Hi! This is CS50.

- If you have trouble seeing projector, see live.cs50.io/screen for a live feed.
- If uncomfy asking questions during lecture, post to cs50.harvard.edu/discourse!
- Problem Set 3, due Thu 10/4 at 11:59pm, on website this eve.

This is CS50

File
Edit
View
PHOTOGRAPHIC EVIDENCE

compiling

preprocessing

compiling

assembling

linking

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


int main(void)
{
    string name = get_string("Name: ");
    printf("hello, %s\n", name);
}
```

```c
...
string get_string(string prompt);
int printf(const char *format, ...);
...

int main(void)
{
    string name = get_string("Name: ");
    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
    ...
```

```
01111111010001010100110001000110
00000010000000010000000100000000
00000000000000000000000000000000
00000000000000000000000000000000
00000001000000000011111000000000
00000001000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
10100000000000100000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
01000000000000000000000000000000
00000000000000000100000000000000
00001010000000000000000100000000
01010101010010001000100111100101
01001000100000111110110000010000
00110001110000001000100111000111
01001000101111110000000000000000
00000000000000000000000000000000
00000000000000001011000000000000
11101000000000000000000000000000
00000000010010001011111100000000
00000000000000000000000000000000
00000000000000000000000001001000
...
```

```
01111111010001010100110001000110
00000010000000010000000100000000
00000000000000000000000000000000
00000000000000000000000000000000
00000001000000000011111000000000
00000001000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
10100000000000010000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
01000000000000000000000000000000
00000000000000000100000000000000
00001010000000000000000100000000
01010101010010001000100111100101
01001000100000111110110000010000
00110001110000001000100111000111
01001000101111110000000000000000
00000000000000000000000000000000
00000000000000001011000000000000
11101000000000000000000000000000
00000000010010001011111100000000
00000000000000000000000000000000
00000000000000000000000001001000
...
```

cs50.c

printf.c

```
01111111010001010100110001000110
00000010000000001000000010000000
00000000000000000000000000000000
00000000000000000000000000000000
00000001000000000011111000000000
00000001000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
10100000000000010000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
01000000000000000000000000000000
00000000000000000100000000000000
00001010000000000000000100000000
01010101010010001000100111100101
01001000100000111110110000010000
00110001110000001000100111000111
01001000101111100000000000000000
00000000000000000000000000000000
00000000000000001011000000000000
11101000000000000000000000000000
00000000010010001011111100000000
00000000000000000000000000000000
00000000000000000000000001001000
...
```

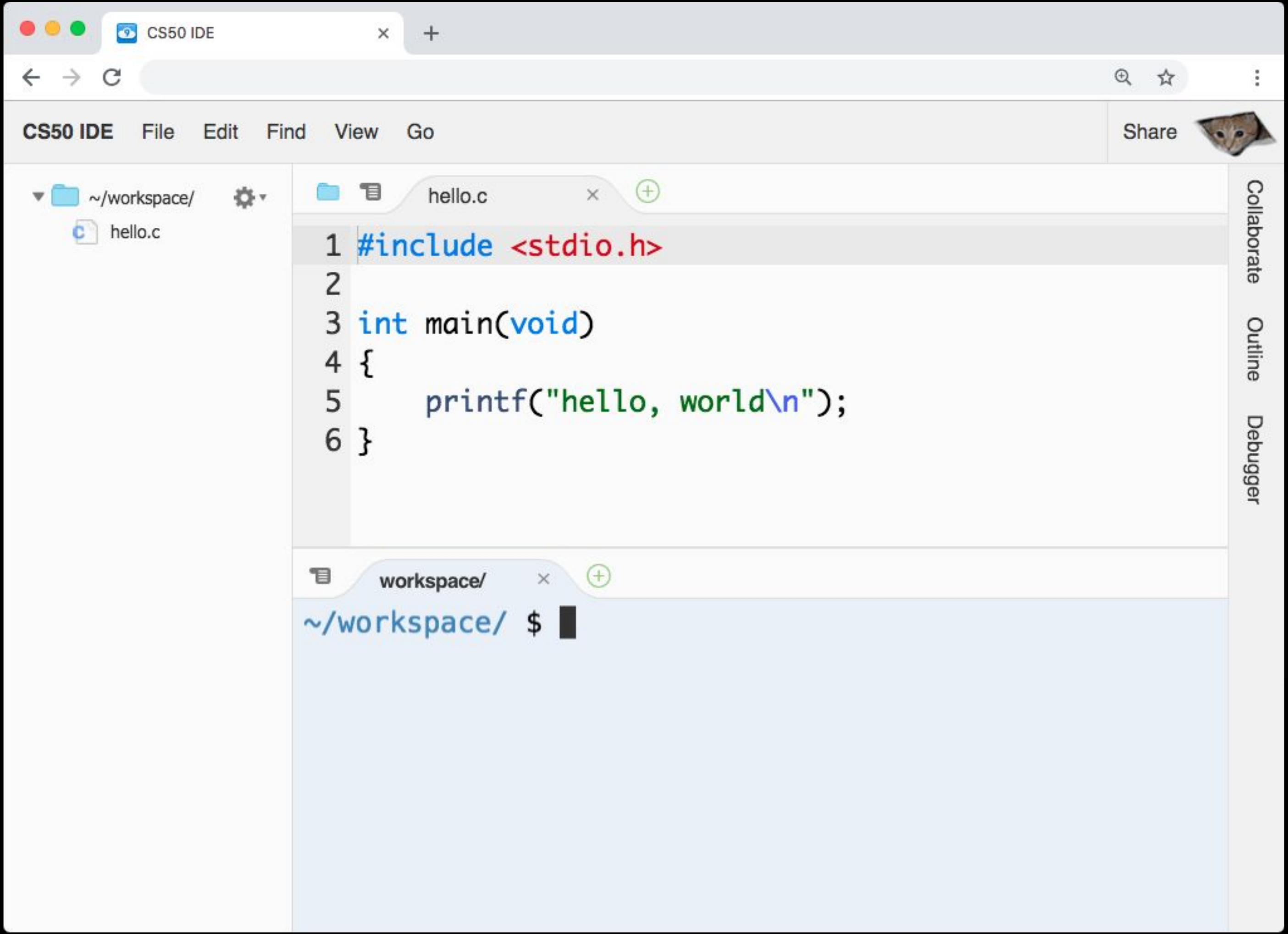
```
01111111010001010100110001000110
00000010000000001000000010000000
00000000000000000000000000000000
00000000000000000000000000000000
00000011000000000011111000000000
00000001000000000000000000000000
11000000000011110000000000000000
00000000000000000000000000000000
01000000000000000000000000000000
00000000000000000000000000000000
00101000001100100000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
01000000000000000011100000000000
00000111000000000100000000000000
00011100000000000001100100000000
00000001000000000000000000000000
00000101000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
01011100001001010000000000000000
00000000000000000000000000000000
...
```

```
00101111011011000110100101100010
01100011001011100111001101101111
00101110001101100010000000101111
01110101011100110111001000101111
01101100011010010110001000101111
01111000001110000011011001011111
00110110001101000010110101101100
01101001011011100111010101111000
00101101011001110110111001110101
00101111011011000110100101100010
01100011010111110110111001101111
01101110011100110110100001100001
01110010011001010110010000101110
01100001001000000010000001000001
01010011010111110100111001000101
01000101010001000100010101000100
00100000001010000010000000101111
01101100011010010110001000101111
01111000001110000011011001011111
00110110001101000010110101101100
01101001011011100111010101111000
00101101011001110110111001110101
00101111011011000110010000101101
01101100011010010110111001110101
01111000001011010111100000111000
00110110001011010011011000110100
...
```

