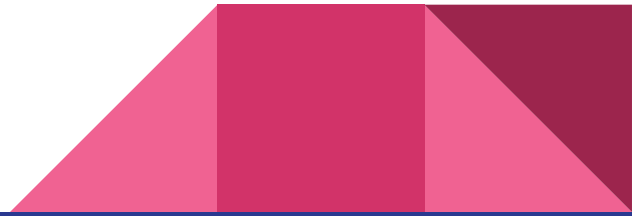
Computer System Organization [Spring 18, Mu]

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

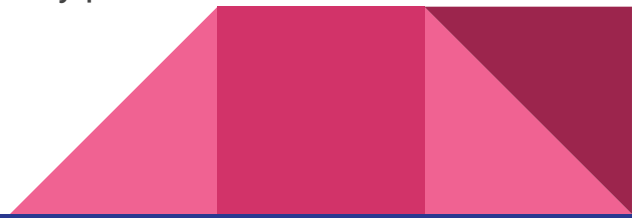
Logistics

- Piazza
  - Check for your answer in other Piazza questions first
    - Check google too!
  - No posting assignment code!
    - Ask general questions (and if you have, post general code)
    - Learning to ask code question in english will make debugging a lot easier
      - Way more relevant Google search results
- We will speak briefly about Lab1
- Go over code examples
- Go through a few coding exercises



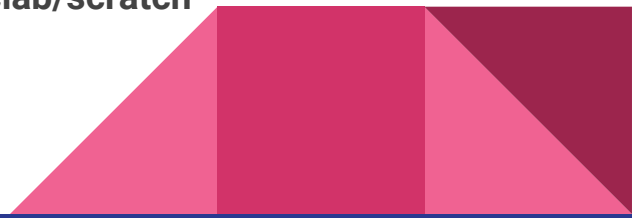
Lab1

- The first part mini:
  - Problems were relatively simple
    - No need to define main
    - Goog template to start from
  - Getting used to C was the tricky part
  - Be comfortable with pointers
- Second part: Scratch
  - You will implement three programs start-to-end
  - Must be main() function somewhere in code
  - You **MUST** create a Makefile so that typing “make” in the directory produces an executable **with the specified name**
  - Learn, and use, file I/O in C



Lab1

- You will need some way to sort in Lab1/scratch word count problem
  - There is qsort in the stdlib
- Resources:
  - “The C Programming Language” -- relatively short (at least compared to C++)
  - The **man** command on the command line
  - Google
- Gracedays for part 2
  - For Lab1 part1 (mini) the gracedays.md file could be either in **.../clab** or **.../clab/mini**
    - Both are OK as I did not specify the location.
  - For lab1 part2 (scratch) the gracedays.md file **must** be in **.../clab/scratch**



