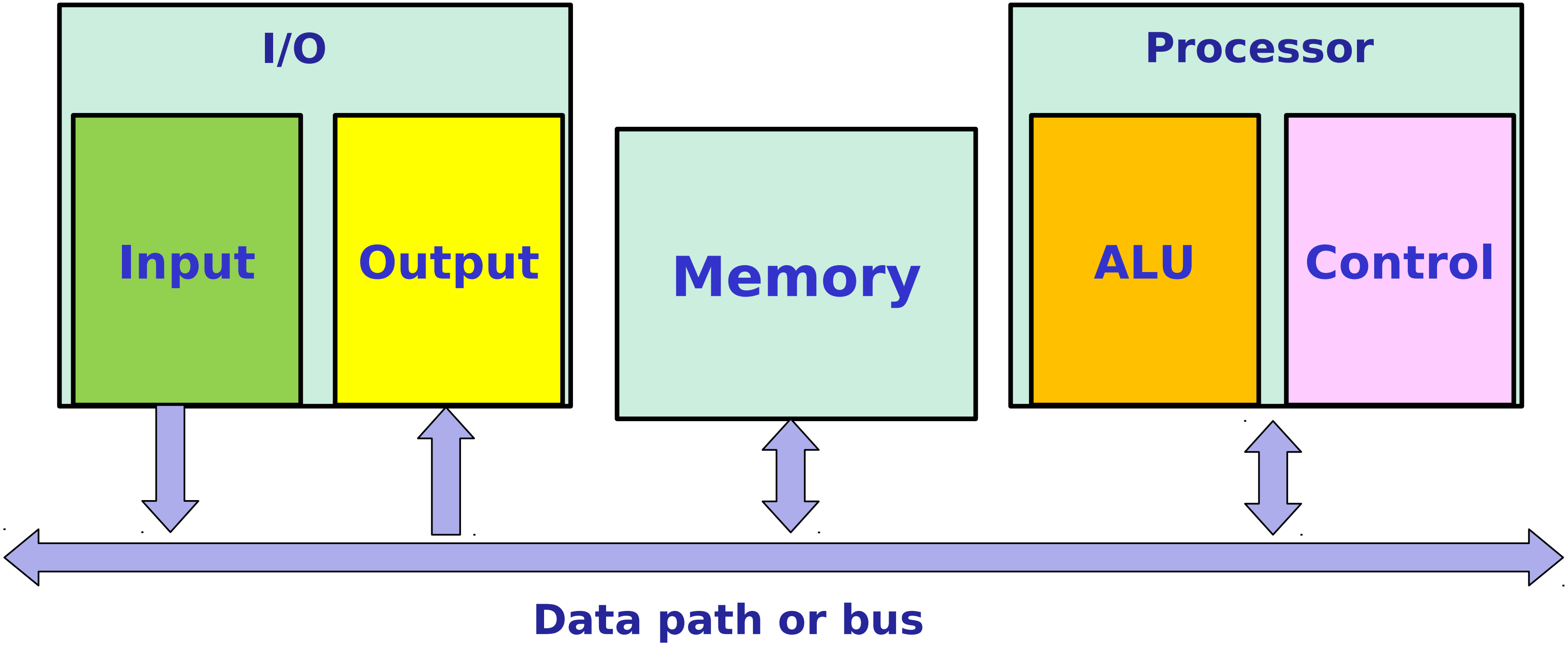
Memory Unit

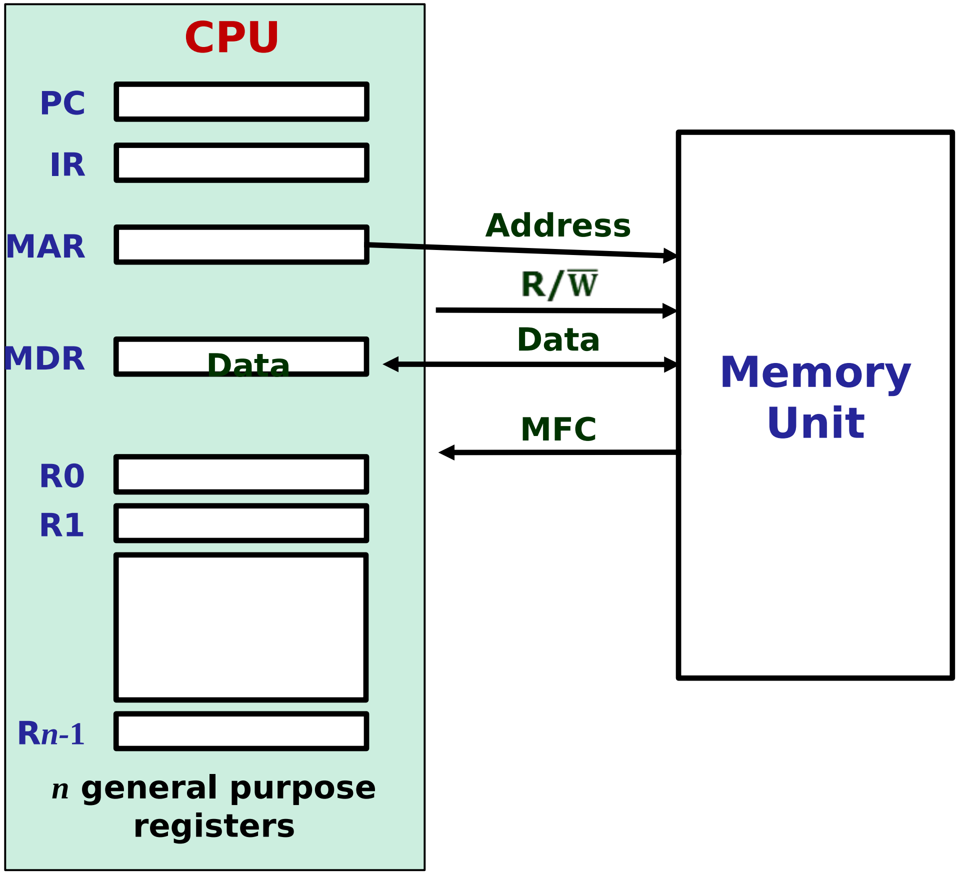
- **Functional Units**:
  - – Input Unit
  - – Memory Unit
  - – Arithmetic and Logic Unit (ALU)
  - – Output Unit
  - – Control Unit



Computer Memory

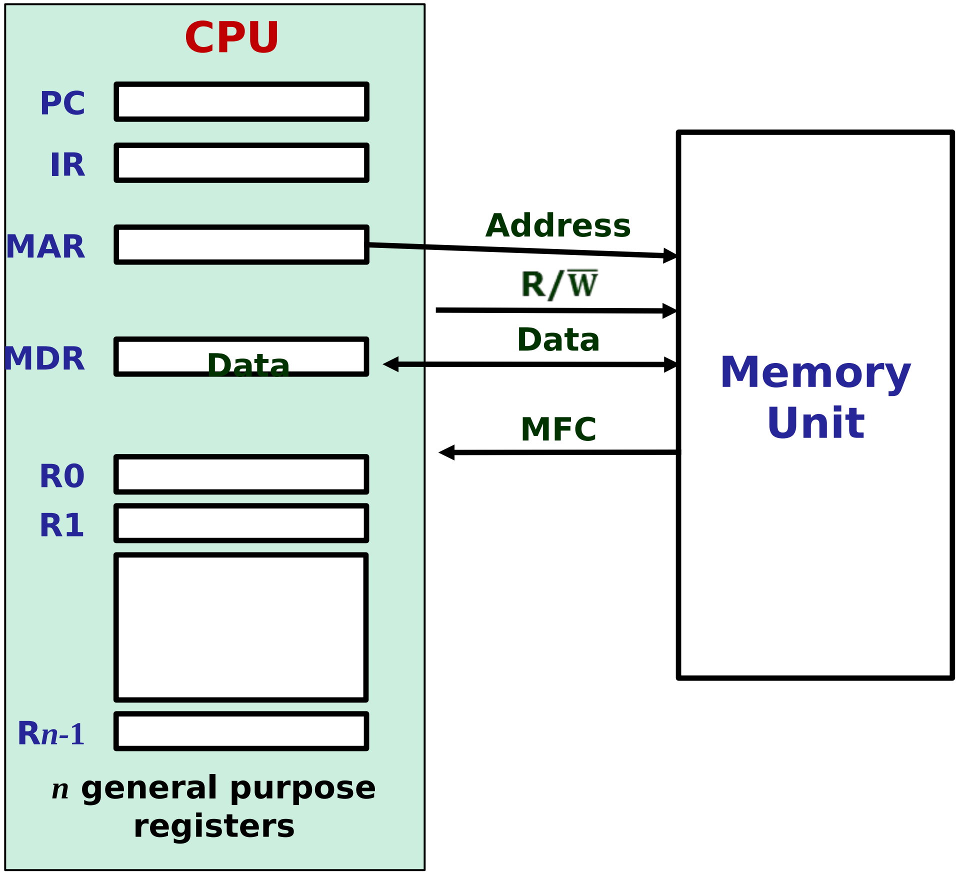
- Number & character operands, as well as instructions are stored in the memory of the computer
- Stored program concept
- CPU executes the instructions for which the instructions and operands have to come from memory unit
- Operations which involve memory:
  - Instruction fetch
    - Memory read
  - Memory operand fetch and store
    - Memory read
    - Memory write
- Instructions involving memory access:
  - LOAD and STORE instructions

Memory Read and Write Operation



- **Read**
  - Processor loads the address of memory location into MAR
  - Set the R/$\overline{W}$ line to 1
  - Memory responds by placing the data from address location onto data line
  - Confirm the action by asserting MFC (memory function complete) signal
  - Upon receiving MFC signal, processor loads the data on data line into MDR

Memory Read and Write Operation

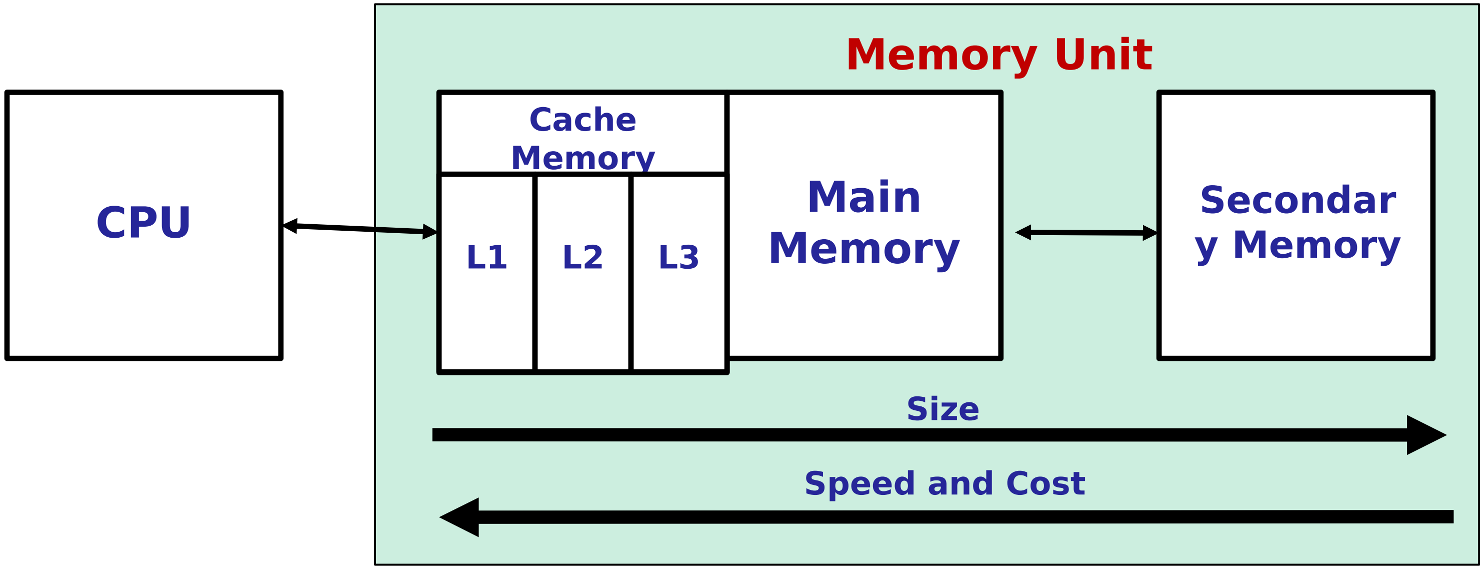


- **Write**
  - Processor loads the address of memory location into MAR
  - Processor loads data into MDR
  - Set the R/$\overline{\text{W}}$ line to 0 to indicate write operation
  - Processor places the data in MDR onto data line
  - Data on data line is written into memory location
  - Memory confirms the action by asserting MFC signal

Memory Latency and Memory Organization

- **Latency**: Time to access the first of the sequence of memory words
- What is involved in determining the latency of the memory operation?
  - Processor issues the logical address to memory unit
  - The logical address need to be converted into physical address
- Memory unit is called random access memory (RAM)
  - Any location can be accessed for read/write operation independent of the location’s address
- Memory unit is organised in hierarchical manner

Memory Hierarchy



- Processor processes instructions and data faster than it can be fetched from memory unit
- Memory access time is the bottleneck
- One way to reduce memory access time is to use faster memory
  - A small and faster memory bridge the gap between processor and main memory

Memory Performance Parameters

- **Memory Access Time**:
  - Time interval between initiation of one operation and completion of that operation
  - Example: Time between assertion of Read signal and MFC signal
- **Memory Cycle Time**:
  - Minimum time delay between the initiation of two successive memory operations
  - Time delay between start of a read/write operation to start of next memory operation
- Memory cycle time is usually slightly larger than access time

Internal Organization of Memory

- The memory is organised such that a group of n-bits can be stored or retrieved in a single basic operation
- Each group of n-bits is referred as one **memory word**
- Accessing the memory to store or retrieve information require address for each location
- Possible number of address locations are decided by the number of address lines in the processor
- For k-address lines, there will be 2^k locations, each of n-bit memory word
- 2^k addresses constitute address space of computer
- Example: Let k=10 and n=32
  - Number of locations: 2^{10}
  - Size of the memory: 2^{10} x 2^5 bits = 2^{15} bits

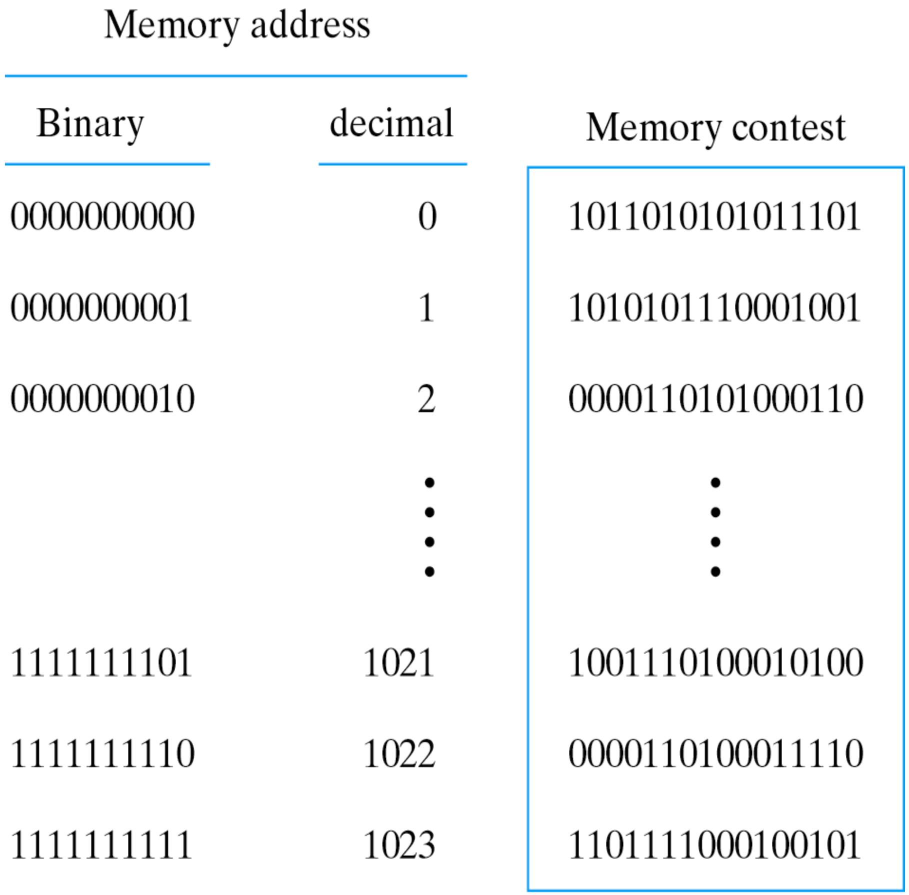
    = 2^{12} Bytes

    = 4 KB

Memory Content Example

1024 memory locations: 1x0 bit address

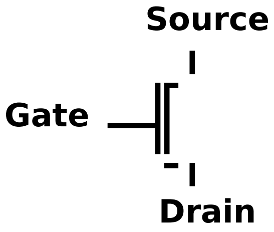
16 bit data



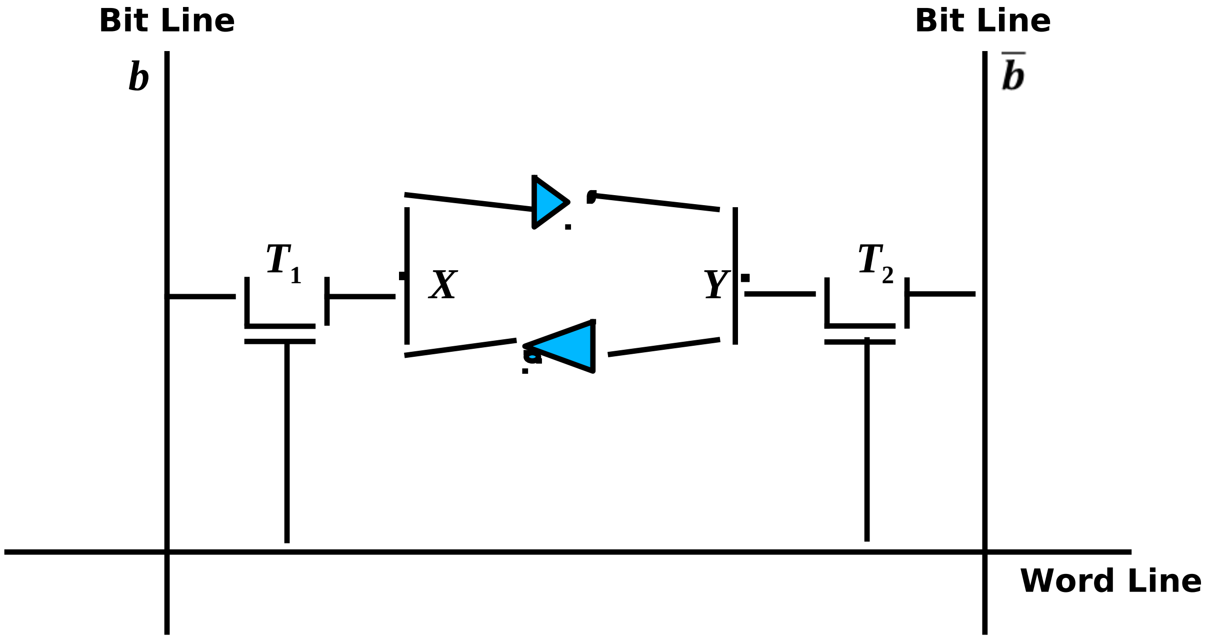
Memory address		
Binary	decimal	Memory contest
0000000000	0	1011010101011101
0000000001	1	1010101110001001
0000000010	2	0000110101000110
	⋮	⋮
1111111101	1021	1001110100010100
1111111110	1022	0000110100011110
1111111111	1023	1101111000100101

Semiconductor Memories

- Two basic ways of designing memory
  - Static RAM (SRAM)
  - Dynamic RAM (DRAM)
- **Static RAM**:
  - Built using metal-oxide semiconductor (MOS) transistors
  - MOS transistors acts as switch
    - +5 v (when Gate input is 1): Transistor conducts: ON state
    - 0 v (when Gate input is 0): Transistor does not conducts: OFF state

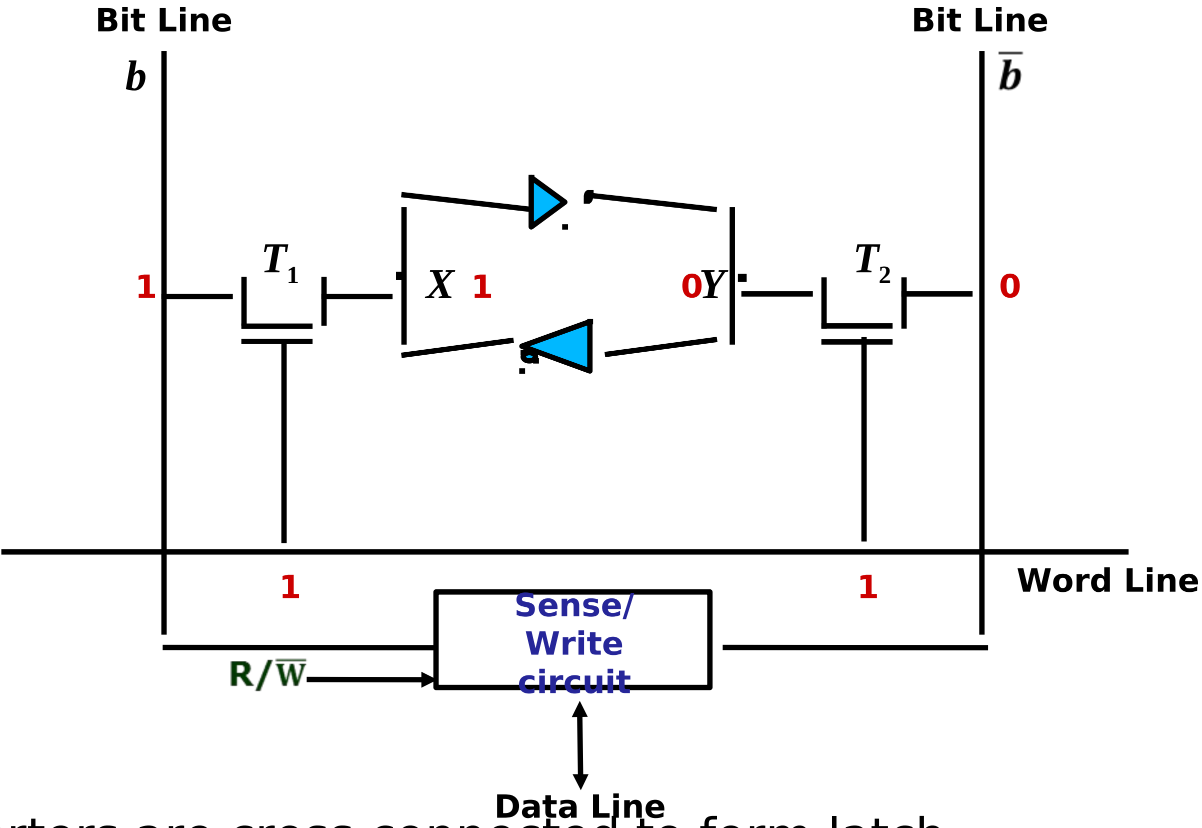


Static RAM Cell



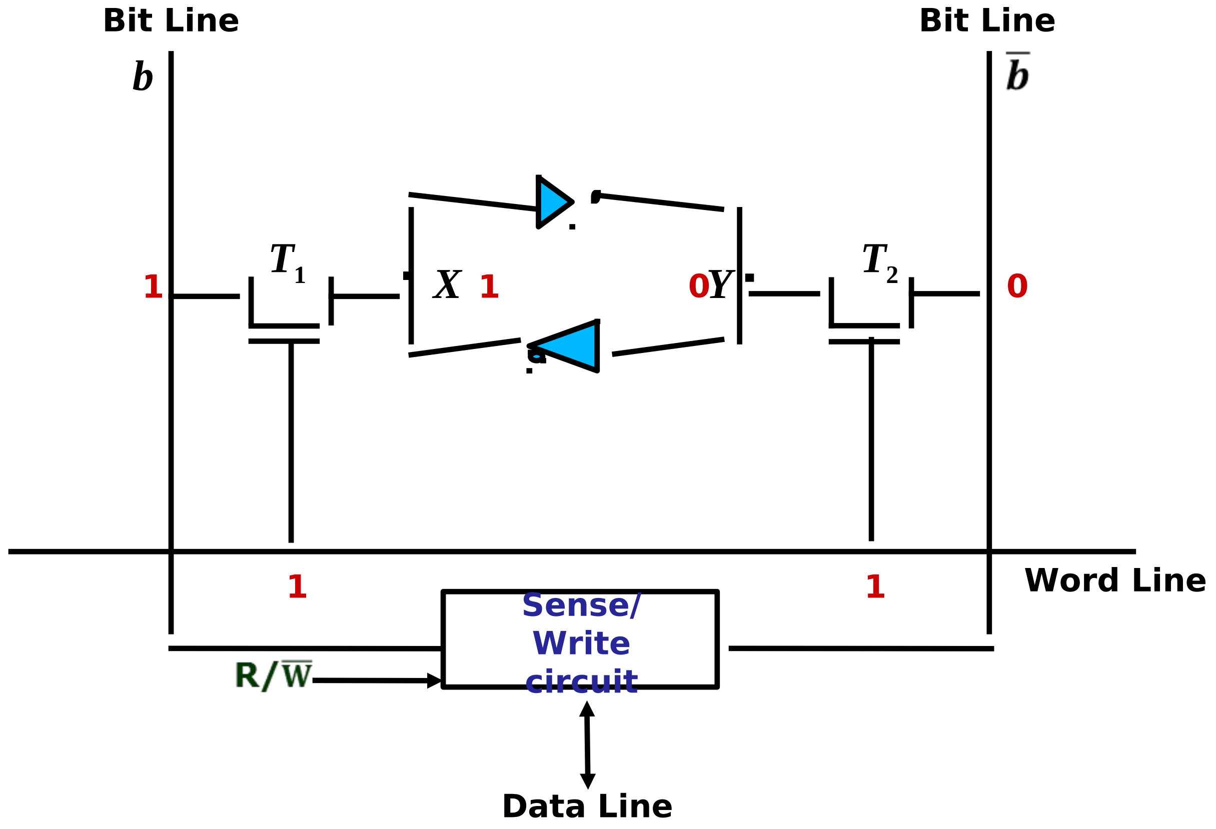
- Two inverters are cross connected to form latch
- Inverters are connected to 2 transistors which act as switches
- Switches are opened or closed under the control of word line
- This circuit retain the state (bit) as long as power is applied (Static Memory)

Static RAM Cell - Read



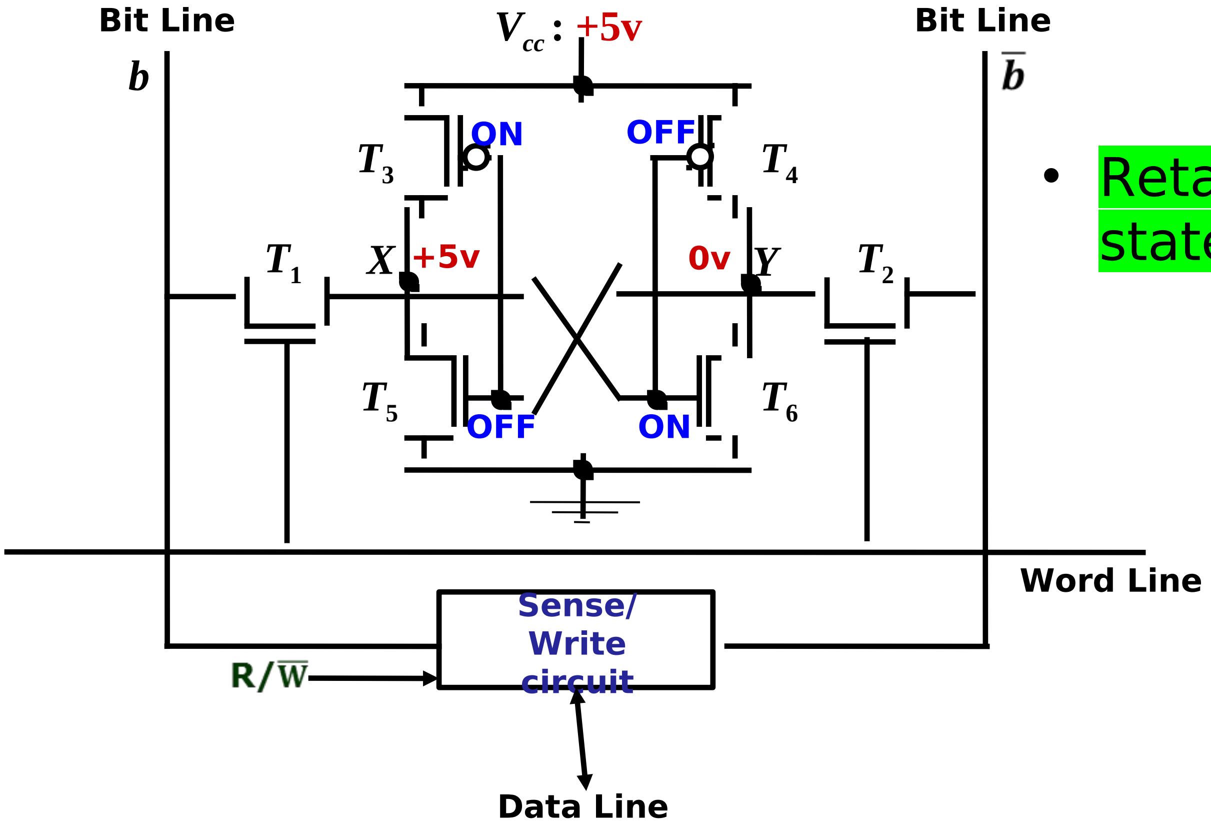
- Two inverters are cross connected to form latch
- Inverters are connected to 2 transistors which act as switches
- Switches are opened or closed under the control of word line
- This circuit retain the state (bit) as long as power is applied (Static Memory)

