FUTURE VISION BIE

One Stop for All Study Materials

& Lab Programs

Future Vision

By K B Hemanth Raj

Scan the QR Code to Visit the Web Page

Or

Visit : https://hemanthrajhemu.github.io

Gain Access to All Study Materials according to VTU,
CSE – Computer Science Engineering,
ISE – Information Science Engineering,
ECE - Electronics and Communication Engineering
& MORE...

Join Telegram to get Instant Updates: https://bit.ly/VTU_TELEGRAM

Contact: MAIL: futurevisionbie@gmail.com

INSTAGRAM: www.instagram.com/hemanthraj_hemu/

INSTAGRAM: www.instagram.com/futurevisionbie/

WHATSAPP SHARE: https://bit.ly/FVBIESHARE

Advanced Computer Architecture

17CS72

MODULE-3

Bus, Cache, and Shared Memory

Book: "Advanced Computer Architecture – Parallelism, Scalability, Programmability", Hwang & Jotwani

https://hemanthrajhemu.github.io

MODULE-3

Syllabus:

MODULE	Bus, Cache, and Shared Memory:	
3	**Bus and Memory:** Bus, Cache	1
	Shared Memory	1
	Bus Systems ,Cache Memory Organizations	2
	Shared Memory Organizations	1
	Sequential and Weak Consistency Models	1
	Pipelining and Superscalar Techniques	1
	Linear Pipeline Processors	1
	Nonlinear Pipeline Processors	1
	Instruction Pipeline Design ,Arithmetic Pipeline Design	1

https://hemanthrajhemu.github.io

Bus, Cache, and Shared Memory

https://hemanthrajhemu.github.io

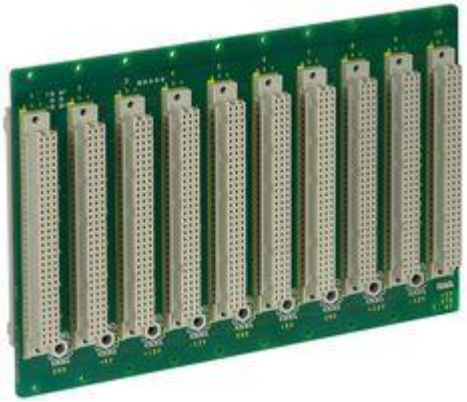
Bus Systems

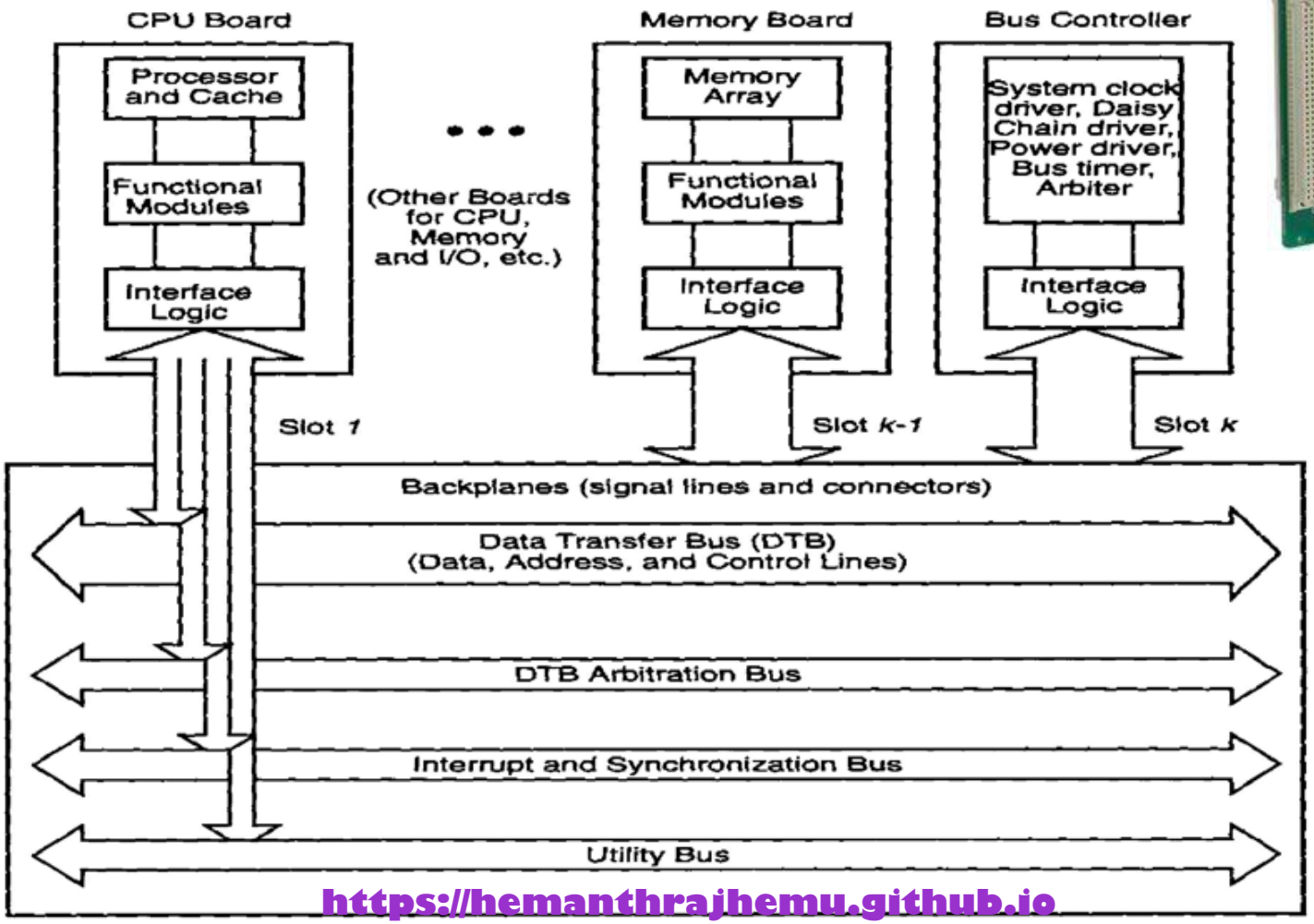
- System bus operates on contention basis
- Only one is granted access to bus at a time
- *Effective bandwidth* available is inversely proportional to # of contending processors
- Hence, simple and low cost (4 – 16 processors)
- *Backplane Bus Specification*
  - Interconnects processors, data storage, and I/O devices
  - Must allow communication b/t devices without interfering working process.
  - Timing protocols for arbitration
  - Operational rules for orderly data transfers
  - Signal lines grouped into several buses

https://hemanthrajhemu.github.io

BUS Systems

Backplane Multiprocessor System





https://hemanthrajhemu.github.io

BUS Systems
Backplane Multiprocessor System

- *Data Transfer Bus (DTB)*
  - DTB composed of data, address, and control lines
  - Address lines broadcast data and device address
    - Proportional to log of address space size
    - Address modifier lines used for special addressing modes.
  - Data lines proportional to memory word length
  - Control lines specify read/write, timing, and bus error conditions

BUS Systems
Backplane Multiprocessor System

- *Bus Arbitration and Control*
  - Arbitration: assign control of bus to a requester
  - Requester: *master* Receiving end: *slave*
  - Interrupt lines for prioritized interrupts
  - Dedicated lines for synchronizing parallel activities among processor modules
  - Utility lines provide periodic timing and coordinate the power-up/down sequences
  - Bus controller board houses control logic

BUS Systems
Backplane Multiprocessor System

- *Functional Modules:*
  - Collection of electronic circuits to implement special bus control functions
  - Arbiter: functional module that performs arbitration
  - Bus timer: measures time for data transfers and cancels if its too long
  - Interrupter: generates interrupt request and provides status/ID to interrupt handler
  - Location monitor: monitors data transfer
  - Power monitor: monitors power source and signals when power becomes unstable.
  - System clock driver: provides clock timing signal on the utility bus
  - Board interface logic: matches signal line impedance, propagation time, and termination values

BUS Systems
Backplane Multiprocessor System

- *Physical Limitations:*
  - Electrical, mechanical, and packaging limitations restrict # of boards
  - Can mount multiple backplane buses on the same backplane chassis
  - Difficult to scale a bus system due to packaging constraints

