Caching

COMP375 Computer Architecture and Organization

"Most good programmers do programming not because they expect to get paid or get adulation by the public, but because it is fun to program."

-Linus Torvalds

Exam in two weeks

- The second exam in COMP375 will be on Wednesday or Friday, October 21 or 23
- It covers all the material since the first exam
  - Interrupts
  - Busses
  - Memory
  - VLSI
  - Cache

Goals

- Understand and be able to calculate the advantages of cache memory
- Understand the design decisions involved in structuring a cache system

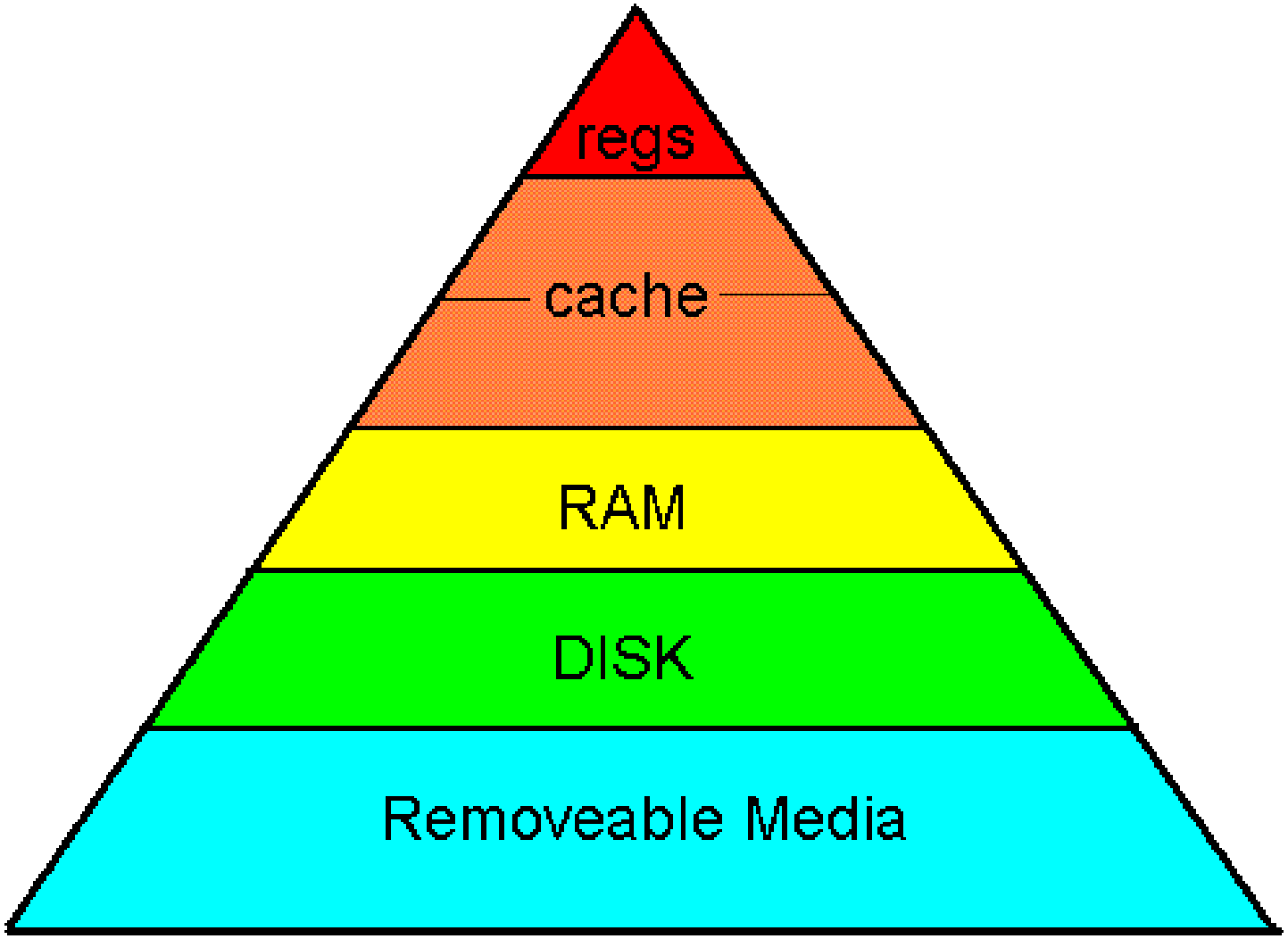
Locality of Reference

- Temporal Locality
  - A memory location that is referenced is likely to be accessed again in the near future
- Spatial Locality
  - Memory locations near the last access are likely to be accessed in the near future

Widespread Use

- Caching is the idea of keeping a copy of information that was time consuming to get in the first place
- Computers using a network will save network addresses that they might need
- The operating system keeps blocks of information from the disk in RAM
- Information in RAM is copied to faster cache memory

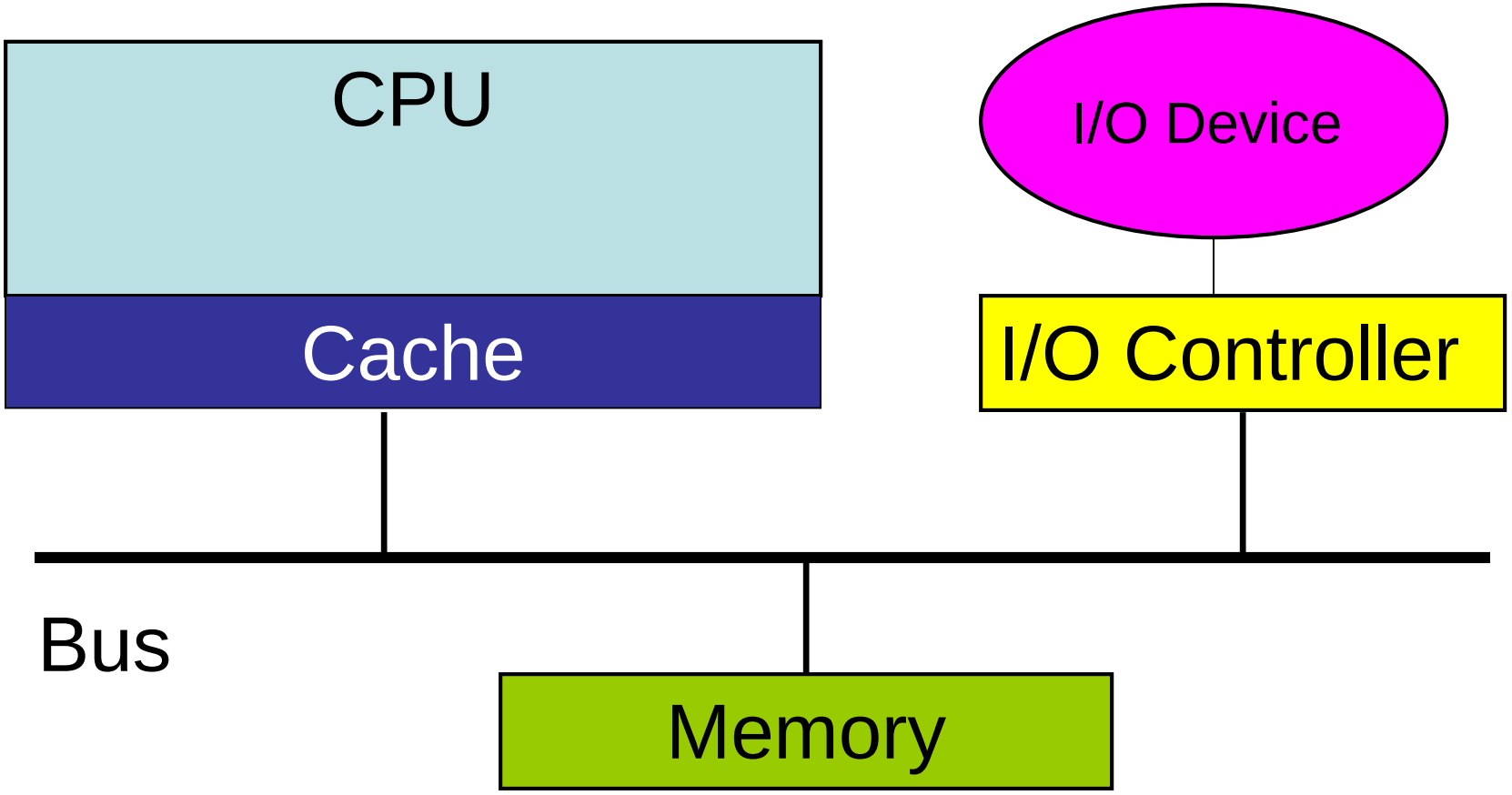
Memory Hierarchy



Cache Basics

- The processor cache is a high speed memory that keeps a copy of the frequently used data
- When the CPU wants a data value from memory, it first looks in the cache
- If the data is in the cache, it uses that data
- If the data is not in the cache, it copies a **line** of data from RAM to the cache and gives the CPU what it wants

Basic Computer Components



Cache Line

- The CPU usually accesses a **word** of memory at a time. A word (4 bytes) will hold an integer, float or instruction
- When memory is copied from the RAM to the cache, it usually copies a **line** of data
- A line is usually much larger than a word, often 4 to 16 words (16 to 64 bytes) in size
- A cache line might also be called a block

How Many Lines?

- You can determine the number of lines in the cache by dividing the cache size by the size of the line

$$lines = \frac{cachesize(bytes)}{linesize(bytes/line)}$$

Addresses in a Line

- A line of cache holds several sequential addresses
- If you have a 32 byte cache line, then any group of 32 sequential bytes whose upper bits are the same, will be in the same cache line

Binary Addresses

- If you group addresses by the upper n bits, you will have 2^n groups
- The lower bits of each group repeat in the same way

Separating Address Into Parts

- If you have a system with 16 bit addresses and 32 byte cache lines, how many of the upper address bits are the same for every byte in the cache?
- $Log_2(32) = 5$ It takes 5 bits to address each byte in a 32 byte line

- There are 16 – 5 = 11 upper address bits

If you have a system with 32 bit addresses and 16 byte cache lines, how many of the upper address bits are the same for every byte in a cache line?