Static RAM Cell - Write



- Inverters are connected to 2 transistors which act as switches
- Two inverters are cross connected to form latch
- Switches are opened or closed under the control of word line
- This circuit retain the state (bit) as long as power is applied (Static Memory)

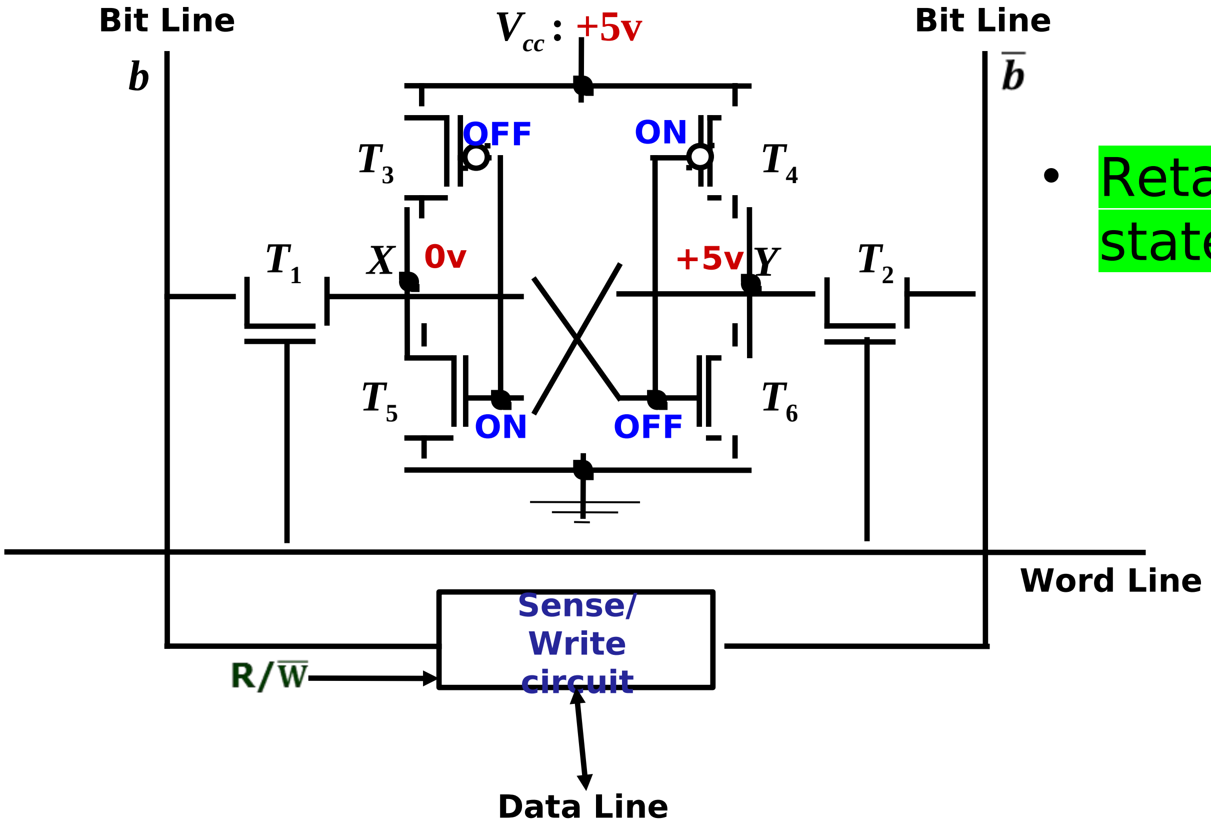
CMOS Static RAM Cell



- Retaining the state 1

- This circuit retain the state (bit) as long as power is applied (Static Memory)
- Continuous power is needed for a cell to retain the state

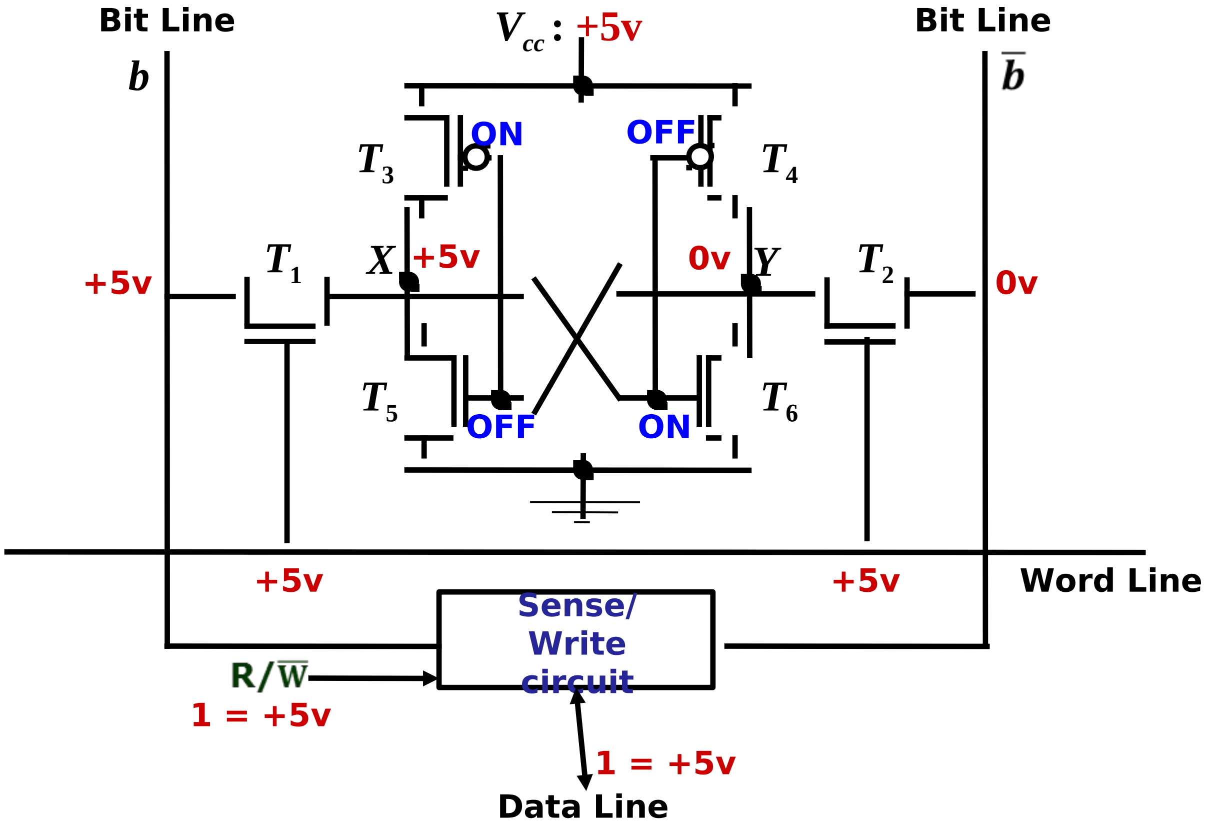
CMOS Static RAM Cell



- Retaining the state 0

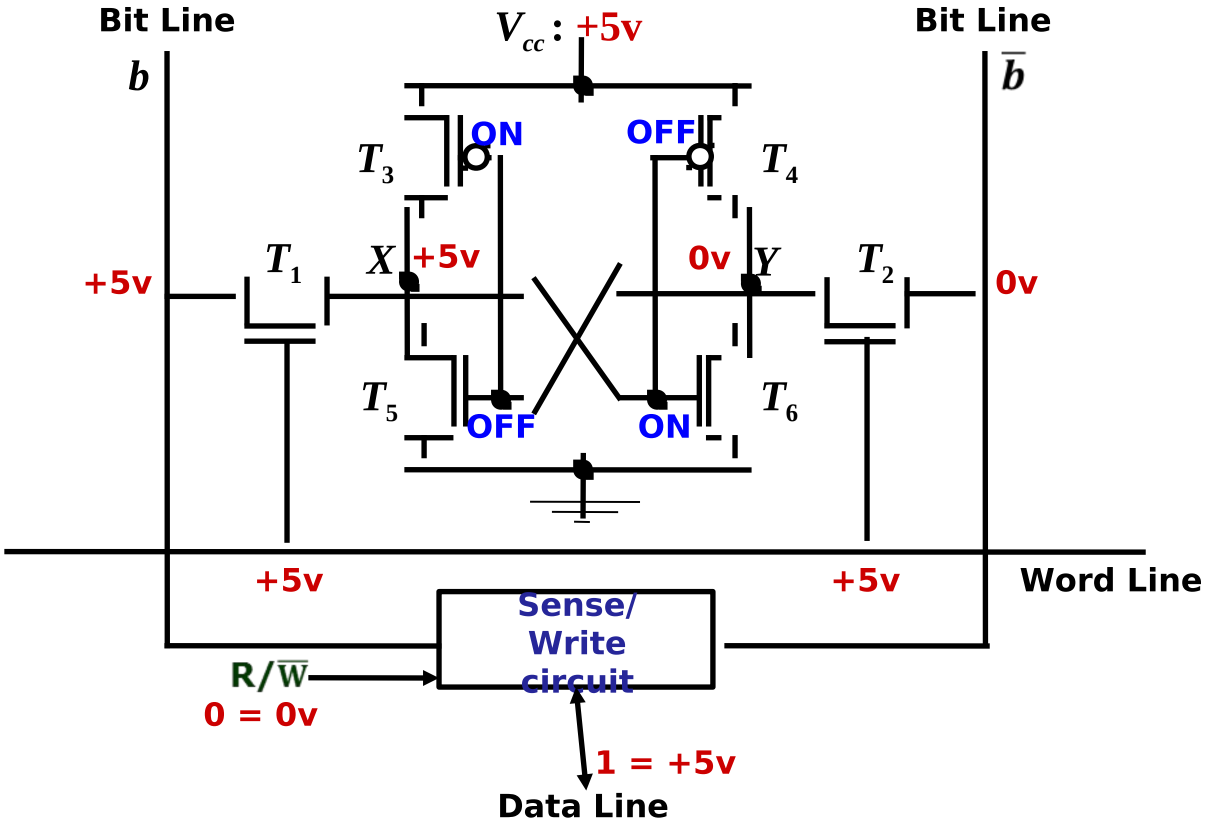
- This circuit retain the state (bit) as long as power is applied (Static Memory)
- Continuous power is needed for a cell to retain the state

CMOS Static RAM Cell - Read



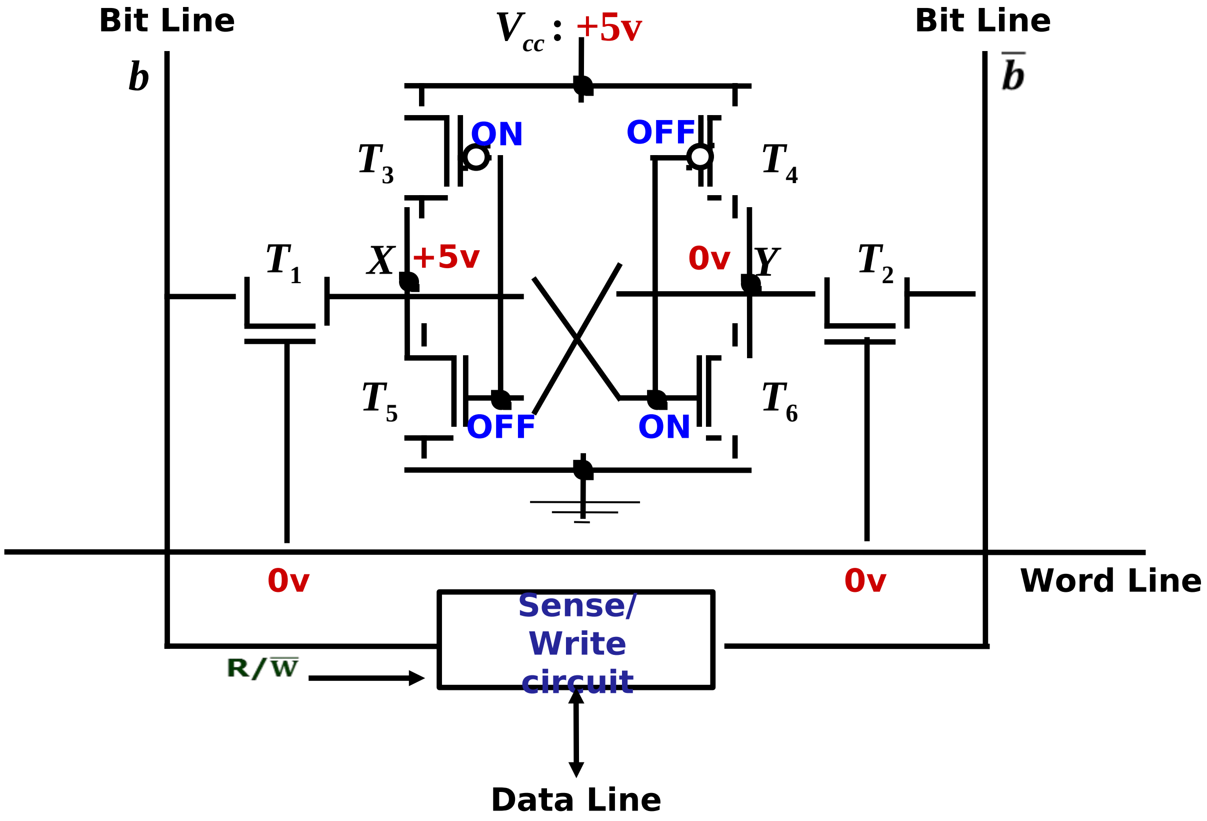
- This circuit retain the state (bit) as long as power is applied (Static Memory)
- Continuous power is needed for a cell to retain the state

CMOS Static RAM Cell - Write



- This circuit retain the state (bit) as long as power is applied (Static Memory)
- Continuous power is needed for a cell to retain the state

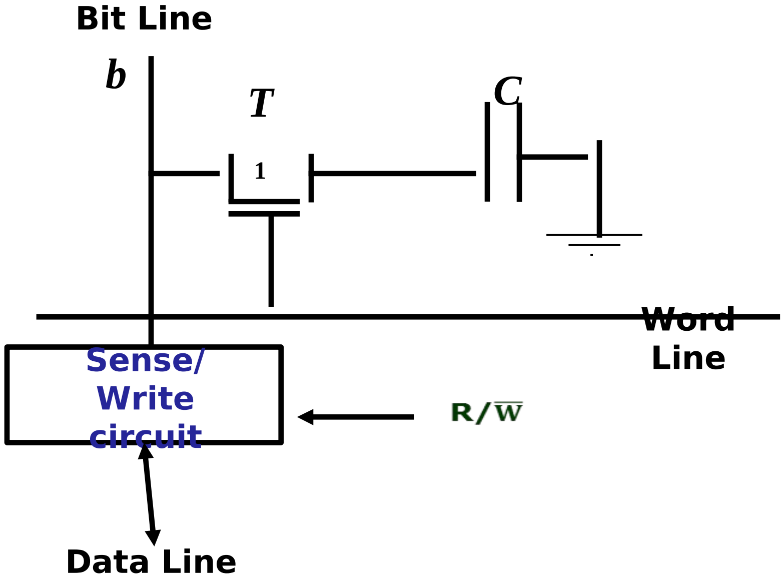
CMOS Static RAM Cell: Illustration



- Access time is less i.e. faster memory
- Low power consumption
- Uses 6 transistors: Costly
- Used in applications where speed is critical concern: Cache

Dynamic RAM (DRAM) Cell

- Less expensive and simpler cell
- Information is stored in the form of a charge on a capacitor (C)
  - Charge in capacitor is stored only for short time
  - However, a cell is required to store information for a much longer time
- To retain information for longer time, content of capacitor mush be periodically refreshed

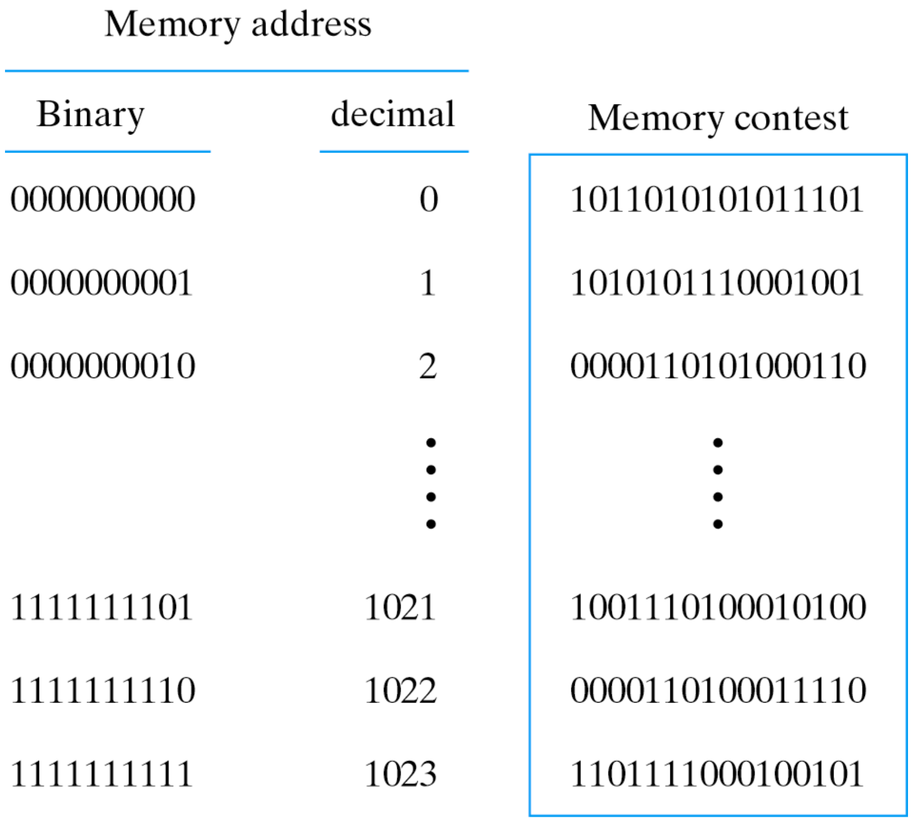


- Low speed as refresh needed
- Only 1 transistor is used
- Used to build main memory

Memory Content Example

1024 memory locations: 10 bit address

16 bit data

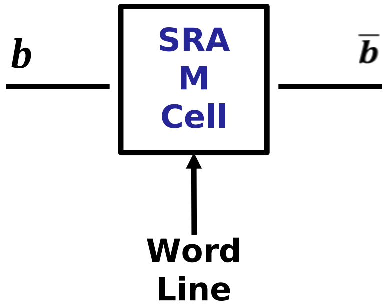


Memory address		
Binary	decimal	Memory contest
0000000000	0	1011010101011101
0000000001	1	1010101110001001
0000000010	2	0000110101000110
	⋮	⋮
1111111101	1021	1001110100010100
1111111110	1022	0000110100011110
1111111111	1023	1101111000100101

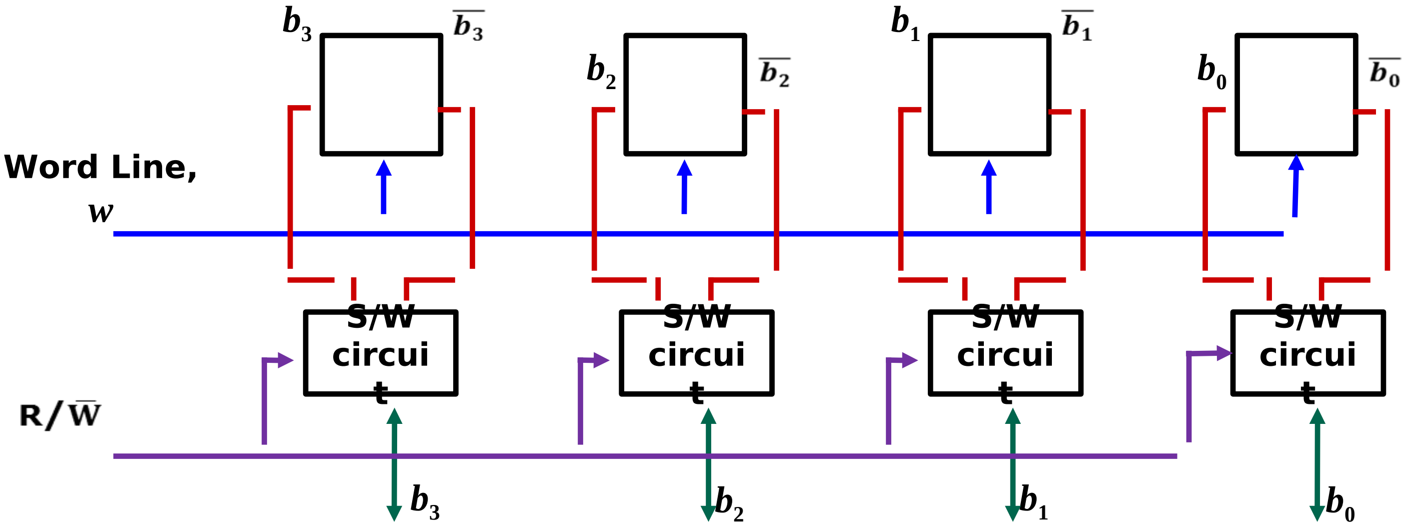
Fig. 7-3 Content of a 1024 × 16 Memory

Internal Organization of SRAM Chip

- SRAM cell (block diagram):



- Memory cells are usually organised in the form of an array
- At each memory location, n SRAM cells are placed next to each other to form n-bit memory word
- Example: 4-bit memory word



Internal Organization of 2^k x n SRAM Chip

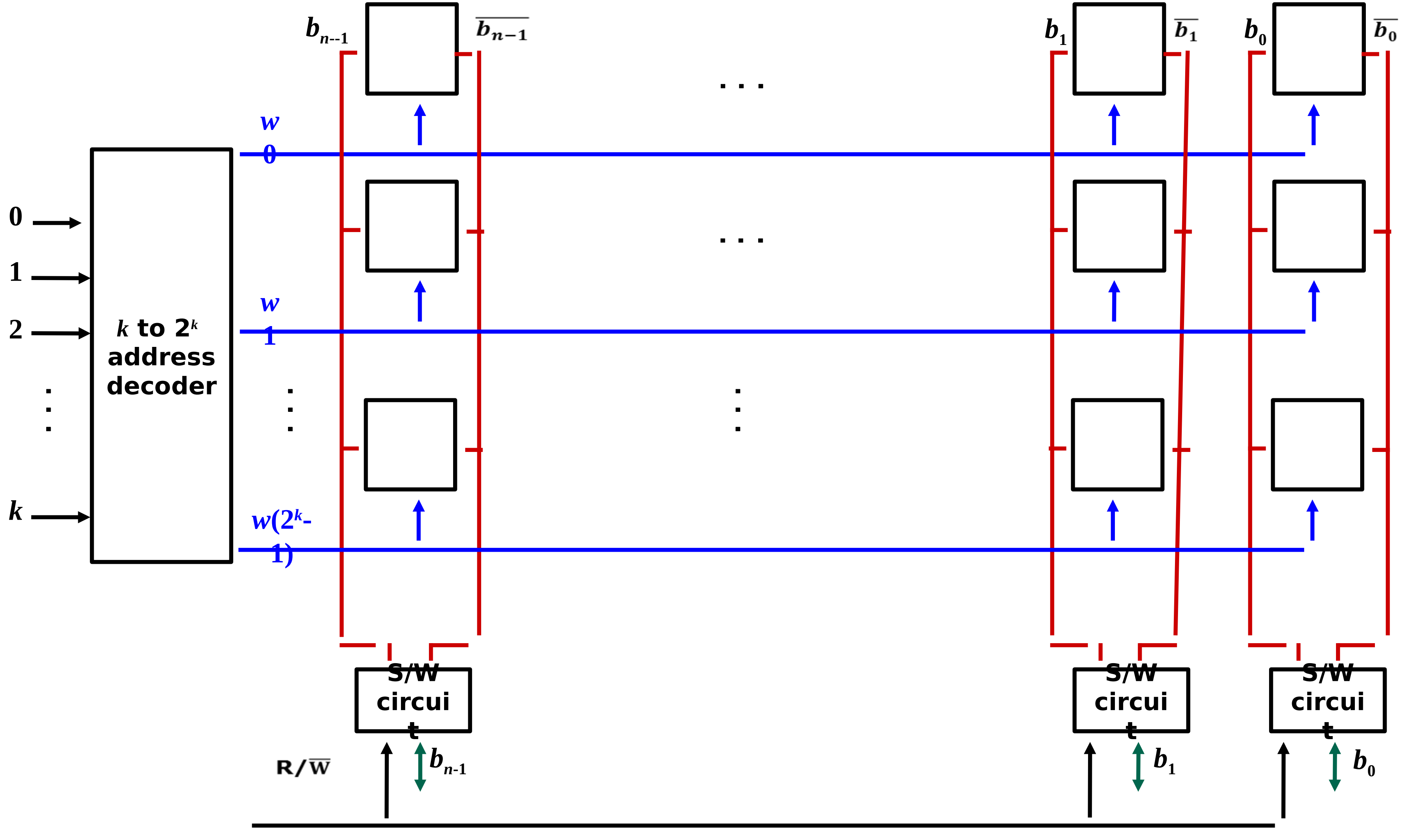


Illustration: 16 x 4 bit Memory Chip

- 1-dimensional address decoding:
  - *Example*: ***Design of*** **64b** ***memory chip***
  - Number of address line be : 4
  - Address decoder configuration : 4-to-2^4

