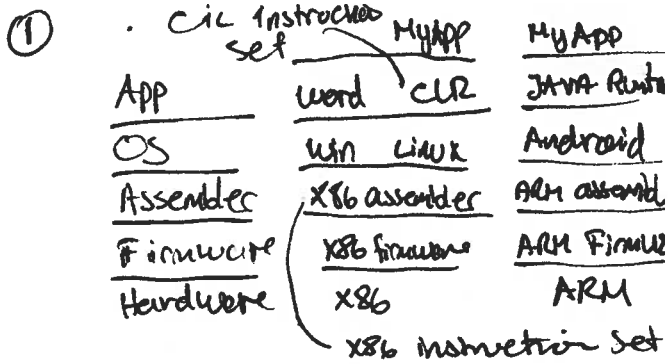
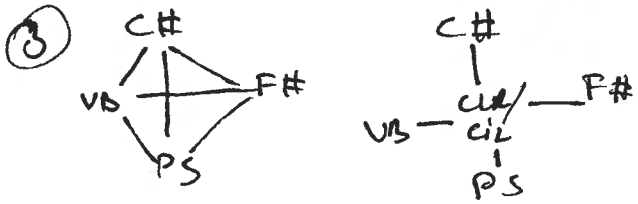
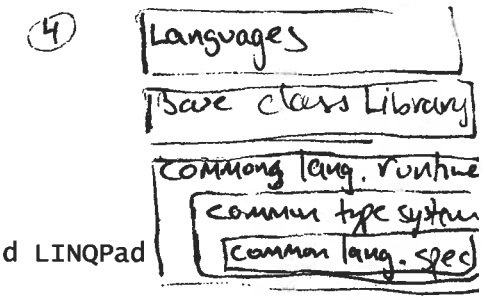
1 time

Lecture 1

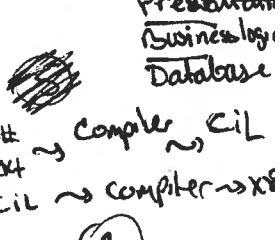
- Lecture 1
  - Agenda
    - Introductions and expectations
    - Formal module description
    - Curriculum proposal
    - Course prerequisites
    - Setting up environment
    - The philosophy of .NET
    - Building C# applications
  - Introductions and expectations
    - Me
    - You
    - Gather mail addresses
  - Formal module description
    - See PDF
  - Curriculum proposal
    - See external file
  - Course prerequisites
    - Literature
      - Pro C# 5.0 and the .NET 4.5 Framework
      - Design Patterns: Elements of Reusable Object-Oriented Software
      - Free web resources
    - Tools
      - Visual Studio Express 2012 For Windows Desktop
      - LINQPad
      - ILSpy
  - Setting up environment
    - Download tools
    - Register Visual Studio with Microsoft
    - Create Hello World application with Visual Studio and LINQPad
  - The philosophy of .NET (how pieces fit together)
    - Pro C# 5.0 and the .NET 4.5 Framework, Chapter 1
  - Common Language Runtime (I)
    - Regular program that reads and executes managed code
    - Simulates a real computer
    - Virtual stack based computer
    - Stack: first-in/first-out data structure
    - Example: manual postfix addition using a stack
      - Prefix, infix, postfix operators
      - Generalize to any function and its operands
    - Example: disassemble add routine to Intermediate Language
    - Common Intermediate Language get Just-in-Time compiled to native code
    - Abstraction layer of CIL above assembler and below C#
  - Common Language Runtime (II)
    - CIL is the protocol between front-end/back-end of compiler
    - CIL is contained in an assembly (dll or exe)
    - With the CLR, the back-end is a runtime component
    - ngen.exe does JIT'ing ahead-of-time of an assembly
    - Applies familiar principle of layering to manage complexity
    - Without the CLR and CIL
      - n languages would require ~~n~~ many two-way bridges
      - CLR/CIL is for languages what a hub is for distributed systems
      - Machine code doesn't contain enough type metadata for interop
      - Without metadata ILSpy wouldn't be able to reverse engineer CIL
      - Before .NET languages on Windows could interop using COM
  - Common Type System
    - Specifies all possible CLR data types and programming constructs
    - A type in one language has same behavior as in any other







general intro to ideas in comp. sci.



- Type is one of class, interface, structure, enumeration, delegate
- An integer in C# shared its characteristics with an integer in F#
- Type members is one of
  - Constructor, finalizer, static constructor, nested type, operator, method, property, indexer, field, read-only field, constant, event
- Data types
  - byte, integer, long, double, ...
- A language doesn't have to support the entire CTS

- Common Language Specification
  - Subset of the CTS that all .NET compilers must support
  - Perhaps some .NET languages don't want pointer support or support unsigned data types
  - CLS as a lowest common denominator guarantees ~~compatibility~~ interoperability

- Base Class Libraries
  - Available to all .NET programming languages
  - Container types, file I/O, threads, graphics, web
  - Every language no longer requires its own libraries
  - Libraries only have to be written once

- Languages
  - Theoretically C# doesn't have to be dependent on the CLR
  - Languages all compile to CIL and that's how they interoperate
  - Example: show interoperability between C# and F# (and ILSpy)

- Building C# applications
  - Pro C# 5.0 and the .NET 4.5 Framework, Chapter 2
  - Easy to read and follow informal chapter