ENGSCI 233 Lecture 10.1

Computer Architecture

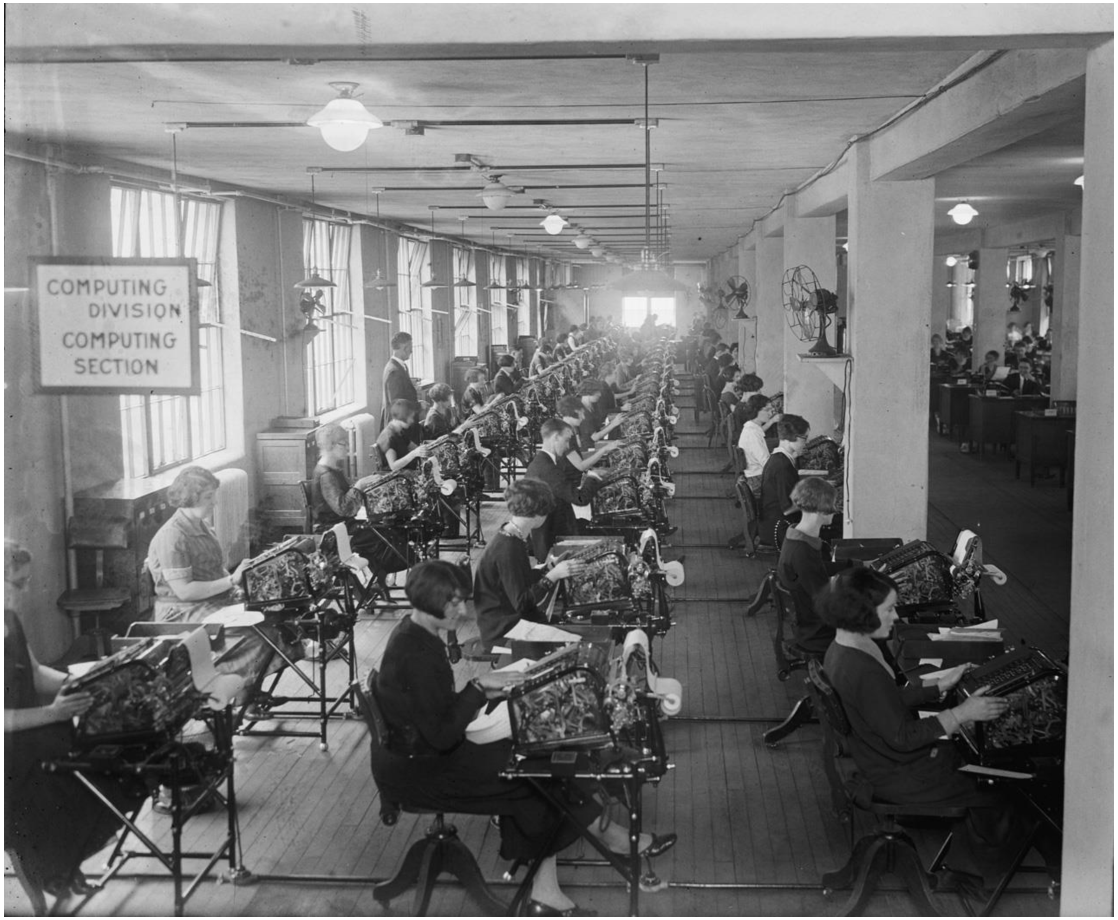
AN x64 PROCESSOR IS SCREAMING ALONG AT BILLIONS OF CYCLES PER SECOND TO RUN THE XNU KERNEL, WHICH IS FRANTICALLY WORKING THROUGH ALL THE POSIX-SPECIFIED ABSTRACTION TO CREATE THE DARWIN SYSTEM UNDERLYING OS X, WHICH IN TURN IS STRAINING ITSELF TO RUN FIREFOX AND ITS GECKO RENDERER, WHICH CREATES A FLASH OBJECT WHICH RENDERS DOZENS OF VIDEO FRAMES EVERY SECOND
BECAUSE I WANTED TO SEE A CAT JUMP INTO A BOX AND FALL OVER.
I AM A GOD.

Today’s learning objectives

- Understand and demonstrate breaking a process into computable steps
- Understand the relationship between programming languages, assembly code, and machine code
- Understand the parts of a computer
- Understand processor performance characteristics
- Understand multi-processor systems

What is a computer anyway?

