L3: Naming systems

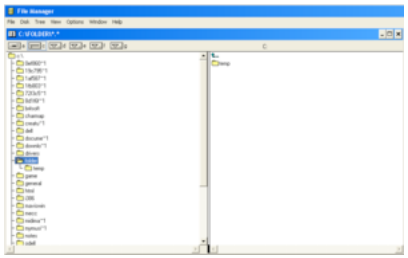
Frans Kaashoek

6.033 Spring 2007

http://web.mit.edu/6.033

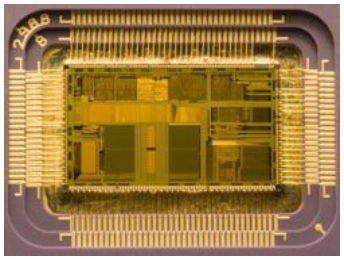
MASSACHUSETTS
INSTITUTE OF
TECHNOLOGY

Fundamental abstractions



- Memory
  - Read/write
- Interpreter
  - Instruction repertoire
  - Environment
  - Instruction pointer
- Communication links
  - Send/receive

(loop (print (eval (read))))



World-wide Web

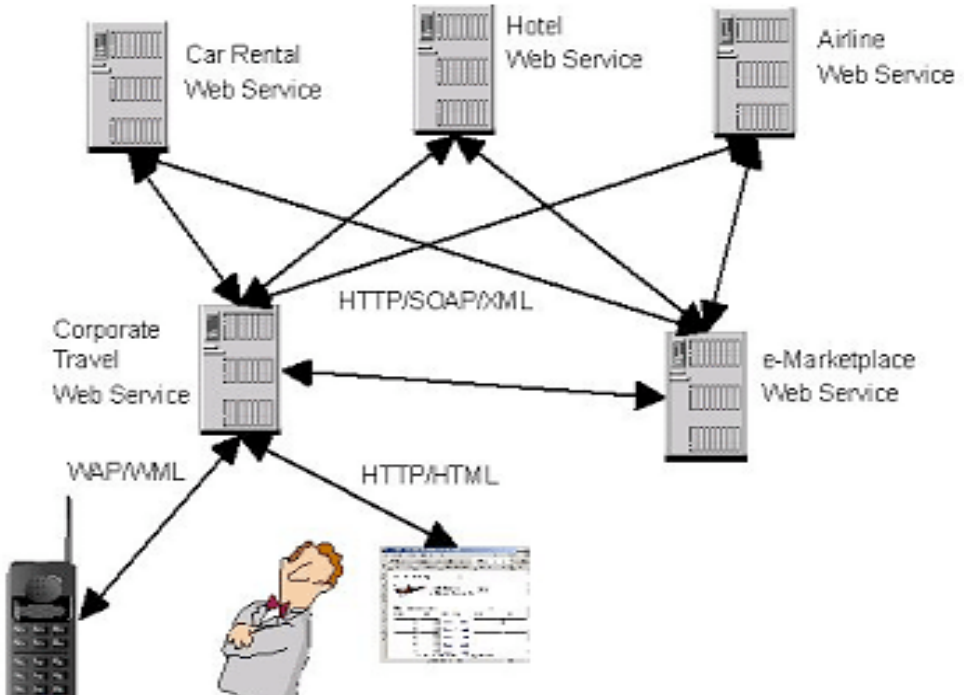
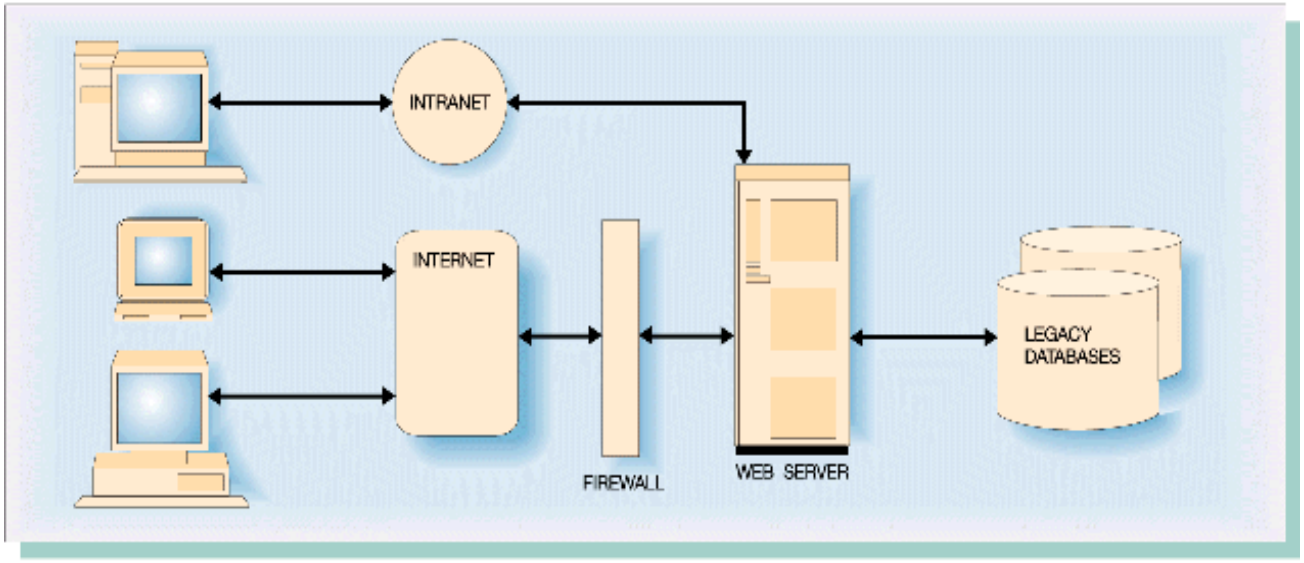


