

Introduction

Rupesh Nasre.

CS3300 Compiler Design
IIT Madras
July 2024

Languages



Languages

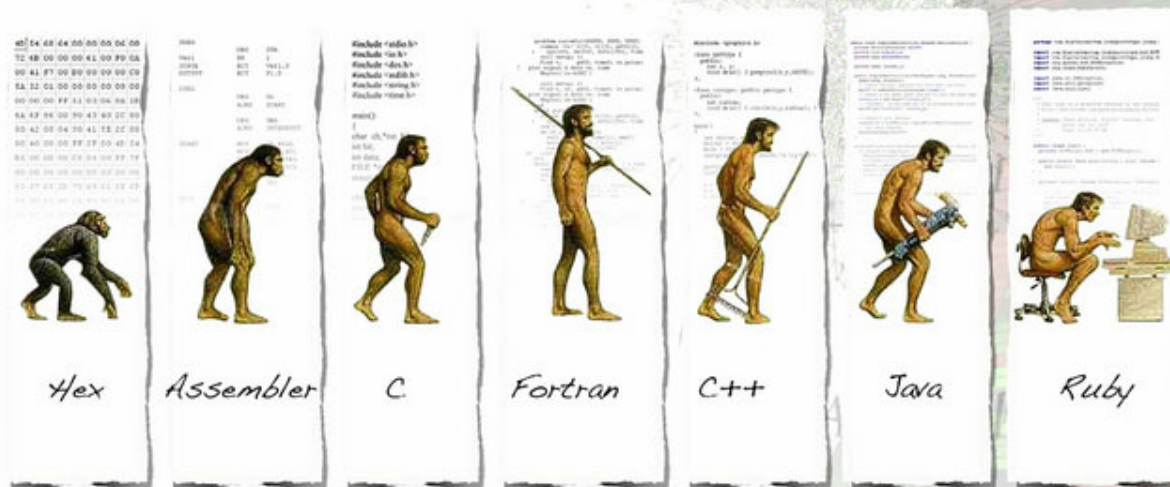
- Why do we need languages?
 - Humans communicate
 - sign language, body language, braille
 - Birds communicate
 - mark territories, attract for mating, warn danger
 - Animals communicate
 - mark territories, convey need, preparation for attack
 - Aliens?

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



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

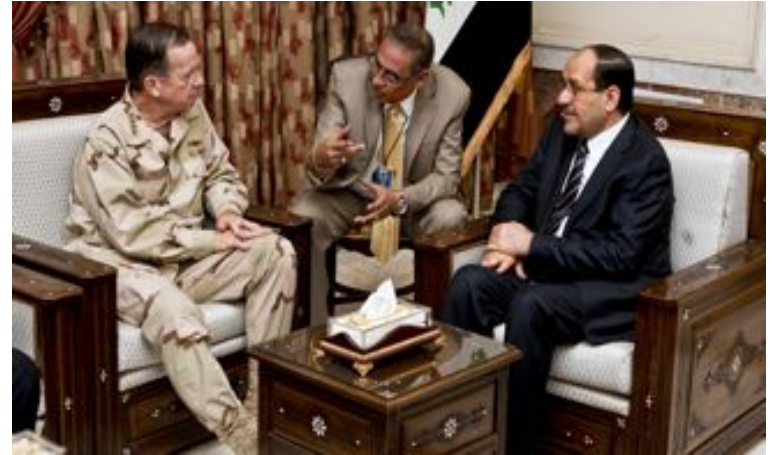
- Using mobile buttons
- Using ipad touch
- Using a calculator
- Using a fan switch
- Using a remote for projector / laser

... 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?
 - எனக்கு தமிழ் தெரியும்
 - मुझे हिंदी आता है
 - Ich kenne Deutsch
 - I know English



Jobs of a Compiler

- **Translate**: input language, output language
- **Maintain correctness**
 - चाचाजी अजमेर गए ।
 - Uncle 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 मिला है।

Good Language (2015)

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।

Good Language (2018)

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

मुझे किताबें मिलीं लेकिन उससे ज्यादा मुझे आपका पत्र मिला।

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

मुझे किताबें मिलीं लेकिन मुझे उससे अधिक पत्र मिला।

I got books but I received more letters from him.

मुझे किताबें मिलीं लेकिन मुझे उससे ज्यादा पत्र मिले।

I got the books but I received more letters from him.

