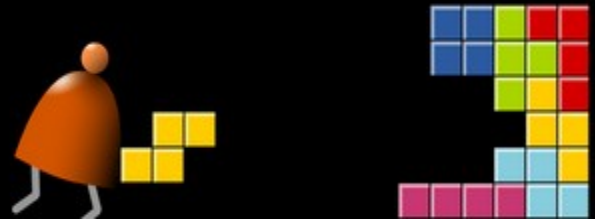


Chapter 7

Virtual Machine, Part I

These slides support chapter 7 of the book

The Elements of Computing Systems

By Noam Nisan and Shimon Schocken

MIT Press

Hello World

Jack program

```
// First example in Programming 101
class Main {
    function void main() {
        do Output.printString("Hello World!");
        do Output.println(); // New line.
        return;
    }
}
```

Hello World Below

Jack program

```
// First example in Programming 101
class Main {
    function void main() {
        do Output.printString("Hello World!");
        do Output.println(); // New line.
        return;
    }
}
```

Issues:

- Program execution
- Writing on the screen
- Handling `class`, `function` ...
- Handling `do`, `while`, ...
- function call and return
- operating system
- ...

Q: How can high-level programmers ignore all these issues?

Hello World Below

Jack program

```
// First example in Programming 101
class Main {
    function void main() {
        do Output.printString("Hello World!");
        do Output.println(); // New line.
        return;
    }
}
```

abstraction

Q: How can high-level programmers ignore all these issues?

A: They treat the high-level language as an ***abstraction***.

Hello World Below

Jack program

```
// First example in Programming 101
class Main {
    function void main() {
        do Output.printString("Hello World!");
        do Output.println(); // New line.
        return;
    }
}
```