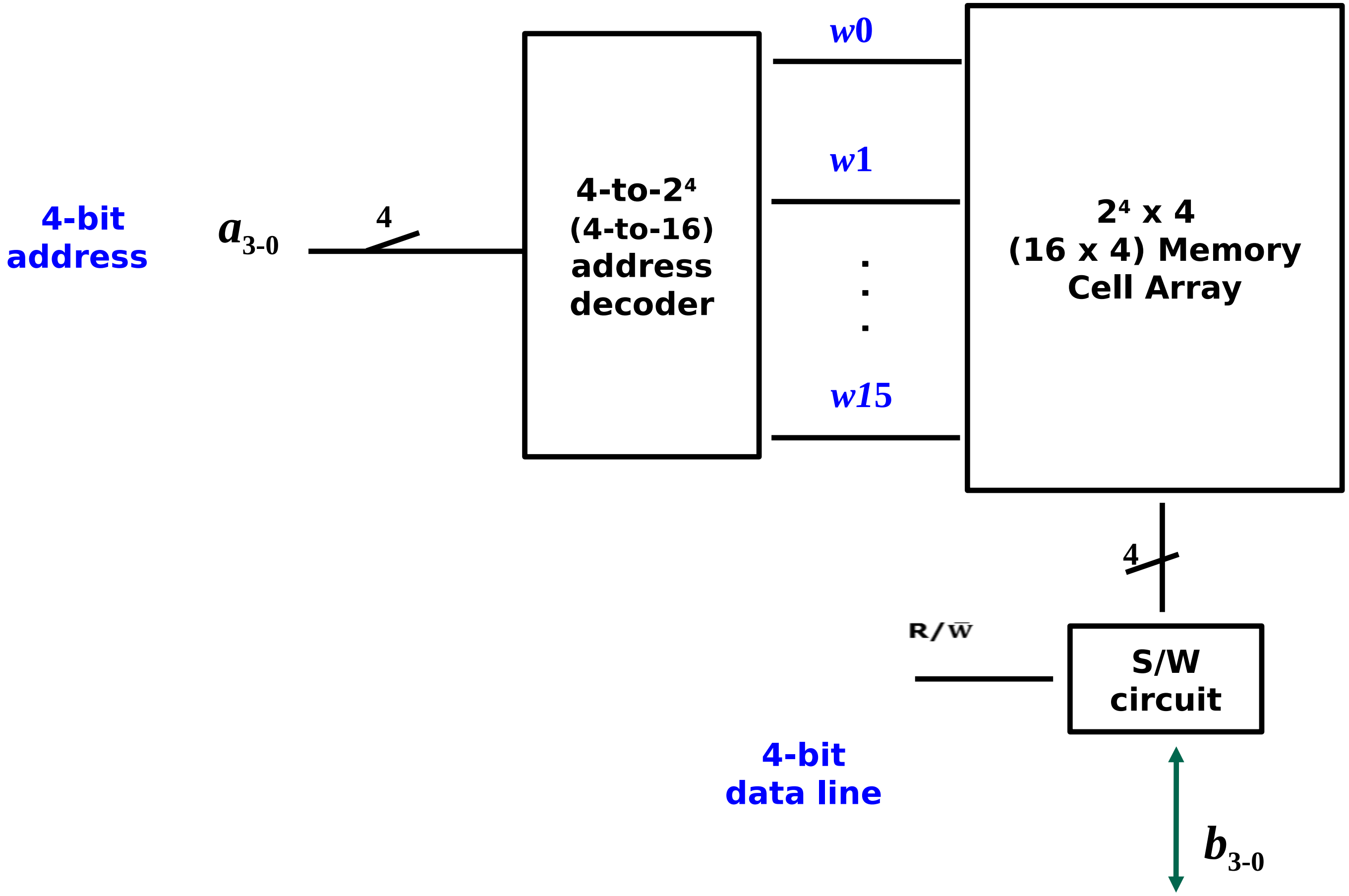


Illustration: 512 x 8 bit Memory Chip

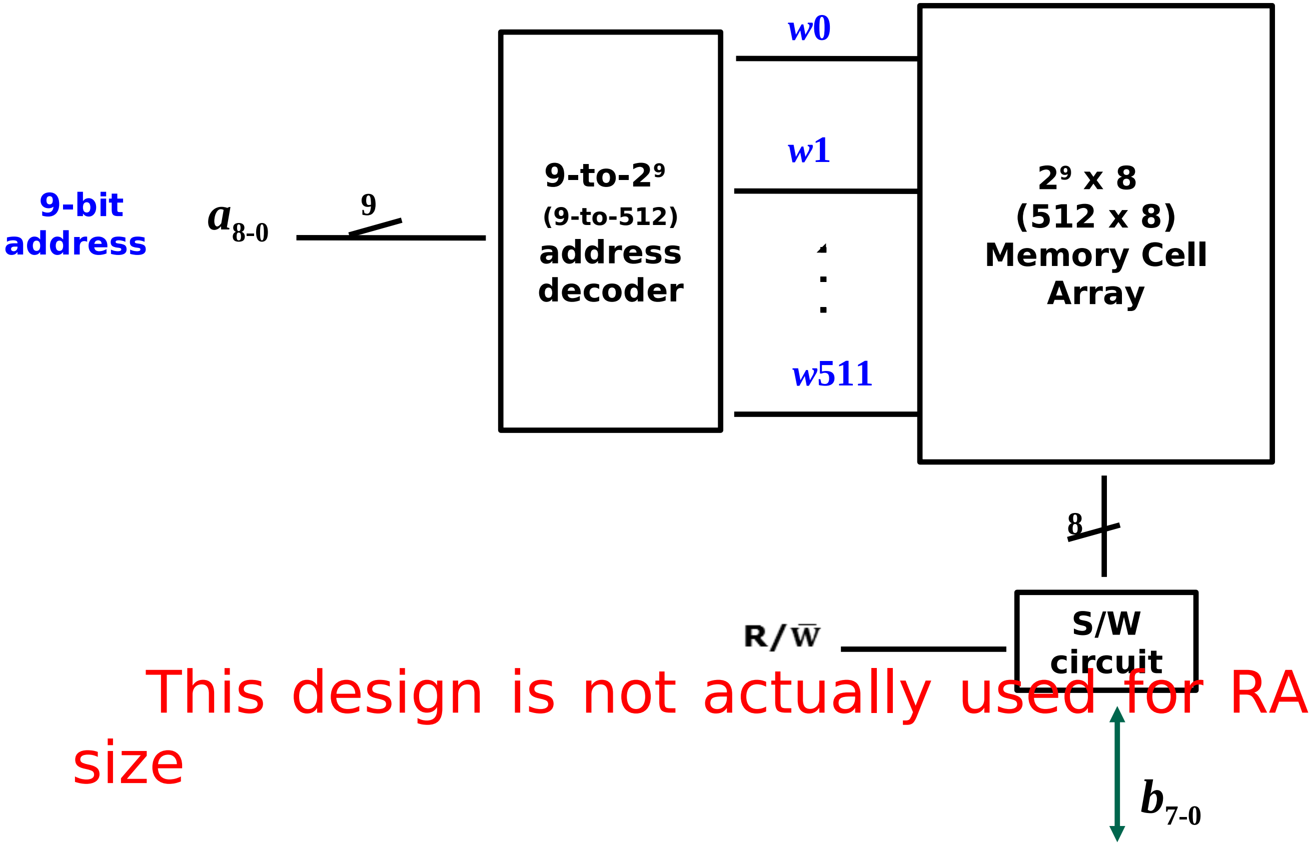
- 1-dimensional address decoding:
  - *Example*:

Memory chip organization : 512 x 8

Memory Size : 512B

Number of address line be: 9

Address decoder configuration : 9-to-2^9



9-bit address
a_{8-0}
9
9-to-2^9 (9-to-512) address decoder
$w0$
$w1$
$w511$
2^9 x 8 (512 x 8) Memory Cell Array
8
S/W circuit
$R/\overline{W}$
b_{7-0}

This design is not actually used for RAM of any size

Coincident Decoding

Regular decoding is costly:

A decoder with k inputs and 2^K outputs requires 2^K AND gates with k inputs per gate.

Total number of gates can be reduced by using two-dimensional decoding:

Basic idea: arrange memory cells in a (as close as possible to) square configuration.

Use two k/2 input decoders instead of one k input decoder

Two-Dimensional Decoding

Instead of using a single 10 x 1024 decoder
we use two 5x32 decoders.

One decoder picks the row, one the column

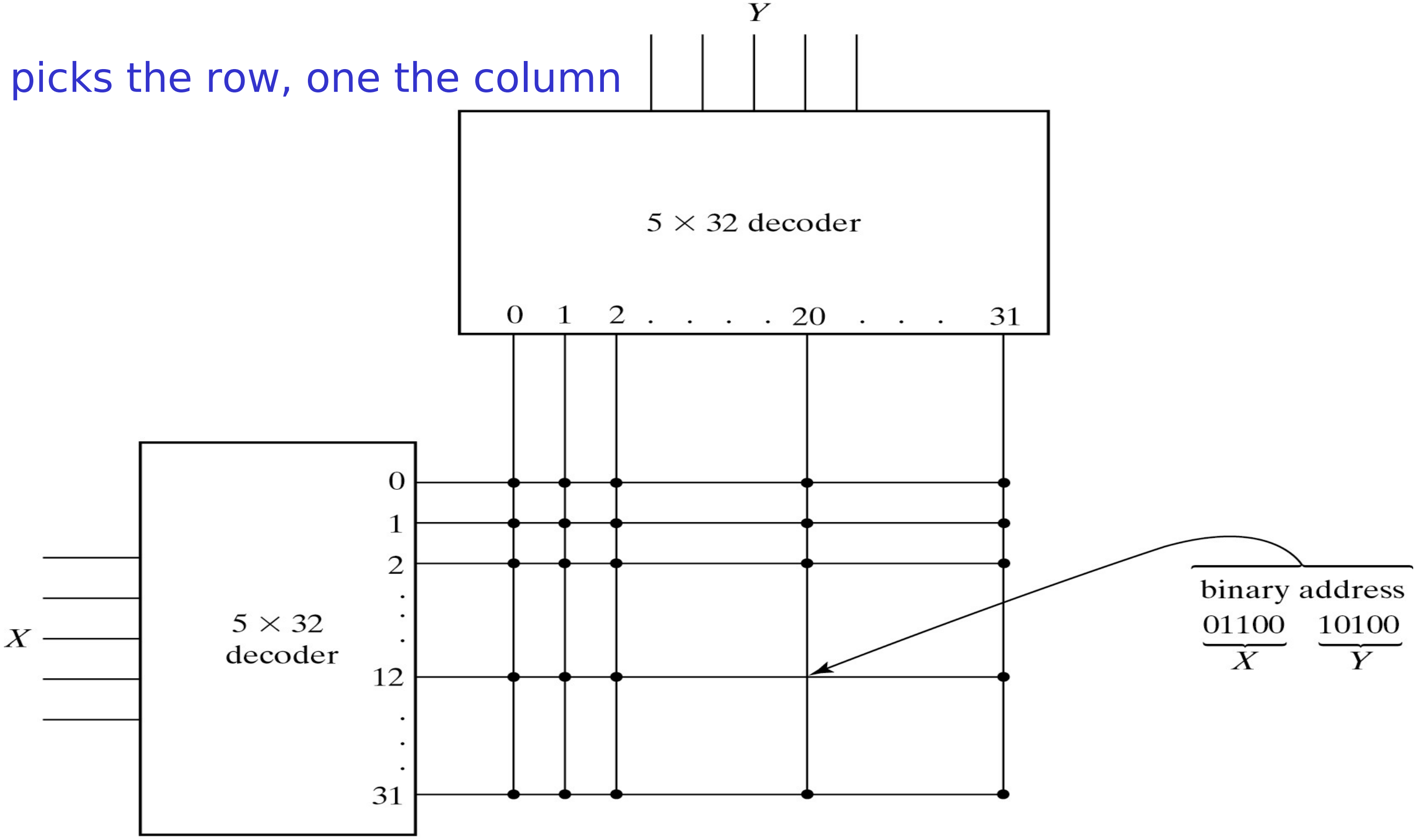


Fig. 7-7 Two-Dimensional Decoding Structure for a 1K-Word Memory

Two-Dimensional Decoding

Needs 64 5-input AND gates instead of 1024 10-input gates.

Address is divided to two equal parts

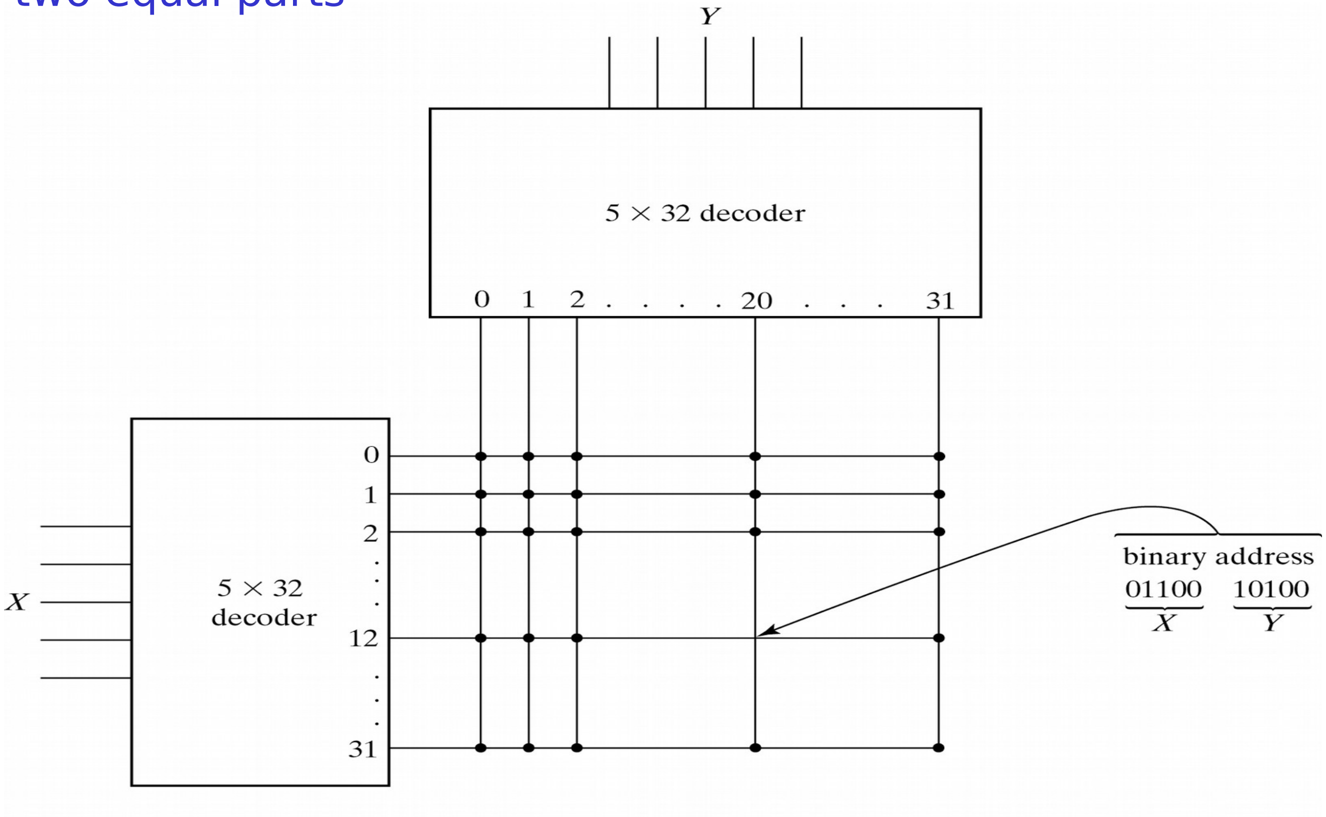
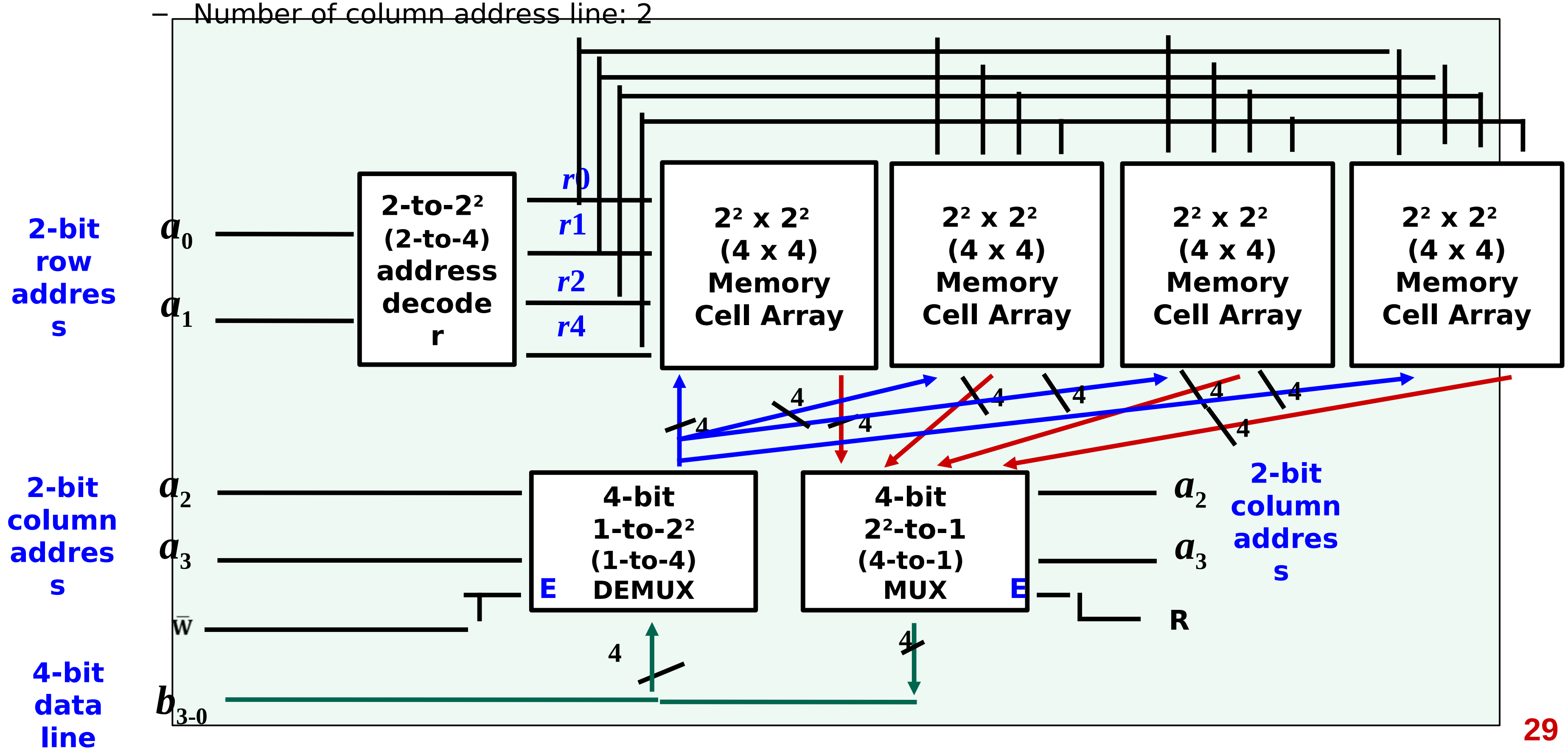


Fig. 7-7 Two-Dimensional Decoding Structure for a 1K-Word Memory

2-dimensional Address Decoding

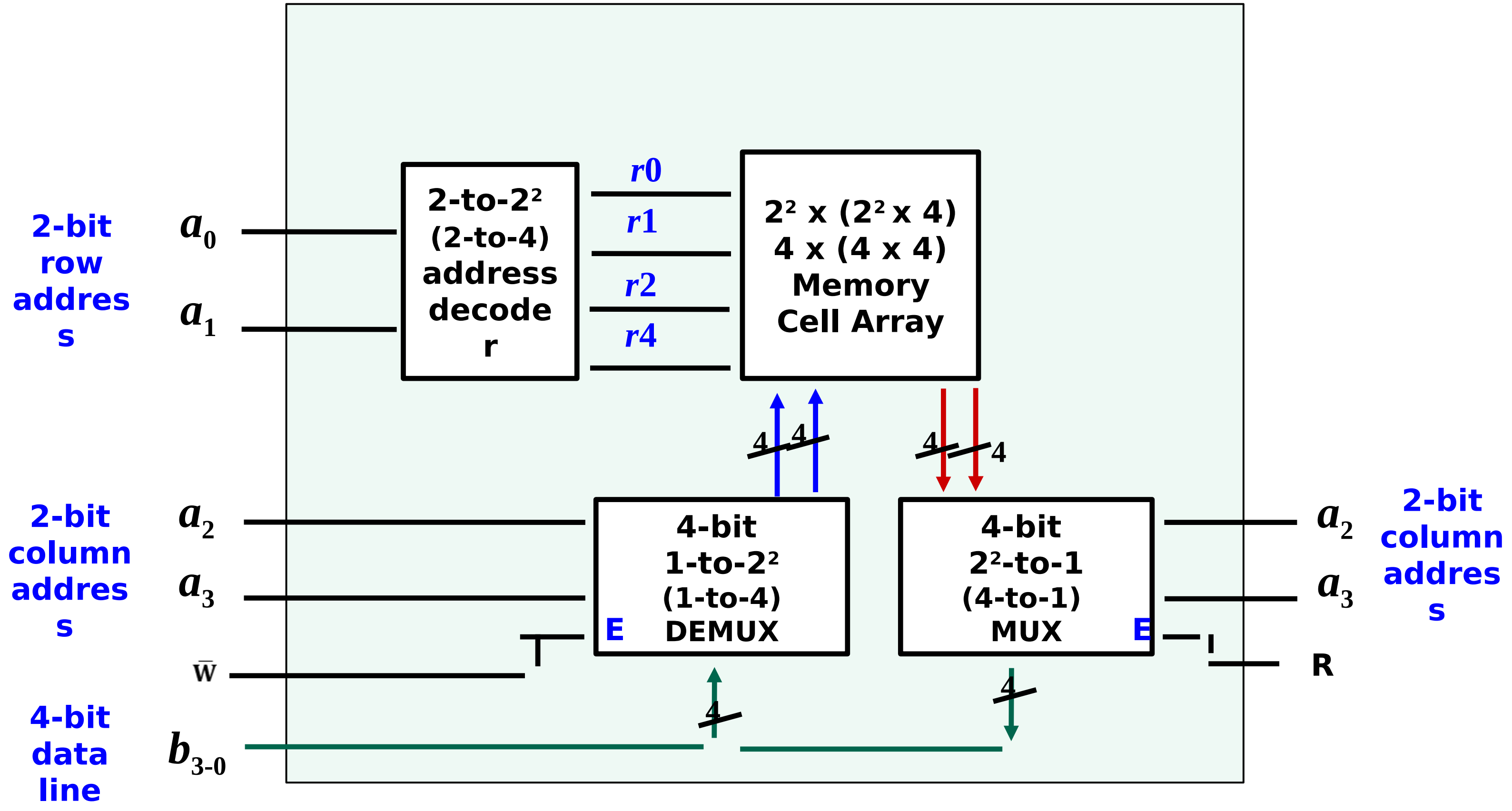
- k-address line is divided into $k/2$ row address and $k/2$ column address
- Now, memory chip is considered as $2^{k/2}$ x $2^{k/2}$ memory cell array
- Example: Design of 16 x 4 bit memory chip
  - Number of address lines: 4
  - Number of row address lines: 2
  - Number of column address line: 2



2-bit row addres s
a_0
a_1
2-to-2^2 (2-to-4) address decode r
$r0$
$r1$
$r2$
$r4$
2^2 x 2^2 (4 x 4) Memory Cell Array
2^2 x 2^2 (4 x 4) Memory Cell Array
2^2 x 2^2 (4 x 4) Memory Cell Array
2^2 x 2^2 (4 x 4) Memory Cell Array
4
4
4
4
4
4
4
4
2-bit column addres s
a_2
a_3
$\bar{W}$
E
4-bit 1-to-2^2 (1-to-4) DEMUX
4-bit 2^2-to-1 (4-to-1) MUX
E
a_2
a_3
2-bit column addres s
R
4
4
4-bit data line
b_{3-0}

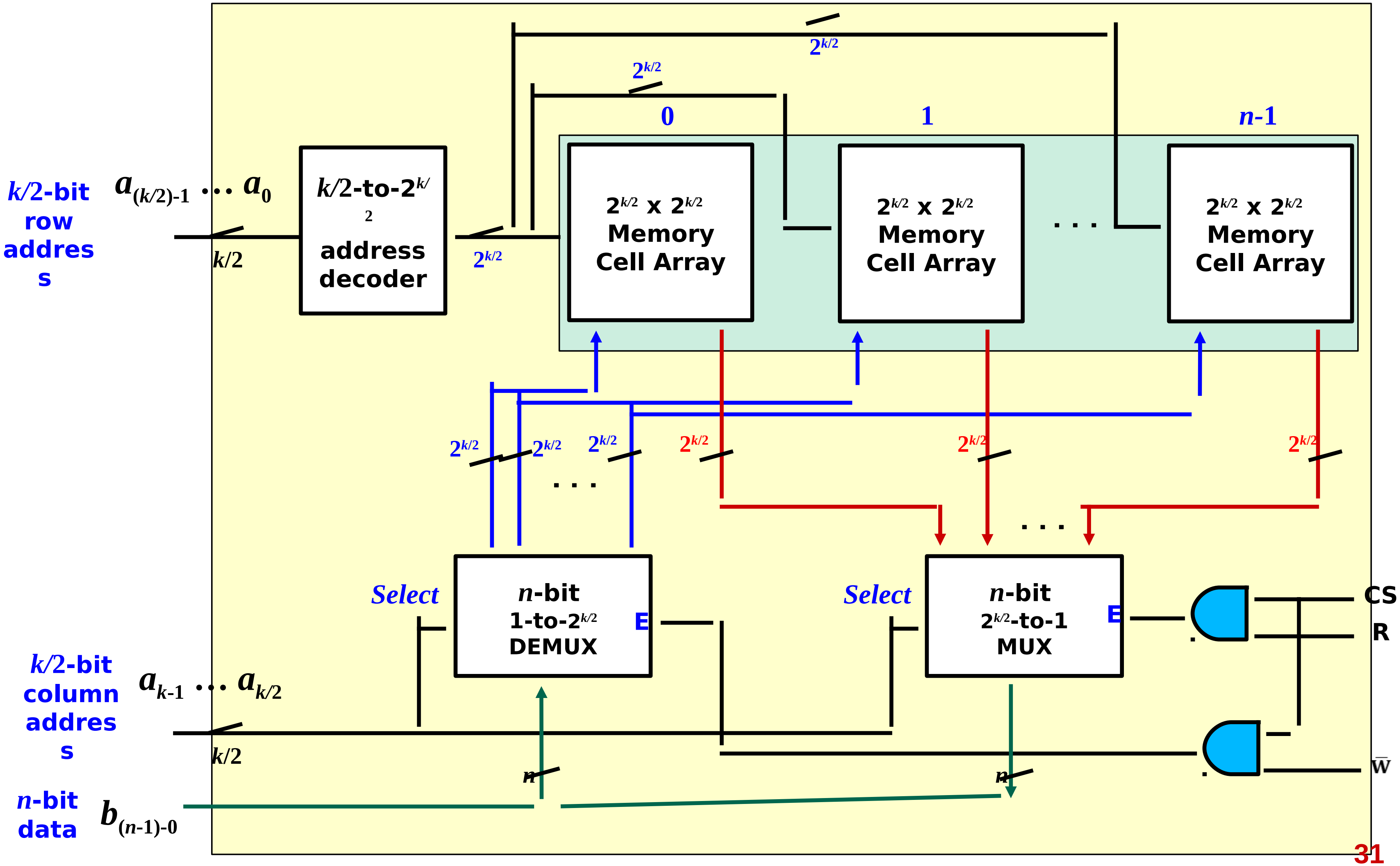
2-dimensional Address Decoding

- *k*-address line is divided into $k/2$ row address and $k/2$ column address
- Now, memory chip is considered as $2^{k/2}$ x $2^{k/2}$ memory cell array
- Example: Design of 16 x 4 bit memory chip
  - Number of address lines: 4
  - Number of row address lines: 2
  - Number of column address line: 2



2-dimensional Address Decoding

- **Organization of 2^k x n SRAM Chip**



2-dimensional Address Decoding

- **Organization of 2^k x n SRAM Chip**
  - The number of cell array depends on the width of the memory chip (word length) expected

