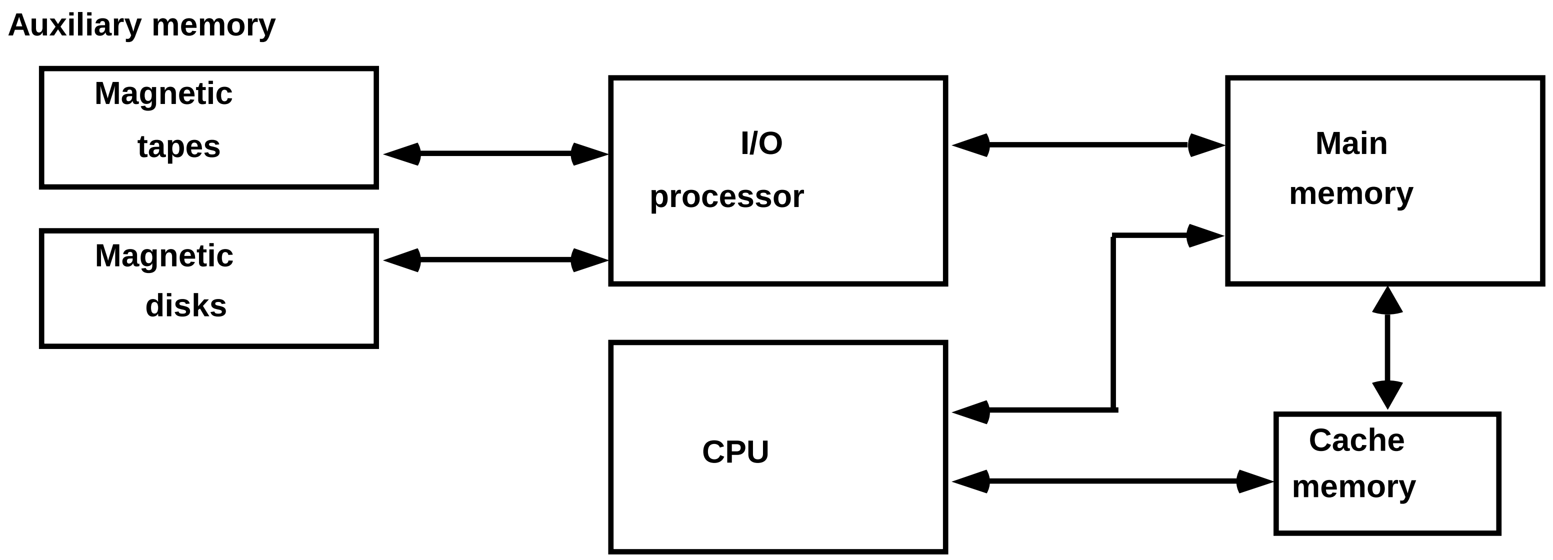
BCS-29
Advanced Computer Architecture

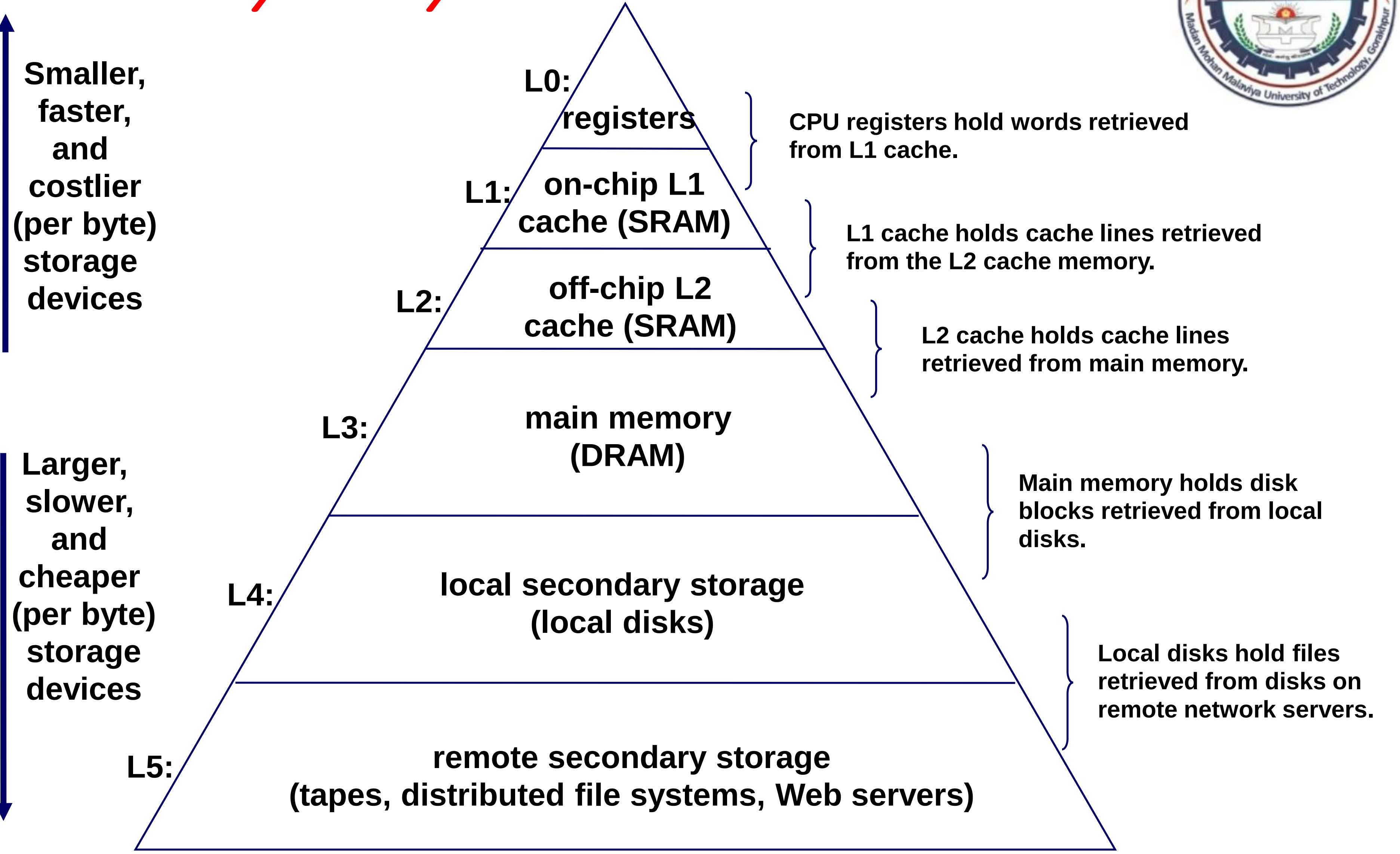
Memory Subsystem

Memory Hierarchy

Memory Subsystem



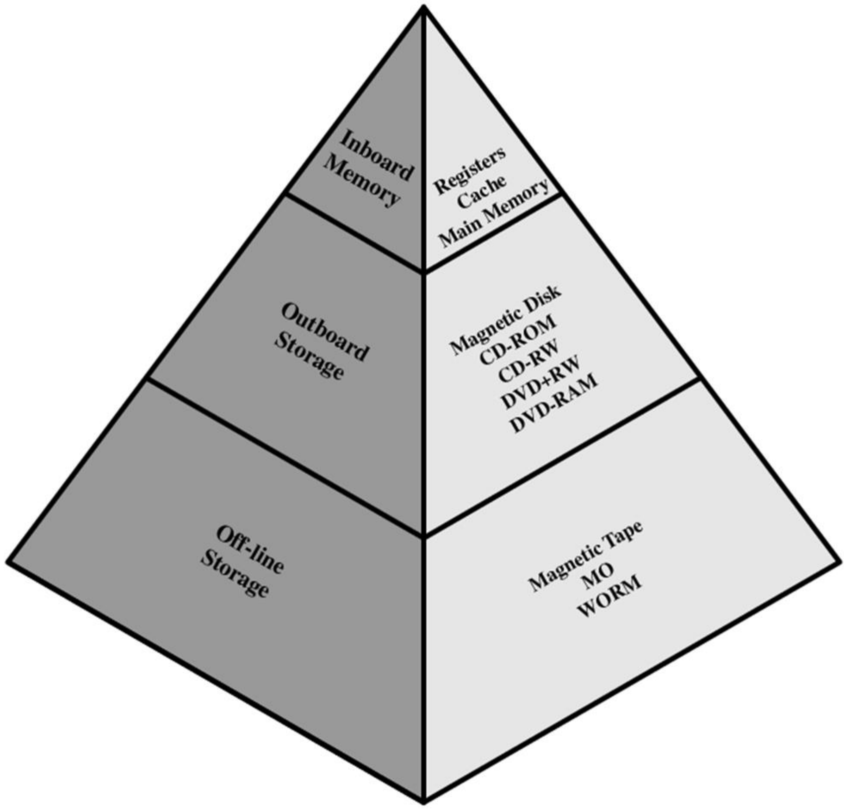
Memory Subsystem



Smaller, faster, and costlier (per byte) storage devices
Larger, slower, and cheaper (per byte) storage devices
L0:
registers
L1:
on-chip L1 cache (SRAM)
L2:
off-chip L2 cache (SRAM)
L3:
main memory (DRAM)
L4:
local secondary storage (local disks)
L5:
remote secondary storage (tapes, distributed file systems, Web servers)
CPU registers hold words retrieved from L1 cache.
L1 cache holds cache lines retrieved from the L2 cache memory.
L2 cache holds cache lines retrieved from main memory.
Main memory holds disk blocks retrieved from local disks.
Local disks hold files retrieved from disks on remote network servers.

Memory Hierarchy

- Registers
  - In CPU
- Internal or Main memory
  - May include one or more levels of cache
  - “RAM”
- External memory
  - Backing store



Internal Memory Types

Memory Type	Category	Erasure	Write Mechanism	Volatility
Random-access memory (RAM)	Read-write memory	Electrically, byte-level	Electrically	Volatile
Read-only memory (ROM)	Read-only memory	Not possible	Masks	Nonvolatile
Programmable ROM (PROM)			Electrically	
Erasable PROM (EPROM)	Read-mostly memory	UV light, chip-level		
Electrically Erasable PROM (EEPROM)		Electrically, byte-level		
Flash memory		Electrically, block-level		

External Memory Types

- HDD
  - Magnetic Disk(s)
  - SDD (Solid State Disk(s))

- Optical
  - CD-ROM
  - CD-Recordable (CD-R)
  - CD-R/W
  - DVD

- Magnetic Tape

Hierarchical Memory Technology

- Memory in system is usually characterized as appearing at various levels (0, 1, …) in a hierarchy, with level 0 being CPU registers and level 1 being the cache closest to the CPU.
- Each level is characterized by five parameters:
  - access time t_i (round-trip time from CPU to ith level)
  - memory size s_i (number of bytes or words in the level)
  - cost per byte c_i
  - transfer bandwidth b_i (rate of transfer between levels)
  - unit of transfer x_i (grain size for transfers)

Memory Generalities

- It is almost always the case that memories at lower-numbered levels, when compare to those at higher-numbered levels
    - are faster to access,
    - are smaller in capacity,
    - are more expensive per byte,
    - have a higher bandwidth, and
    - have a smaller unit of transfer.
- In general, then, $t_{i-1} < t_i$, $s_{i-1} < s_i$, $c_{i-1} > c_i$, $b_{i-1} > b_i$, and $x_{i-1} < x_i$.

The Inclusion Property

- The inclusion property is stated as:

  $M_1 \subset M_2 \subset \ldots \subset M_n$

  The implication of the inclusion property is that all items of information in the "innermost" memory level (cache) also appear in the outer memory levels.

- The inverse, however, is not necessarily true. That is, the presence of a data item in level M_{i+1} does not imply its presence in level M_i. We call a reference to a missing item a "miss."

The Coherence Property

- The inclusion property is, of course, never completely true, but it does represent a desired state. That is, as information is modified by the processor, copies of that information should be placed in the appropriate locations in outer memory levels.
- The requirement that copies of data items at successive memory levels be consistent is called the "coherence property."

Coherence Strategies

- Write-through
  - As soon as a data item in M_i is modified, immediate update of the corresponding data item(s) in M_{i+1}, M_{i+2}, … M_n is required. This is the most aggressive (and expensive) strategy.
- Write-back
  - The update of the data item in M_{i+1} corresponding to a modified item in M_i is not updated unit it (or the block/page/etc. in M_i that contains it) is replaced or removed. This is the most efficient approach, but cannot be used (without modification) when multiple processors share M_{i+1}, …, M_n.