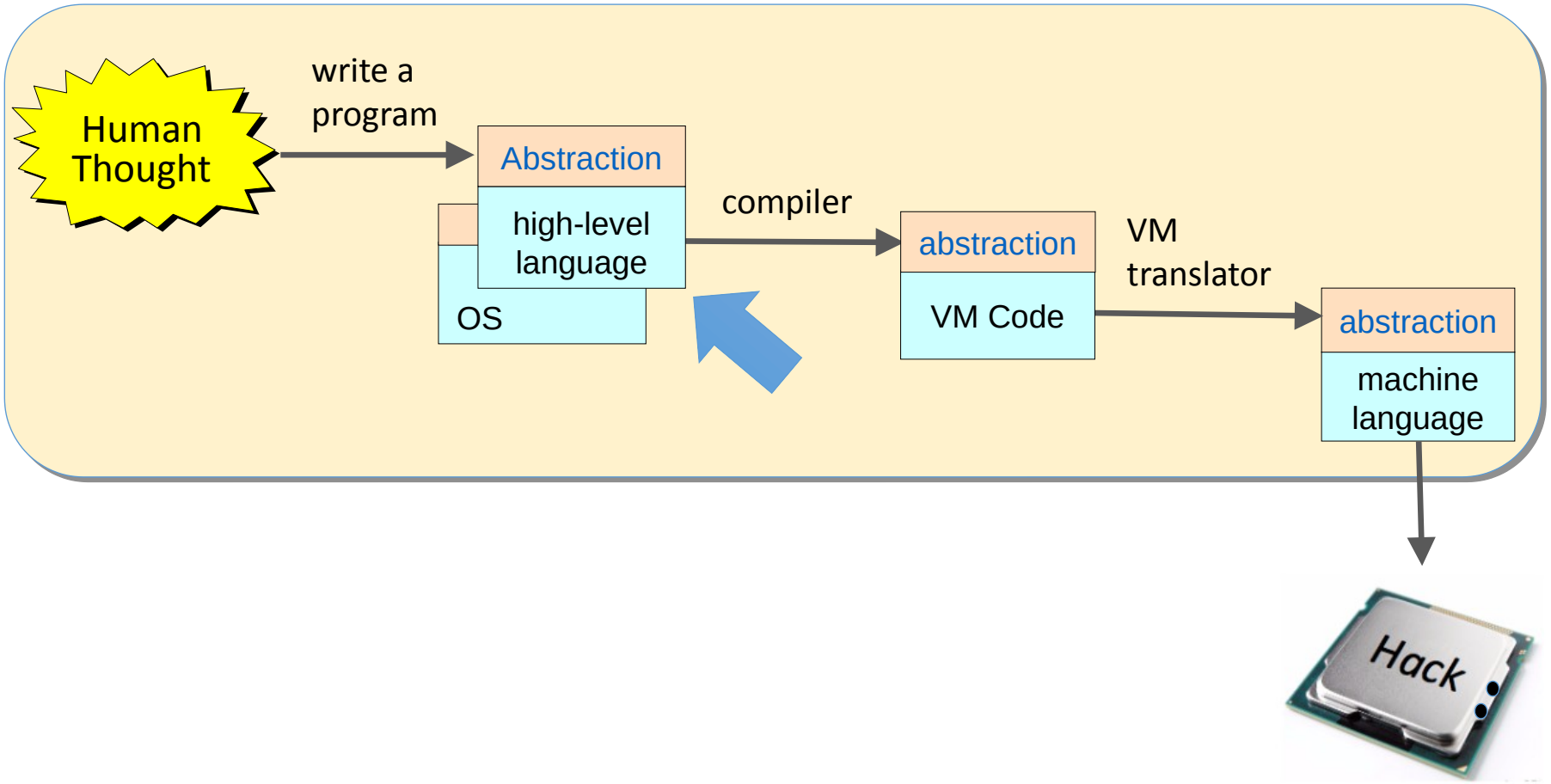
abstraction

Q: What makes the abstraction work?

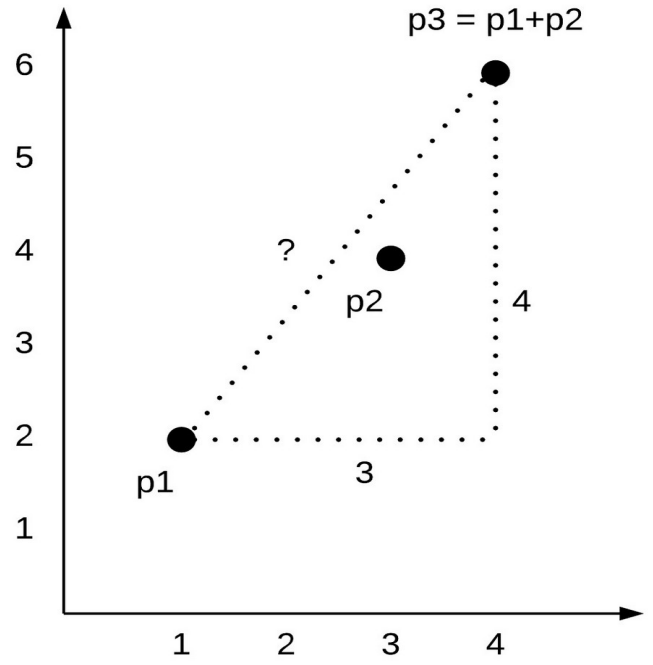
A:
- Assembler
- Virtual machine
- Compiler
- Operating system

Chapters 6-12

The big picture



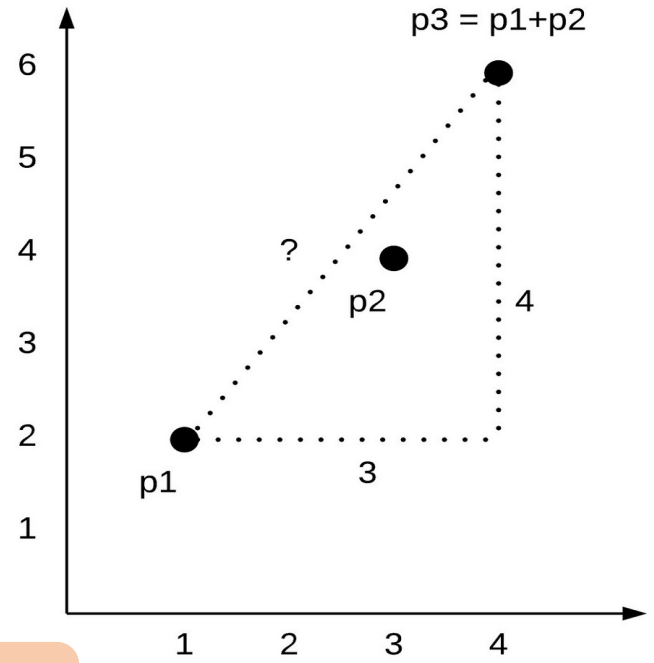
High-level programming




```
class Point {    // API
   /** Constructs a new point with the given coordinates */
   constructor Point new(int ax, int ay) {}
   /** Returns the point which is this point plus the other point */
   method Point plus(Point other) {}
   /** Cartesian distance between this and the other point */
   method int distance(Point other) {
   /** Prints this point, as (x,y) */
   method void print() {
  ...
}
```

High-level programming

```
/** Demo: working with Point objects */
class Main {
  function void main() {
```

