This is CS50

```c
#include <stdio.h>


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

```c
#include <stdio.h>


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

```c
#include <stdio.h>


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

```c
#include <stdio.h>


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

```c
#include <stdio.h>


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

```c
#include <stdio.h>


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

```
clang hello.c

./a.out
```

```
clang -o hello hello.c

./hello
```

```c
#include <stdio.h>


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

```c
#include <cs50.h>
#include <stdio.h>


int main(void)
{
    string name = get_string("What's your name?\n");
    printf("hello, %s\n", name);
}
```

```c
#include <cs50.h>
#include <stdio.h>


int main(void)
{
    string name = get_string("What's your name?\n");
    printf("hello, %s\n", name);
}
```

```c
#include <cs50.h>
#include <stdio.h>


int main(void)
{
    string name = get_string("What's your name?\n");
    printf("hello, %s\n", name);
}
```

```c
#include <cs50.h>
#include <stdio.h>


int main(void)
{
    string name = get_string("What's your name?\n");
    printf("hello, %s\n", name);
}
```

```c
#include <cs50.h>
#include <stdio.h>


int main(void)
{
    string name = get_string("What's your name?\n");
    printf("hello, %s\n", name);
}
```

```c
#include <cs50.h>
#include <stdio.h>


int main(void)
{
    string name = get_string("What's your name?\n");
    printf("hello, %s\n", name);
}
```

```c
#include <cs50.h>
#include <stdio.h>


int main(void)
{
    string name = get_string("What's your name?\n");
    printf("hello, %s\n", name);
}
```

```
clang -o hello hello.c -lcs50

./hello
```

```
make hello

./hello
```

compiling

preprocessing

compiling

assembling

linking

preprocessing

compiling

assembling

linking

```c
#include <cs50.h>
#include <stdio.h>


int main(void)
{
    string name = get_string("What's your name?\n");
    printf("hello, %s\n", name);
}
```

```c
#include <cs50.h>
#include <stdio.h>


int main(void)
{
    string name = get_string("What's your name?\n");
    printf("hello, %s\n", name);
}
```

```
string get_string(string prompt);
#include <stdio.h>


int main(void)
{
    string name = get_string("What's your name?\n");
    printf("hello, %s\n", name);
}
```

```
string get_string(string prompt);
#include <stdio.h>


int main(void)
{
    string name = get_string("What's your name?\n");
    printf("hello, %s\n", name);
}
```

```
string get_string(string prompt);
int printf(string format, ...);


int main(void)
{
    string name = get_string("What's your name?\n");
    printf("hello, %s\n", name);
}
```

```
...
string get_string(string prompt);
int printf(string format, ...);
...

int main(void)
{
    string name = get_string("What's your name?\n");
    printf("hello, %s\n", name);
}
```

preprocessing

compiling

assembling

linking

```
...
string get_string(string prompt);
int printf(string format, ...);
...

int main(void)
{
    string name = get_string("What's your name?\n");
    printf("hello, %s\n", name);
}
```

```
...
main:                                   # @main
    .cfi_startproc
# BB#0:
    pushq    %rbp
.Ltmp0:
    .cfi_def_cfa_offset 16
.Ltmp1:
    .cfi_offset %rbp, -16
    movq    %rsp, %rbp
.Ltmp2:
    .cfi_def_cfa_register %rbp
    subq    $16, %rsp
    xorl    %eax, %eax
    movl    %eax, %edi
    movabsq    $.L.str, %rsi
    movb    $0, %al
    callq    get_string
    movabsq    $.L.str.1, %rdi
    movq    %rax, -8(%rbp)
    movq    -8(%rbp), %rsi
    movb    $0, %al
    callq    printf
    ...
```

```
...
main:                                   # @main
    .cfi_startproc
# BB#0:
    pushq    %rbp
.Ltmp0:
    .cfi_def_cfa_offset 16
.Ltmp1:
    .cfi_offset %rbp, -16
    movq    %rsp, %rbp
.Ltmp2:
    .cfi_def_cfa_register %rbp
    subq    $16, %rsp
    xorl    %eax, %eax
    movl    %eax, %edi
    movabsq    $.L.str, %rsi
    movb    $0, %al
    callq    get_string
    movabsq    $.L.str.1, %rdi
    movq    %rax, -8(%rbp)
    movq    -8(%rbp), %rsi
    movb    $0, %al
    callq    printf
    ...
```

```
...
main:                                   # @main
    .cfi_startproc
# BB#0:
    pushq    %rbp
.Ltmp0:
    .cfi_def_cfa_offset 16
.Ltmp1:
    .cfi_offset %rbp, -16
    movq    %rsp, %rbp
.Ltmp2:
    .cfi_def_cfa_register %rbp
    subq    $16, %rsp
    xorl    %eax, %eax
    movl    %eax, %edi
    movabsq    $.L.str, %rsi
    movb    $0, %al
    callq    get_string
    movabsq    $.L.str.1, %rdi
    movq    %rax, -8(%rbp)
    movq    -8(%rbp), %rsi
    movb    $0, %al
    callq    printf
    ...
```

preprocessing

compiling

assembling

linking

```
...
main:                                   # @main
    .cfi_startproc
# BB#0:
    pushq    %rbp
.Ltmp0:
    .cfi_def_cfa_offset 16
.Ltmp1:
    .cfi_offset %rbp, -16
    movq    %rsp, %rbp
.Ltmp2:
    .cfi_def_cfa_register %rbp
    subq    $16, %rsp
    xorl    %eax, %eax
    movl    %eax, %edi
    movabsq    $.L.str, %rsi
    movb    $0, %al
    callq    get_string
    movabsq    $.L.str.1, %rdi
    movq    %rax, -8(%rbp)
    movq    -8(%rbp), %rsi
    movb    $0, %al
    callq    printf
    ...
```

```
01111111010001010100110001000110
00000010000000010000000100000000
00000000000000000000000000000000
00000000000000000000000000000000
00000001000000000011111000000000
00000001000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
10100000000000100000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
01000000000000000000000000000000
00000000000000000100000000000000
00001010000000000000000100000000
01010101010010001000100111100101
01001000100000111110110000010000
00110001110000001000100111000111
01001000101111100000000000000000
00000000000000000000000000000000
00000000000000001011000000000000
11101000000000000000000000000000
00000000010010001011111100000000
00000000000000000000000000000000
00000000000000000000000001001000
...
```

preprocessing

compiling

assembling

linking

```c
#include <cs50.h>
#include <stdio.h>


int main(void)
{
    string name = get_string("What's your name?\n");
    printf("hello, %s\n", name);
}
```

```c
#include <cs50.h>
#include <stdio.h>


int main(void)
{
    string name = get_string("What's your name?\n");
    printf("hello, %s\n", name);
}
```

```c
#include <cs50.h>
#include <stdio.h>


int main(void)
{
    string name = get_string("What's your name?\n");
    printf("hello, %s\n", name);
}
```

```c
#include <cs50.h>
#include <stdio.h>


int main(void)
{
    string name = get_string("What's your name?\n");
    printf("hello, %s\n", name);
}
```

hello.c

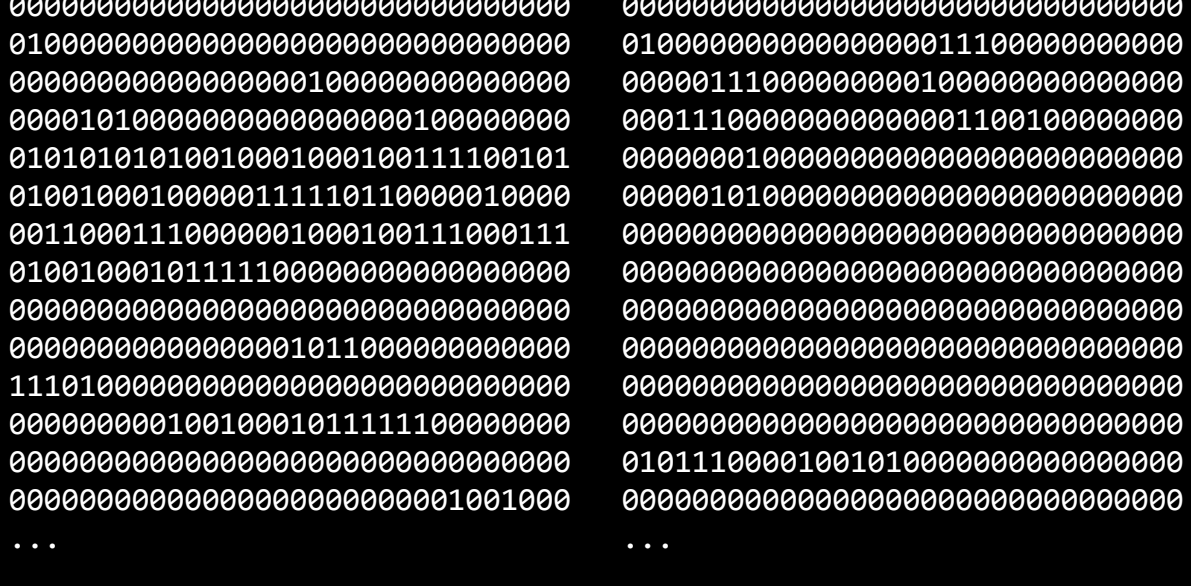
hello.c

cs50.c

hello.c cs50.c stdio.c

hello.c cs50.c printf.c


cs50.c
printf.c



01111111010001010100110001000110
00000010000000010000000100000000
00000000000000000000000000000000
00000000000000000000000000000000
00000011000000000011111000000000
00000001000000000000000000000000
11000000000011110000000000000000
00000000000000000000000000000000
01000000000000000000000000000000
00000000000000000000000000000000
00101000011001000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
01000000000000000111000000000000
00001110000000001000000000000000
00011100000000000011001000000000
00000001000000000000000000000000
00000101000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
01011100001001010000000000000000
00000000000000000000000000000000
...

printf.c

011111110100010101001100010001100000001000000001000000010000000000000000000000000000000000000000
00000000000000000000000000000000010000000000111110000000000000000010000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000010100000000000010000000000000000000000000000000000000000000000000000000000000000000
00000000010000000000000000000000000000000000000000010000000000000000000101000000000000000001000000
00010101010100100010001001111001010100100010000011111011000001000000110001110000001000100111000111010 0
10001011111000000000000000000000000000000000000000000000000000000000001011000000000000011101000000
00000000000000000000000000000001001000101111110000000000000000000000000000000000000000000000000000
0000000001001000...011111110100010101001100010001100000001000000001000000010000000000000000000000000
00000000000000000000000000000000000000000000001100000000001111100000000000000000100000000000000000
00000001100000000000111110000000000000000000000000000000000000000000010000000000000000000000000000
00000000000000000000000000000000001010000011001000000000000000000000000000000000000000000000000000
00000000000000000000000000001000000000000000000111000000000000000000111000000000001000000000000000000111000000
00000000110010000000000000001000000000000000000000000000001010000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000101110000100101000000000000000
00000000000000000000000000000000000...001011110110110001101001011000100110001100101110011100110110111 1
00101110001101100010000000101111011101010111001101110010001011110110110001101001011000100010111101111 0
00001110000001101100101111100110110001101000010110101101100011010010110111001110101011110000010110101 10
01110110111001110101001011110110110001101001011000100110001101011111011011100110111101101110011100110 1
10100001100001011100100110010101100100001011100110000100100000001000000100000101010011010111110100111 0
01000101010001010100010001000101010001000010000000101000001000000001011110110110001101001011000100010 11
11011110000001110000011011001011111001101100011010000101101011011000110100101101110011101010111100000 10
11010110011101101110011101010010111101101100011001000010110101101100011010010110111001110101011110000 0
101101011110000011100000110110001011010011011000110100...

preprocessing

compiling

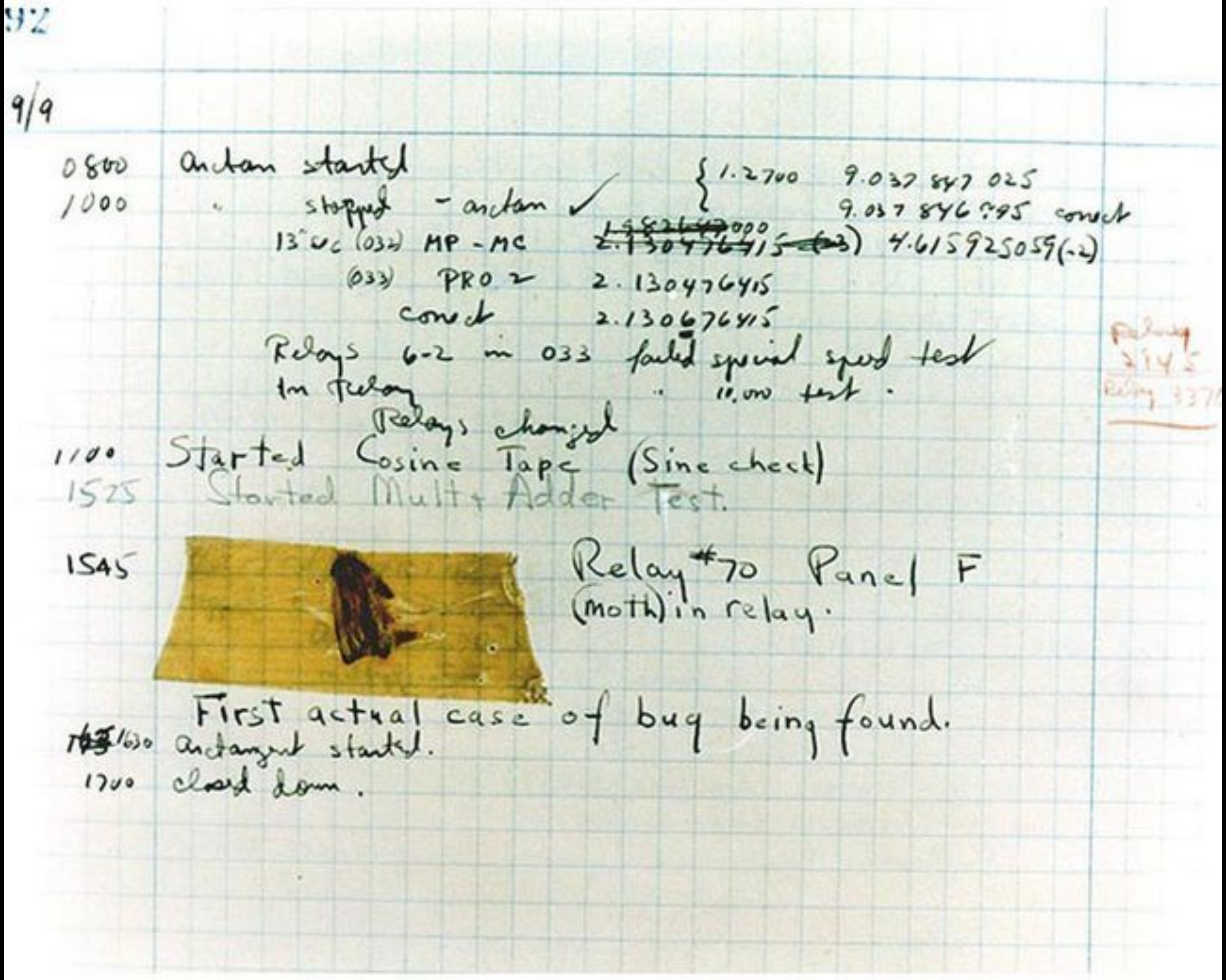
assembling

linking

compiling

debugging

Photo # NH 96566-KN (Color) First Computer "Bug", 1947



92

9/9

0800 Anctan started

1000 " stopped - anctan ✓ {1.2700 9.037 847 025

9.037 846 995 correct

13" uc (032) MP - MC ~~1.982147000~~ 2.130476415 ~~(-3)~~ 4.615925059(-2)

(033) PRO 2 2.130476415

correct 2.130676415

Relays 6-2 in 033 failed special speed test

In relay " 11.000 test.

Relays changed

1100 Started Cosine Tape (Sine check)

1525 Started Mult + Adder Test.

1545 Relay #70 Panel F

(moth) in relay.

First actual case of bug being found.

~~1600~~ 1630 Antangent started.

1700 closed down.

Relay 2145

Relay 3370

In relay . 11.00 test.

Relays changed

1100 Started Cosine Tape (Sine check)

1525 Started Multy Adder Test.

1545  Relay #70 Panel F
(moth) in relay.

First actual case of bug being found.

~~1655~~ 1630 Arctangent started.

1700 closed down.