Locality of References

- In most programs, memory references are assumed to occur in patterns that are strongly related (statistically) to each of the following:
  - *Temporal locality* – if location M is referenced at time t, then it (location M) will be referenced again at some time t+Δt.
  - *Spatial locality* – if location M is referenced at time t, then another location M±Δm will be referenced at time t+Δt.
  - *Sequential locality* – if location M is referenced at time t, then locations M+1, M+2, … will be referenced at time t+Δt, t+Δt', etc.
- In each of these patterns, both Δm and Δt are "small."
- H&P suggest that 90 percent of the execution time in most programs is spent executing only 10 percent of the code.

Working Sets

- The set of addresses (bytes, pages, etc.) referenced by a program during the interval from t to t+ω, where ω is called the *working set parameter*, changes slowly.
- This set of addresses, called the *working set*, should be present in the higher levels of M if a program is to execute efficiently (that is, without requiring numerous movements of data items from lower levels of M). This is called the *working set principle*.

Hit Ratios

- When a needed item (instruction or data) is found in the level of the memory hierarchy being examined, it is called a *hit*. Otherwise (when it is not found), it is called a *miss* (and the item must be obtained from a lower level in the hierarchy).
- The *hit ratio*, *h*, for M_i is the probability (between 0 and 1) that a needed data item is found when sought in level memory M_i.
- The *miss ratio* is obviously just $1-h_i$.
- We assume $h_0 = 0$ and $h_n = 1$.

Access Frequencies

- The access frequency f_i to level M_i is $(1-h_1) \times (1-h_2) \times \ldots \times h_i$.

- Note that $f_1 = h_1$, and

$$\sum_{i=1}^{n} f_i = 1$$

Effective Access Times

- There are different penalties associated with misses at different levels in the memory hierarcy.
  - A cache miss is typically 2 to 4 times as expensive as a cache hit (assuming success at the next level).
  - A page fault (miss) is 3 to 4 magnitudes as costly as a page hit.
- The effective access time of a memory hierarchy can be expressed as

$$
\begin{aligned}
T_{eff} &= \sum_{i=1}^{n} f_i \cdot t_i \\
&= h_1 t_1 + (1-h_1) h_2 t_2 + \cdots + (1-h_1)(1-h_2)\cdots(1-h_{n-1}) h_n t_n
\end{aligned}
$$

- The first few terms in this expression dominate, but the effective access time is still dependent on program behavior and memory design choices.

Memory Technologies

Memory Components

- **Static RAM** Static RAM (SRAM) is a memory technology based on flip-flops. SRAM has an access time of 2 – 10 nanoseconds. From a logical view, all of main memory can be viewed as fabricated from SRAM, although such a memory would be unrealistically expensive.
- **Dynamic RAM** Dynamic RAM (DRAM) is a memory technology based on capacitors – electrical elements that store electronic charge. Dynamic RAM is cheaper than static RAM and can be packed more densely on a computer chip, thus allowing larger capacity memories. DRAM has an access time (to be defined later) in the order of 60 nanoseconds, slower than SRAM.
- **DDR SDRAM (double data rate synchronous dynamic random access memory)**
- **DDR2 SDRAM (double data rate two synchronous dynamic random access memory)**

How does cache memory work?

- Main memory consists of up to 2^n addressable words, with each word having a unique *n*-bit address. For mapping purposes, this memory is considered to consist of a number of fixed-length blocks of *K* words each. Thus, there are $M = 2^n / K$ blocks.
- The cache is split into *C lines* of *K* words each, Figure with number of lines considerably smaller than the number of main memory blocks (C << M).
- At any time, only a subset of the blocks of main memory resides in the lines in the cache.
- If a word in a block of memory is read, that block is transferred to one of the lines of the cache.