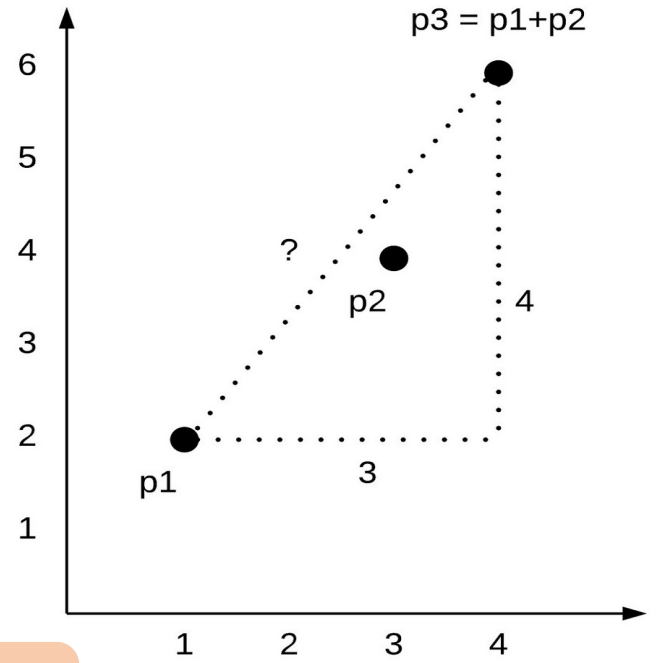


Written in the Jack language

```
class Point {    // API
   /** Constructs a new point with the given coordinates */
   constructor Point new(int ax, int ay) {}
   /** Returns the point which is this point plus the other point */
   method Point plus(Point other) {}
   /** Cartesian distance between this and the other point */
   method int distance(Point other) {
   /** Prints this point, as (x,y) */
   method void print() {
  ...
}
```

High-level programming

```
/** Demo: working with Point objects */
class Main {
  function void main() {
     var Point p1, p2, p3;
     let p1 = Point.new(1,2);
     let p2 = Point.new(3,4);
     let p3 = p1.plus(p2);
     do p3.print();
```



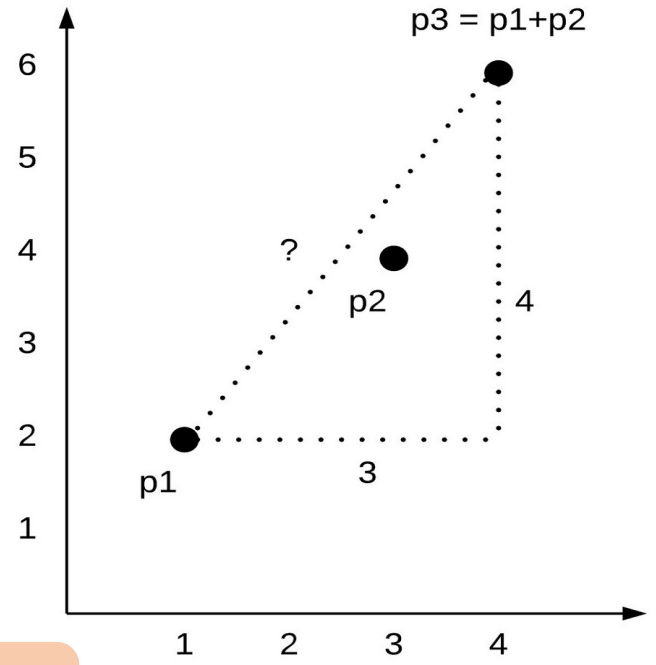
Written in the Jack language

```
class Point {    // API
   /** Constructs a new point with the given coordinates */
   constructor Point new(int ax, int ay) {}
   /** Returns the point which is this point plus the other point */
   method Point plus(Point other) {}
   /** Cartesian distance between this and the other point */
   method int distance(Point other) {
   /** Prints this point, as (x,y) */
   method void print() {
  ...
}
```

High-level programming

```
/** Demo: working with Point objects */
class Main {
  function void main() {
     var Point p1, p2, p3;
     let p1 = Point.new(1,2);
     let p2 = Point.new(3,4);
     let p3 = p1.plus(p2);
     do p3.print();
     do Output.println();
     do Output.printInt(p1.distance(p3));
```

Written in the Jack language

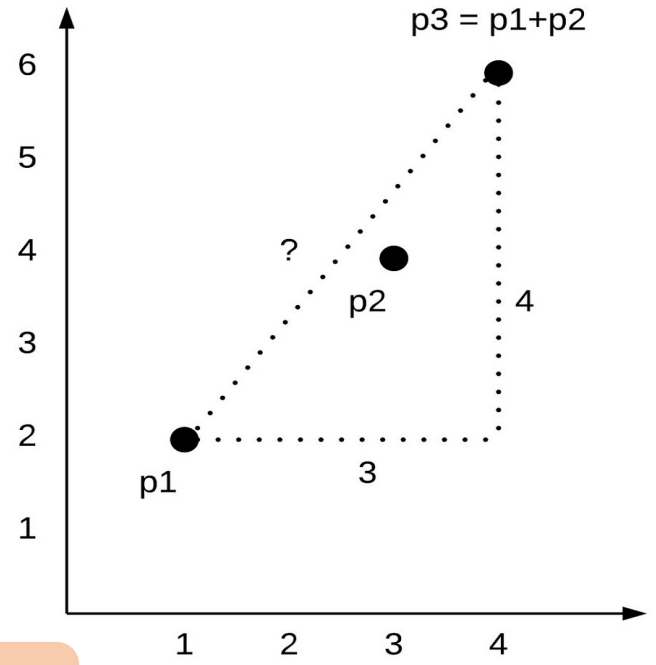



```
class Point {    // API
   /** Constructs a new point with the given coordinates */
   constructor Point new(int ax, int ay) {}
   /** Returns the point which is this point plus the other point */
   method Point plus(Point other) {}
   /** Cartesian distance between this and the other point */
   method int distance(Point other) {
   /** Prints this point, as (x,y) */
   method void print() {
  ...
}
```

