4 Memory hierarchy questions

- Block placement
- Identification
- Replacement
- Write strategy

Q-1

- Where can a block be placed in the upper level?
  - One place
    - Direct-map
    - Block num = (block addr) MOD (num of blocks in cache)
  - Anywhere
    - Fully associative
  - In a restricted number of locations
    - Set associative
    - Block num = (block addr) MOD (num of sets in cache)

Direct map = 1-way set associative
Fully associative = m-way set associative, where m is the number of blocks in the cache

ex

- Cache has 8 blocks
- Memmory block number 12 goes where?
  - Direct map = 12 % 8 = 0 ==> it goes in block-0
  - Fully associative = Anywhere between 0 and 7
  - 2-way Set associative = 12 % 4 = 0 ==> it goes in set-0
    - Cache has 4 sets (8 div 2)
    - Each set has 2 ways: way-0 and way-1

Q-2

- How is a block found if it is in the upper level?
- In comparion
  - Offset not used : all blocks match this
  - Index not used : block placed using value
    - Therefore something there always
- As associativity increases

  Associativity increases

  => blocks per set increases

  => index size decreases

  => tag size increases
- Fully associative cache has no index field

Terms and formulae

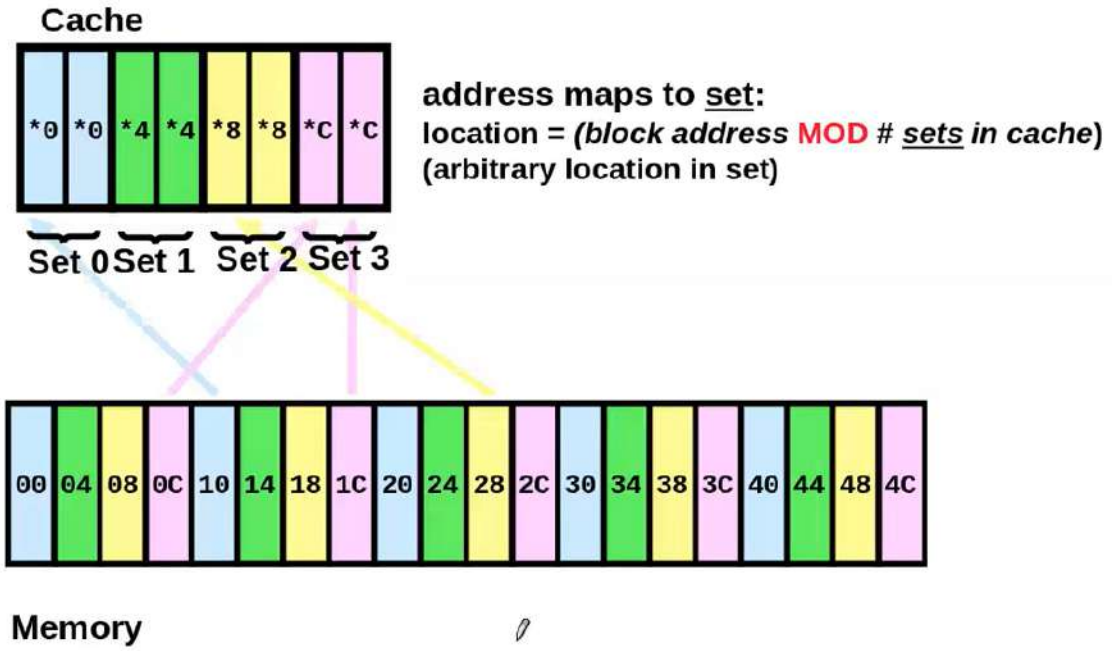
- Capacity = C bytes (e.g. 1 KB)
- Block size = B bytes (32 B)
- Byte select bit = 0 ... log(B)-1 [0..4]
- Number of blocks = C/B [1 KB/32B = 32]
- Address size = A [32 bits]
- Cache index size = I = log(C/B) [log 32 = 5]
- Tag size = A – I – log(B) [32-5-5 = 22]
-
- Block-0 in cache is mapped by main memory addresses: 0, 32, 64, ...
- i.e. if bits <9,...5> are same in address then all such map to same block
- Therefore, how to find which one if in the cache?
  - Tag- field is compared to check this
  - If match => Hit

