CSIS 4244

History and Evolution of Programming Languages

In the beginning there was machine language…

No, wait… in the beginning there was *Ada*!

Augusta Ada King, Countess of Lovelace (10 December 1815 – 27 November 1852), born Augusta Ada Byron, was an English writer chiefly known for her work on Charles Babbage's early mechanical general-purpose computer, the analytical engine. Her notes on the engine include what is recognised as the first algorithm intended to be processed by a machine; as such *she is regarded as the world's first computer programmer*.

- *Wikipedia*

1940's – 1950's

In the beginning there was machine language…

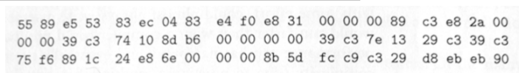
- Numeric codes
- Tedious & error prone

Then assembly language

- Mnemonics replace numbers
- One-to-one: machine instruction to assembler

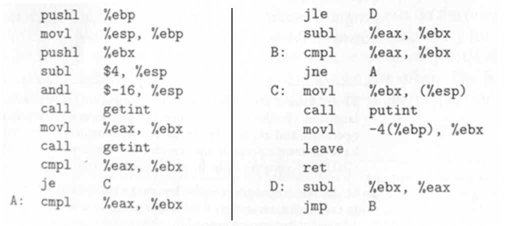
Low Level Code

Example 1.1: GCD program in x86 machine language



```
55 89 e5 53  83 ec 04 83  e4 f0 e8 31  00 00 00 89  c3 e8 2a 00
00 00 39 c3  74 10 8d b6  00 00 00 00  39 c3 7e 13  29 c3 39 c3
75 f6 89 1c  24 e8 6e 00  00 00 8b 5d  fc c9 c3 29  d8 eb eb 90
```

Example 1.2: GCD program in x86 assembler



```
    pushl  %ebp                 jle    D
    movl   %esp, %ebp           subl   %eax, %ebx
    pushl  %ebx             B:  cmpl   %eax, %ebx
    subl   $4, %esp             jne    A
    andl   $-16, %esp       C:  movl   %ebx, (%esp)
    call   getint               call   putint
    movl   %eax, %ebx           movl   -4(%ebp), %ebx
    call   getint               leave
    cmpl   %eax, %ebx           ret
    je     C                D:  subl   %ebx, %eax
A:  cmpl   %eax, %ebx           jmp    B
```

Konrad Zuse: Plankalkül

His greatest achievement was the world's first functional program-controlled Turing-complete computer, the Z3, which became operational in May 1941.

While working on his Z4 computer, Zuse realised that programming in Machine code was too complicated, so he designed the first high-level programming language, Plankalkül ("Plan Calculus"), in 1945/6.

- *Wikipedia*

"The very first attempt to devise an algorithmic language was undertaken in 1948 by K. Zuse. His notation was quite general, but the proposal never attained the consideration it deserved."

- *Heinz Rutishauser* (ALGOL)

Plankalkül Syntax

Advanced data structures

Floating point, arrays, records

Invariants

Example:

An assignment statement to assign the expression A[4] + 1 to A[5]

```
  |  A + 1 => A
V |  4        5          (subscripts)
S |  1.n      1.n        (data types)
```

FORTRAN

FORTRAN I & II – 1957

Designed for the "new" IBM 704

Computers were slow and unreliable

Applications were scientific

No programming methodology or tools

Machine efficiency was most important
(Optimizing compilers)

FORTRAN (continued)

Characteristics

- Fixed format
- Names up to six characters
- Implicit typing
- Formatted I/O
- DO loop (counting)

```
DO I = 1,5
```

- Three-way selection statement (arithmetic IF)

```
IF (X-Y) 10,20,30
```

FORTRAN (continued)

FORTRAN IV – (1960-62)

- Explicit type declarations
- Logical selection statement
- ANSI standard in 1966

FORTRAN 77

- Character string handling
- Logical loop control statement
- IF-THEN-ELSE statement

FORTRAN 90

- Modules, Parameter type checking
- Dynamic arrays, Pointers
- Recursion, CASE statement

FORTRAN 95

FORTRAN 2003

and in Sept. 2010…

FORTRAN 2008

LISP (1959)

LISt **P**rocessing language (Designed at MIT by John McCarthy – didn't think FORTRAN was much better than assembler)

