Number Repr & Bitwise Op

- Unsigned, sign and magnitude, biased ($-k$), one's complement, two's complement
- Signed # convention = Two's complement; e.g. $0b100 = -4, 0b101 = -3, 0b010 = 2$
- One's complement : first bit signifies pos (0) or neg (1). Reverse all if neg, to get the absolute value
- Bias default : $k = -(2^{n-1} - 1)$
- Bitwise $x \wedge 1 = \sim x$; $x \wedge 0 = x$; $x \wedge x = 0$

C

- char = 1 byte; int = 4 bytes (32 bits) usually
- malloc etc. does not return heap addresses in sequential order, but does for stack (top large to bot small) → From top to bottom (large to small addresses) : Stack, Heap, Static, Code
- Common errors
  - Memory leak : fail to free allocated space, e.g. changing a pointer
  - Segmentation fault : accessing or indexing into unassigned space
  - Double free
  - Incorrect use of free : cannot free space not being m/c/realloc-ed
  - Logic error
- Note : == return 0 for False (not equal), strcmp return 0 for True (equal)
- Statically defined strings (char*, not char str[]) canNOT be modified → char* static; char str[] stack
- Only 0 and NULL evals to False
- malloc(size) contains garbage, calloc(#, size of each) initializes all to 0, realloc(???)
- strcpy, strcmp

RISC-V

- 32 registers, each 32-bit wide
- Calling function : prologue (save callee-side registers), body, epilogue (restore callee-side regs)

Extra Sanity Checks

- Check : NULL & str of len 0
- In C, only 0 and NULL evals to False
- Pointer p, then $p + 1$ actually adds by type size, e.g. char* adds 1, int* adds 4, etc.