2-way set assoc

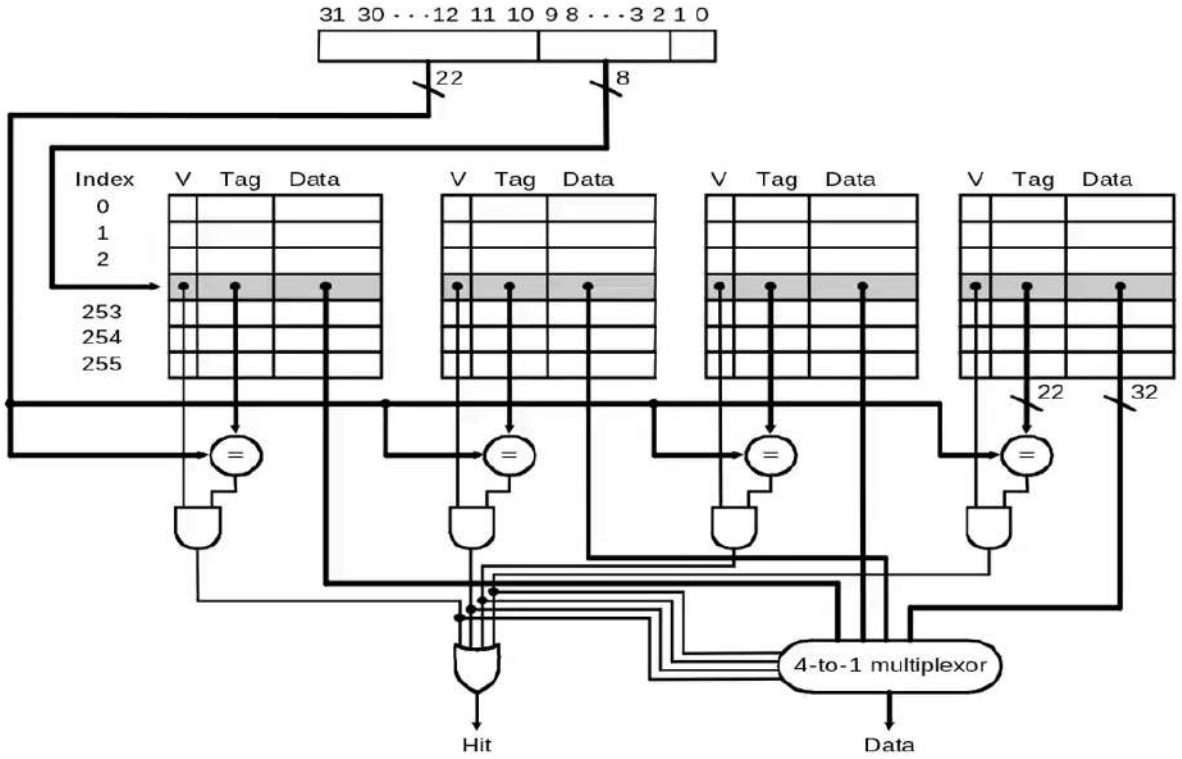
- N-way set associative = N entries for cache index
  - Set has N blocks and when placing block any location can be used
  - N direct mapped caches operated in parallel
  - N is typically 2 to 4
- Cache index select a set
- The two-tags in the set are compared in parallel
- Data selected from matching tag

- Disadvantage of N-way assoc
  - N-way vs Direct map => N comparators vs 1
  - Extra MUX delay
  - Data comes after hit/miss decision
- Direct map
  - Data avaiable before hit/miss decision
  - Possible to assume hit and continue
  - Recover if miss

EX: 2-way set-assoc



4-way set assoc design

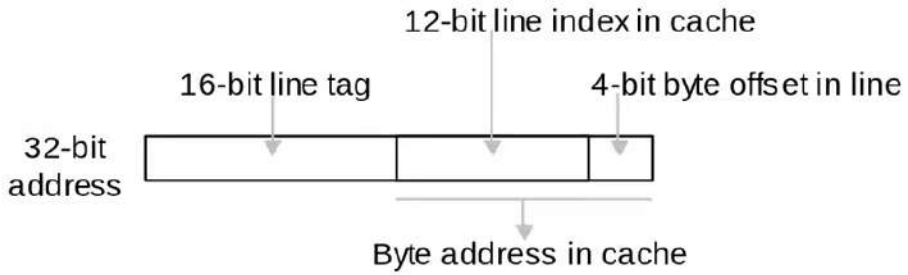


Numerical ex – Direct map

Show cache addressing for a byte-addressable memory with 32-bit addresses. Cache line W = 16 B. Cache size L = 4096 lines (64 KB).

