# EC-252: COMPUTER ARCHITECTURE AND MICROPROCESSORS

Vaskar Raychoudhury

Indian Institute of Technology Roorkee

# Memory Technology

- Static RAM (SRAM)
  - 0.5ns – 2.5ns, $2000 – $5000 per GB
- Dynamic RAM (DRAM)
  - 50ns – 70ns, $20 – $75 per GB
- Magnetic disk
  - 5ms – 20ms, $0.20 – $2 per GB
- Ideal memory
  - Access time of SRAM
  - Capacity and cost/GB of disk

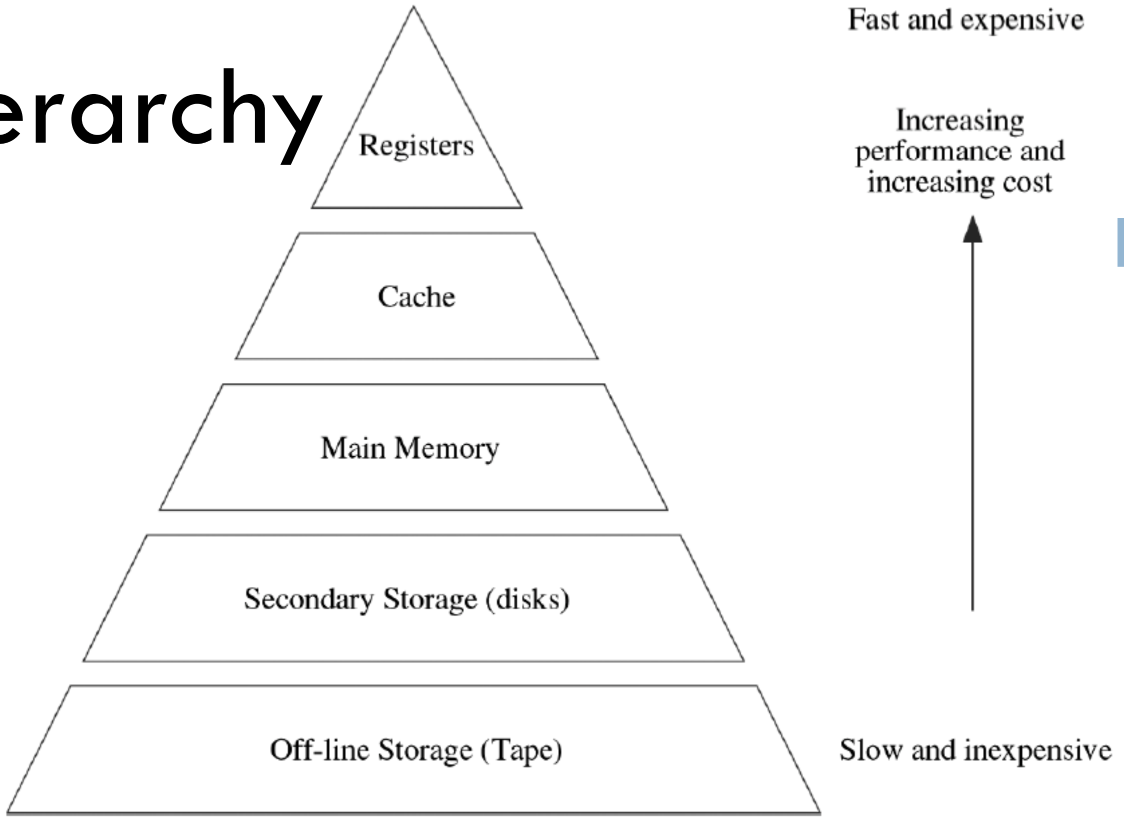
# Principle of Locality

- Programs access a small proportion of their address space at any time
- Temporal locality
  - Items accessed recently are likely to be accessed again soon
  - e.g., instructions in a loop, induction variables
- Spatial locality
  - Items near those accessed recently are likely to be accessed soon
  - E.g., sequential instruction access, array data

## Taking Advantage of Locality

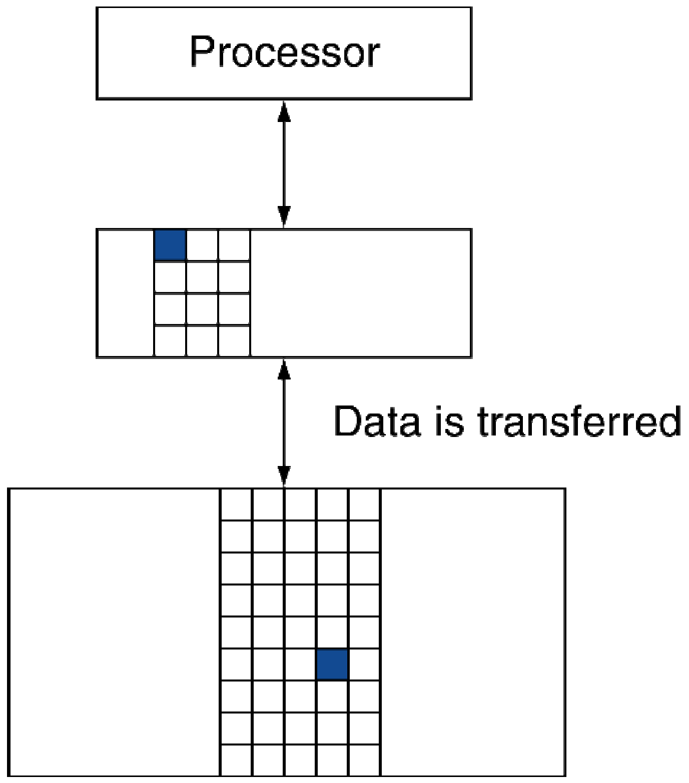
- Memory hierarchy
- Store everything on disk
- Copy recently accessed (and nearby) items from disk to smaller DRAM memory
  - Main memory
- Copy more recently accessed (and nearby) items from DRAM to smaller SRAM memory
  - Cache memory attached to CPU

