Memory System

4.1 Microcomputer Memory

Memory is an essential component of the microcomputer system.It stores binary instructions and datum for the microcomputer.The memory is the place where the computer holds current programs and data that are in use.

The memory unit that communicates directly with the CPU is called main memory.

Devices that provide backup storage are called auxiliary memory or secondary memory.

4.2 Characteristics of memory systems

We can classify memory systems according to their key characteristics .Memory system can be characterised as follows

1.Location
2.Capacity
3.Unit of transfer
4.Access method
5.Performance
6.Physical type
7 Physical characteristics and
8.Organisation.

1.Location

The term location refers to whether memory is internal and external to the computer.

Internal memory is often main memory. The other forms of internal memory are registers (present in processor) and Cache .

External memory consists of peripheral storage devices, such as disk and tape, that are accessible to the processor via I/O controllers.

2.Capacity

A memory is characterized by its capacity. For internal memory, this is typically expressed in terms of bytes (1byte = 8bits) or words. Common word lengths are 8, 16, and 32 bits. External memory capacity is typically expressed in terms of bytes

3.Unit of Transfer

Internal: For internal memory, the unit of transfer is equal to the number of data lines into and out of the memory module.(eg 64 ,128 or 256 bits).Here,unit refers to Word. The size of a word is typically equal to the number of bits used to represent an integer and to the instruction length.The Intel x86 architecture has a a word size of 32 bits.

Relationship between length of bits and number of addressable unit is $2^{noofdatalines}$.•

For a 32 bit datalines,addressable unit is 2^{32}=4GB

Unit of transfer for a main memory is the number of bits read out of or written into memory at a time.

External: The unit of transfer need not equal a word or an addressable unit.For external memory, they are transferred in block which is larger than a word.

4.Access Method

Sequential access:Memory is organized into units of data, called records. Access must be made in a specific linear sequence. Stored addressing information is used to separate records and assist in the retrieval process. A shared read–write mechanism is used, and this must be moved from its current location to the desired location, passing and rejecting each intermediate record. Thus, the time to access an arbitrary record is highly variable. e.g. tape

Direct Access: Individual blocks of records have unique address based on location. Access is accomplished by jumping (direct access) to general vicinity plus a sequential search to reach the final location.

— e.g. disk

Random access: Each addressable location in memory has a unique, physically wired-in addressing mechanism. The time to access a given location is independent of the sequence of prior accesses and is constant. Thus, any location can be selected at random and directly addressed and accessed. Main memory and some cache systems are random access.

— e.g. RAM

Associative access: This is a random access type of memory that enables one to make a comparison of desired bit locations within a word for a specified match, and to do this for all words simultaneously. Thus, a word is retrieved based on a portion of its contents rather than its address.. Cache memories may employ associative access.
— e.g. cache

5.Performance

For random-access memory, access time is the time it takes to perform a read or write operation, that is, the time from the instant that an address is presented to the memory to the instant that data have been stored or made available for use.

For non-random-access memory, access time is the time it takes to position the read–write mechanism at the desired location.

Memory cycle time: This concept is primarily applied to random-access memory and consists of the access time plus any additional time required before a second access can commence. This additional time may be required for transients to die out on signal lines or to regenerate data if they are read destructively. Note that memory cycle time is concerned with the system bus, not the processor.

Transfer rate: This is the rate at which data can be transferred into or out of a memory unit. For random-access memory, it is equal to 1/(cycle time). For non-random-access memory, the following relationship holds:

$$T_n = T_A + n/R$$

where
T_n = Average time to read or write n bits
T_A = Average access time
n = Number of bits
R = Transfer rate, in bits per second (bps)

6.Physical Types

The most common types of memory are semiconductor memory, magnetic surface memory, used for disk and tape, and optical and magneto-optical

7.Physical Characteristics

Volatile memory, information decays naturally or is lost when electrical power is switched off.

Nonvolatile memory, information once recorded remains without deterioration until deliberately changed; no electrical power is needed to retain information. Magnetic-surface memories are nonvolatile. Semiconductor memory (memory on integrated circuits) may be either volatile or nonvolatile. Nonerasable memory cannot be altered, except by destroying the storage unit. Semiconductor memory of this type is known as read-only memory (ROM).

8.Organization

For random-access memory, organization refers to the physical arrangement of bits to form words

Eg .Interleaved ,Associative

4.3 The Memory Hierarchy

A variety of technologies are used to implement memory systems, and across this spectrum of technologies, the following relationships hold:

Faster access time(because size of ram and cache is small), greater cost per bit (semiconductor memory is costlier than secondary memory)

Greater capacity(for eg hard disk), smaller cost per bit (cost of hard disk 1tb is less than 8GB RAM)

Greater capacity(for eg hard disk), slower access time(time taken to retrieve data)

The designer would like to use memory technologies that provide for large-capacity memory, because the capacity is needed and because the cost per bit is low.

However, to meet performance requirements, the designer needs to use expensive, relatively lower-capacity memories with short access times.

The way out of this dilemma is not to rely on a single memory component or technology, but to employ a memory hierarchy

Capacity, cost and speed of different types of memory play a vital role while designing a memory system for computers.

If the memory has larger capacity, more application will get space to run smoothly.

It's better to have fastest memory as far as possible to achieve a greater performance. Moreover for the practical system, the cost should be reasonable.

There is a tradeoff between these three characteristics cost, capacity and access time. One cannot achieve all these quantities in same memory module because

If capacity increases, access time increases (slower) and due to which cost per bit decreases.

If access time decreases (faster), capacity decreases and due to which cost per bit increases.
The designer tries to increase capacity because cost per bit decreases and the more application program can be accommodated. But at the same time, access time increases and hence decreases the performance.

So the best idea will be to use memory hierarchy.

Memory Hierarchy is to obtain the highest possible access speed while minimizing the total cost of the memory system.

Not all accumulated information is needed by the CPU at the same time.

Therefore, it is more economical to use low-cost storage devices to serve as a backup for storing the information that is not currently used by CPU

The memory unit that directly communicate with CPU is called the *main memory*

Devices that provide backup storage are called *auxiliary memory*

The memory hierarchy system consists of all storage devices employed in a computer system from the slow by high-capacity auxiliary memory to a relatively faster main memory, to an even smaller and faster cache memory

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

CPU logic is usually faster than main memory access time, with the result that processing speed is limited primarily by the speed of main memory

The cache is used for storing segments of programs currently being executed in the CPU and temporary data frequently needed in the present calculations

.

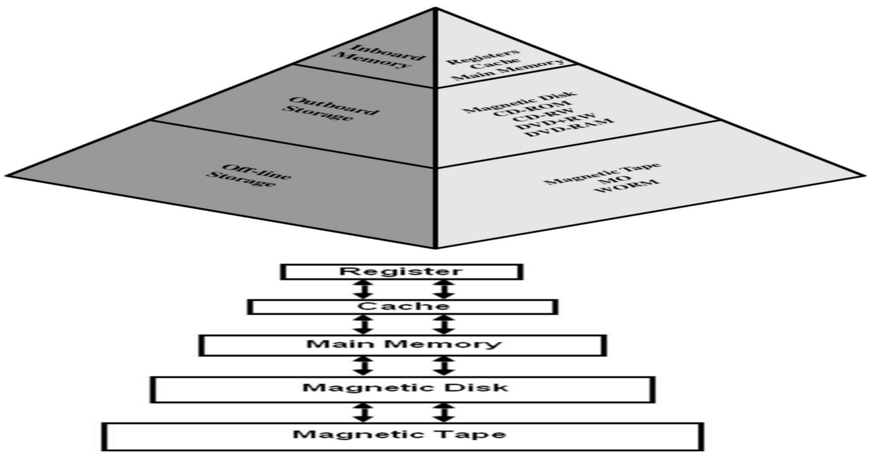


Fig 4.1: Memory Hierarchy

The memory hierarchy system consists of all storage devices employed in a computer system from slow but high capacity auxiliary memory to a relatively faster cache memory accessible to high speed processing logic. The figure below illustrates memory hierarchy

As we go down in the hierarchy.

Cost per bit decreases
Capacity of memory increases
Access time increases
Frequency of access of memory by processor also decreases.

Hierarchy List

1. Registers
2. L1 Cache
3. L2 Cache
4. Main memory
5. Disk cache
6. Disk
7. Optical
8. Tape

4.4 Internal and External memory

4.4.1 Internal or Main Memory

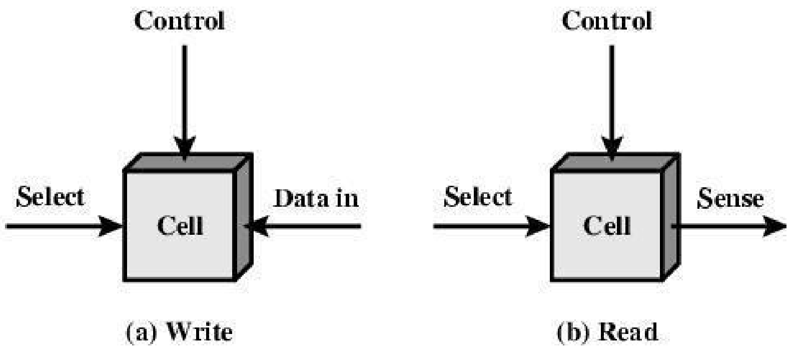
The main memory is the central unit of the computer system. It is relatively large and fast memory to store programs and data during the computer operation. These memories employ semiconductor integrated circuits. The basic element of the semiconductor memory is the memory cell.

The memory cell has three functional terminals which carries the electrical signal.

The select terminal: It selects the cell.

The data in terminal: It is used to input data as 0 or 1 and data out or sense terminal is used for the output of the cell's state.

The control terminal: It controls the function i.e. it indicates read and write.



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

4.4.1.1 RAM– Random Access memory

Memory cells can be accessed for information transfer from any desired random location.