Solution

Byte offset in line is $\log_2 16 = 4$ b. Cache line index is $\log_2 4096 = 12$ b. This leaves 32 – 12 – 4 = 16 b for the tag.



Components of the 32-bit address in an example direct-mapped cache with byte addressing.

Numerical ex – set assoc

Tag 9 bit	Set 13 bits	Word 2 bit

Use set field to determine which set of cache lines to look in (direct)

Within this set, compare tag fields to see if we have a hit (associative)

e.g

Address	Tag	Data	Set number
FFFFFC	1FF	12345678	1FFF
00FFFF	001	11223344	1FFF

Same Set, different Tag, different Word

Example ...

- Given Tag=9 bits, Set=13 bits, Word=2 bits

  Given address $FFFFFD_{16}$

- What are values of Tag, Set, Word?

  First 9 bits are Tag, next 13 are Set, next 2 are Word

  Rewrite address in base 2: 1111 1111 1111 1111 1111 1101

  Group each field in groups of 4 bits starting at right

  Add *zero bits* as necessary to leftmost group of bits

  0001 1111 1111 *0001* 1111 1111 1111 *0001*

  → 1FF 1FFF 1 (Tag, Set, Word)

Q-3

- Which block should be replaced on a cache miss?
- Direct map
  - Simple hardware
  - One choice only
- Fully or set assoc
  - Many options
  - (1) Random
    - Use pseudo random generator
      - Reproducible values from seed --> good to debug
  - (2) LRU (Least Recently Used)
    - Replae block that is unused for longest time
    - Locality of reference concept used
    - But hardware complex as number of blocks increase
  - (3) FIFO (First-in First-out)
    - LRU approximated to FIFO
    - Determine oldest block rather than LRU
  - (4) Optimial algorithm
    - Replace the block that will not be used for the longest time (must know the future)

Q-4

- What happens on a write?
  - Reads dominate processor cache accesses --> Instruction are reads !
- Ex: write are 7% of total memory traffic and 28% of the data cache traffic
- MIPS ex LD = 26%, ST = 10%
- Write Mem traffic = 10% / (100% + 26% + 10%) = 7%
- Data cache traffic = 10% / (10% + 26%) = 28%
- Therefore make common case fast --> make reads fast

Read block

- Read block parallel with tag read+compare
  - If block is hit then send read data to processor
  - If it is miss then ignore the value read
- This is NOT possible for writes
  - Modifying block can begin after tag comparison complete => Hit
  - Writes take longer
  - Write small = 1 to 8 bytes but read can be whole block and no hardm
    - Selective writes in a block takes time

Write options

- Write through
  - When we update cache, data in main memory is old, i.e. inconsistent
  - Write to both cache and lower level memory
- Write back
  - Update only cache. At replacement write to lower level
  - Dirty bit to check if block is modified
    - =1 write to lower level
    - =0 clean block. No need to write

Write back: Advantage

- Writes occurs at speed of cache
- Multiple writes to same block
  - => one final write to lower level
- Needs less memory bandwidth
  - Good for multiprocessors (sharing RAM)
- Saves power (as less accesses to RAM and interconnect)
  - Good for embedded applications

Write through: Advantage

- Easy to implement
- On read-miss
  - -> block replacement
  - -> write the victim to next level is not needed. Simply change the 'v' bit
- Cache always clean
- Next lower level has current copy of data
  - => coherence maintained
  - => good for multiprocessors
- In case of multi-level caches,
  - Write through goes to next lower level and not all the way to main-memory
- Write stall: processor has to wait during write through
- Write buffer: write stall is optimised by use of write buffer
- Processor can continue after putting data in write buffer ... overlap processor execution with memory update

What happens on a write-miss?

- Data is not needed on a write-miss
- (1) write allocate
  - Block allocated on miss
  - Write data
  - Same as read miss
- (2) No-write allocate
  - Write-miss does not affect cache
  - Block directly modified at lower level memory

IITG

What happens on a write-miss? ...

