UNIT - III
Memory System

Prof. Anand Motwani
Faculty, SCSE
VIT Bhopal University

Contents

- Locality
- Memory systems hierarchy
- Memory Types
- Main memory organization

Locality

Principle of Locality:

- **Programs tend to reuse data and instructions near those they have used recently, or that were recently referenced themselves.**
- **Temporal locality: Recently referenced items are likely to be referenced in the near future.**
- **Spatial locality: Items with nearby addresses tend to be referenced close together in time.**

```
sum = 0;
for (i = 0; i < n; i++)
    sum += a[i];
return sum;
```

Locality Example:

- Data
  - Reference array elements in succession (stride-1 reference pattern): Spatial locality
  - Reference `sum` each iteration: Temporal locality
- Instructions
  - Reference instructions in sequence: Spatial locality
  - Cycle through loop repeatedly: Temporal locality

Memory Hierarchies

Some fundamental and enduring properties of hardware and software:

- Fast storage technologies cost more per byte and have less capacity.
- The gap between CPU and main memory speed is widening.
- Well-written programs tend to exhibit good locality.

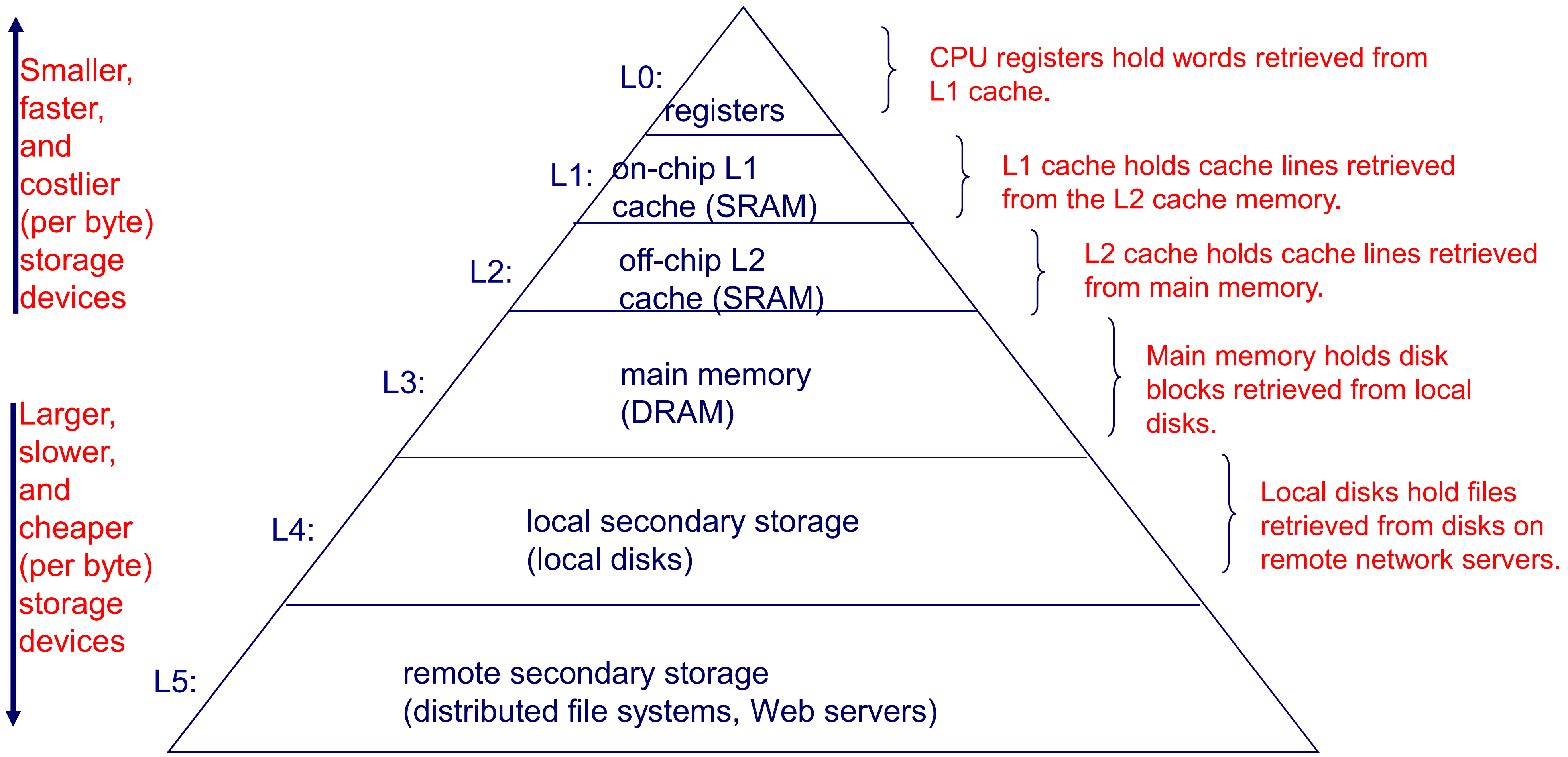
These fundamental properties complement each other beautifully.

They suggest an approach for organizing memory and storage systems known as a memory hierarchy.

Memory hierarchy was developed on a program behaviour known as locality of reference.

The references are generated by CPU for accessing data or instructions. These accesses tend to be clustered in certain regions in time, space and ordering.

An Example Memory Hierarchy


Smaller, faster, and costlier (per byte) storage devices
Larger, slower, and cheaper (per byte) storage devices
L0: registers
L1: on-chip L1 cache (SRAM)
L2: off-chip L2 cache (SRAM)
L3: main memory (DRAM)
L4: local secondary storage (local disks)
L5: remote secondary storage (distributed file systems, Web servers)
CPU registers hold words retrieved from L1 cache.
L1 cache holds cache lines retrieved from the L2 cache memory.
L2 cache holds cache lines retrieved from main memory.
Main memory holds disk blocks retrieved from local disks.
Local disks hold files retrieved from disks on remote network servers.

Internal Memory Types

Memory Type	Category	Erasure	Write Mechanism	Volatility
Random-access memory (RAM)	Read-write memory	Electrically, byte-level	Electrically	Volatile
Read-only memory (ROM)	Read-only memory	Not possible	Masks	Nonvolatile
Programmable ROM (PROM)			Electrically	
Erasable PROM (EPROM)	Read-mostly memory	UV light, chip-level		
Electrically Erasable PROM (EEPROM)		Electrically, byte-level		
Flash memory		Electrically, block-level		

Types of Memory based on Access

- Sequential Access

To find the required piece of information, the system must search the device from beginning of the memory. Memory devices that supports such access are called sequential access.

Example: Magnetic tape

- Direct Access

Refers to the condition when system can go directly to the information that is needed is known as direct access.

Example: Magnetic disk or optical disk.

Memory Access Methods

1. **Sequential Access:**

- **Start at the beginning and read through in order**
- **Access time depends on location of data and previous location**

2. Random Access:

- **Individual addresses identify locations exactly**
- **Access time is consistent across all locations and is independent previous access**

3. Direct Access:

- **Individual blocks have unique address**
- **Access is by jumping to vicinity then performing a sequential search**
- **Access time depends on location of data within "block" and previous location**

4. Associative Access

- Addressing information must be stored with data in a general data location
- A specific data element is located by a comparing desired address with address portion of stored elements
- Access time is independent of location or previous access

Example: cache

Contents

Types of Main memory (RAM) :

- SRAM
- DRAM
- Characteristics and Performance
- Latency
- Cycle time
- Bandwidth
- memory interleaving

Random-Access Memory (RAM): Types

Key features

- RAM is packaged as a chip.
- Basic storage unit is a cell (one bit per cell).
- Multiple RAM chips form a memory.

Static RAM (SRAM)

- Each cell stores bit with a six-transistor circuit.
- Retains value indefinitely, as long as it is kept powered.
- Relatively insensitive to disturbances such as electrical noise.
- Faster and more expensive than DRAM.

Dynamic RAM (DRAM)

- Each cell stores bit with a capacitor and transistor.
- Value must be refreshed every 10-100 ms.
- Sensitive to disturbances.
- Slower and cheaper than SRAM.

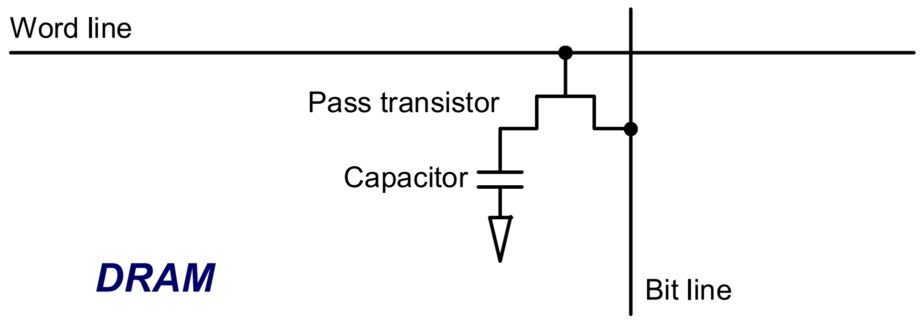
Memories: Review

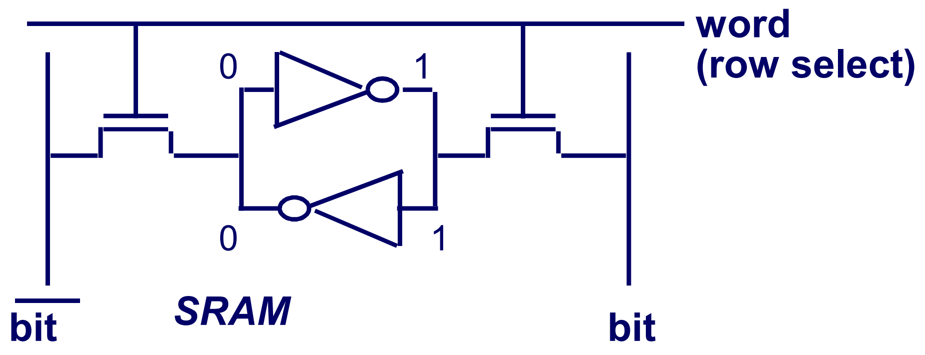
SRAM:

- **value is stored on a pair of inverting gates**
- **very fast but takes up more space than DRAM (4 to 6 transistors). Access times: 5 - 25ns**

DRAM:

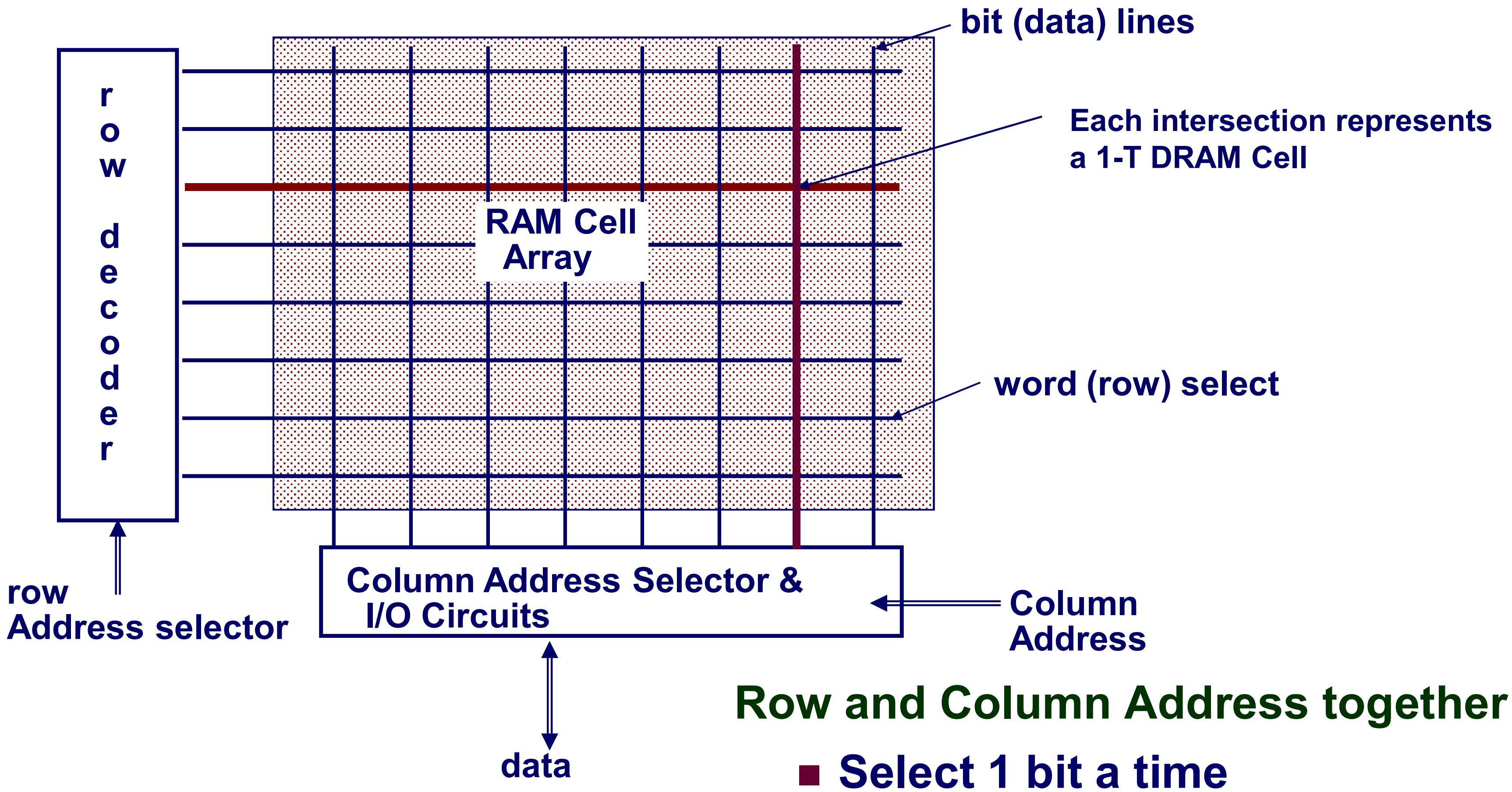
- **value is stored as a charge on capacitor (must be refreshed)**
- **very small but slower than SRAM (factor of 5 to 10)**





	DRAM	SRAM
Type of device	Analog	Digital
Electricity	Needed to refresh	No need to refresh
Speed	Slow	Fast
Construction	Simple	Complicated
Capacity	More	Less
Cost	Low	High
Application	Main Memory	Cache Memory

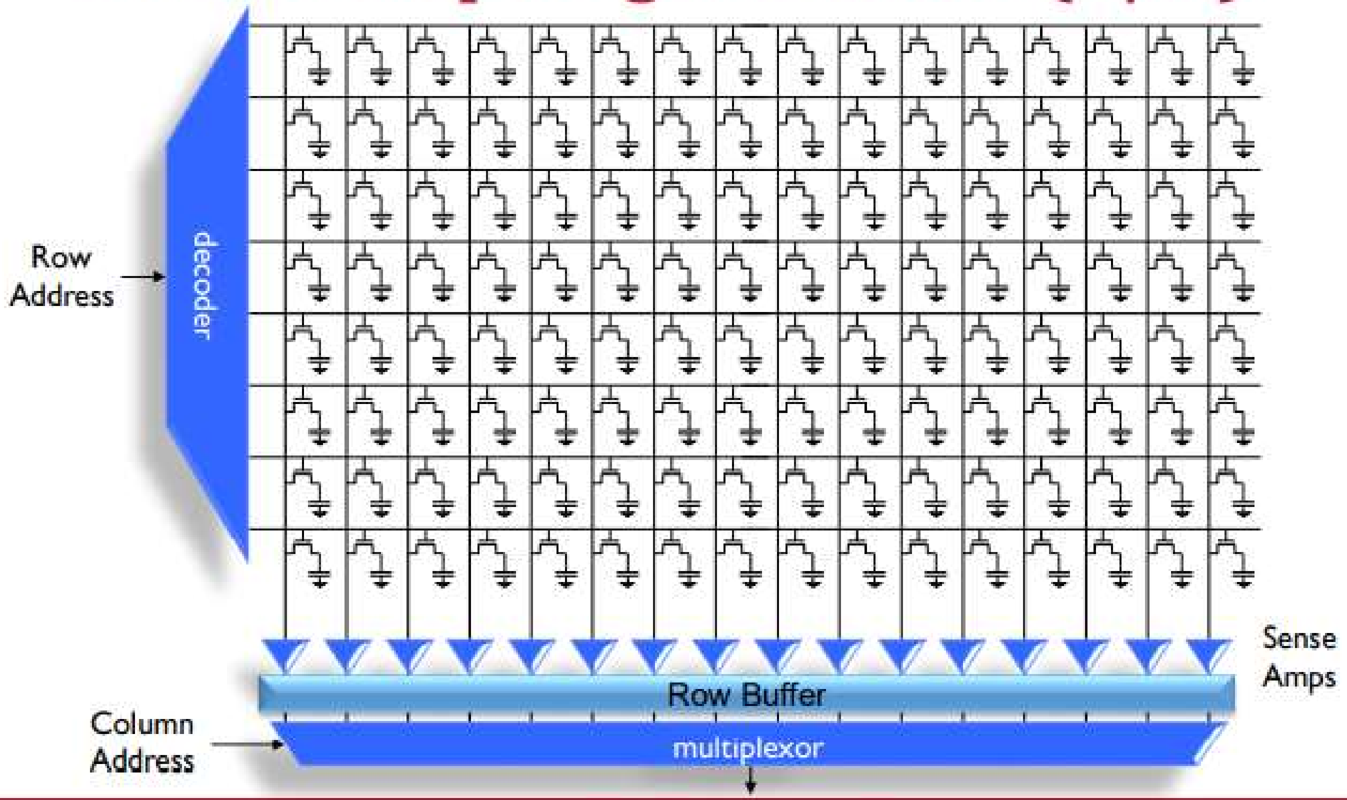
Classical DRAM Organization (square)



Row and Column Address together:

- Select 1 bit a time

DRAM Chip Organization (1/2)

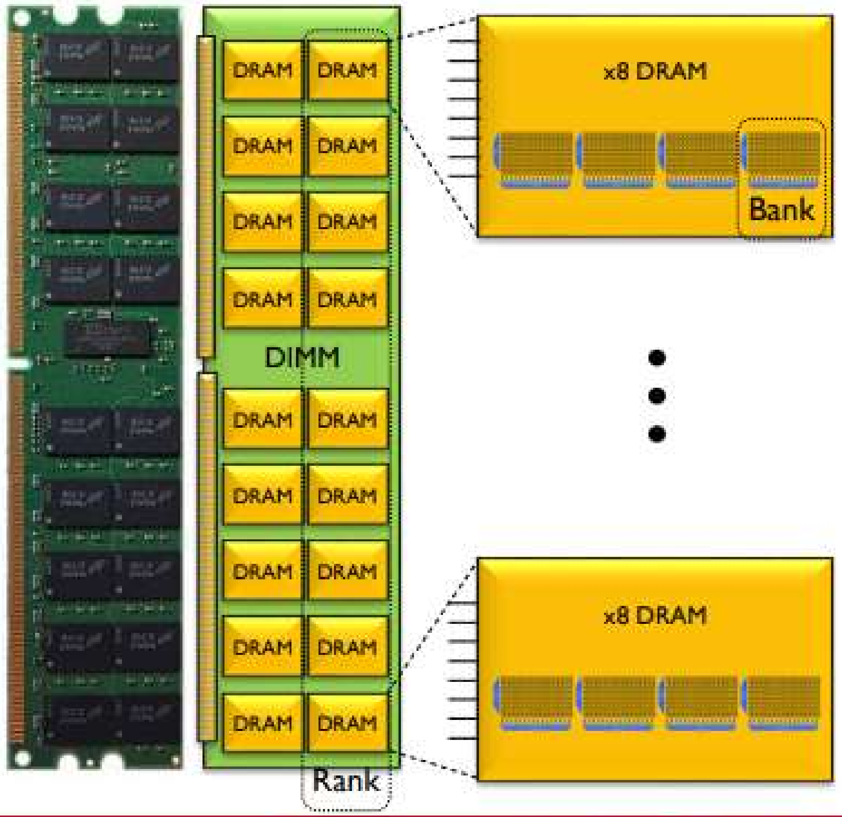


DRAM is much denser than SRAM

DRAM Chip Organization (2/2)

- Low-Level organization is very similar to SRAM
- Reads are *destructive*: contents are erased by reading
- *Row buffer* holds read data
  - Data in row buffer is called a *DRAM row*
    - Often called "page" – do not confuse with virtual memory page
  - Read gets entire row into the buffer
  - Block reads always performed out of the row buffer
    - Reading a whole row, but accessing one block
    - Similar to reading a cache line, but accessing one word

DRAM Organization



All banks within the rank share all address and control pins

All banks are independent, but can only talk to one bank at a time

x8 means each DRAM outputs 8 bits, need 8 chips for DDRx (64-bit)

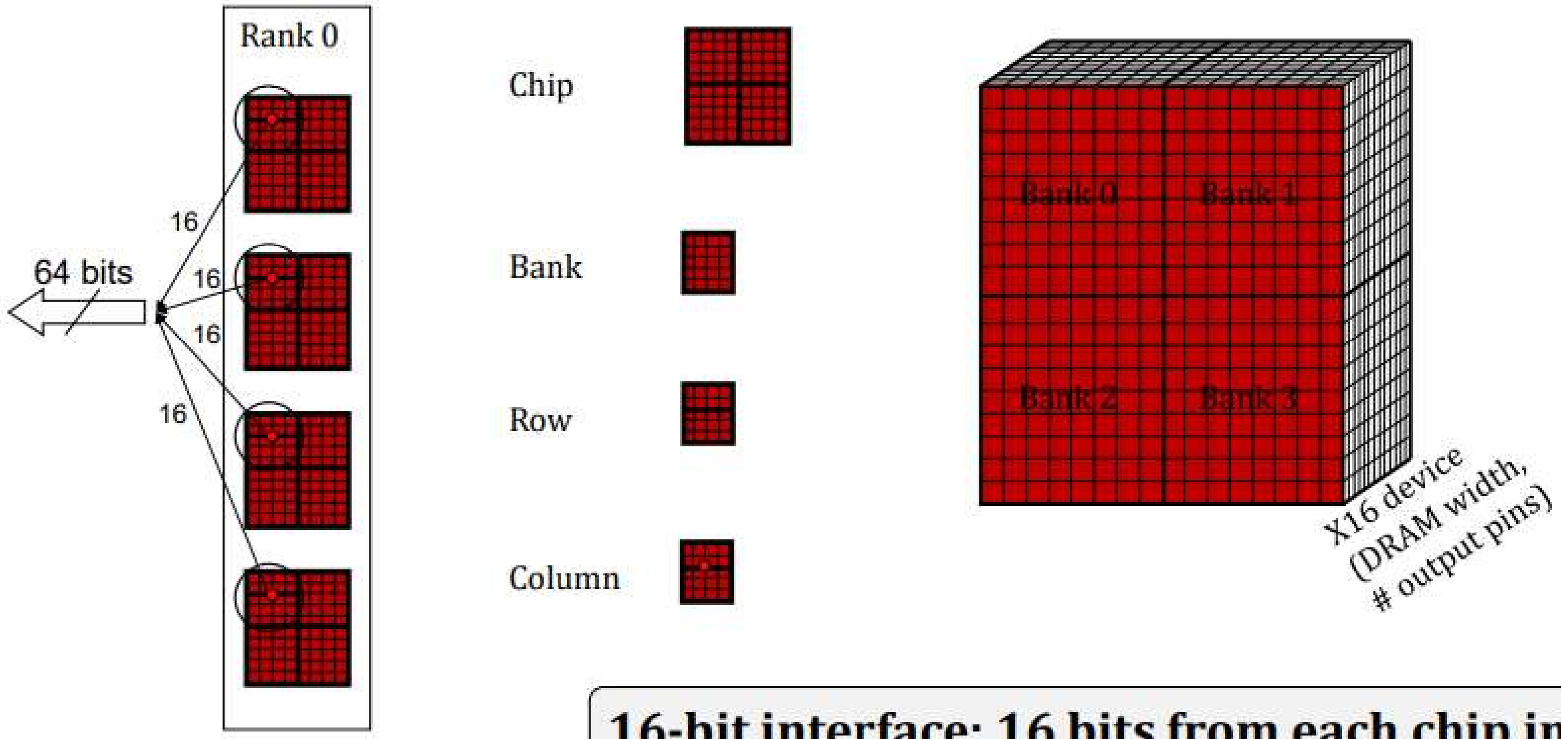
Why 9 chips per rank? 64 bits data, 8 bits ECC

