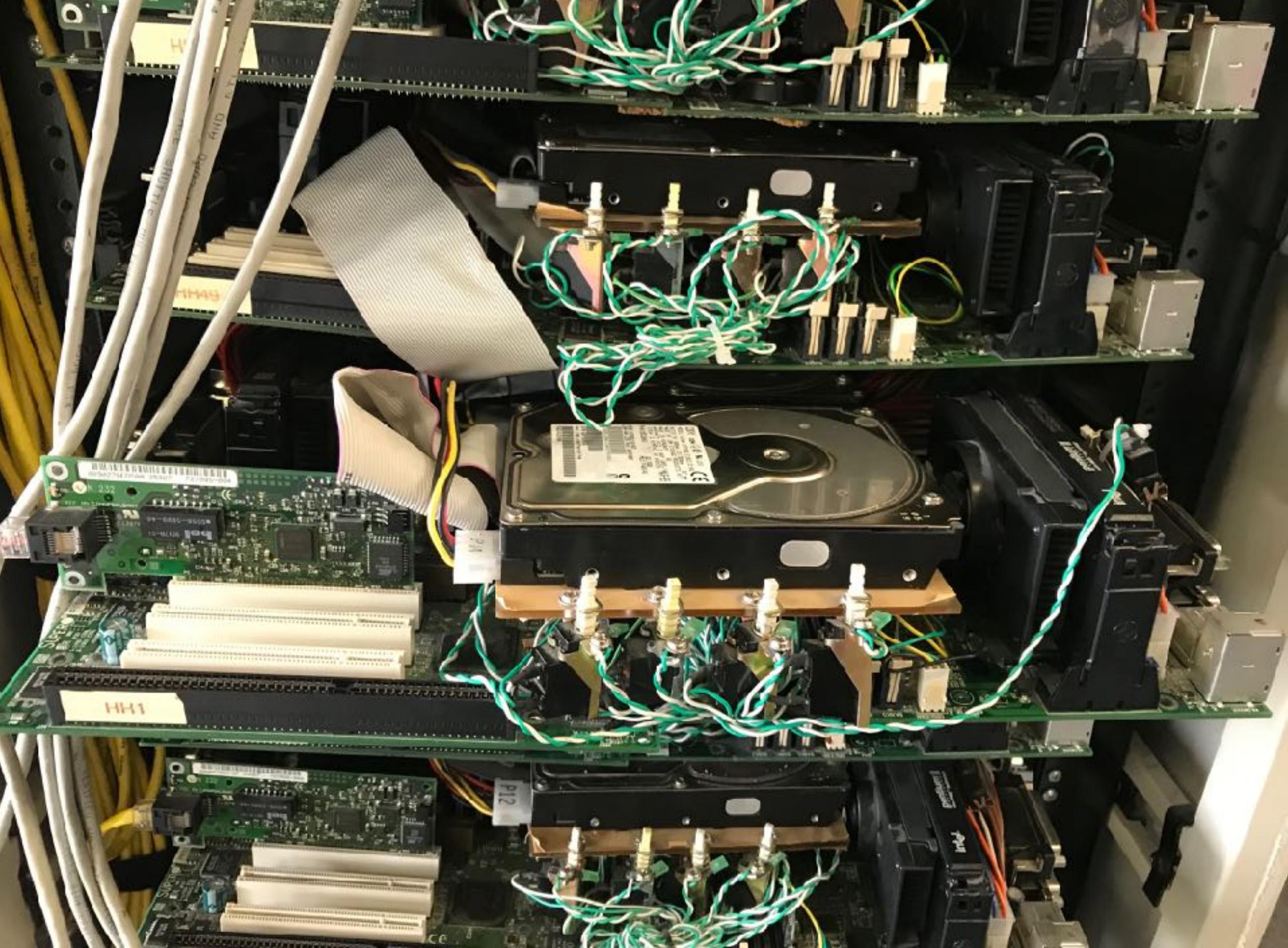