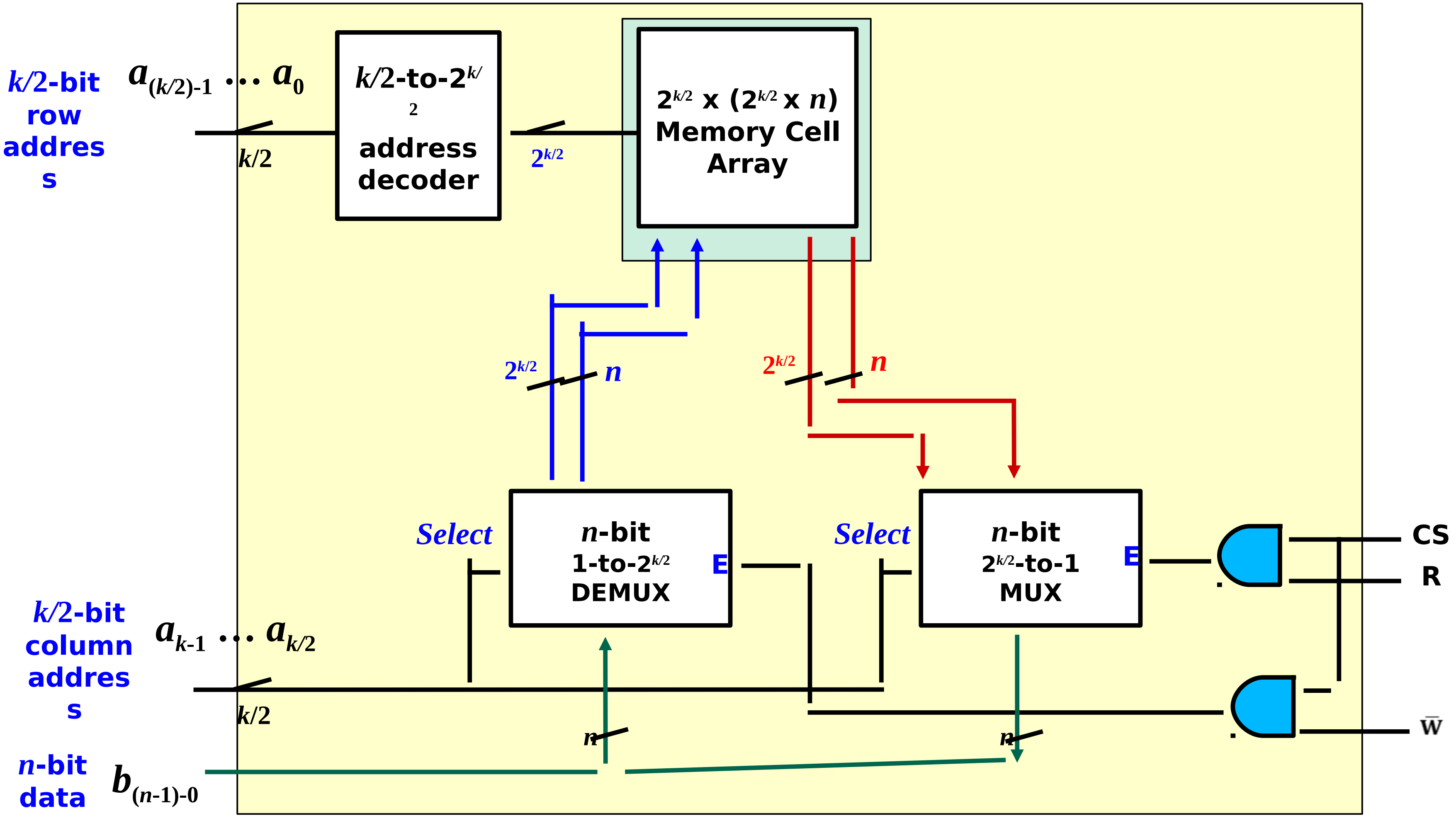


Illustration: Design of 1MB SRAM Chip

- Capacity of memory: 1 MB
- Word length: 8 bits
- Memory organization: 1M x 8

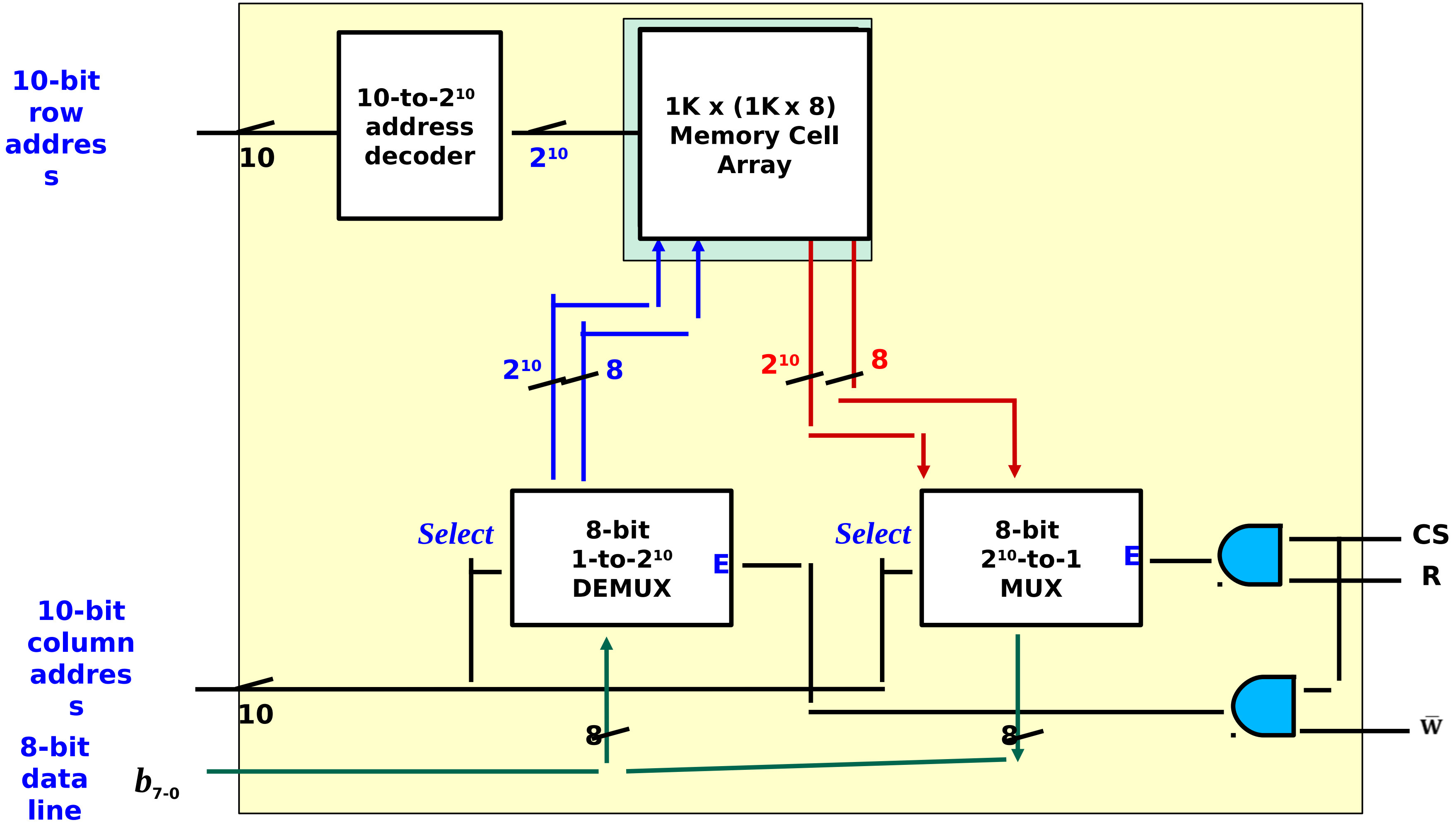
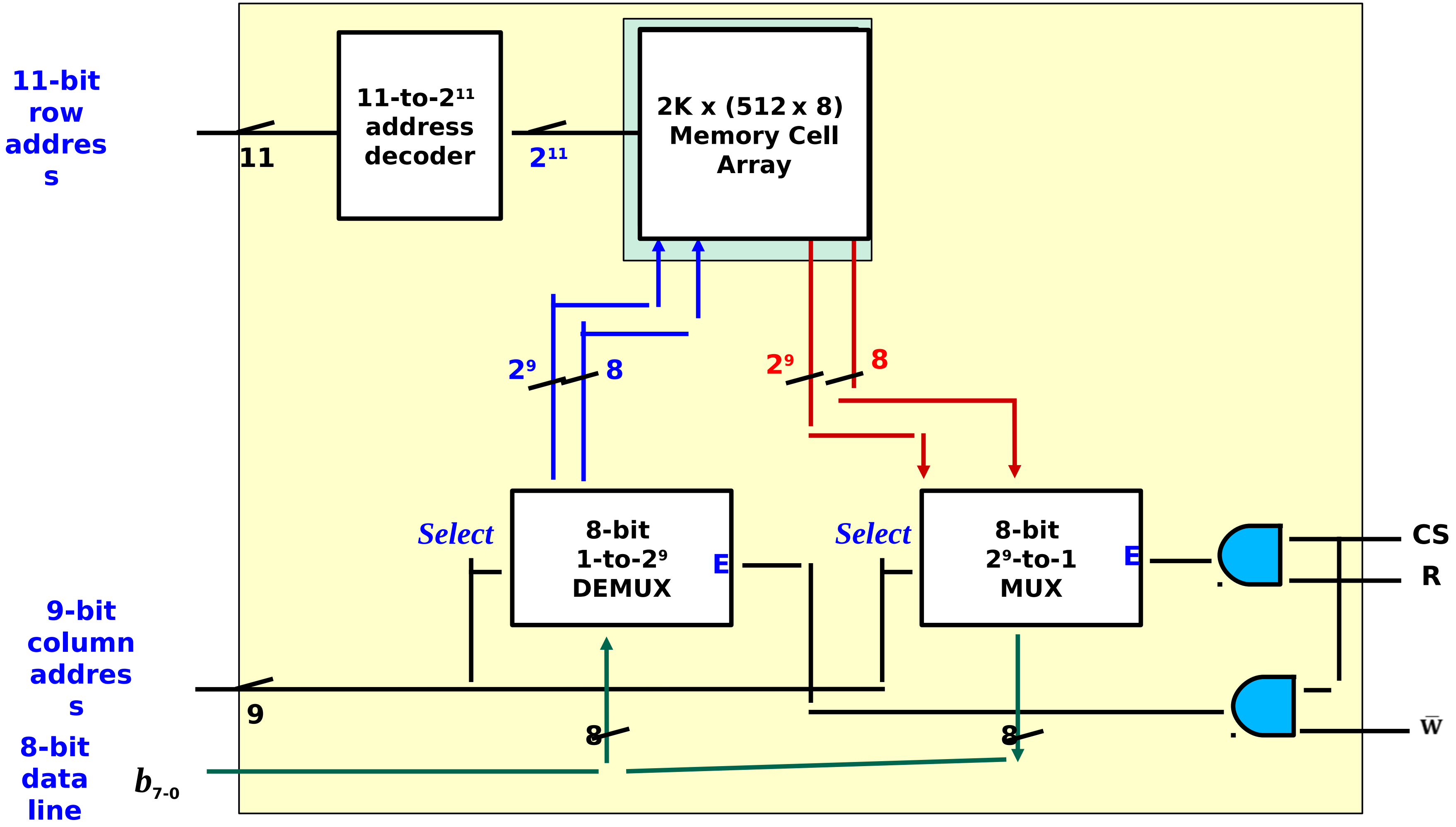


Illustration: Design of 1MB SRAM Chip

- Capacity of memory: 1 MB **(2^{20} B = 2^{23} b)**
- Word length: 8 bits
- Number of row address: 11
- Number of column address: 20-11 = 9



Static RAM Module

- **Byte Addressable Memory**
- Illustration:
  - Task: Design SRAM with capacity 8MB (**2^{23} B**)
  - Requirements:
    - Memory organization depends on the word size and word size decides the width of the data bus
      - Let word size be 32 bit (**4 B**)
      - Now, memory organization: 2M x 32 bit
    - Memory should be byte addressable
      - Let the organization of cell array to incorporate byte addressability be 1M x 8 bit
  - Organization include:
    - 2 rows of chips, each of size 1M x 8 bit
    - Each row contain 4 chips
  - Number of address lines: 23 (**2^{23} B**)