For AI research

- Process data in lists (rather than arrays)
- Symbolic computation (rather than numeric)
- Interactive

Only two data types: atoms and lists

Uniform representation of data and code

Simple syntax

Compact clear semantics

LISP (continued)

Pioneered functional programming

No need for variables or assignment

Control via recursion and conditional expressions

Still a dominant language in AI

Scheme is a smaller and simpler version of LISP popular for educational applications

GCD in Scheme

```
(define gcd
  (lambda (a b)
    (cond ((= a b) a)
          ((> a b) (gcd (- a b) b))
          (else (gcd (- b a) a)))))
```

ALGOL 58/60

Language features
- Concept of type was formalized
- Arrays could have any number of subscripts
- Parameters modes (`in` & `out`)
- Compound statements (`begin..end`)
- Semicolon as a statement separator
- `if` had an `else-if` clause
- Subprogram recursion
- No I/O

ALGOL (continued)

Successes:
- The standard way to publish algorithms for over 20 years
- All subsequent imperative languages are based on it
  - ALGOL 68 strongly influenced Pascal, C, and Ada
- First machine-independent language
- First language whose syntax was formally defined (BNF)

ALGOL (continued)

Failures:
- Never widely used (especially in U.S.) because
  - No I/O and the character set made programs non-portable
  - Too hard to implement
  - IBM supported FORTRAN
  - Formal syntax description considered too complex

COBOL (1960)

COmmon **B**usiness **O**riented **L**anguage - Designed by a committee from government (Grace Hopper) & industry
- DoD's first effort to provide a single language for all military branches
- English-like syntax, verbose
- Source of many Y2K problems
- For many years was the most widely used business application language
- Cobol 2002 is object-oriented

SNOBOL (1964)

String Oriented SymBOlic Language - Designed as a string manipulation language (at Bell Labs by Farber, Griswold, and Polensky)
- Powerful operators for string pattern matching
- Dynamic typing and storage allocation
- Variables are untyped
  - They acquire a type when assigned a value
- Storage is allocated to a variable when it is assigned a value

PL/1 (1965)

Programming Language 1 – Major IBM effort to combine scientific and business languages
- First unit-level concurrency
- First exception handling
- Too large, too complex, many poorly designed features
- Was (and still is) used for both scientific and business applications

SIMULA 67

Simulation Language

- Designed primarily for system simulation (in Norway by Nygaard and Dahl)
- Based on ALGOL 60
- Classes are the basis for data abstraction
- The beginnings of object oriented programming

Pascal (1971)

Named after Blaise Pascal, inventor of mechanical calculator

- Designed by Niklaus Wirth (who quit the ALGOL 68 committee)
- Designed for teaching structured programming
- Nothing new, just small and simple
- Once the most widely used language for teaching programming in colleges

C (1972)

Designed for systems programming (at Bell Labs by Dennis Richie)

- Evolved from B and ALGOL 68
- Powerful set of operators, but poor type checking
- Initially spread as the way to port UNIX

```
int gcd(int a, int b) {
    while (a != b) {
        if (a > b) a = a - b;
        else b = b - a;
    }
    return a;
}
```

Perl

A scripting language developed by Larry Wall

A *script* (file) contains instructions to be executed

Originally for report processing in Unix but widely used as a general purpose language

Powerful but somewhat dangerous

Prolog (1972)

Programming in Logic - Developed at the University of Aix-Marseille, and the University of Edinburgh

- Based on formal logic
- Non-procedural
- For creating rule-based systems
- Intelligent database system that uses an inferencing process to determine the truth of given queries

GCD in Prolog

```
gcd(A,B,G) :- A = B, G = A.
gcd(A,B,G) :- A > B, C is A-B, gcd(C,B,G).
gcd(A,B,G) :- B > A, C is B-A, gcd(C,A,G).
```

Smalltalk (1972-1980)

Developed at Xerox PARC, initially by Alan Kay & later by Adele Goldberg

- First full implementation of an object-oriented language (data abstraction, inheritance, and dynamic type binding)
- Pioneered the graphical user interface that everyone uses now

Ada (1983 but began in mid-’70s)

Motivated by

- The “software crisis”
- Having over 450 languages being used for embedded systems