मुझे किताबें मिलीं लेकिन मुझे उससे अधिक पत्र प्राप्त हुए।

Good Language (2020)

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

मुझे किताबें मिलीं लेकिन उससे ज्यादा मुझे आपका पत्र मिला।

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

मुझे किताबें मिलीं, लेकिन इससे ज्यादा मुझे आपका पत्र मिला।

I received books, but more than that I received your letter.

मुझे किताबें मिलीं, लेकिन इससे भी ज्यादा मुझे आपका पत्र मिला।

I received books, but more than that I received your letter.

मुझे किताबें मिलीं, लेकिन इससे भी ज्यादा मुझे आपका पत्र मिला।

Good Language (2024)

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

मुझे किताबें तो मिलीं हीं, उससे भी अधिक मुझे आपका पत्र मिला।

Not only did I get the books, but more than that I got your letter.

मुझे न केवल किताबें मिलीं, बल्कि उससे भी अधिक मुझे आपका पत्र मिला।

Not only did I get the books, but more than that I got your letter.

मुझे न केवल किताबें मिलीं, बल्कि उससे भी अधिक मुझे आपका पत्र मिला।

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

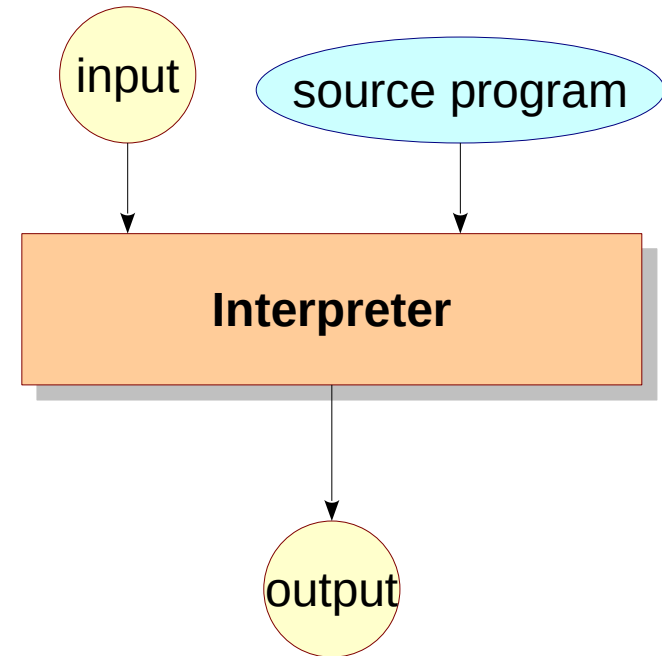
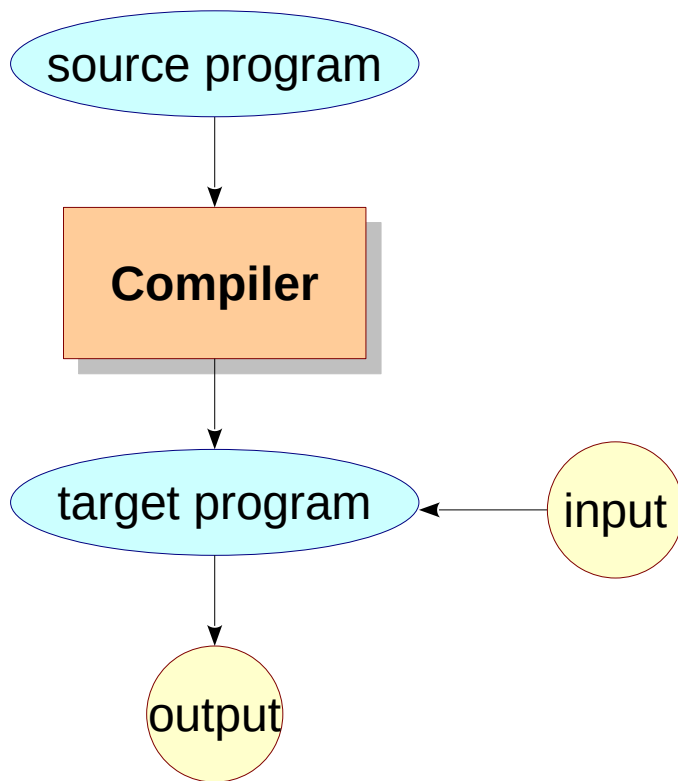
```
char*f="char*f=%c%s%c;main(){printf(f,34,f,34,10);}%c";main(){printf(f,34,f,34,10);}
```

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.