Illustration: 8MB Static RAM Module

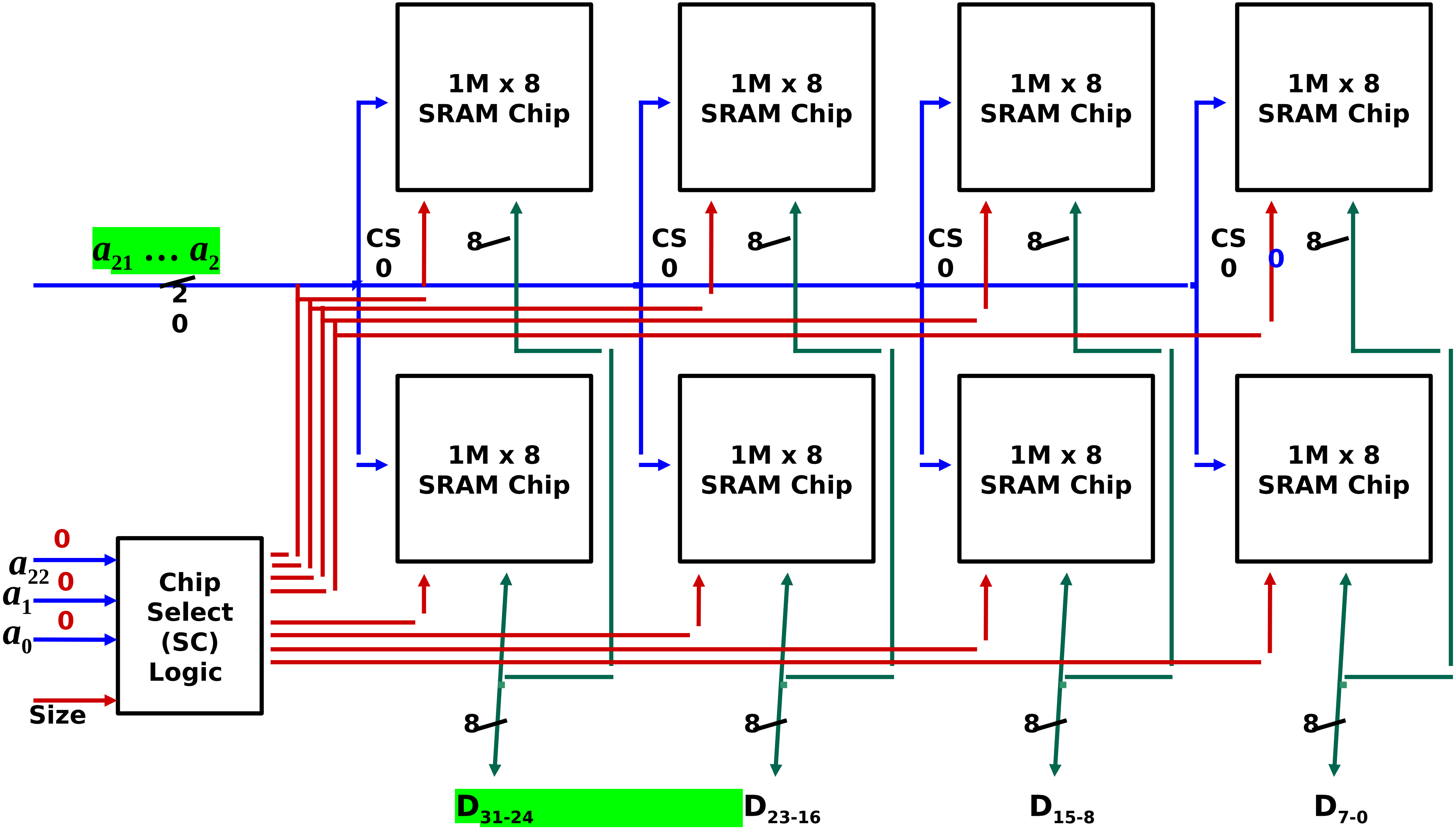


Illustration: 8MB Static RAM Module

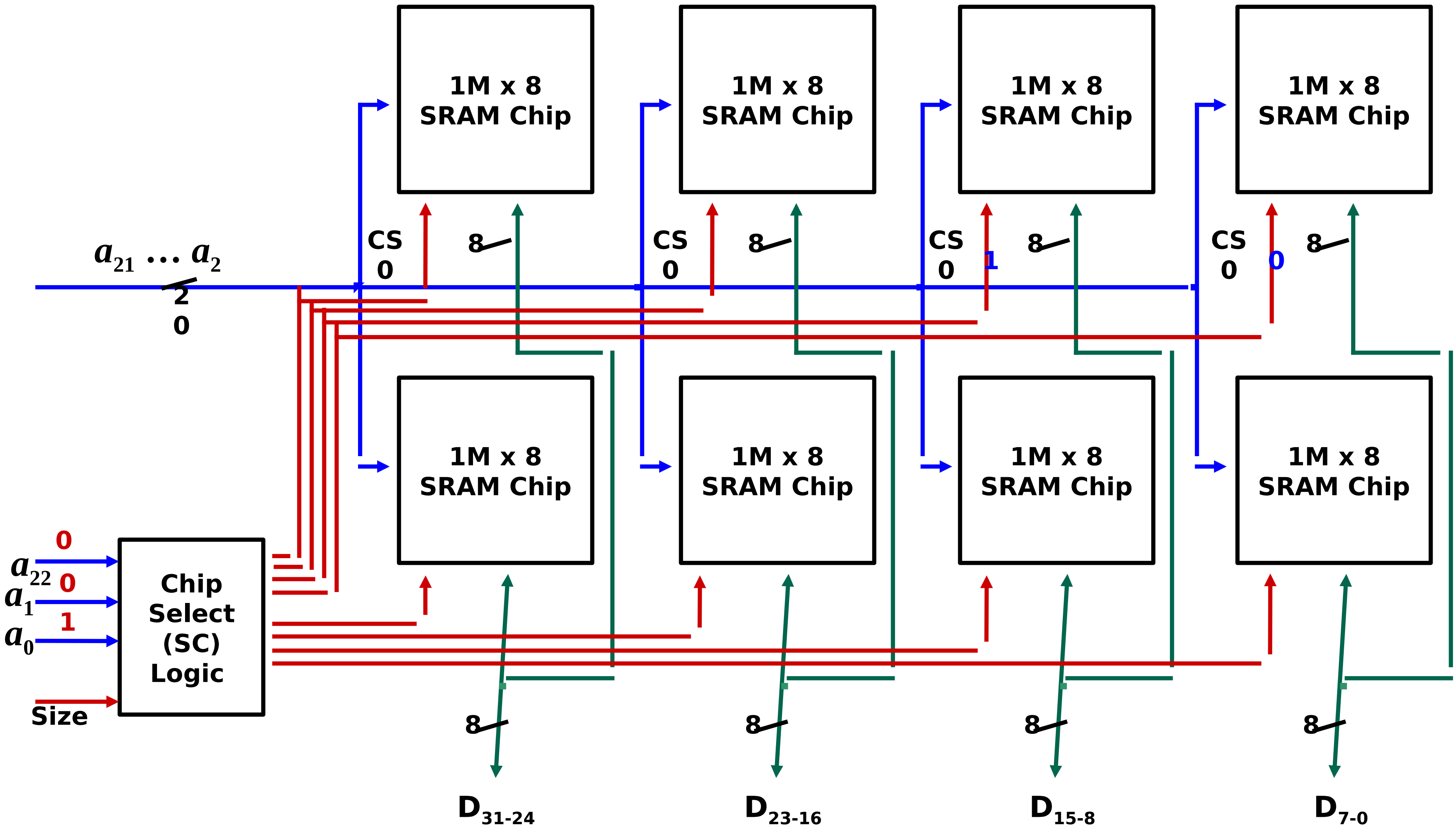


Illustration: 8MB Static RAM Module

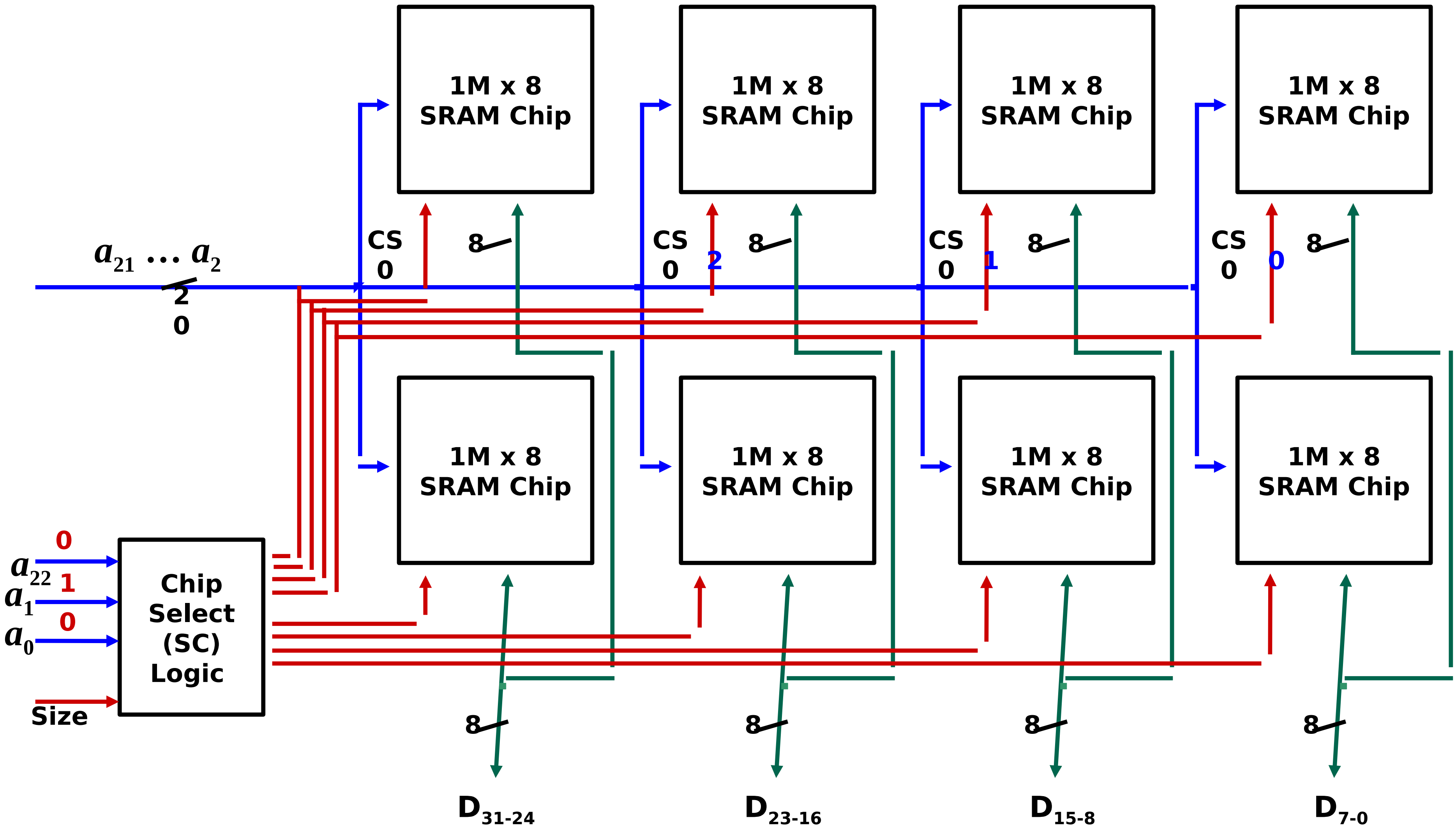


Illustration: 8MB Static RAM Module

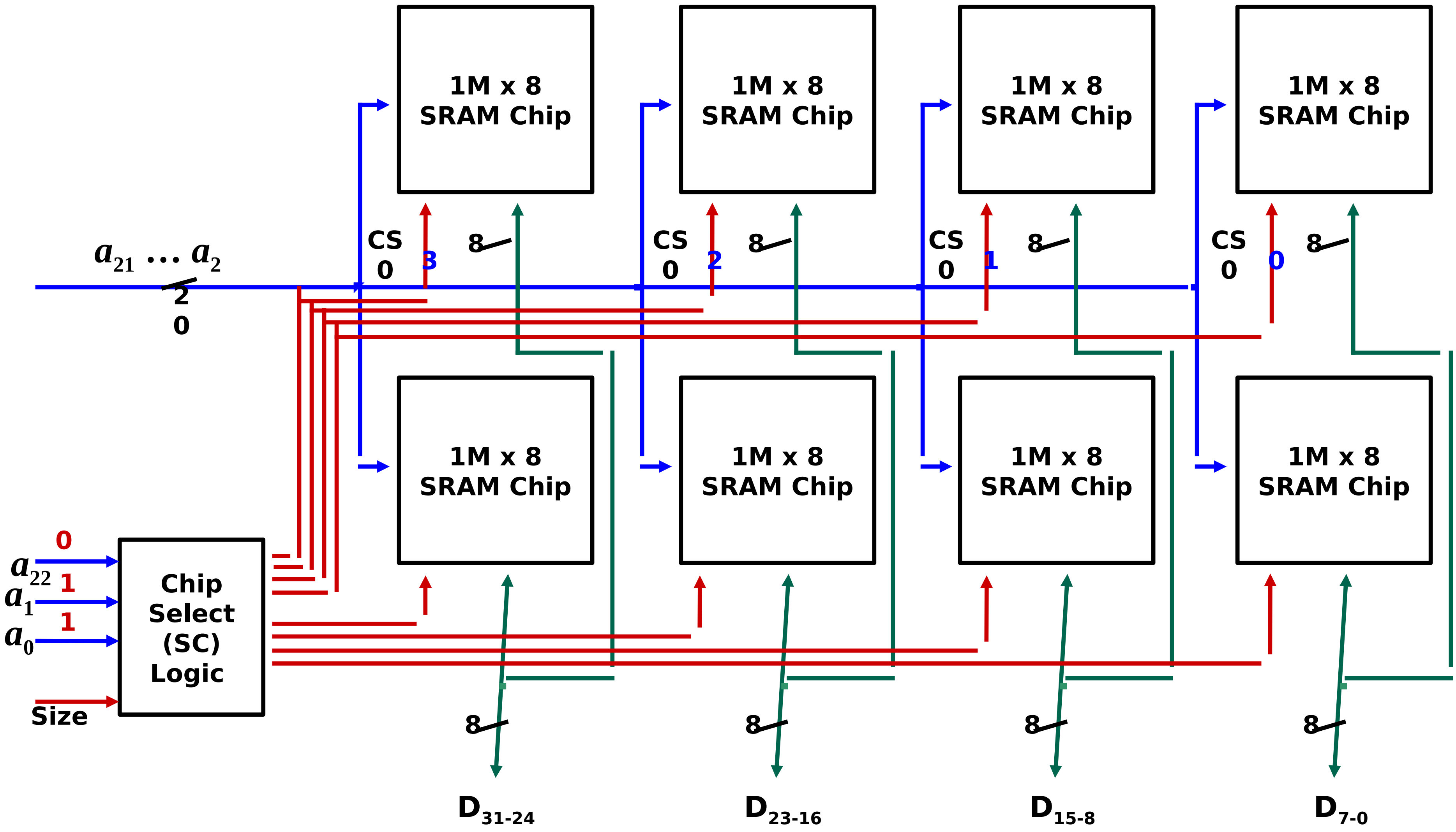


Illustration: 8MB Static RAM Module

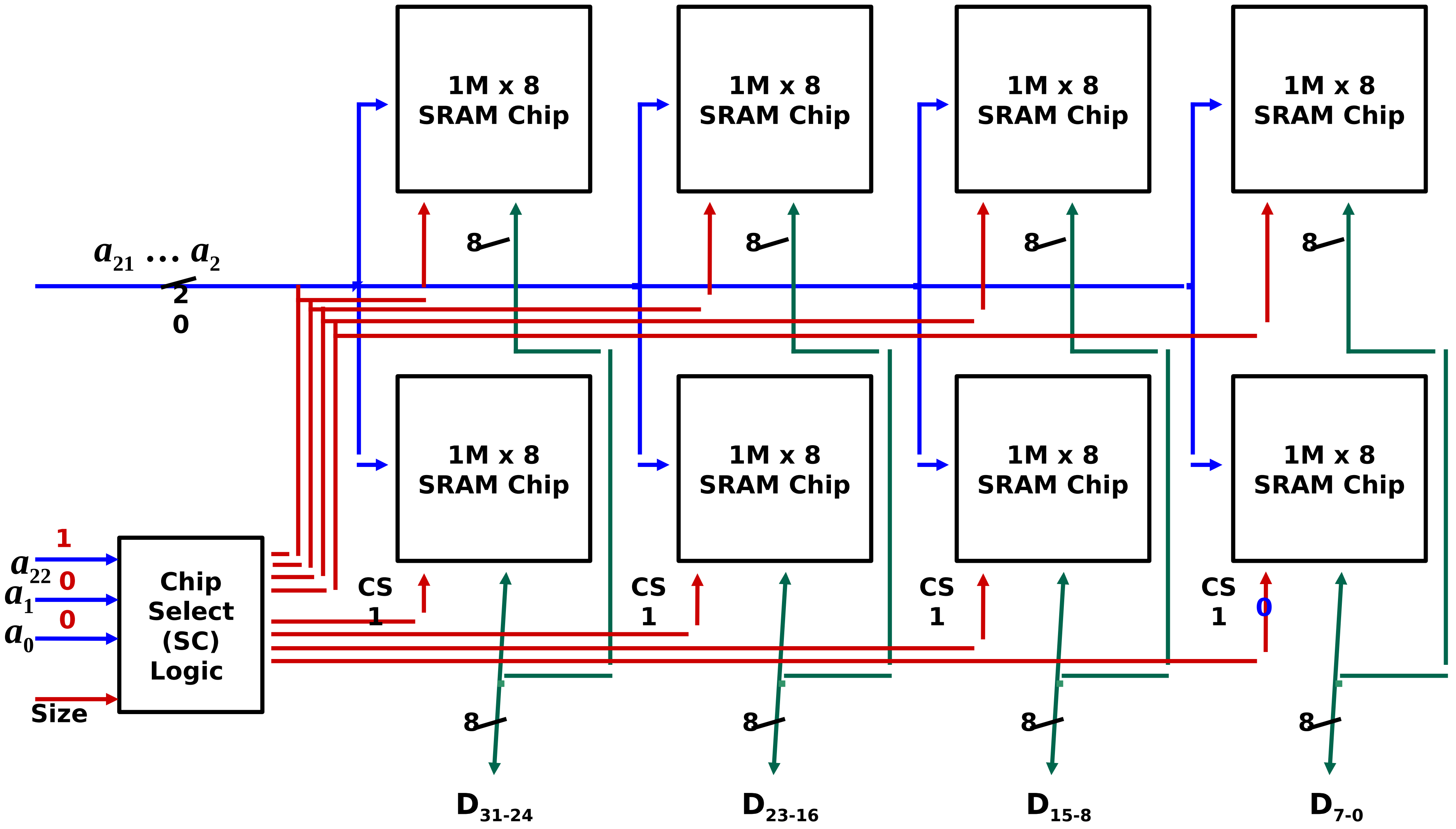


Illustration: 8MB Static RAM Module

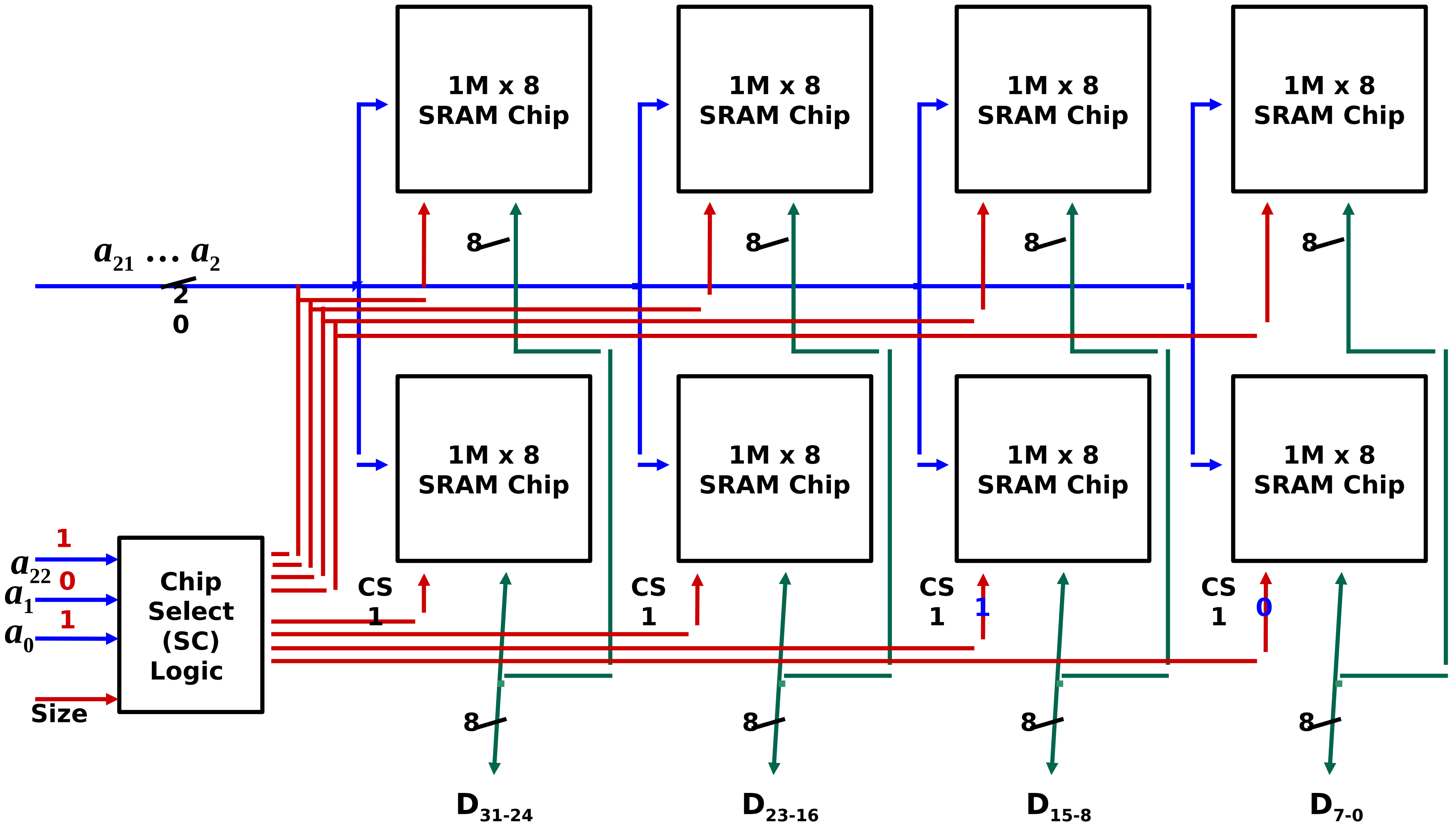


Illustration: 8MB Static RAM Module

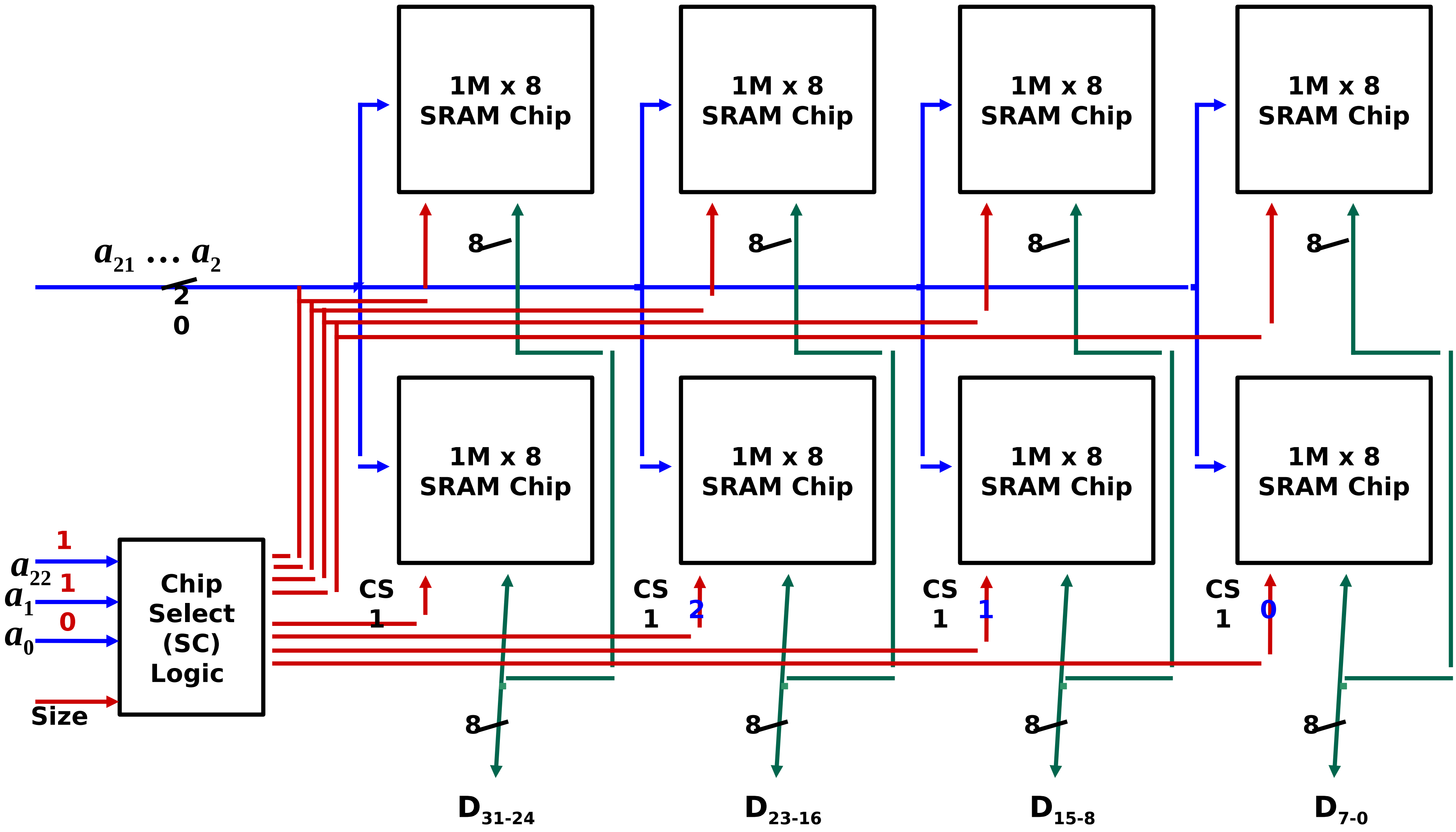
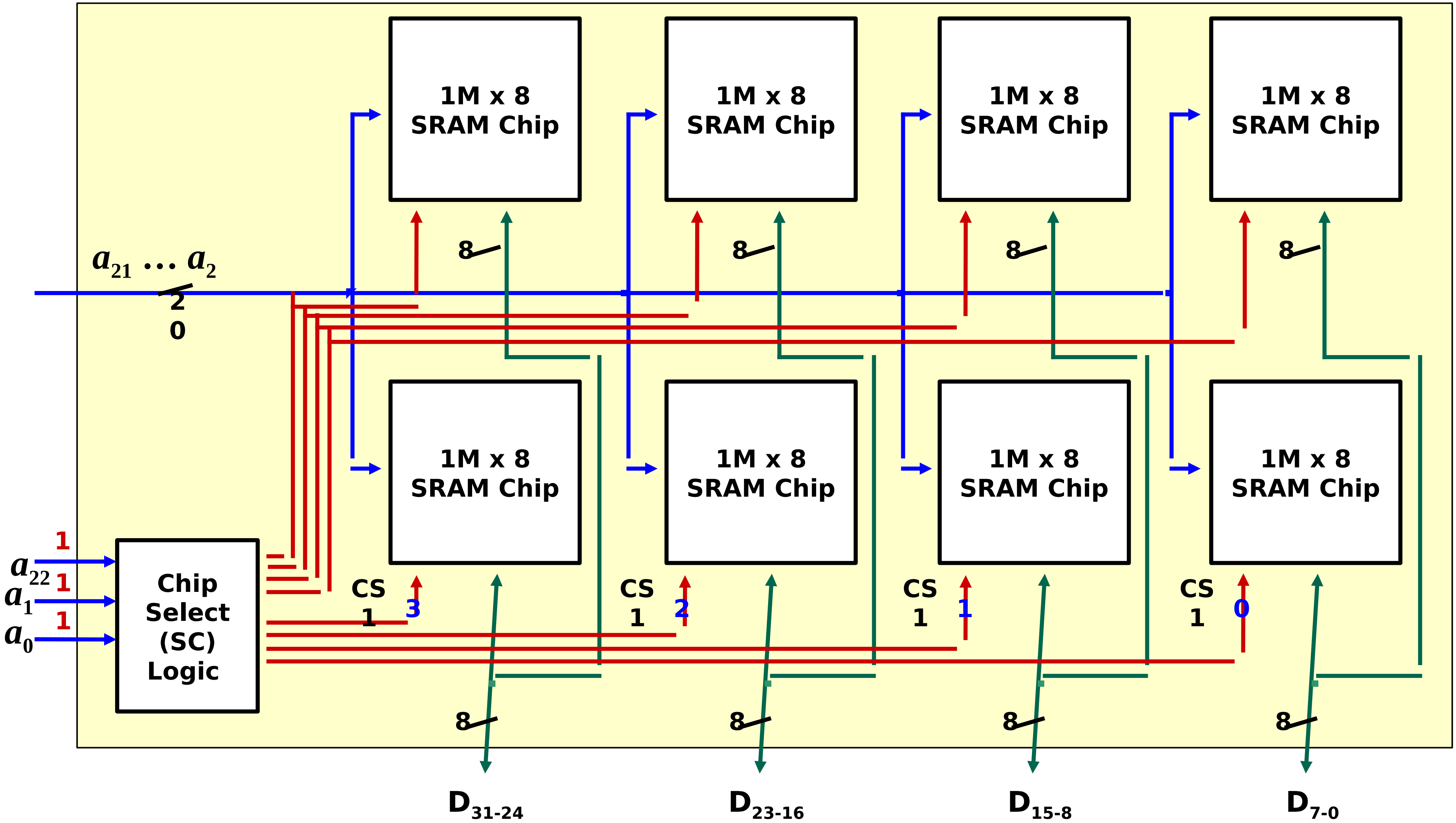


Illustration: 8MB Static RAM Module



Static RAM Module

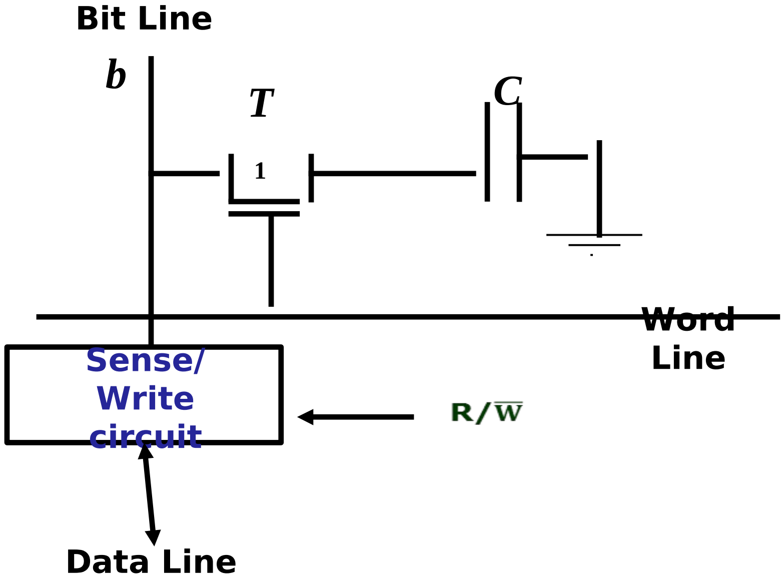
- **4MB SRAM Module**

Static RAM Module

- ***N***: Capacity of SRAM
- ***n*** bits: Width of SRAM
- ***M***: Capacity of one SRAM chip
- ***m*** bits: Width of one SRAM chip
- Number of rows of memory chips: ***N*/*M***
- Number of chips in a row: ***n*/*m***
- According to the number of rows of memory chips and number of chips in a row, Chip Select (CS) logic is designed
- Higher order bits in address select a row of memory chips
- Lower order bits in address select a byte in a word
- Size line in CS logic indicates how many bytes in a word need to be selected

Dynamic RAM (DRAM) Cell

- Less expensive and simpler cell
- Information is stored in the form of a charge on a capacitor (C)
  - Charge in capacitor is stored only for short time
  - However, a cell is required to store information for a much longer time
- To retain information for longer time, content of capacitor mush be periodically refreshed



- Low speed as refresh needed
- Only 1 transistor is used
- Used to build main memory

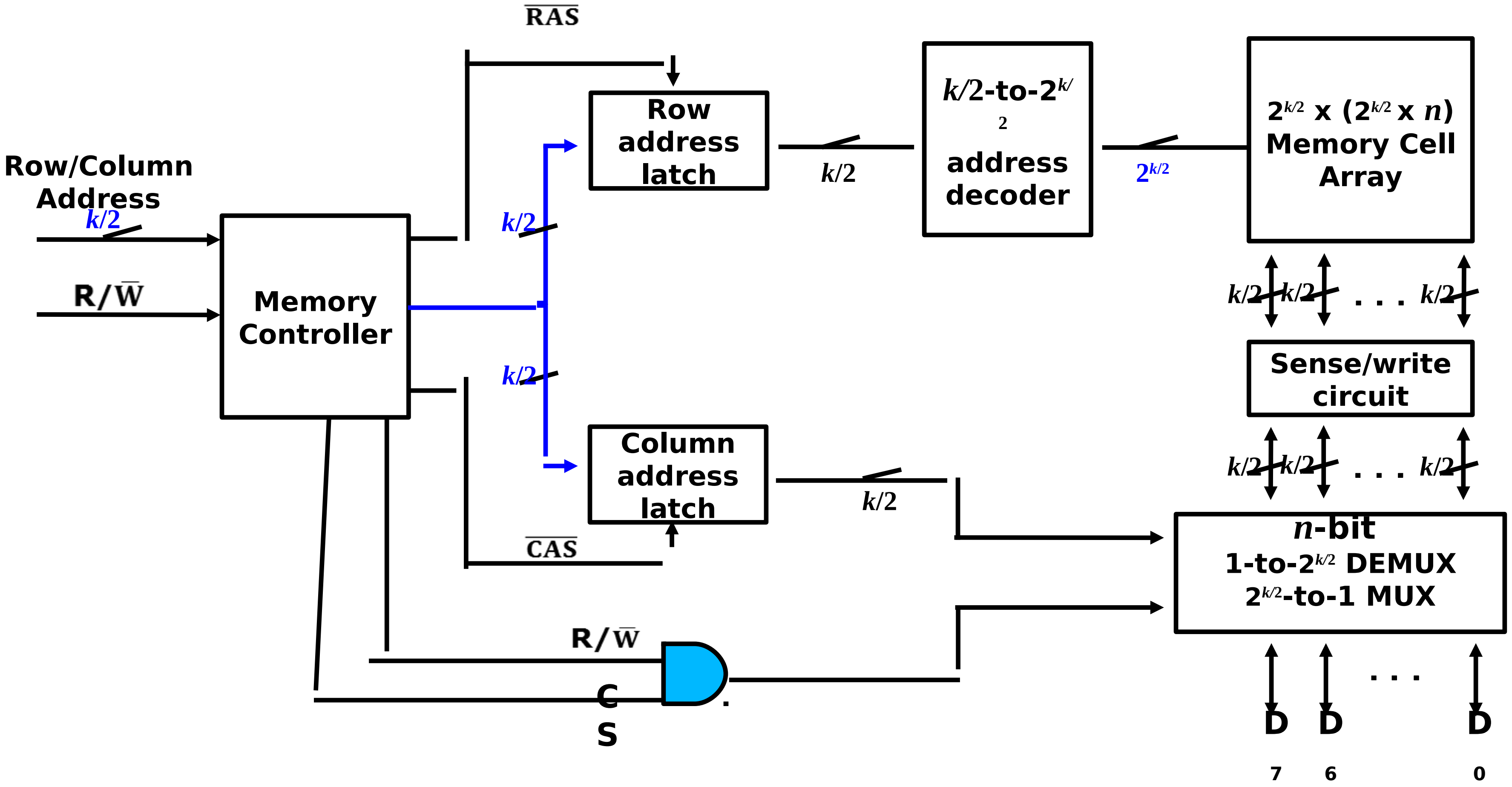
Dynamic RAM (DRAM) Chip Organization

- DRAM cells are also arranged in 2-diemsional array form
- Here also, row address lines are used to select row and column address lines are used to select column
- Two important factors influence the design of the DRAM chip are:
  - Number of input/output pins i.e. external pins
  - Need to refresh the cells
- Scheme for saving pins:
  - Row address and column address are transmitted over the same line one after the other
  - This is called time multiplexing
  - This is usually performed by a memory controller circuit
  - It generates the different control signals

Dynamic RAM (DRAM) Chip Organization

- Two additional control signals are needed to inform the chip when the row address and column address is valid on address line
  - Row address strobe ($\overline{\text{RAS}}$):
    - Inform the chip when the row address is valid on address lines
  - Column address strobe ($\overline{\text{CAS}}$):
    - Inform the chip when the row address is valid on address lines
  - Both $\overline{\text{RAS}}$ and $\overline{\text{CAS}}$ are usually active low

DRAM Chip Organization



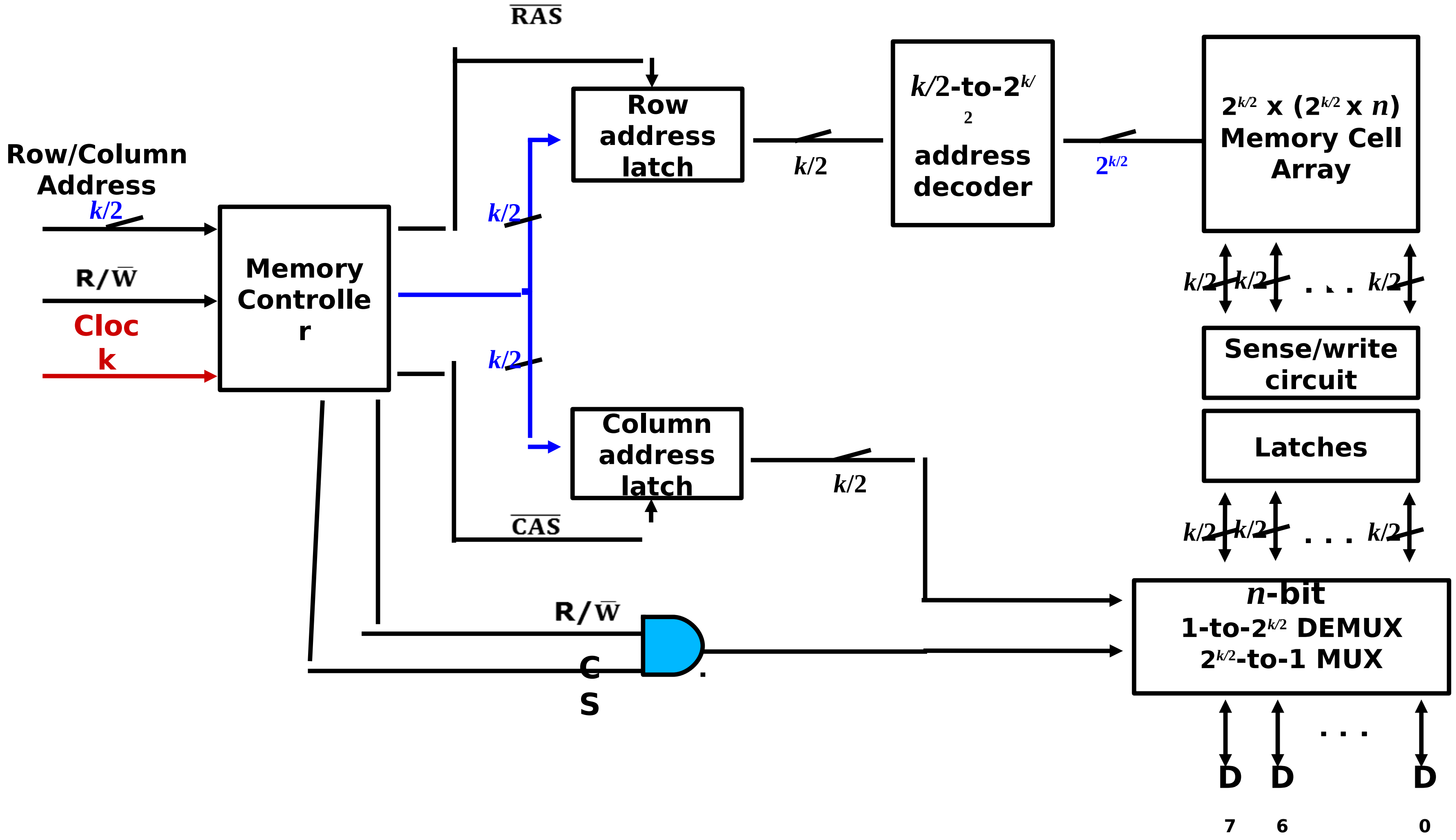
DRAM Chip Organization

- During read or write operation, the row address is applied first
  - It is loaded into row address latch in response to $\overline{RAS}$
  - Then read operation is initiated, in which all cells in a selected row are read and refreshed
- Shortly after row address is loaded, the column address is applied
  - It is loaded onto the column address latch under the control of $\overline{CAS}$
- The information in the column address latch is decoded and appropriate n sense/write circuits are selected
- To ensure that the contents of DRAM are maintained, each row of cells must be accessed and refreshed periodically

DRAM Chip Organization

- Memory controller circuit provide the necessary signals CAS and RAS that governs timing
- These operations are directly synchronised with clock signal
- Such a DRAM chip is called Synchronous DRAM (SDRAM)

SDRAM Chip Organization



Fast Page Mode Feature of SDRAM

- Transfer capability of SDRAM
- Contents of all $2^{k/2}$ cells in a selected row are sensed in each on the n cell arrays
- Only n bits (one from each of the cells arrays) are placed in the data lines, $D_{7\text{-}0}$
- To access other bytes in the same row, without having to reselect the row, a latch is used at the output of the sense/write circuits in each column
- The row address will load the latches corresponding to all bits in the selected row
- Then different column address are applied to place the different bytes on data lines
- Transfer bytes in sequential order
- This arrangement allows transferring a block of data at much faster rate

Refresh Overhead in SDRAM

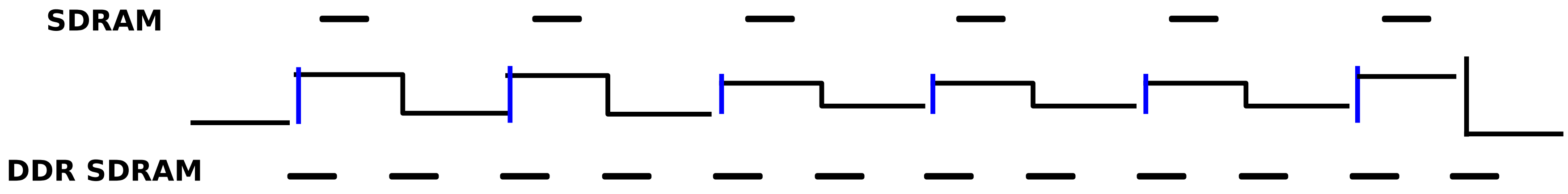
- All dynamic memories need to be refreshed
- In SDRAM typical period of refreshing all rows is ***64ms***
- Each row is refreshed at least in 64*ms*
- Example:
  - *Suppose a SDRAM chip has 8K (8192) rows*
  - *Number of clock cycles to access each row:* 4 clock cycles
  - Number of clock cycles to refresh all rows:

    8192 x 4 = 32768 clock cycles
  - *Suppose clock rate is* 133 MHz
  - Times needed to refresh all rows:

    $32768/133\text{x}10^{6} = 246\text{x}10^{-6}\ s = 0.246\ ms$
  - **Refreshing overhead is 0.246*ms* out of 64*ms***

Double-Data-Rate SDRAM (DDR SDRAM)

- Faster version of SDRAM
- The standard SDRAM performs all actions on the raising edge of the clock cycle
- DDR SDRAM access the cell array in the same way, but transfers the data on both edges of the clock
- Hence, their bandwidth is essentially doubled for long burst transfers
- Bandwidth: The number of bits/bytes that can be transferred in one second



IBM
(11S) IBM P/N Z H/C
FPUT3 47908W
1GB PC2100 CL2.5 2.5V D E (128Mx72)
(4L) Origin: KR OPT:33L5039
HYN
2004-07-09
hynix

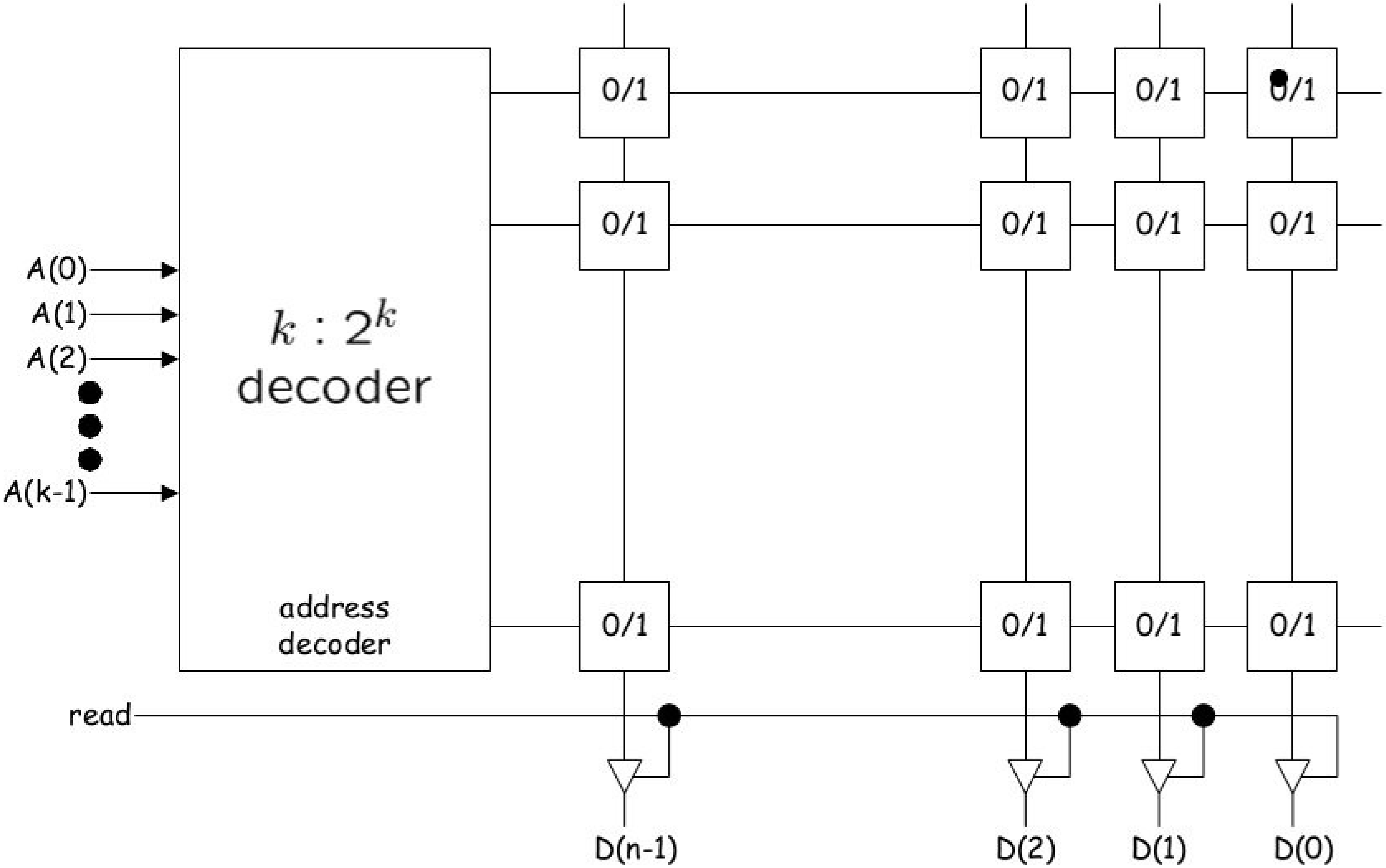
BM 33L5039 RAM Module

(1 GB, DDR2 RAM, 266 MHz, DIMM 184-pin)