The process of locating a word in memory is the same and requires of locating a word in memory is the same and requires an equal amount of time no matter where the cells are located physically in memory thus named 'Random access'.

Integrated RAM are available in two possible operating modes, S*tatic and Dynamic*

I.Static RAM (SRAM)

The static RAM consists of flip flop that stores binary information and this stored information remains valid as long as power is applied to the unit.

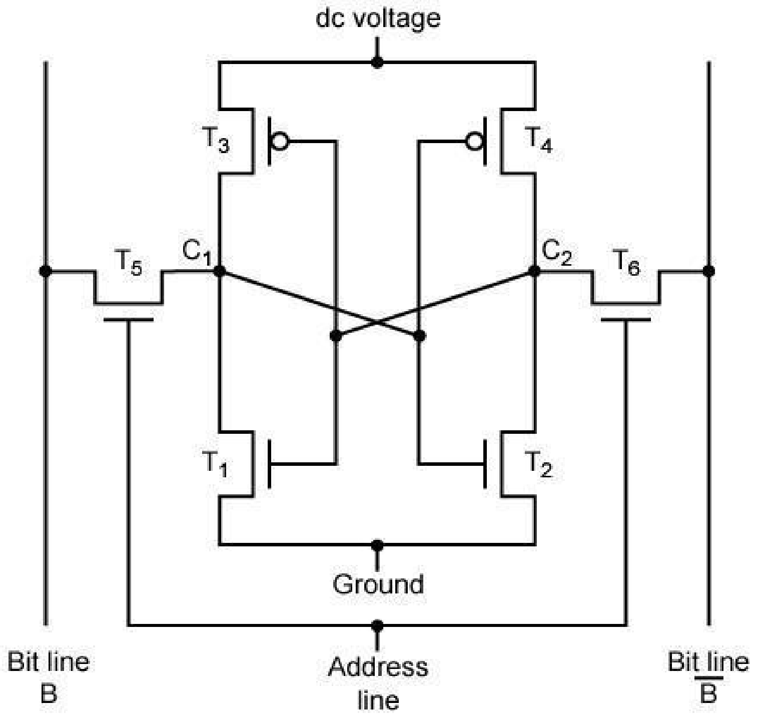


Fig 4.2: SRAM structure

Four transistors T1, T2, T3 and T4 are cross connected in an arrangement that produces a stable logical state.

In logic state 1, point C1 is high and point C2 is low. In this state, T1 & T4 are off and T2 & T3 are on.

In logic state 0, point C1 is low and C2 is high. In this state, T1 & T4 are on and T2 & T3 are off.

The address line controls the two transistors T5 & T6. When a signal is applied to this line, the two transistors are switched on allowing for read and write operation.

For a write operation, the desired bit value is applied to line B while it's complement is applied to line B complement. This forces the four transistors T1, T2, T3 & T4 into a proper state.

For the read operation, the bit value is read from line B.

II Dynamic RAM (DRAM)

The dynamic RAM stores the binary information in the form of electrical charges and capacitor is used for this purpose.

Since charge stored in capacitor discharges with time, capacitor must be periodically recharged and which is also called refreshing memory.

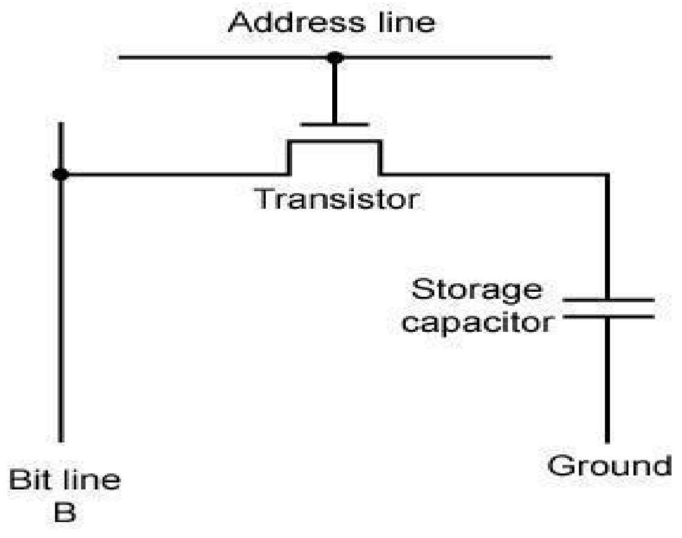


Fig 4.3: DRAM structure

The address line is activated when the bit value from this cell is to be read or written.

The transistor acts as switch that is closed i.e. allowed current to flow, if voltage is applied to the address line; and opened i.e. no current to flow, if no voltage is present in the address line.

For DRAM writing

The address line is activated which causes the transistor to conduct.

The sense amplifier senses the content of the data bus line for this cell.

If the bus line is low, then amplifier will ground the bit line of cell and any charge in capacitor is addressed out.

If data bus is high, then a +5V is applied on bit line and voltage will flow through transistor and charge the capacitor.

For DRAM reading

Address line is activated which causes the transistor to conduct.

If there is charge stored in capacitor, then current will flow through transistor and raise the voltage in bit line. The amplifier will store the voltage and place a 1 on data out line.

If there is no charge stored in capacitor, then no current will flow through transistor and voltage bit line will not be raised. The amplifier senses that there is no charge and places a 0 on data out line.

SRAM versus DRAM
Both volatile
Power needed to preserve data

Static RAM
Uses flip flop to store information
Needs more space
Faster, digital device
Expensive, big in size
Don't require refreshing circuit
Used in cache memory

Dynamic RAM

Uses capacitor to store information

More dense i.e. more cells can be accommodated per unit area
Slower, analog device
Less expensive, small in size o Needs refreshing circuit
Used in main memory, larger memory units

4.4.1.2 ROM– Read Only memory

Read only memory (ROM) contains a permanent pattern of data that cannot be changed.

A ROM is non-volatile that is no power source is required to maintain the bit values in memory.

While it is possible to read a ROM, it is not possible to write new data into it.

The data or program is permanently presented in main memory and never be loaded from a secondary storage device with the advantage of ROM.

A ROM is created like any other integrated circuit chip, with the data actually wired into the chip as part of the fabrication process.

It presents two problems

The data insertion step includes a relatively large fixed cost, whether one or thousands of copies of a particular ROM are fabricated.

There is no room for error. If one bit is wrong, the whole batch of ROM must be thrown out.

Types of ROM
Programmable ROM (PROM)

It is non-volatile and may be written into only once. The writing process is performed electrically and may be performed by a supplier or customer at a time later than the original chip fabrication.

Erasable Programmable ROM (EPROM)

It is read and written electrically. However, before a write operation, all the storage cells must be erased to the same initial state by exposure of the packaged chip to ultraviolet radiation (UV ray). Erasure is performed by shining an intense ultraviolet light through a window that is designed into the memory chip. EPROM is optically managed and more expensive than PROM, but it has the advantage of the multiple update capability.

Electrically Erasable programmable ROM (EEPROM)

This is a read mostly memory that can be written into at any time without erasing prior contents, only the byte or byte addresses are updated. The write operation takes considerably longer than the read operation, on the order of several hundred microseconds per byte. The EEPROM combines the advantage of non-volatility with the flexibility of being updatable in place, using ordinary bus control, addresses and data lines. EEPROM is more expensive than EPROM and also is less dense, supporting fewer bits per chip.

Flash Memory

Flash memory is also the semiconductor memory and because of the speed with which it can be reprogrammed, it is termed as flash. It is interpreted between EPROM and EEPROM in both cost and functionality. Like EEPROM, flash memory uses an electrical erasing technology. An entire flash memory can be erased in one or a few seconds, which is much faster than EPROM. In addition, it is possible to erase just blocks of memory rather than an entire chip. However, flash memory doesn't provide byte level erasure, a section of memory cells are erased in an action or 'flash'.

4.4.2 External Memory

The devices that provide backup storage are called external memory or auxiliary memory. It includes serial access type such as magnetic tapes and random access type such as magnetic disks.

Magnetic Tape

A magnetic tape is the strip of plastic coated with a magnetic recording medium. Data can be recorded and read as a sequence of character through read / write head. It can be stopped, started to move forward or in reverse or can be rewound.

Data on tapes are structured as number of parallel tracks running length wise. Earlier tape system typically used nine tracks. This made it possible to store data one byte at a time with additional parity bit as 9th track. The recording of data in this form is referred to as parallel recording.

Magnetic Disk

A magnetic disk is a circular plate constructed with metal or plastic coated with magnetic material often both side of disk are used and several disk stacked on one spindle which Read/write head available on each surface. All disks rotate together at high speed. Bits are stored in magnetize surface in spots along concentric circles called tracks. The tracks are commonly divided into sections called sectors. After the read/write head are positioned in specified track the system has to wait until the rotating disk reaches the specified sector under read/write head.

Information transfer is very fast once the beginning of sector has been reached. Disk that are permanently attached to the unit assembly and cannot be used by occasional user are called hard disk drive with removal disk is called floppy disk

Optical Disk

The huge commercial success of CD enabled the development of low cost optical disk storage technology that has revolutionized computer data storage. The disk is form from resin such as polycarbonate. Digitally recorded information is imprinted as series of microscopic pits on the surface of poly carbonate. This is done with the finely focused high intensity leaser. The pitted surface is then coated with reflecting surface usually aluminum or gold. The shiny surface is protected against dust and scratches by the top coat of acrylic.

Information is retrieved from CD by low power laser. The intensity of reflected light of laser changes as it encounters a pit. Specifically if the laser beam falls on pit which has somewhat rough surface the light scatters and low intensity is reflected back to the surface. The areas between pits are called lands. A land is a smooth surface which reflects back at higher intensity. The change between pits and land is detected by photo sensor and converted into digital signal. The sensor tests the surface at regular interval.

