Memory Organization

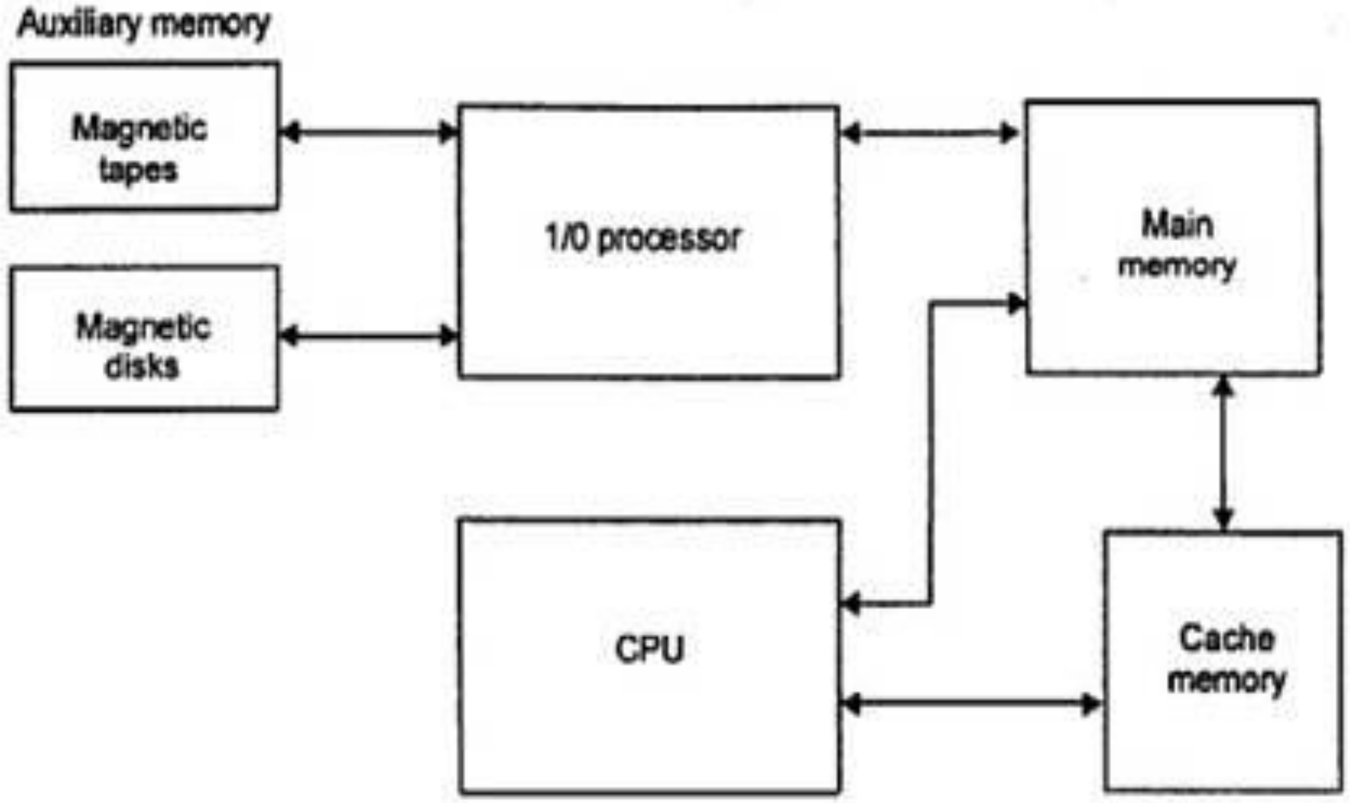
Memory Hierarchy

Memory is used for storing programs and data that are required to perform a specific task.

For CPU to operate at its maximum speed, it required an uninterrupted and high speed access to these **memories** that contain programs and data. Some of the criteria need to be taken into consideration while deciding which **memory** is to be used:

- Cost
- Speed
- Memory access time
- Data transfer rate
- Reliability

How Memories attached to CPU



A computer system contains various types of memories like auxiliary memory, cache memory, and main memory.

- **Auxiliary Memory**

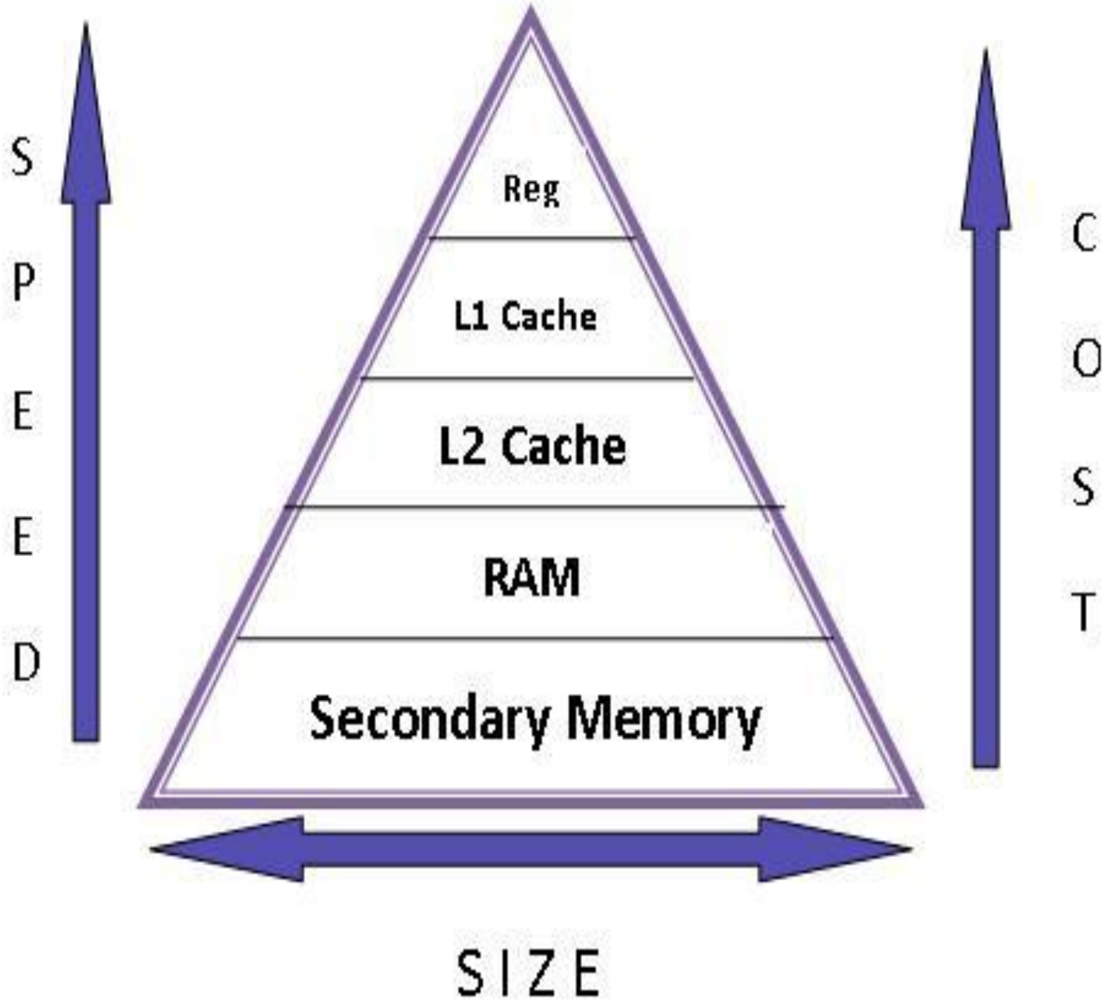
The auxiliary memory is at the bottom and is not connected with the CPU directly. However, being slow, it is present in large volume in the system due to its low pricing. This memory is basically used for storing the programs that are not needed in the main memory. This helps in freeing the main memory which can be utilized by other programs that needs main memory. The main function of this memory is to provide parallel searching that can be used for performing a search on an entire word.

- **Main Memory**

The main memory is at the second level of the hierarchy. Due to its direct connection with the CPU, it is also known as central memory. The main memory holds the data and the programs that are needed by the CPU. The main memory mainly consists of RAM, which is available in static and dynamic mode.

- **Cache Memory**

Cache memory is at the top level of the memory hierarchy. This is a high speed memory used to increase the speed of processing by making current programs and data available to the CPU at a rapid rate. Cache memory is usually placed between the CPU and the main memory.


SPEED
Reg
L1 Cache
L2 Cache
RAM
Secondary Memory
COST
SIZE

Main Memory

- Central storage unit in a computer system
- Large memory
- Made up of Integrated chips
- Types:

  RAM (Random access memory)

  ROM (Read only memory)

RAM (Random Access Memory)

Random access memory (RAM) is the best known form of computer memory. RAM is considered "random access" because you can access any memory cell directly if you know the row and column that intersect at that cell.

Types of RAM:-

- Static RAM (SRAM)
- Dynamic RAM (DRAM)

- **Static RAM (SRAM)**
  - a bit of data is stored using the state of a flip-flop.
  - Retains value indefinitely, as long as it is kept powered.
  - Mostly uses to create cache memory of CPU.
  - Faster and more expensive than DRAM.

- **Dynamic RAM (DRAM)**
  - Each cell stores bit with a capacitor and transistor.
  - Large storage capacity
  - Needs to be refreshed frequently.
  - Used to create main memory.
  - Slower and cheaper than SRAM.

ROM

ROM is used for storing programs that are **Permanently** 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 on.

There are five basic ROM types:

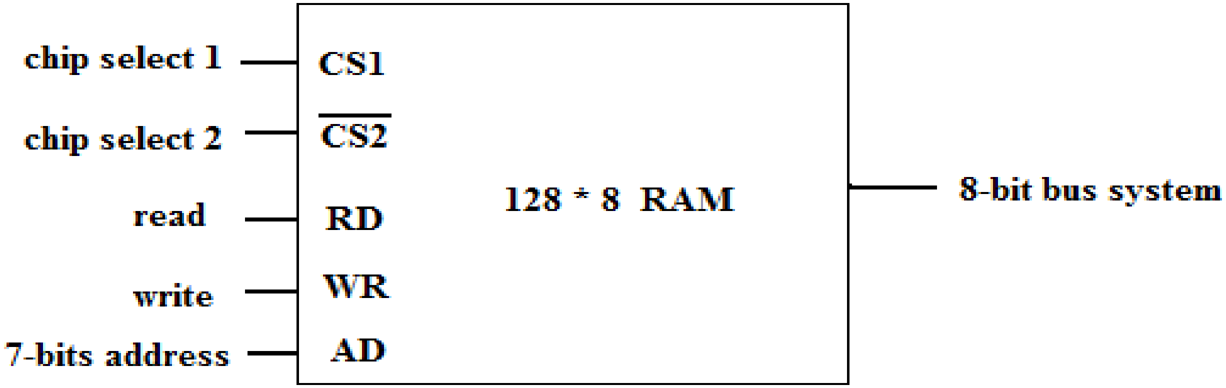
- ROM - Read Only Memory
- PROM - Programmable Read Only Memory
- EPROM - Erasable Programmable Read Only Memory
- EEPROM - Electrically Erasable Programmable Read Only Memory
- Flash EEPROM memory

