Module-3

5 a. What is arbitration? Explain different types of arbitration. (08 Marks)

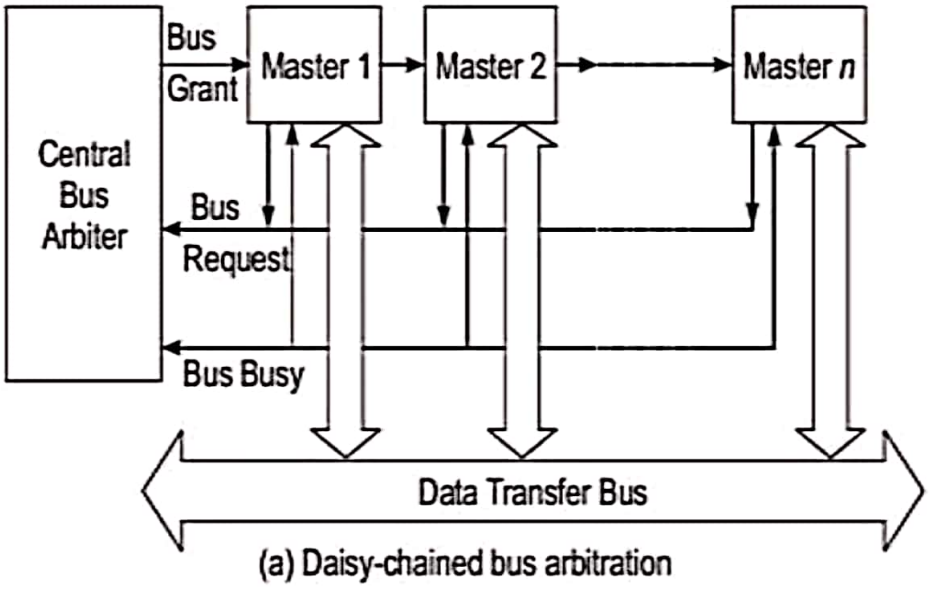
b. Explain sequential and weak consistency models. (08 Marks)

Bus Arbitration and Control

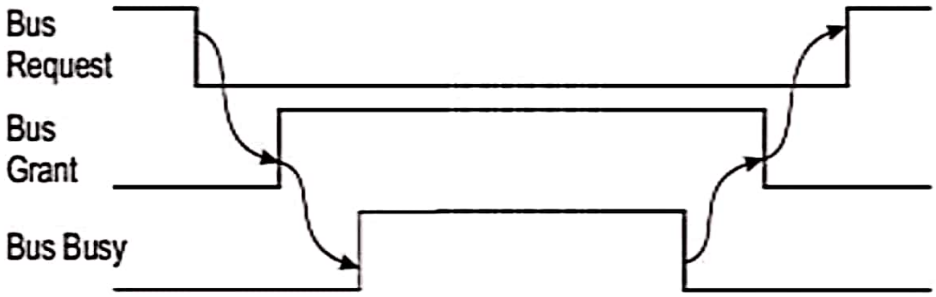
- The process of assigning control of the DTB to a requester is called arbitration. Dedicated lines are reserved to coordinate the arbitration process among several requesters.
- The requester is called a master, and the receiving end is called a slave.
- Interrupt lines are used to handle interrupts, which are often prioritized. Dedicated lines may be used to synchronize parallel activities among the processor modules.
- Utility lines include signals that provide periodic timing (clocking) and coordinate the power-up and power-down sequences of the system.
- The backplane is made of signal lines and connectors.
- A special bus controller board is used to house the backplane control logic, such as the system clock driver, arbiter, bus timer, and power driver.

Central Arbitration

- Uses a central arbiter as shown in Fig 5.4a
- Potential masters are daisy chained in a cascade
- A special signal line propagates *bus-grant* from first master (at slot 1) to the last master (at slot n).
- All requests share the same *bus-request* line
- The *bus-request* signals the rise of the *bus-grant* level, which in turn raises the *bus-busy* level as shown in Fig. 5.4b.



(a) Daisy-chained bus arbitration



(b) Bus transaction timing

Fig. 5.4 Central bus arbitration using shared requests and daisy-chained bus grants with a fixed priority

Distributed Arbitration

- Each master has its own arbiter and unique arbitration number as shown in Fig. 5.5b.
- Uses arbitration number to resolve arbitration competition
- When two or more devices compete for the bus, the winner is the one whose arbitration number is the largest determined by Parallel Contention Arbitration..
- All potential masters can send their arbitration number to shared-bus request/grant (SBRG) lines and compare its own number with SBRG number.
- If the SBRG number is greater, the requester is dismissed. At the end, the winner's arbitration number remains on the arbitration bus. After the current bus transaction is completed, the winner seizes control of the bus.
- Priority based scheme

<1b>

5.4.2 Sequential Consistency Model

- **Sufficient conditions:**
  1. Before a *load* is allowed to perform wrt any other processor, all previous *loads* must be globally performed and all previous *stores* must be performed wrt all processors
  2. Before a *store* is allowed to perform wrt any other processor, all previous *loads* must be globally performed and all previous *stores* must be performed wrt to all processors

