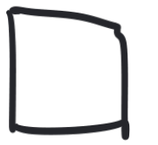
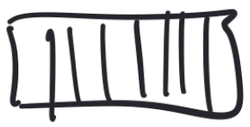
CS 208

29 March 2023

Wednesday



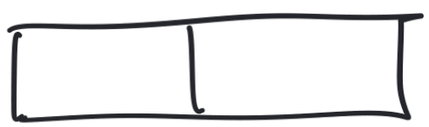
bit ("binary digit")



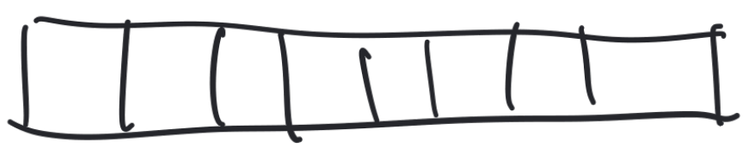
byte (8 bits)

Fred Brooks

2 possible contents: 0 or 1



2 bits: 2^2 possible bit patterns



8 bits : 1 byte

$2^8 = 256$ possible bit patterns

Ways of writing integers

Base ten
"decimal"

$$973 = 9\times10^2 + 7\times10^1 + 3\times10^0$$

Base two
"Binary"

$$10110_{two} = 1\times2^4 + 0\times2^3 + 1\times2^2 + 1\times2^1 + 0\times2^0 = 22_{ten}$$

32 16 8 4 2 1

$$110101_{two} = ?_{ten}$$

$$= 1\times32 + 1\times16 + 1\times4 + 1\times1 = 53_{ten}$$

Base sixteen
"hexadecimal"

Symbols 0 1 2 3 4 5 6 7 8 9 A B C D E F
10 11 12 13 14 15

$256 = 16^2$ 16 1

$AB7_{hexadecimal} = ?_{ten}$

$A \times 256 + B \times 16 + 7$

$= 10 \times 256 + 11 \times 16 + 7 = 2560 + 176 + 7$

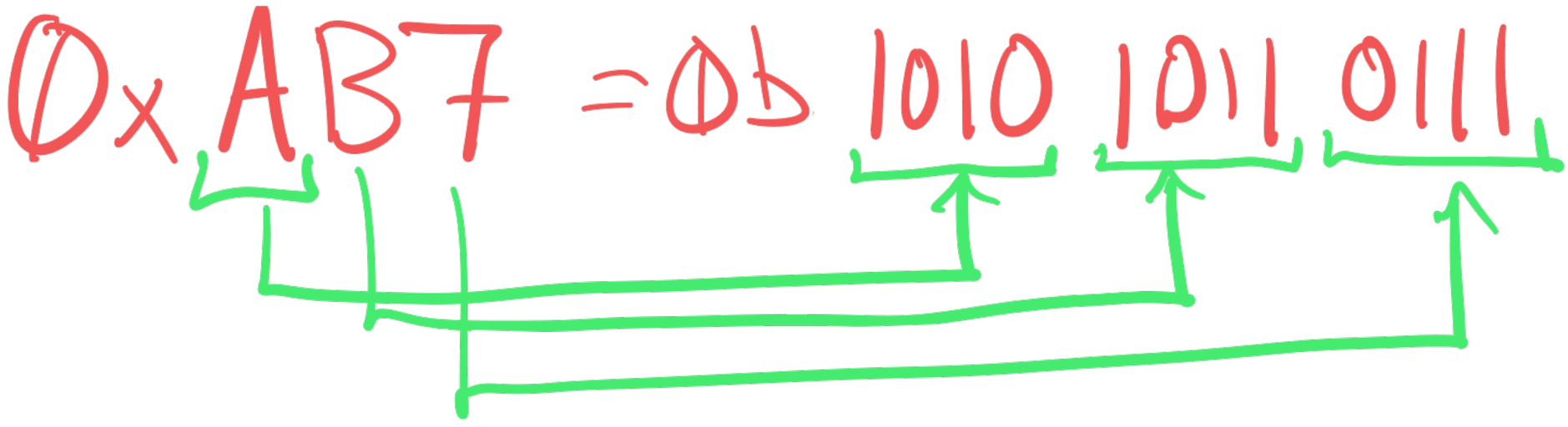
$= 2743_{ten}$

Why bother w/ hexadecimal?

Binary ← we care

Decimal ⟷ Binary hard

Hexadecimal ⟷ Binary easy



know this.