```
011111110100010101001100010001100000001000000001000000010000000000000000000000000000000000000000000000
000000000000000000000000000000000100000000001111100000000000000001000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000001010000000000010000000000000000000000000000000000000000000000000000000000000000000000000
000000000100000000000000000000000000000000000000000000000100000000000000000010100000000000000001000000
000101010101001000100010011110010101001000100000111110110000010000001100011100000010001001110001110100
100010111110000000000000000000000000000000000000000000000000000000000000000000001011000000000000111010
000000000000000000000000000000010010001011111100000000000000000000000000000000000000000000000000000000
0000000001001000...01111111010001010100110001000110000000100000000100000001000000000000000000000000000
000000000000000000000000000000000000000000000000000110000000000111110000000000000000100000000000000000
000000011000000000011111000000000000000000000000000000000000000000000000100000000000000000000000000000
000000000000000000000000000000000001010000011001000000000000000000000000000000000000000000000000000000
000000000000000000000000000010000000000000000111000000000000000011100000000010000000000000000011100000
000000001100100000000000000010000000000000000000000000000010100000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000001011100001001010000000000000
00000000000000000000000000000000000...0010111101101100011010010110001001100011001011100111001101101111
001011100011011000100000001011110111010101110011011100100010111101101100011010010110001000101111011110
000011100000110110010111110011011000110100001011010110110001101001011011100111010101111000001011010110
011101101110011101010010111101101100011010010110001001100011010111110110111001101111011011100111001101
101000011000010111001001100101011001000010111001100001001000000010000001000001010100110101111101001110
010001010100010101000100010001010100010000100000001010000010000000101111011011000110100101100010001011
110111100000111000001101100101111100110110001101000010110101101100011010010110111001110101011110000010
110101100111011011100111010100101111011011000110010000101101011011000110100101101110011101010111100000
101101011110000011100000110110001011010011011000110100...
```

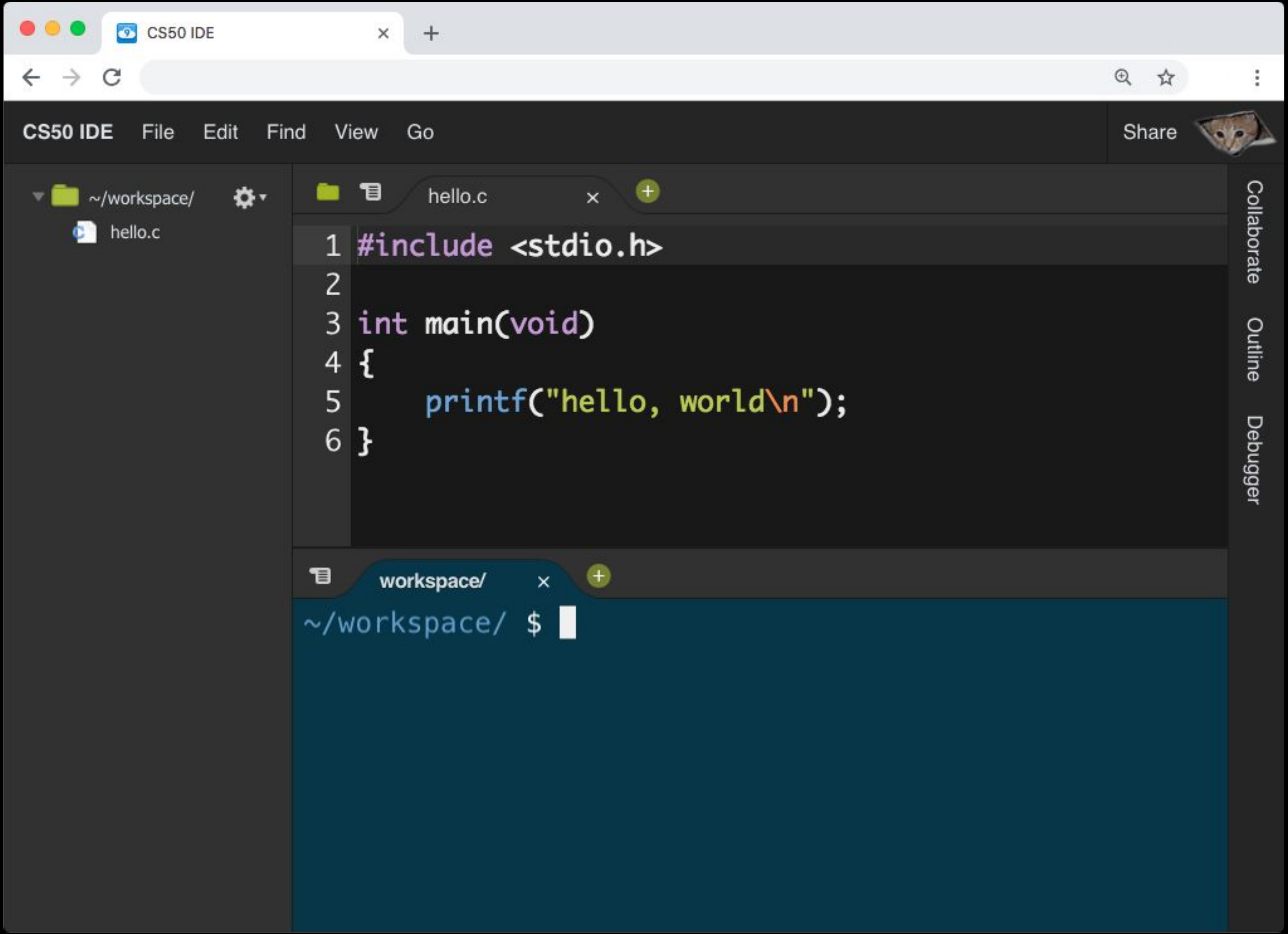
```
help50

