3. Using CPUs for Neural Networks

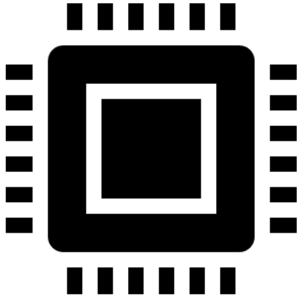
Philipp Holzinger

Content

CPU Architecture

1. Overview of Architecture
2. Optimizations useful for NNs
   - Memory optimizations
   - Vectorization
   - Threading

Platforms for RADL



CPU Architecture

Pseudocode to be Executed

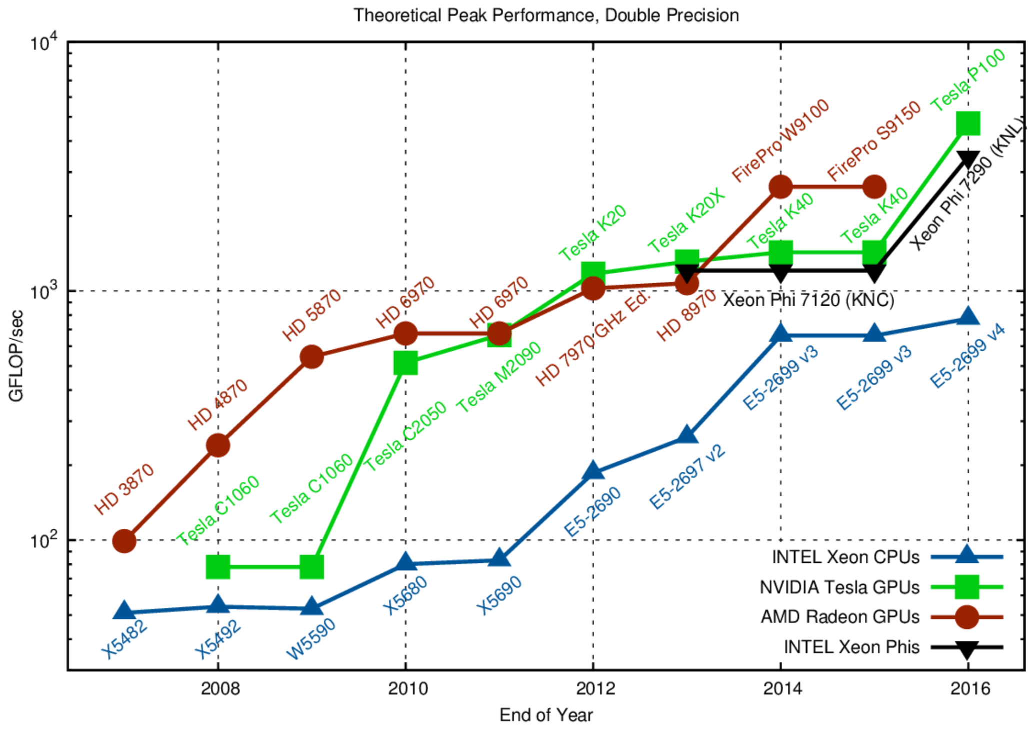
Fully connected layer

```
for (b = 0 to B-1)           // batch
  for (n = 0 to N-1)         // neuron
    for (i = 0 to I-1)       // input
```

