Cache

Computer Engineering 2

CT Team: A. Gieriet, J. Gruber, R. Gübeli, M. Meli, M. Rosenthal,
A. Rüst, J. Scheier, M. Thaler

Agenda

- **Principle of locality**
- **Cache mechanics**
- **Cache organization**
  - Fully associative
  - Direct mapped
  - N-way set associative
- **Performance metrics**
- **Cache misses**
- **Replacement strategies**
- **The programmers perspective**

Motivation

- **Situation**
  - Processor
    - Fast cycle time
  - Fast DRAM[1]
    - Slow cycle time (up to 100x)
    - Efficiently reads only in bursts

  → Bridging the gap such that pipelining is effective!

- **Goal**
  - Access "slower" memory in bursts and maintain a fast cache for fast single accesses
  - But: Data integrity must be carefully managed, such that both, cache and main memory have the same data

[1] *Dynamic RAM*

Learning Objectives

- **At the end of this lesson you will be able**
  - to understand the principles of cache memory
  - to explain the principle of locality
  - to enumerate advantages and disadvantages of different cache models
    - Fully associative
    - Direct mapped
    - N-way set associative
  - to enumerate types of cache misses
  - to understand how cache size and cache hit rate are related
  - to name different replacement strategies

Principle of Locality

- **Principle of locality**

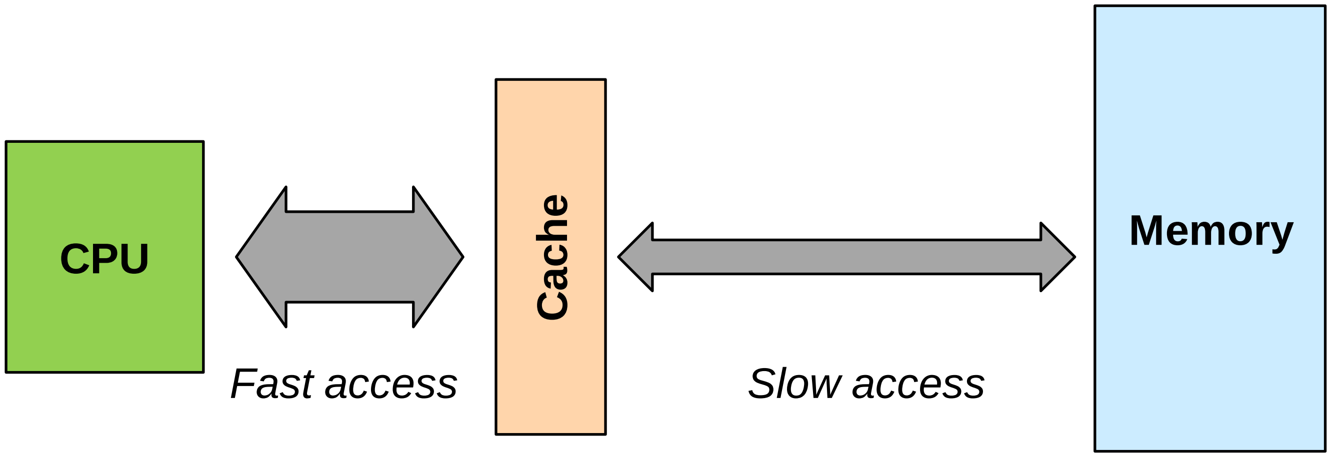
> Programs usually access small regions of memory in a given interval of time

- Temporal locality
  - Current data location is likely being accessed again in near future
- Spatial locality
  - Current data location is likely being close to next accessed location

```
for(i = 0; i < 100000; i++) {     // incremental access
    a[i] = b[i];                  // → spatial locality
}
if (a[1234] == a[4321]) {         // → temporal locality
    a[1234] = 0;
}
```

Cache Mechanics

- **Definition cache**
  - Computer memory with short access time
  - Storage of frequently or recently used instructions or data



CPU
Very small
Registers (Word)

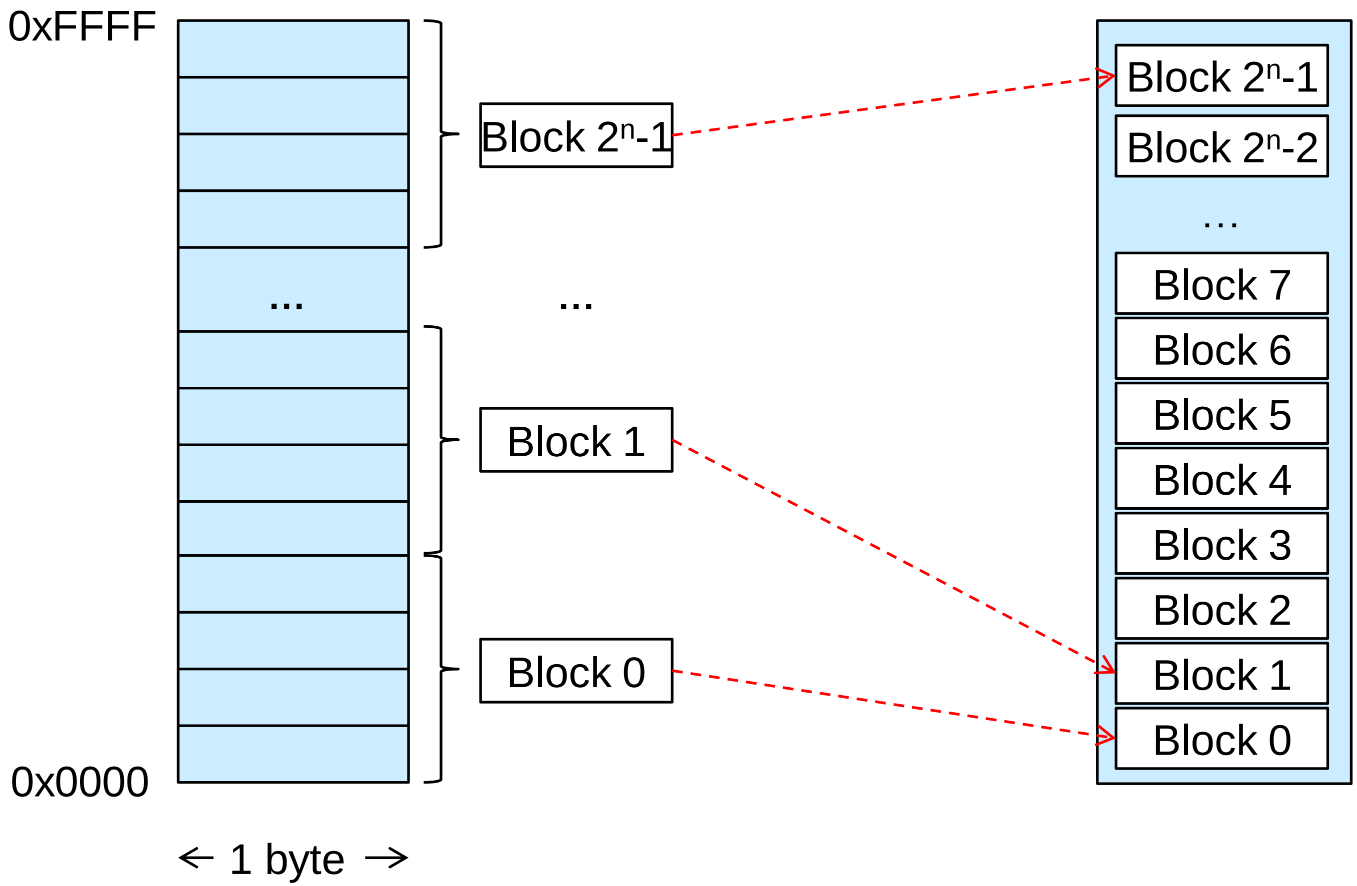
Cache
Small, fast, expensive
Caches subset of memory blocks