RAM and ROM Chips

- 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



chip select 1 — CS1

chip select 2 — $\overline{\text{CS2}}$

read — RD

write — WR

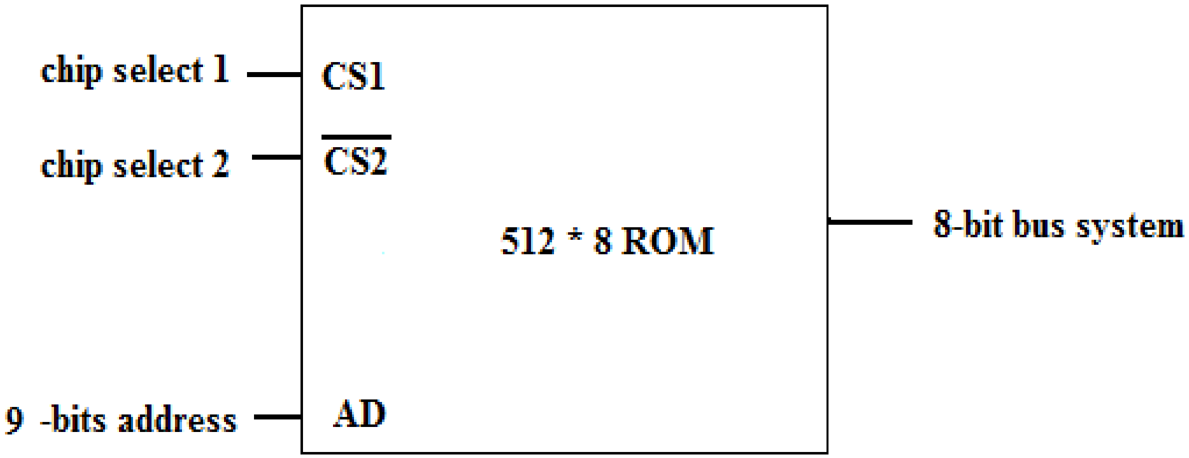
7-bits address — AD

128 * 8 RAM

8-bit bus system

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
$\overline{\text{CS2}}$
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

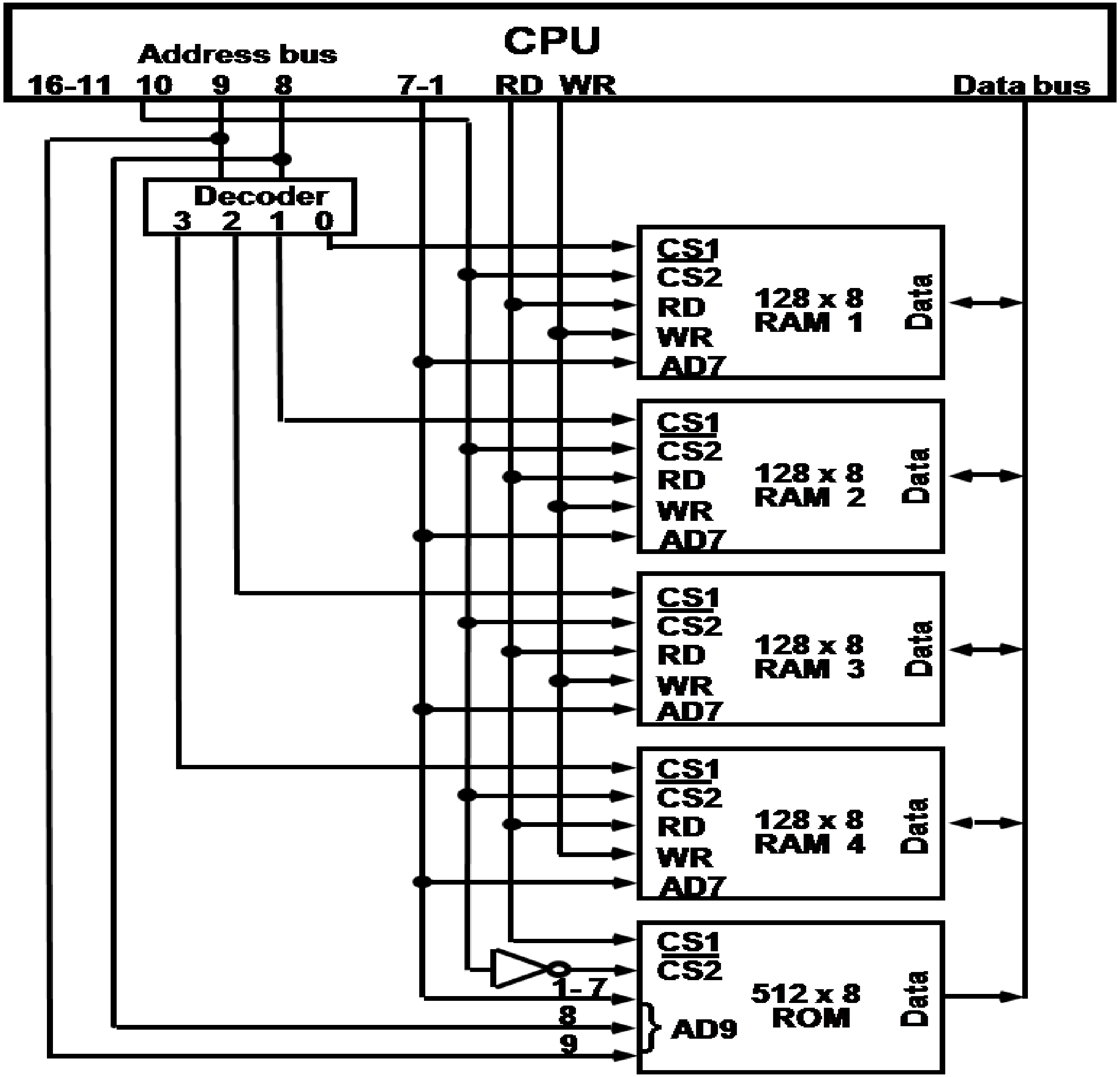
- The RAM have 128 byte and need seven address lines, where the ROM have 512 bytes and need 9 address lines