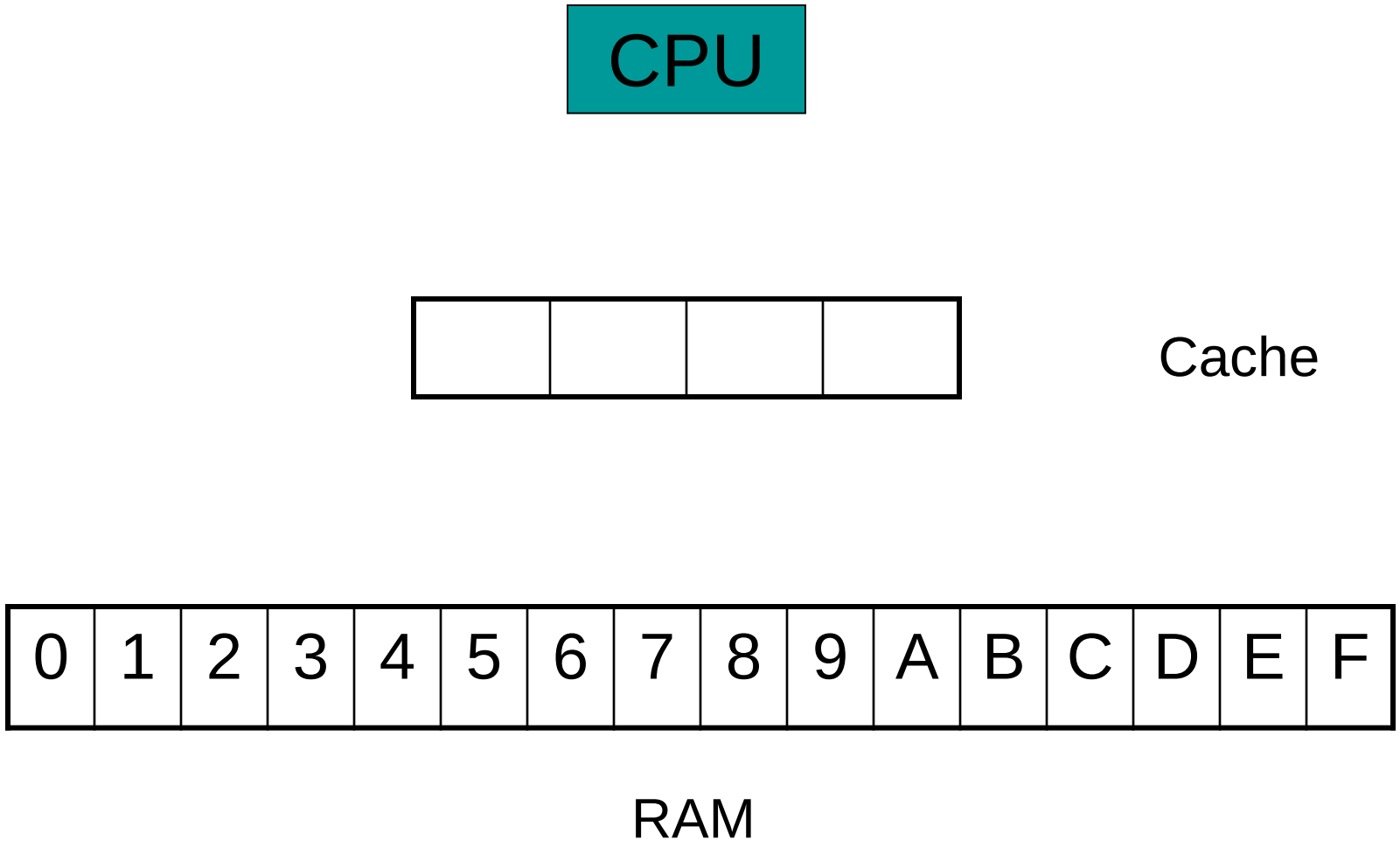
A. 1
B. 4
C. 16
D. 28
E. 63

Hit or Miss

- When the processor wants information at a given address and that information is already in the cache, this is called a **cache hit**
- If the processor wants information that is not already in the cache, this is called a **cache miss**