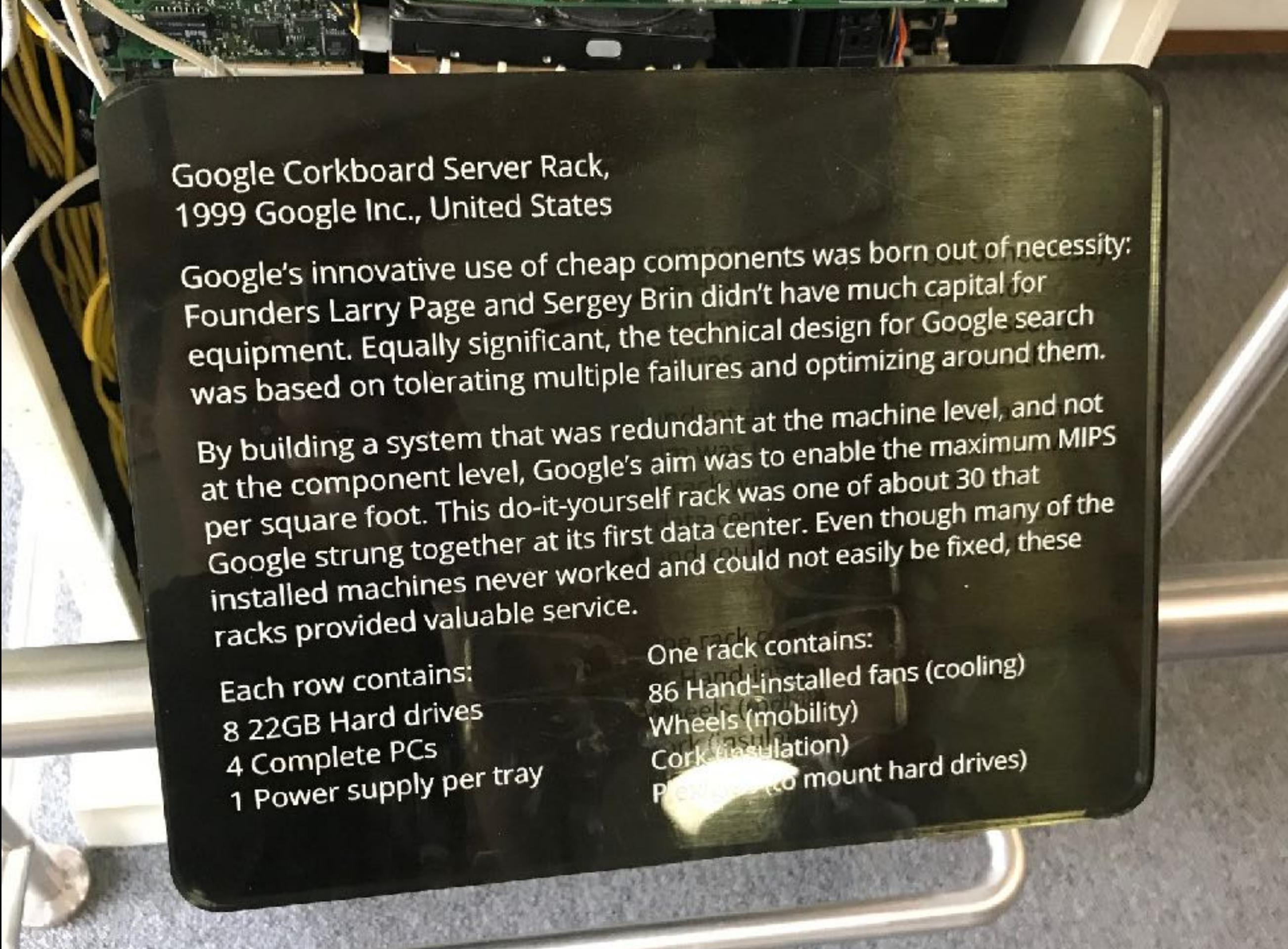




