# CS147 - Lecture 23

Kaushik Patra
(kaushik.patra@sjsu.edu)

1

- Virtual Memory – Concept
- Virtual Memory - Implementation

*Reference Books / Source:*

1) *Chapter 4 of 'Computer Organization & Architecture' by Stallings*
2) *Chapter 7 of 'Computer Organization & Design' by Patterson and Hennessy/*

Virtual Memory Concept ...

2

## Challenges in Multi-program Environment

- Single program environment enjoys entire resources of computing system.
- Multiprogram environment is the usual case now a days.
  - Resources are divided and partitioned for each program
    - temporal (processor)
    - spatial (memory)

3

• Single program is very uncommon in modern computing scenario. It is mostly found in hardware laboratory scenario where hardwares are tested with single program loaded into the memory. In that case the program enjoys entire resources available to single program.

• In multiprogram scenario, more than one programs (one of them is the operating system) are active in the system. This means resources like processor and memory is divided / shared between different active programs. Sharing can be temporal – it is like time sharing vacation rental home. Different parties are using same rental home at different time point of the year. In case of computer system, processor is shared temporarily. Each process / program takes its own turn to be executed on a processor. Operating system orchestrate execution of multiple programs on processor. Compare to that, spatial sharing is like sharing tent with friends in camping. All the persons are using same resource 'tent' at the same time finding their own space inside tent. In terms of computing system, memory if shared spatially. Multiple programs have their own share / space into the memory system. The other program may not violate memory space of other program.

## Challenges in Multi-program Environment

- Temporal and spatial partitioning is managed by operating system with hardware assistance.
  - Context Switching
  - Memory Management
- Let's talk about spatial (memory) partition
  - Two problems to solve
    - How to create and maintain the partition boundary?
    - How to manage with smaller share of main memory?

4

- In a sense OS takes the central role in managing temporal and spatial sharing with special architectural help. Process management (temporal sharing – with context switching) and memory management are two major areas in OS study.
- In architecture study we'll discuss mainly the spatial partition support – or memory partition support. We'll try to answer two major questions on how to create / maintain memory partition boundary and how to manage with much smaller share of memory in multi-program execution environment.

## Virtual Memory – Concept

- Two factors are unknown at program compilation time.
  - How much memory is going to be available at run time?
  - Which addresses are going to be available at run time?
- Virtual memory gives program illusion of enjoying entire memory space of the system and beyond.
  - All the addresses are available to the program.
  - Memory space can be much bigger in size than the actual main memory of the system.

**32-bit** machine can address **4GB** memory space

**64-bit** machine can address **$2^{24}$TB** memory space

5

- In a multi-program environment two major factors are unknown at the program compilation time. One is that how much memory is going to be available at run time. Imagine a program started when there are already 9 other programs are active in the system. It will definitely have less share of memory if there are 4 other programs are active in the system at the same time. This observation also leads to the fact that it is not determined at the compile time that which addresses will be available to the program at run time. It would be better if program can assume it has entire memory space and all the address at the run time. System software's job (compiler, linker, loader, etc) will be much easier in that case.

- Virtual memory system provides exactly this illusion to a program that it has the entire memory space (often which is much bigger than available physical memory present in the system – think of 64-bit machine which can address 16M TB memory space which can not be implemented in a real memory system. What a concurrent memory technology can provide is much less that 1 TB memory space). It also provides illusion of providing all the memory address to a single program. These memory addresses are called virtual memory address – analogous to google phone number which can be associated with any real phone number. During execution time, these virtual memory addresses are translated to actual memory address online.

## Virtual Memory – concept

- The actual memory is called a physical memory.
  - Includes both main memory (along with cache) and disk.
  - The main memory acts as cache for the disk.
- The virtual memory mechanism automates the management of two level hierarchy of main memory and disk.
  - A part of the hard disk is used for implementation of virtual memory.

**SWAP** space amount determines
how much would be limit for virtual memory

6

• In context of virtual memory, the actual implementation of the memory is called physical memory. This contains main memory system (with its cache) and disk. As a concept, virtual memory is not different than concept of cache. One can think the main memory as the cache of the disk storage. However, only a part of the available disk space is used as a part of the system memory. The rest of the part is used to implement file system. The space, which is the part of the system memory, is called a swap space. Virtual memory mechanism automates the management of the two level hierarchy, main memory and disk, freeing programmer from overhead of managing memories by themselves.

