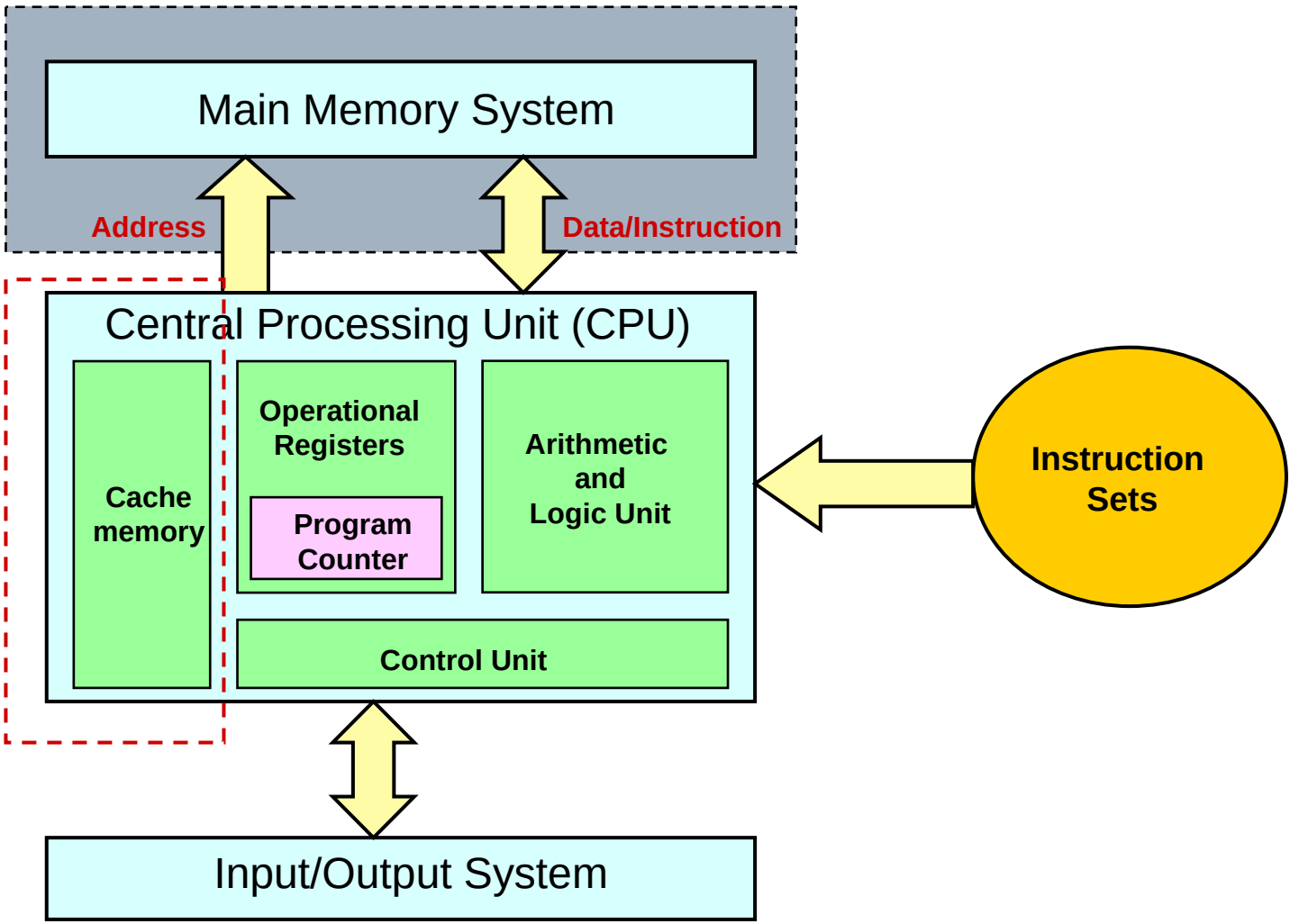
Chapter 6
The Memory System

Jin-Fu Li
Department of Electrical Engineering
National Central University
Jungli, Taiwan

Outline

- Basic Concepts
- Semiconductor Random Access Memories
- Read Only Memories
- Speed, Size, and Cost
- Cache Memories
- Performance Considerations
- Virtual Memories

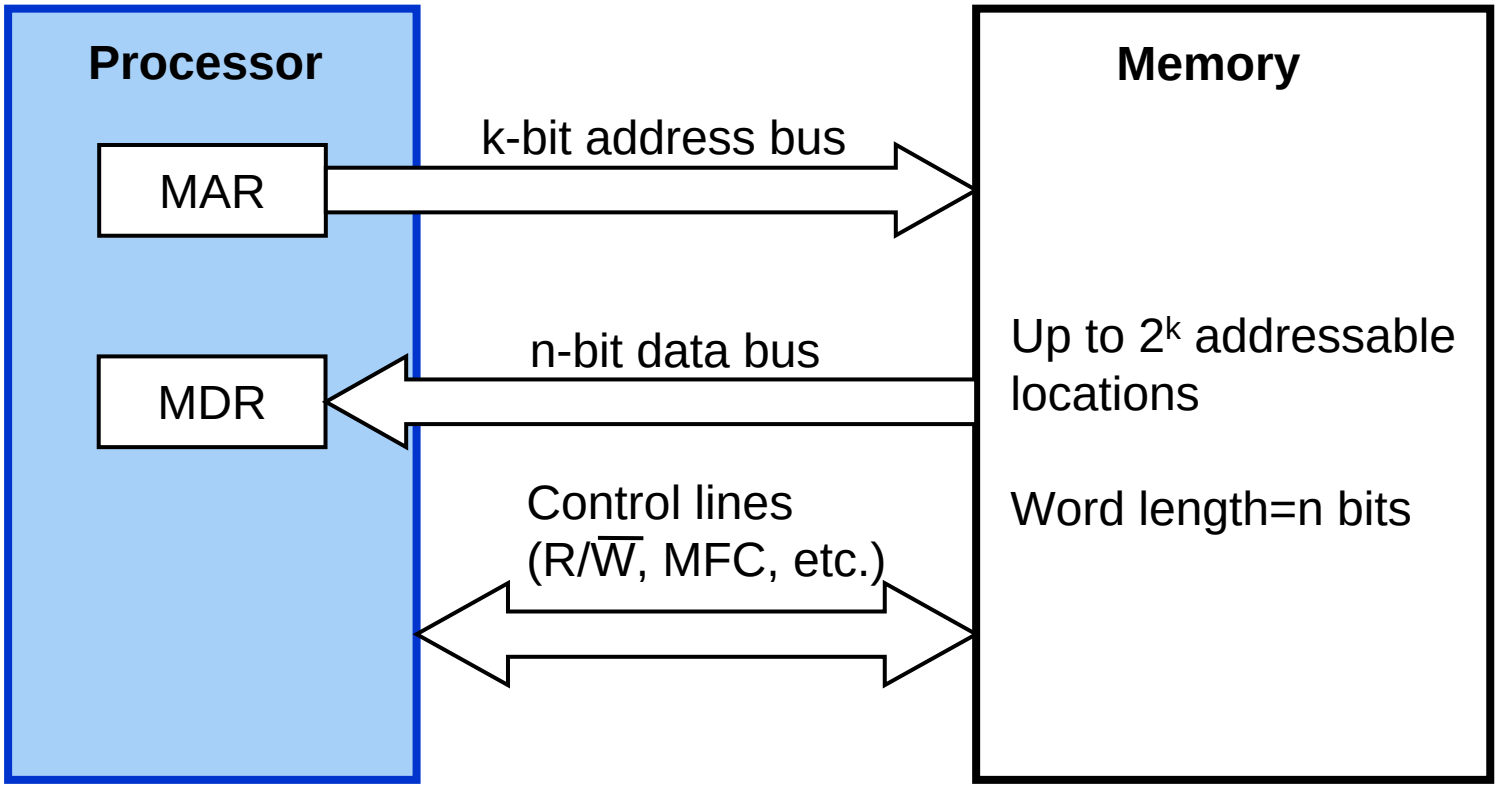
Content Coverage



Basic Concepts

- The maximum size of the memory that can be used in any computer is determined by the addressing scheme
  - For example, a 16-bit computer that generates 16-bit addresses is capable of addressing up to 2^{16}=64K memory locations.
  - Similarly, machines whose instructions generate 32-bit addresses can utilize a memory that contains up to 2^{32}=4G memory locations
- Most modern computers are byte addressable
- Form the system standpoint, we can view the memory unit as a block box
  - Data transfer between the memory and processor takes place through the use of two processor registers, MAR and MDR

Connection of the Memory to the Processor



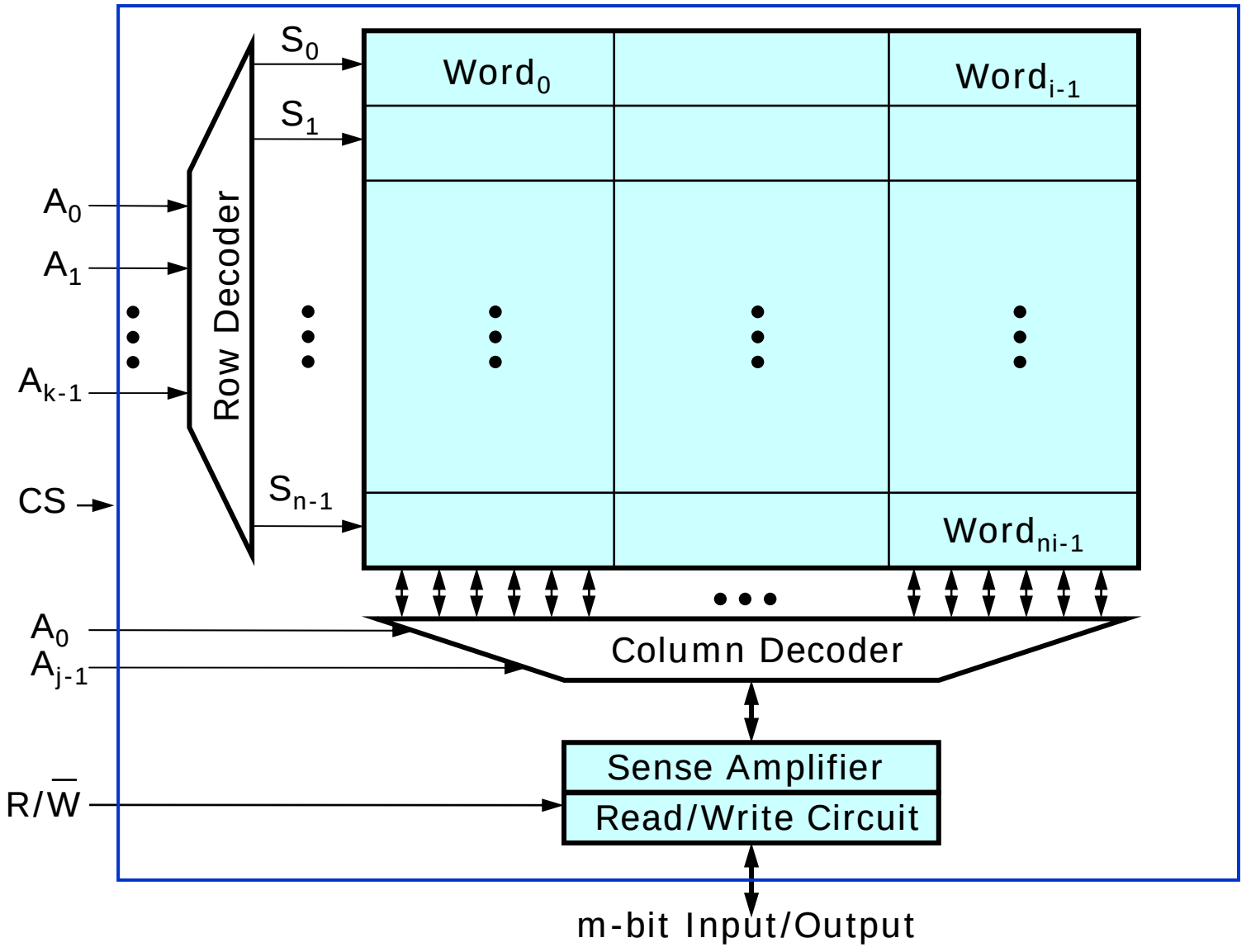
Basic Concepts

- A useful measure of the speed of memory units is the time that elapses between the initiation of an operation and the completion of that operation. This is referred to as the memory access time.
- Another important measure is the memory cycle time, which is the minimum time delay required between the initiation of two successive memory operations
- A memory unit is called random-access memory (RAM) if any location can be accessed for a Read or Write operation in some fixed amount of time that is independent of the location's address
- The memory cycle time is the bottleneck in the system