Optical modules
These are common multi-channel passive components used in optical communication networks. One of the main reasons for their popularity is that they can be used in both passive and active optical networks. They are also used in optical fiber communication systems to provide a cost-effective way to increase bandwidth. These optical modules work by passing light through a series of lenses and mirrors to change its path or intensity. They are often used in conjunction with other optical components such as lasers and detectors to perform specific functions in a network.





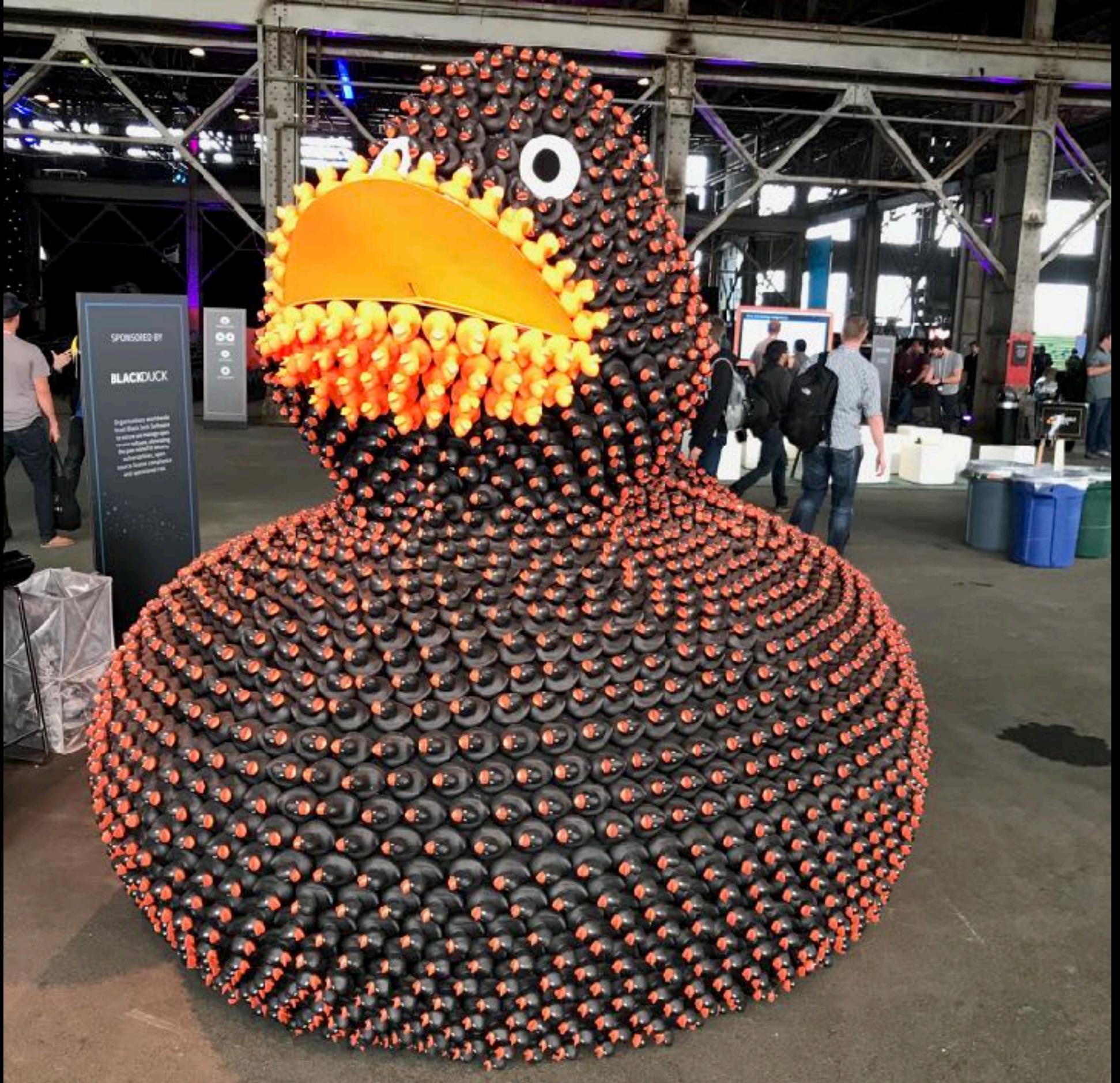
Google Corkboard Server Rack, 1999 Google Inc., United States

