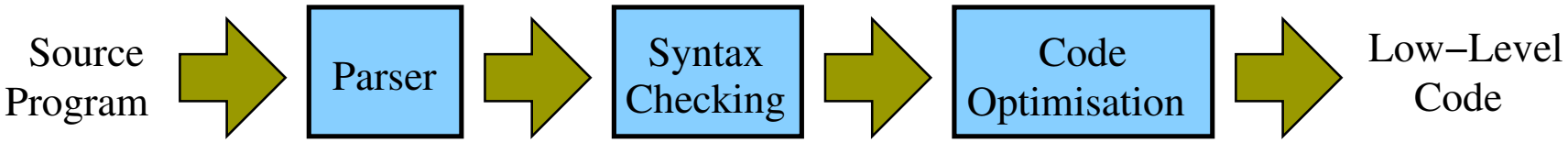
SWEN430 - Compiler Engineering

Lecture 1 - Introduction

Dr David J. Pearce

School of Engineering and Computer Science
Victoria University of Wellington

Compilers

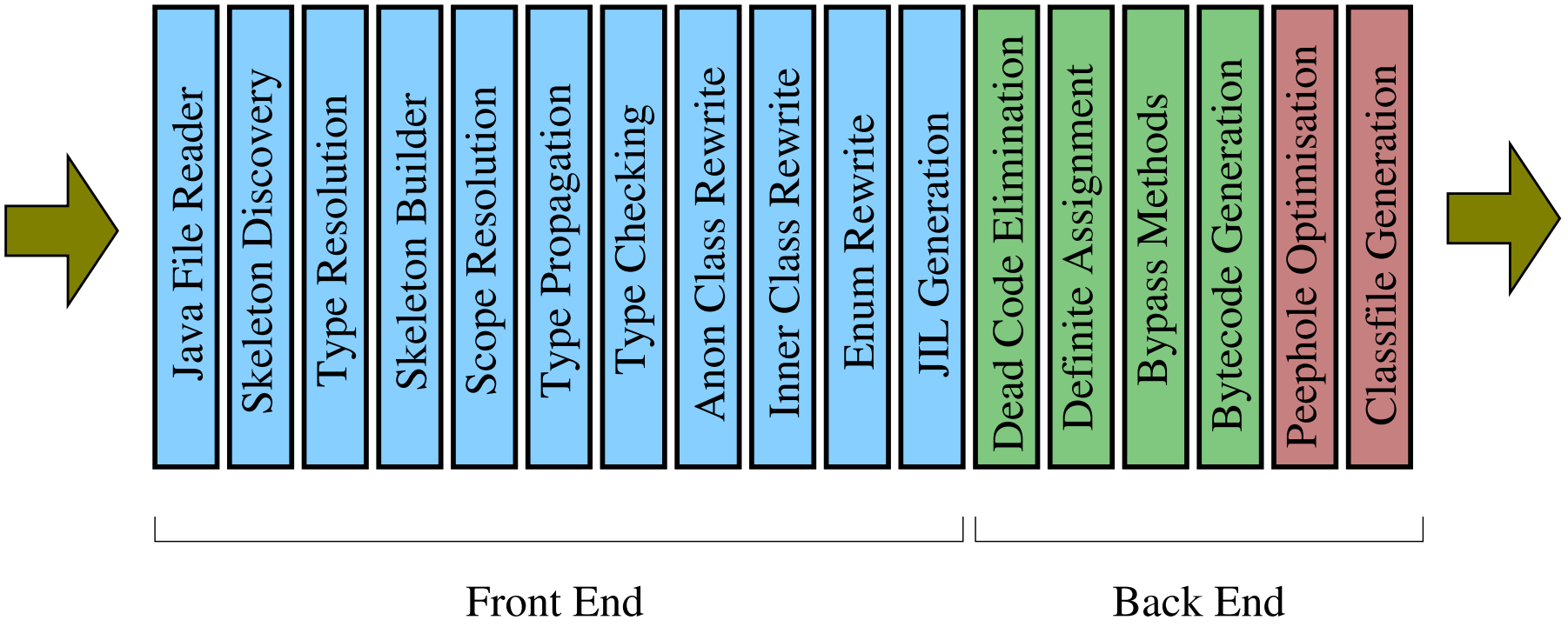


- Compilers translate **source programs** into **low-level code**
- Compilers check for certain errors (e.g. type errors)
- Compilers optimise the program where possible
- Examples (all subject to active research and improvement):
  - Microsoft Visual C#/C++/F#/VB (translates into .NET IL)
  - GCC (e.g. translates C/C++ into x86)
  - GHC (translates Haskell into x86)
  - Javac (translates Java into Java bytecode)

Example: Compiling Java

- Java Language Specification:
  - Details what is **syntactically correct** Java code
  - Details how Java code **should execute**
  - `http://docs.oracle.com/javase/specs/jls/se7/html/index.html`
- Java Virtual Machine Specification:
  - Details what is **syntactically correct** Java Classfile
  - Details how Java bytecodes **should be executed**