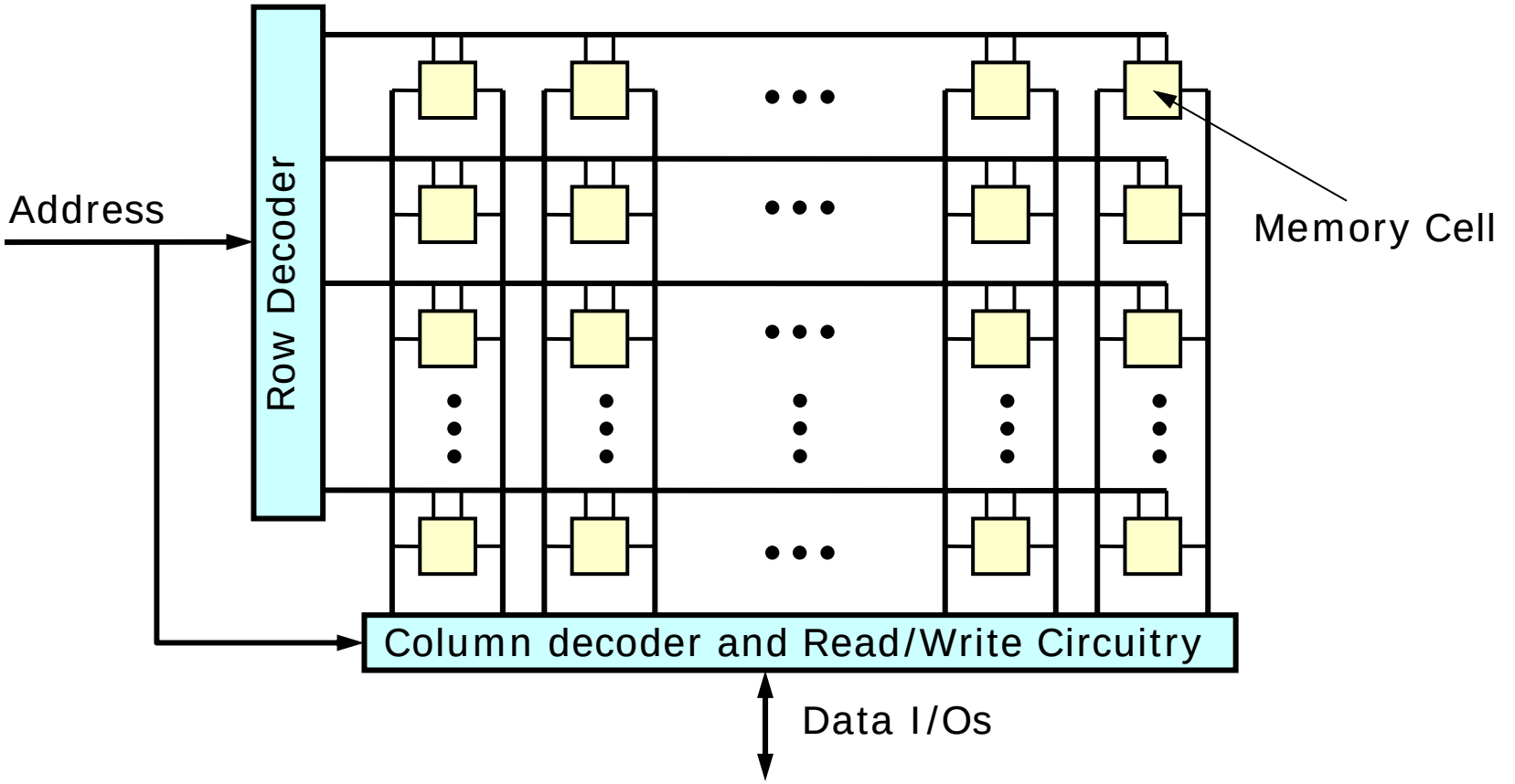
Basic Concepts

- One way to reduce the memory access time is to use a cache memory
- Cache memory is a small, fast memory that is inserted between the larger, smaller main memory and the processor.
- Virtual memory is used to increase the apparent size of the physical memory. Data are addressed in a virtual address space that can be as large as the addressing capability of the processor. But at any given time, only the active portion of this space is mapped onto locations in the physical memory. The remaining virtual addresses are mapped onto the bulk storage devices used, such as magnetic disks

