1 Three kinds of programs

Application          Interpreter                Compiler

appl                 input                   input  xc  output
mach                 interp                  mach

mach

2 Rules

1. A program can run if its implementation (bottom) language is the machine language (floor).

2. A program can run if it is balanced on a running interpreter whose input language matches its implementation languages.

3. A program can be translated by running compiler if the input language of the compiler matches the implementation language of the compiler. The resulting program has the implementation language equal to the compiler output language.

In particular, an interpreter being used as input to a compiler is not running, and thus cannot support any program on it.

3 Examples

These examples show: (1) running the program xtetris; (2) the source code which cannot run; (3) running Eclipse on the JVM; (4) compiling xtetris on the GNU C Compiler.
4 Errors

Each of the following diagrams have something wrong. What?

<table>
<thead>
<tr>
<th>Diagram 1</th>
<th>Diagram 2</th>
<th>Diagram 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Eclipse J</td>
<td>Eclipse J</td>
<td>eclipse.j</td>
</tr>
<tr>
<td>J JVM C</td>
<td>C gcc M</td>
<td>M</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

5 Puzzles

Draw diagrams for the following situations.

1. If one has the pieces of the last error diagram: Eclipse in Java Byte Code, the JVM implemented in C and gcc, how can one run Eclipse?

2. If one has gcc, a Pascal interpreter written in C (pi.c) and a Pascal program for printing grades (grades.p), how can one run it?

3. If one has a Pascal compiler written in Pascal (pc.p) that generates C as well as what is in the previous problem, how can one get a native executable for the grades program?

4. With all the previous pieces, How does one package up a native Pascal compiler that generates machine code?

5. Assuming that one has a working but slow and not optimizing compiler (cc) and the source for an optimizing compiler (gcc.c), how does one get a fast optimizing compiler?

6. The Cool compiler is written in Cool and generates MIPS. Cool (L) is a subset of Scala (S). We have a Scala compiler that generates Java byte code that is in Java byte code. We have a running JVM on Intel’s x86. We have spim.c, a MIPS interpreter written in C. Finally we have the gcc executable. There are two radically different ways we can run primes.cool. What are they? Here are your pieces: