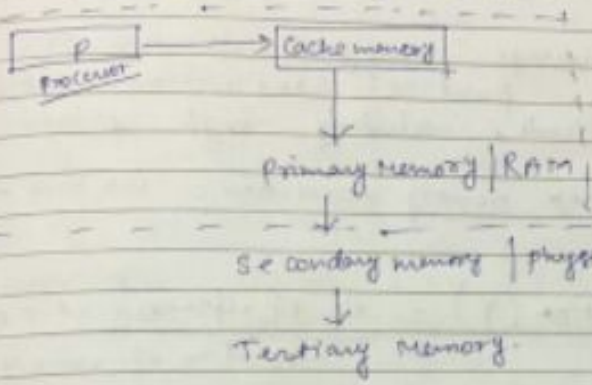
Thread level Parallelism
Memory Hierarchal design.

23/02

Memory Hierarchy design



Cache hit
cache miss

Memory mapping is done by virtual memory.

RISC and CISC

CU

Hardware Microprogram control.
(combination of Software + Hardware)

Prob. locality of reference (b/w P and Cache)

Evaluation of cache performance and optimization of cache.

The average memory access time is calculated as a part of Instruction execution at all levels, including high & low.

average memory access time = Hit time + miss rate * miss penalty

$$\text{Hit ratio } (\gamma) = \frac{\text{No. of refrences found in cache}}{\text{total no. of memory refrences}}$$

Miss rate is the faction of memory references not found in cache.

$$\text{Miss rate} = \frac{\text{No. of misses}}{\text{references}}$$

Miss Penalty is the additional time required because of a 'miss'.

CPU time in the presence of misses, the CPU time can be considered with the following operational entities :-

CPU time = (CPU execution clock cycles + memory stall clock cycles) * clock cycle time

There are 16 Cache optimizations into 4 categories:-

1) Reducing the miss Penalty
   (a) Multilevel Caches
   (b) Critical word first
   (c) Read miss before write miss
   (d) Merging Right buffers.
   (e) (M) No. of victim caches

2) Reducing the miss rate
   (a) ~~Changing~~ Larger block sizes
   (b) Larger Cache Size
   (c) Higher associativity
   (d) Pseudo activity (Pseudo associativity and compiler optimization)

3) Reducing the miss penalty and miss rate via Parallelism: —
   a) Non-blocking Caches
   b) Hardware Pre-fetching
   c) compiler Pre-fetching

4) Reducing the time to hit in the Cache
   (a) Arranging Small and simple Caches.
   (b) Avoiding Address translation using VM
   (c) Pipeline Cache access

Book – Patterson

Locality of reference

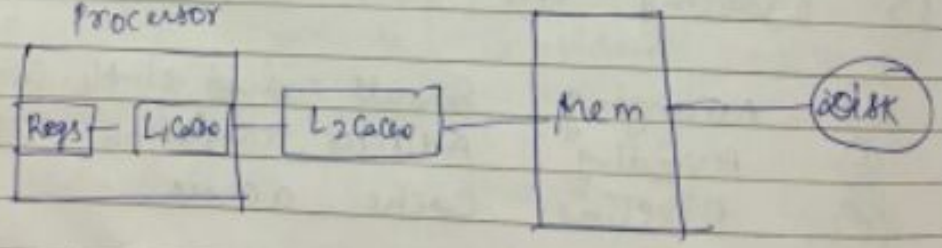
Many of times, a Computer executes instructions from consecutive memory location serially until the information happens to be a branch and branch takes a flow in a different locality in the memory.

with the advancement in the execution of a Program, slowly changes with time. This is referred as Spatial locality.

It is possible that same location may be accessed again in short intervals of time or a nearby location may be accessed in another refrence. This is refered as Principle of locality.

Various techniques to reduce miss Penalty.

1) Arranging Multilevel Caches.



In order to rate the miss penalty, the avg. mem. access time for 2 level caches is calculated as

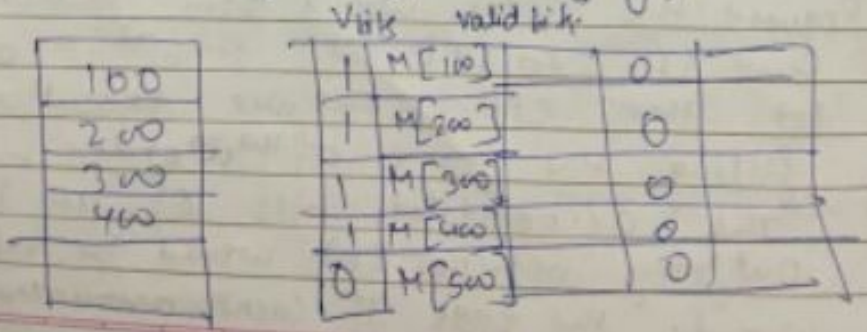
Avg mem Acc time = Hit time of L1 + Miss rate of L1 x Miss Penalty L1

fetch the words in normal order but as the request word of block arrives. Send it to the CPU and let this CPU continue execution.