Memory
Large, slow, cheap
Partitioned into memory blocks

Cache Mechanics

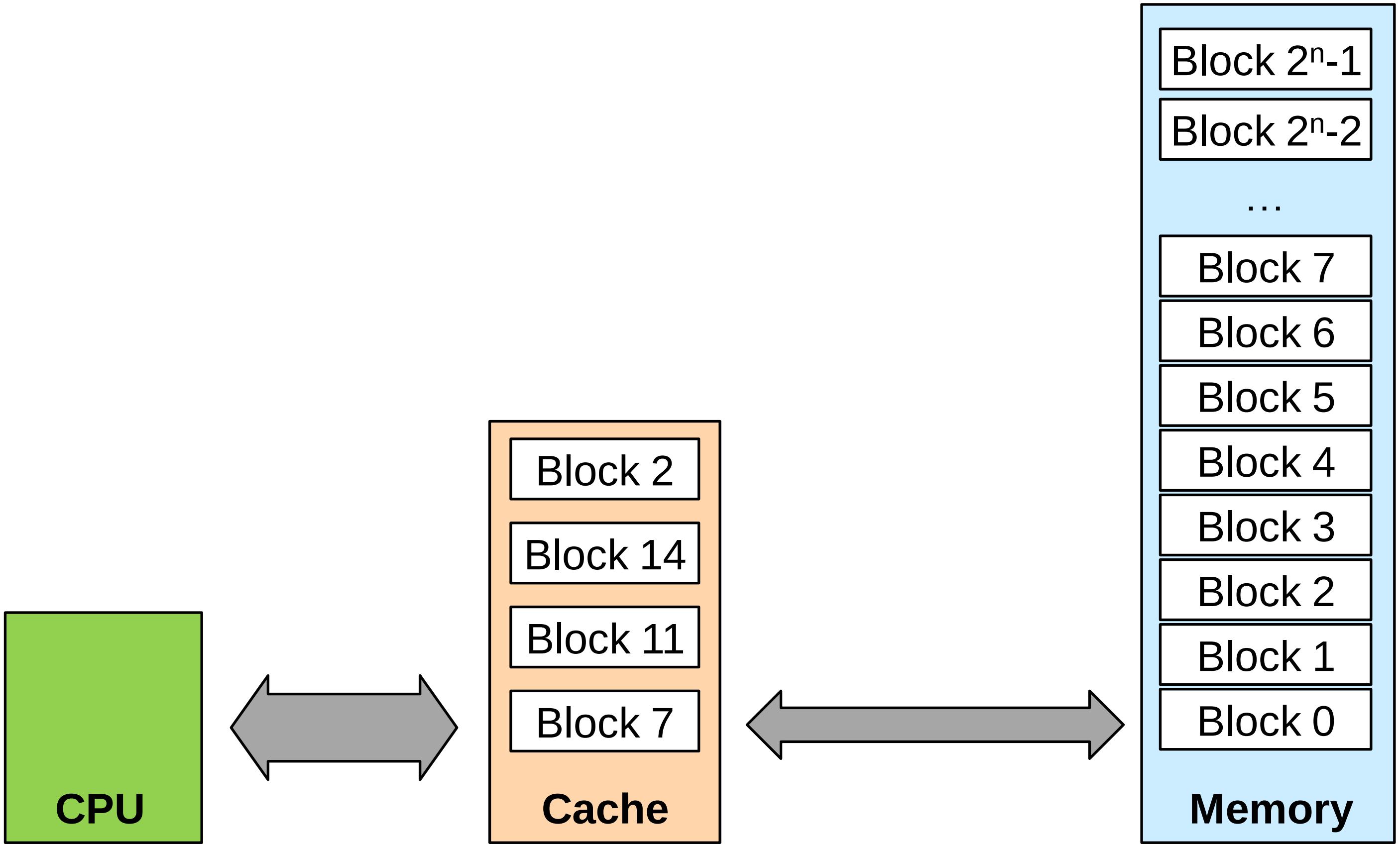
- **Memory blocks**

All examples given in this lecture are using
- ***hypothetical 16 bit addressing***
- ***blocks of 4 bytes***



Cache Mechanics

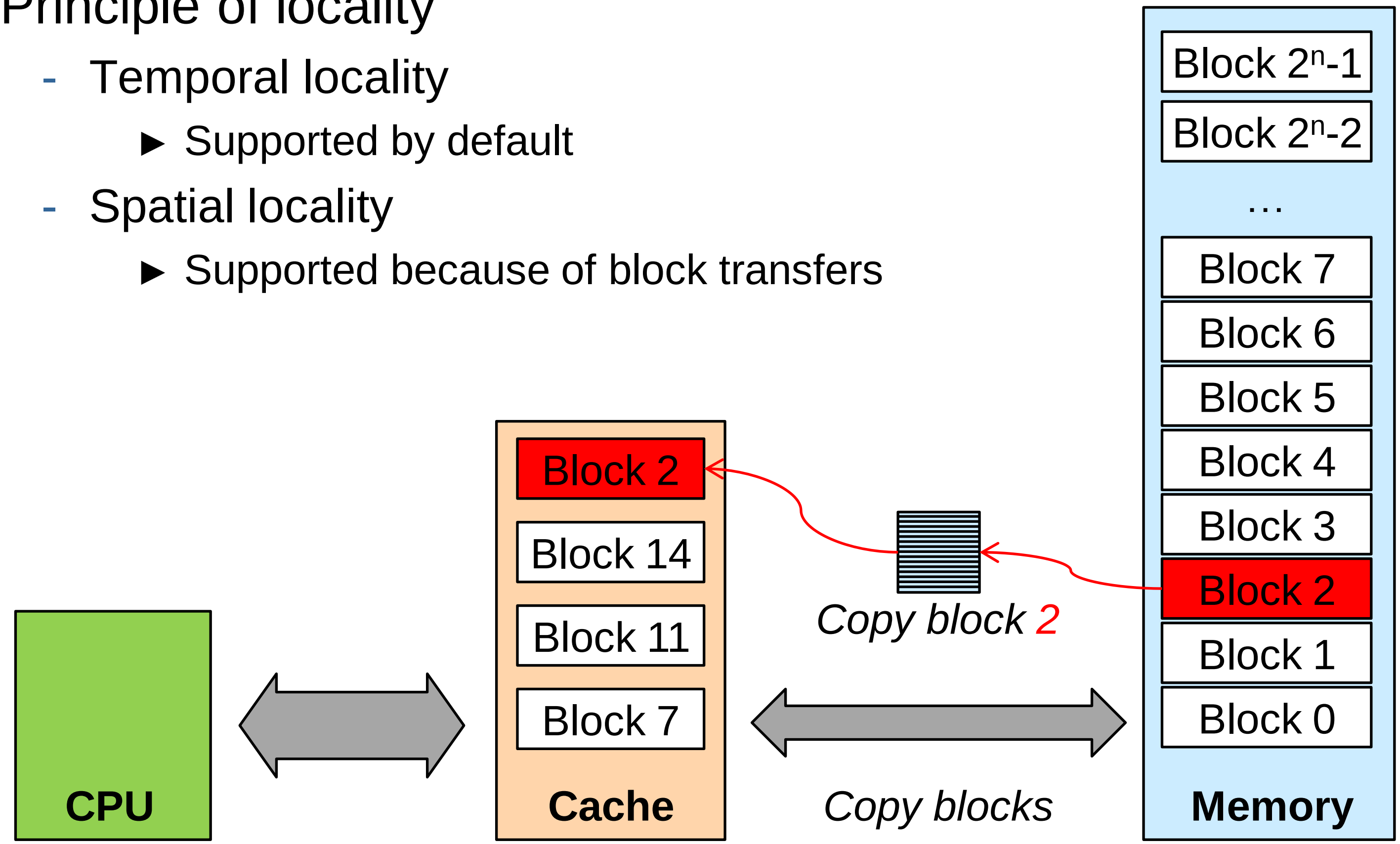
- **Memory blocks**
  - Selected blocks of main memory copied to faster cache memory



Cache Mechanics

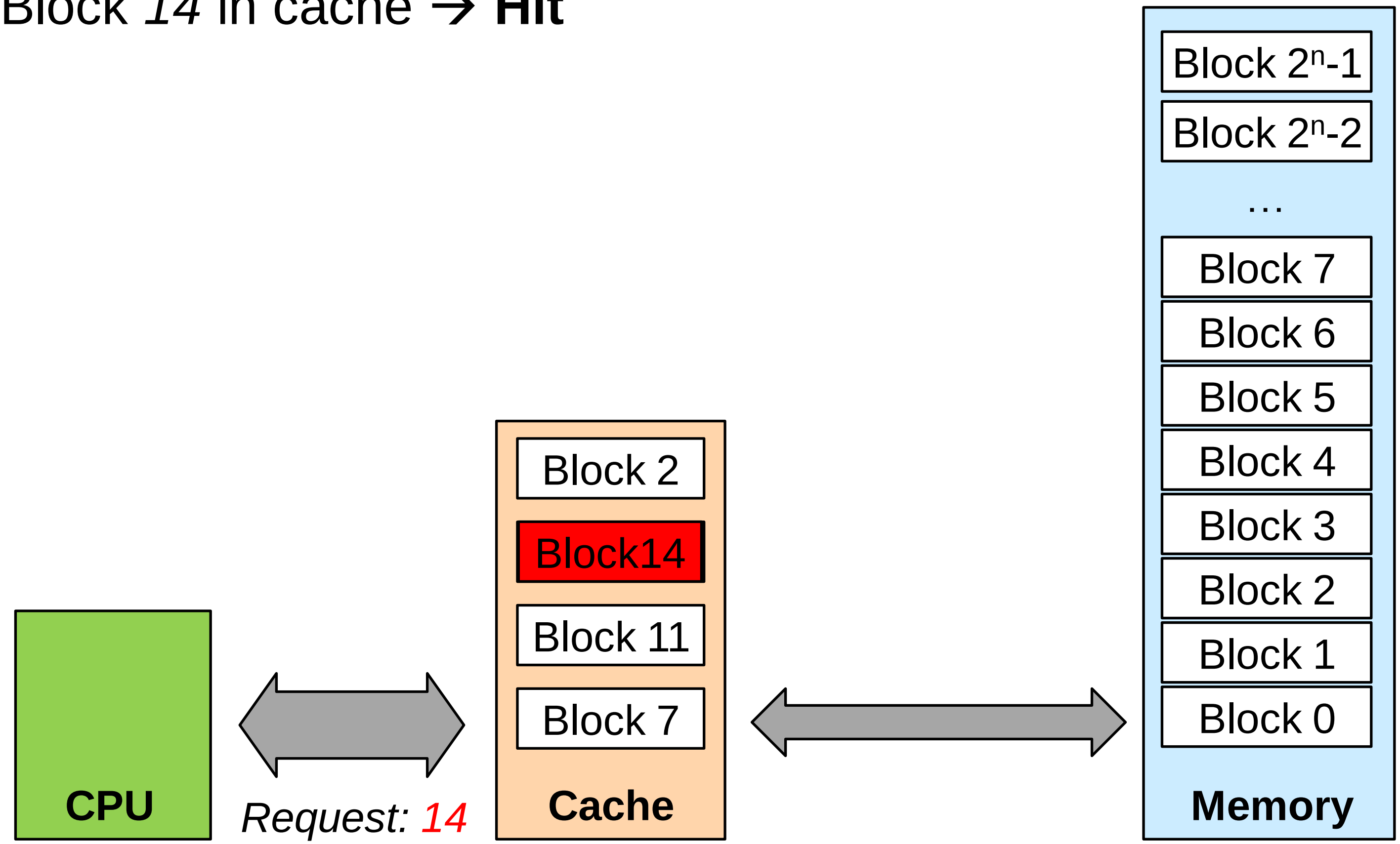
Memory blocks

- Blocks of main memory copied to faster cache memory
- Principle of locality
  - Temporal locality
    - ▶ Supported by default
  - Spatial locality
    - ▶ Supported because of block transfers