printf

style50
```

CS50 IDE



```c
#include <stdio.h>

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

```
~/workspace/ $ 
```



```c
#include <stdio.h>

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

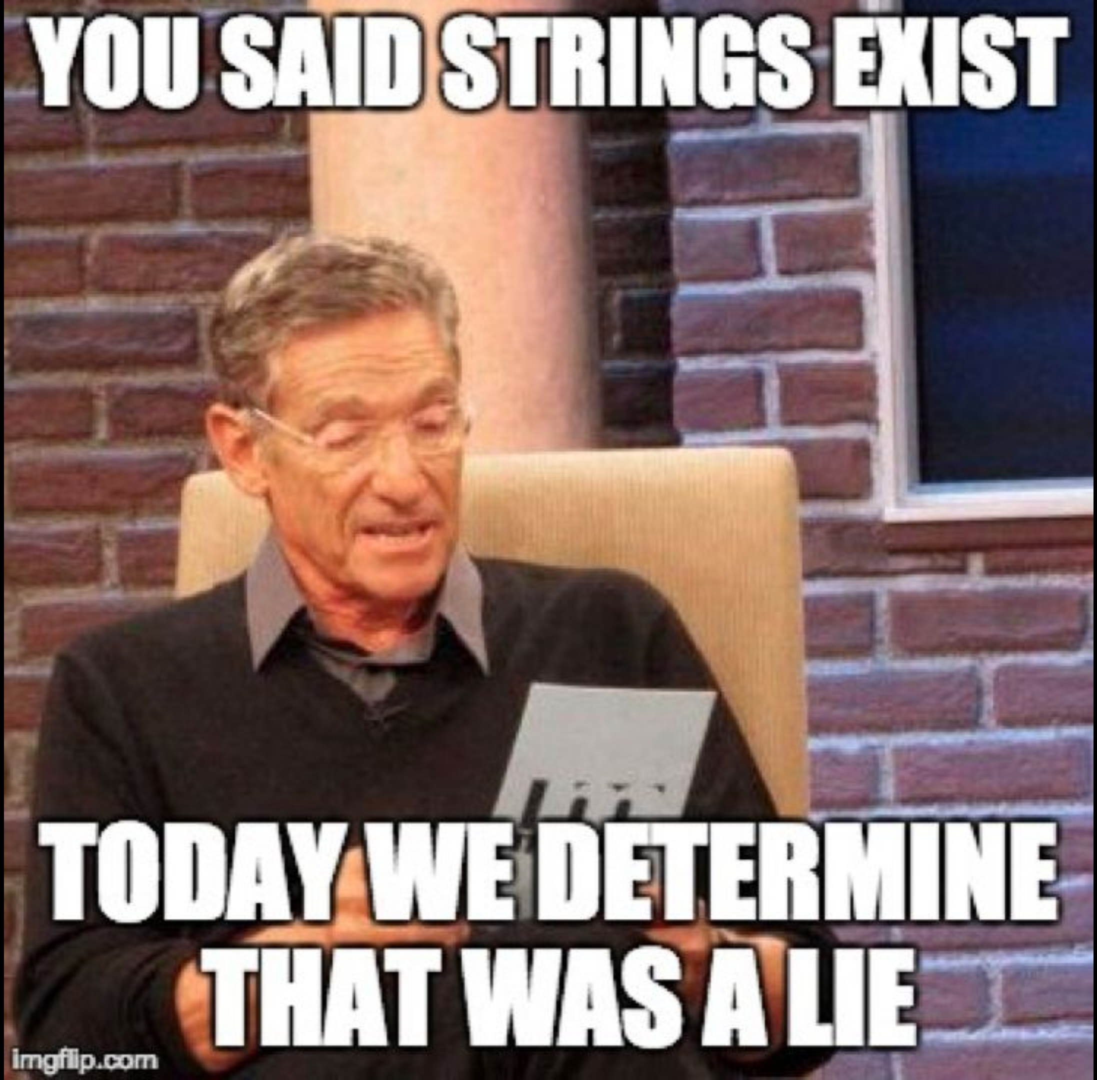
```
~/workspace/ $ 
```