# Memory Hierarchy



| Memory type | Access time | Cost/MB | Typical amount used | Typical cost |
|---|---|---|---|---|
| Registers | 0.5 ns | High | 2 KB | – |
| Cache | 5–20 ns | $80 | 2 MB | $160 |
| Main memory | 40–80ns | $0.40 | 512 MB | $205 |
| Disk memory | 5 ms | $0.005 | 40 GB | $200 |

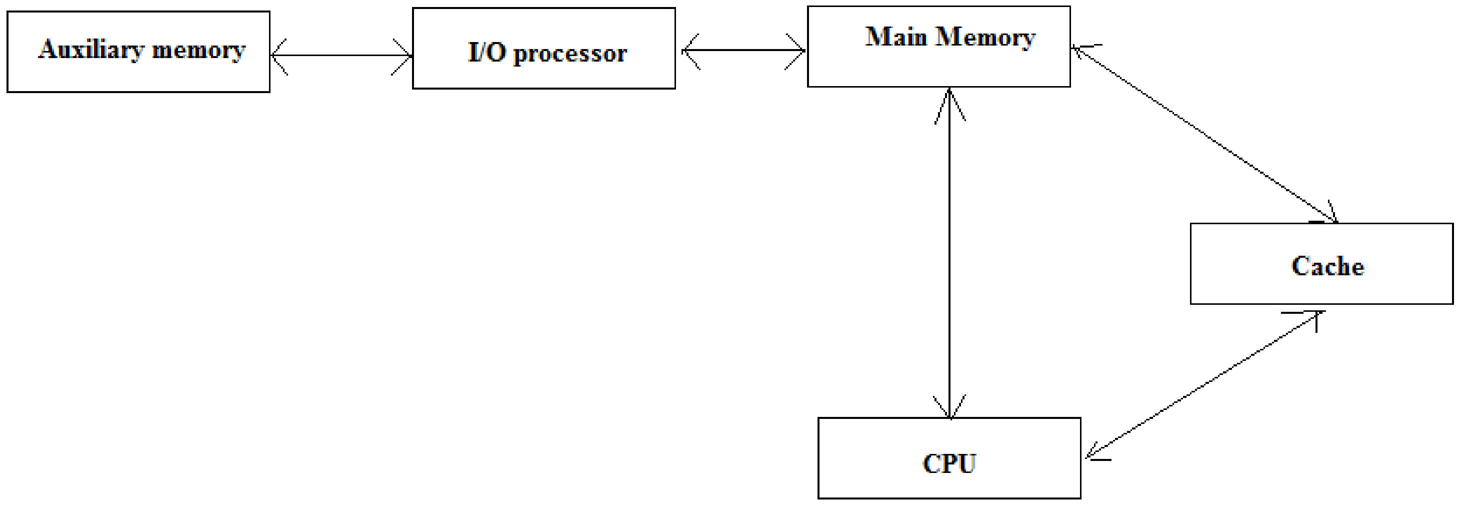
# Memory Hierarchy Levels



- Block (aka line): unit of copying
  - May be multiple words
- If accessed data is present in upper level
  - Hit: access satisfied by upper level
    - Hit ratio: hits/accesses
- If accessed data is absent
  - Miss: block copied from lower level
    - Time taken: miss penalty
    - Miss ratio: misses/accesses
      = 1 – hit ratio
  - Then accessed data supplied from upper level

# Memory Hierarchy

- The main memory occupies a central position by being able to communicate directly with the CPU and with auxiliary memory devices through an I/O processor
- A special very-high-speed memory called **cache** is used to increase the speed of processing by making current programs and data available to the CPU at a rapid rate



## Main Memory

- Most of the main memory in a general purpose computer is made up of RAM integrated circuits chips, but a portion of the memory may be constructed with ROM chips

- RAM– Random Access memory
  - Integated RAM are available in two possible operating modes, *Static and Dynamic*
- ROM– Read Only memory

## Random-Access Memory (RAM)

- Static RAM (SRAM)
  - Each cell stores bit with a six-transistor circuit.
  - Retains value indefinitely, as long as it is kept powered.
  - Relatively insensitive to disturbances such as electrical noise.
  - Faster and more expensive than DRAM.
- Dynamic RAM (DRAM)
  - Each cell stores bit with a capacitor and transistor.
  - Value must be refreshed every 10-100 ms.
  - Sensitive to disturbances.
  - Slower and cheaper than SRAM.