  - `http://docs.oracle.com/javase/specs/jvms/se7/html/index.html`

Example: a Java Compiler (JKit)



- Previously developed at VUW by David J. Pearce
- Used for research, teaching and fun!
- Currently has **90 classes** ($\sim$79 KLOC) and **287 JUnit tests**

Java Bytecode (Slightly Simplified)

Test.java

```
class Test {
    public static void main(String[] args) {
        System.out.println("Hello World"); } }
```

javap -verbose Test

```
Compiled from "Test.java"
class Test extends java.lang.Object
...
public static void main(java.lang.String[]);
 Code:
 Stack=2, Locals=1, Args_size=1
 0: getstatic     #2; //Field java/lang/System.out
 3: ldc           #3; //String Hello World
 5: invokevirtual #4; //Meth java/io/PrintStream.println
 8: return
}
```

Example: the Whiley Compiler

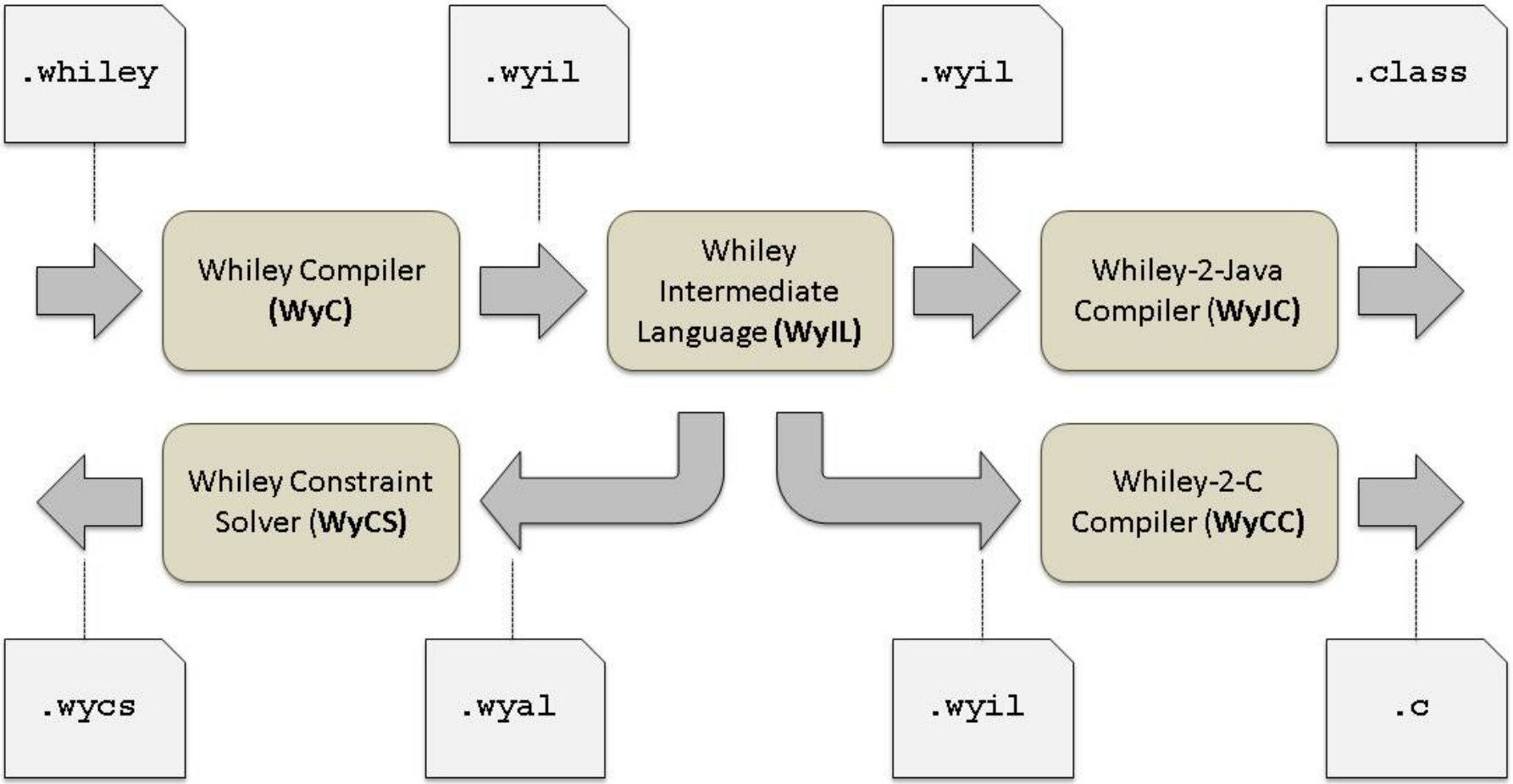
test.whiley

```
type nat is (int n) where n >= 0

function sum([nat] xs) → nat:
    int r = 0
    int i = 0
    while i < |xs| where i >= 0 && r >= 0:
        r = r + xs[i]
    return r
```