- We can use any write miss strategy with write-back/write through cache
- Normally,
  - Write back caches
    - Use write allocate
    - Maybe future write to same block
  - Write through caches
    - Use no-write allocate
    - Anyway we write to cache + lower-level
    - So why bring the block to cache !

Example

- Assume full assoc, wr-back with lot of empty entries
- Sequence of access: (1) WR mem[100], (2) WR mem[100], (3) RD mem[200], (4) WR mem[200], (5) WR mem[100]
- What are number of hits and miss using wr-allocate VS no-write allocate
- ANS:
- No-write allocate
  - (1) and (2) Miss as [100] not read into cache
  - (3) is miss as 200 not in cache
  - (4) Hit as 200 will be in cache now
  - (5) miss as 100 still not there
  - 1 Hit, 4 Miss
- Write-allocate
  - (1) Miss, But 100 will be brought in cache
  - (2) Hit, (3) Miss (4) Hit, (5) Hit
  - 3 Hits, 2 Miss

Cache performance

AMAT = Hit time + Miss rate x Miss Penalty

- Better measure of memory hierarchy performance is Average Memory Access Time

- Ex: Find if split (I$+D$) better than unified cache.

  Which has lower miss rate and lower AMAT? 16KB I$ + 16KB D$ or 32KB unified.

  Percentage of instr ref = 74% (out of all mem ref). Misses per 1000 instr: I$ = 3.83, D$=40.9, Uni=43.3.

  Hit = 1 cycle, Miss penalty = 200cy. LD/ST hit takes 1 cycle extra on unified as single port in case of simultaneous access by two instr. Ignore write stalls. 36% instr are data transfer

- ANS=

Example ... (ans)

- Miss rate = misses/total access = (misses/instr) / (access/instr) = (missses per 1000 instr/ 1000) / (accesses/instr)
- Out of total instr 36% data transfer (given)
- Out of total mem ref 74% are instr, so 26% are data transfer instr
- Miss rate I$ = 3.82/1000/1.0 = 0.004
- Miss rate D$ = 40.9/1000/0.36 = 0.114
- Miss rate Uni = 43.3/1000/(1.0+0.36) = 0.0318
- Overall miss rate for split cache
- Miss rate (split) = 74% x 0.004 + 26% x 0.0114 = 0.0326
- Unified cache has slightly low miss rate

Example .... (ans)

- Finding AMAT
- AMAT = %instr (hit time + instr miss rate x miss penalty) + %data (hit time + data miss rate x miss penalty)
- AMAT_split