Classwork

- Write a program that takes a function name as input and calls that function.



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

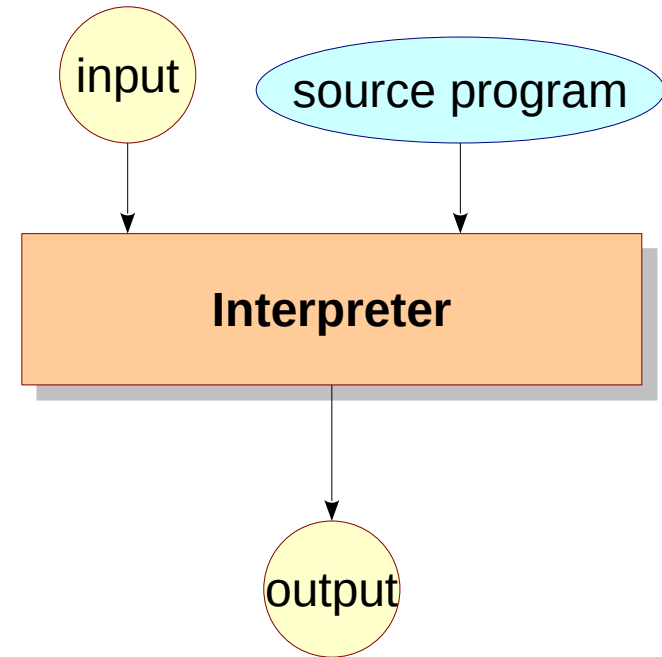
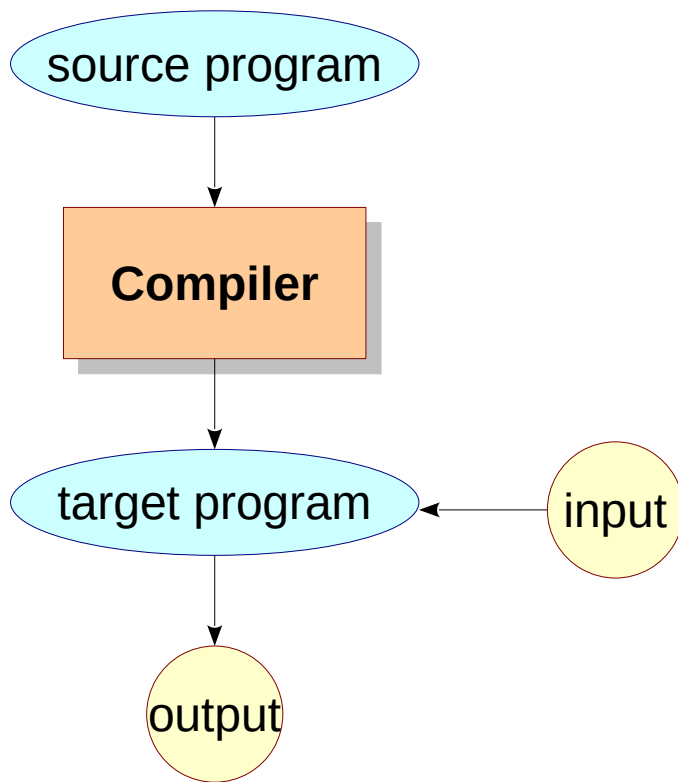
```

$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);
}
  
```

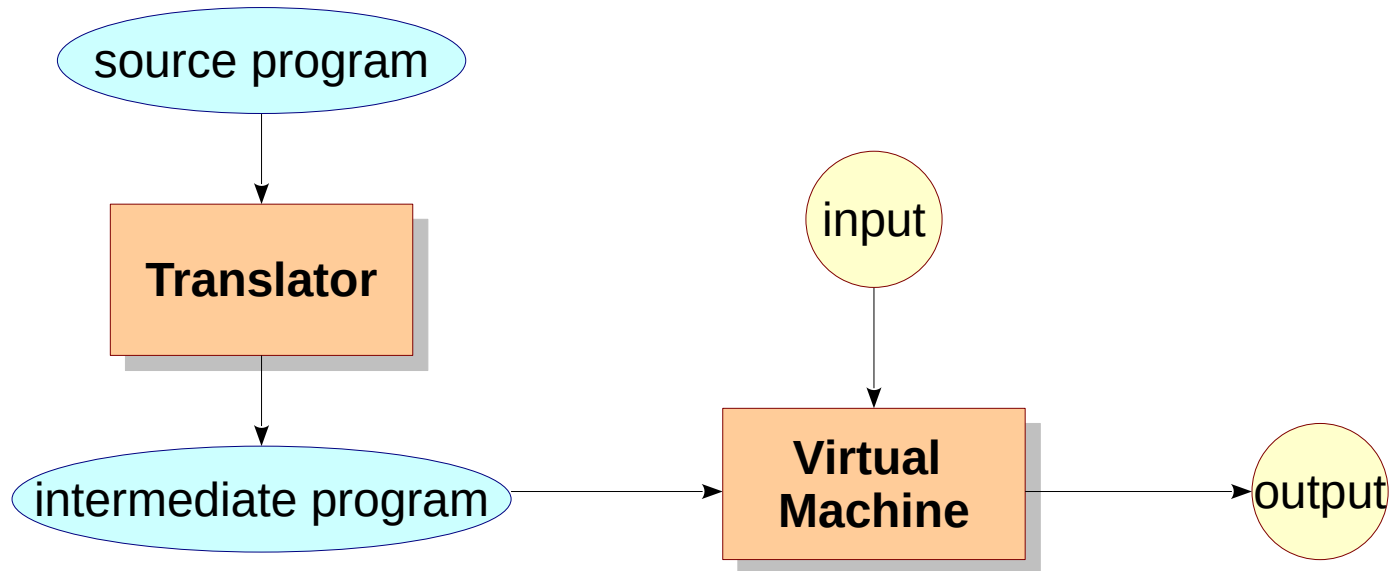
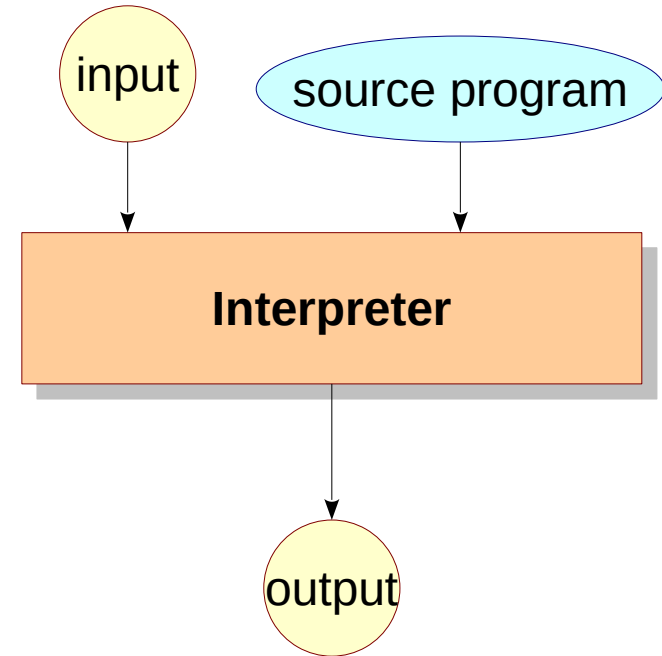
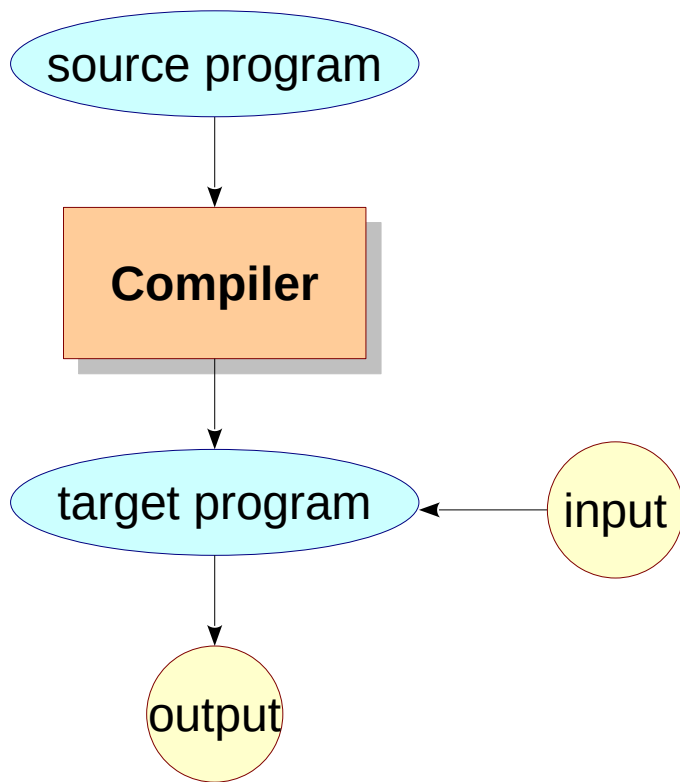


