Programming Paradigms

Mădălina Eraşcu

(after an original presentation by Vitaly Shmatikov)

What Is a Programming Language?

- Formal notation for specifying computations, independent of a specific machine
  - Example: a factorial function takes a single non-negative integer argument and computes a positive integer result
    - Mathematically, written as fact: nat $\rightarrow$ nat
- Set of imperative commands used to direct computer to do something useful
  - Print to an output device: printf("hello world\n");
    - What mathematical function is "computed" by printf?

Computable functions

- Function f is computable if some program P computes it.
- Some functions are computable, some are not
  - Example: f(x) = if x=0 then 1 else f(x-2)
- Programming language implementation
  - Can report error if program result is undefined due to an undefined basic operation (e.g., division by zero)
  - Cannot report error if program will not terminate

Factorial Function

- Imperative (C)

```
#include <stdio.h>
int main()
{
  int c, n, fact = 1;
  printf("Enter a number to calculate it's factorial\n");
  scanf("%d", &n);
  for (c = 1; c <= n; c++)
    fact = fact * c;
  printf("Factorial of %d = %d\n", n, fact);
  return 0;
}
```

Factorial Function (cont'd)

- Object-Oriented (Java)

```
import java.util.Scanner;
class Factorial{
   public static void main(String args[]){
      int n, c, fact = 1;
      System.out.println("Enter an integer to calculate it's factorial");
      Scanner in = new Scanner(System.in);
      n = in.nextInt();
      if ( n < 0 )
         System.out.println("Number should be non-negative.");
      else{
         for ( c = 1 ; c <= n ; c++ )
            fact = fact*c;
          System.out.println("Factorial of "+n+" is = "+fact);
      }
   }
}
```

Factorial Function (cont'd)

- Multi-paradigme: imperative & OO (C++)

```
#include<iostream>
using namespace std;
int main()
{
    int num,factorial=1;
    cout<<" Enter Number To Find Its Factorial:  ";
    cin>>num;
    for(int a=1;a<=num;a++)
    {
        factorial=factorial*a;
    }
cout<<"Factorial of Given Number is ="<<factorial<<endl;
    return 0;
}
```

Factorial Function (cont'd)

- Logic (Lisp)

```
(defun fact (n)
  (if (< n 2)
      1
    (* n (fact(- n 1)))))

(fib 10)
```

Factorial Function (cont'd)

- Functional (Prolog)

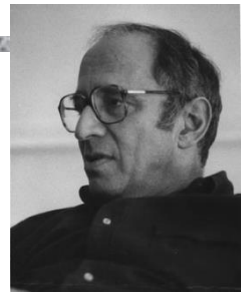
```
fact(X, 1) :- X<2.
fact(X, F) :- Y is X-1, fact(Y,Z), F is Z*X.
```

Principal Paradigms

- Imperative / Procedural
- Object-Oriented
- Logic
- Functional / Applicative
- Concurrent
- Scripting

- In reality, very few languages are “pure”
  - Most combine features of different paradigms

Where Do Paradigms Come From?



- Paradigms emerge as the result of social processes in which people develop ideas and create principles and practices that embody those ideas
  - Thomas Kuhn. "The Structure of Scientific Revolutions."
- Programming paradigms are the result of people's ideas about how programs should be constructed
  - … and formal linguistic mechanisms for expressing them
  - … and software engineering principles and practices for using the resulting programming language to solve problems

In this course: Imperative language C

- Imperative (procedural) programs consists of actions to effect state change, principally through assignment operations or side effects
  - Fortran, Algol, Cobol, PL/I, Pascal, Modula-2, Ada, C
- OO programming is not always imperative, but most OO languages have been imperative
  - Simula, Smalltalk, C++, Modula-3, Java
  - Notable exception: CLOS (Common Lisp Object System)

Functional and Logic Paradigms

- Focuses on function evaluation; avoids updates, assignment, side effects
- Not all functional languages are "pure"
  - In practice, rely on non-pure functions for input/output and some permit assignment-like operators
- Logic programming is based on predicate logic
  - Targeted at theorem-proving languages, automated reasoning, database applications