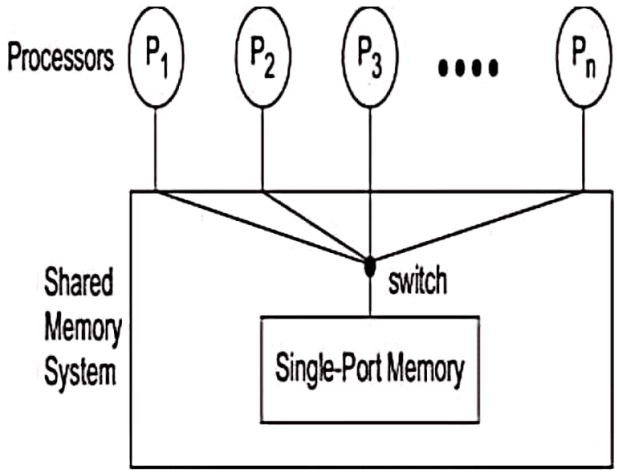


Fig. 5.20 Sequential consistency memory model (Courtesy of Sindhu, Frailong, and Cekleov; reprinted with permission from *Scalable Shared-Memory Multiprocessors*, Kluwer Academic Publishers, 1992)

5.4.3 Weak Consistency Models

- Multiprocessor model may range from strong (sequential) consistency to various degrees of weak consistency
- Two models considered
  - DSB (Dubois, Scheurich and Briggs) model
  - TSO (Total Store Order) model

DSB Model

Dubois, Scheurich and Briggs have derived a weak consistency model by relating memory request ordering to synchronization points in the program. We call this the DSB model specified by the following

3 conditions:

1. All previous *synchronization* accesses must be performed, before a *load* or a *store* access is allowed to perform wrt any other processor.
2. All previous *load and store* accesses must be performed, before a *synchronization* access is allowed to perform wrt any other processor.
3. *Synchronization* accesses sequentially consistent with respect to one another

TSO Model

Sindhu, Frailong and Cekleov have specified the TSO weak consistency model with 6 behavioral axioms.

Explain two types of mapping techniques

Direct Mapping Cache

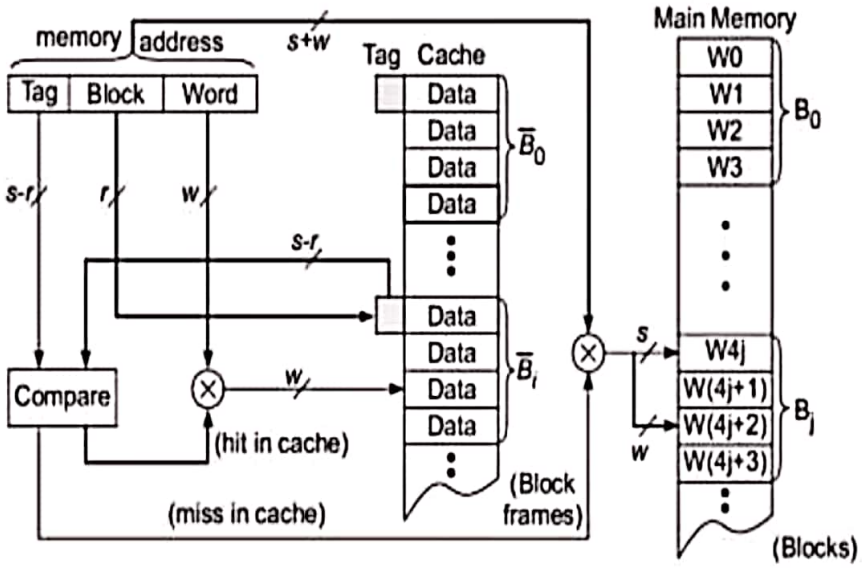
- Direct mapping of $\mathbf{n/m = 2^{s-r}}$ memory blocks to one block frame in the cache
- Placement is by using modulo-m function. Block B_j is mapped to block frame $\underline{B_i}$

