2 Bytecode dispatch techniques

Switch-based

```
while (true)
  switch (*ip++)
    case add:
      <add>
      break;
    :
```

+ Simple
+ Portable
÷ Slow

Direct call threading

```
while (1)
  (*ip++)();
```

+ portable
+ faster
÷ Large code size
÷ Creates and destroys stack sections
÷ code translation

Direct threading

```
goto **ip++;
add:
  <add>
  goto **ip++;
  :
```

+ fast
÷ not portable
÷ large code size
÷ code translation

Indirect threading

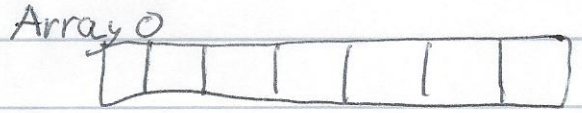
```
t = {&&add, ... };
goto *t[*ip++];
add:
  <add>
  goto *t[*ip++];
  :
```

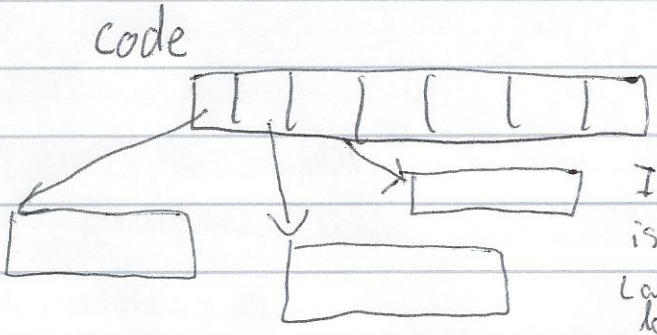
+ relative fast
+ no code translation
÷ Less portable

Experiment viser at GCC bruger jump-tables:
⇒ indirect threading.

JIT

We implemented Jit on the universal machine.





Initially one instruction is compiled.
Later all instructions until loadProgram.

```
if(code[ip] == null)  code[ip] = compile();
if(code[ip] != null) {code[ip](); continue;}
switch( )
  :
```

Approx 2x faster.