## ROM

- ROM is used for storing programs that are **PERMENTLY** resident in the computer and for tables of constants that do not change in value once the production of the computer is completed

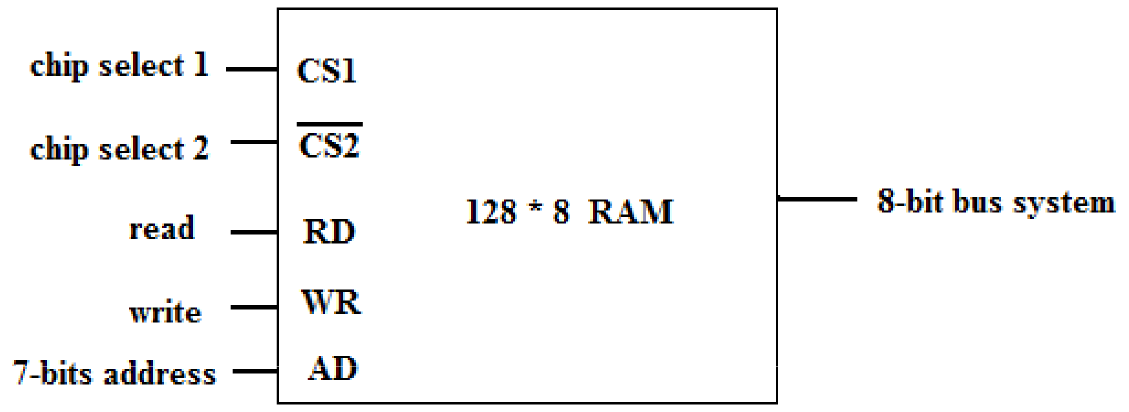
- The ROM portion of main memory is needed for storing an initial program called *bootstrap loader,* witch is to start the computer software operating when power is turned off

## Main Memory

- A RAM chip is better suited for communication with the CPU if it has one or more control inputs that select the chip when needed

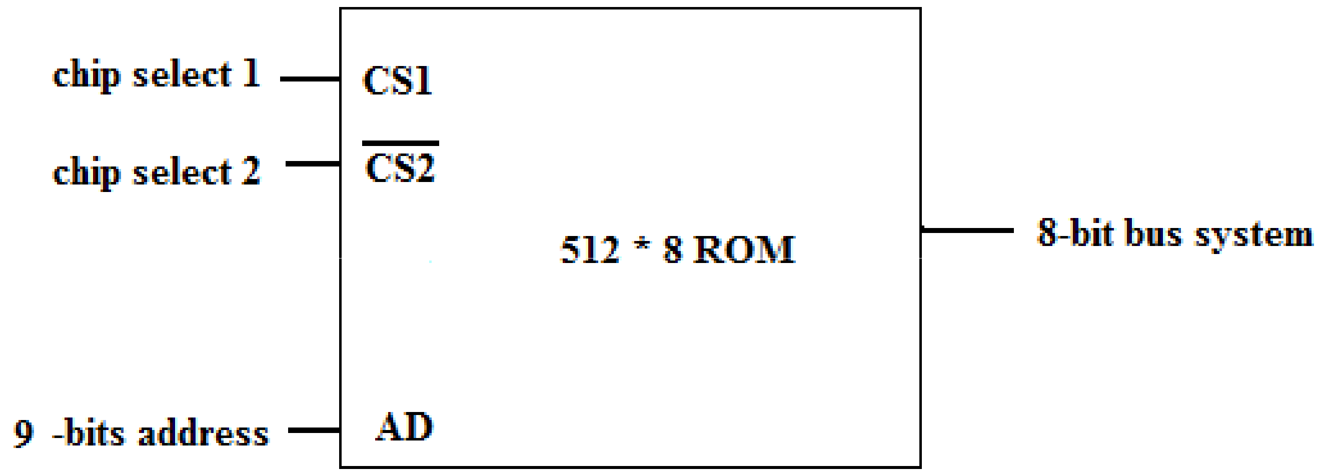
- The Block diagram of a RAM chip is shown next slide, the capacity of the memory is 128 words of 8 bits (one byte) per word

## RAM



| CS1 | $\overline{\text{CS2}}$ | RD | WD | Memory Function | State of data bus |
|---|---|---|---|---|---|
| 0 | 0 | * | * | Inhibit | High-impedance |
| 0 | 1 | * | * | Inhibit | High-impedance |
| 1 | 0 | 0 | 0 | Inhibit | High-impedance |
| 1 | 0 | 0 | 1 | Write | Input data to RAM |
| 1 | 0 | 1 | * | Read | Output data from RAM |
| 1 | 1 | * | * | Inhibit | High-impedance |

## ROM


chip select 1
CS1
chip select 2
$\overline{CS2}$
512 * 8 ROM
8-bit bus system
9 -bits address
AD