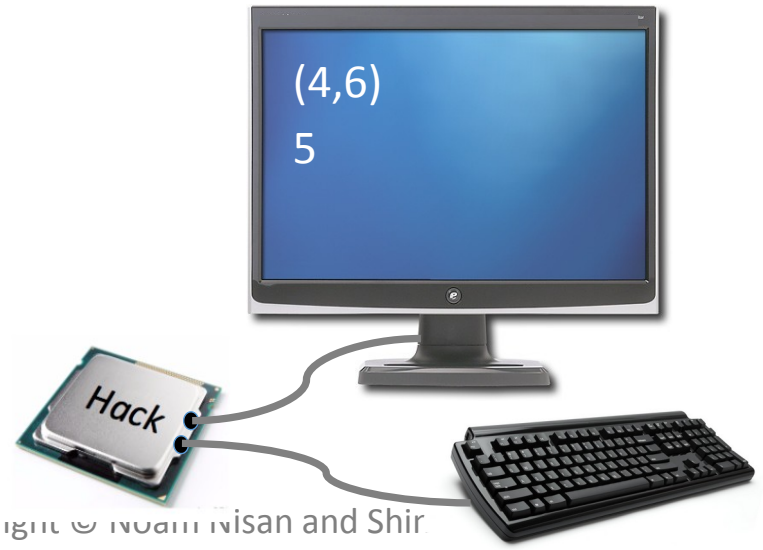
High-level programming

```
/** Demo: working with Point objects */
class Main {
  function void main() {
     var Point p1, p2, p3;
     let p1 = Point.new(1,2);
     let p2 = Point.new(3,4);
     let p3 = p1.plus(p2);
     do p3.print();
     do Output.println();
     do Output.printInt(p1.distance(p3));
```

Written in the Jack language



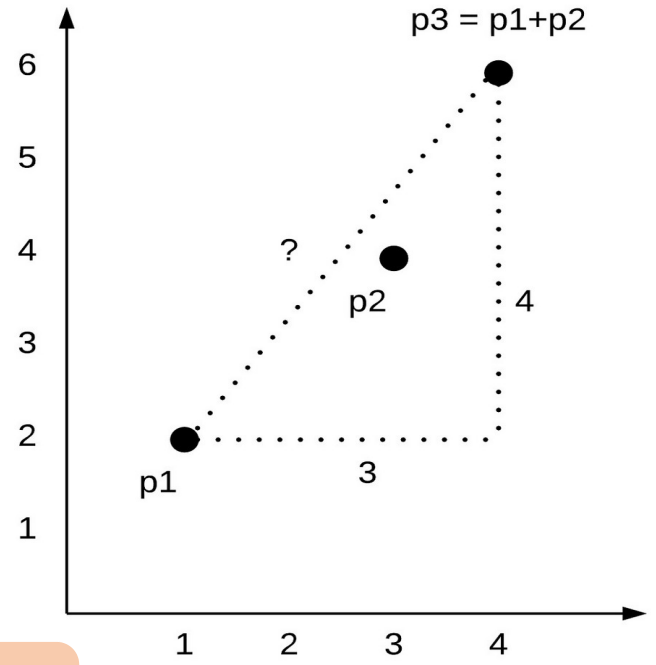

```
class Point {    // API
   /** Constructs a new point with the given coordinates */
   constructor Point new(int ax, int ay) {}
   /** Returns the point which is this point plus the other point */
   method Point plus(Point other) {}
   /** Cartesian distance between this and the other point */
   method int distance(Point other) {
   /** Prints this point, as (x,y) */
   method void print() {
  ...
}
```