Caching

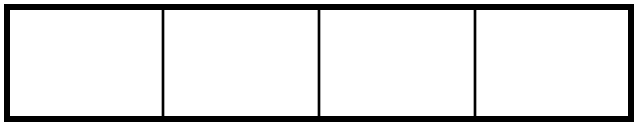
assume 2 word lines



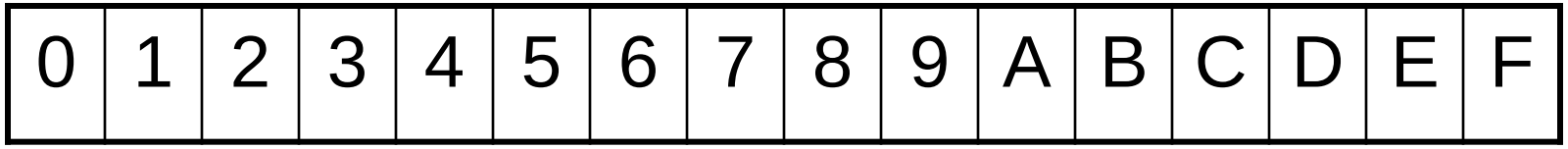
Cache Miss

CPU

Read request address 2

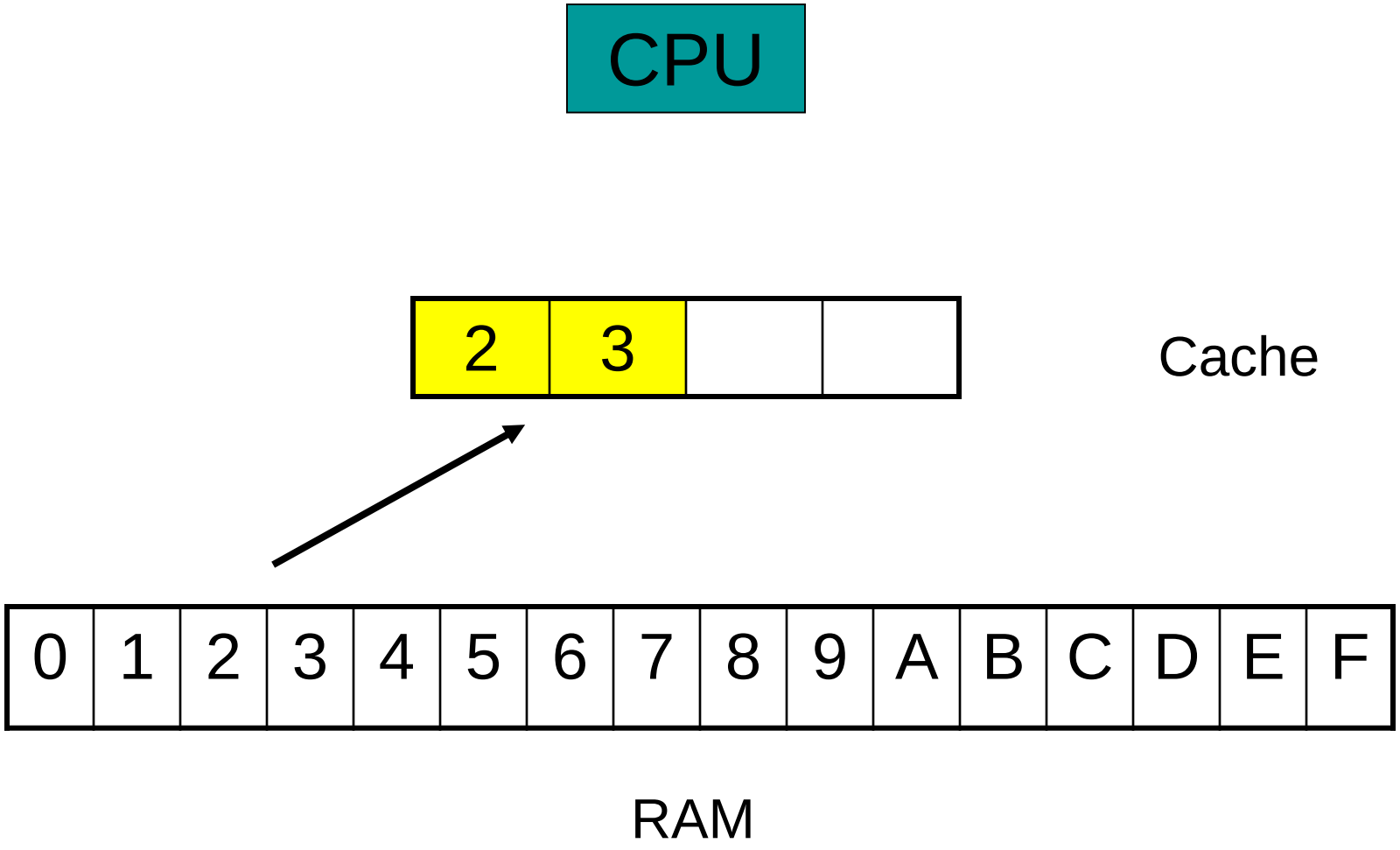


Cache

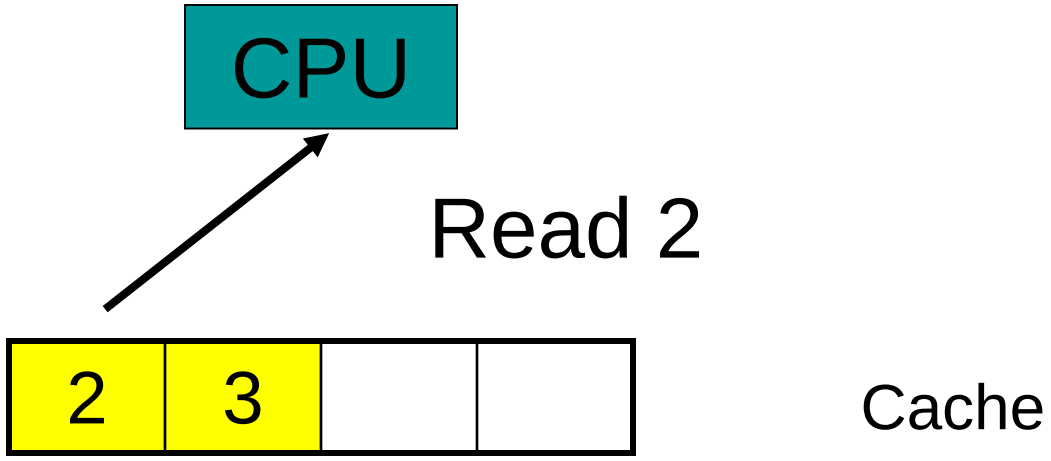


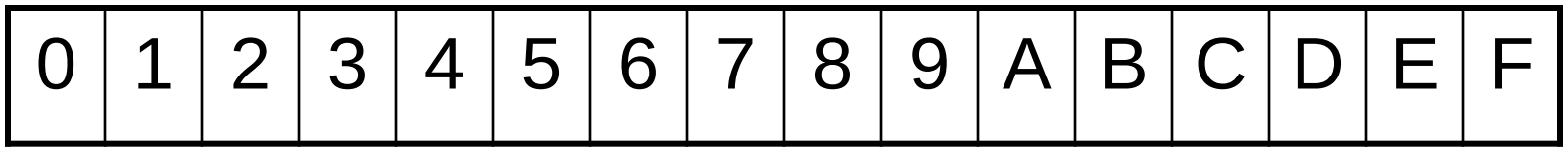
RAM

Line copied from RAM to cache



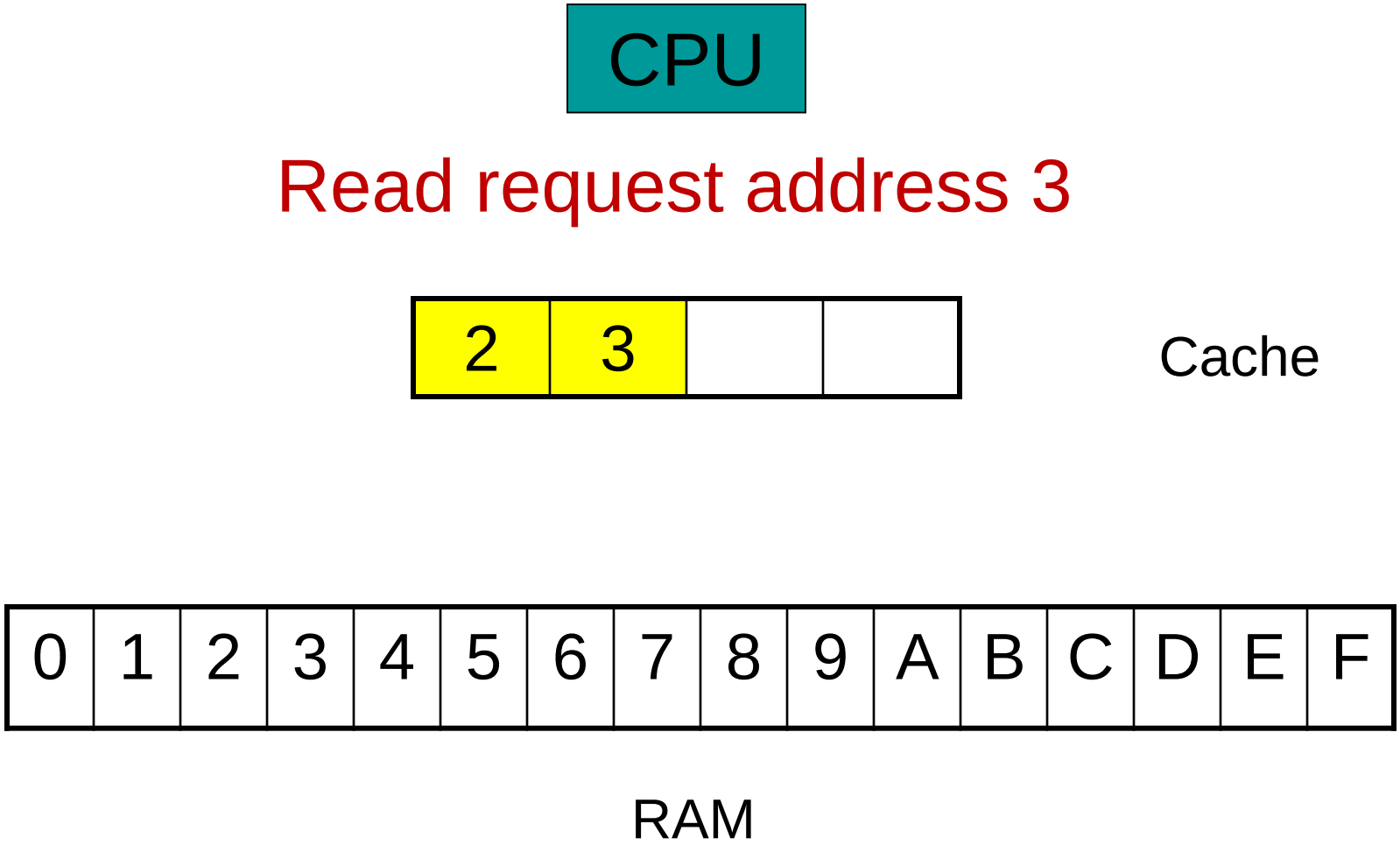
Data copied from cache to CPU



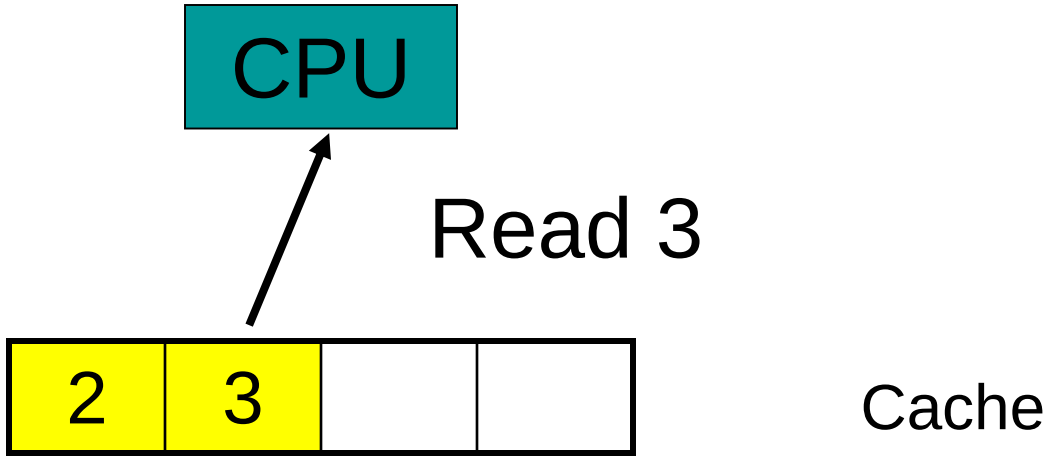


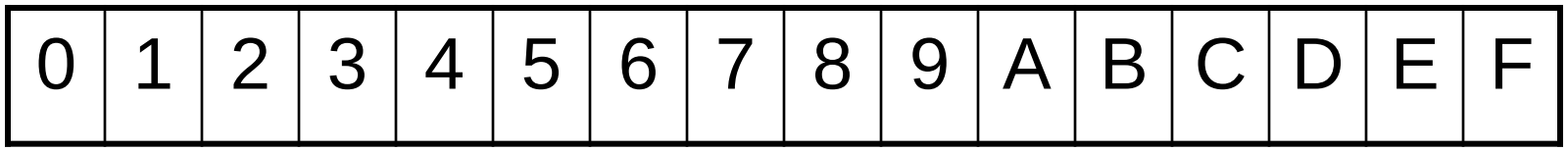
RAM

Cache Hit



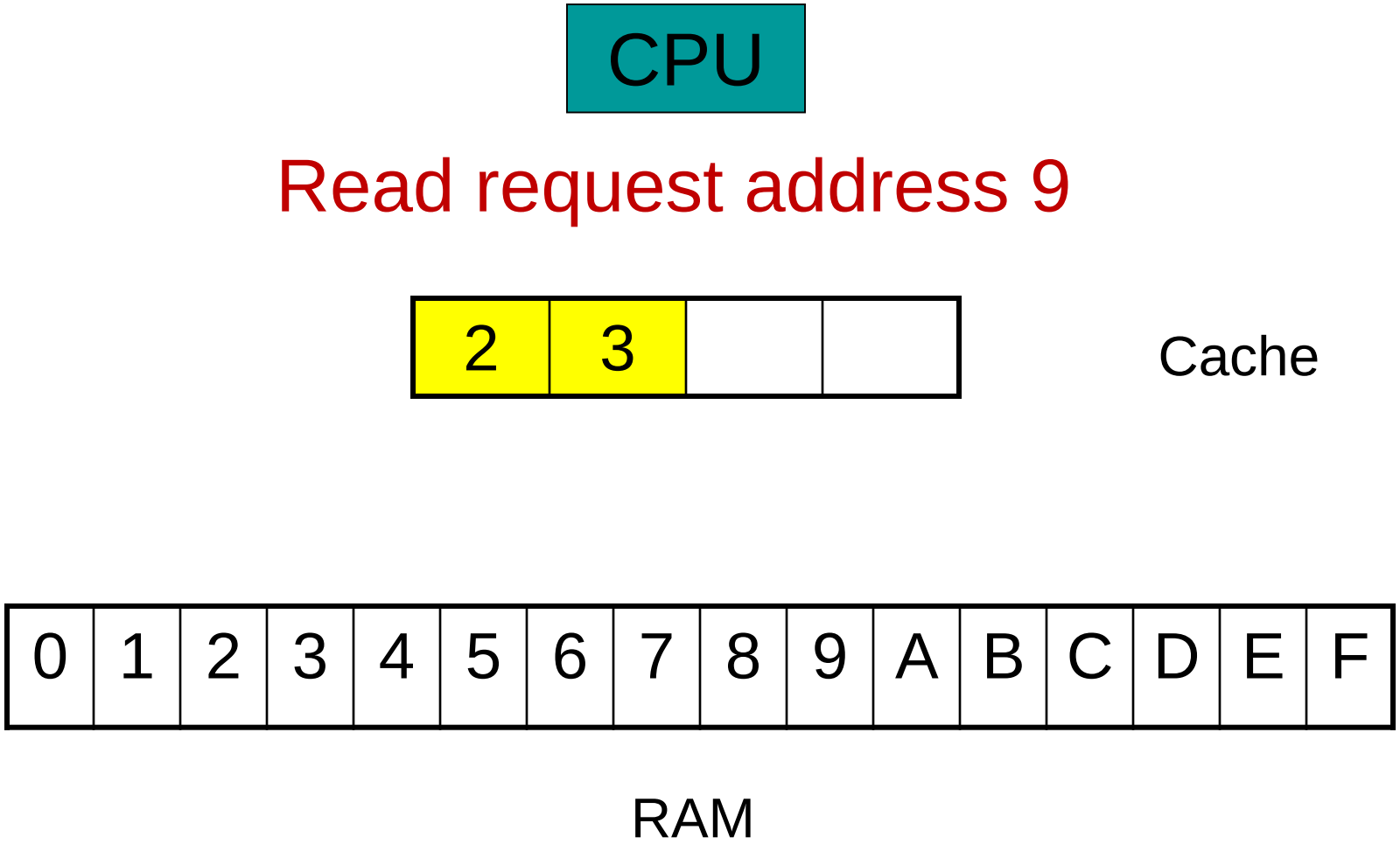
Data copied from cache to CPU



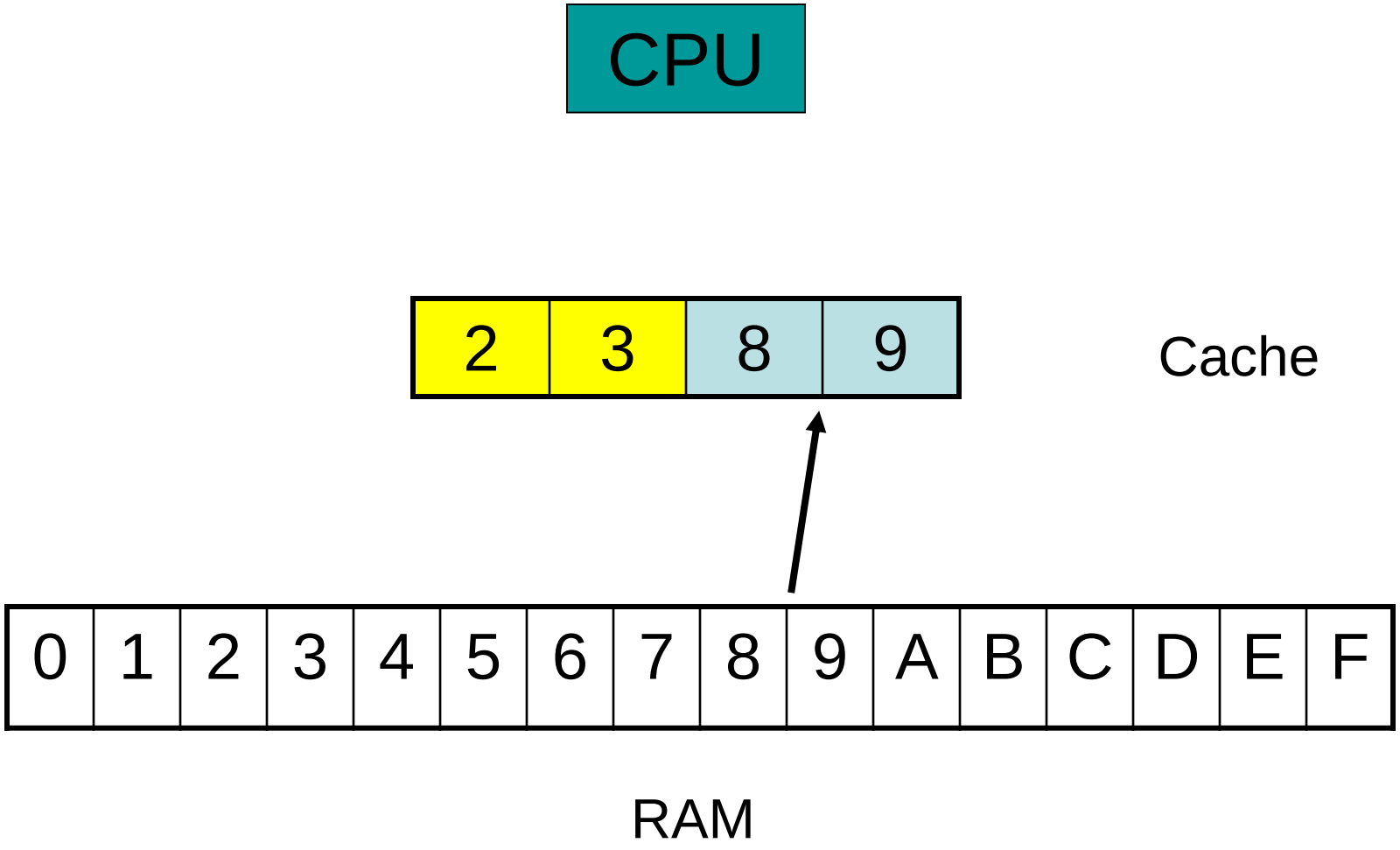


RAM