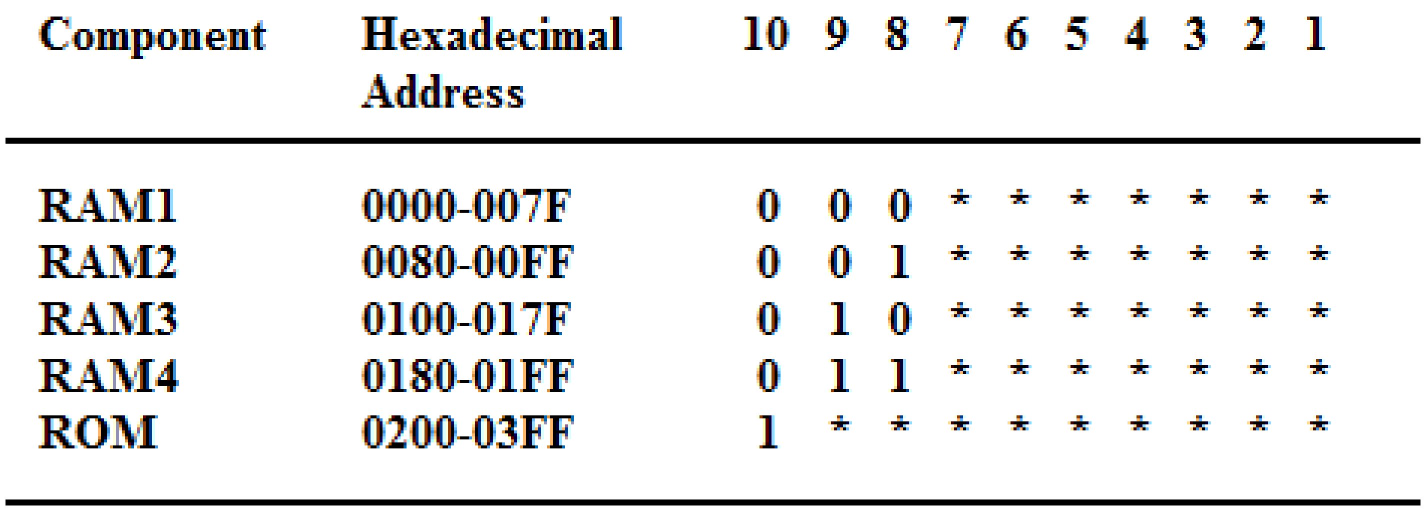
## Memory Address Map

- Memory Address Map is a pictorial representation of assigned address space for each chip in the system

- To demonstrate an example, assume that a computer system needs 512 bytes of RAM and 512 bytes of ROM

- The RAM have 128 byte and need seven address lines, where the ROM have 512 bytes and need 9 address lines

# Memory Address Map



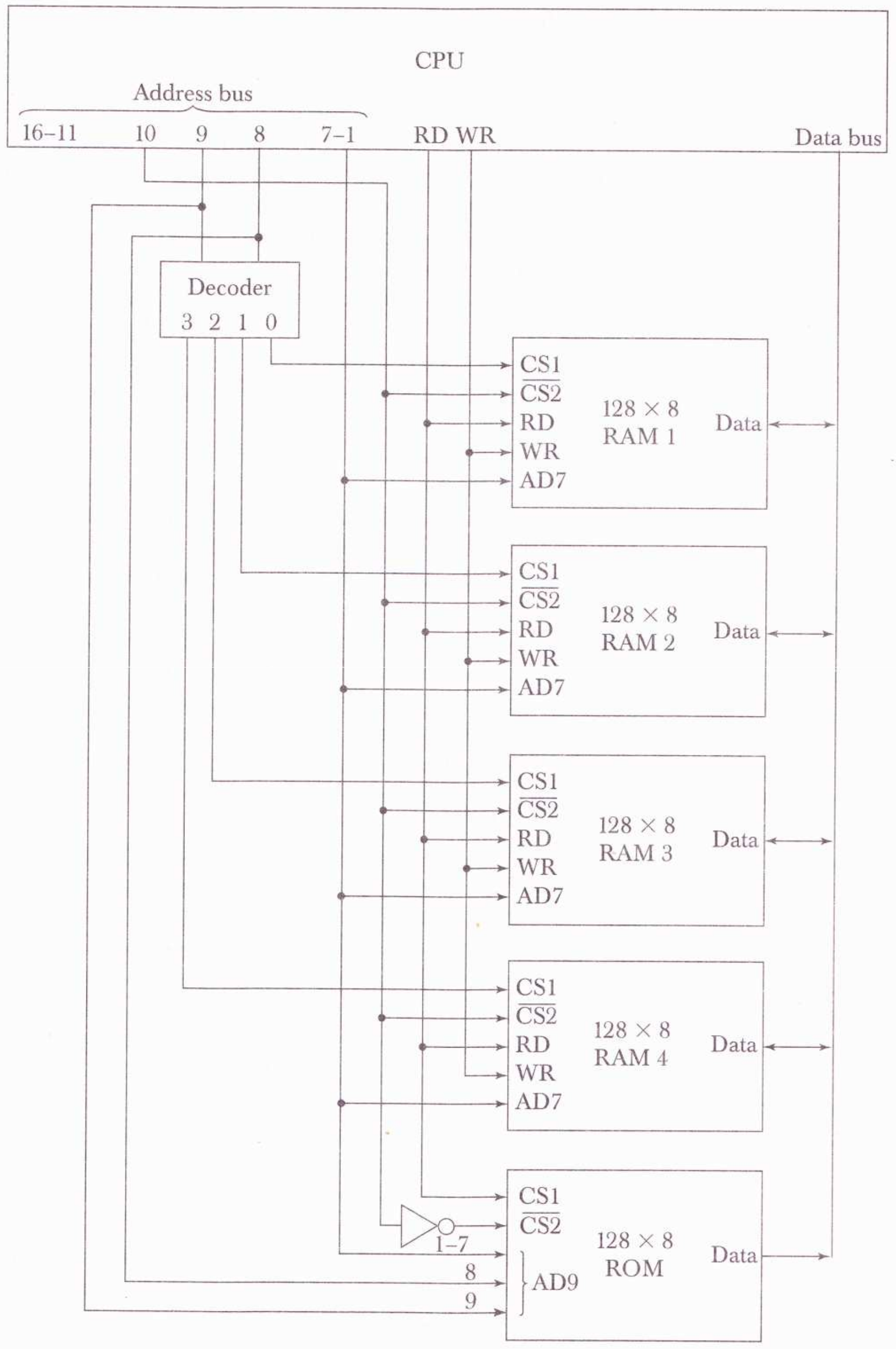
| Component | Hexadecimal Address | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| RAM1 | 0000-007F | 0 | 0 | 0 | * | * | * | * | * | * | * |
| RAM2 | 0080-00FF | 0 | 0 | 1 | * | * | * | * | * | * | * |
| RAM3 | 0100-017F | 0 | 1 | 0 | * | * | * | * | * | * | * |
| RAM4 | 0180-01FF | 0 | 1 | 1 | * | * | * | * | * | * | * |
| ROM | 0200-03FF | 1 | * | * | * | * | * | * | * | * | * |