Random Access Memory (RAM)

- Misnamed as all semiconductor memory is random access
- Read/Write
- Volatile
- Temporary storage
- Static or dynamic

Types of RAM

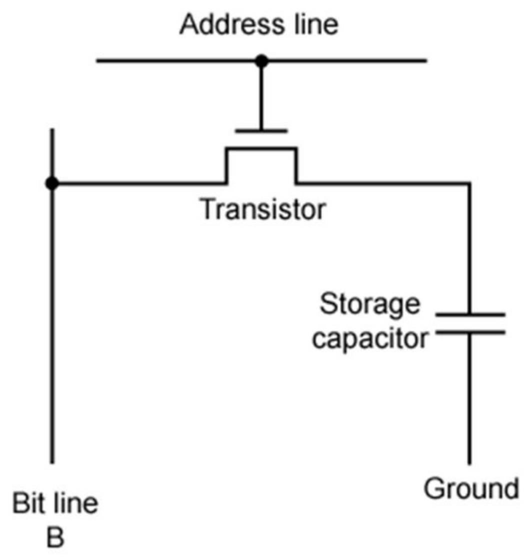
- Dynamic RAM (**DRAM**) – are like leaky capacitors; initially data is stored in the DRAM chip, charging its memory cells to maximum values. The charge slowly leaks out and eventually would go to low to represent valid data; before this happens, a **refresh** circuitry reads the contents of the DRAM and rewrites the data to its original locations, thus restoring the memory cells to their maximum charges
- Static RAM (**SRAM**) – is more like a register; once the data has been written, it will stay valid, it doesn't have to be refreshed. Static RAM is faster than DRAM, also more expensive. Cache memory in PCs is constructed from SRAM memory.

Dynamic RAM

- Bits stored as charge in capacitors
  - Charges leak
  - Need refreshing even when powered
- Simpler construction
- Smaller per bit than SRAM
  - Less expensive
- Need refresh circuits
- Slower
- Used for main memory in computing systems
- Essentially analogue
  - Level of charge determines value

DRAM Structure & Operation

- Address line active when bit read or written
  - Transistor switch closed (current flows)
- Write
  - Voltage to bit line
    - High for 1 low for 0
  - Then signal address line
    - Transfers charge to capacitor
- Read
  - Address line selected
    - transistor turns on
  - Charge from capacitor fed via bit line to sense amplifier
    - Compares with reference value to determine 0 or 1
  - Capacitor charge must be restored



DRAM Refreshing

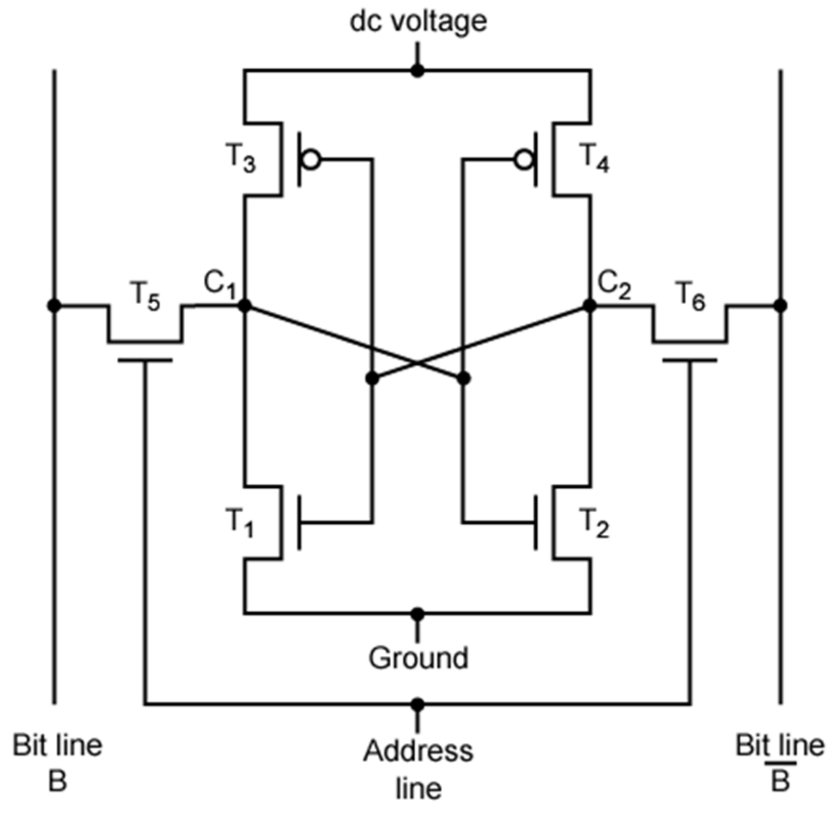
- Refresh circuit included on chip
  - Disable memory array chip
  - Count through rows and select each in turn
  - Read contents & write it back (restore)
- Takes time
- Slows down apparent performance

Static RAM

- Bits stored as on/off switches
- No charges to leak
- No refreshing needed when powered
- More complex construction
- Larger per bit
  - More expensive
- Does not need refresh circuits
- Faster
  - Cache
- Digital
  - Uses flip-flops

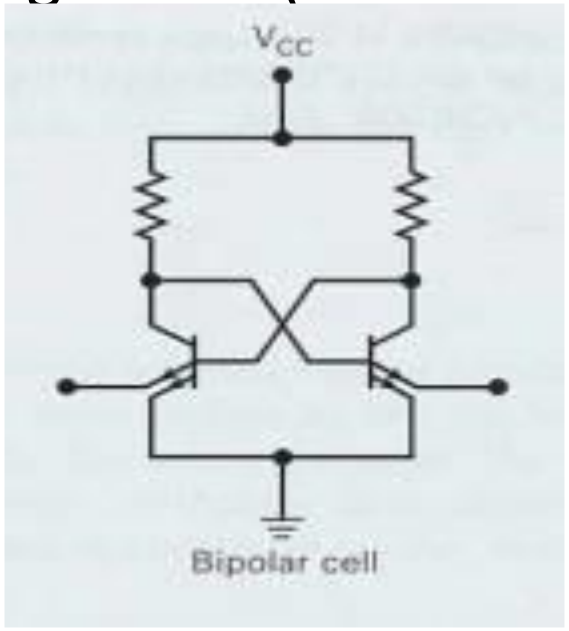
Static RAM Structure & Operation

- Transistor arrangement gives stable logic state
- State 1
  - C_1 high, C_2 low
  - T_1 T_4 off, T_2 T_3 on
- State 0
  - C_2 high, C_1 low
  - T_2 T_3 off, T_1 T_4 on
- Address line transistors T_5 T_6 is switch
- Write – apply value to B & compliment to B
- Read – value is on line B



Static RAM cell

- Cache is much faster than main memory because it is implemented using SRAM (Static Random Access Memory).



SRAM v DRAM

- Both volatile
  - Power needed to preserve data
- Dynamic cell
  - Simpler to build, smaller
  - More dense
  - Less expensive
  - Needs refresh
  - Larger memory units
- Static
  - Faster
  - Cache

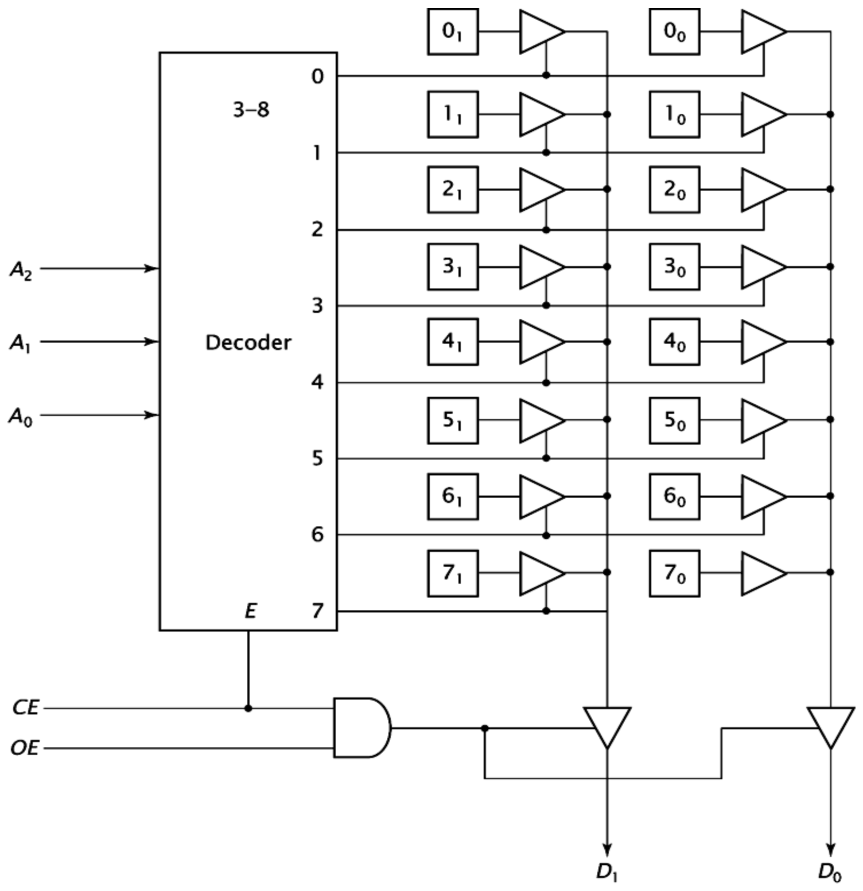
Read Only Memory (ROM)