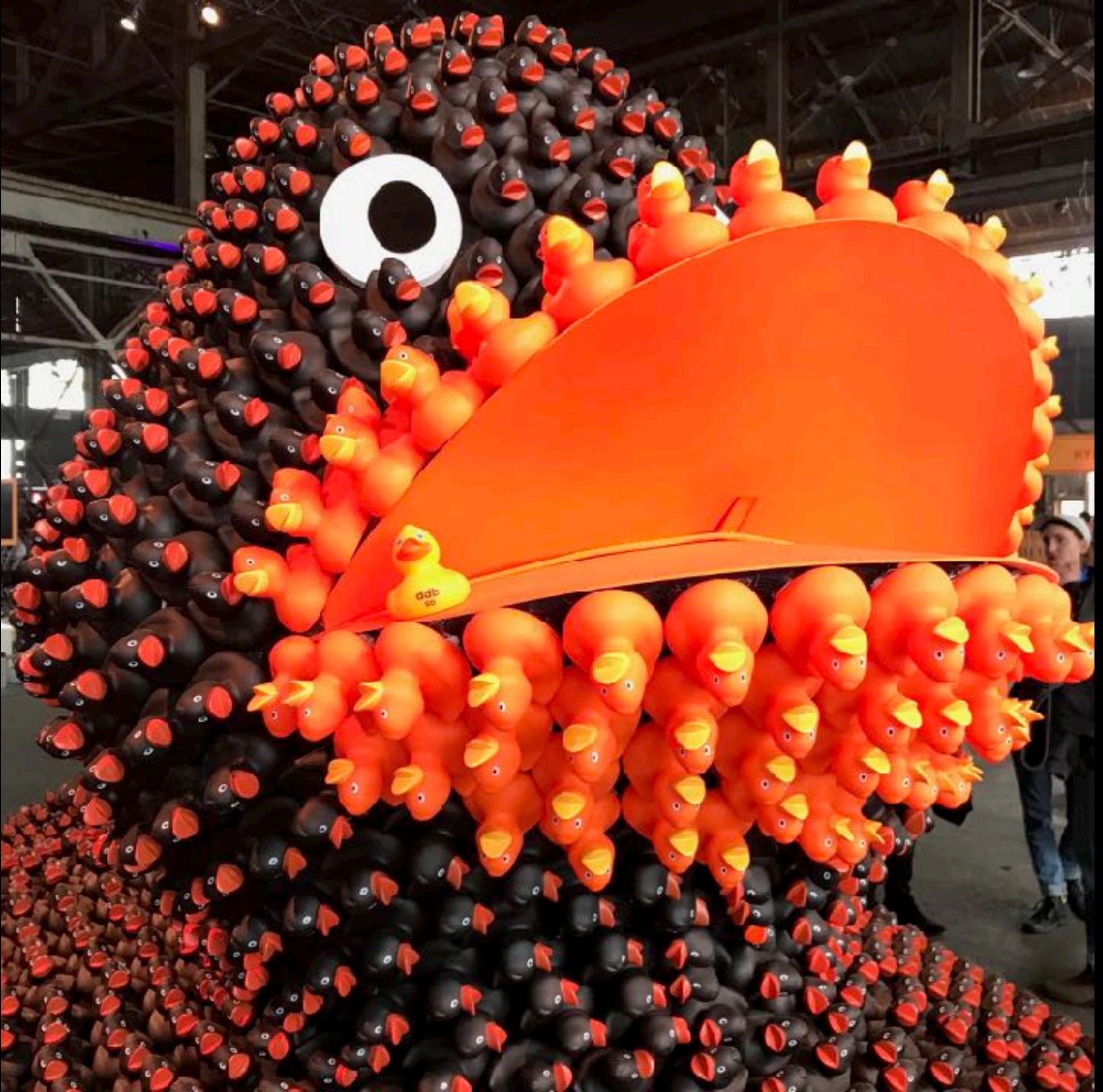
Google's innovative use of cheap components was born out of necessity: Founders Larry Page and Sergey Brin didn't have much capital for equipment. Equally significant, the technical design for Google search was based on tolerating multiple failures and optimizing around them.

By building a system that was redundant at the machine level, and not at the component level, Google's aim was to enable the maximum MIPS per square foot. This do-it-yourself rack was one of about 30 that Google strung together at its first data center. Even though many of the installed machines never worked and could not easily be fixed, these racks provided valuable service.