## Memory Address Map

- The hexadecimal address assigns a range of hexadecimal equivalent address for each chip

- Line 8 and 9 represent four distinct binary combination to specify which RAM we chose

- When line 10 is 0, CPU selects a RAM. And when it’s 1, it selects the ROM


CPU
Address bus
16–11
10
9
8
7–1
RD WR
Data bus
Decoder
3 2 1 0
CS1
$\overline{CS2}$
RD
WR
AD7
128 × 8
RAM 1
Data
CS1
$\overline{CS2}$
RD
WR
AD7
128 × 8
RAM 2
Data
CS1
$\overline{CS2}$
RD
WR
AD7
128 × 8
RAM 3
Data
CS1
$\overline{CS2}$
RD
WR
AD7
128 × 8
RAM 4
Data
CS1
$\overline{CS2}$
1–7
8
9
AD9
128 × 8
ROM
Data

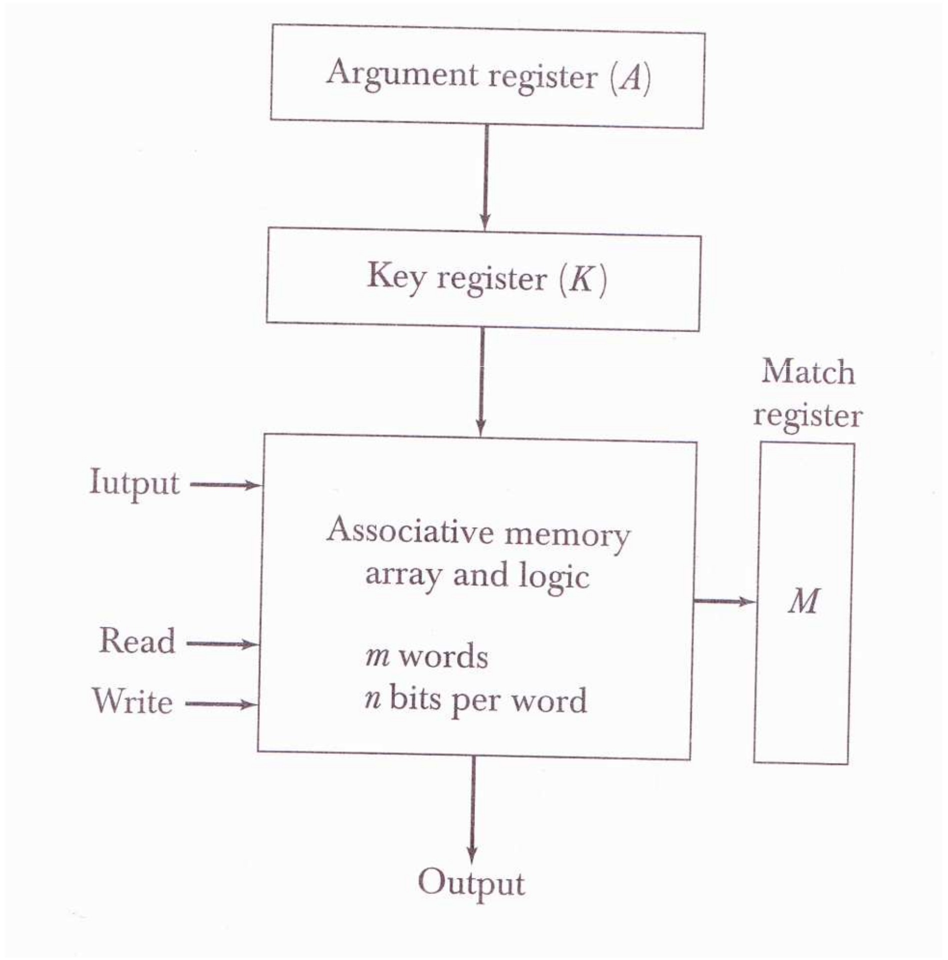
## Associative Memory

- Associative or Content Addressable Memory (CAM)
  - Memory unit accessed by the content of the data rather than by an address
- Writing data to associative memory
  - No address is given while writing a data
  - The memory itself finds an empty unused location to store the word
- Reading data from associative memory
  - The content of the word or part of it (how?) is specified
  - The memory locates all words matching the specified content and marks them for reading