BUS Systems
Addressing and Timing Protocols

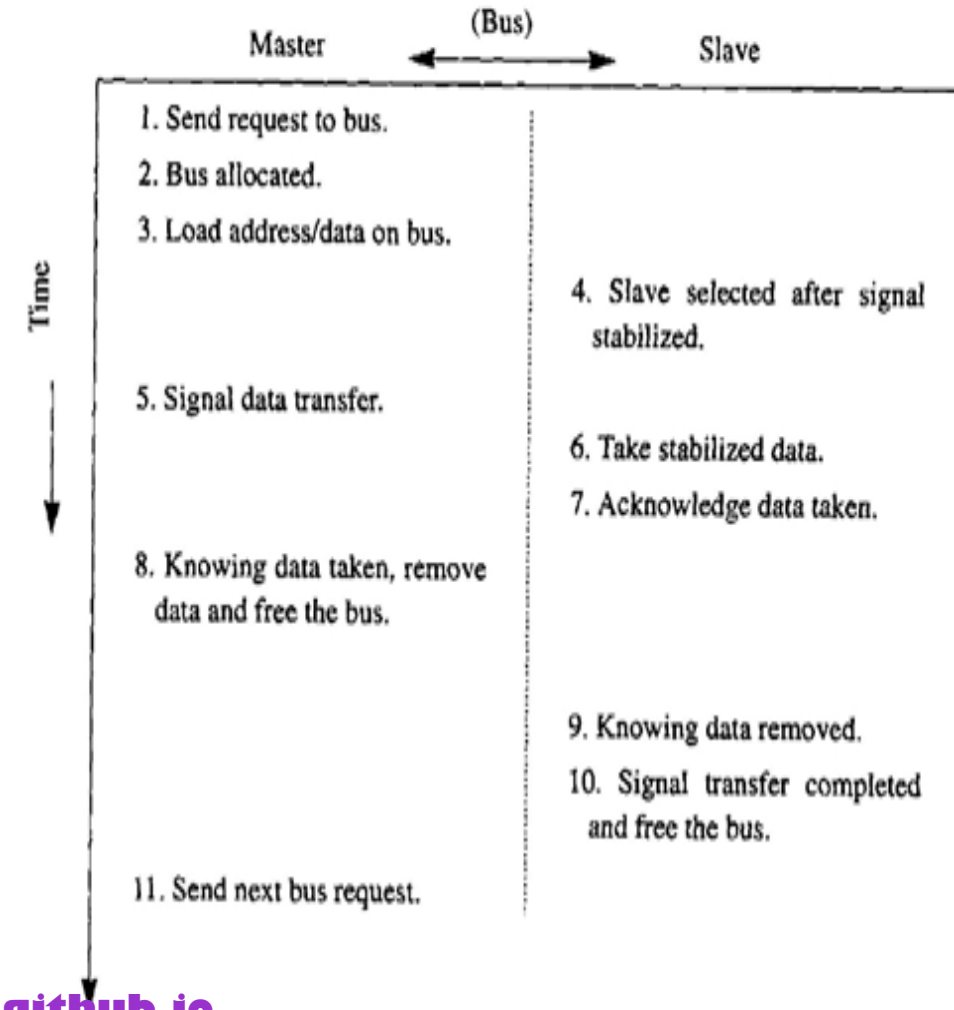
- Two types of printed circuit boards connected to a bus: *active* and *passive*
- The master can initiate a bus cycle (like processor)
  - Only one can be in control at a time
- The slaves respond to requests by a master (like memory)
  - Multiple slaves can respond

BUS Systems
Addressing and Timing Protocols

- *Bus Addressing*
  - The backplane bus is driven by a digital clock with a fixed cycle time: *bus cycle*
  - Factors affecting bus delay:
    - Source's line drivers, destination's receivers, slot capacitance, line length, and # boards attached etc.
  - To optimize performance bus should reduce cycles on request handling, arbitration, addressing and interrupts
  - Each device is identified by device number.
  - When device # matches contents of high-order address lines, the board is selected as a slave

BUS Systems
Addressing and Timing Protocols

- *Broadcall and Broadcast*
  - Most bus transactions have one slave/master
  - Broadcall: read operation where multiple slaves place data on bus
    - detects multiple interrupt sources
    - Special AND / OR operations are used.
  - Broadcast: write operation involving multiple slaves
    - Implements multi-cache coherence on the bus
  - Synchronizing Timing Protocol

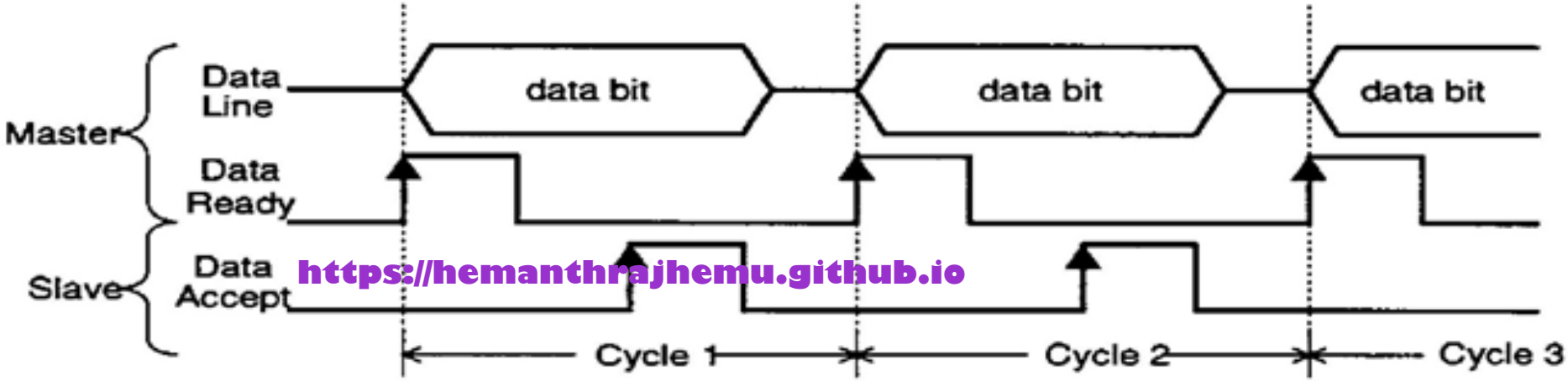


BUS Systems

Addressing and Timing Protocols

Synchronous Timing:

- All bus transaction steps take place at fixed clock edges
- Clock cycle time determined by slowest device on bus (suitable for devices with same speed)
- Data-ready pulse (master) initiates transfer
- Data-accept (slave) signals completion