- Provides permanent storage (nonvolatile)

- Used for: microprogramming, library subroutines (code) and constant data, systems programs (BIOS for PC or entire application + OS for certain embedded systems)

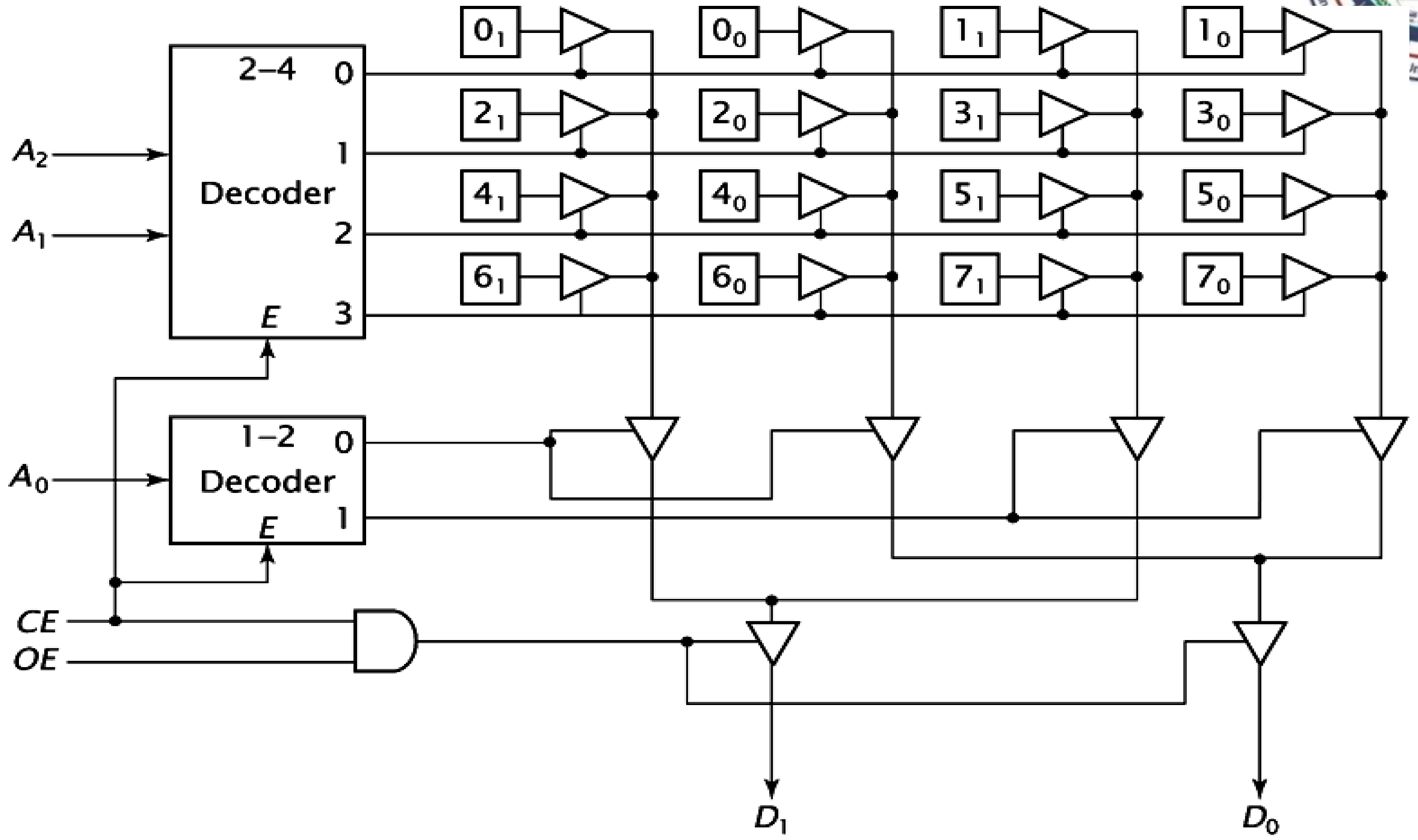
- Types
  - Written during manufacture (very expensive for small runs)
  - Programmable (once) PROM (needs special equipment to program)
  - Read “mostly”
    - Erasable Programmable (EPROM) - Erased by UV
    - Electrically Erasable (EEPROM) - Takes much longer to write than read
    - Flash memory - Erase whole memory electrically

Internal linear organization



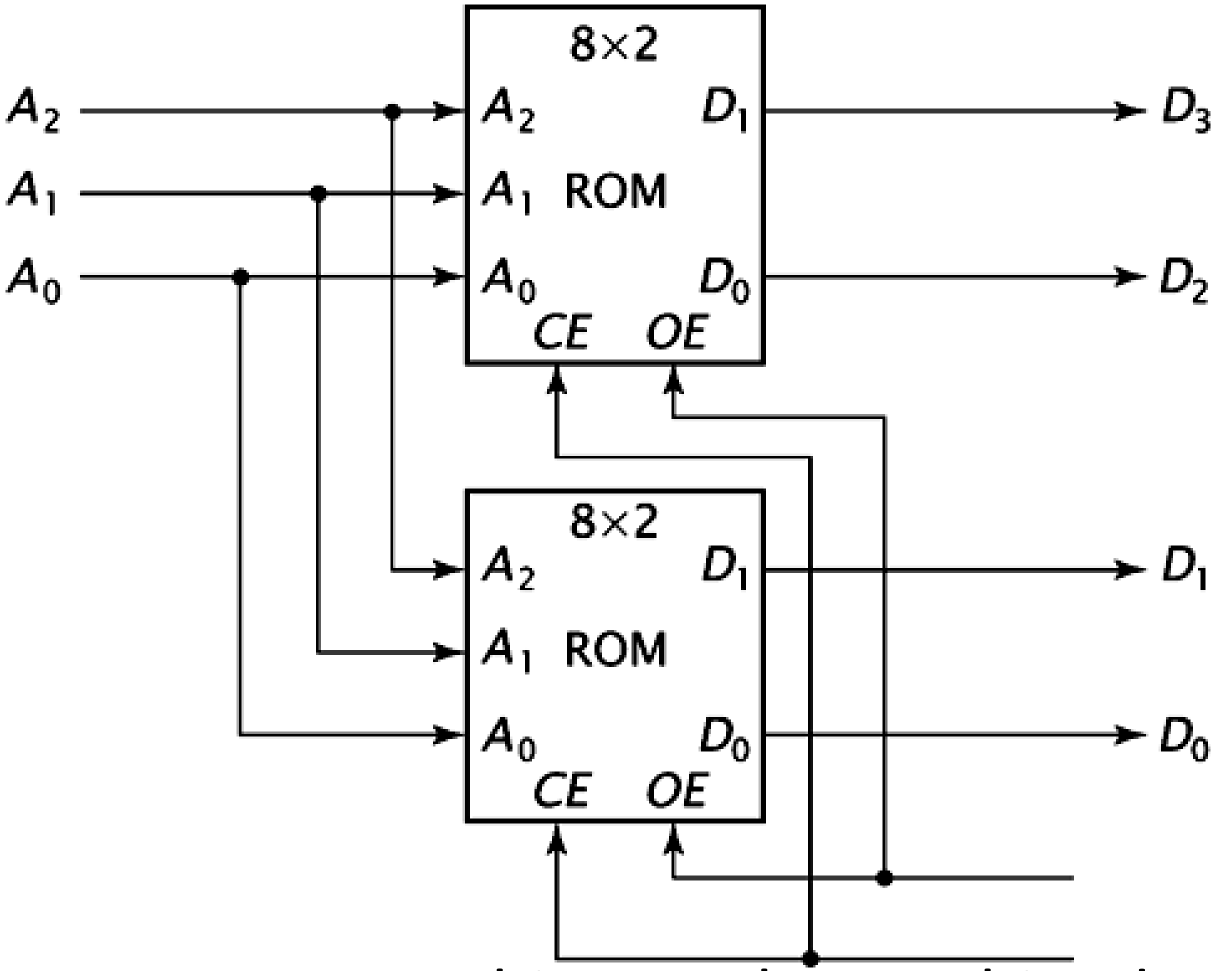
- 8X2 ROM chip
- As the number of locations increases, the size of the address decoder needed, becomes very large
- Multiple dimensions of decoding can be used to overcome this problem

