Set 2

Problem 1) Define the following terms –

a) Cache Line clean	b) Point of Unification	c) Point of Coherency
d) Non-blocking Cache	e) VIPT Caches	f) Page Fault

a) Cache Line Clean – In modern day cores there are many levels of cache organization. Hence when a particular data is present in the processors cache but is not present in the main memory, the cache line is said to be dirty. The operation to copy this data back to the main memory is referred to as cleaning the cache. Hence, when the data is copied from cache to main memory from a dirty cache line, such an operation is termed as a cache line clean operation.

b) Point of unification – Most of the processors these days implements different caches for the instruction accesses and the data accesses. Usually, the level 1 cache is different for data and instruction side which gives the designers freedom to place instruction close to instruction fetch unit and data cache close to data manipulation unit. The point of unification is thus known as that level of cache where both the data and instruction are present together. It is usually the level 2 cache system.

c) Point of Coherency – In a multicore system there are many processors sharing the data and hardware resources. These days most of the processors have an internal L1 and L2 cache but the L3 cache is shared across the cores. Since this data should be visible to all the cores, all the time it is known as the Point of Coherency of the system.

d) Non-blocking cache - Non-blocking caches makes sure that the cache keeps on accepting lookup requests for next instruction/data accesses when there is an on-going miss from the cache. This kind of technique is known as hit-under-miss i.e. the cache allows the next request to get a hit even when there is an on-going cache miss. The next-generation modern processors would typically be out-of-order which means that they wouldn't need to stall on any kind of dependency and/or cache/TLB misses. In such a case a non-blocking cache would definitely help the processor in executing more instructions/data accesses as it would be able to give hit information for the next requests as well.

e) VIPT Caches – VIPT caches stands for virtually indexed and physically tagged caches. Since the processors implement virtual memory system hence this means that we are effectively adding an extra stage of memory lookup which involves the address translation from virtual address to physical address. In order to save time and to better utilize the hardware, the cache are designed to virtually indexed but physically tagged. This ensures that while the address is being translated the cache could simultaneously be looked up and when the physical address is available the tag comparison can be made. This improves the system performance.

f) Page Faults – In a paging system, the hardware is capable of providing certain useful bits to the different pages of the physical memory. These bits could be related to the security state i.e. certain memory can only be accessed only when the processor is in secure mode and even certain bits can be used to determine whether certain location is read only or write only or non-executable. When any access which doesn't honour such conditions is made, a page fault is generated and the system access is transferred to the kernel to take appropriate measures.

Problem 2) Find the tag, index and offset values for a system having a 32 KB L1-cache with each block being 64 B long and a 36-bit address space under the following configuration -
i) The cache is organized as a direct mapped cache
ii) A 4-way set associative organization is chosen for the L1-cache
Which one would result in a faster cache lookup if the time taken for indexing is independent of the number bits used? Justify?
iii) Find the number of cache misses (for the direct mapped case) the following code would encounter if the array is stored in a row major form. Assume int takes 4 B of space in memory.

```
int a[64][64];
for (i = 0; i < 63; i++) {
  for (j = 0; j < 63; j++) {
    sum += a[i][j];
  }
}
```

i) System details -
L1 cache – 32 KB size
Block size – 64B
Direct Mapped
Using the block size and byte addressable memory
Number of offset bits = 6
Number of sets = 32KB/64B = 512 sets
Number of index bits = 9
Hence, the rest of the bits would be used for the tag = 36 – (6 + 9) = 21

ii) 4-way set associative
Using the block size and byte addressable memory
Number of offset bits = 6
Total number of sets = 32KB/64B = 512 sets
Number of sets in each way = 512/4 = 128
Number of index bits = 7
Hence, the rest of the bits would be used for the tag = 36 – (6 + 7) = 23

As it is given that the cache indexing operation is independent of the number of index bits thus it would depend on the tag comparisons. Since for a match it is essential that we should have a complete comparison between the physical address and the tag bits, it would take more time to compare a larger tag. Hence, it would mean that a direct mapped cache would be a bit faster as compared to the associative cache.

iii) The direct mapped cache mentioned in the above problem has the following configuration -
Size = 32 KB, Number of Sets = 512, Block size = 64B
As each access takes 4B of space, hence each cache block would contain 16 entries.
For the first access we would see a miss, followed by 15 hits. Then there would again be a miss followed by 15 hits. So on.
This means in all there would be a miss after every 16^{th} access. Number of misses = 64*64/16 = 256

Problem 3) In a memory system which contains a Translation lookaside buffer, page table and a cache, it can be said that such a system would encounter 3 different types of misses. What are they? Consider all the different but valid combination of these misses and point out the scenarios under which these would occur.

