

→ There are 10 cache optimization techniques .

| Technique | Hit time | Bandwidth | Miss penalty | Miss rate | Power consumption | Hardware cost/complexity | Comment |
|---|----------|-----------|--------------|-----------|-------------------|--------------------------|---|
| Small and simple caches | + | | | – | + | 0 | Trivial; widely used |
| Way-predicting caches | + | | | | + | 1 | Used in Pentium 4 |
| Pipelined cache access | – | + | | | | 1 | Widely used |
| Nonblocking caches | | + | + | | | 3 | Widely used |
| Banked caches | | + | | | + | 1 | Used in L2 of both i7 and Cortex-A8 |
| Critical word first and early restart | | | + | | | 2 | Widely used |
| Merging write buffer | | | + | | | 1 | Widely used with write through |
| Compiler techniques to reduce cache misses | | | | + | | 0 | Software is a challenge, but many compilers handle common linear algebra calculations |
| Hardware prefetching of instructions and data | | | + | + | – | 2 instr., 3 data | Most provide prefetch instructions; modern high-end processors also automatically prefetch in hardware. |
| Compiler-controlled prefetching | | | + | + | | 3 | Needs nonblocking cache; possible instruction overhead; in many CPUs |

Figure 2.11 Summary of 10 advanced cache optimizations showing impact on cache performance, power consumption, and complexity. Although generally a technique helps only one factor, prefetching can reduce misses if done sufficiently early; if not, it can reduce miss penalty. + means that the technique improves the factor, – means it hurts that factor, and blank means it has no impact. The complexity measure is subjective, with 0 being the easiest and 3 being a challenge.