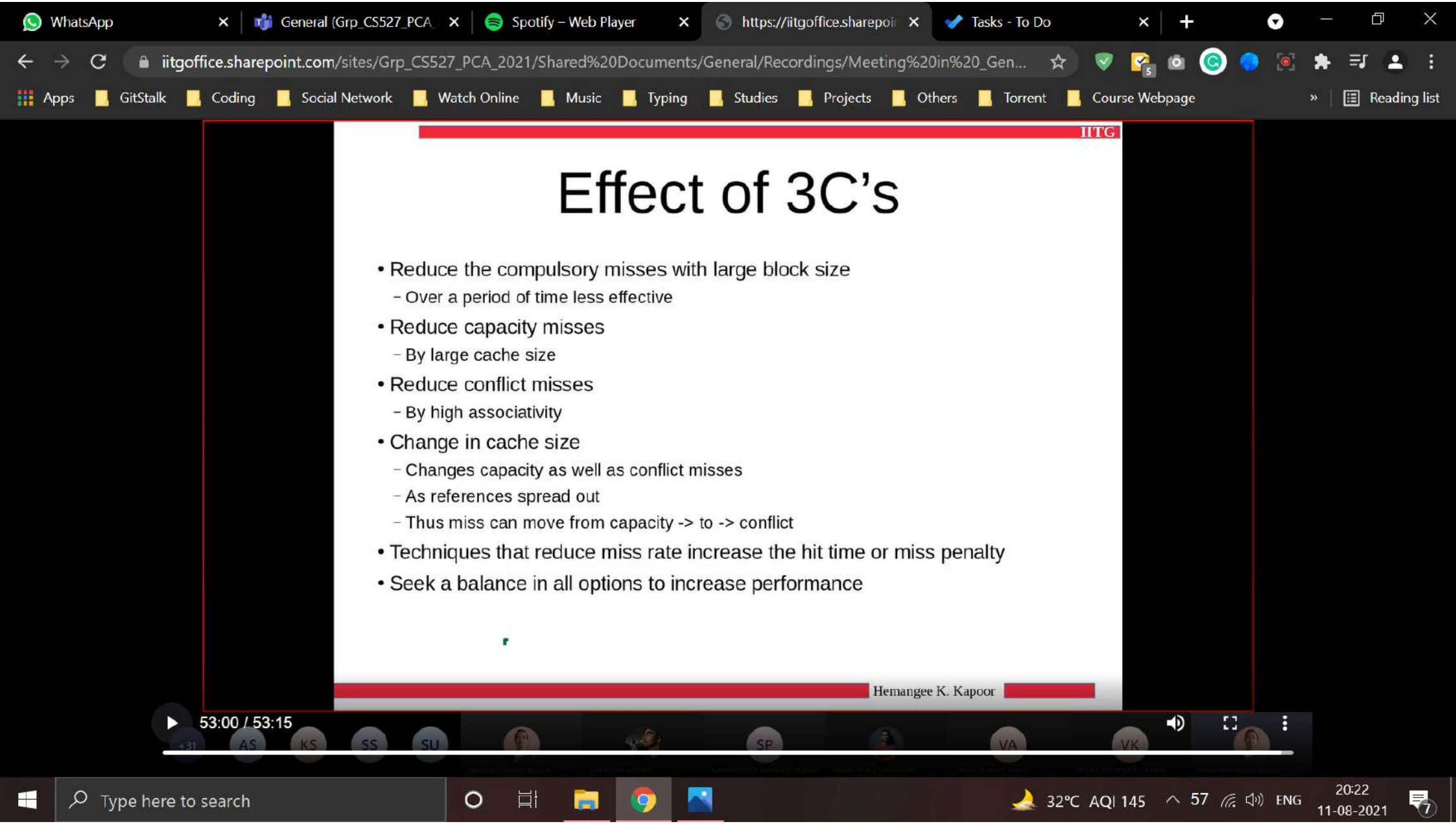
  = 74% (1+0.004x200) + 26%(1+0.0114x200) = 7.52
- AMAT_uni

  = 74% (1+0.0318x200) + 26%(1+1+0.0318x200) = 7.62

- Therefore, Split cache (having 2 ports: I$, D$) has better AMAT even if it has poorer miss rate

Causes of misses

- 3C's model
- Classifying misses
- (1) Compulsory
  - First access to a block not in cache called cold-start misses or first reference miss
  - Misses even in an infinite cache
- (2) Capacity
  - Cache cannot contain all needed blocks, Block discarded and retrieved later
  - Misses in a fully associative cache
- (3) Conflict
  - Placement in set-associative or direct map
  - More than 1 block maps to same set/location
  - Collision misses or interference misses
  - Misses in N-way assoc size cache
- (4) Coherence
  - More recent 4th C
  - Due to cache flishes to keep multiple cache coherent in multi-processors


WhatsApp
General (Grp_CS527_PCA
Spotify – Web Player
https://iitgoffice.sharepoi
Tasks - To Do
iitgoffice.sharepoint.com/sites/Grp_CS527_PCA_2021/Shared%20Documents/General/Recordings/Meeting%20in%20_Gen...
Apps
GitStalk
Coding
Social Network
Watch Online
Music
Typing
Studies
Projects
Others
Torrent
Course Webpage
Reading list
IITG
Effect of 3C's
• Reduce the compulsory misses with large block size
– Over a period of time less effective
• Reduce capacity misses
– By large cache size
• Reduce conflict misses
– By high associativity
• Change in cache size
– Changes capacity as well as conflict misses
– As references spread out
– Thus miss can move from capacity -> to -> conflict
• Techniques that reduce miss rate increase the hit time or miss penalty
• Seek a balance in all options to increase performance
Hemangee K. Kapoor
53:00 / 53:15
Type here to search
32°C AQI 145
57
ENG
20:22
11-08-2021