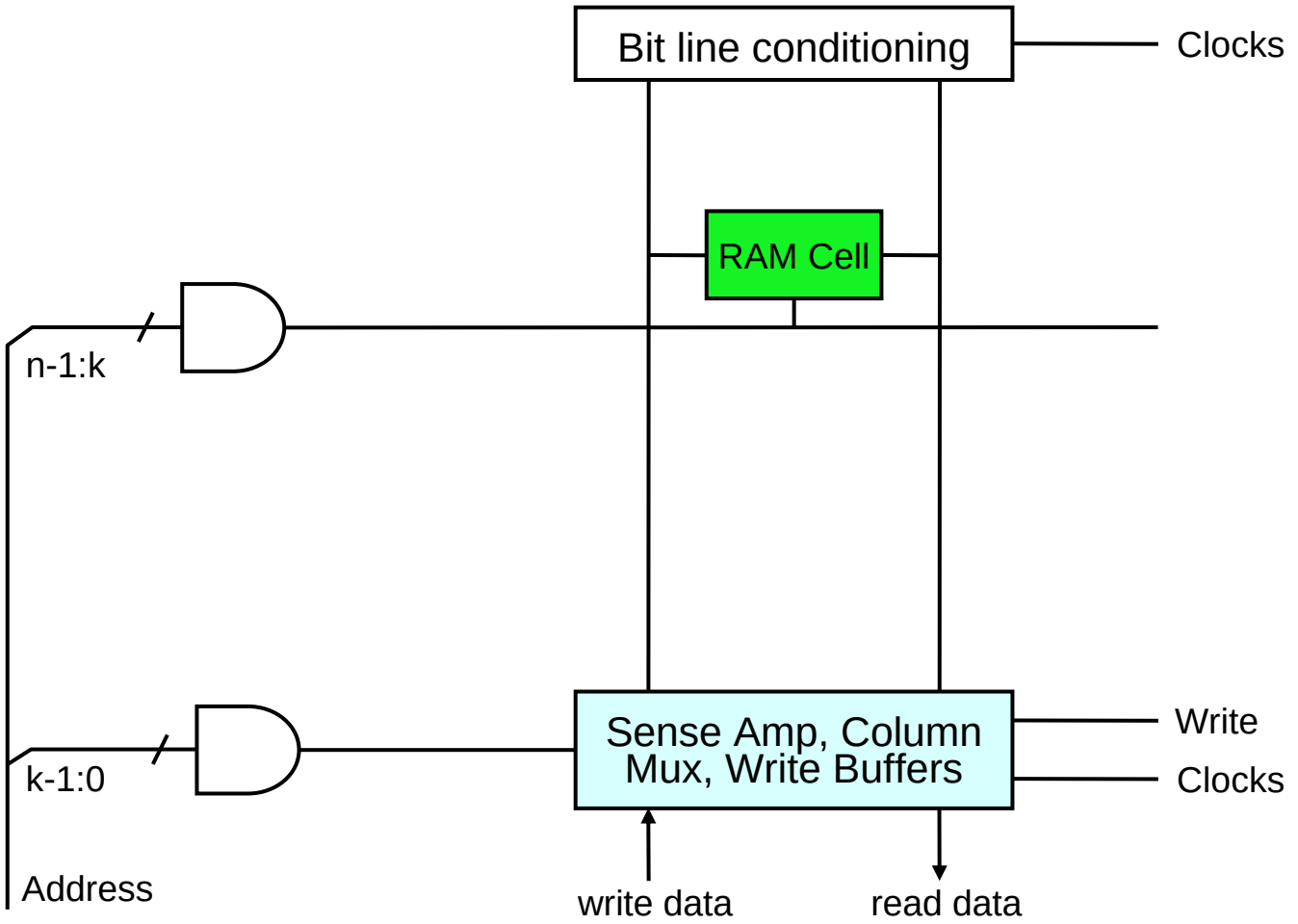
Semiconductor Random Access memories



Organization of Bit Cells in a Memory

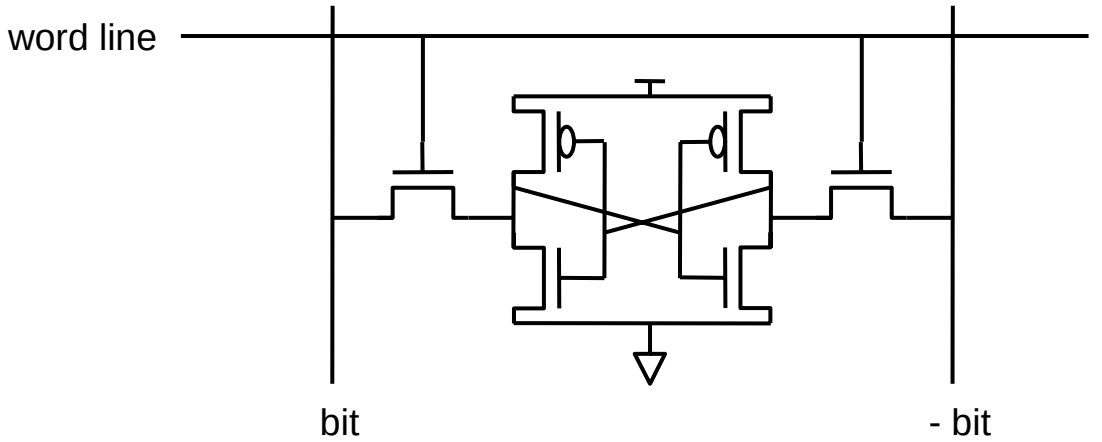


Generic RAM Circuit

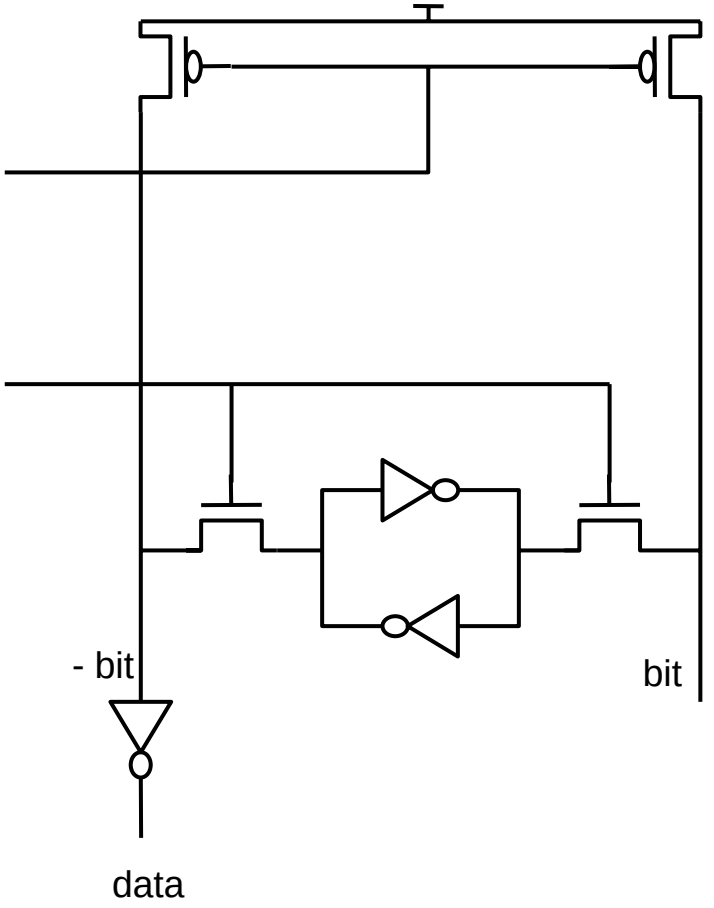


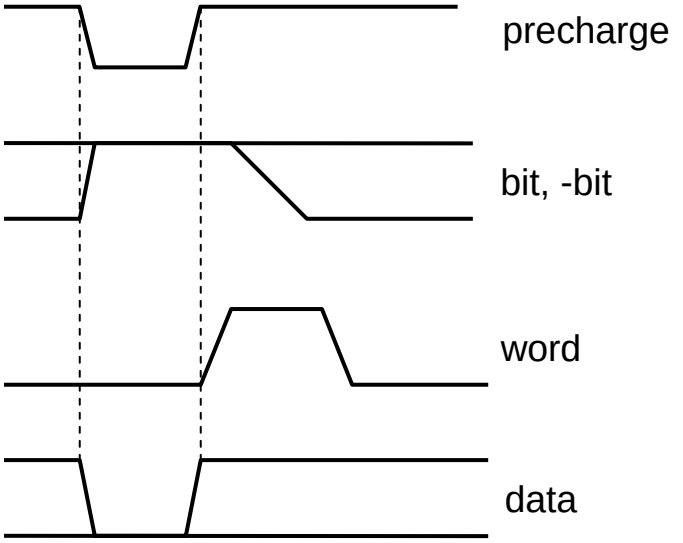
Static RAM

- A 6T SRAM cell

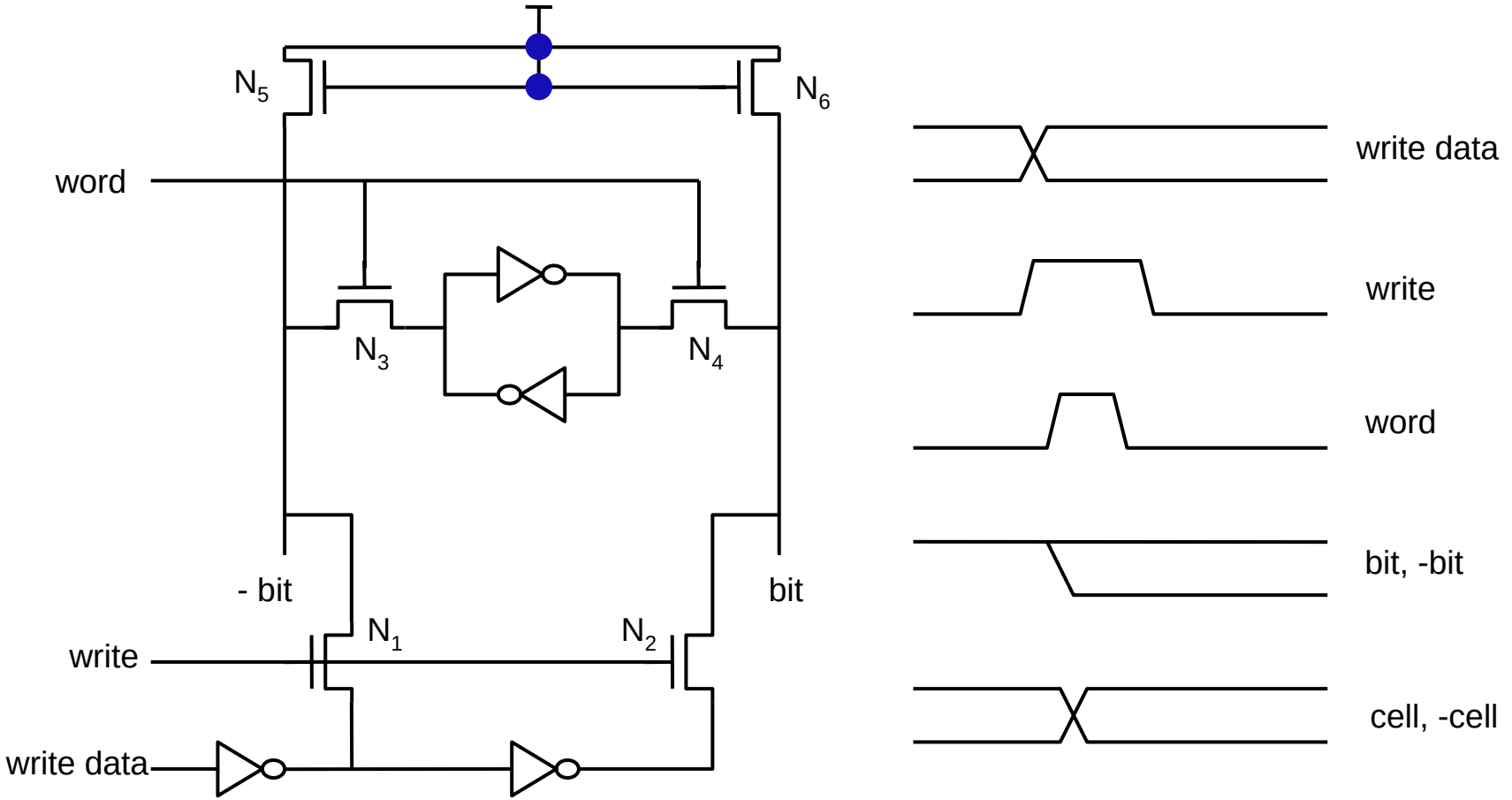


Read Operation



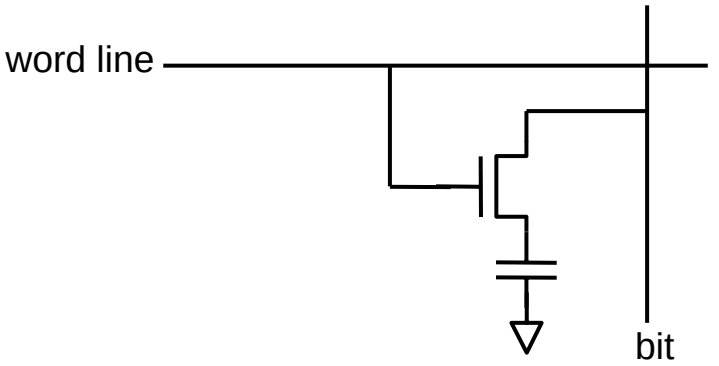


Write Operation



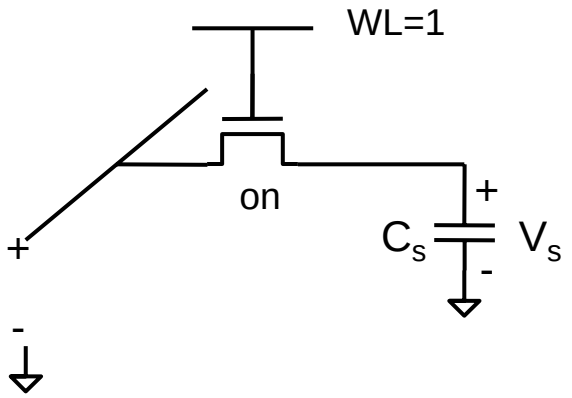
Dynamic RAM

- A typical DRAM cell

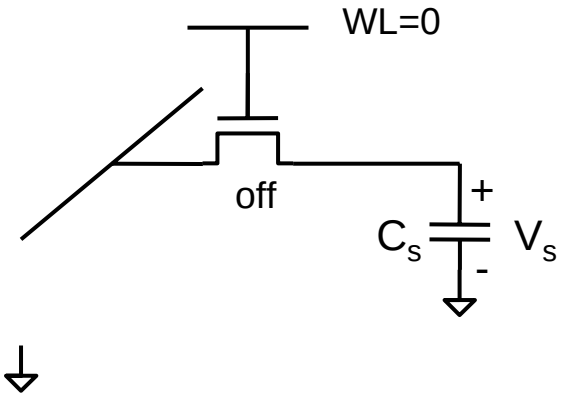


Data Retention Time of a DRAM Cell

- Write and hold operation in a DRAM cell



Write Operation



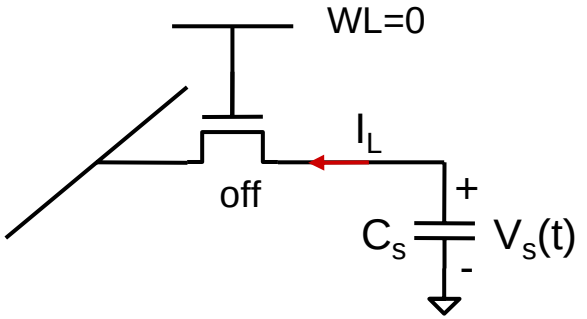
Hold

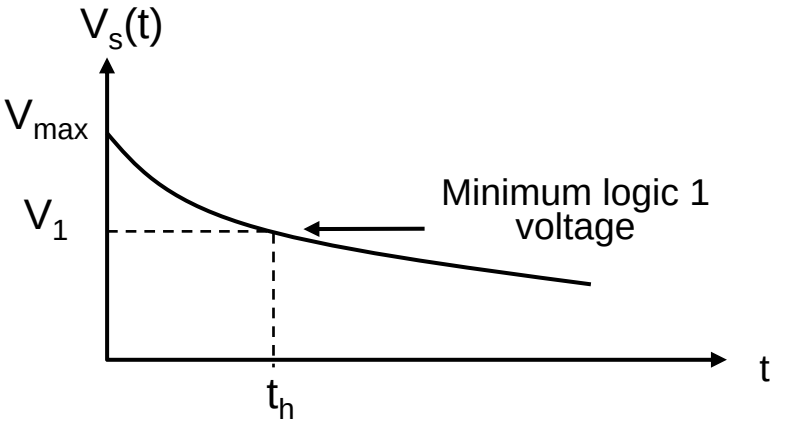
$$V_s = V_{\max} = V_{DD} - V_{tn}$$

$$Q_{\max} = C_s (V_{DD} - V_{tn})$$

Data Retention Time of a DRAM Cell

- Charge leakage in a DRAM cell





$$I_L = -\left(\frac{dQ_s}{dt}\right)$$

$$I_L = -C_s\left(\frac{dV_s}{dt}\right)$$

$$I_L \approx -C_s\left(\frac{\Delta V_s}{\Delta t}\right)$$

$$t_h = |\Delta t| \approx \left(\frac{C_s}{I_L}\right)\Delta V_s$$

DRAM Refresh Operation

As an example, if IL=1nA, Cs=50fF, and the difference of Vs is 1V, the hold time is

$$t_h = \frac{50 \times 10^{-15}}{1 \times 10^{-9}} \times 1 = 0.5 \mu s$$

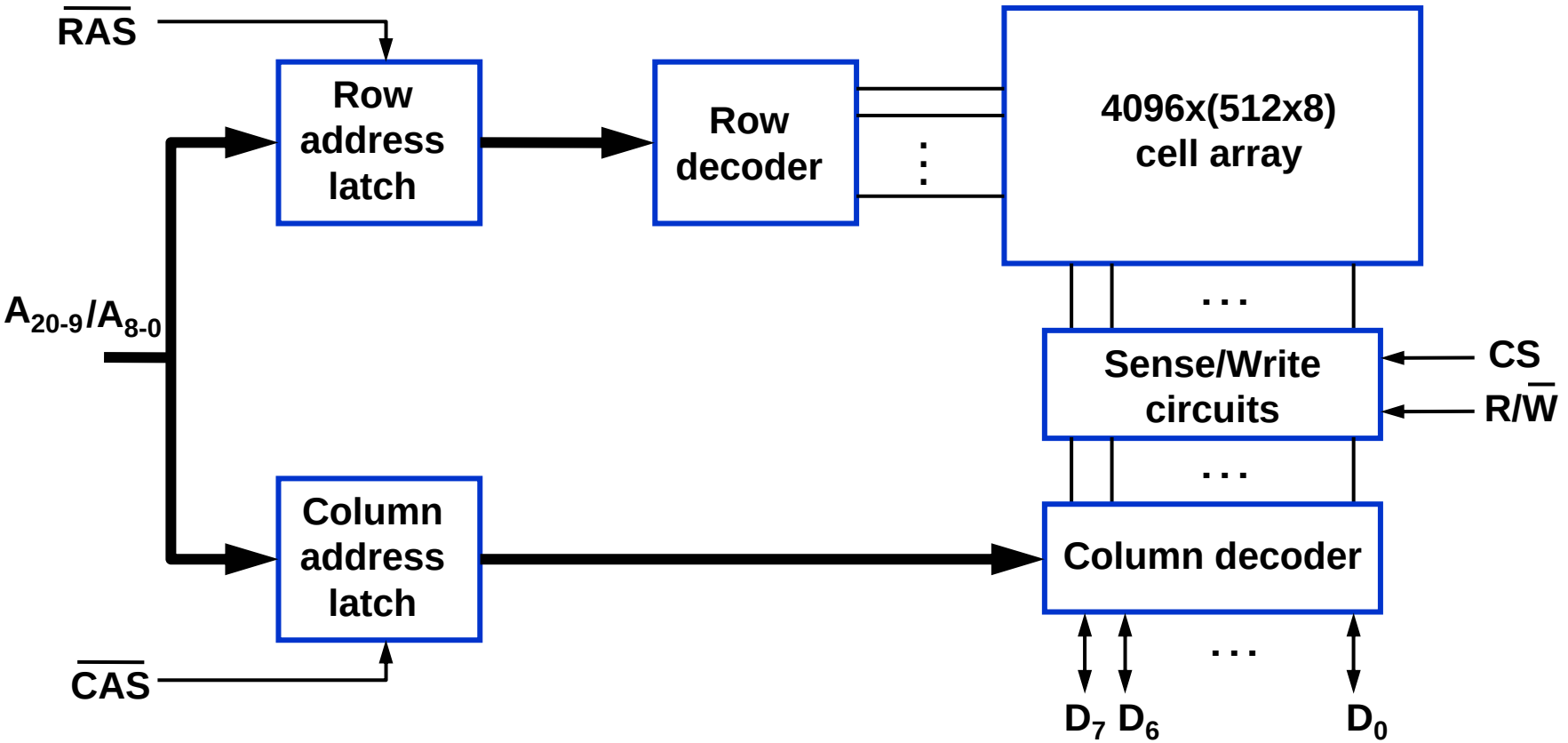
Memory units must be able to hold data so long as the power is applied. To overcome the charge leakage problem, DRAM arrays employ a **refresh operation** where the data is periodically read from every cell, amplified, and rewritten.

The refresh cycle must be performed on every cell in the array with a minimum refresh frequency of about

$$f_{refresh} \approx \frac{1}{2t_h}$$

Asynchronous DRAMs

- A 16M-bit DRAM chip, configured as 2Mx8, is shown below

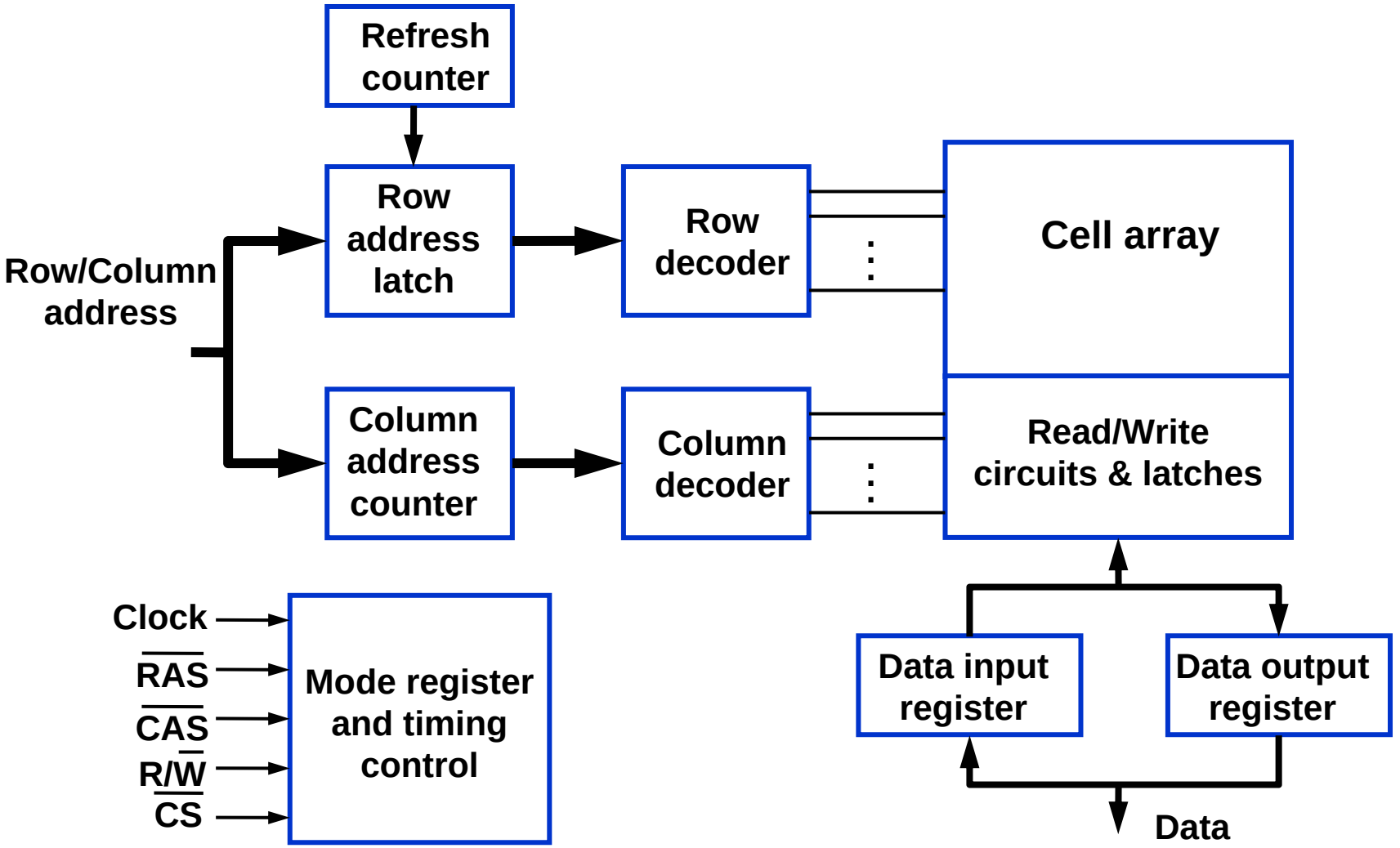


Fast Page Mode

- When the DRAM shown above is accessed, the contents of all 4096 cells in the selected row are sensed, but only 8 bits are placed on the data lines D_{7-0}. This byte is selected by the column address bits A_{8-0}.
- A simple modification can make it possible to access the other bytes in the same row without having to reselect the row
  - A latch can be added at the output of the sense amplifier in each column
  - Transfer of successive bytes is achieved by applying a consecutive sequence of column address under the control of successive CAS signals
- This block transfer capability is referred to as the *fast page mode* feature