Internal two-dimensional organization



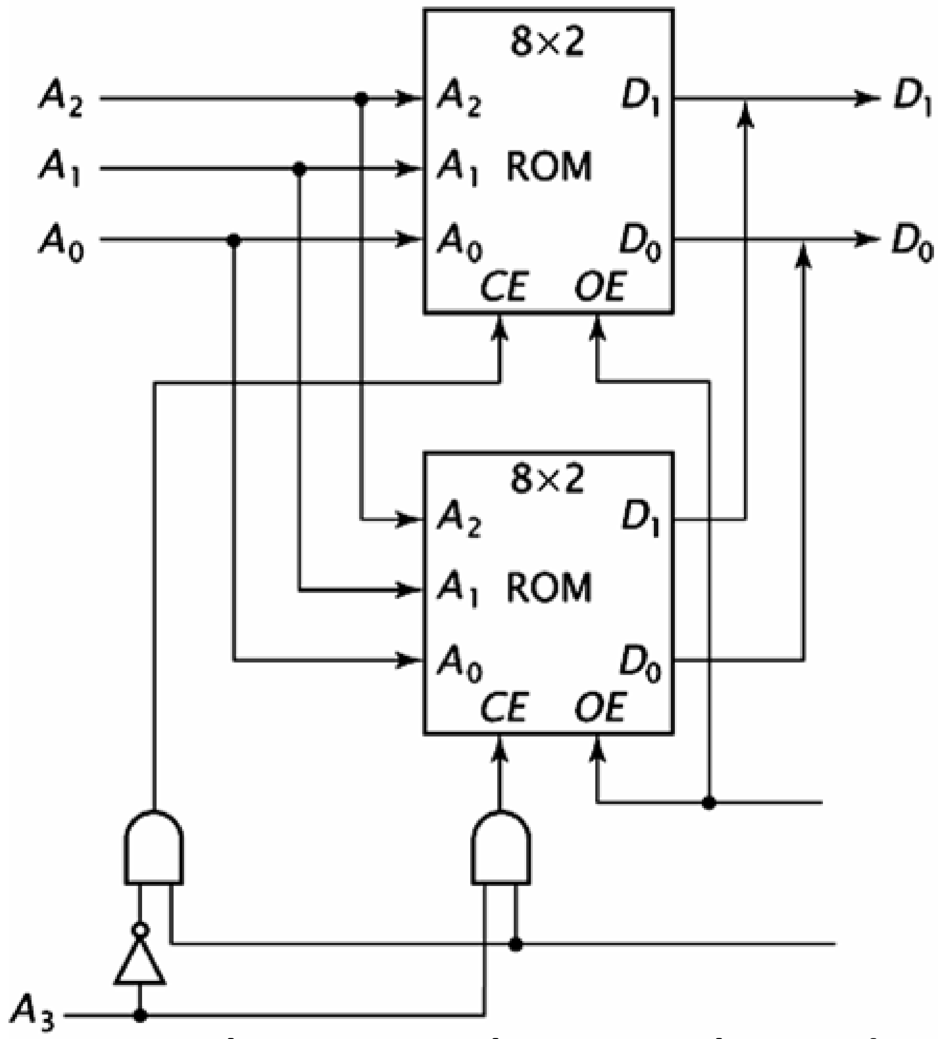
- High order address bits (A2A1) select one of the rows
- The low order address bit selects one of the two locations in the row

Memory Subsystems Organization (1)

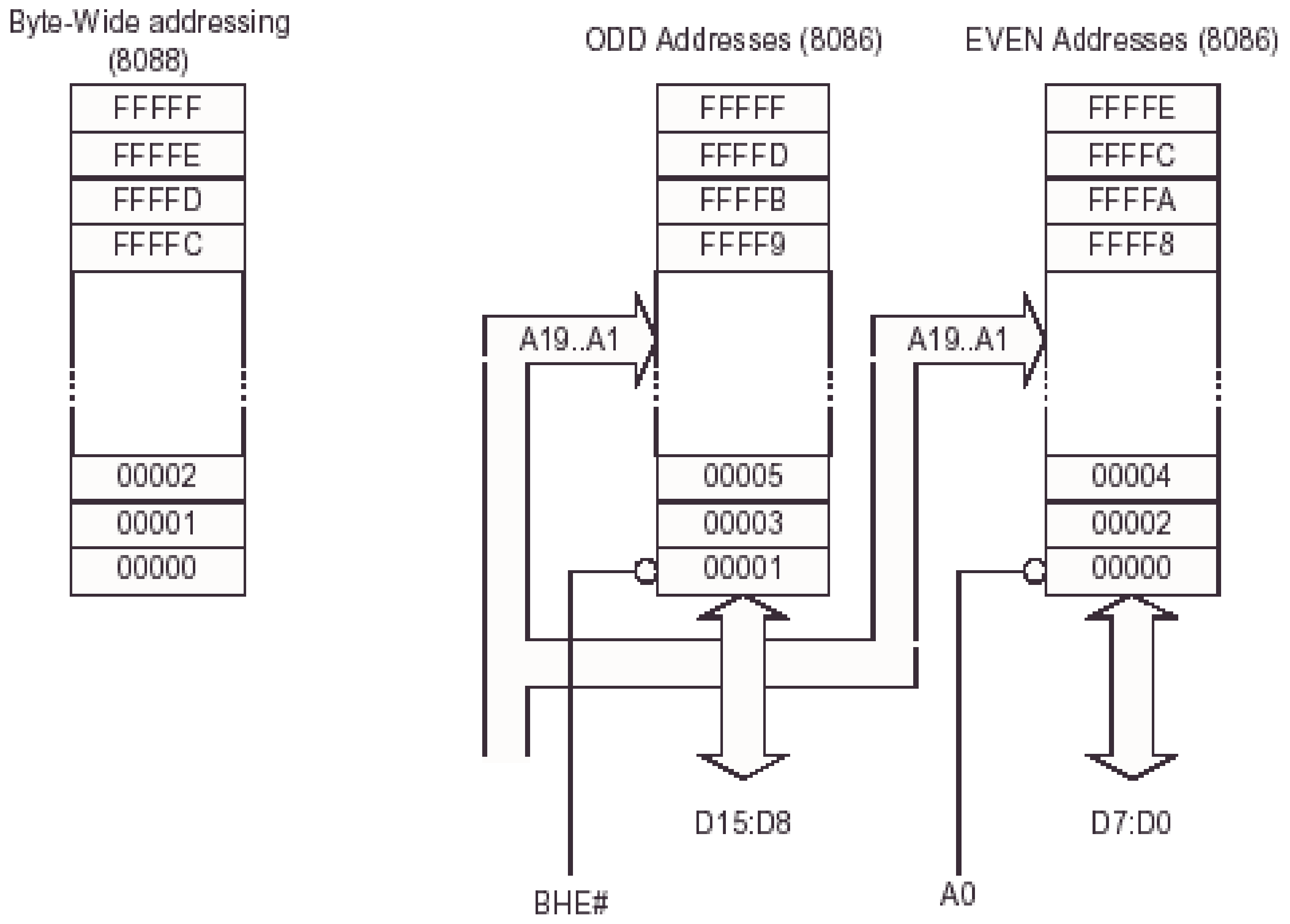


- Two or more memory chips can be combined to create memory with more bits per location (two 8X2 chips can create a 8X4 memory)