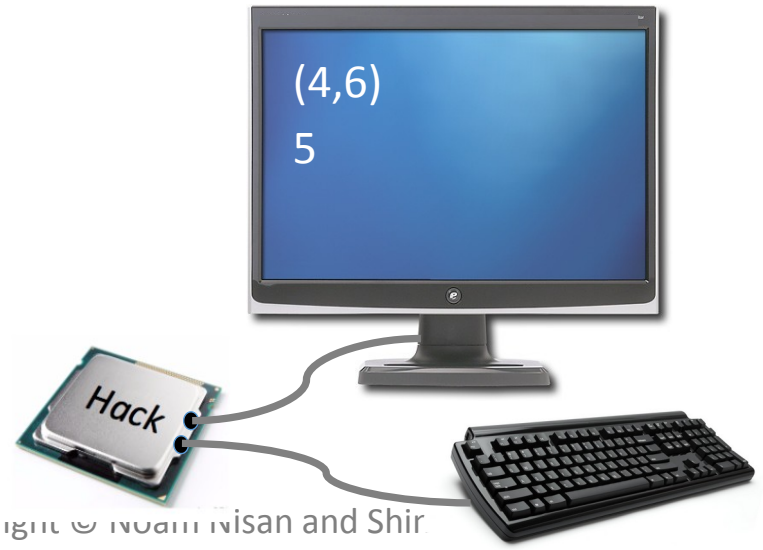
High-level programming

```
/** Demo: working with Point objects */
class Main {
  function void main() {
     var Point p1, p2, p3;
     let p1 = Point.new(1,2);
     let p2 = Point.new(3,4);
     let p3 = p1.plus(p2);
     do p3.print();
     do Output.println();
     do Output.printInt(p1.distance(p3));
     return;
  }
}
```

Written in the Jack language

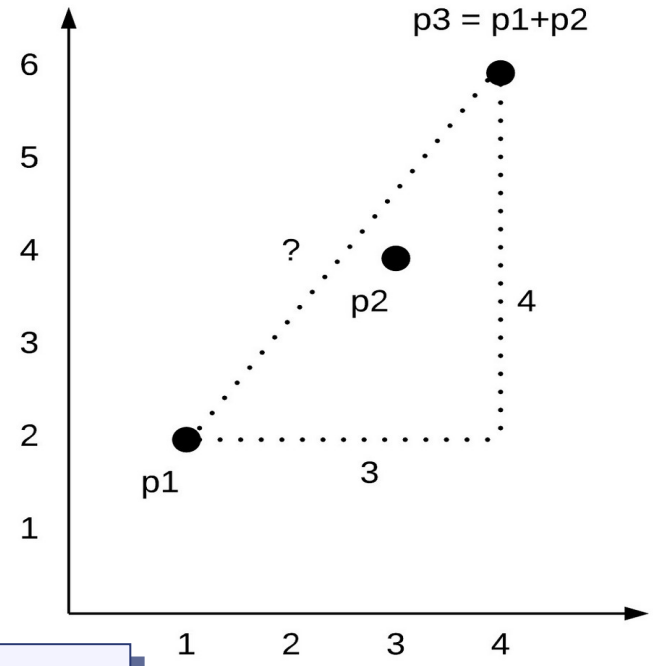



```
class Point {    // API
   /** Constructs a new point with the given coordinates */
   constructor Point new(int ax, int ay) {}
   /** Returns the point which is this point plus the other point */
   method Point plus(Point other) {}
   /** Cartesian distance between this and the other point */
   method int distance(Point other) {
   /** Prints this point, as (x,y) */
   method void print() {
  ...
}
```

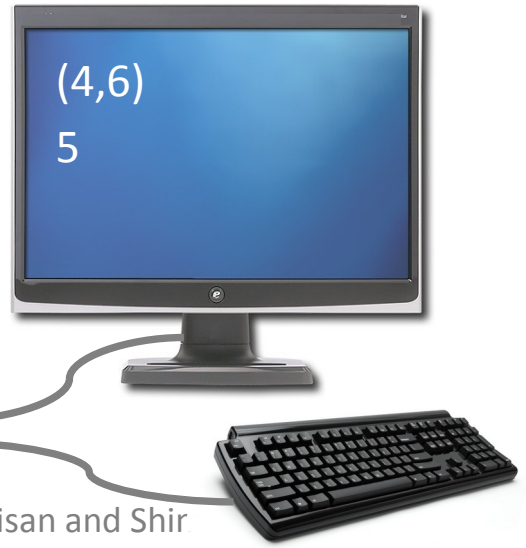


High-level programming

```
/** Demo: working with Point objects */
class Main {
  function void main() {
     var Point p1, p2, p3;
     let p1 = Point.new(1,2);
     let p2 = Point.new(3,4);
     let p3 = p1.plus(p2);
     do p3.print();
     do Output.println();
     do Output.printInt(p1.distance(p3));
     return;
  }
}
```




```
/** Represents a Point */
class Point {
  field int x, y;
  static int pointCount;
  /** Constructs a new point */
  constructor Point new(int ax, int ay) {
      let x = ax;
      let y = ay;
      let pointCount = pointCount + 1;
      return this;
  }
  // More Point methods...
}
```



High-level programming / `Point` class

```
/** Represents a Point; Stored in the file Point.jack */
class Point {

    field int x, y; // the coordinates of this point

    static int pointCount; // the number of Point objects constructed so far

    /** Constructs a new point with the given coordinates */
    constructor Point new(int ax, int ay) {
        let x = ax;
        let y = ay;
        let pointCount = pointCount + 1;
        return this;
    }

    /** Returns the x coordinate of this point */
    method int getx() { return x; }

    /** Returns the y coordinate of this point */
    method int gety() { return y; }

    /** Returns the number of Point objects constructed so far */
    method int getPointCount() { return pointCount; }

    /** Returns the point which is this point plus the other point */
    method Point plus(Point other){
        return Point.new(x + other.getx(), y + other.gety());
    }
```