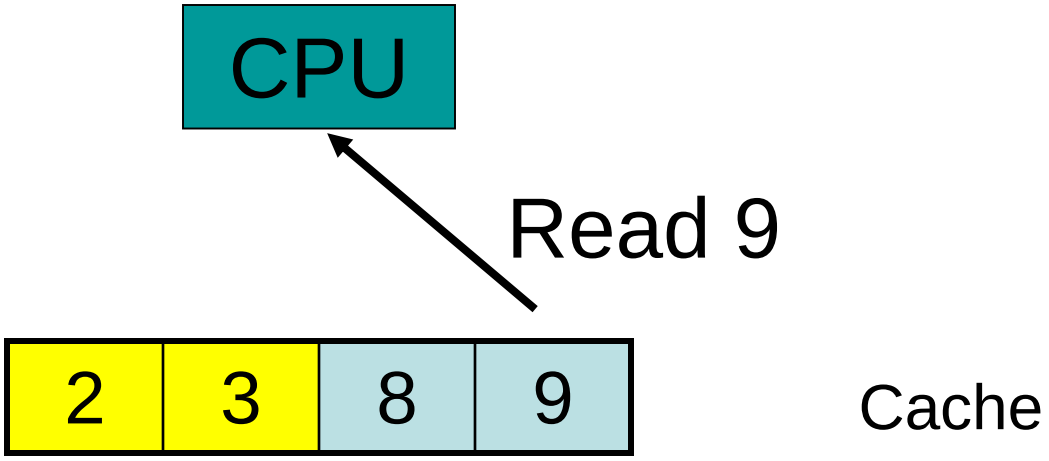
Cache Miss

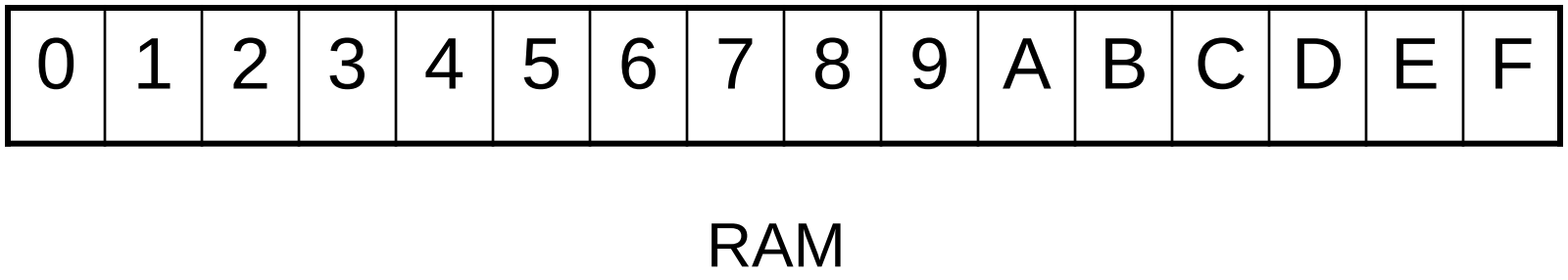


Line copied from RAM to cache

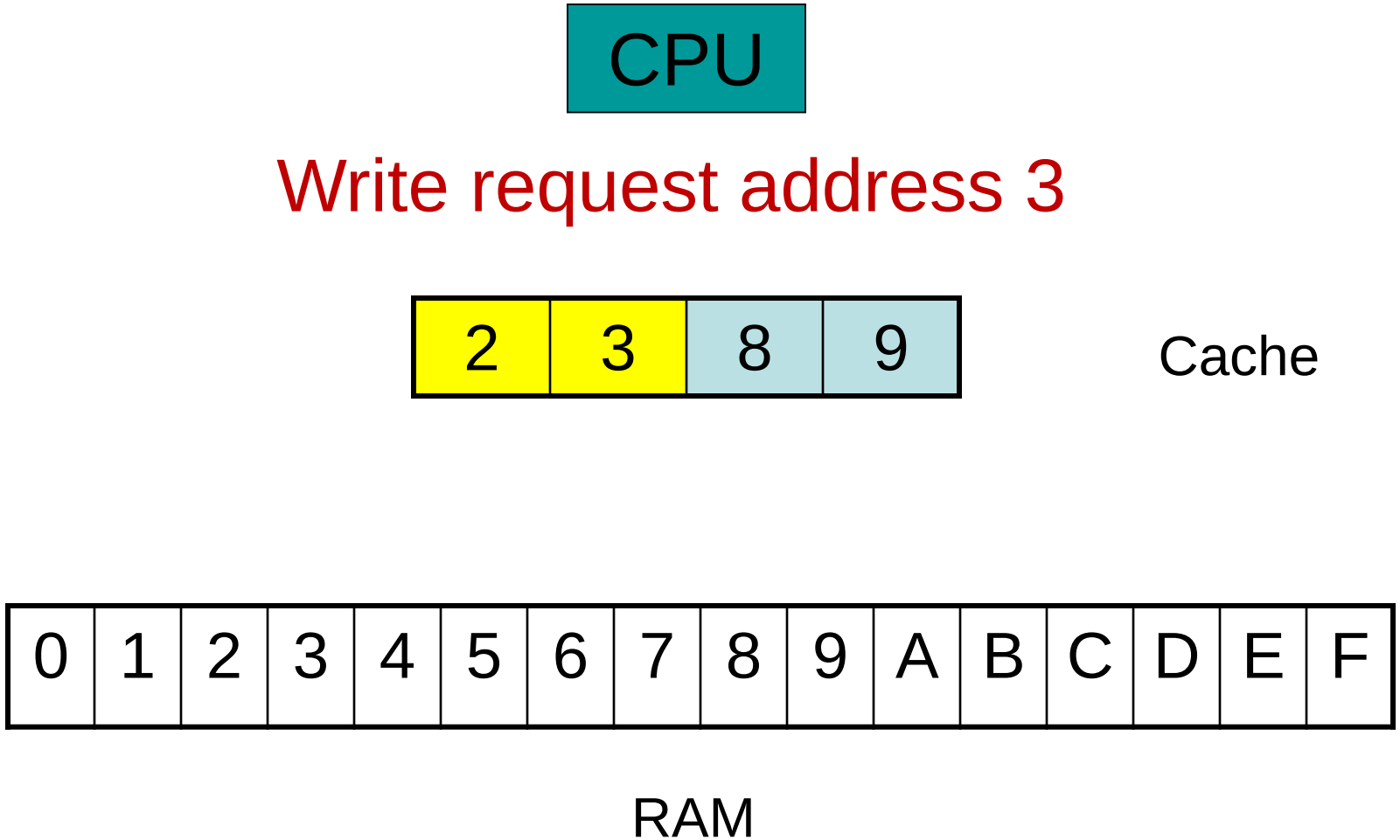


Data copied from cache to CPU

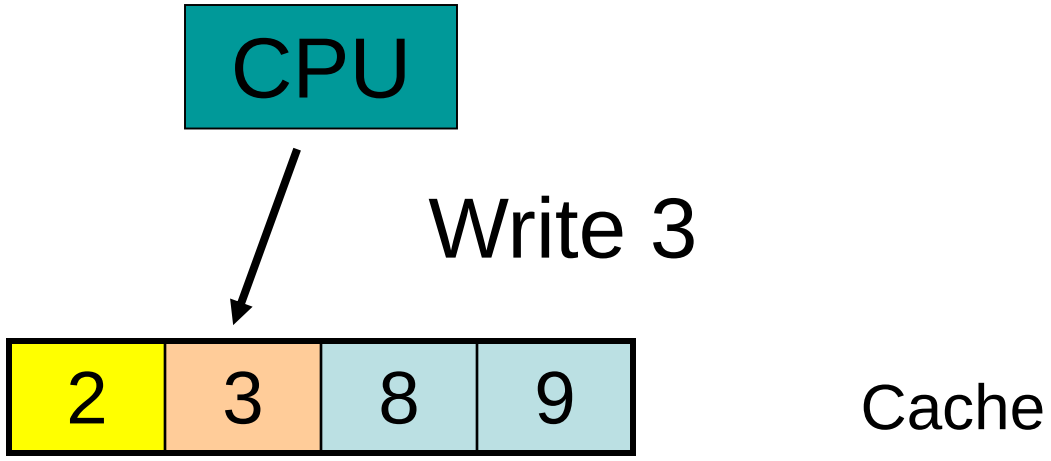


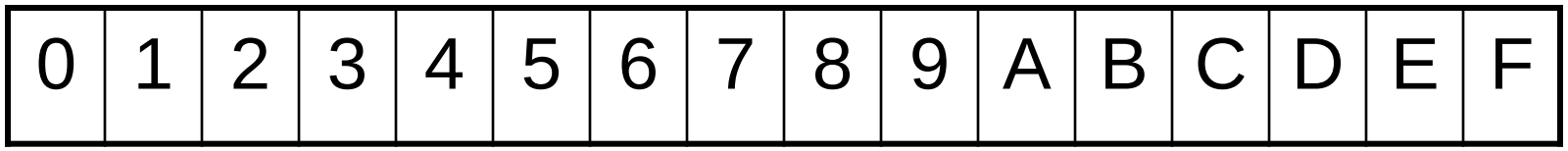


Cache Hit


CPU
Write request address 3
2
3
8
9
Cache
0
1
2
3
4
5
6
7
8
9
A
B
C
D
E
F
RAM

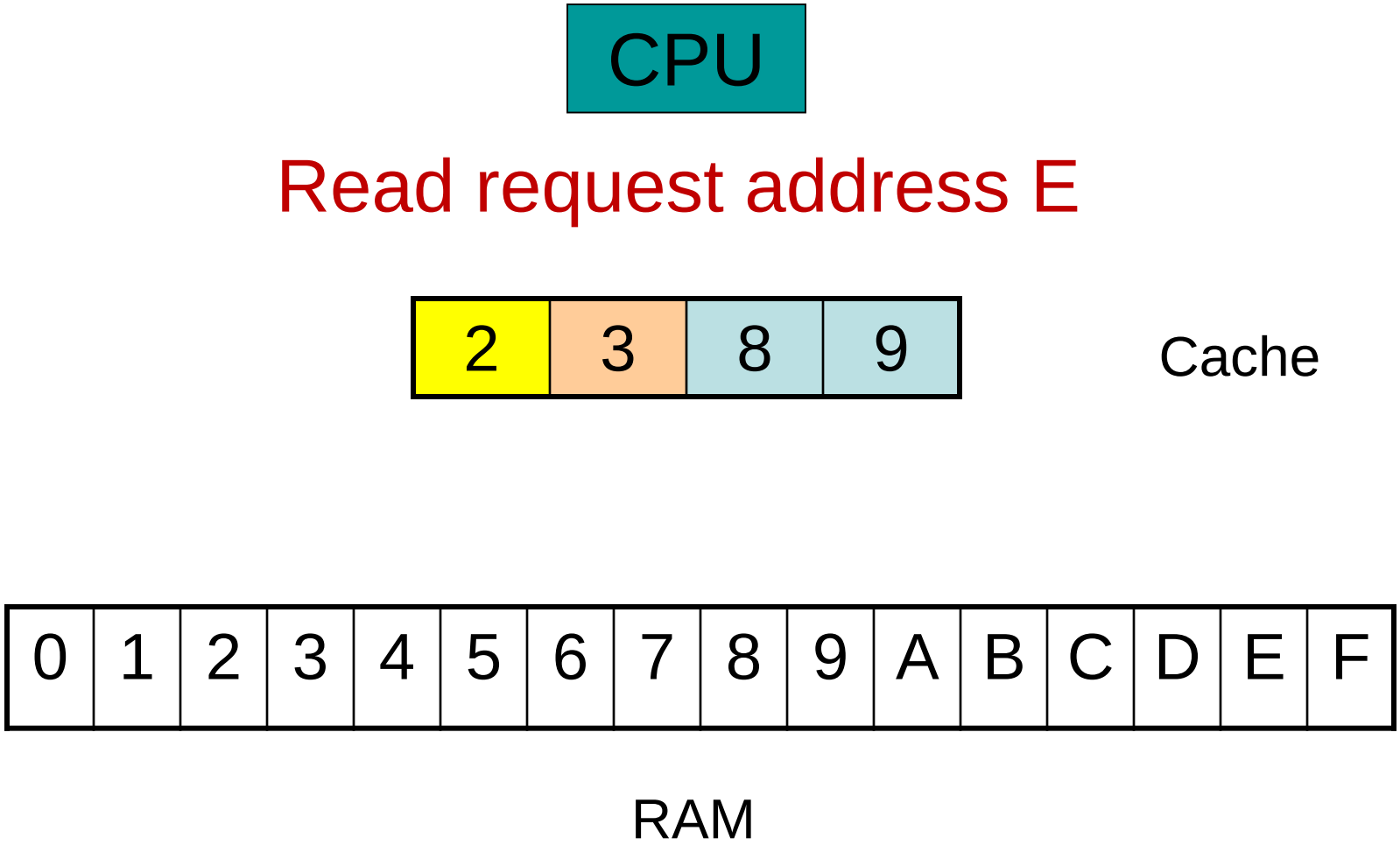
Data copied from CPU to cache



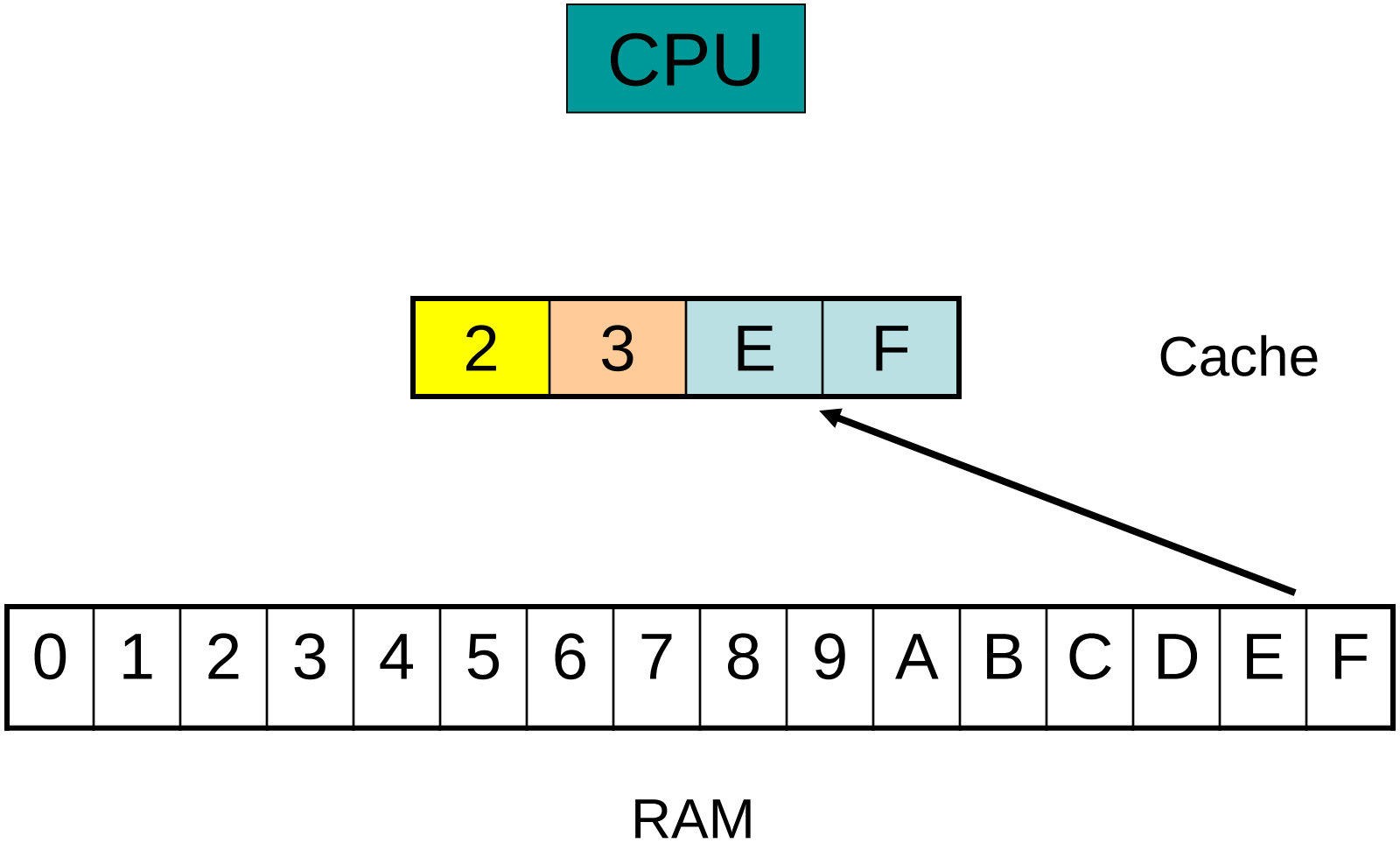


RAM

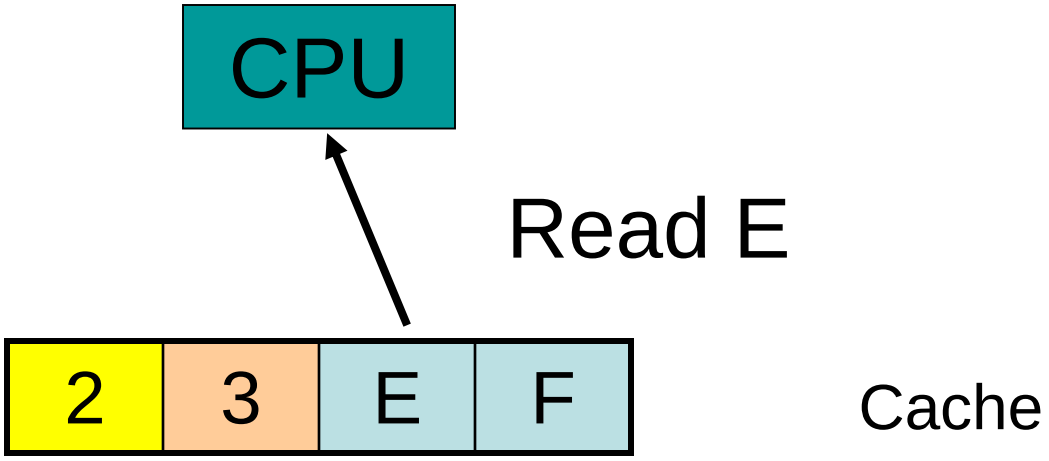
Cache Miss

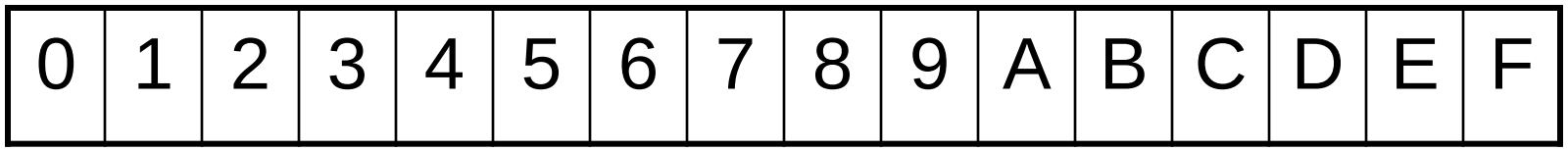

CPU
Read request address E
2
3
8
9
Cache
0
1
2
3
4
5
6
7
8