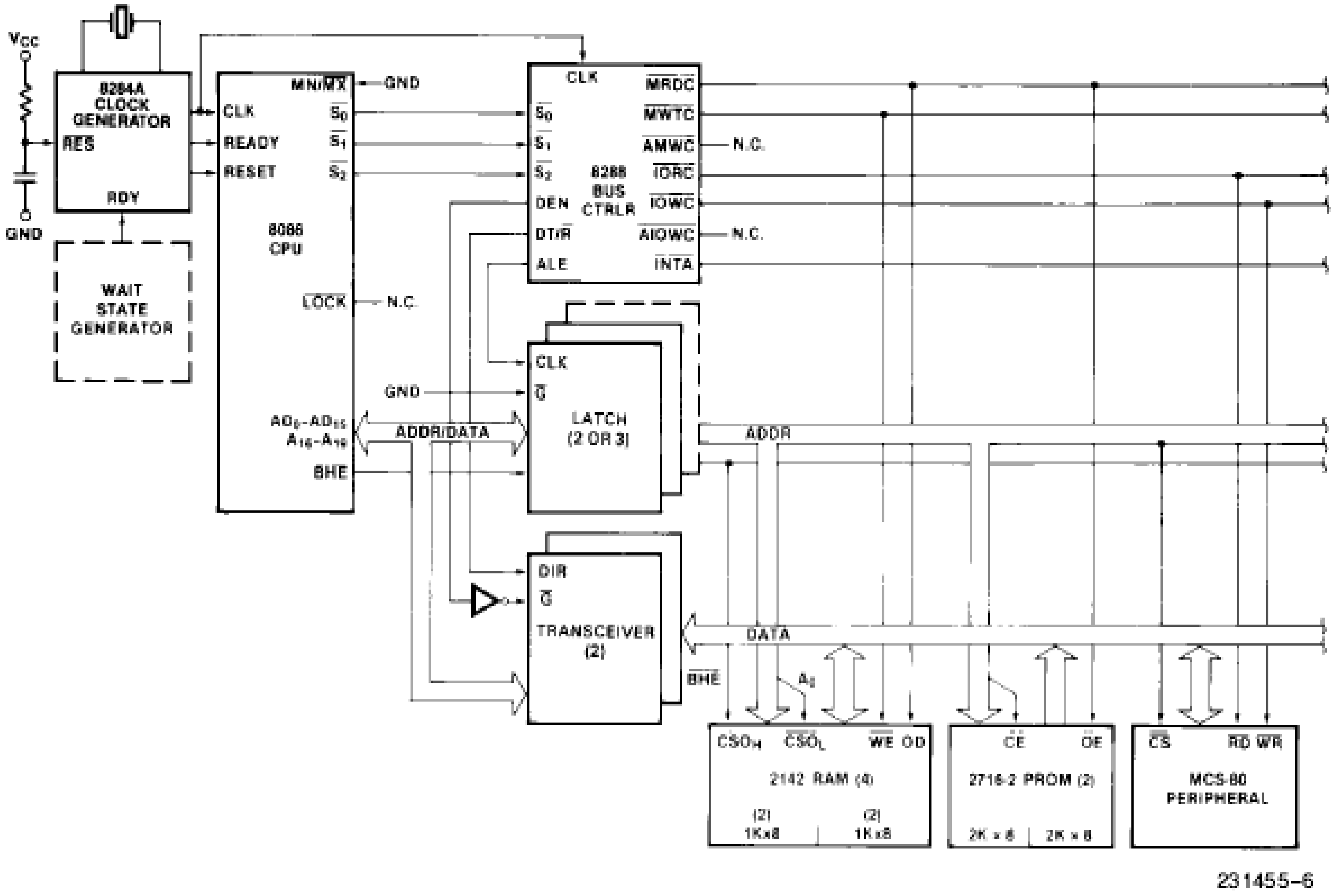
Memory Subsystems Organization (2)



- Two or more memory chips can be combined to create more locations (two 8X2 chips can create 16X2 memory)


Byte-Wide addressing (8088)
FFFFF
FFFFE
FFFFD
FFFFC
00002
00001
00000
ODD Addresses (8086)
FFFFF
FFFFD
FFFFB
FFFF9
00005
00003
00001
EVEN Addresses (8086)
FFFFE
FFFFC
FFFFA
FFFF8
00004
00002
00000
A19..A1
A19..A1
D15:D8
D7:D0
BHE#
A0

Processor and Memory



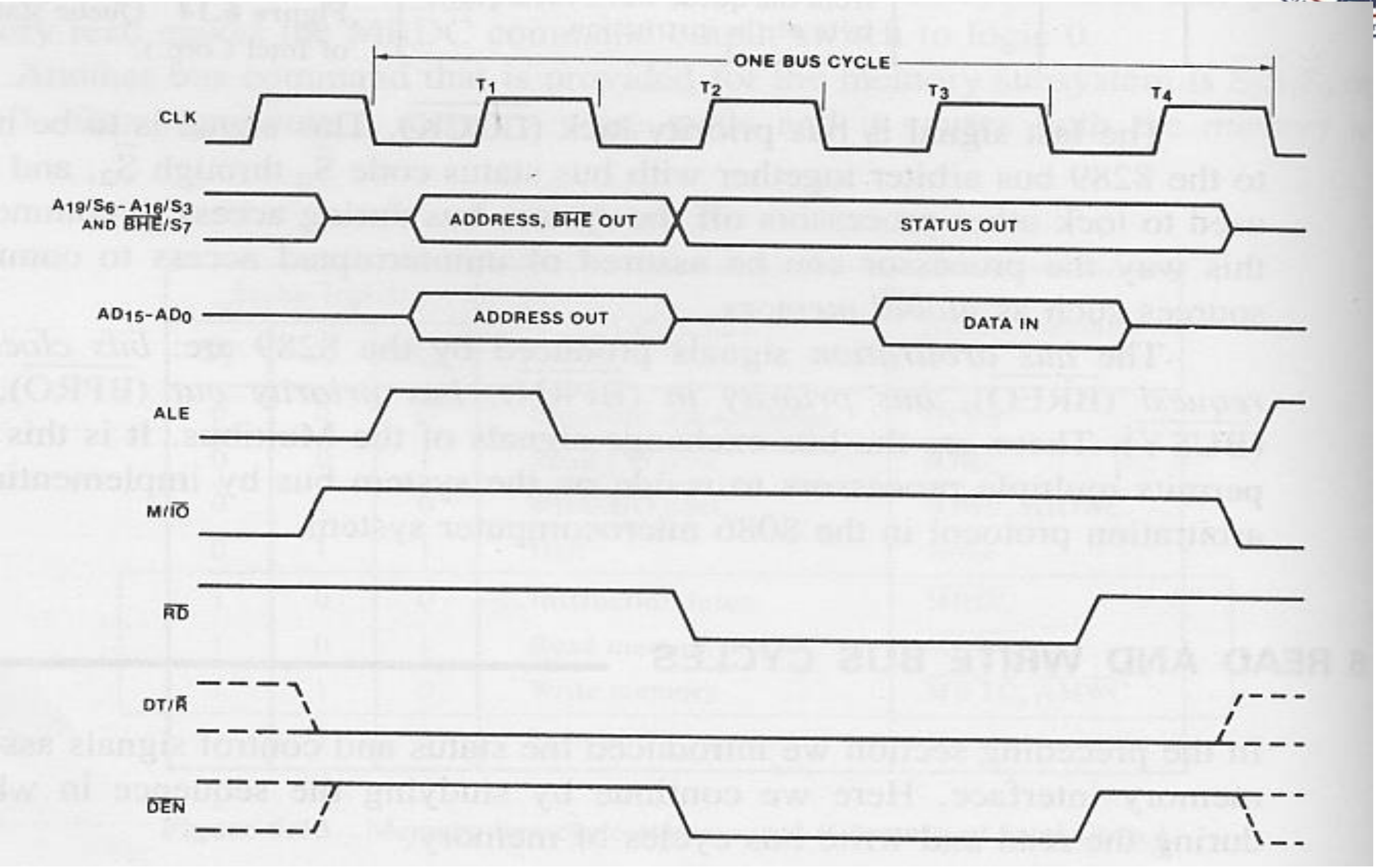
Memory control signals

- To control the memory system in the minimum mode, requires: ALE, /BHE, M/IO, DT/R, /RD, /WR, and /DEN
- ALE – address latch enable, signals external circuitry that a valid address is on the bus (0->1) so the address can be stored in the latch (or buffer)
- M/IO – identify whether it is a memory or IO (Input/Output) operation (high – memory, low – I/O)
- DT/R – transmit or receive (1 – transmit)
- DEN – to enable the data bus