DVD-Technology

Multi-layer
Very high capacity (4.7G per layer)
Full length movie on single disk
Using MPEG compression
Finally standardized (honest!)
Movies carry regional coding
Players only play correct region films

DVD-Writable

Loads of trouble with standards
First generation DVD drives may not read first generation DVD-W disks
First generation DVD drives may not read CD-RW disks

4.5 Cache memory principles

Principles

Intended to give memory speed approaching that of fastest memories available but with large size, at close to price of slower memories *Cache is checked first for all memory references.If not found, the entire block in which that reference resides in main memory is stored in a cache slot, called a line*

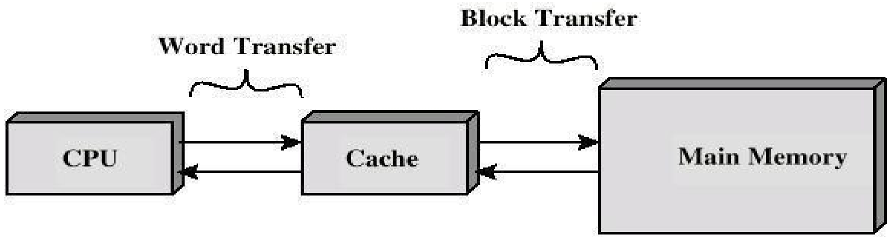
Each line includes a tag (usually a portion of the main memory address) which identifies which particular block is being stored

Locality of reference implies that future references will likely come from this block of memory, so that cache line will probably be utilized repeatedly.
The proportion of memory references, which are found already stored in cache, is called the **hit ratio.**

Cache memory is intended to give memory speed approaching that of the fastest memories available, and at the same time provide a large memory size at the price of less expensive types of semiconductor memories. There is a relatively large and slow main memory together with a smaller, faster cache memory contains a copy of portions of main memory.

When the processor attempts to read a word of memory, a check is made to determine if the word is in the cache. If so, the word is delivered to the processor. If not, a block of main memory, consisting of fixed number of words is read into the cache and then the word is delivered to the processor.

The locality of reference property states that over a short interval of time, address generated by a typical program refers to a few localized area of memory repeatedly. So if programs and data which are accessed frequently are placed in a fast memory, the average access time can be reduced. This type of small, fast memory is called cache memory which is placed in between the CPU and the main memory.



When the CPU needs to access memory, cache is examined. If the word is found in cache, it is read from the cache and if the word is not found in cache, main memory is accessed to read word. A block of word containing the one just accessed is then transferred from main memory to cache memory.

Cache connects to the processor via data control and address line. The data and address lines also attached to data and address buffer which attached to a system bus from which main memory is reached

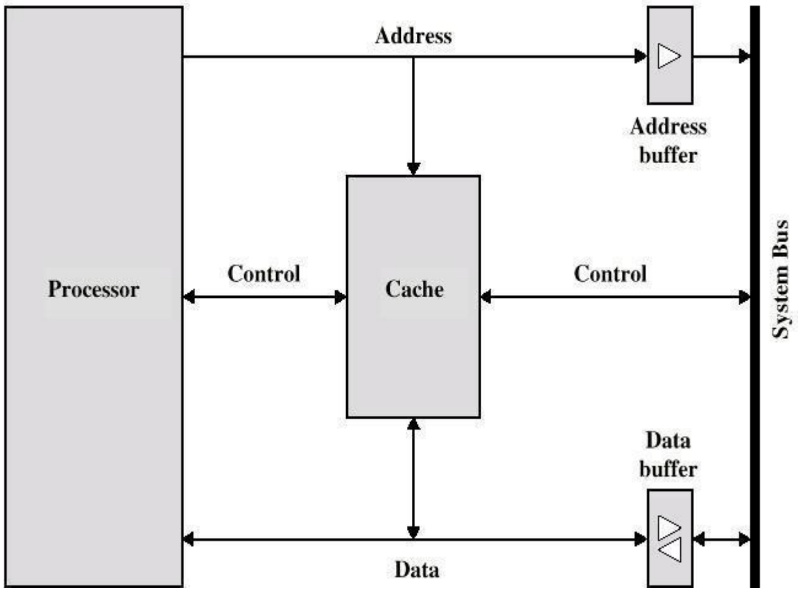
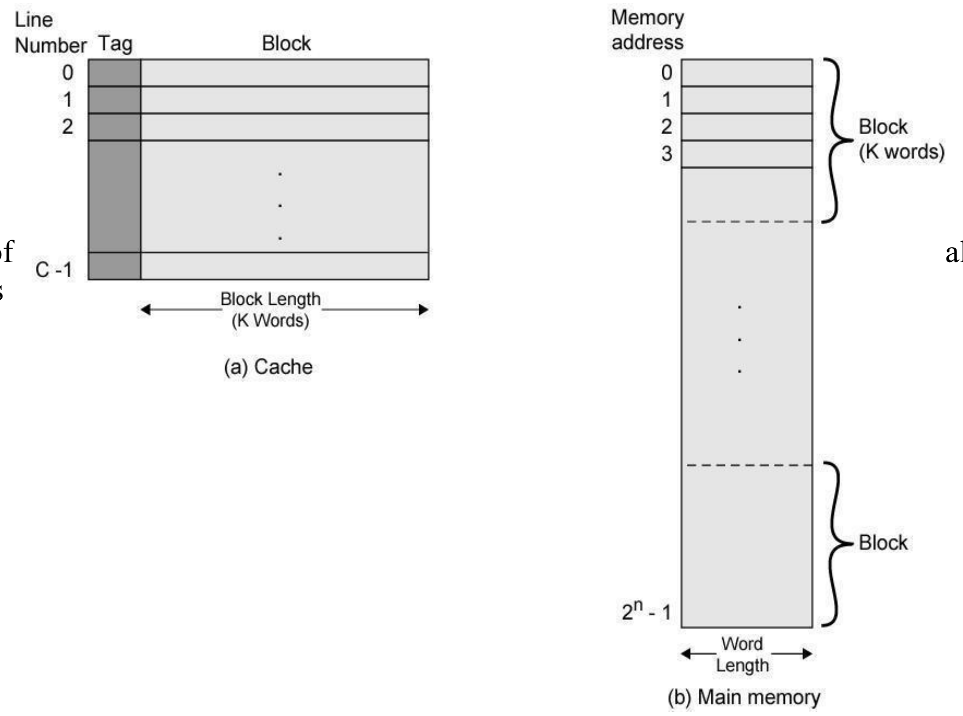


Fig 4.4: Typical Cache organization

When a cache hit occurs, the data and address buffers are disabled and the communication is only between processor and cache with no system bus traffic. When a cache miss occurs, the desired word is first read into the cache and then transferred from cache to processor. For later case, the cache is physically interposed between the processor and main memory for all data, address and control lines.

Cache Operation Overview



05 of 'views all all types

Fig 4.5: Cache memory / Main memory structure

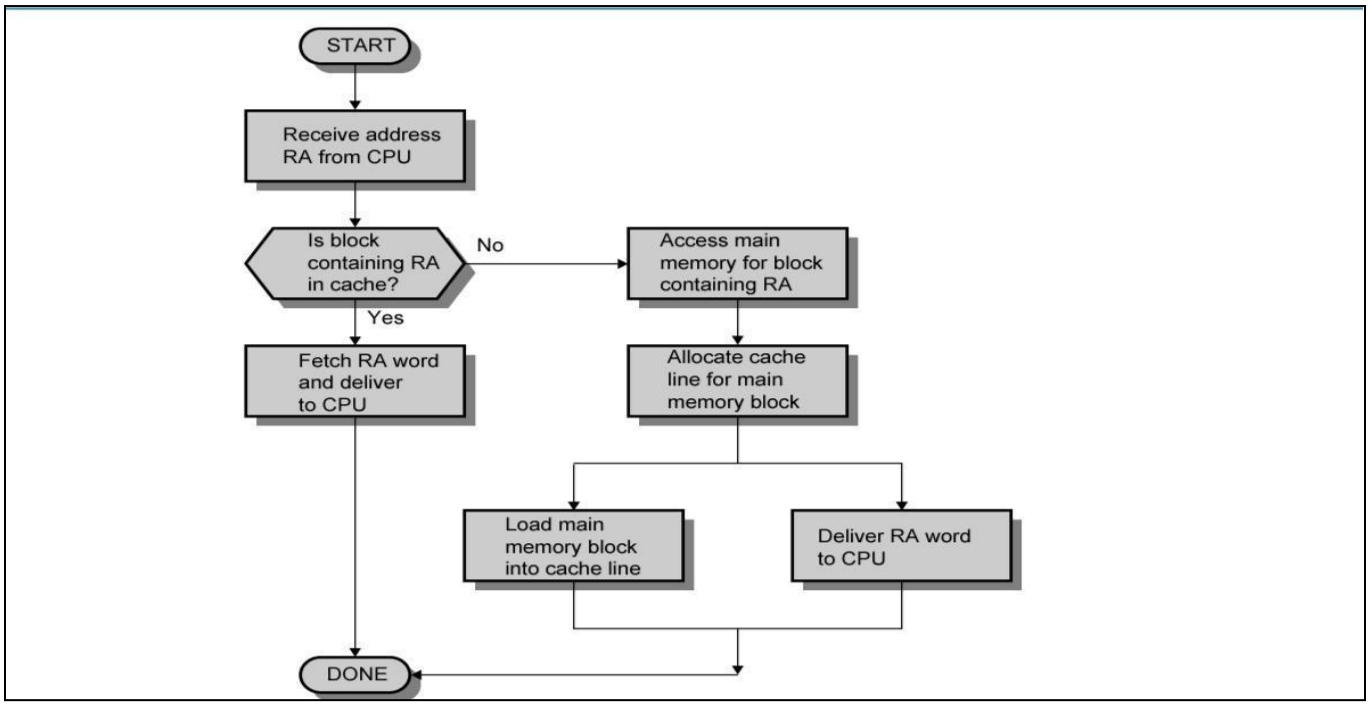

START
Receive address RA from CPU
Is block containing RA in cache?
No
Access main memory for block containing RA
Yes
Fetch RA word and deliver to CPU
Allocate cache line for main memory block
Load main memory block into cache line
Deliver RA word to CPU
DONE

Fig 4.6: Flowchart for cache read operation

CPU generates the receive address (RA) of a word to be moved (read).