- Currently developed at VUW by David J. Pearce
- Currently, **106 KLOC**, spread over **270 classes**
- See: `http://whiley.org`, `http://www.ohloh.net/p/whiley`

Example: the Whiley Compiler



Example: the Wyvern Compiler

- Currently developed in collaboration between CMU (Jonathan Aldrich) and VUW (Alex Potanin)
- General Description: `http://www.cs.cmu.edu/~aldrich/securemobileweb/spec-rationale.html`
- Implementation: `https://github.com/wyvernlang`

Course Overview

- **Lectures**
  - Tuesday and Friday 10:00am to 10:50am in CO228 and FT77 306
- **People**
  - Dr David J. Pearce (course coordinator and lecturer), CO231, 463 5833, djp@ecs.vuw.ac.nz
  - A/Prof Lindsay Groves (lecturer), CO257, 463 5656, lindsay@ecs.vuw.ac.nz

Class Representative Election!!

Assessment

- **Assignment 1 (10%)** — Parsing and Interpretation (due 21st of March)
- **Assignment 2 (10%)** — Type Checking (due 11th of April)
- **Assignment 3 (10%)** — Java Bytecode (due 9th of May)
- **Assignment 4 (10%)** — x86 Machine Code (due 30th of May)
- **Exam (60%)**
  - 3 hours (as usual)
- Mandatory Requirements
  - *At least 40% average across four assignments*
  - *At least 40% on exam*
- Late Penalties:
  - Late work will be penalised 10% per weekday after the deadline
  - Each student has three "late days"

Recommended Books

- There is **no set text**, but the following are recommended:

  Modern Compiler Implementation in Java, Andrew Appel. (closed reserve)

  Engineering a Compiler, Keith D. Cooper and Linda Toczon. See Chapter 8. [1 copy in library]

  Compilers: Principles, Techniques and Tools, Alfred V. Aho, Ravi Sethi and Jeffrey D. Ullman. See Chapter 10. [1 copy in library]

  Advanced Compiler Design and Implementation, Steve S. Muchnick. See Chapter 9.

  Optimizing Compilers for Modern Architectures, Randy Allen and Ken Kennedy. See Chapter 4.4 and 11.