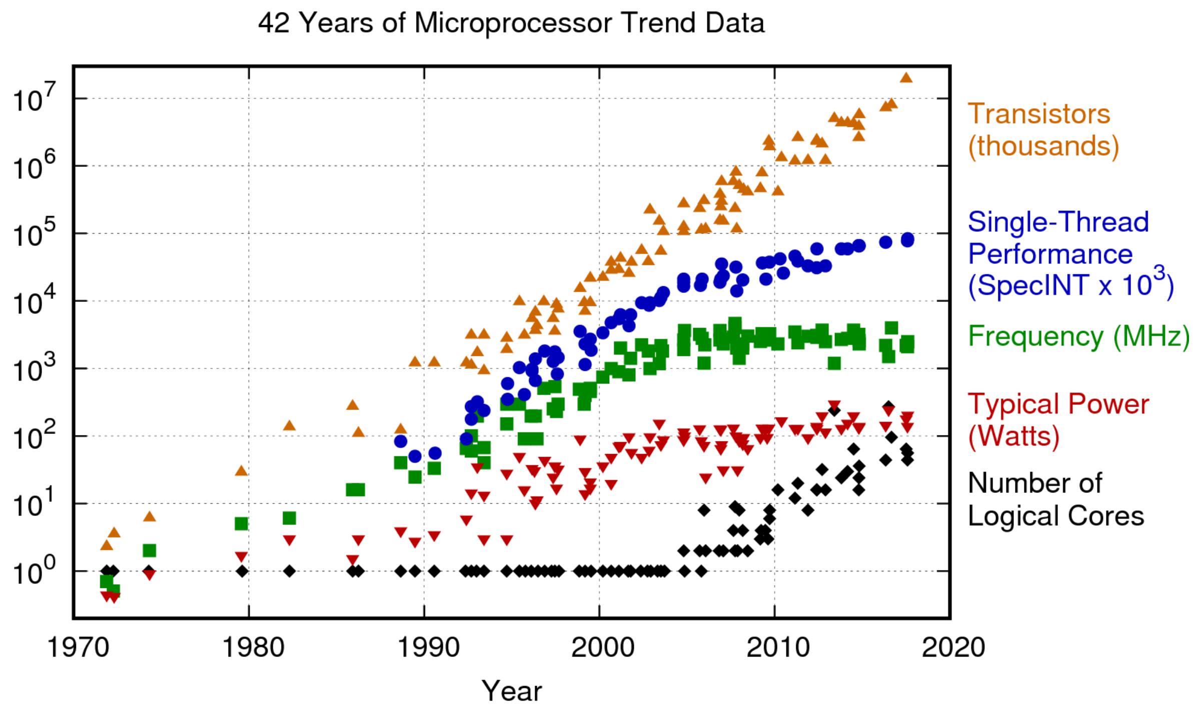
Convolutional layer

```
for (b = 0 to B-1)                  // batch
  for (k = 0 to K-1)                // output channels
    for (c = 0 to C-1)              // input channels
      for (x = 0 to X-1)            // input columns
        for (y = 0 to Y-1)          // input rows
          for (fx = 0 to FX-1)      // filter columns
            for (fy = 0 to FY-1)    // filter rows
```

CPU RAW Performance Evolution



There are some Restrictions



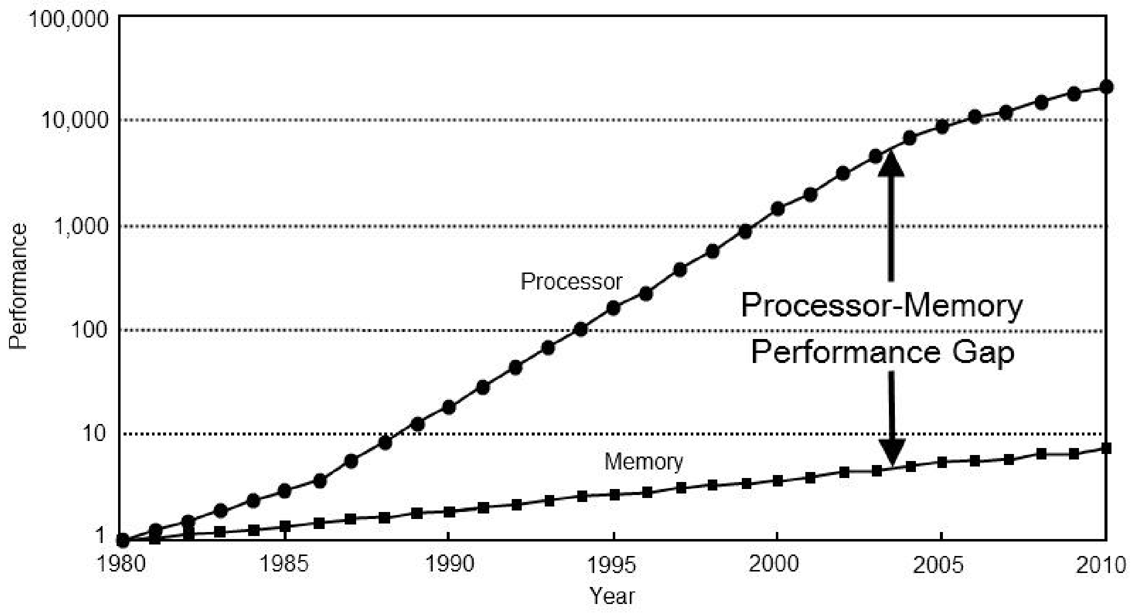
Original data up to the year 2010 collected and plotted by M. Horowitz, F. Labonte, O. Shacham, K. Olukotun, L. Hammond, and C. Batten
New plot and data collected for 2010-2017 by K. Rupp