Cache Mechanics

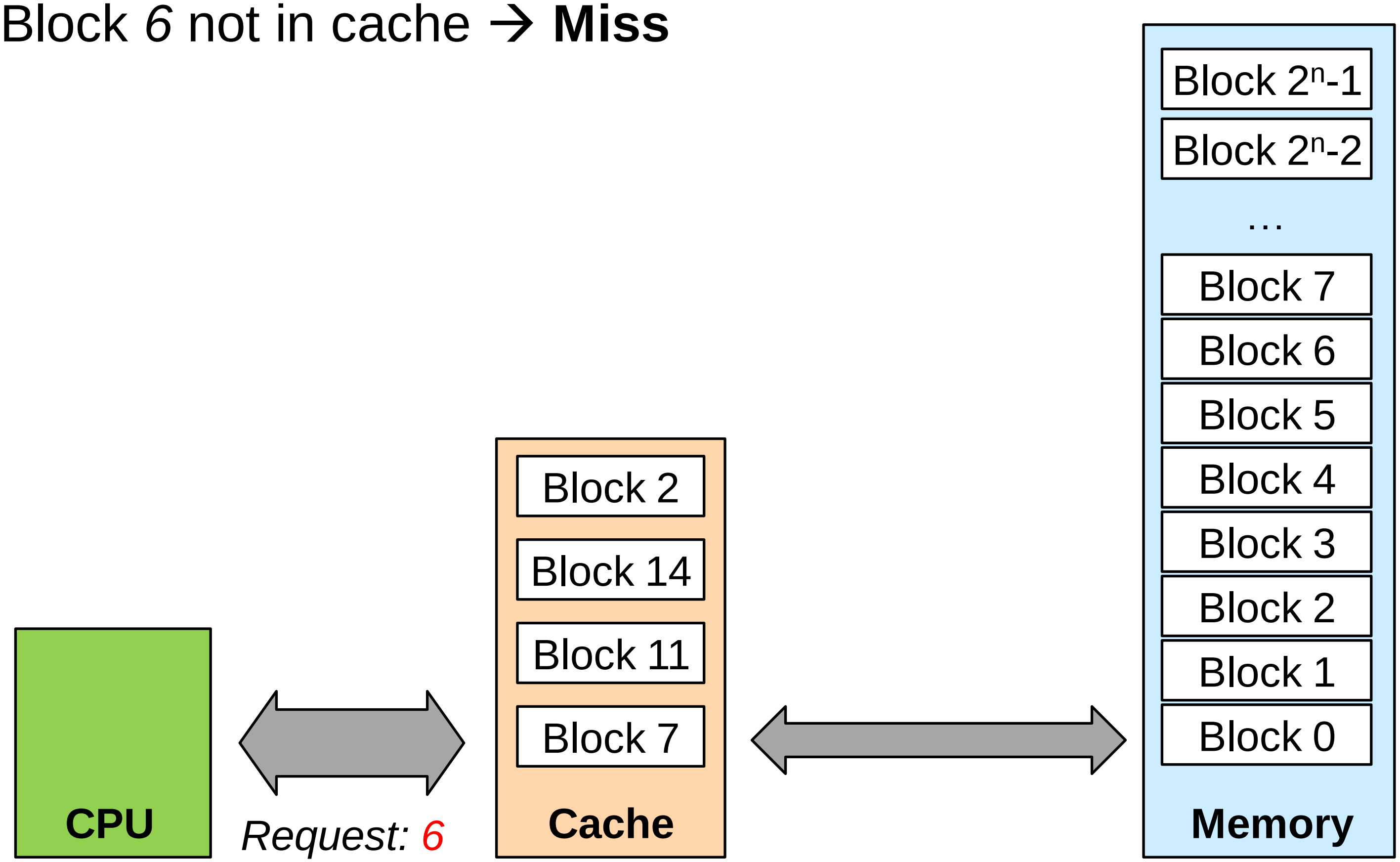
- **Cache hit**
  - Data in block *14* is needed
  - Block *14* in cache → **Hit**



Cache Mechanics

Cache miss (I)

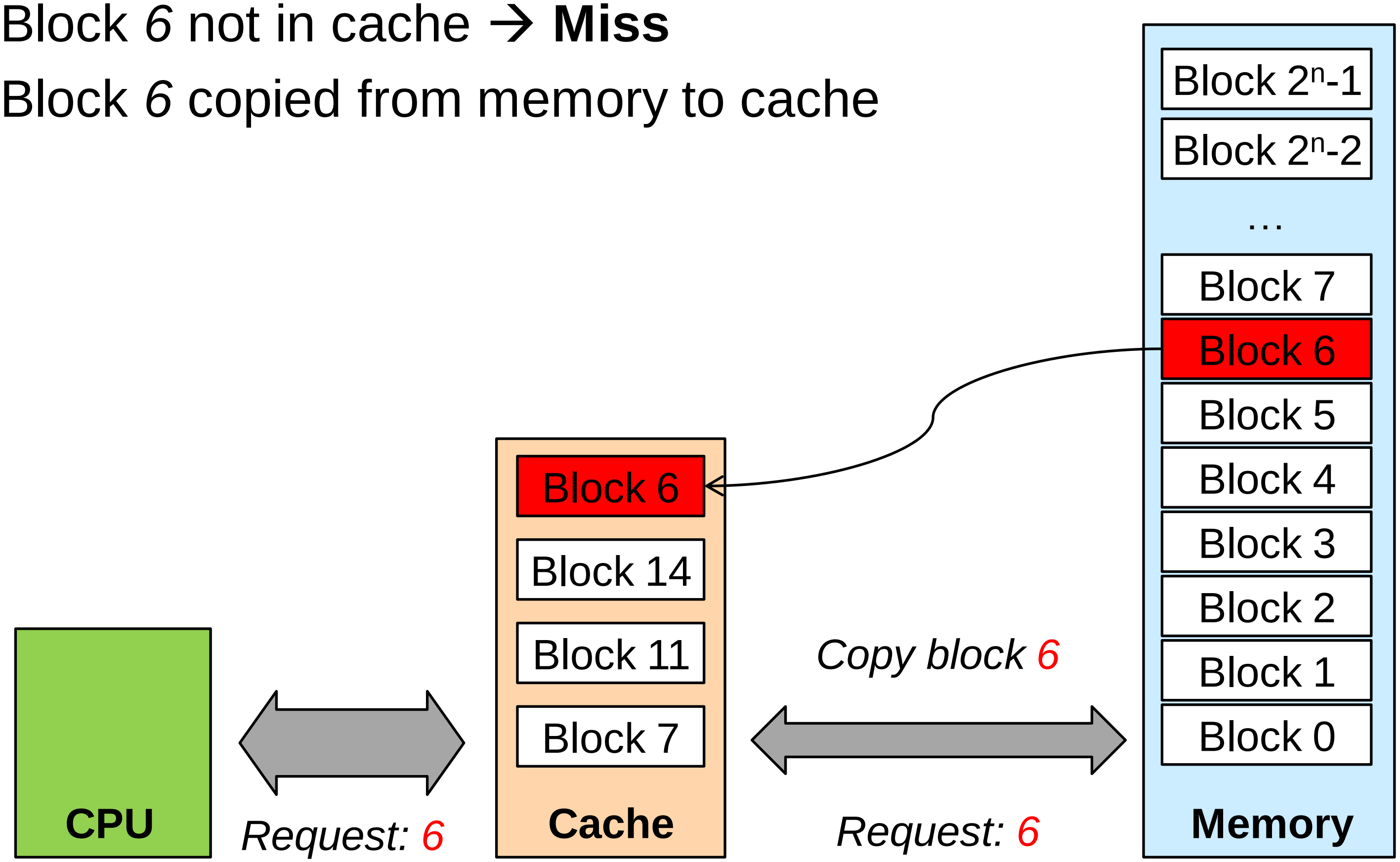
- Data in block *6* is needed
- Block *6* not in cache → **Miss**