Dual-rank x8 (2Rx8) DIMM

DIMM: a PCB with DRAM chips on the back and front

- Rank: a collection of DRAM chips that work together to respond to a request and keep the data bus full
- A 64-bit data will need 8 x8 DRAM chips or 4 x16 DRAM chips or..
- Bank: a subset of a rank that is busy during one request
- Row buffer: the last row (say, 8 KB) read from a bank, acts like a cache

Ranks, Banks, Rows, and Columns



16-bit interface: 16 bits from each chip in one go

Each rank has 64-bit wide data bus

If a rank is of width x8 then # DRAM chips ??

What about x4, # DRAM chips ??

If a rank is of width x8 then # DRAM chips ?? 8

What about if rank is of width x4, # DRAM chips ?? 16

Nonvolatile Memories

DRAM and SRAM are volatile memories

- Lose information if powered off.

Nonvolatile memories retain value even if powered off.

- Generic name is read-only memory (ROM).
- Misleading because some ROMs can be read and modified.

Types of ROMs

- Programmable ROM (PROM)
- Eraseable programmable ROM (EPROM)
- Electrically eraseable PROM (EEPROM)
- Flash memory

Firmware

- Program stored in a ROM
  - Boot time code, BIOS (basic input/ouput system)
  - graphics cards, disk controllers.

Characteristics of Memory

- **Capacity:** It is global volume of information (in bits) that the memory can store.
- **Access time:** The time interval between the read request and the availability of data.
- **Cycle time:** It is the minimum time interval between two successive access.
- **Throughput:** It is the volume of information exchanged per unit of time, expressing in bits/second.
- **Non-Volatility:** The ability of a memory to store data when it is not being supplied with electricity

Memory Performance Metrics

Latency: Time to access one word

- ***Access time***: time between the request and when the data is available (or written)
- ***Cycle time***: time between requests
- Usually cycle time > access time
- Typical read access times for SRAMs in 2004 are 2 to 4 ns for the fastest parts to 8 to 20ns for the typical largest parts

Bandwidth: How much data from the memory can be supplied to the processor per unit time

- width of the data channel * the rate at which it can be used

Size*: DRAM to SRAM *4 to 8

Cost/Cycle time*: SRAM to DRAM *8 to 16

Numerical 1:

How many 256*4 chips are needed to provide a memory capacity of 2048 bytes.

Solution 1:

256 means 8 address lines

4 data lines

Size to be constructed 2048 bytes means : $2048*8 = 2^{11} * 8$

It requires 11 address lines and 8 data lines.

So no. of rows required is 2048/256 = 8

No. of columns 8 / 4 = 2.

Total RAMs required = 16.

Numerical 2:

A computer employs RAM chips of 256*8 and ROM chips of 1024*8.

The computer system needs 2K bytes of RAM, 4K bytes of ROM.

How many RAM and ROM chips are needed.

Step 1 of 5

Size of RAM chip $= 256 \times 8$

Memory size required $= 2$ k bytes $= 2 \times 1024$ bytes $= 2 \times 1024 \times 8$ bits

Total number of RAM chips required

$$\frac{(2 \times 1024 \times 8)}{(256 \times 8)} = 8 \text{ Chips}$$

Size of ROM chip = 1024×8

Memory size required = 4 k bytes $= 4 \times 1024$ bytes $4 \times 1024 \times 8$ bits

Total number of ROM chips required

$$\frac{(4 \times 1024 \times 8)}{(1024 \times 8)} = 4 \text{ Chips}$$

Hence, $\boxed{8}$ RAM chips and $\boxed{4}$ ROM chips are required

Suppose a large memory of 1K x 4 is to be constructed from 512 x 2 RAM chips.

1. For small chip no. of address lines required =

No. of data lines = 2

2. For large memory, address lines = ?

3. data lines = ?

4. No. of total RAM chips required = ?

Cache memory:

Address mapping

- line size

Replacement and policies

- coherence

Cache

Small amount of fast memory

Sits between normal main memory and CPU

May be located on CPU chip or module

Caches

Cache: A smaller, faster storage device that acts as a staging area for a subset of the data in a larger, slower device.

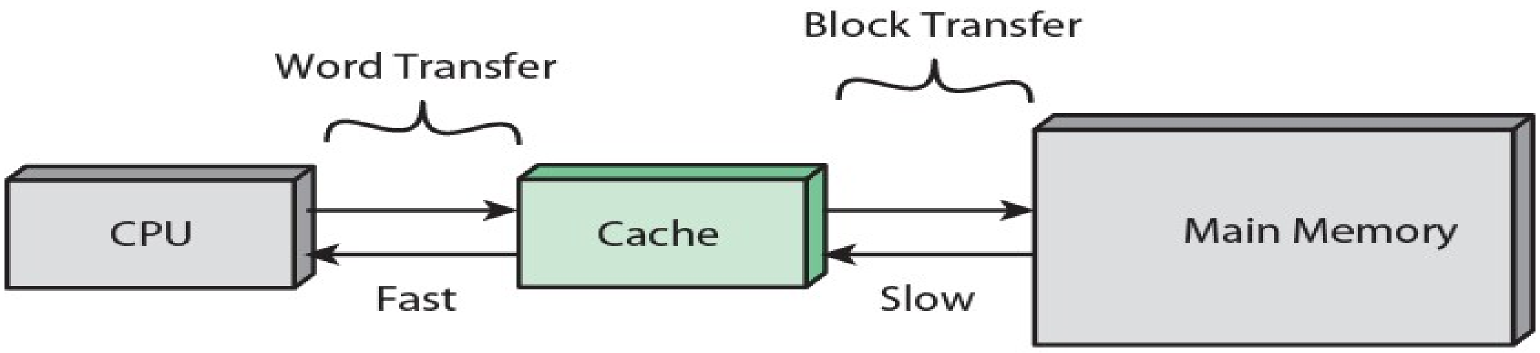
Fundamental idea of a memory hierarchy:

- **For each k, the faster, smaller device at level k serves as a cache for the larger, slower device at level k+1.**

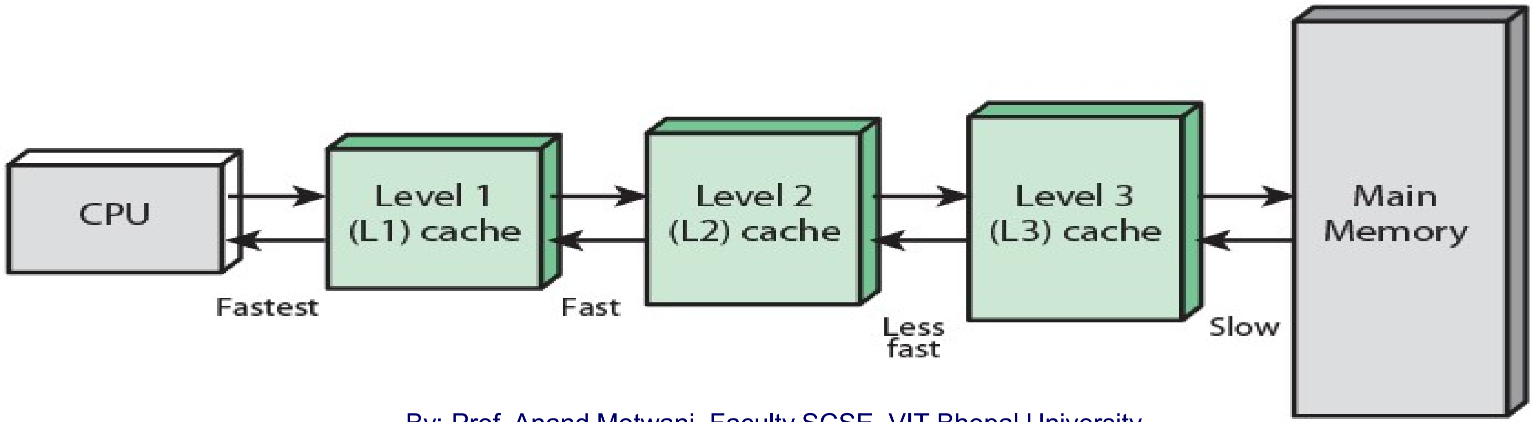
Why do memory hierarchies work?