The given memory system contains a TLB, a page table – this suggests that the memory system implements virtual memory. The presence of cache reflects that the system would definitely suffer from a cache miss. The misses due to TLB based virtual memory system is a TLB miss and a page fault. Therefore the three misses would be TLB miss, Page Table miss and a cache miss.

The possible valid combinations of such a system is shown in the table below with the scenarios under which such a combination would occur –

TLB	Page Table	Cache	Scenario
Hit	Hit	Miss	A cache miss for a request which hits in TLB. Page table won't be looked up on TLB hit
Miss	Hit	Hit	The access misses in TLB but hits in page table. The access would be re-issued after the physical address is calculated and would hit in cache
Miss	Hit	Miss	The access misses in TLB but hits in page table. The access would be re-issued after the physical address is calculated and would miss in cache
Miss	Miss	Miss	There is a TLB miss, followed by a page table miss (possibly a page fault) and then the access goes on and misses in cache
Hit	Miss	Miss	This doesn't seem a correct scenario. The entry is present in TLB but not in page table – this cannot happen
Hit	Miss	Hit	This doesn't seem a correct scenario. The entry is present in TLB but not in page table – this cannot happen
Miss	Miss	Hit	Again not possible since that data cannot be in cache if there isn't a valid physical address possible due to no pagin available

Problem 4) Consider a virtual memory system with 1 GB of physical memory and a page size of 4KB. The virtual memory system spans 32-bit of address space. The system also contains a 32KB direct mapped cache with 8B blocks. The cache is physically tagged and indexed. Answer the following with respect to the above system -
a) Find the virtual address mapping of the system
b) Find the physical address mapping of the system
c) Find the tag, index and offset bits of the VIPT cache. Assume that the size of Cache Tag RAMs when completely filled doesn't exceed 12 KB.
d) Let the time taken for a TLB lookup be 30ps, 50ps to activate the cache line from index and it takes 20ps to compare tags from cache and the physical address. Also it takes 300ps for the data to be available to the CPU from cache. Find how much time it will take for a memory access which has a TLB hit as well as a cache hit? Compare the time taken if the cache was Physically tagged and indexed?

a) Page Size = 4 KB
This means the offset bits would be equal to 12
The remaining bits would correspond to the Virtual Page number = 32 -12 = 20 bits