- Another DoD attempt at a "do everything" language
- Huge design effort, involving hundreds of people, much money, and about eight years

Ada (continued)

Competitive design

Included all that was then known about software engineering and language design

Compilers were very difficult to implement; the first really usable compiler came nearly five years after the language design

Ada (continued)

Contributions:

- Packages (support for data abstraction)
- Exception handling
- Generic program units
- Concurrency (through the tasking model)

Ada (continued)

Ada 95 - (began in 1988)

- Support for OOP through type derivation
- Better control mechanisms for shared data (new concurrency features)

Ada 2005

- Emphasis on “safety-critical” systems
- Language subsets
- No DoD funding this time

C++ (1985)

Developed at Bell Labs by Bjarne Stroustrup

- Evolved from C and SIMULA 67
- A large and complex language, in part because it supports both procedural and OO programming
- Popularity grew rapidly, along with OOP
- ANSI standard approved in November 1997

Eiffel (1992)

Designed by Bertrand Meyer
- Not directly derived from any other language
- Influenced by many (Ada, SIMULA, Smalltalk, etc.)
- True object oriented language
- Support for assertions

Java (1995)

Developed at Sun Microsystems
- Based on C++ but significantly simplified
- Supports only OOP
- Has references, but not pointers
- Concurrency using threads

Other Scripting Languages

PHP
- Used for Web applications (server-side); produces HTML markup as output

JavaScript
- Used in Web programming (client-side) to create dynamic HTML documents
- Related to Java only through similar syntax

Python
- An OO interpreted scripting language
- Type checked but dynamically typed

Ruby
- Pure object-oriented language
- Both objects and classes are dynamic

C#

Part of the .NET development platform

Based on C++ and Java

A language for component-based software development

All .NET languages (C#, Visual BASIC .NET, Managed C++, J#, and ECMAscript) use Common Type System (CTS), which provides a common class library

Becoming widely used

Markup/Programming Hybrid Languages

XML
- eXtensible Markup Language: a metamarkup language

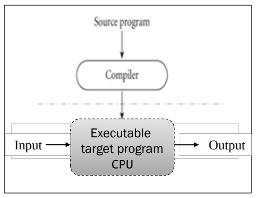
XSLT
- eXtensible Stylesheet Language Transformation (XSTL) transforms XML documents for display
- Programming constructs (e.g., looping)

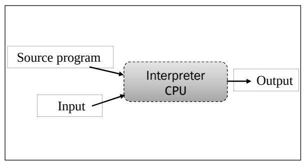
JSP
- Java Server Pages: a collection of technologies to support dynamic Web documents
- servlet: a Java program that resides on a Web server; servlet's output is displayed by the browser

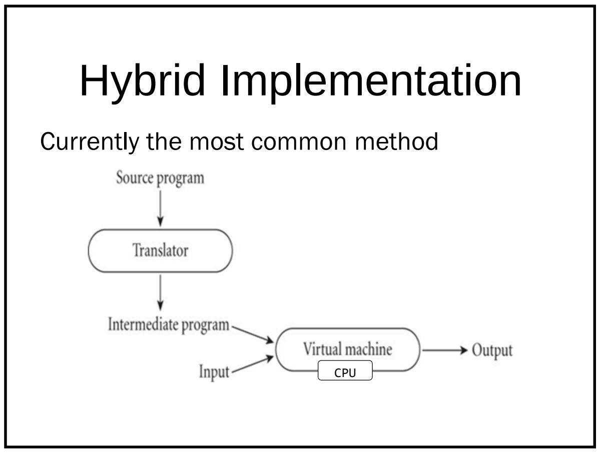
Implementation Methods – Compiled vs. Interpreted

Compilation: A *translator* brings the language down to the level of the machine.

Interpretation: A *virtual machine* brings the machine up to the level of the language.

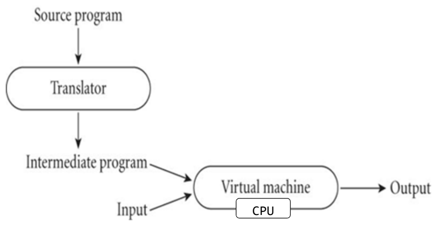






Hybrid Implementation

Currently the most common method


Source program
Translator
Intermediate program
Input
Virtual machine
CPU
Output