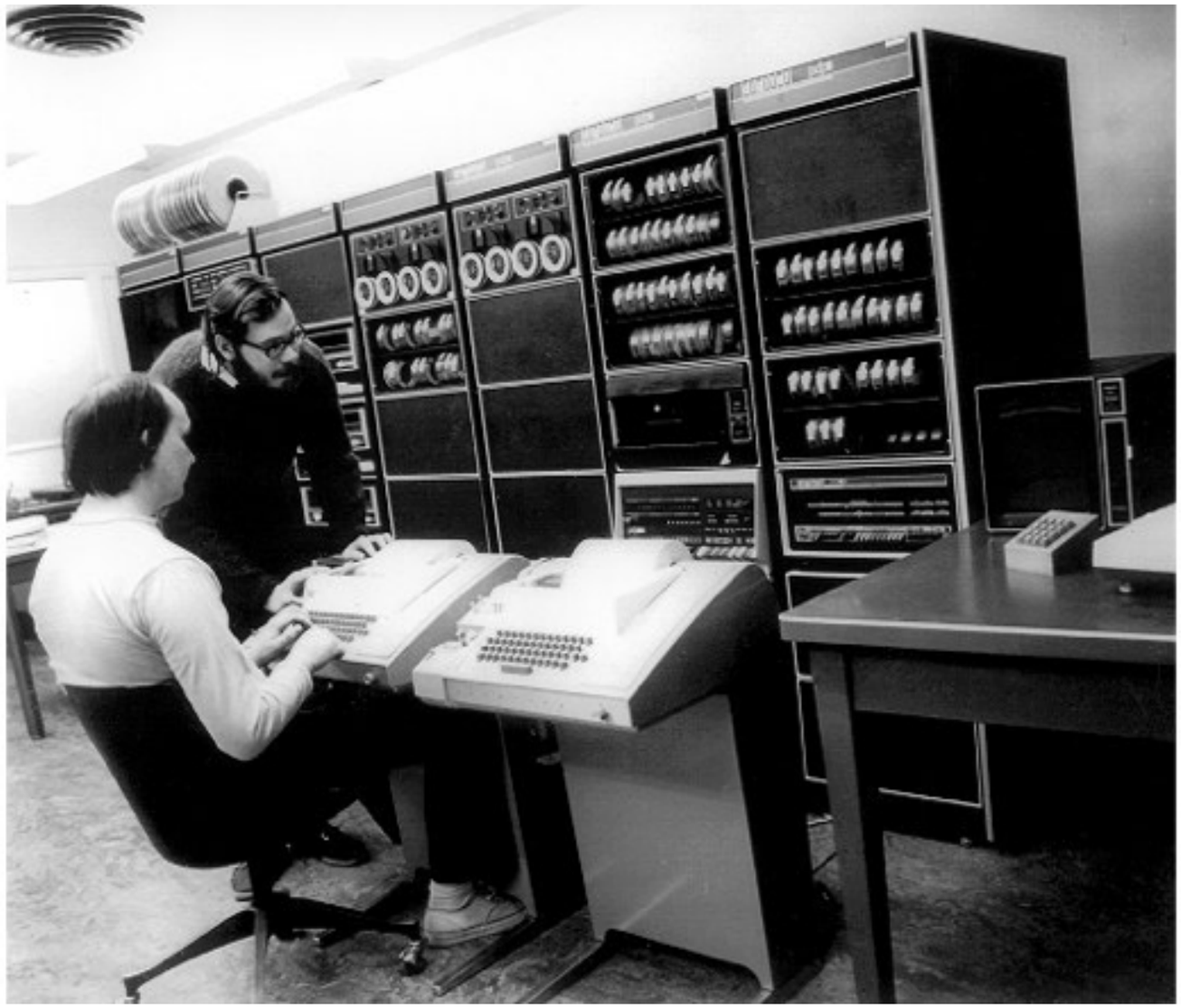
Spirit of C

"to get a deeper understanding of the language"



Deep C - a 3 day course
Jon Jagger & Olve Maudal

trust the programmer

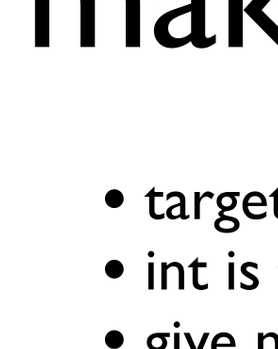
- BCPL ->B ->C
- let them do what needs to be done
- the programmer is in charge, not the compiler

"The philosophy of BCPL is not one of the tyrant who thinks he knows best and lays down the law on what is not allowed; rather BCPL acts more as a servant offering his services to the best of his ability and without complaint, even when confronted with apparent nonsense.
The programmer is always assumed to know what he is doing and is not hemmed in by petty restrictions."

page 5, BCPL the language and its compiler
by Martin Richards and Colin Whitby-Strevens

make it fast even if non portable

- target efficient code generation
- int is the natural word size of the machine
- give maximum opportunities to the compiler, eg sequence points