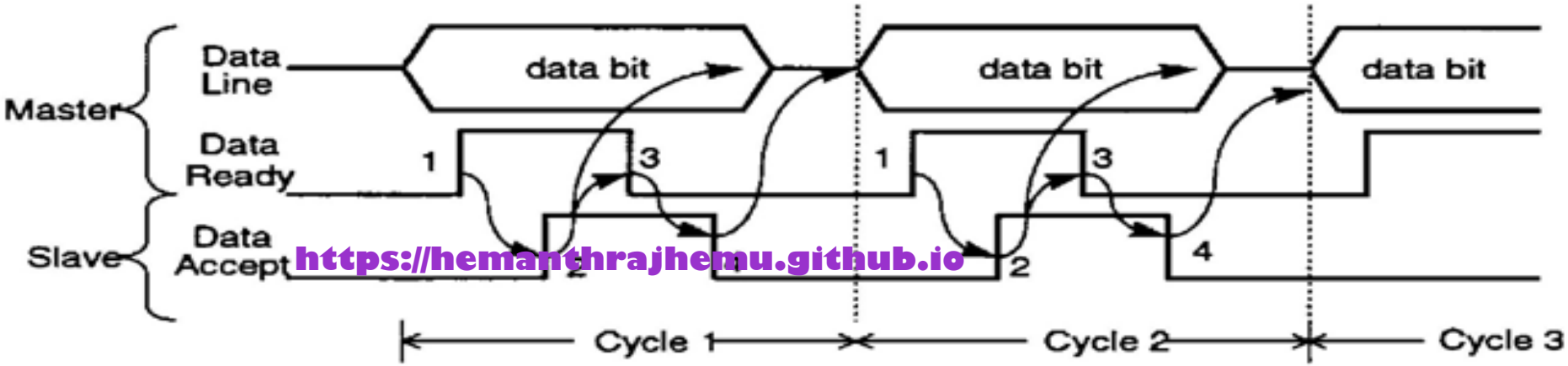


BUS Systems

Addressing and Timing Protocols

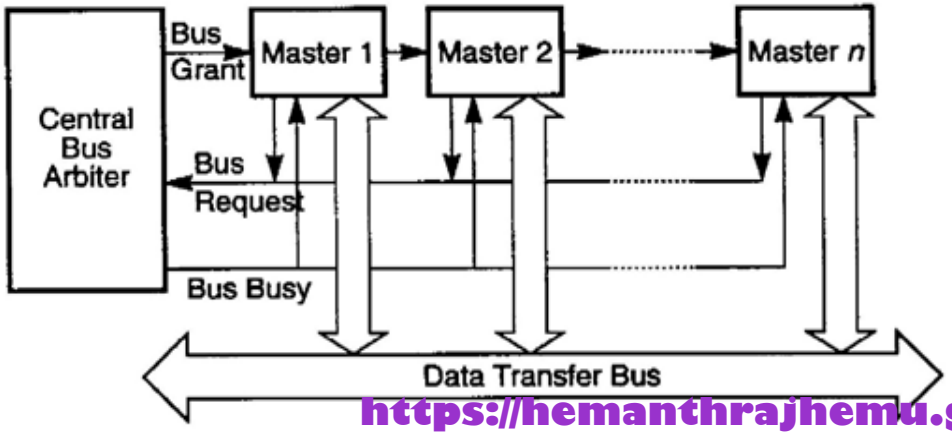
Asynchronous Timing:

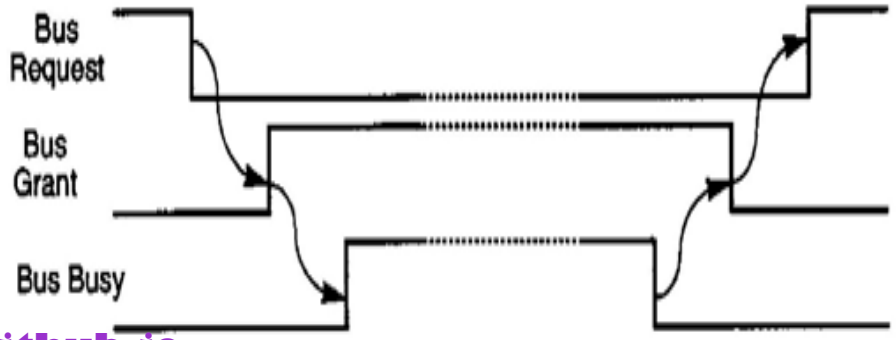
- Based on handshaking or interlocking
- Provides freedom of variable length clock signals for different speed devices
- The (1) raising edge of *data ready* signal from signal triggers (2) *data accept* signal from slave.
- Falling of *data ready* clock (3) removes data from bus and (4) triggers trailing edge of *data accept* clock.
- 4 - way hand shake process is repeated till all data is transferred.
- More complex and costly, but more flexible



BUS Systems
Arbitration, Transaction, and interrupt

- Process of selecting next bus master – *Arbitration*.
- *Bus tenure* is duration of control (to restrict)
- Arbitrate on a fairness or priority basis
- Arbitration competition and bus transactions take place concurrently on a parallel bus over separate lines

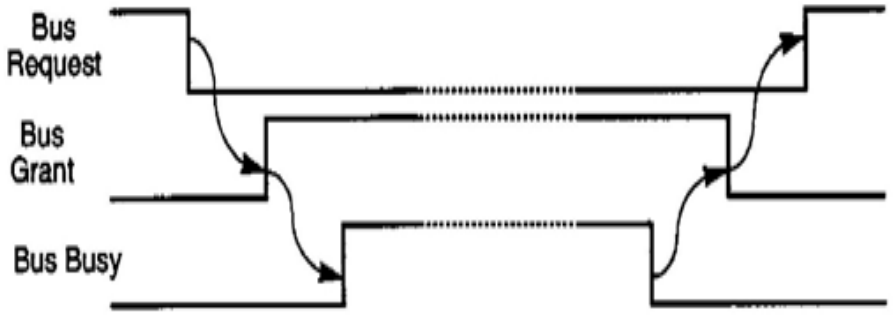


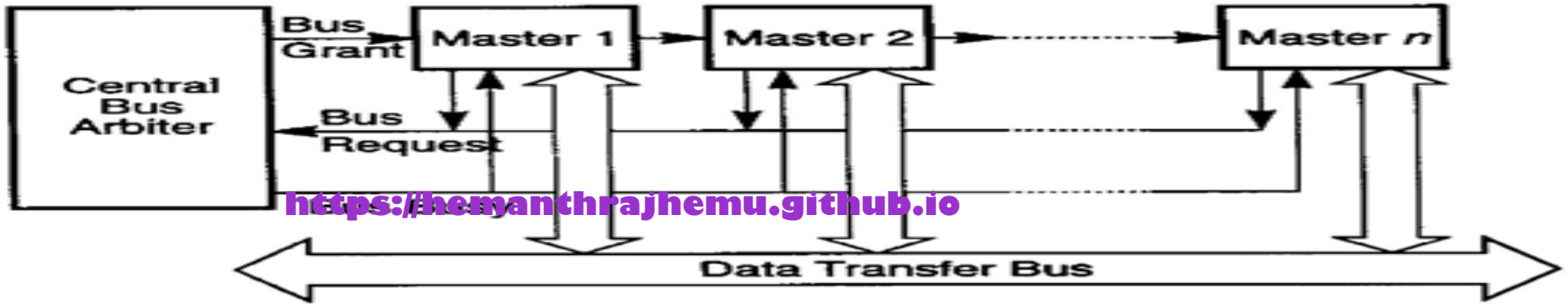


https://hemanthrajhemu.github.io

BUS Systems
Arbitration, Transaction, and interrupt

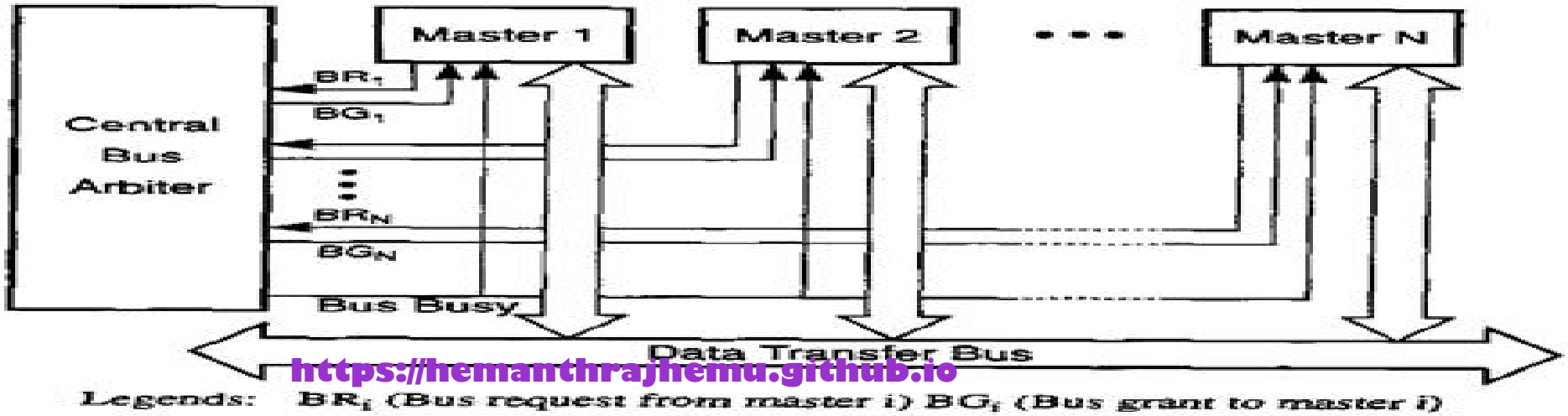
- ***Central Arbitration***
  - Potential masters are daisy chained
  - Signal line propagates *bus-grant* from first master to the last master
  - One *bus-request* line from master activates *bus-grant* line which activates *bus-busy* line.
  - Easy to add devices
  - Fixed-priority from left to right (not fair)
  - Propagation of bus-grant signal is slow
  - Not fault tolerant (if a device fails)