High-level programming / `Point` class

```
/** Represents a Point; Stored in the file Point.jack */
class Point {
    field int x, y; // the coordinates of this point
    static int pointCount; // the number of Point objects constructed so far

    // The methods from the previous slide...

    /** Returns the Cartesian distance between this and the other point */
    method int distance(Point other) {
        var int dx, dy;
        let dx = x - other.getx();
        let dy = y - other.gety();
        return Math.sqrt((dx*dx)+ (dy*dy));
    }

    /** Prints this point, as (x,y) */
        method void print() {
        do Output.printString("(");
        do Output.printInt(x);
        do Output.printString(",");
        do Output.printInt(y);
        do Output.printString(")");
        return;
    }
}  // class Point ends here
```

Jack has the look and feel of a typical high-level, object-based language

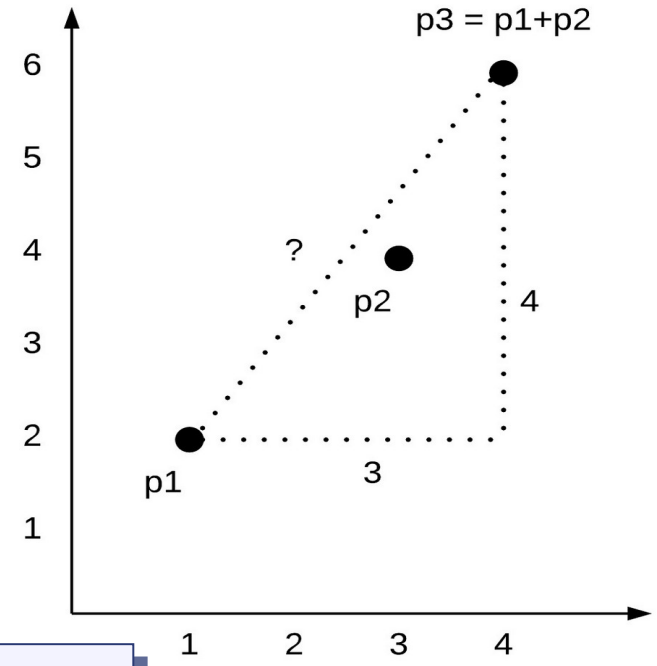
Recap

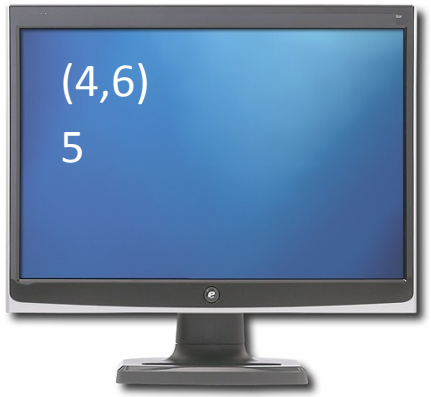
```
/** Demo: working with Point objects */
class Main {
  function void main() {
     var Point p1, p2, p3;
     let p1 = Point.new(1,2);
     let p2 = Point.new(3,4);
     let p3 = p1.plus(p2);
     do p3.print();
     do Output.println();
     do Output.printInt(p1.distance(p3));
     return;
  }
}
```