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	*	*	*	*	*	*	*	*	*

- 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



Memory connection to the CPU

Cache memory

- If the active portions of the program and data are placed in a fast small memory, the average memory access time can be reduced
- Thus reducing the total execution time of the program
- Such a fast small memory is referred to as cache memory
- The cache is the fastest component in the memory hierarchy and approaches the speed of CPU component

- When CPU needs to access memory, the cache is examined
- If the word is found in the cache, it is read from the fast memory
- If the word addressed by the CPU is not found in the cache, the main memory is accessed to read the word

- When the CPU refers to memory and finds the word in cache, it is said to produce a **hit**

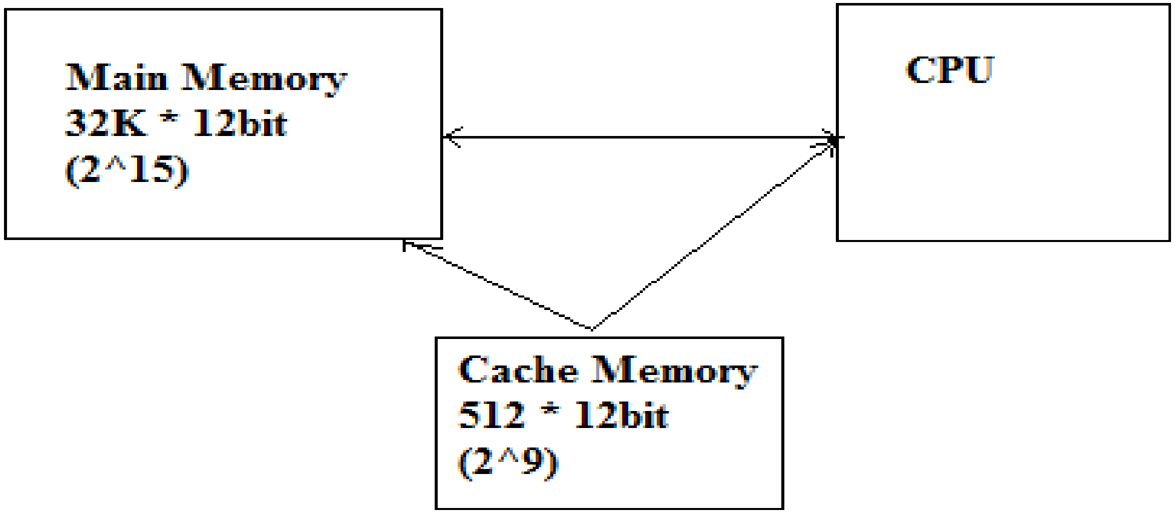
- Otherwise, it is a **miss**

- The performance of cache memory is frequently measured in terms of a quantity called **hit ratio**

  Hit ratio = hit / (hit+miss)