```
help50
```

```
help50

printf
```

CS50 IDE

ide.cs50.io

```
help50

printf

debug50
```

```
help50

printf

debug50

check50
```

```
help50

printf

debug50

check50

style50
```

`help50`	correctness
`printf`	correctness
`debug50`	correctness
`check50`	correctness
`style50`	style

`help50`	correctness
`printf`	correctness
`debug50`	correctness
`check50`	correctness
`style50`	style
`ddb`	

correctness

design

style

arrays

`char` 1 byte

`char` 1 byte

`int` 4 bytes

`char`	1 byte
`int`	4 bytes
`float`	4 bytes

`char`	1 byte
`int`	4 bytes
`float`	4 bytes
`long`	8 bytes

`char`	1 byte
`int`	4 bytes
`float`	4 bytes
`long`	8 bytes
`double`	8 bytes

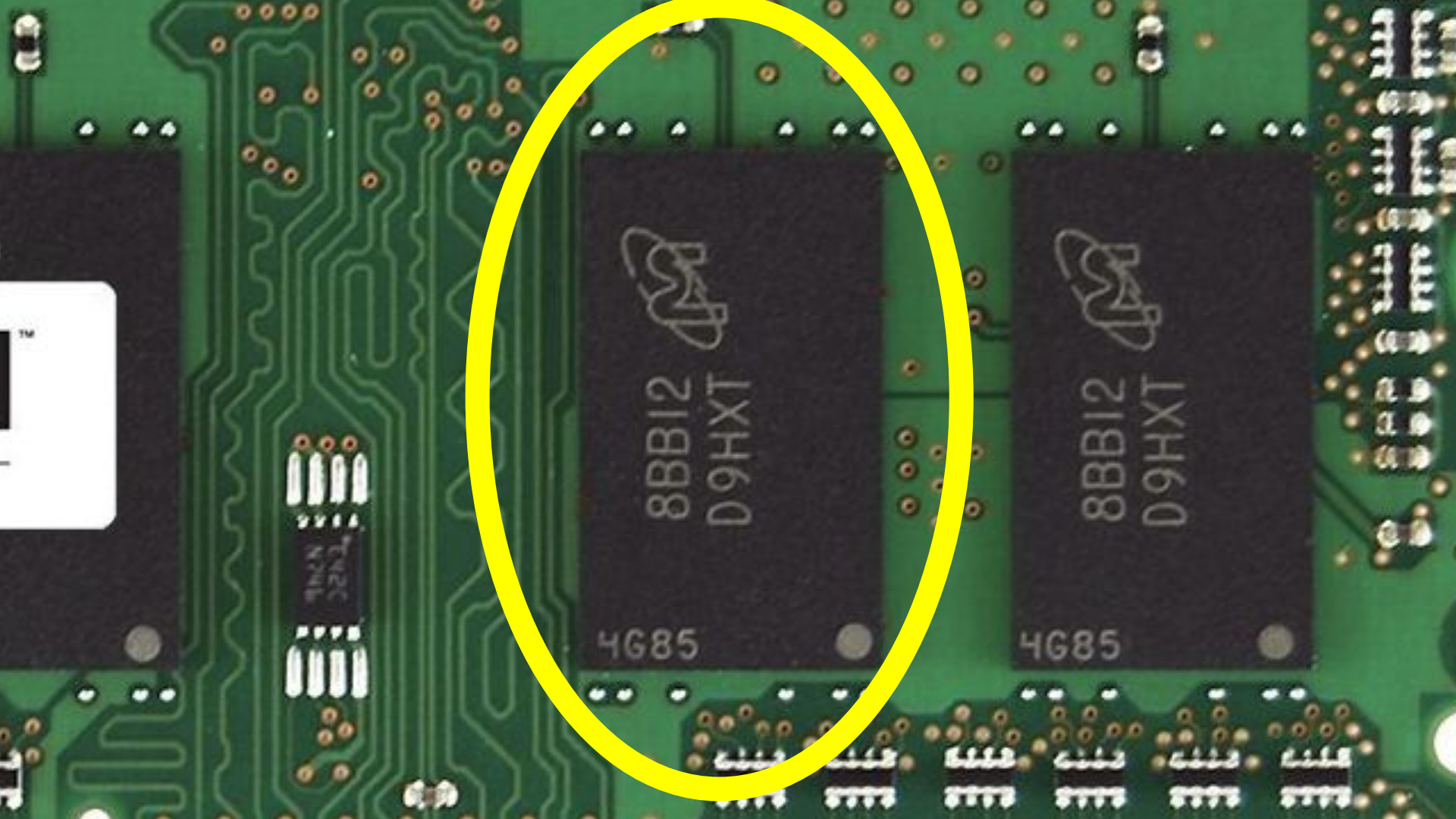
`bool`	1 byte
`char`	1 byte
`int`	4 bytes
`float`	4 bytes
`long`	8 bytes
`double`	8 bytes