Figure 2 Three-tier architecture



Web names

http://web.mit.edu/6.033

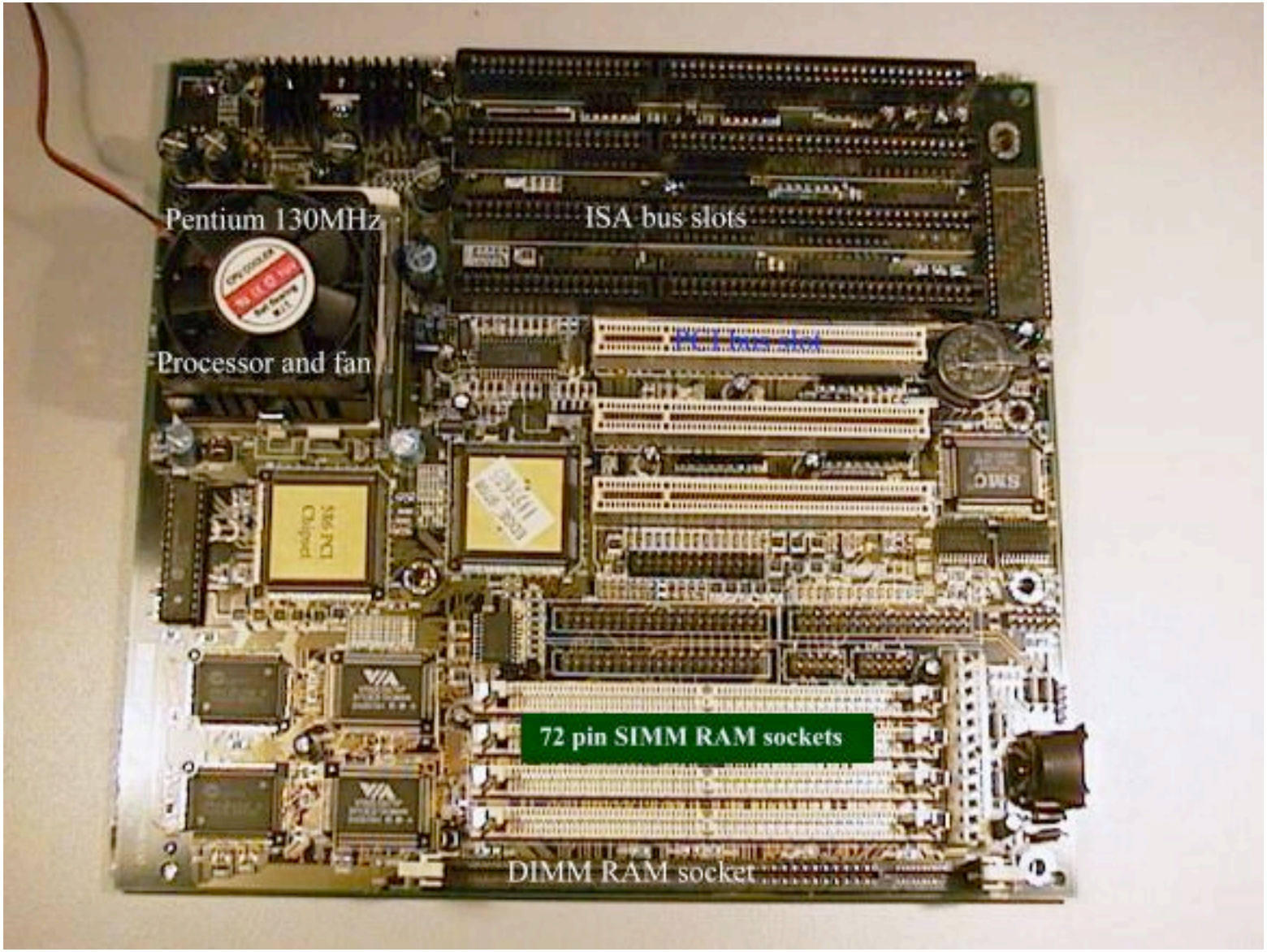
3 naming systems

https://apply.eecs.mit.edu/ex?whatnext=ctlmyfolders&submitted=on&onlyreader=kaashoek

