# Lecture 14.01.2020

- we will be using ARM to do our work and learn a bit of assembly
- a *computer* architect has technology, goals, use, manufacturing in mind
- understand the principles and precedents of computer architecture
- evaluate trade-offs of different designs, develop principled or new designs
- architects need to keep past, future, goals, limitations and so on in mind
- general loop: fetch instruction from memory -> decode instruction -> execute instruction
- CISC - complete instruction set computing
- RISC - reduced instruction set computing – the decode part is more efficient here
- pipeline-ing: two pipelines to perform instructions
- our focus for design goals will be on performance, touch on power, cost, reliability
- another shaping force is the application of the chips: is it heavy floating point stuff or is it high data bandwidth and throughput?
- some domains: desktop (intel i7, amd ryzen), mobile (snapdragon), embedded (ARM chips), deeply embedded
- we will deal with general purpose CPUs, not special purpose chips
- EPIC - explicitly parallel instruction computing
- FPGA - field-programmable gate array – circuits are configurable in the field, so after purchase
- Moore's law: every ~18 months the number of transistors in a chip doubles
- current maximum is AMD's Epyc Rome with 39.54 billion MOSFETs in one chip
- DRAM - dynamic random access memory – constructed from one transistor and capacitor, this is faster than SRAM (static RAM), but consumer more power
- in is the 1970s ~25k resistors were enough to get 16-bit processors on one chip – the revolution began
- PDP - programmed data processors – old "computer" used for research and the like
- first microprocessor: Intel 4004, 10,000 nm, 2300 transistors, 108 kHz, 4 bit
- today: AMD Epyc Rome, 7 nm, 39.54 billion transistors, 2.25-3.4 GHz, 64 bit
- Intel Pentium 4 was a big improvement in 2003
- classifications of computers: SISD - single instruction stream single data stream; SIMD - single instruction stream multiple data stream; MISD - multiple instruction stream, single data stream; MIMD - multiple instruction stream multiple data stream (*most common today*)
- parallelism is good for performance, supercomputers today are *very* parallel in their design
- the types are cluster systems, NUMA (non-uniform memory access) system, MPP (massive parallel processing) system, SMP (symmetric multiprocessing) system
- implicit instruction-level parallelism: hardware provides parallel resources, figures out how to use them, the software is not involved
- **the main good textbooks are Patterson & Hennessey**
- MIPS ~ RISC in nature (microprocessor without interlocked pipelined stages)
- another revolution: explicit data and thread level parallelism: hardware has parallel resources and software specifies their usage, first for 4 parallel multiplications – GPUs with highly parallel architectures can also be used for certain computing jobs