# Advanced CPU design and optimization

Nico Fröhlich

Troblecodings

July 22, 2020

# Quellen und weiterführende Inhalte

- Intel® 64 and IA-32 Architectures Software Developer's Manual
- Software Optimization Guide for AMD64 Processors
- Intel® 64 and IA-32 Architectures Optimization Reference Manual
- Chandler Carruth, Matt Godbolt, Andrei Alexandrescu ...
- Mike Acton
- https://www.quick-bench.com/q/Ev6U1Owr7WKTbzyNgCol7vEJq4o

# Einführung

Das was alle CPUs gemeinsam haben:

- Clock (Oszillator)
- Fetch-Decode Cycle
- Instruction Counter

# Fetch - Decode - Execute Cycle

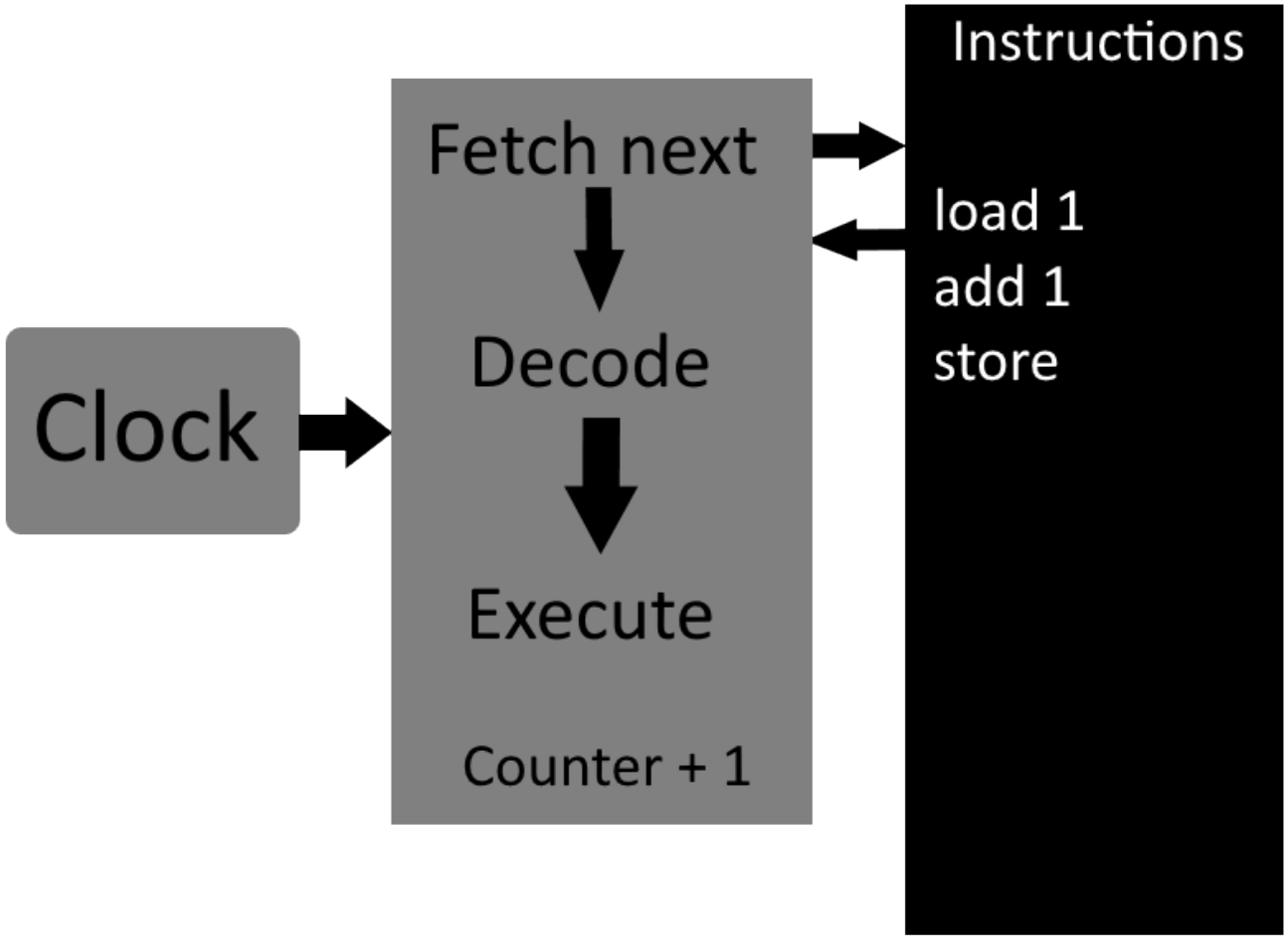


Figure: CPU Cycle

# Let's talk cache

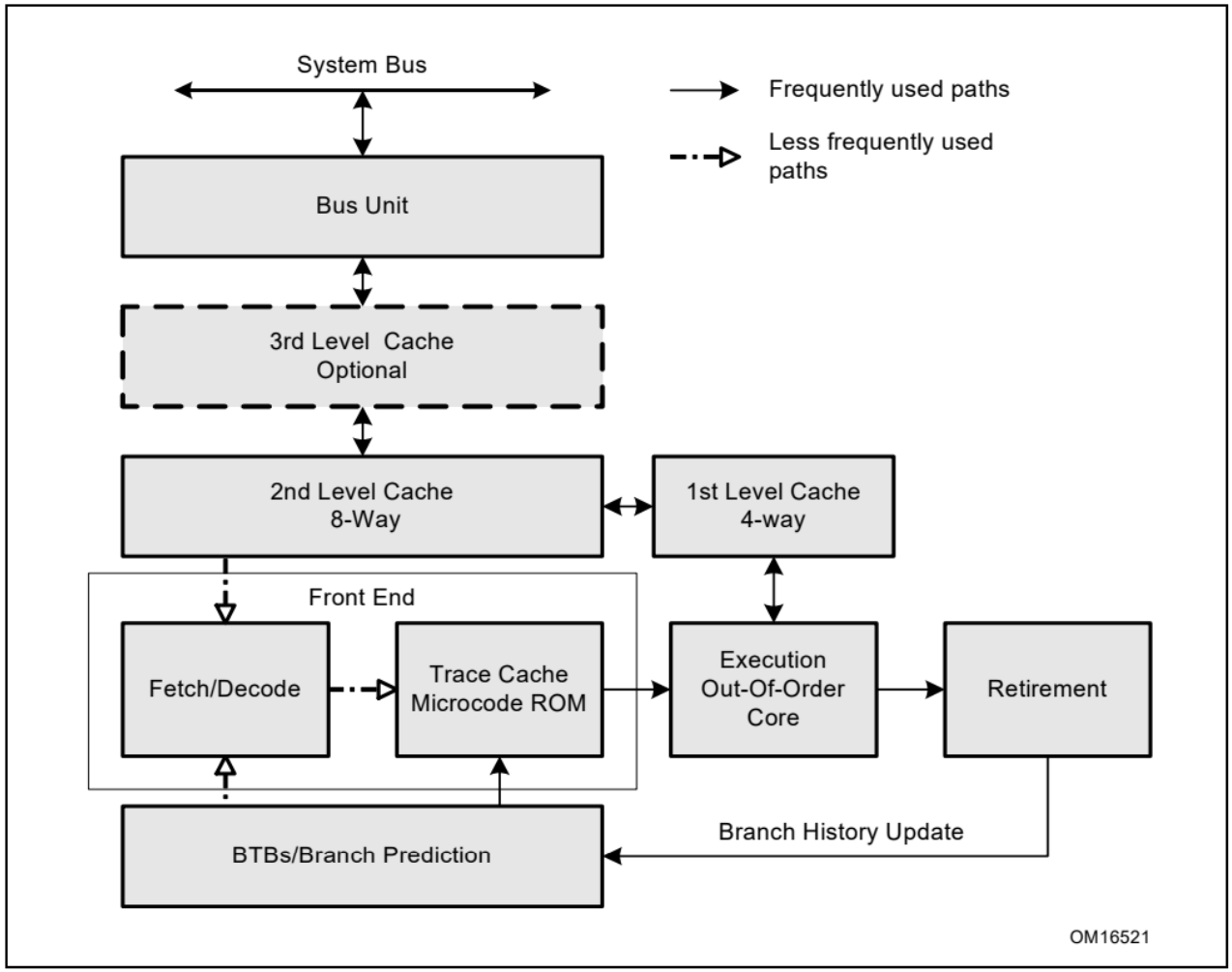


Figure: The Intel NetBurst Microarchitecture

# Was passiert?

- Die OOE zerlegt die gecacheten instructions
- Schaut nach was für memory gentutzt wurde
- Fängt an die wahrscheinlichsten Sachen in den Cache zu laden
- Executed was er kann
- RU Updatet history und reordert instructions

# Cache misses

```
Core i7 Xeon 5500 Series Data Source Latency (approximate)              [Pg. 22]

local  L1 CACHE hit,                                    ~4 cycles (   2.1 -  1.2 ns )
local  L2 CACHE hit,                                   ~10 cycles (   5.3 -  3.0 ns )
local  L3 CACHE hit, line unshared                     ~40 cycles (  21.4 - 12.0 ns )
local  L3 CACHE hit, shared line in another core       ~65 cycles (  34.8 - 19.5 ns )
local  L3 CACHE hit, modified in another core          ~75 cycles (  40.2 - 22.5 ns )

remote L3 CACHE (Ref: Fig.1 [Pg. 5])          ~100-300 cycles ( 160.7 - 30.0 ns )

local  DRAM                                                      ~60 ns
remote DRAM                                                     ~100 ns
```