BUS Systems
Arbitration, Transaction, and interrupt

- ***Independent Requests and Grants***
  - Provide independent bus-request and grant signals for each master.
  - Require a central arbiter, but can use a priority or fairness based policy
  - More flexible and faster than a daisy-chained policy
  - Larger number of lines – costly

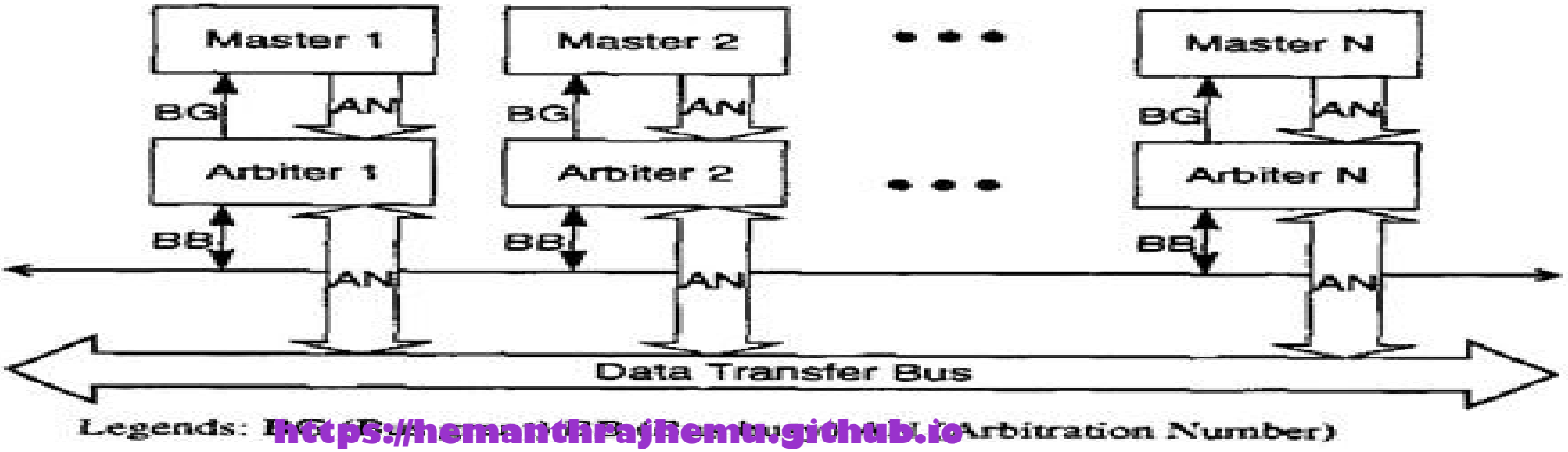


(a) Independent requests with a central arbiter

BUS Systems

Arbitration, Transaction, and interrupt

- ***Distributed Arbitration***
  - Each master has its own arbiter and unique arbitration number
  - Use Arbitration Number (AN) to resolve competition.
  - Send AN to shared - bus request / grant (**SBRG**) lines and compare own number with SBRG number



Legends: [illegible] Arbitration Number)

(b) Using distributed arbiters

BUS Systems
Arbitration, Transaction, and interrupt

- ***Transaction modes***
  - *Address-only* transfer: only address no data
  - *Compelled-data* transfer: address transfer followed by a block of one or more data transfers to one or more contiguous addresses.
  - *Packet-data* transfer: address transfer followed by a fixed-length block of data transfers from set of continuous address.
  - A bus transaction consist of request followed by response.
    - *Connected transaction*: carry out master's request and a slave's response in a single bus transaction
    - *Split*: splits request and response into separate bus transactions

https://hemanthrajhemu.github.io

BUS Systems
Arbitration, Transaction, and interrupt

- ***Interrupt Mechanisms***
  - *Interrupt:* request from I/O to a processor for service or attention
  - Priority interrupt bus sends interrupt signals
  - Interrupter provides status and ID info
  - Have an interrupt handler for each request line
  - Can use message passing on data lines
    - Save lines, but use cycles
    - Use of time-shared data bus lines is a *virtual-interrupt*

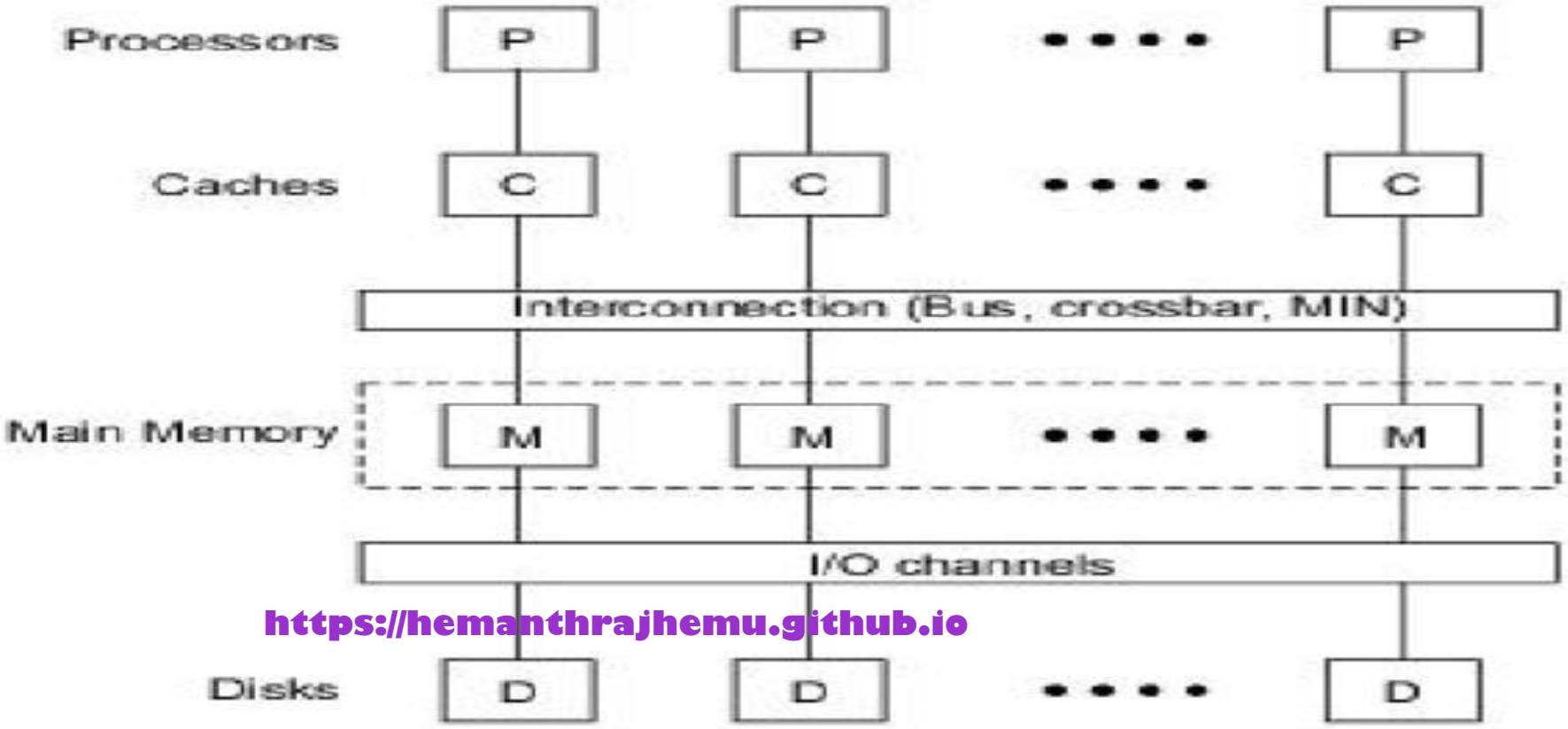
Futurebus+ Goals

- Open bus standard to support:
  - 64 bit address space
  - Throughput required by multi-RISC or future generations of multiprocessor architectures
- Expandable or scalable
- Independent of particular architectures and processor technologies
- *Standard Requirements:*
  - Independence for an open standard
  - Asynchronous timing protocol
  - Optional packet protocol
  - Distributed arbitration protocols
  - Support of high reliability and fault tolerant applications
  - Ability to lock modules w/o deadlock or livelock
  - Circuit-switched and split transaction protocols
  - Support of real-time mission critical computations w/multiple priority levels
  - 32 or 64 bit addressing
  - Direct support of snoopy cache-based procs
  - Compatible message passing protocols

