ECE 177 – Programming I: From C Foundations to Hardware Interaction Lecture 2

Vince Weaver

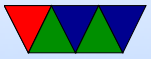
https://web.eece.maine.edu/~vweaver

vincent.weaver@maine.edu

23 January 2026

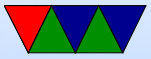
Announcements

- Labs Start Next Week!
  Remember to bring your laptop!
- What happens if bad weather cancels a lab?
  I'll send an e-mail if that happens
- Will start taking attendance next week!
- Added a few optional textbook recommendations on the website



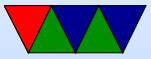
History of C

- Originally written in early 1970s by Dennis Ritchie (1941-2011) (Turing Award Winner)
- Deeply tied to the UNIX operating system, a highly influential operating system that's a predecessor to Linux



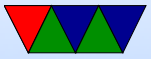
History of UNIX

- UNIX developed in early 1970s Bell Labs
- Originally written in assembly language
- C language invented while making UNIX portable
  - Various computers have own assembly language, generally not compatible
  - For example: modern days might have x86, ARM, RISC-V
  - Having a higher-level language allows single codebase to target multiple systems



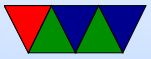
First machine ported to was PDP-11

- Made by DEC, major New England computer company that no longer exists
- Size of refrigerator, low-end model was $11,000 in 1970 which would be $80,000 now
- Show off my PiDP-11 6:10 scale replica



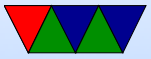
C progression

- Previous project Multics had a language CPL / BCPL
- Based on BCPL, Thompson made a subset B. It only supported integers as a type.
- Ritchie wrote C (follow up of new-B)
- Traditionally people joke that since origin B-C-P-L if the followup to C would be called P or D
  (there is a D, but most direct follow up C++)



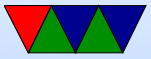
History of C – Some Links

- https://arstechnica.com/features/2020/12/a-damn-stupid-thing-to-do-the-origins-of-c/
- Video: Ken Thompson interviewed by Brian Kernighan https://www.youtube.com/watch?v=EY6q5dv_B-o
- Interesting thing about Computer history is it's recent enough many people involved are still alive



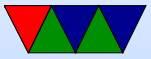
More History of C

- K&R C from book (C78)
  K&R (Brian Kernighan and Dennis Ritchie) weird function declarations, only `/* */` comments
- ANSI C (C89)
- ISOC (C90) international standard
- C99. Various features including C++ style comments
- C11 / C13 / C17
- C18
- C23



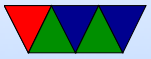
The purpose of C

- Low-level system programming language
- Best when dealing directly with hardware (Operating Systems, Embedded Systems)



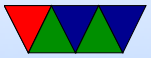
C Positives

- Relatively simple
- Close to hardware
- Generally generates fast code
  Especially compared to interpreted languages like Python, Java, Jacascript
- Low-resources, originally written on machines from 1970s
- Portable



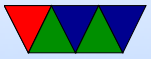
C Downsides

- Undefined Behavior (the spec is intentionally vague in places and this can lead to problems)
- Security (easier than it should be to crash)
- Strings (easy to get wrong)
- Pointers
- Memory Management (malloc/free)



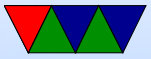
What Makes a Computer

- Processing part (CPU / ALU)
- Temporary Memory (RAM)
- Permanent Memory (DISK)
- Input / Output (I/O) for communicating with outside world



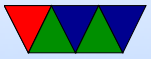
Programs / Executables

- After you compile them, live on disk (permanent storage: slower, cheaper)
- When you run them, get copied into RAM (temporary storage: faster, more-expensive)
- While running will generally use I/O to interact with users
  - Input: Keyboard, mouse, touchscreen
  - Output: text, graphics, sound



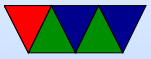
Bits

- Binary
- Just a 0 or 1
- Stored in either a flip-flop or capacitor
- Most computers cannot address individual bits (though some, Intel 432 could)



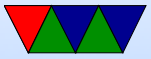
Aside on Memory – How Are Bits Stored?

- DRAM (dynamic RAM) – stored in capacitors (plus a transistor each bit). Leaks away and has to be refreshed every few milliseconds
- SRAM (static RAM) – stored in flip-flops, circuit made with 6 transistors
- Disk/Flash – way more complicated than RAM



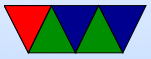
Aside – can you have a non-binary Computer?

- Yes, you could design logic with base3
- The difference/analytical engine was base10
- Binary is easier for a lot of reasons



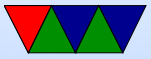
Bytes

- Most modern computers memory is chunked together in octets, or bytes
- There are 8 bits in a byte
- 0000.0000b to 1111.1111b (0..255)
- For given number of bits X, the number of values is 2^X, so for 8-bits it is 2^8 (256)



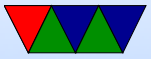
Nibbles/Nyblles

- Occasionally you might want to operate on half a byte
- For humor reasons this is often called a nibble



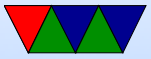
Addressing Memory

- Modern computers can have billions of bytes (4GB is 2^{32} bytes)
- Each byte has an address uniquely identifying it
- Traditionally in RAM this was a grid, but it has gotten complex
- Generally your OS, code, programs, variables, will all be living in RAM and each part will have an address for each byte

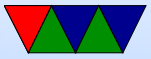


Asside on Memory Prefixes

- With RAM use SI prefixes, kilo, mega, giga, terra
- RAM is always a power of 2 though, so not accurate. 2^{10} bytes is called a kilobyte despite being 1024
- There was an attempt to have power-of-two prefixes, so a kiB "kibibyte" would be 1024 not 1000
- This mostly came up because hard disk manufacturers realized they could use the SI units meaning powers of 10 to make their disks look larger
- You don't always see the power-of-two prefixes used

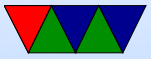


much



Do you ever want to have values bigger than 8-bits?

- Older computers were 8-bits but even they wanted more
- Modern computers are 64-bit (or 32-bit) but still memory is usually byte-addressable (the DEC Alpha wasn't and it was a huge pain so they had to add it back in)



Data Types

- We briefly started talking about C data types but the notes for that will be in the next lecture