9
A
B
C
D
E
F
RAM

Line copied from RAM to cache



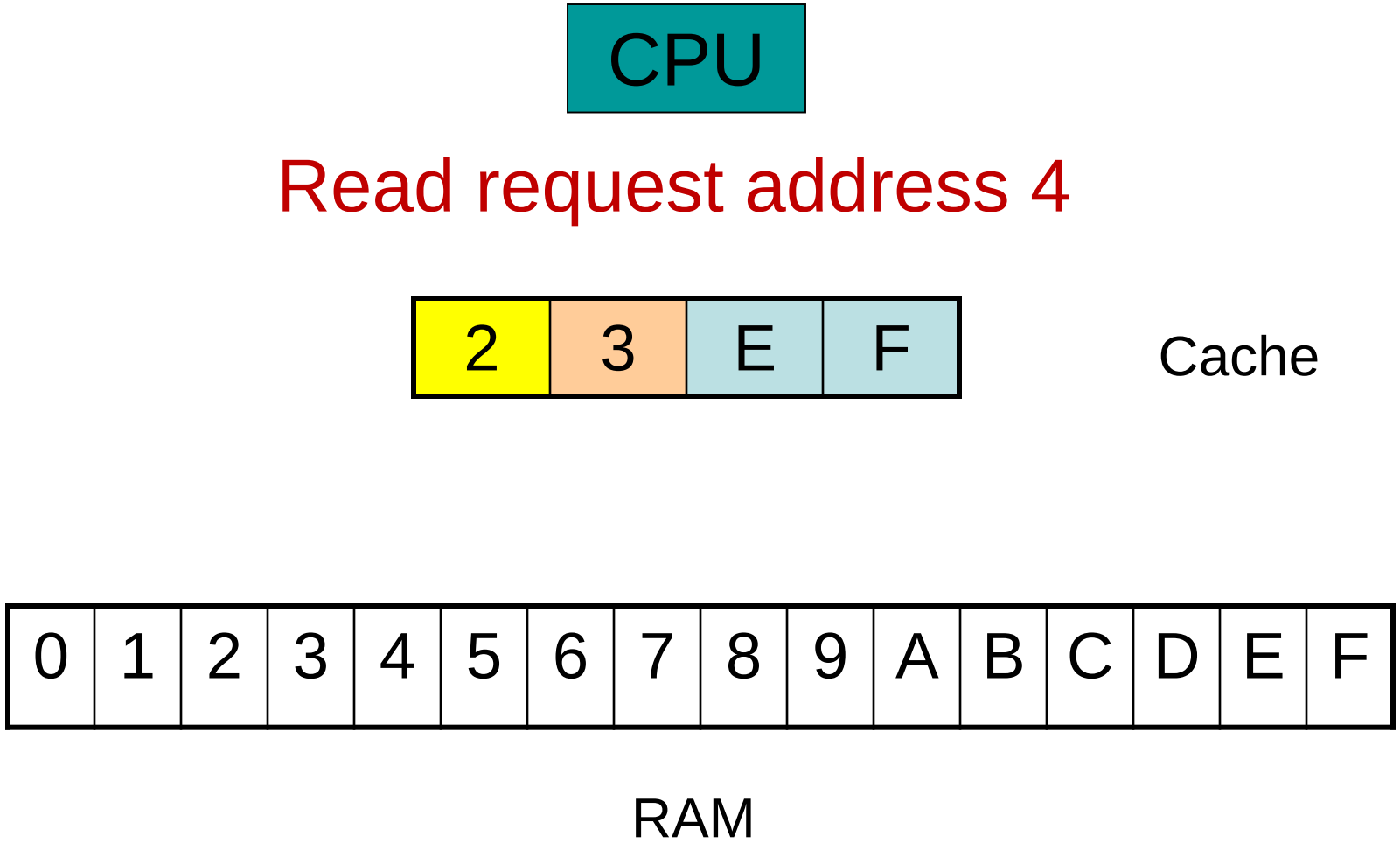
Data copied from cache to CPU



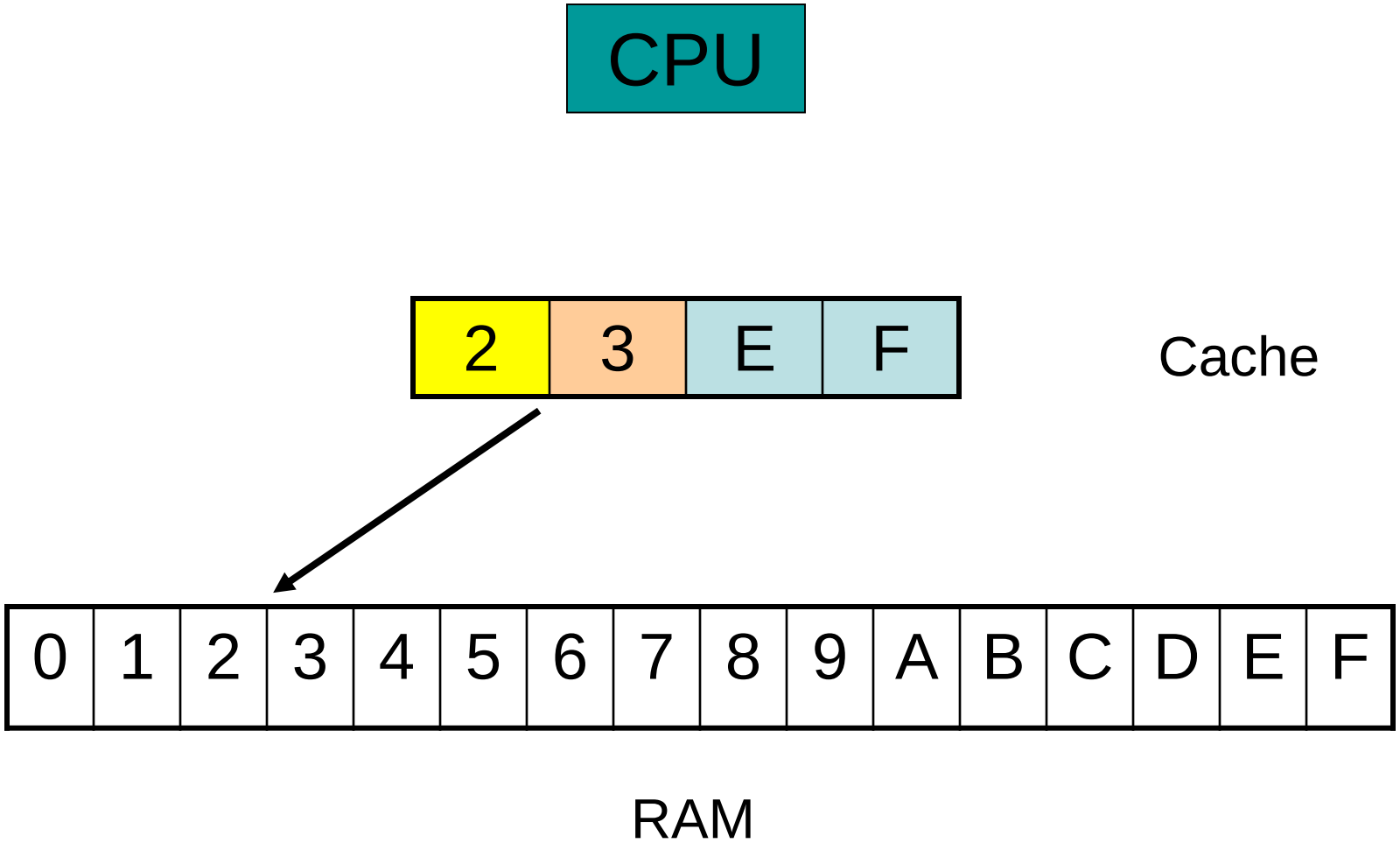


RAM

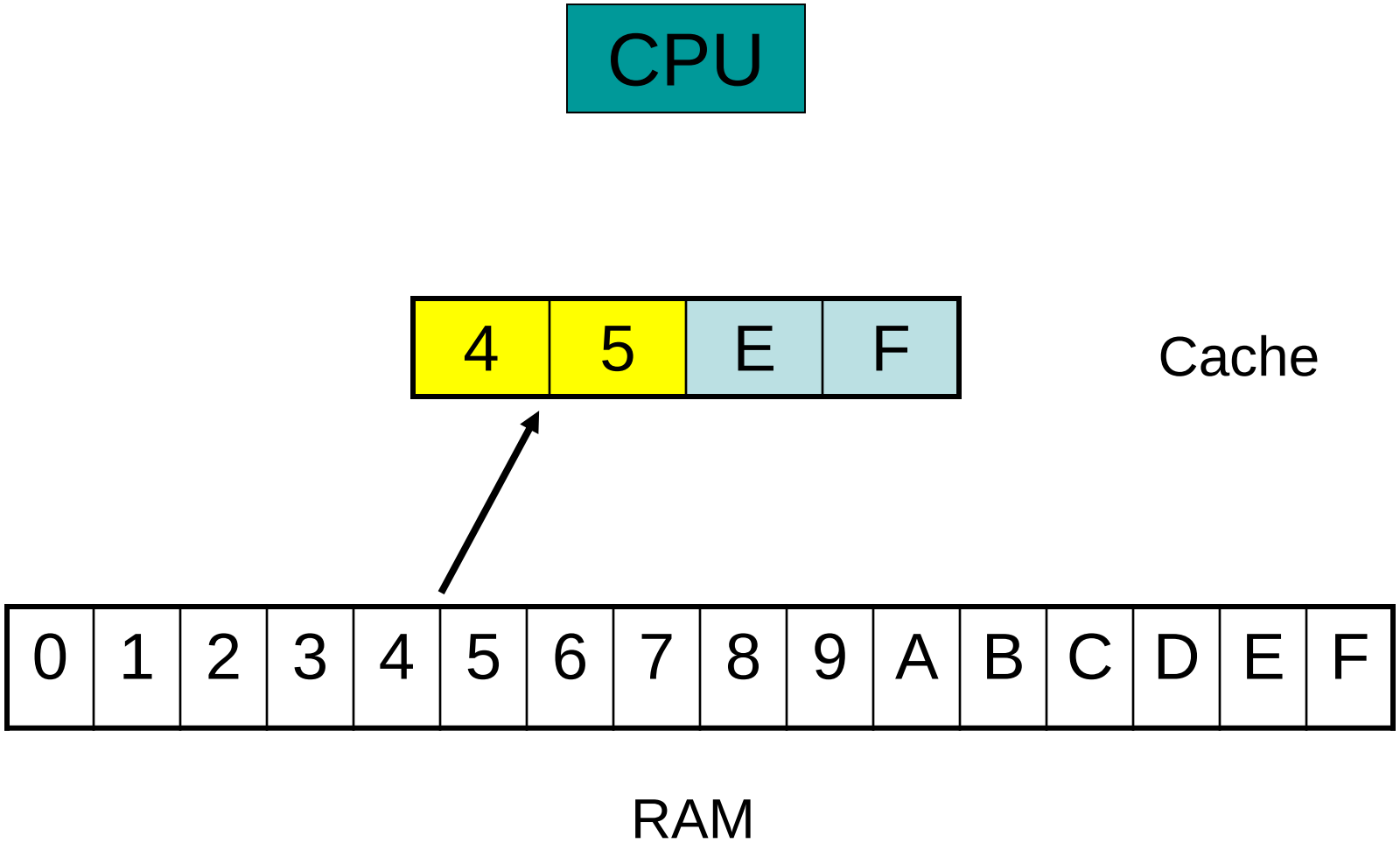
Cache Miss


CPU
Read request address 4
2
3
E
F
Cache
0
1
2
3
4
5
6
7
8
9
A
B
C
D
E
F
RAM

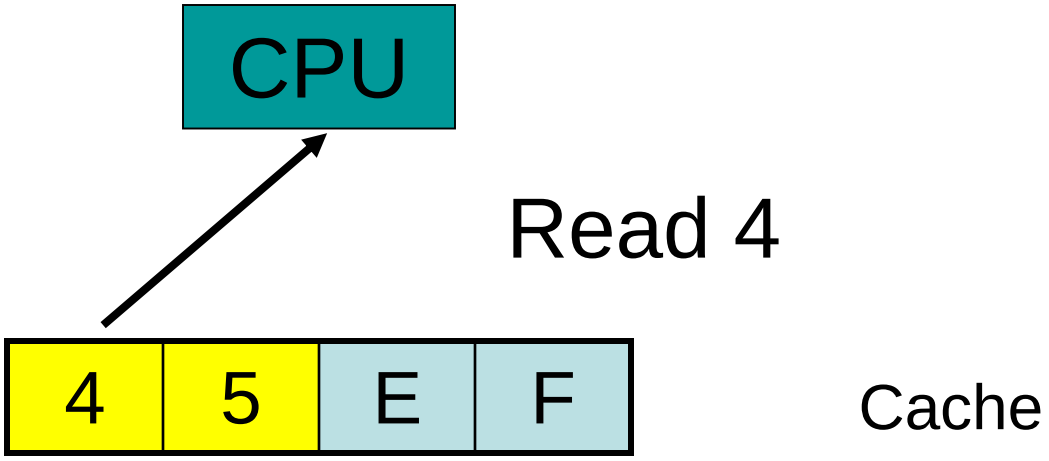
Line copied from cache to RAM



Line copied from RAM to cache



Data copied from cache to CPU



0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F

RAM

When should we hold the second COMP275 exam?

A. Wednesday, October 21

B. Friday, October 23

Hit Ratio