- Signal Lines
  - Information (150 – 306)
  - Synchronization (7)
  - Bus arbitration (18)
  - Handshake (6)
- Information Lines
  - 64 address lines multiplexed with lower order 64 data lines
  - Data path can be up to 256 bits wide
  - Tag lines extend address/data modes (opt)
  - Command lines carry info from master
  - Status lines used by slaves to respond
  - Capability lines to declare special bus transactions
  - Parity check lines for protection
- Synchronization Lines
  - Coordinate exchange of address, command, capability status and data
  - Address/data handshake lines used by both master and slaves
  - Bus tenure line used to coordinate transfer of bus control

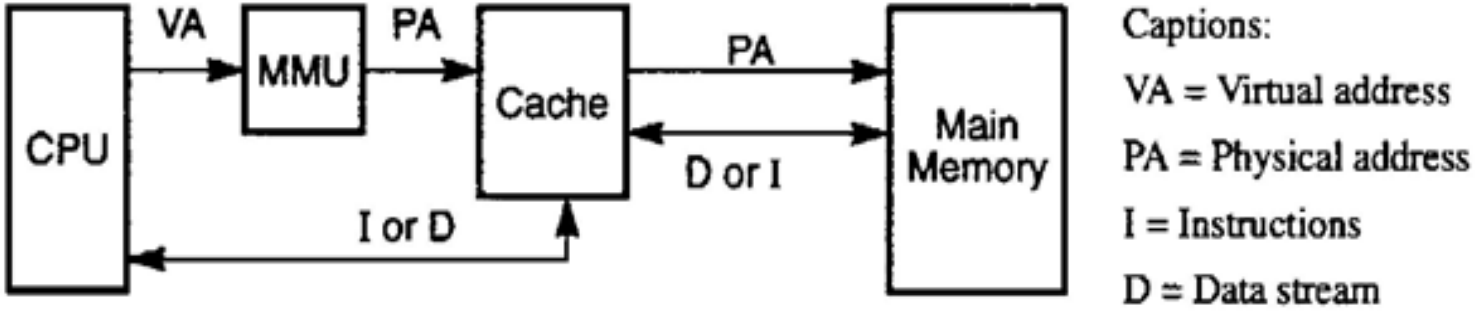
Cache Memory Organization

- Cache Addressing Models
- Most systems use private caches for each processor



Physical Address Caches

- Address to caches can be given using **physical address** or virtual address



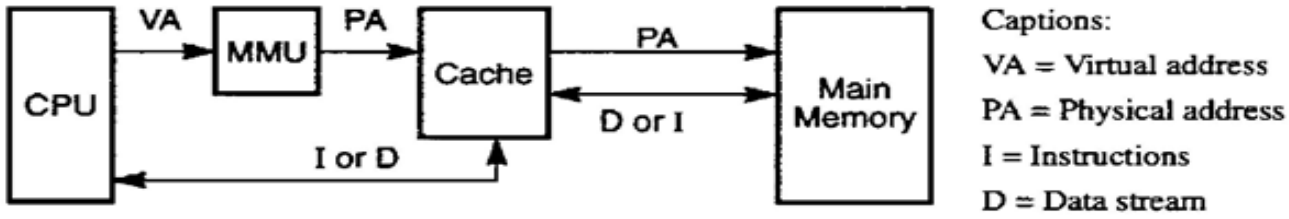
(a) A unified cache accessed by physical address

- Cache is indexed and tagged with the physical address
- Cache lookup occurs after address translation in TLB or MMU (no aliasing)

https://hemanthrajhemu.github.io

Physical Address Caches

- Address to caches can be given using physical address or virtual address



(a) A unified cache accessed by physical address

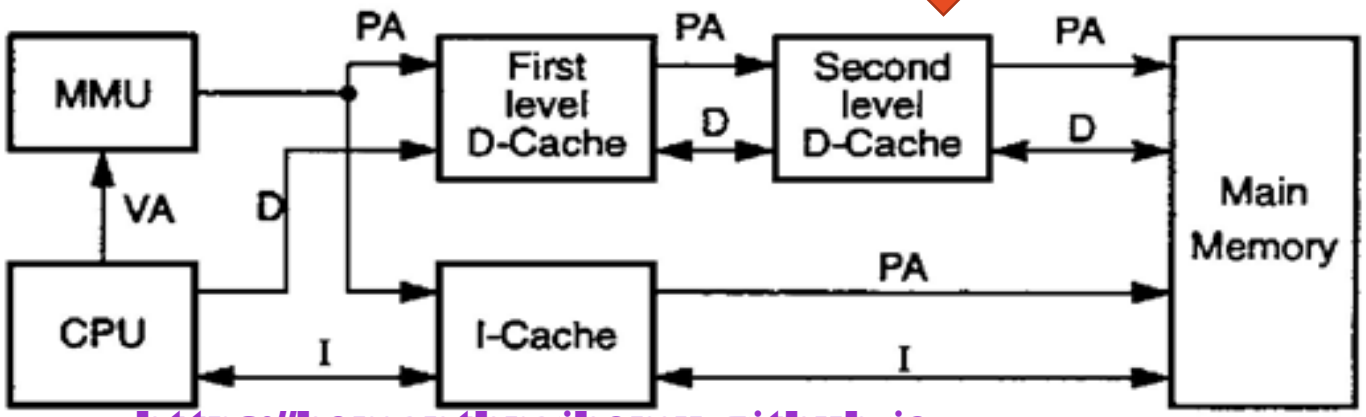
- Read:
  - HIT if data is present in cache
  - MISS if data is not in cache, load a block from main memory to cache
- Write:
  - write-back (data written to main memory immediately) (needs more cycles)
  - write-through policy (write to main memory is delayed until the cache block is replaced)

https://hemanthrajhemu.github.io

Physical Address Caches

- **Advantages:**
  - No aliasing problems
  - Simplistic design
  - Requires little intervention from OS kernel
- **Disadvantage**:
  - Slowdown in accessing cache until the MMU/TLB finishes translation

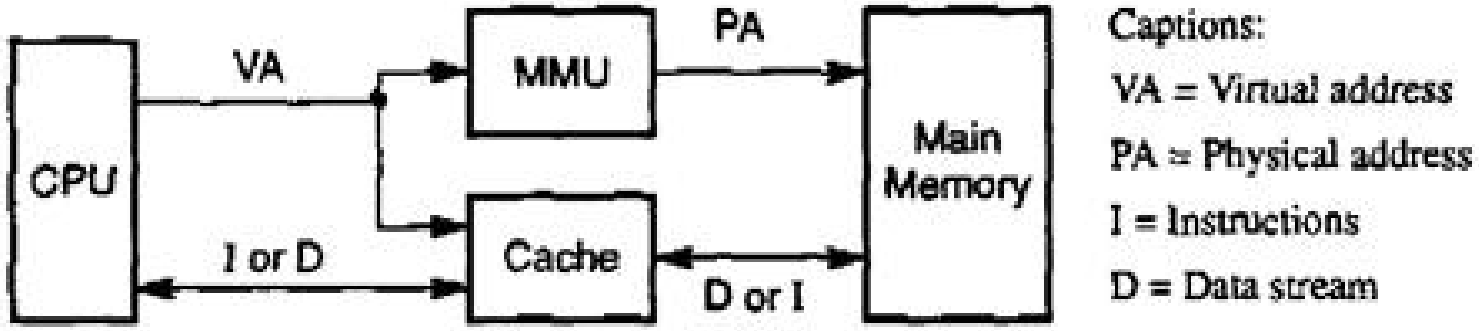
Example:



Split caches accessed by physical address in the Silicon Graphics workstation

Virtual Address Caches

- Cache indexed or tagged with virtual address
- Cache and MMU translation/validation performed in parallel
- Physical address saved in tags for write back
- More efficient as cache access is faster