Each row contains:
8 22GB Hard drives
4 Complete PCs
1 Power supply per tray

One rack contains:
86 Hand-installed fans (cooling)
Wheels (mobility)
Cork (insulation)
Plastic trays to mount hard drives









C



Python



```
printf("hello, world\n");
```



```
print("hello, world")
```



```
#include <stdio.h>

int main(void)
{
    printf("hello, world\n");
}
```



```
def main():
    print("hello, world")
if __name__ == "__main__":
    main()
```



```
def main():
    print("hello, world")

if __name__ == "__main__":
    main()
```



```
def main():
    print("hello, world")

if __name__ == "__main__":
    main()
```



```
def main():
    print("hello, world")
if __name__ == "__main__":
    main()
```



```
def main():
    •••print("hello, world")

if __name__ == "__main__":
    •••main()
```





```
while (true)
{
    printf("hello, world\n");
}
```



```
while True:  
    print("hello, world")
```



```
while True:  
    print("hello, world")
```





```
for (int i = 0; i < 50; i++)  
{  
    printf("hello, world\n");  
}
```



```
for i in range(50):  
    print("hello, world")
```

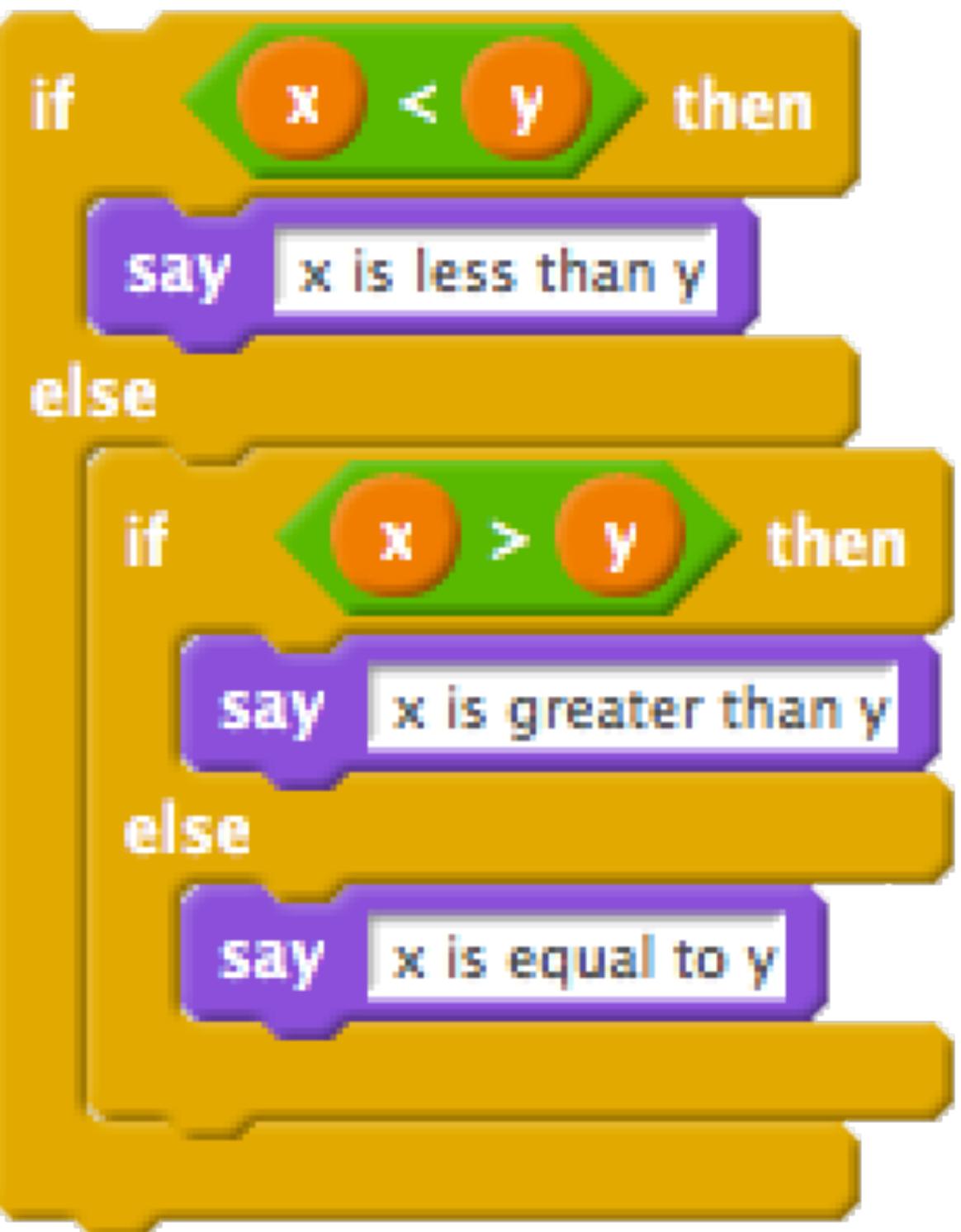


```
for i in range(50):  
    print("hello, world")
```