Main.jack

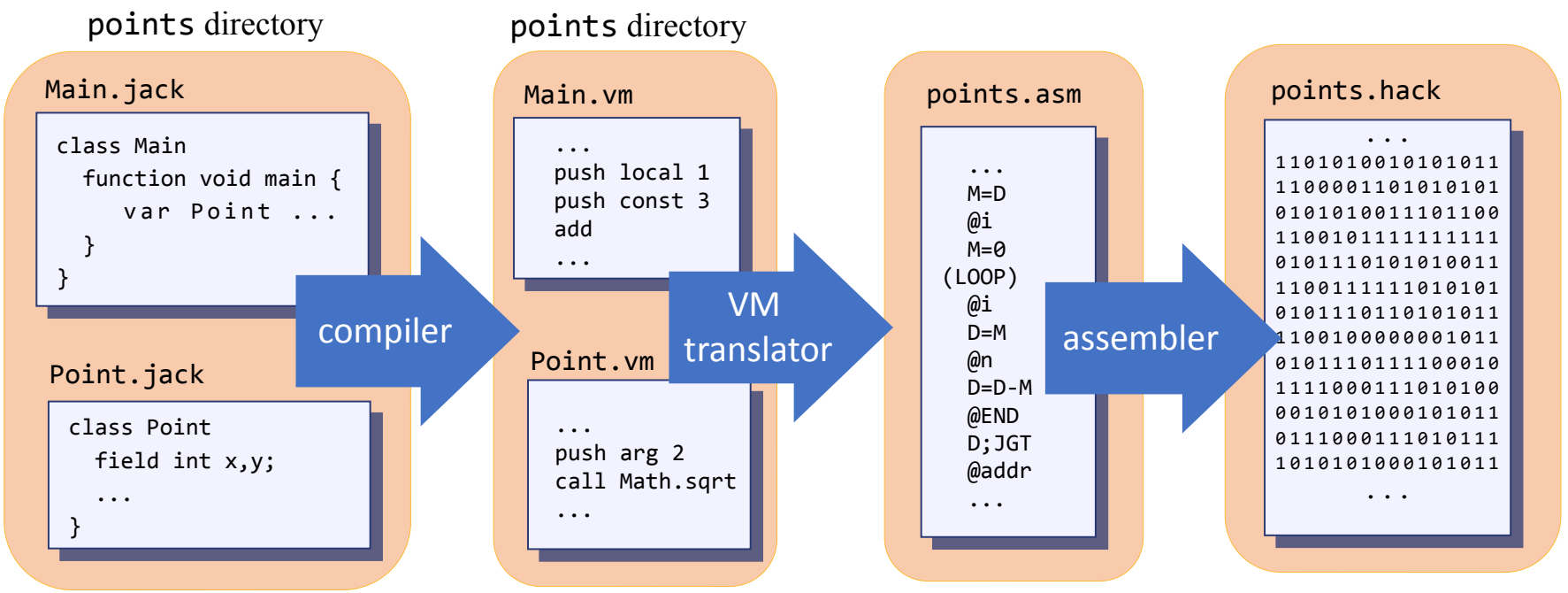
```
/** Represents a Point */
class Point {
  field int x, y;
  static int pointCount;
  /** Constructs a new point */
  constructor Point new(int ax, int ay) {
      let x = ax;
      let y = ay;
      let pointCount = pointCount + 1;
      return this;
  }
  // More Point methods...
}
```

