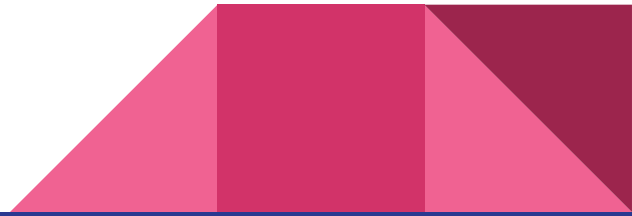
Computer System Organization [Spring 18, Mu]

R02: C variables, pointers, arrays, and bit operations

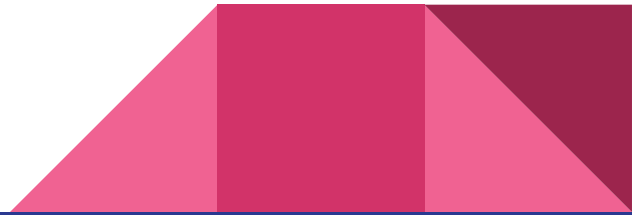
Logistics

- Stay current with Piazza
  - Ask questions there
  - Answer fellow classmates questions if you can
  - You are responsible for, at least, knowledge from instructor posts
- Lab 1 has been posted -- you would benefit to start right away
  - We will not go over part 1 in recitation
  - Next week I will speak on part 2, but you should have started by then!
- You have 5 grace days total for labs
  - used in half day increments



Using a graceday

- Inside of the lab directory
  - e.g. (this will not be the exact command): **cd cso-spring18-labs-USERNAME/clab**
  - **echo “0.5” > gracedays.md**
  - **git add gracedays.md**
  - **git commit gracedays.md -m “Using extension”**
  - **git push**
- We will automatically detect that a graceday was used as long as it is *pushed to github!!!*
- No grace days for recitations. Recitations due Wednesday 11:59pm



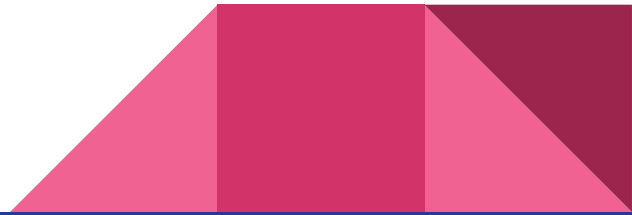
Variables in C

- Three important pieces of information associated with variable
  - Memory address of variable
  - Size of variable in bytes
  - Interpretation of variable

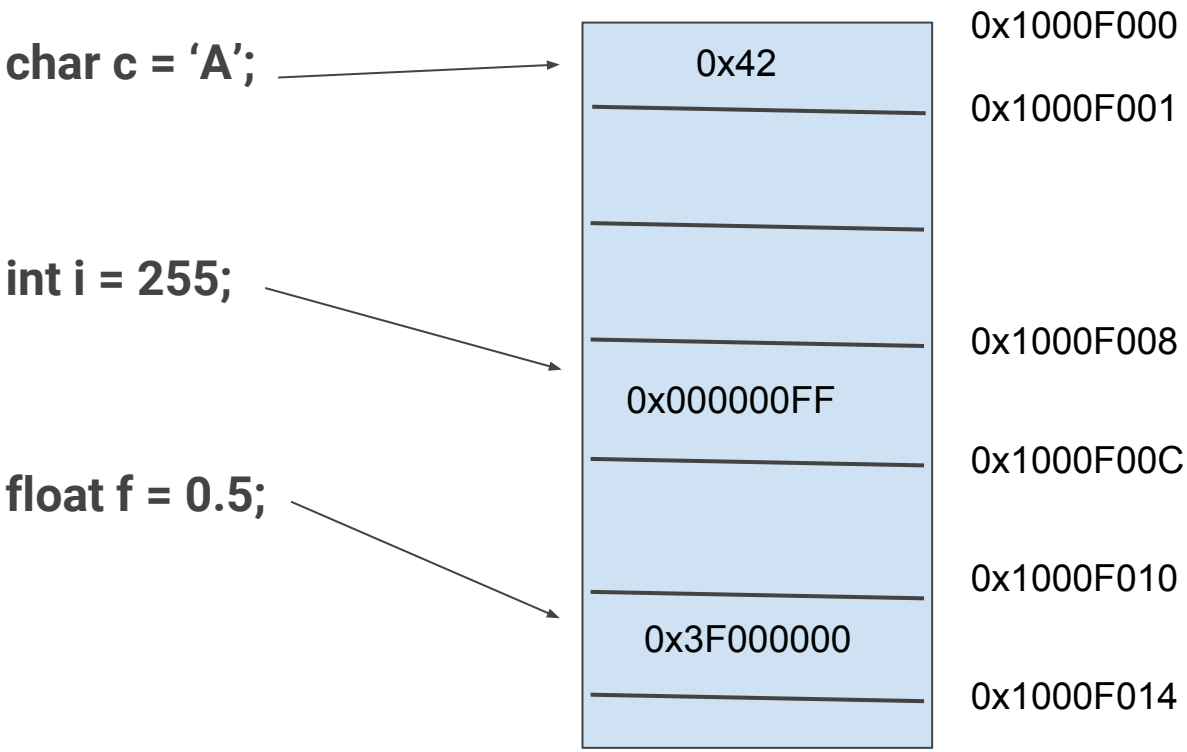
int a = 4; // Declaration and initialization. Interpret memory as *integer*

