May 28th:

https://en.wikibooks.org/wiki/The_Computer_Revolution

http://en.wikibooks.org/wiki/The_Computer_Revolution/Hardware/CPU
https://en.wikibooks.org/wiki/The_Computer_Revolution/Hardware/Memory
https://en.wikibooks.org/wiki/The_Computer_Revolution/Hardware/Virtual_Memory
https://en.wikibooks.org/wiki/The_Computer_Revolution/Hardware/Microprocessor

CPU (Central Processing Unit): "Brain" of the computer. Executes instructions, manages tasks. Older PCs relied on one CPU; modern systems use multi-core CPUs for multitasking and performance (e.g., Intel Core, AMD Ryzen). Delegates tasks to specialized chips (e.g., GPU). Connected via the motherboard, which also connects all subsystems.

Memory Types:

RAM (Random Access Memory): Temporary, volatile memory for active tasks. Installed as modules (DIMM, SO-DIMM). Measured in MB/GB. Overloading causes fallback to slower HDD/SSD memory.

Types: DRAM, SDRAM, SRAM, DDR-SDRAM

ROM (Read-Only Memory): Non-volatile, read-only chip memory for permanent firmware (BIOS/POST). Types include PROM. Modern systems use flash-based ROM, allowing updates.

CMOS (Complementary Metal-Oxide Semiconductor): Battery-powered chip storing BIOS config/time. Similar to PRAM on Macs.

Flash Memory: Non-volatile, rewritable storage (e.g., SD cards, USB drives).

Solid-state, widely used in phones, cameras, laptops.
Sizes: 1GB to 32GB+. Portable, durable, no moving parts.

Future RAM: Magnetic RAM (MRAM) being developed for nonvolatile memory, retaining data without power.

Microprocessor: Miniaturized CPU chip. Executes logic/instructions. Controls all digital systems (PCs, phones, clocks).
Key traits: instruction set, bandwidth, clock speed.
Converts input -> machine code -> output. Enables embedded systems. Rapid evolution (e.g., 7nm dies).
Brands: Intel Core, AMD Ryzen, Apple M-series, Snapdragon.
Found in everything from PCs to smart devices.

https://en.wikibooks.org/wiki/Microprocessor_Design
https://en.wikibooks.org/wiki/Microprocessor_Design/Memory
Importance of Memory
- Essential in microcontroller and processor design.
- Needed to understand overall system functionality.

Memory Hierarchy
- Trade-off: memory can be fast or large, not both.
- Hierarchy structure:
  - Fast, small memory is close to the CPU.
  - Large, slow memory is farther from the CPU.

Hard Disk Drives (HDDs) and SSDs
- Known as secondary memory or non-volatile memory.
- HDD:
  - Stores data magnetically.
  - Keeps data without power.
  - Very slow compared to RAM or cache.
  - Mechanical parts make it prone to wear/failure.
- SSD:
  - Some newer models use flash memory.

Flash Memory in Microcontrollers
- As of 2013, many single-chip CPUs use flash memory as program memory.
- CPUs can directly run code from flash.
- In Harvard architecture CPUs, flash is often the only memory type from which native code can be executed.

RAM (Random Access Memory)
- Volatile memory – loses data when power is off.

- Acts as the processor’s main working memory.
- Usually has limited capacity (few GBs).
- Two primary types:
  - SRAM
  - DRAM

SRAM (Static RAM)
- Uses 6 transistors per bit.
- Doesn't need refresh while powered.
- Faster, but larger and more expensive.
- Commonly used in CPU caches, not as main memory.