$$B_j \rightarrow \underline{B_i} \qquad \text{if } i=j \bmod m$$

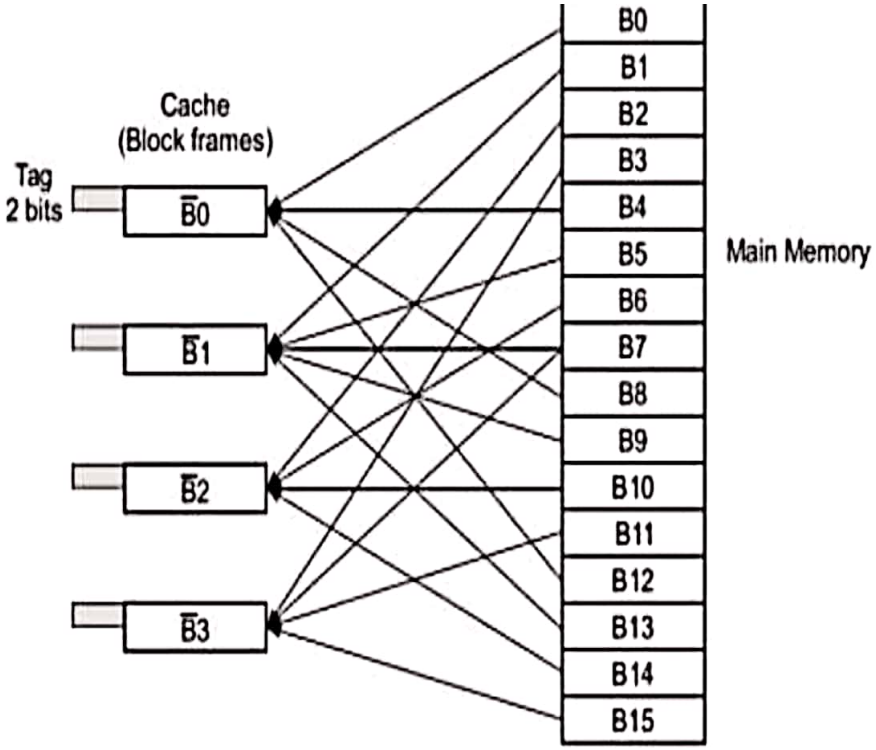
- There is a unique block frame $\underline{B_i}$ that each B_j can load into.
- There is no way to implement a block replacement policy.
- This Direct mapping is very rigid but is the simplest cache organization to implement.

The memory address is divided into 3 fields:

- The lower w bits specify the word offset within each block.
- The upper s bits specify the block address in main memory
- The leftmost (s-r) bits specify the tag to be matched



(a) The cache/memory addressing



(b) Block B_j can be mapped to block frame $\overline{B}_i$ if $i = j$ (modulo 4)

Fig. 5.9 Direct-mapping cache organization and a mapping example

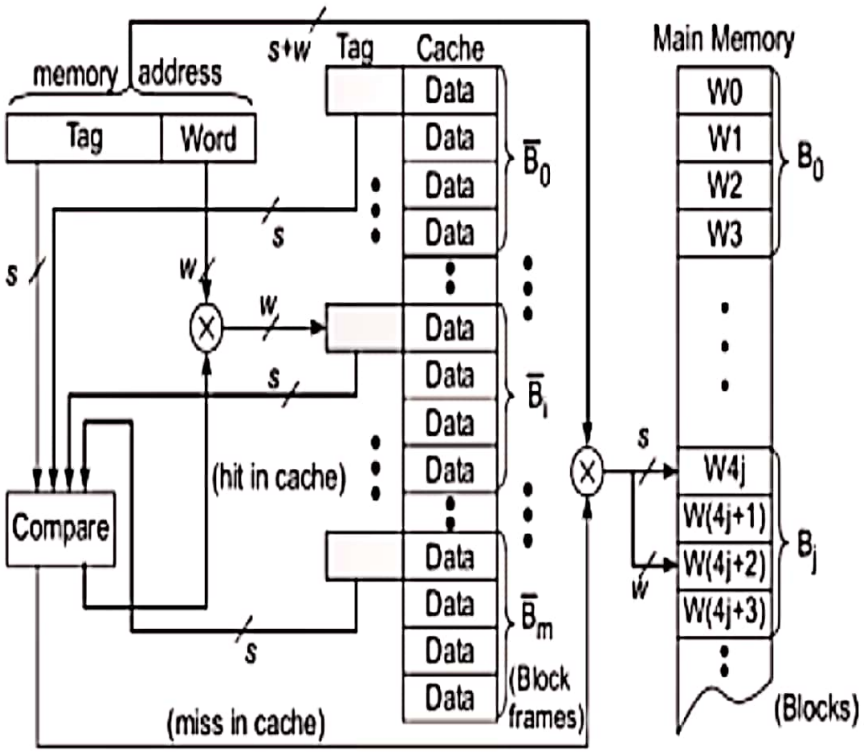
The block field (r bits) is used to implement the (modulo-m) placement, where $\mathbf{m=2^r}$

Once the block $\mathbf{B_i}$ is uniquely identified by this field, the tag associated with the addressed block is compared with the tag in the memory address.