- The hit ratio is the percentage of CPU memory references that are found in the cache

  hit ratio = 100% * hit count / access count

- A high hit ratio means that most of the time the CPU could get the information it wanted from the cache

Cache Effectiveness

- Let
  - h = hit rate
  - c = Time to access the cache
  - m = Time to access the RAM

Access time = h * c + (1-h) * m

Example:

c = 6 ns, m = 60ns and h = 90%

Access = 0.9*6ns + (1-0.9)*60ns = 11.5ns

High Hit Rate is Important

- There is a significant reduction in speed as the hit rate drops
- The challenge in cache design is to ensure that the desired data and instructions are in the cache
- The cache system must be quickly searchable as it is checked every memory reference

Consider a system with 4.5 nsec cache and 55 nsec RAM. If the system has a 92% hit rate, what is the average memory access time?

1. 4.1 nsec
2. 8.5 nsec
3. 30 nsec
4. 51 nsec
5. 59.5 nsec

Cache Design Considerations

- Size
- Size of cache line
- Combined or separate data and instruction
- Cache levels
- Write policy
- Cache on physical or program addresses Mapping
- Replacement policy
- Multiprocessor cache coherence

Cache Size

- Bigger is better
- Cache size is limited by
  - Space on the processor chip
  - Manufacturing limitations
  - Cost