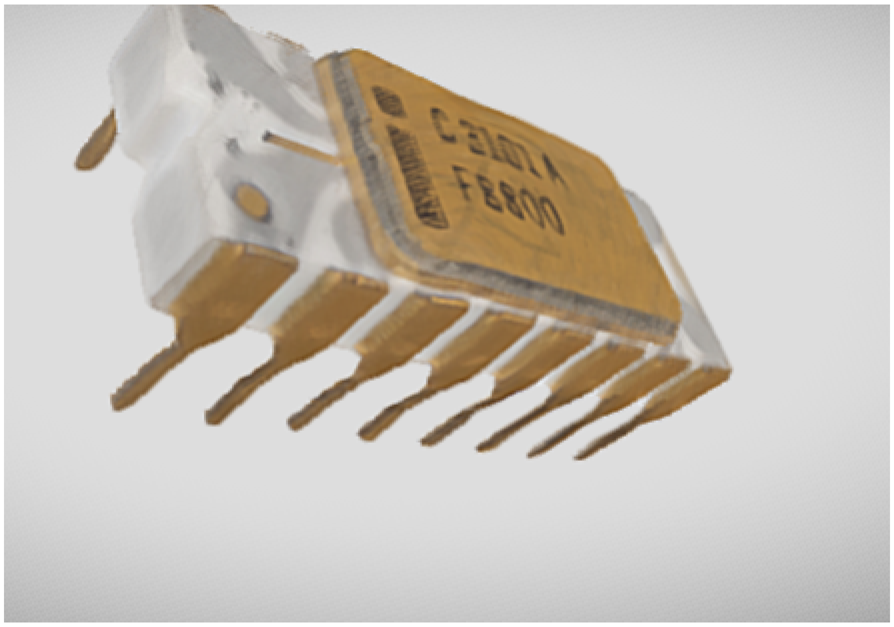
DRAM (Dynamic RAM)
- Uses 1 transistor + 1 capacitor per bit.
- Higher density, stores more data in less space.
- Needs constant refreshing, making it slower.
- Cheaper than SRAM.
- Basis for most main memory.
- Variant: SDRAM, a type of DRAM (not related to SRAM).

Note on Moore’s Law:
- CPUs keep getting faster, but DRAM speed is lagging.
- Leads to increasing inefficiency in communication between CPU and RAM.

Cache Memory
- Smaller and faster than main RAM.
- Sits closer to the CPU.
- Operates at CPU clock speed (much faster than system bus).
- Has levels:
  - L1 – smallest, fastest.
  - L2 – medium.
  - L3 – largest, slowest.

Registers
- Fastest and smallest memory type.
- Stored directly inside the CPU.
- Used for immediate data access during operations.
- Most CPUs have between 4 to 256 registers.

https://virtualmuseum.intel.com/



1972:

- Intel acquires Microma, a digital watchmaker with a prototype LCD watch.

- Digital Watches (early 1970s):

- Seen as high-tech gadgets.

- Sold for around $200.

1974:

- Intel expands its LCD watch circuits.

- New features include:

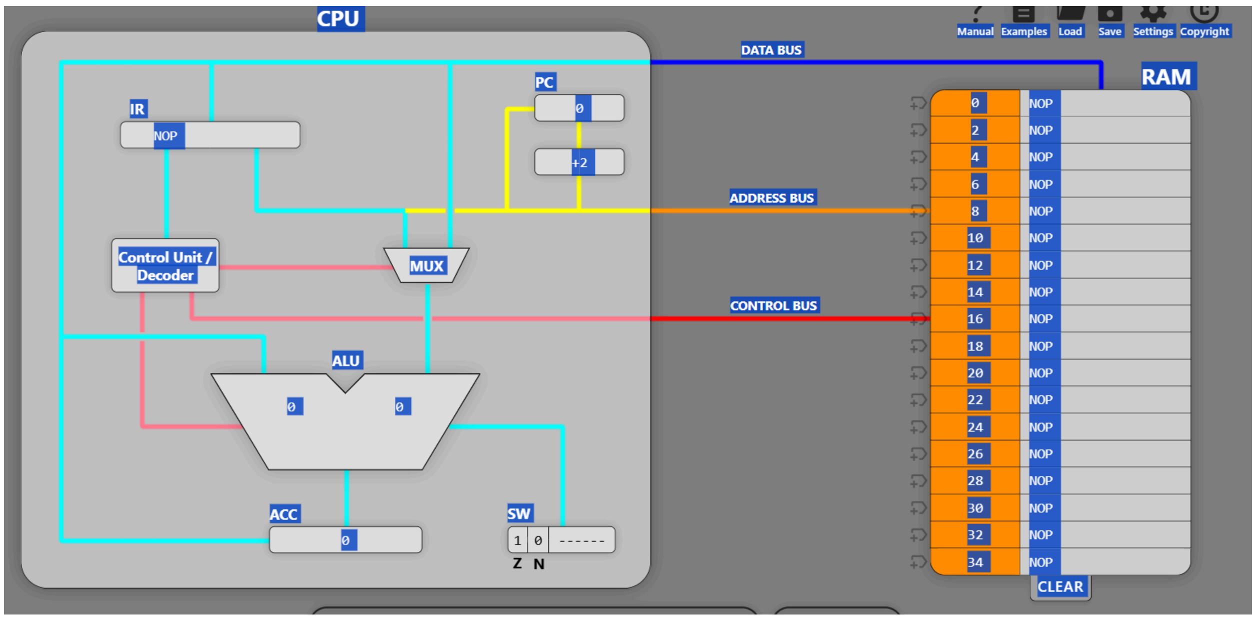
- Seconds display

- Date display

1978:

- Intel sells Microma.
  Reason: Could not add enough value to watch electronics.

https://cpuvisualsimulator.github.io/



The simulator lets you run and see how assembly code works.

You can add or change instructions and numbers in RAM.

You can make labels to use instead of memory addresses.

- Labels help with jumps and variables.

You can switch views between symbols and binary.

You can change the Program Counter, Accumulator, and flags (Negative, Zero).

Can run the program:

- All at once

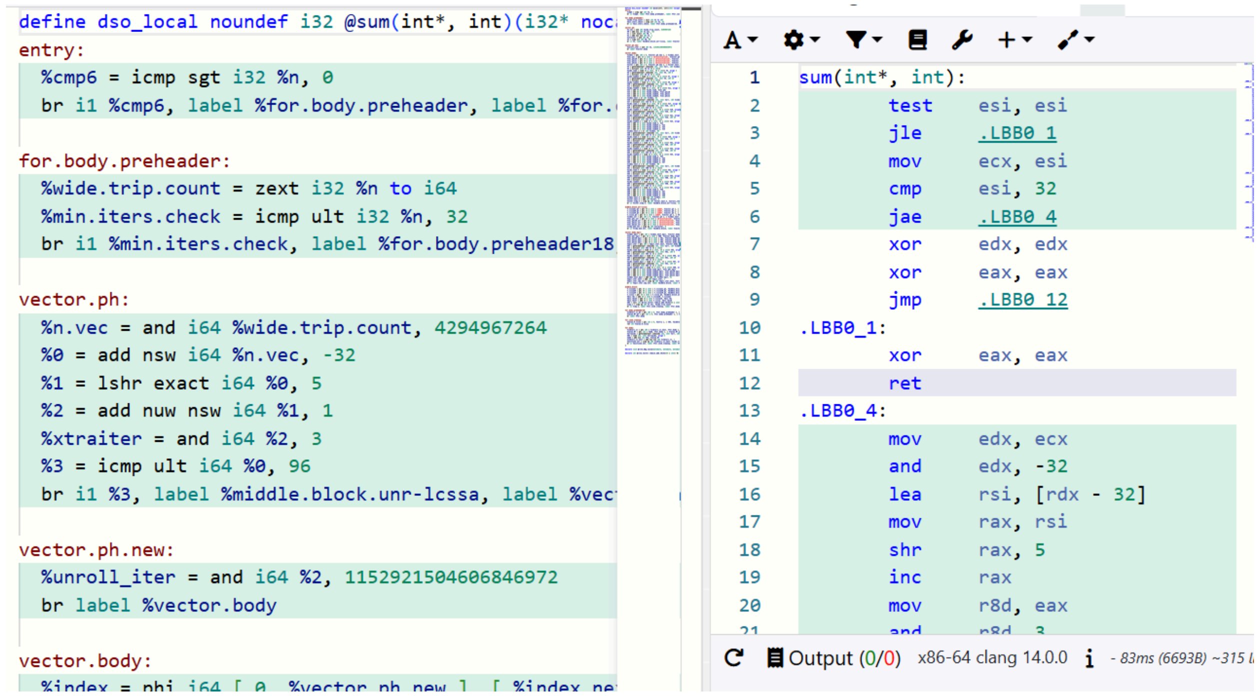
- One instruction at a time
- One step at a time

June 2nd:

https://godbolt.org/

This site is basically a compiler site where we see different examples of different programming languages like C++, python, Java, Html, Pascal, Ada, C, etc.