Chip Area
Data Transport

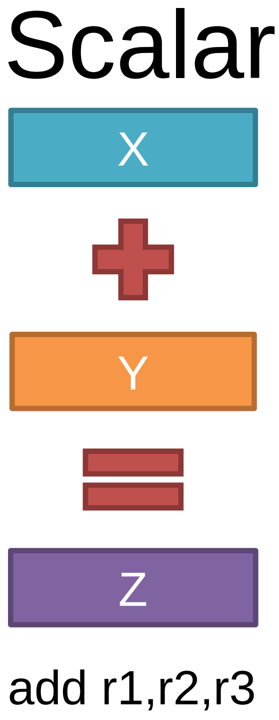
Dissipated Power
Physical Limits

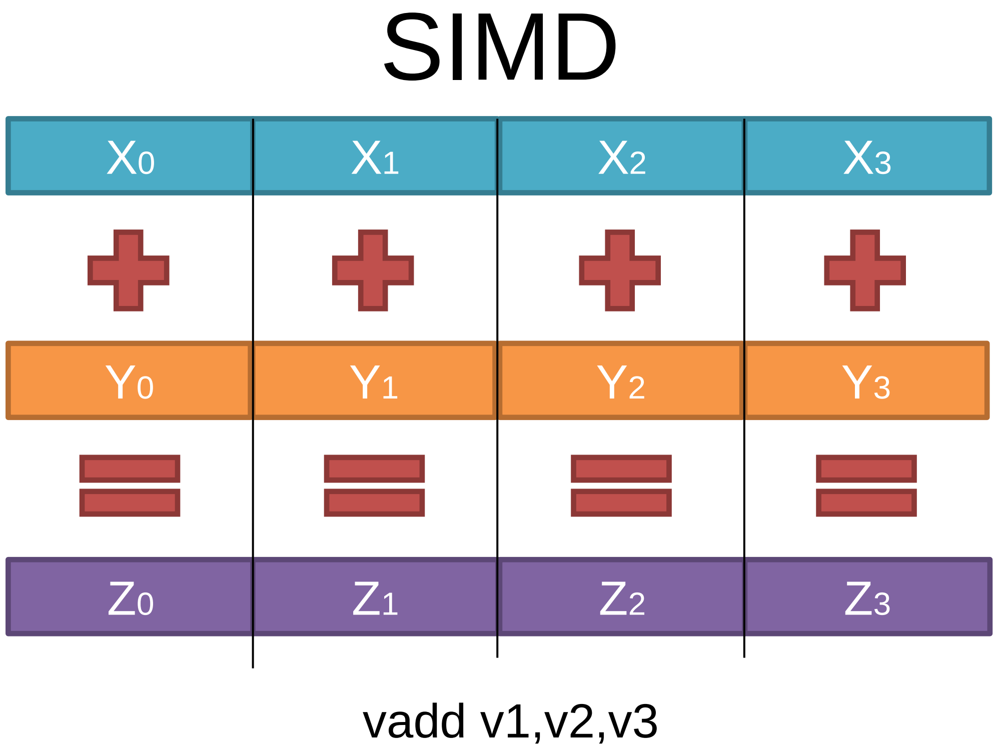
Chip Area
Data Transport

The Memory Gap

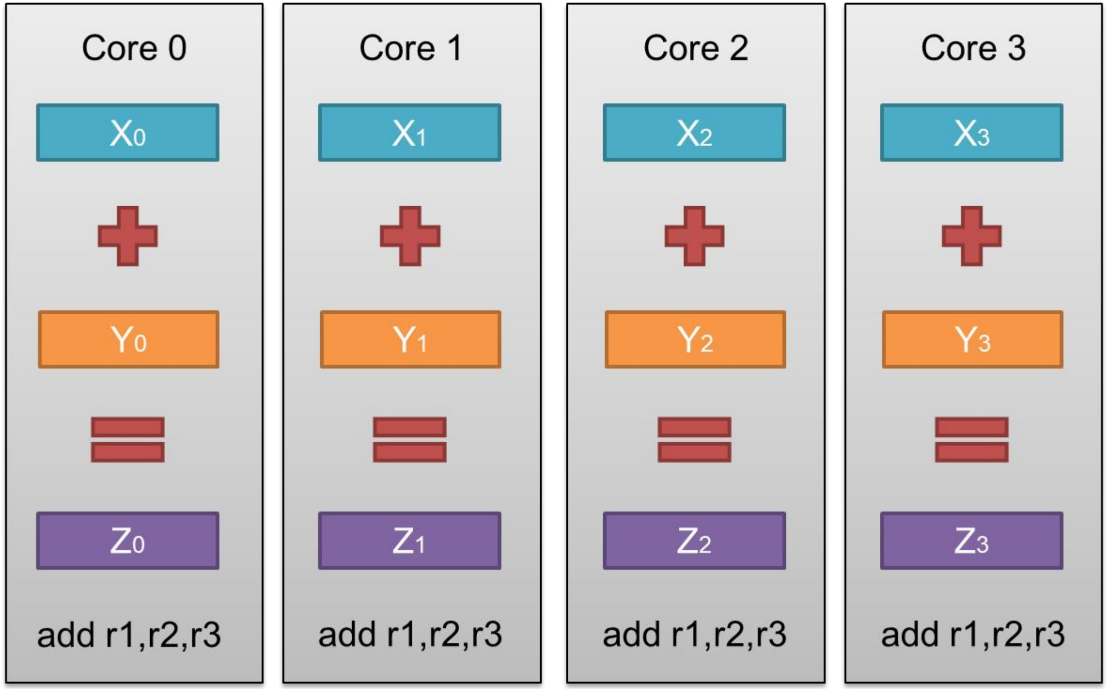