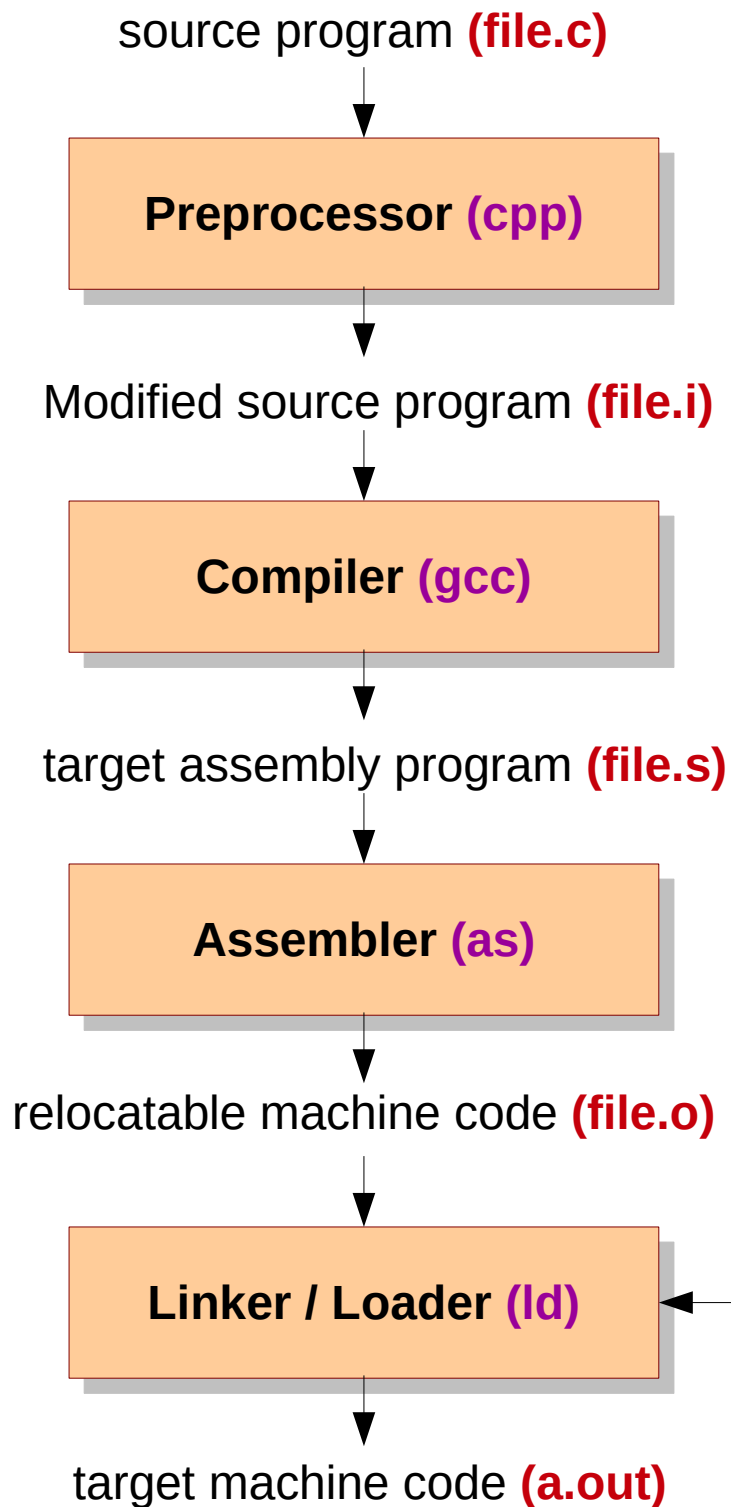
- What does this mean?
 - You may be able to do the following with compilers.