Animation

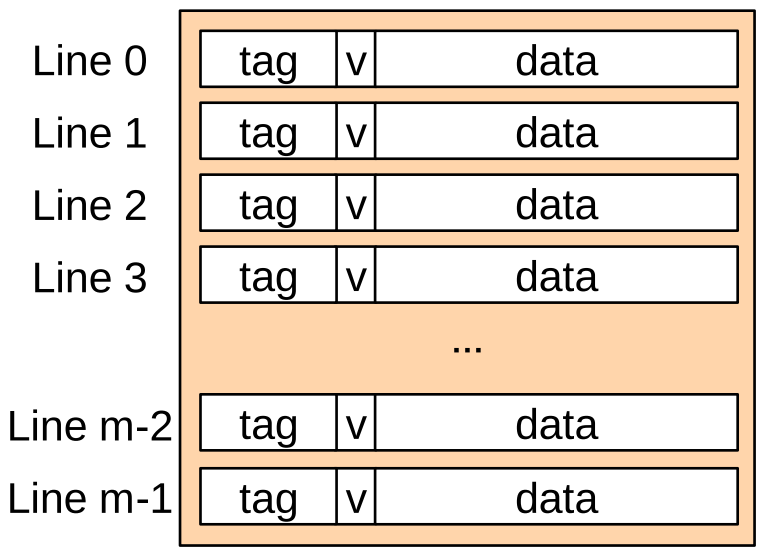
Cache Mechanics

- **Cache miss (II)**
  - Data in block *6* is needed
  - Block *6* not in cache → **Miss**
  - Block *6* copied from memory to cache

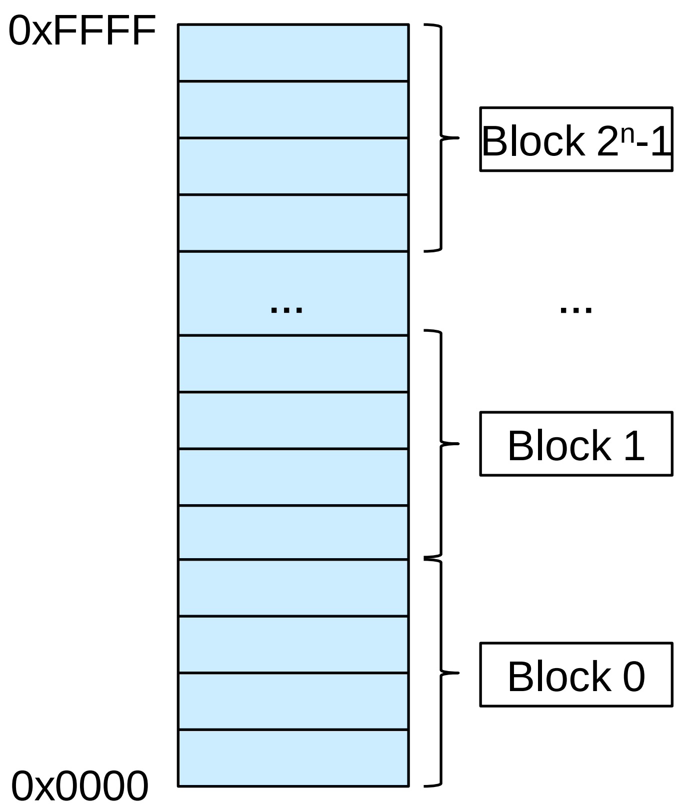


Cache Organization

- **Organized in lines**
  - Valid bit v → indicates that line contains valid data
  - Tag → unique identifier for memory location
  - Data → data of exactly one memory block
  - m = overall number of cache lines



Cache Organization



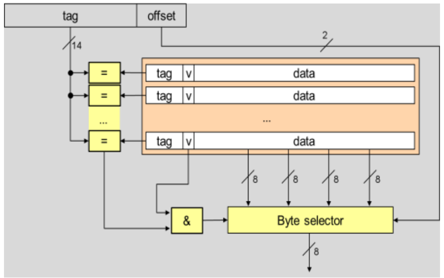
- **Addressing**

Block	Address	Block identification bits	offset
Block 2^{n-1}	0xFFFF	1111 1111 1111 11	11
	0xFFFE	1111 1111 1111 11	10
	0xFFFD	1111 1111 1111 11	01
	0xFFFC	1111 1111 1111 11	00
...			
Block 1	0x0007	0000 0000 0000 01	11
	0x0006	0000 0000 0000 01	10
	0x0005	0000 0000 0000 01	01
	0x0004	0000 0000 0000 01	00
Block 0	0x0003	0000 0000 0000 00	11
	0x0002	0000 0000 0000 00	10
	0x0001	0000 0000 0000 00	01
	0x0000	0000 0000 0000 00	00

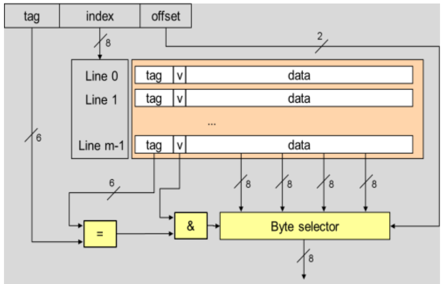
Cache Organization

■ Three different cache models

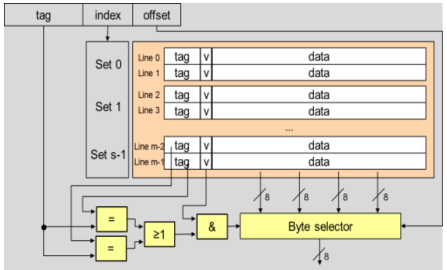
- Fully associative



- Direct mapped



- N-way set associative

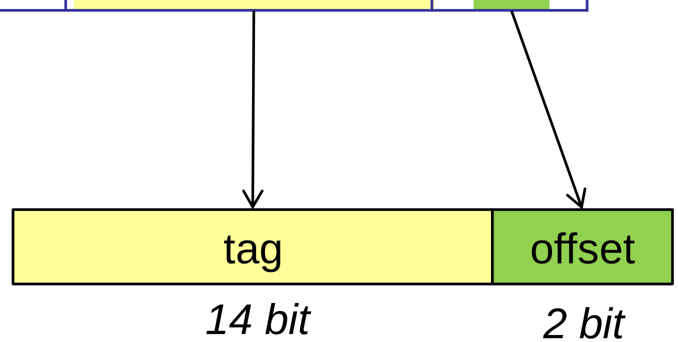


Fully Associative

- **Addressing**
  - Tag contains complete block identification

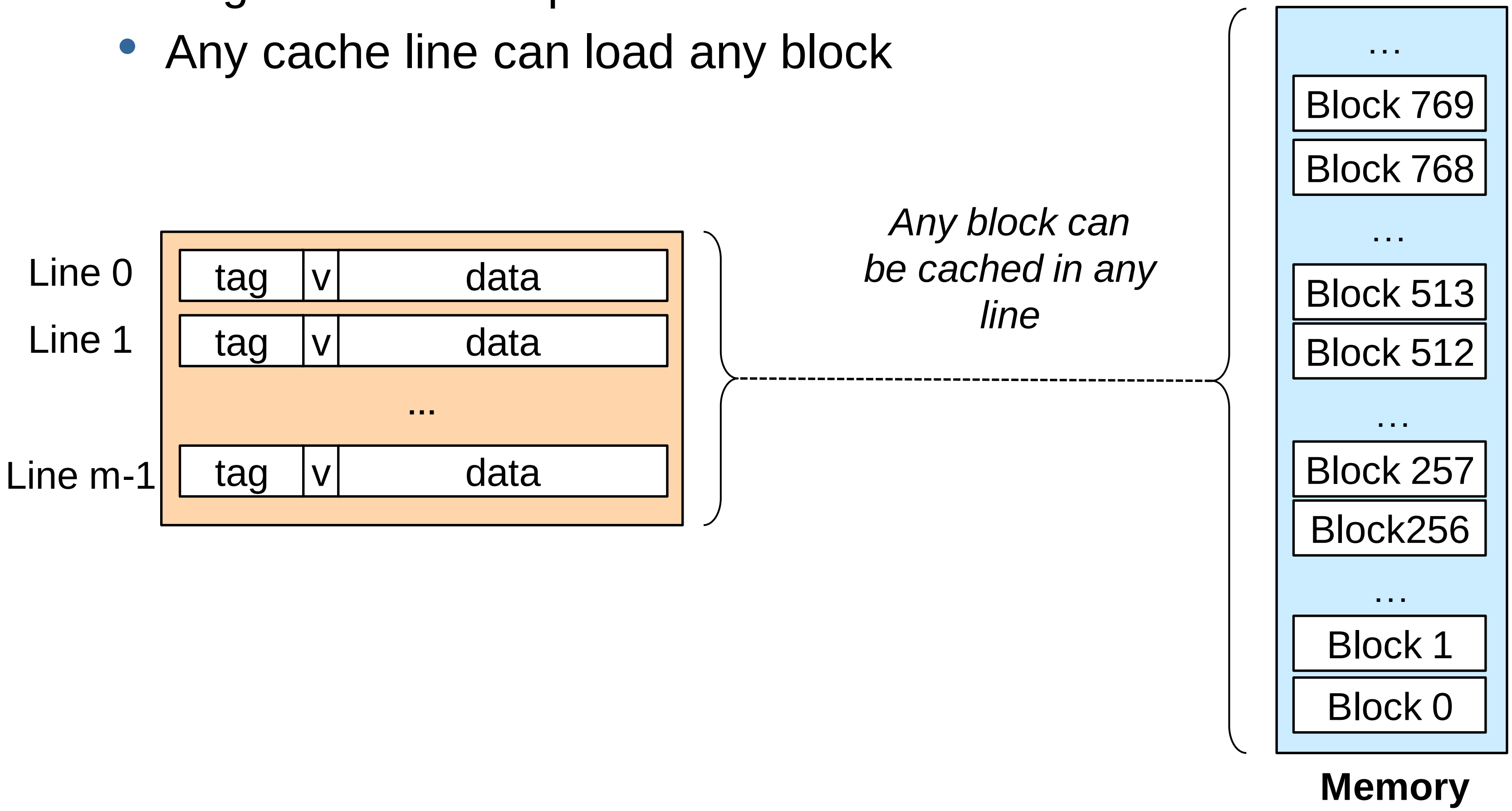
Block	Address	Block identification bits	offset
Block 1	0x0007	0000 0000 0000 01	11
	0x0006	0000 0000 0000 01	10
	0x0005	0000 0000 0000 01	01
	0x0004	0000 0000 0000 01	00