`bool`	1 byte
`char`	1 byte
`int`	4 bytes
`float`	4 bytes
`long`	8 bytes
`double`	8 bytes
`string`	? bytes

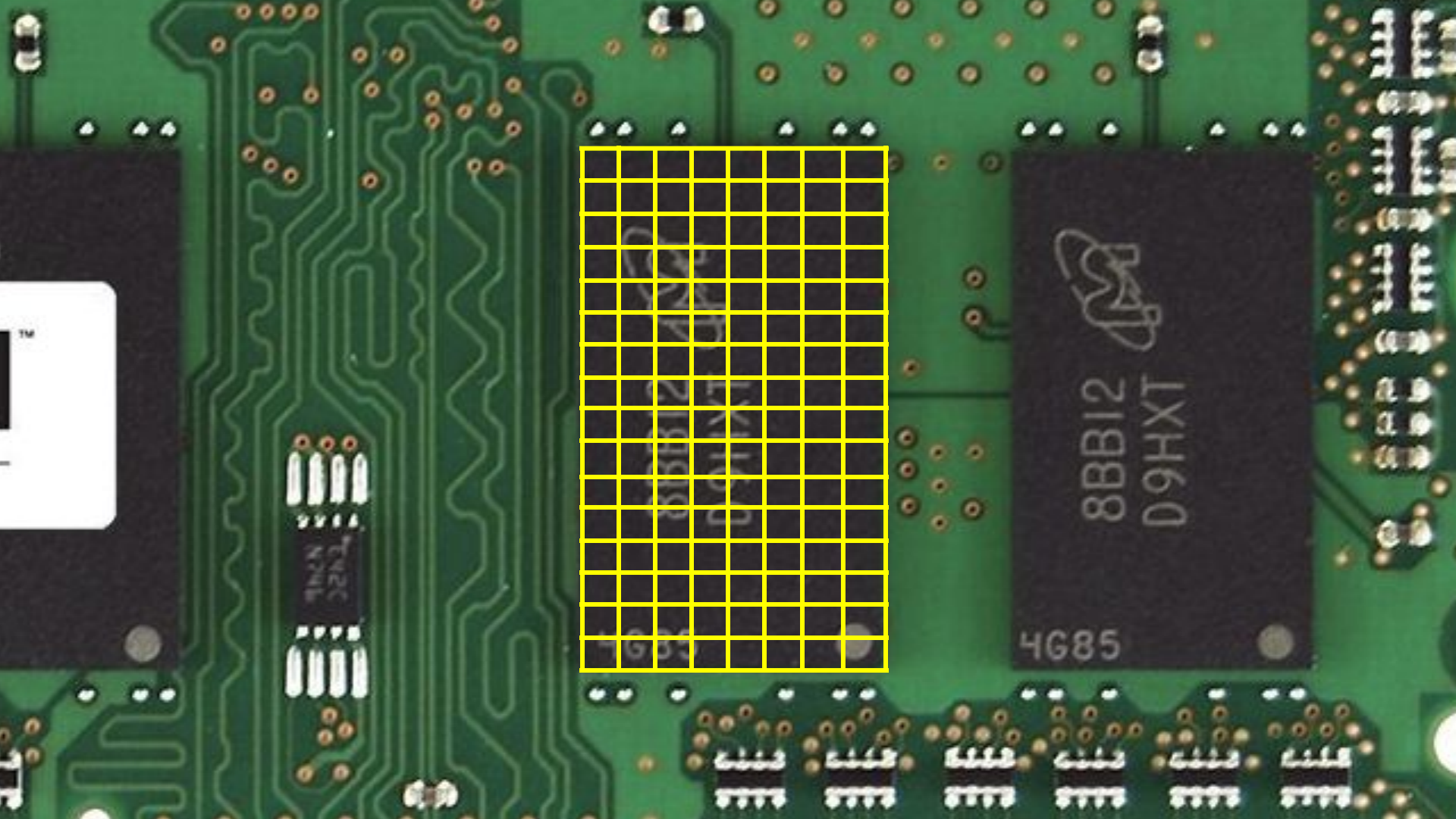
`...`

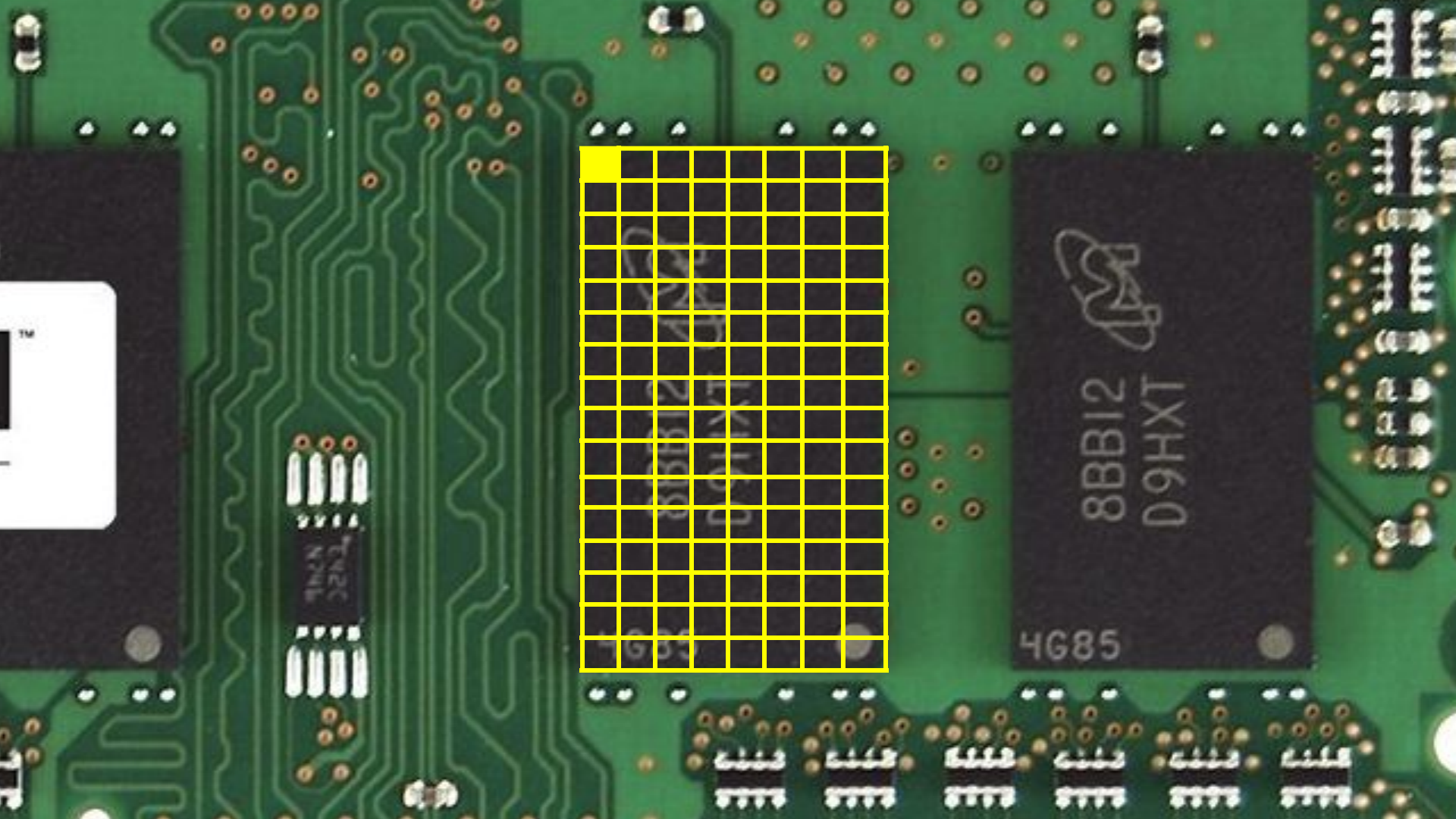

crucial
by Micron
8BBI2
D9HXT
8BBI2
D9HXT
4G85
4G85
4G85
47J6

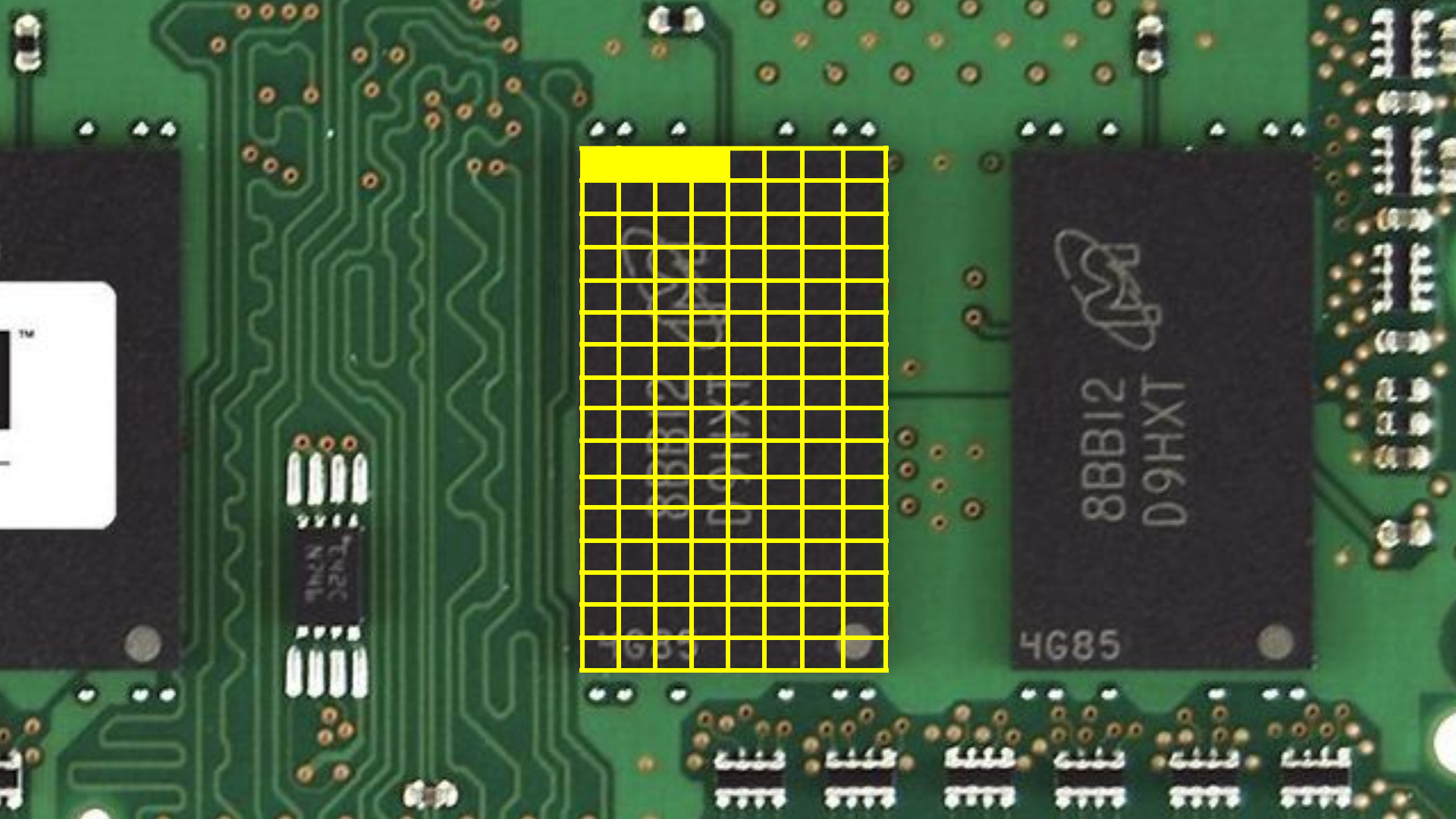

crucial
by Micron
8BBI2
D9HXT
4G85
8BBI2
D9HXT
4G85
4G85
47J6
4G85

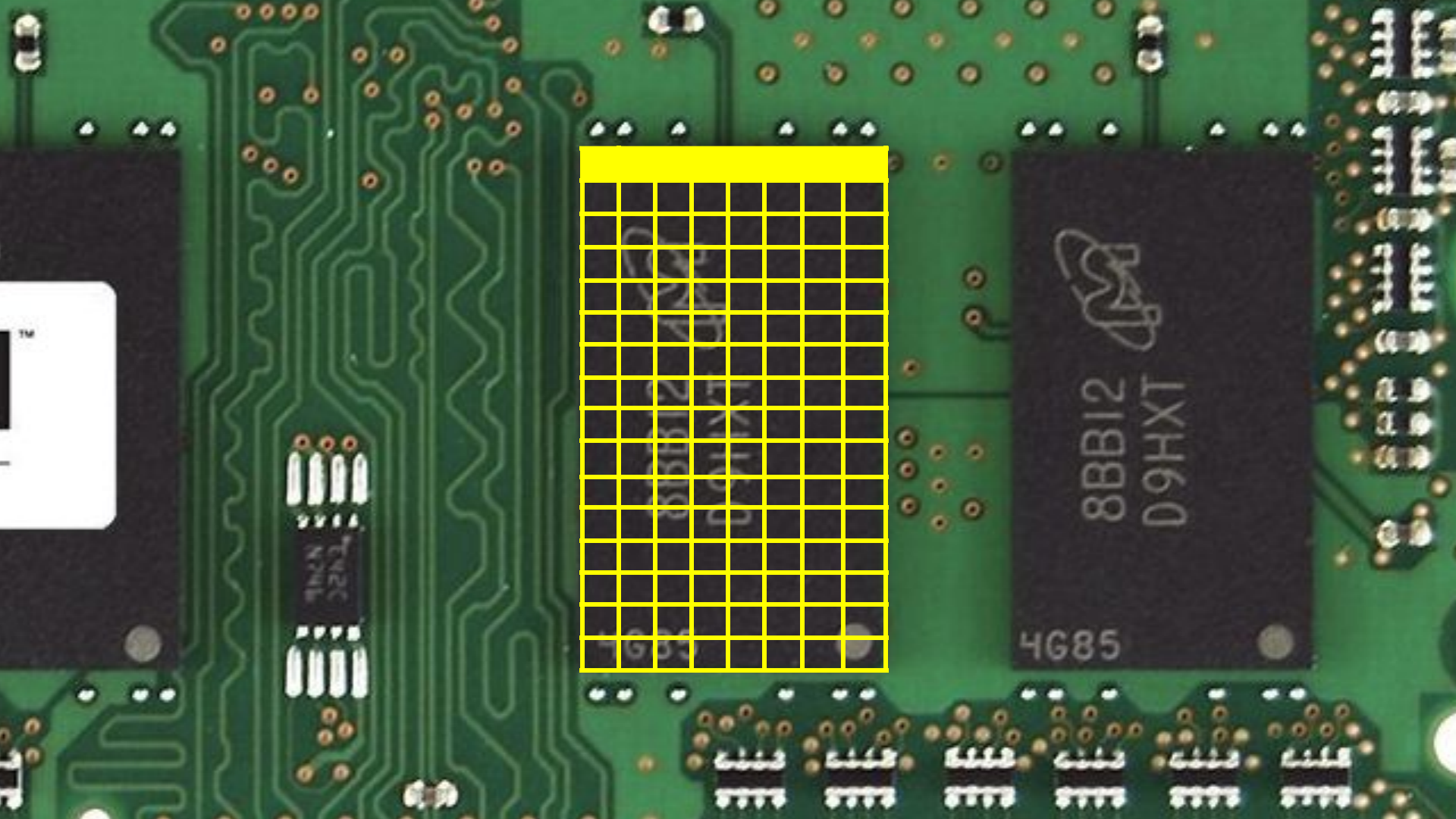

8BBI2
D9HXT
4G85
8BBI2
D9HXT
4G85

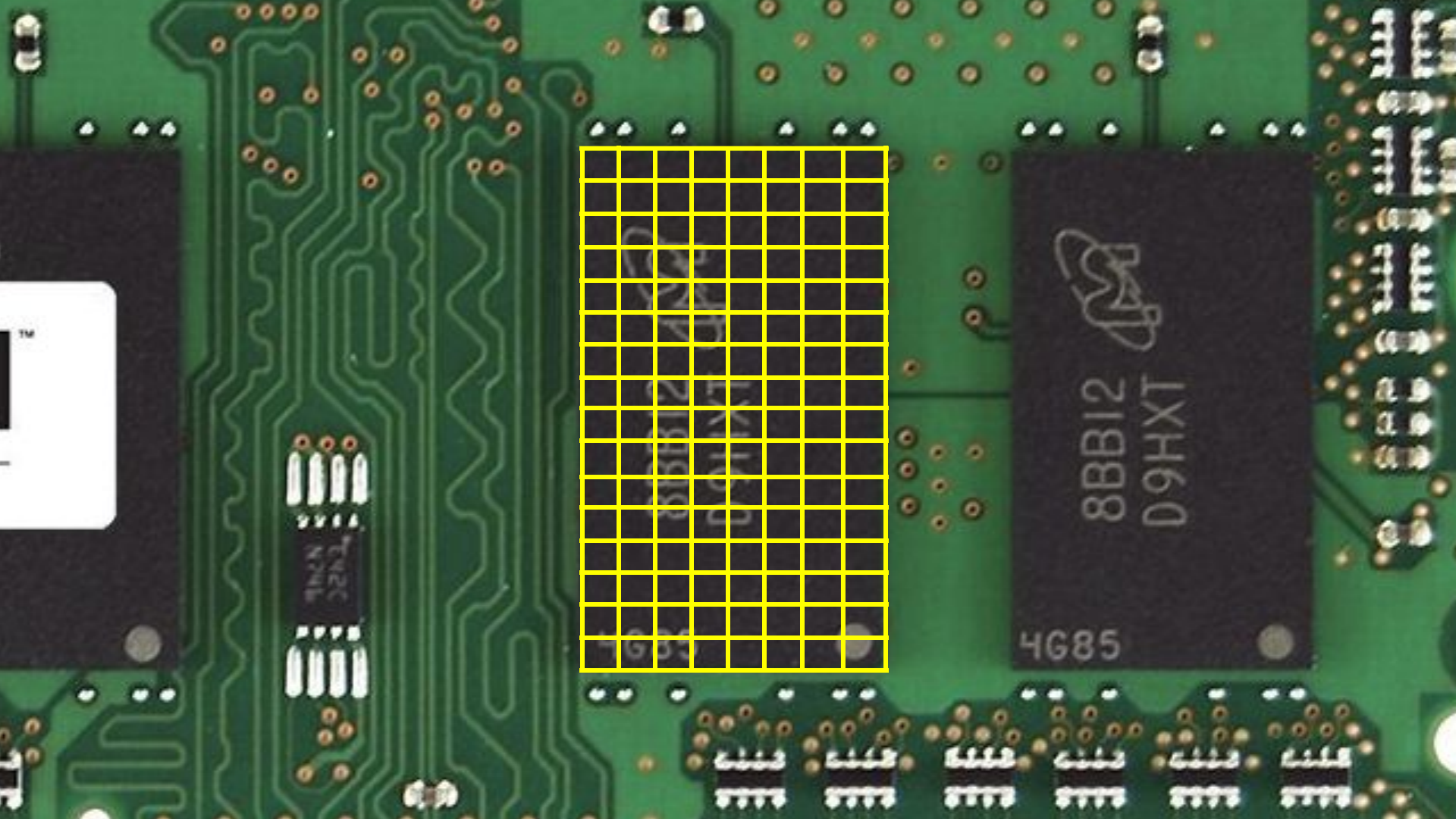

8BBI2
D9HXT
4G85
8BBI2
D9HXT
4G85

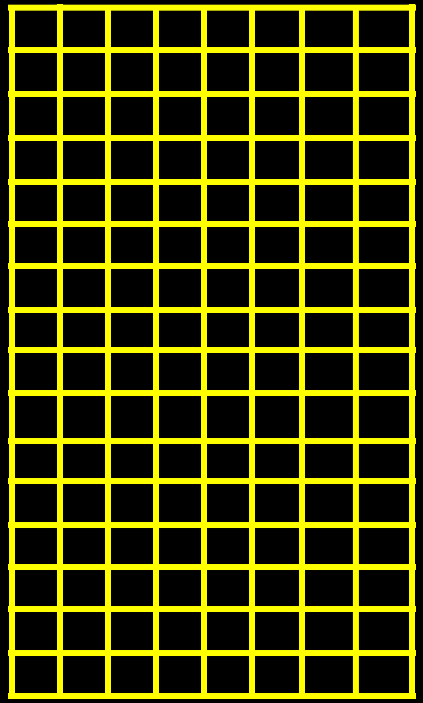

8BBI2
D9HXT
4G85


8BBI2
D9HXT
4G85

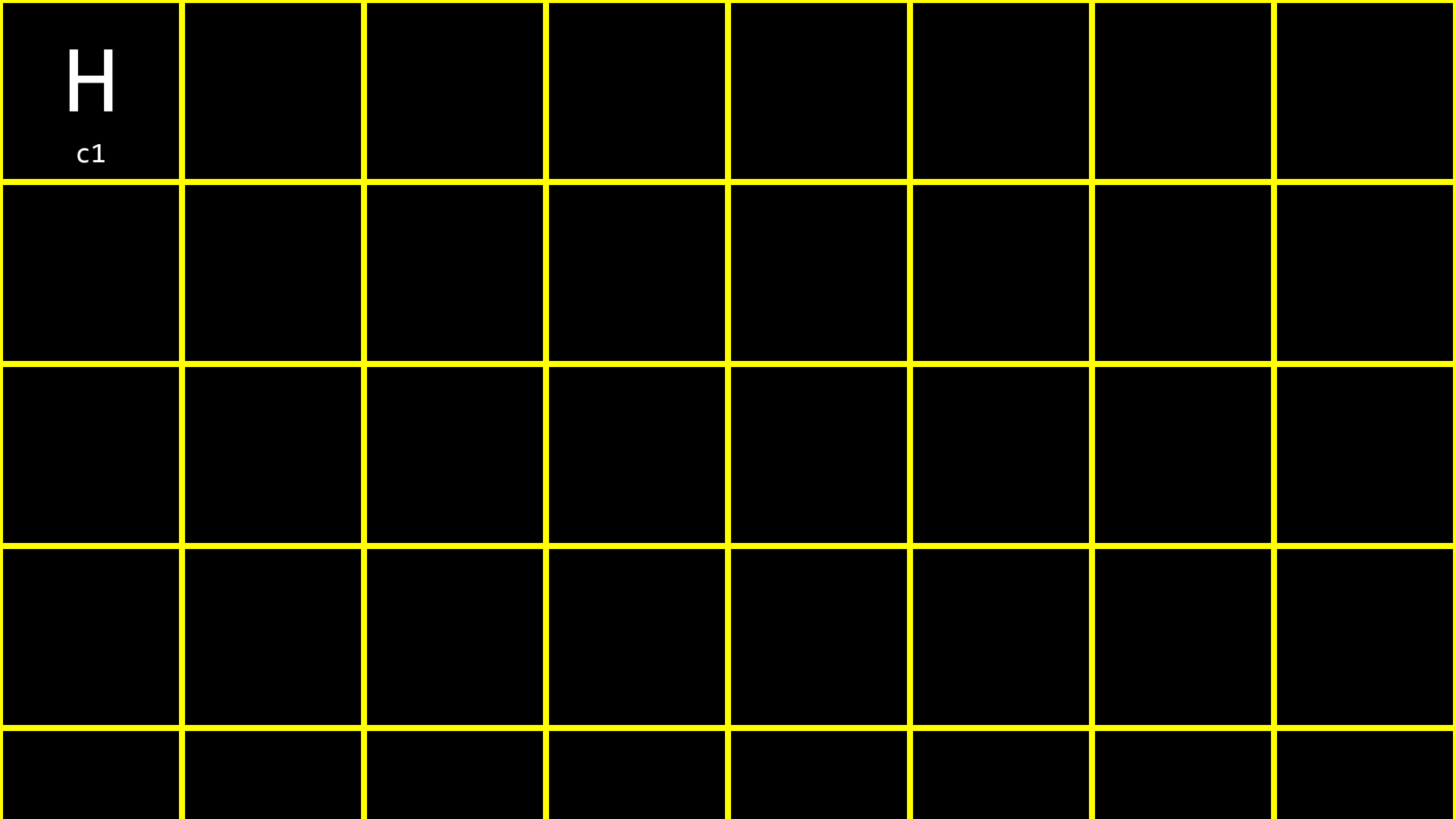

8BBI2
D9HXT
4G85

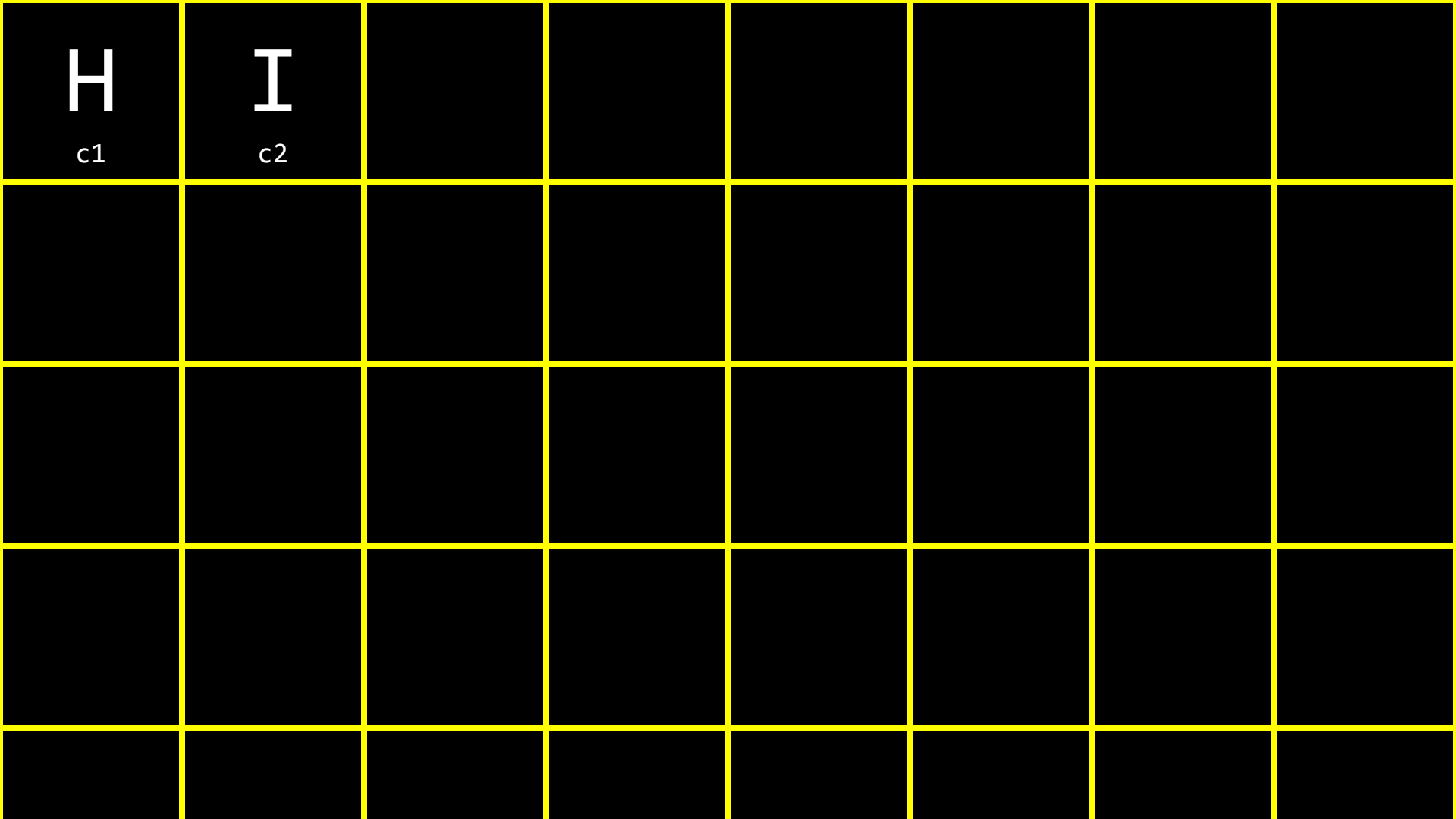

8BBI2
D9HXT
4G85

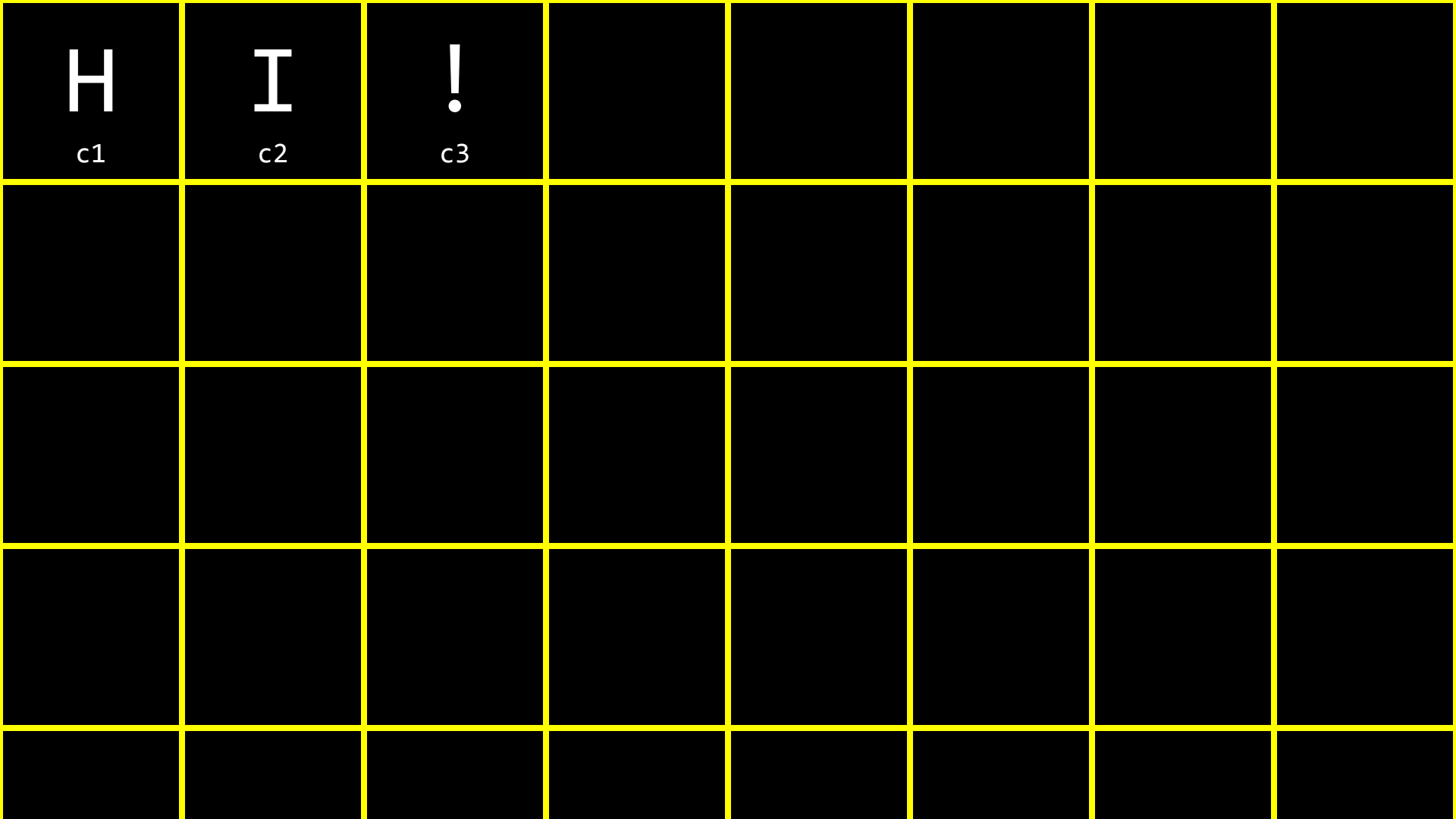

8BBI2
D9HXT
4G85