## Virtual Memory – concept

- Virtual memory is similar to cache.
  - Main memory acts as cache for the disk.
    - Both exploits locality of program execution.
  - Blocks are called Pages.
  - Miss is called page fault.
- Processor issues virtual address.
  - It is translated into a physical address.
    - Address mapping or translation
  - Provides relocation facilities for program execution.
    - Programs are loaded into individual pages.

7

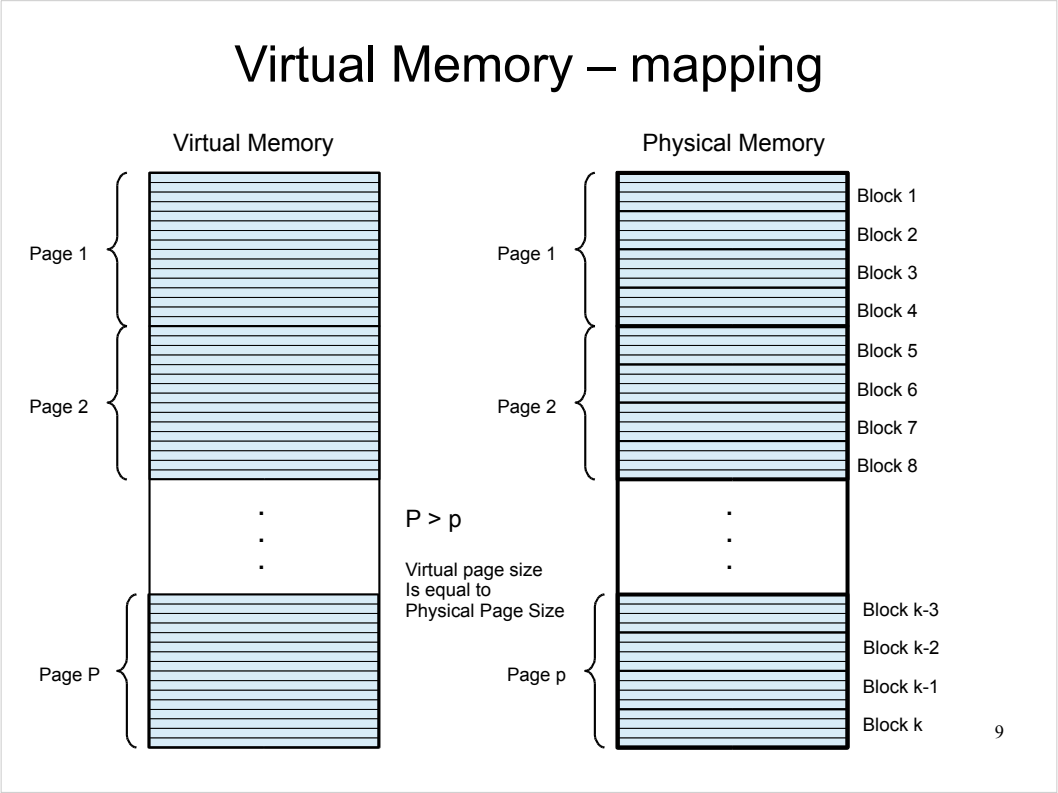
• Though cache and virtual memory systems are similar in concept, they have their difference in origin. The cache was developed to make the memory access faster and virtual memory was developed to provide illusion of having more available memory to program than system hardware offers. Therefore they developed different terminology for similar actions or phenomenon. A virtual memory equivalent to 'block' in cache is called a 'page'. Equivalent phenomenon of cache miss is called a 'page fault' in virtual memory context. The process of address mapping from virtual address to physical address is called 'address mapping' or 'address translation'.

• The processor of a computing system supporting virtual memory issues virtual address to fetch instructions or access data (for read or write). Memory management system maps virtual address into physical address to access corresponding information.