Read Cycle

- Consists of 4 time states
- T1 – memory address is on the address bus, /BHE is also output, ALE is enable
- Address is latch to external device at the trailing edge of ALE
- T2 – M/IO and DT/R are set to 1 and 0 respectively. These signals remain their status during the cycle
- Late in T2 - /RD is switched to 0 and /DEN also set to 0
- T3 and T4 – status bits S3, S4 are output
- Data are read during T3
- /RD and /DEN return to 1 at T4

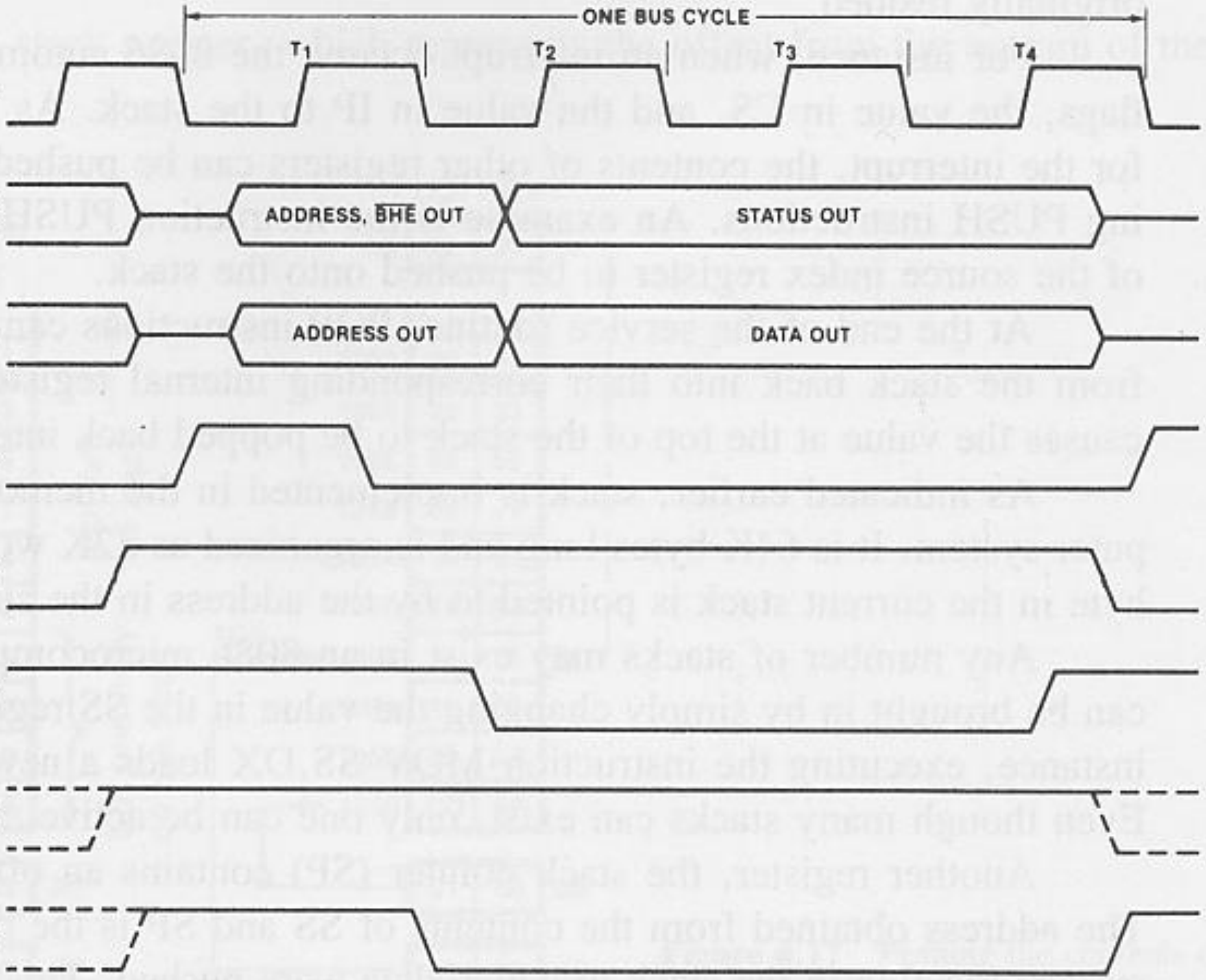
Read Cycle


ONE BUS CYCLE
T1
T2
T3
T4
CLK
A19/S6 - A16/S3 AND BHE/S7
ADDRESS, BHE OUT
STATUS OUT
AD15-AD0
ADDRESS OUT
DATA IN
ALE
M/IO
RD
DT/R
DEN

Write Cycle

- T1 – address and /BHE are output and latched with ALE pulse
- M/IO is set to 1, DT/R is also set to 1
- T2 - /WR set to 0 and data put on data bus
- Data remain in the data bus until /WR returns to 1
- When /WR returns to 1 at T4, data is written into memory