```
char c1 = 'H';

char c2 = 'I';

char c3 = '!';
```


H
c1


H
c1
I
c2

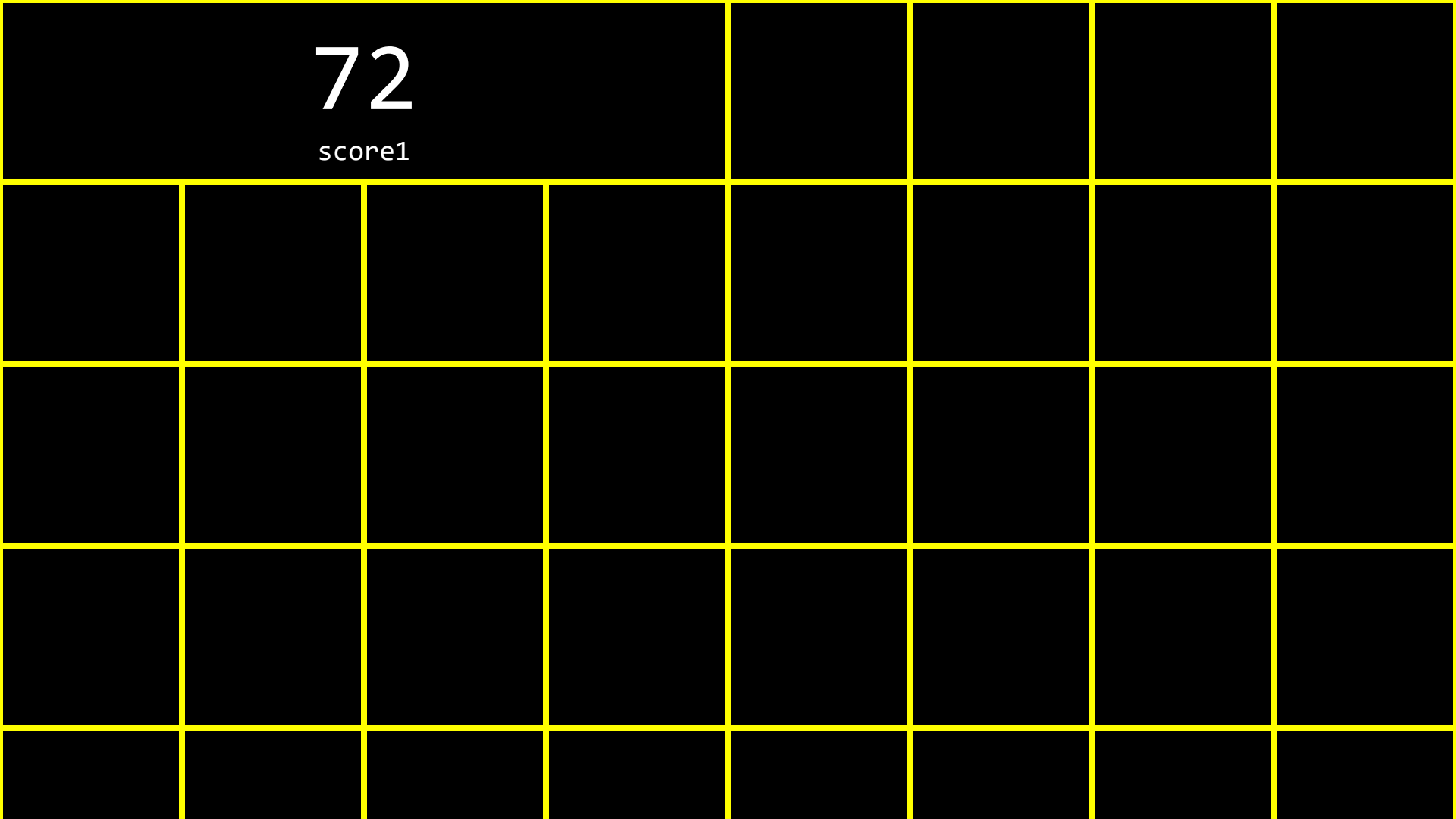

H
I
!
c1
c2
c3

72 c1	73 c2	33 c3					

01001000 c1	01001001 c2	00100001 c3					

72 c1	73 c2	33 c3					

```
int score1 = 72;

int score2 = 73;

int score3 = 33;
```


72
score1

72

score1

73

score2

72

score1

73

score2

33

score3

00000000000000000000001001000

score1

00000000000000000000001001001

score2

00000000000000000000100001

score3

