CSE 216
Class Notes

Ben Feuer

Spring 2024

Contents

Chapter 1

Concepts and terms to know

1.1 From Lecture 1

Note:-

Bit, Byte, Word, RAM, CPU, Fetch-Decode-Execute, Instruction Set Architecture, Hertz, K, M, G, T, Assembler, Byte-Addressable, Compiler, Interpreter, Native Code, Virtual Machine, Call stack segment, Heap segment, Text segment, Global segment.

Definition 1.1.1: Bit

A bit is the smallest unit of data in a computer. It can be either a 0 or a 1. Has two possible values. 2^1.

Definition 1.1.2: Byte

A byte is a group of 8 bits. Has 2^8 possible values. ASCII = 1 byte, Unicode = 2 bytes.

Definition 1.1.3: Word

A word is a group of bytes. The size of a word is dependent on the architecture of the computer. For example, a 32-bit computer has 4-byte words. A 64-bit computer has 8-byte words. Unit of data that a CPU can process in one go. A word is essentially one instruction.

Definition 1.1.4: Integers - Java

Integer - 4 bytes, Short - 2 bytes, Long - 8 bytes. Two's complement representation.

Definition 1.1.5: Real Numbers - Java

Float - 4 bytes, Double - 8 bytes. IEEE 754 standard. Have three parts: sign, exponent, and mantissa.

Definition 1.1.6: RAM

Random Access Memory. A type of computer memory that can be accessed randomly. It is volatile, meaning that it loses its contents when the power is turned off. Ram is used for storing data and instructions that are currently being used by the CPU. Short term memory for computer's current needs. Ram stores data and instructions that are currently being used by the CPU.

Definition 1.1.7: CPU

Central Processing Unit. The part of the computer that performs most of the processing. It is the brain of the computer.

Definition 1.1.8: ALU

Arithmetic Logic Unit. The part of the CPU that performs arithmetic and logical operations. Part of CPU that does math.

Definition 1.1.9: Fetch-Decode-Execute

The process by which a CPU executes instructions. The CPU fetches an instruction from memory, decodes it, and then executes it. Size of instruction is dependent on the architecture of the computer. Fetches from RAM.

Definition 1.1.10: Jump instruction

An instruction that changes the flow of control in a program. It is used to implement loops and conditional statements.

Definition 1.1.11: Instruction

A command that tells the CPU to perform a specific operation. It is the basic unit of execution in a computer program.

Definition 1.1.12: Instruction Set Architecture

The set of instructions that a CPU can execute. It is the interface between the hardware and the software. x86, x64 (32bit), ARM, MIPS, RISC-V.

Definition 1.1.13: Hertz

A unit of frequency equal to one cycle per second. It is used to measure the clock speed of a CPU. How many fetch-decode-execute cycles per second can our CPU do. $1Hz = 1 cyclepersecond$.

Definition 1.1.14: K, M, G, T

Kilo, Mega, Giga, Tera. Units of measurement used to represent large numbers. $1K = 10^3$, $1M = 10^6$, $1G = 10^9$, $1T = 10^{12}$.

Definition 1.1.15: Assembler

A program that translates assembly language into machine code.

Definition 1.1.16: Byte-Addressable

A memory system in which each byte has a unique address. This is the most common type of memory system.

Definition 1.1.17: Compiler

A program that translates high-level code into machine code. Originally, compilers were used to translate code from high level languages (HLLs) to assembly. Now, they are used to translate high-level code into machine code (two steps).

Note:-

Java vs. C/C++
Java -¿ IJVM (instruction set for java virtual machine) -¿ JVM (native program that selects proper instructions for the computer, go between java and computer) -¿ managed system. Right once, run anywhere. Same complier, different JVM.
C/C++ -¿ compiled to native code -¿ runs on the computer. Different compiler for different instruction set architectures (ISAs).

Definition 1.1.18: Interpreter

A program that executes code line by line without translating it into machine code (raw binary code).