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

is equivalent to

```
x += 18;
```





- `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`

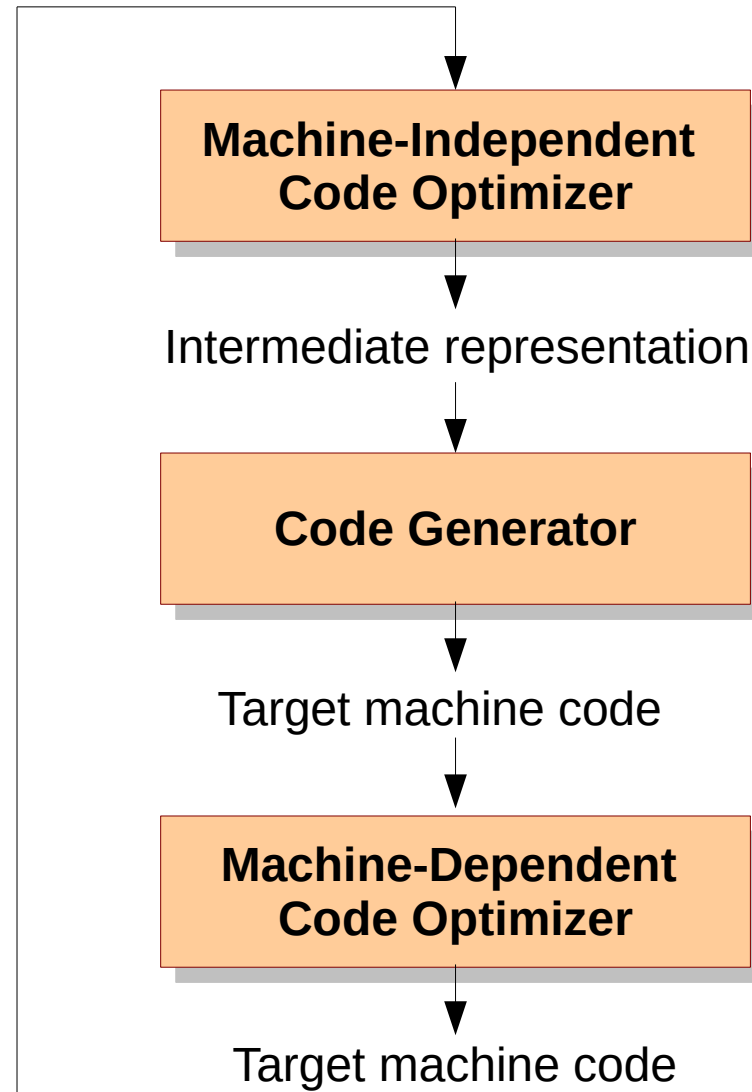
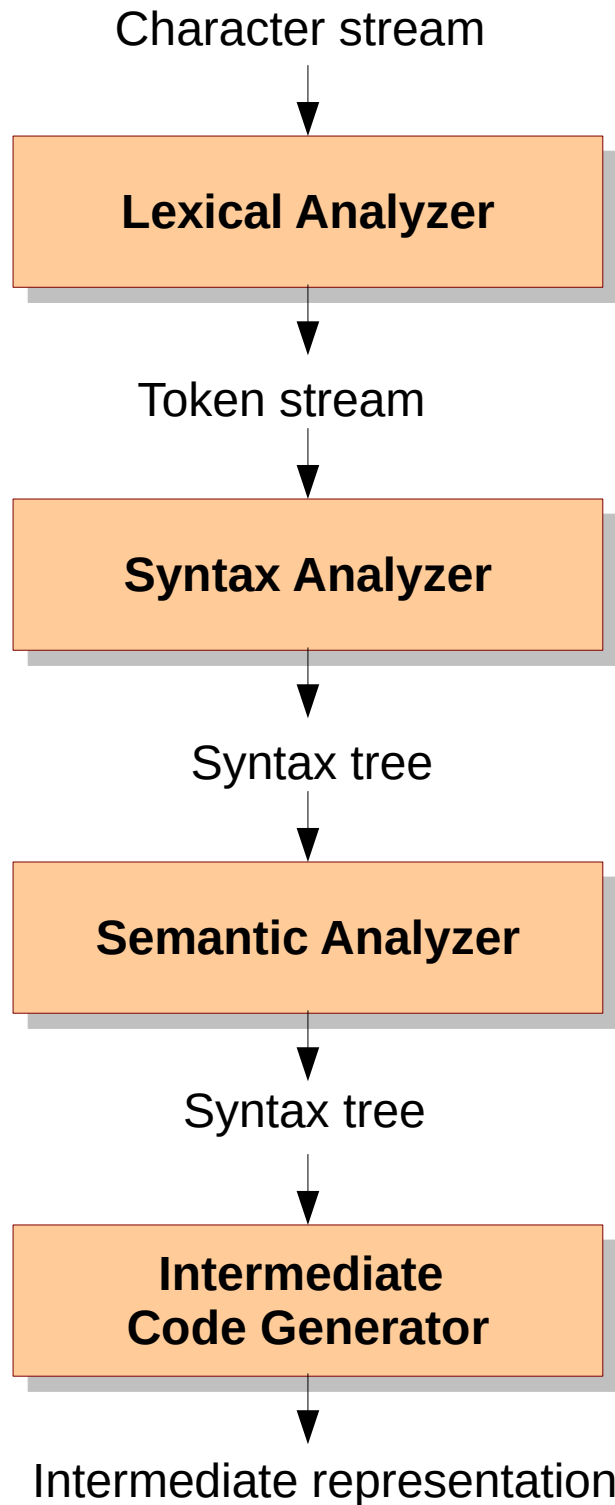
Language Translators