- **Programs tend to access the data at level k more often than they access the data at level k+1.**
- **Thus, the storage at level k+1 can be slower, and thus larger and cheaper per bit.**
- **Net effect: A large pool of memory that costs as much as the cheap storage near the bottom, but that serves data to programs at the rate of the fast storage near the top.**

Cache and Main Memory

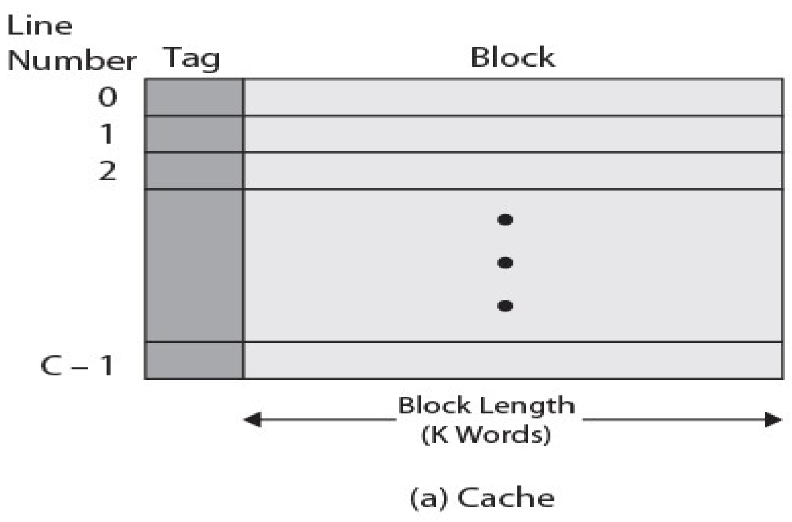


(a) Single cache

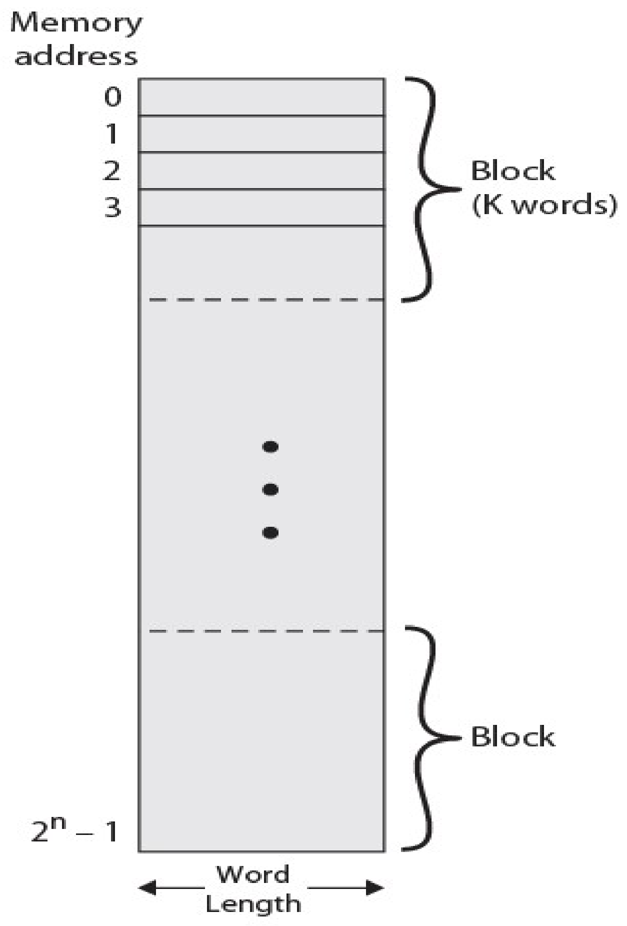


(b) Three-level cache organization

Cache/Main Memory Structure



(a) Cache

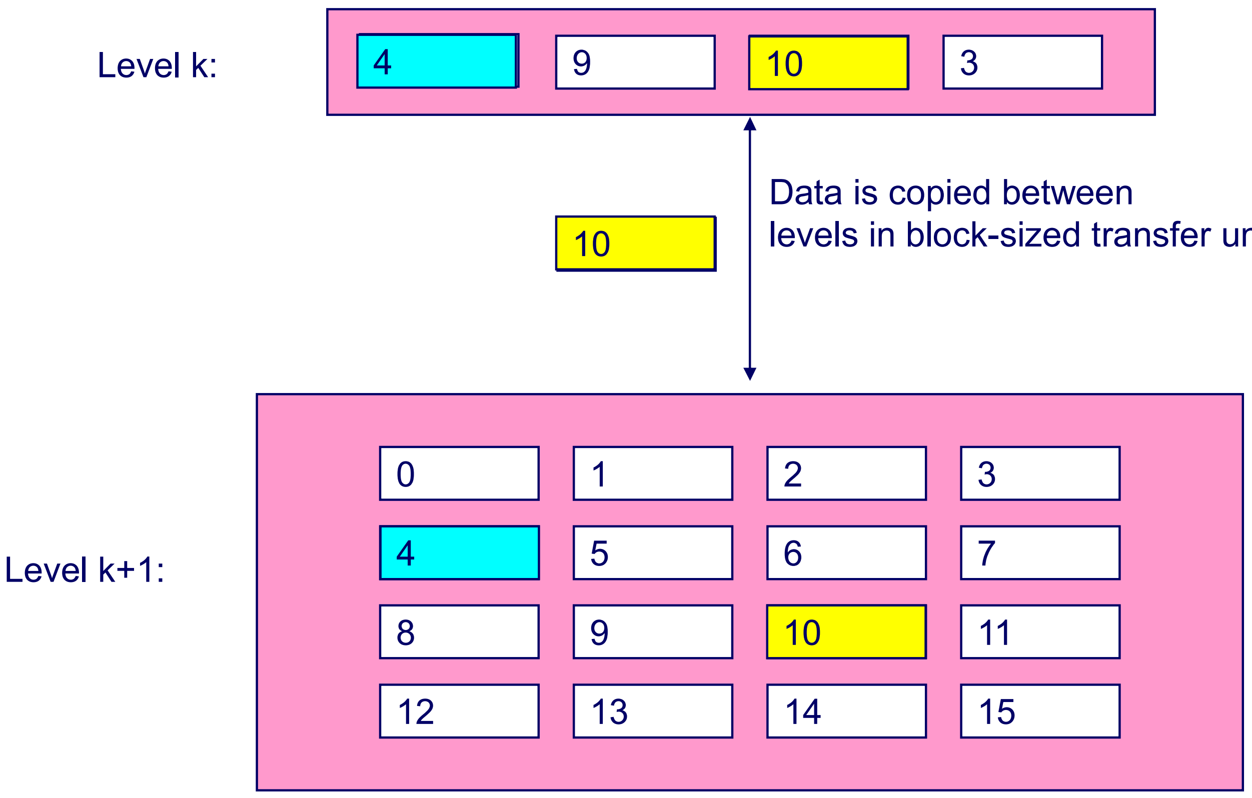


(b) Main memory

Cache operation – overview

- CPU requests contents of memory location
- Check cache for this data
- If present, get from cache (fast)
- If not present, read required block from main memory to cache
- Then deliver from cache to CPU
- Cache includes tags to identify which block of main memory is in each cache slot

Caching in a Memory Hierarchy

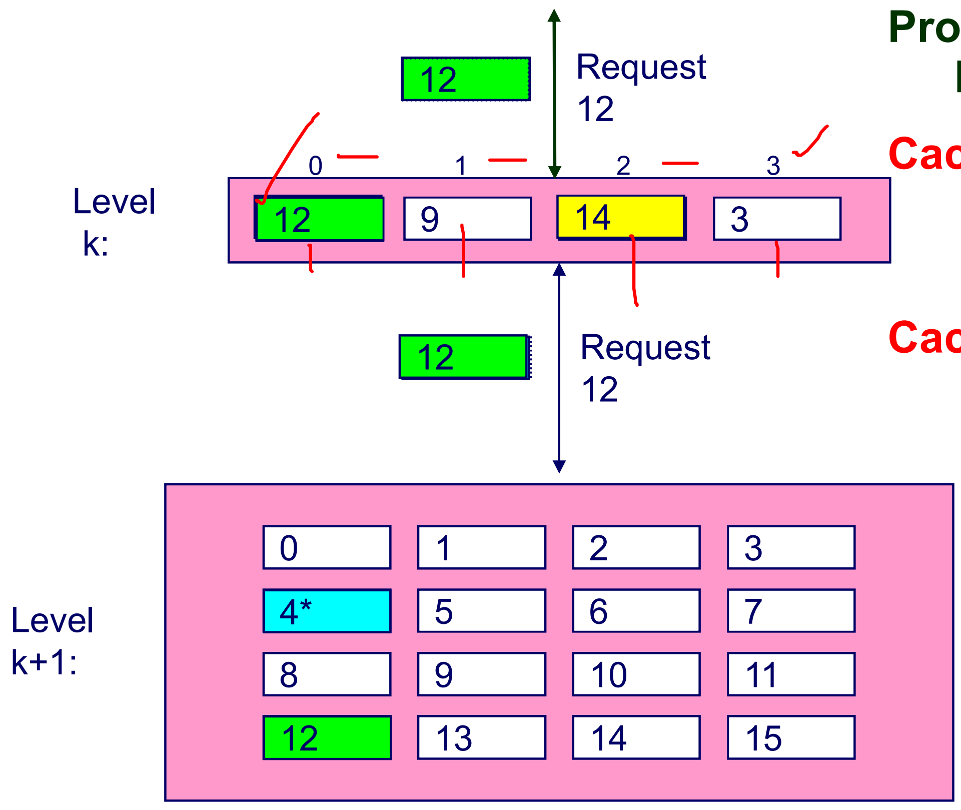


Smaller, faster, more expensive device at level k caches a subset of the blocks from level k+1

Data is copied between levels in block-sized transfer units

Larger, slower, cheaper storage device at level k+1 is partitioned into blocks.

General Caching Concepts



Program needs object d, which is stored in some block b.

Cache hit

- Program finds b in the cache at level k. E.g., block 14.

Cache miss

- b is not at level k, so level k cache must fetch it from level k+1. E.g., block 12.
- If level k cache is full, then some current block must be replaced (evicted). Which one is the "victim"?
  - **Placement policy:** where can the new block go? E.g., b mod 4
  - **Replacement policy:** which block should be evicted? E.g., LRU

12 mod 4 = 0

General Caching Concepts

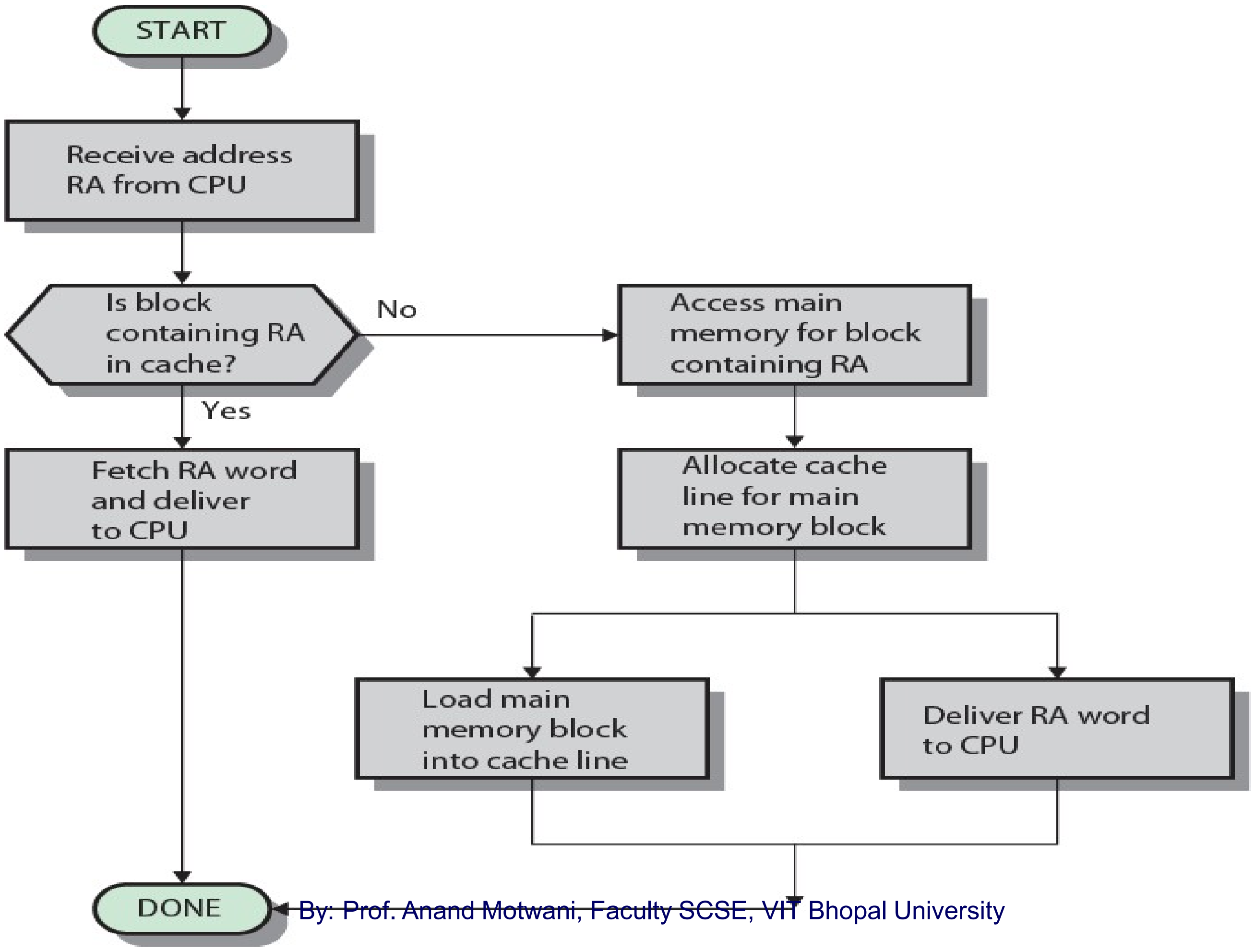
Types of cache misses:

- **Cold (compulsary) miss**
  - **Cold misses occur because the cache is empty.**
- **Conflict miss**
  - **Most caches limit blocks at level k+1 to a small subset (sometimes a singleton) of the block positions at level k.**
  - **E.g. Block i at level k+1 must be placed in block (i mod 4) at level k+1.**
  - **Conflict misses occur when the level k cache is large enough, but multiple data objects all map to the same level k block.**
  - **E.g. Referencing blocks 0, 8, 0, 8, 0, 8, ... would miss every time.**
- **Capacity miss**
  - **Occurs when the set of active cache blocks (working set) is larger than the cache.**

Examples of Caching in the Hierarchy

Cache Type	What Cached	Where Cached	Latency (cycles)	Managed By
Registers	4-byte word	CPU registers	0	Compiler
TLB	Address translations	On-Chip TLB	0	Hardware
L1 cache	32-byte block	On-Chip L1	1	Hardware
L2 cache	32-byte block	Off-Chip L2	10	Hardware
Virtual Memory	4-KB page	Main memory	100	Hardware+OS
Buffer cache	Parts of files	Main memory	100	OS
Network buffer cache	Parts of files	Local disk	10,000,000	AFS/NFS client
Browser cache	Web pages	Local disk	10,000,000	Web browser
Web cache	Web pages	Remote server disks	1,000,000,000	Web proxy server

Cache Read Operation - Flowchart


START
Receive address RA from CPU
Is block containing RA in cache?
No
Yes
Access main memory for block containing RA
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

Cache Design

- Addressing
- Size
- Mapping Function
- Replacement Algorithm
- Write Policy
- Block Size
- Number of Caches

Size does matter

- Cost
  - More cache is expensive
- Speed
  - More cache is faster (up to a point)
  - Checking cache for data takes time

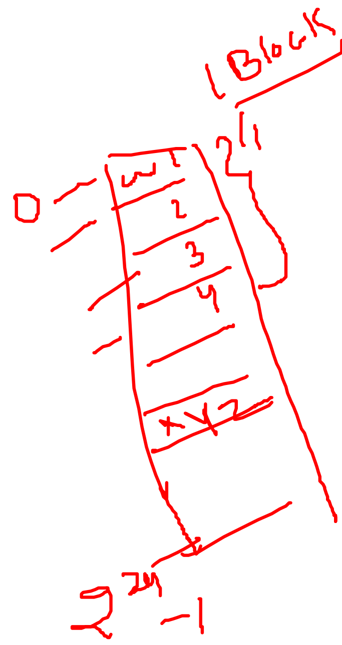
Elements of Cache Design

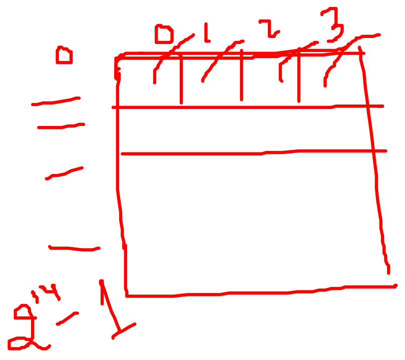
- **Cache Size: It is the amount of main memory data that cache can hold.**
- **Block Size: It is the no. Of words(bytes) grouped together into a single unit called a block.**
- **Mapping Function:(Block placement phenomenon): Direct, Associative and Set associative.**
- **Replacement Algorithms:**
- **Write Policy:**

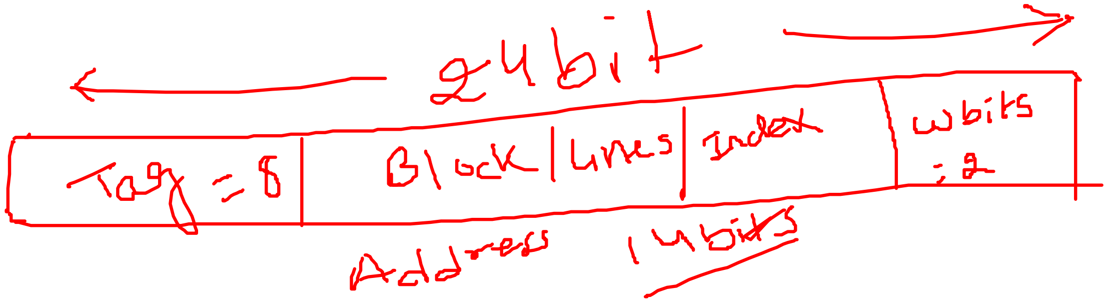
Mapping Function

- Cache of 64kByte $= 2^6 \cdot 2^{10}$ Byte $= 2^{16}$ Bytes
- Cache block of 4 bytes
  - i.e. cache is 16k (2^{14}) lines of 4 bytes
- 16MBytes main memory $= 2^4 \cdot 2^{20}$ Bytes $= 2^{24}$ Bytes
- 24 bit address
  - (2^{24}=16M)

$$\frac{2^{16}}{2^2} = 2^{14}$$

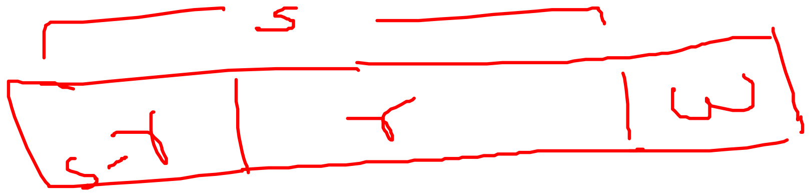




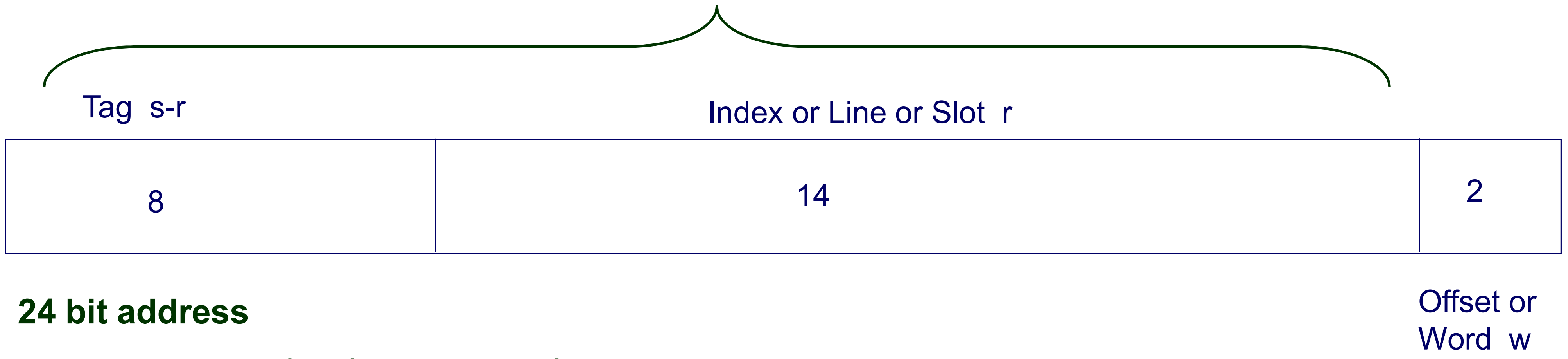


Direct Mapping

- Each block of main memory maps to only one cache line
  - i.e. if a block is in cache, it must be in one specific place
- Address is in two parts
- Least Significant w bits identify unique word
- Most Significant s bits specify one memory block
- The MSBs are split into a cache line field r and a tag of s-r (most significant)



Direct Mapping Address Structure



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

Consider a cache consisting of 128 blocks of 16 words

each, for a total of 2048 (2K) words, and assume that

✓ the main memory is addressable by a 16-bit address.

The main memory has 64K words, which we will

view as 4K blocks of 16 words each

1w = 8bit = 1Byte

$128 \times 16 = 2^{11} = 2 \cdot 2^{10}$

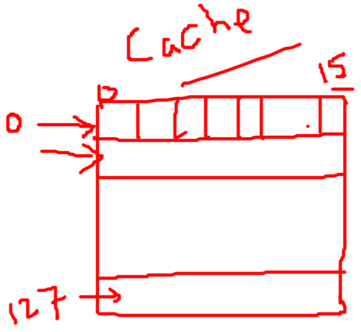
$= 2k$ words

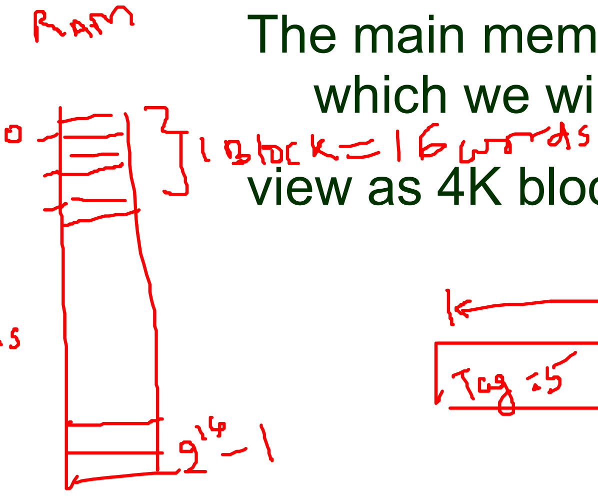
$2^{16} = 2^{6} \cdot 2^{10} = 64K$ words