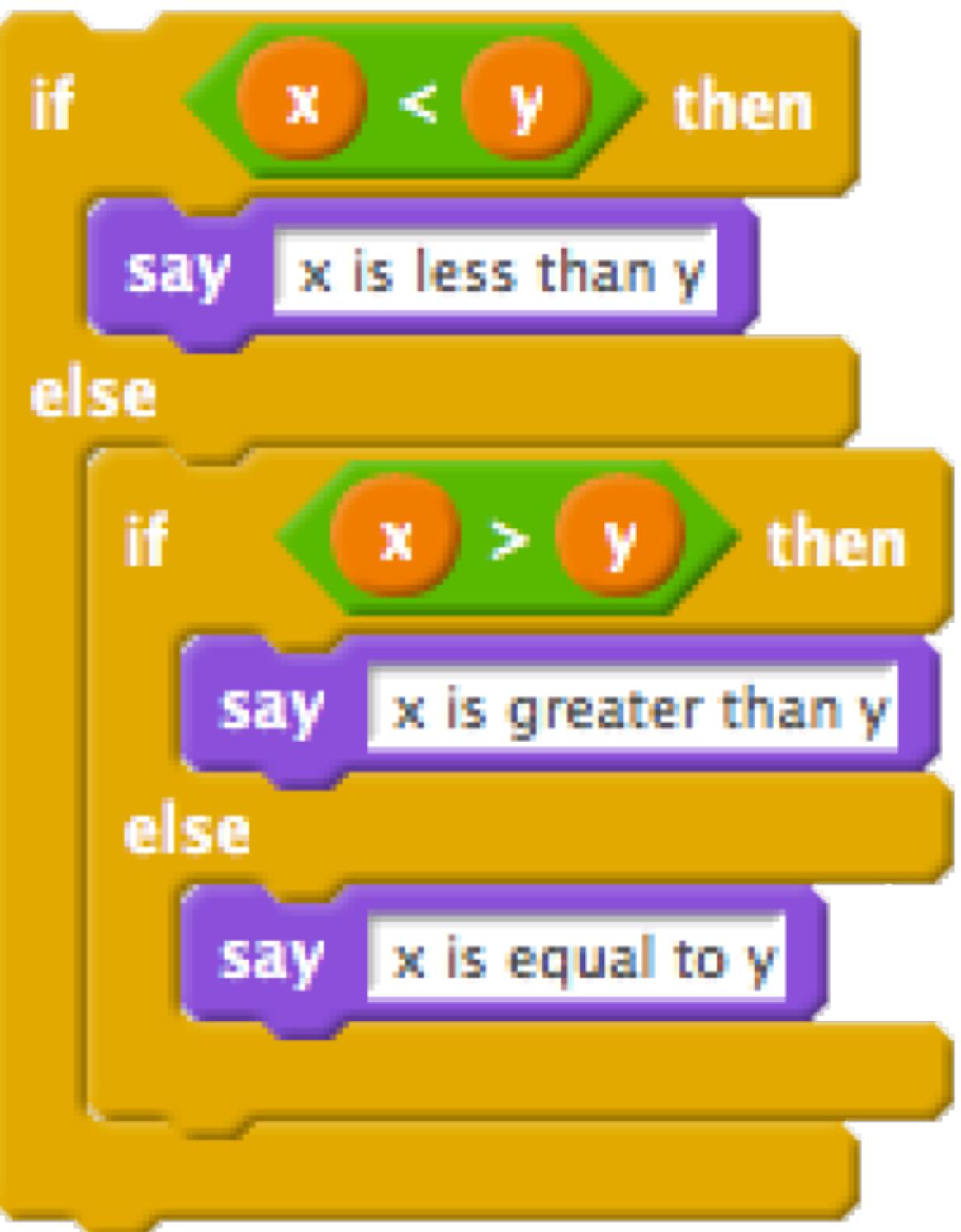


```
for i in range(50):  
    print("hello, world")
```