```
cd

ls

mkdir

rm

rmdir

…
```

```
check50

debug50
```

```
get_char

get_double

get_float

get_int

get_long

get_string

...
```

string



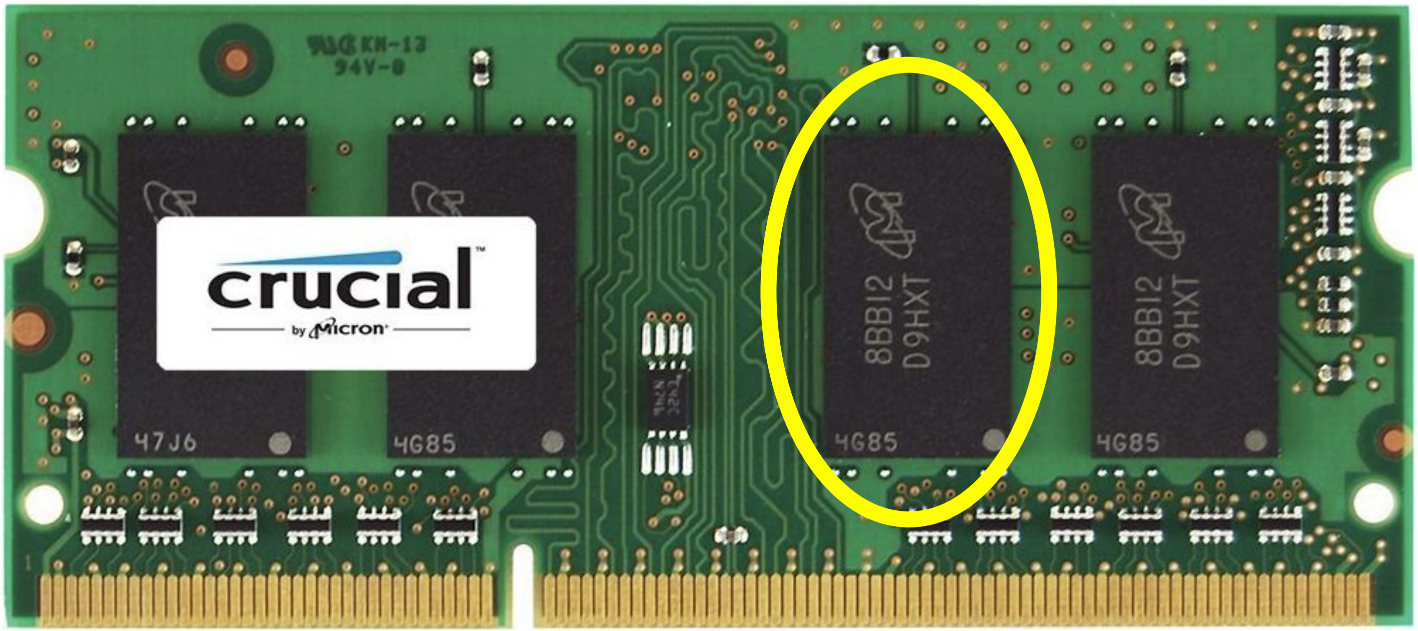
YOU SAID STRINGS EXIST
TODAY WE DETERMINE THAT WAS A LIE
imgflip.com

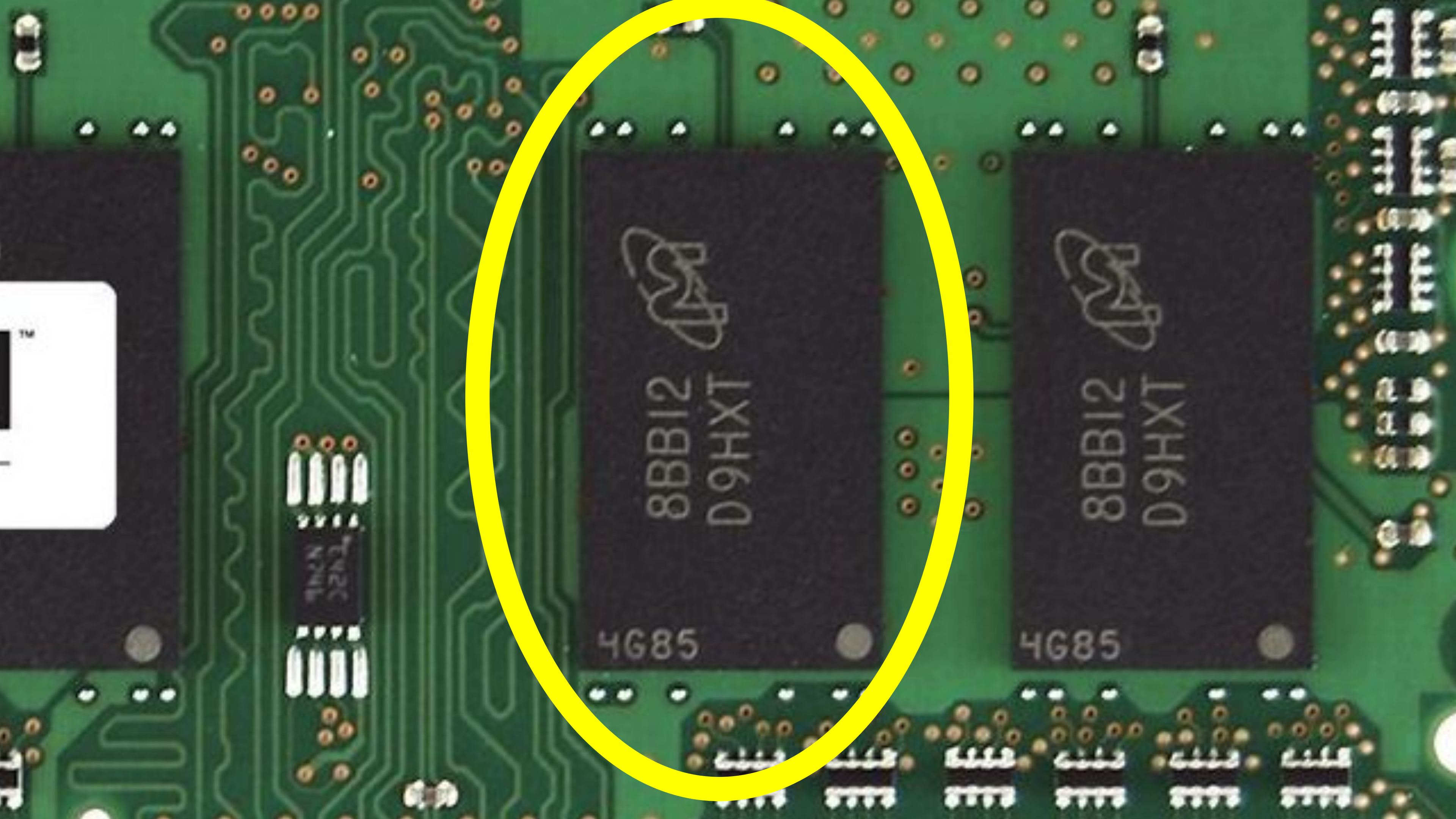
string

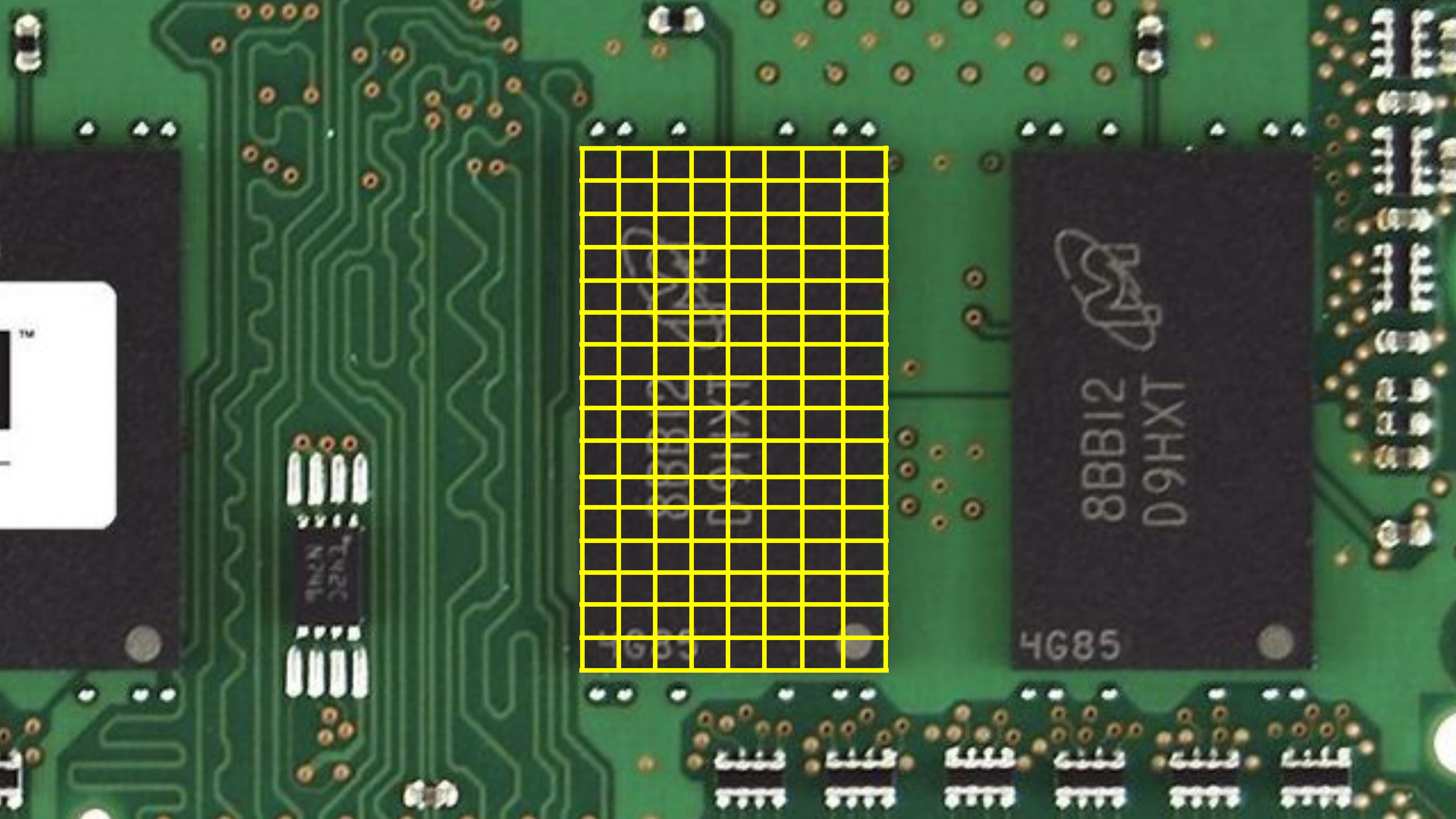
```
char *
```

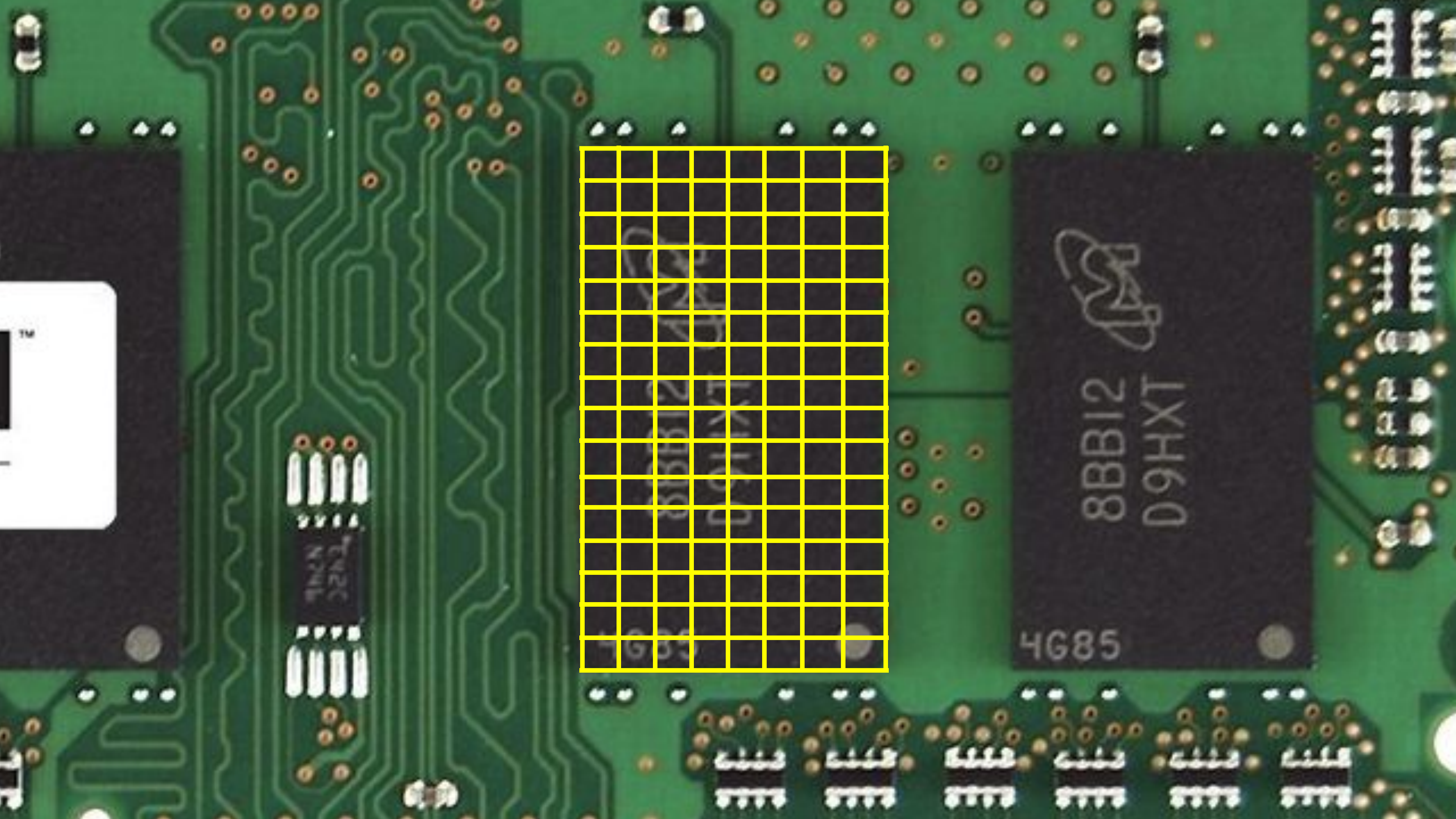
```
malloc