What is a computer anyway?



Let’s run a demonstration.

- 1 volunteer for “memory”
- 1 volunteer for “processor”
- 1 volunteer for “data bus”

- Add two large numbers:

12345 + 67890

Demonstration rules:

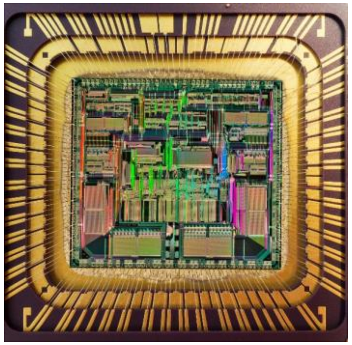
- One digit per post-it note
- Operations happen in-place if destination is not specified
- The processor can only store 4 digits
    - Everything else goes in main memory
- The processor must tell the data bus what to get from memory

Let’s run a demonstration.

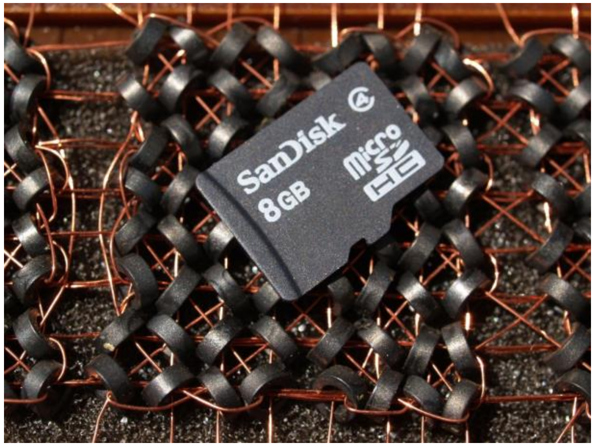
1. Start with an empty output, zero carry *c*, and a digit counter *k* = 0. (Count from the right.)
2. Retrieve digit *k* from each of the numbers to be added.
3. Add the digits *k* together, as well as the carry *c*.
4. Store the ones digit of the result as digit *k* of the output.
5. Replace the value in *c* with the tens digit of the result.
6. If *k* <= 4, increment *k* and go to step 2.
7. Display the output.

Computers have four parts.