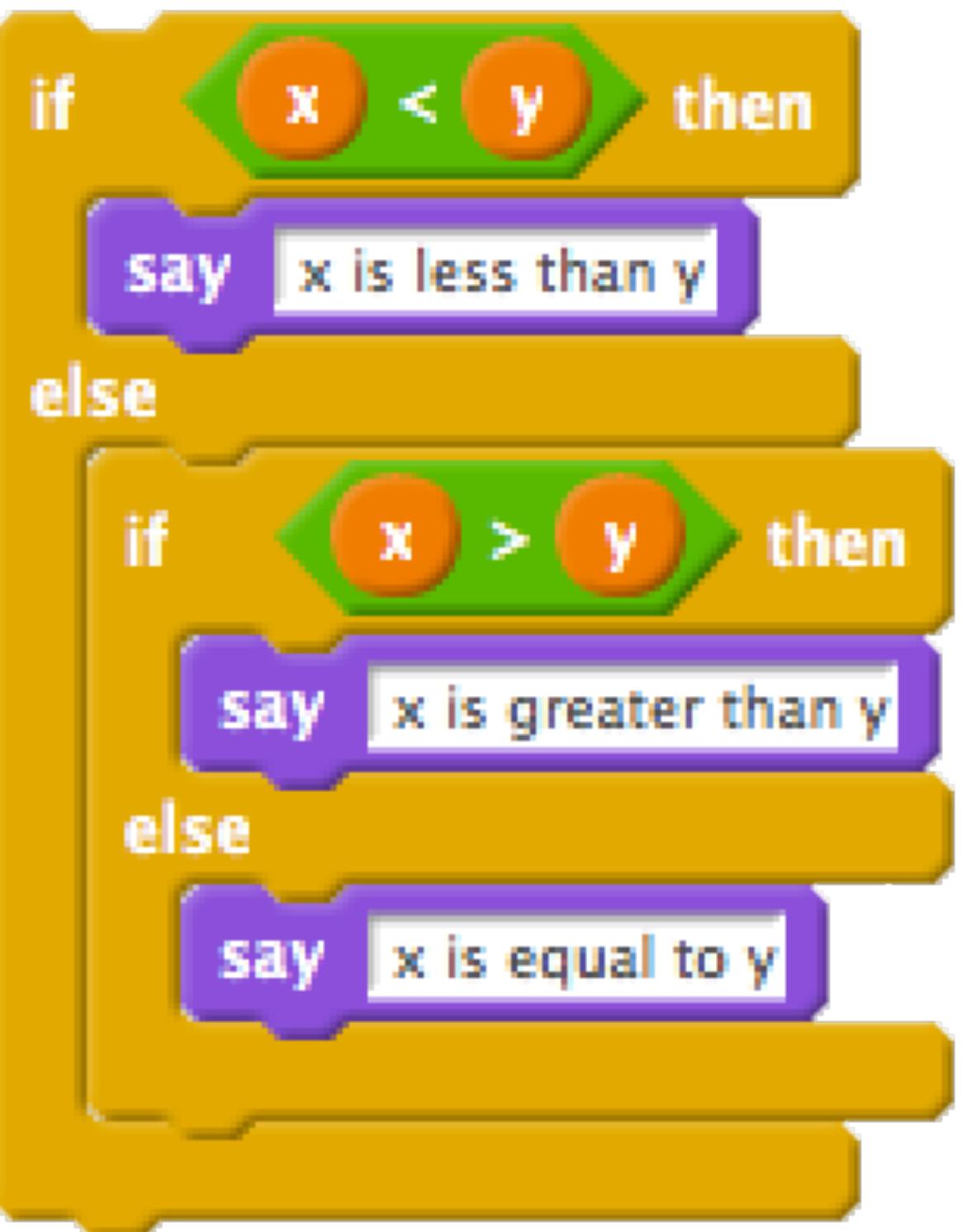
```
if x < y then  
    say x is less than y  
else  
    if x > y then  
        say x is greater than y  
else  
    say x is equal to y
```



```
if (x < y)
{
    printf("x is less than y\n");
}
else if (x > y)
{
    printf("x is greater than y\n");
}
else
{
    printf("x is equal to y\n");
}
```