Definition 1.1.19: Transpiler

A program that translates code from one high-level language to another. Typescript.

Definition 1.1.20: Native Code

Machine code that is specific to a particular CPU architecture.

Definition 1.1.21: Virtual Machine

A software implementation of a computer that runs on another computer. It allows you to run code that is specific to a different architecture.

Definition 1.1.22: Call stack segment

A segment of memory that is used to store function calls and local variables.

Definition 1.1.23: Heap segment

A segment of memory that is used to store dynamically allocated memory.

Definition 1.1.24: Text segment

A segment of memory that is used to store the code of a program.

Definition 1.1.25: Global segment

A segment of memory that is used to store global variables.

Chapter 2

Programming Abstractions - OOP++ and C++

Note:-

Why do rules in a language exist? Use the reasoning to help you understand the language and be a better programmer.

Definition 2.0.1: What is memory

A giant array of bytes.
How do we assign data to/get data from memory?

- In java, we don't
- the JVM does
- Using memory addresses

We use object ids.

Note:-

There are four regions of memory:

- Stack segment - call stack segment of memory for all the local variables for methods that are still active. Active method means that the method hasn't returned yet. Temporary variables declared inside methods, method arguments, removed from memory when a method returns. Dynanic - allocated at runtime.
- Heap segment - for dyanmic data (whenever you use new), data for constructed objects, persistant as long as an existing object variable references this region of memory, for C, C++, Java, C#, Python, etc. Dynamic - allocated at runtime.
- Text segment - instructions
- Global segment - data known at compile time - static/global variables

Instant variables (iv): associated with an instanciated object. Not known at compile time. Stored in heap segment of memory. New instruction signifies heap segment. Malloc for C or callco or realloc.

Static Variables(sv)/Global Variables: known at compile time -¿ stored in global segment of memory. Only one copy of the variable is needed.

Local Variables(lv): (declared in methods) not known at runtime -¿ stored in stack segment of memory.

Top of memory is the stack segment, bottom is the global segment. 0xffffffff to 0x00000000.

Definition 2.0.2: When a function is active

When a function is active, it is on the call stack. When the function has not returned yet, it is active. When the function returns, it is no longer active.

2.1 OOP terms to know

Note:-
Abstraction, strongly typed, class casting, actual vs. apparent, abstract class, concrete class, interface, virtual functions, anonymous class, generics, global, static, call-by-value, HAS-A vs IS-A.

2.1.1 What is abstraction

Definition 2.1.1: Abstraction

Ignoring certain low-level details of a problem to get a simpler solution.
Not focusing on the details, but on the big picture.

Definition 2.1.2: Abstraction Techniques

- Type Abstraction.
- Iteration Abstraction (iterator design pattern).
- Data Abstraction (State design pattern).
- Etc.

Chunking is a form of abstraction our brain uses.

Definition 2.1.3: Types

A type specifies a well-defined set of values. Ex: int, string.

Java, C++, C# are strongly typed languages.

- compiled code is guaranteed to be type safe.
- only exception is casting.

Student s = new Student();
Person p = (Person) s; // ok when Student is a subclass of Person - IS-A relationship.

Person p = new Person();
Student s = (Student) p; // not ok - ClassCastException. Runtime error.

Note:-
An instanciated object has data stored but not in the object itself it is just data associated with the type.

Example 2.1.1 (Type casting)
public class Person {

public String firstName; public String lastName;
public String toString() { return firstName + " " + lastName; } }

public class Student extends Person {
public double GPA;
public String toString() { return super.toString() + " " + GPA; } }

Type casting:
Person p = **new (means instanciated)** Student(); // ok
Student s = new Student(); // ok
p = new Student(); //

Definition 2.1.4: Apparent vs. Actual type

Apparent type is the declared type of a variable. - Compiler enforces all rules
Actula type is the type of a variable at runtime. - Machine (new, heap, JVM)

Note:-
JVM has a table of all variables. Stores name, id, and memory address.