- **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**.

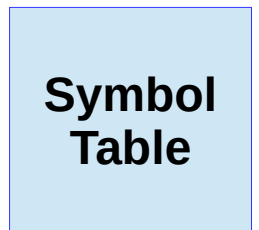
Homework

- Exercises 1.1.1-5 from ALSU.

Frontend



Backend

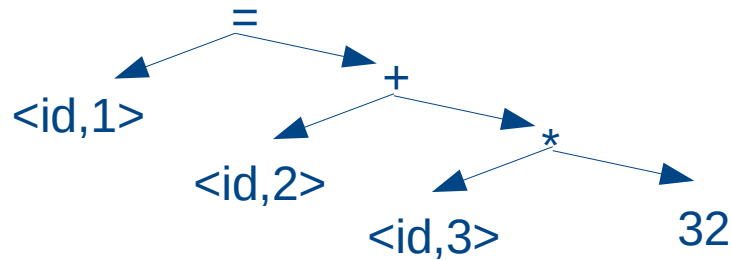


$z = x + y * 32$

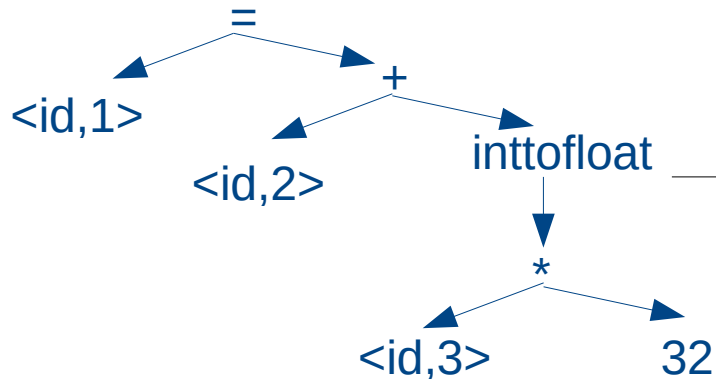
Lexical Analyzer

`<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`

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

Symbol Table

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

**Machine-Dependent
Code Optimizer**

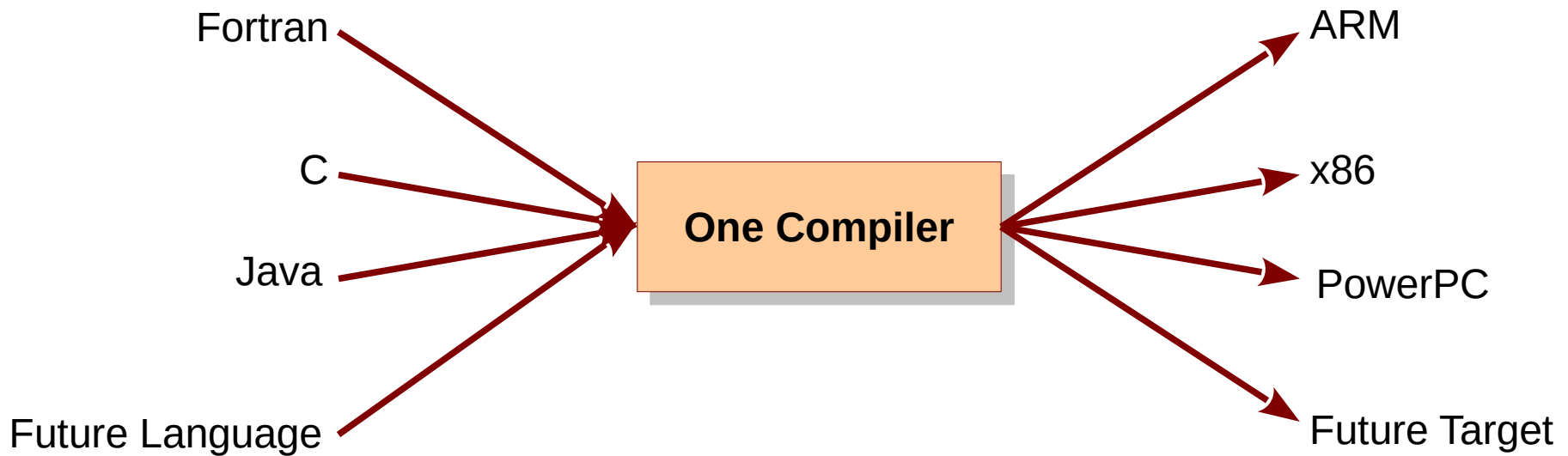
Symbol Table

- Records variable names
- Collects 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;`