Vectorization



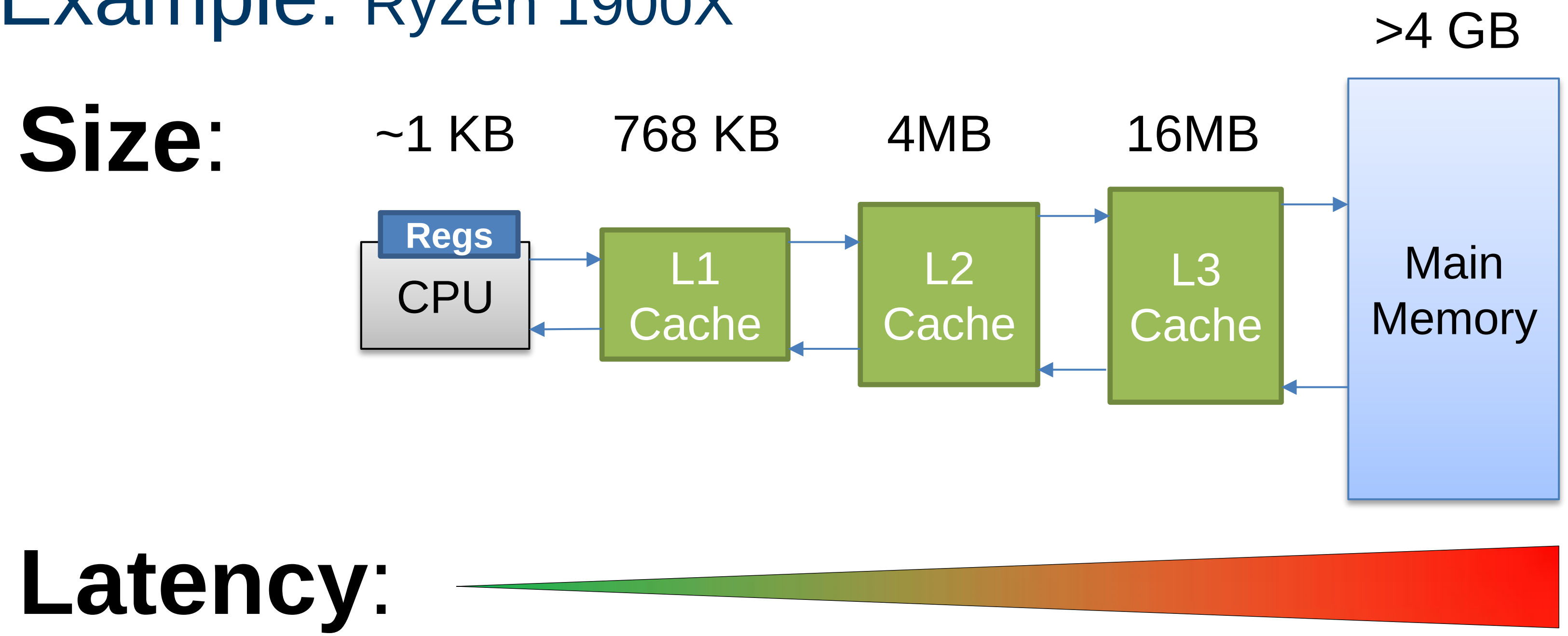


Threading

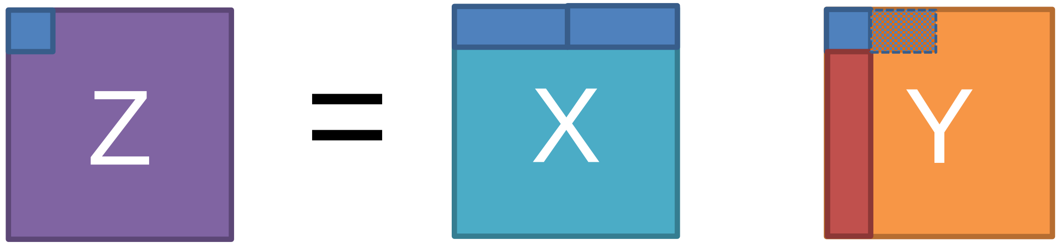

Core 0
X_0
Y_0
Z_0
add r1,r2,r3
Core 1
X_1
Y_1
Z_1
add r1,r2,r3
Core 2
X_2
Y_2
Z_2
add r1,r2,r3
Core 3
X_3
Y_3
Z_3
add r1,r2,r3