16 bit address
from processor



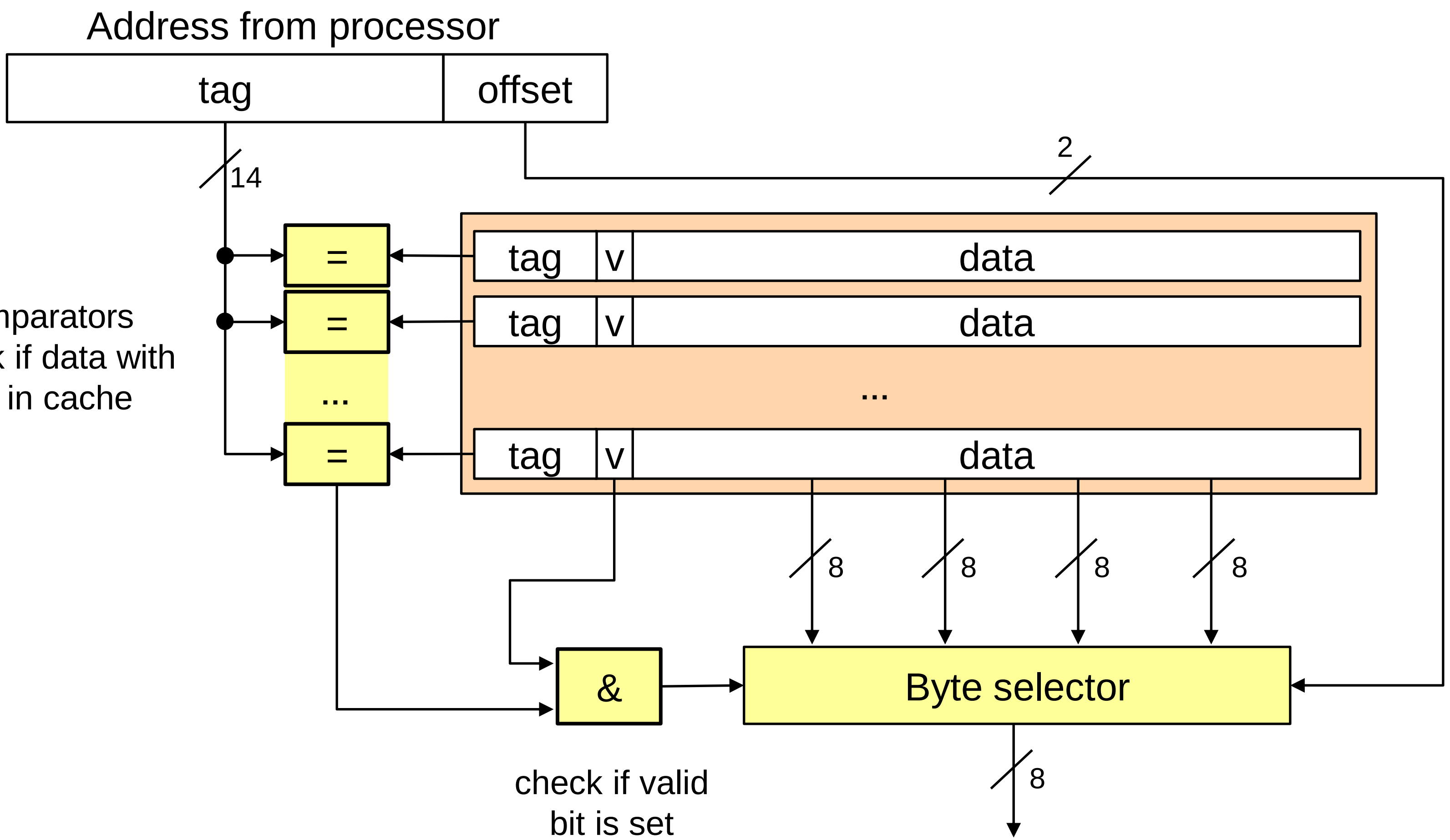
Fully Associative

- **Organization**
  - Tag contains complete block identification
  - Any cache line can load any block



Fully Associative

Architecture



n comparators check if data with tag is in cache

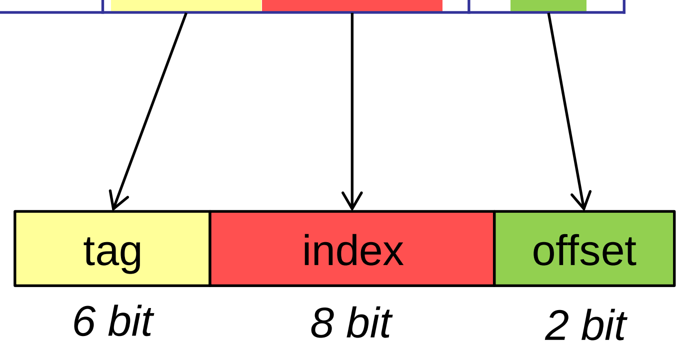
check if valid bit is set

Direct Mapped

- **Addressing**
  - Block identification spitted into tag and index

Block	Address	Block identification bits	offset
Block 1	0x0007	0000 0000 0000 01	11
	0x0006	0000 0000 0000 01	10
	0x0005	0000 0000 0000 01	01
	0x0004	0000 0000 0000 01	00