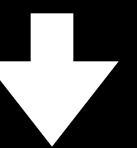


```
if x < y:  
    print("x is less than y")  
elif x > y:  
    print("x is greater than y")  
else:  
    print("x is equal to y")
```

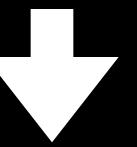


```
if x < y:  
    print("x is less than y")  
elif x > y:  
    print("x is greater than y")  
else:  
    print("x is equal to y")
```

source code



compiler



machine code

```
#include <stdio.h>

int main(void)
{
    printf("hello, world\n");
}
```

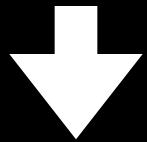
clang hello.c

01111111	01000101	01001100	01000110	00000010	00000001	00000001	00000001	00000000
00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000
00000010	00000000	00111110	00000000	00000001	00000000	00000000	00000000	00000000
10110000	00000101	01000000	00000000	00000000	00000000	00000000	00000000	00000000
01000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000
11010000	00010011	00000000	00000000	00000000	00000000	00000000	00000000	00000000
00000000	00000000	00000000	00000000	01000000	00000000	00111000	00000000	00000000
00001001	00000000	01000000	00000000	00100100	00000000	00100001	00000000	00000000
00000110	00000000	00000000	00000000	00000101	00000000	00000000	00000000	00000000
01000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000
01000000	00000000	01000000	00000000	00000000	00000000	00000000	00000000	00000000
01000000	00000000	01000000	00000000	00000000	00000000	00000000	00000000	00000000
11111000	00000001	00000000	00000000	00000000	00000000	00000000	00000000	00000000
11111000	00000001	00000000	00000000	00000000	00000000	00000000	00000000	00000000
00001000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000
00000011	00000000	00000000	00000000	00000100	00000000	00000000	00000000	00000000
00111000	00000010	00000000	00000000	00000000	00000000	00000000	00000000	00000000
00111000	00000010	01000000	00000000	00000000	00000000	00000000	00000000	00000000
00111000	00000010	01000000	00000000	00000000	00000000	00000000	00000000	00000000
00011100	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000

...

./a.out

source code

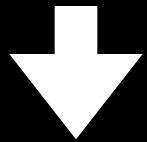


interpreter

```
def main():
    print("hello, world")
```

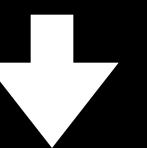
```
python hello.py
```

source code

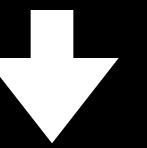


interpreter

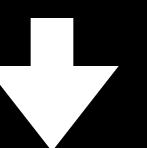
source code



compiler



byte code



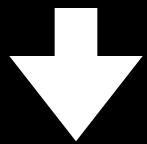
interpreter

```
2      0 LOAD_GLOBAL
      3 LOAD_CONST
      6 CALL_FUNCTION
      9 POP_TOP
     10 LOAD_CONST
     13 RETURN_VALUE

          0 (print)
          1 ('hello, world')
          1 (1 positional, 0 keyword pair)

          0 (None)
```

source code



interpreter

bool

float

int

str

...

`get_char`

`get_float`

`get_int`

`get_string`

...

...

complex

dict

list

range

set

tuple

...



CS50

H

Y