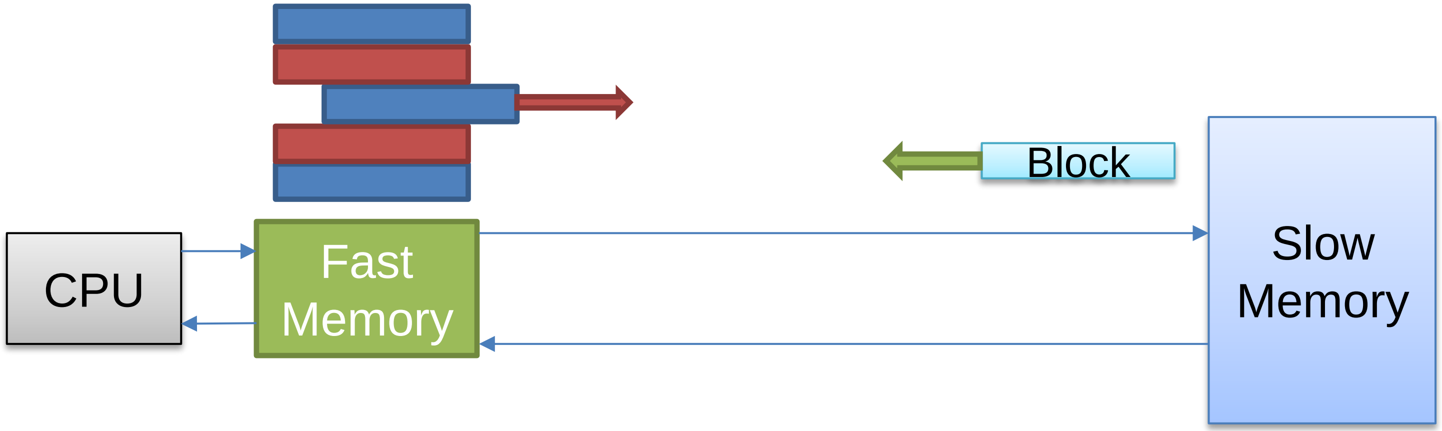
CPU Memory Hierarchy

Example: Ryzen 1900X

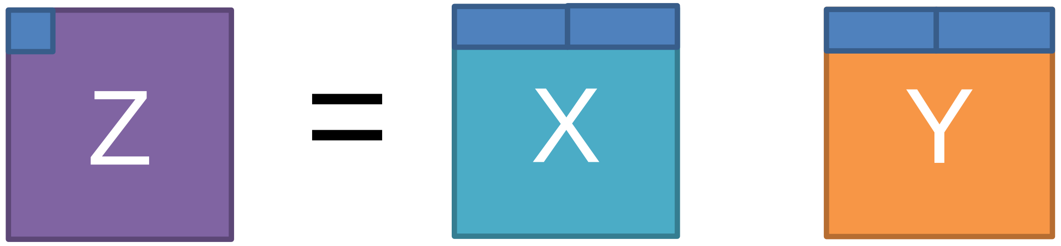


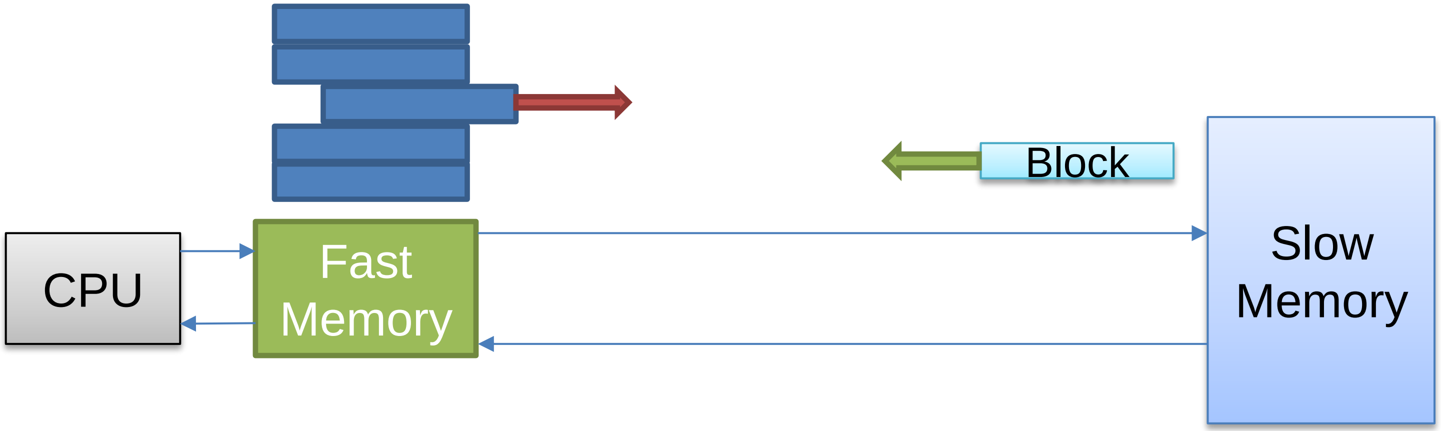
Data Layout and Access





Data Layout and Access





Data Layout and Access

The faster the memory, the more it is limited in size!

- Choose which data to keep in memory („**Cache Blocking**“)
  - Data Stationarity / Parallelism
- Consider smaller or fixed point data types („**Quantization**“)
  - More values per memory access and area