Synchronous DRAMs

- The structure of an synchronous DRAM (SDRAM)



Burst Read Operation in an SDRAM

- SDRAMs have several different modes of operation, which can be selected by writing control information into a mode register
- The burst operations use the block transfer capability described above as the fast page mode feature
- In SDRAMs, it is not necessary to provide externally generated pulses on the CAS line to select successive columns. The necessary control signals are provided internally using a column counter and the clock signal

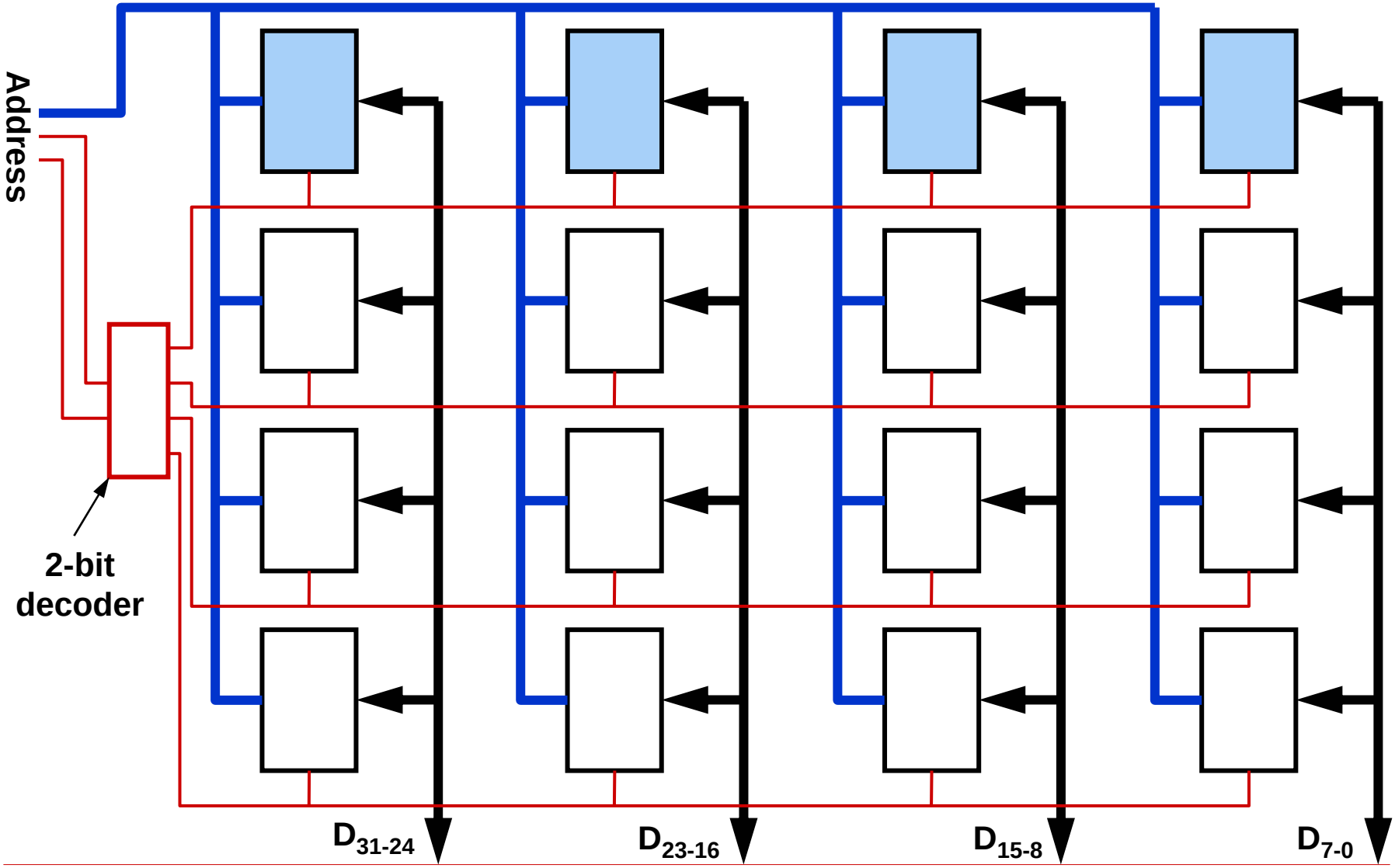
Latency

- Transfers between the memory and the processor involve single words of data or small blocks of words
- The speed and efficiency of these transfers have a large impact on the performance of a computer system
- A good indication of the performance is given by two parameters: latency and bandwidth
- The term memory latency is used to refer to the amount of time it takes to transfer a word of data to or from the memory
- In block transfers, the term latency is used to denote the time it takes to transfer first word of data

Bandwidth

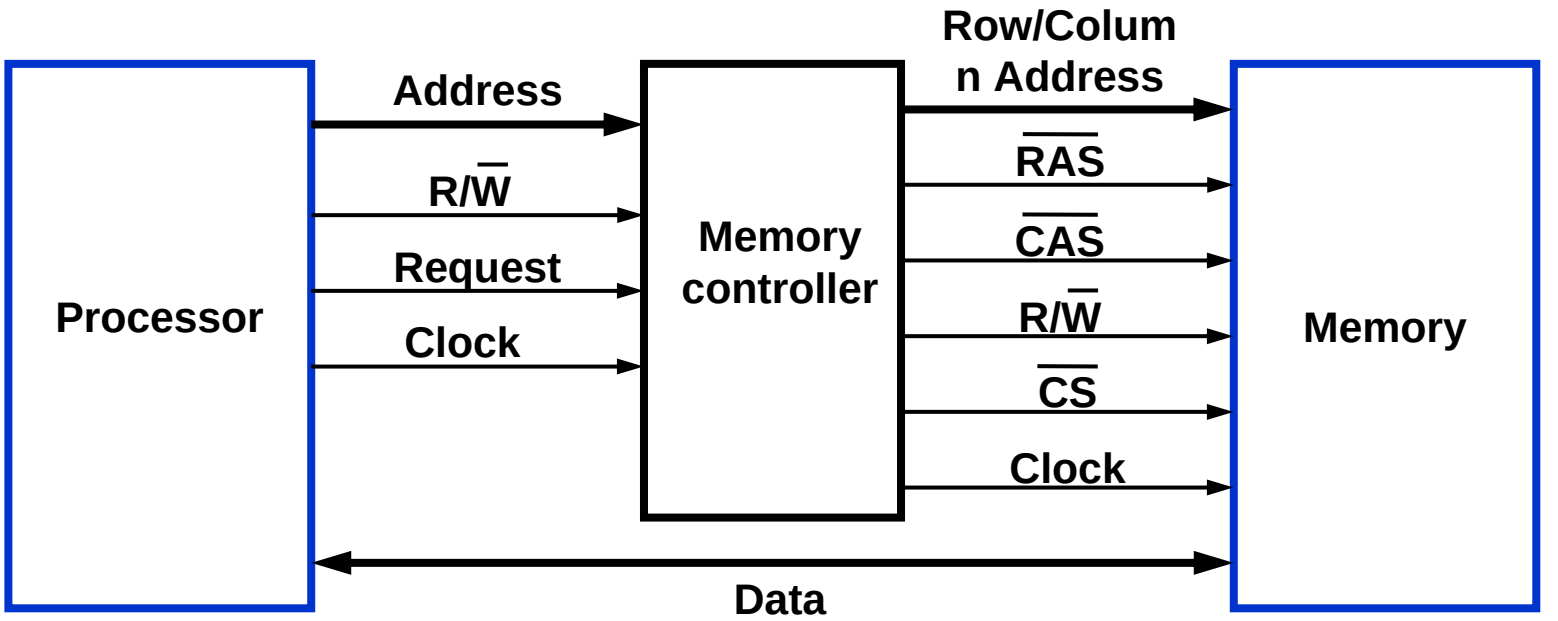
- When transferring blocks of data, it is of interest to know how much time is needed to transfer an entire block
- Since blocks can be variable in size, it is useful to define a performance measure in terms of the number of bits or bytes that can be transferred in one second. This measure is often referred to as the memory bandwidth
- The bandwidth of a memory unit depends on the speed of access to the stored data and on the number of bits that can be accessed in parallel

Structure of Larger Memories



Memory System Considerations

- Memory controller
  - To reduce the number of pins, the DRAM chips use multiplexed address inputs
  - A typical processor issues all bits of an address at the same time
  - The required multiplexing of address bits is usually performed by a memory controller circuit, which is interposed between the processor and the DRAM as shown below

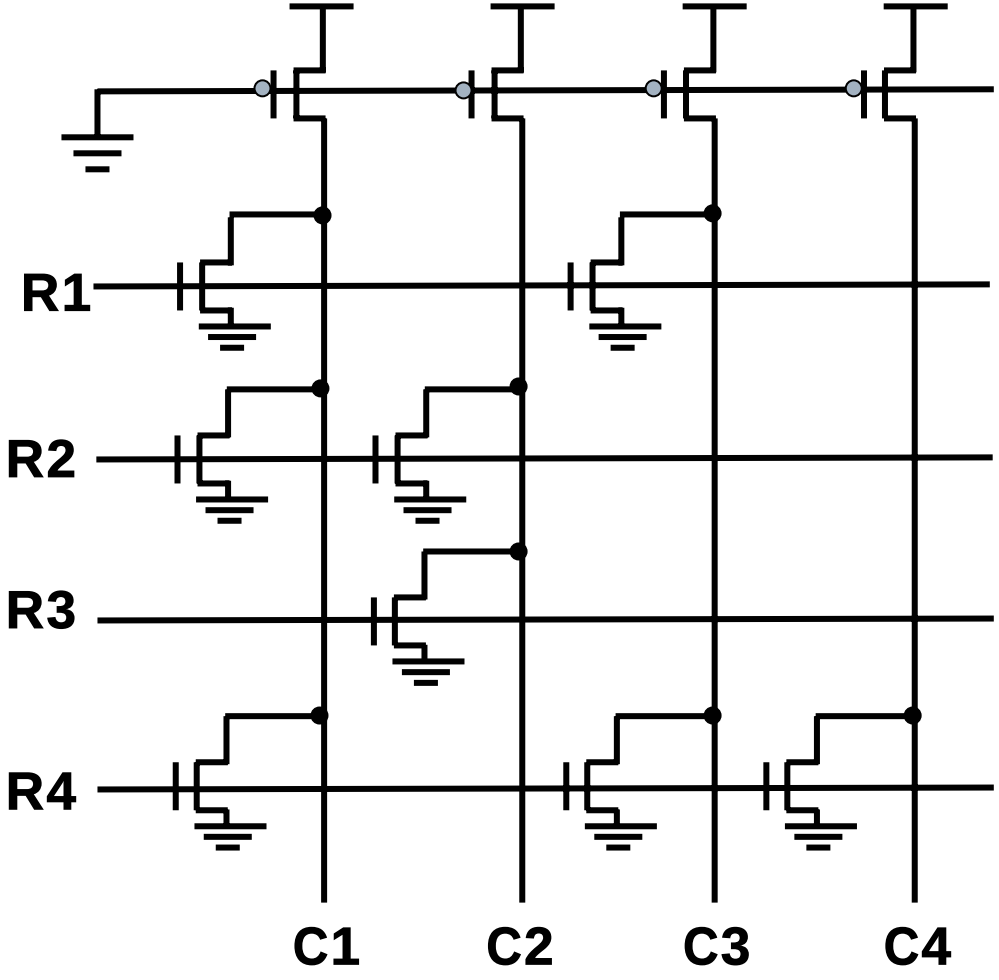


Refresh Overhead

- All DRAMs have to be refreshed
- Consider an SDRAM whose cells are arranged in 8K (8192) rows. Suppose that it takes four clock cycles to access (read) each row. Then it takes 8192x4=32768 cycles to refresh all rows
- At a clock rate of 133MHz, the time needed to refresh all rows is $32768/(133\times10^{6})=246\times10^{-6}$ seconds
- Thus, the refreshing process occupies 0.246ms in each 64-ms time interval. The refresh overhead is 0.246/64=0.0038.