c) Giving Priority to read misses over write

with a right through cache, the most imp improvement is right buffer of the proper size (i.e. 2).
The simplest way out of writting is for the read miss to wait until, the right buffer is empty.

d) Merging Right buffer → This technique involves right buffers with an increase in time efficiency. write through caches rely on write buffers.
If the buffer contains other modified blocks, the addresses can be checked to see if the address of the new data matches the address of valid write buffer entry. If so new data are combined with entry. (Right Merging).

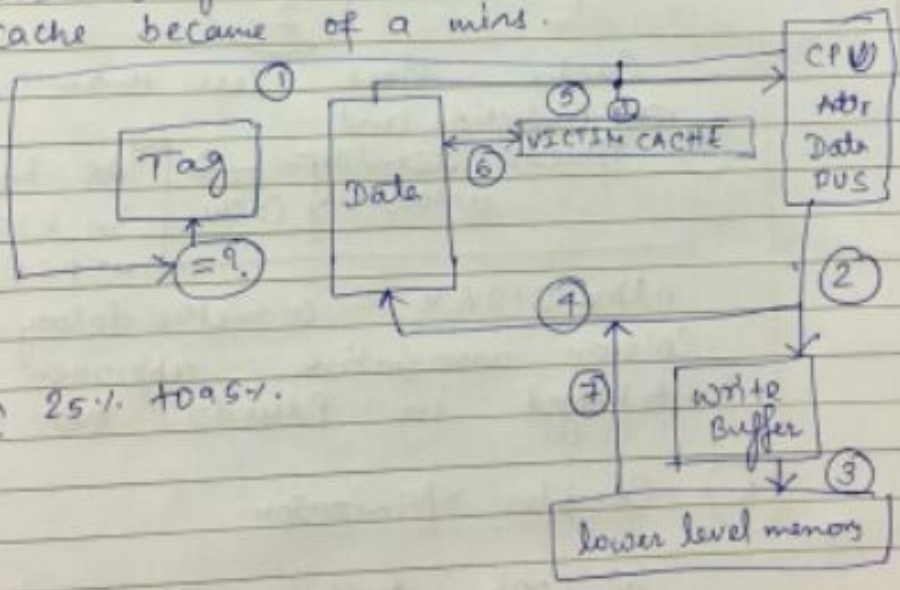


If there is no multiple writes for same memory location

b) No. of victim caches.

The cache which is suffering from miss operation.

① one approach to lower miss penalty is to remember what was discarded in case it is needed again. Since the discarded data has already been fetched, it can be used again at small cost. This is called recycling. with this activity related cache is referenced as a victim cache that which contains only blocks that are discarded from a cache because of a miss.



maximum 25% to 95%.

② Reducing the miss rate

ⓐ arranging large blocks with required size
↓
It is a concept of temporal and spatial locality.

ⓑ Arranging larger caches

ⓒ Higher associativity → It is a concept of direct mapped cache of size N has about the same miss rate as 2 Way set associativity Cache of size $\frac{N}{2}$. This helped for Cache sizes less than 128 KB.

④ Way Prediction and Pseudo ~~activity~~ associativity Cache. →
As per the reality in RISC architecture MIPS R4300 and Alpha 21264, a methodology named Column associative approach has been deployed in Pseudo Set.

⑤ Compiler optimization

ⓐ Loop interchange
ⓑ Blocking

Array can't be accomodated in Cache memory

Loop interchange → Some programs have nested loops that access data in memory in non-sequential order/random order. Simply exchanging the nesting of the loops can make the code access flexible. Assuming that the arrays don't fit into the cache. This technique reduces misses by improving spatial locality.

Note: Reordering maximizes uses of data in a cache block before it is discarded.

Blocking ——→ This optimization tries to reduce misses through improved temporal locality where the arrays are stored by row major order and column major order to transform the block interchange addresses.

Ranges → find size

Segment → virtual size.

Smallest – 1 Byte.

Largest → 2^{32} byte.

2^{16} to 2^{32} byte

Virtual memory → TLB

Translation looked ahead Buffer

physical frame address.

valid bit

dirty bits

- A TLB entry is like a cash entry where the Tag holds portion of the virtual address and data portion holds a physical frame number

Valid bit (0,1) and dirty bit.
TO change the Physical Page frame number OS must make sure that old entry is not in the TLB, otherwise System will not work properly.
The dirty ~~bit~~ bit means Corresponding page is dirty, not that address translation in TLB is dirty, nor a particular block in data cache is dirty. The OS resets these bits by changing the value in page table and then ~~invalidating~~ invalidating the corresponding TLB entry.