Concurrent and Scripting Languages

- Concurrent programming cuts across imperative, object-oriented, and functional paradigms
- Scripting is a very "high" level of programming
  - Rapid development; glue together different programs
- Very popular in Web development
  - Especially scripting active Web pages

Design Choices

- C: Efficient imperative programming with static types
- C++: Object-oriented programming with static types, subtype and parametric polymorphism
- Java: Imperative, object-oriented, and concurrent programming with static types and garbage collection
- Lisp: recursive programming with dynamic types
- Prolog: Practical functional programming with strict (eager) evaluation and polymorphic type inference
- Python: scripting language for web applications; its main feature is code readability

Billion-Dollar Mistake

Failed launch of Ariane 5 rocket (1996)

- $500 million payload; $7 billion spent on development

Cause: software error in inertial reference system

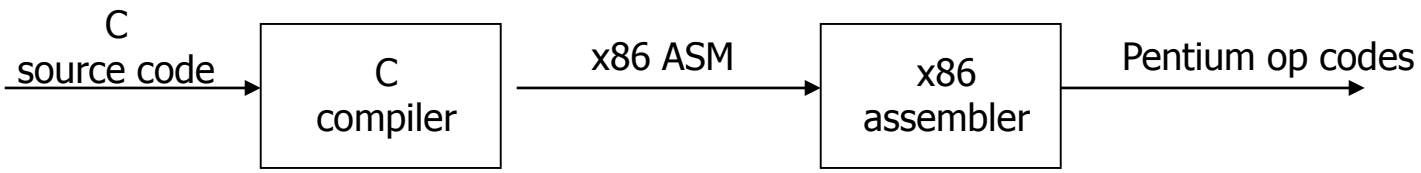
- Re-used Ariane 4 code, but flight path was different
- 64-bit floating point number related to horizontal velocity converted to 16-bit signed integer; the number was larger than 32,767; inertial guidance crashed

Program Correctness

- Assert formal correctness statements about critical parts of a program and reason effectively
  - A program is intended to carry out a specific computation, but a programmer can fail to adequately address all data value ranges, input conditions, system resource constraints, memory limitations, etc.
- Language features and their interaction should be clearly specified and understandable
  - If you do not or can not clearly understand the semantics of the language, your ability to accurately predict the behavior of your program is limited
- Some errors can be detected at compile time.

