

# The Cache Memory

An Unpredictable Hardware Component in Single- and Multiprocessor Real-Time Systems

## What is a cache memory?

### A problem

New VLSI technology has resulted in more dense structures in electronic components which ends with either *faster* or more *complex* chips. The chips can get faster since they will get smaller due to higher "gate-density" and when the speed of light is one limiting factor, smaller chips can run at a higher clock rate than it's predecessors. The new technology can also be used to put more gates and larger constructions on the same chip size. Fast CPU:s and large primary memories are classical examples to the above story.

The performance gap between CPU:s and primary memory has widened the past decades and the trend is increasing.

```
int funk(int term){
  int vector[SIZE];
  int i, sum=0;
  for(i=0;i<SIZE;i++){
    vector[i] +=term;
    sum +=vector[i];
  }
  return sum;
}
```

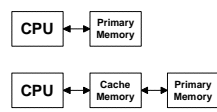
### Observation of locality

Studies has been made on program structure and implementations, and they show that *locality* is a key word. Temporal locality states that if a program use an address the chance is bigger to use it in the future than an arbitrary other address. The other kind of locality, the spatial, states that items that are close to each other in address space tend to be referred close in time too.

The statements above is proved by the fact that instructions are formed in sequences and loops, and that data is often allocated as stacks, vectors, strings and matrices.

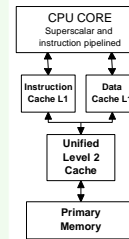
### A solution: the cache memory

A cache memory is a small memory that is located between the CPU and the main memory. It's small and fast, but it can only have a fraction of all data and instructions in mind. A typical size of a cache memory is 8-32kB and the access time is one CPU-clockcycle.



When the CPU puts an address on the bus, the cache controller checks if that address' data is in the cache. If it is (called a *hit*), the cache provides the data fast back to the CPU. If the check is a miss, the cache must swap out some data that probably won't be used with the requested one from the primary memory. Typical hit ratios are between 80-97% which can increase the average performance on a modern CPU by 5-50 times.

With the spatial behavior of programs in mind, much sense is to load a block of data from the underlying hierarchic memory at once to take advantage over the greater bandwidth and long latency that characterizes primary memory.

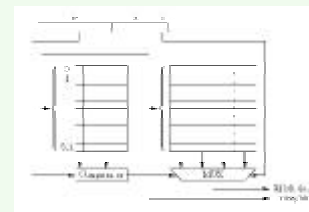


### Enhancements

The cache memory concept can be exploited by splitting the cache memory into a data and an instruction part and by this double bandwidth which is desirable in an instruction pipeline structure of the CPU. By adding a level-2 cache that fetches up the majority of the level-1 caches misses can also increase performance. A typical size of an L2-cache is 64kB-2MB.

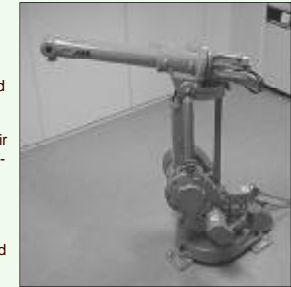
### Implementation

A cache memory resembles of a hash-table that has been implemented in hardware. The address is the key and the set of blocks associated with the key is the location where the datablock can be placed. A simple form of a cache memory is direct mapped, where only one location to place the block in the cache is possible.



## Real-Time and Unpredictable Hardware

A Real-Time system is when the correctness of the answer of a task not only depends of the correct answer but also when it is given. If the answer is given too late or to early it is considered as wrong. Early answers can however easily be corrected by a delay. In a safety critical real-time system, such as air bags in cars or air craft steering mechanism, the constructor must be able to guarantee the correctness in both answer and time. To try to solve the problem by just replacing a slow computer with a fast will not always succeed in complicated processes such as a walking legs or a robot that would scratch the surface of the car its painting - it would maybe just do the flaw faster!



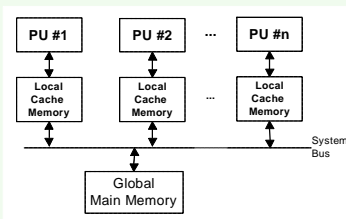
How long it takes for the computer to give an answer can be calculated by adding the time it takes for each instruction to execute. Problems occur when the analyst doesn't know how many times a loop will run or if a large piece of code will be omitted by a selective jump. In these cases a Worst Case Execution Time (WCET) can be calculated, which in many cases is too pessimistic - but safe.

The cache memory makes it even more difficult to calculate the WCET since the analyst doesn't know whether the data or instruction is in the cache or not - the ratio between these states can be as much as 100 times the execution time. In an event-driven system where no-one really knows in what order the programs will execute, it is almost impossible to give any hard guarantees without putting restrictions into the system.

## Tightly Coupled Multiprocessor Systems

A multiprocessor (MP) system is a computer with several connected Processing Units (PU) where there are several strategies to split the work between the PU:s. A "Multiple Instruction Multiple Data-structure" (MIMD) lets all the PU:s work with own instructions and own data. A loosely coupled MIMD sends messages between the PU:s in a network and a tightly coupled MIMD communicates through a global memory.

The tightly coupled MIMD can take great advantage of cache memories. Cache memories will also unload the heavily used system bus and bring down latency in the normal locally way, but also globally in the whole system. To keep the caches and the global memory consistent, special cache coherency protocols must be used.



## The SARA Project

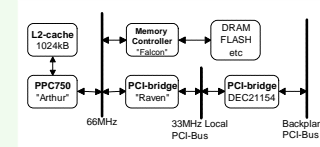
"SARA" stands for "Scaleable Architecture on Real-time Application" and is an umbrella project at the Computer Architecture Laboratory (CAL) at Mälardalen University, Sweden. SARA's first step is to solve performance problems in industrial applications. The concept is to make an extremely scaleable framework for hard- and software. The 9th objective of the program deals with predictability.

"9. Predictability. The soft- and hardware should be partly predictable. Some part could meet hard predictability requirements."

Why the objective only requires some parts of the system should be predictable, instead of the complete system has to do with the simple fact that in most cases only a (small) part of a system has real hard Real-Time requirements. Many parts of a sys-

tem have soft real-time requirements and if they don't meet the time constraints the system will still work, but with a reduced quality of service.

SARA will be implemented with new modern processors ... with cache memories. As mentioned in the section about Real-Time and unpredictable hardware components, cache memories can introduce difficulties to prove if a Real-Time system works. SARA will try to solve these problems by for instance adding restrictions, special scheduling and/or prefetching by other co-operating components that are connected with the operating system.



This is a Motorola CPX2000 system with a Compact-PCI backplane bus.

Evaluation tests has been made in the SARA project with three Power PC750-CPU:s that runs at 366MHz and are loosely coupled via PCI bridges on the backplane.

Each PPC750 is equipped with 2x32KB level-1 cache memory that is 8-way set-associative and a unified 1MB level-2 cache memory with 2-way set-associativity.