- Processor



- Memory

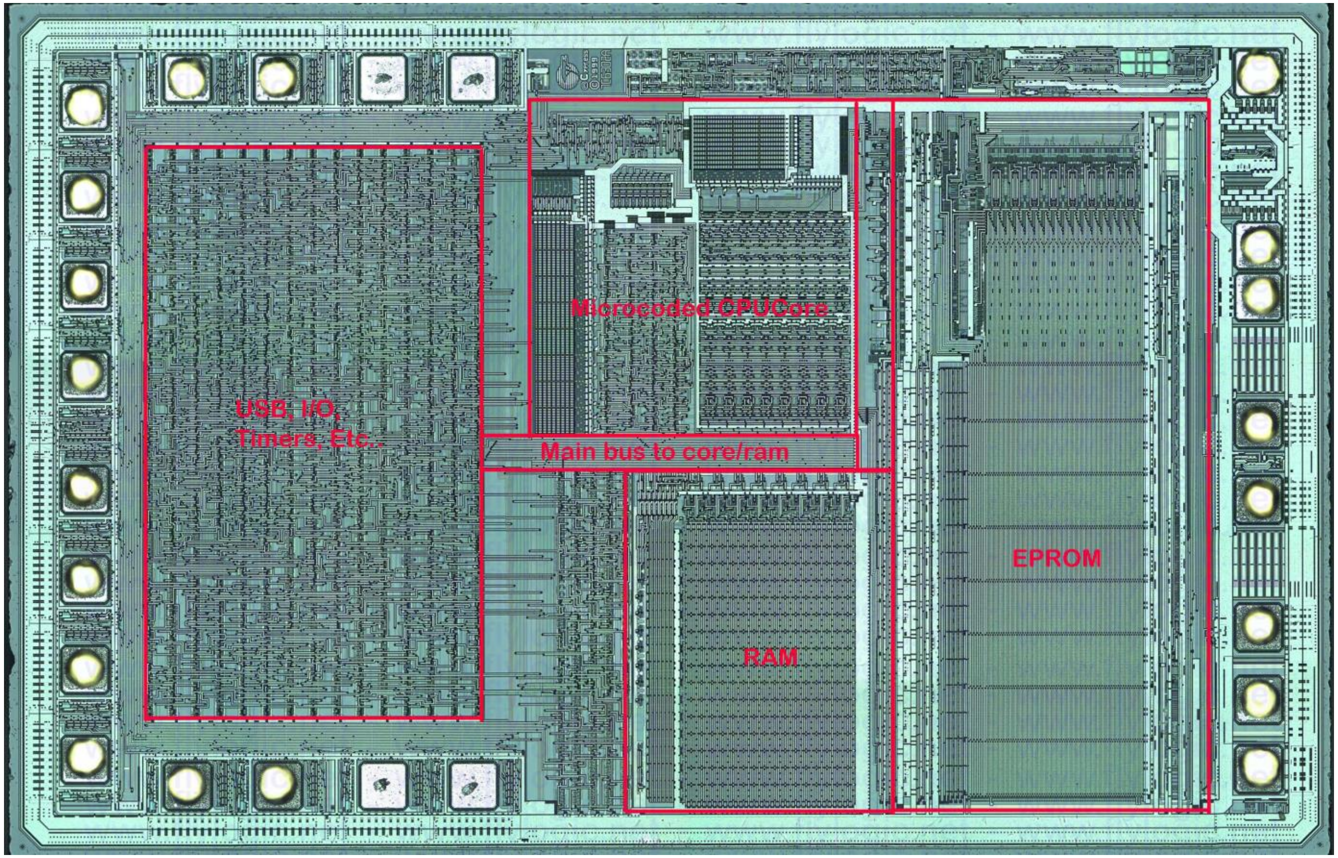


- Data transfer

- Input and output

Computers have four parts.

- These parts could all be on the same silicon chip!



Computer systems add software.

- Operating System
  - Sets standards for access to storage, transport, and I/O
  - Manages multiple programs running together

- Drivers
  - Provide interface between hardware and software.

- We will discuss this more in week 12.

How do programs run on computers?

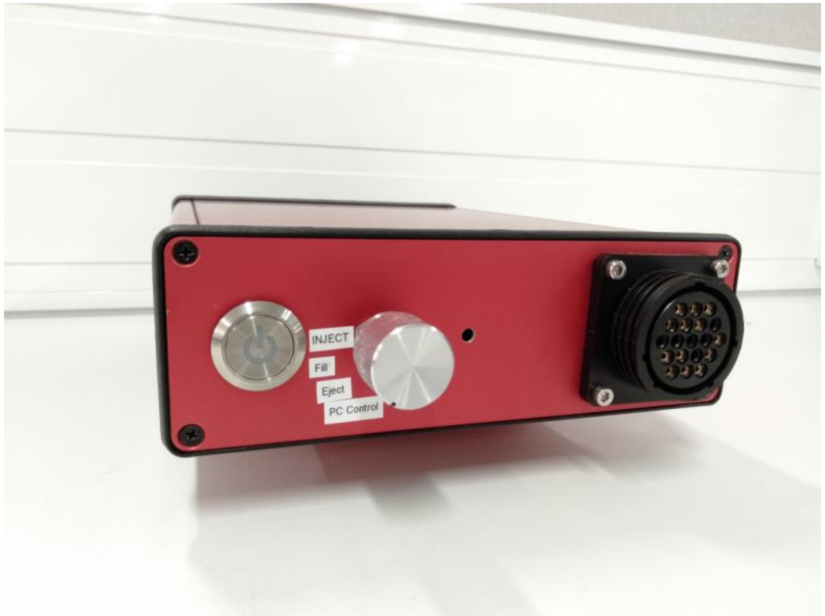
- Interpreter
  - Software that follows instructions written in code

python™

- Compiler
  - Converts code to machine language to run on its own

THE
C
PROGRAMMING
LANGUAGE

An example computer system:



What does code really do?

```
void Commutate(uint16_t *A, uint16_t *B, uint16_t *C, ...
               int16_t Command, uint16_t Position)
{
       if(Command > PWM_IDLE)
       {
              Command = PWM_IDLE;
       }
       else if(Command < -PWM_IDLE)
       {
              Command = -PWM_IDLE;
       }
       *A = 0;
       *B = PWM_IDLE + Command;
       *C = PWM_IDLE - Command;
}
```

- C code
- PWM_IDLE is a constant
- Runs on an ARM microcontroller
- How does this compile?