Check a block containing RA is in cache.

If present, get from cache (fast) and return.

If not present, access and read required block from main memory to cache.

Allocate cache line for this new found block.

Load bock for cache and deliver word to CPU

Cache includes tags to identify which block of main memory is in each cache slot

Locality of Reference

The reference to memory at any given interval of time tends to be confined within a few localized area of memory. This property is called locality of reference. This is possible because the program loops and subroutine calls are encountered frequently. When program loop is executed, the CPU will execute same portion of program repeatedly. Similarly, when a subroutine is called, the CPU fetched starting address of subroutine and executes the subroutine program. Thus loops and subroutine localize reference to memory.

This principle states that memory references tend to cluster over a long period of time, the clusters in use changes but over a short period of time, the processor is primarily working with fixed clusters of memory references.

Spatial Locality

It refers to the tendency of execution to involve a number of memory locations that are clustered.

It reflects tendency of a program to access data locations sequentially, such as when processing a table of data.

Temporal Locality

It refers to the tendency for a processor to access memory locations that have been used frequently. For e.g. Iteration loops executes same set of instructions repeatedly.

4.6 Elements of Cache design

4.6.1 Cache size

Size of the cache to be small enough so that the overall average cost per bit is close to that of main memory alone and large enough so that the overall average access time is close to that of the cache alone.

The larger the cache, the larger the number of gates involved in addressing the cache.
Large caches tend to be slightly slower than small ones – even when built with the same integrated circuit technology and put in the same place on chip and circuit board.

The available chip and board also limits cache size.

4.6.2 Mapping function

The transformation of data from main memory to cache memory is referred to as memory mapping process.

Because there are fewer cache lines than main memory blocks, an algorithm is needed for mapping main memory blocks into cache lines.

There are three different types of mapping functions in common use and are direct, associative and set associative. All the three include following elements in each example.

The cache can hold 64 Kbytes

Data is transferred between main memory and the cache in blocks of 4 bytes each. This means that the cache is organized as 16Kbytes = 2^{14} lines of 4 bytes each.

The main memory consists of 16 Mbytes with each byte directly addressable by a 24 bit address (2^{24} = 16Mbytes). Thus, for mapping purposes, we can consider main memory to consist of 4Mbytes blocks of 4 bytes each.

4.6.2.1 Direct Mapping

It is the simplex technique, maps each block of main memory into only one possible cache line i.e. a given main memory block can be placed in one and only one place on cache.

$$i = j \text{ modulo } m$$

Where i = cache line number; j = main memory block number; m = number of lines in the cache

The mapping function is easily implemented using the address. For purposes of cache access, each main memory address can be viewed as consisting of three fields.

The least significant w bits identify a unique word or byte within a block of main memory. The remaining s bits specify one of the 2^s blocks of main memory.

The cache logic interprets these s bits as a tag of (s-r) bits most significant position and a line field of r bits. The latter field identifies one of the $m = 2^r$ lines of the cache.

Tag s-r	Line or Slot r	Word w
8	14	2

Address length = (s + w) bits
Number of addressable units = 2^{s+w} words or bytes
Block size = line size = 2^w words or bytes
Number of blocks in main memory = $2^{s+w}/2^w = 2^s$
Number of lines in cache = $m = 2^r$
Size of tag = (s – r) bits

24 bit address
2 bit word identifier (4 byte block)
22 bit block identifier
8 bit tag (=22-14), 14 bit slot or line
No two blocks in the same line have the same Tag field
Check contents of cache by finding line and checking Tag

Cache line	**Main Memory blocks held**
0	0, m, 2m, 3m…2s-m
1	1,m+1, 2m+1…2s-m+1
m-1	m-1, 2m-1,3m-1…2s-1

Cache Line	**0**	**1**	**2**	**3**	**4**
Main Memory Block	0	1	2	3	4
	5	6	7	8	9
	10	11	12	13	14
	15	16	17	18	19
	20	21	22	23	24

Note that

o all locations in a single block of memory have the same higher order bits (call them the block number), so the lower order bits can be used to find a particular word in the block.

o within those higher-order bits, their lower-order bits obey the modulo mapping given above (assuming that the number of cache lines is a power of 2), so they can be used to get the cache line for that block

the remaining bits of the block number become a tag, stored with each cache line, and used to distinguish one block from another that could fit into that same cache *line.*