Language Compilation

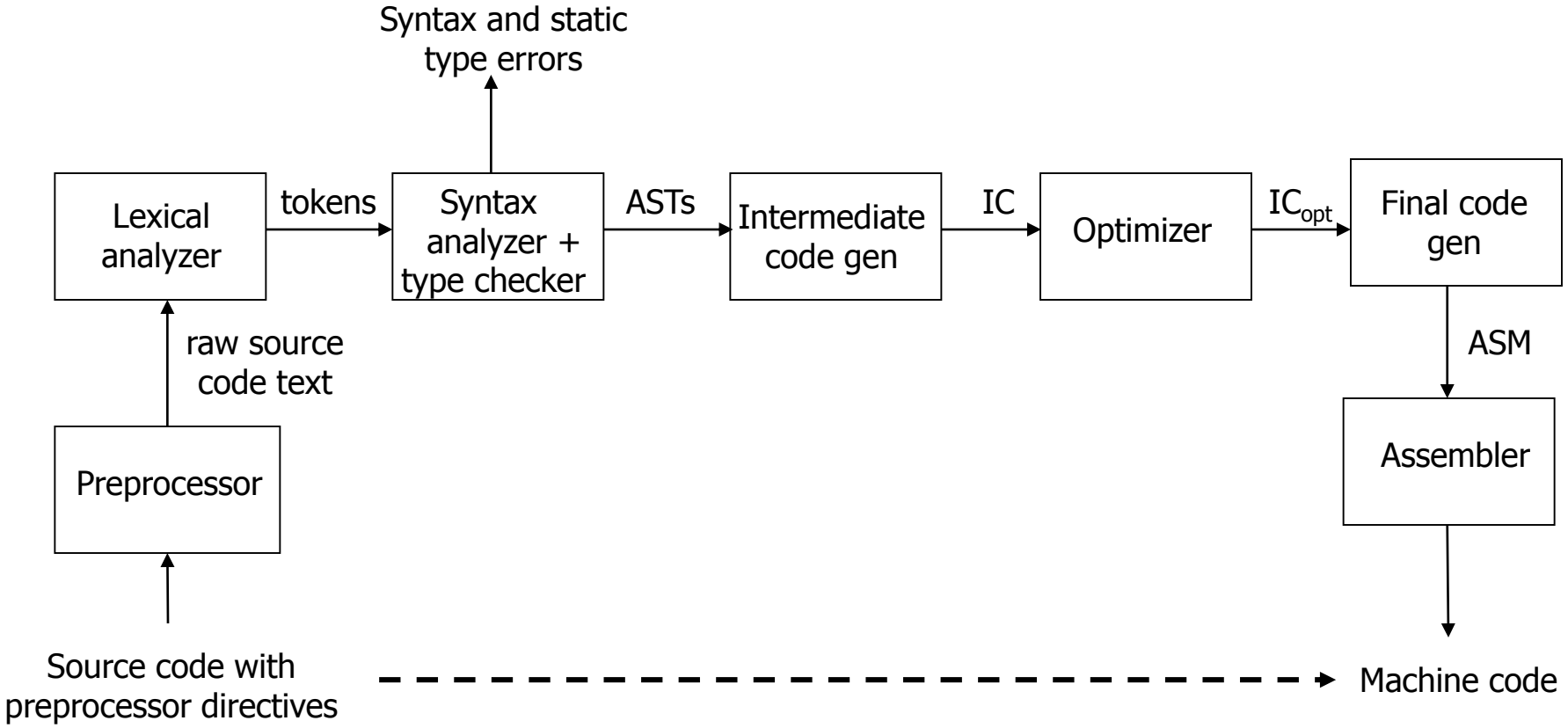
- Compiler: program that translates a source language into a target language
  - Target language is often, but not always, the assembly language for a particular machine



Checks During Compilation

- Syntactically invalid constructs
- Invalid type conversions
  - A value is used in the "wrong" context, e.g., assigning a float to an int
- Static determination of type information is also used to generate more efficient code
  - Know what kind of values will be stored in a given memory region during program execution
- Some programmer logic errors
  - Can be subtle: if (a = b) … instead of if (a == b) …

Compilation Process



Phases of Compilation

- Preprocessing: conditional macro text substitution
- Lexical analysis: convert keywords, identifiers, constants into a sequence of tokens
- Syntactic analysis: check that token sequence is syntactically correct
  - Generate abstract syntax trees (AST), check types
- Intermediate code generation: "walk" the ASTs and generate intermediate code
  - Apply optimizations to produce efficient code
- Final code generation: produce machine code

The gcc (http://www.gnu.org/) Compiler

- One of the most popular C compilers, supplied with Linux but available for other platforms as well
- CodeBlocks (http://www.codeblocks.org/) the IDE that we use in the lab comes with a gcc compiler. (Read here http://en.wikipedia.org/wiki/Integrated_development_environment about what an IDE is)
- Versions of gcc: ansi, c89, c99

Standardization

- The C language was developed by Denis Ritchie & Brian Kernighan between 1969 and 1973 at AT&T Bell Labs.
- C continued to evolve during the 1970s; during this period that the first book on C appeared „The C Programming Language" (Kernighan & Ritchie) which became the reference book of programmers since no official standard existed
- Around 1980s C becomes very popular: many compilers for different compuers running different operating systems.

Standardization (cont'd)

- Compilers used K&R book as reference which was fuzzy about some features of the language; moreover C continuosly evolved.
- Necesity of a C standard arised.
- First standard: 1983 – 1989 supervised by American National Standards Institute (ANSI).
- Formally aproved 1989 ANSI standard X1159-I989 (known as C89 or C90)
- The language underwent a few changes in 1995

-> C99