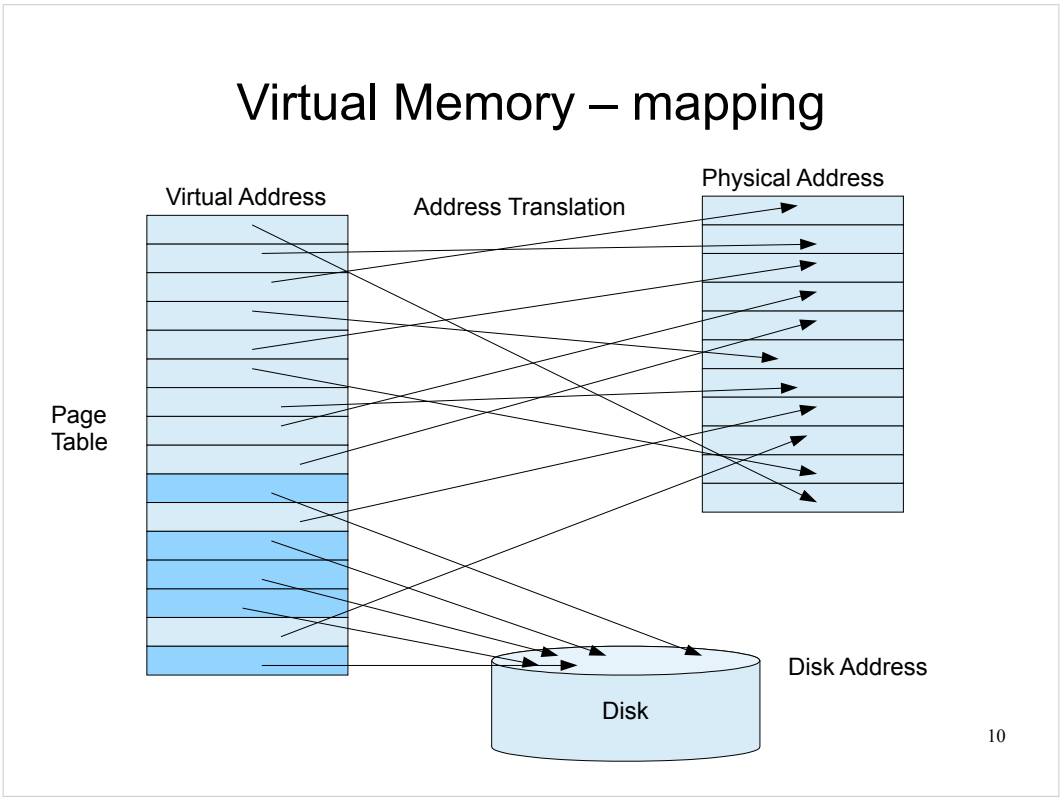
• Usage of virtual memory also facilitates easy loading and relocation of program execution. A program can be loaded into memory at any physical location. All the modern system relocates / load a program as individual pages into memory. Therefore there is no need to find a contiguous memory. Any free physical page location can be used to load the requested page into memory.

Virtual Memory Implementation ...

8

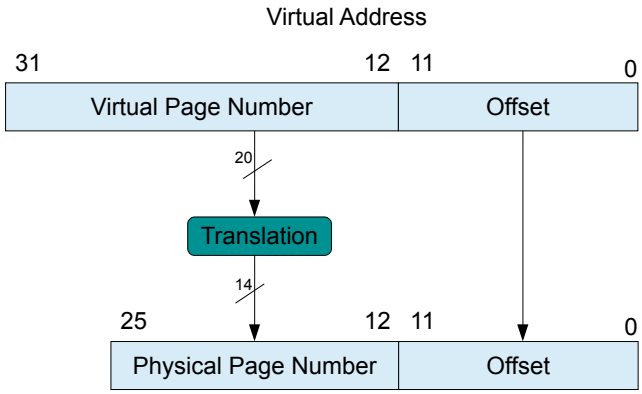



The virtual memory space is divided into pages. The page size is fairly big, in the order of 4K to 16K, even 64K. System may support multiple size pages depending on its need. For example, for a 32-bit machine, virtual memory space is 4GB. With a page size of 4K, the virtual memory space is divided in 1M pages of equal size. Similarly physical memory (main memory) is also broken into pages of similar size. However, usually system deploys less amount of main memory than that virtual memory space supports. Therefore, number of pages in main memory is less than (or equal to) that in the virtual memory space. Additionally, each physical page are divided into blocks for cache operations.