Assembly language is messy.

C Code

```
        *B = PWM_IDLE + Command;
08005050:    887b         ldrh   r3, [r7, #2]
08005052:    f603 0334    addw   r3, r3, #2100 ; 0x834
08005056:    b29a         uxth   r2, r3
08005058:    68bb         ldr    r3, [r7, #8]
0800505a:    801a         strh   r2, [r3, #0]
        *C = PWM_IDLE - Command;
0800505c:    887b         ldrh   r3, [r7, #2]
0800505e:    f5c3 6303    rsb    r3, r3, #2096 ; 0x830
08005062:    3304         adds   r3, #4
08005064:    b29a         uxth   r2, r3
08005066:    687b         ldr    r3, [r7, #4]
08005068:    801a         strh   r2, [r3, #0]
```

Assembly language is messy.

Assembly Code

```
      *B = PWM_IDLE + Command;
08005050:    887b         ldrh   r3, [r7, #2]
08005052:    f603 0334    addw   r3, r3, #2100 ; 0x834
08005056:    b29a         uxth   r2, r3
08005058:    68bb         ldr    r3, [r7, #8]
0800505a:    801a         strh   r2, [r3, #0]
      *C = PWM_IDLE - Command;
0800505c:    887b         ldrh   r3, [r7, #2]
0800505e:    f5c3 6303    rsb    r3, r3, #2096 ; 0x830
08005062:    3304         adds   r3, #4
08005064:    b29a         uxth   r2, r3
08005066:    687b         ldr    r3, [r7, #4]
08005068:    801a         strh   r2, [r3, #0]
```

Assembly language is messy.

Machine Code

```
      *B = PWM_IDLE + Command;
 08005050:    887b           ldrh   r3, [r7, #2]
 08005052:    f603 0334      addw   r3, r3, #2100 ; 0x834
 08005056:    b29a           uxth   r2, r3
 08005058:    68bb           ldr    r3, [r7, #8]
 0800505a:    801a           strh   r2, [r3, #0]
      *C = PWM IDLE - Command;
 0800505c:    887b           ldrh   r3, [r7, #2]
 0800505e:    f5c3 6303      rsb    r3, r3, #2096 ; 0x830
 08005062:    3304           adds   r3, #4
 08005064:    b29a           uxth   r2, r3
 08005066:    687b           ldr    r3, [r7, #4]
 08005068:    801a           strh   r2, [r3, #0]
```

Assembly language is messy.

```
      *B = PWM_IDLE + Command;
08005050:   887b        ldrh   r3, [r7, #2]
08005052:   f603 0334   addw   r3, r3, #2100 ; 0x834
08005056:   b29a        uxth   r2, r3
08005058:   68bb        ldr    r3, [r7, #8]
0800505a:   801a        strh   r2, [r3, #0]
      *C = PWM_IDLE - Command;
0800505c:   887b        ldrh   r3, [r7, #2]
0800505e:   f5c3 6303   rsb    r3, r3, #2096 ; 0x830
08005062:   3304        adds   r3, #4
08005064:   b29a        uxth   r2, r3
08005066:   687b        ldr    r3, [r7, #4]
08005068:   801a        strh   r2, [r3, #0]
```

Assembly language is messy.

```
     *B = PWM_IDLE + Command;
08005050:    887b         ldrh    r3, [r7, #2]
08005052:    f603 0334    addw    r3, r3, #2100 ; 0x834
08005056:    b29a         uxth    r2, r3
08005058:    68bb         ldr     r3, [r7, #8]
0800505a:    801a         strh    r2, [r3, #0]
     *C = PWM_IDLE - Command;
0800505c:    887b         ldrh    r3, [r7, #2]
0800505e:    f5c3 6303    rsb     r3, r3, #2096 ; 0x830
08005062:    3304         adds    r3, #4
08005064:    b29a         uxth    r2, r3
08005066:    687b         ldr     r3, [r7, #4]
08005068:    801a         strh    r2, [r3, #0]
```