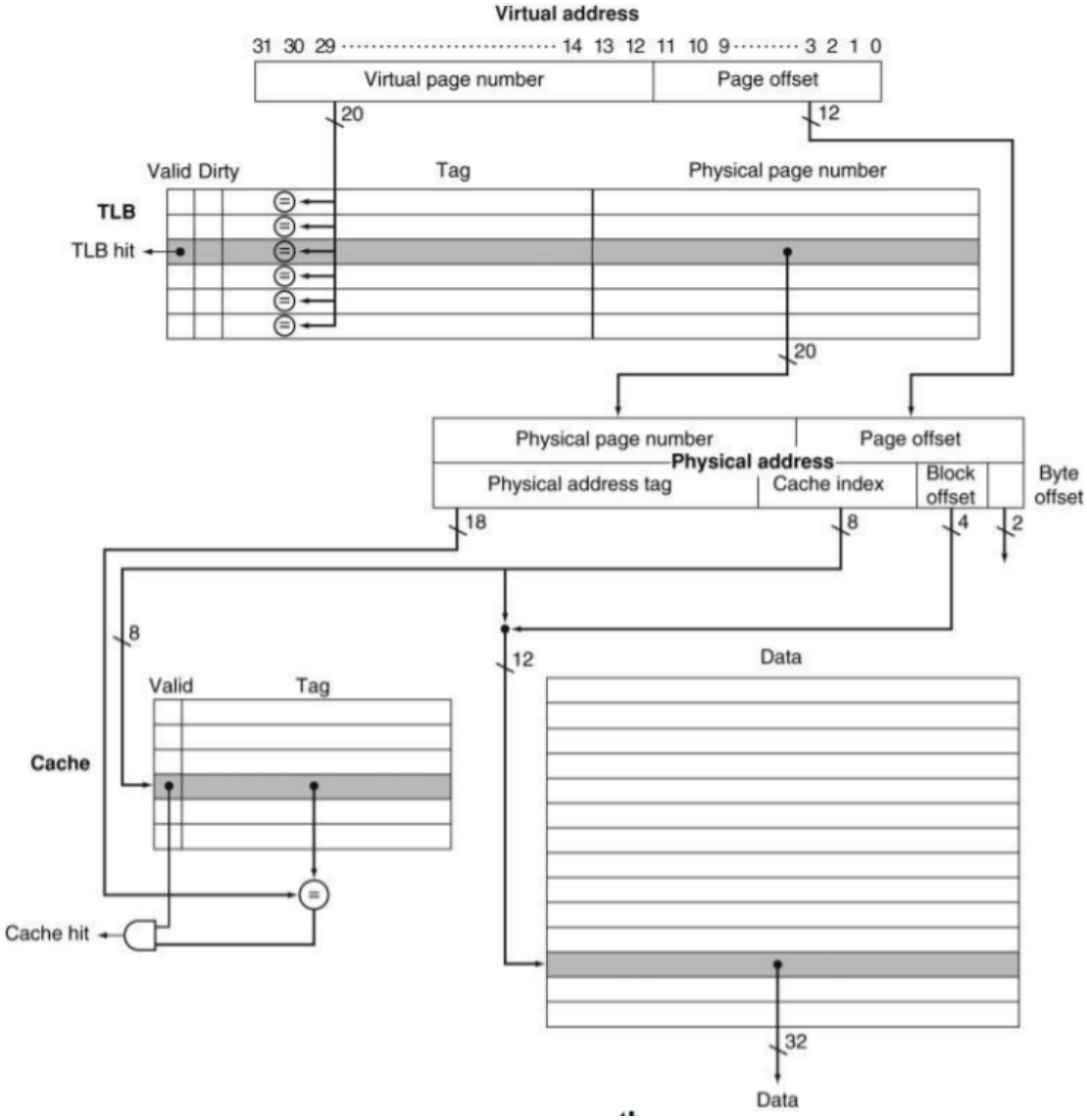
b) Physical memory size = 1GB
Physical address space spans 30-bits address space
Offset bits would be = 12 bits
The remaining bits would correspond to the Physical page number = 30 – 12 = 18

c) Since the cache is virtually indexed and physically tagged – the index bits and the offset bits would be derived from the virtual address and tag bits would correspond to the physical address.
Size of the cache = 32 KB
Direct mapped
Block size = 8 B
Offset bits = 3
Number of sets in the cache = 32 KB/8B = 4K = 4096 sets
Number of index bits required = 12
Maximum tag RAM size = 12 KB
Tag RAM would hold the tag bits. For a better comparison it should the tag should be as big as possible. For our system the best possible tag bits could be 27 bits (out of the 30 bit physical address the last 3 would be offset bits hence there would be no point comparing those).
The 27 bits would correspond to a maximum size of 27 * 4/8 KB = 13.5 KB
This exceeds the maximum allowable size. Hence we would need to adjust the tag bits. As per the size constraints 24-bits would satisfy our needs. (size = 24*4/8 KB = 12KB).
Hence the cache could be accesses as follows -
Tag = 24 (from the physical address)
Index = 12 (from virtual address)
Offset = 3 (from virtual address)

d) Since it is a virtually indexed cache and physically tagged before the comparison we can actually active the cache line. This would take 50ps. By that time the physical address would be available and hence 20 ps for the comparison and 300ps for the data transfer. Total time = 50 + 20 + 300 = 370 ns

If the cache was physically tagged we would need another 30ps before activating the index search. Hence the time taken would be = 400ps

Problem 5) The following figure describes a memory system with a TLB and a cache -



Comment on the following statements -

i) TLB organization?

ii) Cache organization?

iii) Number of comparators required for a cache lookup?

iv) Whether VPN or PPN is used for tag ram comparison?

v) The shaded area in the cache data RAM is?

vi) The size of the data block is?

vii) Do we need to wait for the translation to be completed before doing a cache lookup?

i) As per the figure above, TLB seems to Directly mapped.

ii) Same for cache as well, it is a direct mapped organization.

iii) Number of comparators required for cache lookup would be 1 only. Since the cache is direct

mapped hence there is a single way only.
iv) For the tag comparison the physical page number (from the physical address) is used for the comparison.
v) The shared area represents the data stored in the cache (as we are looking at the data ram). This data could be either a normal data access or an instruction access.
vi) The data ram stores 16B of data. This can be judged as the block offset bits are 4 in number.
vii)Since the cache uses physical address for index and tag lookup, we would need for the translation to get completed before the cache lookup.