Read Only Memory (ROM)

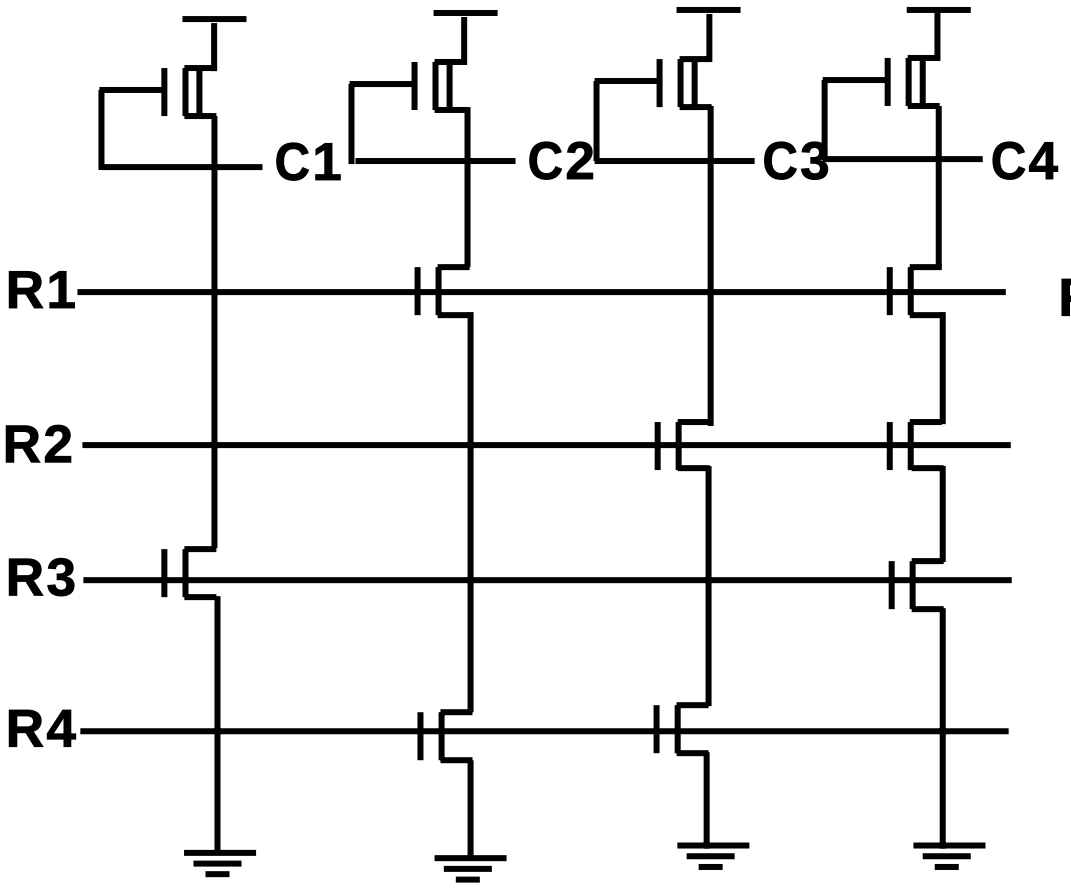
- A 4x 4-bit NOR-based ROM array



R1	R2	R3	R4	C1	C2	C3	C4
1	0	0	0	0	1	0	1
0	1	0	0	0	0	1	1
0	0	1	0	1	0	0	1
0	0	0	1	0	1	1	0

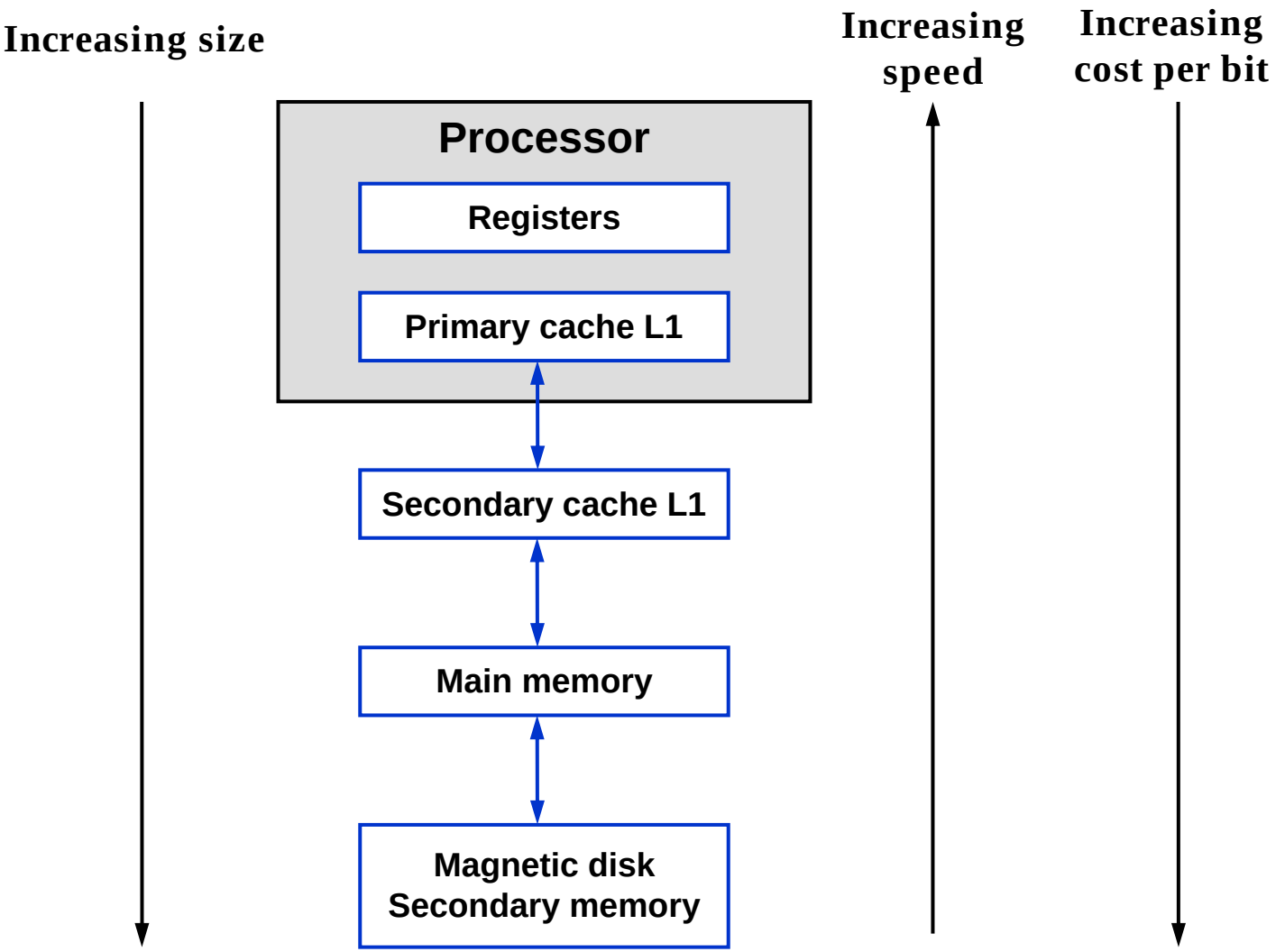
Read Only Memory (ROM)

- A 4x 4-bit NAND-based ROM array



R1	R2	R3	R4	C1	C2	C3	C4
0	1	1	1	0	1	0	1
1	0	1	1	0	0	1	1
1	1	0	1	1	0	0	1
1	1	1	0	0	1	1	0

Memory Hierarchy



Cache Memories

- The speed of the main memory is very low in comparison with the speed of modern processors
- Hence, it is important to devise a scheme that reduces the time needed to access the necessary information
- Since the speed of main memory unit is limited by electronic and packaging constraints, the solution must be sought in a different architectural arrangement
- An efficient solution is to use a fast cache memory which essentially makes the main memory appear to the processor to be faster than it really is

Effectiveness of the Cache Memory

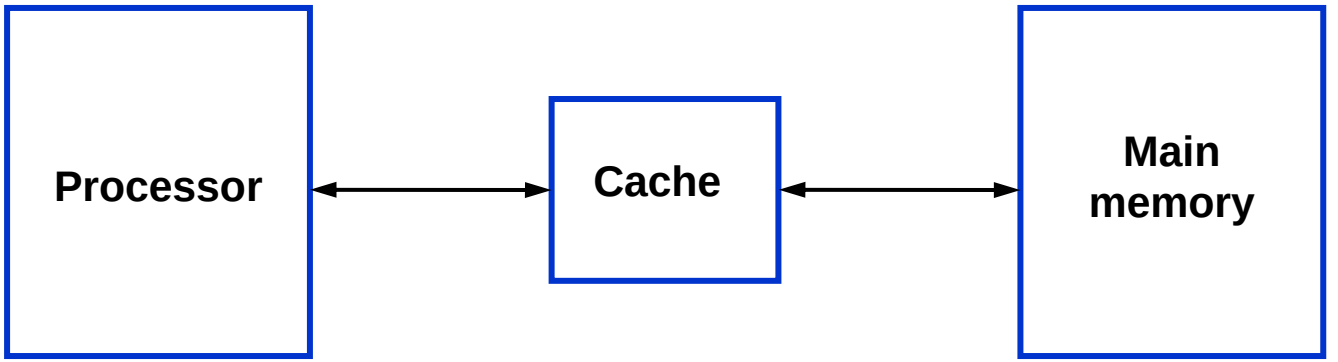
- The effectiveness of the cache mechanism is based on a property of computer programs called locality of reference
- Analysis of programs shows that most of their execution time is spent on routines in which many instructions are executed repeatedly. These instructions may constitute a simple loop, nested loops, or a few procedures that repeatedly call each other
- Memory instructions in localized areas of the program are executed repeatedly during some time period, and the remainder of the program is accessed relatively infrequently. This is referred to as *locality of reference*

Temporal and Spatial Localities

- The temporal aspect of the locality of reference suggests that whenever an information item (instruction or data) is first needed, this item should be brought into the cache where it will hopefully remain until it is needed again
- The spatial aspect of the locality of reference suggests that instead of fetching just one item from the main memory to the cache, it is useful to fetch several items that reside at adjacent addresses as well. We will use the term *block* to refer to a set of continuous address locations of some size. Another item that is often used to refer to a cache block is *cache line*

Use of a Cache Memory

- Consider the simple arrangement shown below



- Usually, the cache memory can store a reasonable number of blocks at any given time, but this number is small compared to the total number of blocks in the main memory
- The correspondence between the main memory blocks and those in the cache is specified by a mapping function

Cache Replacement Algorithm

- When the cache is full and a memory word that is not in the cache is referenced, the cache control hardware must decide which block should be removed to create space for the new block that contains the referenced word
- The collection of rules for making this decision constitutes the *replacement algorithm*

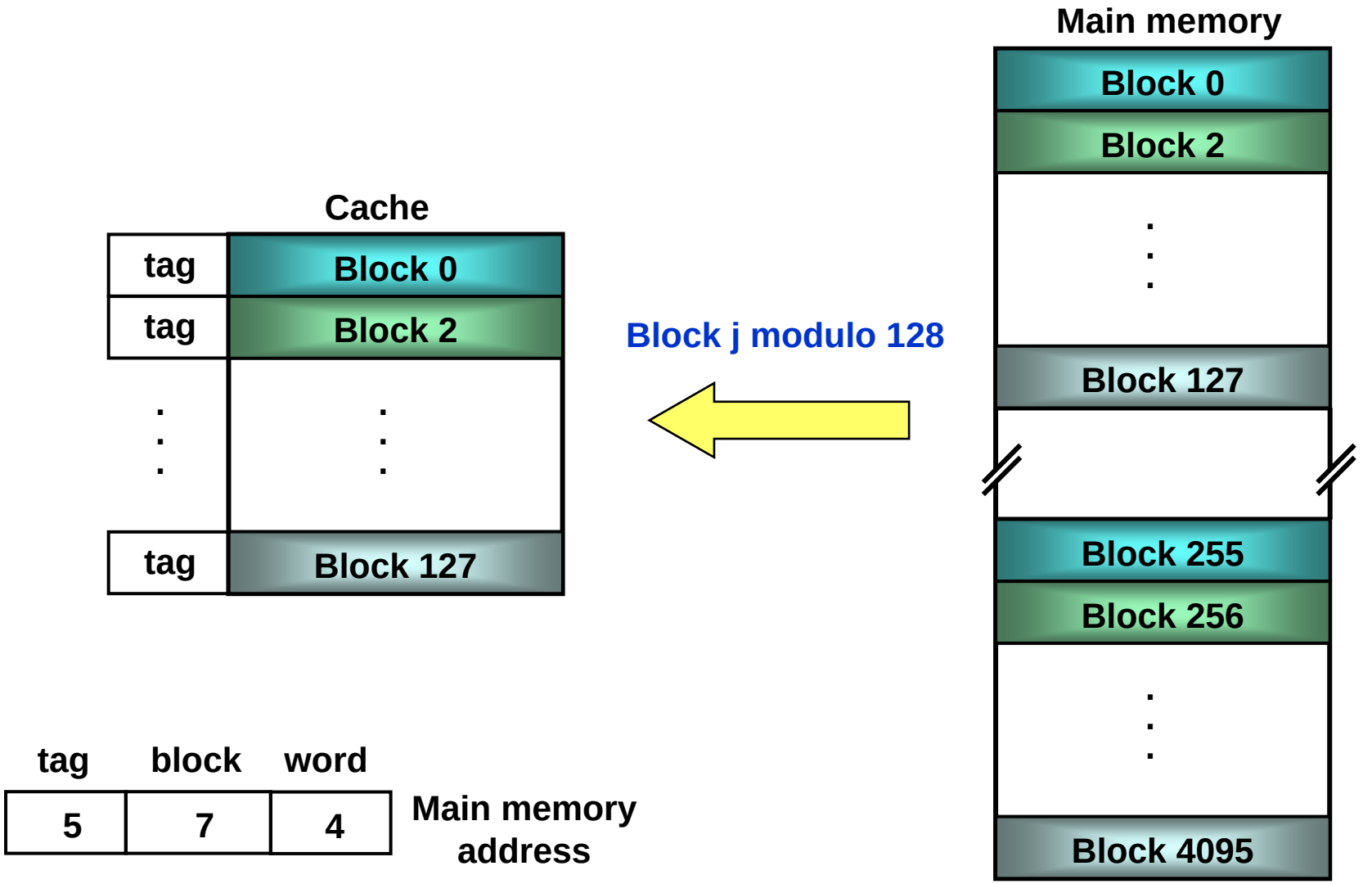
Hits & Misses

- Read hits
  - this is what we want
- Read misses
  - stall the CPU, fetch block from memory, deliver to cache, restart
- Write hits:
  - can replace data in cache and memory (write-through)
  - write the data only into the cache (write-back the cache later)
- Write misses:
  - read the entire block into the cache, then write the word