Assembly language is messy.

```
      *B = PWM_IDLE + Command;
08005050:    887b          ldrh   r3, [r7, #2]
08005052:    f603 0334     addw   r3, r3, #2100 ; 0x834
08005056:    b29a          uxth   r2, r3
08005058:    68bb          ldr    r3, [r7, #8]
0800505a:    801a          strh   r2, [r3, #0]
      *C = PWM_IDLE - Command;
0800505c:    887b          ldrh   r3, [r7, #2]
0800505e:    f5c3 6303     rsb    r3, r3, #2096 ; 0x830
08005062:    3304          adds   r3, #4
08005064:    b29a          uxth   r2, r3
08005066:    687b          ldr    r3, [r7, #4]
08005068:    801a          strh   r2, [r3, #0]
```

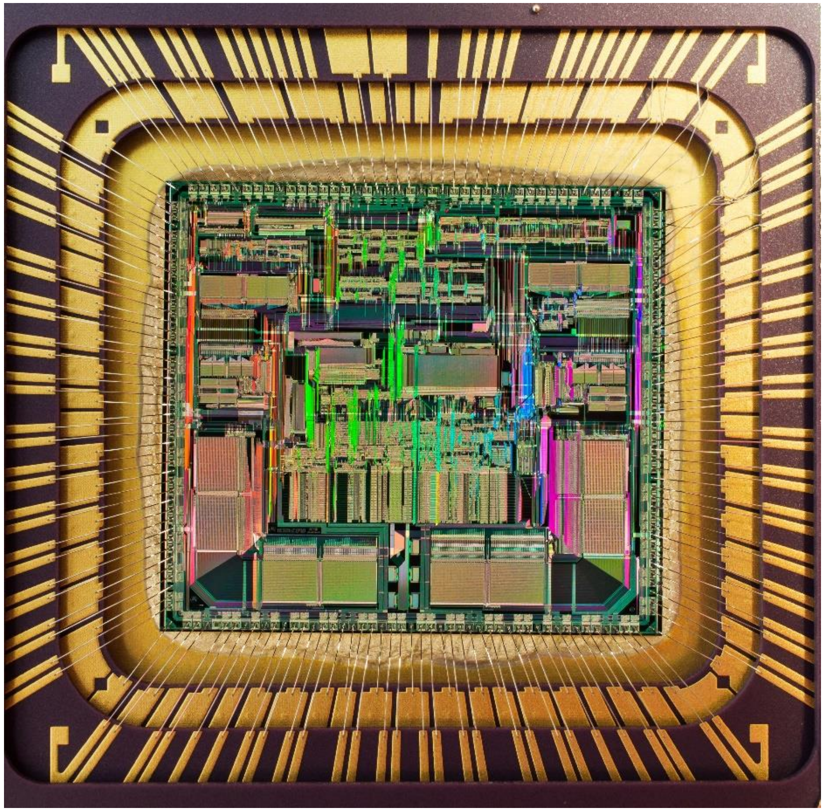
Assembly language is messy.

```
      *B = PWM_IDLE + Command;
08005050:   887b          ldrh   r3, [r7, #2]
08005052:   f603 0334     addw   r3, r3, #2100; 0x834
08005056:   b29a          uxth   r2, r3
08005058:   68bb          ldr    r3, [r7, #8]
0800505a:   801a          strh   r2, [r3, #0]
      *C = PWM_IDLE - Command;
0800505c:   887b          ldrh   r3, [r7, #2]
0800505e:   f5c3 6303     rsb    r3, r3, #2096; 0x830
08005062:   3304          adds   r3, #4
08005064:   b29a          uxth   r2, r3
08005066:   687b          ldr    r3, [r7, #4]
08005068:   801a          strh   r2, [r3, #0]
```

Assembly language is messy.

```
      *B = PWM_IDLE + Command;
08005050:    887b          ldrh   r3, [r7, #2]
08005052:    f603 0334     addw   r3, r3, #2100 ; 0x834
08005056:    b29a          uxth   r2, r3
08005058:    68bb          ldr    r3, [r7, #8]
0800505a:    801a          strh   r2, [r3, #0]
      *C = PWM_IDLE - Command;
0800505c:    887b          ldrh   r3, [r7, #2]
0800505e:    f5c3 6303     rsb    r3, r3, #2096 ; 0x830
08005062:    3304          adds   r3, #4
08005064:    b29a          uxth   r2, r3
08005066:    687b          ldr    r3, [r7, #4]
08005068:    801a          strh   r2, [r3, #0]
```

What does the processing?