free

…
```

crucial™
by Micron
KN-13
94V-0
47J6
4G85
8BBI2
D9HXT
4G85
8BBI2
D9HXT
4G85


KN-13
94V-0
crucial
by Micron
47J6
4G85
8BBI2
D9HXT
4G85
8BBI2
D9HXT
4G85


8BB12
D9HXT
4G85
8BB12
D9HXT
4G85


8BB12
D9HXT
4G85
™


8BB12
D9HXT
4G85
8BB12
D9HXT
4G85

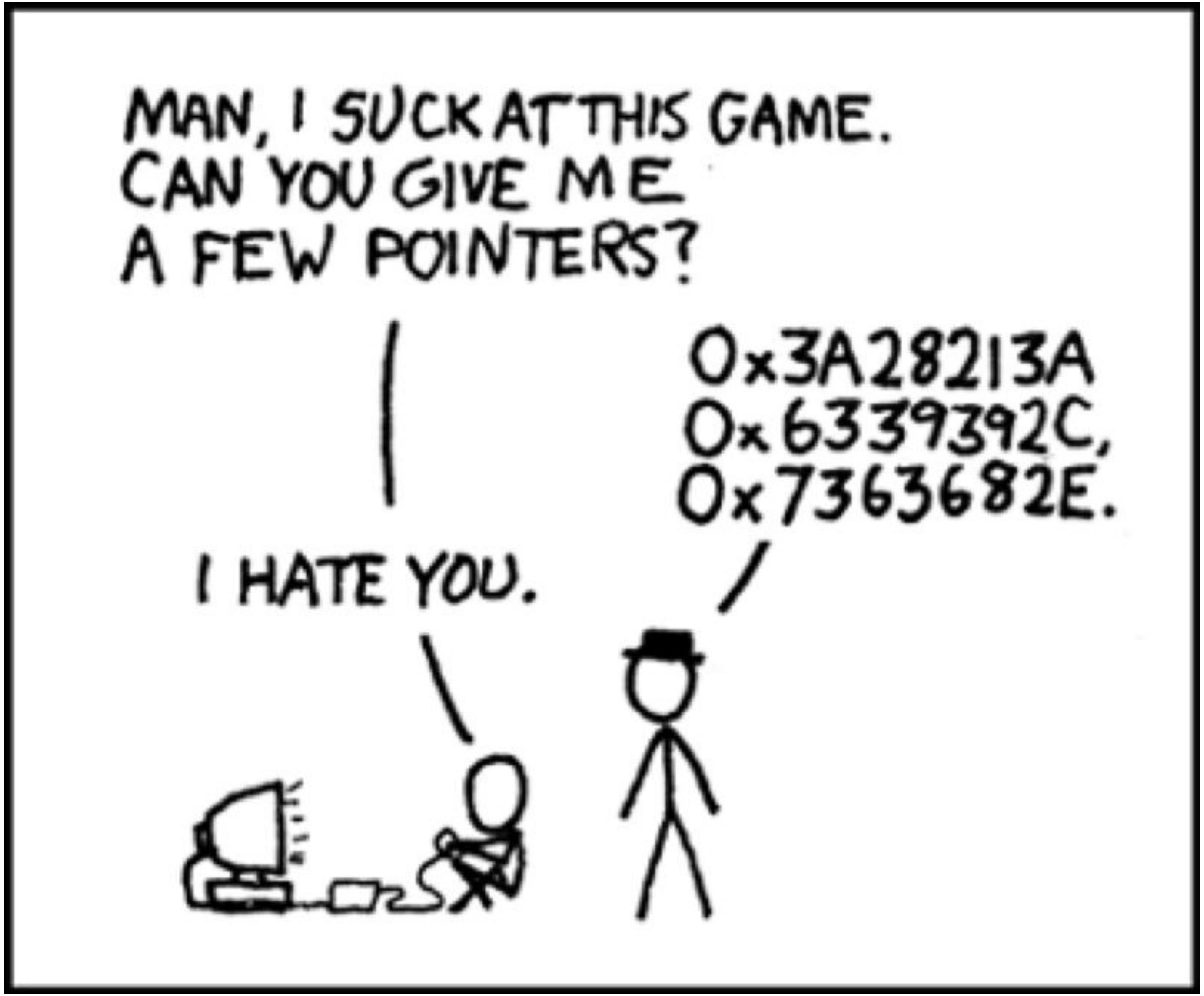
255 216 255

255	216	255
11111111	11011000	11111111

255	216	255
1111 1111	1101 1000	1111 1111

255	216	255
1111 1111	1101 1000	1111 1111
f f	d 8	f f