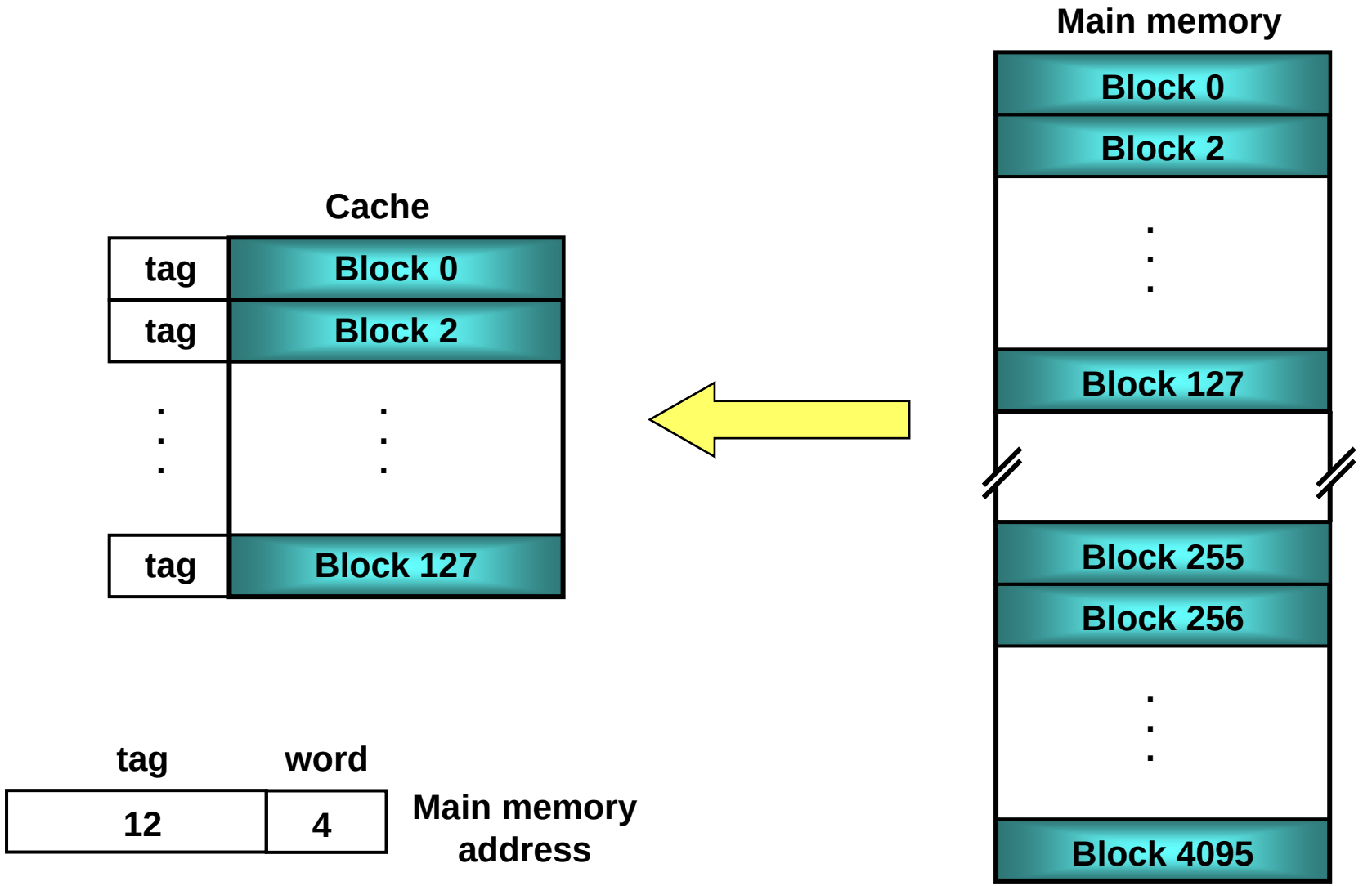
Mapping Functions

- To discuss possible methods for specifying where memory blocks are placed in the cache, we use a specific small example
- Consider a cache consisting of 128 blocks of 16 words each, for a total of 2048 (2K) words, and assume that the main memory is addressable by a 16-bit address. The main memory has 64K words, which we will view as 4K blocks of 16 words each
- Cache mapping functions
  - Direct mapping
  - Associative mapping
  - Set-associative mapping

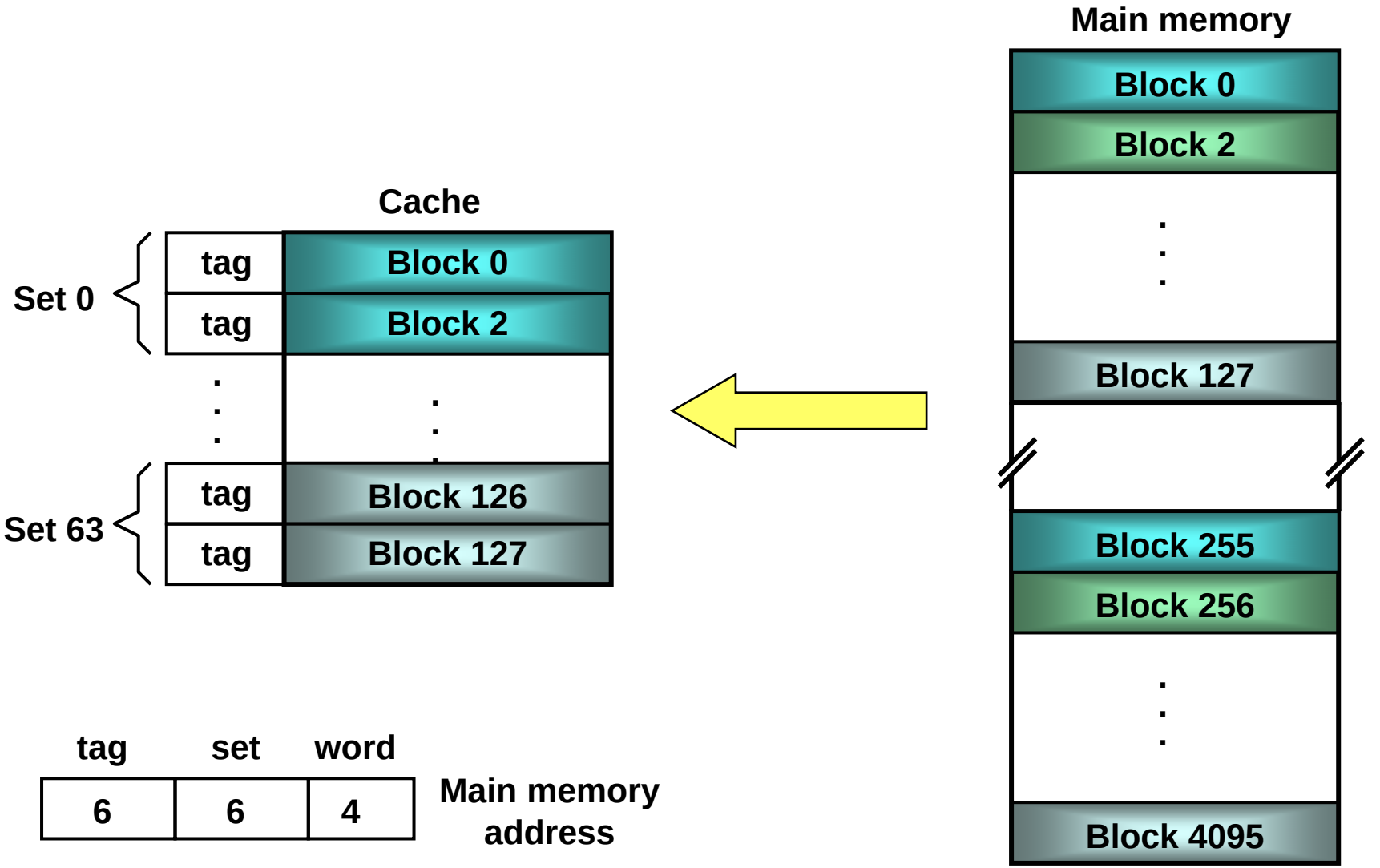
Direct-Mapped Cache



Associative-Mapped Cache



Set-Associative-Mapped Cache

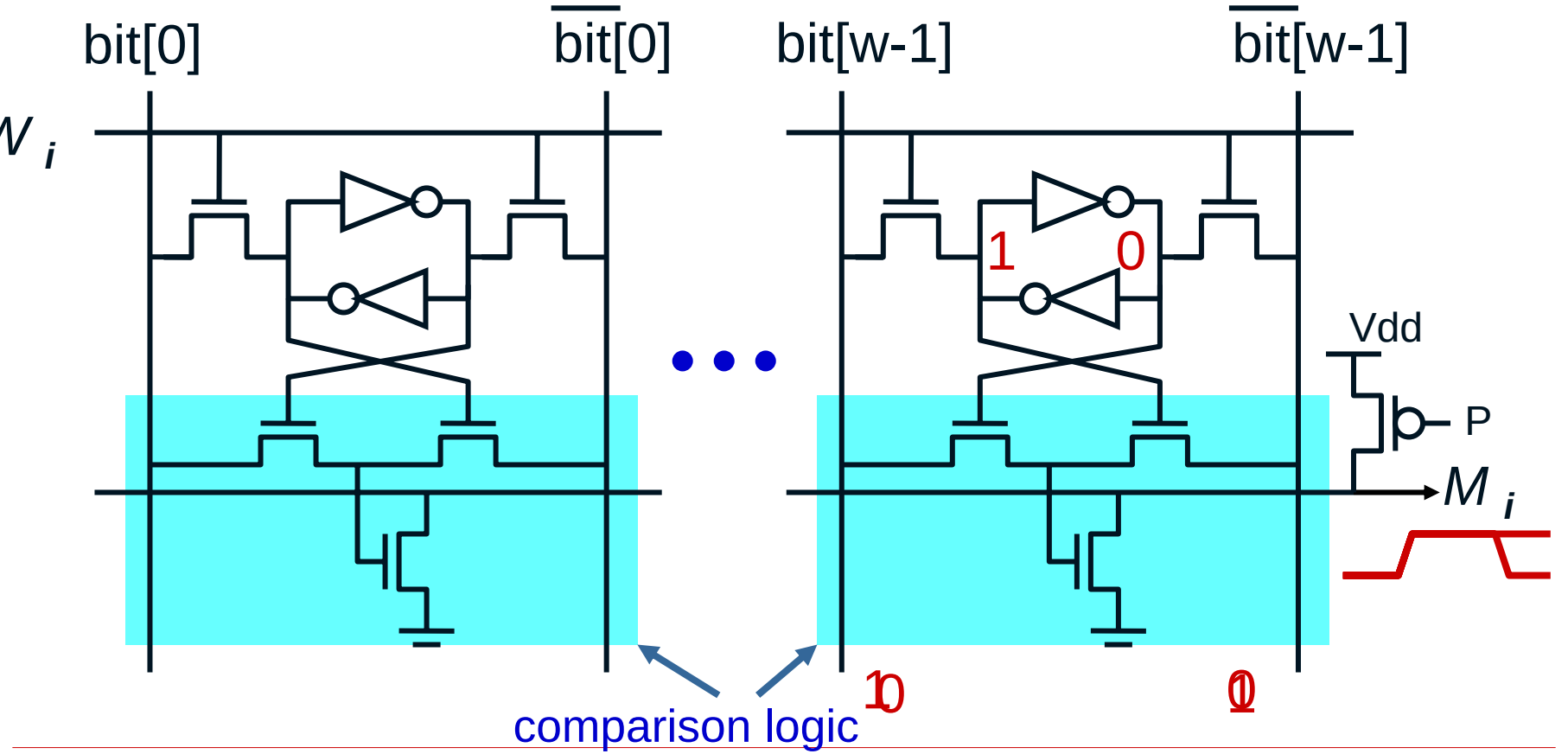


K-Way Set-Associative Cache

- The number of blocks per set is a parameter that can be selected to suit the requirements of a particular computer
- A cache that has k blocks per set is referred to as a *k-way set-associative* cache

Tag Memory Structure

- For performance reasons, the tags must be searched in parallel
- An example of the structure of a tag word



Comparison of Different Mapping Scheme

- Direct-mapped cache
  - Cost: low
  - Flexibility: low
- Associative-mapped cache
  - Cost: high
  - Flexibility: high
- Set-associative-mapped cache
  - Cost: medium
  - Flexibility: medium