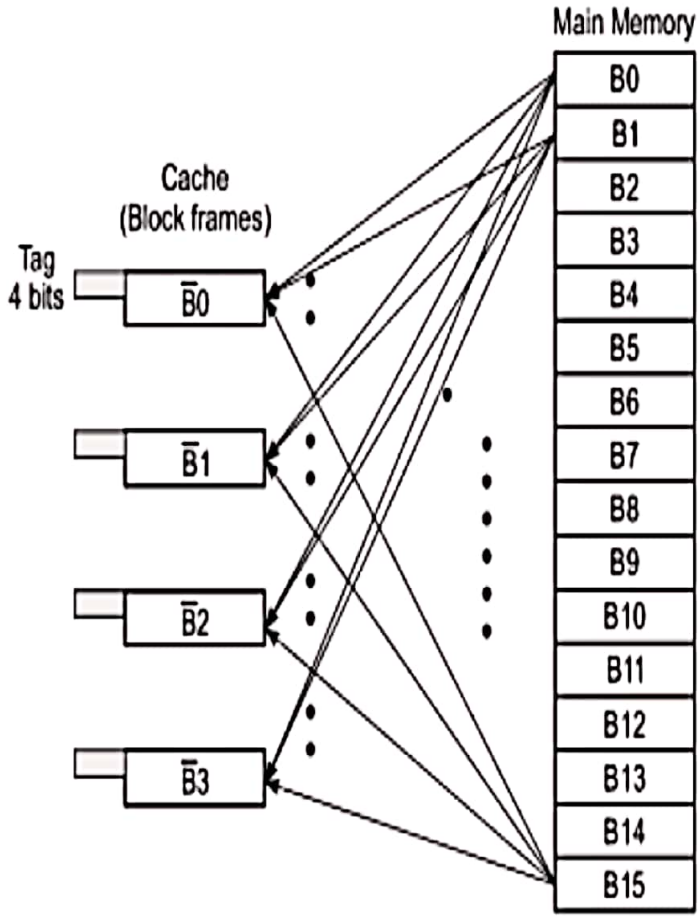
- **Advantages**
  - Simple hardware
  - No associative search
  - No page replacement policy
  - Lower cost
  - Higher speed
- **Disadvantages**
  - Rigid mapping
  - Poorer hit ratio
  - Prohibits parallel virtual address translation
  - Use larger cache size with more block frames to avoid contention

Fully Associative Cache

- Each block in main memory can be placed in any of the available block frames as shown in Fig. 5.10a.
- Because of this flexibility, an s-bit tag needed in each cache block.
- As $s > r$, this represents a significant increase in tag length.
- The name fully associative cache is derived from the fact that an m-way associative search requires tag to be compared with all block tags in the cache. This scheme offers the greatest flexibility in implementing block replacement policies for a higher hit ratio.
- An m-way comparison of all tags is very time consuming if the tags are compared sequentially using RAMs. Thus an associative memory is needed to achieve a parallel comparison with all tags simultaneously.
- This demands higher implementation cost for the cache. Therefore, a Fully Associative Cache has been implemented only in moderate size.
- Fig. 5.10b shows a four-way mapping example using a fully associative search. The tag is 4-bits long because 16 possible cache blocks can be destined for the same block frame.



(a) Associative search with all block tags



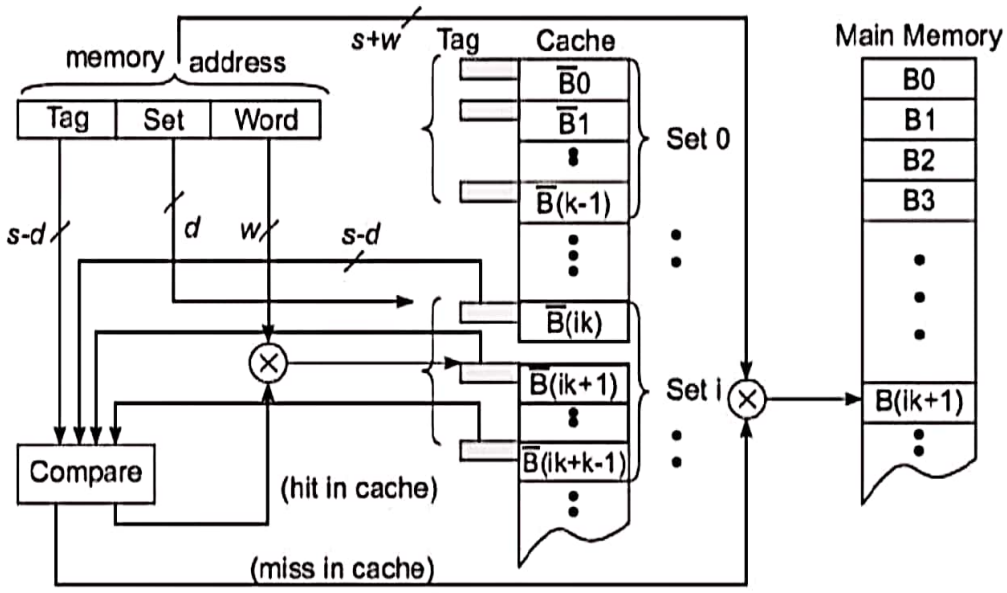
(b) Every block is mapped to any of the four block frames identified by the tag

