Moore's Law

Transistor count doubles every two years.

Different types of parallelism

- Hyperthreading: Two Reg-Sets per core
- Multi-Core: Multiple Cores per CPU
- Multi-Processor: Multiple CPUs per machine
- Compute-Cluster

Parallelism	**Concurrency** (Nebenläufigkeit)
Decomposition of a program into several sub programs, which run simultaneously on several processors ⇒ Faster Programs	Interleaved (time shared) execution that accesses shared resources ⇒ Simpler programs. Sometimes with time slicing (but not necessarily).

Process	**Thread**
Heavyweight, OS only needs process context to run a program correctly, own address space.	Lightweight, a process can have multiple threads, parallel sequence in a program, same address space, separate stack and registers.

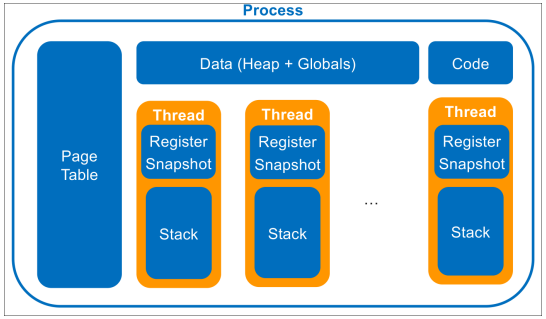

Figure 1: Memory utilization/resources

Multiple threads can write in the same memory locations ⇒ Needs explicit synchronization.

Multiplexing Interleaved execution by using context switching.

Context switching

- Synchron: Waits for condition
  - Queues itself as waiting and gives processor free
- Asynchron: Timing
  - After a defined time, the thread should release the processor
  - Prevents a thread from permanently occupying the processor

Multitasking (scheduling)

- Cooperative (rarely used nowadays)
  - Threads must explicitly initiate context switches at intervals
- Preemptive (nowadays standard)
  - Scheduler interrupts the running thread asynchronously via timer interrupt
  - Time sliced scheduling: Each thread has the processor for maximum time interval

Thread states

Running, Waiting, Ready

JVM

- Single process system
- Runs as long as threads are running (not until main() is done!), unless marked as daemon-thread
- `System.exit() / Runtime.exit()`: Uncontrolled stop of all threads
- Threads realized by `Thread`-class and `Runnable`-interface
  - `void run()` can be overridden for custom behaviour
- `thread.start()` starts a Thread, JVM calls `run()` (do not call run manually!)
- If exception is unhandled, other Threads still continue
- Just like any other application threads
- Scheduling of threads handled by the OS
- Allows setting priorities to threads → still managed by OS

Warn

```
var t1 = new Thread(() ->
{ System.out.println("Hi from t1"); })
t1.setDaemon(true) // Stops running when main-thread is finished
t1.start();
```