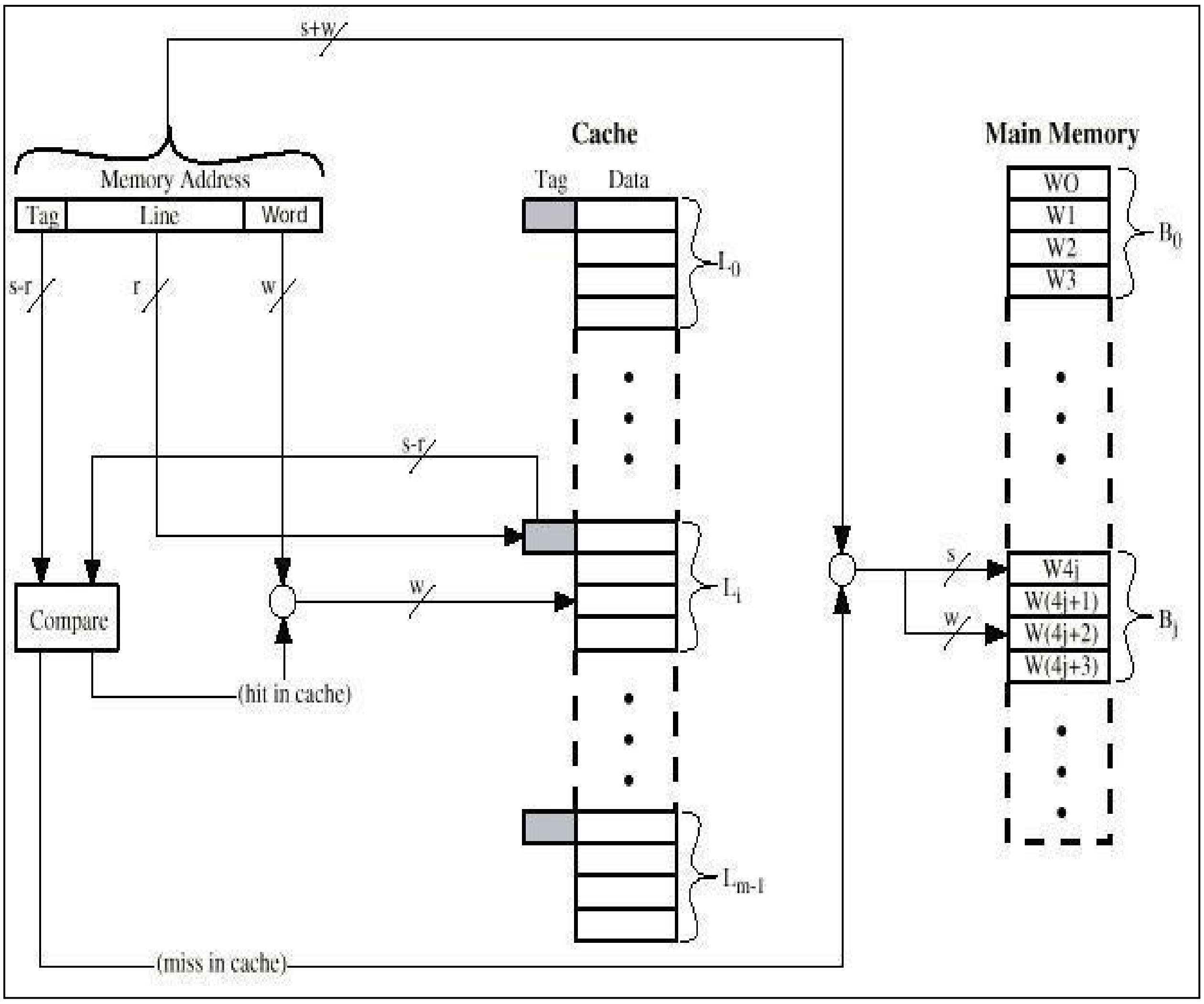


Fig 4.7: Direct mapping structure

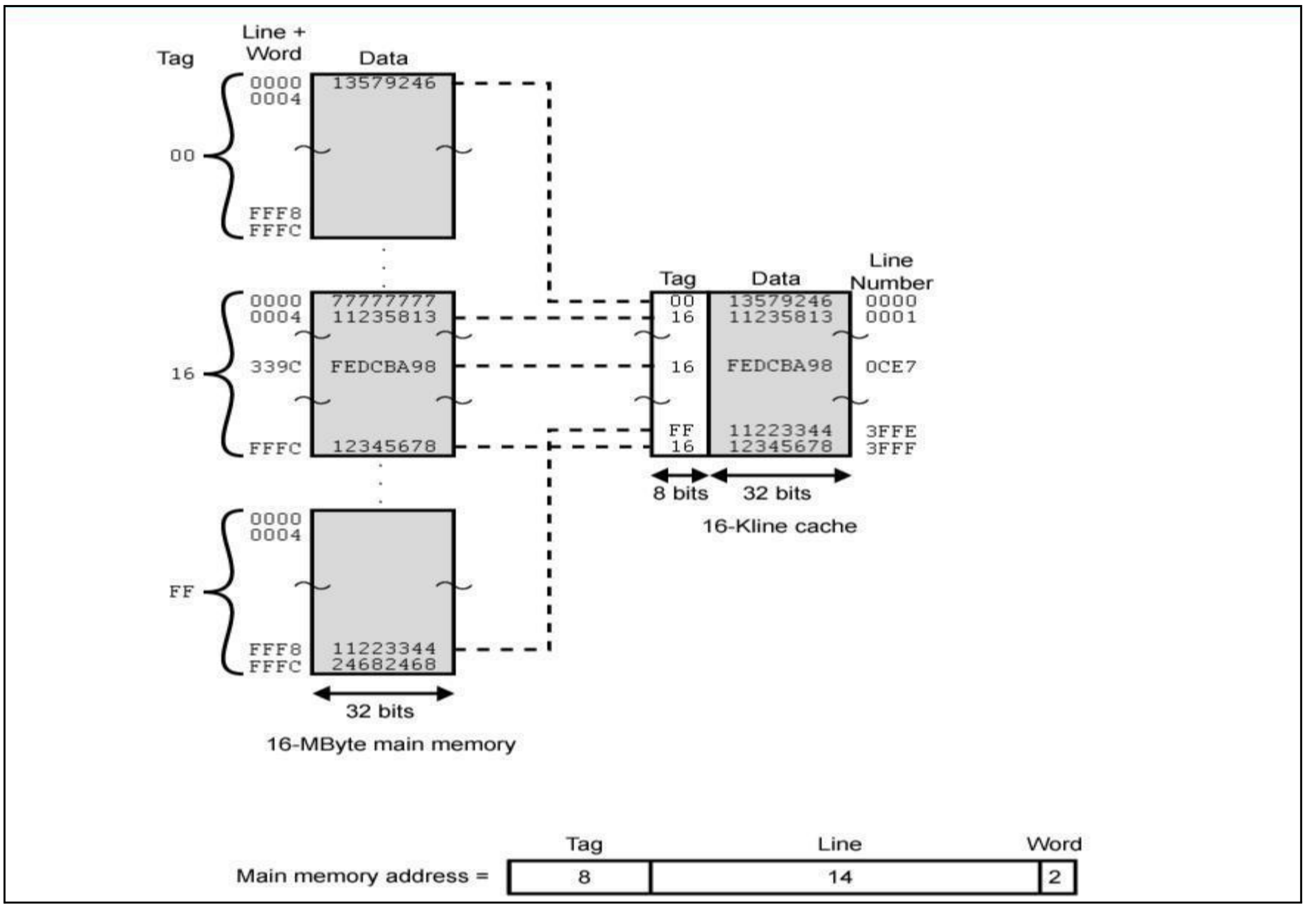


Fig:4.8 Direct mapping example

Pros and Cons
Simple
Inexpensive
Fixed location for given block

If a program accesses 2 blocks that map to the same line repeatedly, cache misses are very high

4.6.2.2 Associated Mapping

It overcomes the disadvantage of direct mapping by permitting each main memory block to be loaded into any line of cache.

Cache control logic interprets a memory address simply as a tag and a word field
Tag uniquely identifies block of memory

Cache control logic must simultaneously examine every line's tag for a match which requires fully associative memory

very complex circuitry, complexity increases exponentially with size
Cache searching gets expensive

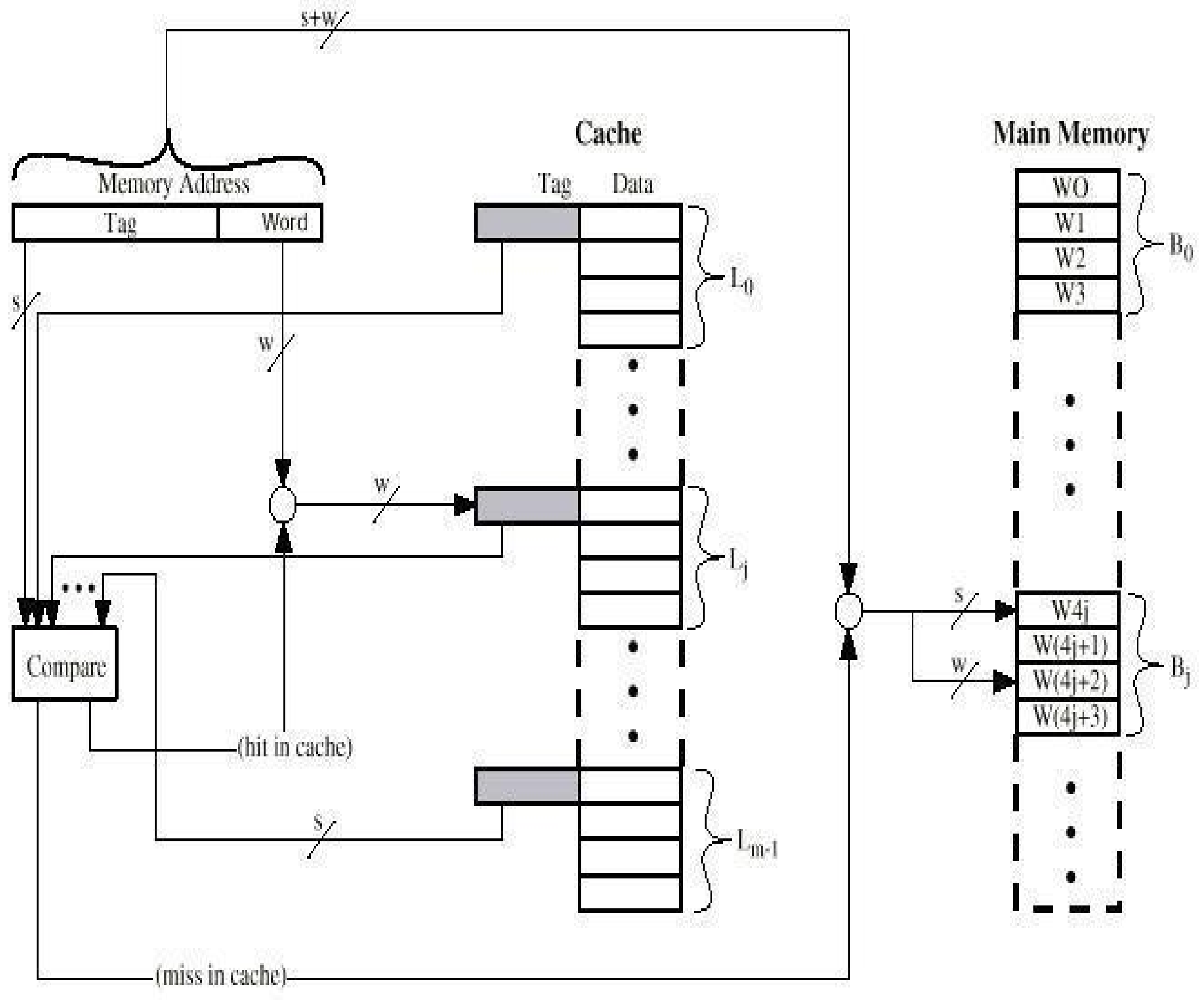


Fig 4.9: Associative structure

Address length = (s + w) bits eg 22+2=24

Number of addressable units = 2^{s+w} words or bytes eg 2^{22+2} =16777216

Block size = line size = 2^{w} words or bytes

Number of blocks in main memory = $2^{s+w}/2^{w} = 2^{s}$

Number of lines in cache = undetermined, Size of tag = s bits

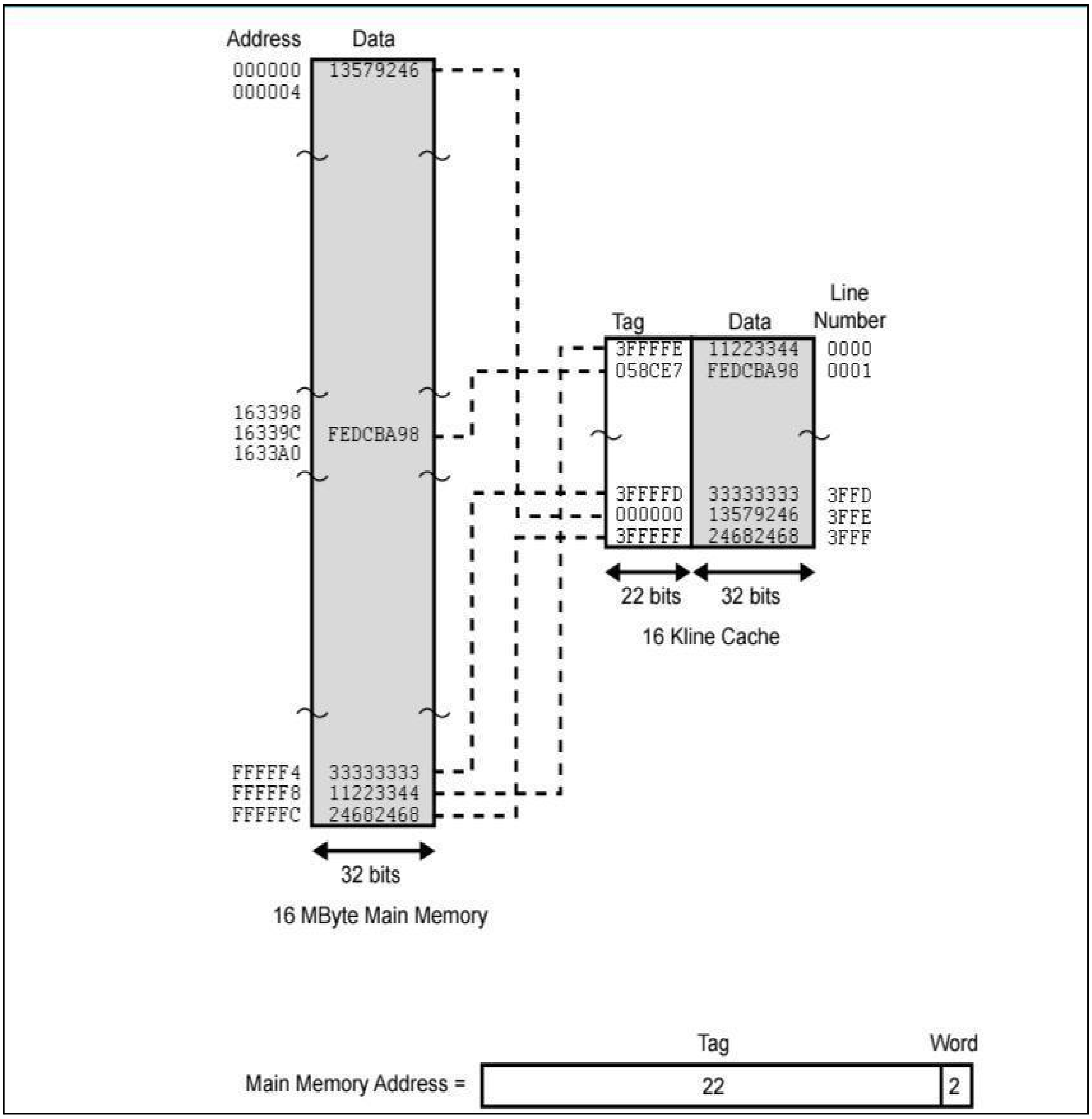


Fig 4.10: Associative mapping example

22 bit tag stored with each 32 bit block of data
Compare tag field with tag entry in cache to check for hit
Least significant 2 bits of address identify which 16 bit word is required from 32 bit data block