```
int score1 = 72;

int score2 = 73;

int score3 = 33;
```

```
int scores[3];

scores[0] = 72;

scores[1] = 73;

scores[2] = 33;
```

72

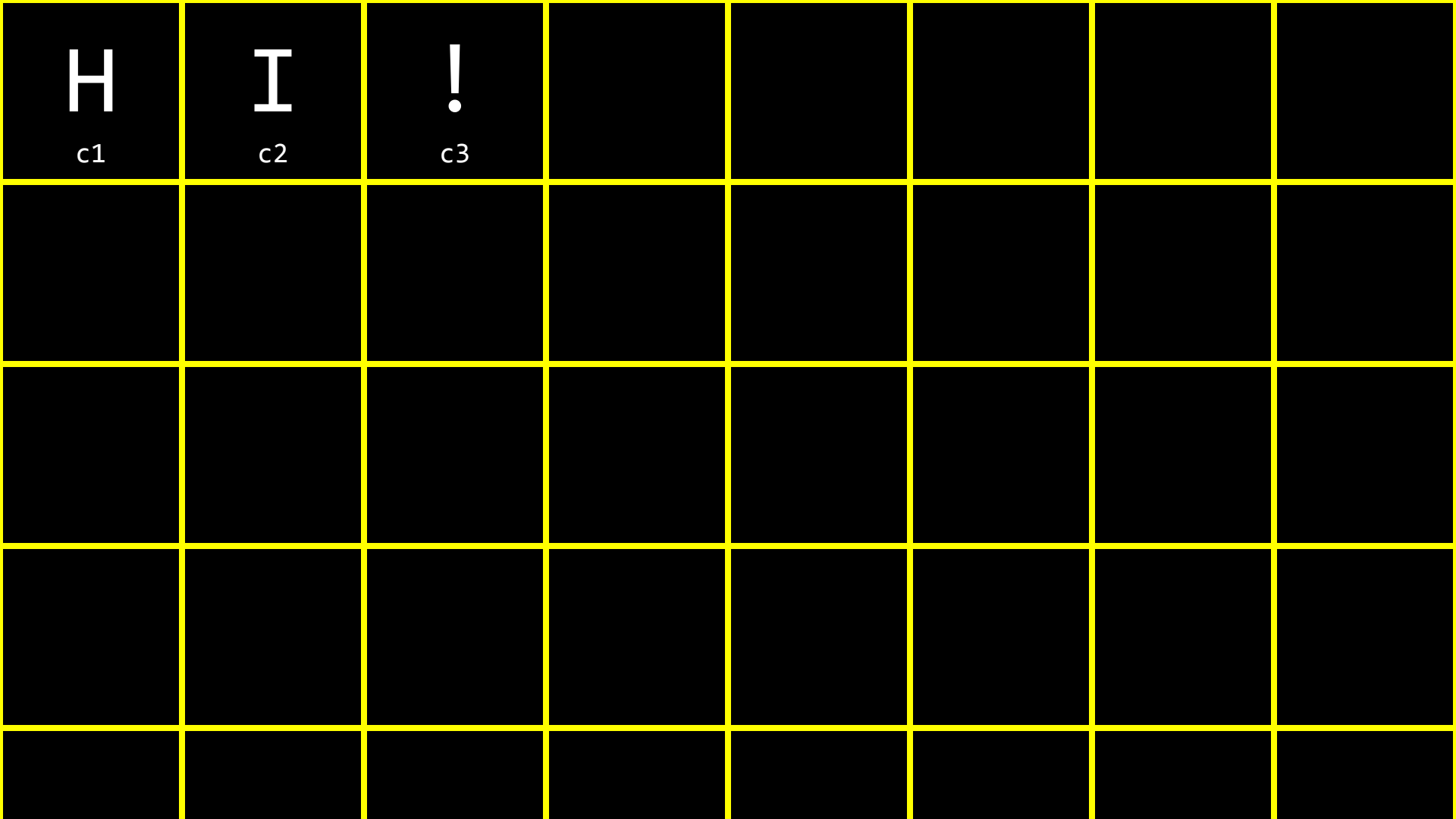
scores[0]

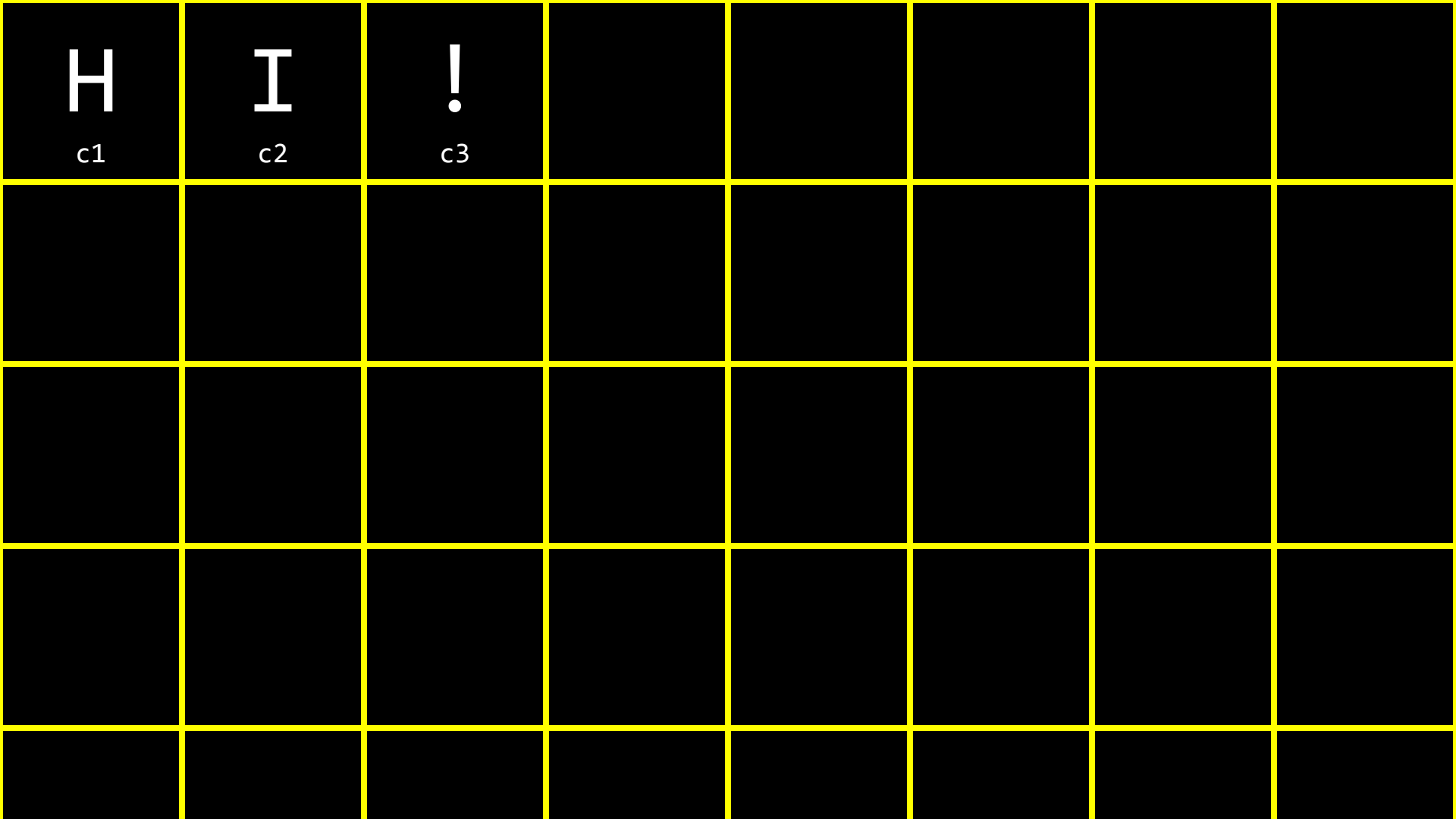
73

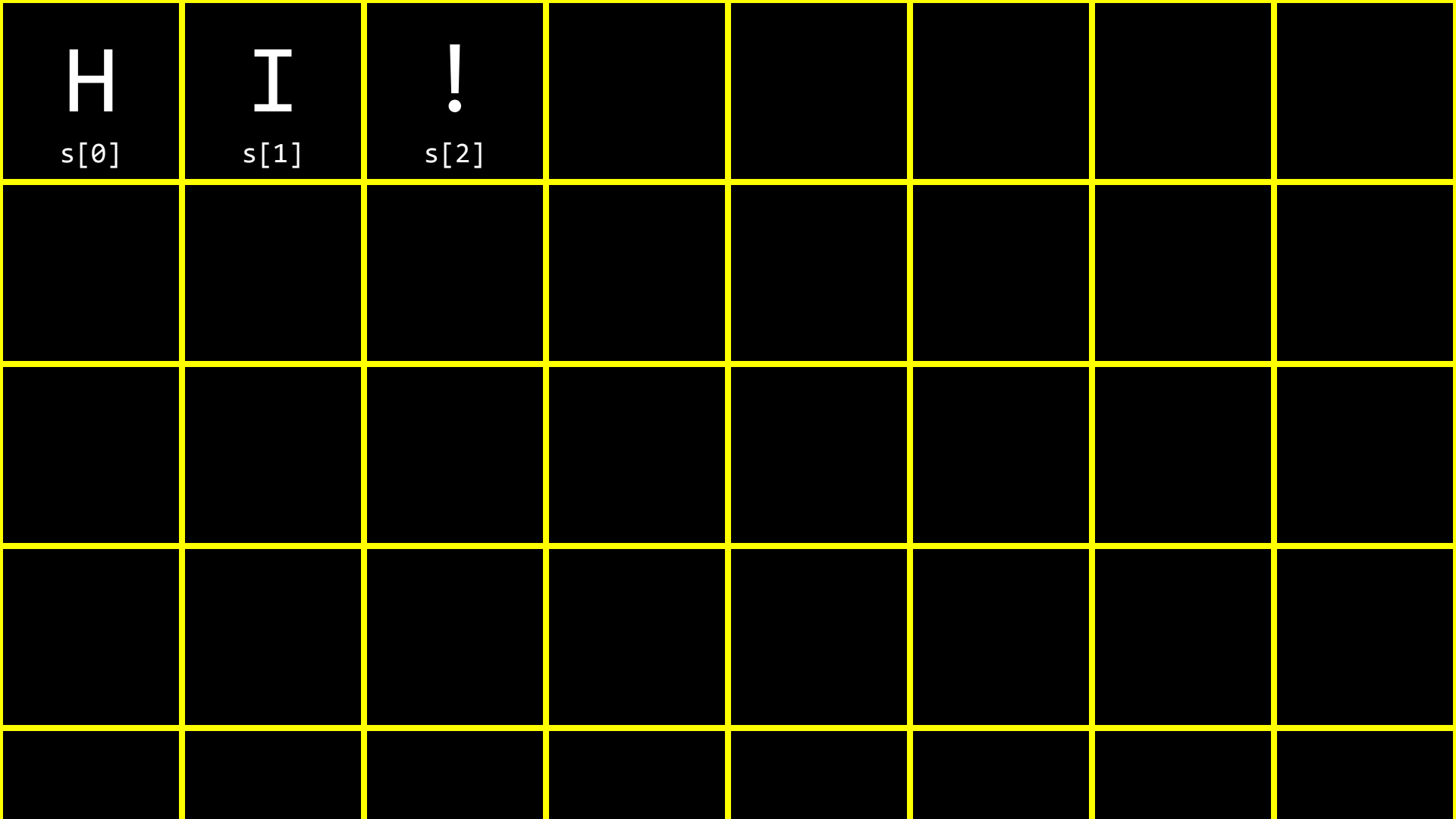
scores[1]

33

scores[2]


H
I
!
c1
c2
c3


H
I
!
c1
c2
c3

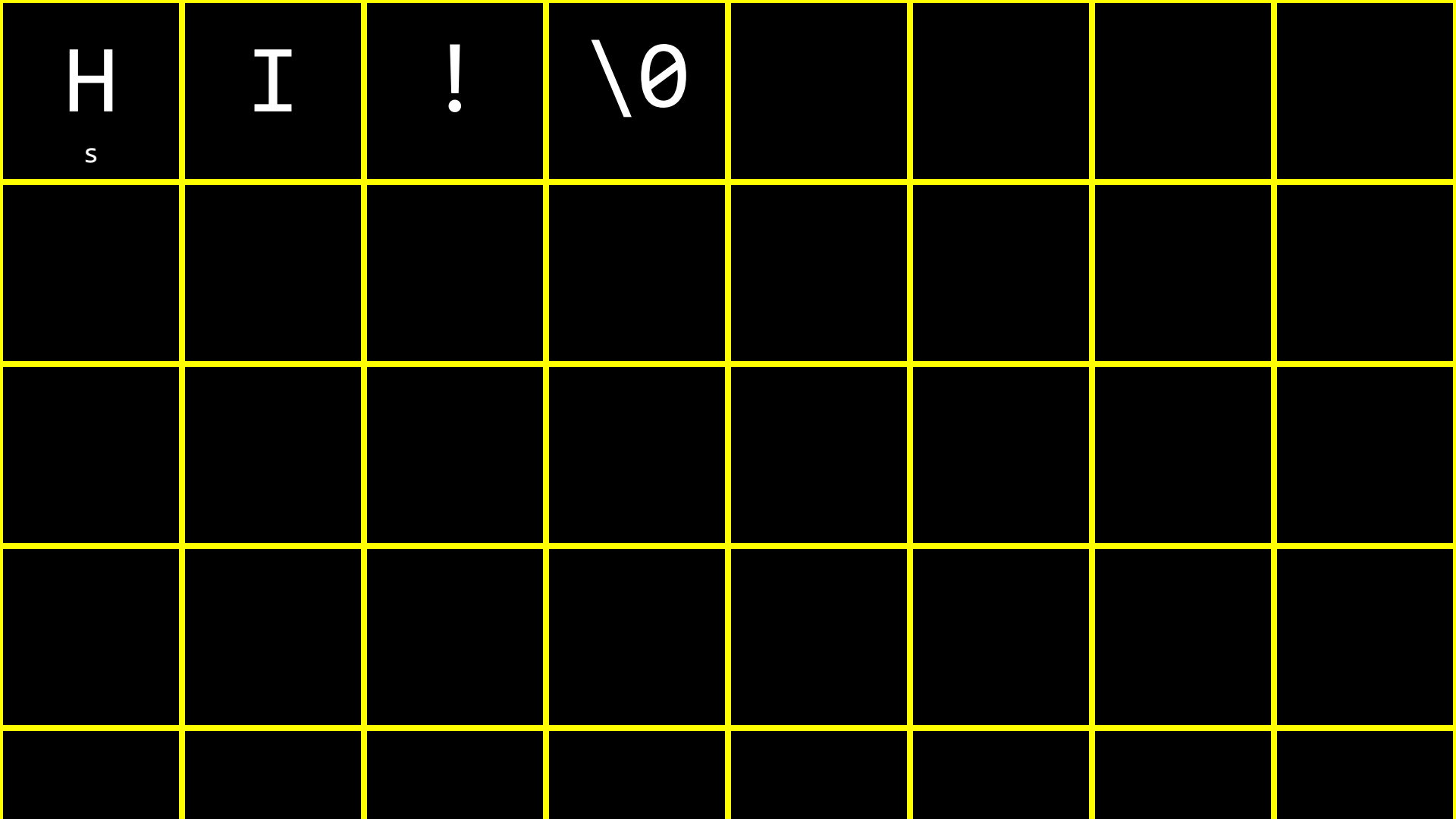

H
I
!
s[0]
s[1]
s[2]

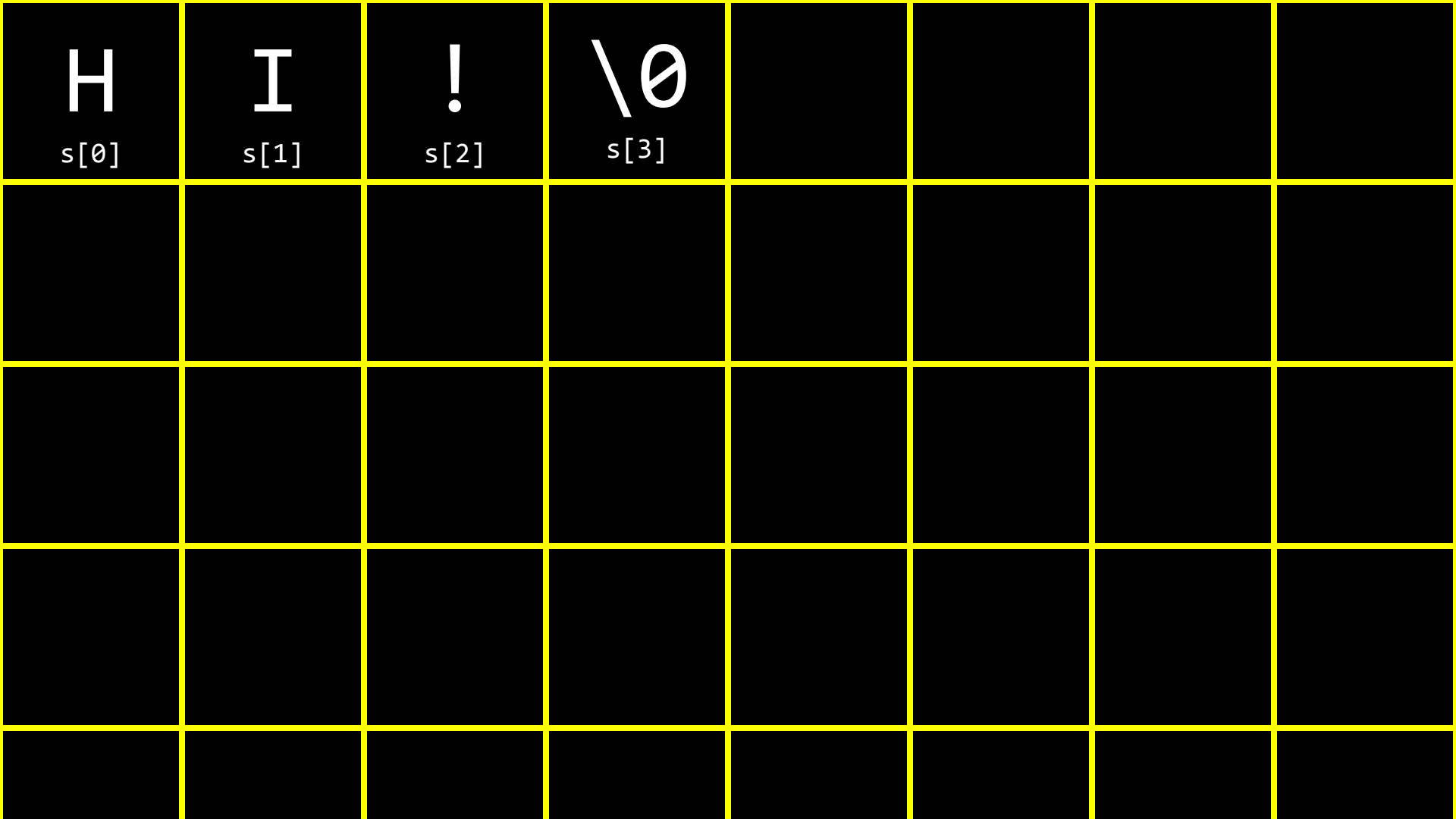
string