16 bit address from processor

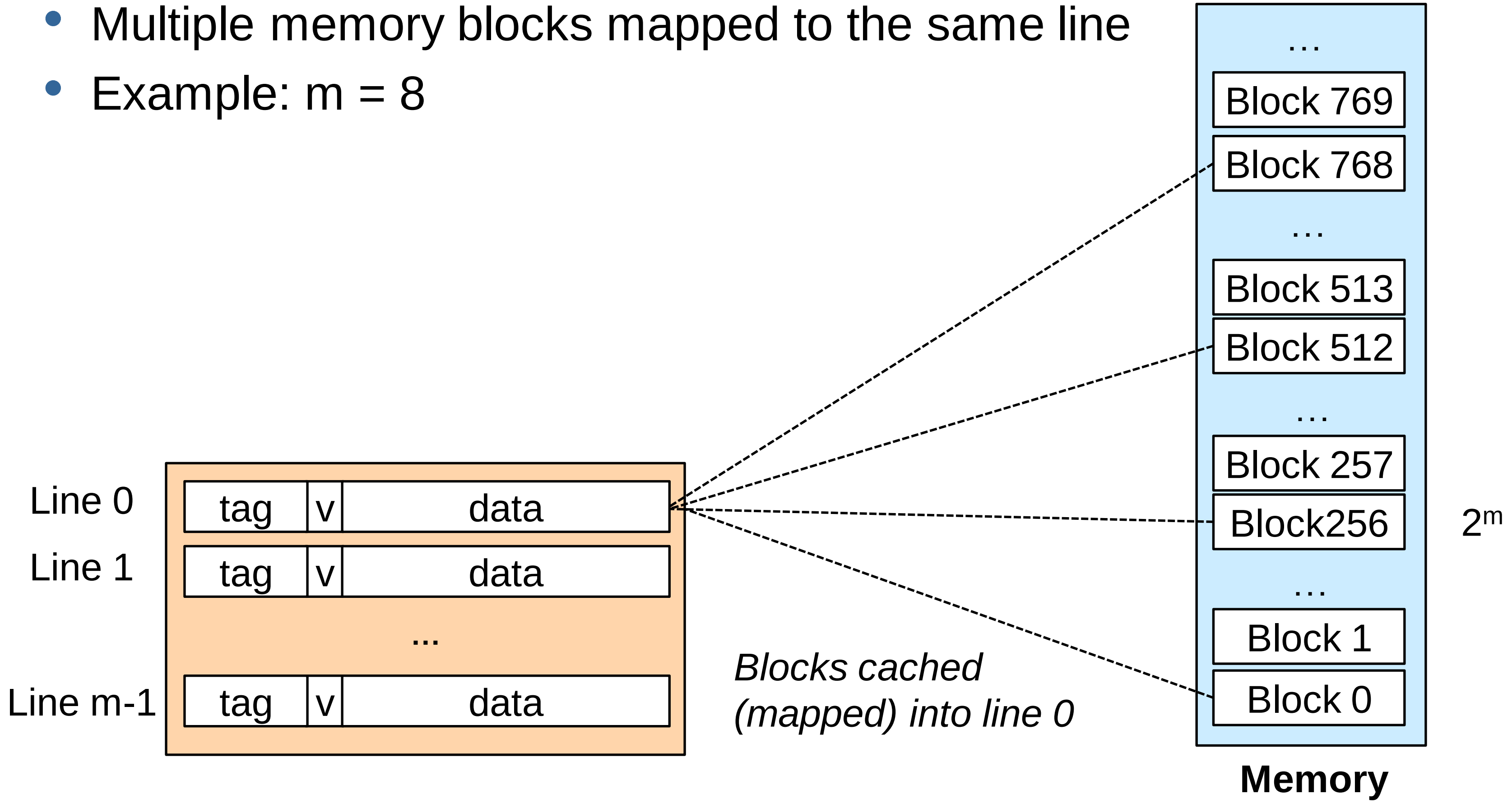


Direct Mapped

School of Engineering
InES Institute of Embedded Systems

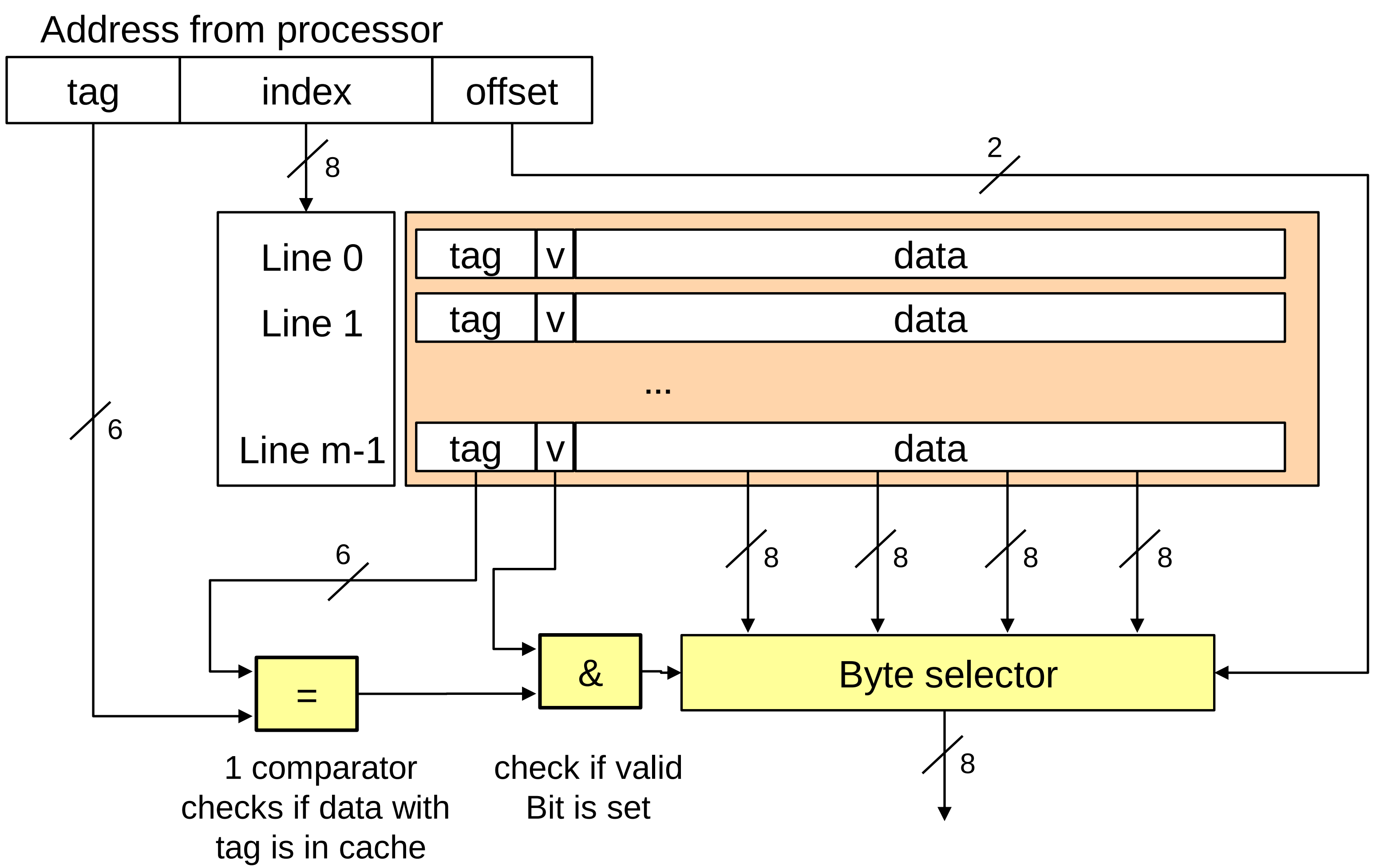
Organization

- Each memory block is mapped to exactly one cache line
- Multiple memory blocks mapped to the same line
- Example: m = 8



Direct Mapped

- **Architecture**

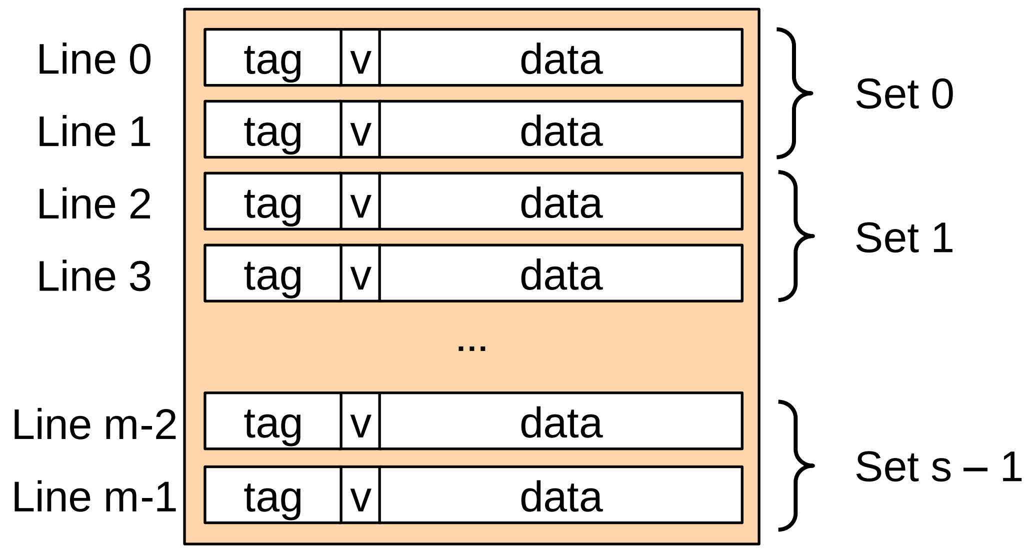


N-Way Set Associative

Organization

- Partition into sets
  - s = m/n number of sets
  - n lines per set („N-way“)
  - b bytes per line
- s x n x b data bytes

Example: n = 2

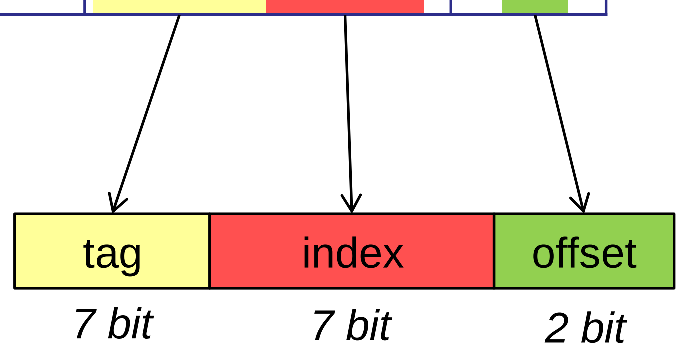


N-Way Set Associative

- **Addressing**
  - Example: n = 2
    - Maximum index corresponds to number of sets (s = m/n)