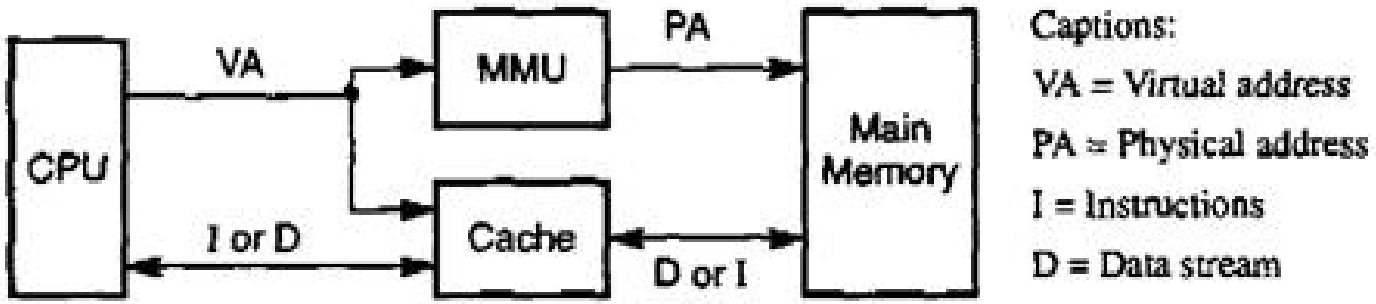


(a) A unified cache accessed by virtual address

https://hemanthrajhemu.github.io

Virtual Address Caches

- **Aliasing Problem**
  - Different logically addressed data have the same index/tag in the cache
  - Confusion if two or more processors access the same physical cache location
  - Flush cache when aliasing occurs, but leads to slowdown
  - Apply special tagging with physical address



(a) A unified cache accessed by virtual address

https://hemanthrajhemu.github.io

Block Placement Schemes

- Performance depends upon cache access patterns, organization, and management policy
- Blocks in caches are block frames ($\underline{B}_i$), and blocks in main memory (B_j)
- **$\underline{B}_i$ ($i \leq m$), B_j ($i \leq n$), $m<<n$, $n=2^s$, $m=2^r$**
- Each block has b words $b=2^w$, for cache total of $m.b$ words, main memory of $n.b$ words

https://hemanthrajhemu.github.io

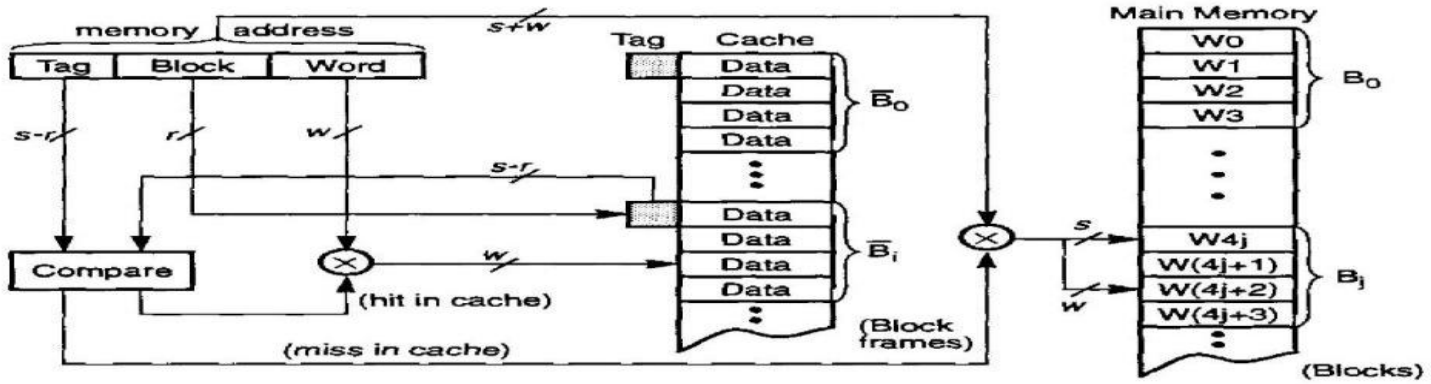
Direct Mapping Cache

- Direct mapping of *n/m* memory blocks to one block frame in the cache
- Placement is by using modulo-*m* function
- $B_j \rightarrow \underline{B_i}$ if $i = j \bmod m$
- Unique block frame $\underline{B_i}$ that each B_j loads to
- Simplest organization to implement

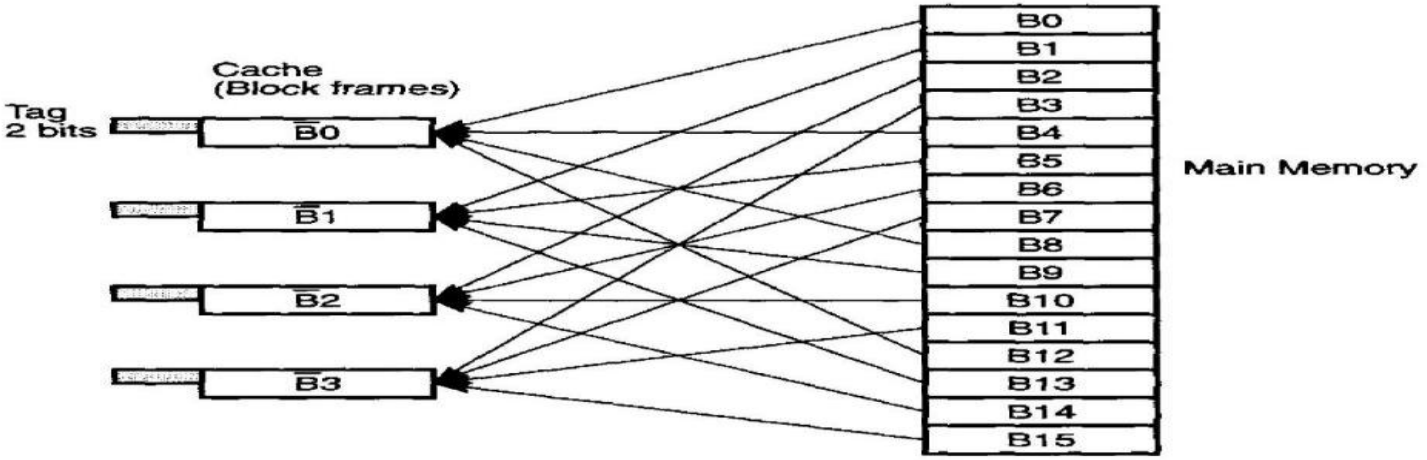
Direct Mapping Cache

- **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

Direct Mapping Cache



(a) The cache/memory addressing



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

Figure 5.10 Direct-mapping cache organization and a mapping example.