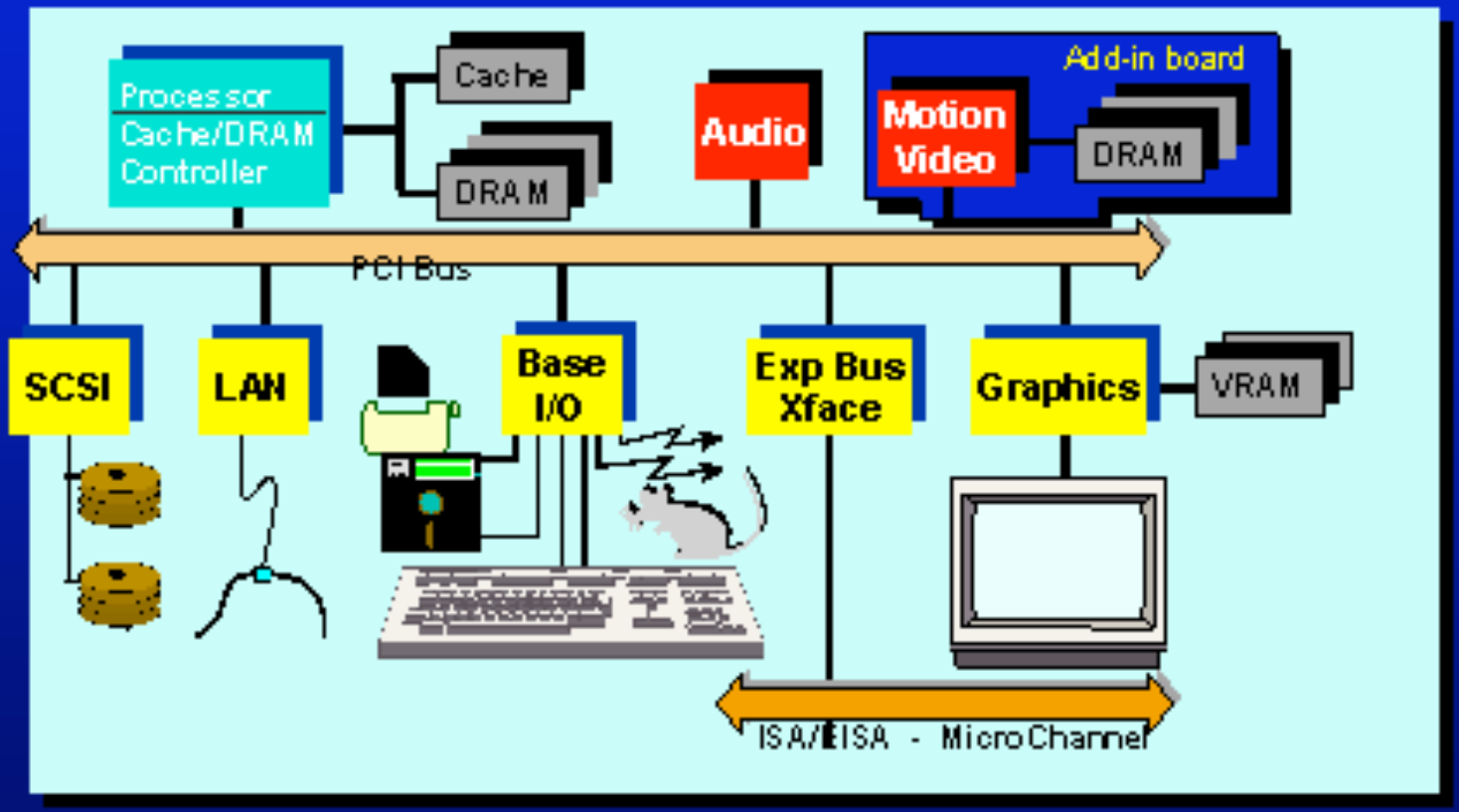
name overloading: user query in the name

ex is a new interpreter

PC board


Pentium 130MHz
Processor and fan
ISA bus slots
72 pin SIMM RAM sockets
DIMM RAM socket

Anatomy Of A Personal Computer



©Courtesy, Shirpal Rawat, Intel Corporation.