```
0xff     0xd8     0xff
```


MAN, I SUCK AT THIS GAME. CAN YOU GIVE ME A FEW POINTERS?
0x3A28213A 0x6339392C, 0x7363682E.
I HATE YOU.

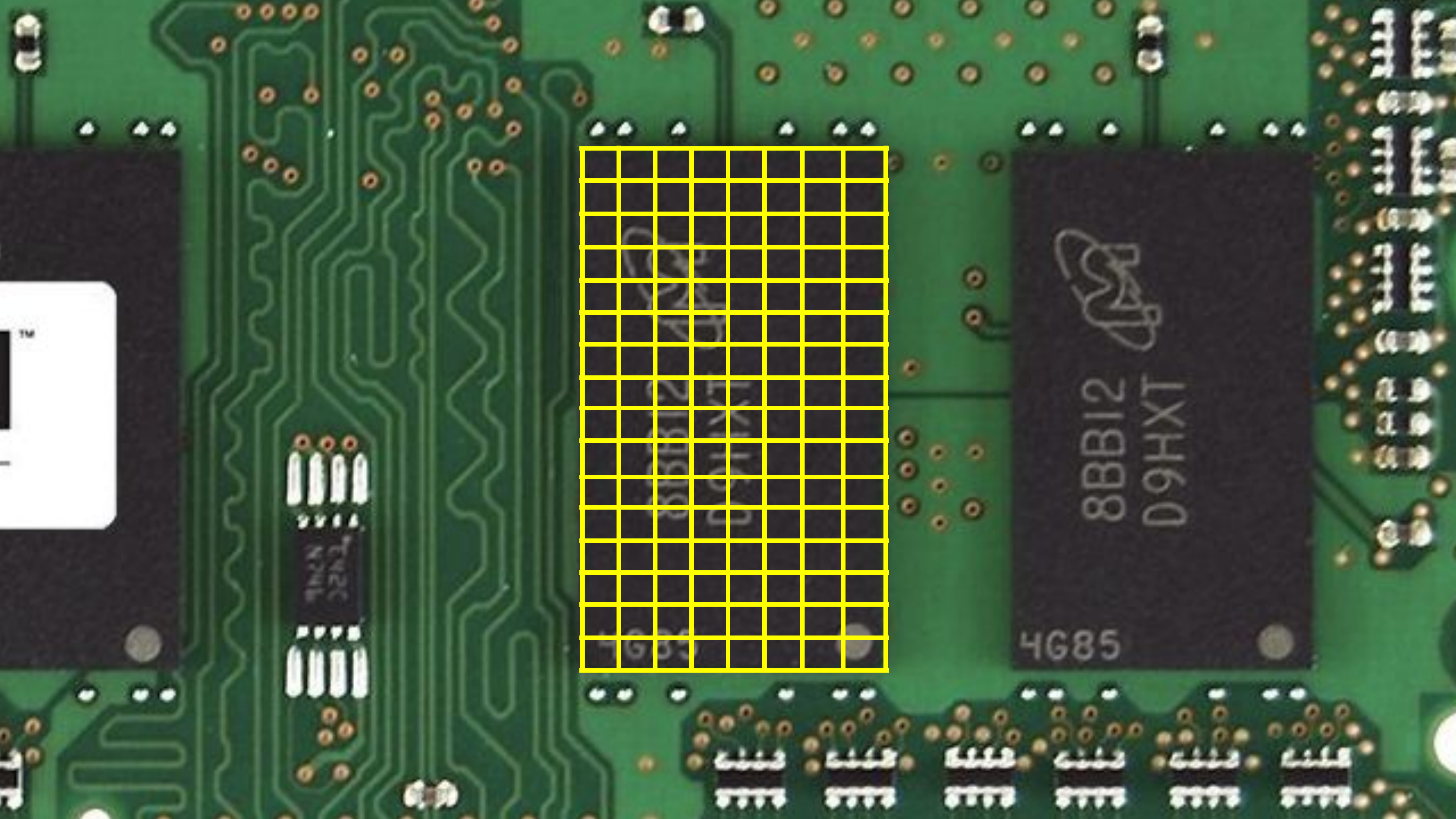
```
void swap(int a, int b)
{



}
```

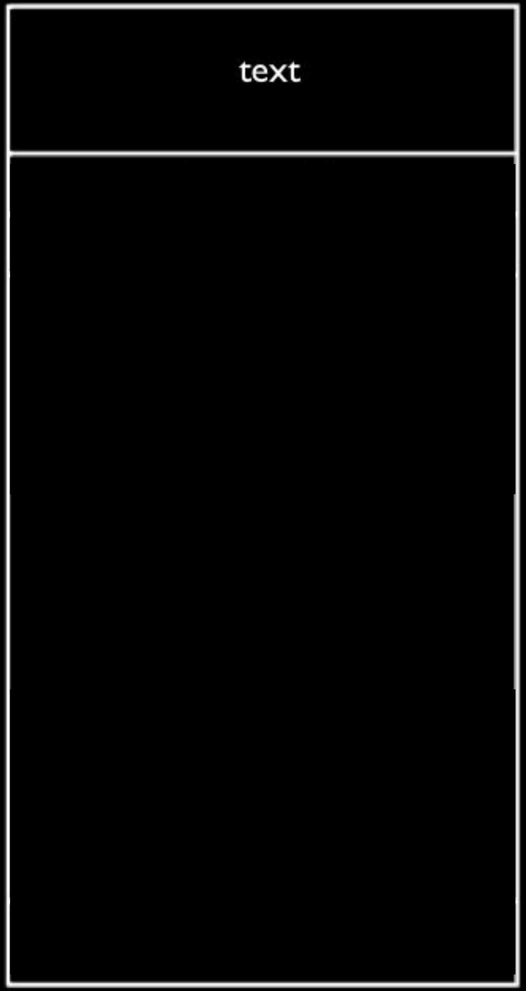
```
void swap(int a, int b)
{
    int tmp = a;
    a = b;
    b = tmp;
}
```

```
void swap(int a, int b)
{
    int tmp = a;
    a = b;
    b = tmp;
}
```

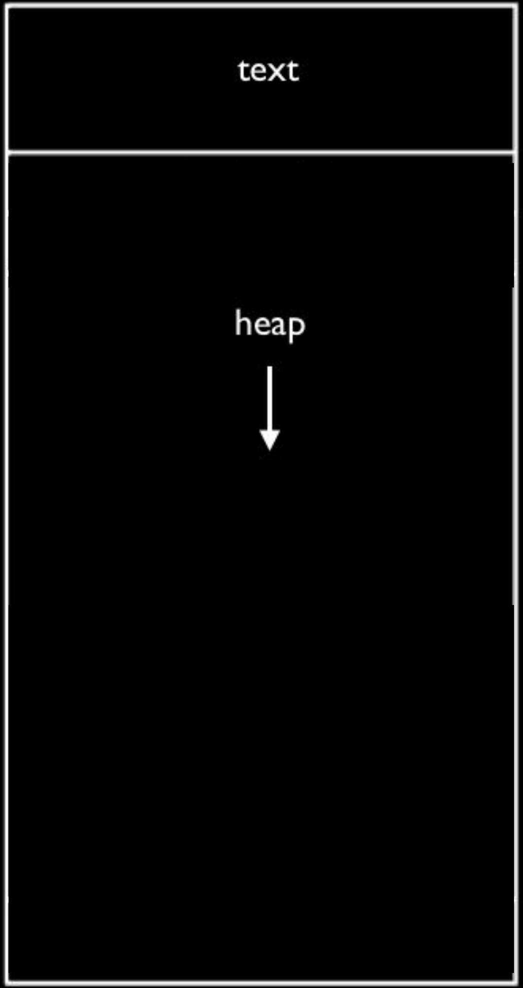
```
void swap(int a, int b)
{
    int tmp = a;
    a = b;
    b = tmp;
}
```