Write Cycle

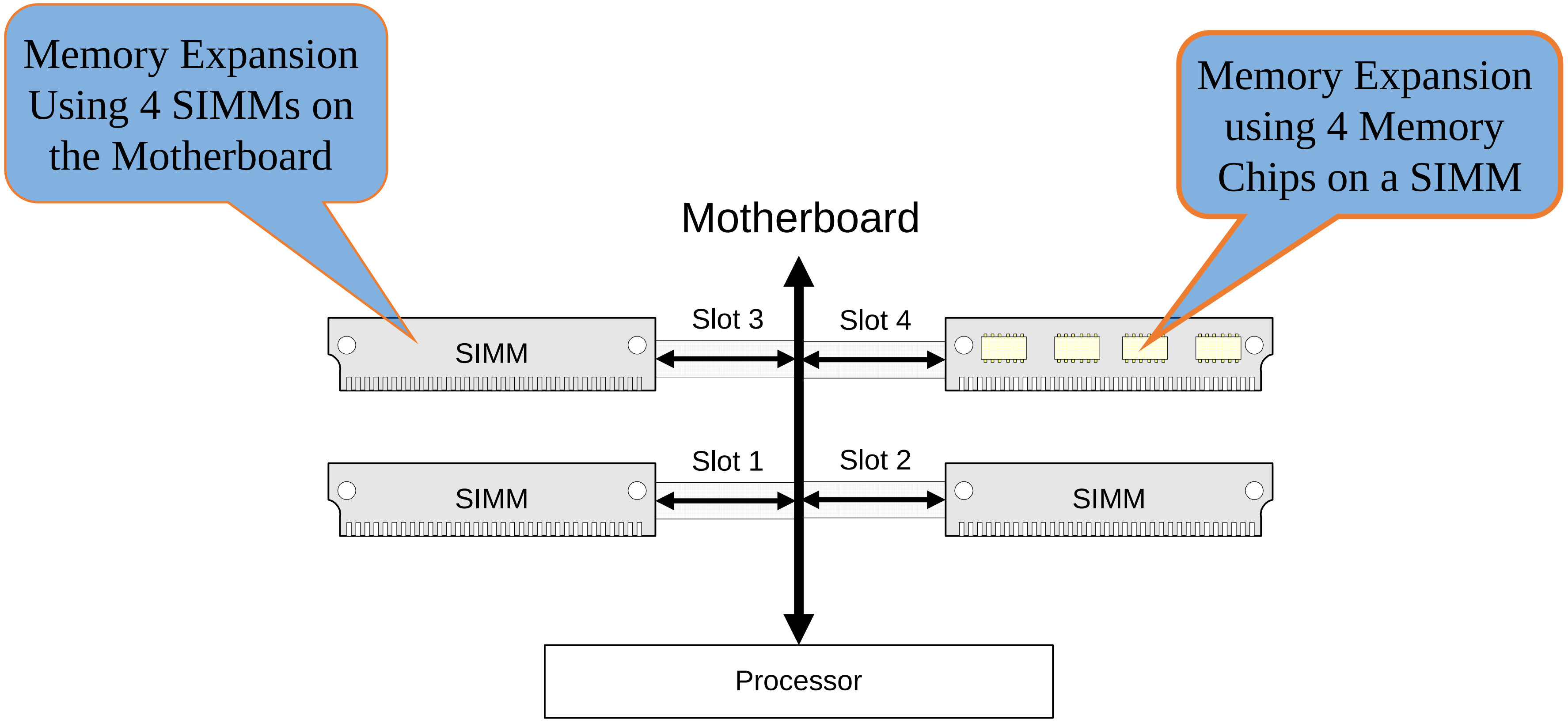


ONE BUS CYCLE
T1
T2
T3
T4
CLK
A19/S6 - A16/S3 AND BHE/S7
ADDRESS, BHE OUT
STATUS OUT
AD15-AD0
ADDRESS OUT
DATA OUT
ALE
M/IO
WR
DT/R
DEN

Double Data Rate (DDR) DRAM

- An SDRAM type of memory where data are transferred on both the rising and the falling clock edge, effectively doubling the transfer rate without increasing the clock frequency
- DDR-200 means a transfer rate of 200 million transfers per second, at a clock rate of 100 MHz
- DDR1 upto 400 MHz
- DDR2 standard allows higher clock frequencies

Memory Expansion on Motherboards


Memory Expansion Using 4 SIMMs on the Motherboard
Memory Expansion using 4 Memory Chips on a SIMM
Motherboard
Slot 3
Slot 4
SIMM
Slot 1
Slot 2
SIMM
SIMM
Processor

RAM Example

Design an 8KX8 RAM module using 2KX8 RAM chips. The module should be connected on an 8-bit processor with a 16-bit address bus and occupy the address range starting from the address A000. Show the circuit and the memory map.

- Number of memory devices needed = 8K/2K = 4
- Decoder needed = 2X4
- Number of address lines on each 2KX8 memory chip = 11

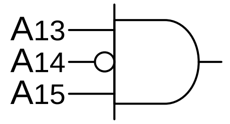
  $2^m = 2K = 2^1 \times 2^{10} = 2^{11} \Rightarrow$ (A0..A10)
- Decoder needed = 2X4

  $\Rightarrow$ 2 address lines are needed for the decoder.
  $\Rightarrow$ (A11..A12)
- Number of address lines needed for the address selection circuit

  = 16 - 11 - 2 = 3 $\Rightarrow$ (A13, A14 A15)

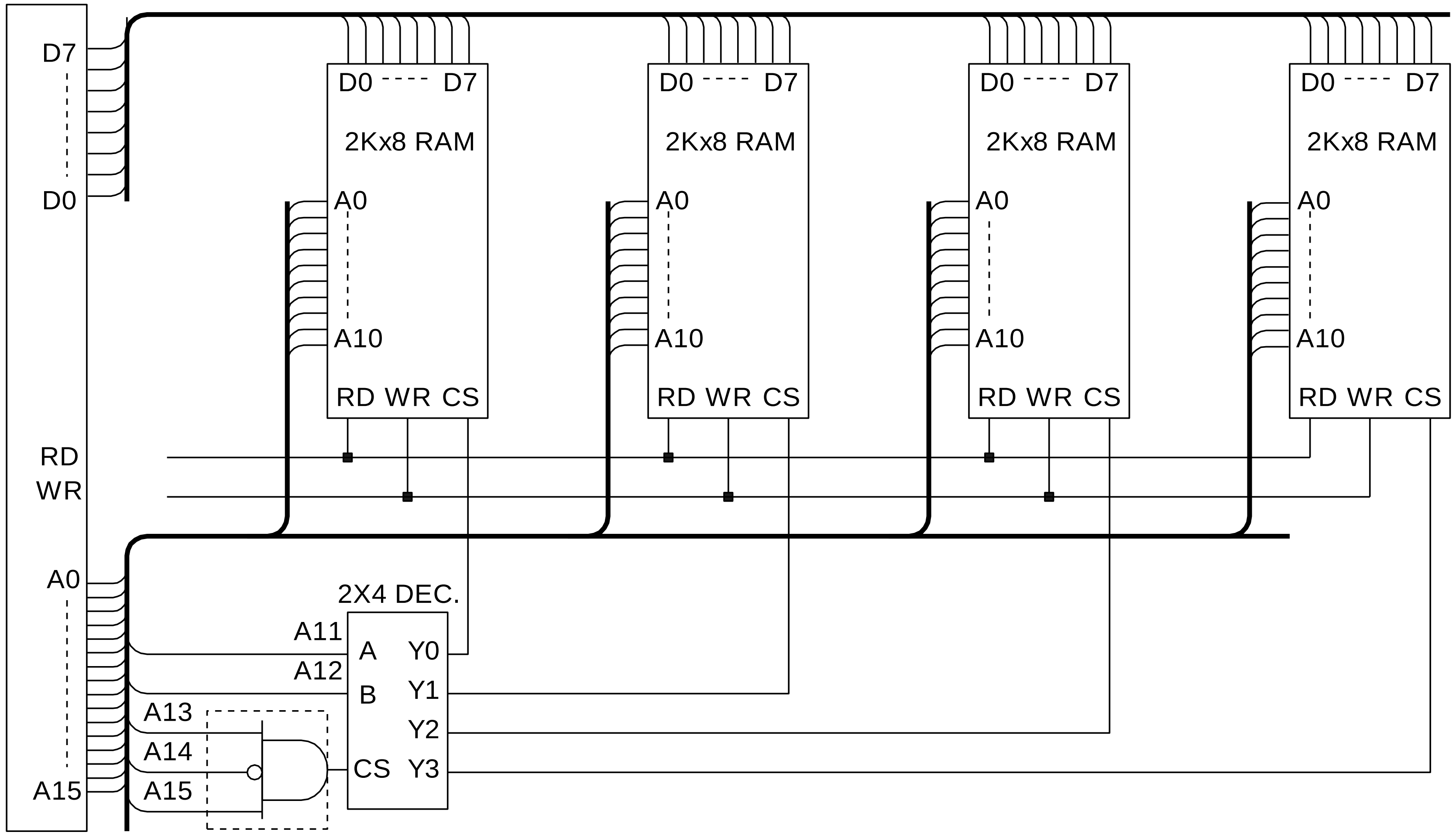
Starting Address = A000 = 1010-0000-0000-0000
==> A15 = 1, A14 = 0 and A13 = 1

Address Selection Circuit



A_{15}	A_{14}	A_{13}	A_{12}	A_{11}	A_{10}	······	A_0	Mem. Map	
0	0	0	0	0	0	······	0	0000	Not Used
1	0	0	1	1	1	······	1	9FFF	
1	0	1	0	0	0	······	0	A000	RAM1
1	0	1	0	0	1	······	1	A7FF	
1	0	1	0	1	0	······	0	A800	RAM2
1	0	1	0	1	1	······	1	AFFF	
1	0	1	1	0	0	······	0	B000	RAM3
1	0	1	1	0	1	······	1	B7FF	
1	0	1	1	1	0	······	0	B800	RAM4
1	0	1	1	1	1	······	1	BFFF	
1	1	0	0	0	0	······	0	C000	Not Used
1	1	1	1	1	1	······	1	FFFF	

Circuit Diagram



D7
D0
RD
WR
A0
A15
D0 ---- D7
2Kx8 RAM
A0
A10
RD WR CS
D0 ---- D7
2Kx8 RAM
A0
A10
RD WR CS
D0 ---- D7
2Kx8 RAM
A0
A10
RD WR CS
D0 ---- D7
2Kx8 RAM
A0
A10
RD WR CS
2X4 DEC.
A11
A12
A13
A14
A15
A
B
CS
Y0
Y1
Y2
Y3