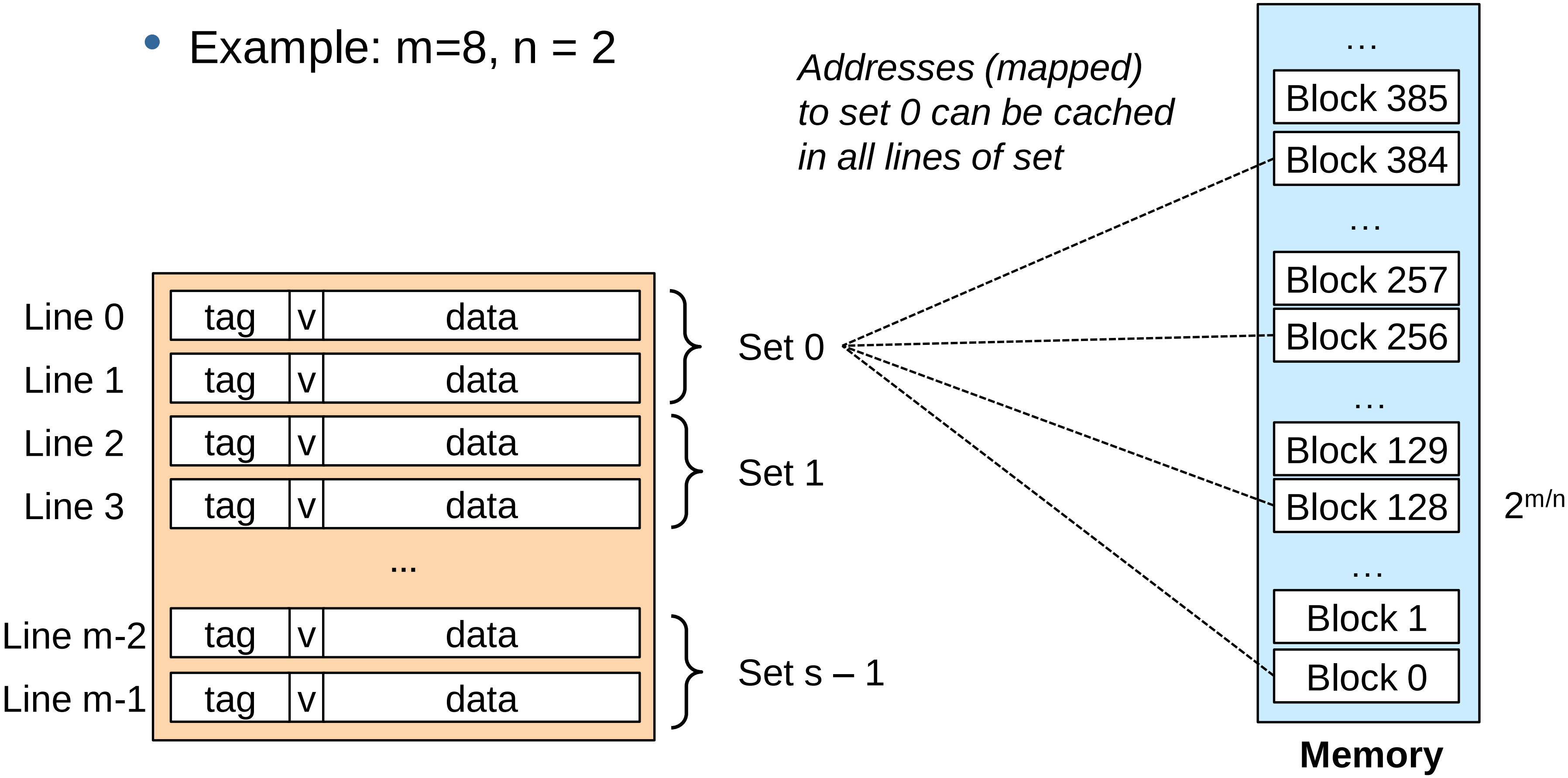
Block	Address	Block identification bits	offset
Block 1	0x0007	0000 0000 0000 01	11
	0x0006	0000 0000 0000 01	10
	0x0005	0000 0000 0000 01	01
	0x0004	0000 0000 0000 01	00

16 bit address from processor



N-Way Set Associative

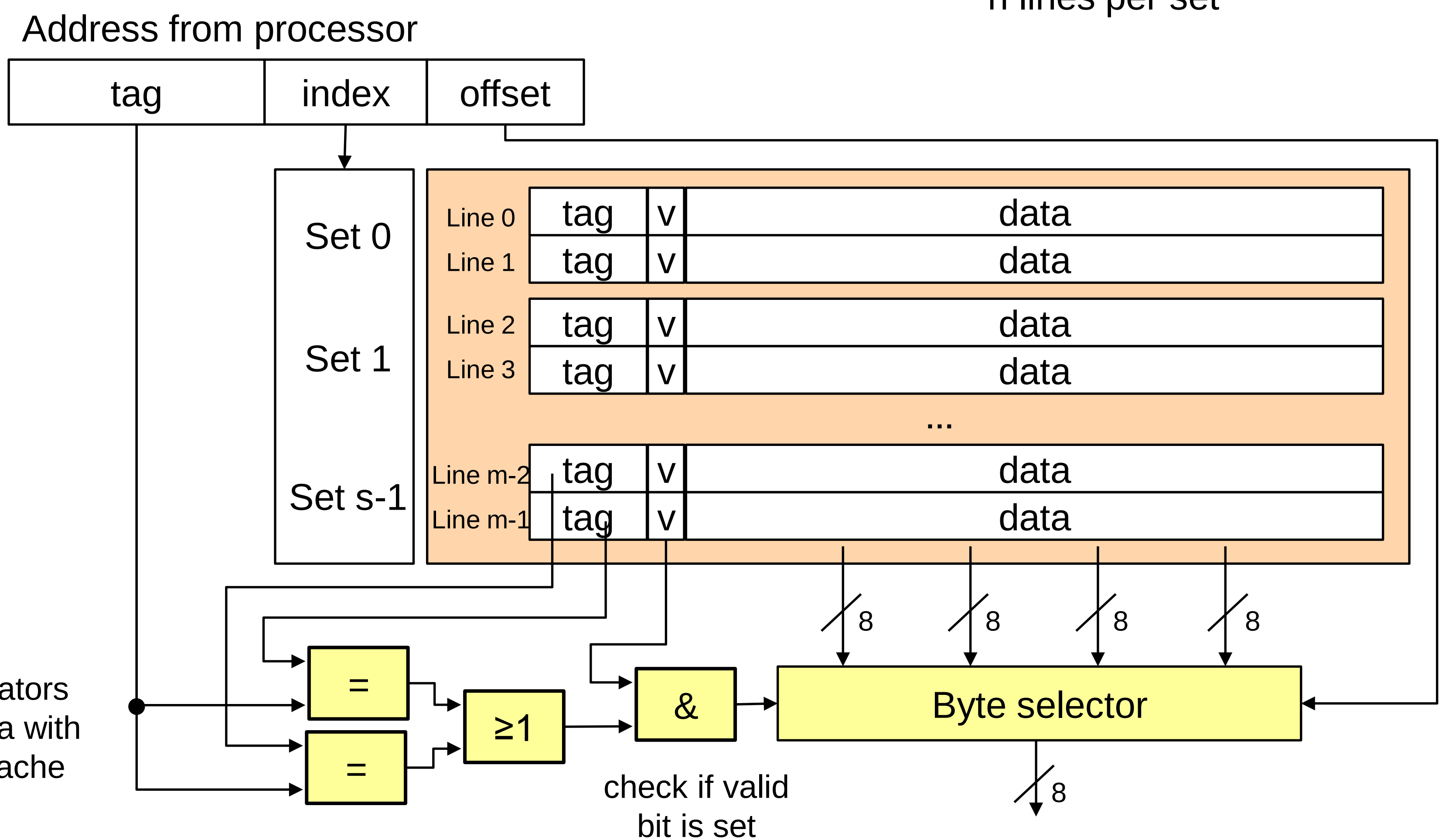
- **Organization**
  - Partition into sets
  - Example: m=8, n = 2



N-Way Set Associative

- **Architecture**

Example: n = 2
s = m/n number of sets
n lines per set



n Comparators check if data with tag is in cache

Cache Organization

School of Engineering
InES Institute of Embedded Systems

Comparison

Organization	Fully associative	Direct mapped	N-way set associative
Number of sets	1	m	m/n
Associativity	m(=n)	1	n
Advantages	• Fast, flexible • Highest hit rates • Advanced replacement strategies	• Simple logic • Replacement strategy defined by organization	Combination of both other concepts to combine advantages and to compensate disadvantages
Disadvantages	• Complex logic: one comparator per line • Requires large area on silicon • Replacement can be complex	• Lower hit rates	

Cache Misses

- **Cold miss**
  - First access to a block
- **Capacity miss**
  - Working set larger than cache
- **Conflict miss**
  - Multiple data objects map to same slot

Performance Metrics

- **Miss rate**
  - Fraction of memory references not found in cache

    misses / accesses = 1 – hit rate

- **Hit time**
  - Time to deliver a block in the cache to the processor

- **Miss penalty**
  - Additional time required to fetch data from memory because of a cache miss