Why SWEN430?

Why should I take SWEN430?

- To learn **how compilers work**
- To create **a working compiler for a realistic imperative language**
- To learn **Java Bytecode** and **x86 Assembly**
- To improve your **programming skills**

The While Language

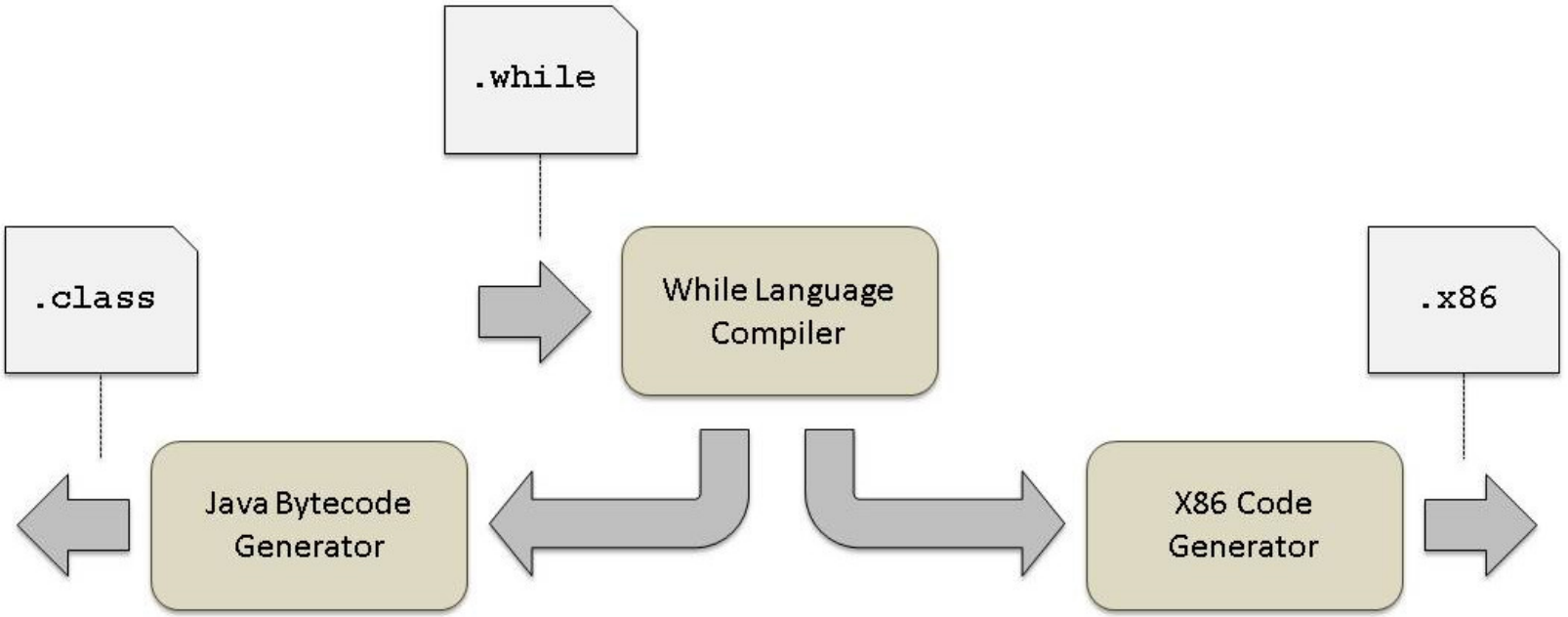
test.while

```
type Point is {int x, int y}

Point move(Point p, int dx, int dy) {
  return {x: p.x + dx, y: p.y + dy};
}
```

- A **simple** imperative language
- **Statements**: for, while, if, switch
- **Expressions**: binary, unary, invocation
- **Types**: bool, int, real, strings, lists, records

While Language Compiler



- Lack of modules / imports will **simplify internals**
- **No intermediate language**: code generation directly off AST
- Targets: **Java Bytecode** and **x86 Assembly Language**