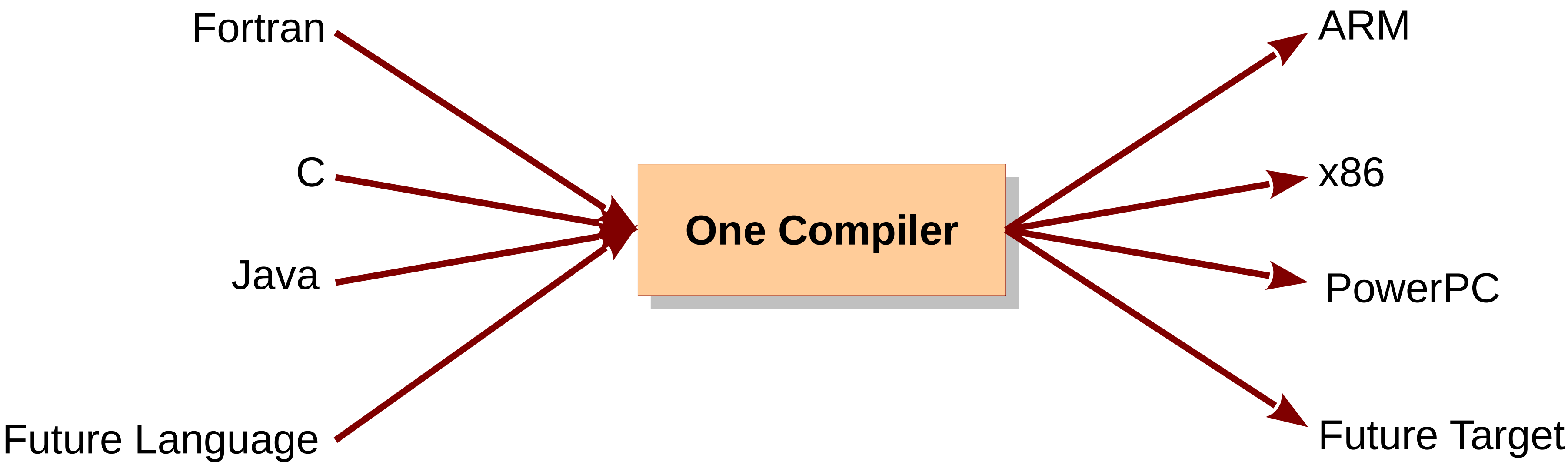
Symbol Table

- Record variable names
- Collect their attributes
  - Type (int, char)
  - Storage requirement ([30], 1)
  - Type modifiers (const, static)
  - Scope (global, static)
  - Information about arguments (for functions)
- Efficient insertion, search (sometimes deletion)
  - C: **int** x, y, z;
  - Pascal: **var** x, y, z: **integer**;
  - Javascript:

1	z	...
2	x	...
3	y	...

Symbol Table

Ideal World



Fortran

C

Java

Future Language

One Compiler

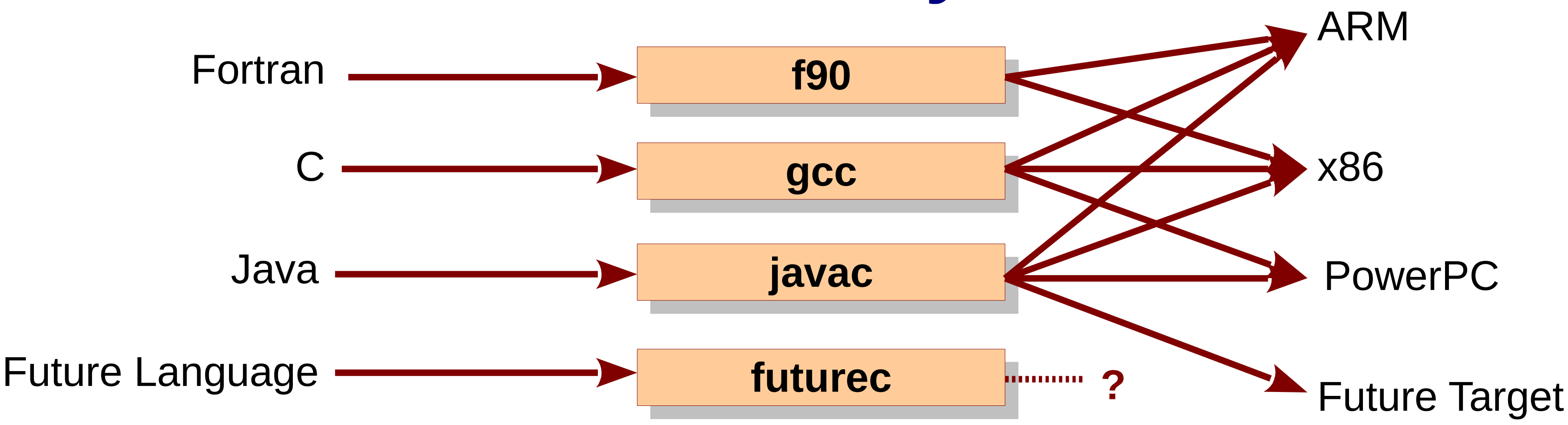
ARM

x86

PowerPC

Future Target

Reality



Fortran

f90

ARM

C

gcc

x86

Java

javac

PowerPC

Future Language

futurec

?