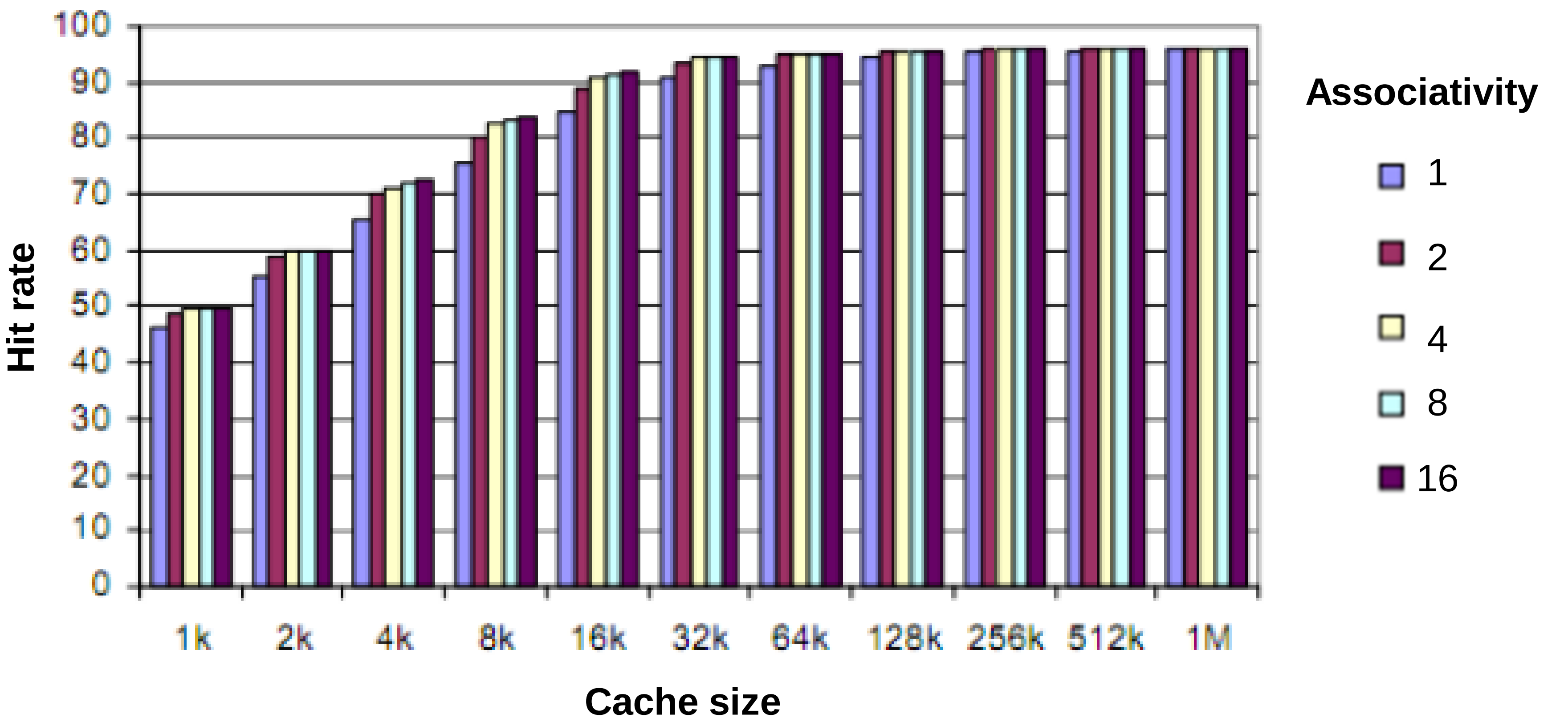
Performance Metrics

- **High cache hit rate is important!**
- **99% hit rate can be twice as fast as 97%**

- **Example:**
  - Cache hit time: 1 processor cycle
  - Miss penalty: 100 processor cycles
  - Average access time is:
    - **97%** hits: 1 cycle + 0.03 * 100 cycles = **4 cycles average**
    - **99%** hits: 1 cycle + 0.01 * 100 cycles = **2 cycles average**

Performance Metrics

- **Cache size versus hit rate**
  - Typical hit rate for Associativity 1,2,4,8,16 and cache size



http://www.extremetech.com/extreme/188776-how-l1-and-l2-cpu-caches-work-and-why-theyre-an-essential-part-of-modern-chips

Replacement Strategies

- **Selecting cache line to replace by**
  - LRU: Least recently used
  - LFU: Least frequently used
  - FIFO: First In–First Out oldest
  - Random Replace: randomly chosen

additional information needed in cache

simple, but still good performance

→ Relevance only for Fully- / N-way associative cache
→ Hard coded in cache implementation

Write Strategies

- **What to do on a write hit*?**
  - Write-through
    - Write immediately to memory
  - Write-back
    - Delay write to memory until replacement of line (needs a valid bit)

- **What to do on a write miss**?**
  - Write-allocate
    - Load into cache and update line in cache
  - No-write-allocate
    - Writes immediately to memory

* *Write hit: data already in cache*

** *Write miss: data not in cache*

Animation

The Programmers Perspective

School of Engineering
InES Institute of Embedded Systems

- **For loops over multi-dimensional arrays**
  - Example: matrices (2-dim arrays)
- **Change order of iteration to match layout**
  - Gets better spatial locality
  - Layout in C: **last index changes first!**

```
for(j = 0; j < 10000; j++){
  for(i = 0; i < 40000; i++){
    c[i][j]=a[i][j]+b[i][j];
  }
}
// a[i][j] and a[i+1][j]
// are 10'000 elements apart
```

```
for(i = 0;i < 40000; i++){
  for(j = 0;j < 10000; j++){
    c[i][j]=a[i][j]+b[i][j];
  }
}
// a[i][j] and a[i][j+1]
// are next to each other
```

The Programmers Perspective

School of Engineering
InES Institute of Embedded Systems

- **Change order of iteration to match layout**

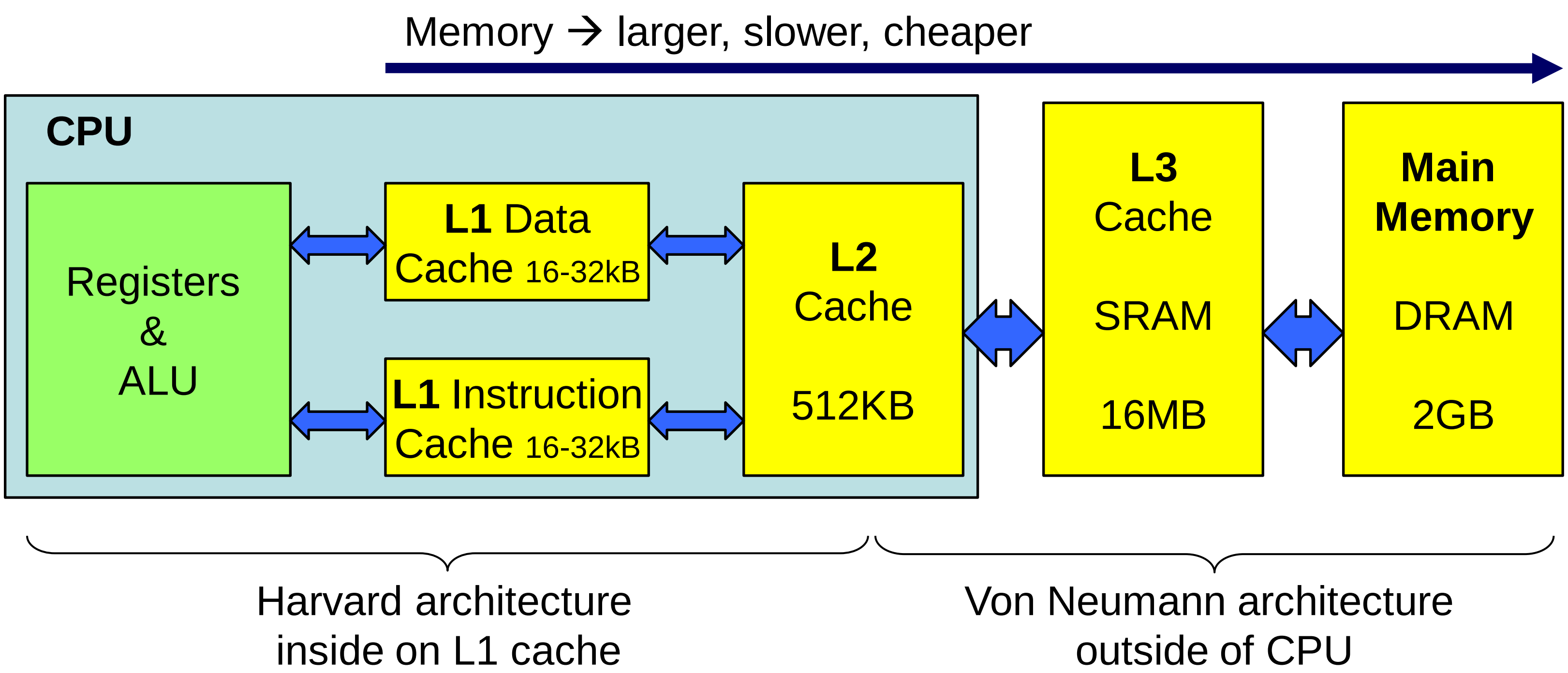
```c
#include<stdio.h>

#include<time.h>

int main( int argc, char** argv )
{
  int iterations = 1000;
  int size = 1024;

  int array[size][size];
  int v = 0;

  if (argc==2) {
  printf("j is inner\n");
  for (int n=0; n<iterations; n++) {
    for (int i=0; i<size; i++) {
      for (int j=0; j<size; j++) {
        array[i][j] = v;
        v++;
      }
    }
  }

  } else {
  printf("j is outer\n");
  for (int n=0; n<iterations; n++) {
    for (int j=0; j<size; j++) {
      for (int i=0; i<size; i++) {
        array[i][j] = v;
        v++;
      }
    }
  }
  }

  printf("%i-%i done\n", array[0][0], array[size-1][size-1]);

  return 0;
}
```

```
j is inner
1047527424-1048575999 done

real    0m3.460s
user    0m3.452s
sys     0m0.000s


j is outer
1047527424-1048575999 done

real    0m18.509s
user    0m18.472s
sys     0m0.000s
```

Advanced Cache Architecture

Cache Levels

- Typical Cache Architecture
- Example: ARM Cortex M3/M4 with external memory



Conclusion

- **Cache**
  - Cache allows fast data access to a certain part of the main memory which is mirrored in a cache
  - Spatial and temporal locality make cache principle
  - Different cache models were discussed
    - Fully associative
    - Direct mapped
    - N-way associative
  - Replacement / Write strategies
  - Reducing cache misses by optimizing memory access
  - Example for advanced cache architecture