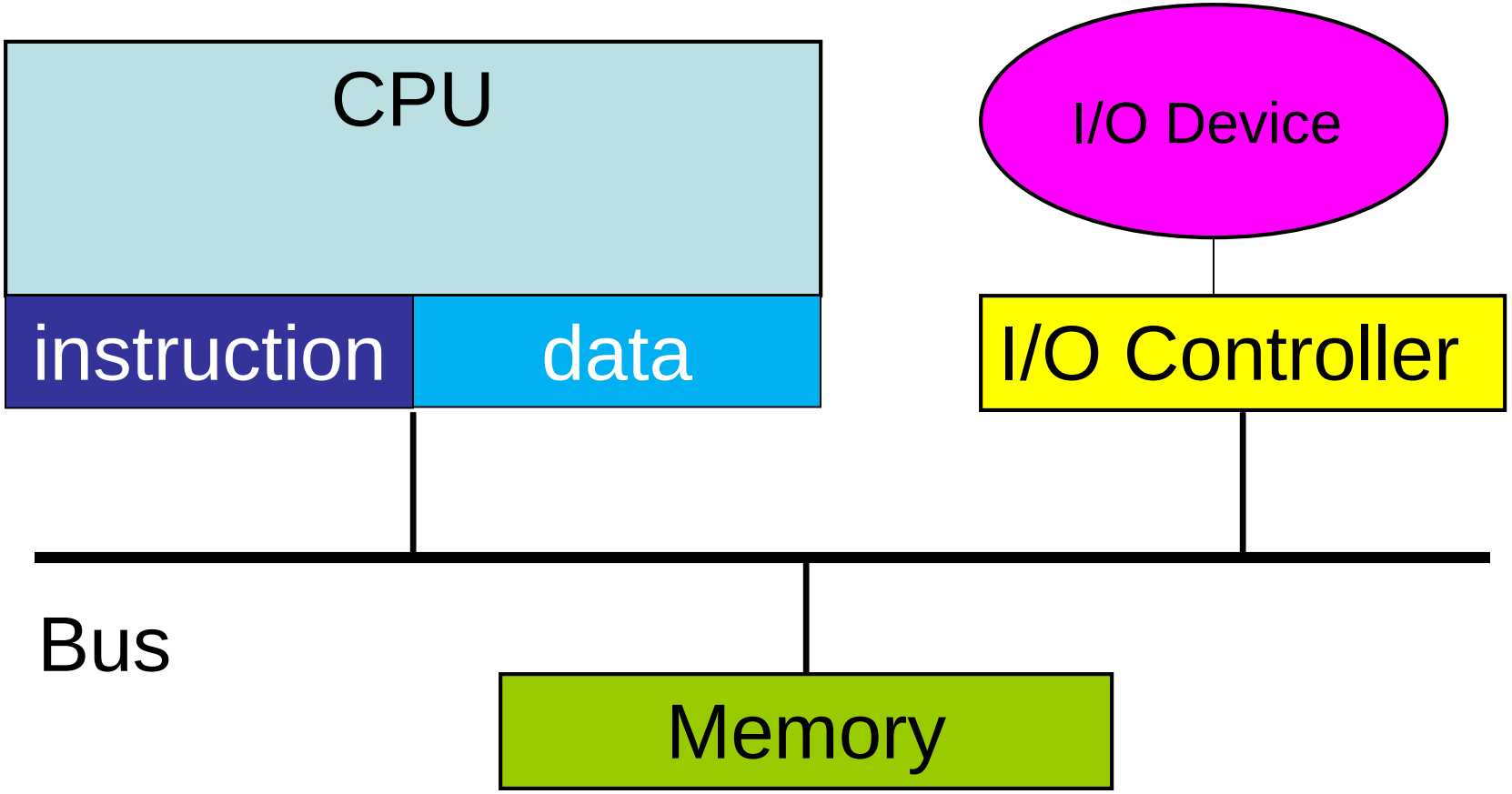
Cache Line Size

- A cache line is the amount of data copied into the cache at one time. 16 to 64 bytes is common
- When there is a cache miss, a line of memory is copied into the cache
- The bigger the line size, the fewer lines in the cache
- Big lines will copy more nearby addresses
- The bigger the line the more likely all of the addresses will not be referenced

Combined or Separate Caches

- Some systems use one cache for instructions and another cache for data
- Intel Pentium Level 1 cache uses a separate 16 KB cache for data and instructions
- Allow simultaneous instruction and operand fetch
- Data and instructions should not be mixed together in the program
- Instructions should be read-only

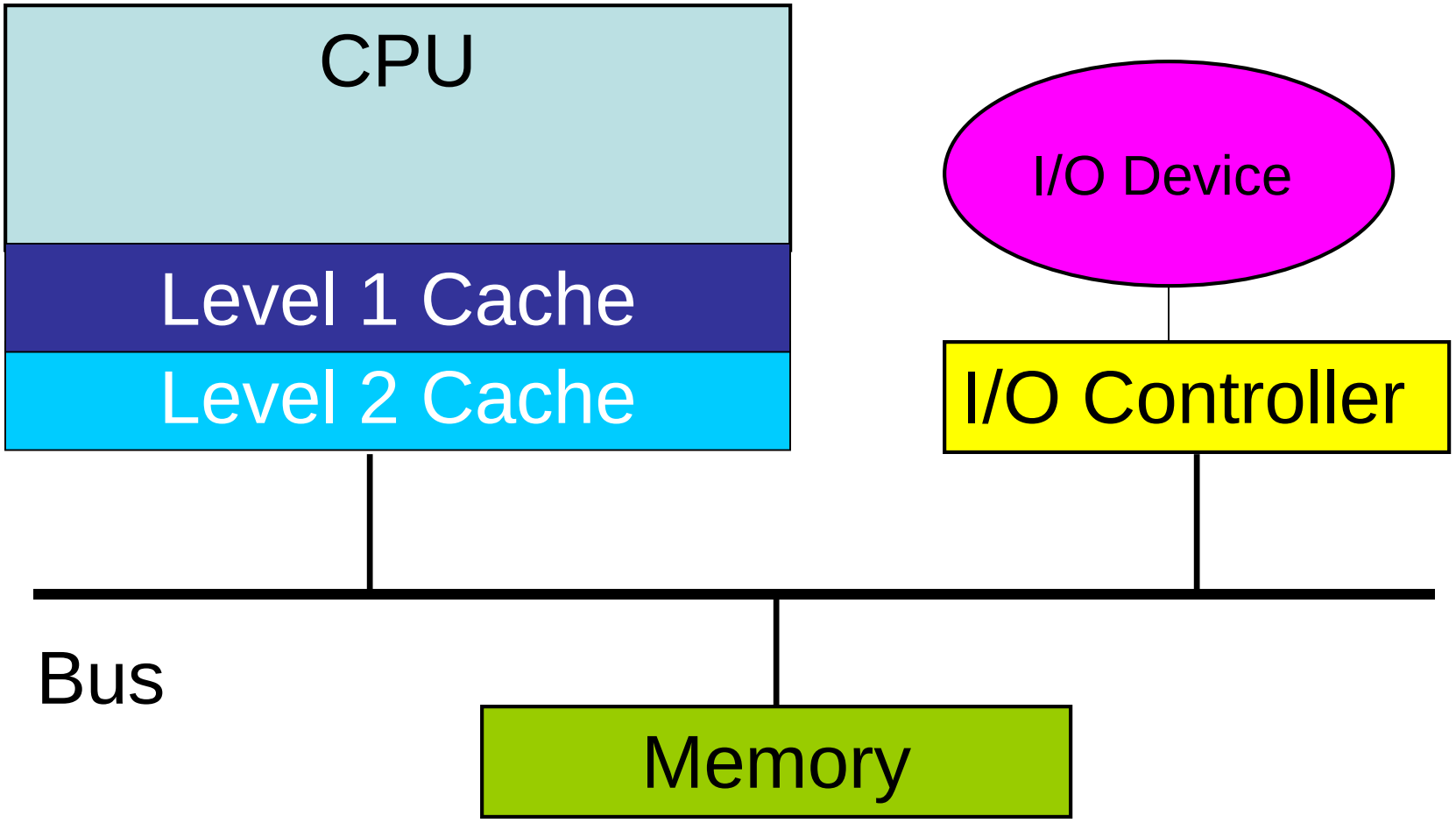
Basic Computer Components



Cache levels

- Most systems have two or three cache levels. Higher level caches are usually larger.
- If data is not in level 1 cache (L1), the memory system looks for the data in the level 2 cache (L2)
- Only if the data is not in L2 does the system get the data from RAM
- The sequence of accesses received by L2 will be different from L1

Basic Computer Components



Student Cache Survey

64 byte lines, 32K L1 8-way, separate date/inst

CPU	L2map	L2size
Intel(R) Core(TM)2 Duo CPU E8500 @ 3.16GHz	24 way	6M
Intel(R) Core(TM)2 CPU 6320 @ 1.86GHz	16 way	4M
Intel(R) Core(TM)2 Duo CPU T7250 @ 2.00GHz	8 way	2M
Intel(R) Core(TM)2 Duo CPU T5470 @ 1.60GHz	8 way	2M
Intel T2400 @ 1.83 GHz	8 way	2M
Genuine Intel® CPU T2250 @ 1.73GHz	8 way	2M
Genuine Intel(R) CPU T1300 @ 1.66GHz	8 way	2M
Intel(R) Pentium(R) M processor 1.40GHz	8 way	2M
Genuine Intel(R) CPU 585 @ 2.16GHz	8 way	1M
Intel(R) Pentium(R) Dual CPU T2370 @ 1.73GHz	4 way	1M
Intel(R) Pentium(R) Dual CPU T2330 @ 1.60GHz	4 way	1M
Genuine Intel(R) CPU 575 @ 2.00GHz	4 way	1M
Intel(R) Celeron(R) M CPU 410 @ 1.46GHz	4 way	1M
Intel Pentium 4 CPU 2.53GHz	8 way	512K
Celeron(R) CPU 2.53GHz	Only 16K level one cache	

If you have 512 KB of cache and 2 GB of RAM, what is the RAM to cache ratio?

1. 4
2. 512
3. 4096
4. 2 billion

Write policy

- When the processor changes a data item, it is changed in the cache. If this line of cache is replaced, RAM needs to be updated
- **Write through** – RAM is updated whenever data is changed
- **Write back** – RAM is updated when the cache line is replaced

Physical or Program Addresses

- Is the caching based on the program effective address or the RAM physical address?
- Program address caching does not have to wait for the memory system to convert the virtual address to a physical address
- If you cache on program addresses, you must clear the cache every time you change process

Multiprocessor Cache Coherence

- Each processor has its own cache
- In a two processor system, both processors may access the same address at the same (or almost the same) time
- Data values might be located in the cache of different processors
- If a CPU updates a data value, how does the other CPU’s cache get the new value?

Mapping

- The memory system has to quickly determine if a given address is in the cache
- There are three popular methods of mapping addresses to cache locations
- **Fully Associative –** Search the entire cache for an address
- **Direct** – Each address has a specific place in the cache
- **Set Associative –** Each address can be in any of a small set of cache locations

Replacement policy

- When a cache miss occurs, data is copied into some location in cache
- With Set Associative or Fully Associative mapping, the system must decide where to put the data and what values will be replaced
- Cache performance is greatly affected by properly choosing data that is unlikely to be referenced again

Replacement Options

- First In First Out (FIFO)
- Least Recently Used (LRU)
- Pseudo LRU
- Random

Exam in two weeks

- The second exam in COMP375 will be on Wednesday or Friday, October 21 or 23
- It covers all the material since the first exam
  - Interrupts
  - Busses
  - Memory
  - VLSI
  - Cache