```
string s = "HI!";
```

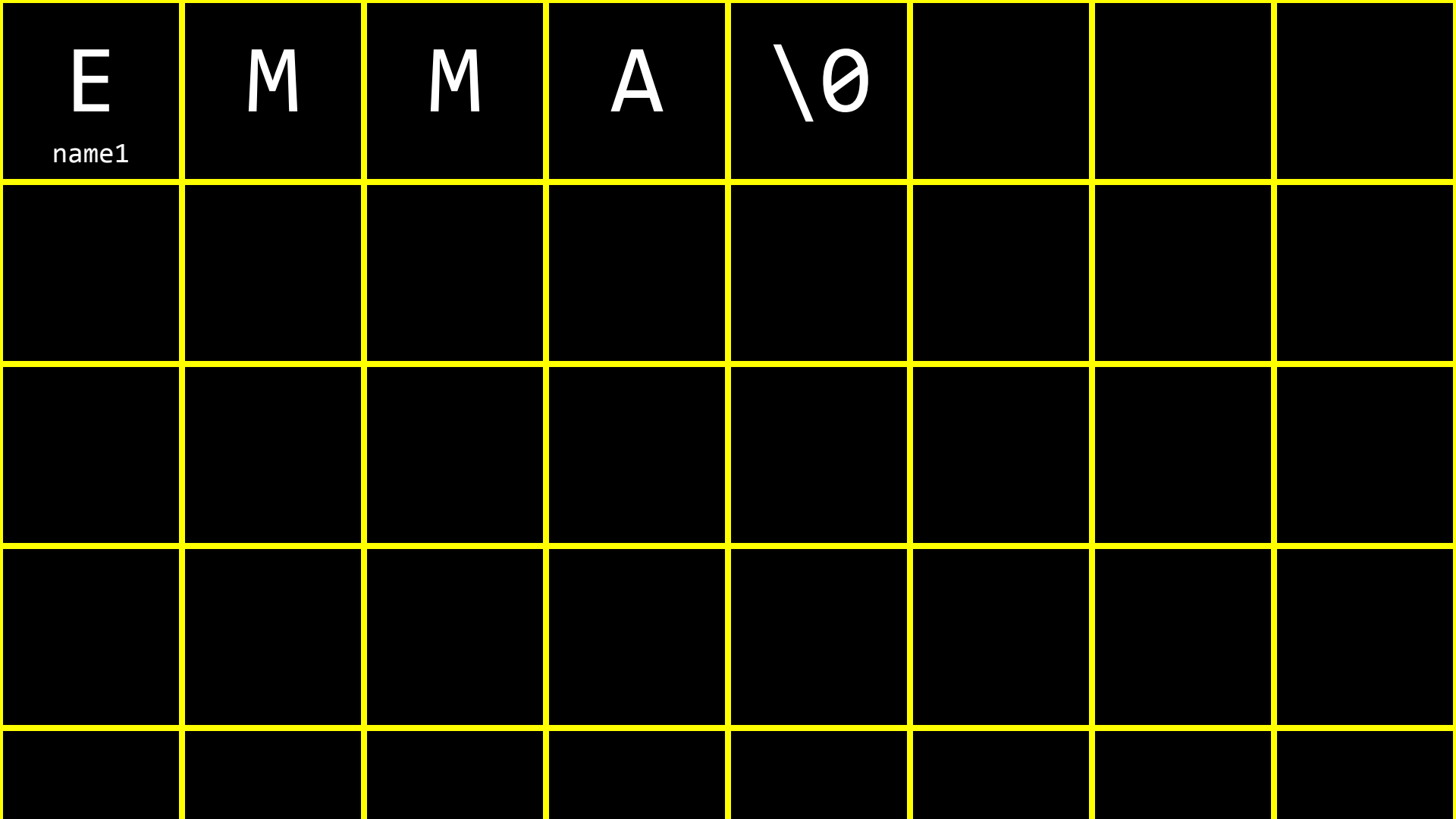
H I !

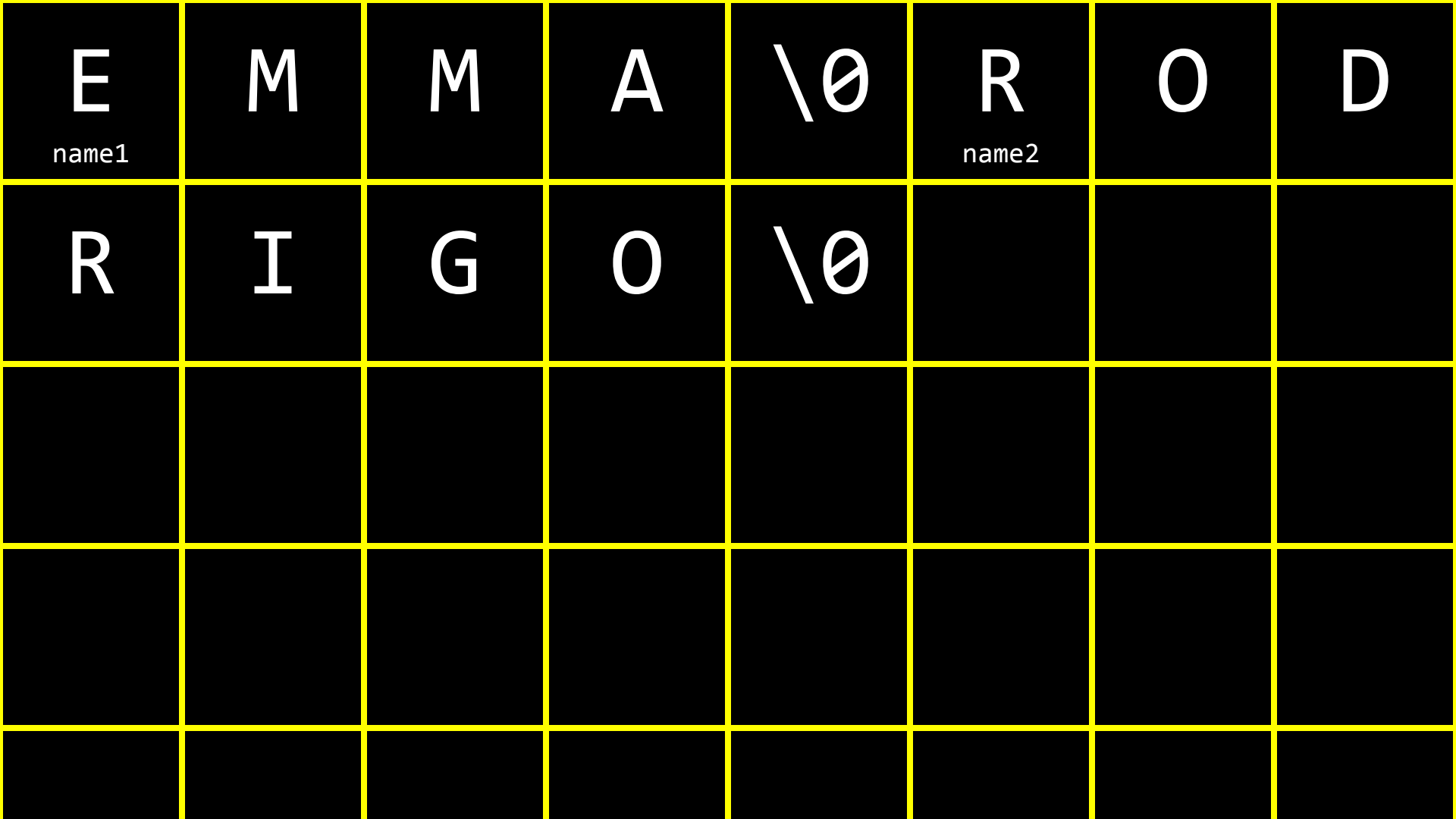
s

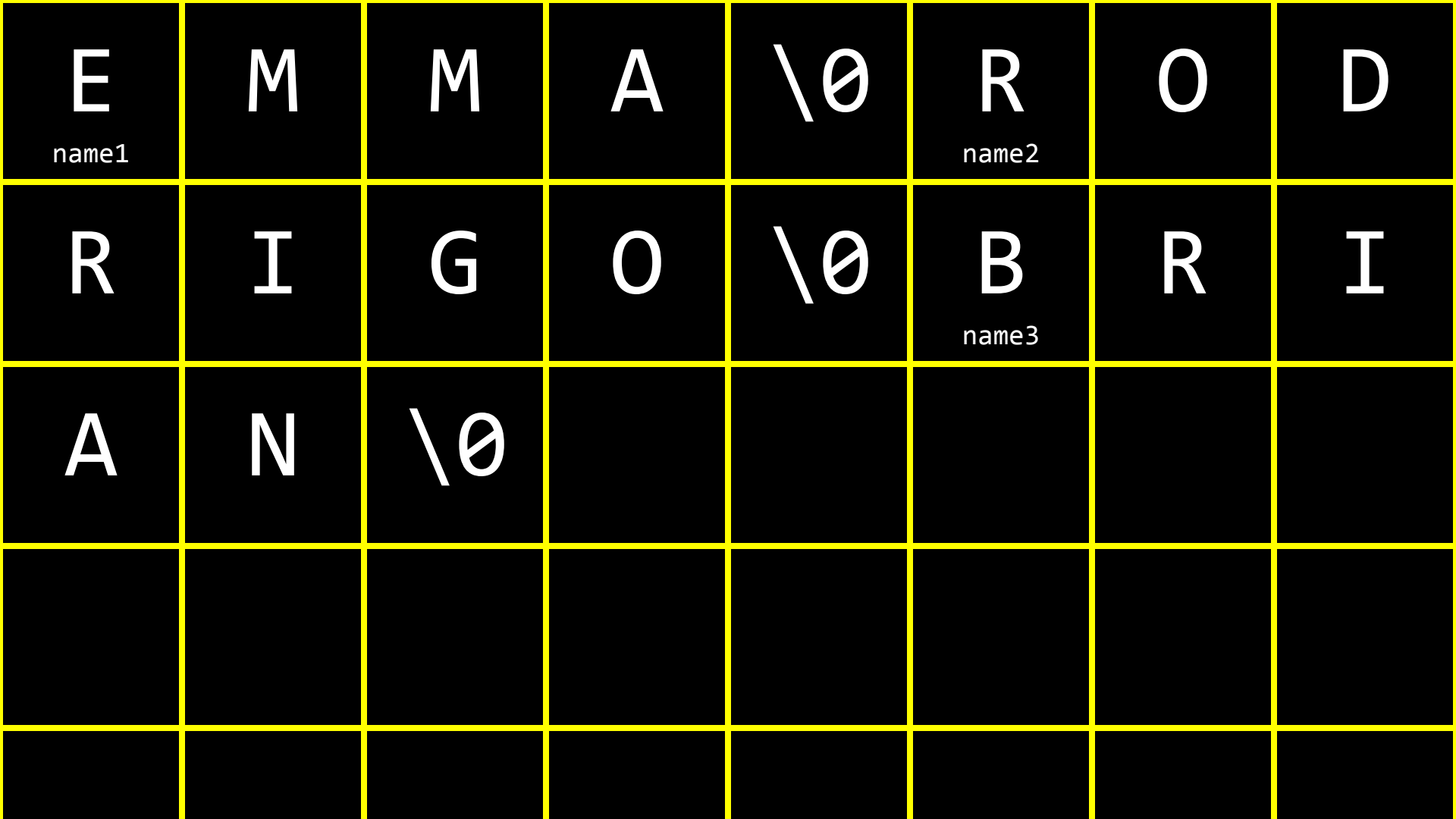

H
I
!
\0
s

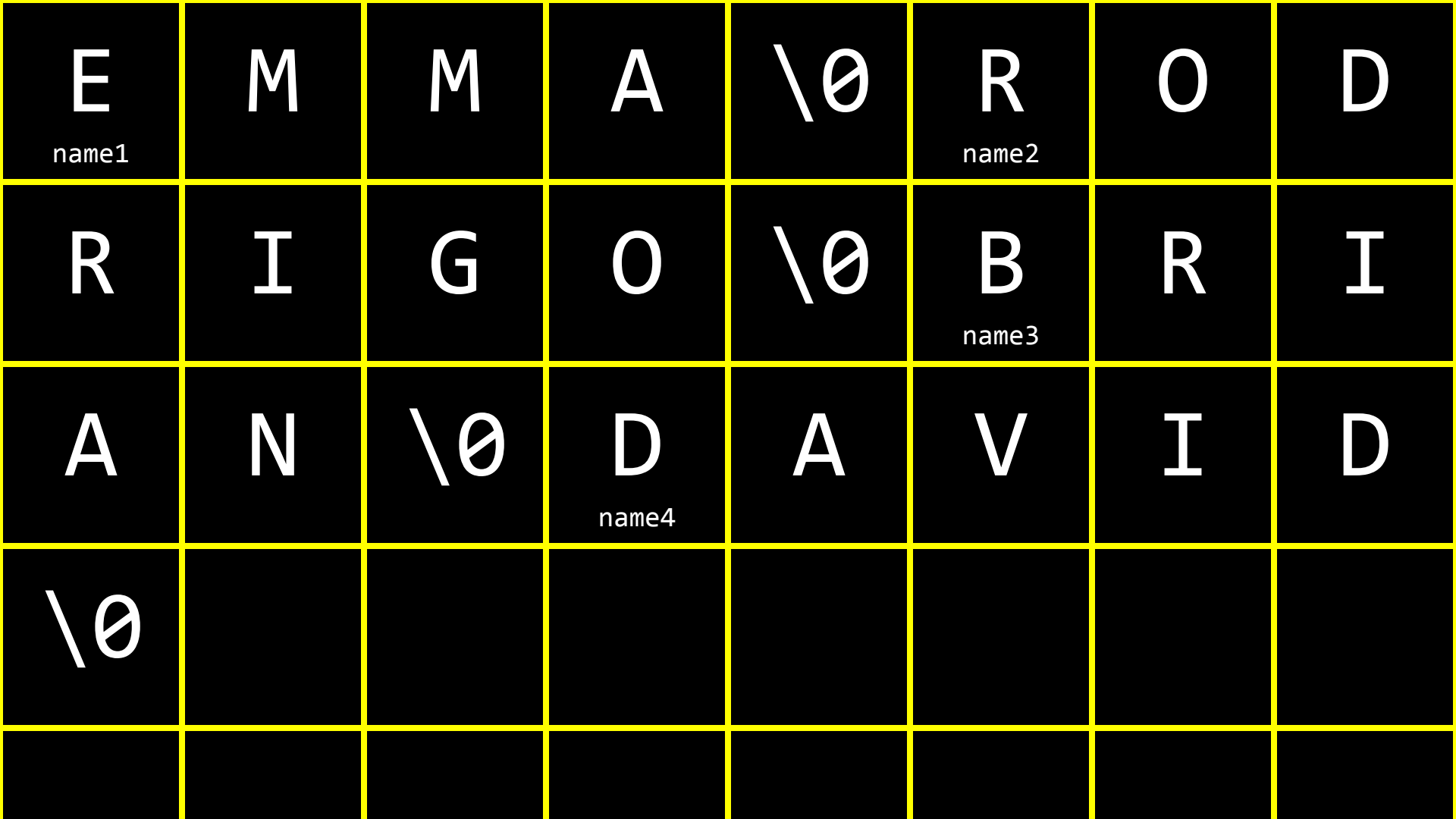

H
I
!
\0
s[0]
s[1]
s[2]
s[3]