Fig. 5.10 Fully associative cache organization and a mapping example

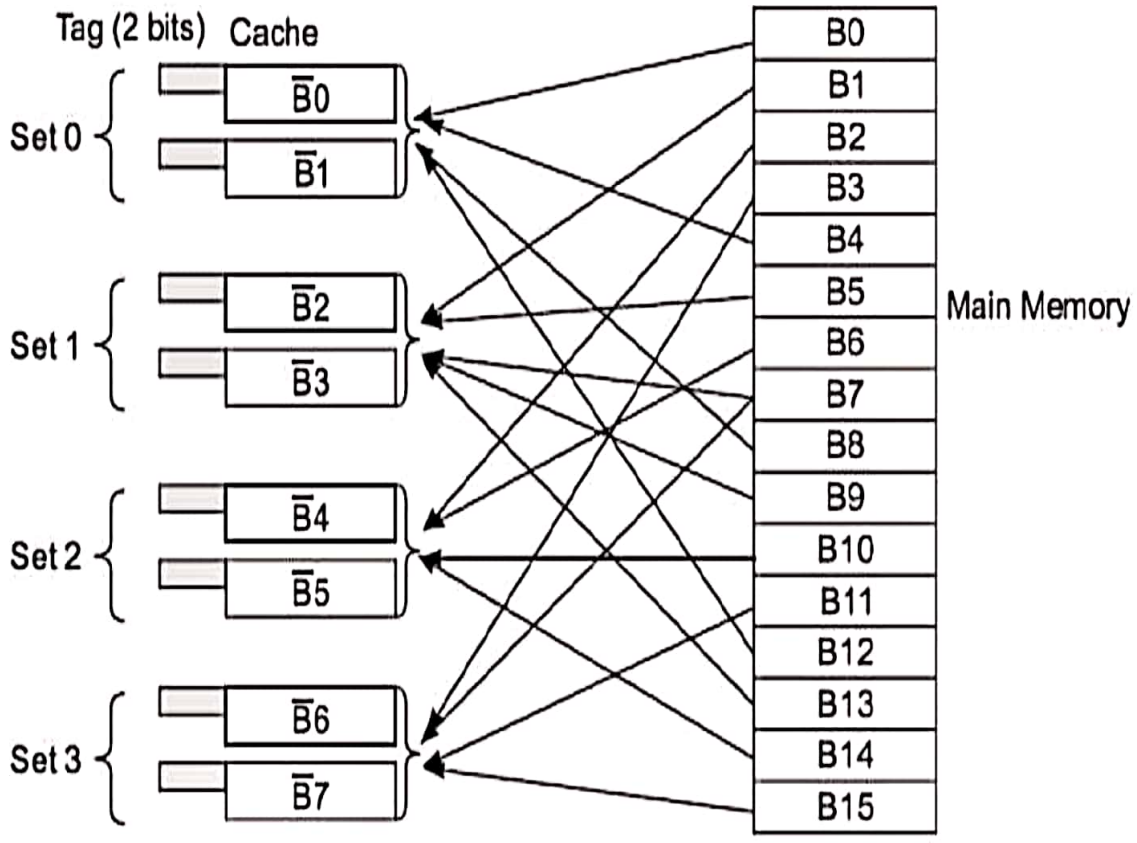
- **Advantages:**
  - Offers most flexibility in mapping cache blocks
  - Higher hit ratio
  - Allows better block replacement policy with reduced block contention
- **Disadvantages:**
  - Higher hardware cost
  - Only moderate size cache
  - Expensive search process

Set Associative Caches

- In a *k*-way associative cache, the *m* cache block frames are divided into **v=m/k** sets, with *k* blocks per set
- Each set is identified by a **d**-bit set number, where $\mathbf{2^d = v}$.
- The cache block tags are now reduced to **s-d** bits.
- In practice, the set size k, or associativity, is chosen as 2, 4, 8, 16 or 64 depending on a tradeoff among block size w, cache size m and other performance/cost factors.



(a) A *k*-way associative search within each set of *k* each blocks



(b) Mapping cache blocks in a two-way associative cache wit four sets

Fig. 5.11 Set-associative cache organization and a two-way associative mapping example

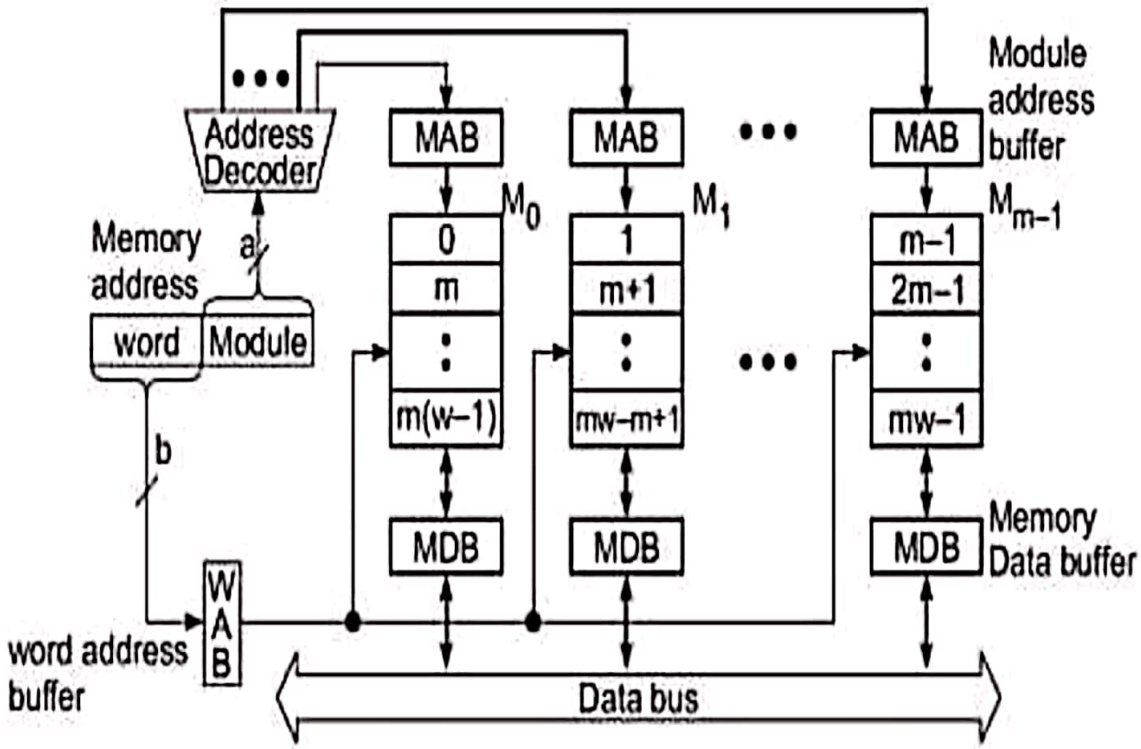
- Compare the tag with the *k* tags within the identified set as shown in Fig 5.11a.
- Since k is rather small in practice, the k-way associative search is much more economical than the full associativity.
- In general, a block B_j can be mapped into any one of the available frames $\underline{B_f}$ in a set S_i defined below.