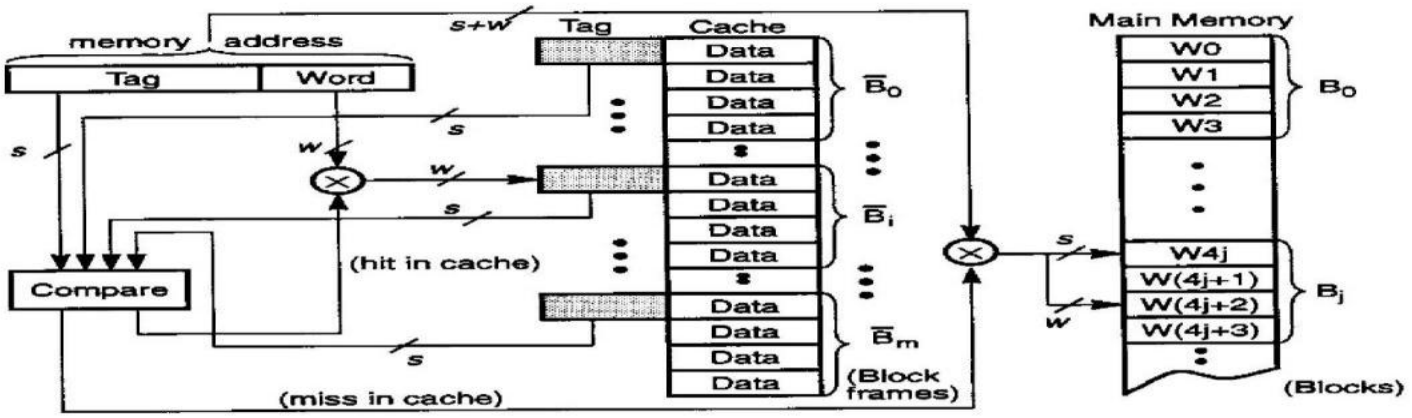
https://hemanthrajhemu.github.io

Fully Associative Cache

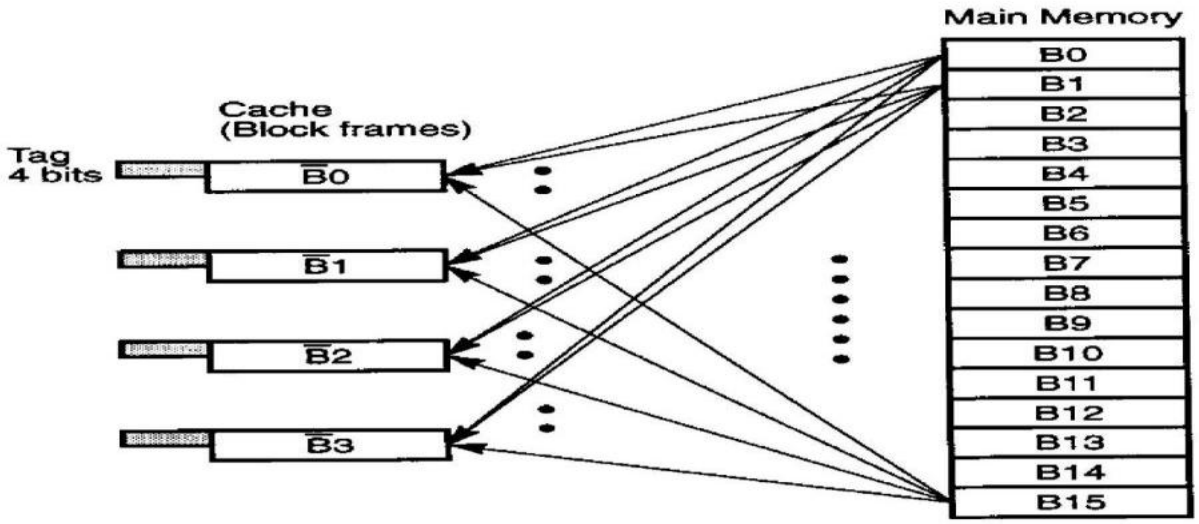
- Each block in main memory can be placed in any of the available block frames
- *s*-bit tag needed in each cache block ($s > r$)
- An *m*-way associative search requires the tag to be compared w/ all cache block tags
- Use an associative memory to achieve a parallel comparison w/all tags concurrently
- 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

https://hemanthrajhemu.github.io

Fully Associative Cache



(a) Associative search with all block tags



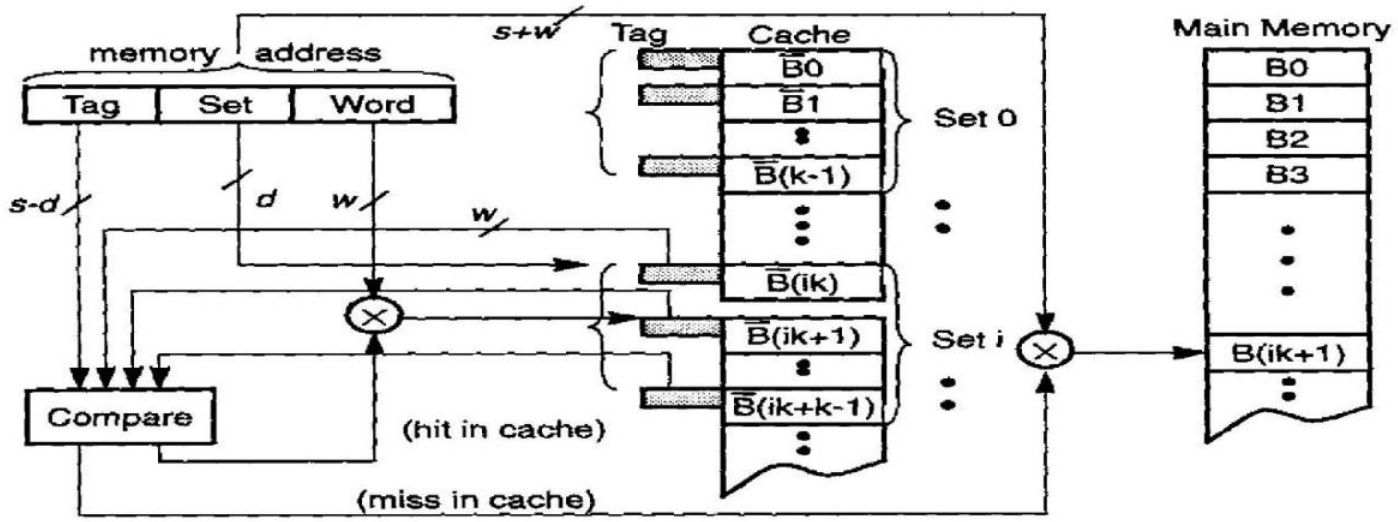
(b) Set-associative mapping between four block frames and 16 memory blocks

identified by the tag

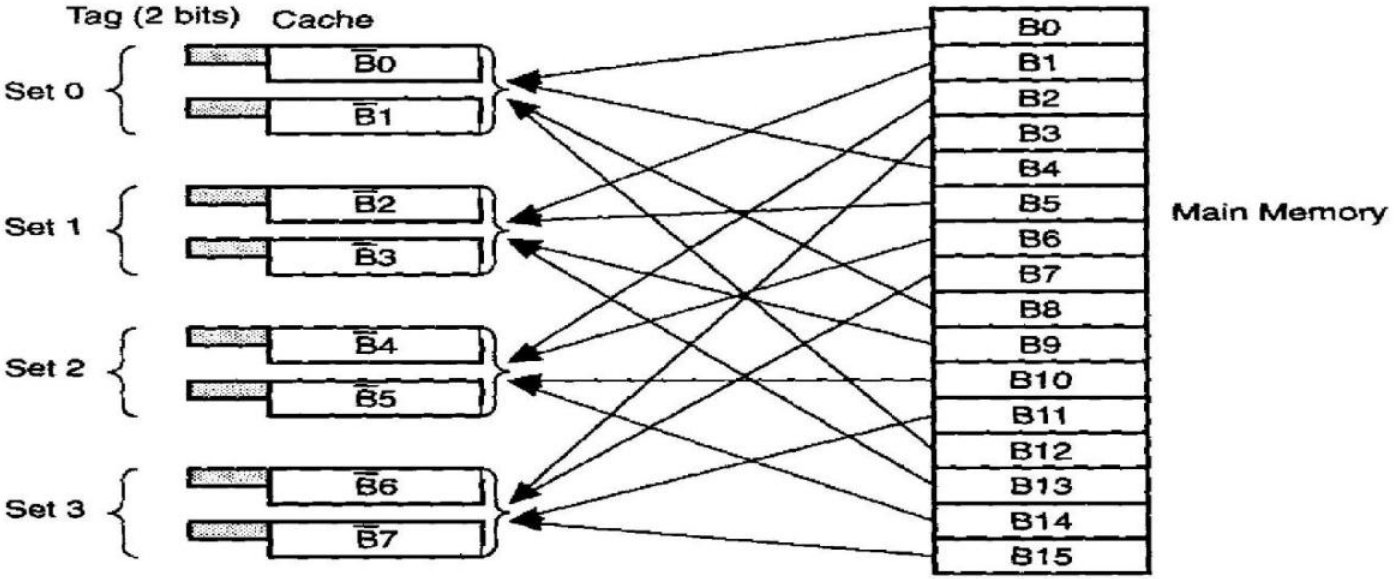
Figure 5.11 Fully associative cache organization and a mapping example.

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
- Compare the tag w/the *k* tags w/in the identified set
- $B_j \rightarrow \underline{B_f} \in S_i$ if $j \pmod v = i$



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



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

Figure 5.12 Set-associative cache organization and a two-way associative mapping example.