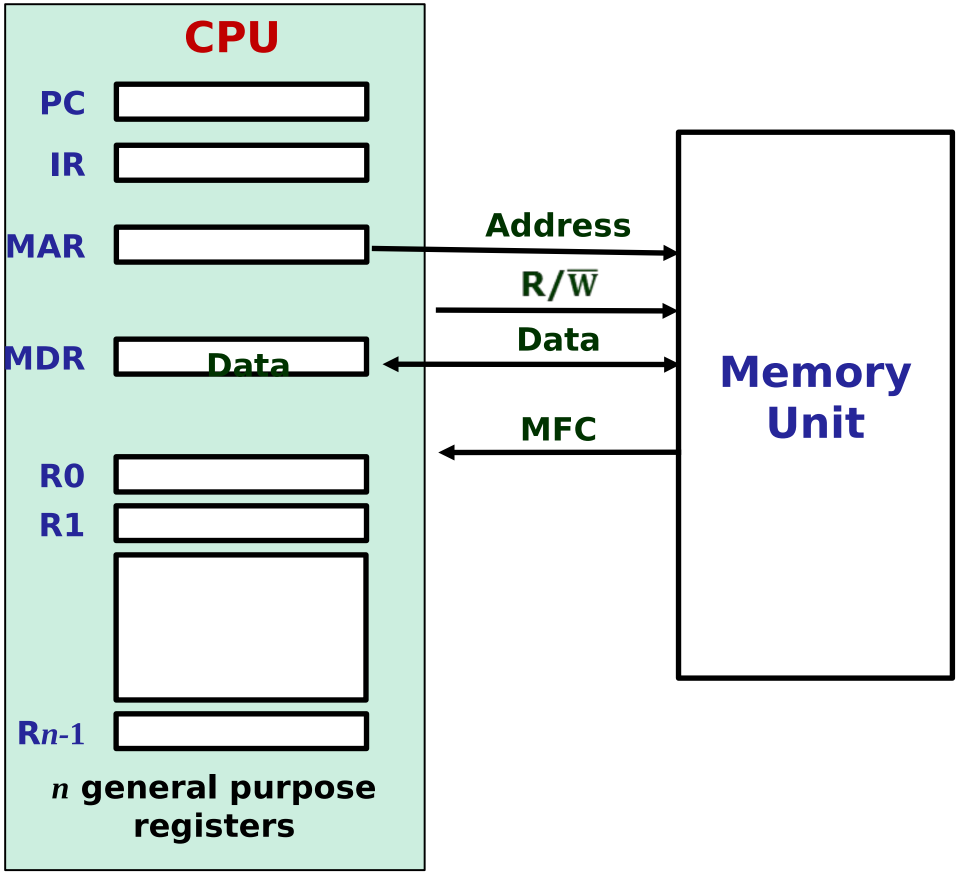
Read-Only Memories (ROMs)

- SRAM and DRAM are volatile i.e. they loose the stored information if power is turned off
- Read-only memories are semiconductor, non-volatile memories
- Their normal operation involve only reading the stored data
- They are extensively used in embedded systems
- Different types of ROMs
  - Read Only Memory (ROM)
  - Programmable ROM (PROM)
  - Erasable, reprogrammable ROM (EPROM)
  - Electicrally erasable reprogrammable ROM (EEPROM)
  - Flash memory

Read-Only Memories (ROMs)

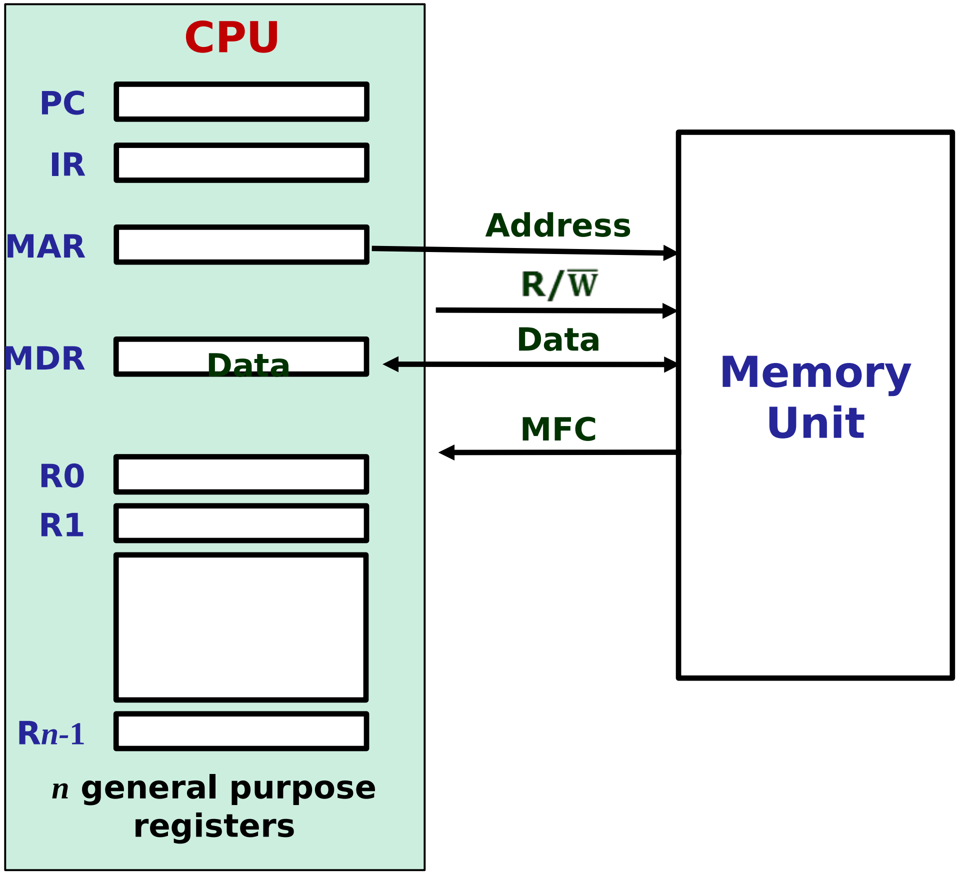


Memory Read and Write Operation



- **Read**
  - Processor loads the address of memory location into MAR
  - Set the R/$\overline{W}$ line to 1
  - Memory responds by placing the data from address location onto data line
  - Confirm the action by asserting MFC (memory function complete) signal
  - Upon receiving MFC signal, processor loads the data on data line into MDR

Memory Read and Write Operation



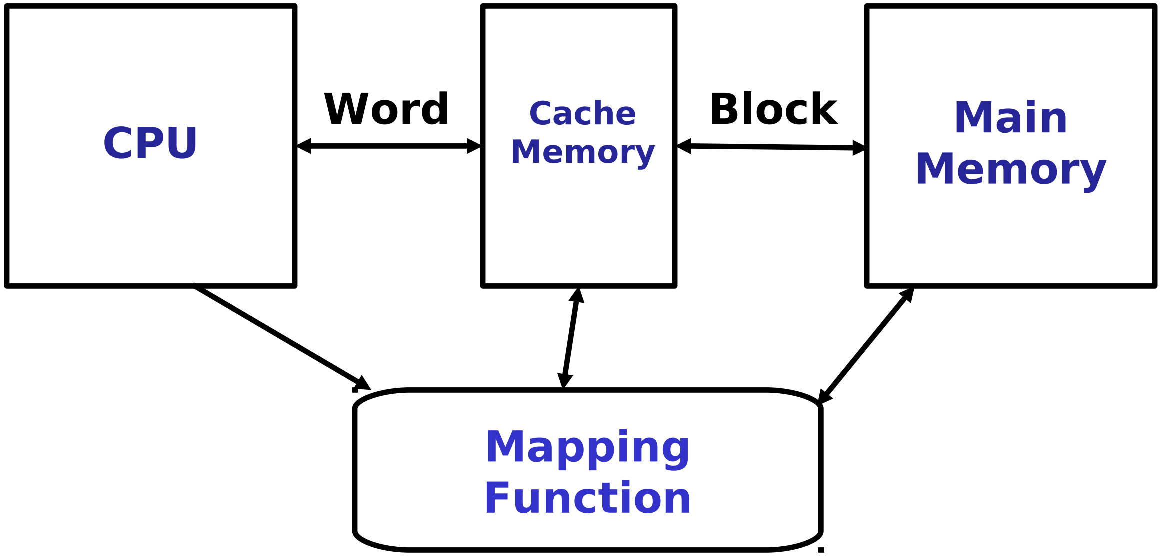
- **Read**
  - Processor loads the address of memory location into MAR
  - Set the R/$\overline{W}$ line to 1
  - Memory responds by placing the data from address location onto data line
  - Confirm the action by asserting MFC (memory function complete) signal
  - Upon receiving MFC signal, processor loads the data on data line into MDR

Cache Memory

- The cache memories are designed to exploit the locality of reference in the program
- **Locality of reference**:
  - Many instructions in localized areas of the program are executed repeatedly during some time period, and the remainder of the program is accessed relatively infrequently
- Different ways of locality of reference

1. Temporal locality:
   - Recently executed instruction/data is likely to be used again very soon
2. Spatial locality:
   - Instructions in close proximity to a recently executed instruction/data (with respect to instruction address) are likely to be executed very soon

Use of a Cache Memory



- Unit of transfer between main memory and cache is block
  - A block is a set of fixed number of words in contiguous address locations
  - Cache block is also called as Cache line
- Mapping function: Correspondence between main memory blocks and those in the cache

Replacement Algorithm

- Read hit/Write hit [Cache hit]:
  - When processor issues read or write request, cache control circuit determines whether the requested word exits in cache
  - If it exists in cache, the read or write operation is performed on the appropriate cache location
  - This means, read hit or write hit is said to have occurred
- Cache miss:
  - If the desired word is not there in cache during read/write operation, then cache miss is said to have occurred
  - During that time new block need to be brought into cache
- Cache control hardware decide which block to removed to create space for new block that contain referenced word when cache is full
- The collection of rules for making this decision constitutes the replacement algorithms

Read and Write Operations on Cache

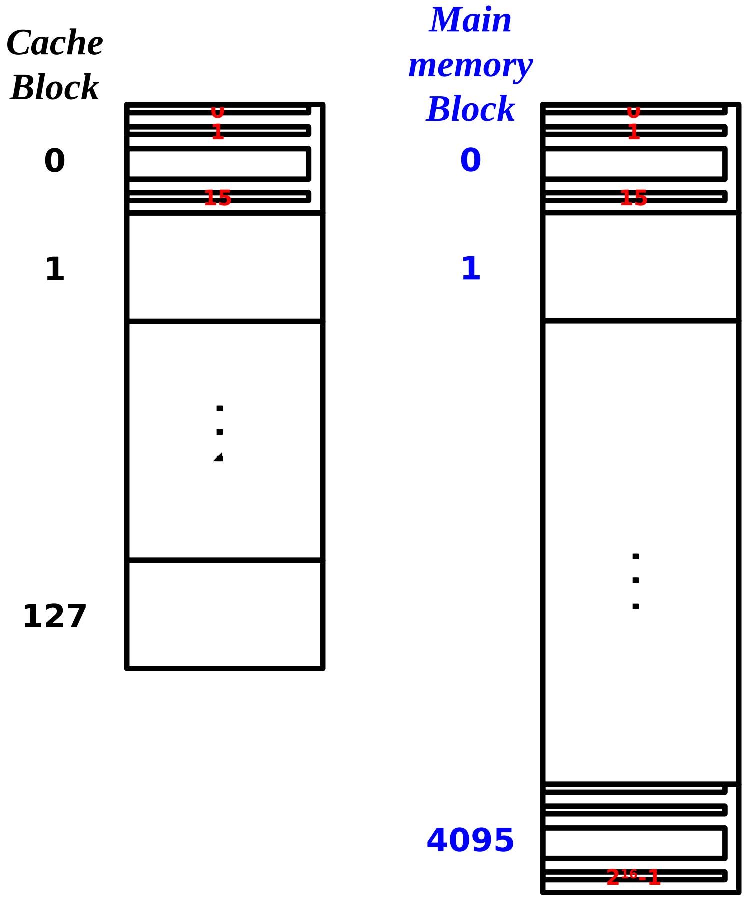
- Read operation:
  - Handling read miss:
    - Approach 1:
      - The block of words that contains the requested word is copied into the cache
      - After entire block is loaded into cache, particular requested word is forwarded to processor
    - Approach 2: Load through or early restart
      - A word is sent to processor as soon as it is read from the main memory
      - At the same time, the block of words that contains the requested word is also copied into the cache
      - This approach reduces the processor waiting period, but with the expense of complex circuitry

Read and Write Operations on Cache

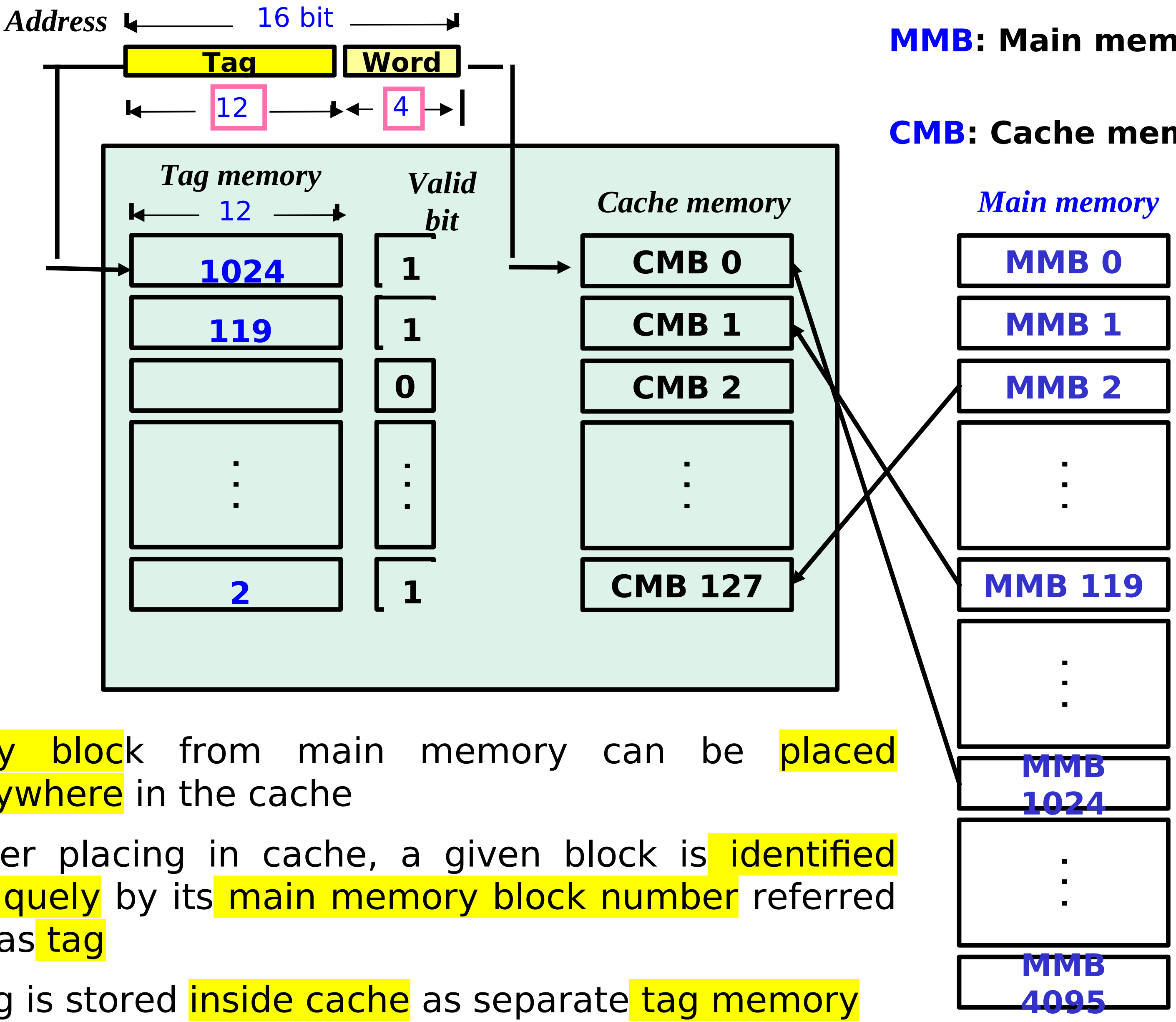
- Write operation:
  - Two techniques for write operation:
    - Write-through protocol
      - The cache location and main memory location are updated simultaneously
    - Write-back (copy-back) protocol
      - Update only the cache location and mark it as updated in a flag bit called dirty bit or modified bit
      - The main memory location is updated later when the block containing that modified word need to be removed from the cache
  - Handling write miss:
    - Write-through protocol:
      - The information is written directly into the main memory
    - Write-back (copy-back) protocol:
      - The block containing the addressed word is first brought into the cache
      - Then the desired word in the cache is overwritten with new information

Mapping Function

- Specifies where memory blocks are placed in the cache
- Cache and main memory are viewed as collection of fixed number of blocks
- Example: Consider a cache and main memory with 2K words and 64K words respectively and each blocks are of size 16 words
  - Block size: 16 words
  - Number of blocks in a cache, N = 2K/16 = 128
  - Number of blocks in main memory, M = 64K/16 = 4K = 4096
  - Addressable location in main memory: 2^{16}



Mapping Function:
Associative Mapping (Associative Mapped Cache)



MMB: Main memory block

CMB: Cache memory block

- Any block from main memory can be placed anywhere in the cache
- After placing in cache, a given block is identified uniquely by its main memory block number referred to as tag
- Tag is stored inside cache as separate tag memory

Mapping Function:
Associative Mapping (Associative Mapped Cache)

- Any block from main memory can be placed anywhere in the cache
- After placing in cache, a given block is identified uniquely by its main memory block number referred to as tag
- Tag is stored inside cache as separate tag memory
- Cache maintains a control bit called valid bit for each block to indicate whether the block contains valid data
- Main memory address reference is partitioned into two parts: Tag and word
- Advantage:
  - Simple and most flexible
  - Complete use of its capacity
- Disadvantage:
  - Tag memory must be searched entirely for each memory reference: Associative search
  - Expensive

Mapping Function:
Direct Mapping (Direct Mapped Cache)

- Main memory block is placed in one and only one place in the cache
- Simplest way to determine the cache location
- Let N be the number of blocks in cache
- The jth main memory block is placed (mapped) onto (j mod N)th block of the cache

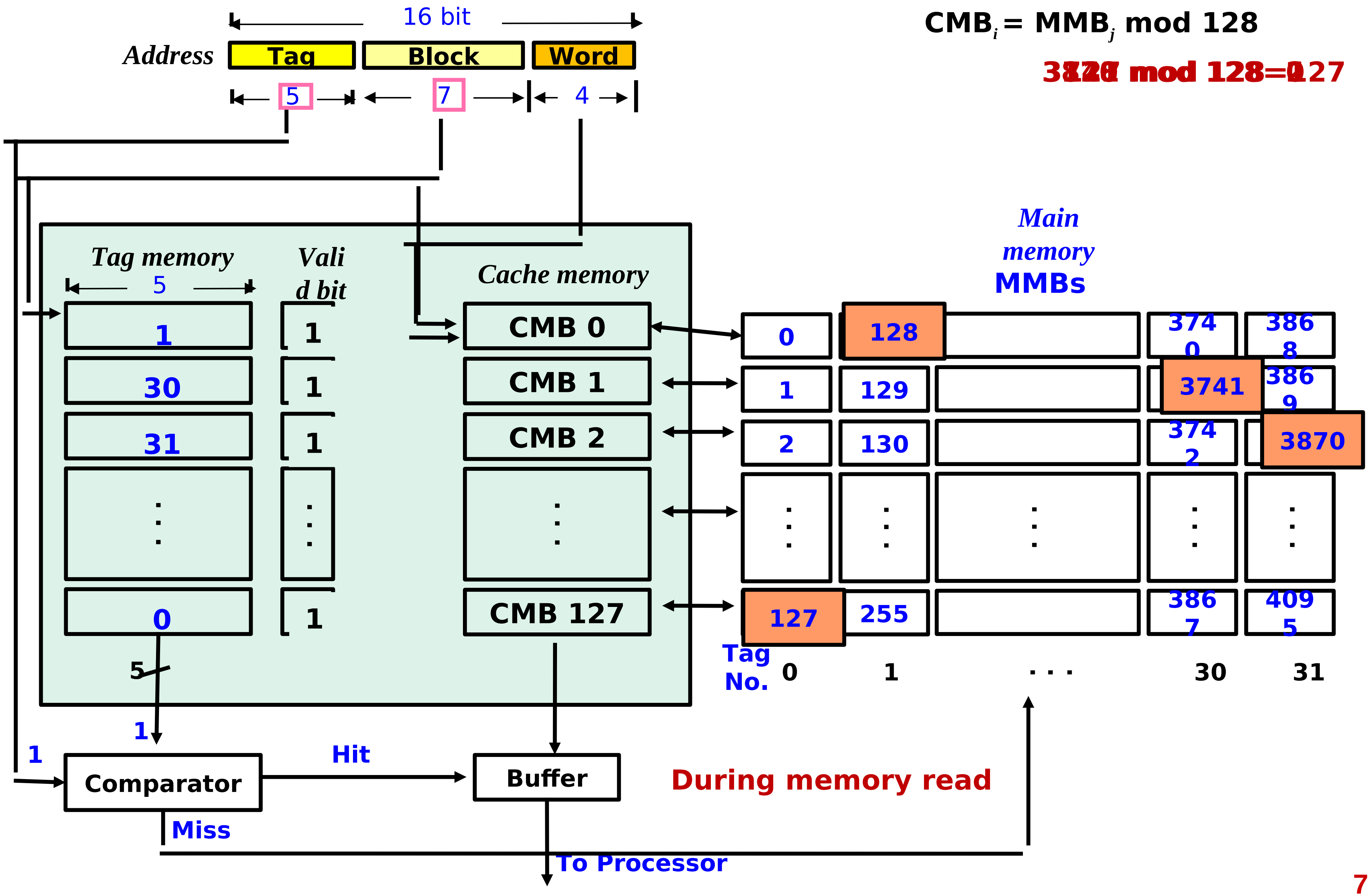
$$\text{CMB}_i = \text{MMB}_j \bmod N$$

- Example: For number of blocks in a cache, $N = 128$ and the number of blocks in main memory, $M = 4096$
  - The main memory blocks 0 or 128 or 256 etc. are mapped onto cache block 0
  - The main memory blocks 1 or 129 or 257 etc. are mapped onto cache block 1
- For the understanding sake, lets consider the main memory as rectangular array of blocks

Mapping Function:
Direct Mapping (Direct Mapped Cache)

$CMB_i = MMB_j \bmod 128$

3870 mod 128=0127



During memory read

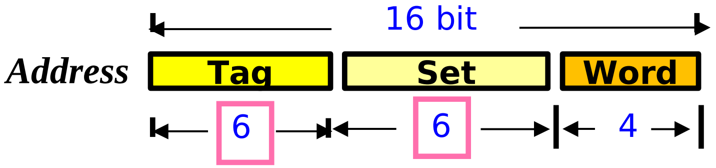
Mapping Function:
Direct Mapping (Direct Mapped Cache)

- Advantage:
  - Simplicity
  - Cost is less compared to fully associative cache
- Disadvantage:
  - Only a single block from a given group can be present in the cache
  - Not flexible
    - Imposes considerable amount of rigidity on the cache organization
- It relies on the principle of locality of reference for its success
- If two blocks of same group are frequently referenced, it leads to trashing

Mapping Function: Set-Associative Mapping

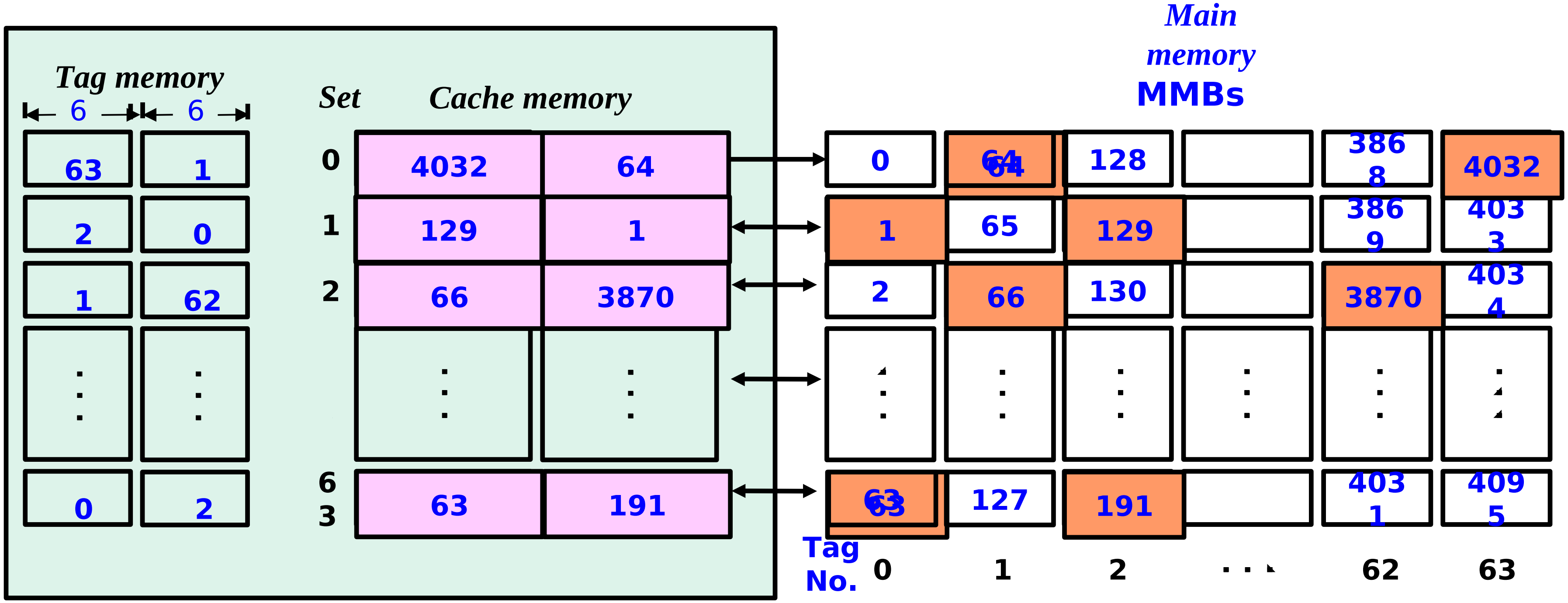
- Combination of the direct and fully associative mapping
- Blocks of the cache are grouped into sets
- Mapping allows block of the main memory to reside in any block of a specific set
- With in a set, it is fully associative
- ***K*-way set associative**:
  - A set may hold *K* number of blocks
  - Number of sets = N/K
  - $\text{CMS}_i = \text{MMB}_j \bmod (N/K)$ where CMS = Cache memory set
- Intel P-III and P-IV processors contain 8-way set associative cache
- Example: Let number of blocks in a cache, N = 128 and the number of blocks in main memory, M = 4096

Mapping Function:
2-way Set-Associative Mapping



CMS$_i$ = MMB$_j$ mod 128/2

3870 mod 64=63



Tag memory

6	6
63	1
2	0
1	62
⋮	⋮
0	2

Set *Cache memory*

Set		
0	4032	64
1	129	1
2	66	3870
	⋮	⋮
63	63	191

Tag No.	0	1	2	. . .	62	63
	0	64	128		3868	4032
	1	65	129		3869	4033
	2	66	130		3870	4034
	⋮	⋮	⋮	⋮	⋮	⋮
	63	127	191		4031	4095