Address	Tag	Data	Cache line
FFFFFC	FFFFFC	24682468	3FFF

4.6.2.3 Set Associated Mapping

It is a compromise between direct and associative mappings that exhibits the strength and reduces the disadvantages

Cache is divided into v sets, each of which has k lines;
number of cache lines = vk
$M = v \, X \, k$

$$I = j \text{ modulo } v$$

Where, i = cache set number; j = main memory block number; v = number of lines in the cache

So a given block will map directly to a particular set, but can occupy any line in that set (associative mapping is used within the set)

Cache control logic interprets a memory address simply as three fields tag, set and word. The d set bits specify one of $v = 2^d$ sets. Thus s bits of tag and set fields specify one of the 2^s block of main memory.

The most common set associative mapping is 2 lines per set, and is called two-way set associative. It significantly improves hit ratio over direct mapping, and the associative hardware is not too expensive.

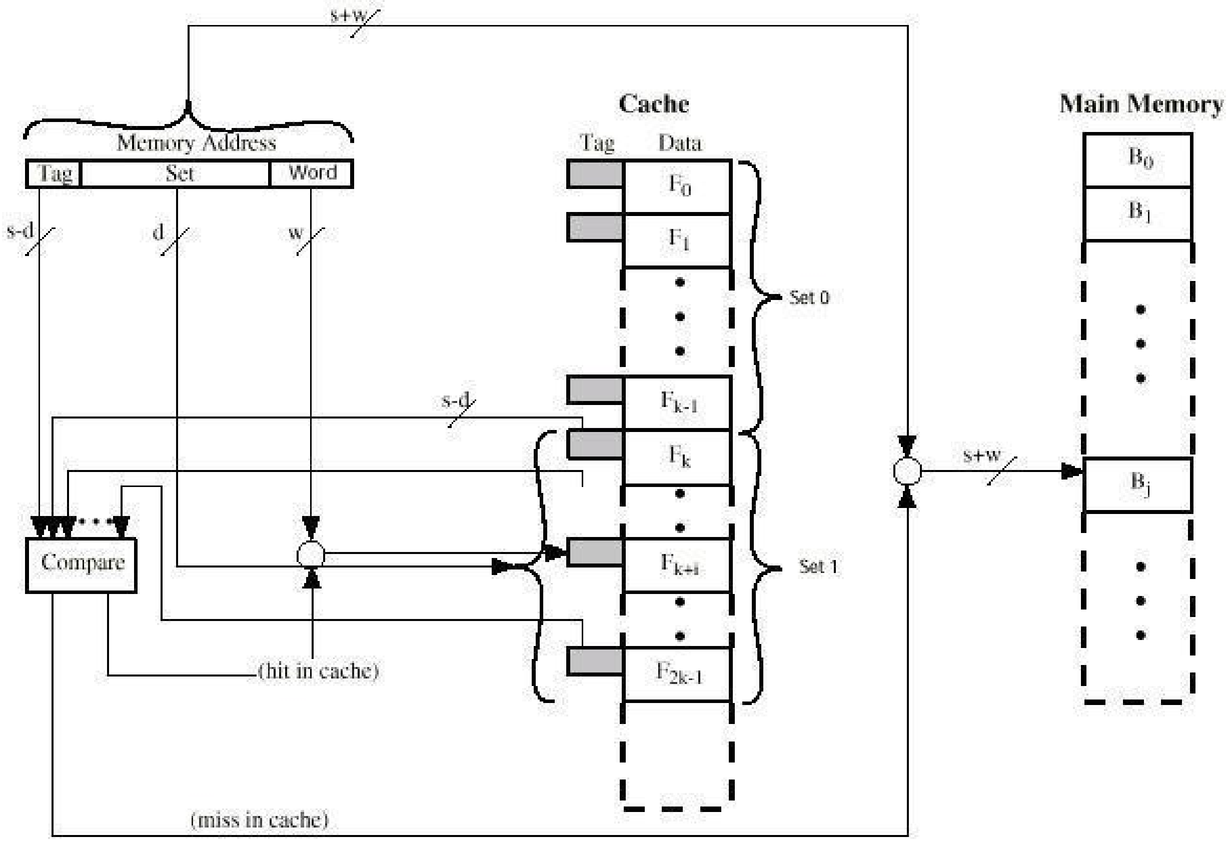


Fig 4.11: Set associative mapping structure

Address length = (s + w) bits eg 22+2=24
Number of addressable units = 2^{s+w} words or bytes eg 2^{22+2} =16777216
Block size = line size = 2^w words or bytes eg w=2 2^2=4
Number of blocks in main memory = 2^s 2^{22}=4194304
Number of lines in set = k k=2
Number of sets = v = 2^d 2^{13}=8192
Number of lines in cache = kv = k * 2^d 2*8192=16384
Size of tag = (s – d) bits

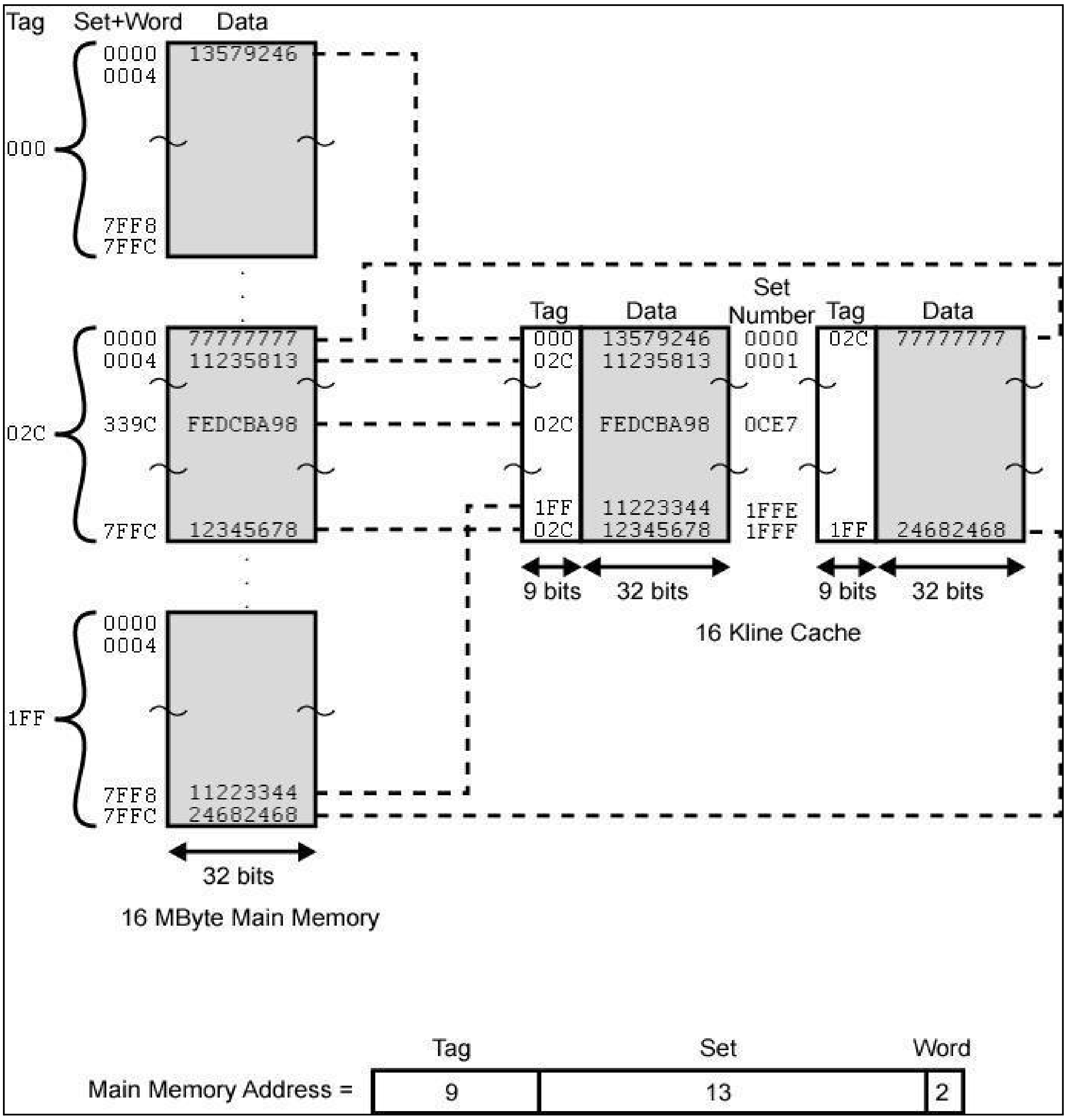


Fig: Set associative mapping example

13 bit set number
Block number in main memory is modulo 2^{13}
000000, 00A000, 00B000, 00C000 … map to same set
Use set field to determine cache set to look in

Compare tag field to see if we have a hit

e.g

Address	Tag	Data	Set number
1FF 7FFC	1FF	12345678	1FFF
001 7FFC	001	11223344	1FFF

4.6.3 Replacement algorithm

When all lines are occupied, bringing in a new block requires that an existing line be overwritten.