# Pros & Cons of Associative Memory

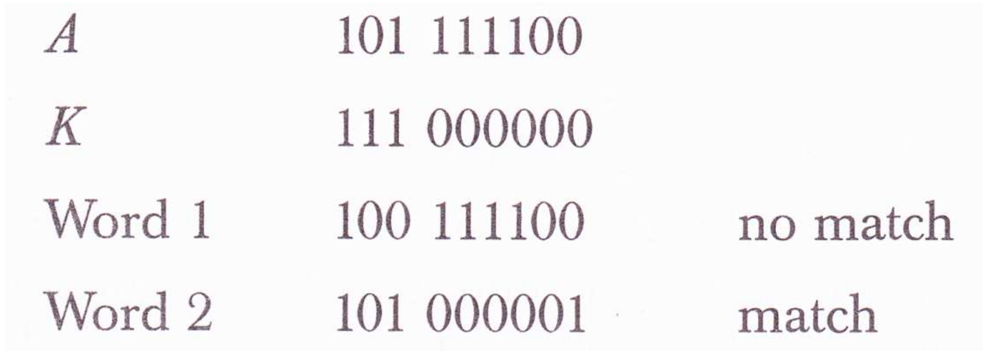
- Uniquely suited for parallel searches by data association
- Searches for an entire word or a part thereof (specific field within the word) is possible
- More expensive than a RAM as
  - each cell must have storage capacity, and
  - logic circuits to match the cell content with an external argument
- Used in applications where the search time is very critical and must be very short