Replacement Algorithms

- In a direct-mapped cache, the position of each block is predetermined; hence, no replacement strategy exists
- In associative and set-associative caches there exists some flexibility. When a new block is to be brought into the cache and all the positions that it may occupy are full, the cache controller must decide which of the old blocks to overwrite. This is an important issue because the decision can be a strong determining factor in system performance
- Thus efficient replacement algorithms are needed. In general, the objective is to keep blocks in the cache that are likely to be referenced in the near future

LRU Replacement Algorithm

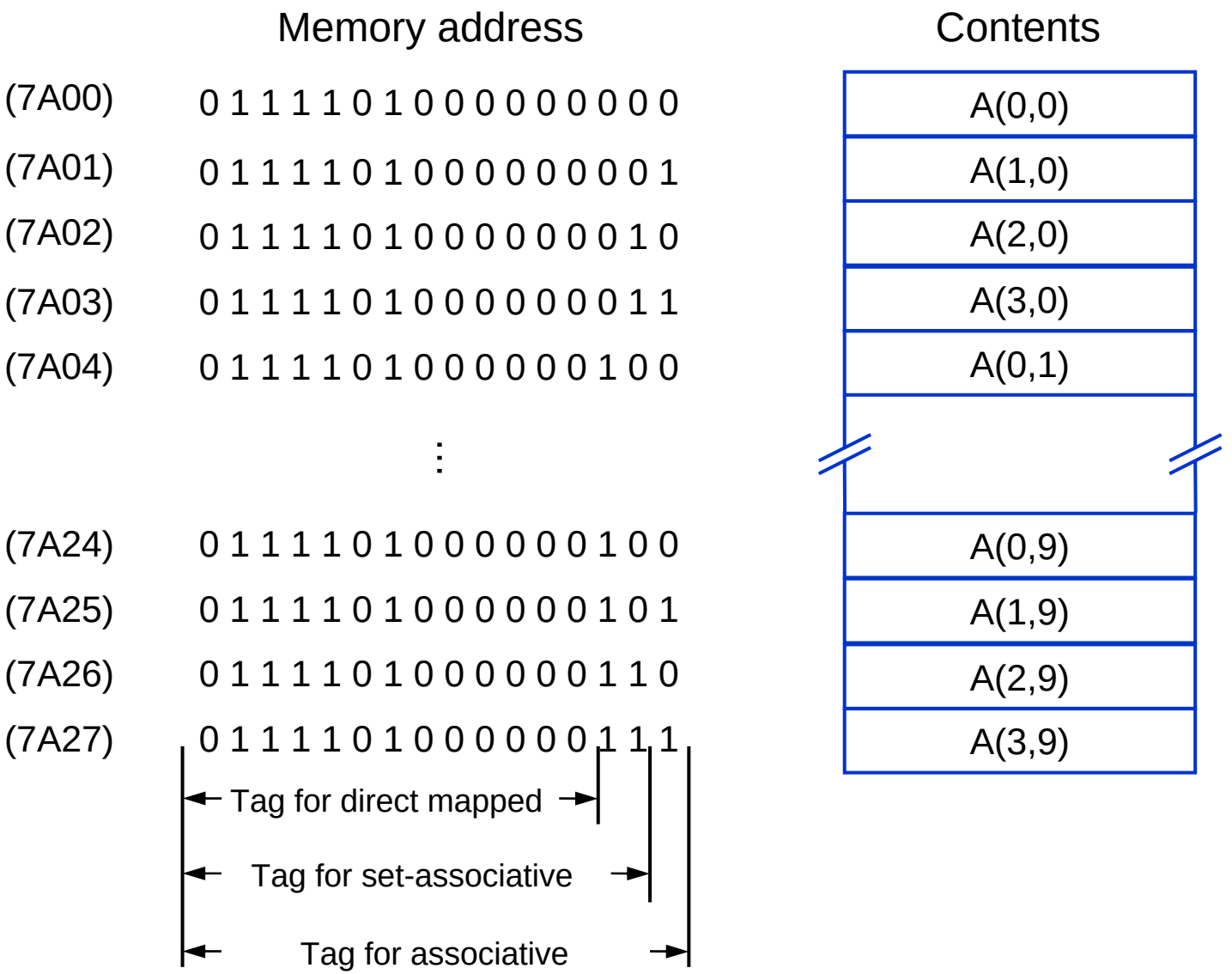
- Because programs usually stay in localized areas for reasonable periods of time, there is a high probability that the blocks that have been referenced recently will be referenced again soon
- Therefore, when a block is to be overwritten, it is sensible to overwrite the one that has gone the longest time without being referenced. This block is called the least recently used (LRU) block, and the technique is called the LRU replacement algorithm

Example of Mapping Techniques

- Assume the data cache has space for only eight blocks of data. Also assume that each block consists of only one 16-bit word of data and the memory is word addressable with 16-bit addresses
- Assume the LRU replacement algorithm is used for block replacement in the cache
- Let the following program be run and a 4x10 array of number is stored in main memory locations 7A00 through 7A27

```
SUM:=0
for j:=0 to 9 do
      SUM:=SUM+A(0,j)
end
AVE:=SUM/10
for i:=9 downto 0 do
       A(0,i):=A(0,i)/AVE
end
```

Data Stored in the Main Memory



Contents of a Direct-Mapped Data Cache

Block position	Contents of data cache after pass:								
	j=1	j=3	j=5	j=7	j=9	j=6	j=4	j=2	j=0
0	A(0,0)	A(0,2)	A(0,4)	A(0,6)	A(0,8)	A(0,6)	A(0,4)	A(0,2)	A(0,0)
1									
2									
3									
4	A(0,1)	A(0,3)	A(0,5)	A(0,7)	A(0,9)	A(0,7)	A(0,5)	A(0,3)	A(0,1)
5									
6									
7									

Contents of an Associative Data Cache

Block position	Contents of data cache after pass:				
	j=7	j=8	j=9	j=1	j=0
0	A(0,0)	A(0,8)	A(0,8)	A(0,8)	A(0,0)
1	A(0,1)	A(0,1)	A(0,9)	A(0,1)	A(0,1)
2	A(0,2)	A(0,2)	A(0,2)	A(0,2)	A(0,2)
3	A(0,3)	A(0,3)	A(0,3)	A(0,3)	A(0,3)
4	A(0,4)	A(0,4)	A(0,4)	A(0,4)	A(0,4)
5	A(0,5)	A(0,5)	A(0,5)	A(0,5)	A(0,5)
6	A(0,6)	A(0,6)	A(0,6)	A(0,6)	A(0,6)
7	A(0,7)	A(0,7)	A(0,7)	A(0,7)	A(0,7)

Contents of a Set-Associative Data Cache

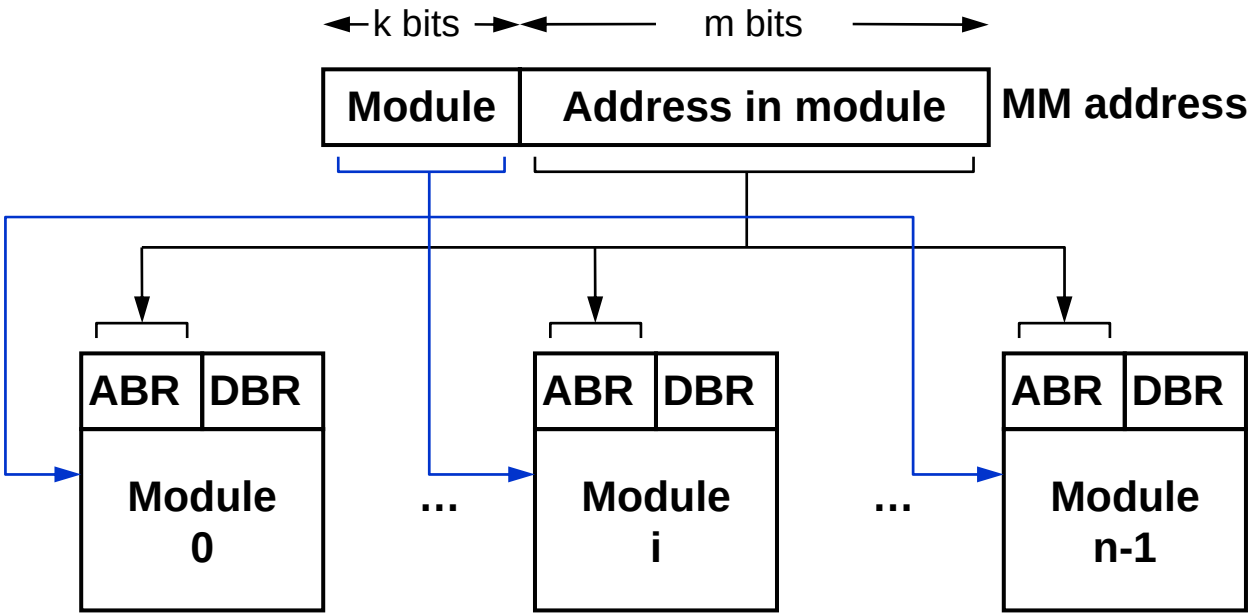
Block position	Contents of data cache after pass:						
	j=3	j=7	j=9	j=4	j=2	j=0	
0	A(0,0)	A(0,4)	A(0,8)	A(0,4)	A(0,4)	A(0,0)	Set 0
1	A(0,1)	A(0,5)	A(0,9)	A(0,5)	A(0,5)	A(0,1)	
2	A(0,2)	A(0,6)	A(0,6)	A(0,6)	A(0,2)	A(0,2)	
3	A(0,3)	A(0,7)	A(0,7)	A(0,7)	A(0,3)	A(0,3)	
0							Set 1
1							
2							
3							

Performance Considerations

- Two key factors in the commercial success of a computer are performance and cost; the best possible performance at the lowest cost is the objective
- The challenge in considering design alternatives is to improve the performance without increasing the cost
- A common measure of success is the *price/performance ratio*

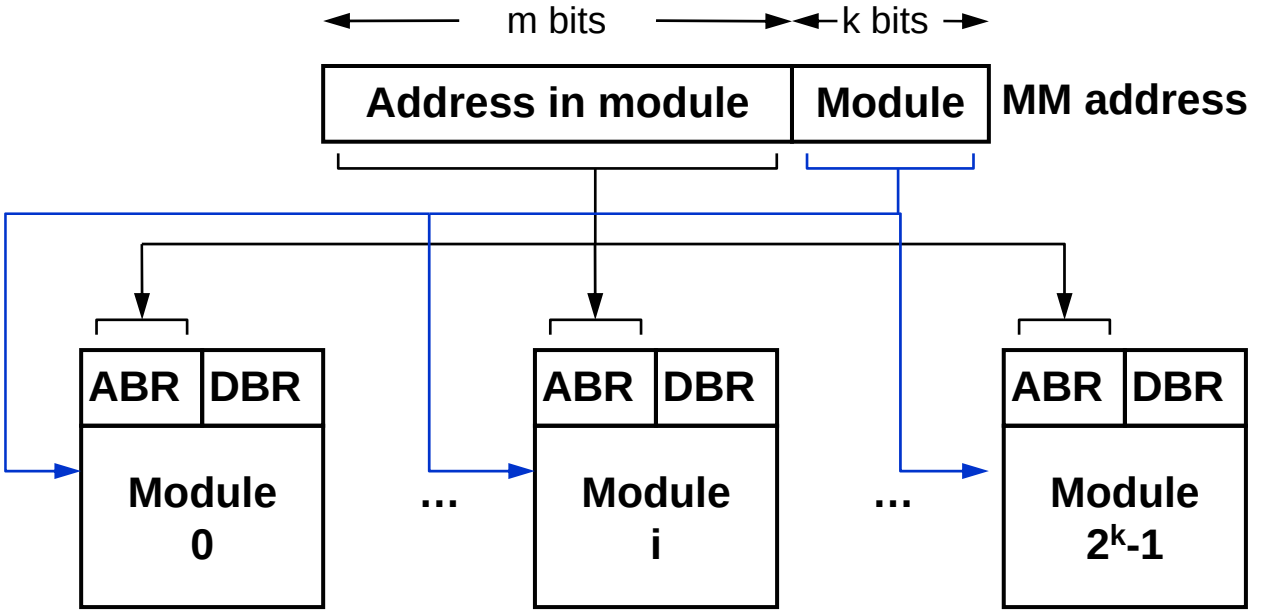
Addressing Multiple-Module Memories

- Consecutive words in a module



Addressing Multiple-Module Memories

- Consecutive words in consecutive modules



- This memory structure is called *memory interleaving*. The low-order k bits of the memory address select a module, and the high-order m bits name a location within that module.