```
string name1 = "EMMA";

string name2 = "RODRIGO";

string name3 = "BRIAN";

string name4 = "DAVID";
```

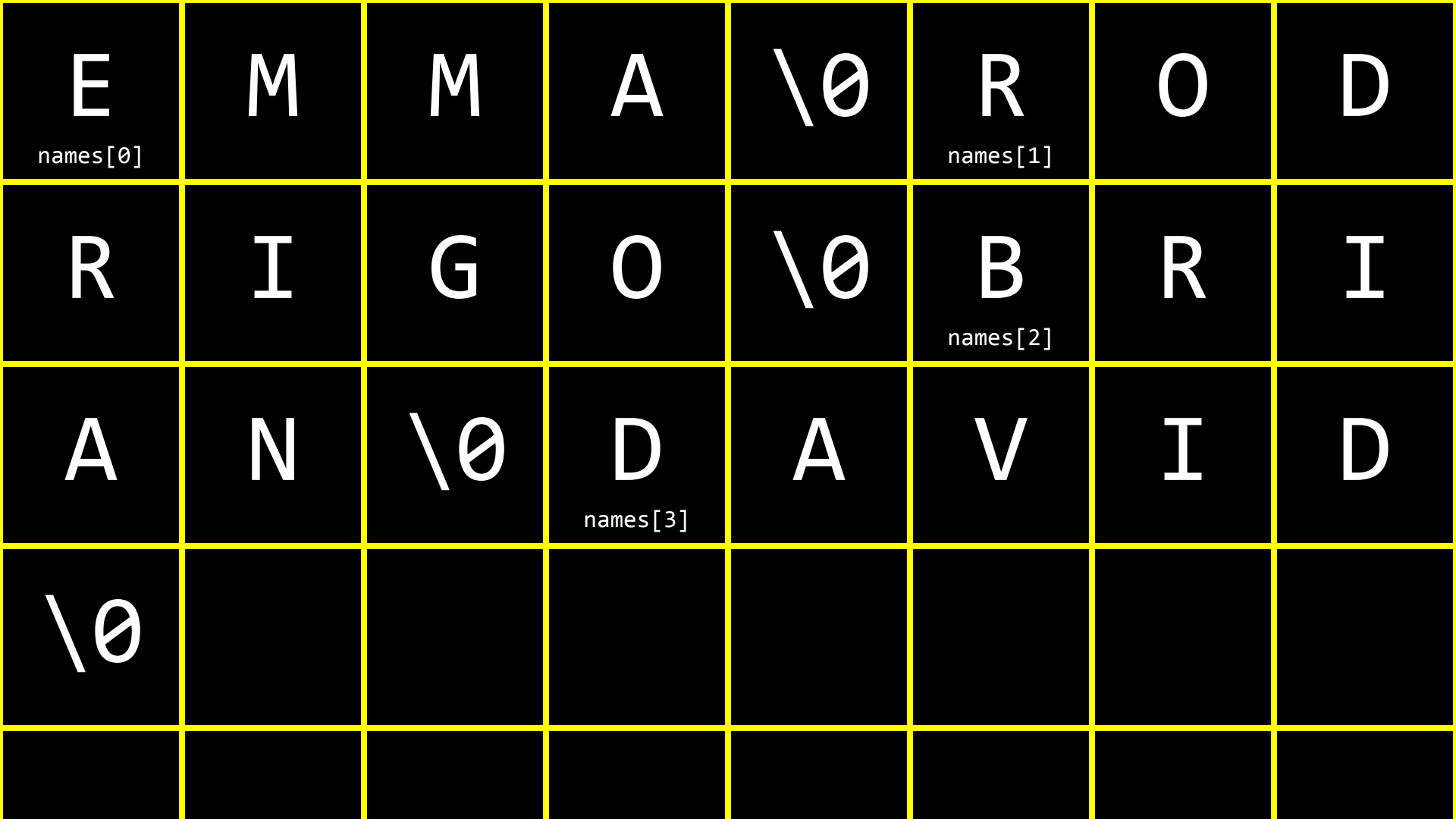

E
M
M
A
\0
name1

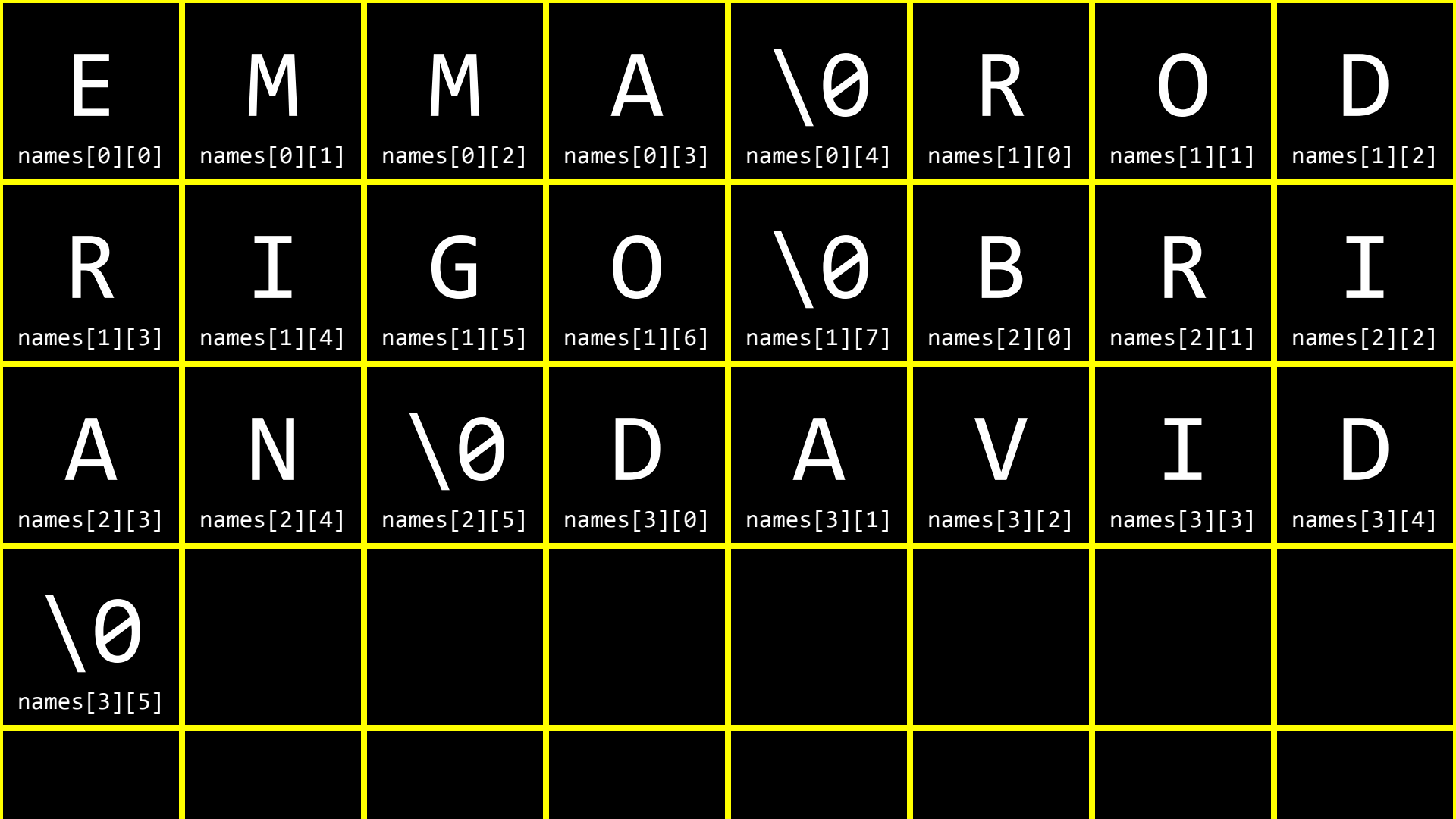

E
M
M
A
\0
R
O
D
name1
name2
R
I
G
O
\0


E
name1
M
M
A
\0
R
name2
O
D
R
I
G
O
\0
B
name3
R
I
A
N
\0


E
M
M
A
\0
R
O
D
name1
name2
R
I
G
O
\0
B
R
I
name3
A
N
\0
D
A
V
I
D
name4
\0

```
string names[4];

names[0] = "EMMA";

names[1] = "RODRIGO";

names[2] = "BRIAN";

names[3] = "DAVID";
```


E
names[0]
M
M
A
\0
R
names[1]
O
D
R
I
G
O
\0
B
names[2]
R
I
A
N
\0
D
names[3]
A
V
I
D
\0


E
names[0][0]
M
names[0][1]
M
names[0][2]
A
names[0][3]
\0
names[0][4]
R
names[1][0]
O
names[1][1]
D
names[1][2]
R
names[1][3]
I
names[1][4]
G
names[1][5]
O
names[1][6]
\0
names[1][7]
B
names[2][0]
R
names[2][1]
I
names[2][2]
A
names[2][3]
N
names[2][4]
\0
names[2][5]
D
names[3][0]
A
names[3][1]
V
names[3][2]
I
names[3][3]
D
names[3][4]
\0
names[3][5]

string

manual pages

command-line arguments