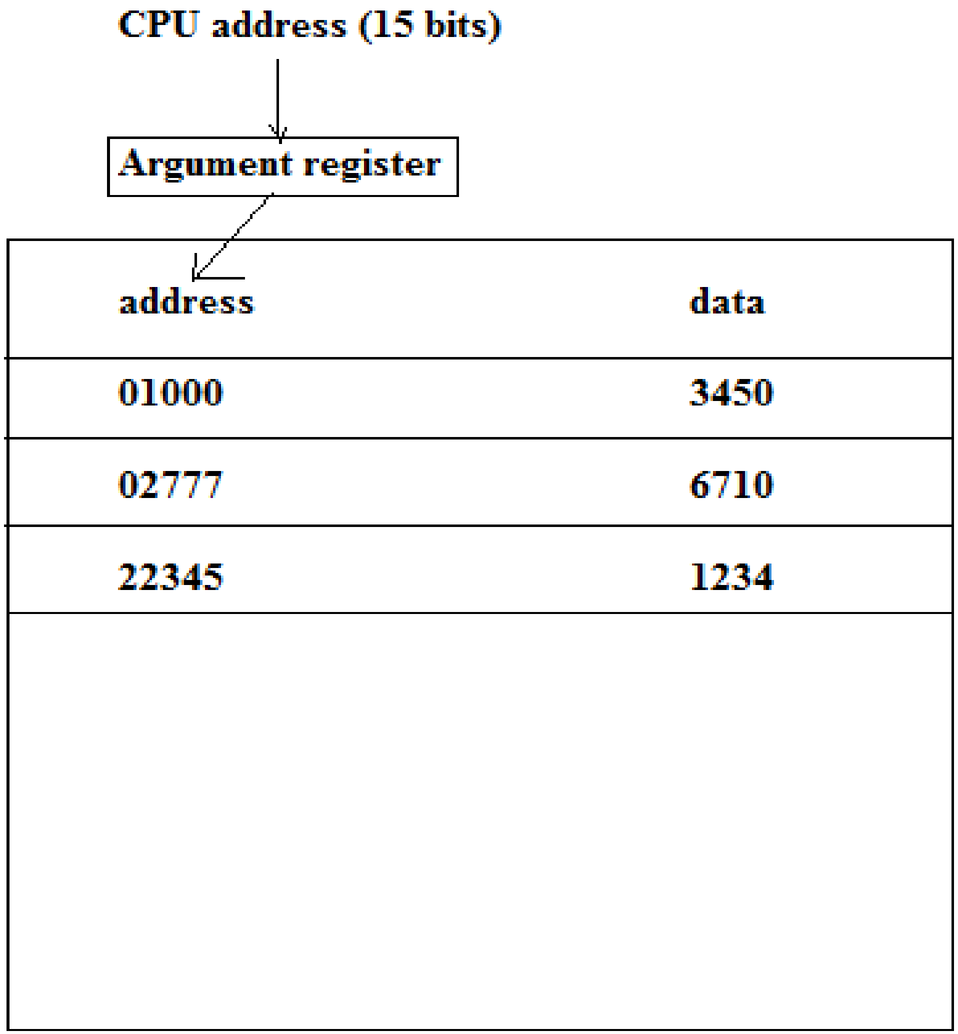
- The basic characteristic of cache memory is its fast access time
- Therefore, very little or no time must be wasted when searching the words in the cache
- The transformation of data from main memory to cache memory is referred to as a **mapping** process, there are three types of mapping:
  - **Associative mapping**
  - **Direct mapping**
  - **Set-associative mapping**

- To help understand the mapping procedure, we have the following example:



Associative mapping

- The fastest and most flexible cache organization uses an associative memory
- The associative memory stores both the address and data of the memory word
- This permits any location in cache to store ant word from main memory
- The address value of 15 bits is shown as a five-digit **octal** number and its corresponding 12-bit word is shown as a four-digit octal number



CPU address (15 bits)