Octal - 01234567

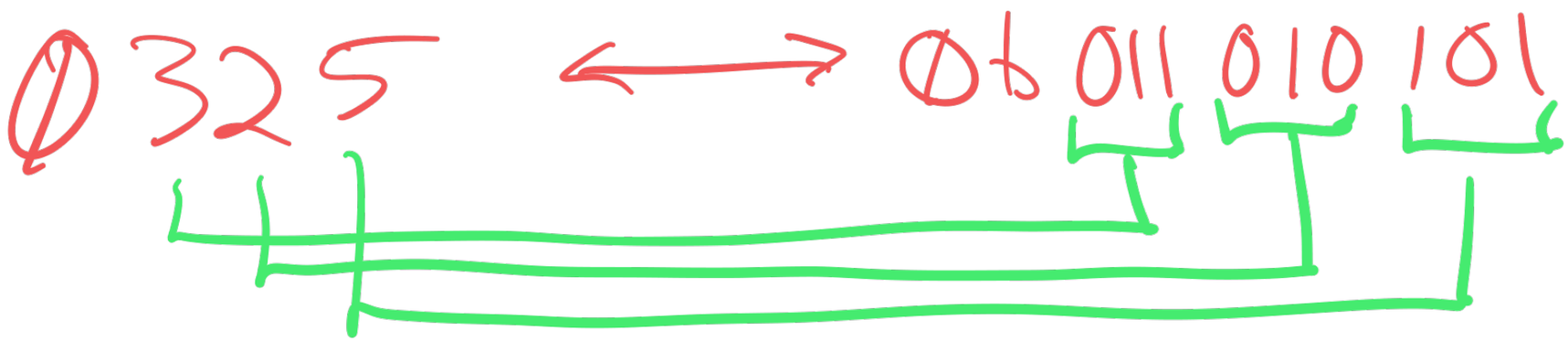
0325 ← octal

8^2 8^1 8^0

64 8 1

$= 3\times64 + 2\times8 + 5$

$= 192 + 16 + 5 = 213_{ten}$

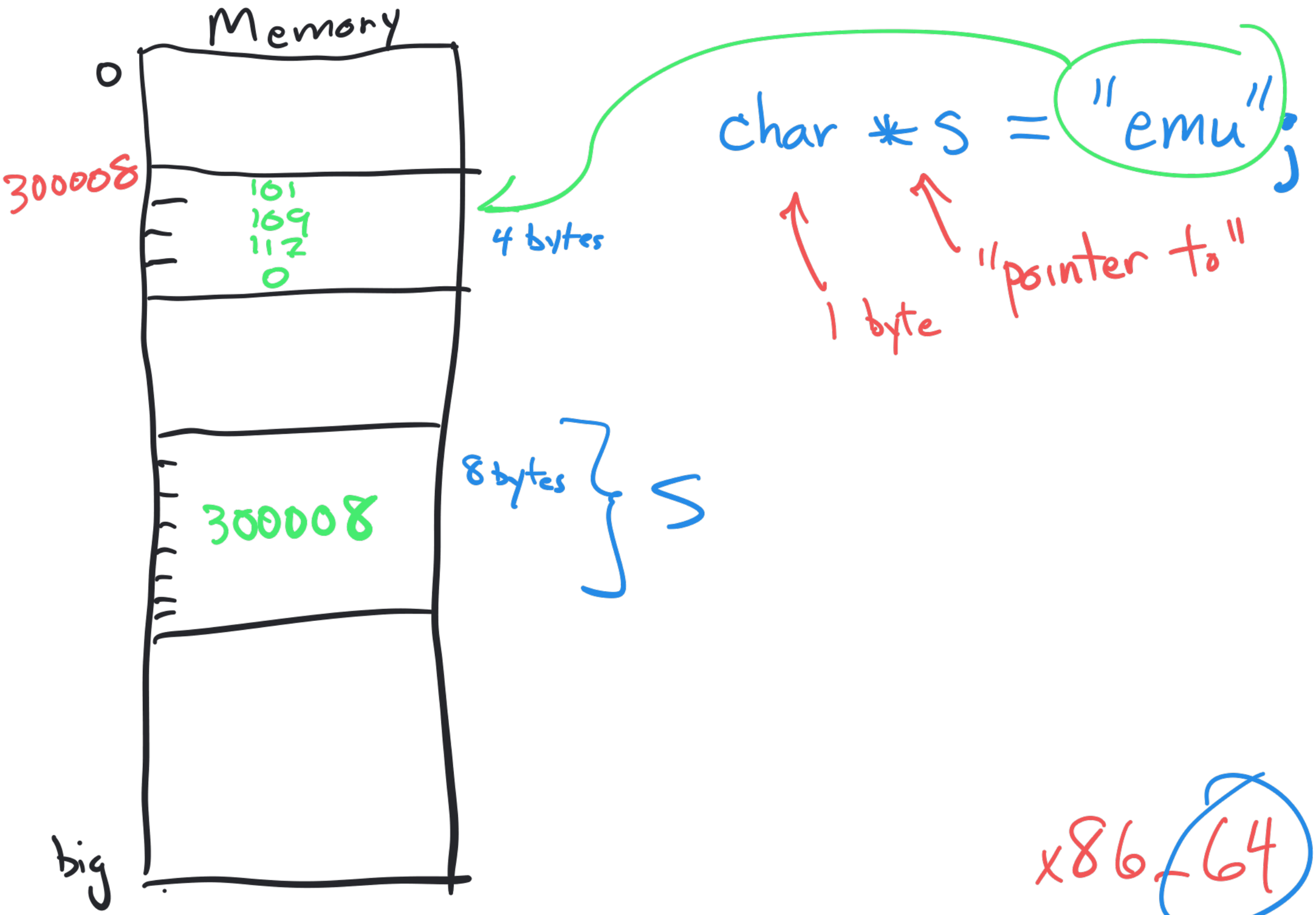


Remember:

hexdump -C filename

man ascii

ASCII — conventional mapping character-to-integer