Access Time Reduction

- Consider the time needed to transfer a block of data from the main memory to the cache when a read miss occurs.
  - Suppose that a cache with 8-word blocks is used. On a read miss, the block that contains the desired word must be copies from the memory into the cache
  - Assume that the hardware has the following properties. It takes one clock cycle to send an address to the main memory. The main memory allows the first word to be accessed in 8 cycles, but subsequent words of the block area accessed in 4 cycles per word
  - Also, one clock cycle is needed to send one word to the cache
- If a single memory module is used, then the time needed to load the desired block into the cache is
  - 1+8+(7x4)+1=38 cycles

Access Time Reduction

- Suppose now that the memory is constructed as four interleaved modules, using the scheme shown above.
- When the starting address of the block arrives at the memory, all four modules begin accessing the required data, using the high-order bits of the address. After 8 cycles, each module has one word of data in its data buffer register (DBR). These words are transferred to the cache, one word at a time, during the next 4 cycles. During this time, the next word in each module is accessed. Then it takes another 4 cycles to transfer these words to the cache
- Therefore, the total time needed to load the block from the interleaved memory is
  - 1+8+4+4=17 cycles

Hit Rate and Miss Penalty

- A successful access to data in a cache is called a hit. The number of hits stated as a fraction of all attempted accesses is called the hit rate, and the miss rate is the number of misses stated as a fraction of attempted accesses
- High hit rates, well over 0.9, are essential for high-performance computers
- Performance is adversely affected by the actions that must be taken after a miss. The extra time needed to bring the desired information into the cache is called the *miss penalty*

Average Access Time

- Consider the impact of the cache on the overall performance of the computer. Let h be the hit rate, M the miss penalty, that is, the time to access information in the main memory, and C the time to access information in the cache. The average access time experienced by the processor is
  - $t_{ave}=hC+(1-h)M$
- Consider a high-performance processor has two levels of caches, L1 and L2. The average access time experienced by the processor is
  - $t_{ave}=h_1C_1+(1-h_1)h_2C_2+(1-h_1)(1-h_2)M$
  - Where h_1 is the hit rate in the L1 cache; h_2 is the hit rate in the L2 cache; C_1 is the time to access information in the L1 cache; C_2 is the time to access information in the L2 cache

Example

- If the computer has no cache, then, using a fast processor and a typical DRAM main memory, it takes 10 clock cycles for each memory read access. Suppose the computer has a cache that holds 8-word blocks and an interleaved main memory. Then as we shown above, 17 cycles are needed to load a block into the cache. Assume that 30% of the instructions in a typical program perform a read or a write operation, which means that there are 130 memory accesses for every 100 instructions executed. Assume that the hit rates in the cache are 0.95 for instructions and 0.9 for data. Let us further assume that the miss penalty is the same for both read and write accesses. Then a rough estimate of the improvement in performance that results from using the cache can be obtained as follows:

$$\frac{\text{Time without cache}}{\text{Time with cache}} = \frac{130 \times 10}{100(0.95 \times 1 + 0.05 \times 17) + 30(0.9 \times 1 + 0.1 \times 17)} = 5.04$$

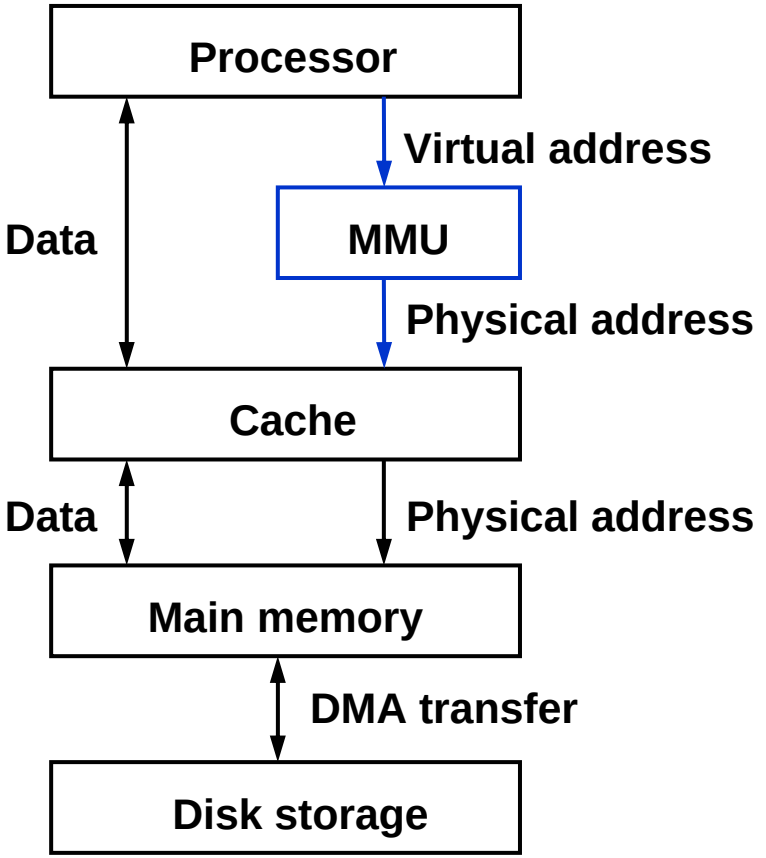
- It is also interesting to consider how effective this cache is compared to an ideal cache that has a hit rate of 100%

$$\frac{100(0.95 \times 1 + 0.05 \times 17) + 30(0.9 \times 1 + 0.1 \times 17)}{130 \times 1} = 1.98$$

Virtual Memories

- Techniques that automatically move program and data blocks into the physical main memory when they are required for execution are called *virtual-memory* techniques
- Programs, and hence the processor, reference an instruction and data space that is independent of the available physical main memory space. The binary addresses that the processor issues for either instructions or data are called *virtual or logical addresses*. These addresses are translated into physical addresses by a combination of hardware and software components

Virtual Memory Organization

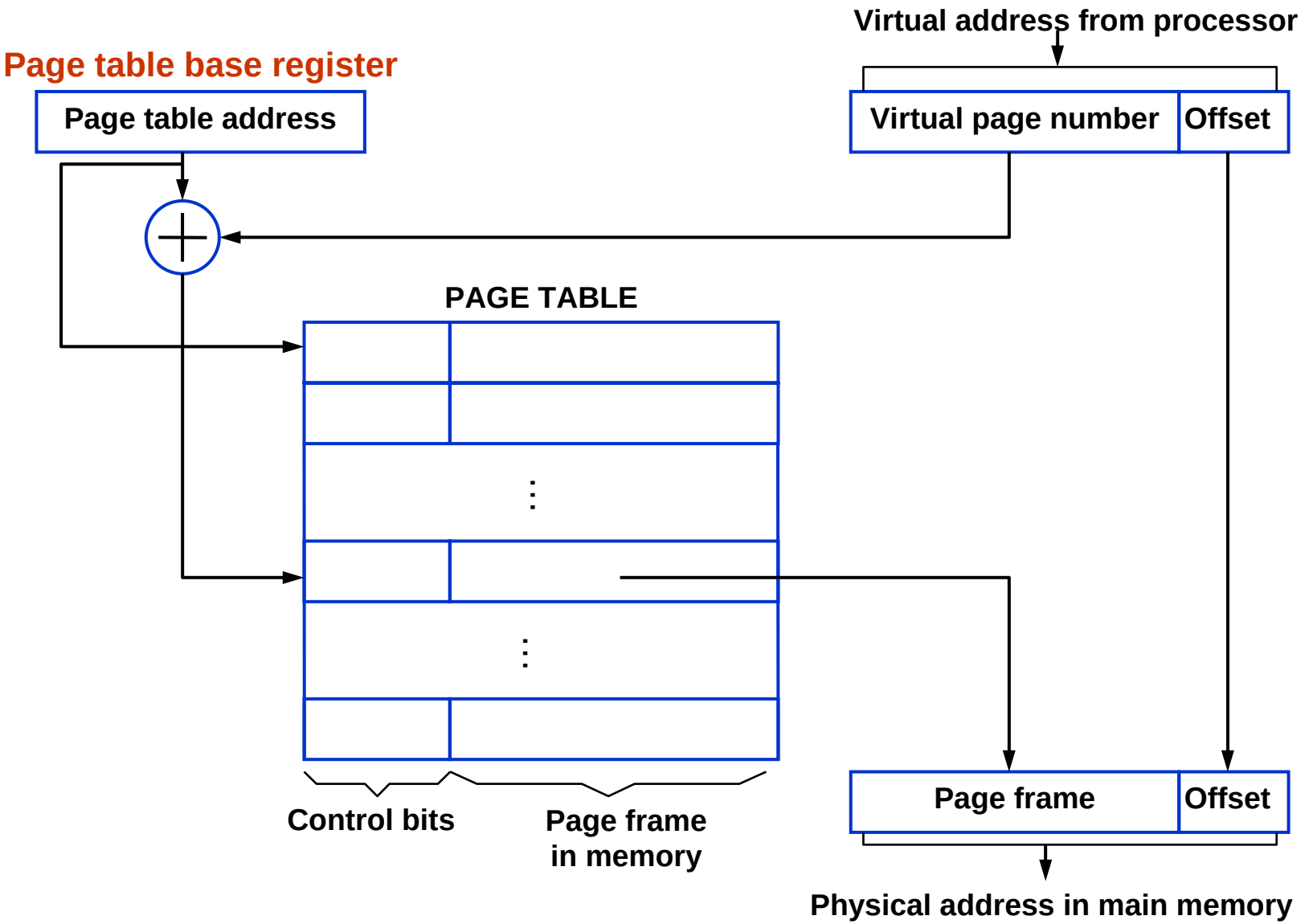


A special hardware unit, called Memory Management Unit (MMU), translates virtual addresses into physical addresses. When the desired data are in the main memory, these data are fetched as described in our presentation of the cache mechanism. If the data are not in the main memory, the MMU causes the operating system to bring the data into the memory from the disk. Transfer of data between the disk and the main memory is performed using the DMA scheme.

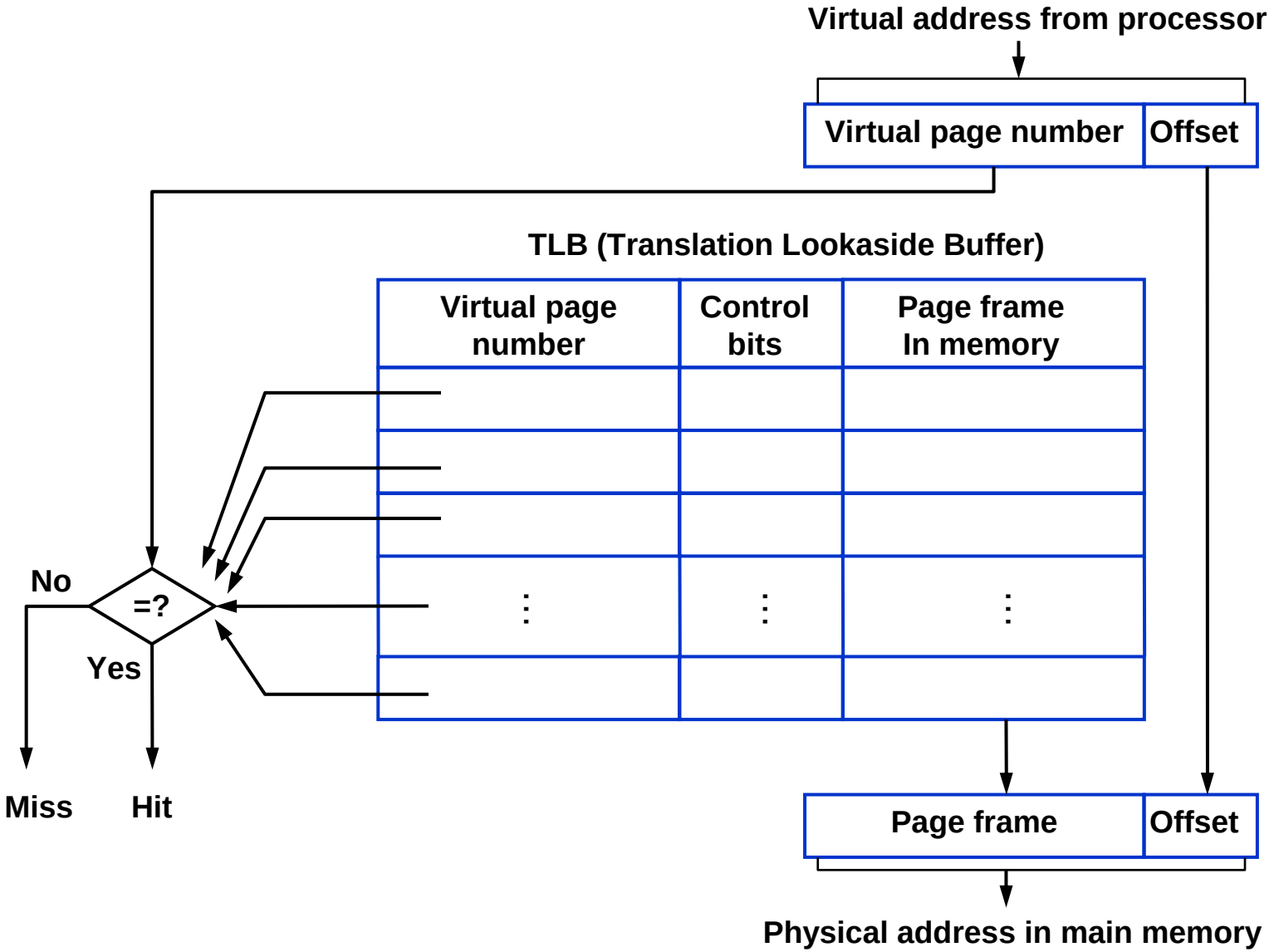
Address Translation

- A simple method for translating virtual addresses into physical addresses is to assume that all programs and data are composed of fixed-length units called pages, each of which consists of a block of words that occupy contiguous locations in the main memory.
- Conceptually, cache techniques and virtual-memory techniques are very similar. They differ mainly in the details of their implementation
  - The cache bridges the speed gap between the processor and the processor and the main memory and is implemented in hardware
  - The virtual-memory mechanism bridges the size and speed gaps between the main memory and secondary storage and is usually implemented in part by software techniques

Virtual-Memory Address Translation



Use of an Associative-Mapped TLB



Use of an Associative-Mapped TLB

- An essential requirement is that the contents of the TLB be coherent with the contents of page tables in the memory. When the operating system changes the contents of page tables, it must simultaneously invalidate the corresponding entries in the TLB.
- Address translation proceeds as follows
  - Given a virtual address, the MMU looks in the TLB for the referenced page. If the page table entry for this page is found in the TLB, the physical address is obtained immediately. If there is a miss in the TLB, then the required entry is obtained from the page table in the main memory and the TLB is updated

Page Fault

- When a program generates an access request to a page that is not in the main memory, a page fault is said to have occurred. The whole page must be brought from the disk into the memory before access can proceed
- When it detects a page fault, the MMU asks the operating system to intervene by raising an exception (interrupt). Processing of the active task is interrupted, and control is transferred to the operating system. The operating system then copies the requested page from the disk into the main memory and returns control to the interrupt task.