  - Performance vs accuracy!
- Consider sparse matrices („**Pruning**“)
  - Set close to 0 weights to 0 (do retraining if needed)
  - Use optimized matrix format (e.g. CSR) and only calcualte needed operations
  - Performance vs accuracy!

Platforms for RADL

X86 CPU - AMD Ryzen Threadripper 1900X

AMD Ryzen Threadripper 1900X

- 8 Cores / 16 Threads
- 3.8 GHz Base-Clock / 4.0 GHz Turbo

- 768 KB L1 Cache
- 4 MB L2 Cache
- 16 MB L3 Cache

180 W TDP

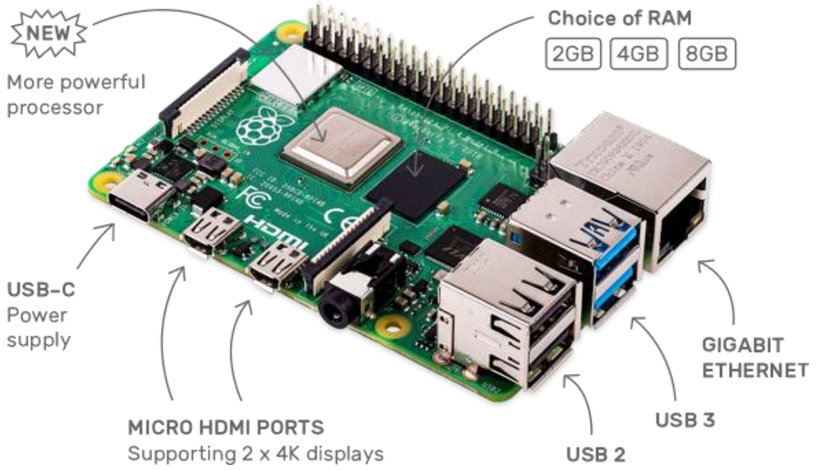
RasberryPI (ARM)