Page size Choosing the Page size is a difficult operation. So OS forces that a larger page size vs Smaller size could be created to favour larger size creation.
OS always favour large size creation.

Cache coherence.

Two Processors — Two different value.

↓

Same memory location.

	Processor Activity	Bus Activity	Content of CPU_A's Cache	CPU_B's Cache	Contents of Mem loc X
OP_1					0
OP_2	CPU_A reads X	Cache miss for X	0	—	0
OP_3	CPUB reads X	"	0	0	0
OP_4	CPUA writes X	Write front last X	1	1	1
OP_5	CPU_B reads X	—	1	1	1

37 Symmetric Shared memory architecture.

(a) SSMA consist of several processors with a single Physical memory

(b) SSMA machines usually support the caching of both shared and private data.
Private data is used by single processor while shared data is used by multiple processors.
When a private data item is cached, its location is migrated to the cache, reducing the average access time as well as the memory bandwidth required.
Note: Since no other processors uses the data, the program behaviour is identical to that in a uniprocessor.

An enforcing based protocol SBP (Snooping based protocol) maintain coherence for multiple processors where each cache that has a copy of data from a block of physical memory also has a copy of sharing status of the block. and no centralized state has been kept. The caches are usually on a shared memory bus and all cache controller monitor (snoop) on the bus to determine whether or not they have a copy of a block ~~that is~~ that is ~~required~~ requested. on the buses.

Implementation The serialization of access enforced by the bus also ~~enforces~~ forces serilization of a writes. Since when two processors compete to write to the same location, one must obtain bus access before the other. The first processor to obtain bus access will cost ~~cost~~ other processors copy to be 'invalidated, causing writes to be strictly serialized.
one implication of this scheme is that a write to a shared data item can't complete until it obtains bus access.

Directory Based Protocol (DBP)

In this the sharing status of a block of Physical memory is kept in just one location called the directory.

The drawbacks of (Symmetic-SMA), ~~compare~~
① Compiler mechanism for ~~transfers~~ software cache coherence are very limited
② without cache coherence the multiprocessor loses advantage of being able to fetch and use multiple words in a single cache block.
3> The mechanism for tolerancy the latency such as Pre-fetch are more useful when they can fetch multiple words such as a cache-block.

1> A direc
that ma
directo
of blo
TO P
bottle
along
direc
location
go
A
tha
alw
is
av
TO
writ
dire
of
with the
(a) Sho
on
Cach
u
(b) Unc
a cop
(c) Ex
ca
1.
(2)

1) A directory keeps the state of every block that may be cached. Information in the directory includes which caches have the copy of block whether it is dirty and so on. To prevent directory from becoming the bottle neck, directory entries can be distributed along with the memory so that different directory accesses can go to different locations just as different memory request go to different memories.
A directory distributed retains characteristic that A sharing status of block is always in a single-node-location. This property is what allows coherence protocol to avoid broadcast.
To implement the operations like read miss, handling a write and Cleaning the cache block, the directory must be able to track the state of each cache block. In simple directory based protocol, there states could be previous with the following:—

(a) Shared Operations; It illustrates that whether one or more processors have the blocks cached and the value in memory is up to date or not.

(b) Uncached Operations: No processor has the a copy of cached block.

(c) Exclusive: Exactly one processor has a copy of cache block and it has written the block, so the memory copies is out of date. This processor is called owner of the block.
Owner Node / Home node.

~~chapter → 4, 5~~

1, 3, 4, 5,
ch 2

1> chapter - 4
2> chapter - 5

Multithreading.

It

(1) fine grained multithreading: → Switches b/w threads on each ~~or~~ instruction causing the execution of multiple threads to be interleaved. This interleaving often done in round robin fashion. Skipping any threads that are stalled/problematic at that particular time.

(2) Coarse grained multithreading → It is alternative of fine grained multithreading. It switches the threads only on costly stalls, such as level 2 cache misses. This change relieves the need to have thread switching be essentially free and is much less likely to ~~stop~~ slow the processor down. (means instruction from other threads will only be issued when a thread encounters a costly stall).

3) Simultaneous Multithreading → It works on instruction level parallelism.

04/2/2018:

Thread level Parallelism (TLP) (multiprocessors)

↓ s/w routine

↓ compiler

↓ Instruction exe.