Replacement Algorithm

- Direct mapped cache:
  - The position of each block is predetermined
  - No replacement strategy exists
- In associative and set-associative cache, there exists flexibility
  - Replacement strategy needed
- Replacement algorithms helps in deciding which of the old blocks in the cache to be replaced when the cache is full and a new block is brought into the cache
- Objective: To keep the blocks in cache that are likely to be referenced
- Locality of reference in program gives a clue to a reasonable strategy
- It is sensible to replace the block that has gone longest time without being referenced

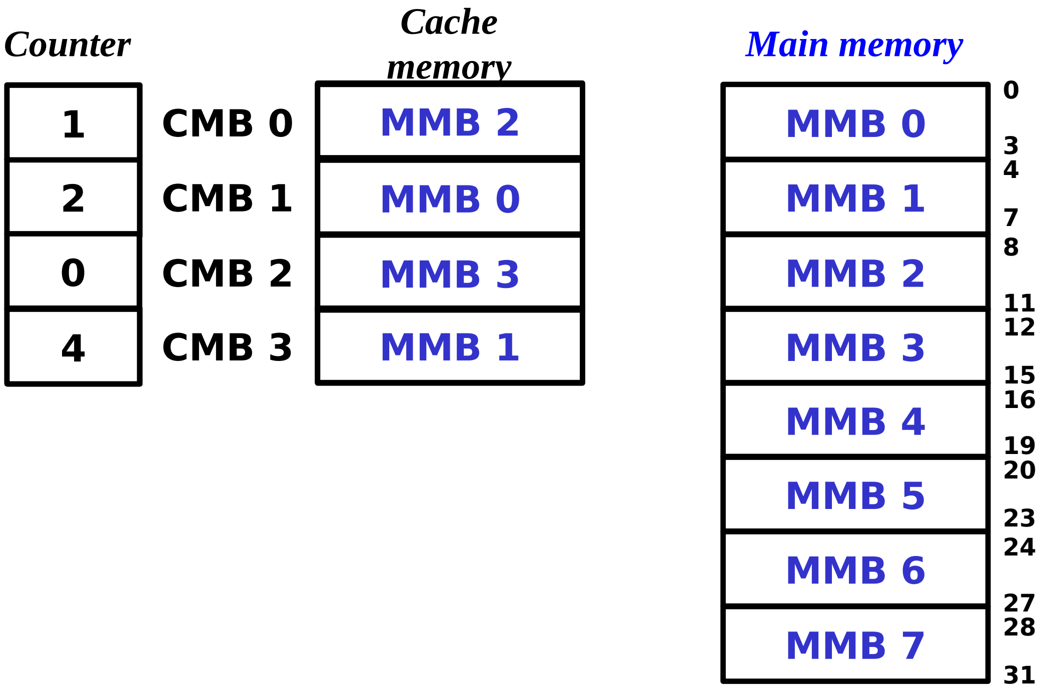
Least Recently Used (LRU) Algorithm

- The cache controller need to track references to all blocks
- LRU algorithm is implemented by using counters for each block
- LRU algorithm:
  - **During cache hit**:
    - Counter of the block referenced is set to 0
    - Counter of the empty block locations are unchanged
    - Counter of other occupied block locations are incremented by 1
  - **During cache miss and cache is not full**:
    - Load the main memory block to empty space and set the counter of that block to 0
    - Increment the counter of other block location by 1
  - **During cache miss and cache is full**:
    - Block with largest counter value (LRU) is removed
    - New block is loaded in that emptied location
    - Counter of that location is set to 0
    - Increment the counter of other locations by 1

Illustration: LRU Algorithm

- Consider fully associative cache
  - Let number of blocks in a cache, $N = 4$ and the number of blocks in main memory, $M = 8$
  - Let each block is 4 words

Read LOC#3
Read LOC#7
Read LOC#11
Read LOC#2
Read LOC#8
Read LOC#14
Read LOC#16



Counter		*Cache memory*
1	CMB 0	MMB 2
2	CMB 1	MMB 0
0	CMB 2	MMB 3
4	CMB 3	MMB 1

Main memory	
MMB 0	0 3
MMB 1	4 7
MMB 2	8 11
MMB 3	12 15
MMB 4	16 19
MMB 5	20 23
MMB 6	24 27
MMB 7	28 31

Illustration: LRU Algorithm

- Consider fully associative cache
  - Let number of blocks in a cache, N = 4 and the number of blocks in main memory, M = 8
  - Let each block is 4 words

Read LOC#3

Read LOC#7

Read LOC#11

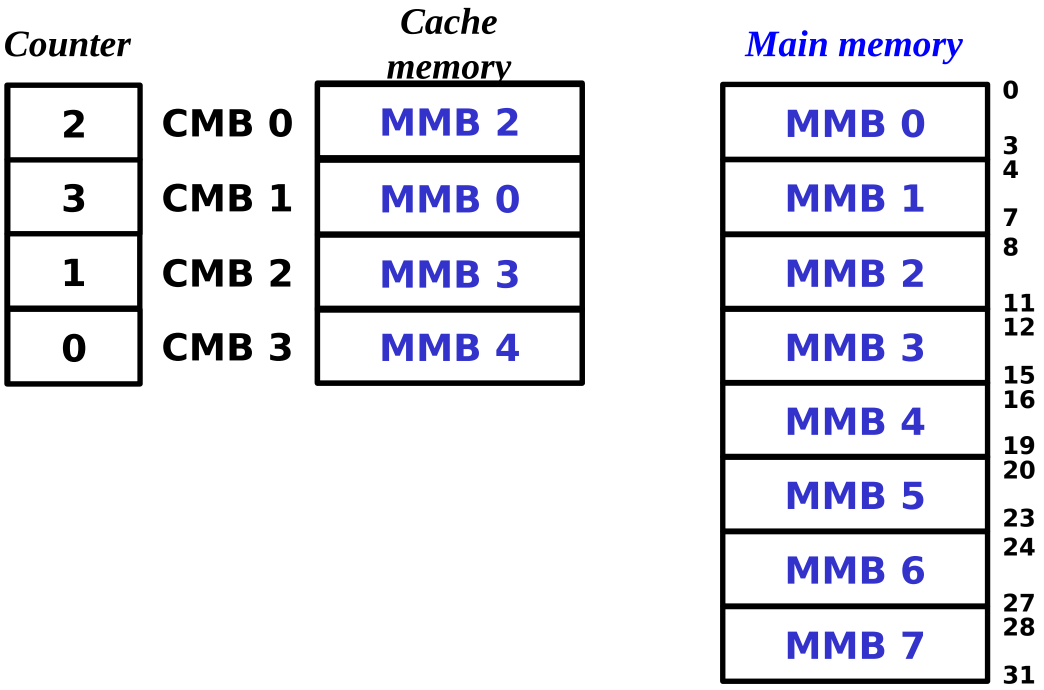
Read LOC#2

Read LOC#8

Read LOC#14

Read LOC#16

Read LOC#23



Counter		*Cache memory*
2	CMB 0	MMB 2
3	CMB 1	MMB 0
1	CMB 2	MMB 3
0	CMB 3	MMB 4

Main memory

MMB 0	0 3
MMB 1	4 7
MMB 2	8 11
MMB 3	12 15
MMB 4	16 19
MMB 5	20 23
MMB 6	24 27
MMB 7	28 31

Illustration: LRU Algorithm

- Consider fully associative cache
  - Let number of blocks in a cache, N = 4 and the number of blocks in main memory, M = 8
  - Let each block is 4 words

Read LOC#3

Read LOC#7

Read LOC#11

Read LOC#2

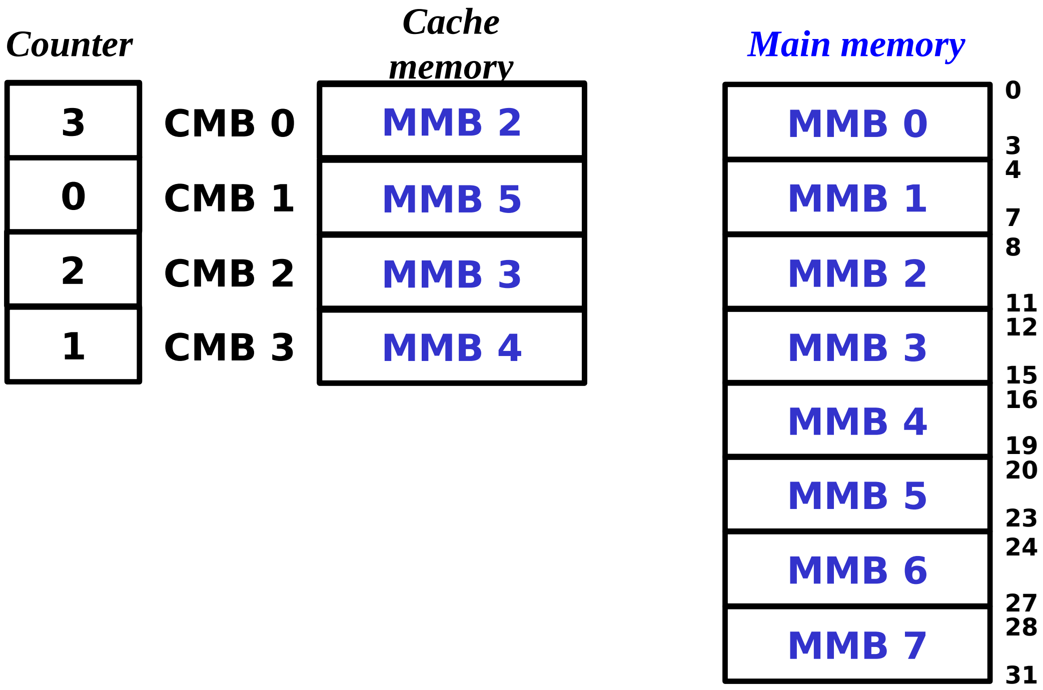
Read LOC#8

Read LOC#14

Read LOC#16

Read LOC#23

Read LOC#25



Counter		*Cache memory*
3	CMB 0	MMB 2
0	CMB 1	MMB 5
2	CMB 2	MMB 3
1	CMB 3	MMB 4

Main memory	
MMB 0	0 3
MMB 1	4 7
MMB 2	8 11
MMB 3	12 15
MMB 4	16 19
MMB 5	20 23
MMB 6	24 27
MMB 7	28 31

Illustration: LRU Algorithm

- Consider fully associative cache
  - Let number of blocks in a cache, N = 4 and the number of blocks in main memory, M = 8
  - Let each block is 4 words

Read LOC#3

Read LOC#7

Read LOC#11

Read LOC#2

Read LOC#8

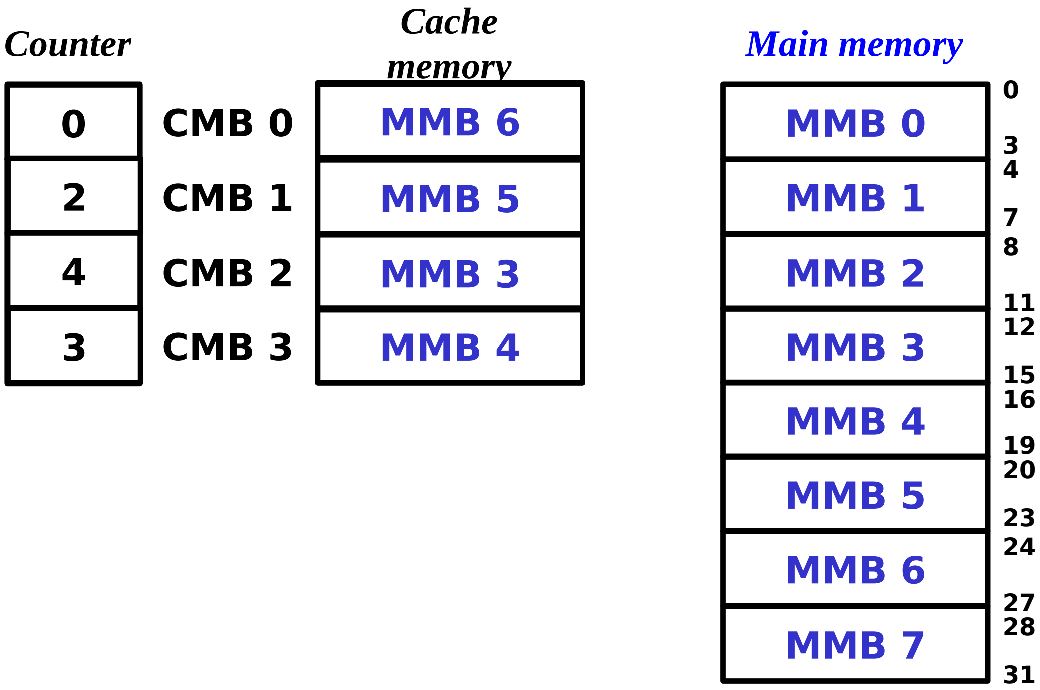
Read LOC#14

Read LOC#16

Read LOC#23

Read LOC#25

Read LOC#27



Cache Memory Architectures

- Cache memory hierarchy: L1 cache, L2 cache and L3 cache
- Two architectures of cache memories:
  - Harvard architecture cache
    - Separate data and instruction cache
    - Advantages:
      - Prevents conflicts between blocks of instruction and data that might map onto same location
      - Program generally do not modify instructions
      - Instructions take less memory than program data
    - Generally used in L1 cache
  - Unified cache (Princeton architecture cache)
    - Cache contain both instruction and data

Categories of Cache Misses

- Goal: To reduce number of cache misses
- This requires to know the reasons for cache misses

1. Compulsory miss:
   - Occurs when the cache is first referenced
   - It causes the block to be brought into the cache
2. Capacity miss:
   - Occurs when the amount of data referenced by the program exceeds the capacity of the cache
   - It causes some blocks to be evicted to make room to new data
   - If the evicted data is referenced again by the program, capacity miss occurs
3. Conflict miss:
   - Occurs in associative/set-associative mapping
   - Occurs when program references more blocks of data mapped on to the same set
   - It causes one of the block to be removed
   - If the removed block is referenced again, conflict miss occurs

Cache Performance Considerations

- Ideally, entire memory hierarchy would appear to the processor as a single memory unit, that has the access time of a cache on processor (L1) and the size of memory disk
- Performance is adversely affected by the actions that must be taken after a miss
- Hit: Successful access to data in a cache
- Hit rate (h): Ratio of number of hits over all attempted access
  - Example: h=0.9 means 90% of the time required block is in cache
- Miss rate: Ratio of number of misses over all attempted access

Cache Performance Considerations

- Miss penalty:
  - The extra time needed to bring the desired information into the cache
  - It is the time that the processor is stalled during waiting
  - It is also the time needed to bring a block of data from a slower unit in the hierarchy to a faster unit
- Suppose we have a cache and a main memory in the hierarchy
  - Let h be the hit rate
  - t_M be the miss penalty i.e. time to access information in the main memory
  - t_C be the cache hit latency i.e. time to access information in cache
  - The average access time experienced by the processor:

$\boldsymbol{t_{avg} = h\, t_C + (1-h)\, t_M}$ i.e. $t_{avg} =$ hit rate * t_C + miss rate * t_M

Cache Performance Considerations

- Suppose we have L1 & L2 caches and a main memory in the hierarchy
  - h_1 : hit rate of L1 cache
  - h_2 : hit rate of L2 cache
  - t_M be the miss penalty i.e time to access information in the main memory
  - t_{C1}: Time to access L1 cache (L1 cache latency)
  - t_{C2}: Time to access L2 cache (L2 cache latency)
  - The average access time experienced by the processor in the two level cache:

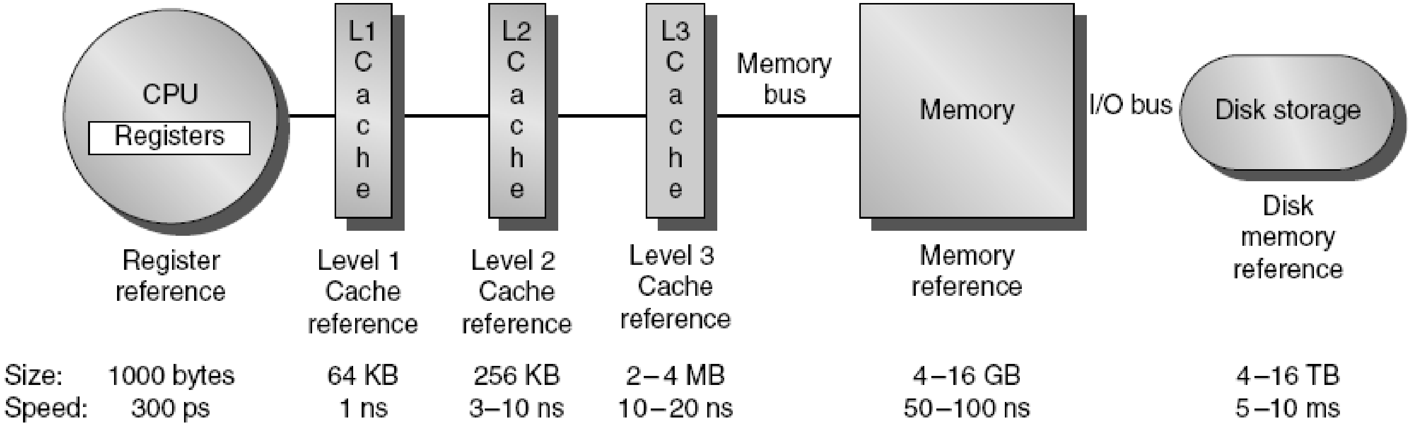
$$t_{avg}=h_1\, t_{C1} + (1-h_1)\, h_2\, t_{C2} + (1-h_1)\,(1-h_2)\, t_M$$

Cache Performance Considerations

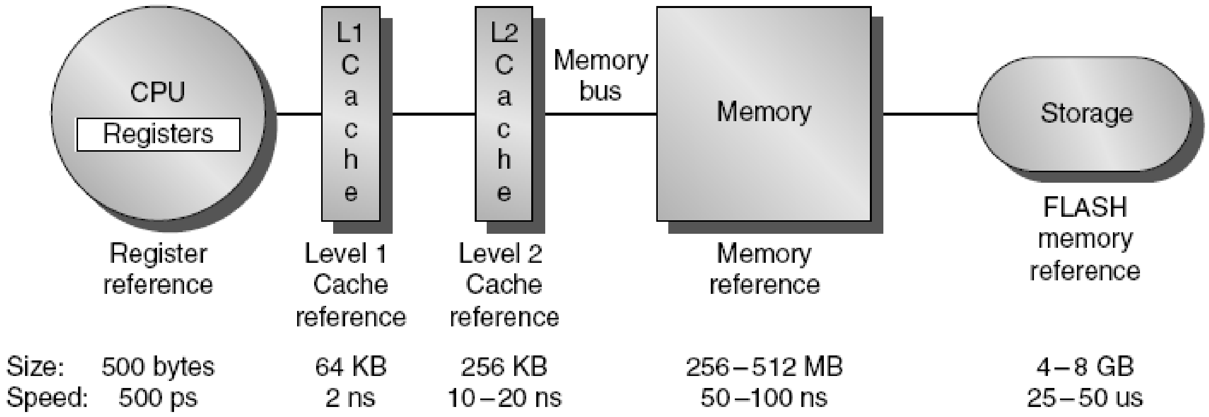
- A cache has a hit rate of 95%, and a block capacity of 128-byte and a cache hit latency of 5 ns. Each word in a block is 32 bits. The main memory takes 100 ns to return a block.
  - What is the cache block size?
  - What is the average memory access time?
- Ans: 1. Cache block size: 32 words

  2. Average memory access time: 9.75 ns

Memory Hierarchy

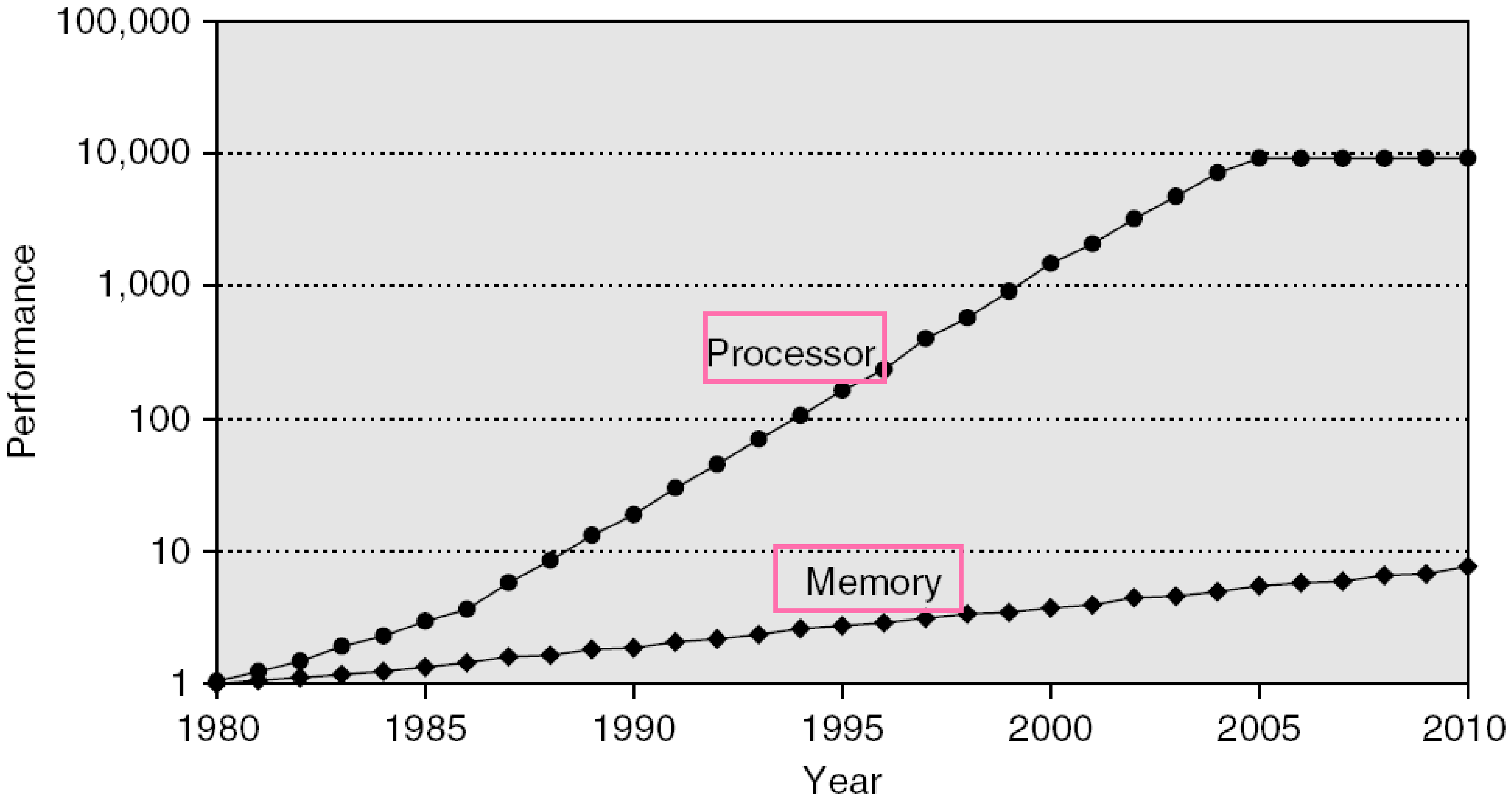


(a) Memory hierarchy for server



(b) Memory hierarchy for a personal mobile device

Memory Performance Gap



Memory Hierarchy Design

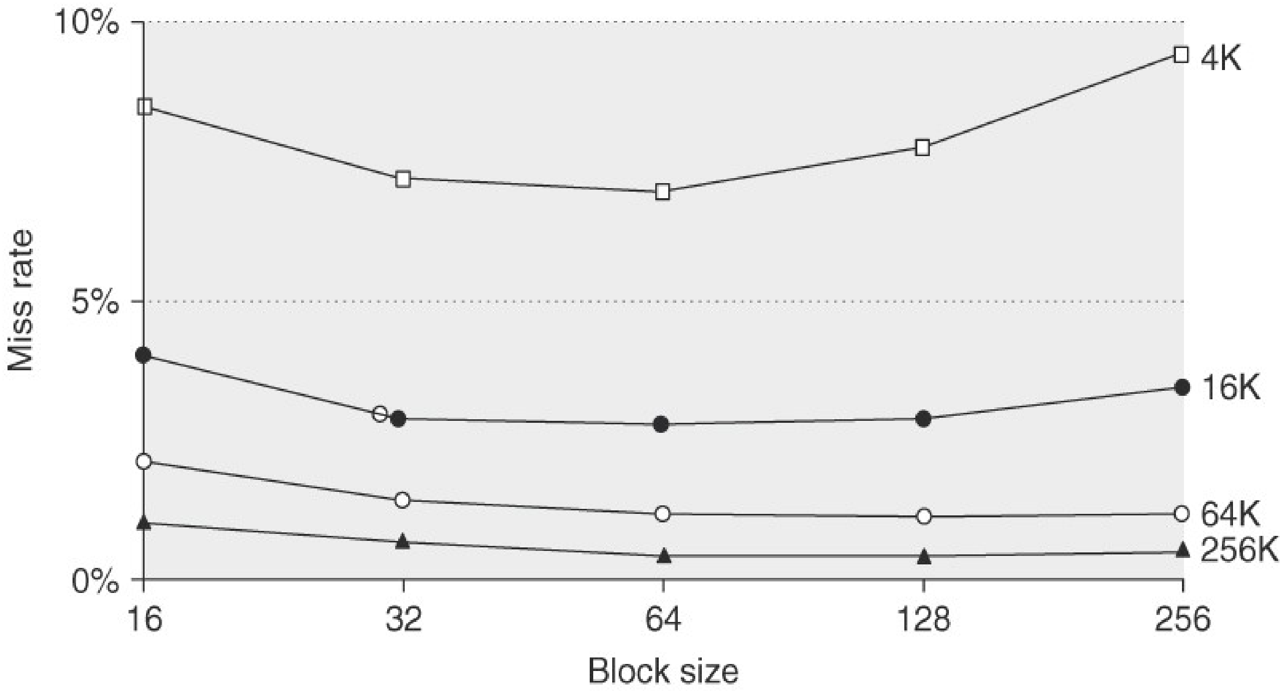
- Memory hierarchy design becomes more crucial with recent multi-core processors:
  - Aggregate peak bandwidth grows with # cores:
    - Intel Core i7 can generate two references per core per clock
    - Four cores and 3.2 GHz clock
      - 25.6 billion 64-bit data references/second +
      - 12.8 billion 128-bit instruction references
      - = 409.6 GB/s!
    - DRAM bandwidth is only 6% of this (25 GB/s)
    - Requires:
      - Multi-port, pipelined caches
      - Two levels of cache per core
      - Shared third-level cache on chip

Memory Hierarchy

- Average memory access time= Hit time+ Miss rate x miss penalty
- basic cache optimizations:
  - Reducing miss rate
    - Larger block size
    - Larger cache
    - Higher associativity
  - Reducing miss penalty
    - Multi level cache
  - Reducing the time to hit in the cache
    - Avoid address translation when indexing the cache