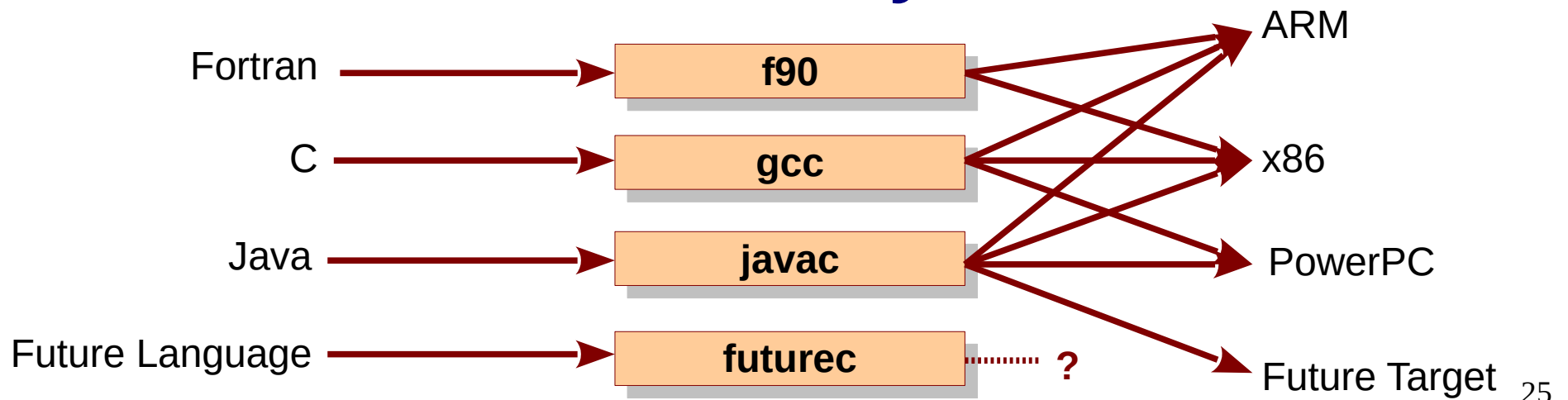
1	z	...
2	x	...
3	y	...

Symbol Table

Ideal World



Reality



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 5.4, clang 5.5, gcc 5.0.1 on x86_64 with lonestar 1.2 or above on Ubuntu 16 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
 - ...
- S I o w, T_e**D**i^u**s**, and ErorrP run.

Maggie and Buildings



Punched Tape



Punched Card

PDP

DOM 1435 PRINTED IN INDIA

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.
- How popularity changed

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 and virtual functions
- 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.