$$B_j \rightarrow \underline{B_f} \in S_i \qquad \text{if } j(\text{mod } v) = i$$

- The matched tag identifies the current block which resides in the frame.

OR

6 a. Explain the following terms associated with cache and memory architecture:
(i) Low order memory interleaving
(ii) Atomic v/s non-atomic memory
(iii) Physical address cache vs virtual address cache
(iv) Memory bandwidth and fault tolerance.



(a) Low-order *m*-way interleaving (the C-access memory scheme)

Low-order interleaving

- Low-order interleaving spreads contiguous memory locations across the **m** modules horizontally (Fig. 5.15a).
- This implies that the low-order **a** bits of the memory address are used to identify the memory module.
- The high-order **b** bits are the word addresses (displacement) within each module.
- Note that the same word address is applied to all memory modules simultaneously. A module address decoder is used to distribute module addresses.

Atomic memory accesses: memory updates are known to all processors at the same time

Non-atomic: having individual program orders that conform is not a sufficient condition for sequential consistency

- Multiprocessor cannot be strongly ordered

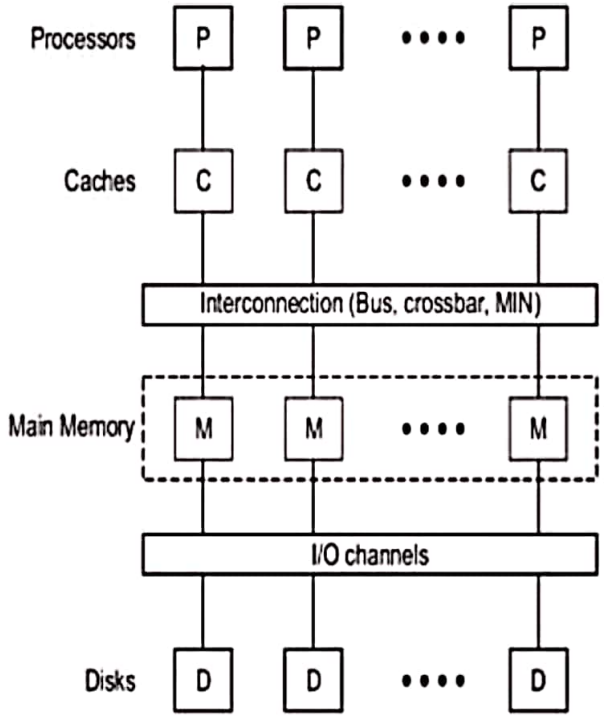


Fig. 5.6 A memory hierarchy for a shared-memory multiprocessor

Physical address cache

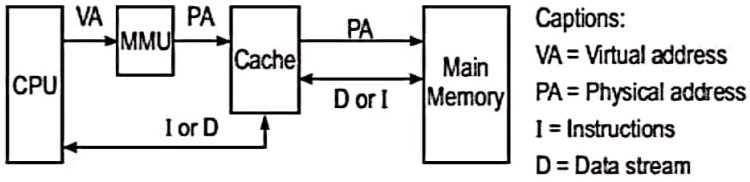
- When cache is addressed by physical address it is called physical address cache. The cache is indexed and tagged with physical address.
- Cache lookup must occur after address translation in TLB or MMU. No aliasing is allowed so that the address is always uniquely translated without confusion.
- After cache miss, load a block from main memory
- Use either write-back or write-through policy

Advantages:

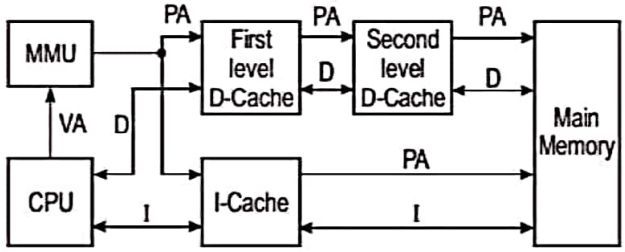
- No cache flushing on a context switch
- No aliasing problem thus fewer cache bugs in OS kernel.
- Simplistic design
- Requires little intervention from OS kernel

Disadvantages:

Slowdown in accessing the cache until the MMU/TLB finishes translating the address



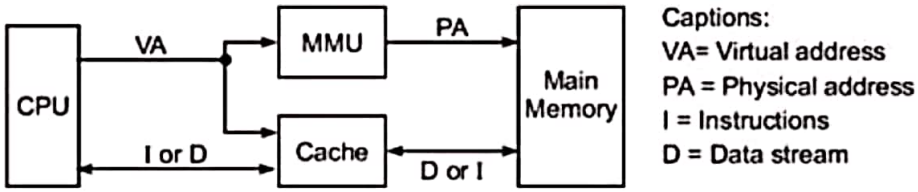
(a) A unified cache accessed by physical address



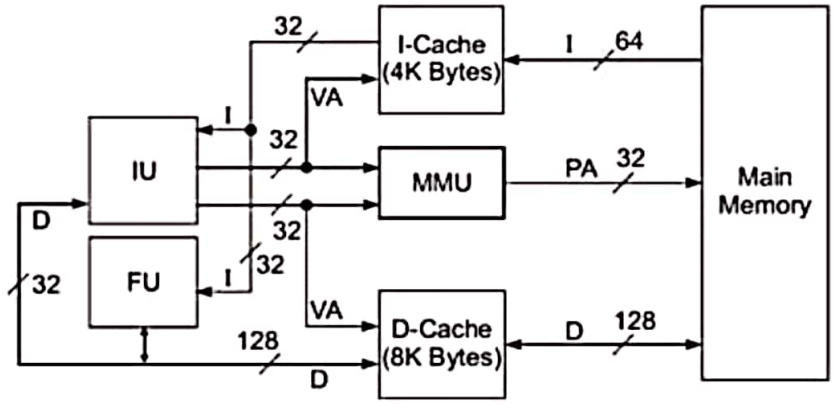
(b) Split caches accessed by physical address in the Silicon Graphics workstation

Fig. 5.7 Physical address models for unified and split caches

Virtual Address caches



(a) A unified cache accessed by virtual address



(b) A split cache accessed by virtual address as in the Intel i860 processor

Fig. 5.8 Virtual address models for unified and split. caches (Courtesy of Intel Corporation, 1989)

- When a cache is indexed or tagged with virtual address it is called virtual address cache.
- In this model both cache and MMU translation or validation are done in parallel.
- The physical address generated by the MMU can be saved in tags for later write back but is not used during the cache lookup operations.

Advantages:

- do address translation only on a cache miss
- faster for hits because no address translation
- More efficient access to cache

Disadvantages:

- Cache flushing on a context switch (example : local data segments will get an erroneous hit for virtual addresses already cached after changing virtual address space, if no cache flushing).
- Aliasing problem (several different virtual addresses cannot span the same physical addresses without being duplicated in cache).

Memory Bandwidth

The memory bandwidth B of an m-way interleaved memory is upper-bounded by m and lower-bounded by *I*. The Hellerman estimate of *B* is

$$B = m^{0.56} \sim \sqrt{m} \qquad (5.5)$$

where m is the number of interleaved memory modules.

- This equation implies that if 16 memory modules are used, then the effective memory bandwidth is approximately four times that of a single module.
- This pessimistic estimate is due to the fact that block access of various lengths and access of single words are randomly mixed in user programs.
- Hellerman's estimate was based on a single-processor system. If memory-access conflicts from multiple processors (such as the hot spot problem) are considered, the effective memory bandwidth will be further reduced.
- In a vector processing computer, the access time of a long vector with n elements and stride distance 1 has been estimated by Cragon (1992) as follows:
- It is assumed that the *n* elements are stored in contiguous memory locations in an m-way interleaved memory system.

The average time t_1 required to access one element in a vector is estimated by

$$t_1 = \frac{\theta}{m}\left(1 + \frac{m-1}{n}\right) \qquad (5.6)$$

When $n \rightarrow \infty$ (very long vector), $t_1 \rightarrow \theta/m = \tau$.

As $n \rightarrow 1$ (scalar access), $t_1 \rightarrow \theta$.

Equation 5.6 conveys the message that interleaved memory appeals to pipelined access of long vectors; the longer the better.

Fault Tolerance

- High- and low-order interleaving can be combined to yield many different interleaved memory organizations.
- Sequential addresses are assigned in the high-order interleaved memory in each memory module.
- This makes it easier to isolate faulty memory modules in a *memory bank* of m memory modules.
- When one module failure is detected, the remaining modules can still bo used by opening a window in the address space.
- This fault isolation cannot be carried out in a low-order interleaved memory, in which a module failure may paralyze the entire memory bank.
- Thus low-order interleaving memory is not fault-tolerant.