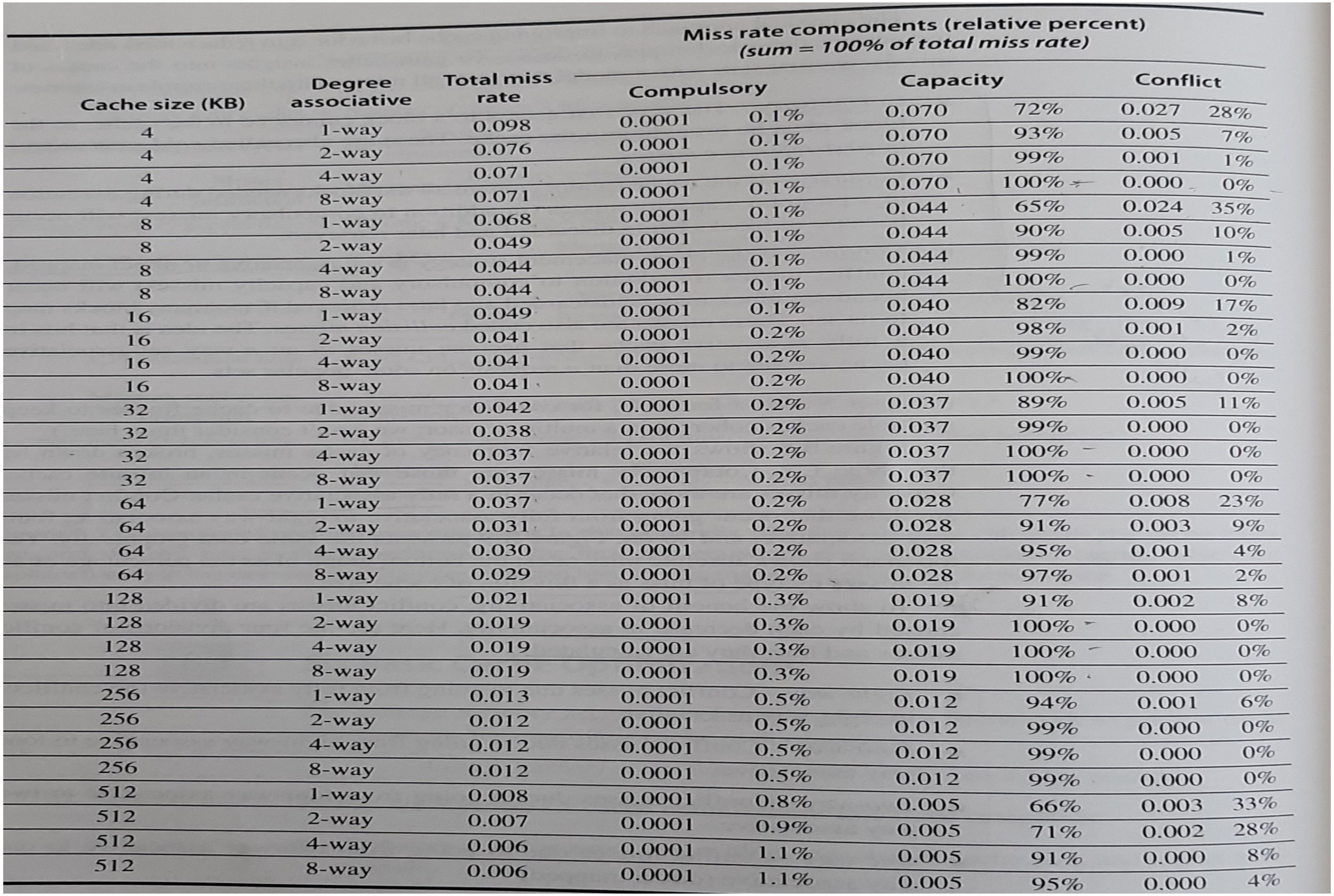
Memory Hierarchy

- basic cache optimizations:
  - Larger block size
    - Reduces compulsory misses
    - Increases capacity and conflict misses, increases miss penalty
    - Average access time is used to decide the size of cache and block



Memory Hierarchy

- Basic cache optimizations:
  - Higher associativity
    - Reduces conflict misses
    - Eight-way set associativity is as effective in reducing miss rate for these sized caches as fully associativity
    - 2:1 cache rule of thumb: Direct map of cache of size N has about the same miss rate as two-way set associativity of size N/2
    - Increases hit time, increases power consumption



			Miss rate components (relative percent) *(sum = 100% of total miss rate)*					
Cache size (KB)	Degree associative	Total miss rate	Compulsory		Capacity		Conflict	
4	1-way	0.098	0.0001	0.1%	0.070	72%	0.027	28%
4	2-way	0.076	0.0001	0.1%	0.070	93%	0.005	7%
4	4-way	0.071	0.0001	0.1%	0.070	99%	0.001	1%
4	8-way	0.071	0.0001	0.1%	0.070	100%	0.000	0%
8	1-way	0.068	0.0001	0.1%	0.044	65%	0.024	35%
8	2-way	0.049	0.0001	0.1%	0.044	90%	0.005	10%
8	4-way	0.044	0.0001	0.1%	0.044	99%	0.000	1%
8	8-way	0.044	0.0001	0.1%	0.044	100%	0.000	0%
16	1-way	0.049	0.0001	0.1%	0.040	82%	0.009	17%
16	2-way	0.041	0.0001	0.2%	0.040	98%	0.001	2%
16	4-way	0.041	0.0001	0.2%	0.040	99%	0.000	0%
16	8-way	0.041	0.0001	0.2%	0.040	100%	0.000	0%
32	1-way	0.042	0.0001	0.2%	0.037	89%	0.005	11%
32	2-way	0.038	0.0001	0.2%	0.037	99%	0.000	0%
32	4-way	0.037	0.0001	0.2%	0.037	100%	0.000	0%
32	8-way	0.037	0.0001	0.2%	0.037	100%	0.000	0%
64	1-way	0.037	0.0001	0.2%	0.028	77%	0.008	23%
64	2-way	0.031	0.0001	0.2%	0.028	91%	0.003	9%
64	4-way	0.030	0.0001	0.2%	0.028	95%	0.001	4%
64	8-way	0.029	0.0001	0.2%	0.028	97%	0.001	2%
128	1-way	0.021	0.0001	0.3%	0.019	91%	0.002	8%
128	2-way	0.019	0.0001	0.3%	0.019	100%	0.000	0%
128	4-way	0.019	0.0001	0.3%	0.019	100%	0.000	0%
128	8-way	0.019	0.0001	0.3%	0.019	100%	0.000	0%
256	1-way	0.013	0.0001	0.5%	0.012	94%	0.001	6%
256	2-way	0.012	0.0001	0.5%	0.012	99%	0.000	0%
256	4-way	0.012	0.0001	0.5%	0.012	99%	0.000	0%
256	8-way	0.012	0.0001	0.5%	0.012	99%	0.000	0%
512	1-way	0.008	0.0001	0.8%	0.005	66%	0.003	33%
512	2-way	0.007	0.0001	0.9%	0.005	71%	0.002	28%
512	4-way	0.006	0.0001	1.1%	0.005	91%	0.000	8%
512	8-way	0.006	0.0001	1.1%	0.005	95%	0.000	4%

2:1 cache rule of thumb: Direct mapped cache of size N has about the same miss rate as a two-way set associative cache of size N/2

Virtual Memory

- Ideally, entire memory hierarchy would appear to the processor as a single memory unit

- In modern computer system, the physical main memory is not as large as the address space spanned by the address issued by the processor

- When a program (or process) does not completely fits into the main memory, parts of it will be there in secondary memory

- In modern computers, operating system moves the data automatically between main memory and secondary storage

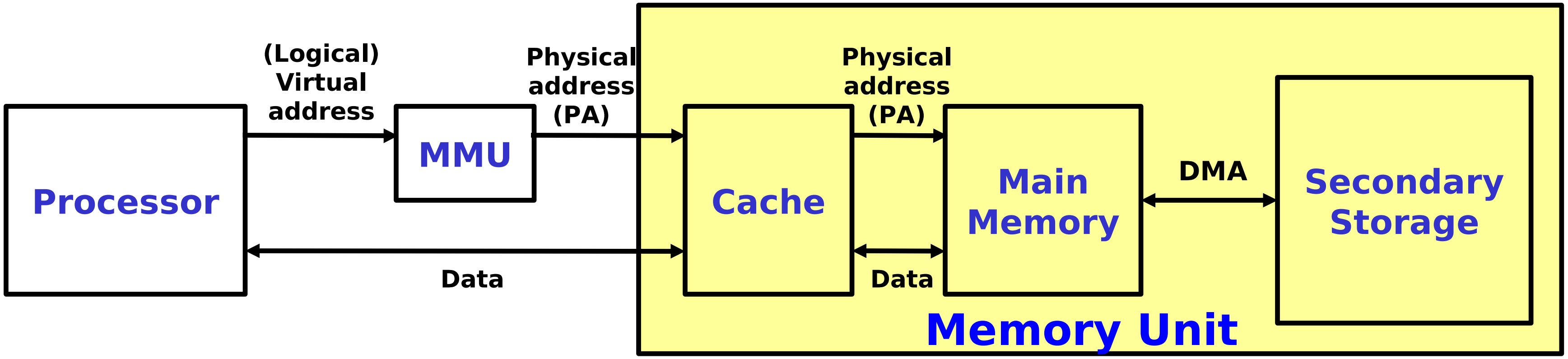
- Programmer does not need to aware of the limitations imposed by the main memory

Virtual Memory Technique

- Technique that automatically move program and data blocks into the physical main memory when they are required for execution
- Using virtual program concept, each program may use entire CPU local address space, at least up to secondary storage
- The address issued by the processor either for instruction or data are called virtual address or logical address
- These addresses are translated into physical memory addresses by a combination of hardware and software

Memory Management Unit (MMU)

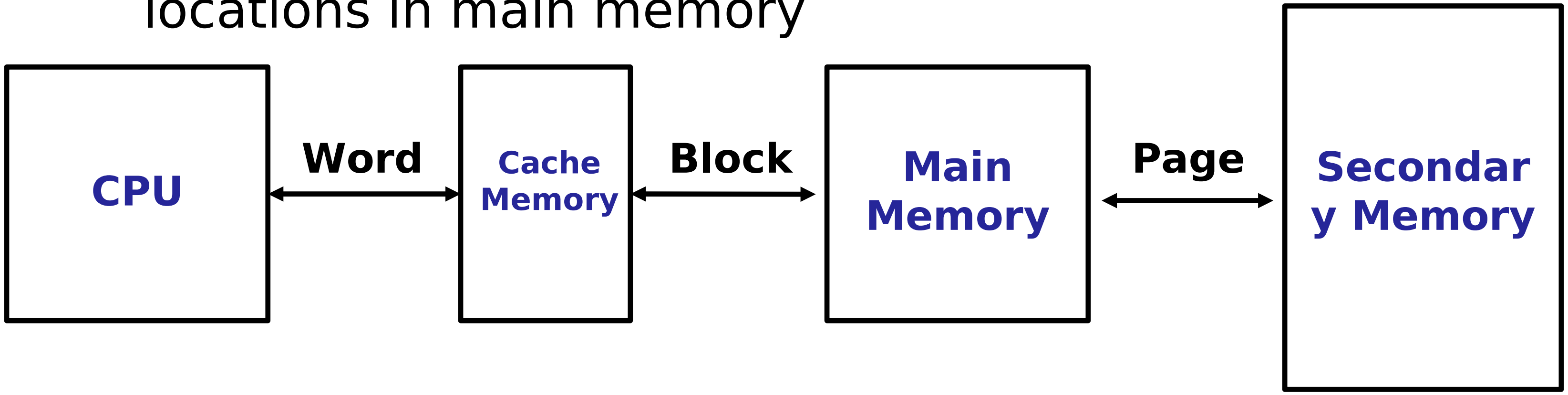
- MMU translate the logical address into physical main memory address
- It is a part of the processor



- If the data is not in main memory, MMU causes the operating system to bring data into memory from the disk
- Transfer of data between disk and main memory is performed using direct memory access (DMA) scheme

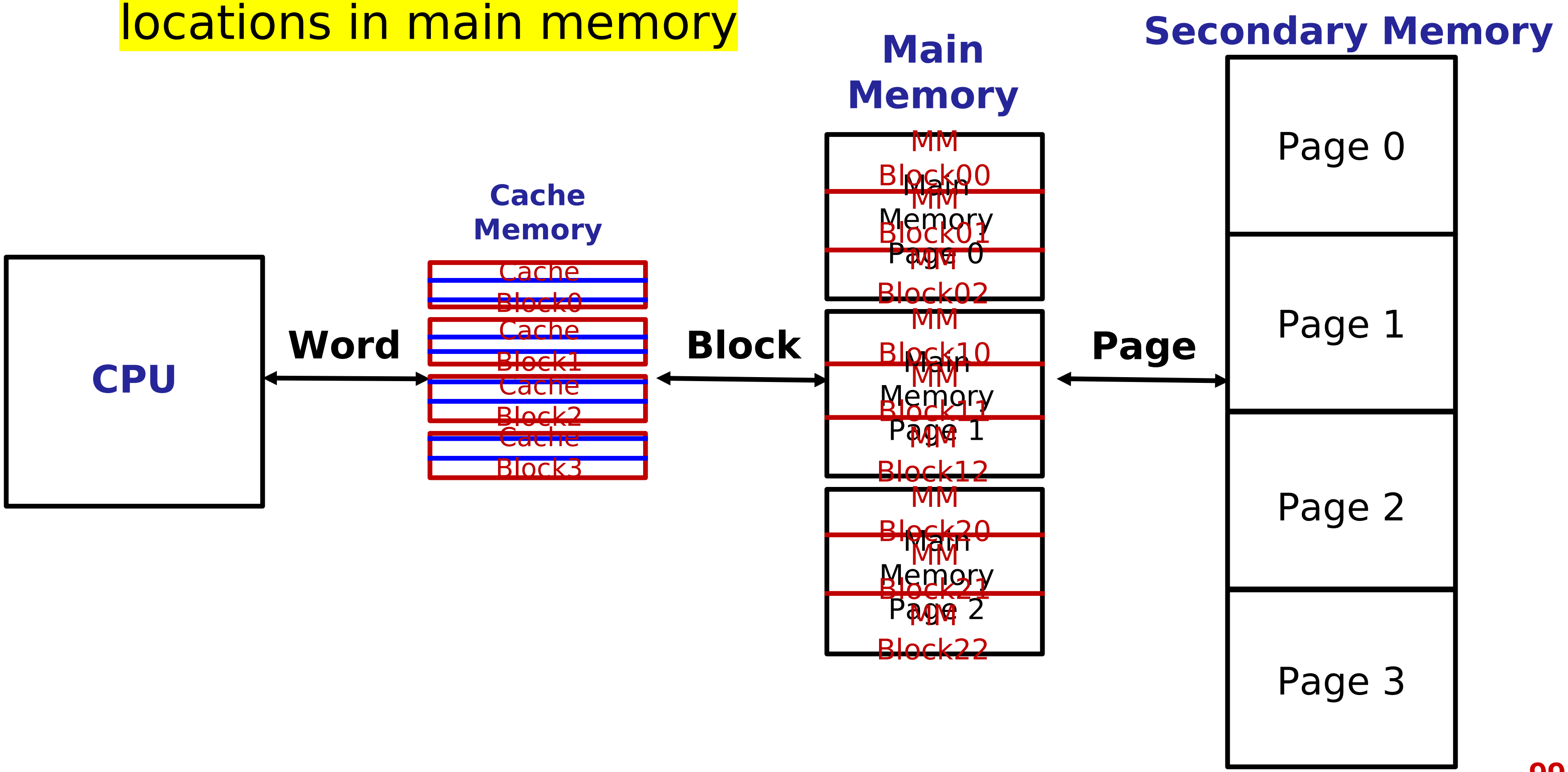
Address Translation

- The virtual memory address translation method based is based on the concept of fixed length pages
- The address translations assumes that programs and data are composed of fixed size units called pages
- Unit of transfer between secondary memory and main memory is page
  - A page is a block of words that occupy contiguous locations in main memory



Address Translation

- The address translations assumes that programs and data are composed of fixed size units called pages
- Unit of transfer between secondary memory and main memory is page
  - A page is a block of words that occupy contiguous locations in main memory



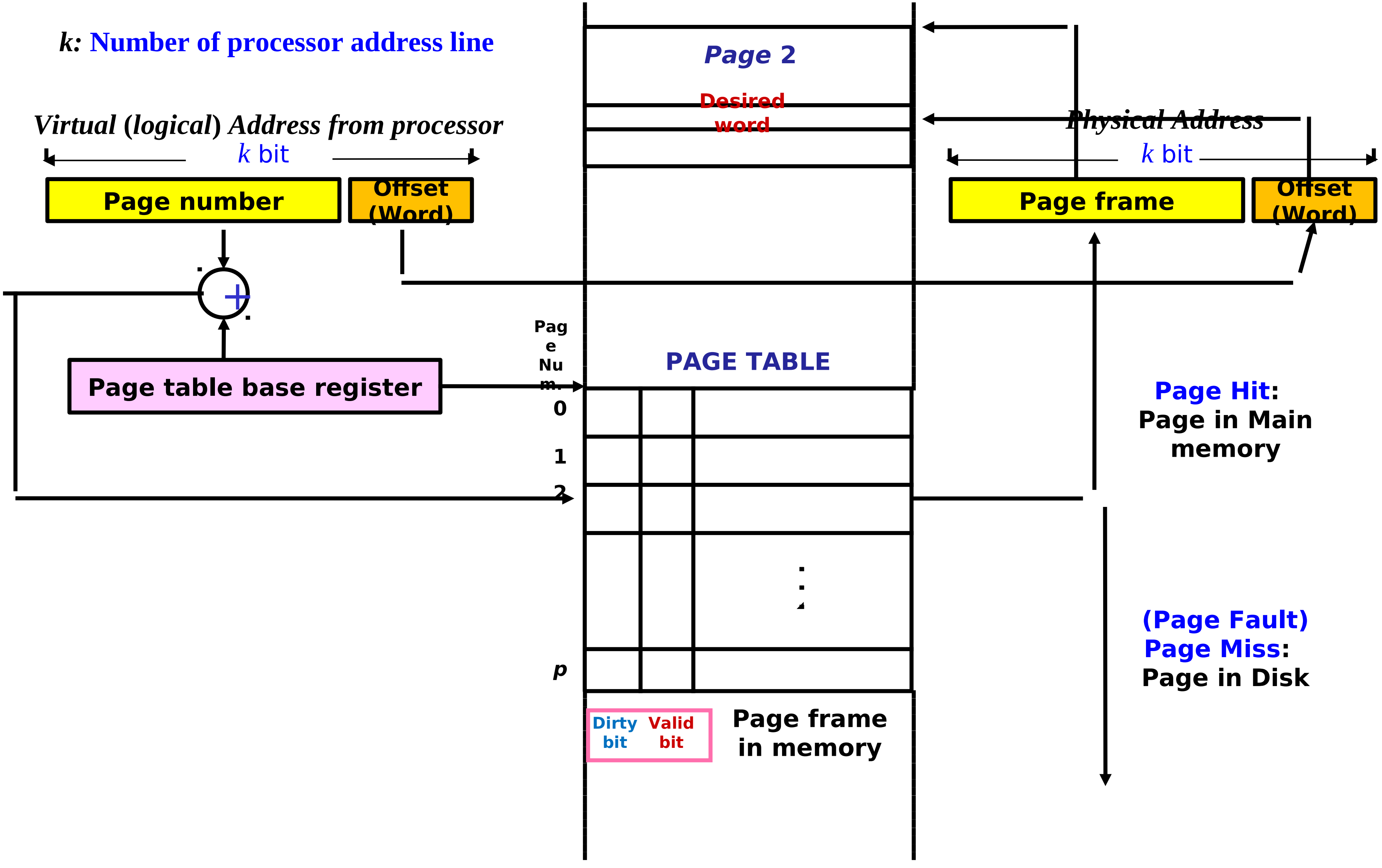
Page

- The programs or data in the disk are seen by the virtual memory as a collection of pages
- This page is the basic unit of information that is moved between the main memory and the secondary memory
- Each page is of the size 2 KB to 16 KB
- Page should not be too small
  - Disk access time is much longer
  - It take considerable time to locate data in the disk
- Page should not be too large
  - Substantial portion of a page may not be used
- Demand paging: Pages are copied to main memory when requested

Parallels Between the Concepts of Cache and Virtual Memory

- Cache:
  - Bridges the speed gap between the processor and the main memory
  - It is implemented in hardware
- Virtual memory mechanism:
  - Bridges the size and speed gap between the main memory and secondary storage
  - It is usually implemented in part by software techniques
- Conceptually, cache techniques and main memory techniques are very similar
- They differ mainly in the details of their implementation

Virtual Memory Address Translation



k: Number of processor address line

Virtual (logical) Address from processor

k bit

Page number

Offset (Word)

Page table base register

Page 2

Desired word

Physical Address

k bit

Page frame

Offset (Word)

Page Num.

0

1

2

p

PAGE TABLE

Dirty bit

Valid bit

Page frame in memory

Page Hit: Page in Main memory

(Page Fault) Page Miss: Page in Disk

Main Memory

Virtual Memory Address Translation

- The virtual address generated by the processor contain page number and offset (word) in the page
- To make sure that required page is in main memory, operating system create **page table for each process**
- This page table is kept in main memory
- Page table base register: Operating system keep the starting address of page table in it
- Page table hold the main memory location for each page
  - The area in main memory that can hold one page is called page frame
- Every entry in page table also include valid bit and dirty bit to describe the status of the page while it is in main memory

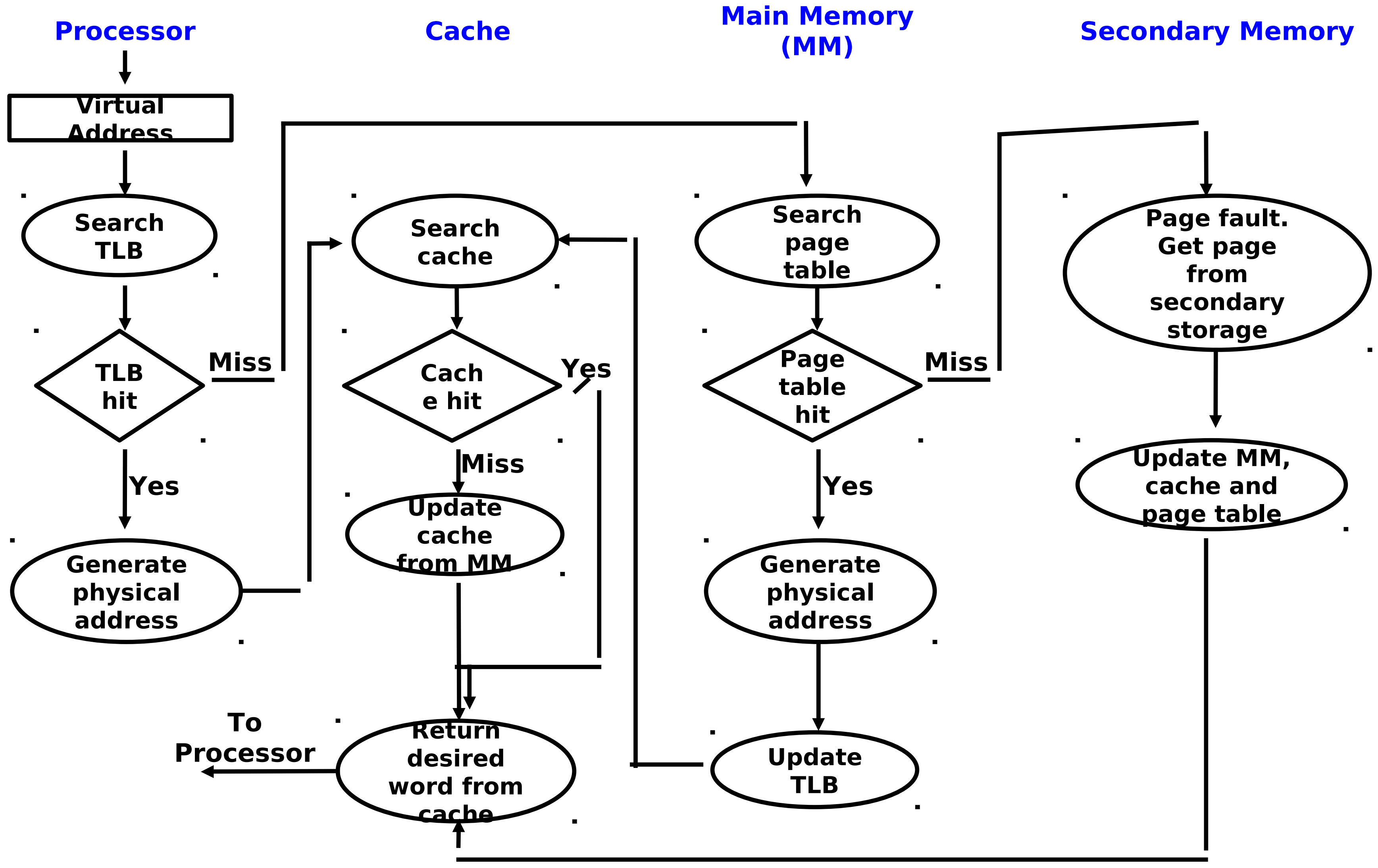
Virtual Memory Address Translation using Translation Lookaside Buffer

- Page table information is used by the MMU for every read and write access
- In order to speed up the address translation procedure, a small cache called Translation Lookaside Buffer (TLB) is incorporated in MMU
- It uses associative/set-associative mapping technique
- It hold a portion of page table corresponding to most recently accessed pages
- TLB holds only the page number and page frame number

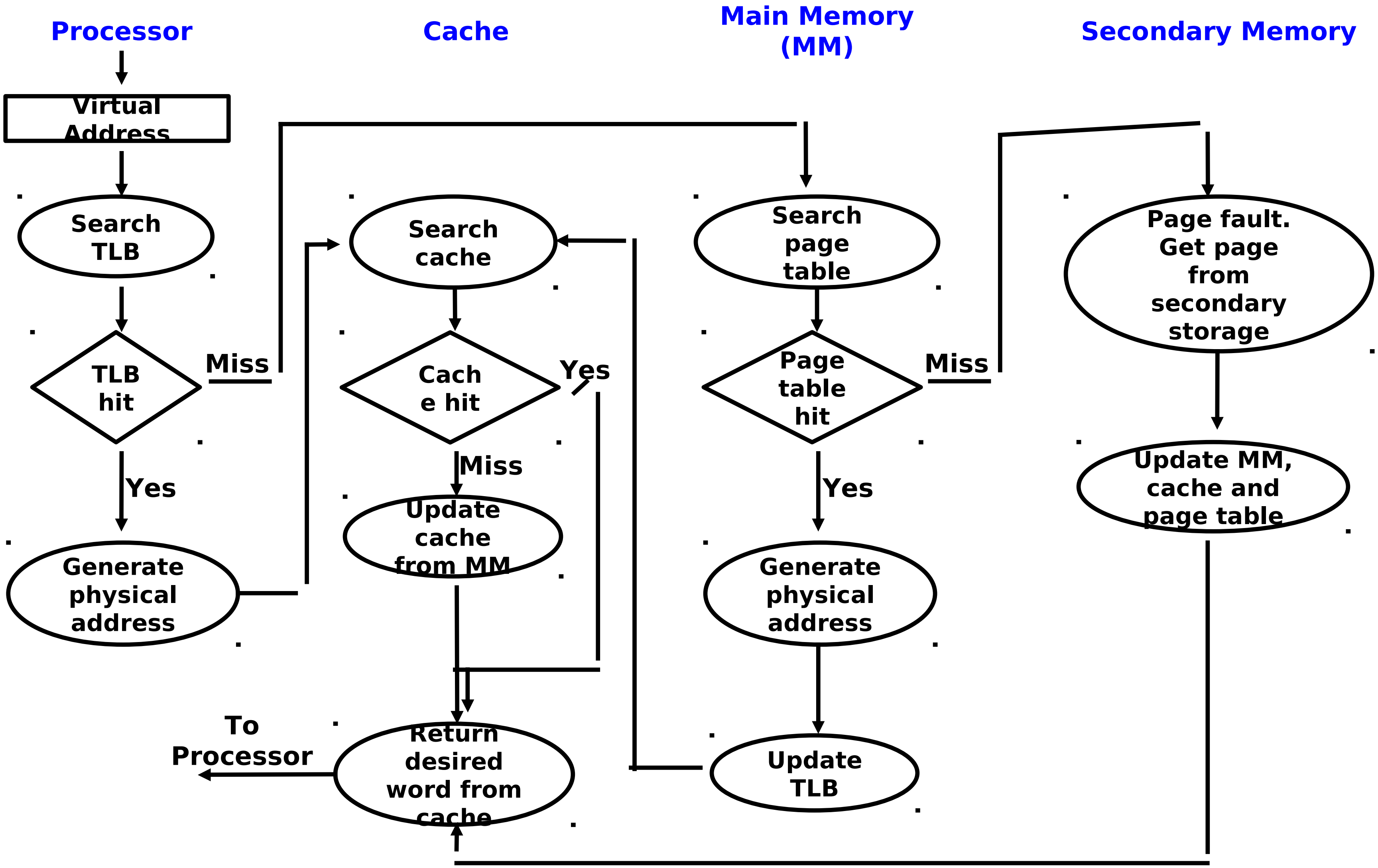
Virtual Memory Address Translation

- Page table information is used by the MMU for every read and write access
- Page fault: Whenever a requested page is not present in the main memory, page fault is said to have occurred
- When a page fault occurs, MMU asks operating system to intervene and raise an exception (interrupt)
  - Process in active get interrupted and control goes to operating system
  - Operating system then copies the requested page from disk to main memory
  - Then returns the control to the interrupted task
- During write operation pages get modified are indicated by dirty bit
- Modified page has to be written back to disk before removed from main memory
- Uses write back policy only

Operation of Memory Hierarchy and Virtual Memory Technique



Operation of Memory Hierarchy and Virtual Memory Technique



Processor

Virtual Address

Search TLB

TLB hit

Miss

Yes

Generate physical address

To Processor

Cache

Search cache

Cach e hit

Yes

Miss

Update cache from MM

Return desired word from cache

Main Memory (MM)

Search page table

Page table hit

Miss

Yes

Generate physical address

Update TLB

Secondary Memory

Page fault. Get page from secondary storage

Update MM, cache and page table