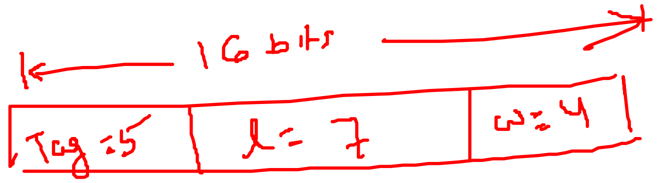
No of blocks in RAM

$= \frac{2^{16}}{2^{4}} = 2^{12}$ blocks

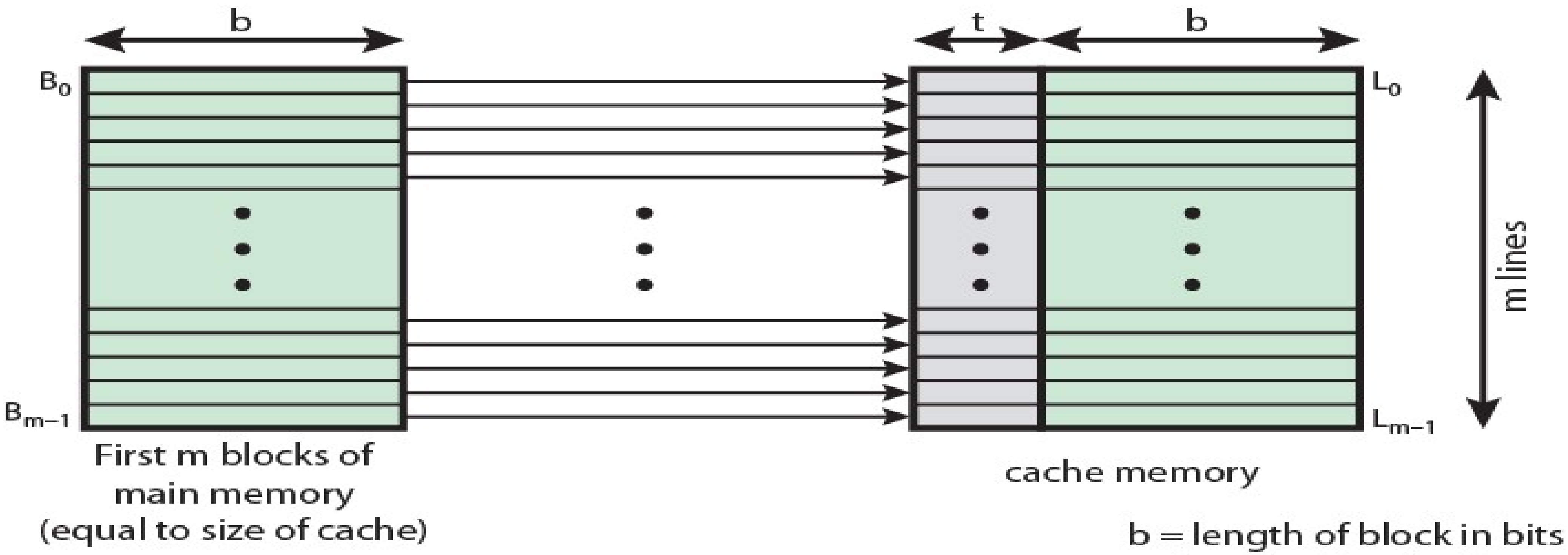
$= 4096$







Direct Mapping from Cache to Main Memory

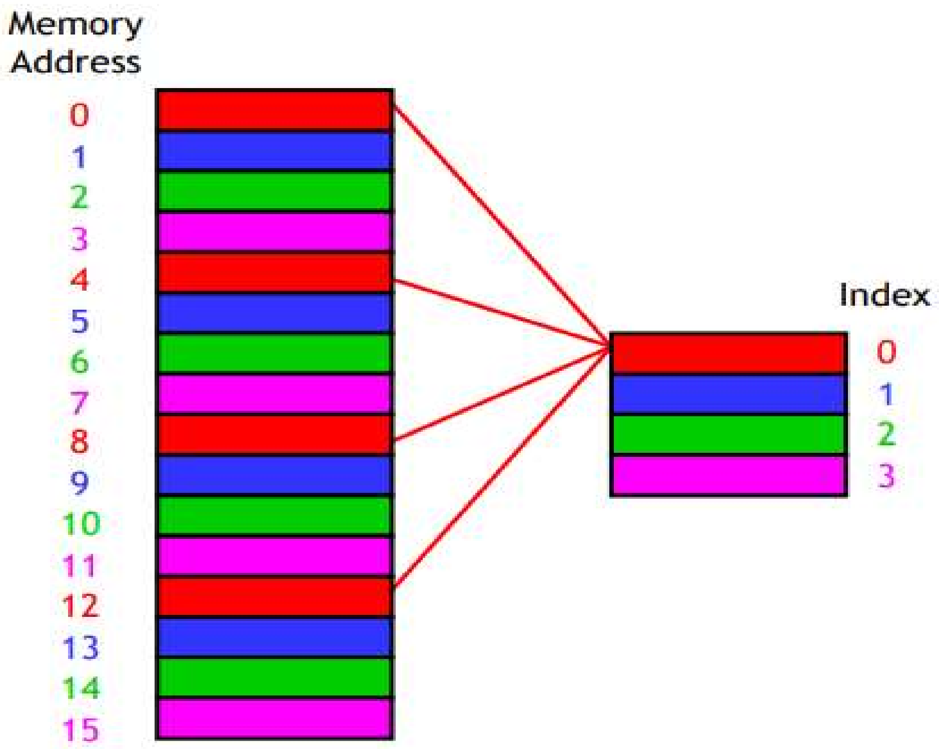


b
t
b
B_0
B_{m-1}
L_0
L_{m-1}
m lines
First m blocks of main memory (equal to size of cache)
cache memory
b = length of block in bits
t = length of tag in bits

(a) Direct mapping

Where should we put data in the cache?

- A direct-mapped cache is the simplest approach: each main memory address maps to exactly one cache block.
- For example, on the right is a 16-byte main memory and a 4-byte cache (four 1-byte blocks).
- Memory locations 0, 4, 8 and 12 all map to cache block 0.
- Addresses 1, 5, 9 and 13 map to cache block 1, etc.
- How can we compute this mapping?



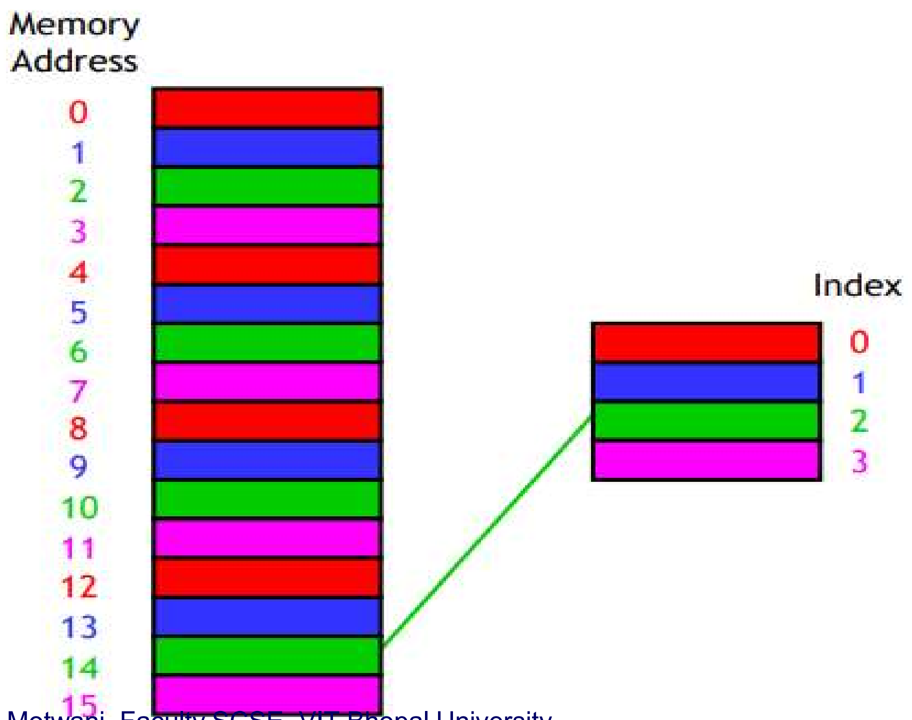
It's all divisions...

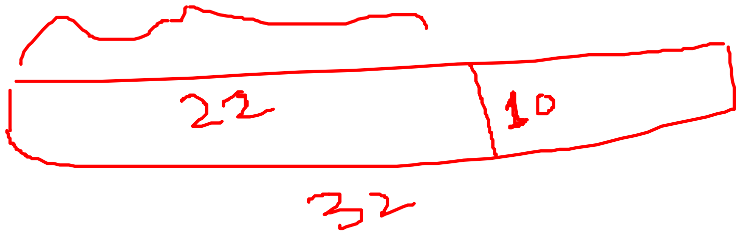
- One way to figure out which cache block a particular memory address should go to is to use the mod (remainder) operator.
- If the cache contains 2^k blocks, then the data at memory address i would go to cache block index

  $i \bmod 2^k$

- For instance, with the four-block cache here, address 14 would map to cache block 2.

  $14 \bmod 4 = 2$





Suppose 32 bit memory address ?

2^{10} byte cache means ?

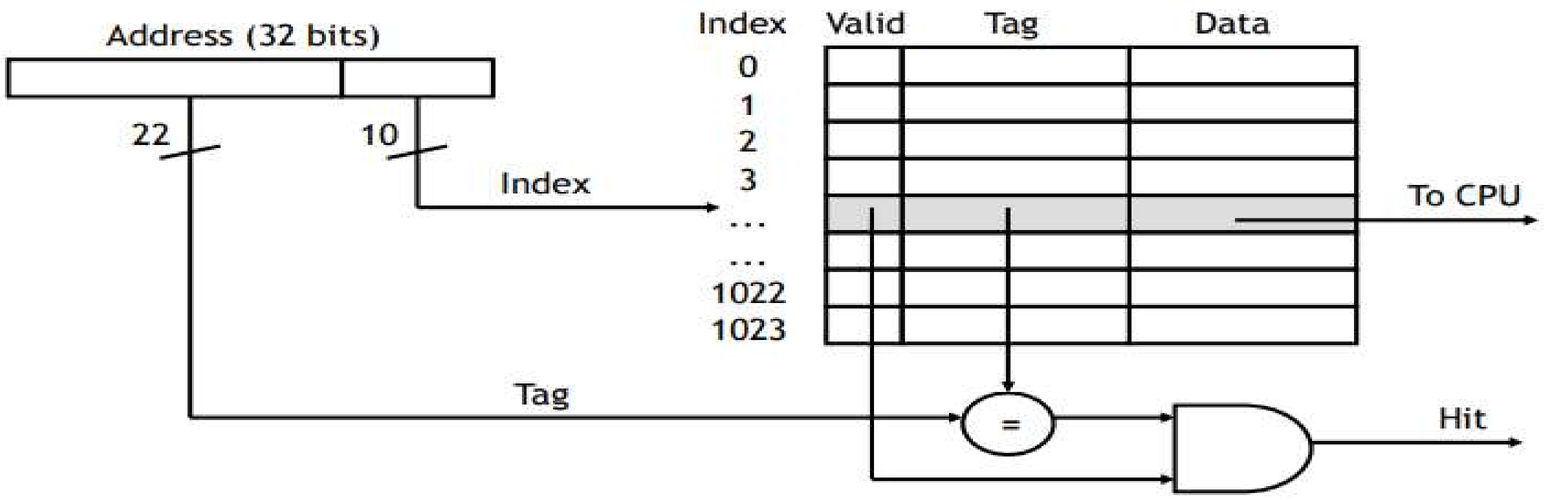
Cache Lines = ? 10

No. of bits in MSB = ?

Tag bits = ?

What happens on a cache hit

- When the CPU tries to read from memory, the address will be sent to a cache controller.
  - The lowest *k* bits of the address will index a block in the cache.
  - If the block is valid and the tag matches the upper (s - r) bits of the s -bit address, then that data will be sent to the CPU.
- Here is a diagram of a 32-bit memory address and a 2^{10}-byte cache.



$2^{16} = 2^{6} \cdot 2^{10}$

64K

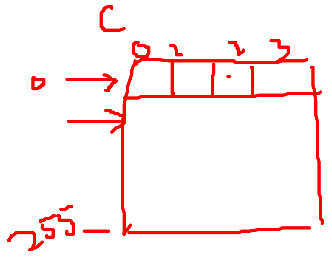
Numerical

A computer has a main memory of 64k ~~x~~ 16, and a cache of 1K words. The cache uses direct mapping with a block size of 4 words.

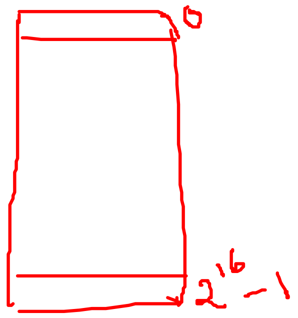
(a) how many bits are there in tag, index and word fields of address format?