## Block Diagram of Associative Memory



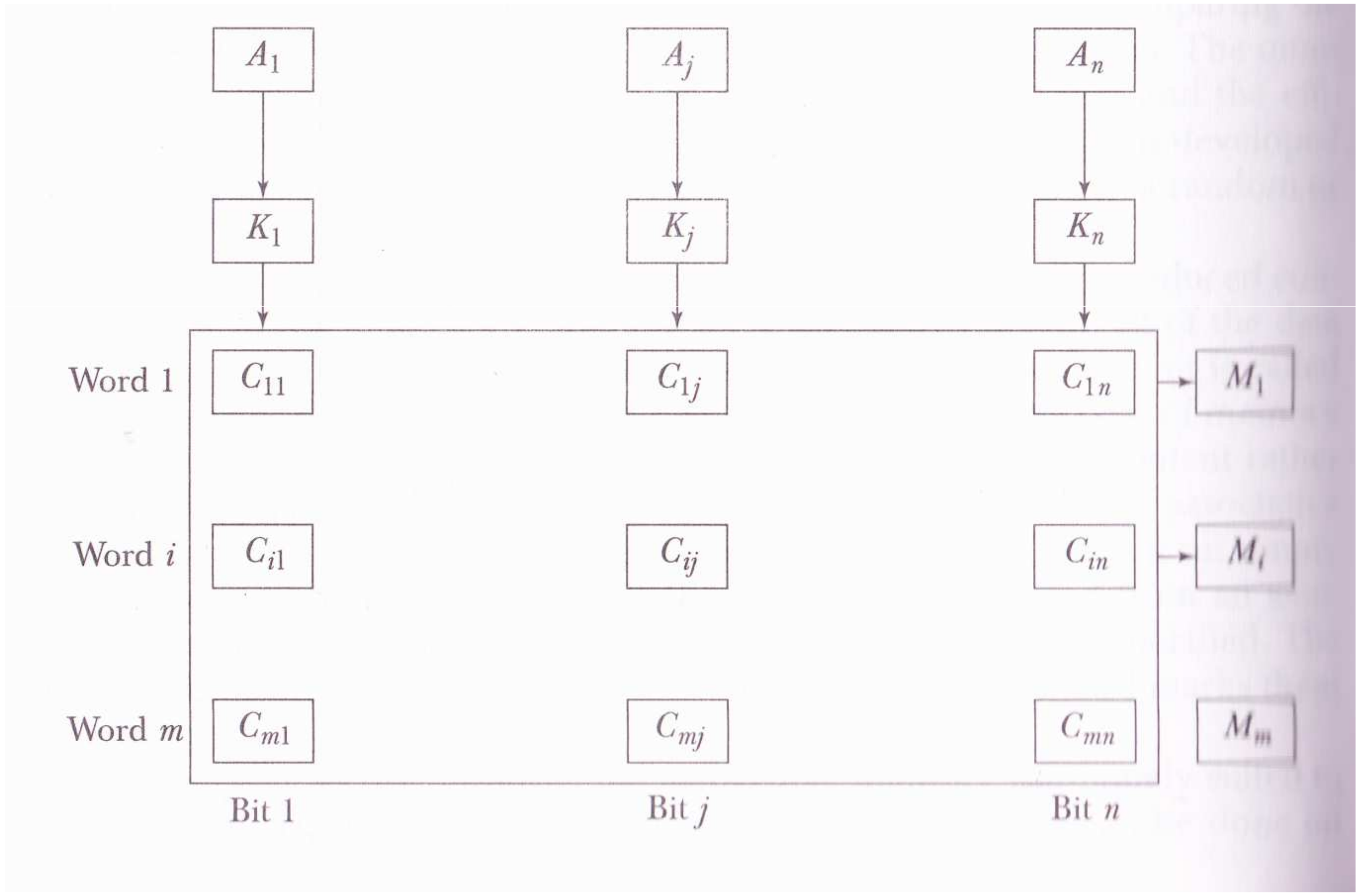
## Total or Partial Content Search

- The key register provides a mask for choosing a particular field or key in the argument word
  - The argument bits for which the key bits are 1, are compared with each memory word



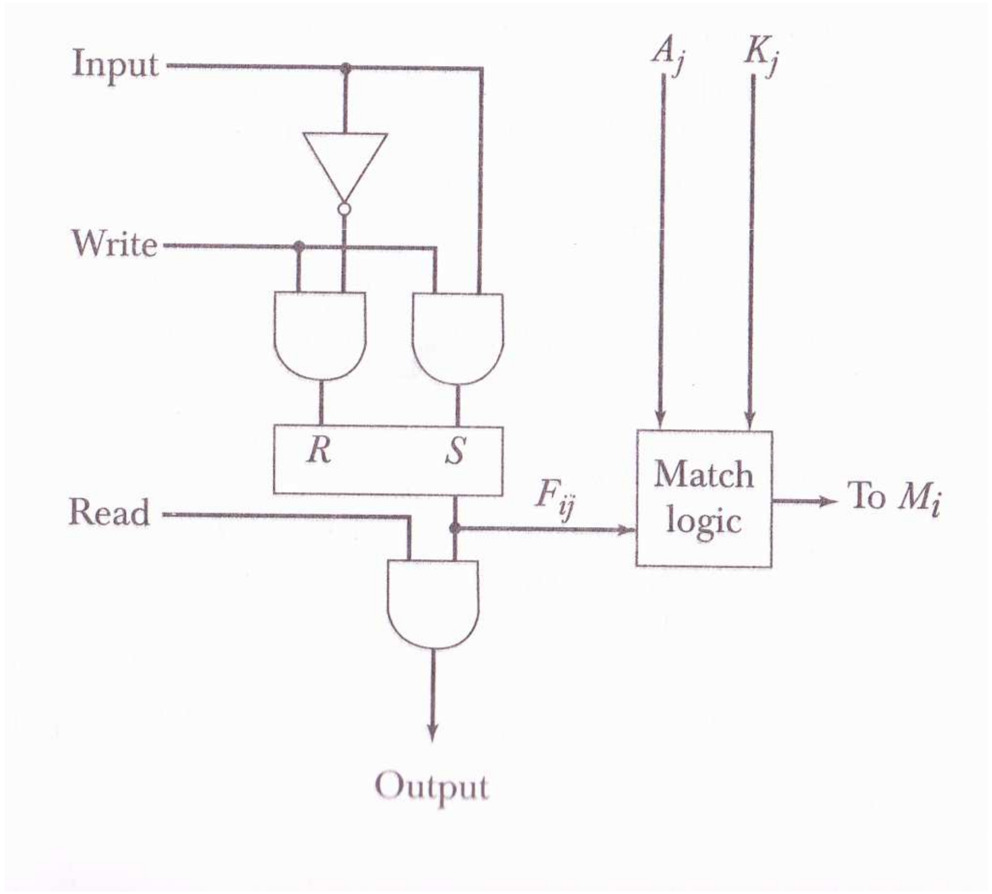
| | | |
|---|---|---|
| *A* | 101 111100 | |
| *K* | 111 000000 | |
| Word 1 | 100 111100 | no match |
| Word 2 | 101 000001 | match |

# *m* x *n* Associative Memory / CAM



## Single Cell of CAM

- The flip-flop can store 1-bit element $F_{ij}$



## Match Logic

- Without considering the key bits
  - Word $i$ is equal to the argument in A if $A_j = F_{ij}$ for $j$=1, 2,…, $n$
  - The equality condition can be expressed logically as

$$x_j = A_j F_{ij} + A_j' F_{ij}'$$

  - where $x_j$ = 1, if the pair of bits in position $j$ are equal; otherwise, $x_j = 0$
  - So, the condition to set the corresponding match bit $M_i$ to 1 is:

$$M_i = x_1\, x_2\, x_3 \ldots\ldots\, x_n$$

# Match Logic (cont'd)

- Considering the key bits
  - If $K_j = 0$, don't compare the corresponding $A_j$ and $F_{ij}$
  - Logically expressing the requirement:

$$x_j + K_j' = \begin{cases} x_j & \text{if } K_j = 1 \\ 1 & \text{if } K_j = 0 \end{cases}$$

  - A term ($x_j + K_j$') will be 1 if $A_j$ and $F_{ij}$ are not compared
  - So, the match logic is expressed as:

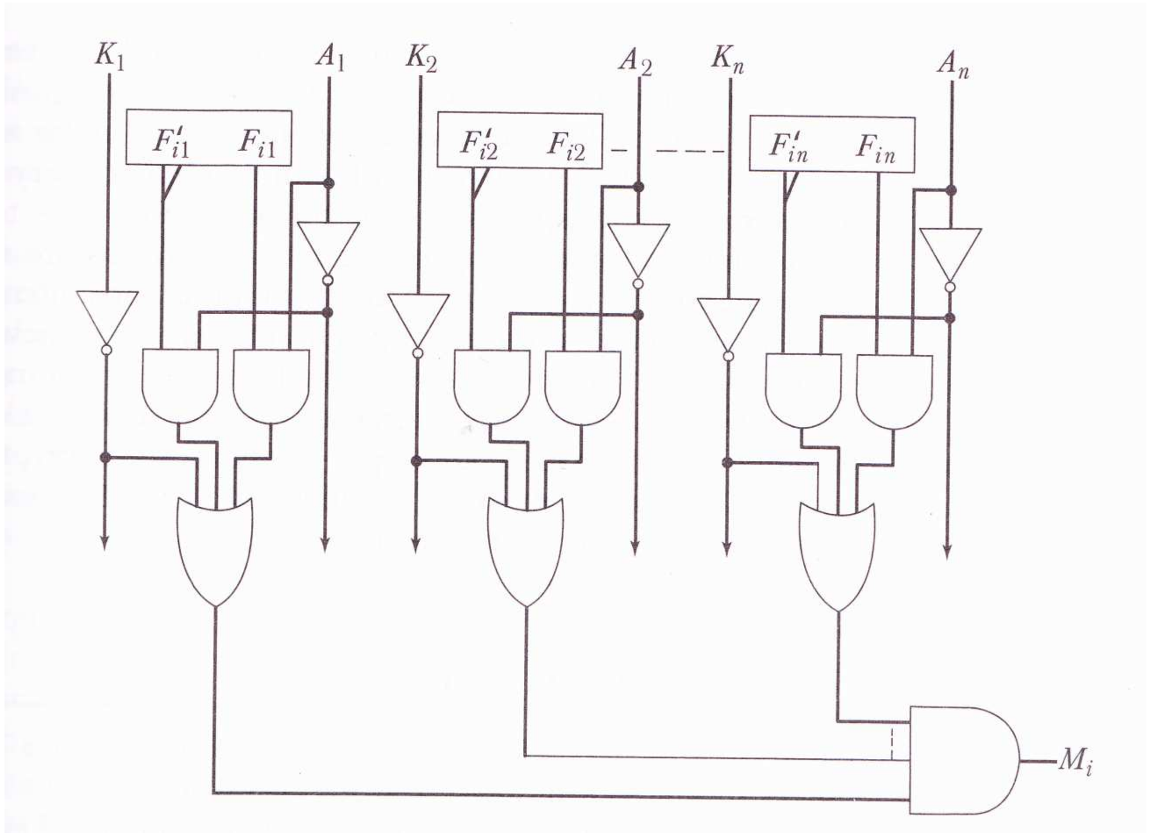
$$M_i = (x_1 + K_1')(x_2 + K_2')(x_3 + K_3') \ldots (x_n + K_n')$$

  - Replacing $x_j$ with the original definition

$$M_i = \prod_{j=1}^{n} (A_j F_{ij} + A_j' F_{ij}' + K_j')$$

# Match Logic for One Word in CAM

- Each cell has 2 AND gates and 1 OR gate
- The inverters for $A_j$ and $K_j$ are needed once for each column
- Avoid all zero (0) key condition



# Read Operation in CAM

- If more than one word in memory matches the unmasked argument field
  - Corresponding match bits of all the matched words are 1
  - Each matched word is then read in sequence by applying a special read signal
  - Usually, associative memory does not store two identical items under a given key
    - So, a single word match occurs and no special read signal is needed
    - We can exclude all-zero-words (think as empty position) and consider such output as a no-match

# Write Operation in CAM

- If the entire memory is loaded at one go
  - Write by addressing each location in sequence
    - Same as writing in RAM
    - No need $m$ address lines, but $d$ lines, where $m = 2^d$
- Deleting stale words and inserting new ones
  - Need special *tag register* to distinguish between active and inactive words
    - For each word there can be a tag bit which will indicate an active and an inactive word by setting to 1 and 0, respectively
    - Delete stale words by setting the tag bit to 0
    - Store new word at the first position for tag bit = 0 and set it to 1

## Uses of CAM

- CAM is often used in **Network Search Engines / Elements** (NSE), such as, network switch or router
  - A network switch carries out address lookup to forward incoming packets to a destination MAC addresses via some port
  - The MAC address table is implemented with a CAM so the destination port can be found very quickly, reducing the switch's latency

# Advantages of CAM

- In case of RAM, the user supplies a memory address and the RAM returns the data word stored at that address
- However, a CAM is designed, such that, the user supplies a data word and the CAM searches its entire memory (in a single operation) to see if that data word is stored anywhere in it
  - So, CAM is much faster than RAM in virtually all search applications
  - Every comparison circuit is active on every clock cycle