Sector Mapping Cache

- Partition cache and main memory into fixed size sectors then use fully associative search
- Use sector tags for search and block fields within sector to find block
- Only missing block loaded for a miss
- The *i*th block in a sector placed into the ith block frame in a destined sector frame
- Attach a valid/invalid bit to block frames

https://hemanthrajhemu.github.io

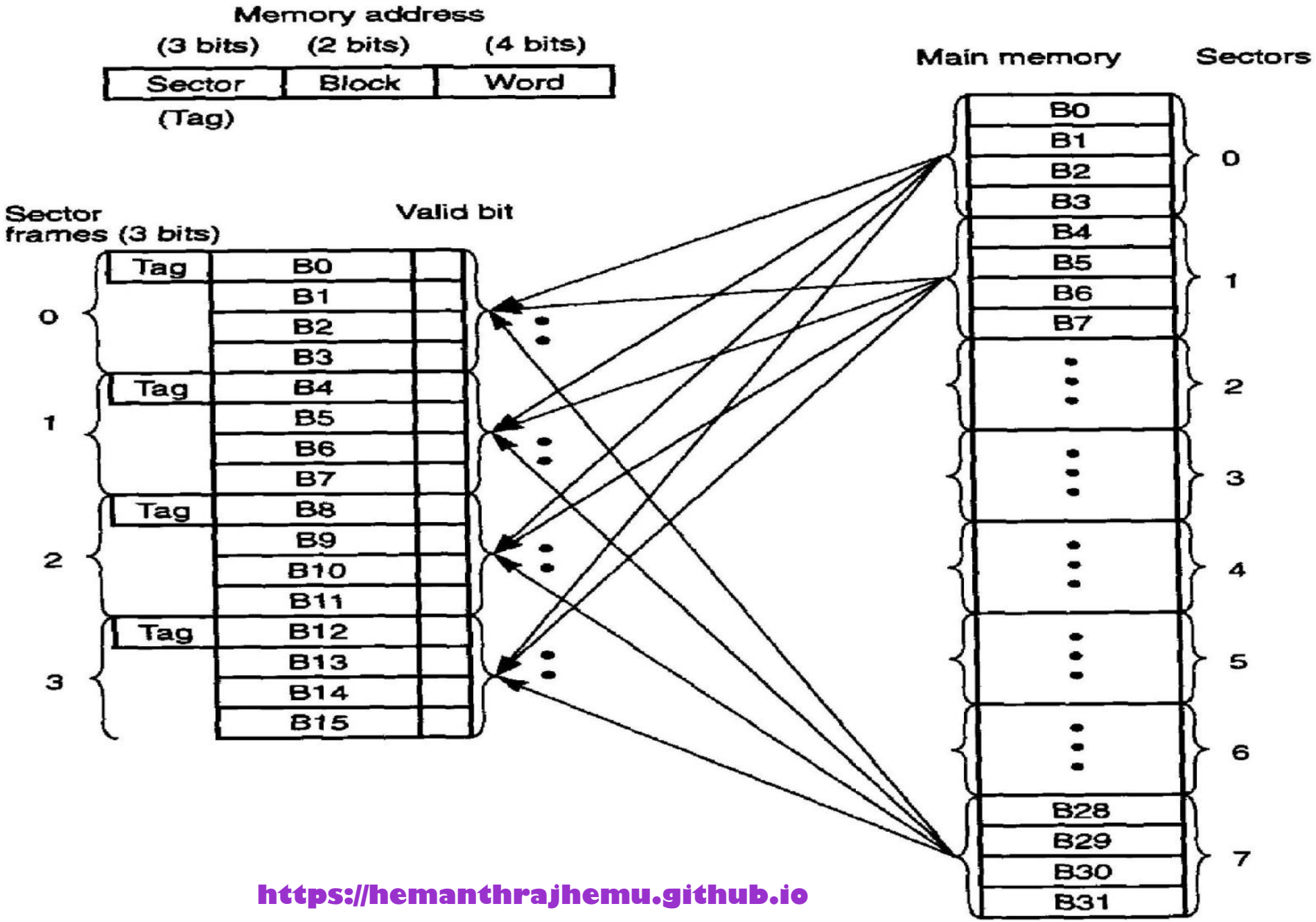


Figure 5.13 A four-way sector mapping cache organization.

Cache Performance Issues

- **Cycle count:** # of m/c cycles needed for cache access, update, and coherence
- **Hit ratio:** how effectively the cache can reduce the overall memory access time
- **Program trace driven simulation:** present snapshots of program behavior and cache responses
- **Cycle Counts**
  - Cache speed affected by underlying static or dynamic RAM technology, organization, and hit ratios
  - Write-thru/write-back policies affect count
  - Cache size, block size, set number, and associativity affect count
  - Directly related to hit ratio

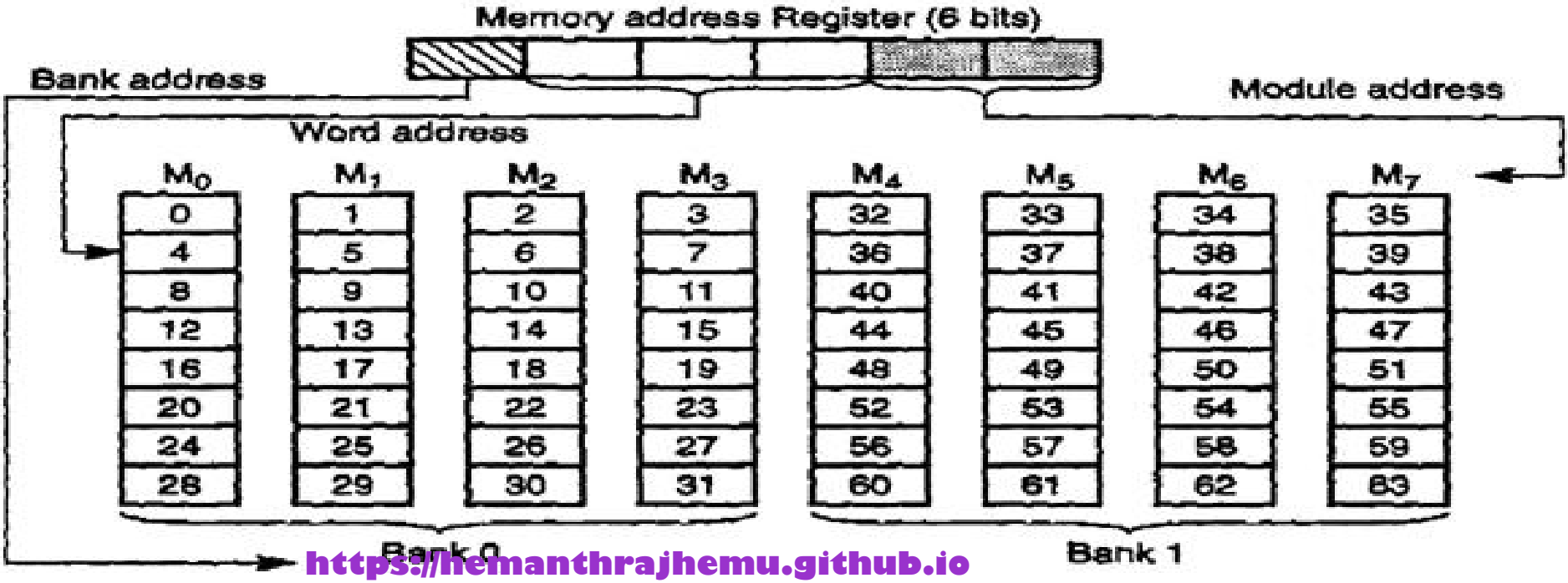
https://hemanthrajhemu.github.io

Cache Performance Issues

- **Hit Ratio**
  - Affected by cache size and block size
  - Increases w.r.t. increasing cache size
  - Limited cache size, initial loading, and changes in locality prevent 100% hit ratio
- **Effect of Block Size**
  - With fixed cache size, block size has impact
  - As block size increases, hit ratio improves due to spatial locality
  - Peaks at optimum block size, then decreases
  - If too large, many words in cache not used

Interleaved Memory Organization

- Goal is to close the speed gap b/t CPU/cache and main memory access
- Provides higher b/w for pipelined access of contiguous memory locations



(a) Four-way interleaving within each memory bank

https://hemanthrajhemu.github.io