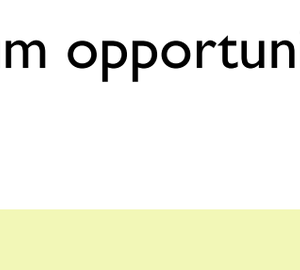
Wibble.java

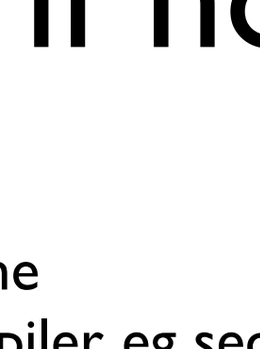
Wibble.class

JVM

0100100010

any machine





Wibble.c

01001001101

machine

make it fast even if non portable

foo.c

```
#include <stdio.h>

int main(void)
{
    int v[] = { 0,2,4,6,8 };
    int i = 1;
    int n = i + v[i++] + v[++i];
    printf("%d\n", n);
}
```

```
$ gcc foo.c && ./a.out
12
```

gcc 4.2.1

```
$ clang foo.c && ./a.out
11
```

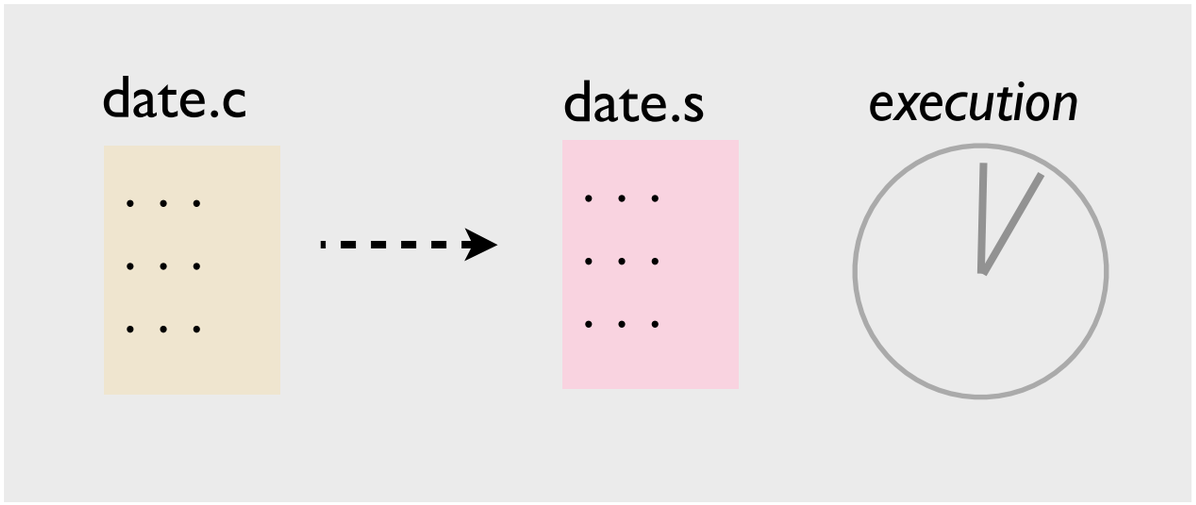
clang 4.1

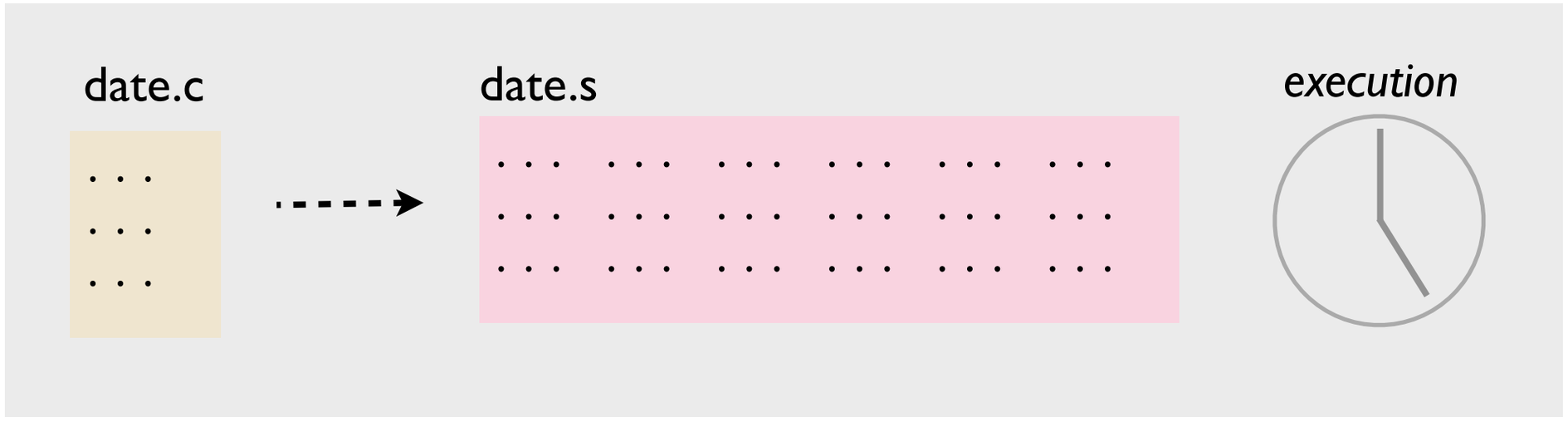
```
$ icc foo.c && ./a.out
13
```

icc 13.0.1

keep the language small & simple

- provide only one way to do an operation
- new inventions are not entertained





rich expressions

- lots of operators
- numerous implicit conversion rules
- expressions freely combine into larger expressions
- allows conciseness

```
i++ i-- () [] . -> (type){list}
++i --i ! ~ (type) * & sizeof _Alignof
* / %    + -    << >>    < <= > >=
== !=    &    ^    |    &&    ||    ?:
= += -= *= /= %= <<= >>= &= ^= |=
,
```

key

buggy code

compiles ok

does not compile

compiles but poor style

c1999 feature

c2011 feature

slide exercise

spirit of C

summary

- trust the programmer
  - let them do what needs to be done
  - the programmer is in charge not the compiler
- make it fast even if non portable
  - target efficient code generation
  - sequence points
  - give maximum opportunities to the compiler
- keep the language small and simple
  - provide only one way to do an operation
  - new inventions are not entertained
  - small amount of code, small amount of assembler
- rich expressions
  - lots of operators
  - expressions freely combine into larger expressions