Broadcom BCM2711 (Cortex-A72 64-bit SoC)

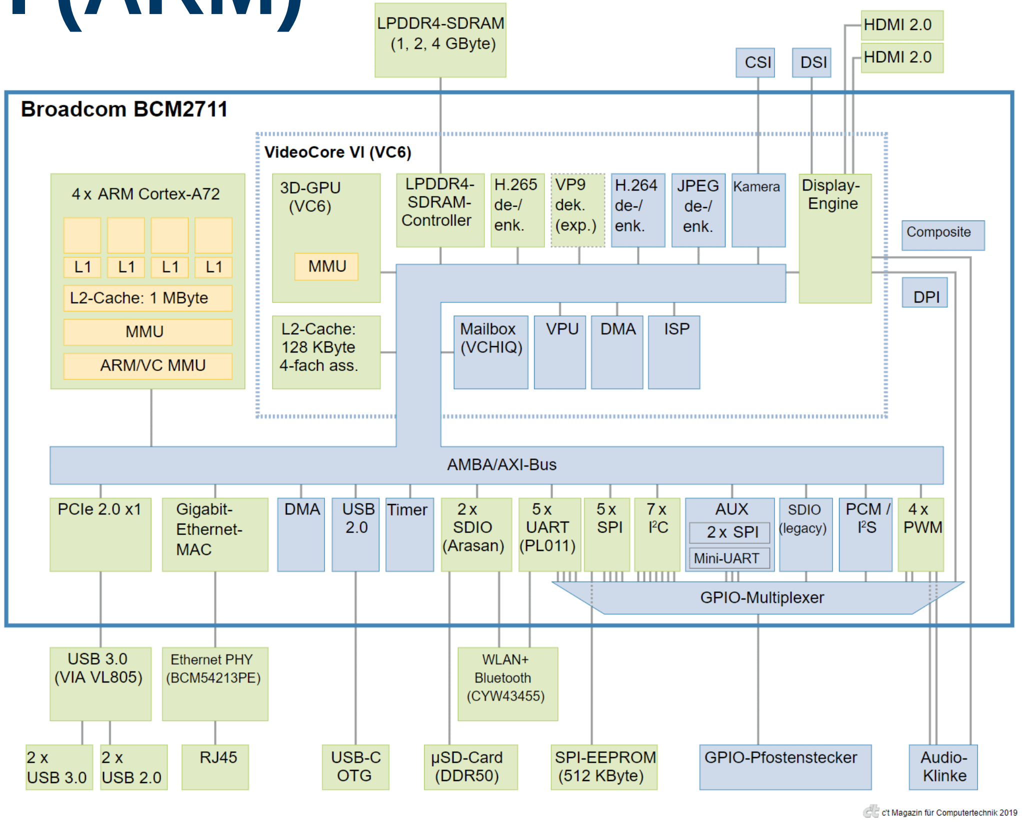
- 4 Cores / 4 Threads
- 1.5GHz Clock

- 32 KB + 48 KB * 4 L1 Cache
- 1 MB L2 Cache
- Up to 4GB LPDDR4

4 W TDP



RasberryPI (ARM)



Arduino (AVR)

ATmega328P 8Bit AVR

- 1 Core /1 Thread
- 16 MHz Clock (depends on occilator used)

- 2 KB RAM

~4 mW TDP
(heavily depending on peripherals!)

Sources

- https://www.karlrupp.net/2013/06/cpu-gpu-and-mic-hardware-characteristics-over-time/
- https://www.amd.com/de/products/cpu/amd-ryzen-threadripper-1900x
- https://www.raspberrypi.org/documentation/hardware/raspberrypi/bcm2711/README.md
- https://www.heise.de/ct/artikel/Raspberry-Pi-4-Model-B-Blockschaltbild-des-Broadcom-BCM2711-4514399.html
- https://store.arduino.cc/arduino-uno-rev3
- http://ww1.microchip.com/downloads/en/DeviceDoc/ATmega48A-PA-88A-PA-168A-PA-328-P-DS-DS40002061A.pdf
- https://learn.sparkfun.com/tutorials/reducing-arduino-power-consumption/all