Memory
0
300008
101
109
112
0
4 bytes
300008
8 bytes
s
big
char *s = "emu";
1 byte
"pointer to"

x86-64

(later: (const))

```
char s[10];
```

① says "s is a char*"

② sets aside 10 bytes to hold the chars

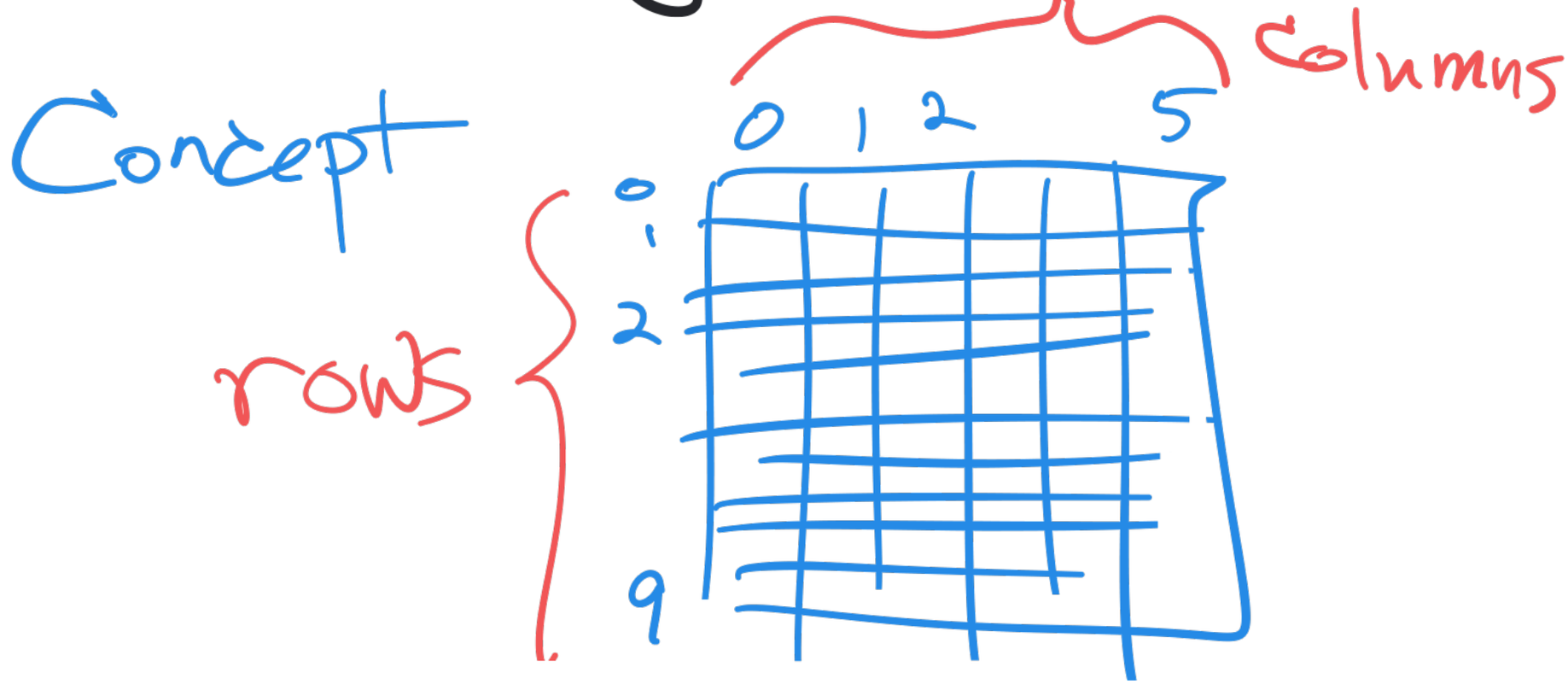
```
char *t;
```

① says "t is a char*"