int* a_address = &a; // Address of a

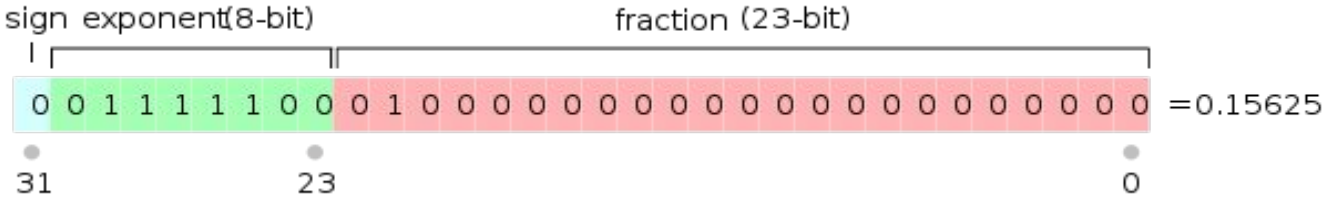
int a_size = sizeof(a); // Size of variable



Variables in C



Floating point representation



Source

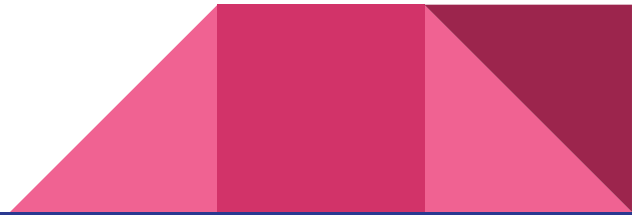
Calculation of number from bits as follows:

```
floating_point = -1^sign * 2^(exponent - 127) * (1.[mantissa])

0.5 = 2^-1 = -1^0 * 2^(126 - 127) * 1.[000…]

0 01111110 00000000000000000000000
```

Pointers in C

- Pointers are just variables too!
  - Pointers have memory addresses too
  - Pointers have a size (the amount of bytes to address into memory)
  - Pointers' value (interpretation) is a memory address

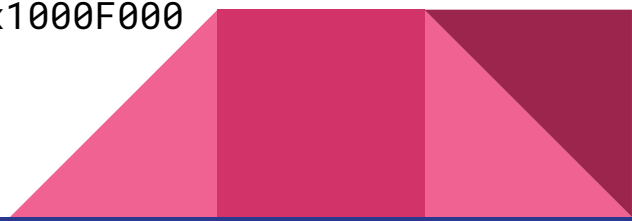
```
int a = 42;
int* a_ptr = &a; // Possible memory location: 0x1000F008
printf("%p", a_ptr); // 0x1000F008
*a_ptr = 43;
printf("%d", *a_ptr); // 43
printf("%d", a); // What is this?
```



Arrays in C

- Arrays are variables! However, the size is different
  - They have memory address
  - Bytes used by the variable => depends on the # elements and size of the elements
  - Interpretation: its the address of the start of the array

```
int arr[] = {1, 1, 2, 3, 5, 8, 13};
printf("%d", arr[4]); // 5
int* ptr = arr;
printf("%d", *(ptr + 4));/ // 5

printf("%p -- %p", arr, ptr); // e.g. 0x1000F000 -- 0x1000F000
```



Bit operations

```
int a = 13; // ...      0000 0000 1011

int b = 1023; // ...    0011 1111 1111

int c = a ^ b; // XOR: 0011 1111 0100

printf("%d", c); // (decimal) 1010


int d = a << 2; //      0000 0010 1100

int e = a >> 3; //      0000 0000 0001
```

Symbol	Description
&	bitwise AND
|	‘’ OR
^	‘’ XOR
<<	left shift
>>	right shift
~	bitwise NOT