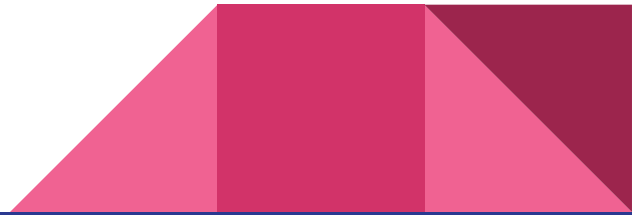
Variables in C

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

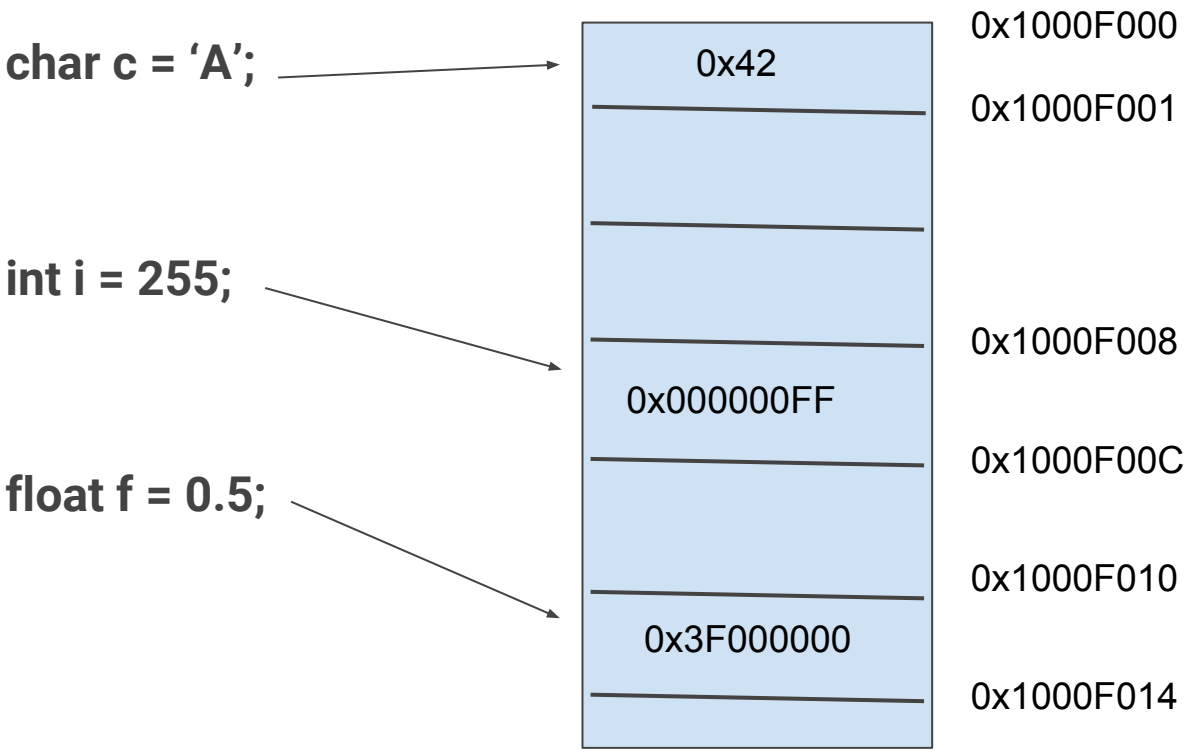
int a = 4; // Declaration and initialization. Interpret memory as *integer*

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

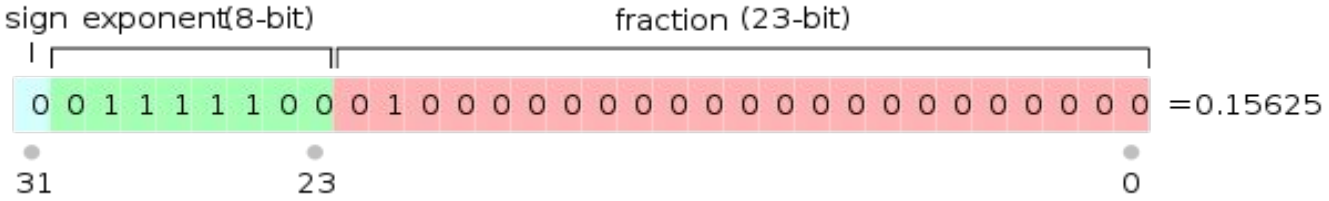
int a_size = sizeof(a); // Size of variable



Variables in C



Floating point representation



Source

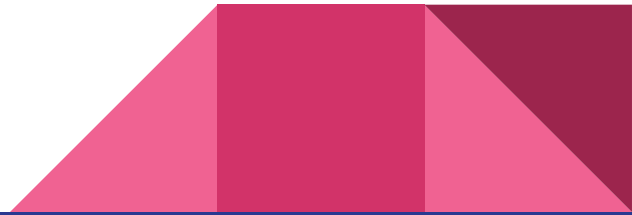
Calculation of number from bits as follows:

```
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

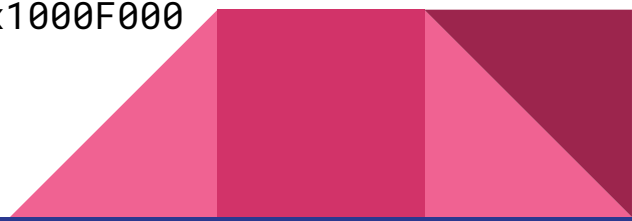
```
int a = 42;
int* a_ptr = &a; // Possible memory location: 0x1000F008
printf("%p", a_ptr); // 0x1000F008
*a_ptr = 43;
printf("%d", *a_ptr); // 43
printf("%d", a); // What is this?
```



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