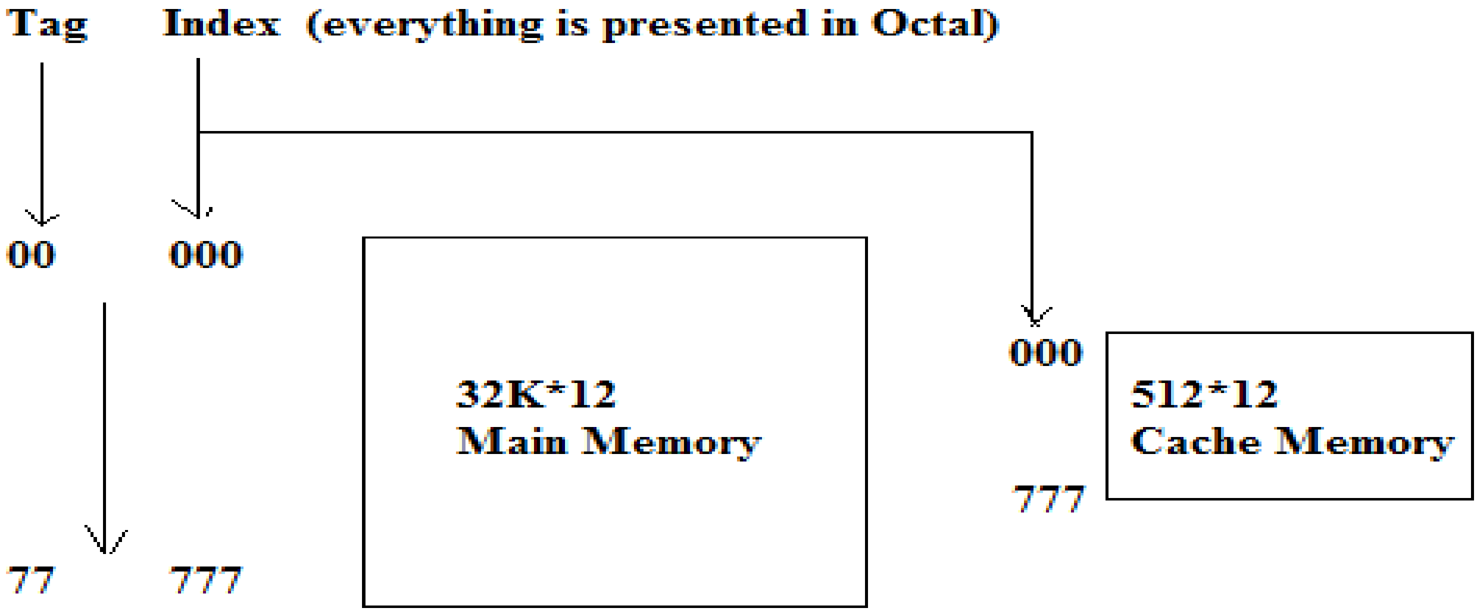
Argument register

address	data
01000	3450
02777	6710
22345	1234

- A CPU address of 15 bits is places in the argument register and the associative memory us searched for a matching address
- If the address is found, the corresponding 12-bits data is read and sent to the CPU
- If not, the main memory is accessed for the word
- If the cache is full, an address-data pair must be displaced to make room for a pair that is needed and not presently in the cache

Direct Mapping

- Associative memory is expensive compared to RAM
- In general case, there are 2^k words in cache memory and 2^n words in main memory (in our case, k=9, n=15)
- The n bit memory address is divided into two fields: k-bits for the index and n-k bits for the tag field



Addressing relationships between main and cache memories

Memory Address	Memory Data
00000	1220
00777	2340
01000	3450
01111	2222
01777	4560
02000	5670
02777	6710

Index Address	Tag	Data
000	00	1220
111	01	2222
777	02	6710

Set-Associative Mapping

- The disadvantage of direct mapping is that two words with the same index in their address but with different tag values cannot reside in cache memory at the same time

- Set-Associative Mapping is an improvement over the direct-mapping in that each word of cache can store two or more word of memory under the same index address

Memory Address	Memory Data
00000	1220
00777	2340
01000	3450
01111	2222
01777	4560
02000	5670
02777	6710

Index Address	Tag	Data	Tag	Data
000	01	3450	02	5670
111	01	2222		
777	02	6710	00	2340

- Each index address refers to two data words and their associated tags
- Each tag requires six bits and each data word has 12 bits, so the word length is 2*(6+12) = 36 bits