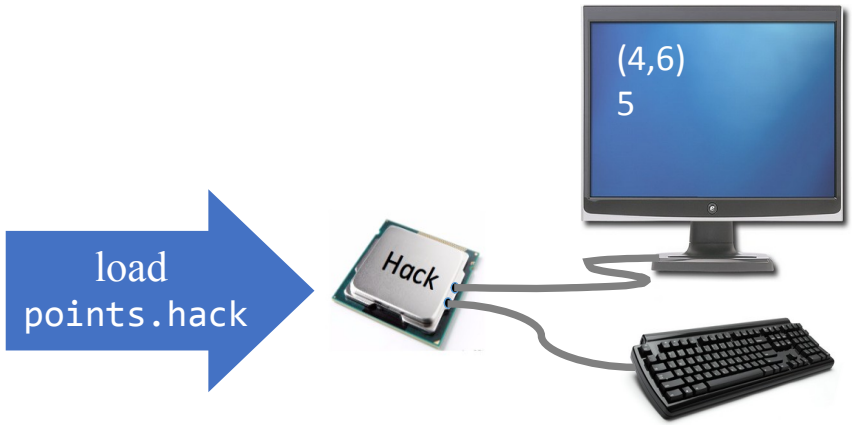
Point.jack



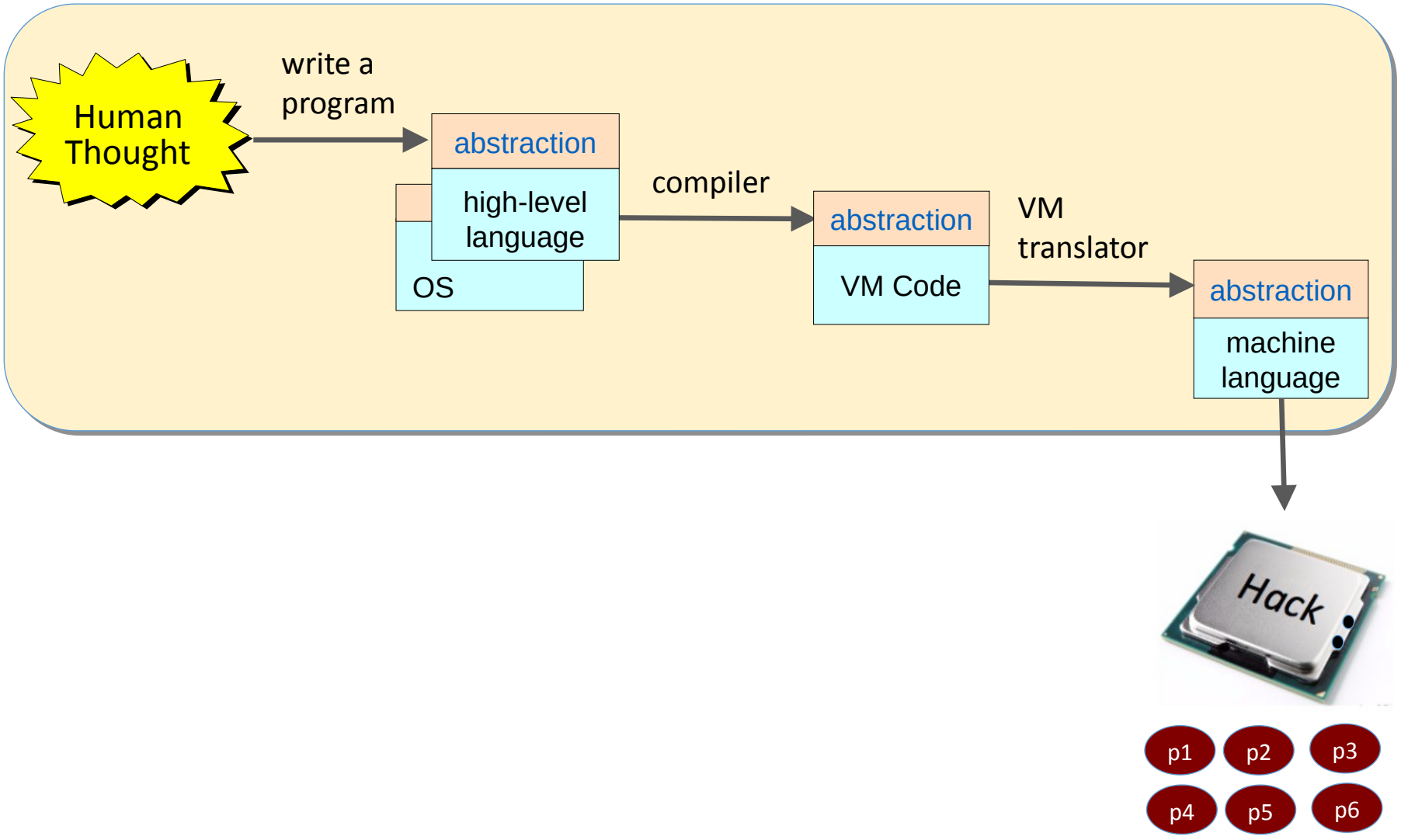


From high-level to low-level

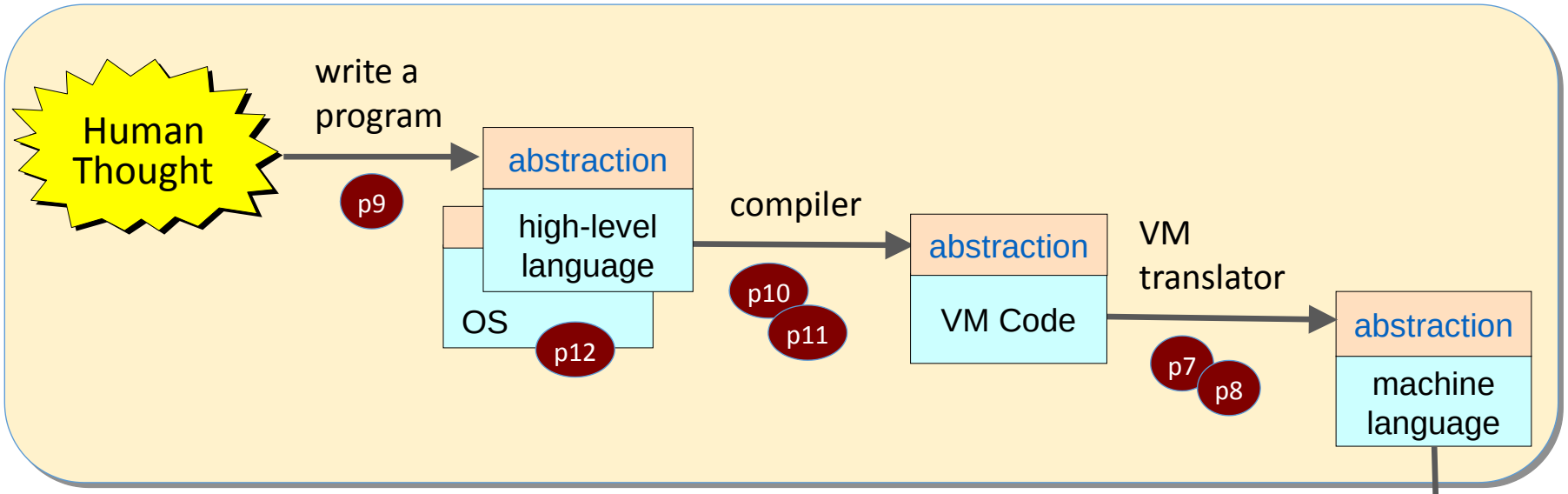




The road ahead



The road ahead



Projects:

p7, p7: building a virtual machine

p9: writing a computer game

p10, p11: developing a compiler

p12: developing an operating system

p1 p2 p3

p4 p5 p6