Future Target

Reality getting worse

- I don't have a compiler for this platform.
- My program compiles with an older version of gcc.
- My program compiles with the new version, but does not run on this new platform.
- My program compiles with an older gcc if you disable optimizations.
- My program compiles if you have llvm 3.4, clang 3.5, gcc 4.7.1 on x86_64 with lonestar 1.2 or above on Ubuntu 12 or below.

Evolution of Programming Languages

- First electronic computers in 1940s.
- Programmed in machine language (0 and 1).
  - Move data from one location to another.
  - Add the content of two registers.
  - Compare two values
  - ...
- Slow, TeDius, and ErorrP run.

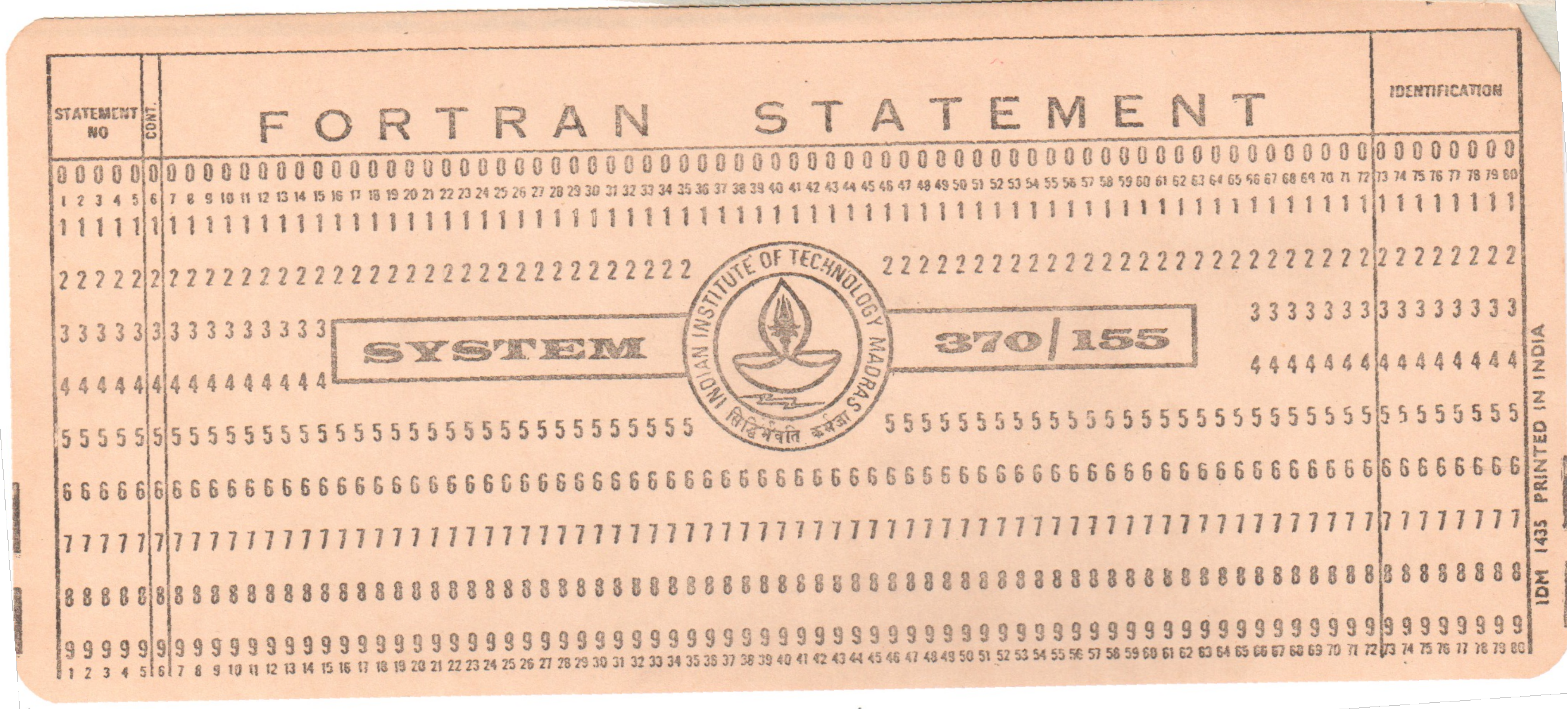
Maggie and Buildings

Punched Tape

Punched Card

Image source: wikipedia

Punched Tape



Punched Card

Courtesy: Deepak Khemani

Evolution of Programming Languages

- Assembly languages in early 1950s.
  - Initially, only mnemonics for machine instructions
  - Later, support for macros
- High-level languages in late 1950s.
  - **Fortran** for scientific computing
  - **Cobol** for data processing
  - **Lisp** for symbolic computation
  - These were so successful that they are still in use.

PL Classification

- Thousands of languages
  - Need to be categorized
- Based on paradigm
  - Imperative (c, c++, java), declarative (lisp, prolog)
- Based on generation *(think of generation gap)*
  - First (machine), second (assembly), third (fortran, cobol, lisp, c), fourth (sql, ps), fifth (prolog)
- Others
  - OO (c++, c#, Ruby), scripting (awk, js, php, python, ruby)

Compiler Writing

- is challenging.
- A compiler is a large program.
- A compiler must translate correctly potentially infinite set of programs that could be written in the source language.
- The problem of generating the optimal target code from a source program is undecidable.
  - Heuristics and Trade-offs.
- **Compilers is an area where Theory meets Practice.**

Static versus Dynamic

- Time
- Compilation
- Optimization
- Analysis
- Type
- Linking
- Scoping

Static versus Dynamic

- Time: compilation versus execution, preprocessor versus compilation
- Compilation: gcc versus jit