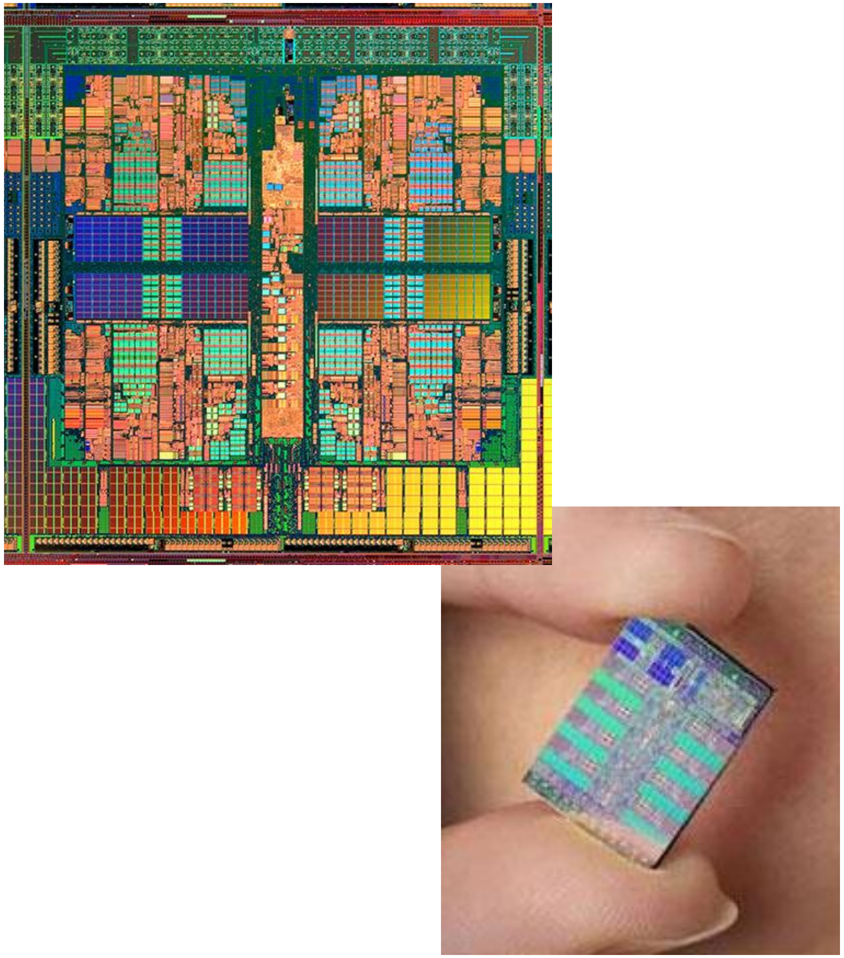


- Semiconductor chips!
- May use many computing units in parallel on one or many separate chips

Processor performance varies.

- How many bits?
- What kinds of instructions?
- How many instructions per second?

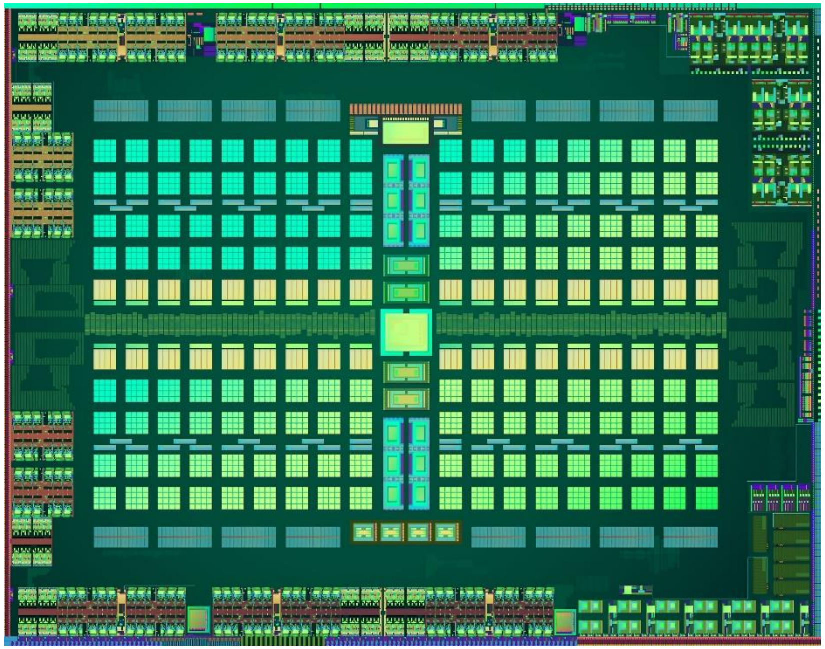
Multiple processors add challenges.