In her we can see different examples, and self made example templates that showcase what different programming languages look like in compilation.
For example: LLVM IR:

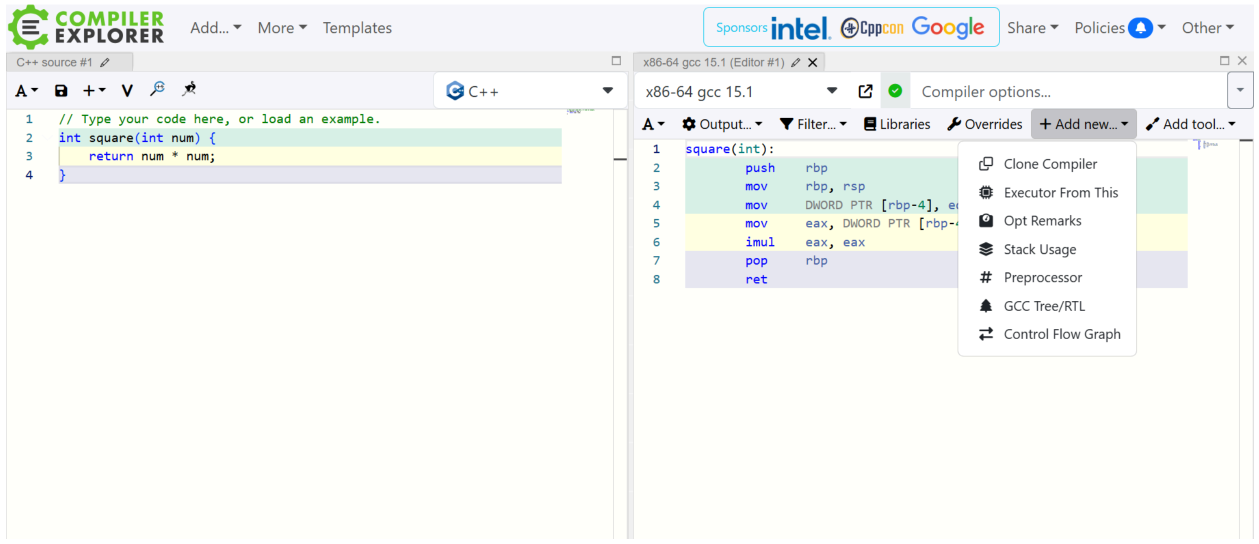


```
define dso_local noundef i32 @sum(int*, int)(i32* noc
entry:
  %cmp6 = icmp sgt i32 %n, 0
  br i1 %cmp6, label %for.body.preheader, label %for.

for.body.preheader:
  %wide.trip.count = zext i32 %n to i64
  %min.iters.check = icmp ult i32 %n, 32
  br i1 %min.iters.check, label %for.body.preheader18

vector.ph:
  %n.vec = and i64 %wide.trip.count, 4294967264
  %0 = add nsw i64 %n.vec, -32
  %1 = lshr exact i64 %0, 5
  %2 = add nuw nsw i64 %1, 1
  %xtraiter = and i64 %2, 3
  %3 = icmp ult i64 %0, 96
  br i1 %3, label %middle.block.unr-lcssa, label %vec

vector.ph.new:
  %unroll_iter = and i64 %2, 1152921504606846972
  br label %vector.body

vector.body:
```

```
sum(int*, int):
        test    esi, esi
        jle     .LBB0_1
        mov     ecx, esi
        cmp     esi, 32
        jae     .LBB0_4
        xor     edx, edx
        xor     eax, eax
        jmp     .LBB0_12
.LBB0_1:
        xor     eax, eax
        ret
.LBB0_4:
        mov     edx, ecx
        and     edx, -32
        lea     rsi, [rdx - 32]
        mov     rax, rsi
        shr     rax, 5
        inc     rax
        mov     r8d, eax
```

Output (0/0) x86-64 clang 14.0.0 - 83ms (6693B) ~315

There is not much else to do here other than that.
You can see more specific settings such as:



C++ — x86-64 gcc 15.1

```
// Type your code here, or load an example.
int square(int num) {
    return num * num;
}
```

```
square(int):
        push    rbp
        mov     rbp, rsp
        mov     DWORD PTR [rbp-4], e
        mov     eax, DWORD PTR [rbp-
        imul    eax, eax
        pop     rbp
        ret
```

__Aside from that, flash memory is: a type of non-volatile memory, meaning it retains data even when the power is off.__