Figure: I7 Messungen

# Continues Memory ist gut

```
for(auto& x : vec) {
  x++;
}
for(auto& x : lis) {
  x++;
}
```

Figure: Linked Lists

```
for(auto& x : vec) {
  x++;
}
for(auto& x : vec2) {
  x++;
}
```

Figure: Continues memory

# Ergebnisse

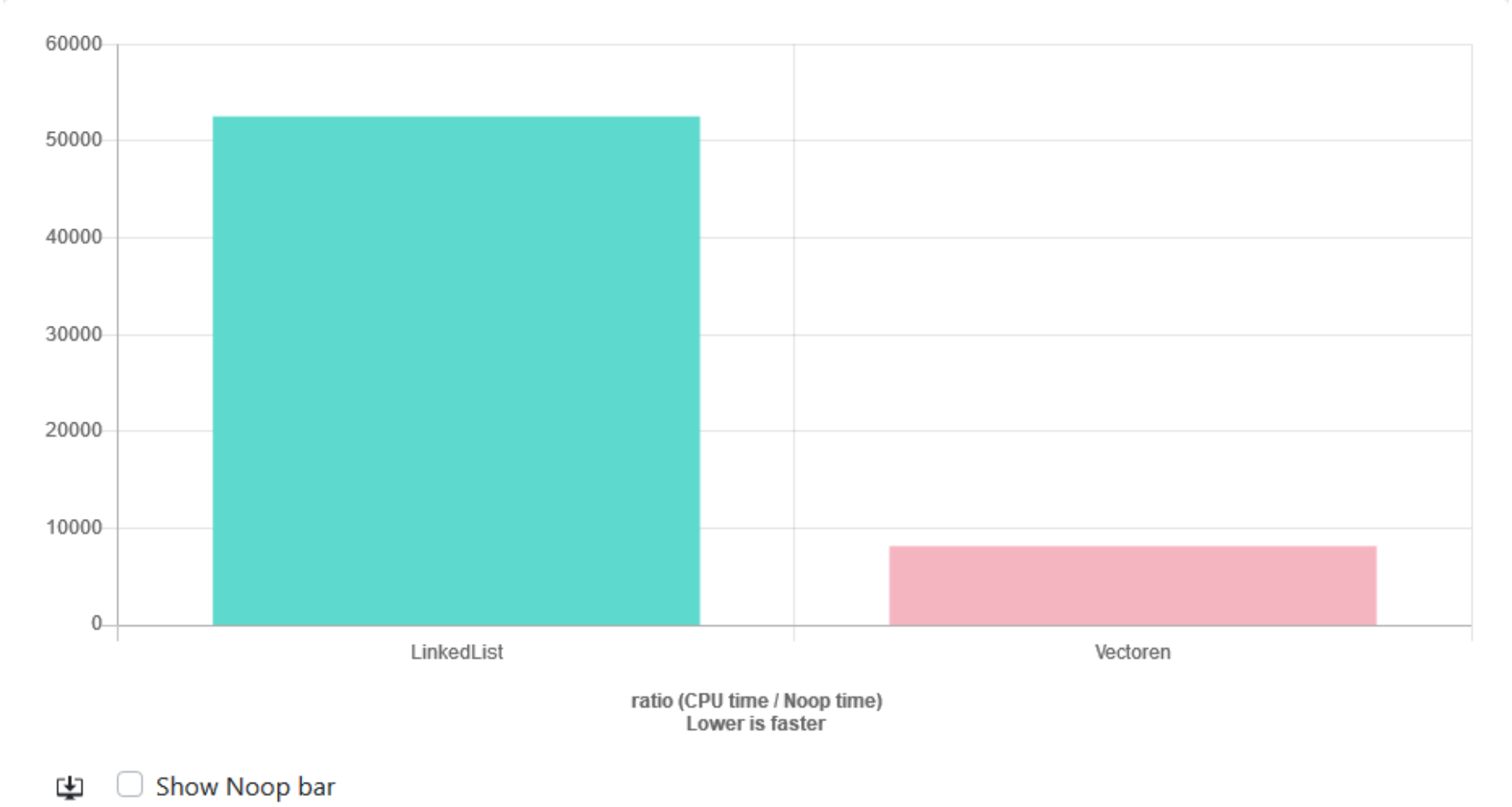


Figure: Quickbench ergebnisse