Computer Hardware Architecture
Lecture 3

Manfred Liebmann
Technische Universität München
Chair of Optimal Control
Center for Mathematical Sciences, M17
manfred.liebmann@tum.de

November 9, 2015

Classical Von Neumann Architecture

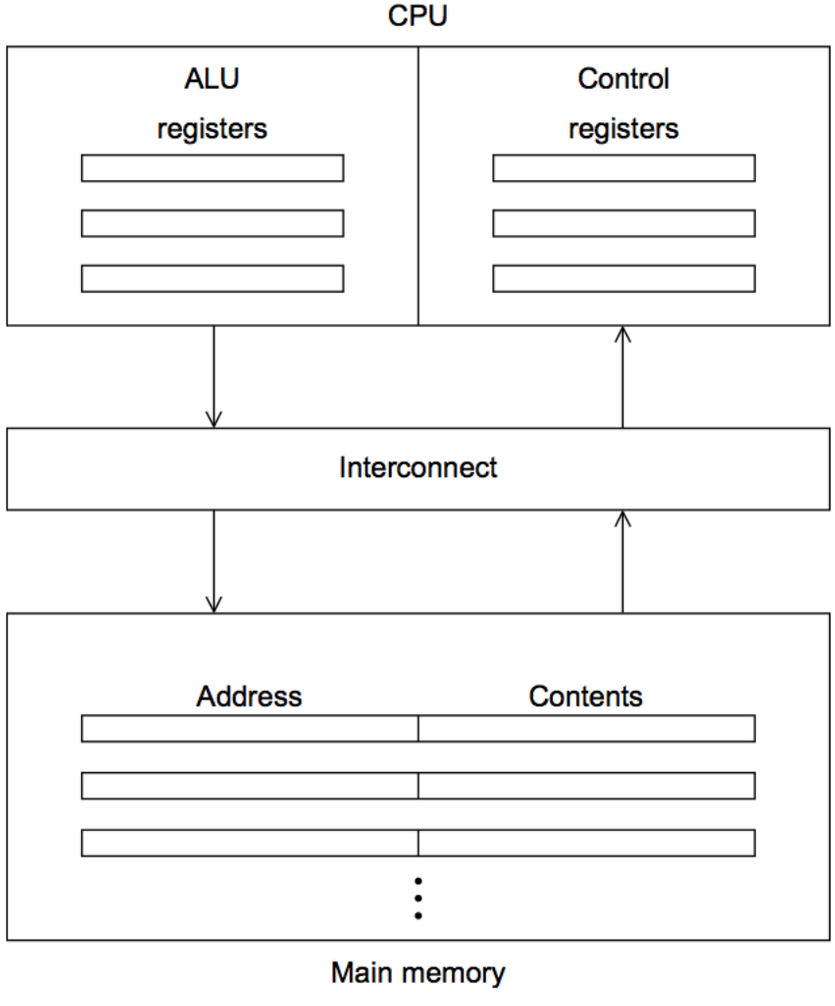

Figure 1: The von Neumann architecture consists of main memory, a central processing unit (CPU) or processor or core, and an interconnect between memory and CPU

Classical Von Neumann Architecture

- **Overview of the Architecture**
  - Main memory is a collection of locations identified by addresses
  - Addresses are used to access the data or instructions stored in these locations
  - The CPU is divided into a *control unit* and *arithmetic and logic unit* (ALU)
  - The control unit decides which instructions in a program are executed
  - The arithmetic and logic unit (ALU) actually executes the instructions
  - The CPU contains very fast storage called *registers*
  - Registers are used as input and output for arithmetic and logic instructions
  - Control registers store the *program counter* (PC)
  - The PC stores the address of the next instruction to be executed
  - Instructions and data are transferred over the interconnect
  - Data or instructions are *fetched* or *read* from memory
  - Data or instructions are *stored* or *written* to memory
  - *Von Neumann bottleneck* caused by the separation of CPU and memory

Processes and Threads

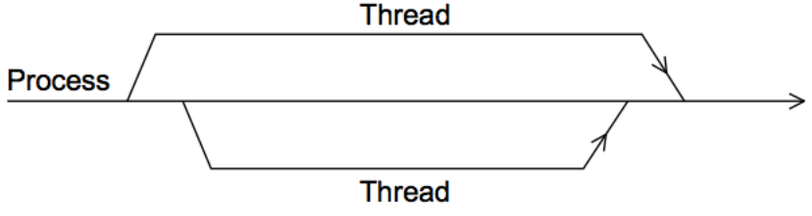

Figure 2: A process with two threads

The *operating system* (OS) manages the hardware and software resources of a computer. When a user runs a program the operating system creates a *process*, an instance of the program being executed. A process can execute multiple threads with access to the process resources.

Processes and Threads

- **A process consists of several entities**
  - The executable machine language program
  - A block of memory for the executable program
  - A *call stack* that keeps track of active functions
  - A *heap* for memory management
  - Descriptors of OS resources, i.e. files descriptors
  - Security information, i.e. access to hardware and software resources
  - State of the process: Ready to run, Waiting for resource
  - Multiple threads with their own execution context

Improving the Von Neumann Architecture: Caches

- **Problem**: The interconnect between the CPU and main memory is a bottleneck.
- **Solution**: Add *cache* memory.

In general a *cache* is a collection of memory locations that can be accessed much faster than main memory. There is often a hierarchy of caches divided into *cache levels*.

Caches are usually small compared to main memory and are build around the concept of **temporal locality** and **spacial locality**.

When a cache is checked for information and the information is available, it is called a *cache hit*, otherwise a *cache miss*.

Caches are partitioned into **cache blocks** or **cache lines**, typically with a size of 64 bytes.

n-Way Set Associative Caches

	Cache Location		
Memory Index	*Fully Assoc*	*Direct Mapped*	*2-way*
0	0, 1, 2, or 3	0	0 or 1
1	0, 1, 2, or 3	1	2 or 3
2	0, 1, 2, or 3	2	0 or 1
3	0, 1, 2, or 3	3	2 or 3
4	0, 1, 2, or 3	0	0 or 1
5	0, 1, 2, or 3	1	2 or 3
6	0, 1, 2, or 3	2	0 or 1
7	0, 1, 2, or 3	3	2 or 3
8	0, 1, 2, or 3	0	0 or 1
9	0, 1, 2, or 3	1	2 or 3
10	0, 1, 2, or 3	2	0 or 1
11	0, 1, 2, or 3	3	2 or 3
12	0, 1, 2, or 3	0	0 or 1
13	0, 1, 2, or 3	1	2 or 3
14	0, 1, 2, or 3	2	0 or 1
15	0, 1, 2, or 3	3	2 or 3

Figure 3: Assignment of 16 lines of main memory to 4 lines of cache memory

Cache blocks are not arbitrarily mapped from main memory, but are organized n-way set associative on modern CPUs!

Virtual Memory and TLB

Virtual memory was developed so that main memory can function as a cache.

The main memory is partitioned into **pages**, i.e. memory blocks of typically 4096 Bytes.

The pages can be stored in *swap space* on secondary storage, i.e. hard drives.

The CPU keeps a **page table** that keeps track where a page is stored in physical memory or on disk and transfers the data from disk to memory if a memory location within the page is accessed by the CPU.

The CPU keeps a special cache for the address translations of the page table, the so called **translation-lookaside buffer** (TLB).

Virtual to physical address translation is expensive and can cause performance degradation!

TLB cache misses are expensive!