```
void swap(int *a, int *b)
{
    int tmp = *a;
    *a = *b;
    *b = tmp;
}
```


8BB12
D9HXT
4G85
8BB12
D9HXT
4G85

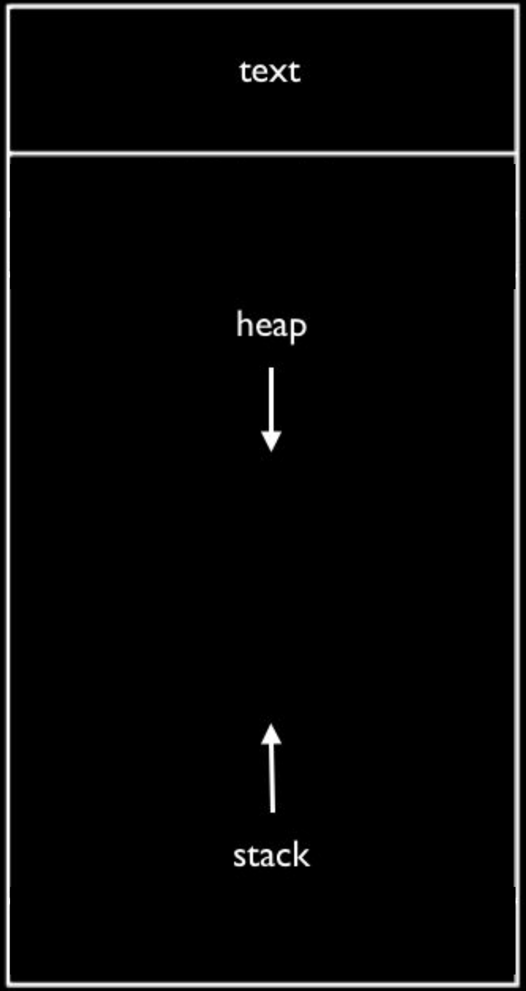


text



text

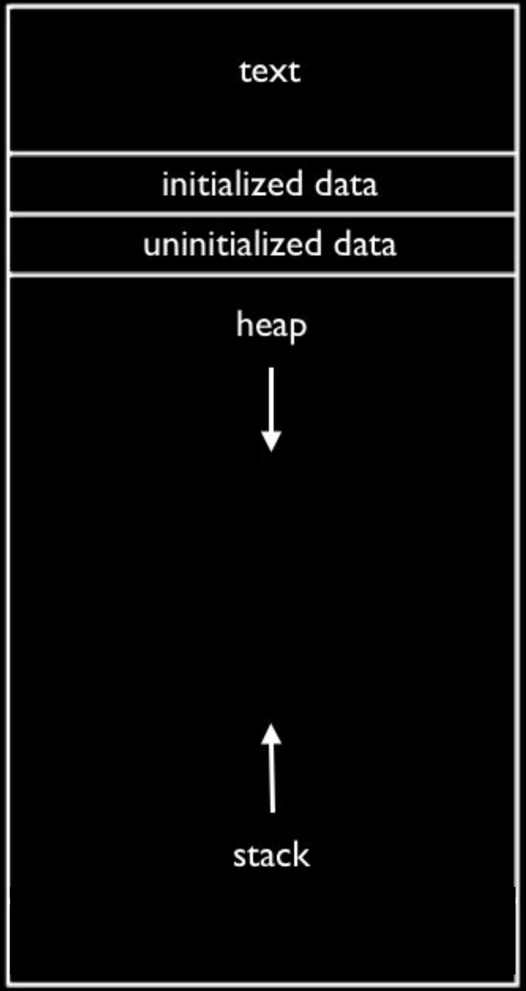
heap

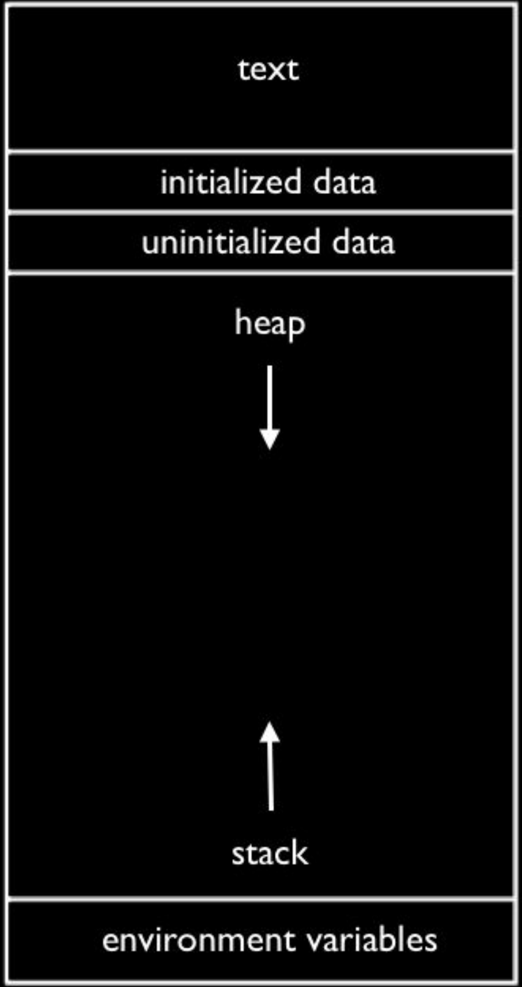


text

heap

stack


text
initialized data
uninitialized data
heap
stack



text

initialized data

uninitialized data

heap

stack

environment variables

```
int main(void)
{
    int *x;
    int *y;

    x = malloc(sizeof(int));

    *x = 42;
    *y = 13;

    y = x;

    *y = 13;
}
```

```
int main(void)
{
    int *x;
    int *y;

    x = malloc(sizeof(int));

    *x = 42;
    *y = 13;

    y = x;

    *y = 13;
}
```

```
int main(void)
{
    int *x;
    int *y;

    x = malloc(sizeof(int));

    *x = 42;
    *y = 13;

    y = x;

    *y = 13;
}
```

```
int main(void)
{
    int *x;
    int *y;

    x = malloc(sizeof(int));

    *x = 42;
    *y = 13;

    y = x;

    *y = 13;
}
```

```
int main(void)
{
    int *x;
    int *y;

    x = malloc(sizeof(int));

    *x = 42;
    *y = 13;

    y = x;

    *y = 13;
}
```

```
int main(void)
{
    int *x;
    int *y;

    x = malloc(sizeof(int));

    *x = 42;
    *y = 13;

    y = x;

    *y = 13;
}
```

```
int main(void)
{
    int *x;
    int *y;

    x = malloc(sizeof(int));

    *x = 42;
    *y = 13;

    y = x;