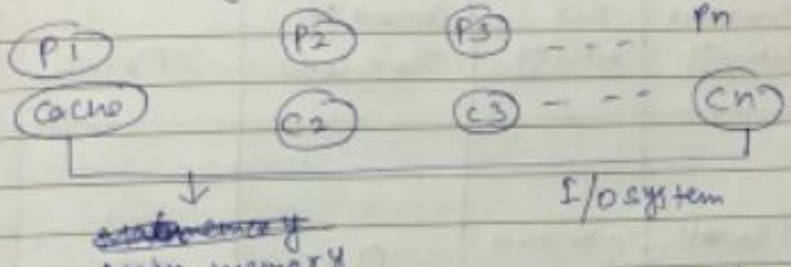
↓ sharing of cache

↓ Thread level Parallelism

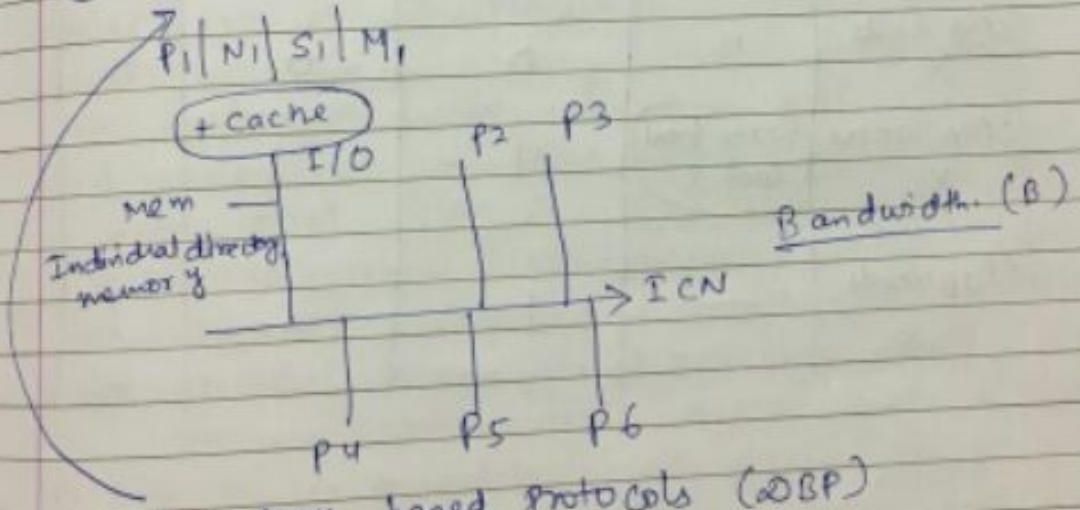
↓ Hit / Miss

Issues in SMA:

① Shared memory architecture (SMA) → Snooping based protocol



② Distributed shared memory architecture (D-SMA)



Directory based protocols (DBP)

Write Invalidate Protocol (WIP)

Write validate Protocol (WVP)

11 Buffer management

Miss Penalty L_1 = Hit time L_2 + Miss rate L_2 × Miss Penalty L_2

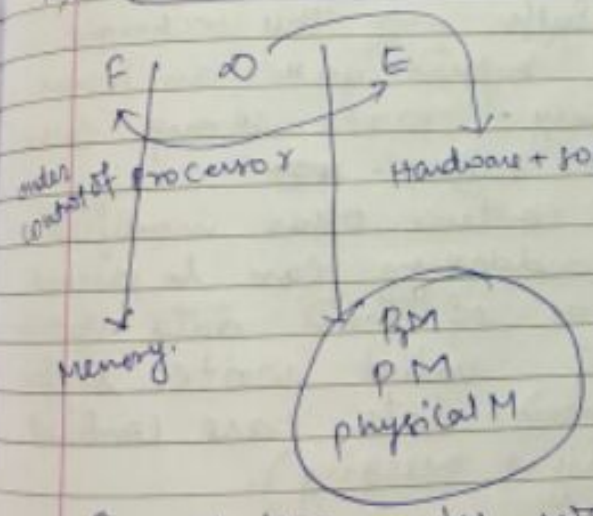
~~Do Avg~~

total miss rate → $\frac{\text{The no. of misses in the cache}}{\text{The total no. of memory accesses to that cache.}}$

global miss rate → $\frac{\text{The no. of misses in the cache}}{\text{Total no. of memory access generated by the CPU.}}$

Avg memory stalls Per Instruction Can be calculated with misses Per instruction L_1 × Hit time L_2 + (wrapped word) misses Per Instruction L_2 × Miss penalty L_2

12) Critical word first and Early restart



F	Store	D	OPr	E
I_1	X	X	X	X
I_2	I_1	X	X	X
I_3	I_2	I_1	X	X
I_4	I_3	I_2	I_1	X
I_5	I_4	I_3	I_2	I_1

Request the miss word from memory and Send it to CPU as soon as it arrives; let the CPU continue execution while filling the rest of the words in the block. The accomodated bits of an instruction (Outgoing instruction IE) would be referenced w.r.t. the tags of cache associated with its corresponding blocks in memory. Which is called wrapped fetch/Req. word first.