Explain with daigram,the backplane bus specification

5.1.1 Backplane Bus Specification

- A backplane bus interconnects processors, data storage and peripheral devices in a tightly coupled hardware.
- The system bus must be designed to allow communication between devices on the devices on the bus without disturbing the internal activities of all the devices attached to the bus.
- Timing protocols must be established to arbitrate among multiple requests. Operational rules must be set to ensure orderly data transfers on the bus.
- Signal lines on the backplane are often functionally grouped into several buses as shown in Fig 5.1. Various functional boards are plugged into slots on the backplane. Each slot is provided with one or more connectors for inserting the boards as demonstrated by the vertical arrows.

Data Transfer Bus (DTB)

- Data address and control lines form the data transfer bus (DTB) in VME bus.
- Address lines broadcast data and device address
  - Proportional to log of address space size
- Data lines proportional to memory word length
- Control lines specify read/write, timing, and bus error conditions

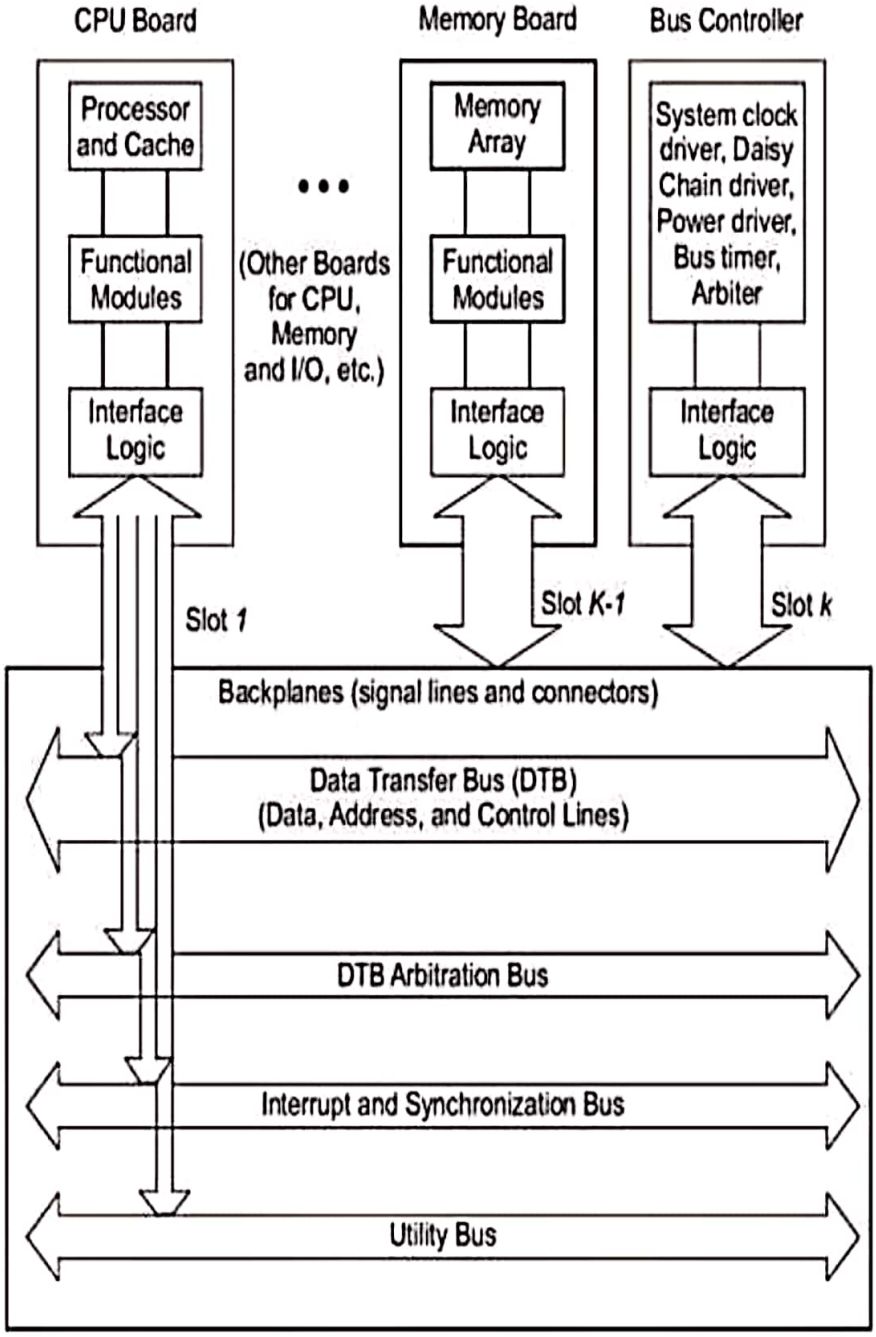


Fig. 5.1 Backplane buses, system interfaces, and slot connections to various functional boards in a multiprocessor system