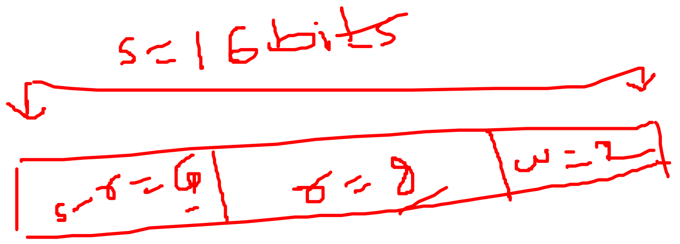
(b) how many bits are there in each ~~word~~ (Line) of cache? 32

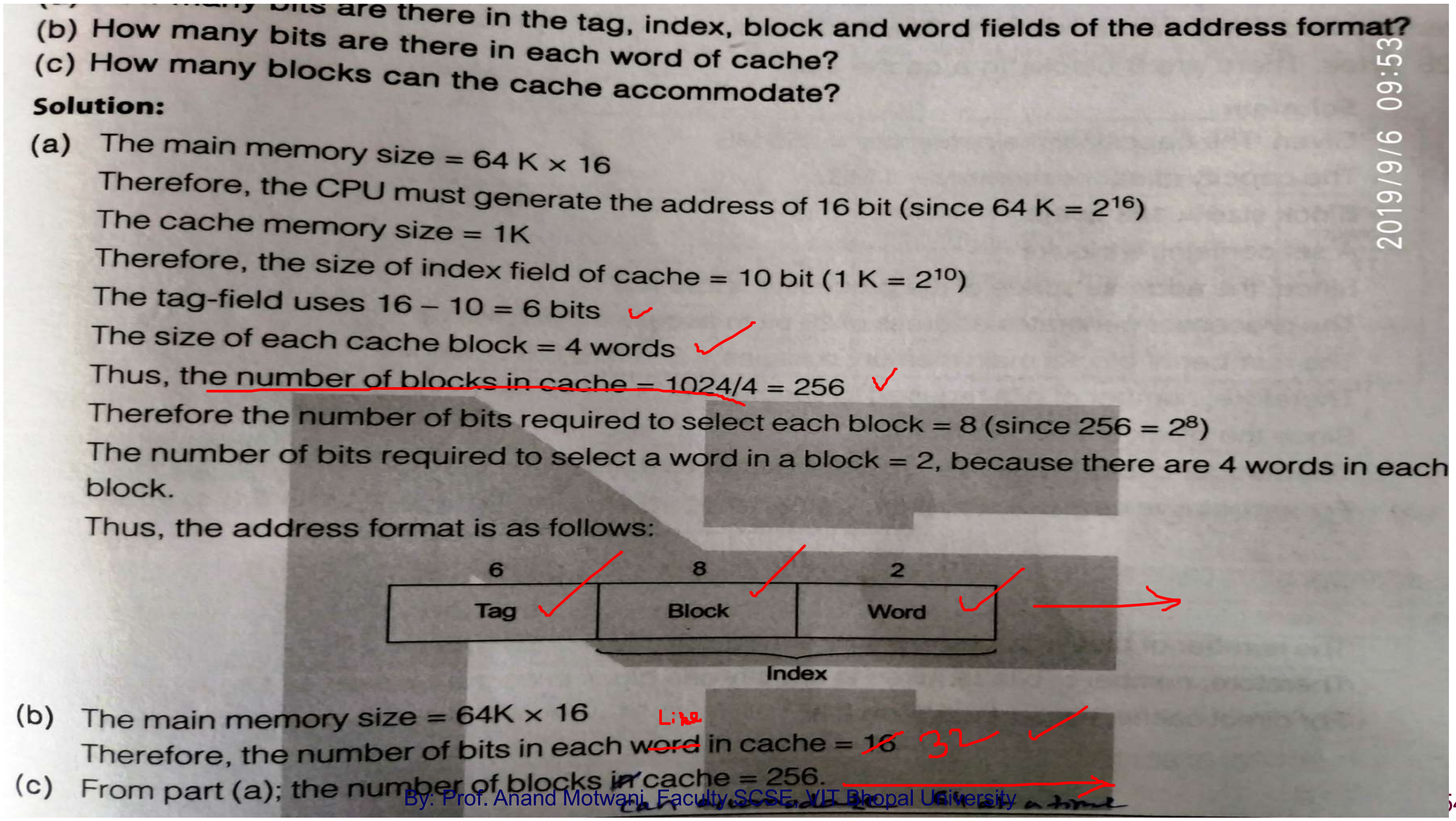
(c) how many blocks the cache can accommodate? $256 = 2^8$



$\frac{2^{10}}{2^2} = 2^8$







bits are there in the tag, index, block and word fields of the address format?

(b) How many bits are there in each word of cache?

(c) How many blocks can the cache accommodate?

2019/9/6 09:53

Solution:

(a) The main memory size = 64 K × 16

Therefore, the CPU must generate the address of 16 bit (since 64 K = 2^{16})

The cache memory size = 1K

Therefore, the size of index field of cache = 10 bit (1 K = 2^{10})

The tag-field uses 16 – 10 = 6 bits ✓

The size of each cache block = 4 words ✓

Thus, the number of blocks in cache = 1024/4 = 256 ✓

Therefore the number of bits required to select each block = 8 (since 256 = 2^8)

The number of bits required to select a word in a block = 2, because there are 4 words in each block.

Thus, the address format is as follows:

6	8	2
Tag ✓	Block ✓	Word ✓

$\underbrace{\text{Block} \quad \text{Word}}_{\text{Index}}$

(b) The main memory size = 64K × 16

Therefore, the number of bits in each ~~word~~ Line in cache = ~~16~~ 32 ✓

(c) From part (a); the number of blocks in cache = 256.

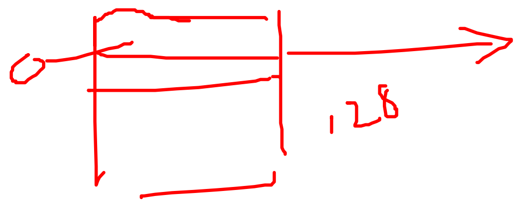
can accommodate [illegible] at a time

Direct Mapping pros & 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 – condition called thrashing

$256 \bmod 128 = 0$

$512 \bmod 128 = 0$



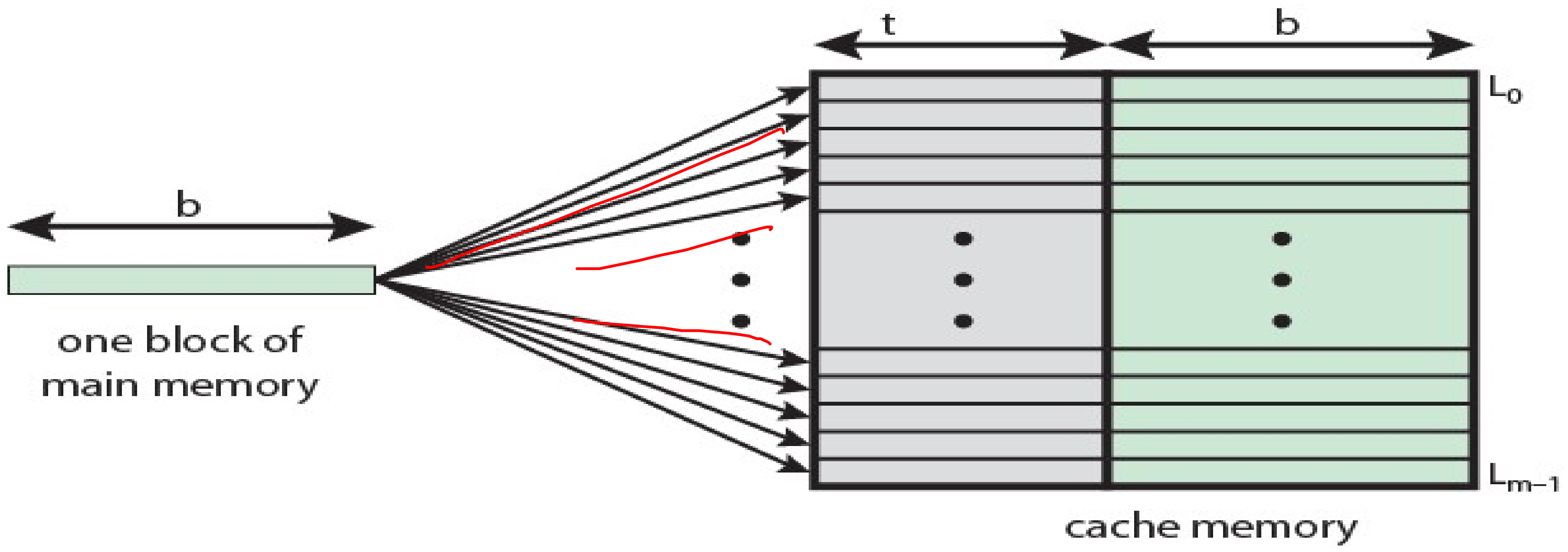
Victim Cache

- Lower miss penalty
- Remember what was discarded
  - Already fetched
  - Use again with little penalty
- Fully associative
- 4 to 16 cache lines
- Between direct mapped L1 cache and next memory level

Associative Mapping

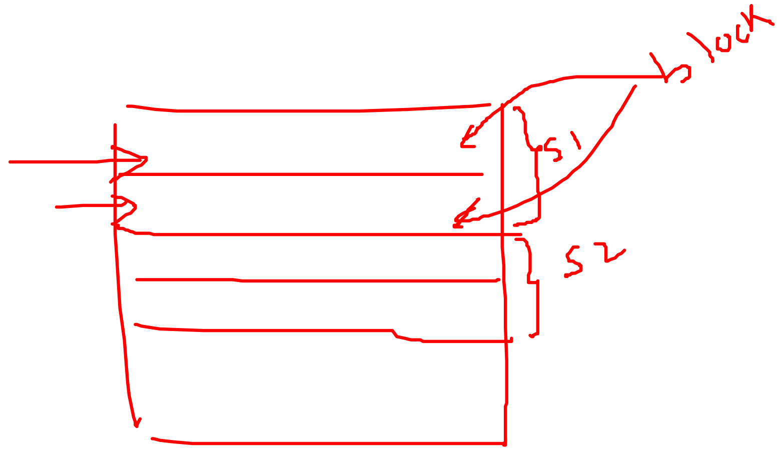
- A main memory block can load into any line of cache
- Memory address is interpreted as tag and word
- Tag uniquely identifies block of memory
- Every line’s tag is examined for a match
- Cache searching gets expensive

Associative Mapping from Cache to Main Memory



Set Associative Mapping

- Cache is divided into a number of sets
- Each set contains a number of lines
- A given block maps to any line in a given set
  - e.g. Block B can be in any line of set i
- e.g. 2 lines per set
  - 2 way associative mapping
  - A given block can be in one of 2 lines in only one set