- Optimization: without and with input
- Analysis: without and with environment
- Type:
  - strongly typed versus scripting languages
  - inheritance
- Linking: .a versus .so
- Scoping

Static versus Dynamic

- Time
- Compilation
- Optimization
- Analysis
- Type
- Linking
- Scoping

```
int i = 1;
void f() {
    printf(“%d”, i);
}
void main() {
    int i = 2;
    f();
}
```

Static	Dynamic
1	2

Where do we use dynamic scoping?

Classwork

- Find the output of the program under static and dynamic scoping.

```
int a = 1, b = 2, y = 3;
void gun(int x, int b) {
  printf("%d %d\n", x, b);
}
void fun(int x, int y) {
  printf("%d %d\n", x, y);
  gun(a, y);
}
void main() {
  int a = 3;
  {
    int b = 4;
    fun(a, b);
  }
  gun(a, b);
  fun(a, b);
}
```

Parameter Passing

- Call by value
  - This happens in C
- Call by reference
  - Supported in C++, aliasing
- Call by name
  - Macros
- Call by value-result
  - Supported in ADA

```
int i = 1;
int *ip = &i;
void f(int x) {
    int y;
    x = 3;
    ip = &y;
    x = i+x+2;
}
void main() {
    f(*ip);
    printf("%d", i);
}
```

Call by value: 1
Call by reference: 8
Call by name: 3
Call by value-result: 6

Classwork

- Create an example that does not use pointers which produces different output under the four parameter passing schemes.

Reordering Transformation

- When can a compiler reorder instructions?

```
int f(int &a, int &b) {
    a = 4;
    c = b;
}
```

```
int f(int &a, int &b) {
    c = b;
    a = 4;
}
```

Such a transformation requires that

- a and b are not referring to the same memory location (RAW).
- a and c are not referring to the same memory location (WAW).

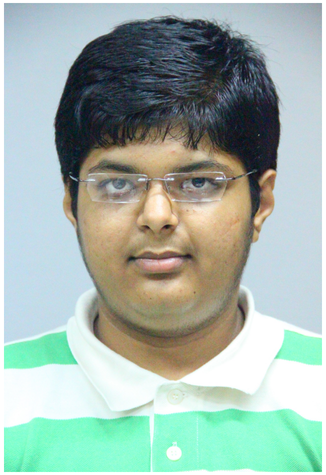
Aliasing

- In the example, a and b may be aliases, if the function call is f(x, x);

```
int f(int &a, int &b) {
    a = 4;
    c = b;
}
```

The student whose parents stay at ...

The one whose roll number is CS13B036



The one who wears spects, has cgpa of ..., and ...

The one who stays in hostel ..., is healthy, and ...

- Can you have aliasing in C?
- If & operator is disallowed, can there be aliasing?

Homework:

- Find out what **restrict** keyword does in gcc.
- Exercises 1.6.1 – 1.6.4 from ALSU.