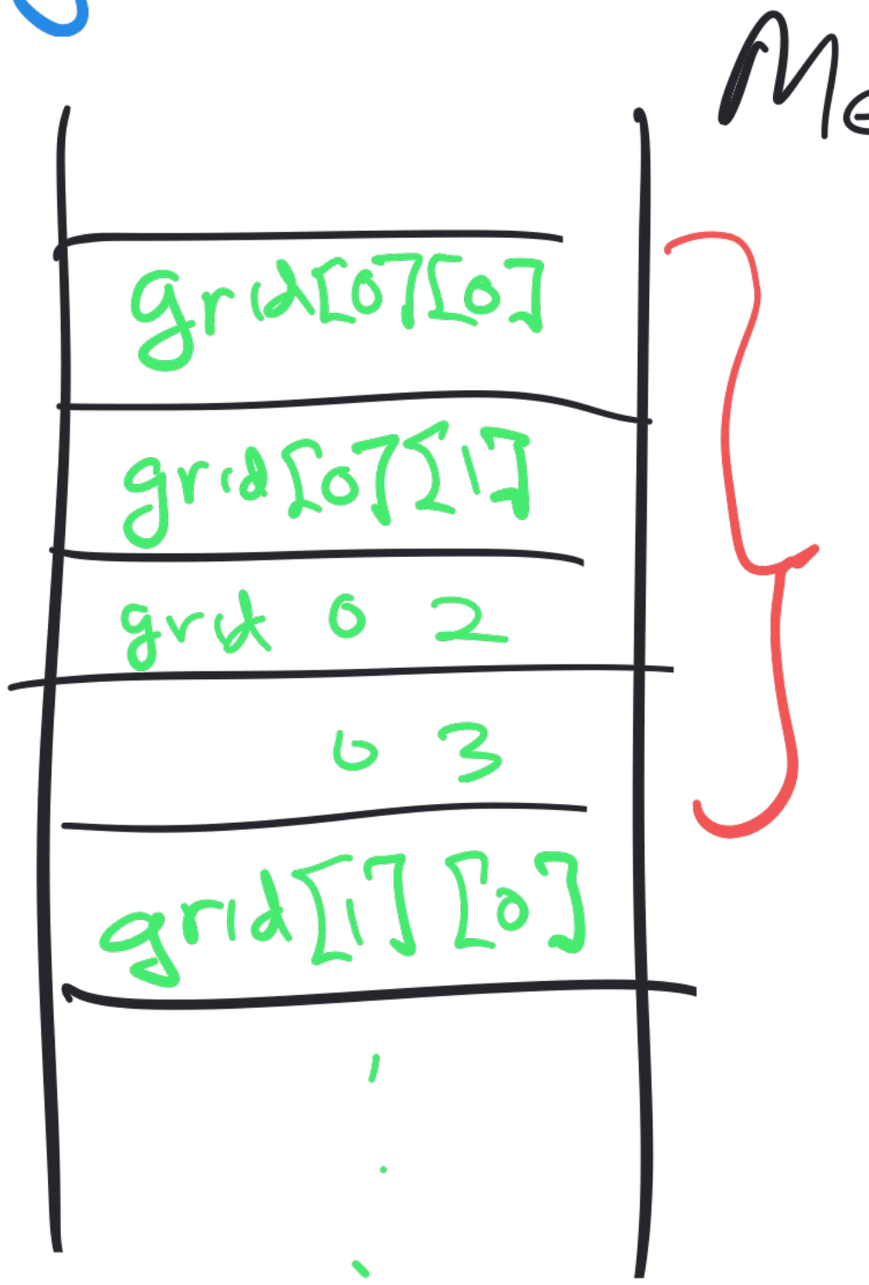
② sets aside 8 bytes to hold t.

2D arrays in C

```
char grid[10][6];
```



char grid[3][4]



"row-major order" (for storing a 2D matrix)

grid — pointer to the first char

grid[2] — pointer to the third row

grid[2][1] — second char in third row.