Bridge Architecture

Abstract bus picture

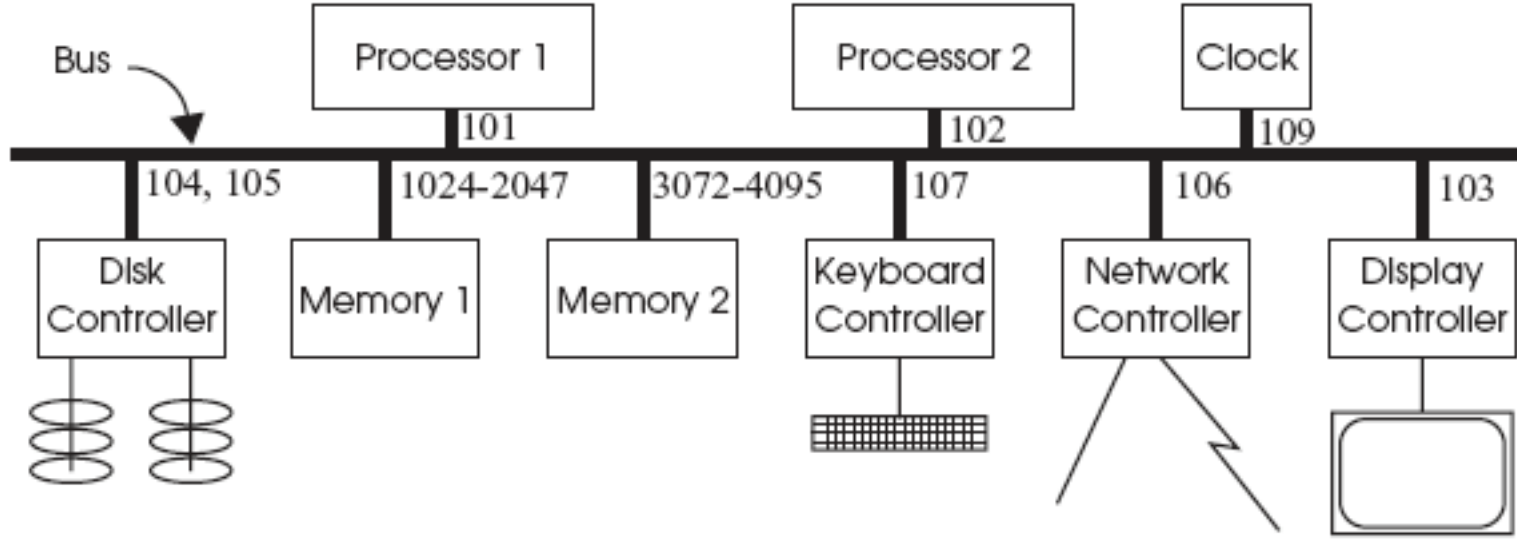


Figure 2–16: A computer with several modules connected by a shared bus. The numbers are the bus addresses to which the attached module responds.

- address is overloaded name with location info
- LOAD 1742, R1

Names

- R1
- 1742
- 18.7.22.69
- web.mit.edu
- http://web.mit.edu/6.033
- 6.033-staff@mit.edu
- amsterdam
- /mit/6.033/www
- foo.c
- .. (as in cd .. or ls ..)
- wc
- (617)253-7149, x37149
- 021-84-2030

Naming example

```
foo (…) {
    int x, y;
    init()
    x = bar(…)
    y = sqrt(x)
    printf (“answer is %d”, y)
}

init() { …}
```

```
0xff00: // foo:
   …
jmp 0xff20 // init
   …
jmp 0xffd0 // bar
   ….
jmp 0xffc0  // sqrt
   …
jmp 0xdc00 // printf
```

Step 1: compilation

```
foo (…) {
    int x, y;
    init();
    x = bar(…);
    y = sqrt(x);
    printf (“answer is %d”, y);
}

init() { …}
```

gcc -c foo.c

foo.o:

- Text:

  0xff00: // foo:
  jmp +20 // init()
  …
  jmp ??? // sqrt
  ….

- Symbol table:

  [init, L, +20]
  [bar, U, ??]
  [sqrt, U, ??]
  [printf, U, ??]

Step 2: linking

- Gcc foo.o bar.o /usr/lib/libm.a /usr/lib/libc.a

foo.o:

- Text:

  0xff00: // foo:
  jmp +20 // init()
  …
  jmp ??? // sqrt
  ….

- Symbol table:

  [init, L, +20]
  [bar, U, ??]
  [sqrt, U, ??]
  [printf, U, ??]

+

libm.a (sqrt.o, foo.o):

sqrt.o:

- Text:

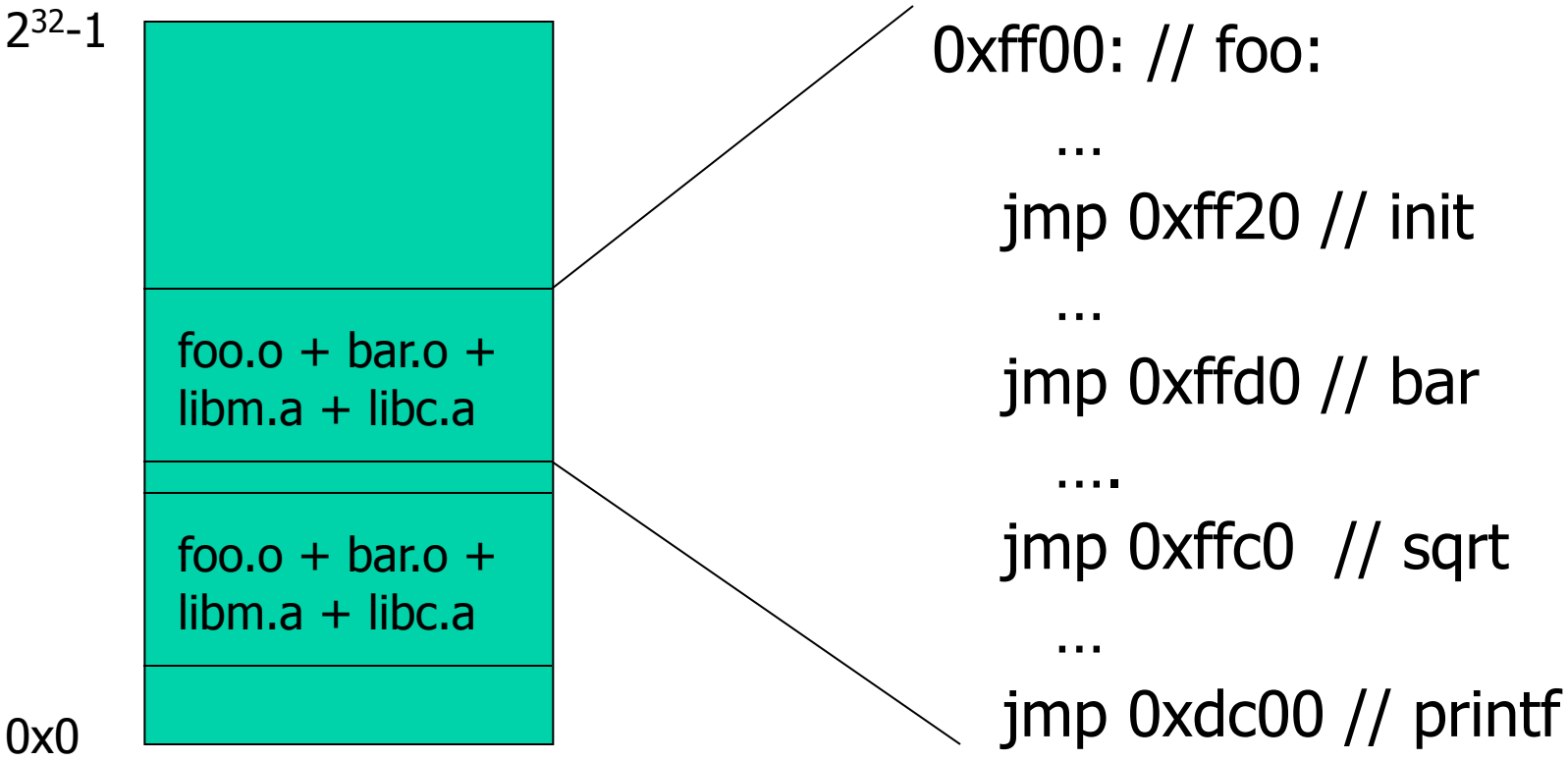
  0xff00: // sqrt:
  jmp +20 // init()
  …

- Symbol table:

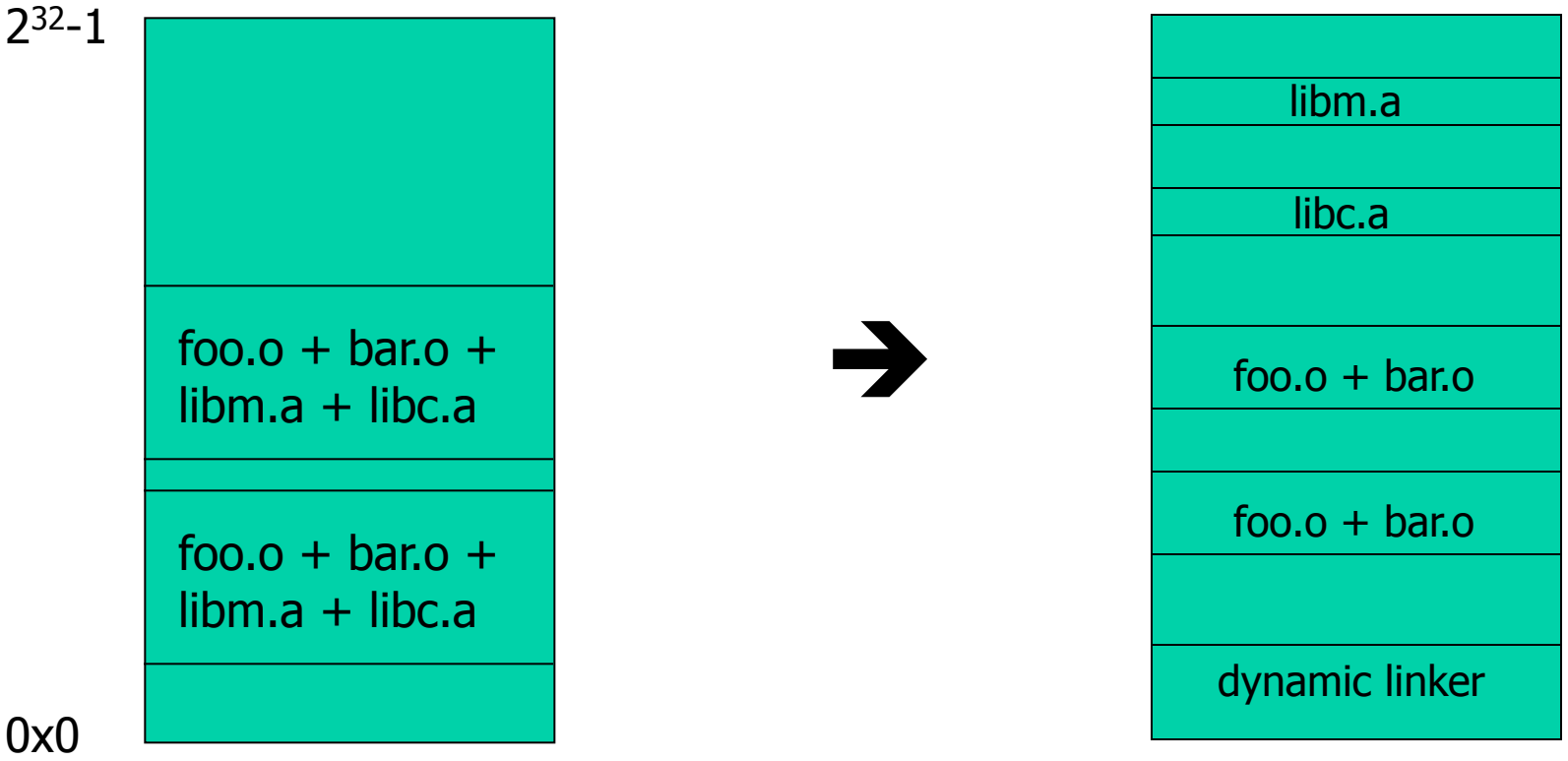
  [init, L, +20]
  [sqrt, G, +100]

Step 3: loading

- Load in memory and adjust addresses:



Sophistication: dynamic linking



- Resolve names at runtime through indirection

Summary

- Understanding a naming system:
  - What is the syntax for name?
  - What are the values?
  - What is the naming resolution algorithm?
  - Where does a name’s context from?