Direct mapping
No choice possible with direct mapping
Each block only maps to one line
Replace that line

Associative and Set Associative mapping
Algorithms must be implemented in hardware for speed

4.6.3.1 First in first out (FIFO)

Replace that block in the set which has been in the cache longest

Implementation: use a round-robin or circular buffer technique (keep up with which slot's "turn" is next

Example 1: Consider page reference string 3, 5, 2, 5, 7, 8, 5 with 3 page frames.Find the number of page faults.

3	5	2	5	7	8	5			
		2	2	2	2	5			
	5	5	5	5	8	8			
3	3	3	3	7	7	7			
M	M	M	H	M	M	M			

Example-2: Consider the page references 5,2,7,3, 2, 0, 2,1, 3,0,2, 0, 3, 0 with 3 page frame. Find number of page fault. Using FIFO

5	**2**	7	**3**	2	0	2	1	3	0	2	0	3	0
		7	7	7	7	2	2	2	0	0	0	0	0
	2	2	2	2	0	0	0	3	3	3	3	3	3
5	5	5	3	3	3	3	1	1	1	2	2	2	2
5	5,2	5,2,7	2,7,3	2,7,3	7,3,0	3,0,2	0,2,1	2,1,3	1,3,0	3,0,2	3,0,2	3,0,2	3,0,2
M	M	M	M	H	M	M	M	M	M	M	H	H	H

MISS =10 HITS=4

4.6.3.2 Least Recently used (LRU)

Replace that block in the set which has been in cache longest with no reference to it

Example 1: Consider page reference string 3, 5, 2, 5, 7, 8, 5 with 3 page frames.Find the number of page faults.

3	5	2	5	7	8	5													
		2	2	2	8	8													
	5	5	5	5	5	5													
3	3	3	3	7	7	7													
M	M	M	H	M	M	H													

MISS =6 HITS=8

Example-2: Consider the page references 5,2,7,3, 2, 0, 2,1, 3,0,2, 0, 3, 0 with 3 page frame. Find number of page fault using LRU

5	**2**	7	**3**	2	0	2	1	3	0	2	0	3	0
		7	7	7	0	0	1	1	1	2	2	2	2
	2	2	2	2	2	2	2	2	0	0	0	0	0
5	5	5	3	3	3	3	0	3	3	3	3	3	3
5	5,2	5,2,7	2,7,3	7,3,2	3,2,0	3,0,2	0,2,1	2,1,3	1,3,0	3,0,2	3,2,0	2,0,3	2,3,0
M	M	M	M	H	M	H	M	M	M	M	H	H	H

MISS =9 HITS=5

4.6.3.3 Optimal Page replacement:

In this algorithm, pages are replaced which would not be used for the longest duration of time in the future.

Example-2: Consider the page references 5,2,7,3, 2, 0, 2,1, 3,0,2, 0, 3, 0 with 4 page frame. Find number of page fault.

5	**2**	7	**3**	2	0	2	1	3	0	2	0	3	0						
			3	3	3	3	3	3	3	3	3	3	3						
		7	7	7	7	7	1	1	1	1	1	1	1						
	2	2	2	2	2	2	2	2	2	2	2	2	2						
5	5	5	5	5	0	0	0	0	0	0	0	0	0						
M	M	M	M	H	M	H	M	H	H	H	H	H	H						

MISS =6 HITS=8

4.6.4 Write policy

When a line is to be replaced, must update the original copy of the line in main memory if any addressable unit in the line has been changed

If a block has been altered in cache, it is necessary to write it back out to main memory before replacing it with another block (writes are about 15% of memory references)

Must not overwrite a cache block unless main memory is up to date
I/O modules may be able to read/write directly to memory

Multiple CPU's may be attached to the same bus, each with their own cache

4.7 Cache Coherency

4.7.1 Write Through

All write operations are made to main memory as well as to cache, so main memory is always valid

Other CPU's monitor traffic to main memory to update their caches when needed.This generates substantial memory traffic and may create a bottleneck.Anytime a word in cache is changed, it is also changed in main memory.Both copies always agree

Generates lots of memory writes to main memory.Multiple CPUs can monitor main memory traffic to keep local (to CPU) cache up to date

Lots of traffic
Slows down writes
Remember bogus write through caches!

4.7.1.2 Write back

When an update occurs, an UPDATE bit associated with that slot is set, so when the block is replaced it is written back first
During a write, only change the contents of the cache
Update main memory only when the cache line is to be replaced

Causes "cache coherency" problems -- different values for the contents of an address are in the cache and the main memory
Complex circuitry to avoid this problem
Accesses by I/O modules must occur through the cache

Multiple caches still can become invalidated, unless some cache coherency system is used. Such systems include:

o Bus Watching with Write Through - other caches monitor memory writes by other caches (using write through) and invalidates their own cache line if a match

o Hardware Transparency - additional hardware links multiple caches so that writes to one cache are made to the others

o Non-cacheable Memory - only a portion of main memory is shared by more than one processor, and it is non-cacheable

4.8 Number of caches

L1 and L2 Cache

On-chip cache (L1 Cache)

It is the cache memory on the same chip as the processor, the on-chip cache. It reduces the processor's external bus activity and therefore speeds up execution times and increases overall system performance.

Requires no bus operation for cache hits
Short data paths and same speed as other CPU transactions

Off-chip cache (L2 Cache)

It is the external cache which is beyond the processor. If there is no L2 cache and processor makes an access request for memory location not in the L1 cache, then processor must access DRAM or ROM memory across the bus. Due to this typically slow bus speed and slow memory access time, this results in poor performance. On the other hand, if an L2 SRAM cache is used, then frequently the missing information can be quickly retrieved.

It can be much larger

It can be used with a local bus to buffer the CPU cache-misses from the system bus

Unified and Split Cache

Unified Cache

Single cache contains both instructions and data. Cache is flexible and can balance "allocation" of space to instructions or data to best fit the execution of the program.

Has a higher hit rate than split cache, because it automatically balances load between data and instructions (if an execution pattern involves more instruction fetches than data fetches, the cache will fill up with more instructions than data)

Only one cache need be designed and implemented

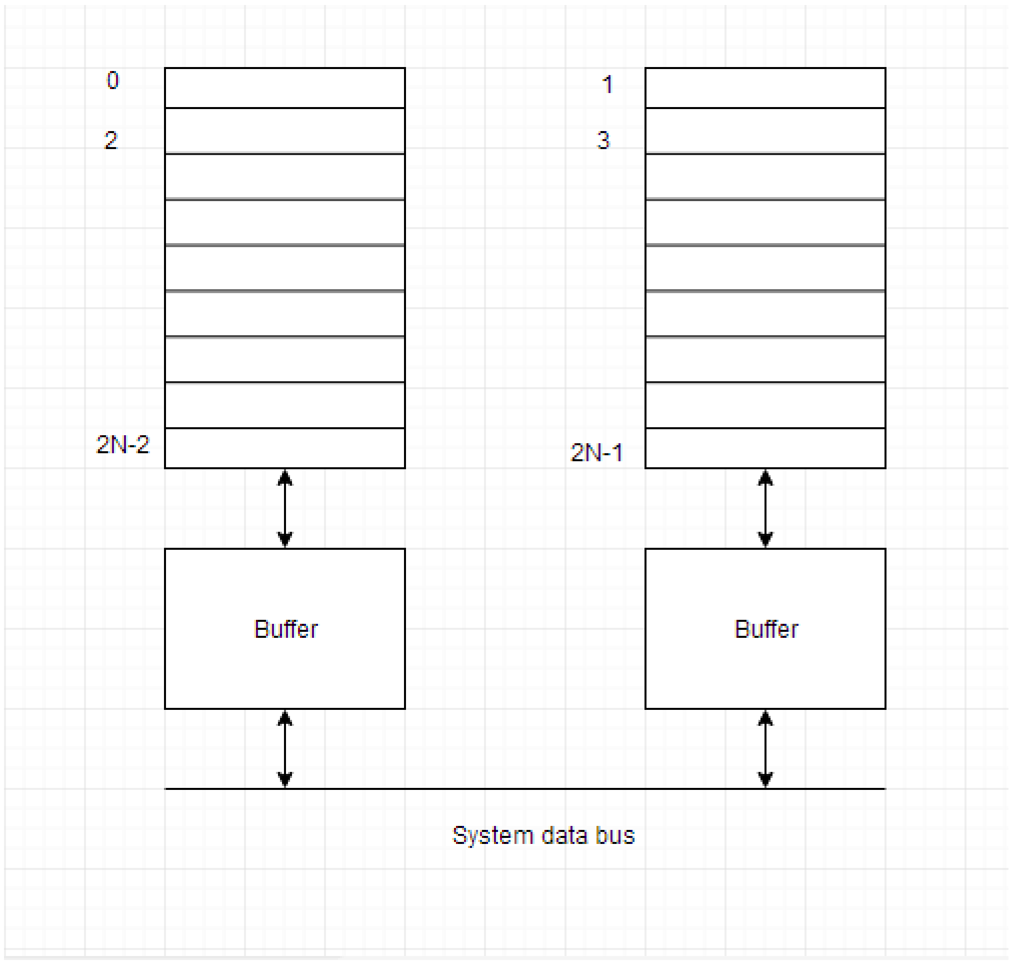
Split Cache

Cache splits into two parts first for instruction and second for data. Can outperform unified cache in systems that support parallel execution and pipelining (reduces cache contention)Trend is toward split cache because of superscalar CPU's
Better for pipelining, pre-fetching, and other parallel instruction execution designs
Eliminates cache contention between instruction processor and the execution unit (which uses data)

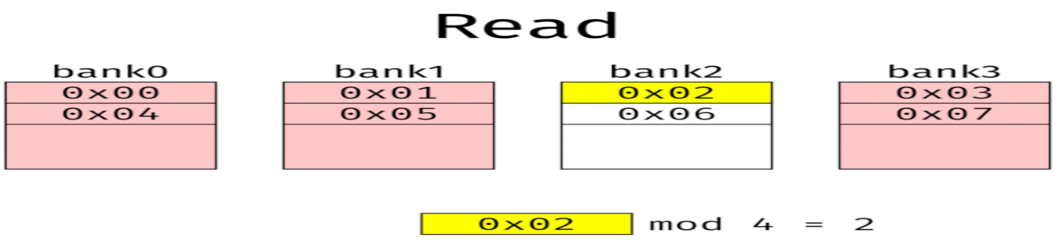
Interleaved Memory

It is a technique for compensating the relatively slow speed of DRAM(Dynamic RAM). In this technique, the main memory is divided into memory banks which can be accessed individually without any dependency on the other.

