The Role of Programming Languages

Chapter 1:

Programming Languages: Concepts and Constructs by Ravi Sethi

What is a Programming Language?

- a tool for instructing machines
- a means of communicating between programmers
- a vehicle for expressing high-level designs
- a notation for algorithms
- a way of expressing relationships between concepts
- a tool for experimentation
- a means for controlling computerized devices

Language Designers

- Balance
- … making computing convenient for *programmers (a fool with a tool is still a fool)*
- and making efficient use of computing machines *(... Why do I have to state this?)*

Levels

- Gross distinction between programming language
- based on readability
- based on independence
- based on purpose (specific … general)

Levels

- Machine *level* language
- Assembly *level* language
- High-level language (3GL)
- sometimes 4GL - fourth Generation Language

Machine Level

- 00000010101111001010
- 00000010101111001000
- 00000011001110101000
- Can you tell what this code fragment does?
- Can it be executed on any machine?
- Is it general purpose?

Assembly Language

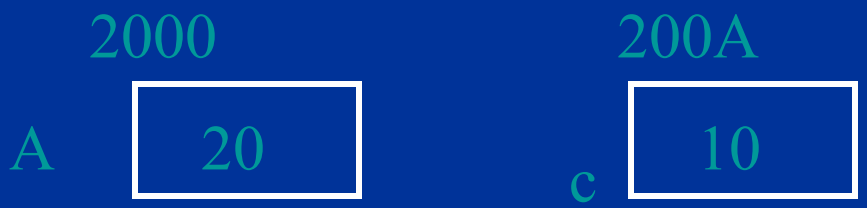
- Look at figure 1.1
- LD R1,”0”
- LD R2, M
- ST R2, R1
- … real assembly used mnemonics
- Add A(M), …. Had to do your own indexing
- What does this program do?

Assembly Language

- Look at page 63 in your text and figure 3.1
- Can you understand what it does now?

Basic Concepts of a RAM machine

- Memory: addresses, contents
- Program: instructions
- input/output:(files)



A = 3 + c

lvalue-> address

rvalue->contents

High Level

- Readable familiar notations
- machine independence
- availability of program libraries
- consistency check (check data types)

Problems of Scale

- Changes are easy to make
- isolated program fragments can be understood
- BUT… one small bug can lead to disaster
- read the NOT story about Mariner rockets
- Notice how the chairman does not understand that a “small” problem can lead to devastating result and why it was not caught

Bugs

- *Programming testing can be used to show the presence of bugs, but never their absence!*
- Dijkstra
- Programming Languages can help
- readable and understandable
- organize such that parts can be understood

Role of Programming Languages

- Art (science) of programming is organizing complexity
- Must organize in such a way that our limited powers are sufficient to guarantee that the computation will establish the desired effect
- (Dijkstra - structured programming, sometimes referred to as goto-less programming)

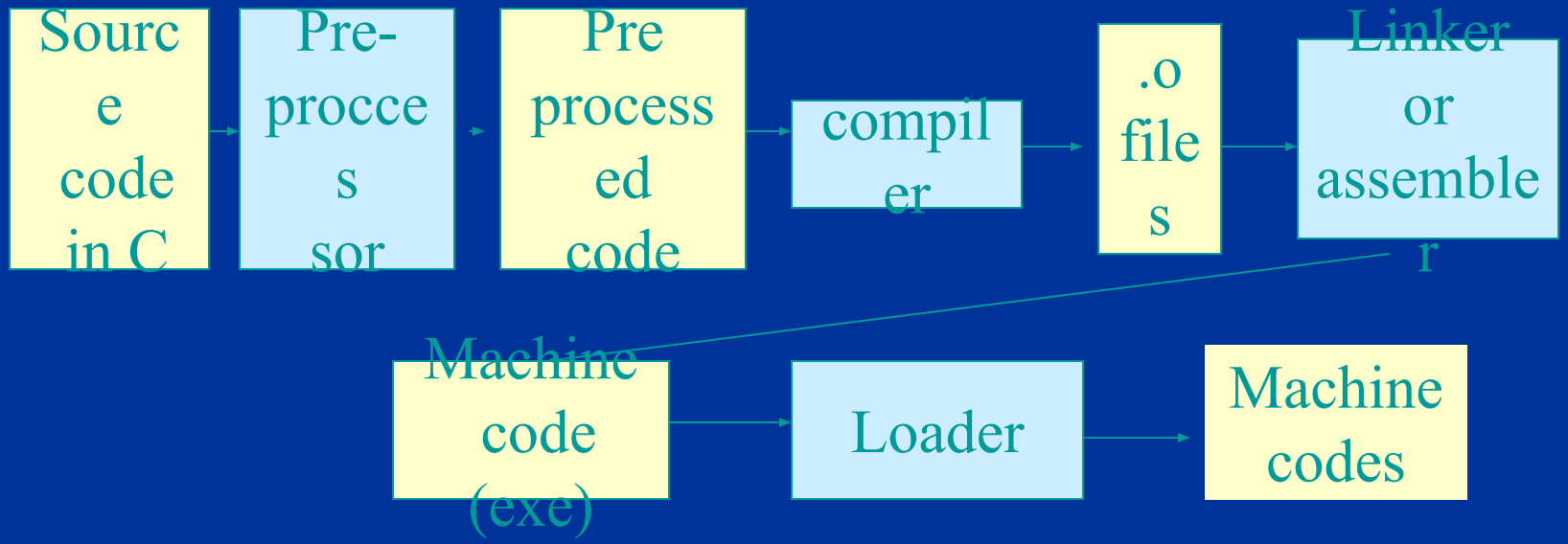
Programming Paradigms

- Imperative - action oriented, sequence of actions
- Functional - LISP, symbolic data processing
- Object-Oriented
- Logic - Prolog, logic reasoning
- Sequential and concurrent

Language Implementation

- Compiler - source code it translated into machine code (all at once)
- Interpreter - machine is brought up to the language (one statement at a time)

Compiled C


Sourc e code in C
Pre- procce s sor
Pre process ed code
compil er
.o file s
Linker or assemble r
Machine code (exe)
Loader
Machine codes

Interpreted Code

- Each instruction is interpreted by machine interpreter
- does not produce object code

Comparisons

- Compilation more efficient
- interpreted more flexible

Testing your skill

- Do 1.4 (a,b,c) in PL book
- Do 1.5
- For each file, include a file header:
  - what this file accomplishes - description
  - what “entities” are in this file
  - dependencies
  - structure

Testing your skill

- For each module, include a module header:
  - what this module accomplishes - description
  - dependencies (parameters(in, out, inout), global data (accessed or modified), called by (fanin), calls (fanout))
  - restrictions
  - programmer
  - date created
  - modifications

Testing your skill

- For the test cases, include a test header:
  - for each input, put the expected output, date executed, name of tester and passed/failed