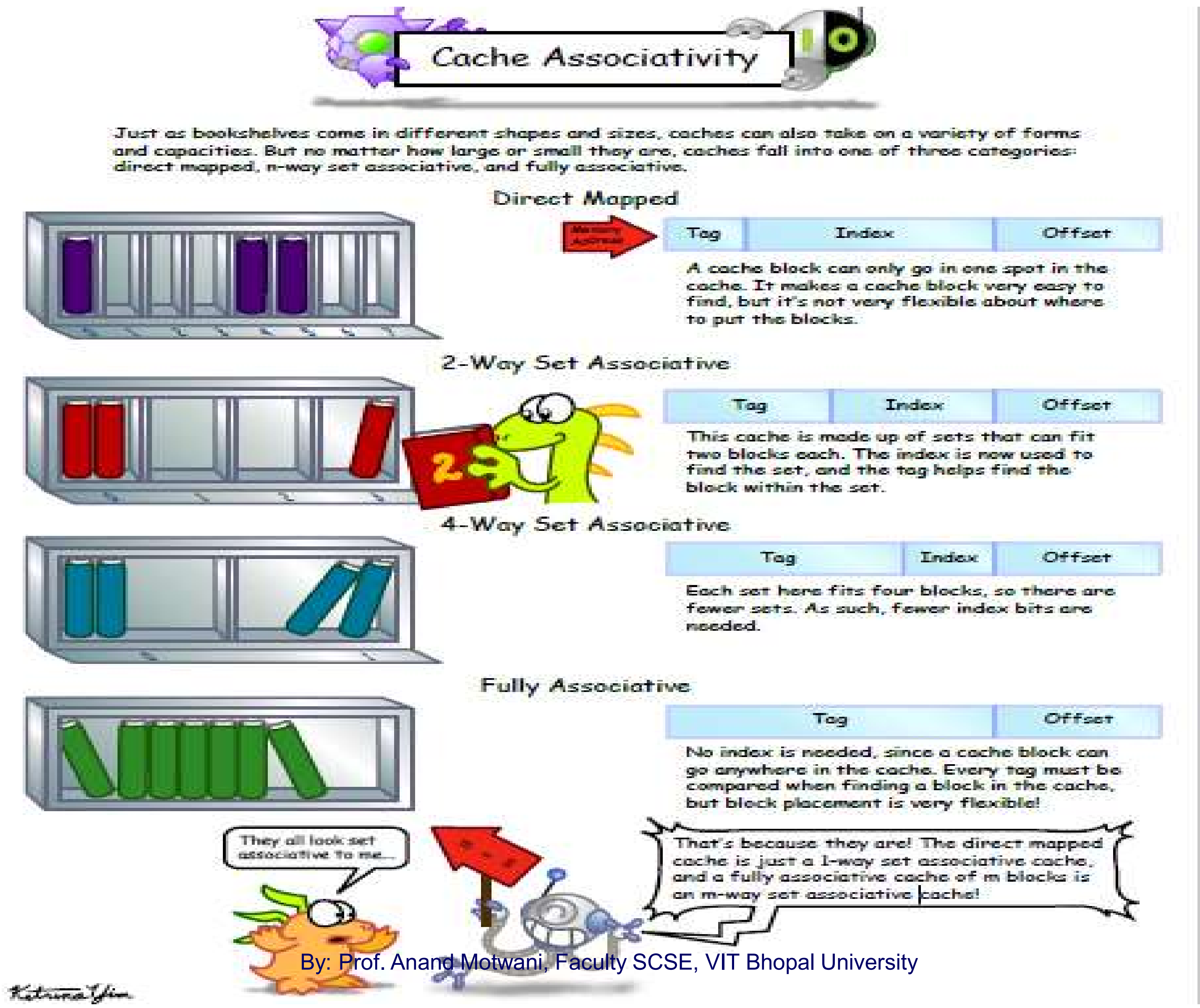


Set Associative Cache

It is also called as m-way set associative cache, where m is number of blocks and all blocks fit into one set.

Instead of using cache index, compare the tags of all cache entries in parallel.

The cache size is not determined by the address size.



Cache Associativity

Just as bookshelves come in different shapes and sizes, caches can also take on a variety of forms and capacities. But no matter how large or small they are, caches fall into one of three categories: direct mapped, n-way set associative, and fully associative.

Direct Mapped

Tag	Index	Offset

A cache block can only go in one spot in the cache. It makes a cache block very easy to find, but it's not very flexible about where to put the blocks.

2-Way Set Associative

Tag	Index	Offset

This cache is made up of sets that can fit two blocks each. The index is now used to find the set, and the tag helps find the block within the set.

4-Way Set Associative

Tag	Index	Offset

Each set here fits four blocks, so there are fewer sets. As such, fewer index bits are needed.

Fully Associative

Tag	Offset

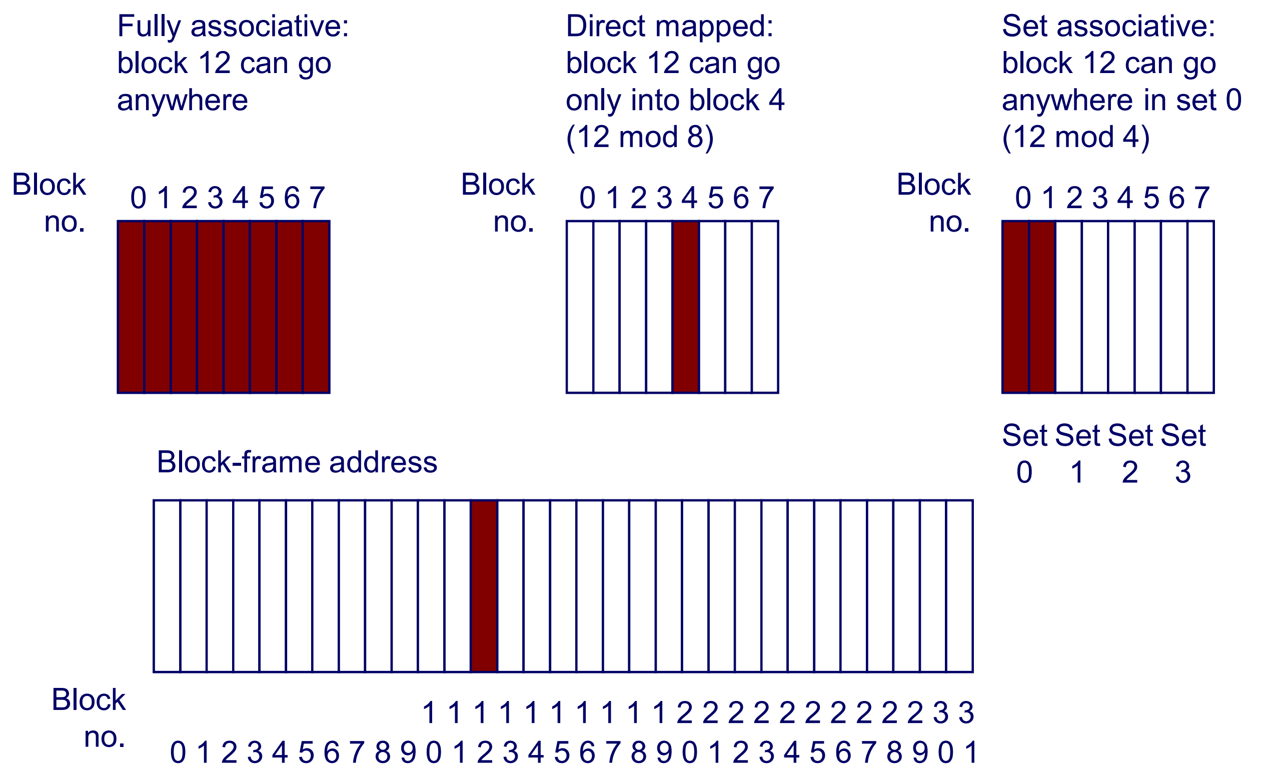
No index is needed, since a cache block can go anywhere in the cache. Every tag must be compared when finding a block in the cache, but block placement is very flexible!

They all look set associative to me...

That's because they are! The direct mapped cache is just a 1-way set associative cache, and a fully associative cache of m blocks is an m-way set associative cache!

Associative Caches

- Block 12 placed in 8 block cache:
  - Fully associative, direct mapped, 2-way set associative
  - S.A. Mapping = Block Number Modulo Number Sets



Numerical question

A block-set associative cache memory consists of 128 blocks divided into four block sets. The main memory consists of 16,384 blocks and each block contains 256 eight bit words.

1. How many bits are required for addressing the main memory?
2. How many bits are needed to represent the TAG, SET and WORD fields?

Given Information:

Given-

Number of blocks in cache memory = 128

Number of blocks in each set of cache = 04

Main memory size = 16384 blocks

Block size = 256 bytes

1 word = 8 bits = 1 byte

= 2^{22} Bytes

Solution

Number of Bits in Block Offset-

We have-

Block size

= 256 bytes

= 2^8 bytes

Thus, Number of bits in block offset or word = 8 bits

Number of Bits in Set Number-

Number of sets in cache

= Number of lines in cache / Set size

= 128 blocks / 4 blocks

= 32 sets

= 2^5 sets

Thus, Number of bits in set number = 5 bits

Solution (cntd.)

Number of Bits in Tag Number-

Number of bits in tag

= Number of bits in physical address – (Number of bits in set number + Number of bits in word)

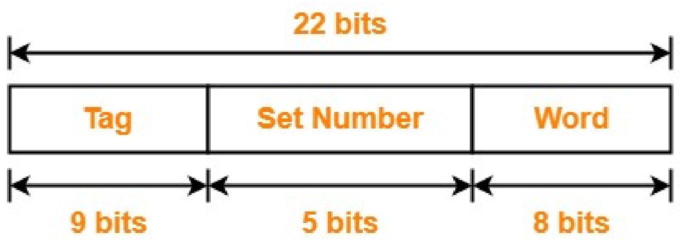
= 22 bits – (5 bits + 8 bits)

= 22 bits – 13 bits

= 9 bits

Thus, Number of bits in tag = 9 bits

Thus, physical address is-



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

Replacement Algorithms (1) Direct mapping

- No choice
- Each block only maps to one line
- Replace that line

Replacement Algorithms (2) Associative & Set Associative

- Hardware implemented algorithm (speed)
- Least Recently used (LRU)
- e.g. in 2 way set associative
  - Which of the 2 block is lru?
- First in first out (FIFO)
  - replace block that has been in cache longest
- Least frequently used
  - replace block which has had fewest hits
- Random
  - Almost as good as LFU and simple to implement

Write Policy

- Must not overwrite a cache block unless main memory is up to date
- Multiple CPUs may have individual caches
- I/O may address main memory directly

Write through

- All writes go to main memory as well as cache
- 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!

Write back

- Updates initially made in cache only
- Update bit (dirty bit) for cache slot is set when update occurs
- If block is to be replaced, write to main memory only if update bit is set
- Other caches get out of sync
- I/O must access main memory through cache
- N.B. 15% of memory references are writes

Line Size

- Retrieve not only desired word but a number of adjacent words as well
- Increased block size will increase hit ratio at first
  - the principle of locality
- Hit ratio will decreases as block becomes even bigger
  - Probability of using newly fetched information becomes less than probability of reusing replaced
- Larger blocks
  - Reduce number of blocks that fit in cache
  - Data overwritten shortly after being fetched
  - Each additional word is less local so less likely to be needed
- No definitive optimum value has been found
- 8 to 64 bytes seems reasonable
- For HPC systems, 64- and 128-byte most common

Multilevel Caches

- High logic density enables caches on chip
  - Faster than bus access
  - Frees bus for other transfers
- Common to use both on and off chip cache
  - L1 on chip, L2 off chip in static RAM
  - L2 access much faster than DRAM or ROM
  - L2 often uses separate data path
  - L2 may now be on chip
  - Resulting in L3 cache
    - Bus access or now on chip…

CAT– II

Topics that will be asked:

- **Memory hierarchy**
- **RAM organization**
- **Locality concept**
- **Usage of memory**
- **Concept of Memory Mapping: Direct, Associative and Set Associative.**
- **Disk Organization**