```c
#include <stdio.h>

int main(void)
{
    ...
}
```

```c
#include <stdio.h>

int main(void)
{
    ...
}
```

```
#include <stdio.h>

int main(int argc, string argv[])
{
    ...
}
```

exit status

```
#include <stdio.h>

int main(int argc, string argv[])
{
    ...
}
```

```
#include <stdio.h>

int main(int argc, string argv[])
{
    ...
}
```

```c
#include <stdio.h>

int main(void)
{
    ...
}
```

readability

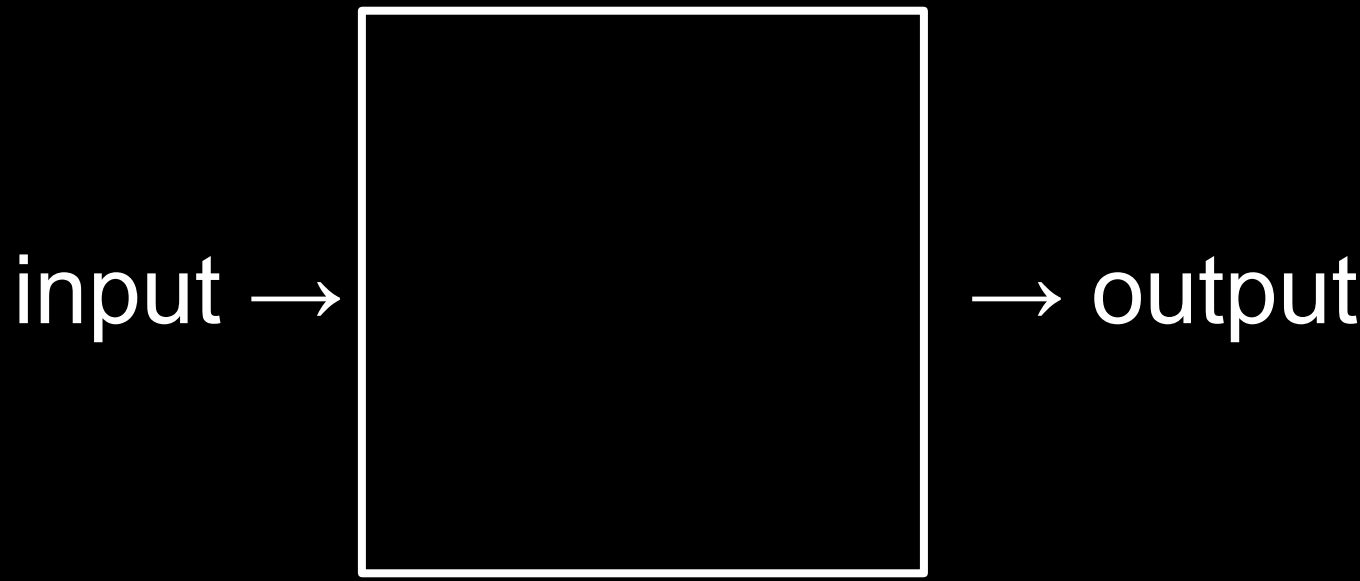
"Mr. and Mrs. Dursley, of number four, Privet Drive, were proud to say that they were perfectly normal, thank you very much. They were the last people you'd expect to be involved in anything strange or mysterious, because they just didn't hold with such nonsense..."

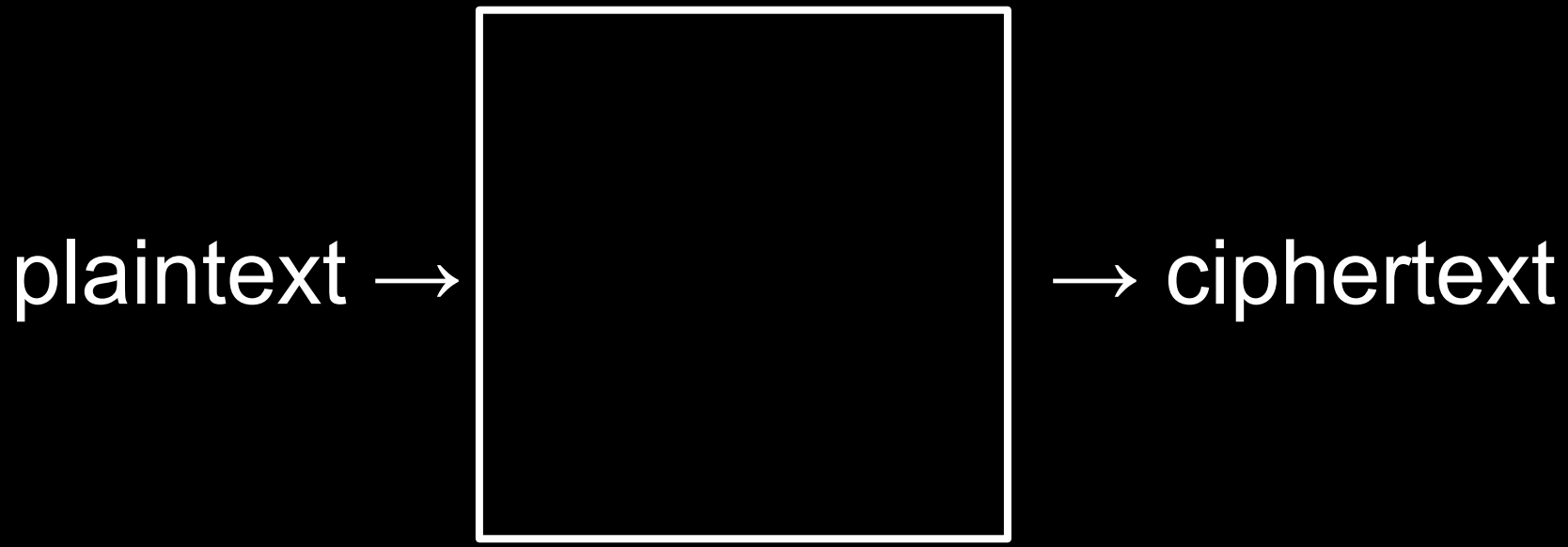
grade 7

"In computational linguistics, authorship attribution is the task of predicting the author of a document of unknown authorship. This task is generally performed via the analysis of stylometric features — particular characteristics of an author’s writing that can be used to identify his or her works in contrast with the works of other authors…"

grade 16

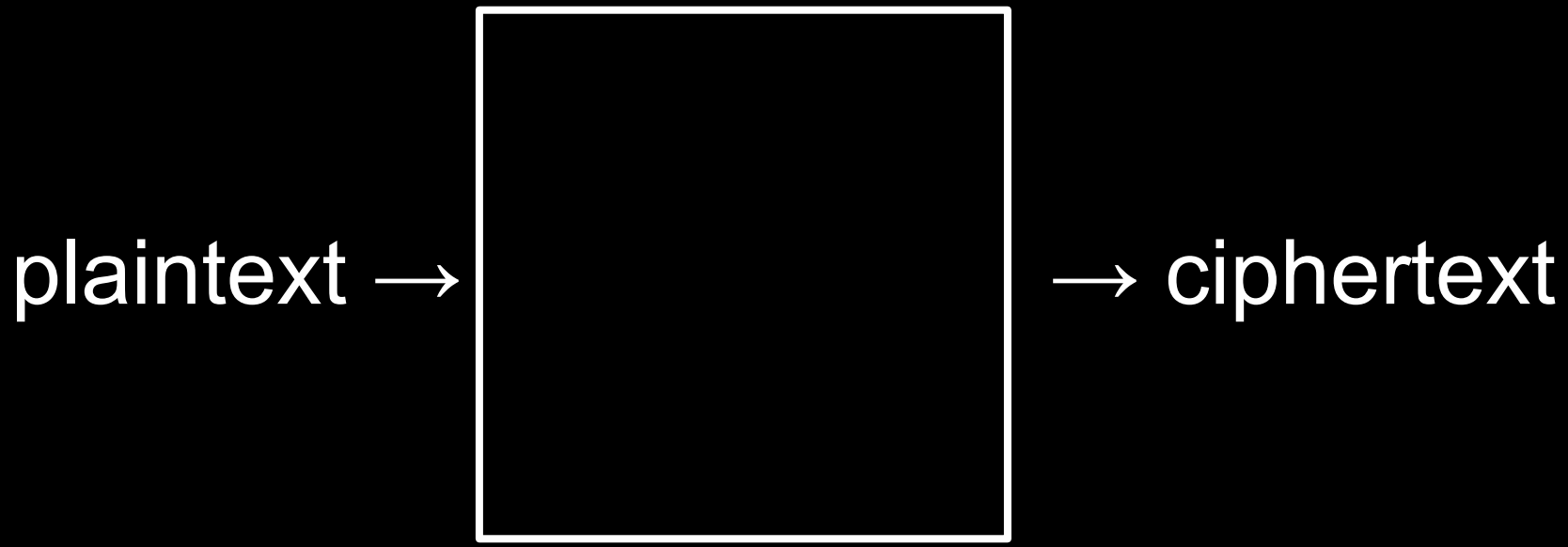
cryptography

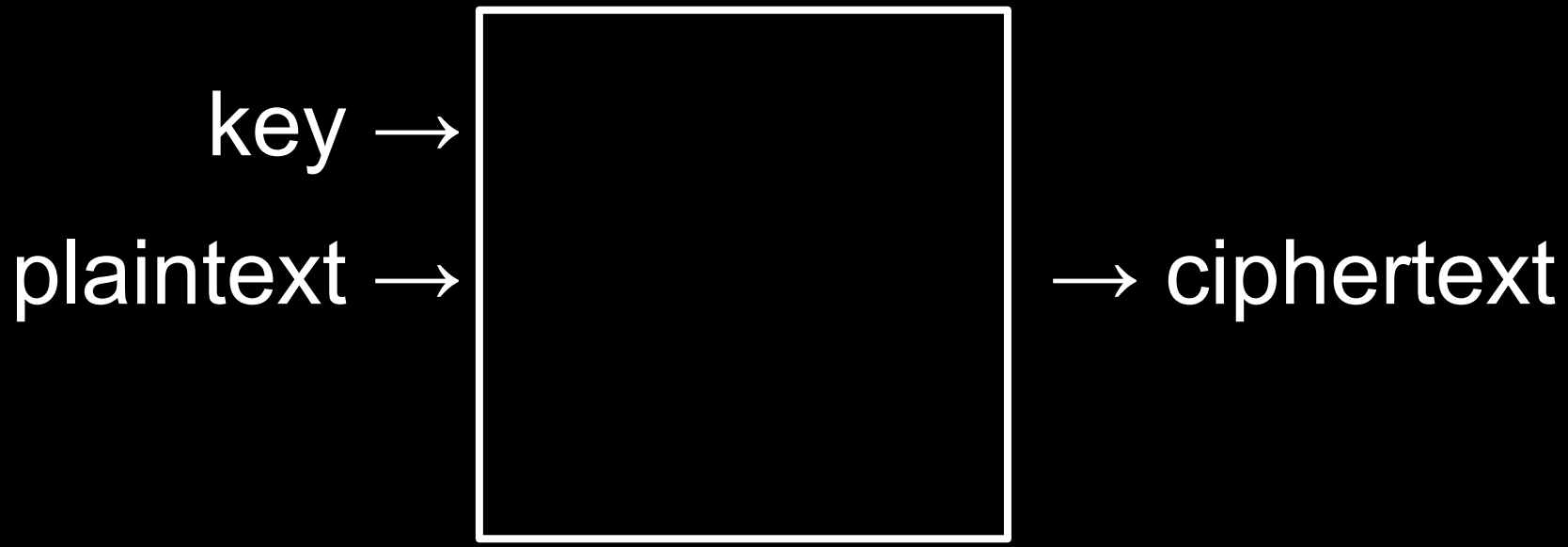

input →
→ output

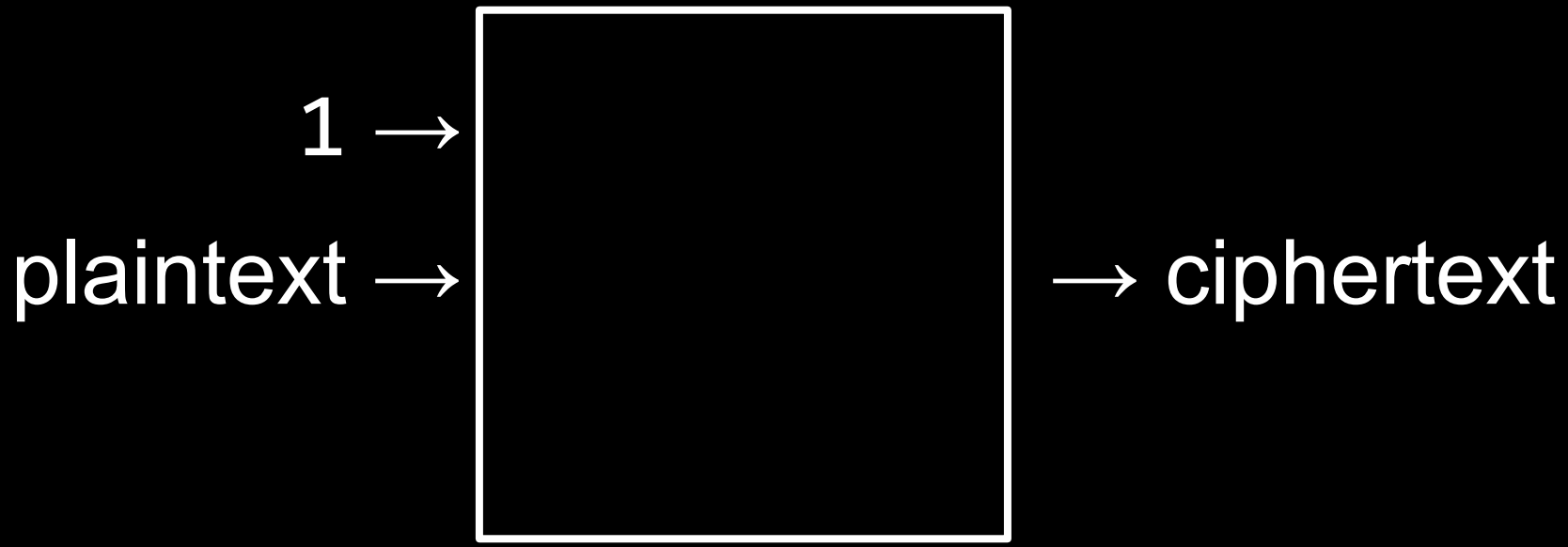

plaintext →
→ ciphertext

H I !

72 73 33


plaintext →
→ ciphertext


key →
plaintext →
→ ciphertext


1 →
plaintext →
→ ciphertext

I L O V E Y O U

73 L O V E Y O U

73 76 O V E Y O U

73 76 79 V E Y O U

73 76 79 86 E Y O U

73 76 79 86 69 Y O U

73 76 79 86 69 89 O U

73 76 79 86 69 89 79 U

73 76 79 86 69 89 79 85

74 76 79 86 69 89 79 85

74 77 79 86 69 89 79 85

74 77 80 86 69 89 79 85

74 77 80 87 69 89 79 85

74 77 80 87 70 89 79 85

74 77 80 87 70 90 79 85

74 77 80 87 70 90 80 85

74 77 80 87 70 90 80 86

```
J   77 80 87 70  90 80 86
```

J M 80 87 70 90 80 86

J M P 87 70 90 80 86

J M P W 70 90 80 86

J M P W F 90 80 86

J M P W F Z 80 86

J M P W F Z P 86

J M P W F Z P V

This is CS50