    *y = 13;
}
```

Pointer Fun with Binky

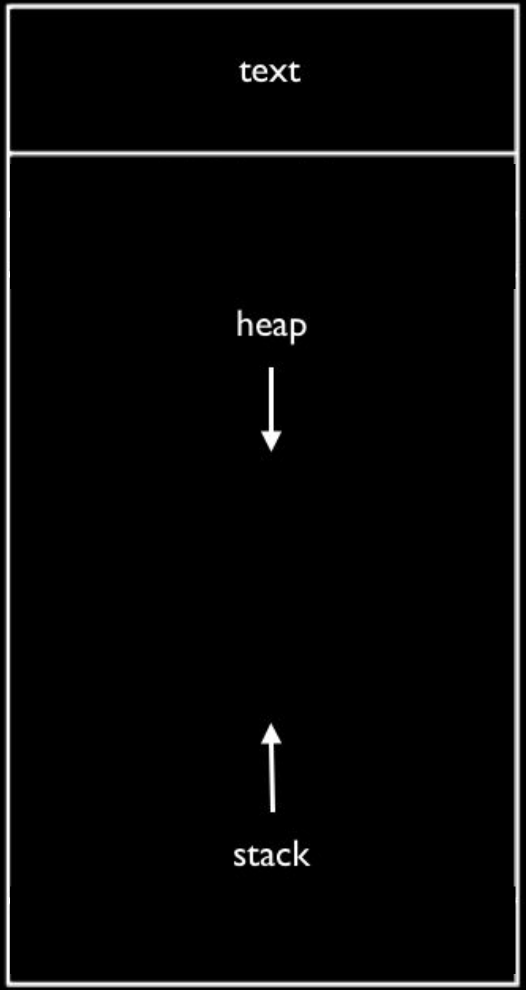
by Nick Parlante

This is document 104 in the Stanford CS Education Library — please see cslibrary.stanford.edu for this video, its associated documents, and other free educational materials.

Carpe Post Meridiem!

valgrind

stackoverflow

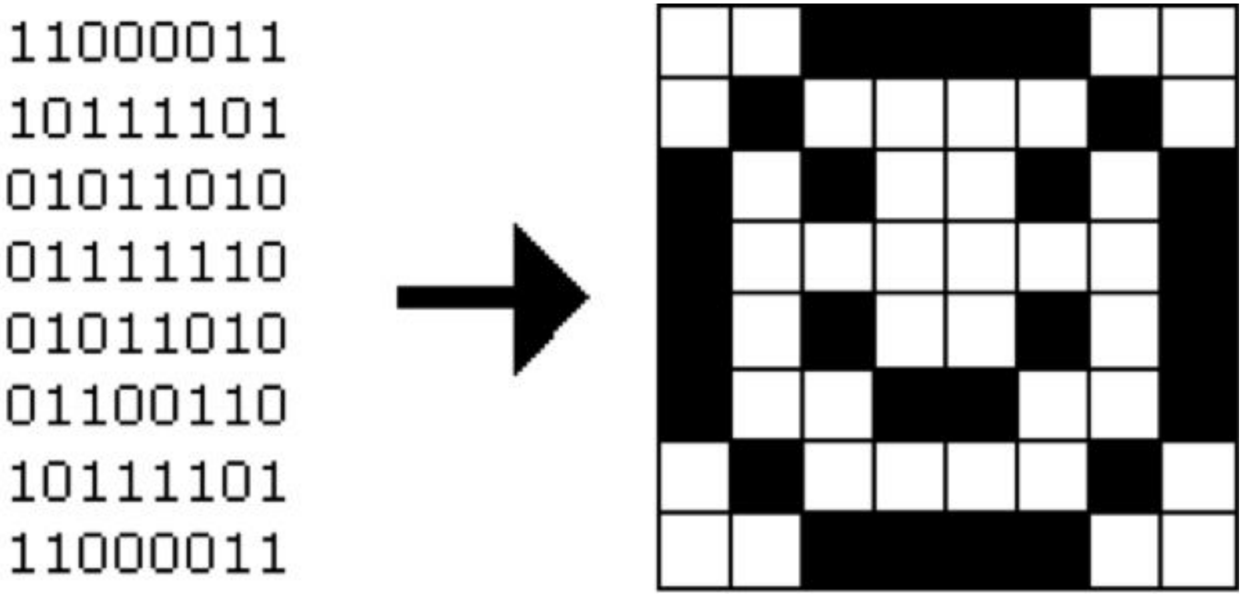

text
heap
stack

stack overflow

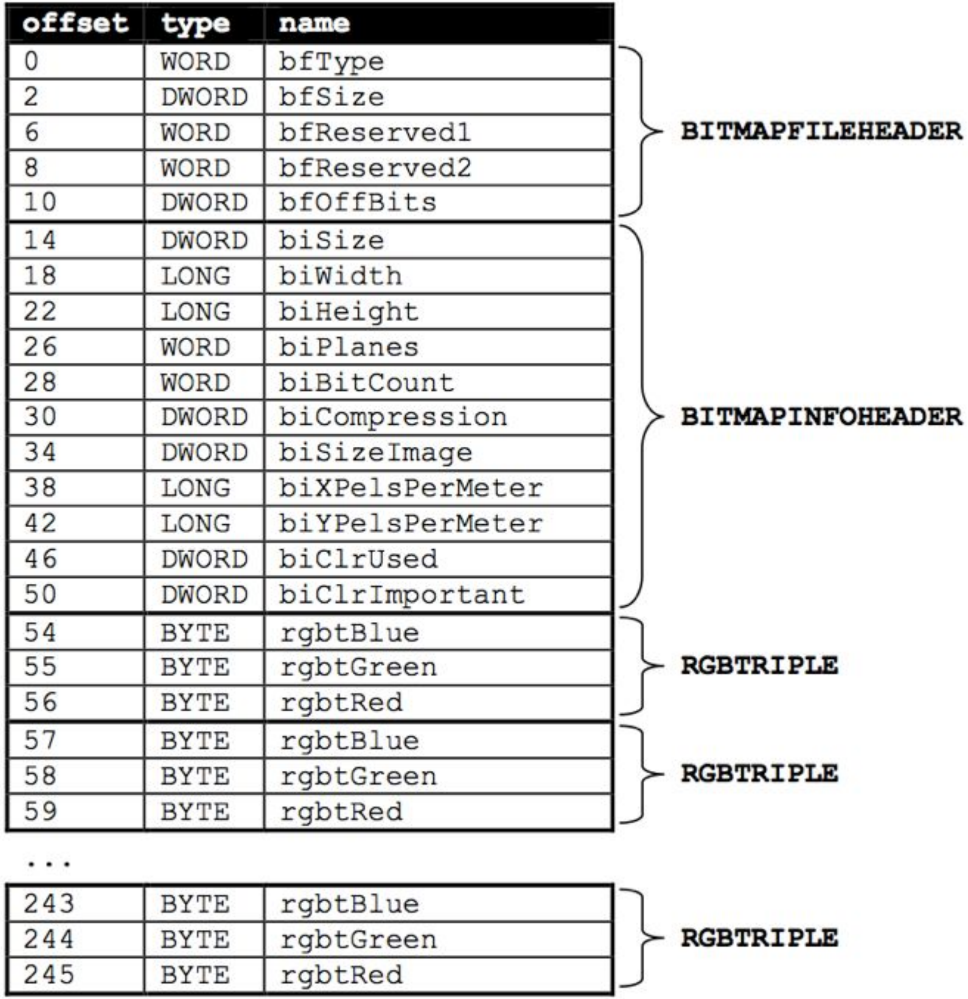
heap overflow

buffer overflow

struct



```
11000011
10111101
01011010
01111110
01011010
01100110
10111101
11000011
```



offset	type	name	
0	WORD	bfType	BITMAPFILEHEADER
2	DWORD	bfSize	
6	WORD	bfReserved1	
8	WORD	bfReserved2	
10	DWORD	bfOffBits	
14	DWORD	biSize	BITMAPINFOHEADER
18	LONG	biWidth	
22	LONG	biHeight	
26	WORD	biPlanes	
28	WORD	biBitCount	
30	DWORD	biCompression	
34	DWORD	biSizeImage	
38	LONG	biXPelsPerMeter	
42	LONG	biYPelsPerMeter	
46	DWORD	biClrUsed	
50	DWORD	biClrImportant	
54	BYTE	rgbtBlue	RGBTRIPLE
55	BYTE	rgbtGreen	
56	BYTE	rgbtRed	
57	BYTE	rgbtBlue	RGBTRIPLE
58	BYTE	rgbtGreen	
59	BYTE	rgbtRed	

...

243	BYTE	rgbtBlue	RGBTRIPLE
244	BYTE	rgbtGreen	
245	BYTE	rgbtRed	

This is CS50