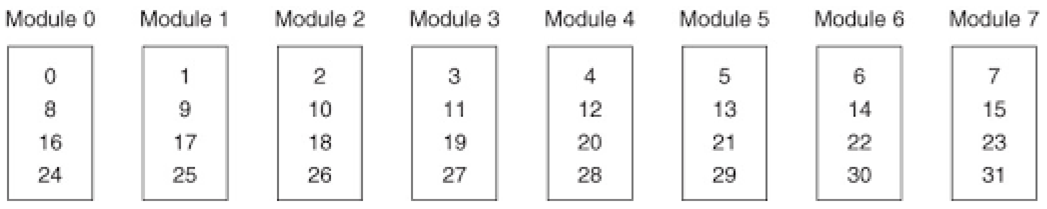
Types:
There are two methods for interleaving a memory:
2-Way Interleaved:-Two memory blocks are accessed at same time for writing and reading operations.
4-Way Interleaved:-Four memory blocks are accessed at the same time.
2-Way Interleaved Example
The idea of interleaving memory is that memory is divided into banks.
In 2-Way Interleaved there will be two banks.One will hold data whose address is an odd number and other bank will hold data whose address are even



If the memory is 4–way interleaved, it is implemented using four banks, numbered 0 through 3.



If the memory is 8–way interleaved, it is implemented using eight banks, numbered 0 through 7.



Consider a 64–word memory that is 2–way interleaved. This means that there are two memory banks, each holding 32 words.

If this memory is also low–order interleaved, we have the following allocation of words to banks.

Bank 0(even): Words 00, 02,04,06 08,10 12,14 16,18 20,22 24, …, 60,62

Bank 1(odd): Words 01,03,05,07, 09,11,13,15,17,19, 21,23 25, …, 61,63

When a memory is N–way interleaved, we always find that N = 2^K.

This is due to the structure of the memory address.

For K = 1, we have 2–way interleaving.

For K = 2, we have 4–way interleaving.

For K = 3, we have 8–way interleaving.

For K = 4, we have 16–way interleaving.

For each scheme, the low–order K bits of the address select the bank.

For a 64–word,**2-Way** low–order interleaved memory, the address structure is as follows. Each address is a 6–bit unsigned binary number.

Bit	5	4	3	2	1	0
Use	Offset of word in a given bank					Bank number

<u>An Example of Low–Order Interleaving</u>

The first 12 words in the 64 word memory that is 2–way low–order interleaved.

Bank	Word Address	Binary Address	Split Address	Offset in Bank	Bank Number
Bank 0:	00	000000	00000 0	0	0
	02	000010	00001 0	1	0
	04	000100	00010 0	2	0
06	000110	00011 0	3	0	
	08	001000	00100 0	4	0
Bank 1:	01	000001	00000 1	0	1
	03	000011	00001 1	0	1
	05	000101	00010 1	1	1
	07	000111	00011 1	1	1
	09	001001	00100 1	2	1
	11	001011	00101 1	2	1

4-Way Interleaved example

For a 64–word, 4–way low–order interleaved memory, the address structure is as follows. Each address is a 6–bit unsigned binary number.

Bit	5	4	3	2	1	0
Use	Offset of word in a given bank				Bank number	

An Example of Low–Order Interleaving

The first 12 words in the 64 word memory that is 4–way low–order interleaved.

Bank	Word Address	Binary Address	Split Address	Offset in Bank	Bank Number
Bank 0:	00	000000	0000 00	0	0
	04	000100	0001 00	1	0
	08	001000	0010 00	2	0
Bank 1:	01	000001	0000 01	0	1
	05	000101	0001 01	1	1
	09	001001	0010 01	2	1
Bank 2:	02	000010	0000 10	0	2
	06	000110	0001 10	1	2
	10	001010	0010 10	2	2
Bank 3:	03	000011	0000 11	0	3
	07	000111	0001 11	1	3
	11	001011	0010 11	2	3

Another Example of Low–Order Interleaving

Consider a 64–word memory that is 8–way low–order interleaved.

Here is the address structure. Here $8 = 2^3$.

Bit	5	4	3	2	1	0
Field	Offset in the bank			Bank number		

Here is the structure seen in the first two banks of this memory.

Bank	**Word Address**	**Binary Address**	**Split Address**	**Offset in Bank**	**Bank Number**
Bank 0:	00	000000	000 000	0	0
	08	001000	001 000	1	0
	16	010000	010 000	2	0
Bank 1:	01	000001	000 001	0	1
	09	001001	001 001	1	1
	17	010001	010 001	2	1

Associative memory

It is also known as content addressable memory (CAM). It is a memory chip in which each bit position can be compared. In this the content is compared in each bit cell which allows very fast table lookup. Since the entire chip can be compared, contents are randomly stored without considering addressing scheme. These chips have less storage capacity than regular memory chips.

The block diagram of an associative memory is displayed in Figure below. It comprises of a memory array and logic for m words with n bits per word. Argument register A and key register K both have n bits, one for every bit of a word. Match register M has m bits, one for each memory word. Every word in memory is compared in parallel with content of argument register then the words which match the bits of argument register set a corresponding bit in match register. After matching process those bits in match register which have been set denote the fact that their corresponding words have been matched. Reading is achieved by a sequential access to memory for those words whose corresponding bits in match register have been set.

Key register offers a mask for choosing a specific key or field in argument word. The complete argument is compared with every memory word if key register contains all 1s. Or else only those bits in argument which have 1s in their corresponding positions of key register are compared. So the key offers a mask or identifying information that specifies how reference to memory is made.

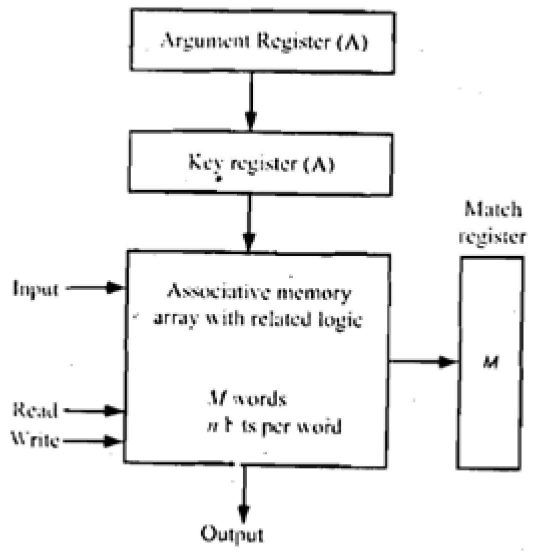


Figure: Associative Memory - Block Diagram

To explain with a numerical illustration assume that argument register A and key register K have the bit configuration displayed below. Only three leftmost bits of a compared with memory words since K has 1's on these positions.

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

Word 2 matches unmasked argument field since the three leftmost bits of argument and word are equal.

Question Bank

Two marks

1.Define locality of reference

2.List any two difference between DRAM and static RAM

3.List any two difference between RAM and ROM

4.Define spatial locality

5.Define temporal locality

6.List any two difference between spatial locality and temporal locality

1	List different memory organization characteristics	5
2	Compare SRAM & DRAM.	10
3	What are elements of cache design? Explain in details.	10
4	Explain in set associative and associative cache mapping techniques (or) Explain different mapping techniques of Cache memory.	10
5	Explain in set associative and associative cache mapping techniques (or) Explain different mapping techniques of Cache memory.	10
6	Calculate the hit and missing various page replacement policy LRU,OPT,FIFO for following sequence (page,The size 3) 2,3,1,2,4,3,2,5,3,6,7,9,3,7 . State which one is best for above example?	10
7	Calculate the hit and missing various page replacement policy LRU,OPT,FIFO for following sequence (page,The size 3) 4,7,3,0,1,7,3,8,5,4,5,3,4,7 . State which one is best for above example?	10
8	Consider the string 1,3,2,4,2.1,5,1,3,2,6,7,5,4,3,2,4,2,3,1,4 Find the page faults for 3 frames using FIFO and LRU page replacement algorithms.	10
9	Calculate the hit and missing various page replacement policy LRU,OPT,FIFO for following sequence (page,The size 3) 2,3,3,1,5,2,4,5,3,2,5,2. State which one is best for above example?	

10	Write short notes on L 1, L2 and L3 Cache memory.	5
11	Explain in details Cache Coherency.	7
12	Describe what are the features of cache design	10
13	Explain various high speed memories such as interleaved memories and caches.	5
14	Write short notes on interleaved and associated memory	10