- Processors stopped getting faster in 2010
  - Still getting smaller and cheaper!
- Multiple programs at once
  - The OS handles it.
- Parallel processing
  - Your job!

GPUs have thousands of processors.

- 3D graphics requires a lot of math
- A GPU is optimized for math
  - >1000 processor cores
  - Many operate in lockstep
  - Simple instruction set
- Limited performance memory interface



Supercomputers have even more.

- Combination of thousands of ordinary desktop CPUs and GPUs
- >2,000,000 processor cores at the top end
- Requires special software

Next time: Storage and transport

Image References

Slide 1: *Abstraction*, by Randall Munroe, from https://xkcd.com/676/ (CC BY-NC 2.5)

Slide 3 (top left): by LG전자, from https://commons.wikimedia.org/wiki/File:Inductive_charging_of_LG_smartphone_(2).jpg (CC BY 2.0)

Slide 3 (top centre): by Raimond Spekking, from https://commons.wikimedia.org/wiki/File:Lenovo_G500s_laptop-2905.jpg (CC BY-SA 4.0)

Slide 3 (top right): by Gareth Halfacree, from https://www.flickr.com/photos/120586634@N05/26212930836 (CC BY-SA 2.0)

Slide 3 (bottom): from https://www.nesi.org.nz/services/high-performance-computing/platforms (UoA copyright)

Slide 4: from http://sites.harvard.edu/~chsi/markone/about.html

Slide 8 (top left): by Gregg M. Erickson, from https://commons.wikimedia.org/wiki/File:Motorola68040die.jpg (CC BY 3.0)

Slide 8 (top right): by Maurizio Pesce, from https://commons.wikimedia.org/wiki/File:USB_Type-C_macbook.png (CC BY 2.0)

Slide 8 (bottom left): by Daniel Sancho, from https://commons.wikimedia.org/wiki/File:8_bytes_vs._8Gbytes.jpg (CC BY 2.0)

Slide 8 (bottom right): by Zzubnik, from https://commons.wikimedia.org/wiki/File:Computer_monitor.jpg (Public domain)

Slide 9: from https://web.archive.org/web/20140414084027/http://blog.ioactive.com/2007_11_01_archive.html

Slide 13: by Jonathan Joseph Bondhus, from https://commons.wikimedia.org/wiki/File:Stack_of_Copy_Paper.jpg (CC BY-SA 3.0)

Slide 14: by Perplexeus, from https://commons.wikimedia.org/wiki/File:Haystack.png (CC BY-SA 4.0)

Slide 22: by Gregg M. Erickson, from https://commons.wikimedia.org/wiki/File:Motorola68040die.jpg (CC BY 3.0)

Slide 23 (top): from https://www.anandtech.com/show/13910/intels-core-i9-9900kf-listed-in-the-usa-582-usd

Slide 23 (bottom): from https://electronics.stackexchange.com/a/84851

Slide 24 (top): by AMD, from https://commons.wikimedia.org/wiki/File:AMD_Phenom_die_equalized.png (freely licensed)

Slide 24 (bottom): by Érick Luiz Wutke Ribeiro, also known as: Delemon, from https://commons.wikimedia.org/wiki/File:Cell-Processor.jpg (CC BY 3.0)

Slide 25: from https://www.extremetech.com/gaming/272764-new-amd-gpu-rumors-suggest-polaris-refresh-in-q4-2018

Slide 26: from https://www.extremetech.com/extreme/146967-stanford-breaks-million-core-supercomputing-barrier2