- The mapping concept is fairly straight forward. OS maintain all the physical page location for a program loaded into a map table indexed with virtual page number. This means, a system with 1M virtual pages (as in the example of the previous slide) will have a mapping table of 1M entry per process / active program. This mapping table is stored into primary memory (the reserved space in the memory map similar to what has been described in the CS147sec05 ISA). Each entry in this mapping table is either a physical page number or address to disk location where the page is stored. Since a virtual page can be present in disk as well, system can support larger memory need from a program that is larger than even the primary memory or available primary memory size (in case of multiprogram environment). The mapping table of virtual page to physical page is called a page table.

# Virtual Memory – mapping




Page size of 4K

11

- The virtual address is divided into two parts – one is the virtual page number (the MSB side) and offset (LSB side) or page offset. The virtual page number is translated into physical page number which may be represented in equal or less number of bits than that of the virtual page number. The offset remains the same. Therefore, the relative location of the requested data is same in the physical page as in the virtual page. In this example, virtual address supports 32-bit address space, where the physical address space supports 26-bit address. The page size is 4K. Hence, LSB side 12 bit of the virtual address is the page offset (which is the offset in the physical page also) and 20 MSB side bits represents the virtual page number. This page number is translated into 14 bit physical page number and the offset remains the same.

- For example is virtual page number 0x23ab4 is mapped to 0x18b0 physical page, then the virtual address 0x23ab4183 will be mapped to 0x18b0183 physical address.

## Virtual Memory – page size

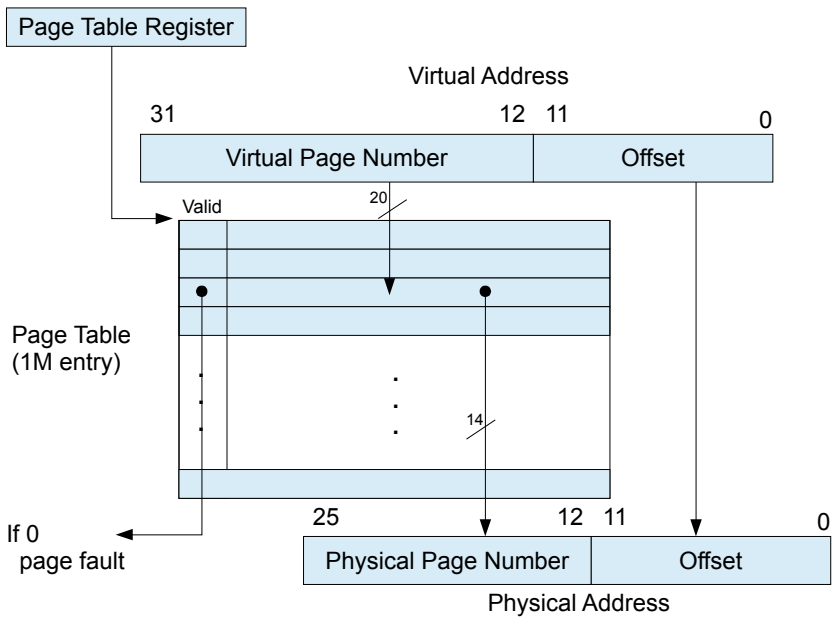
- A page fault is costly.
  - Costs millions of cycles due to high access time of disk.
  - Page size is very important decision.
- Key decisions in designing virtual memory system.
  - Page size should be large enough to balance high access time.
  - Reduce page fault rate by using fully associative page placements.
  - Page faults are to be handled by softwares (OS).
  - Write-back is the strategy to sync content across hierarchy.

12

• Upon page fault, corresponding pages are retrieved from disk. This process is really costly – almost in order of million cycles. Major part of this high cost is coming from access time of the hard disk (time to place the read/write head into place). Therefore, we need to determine such a page size which should be large enough to avoid further page fault in near future. It should try to amortize the high access time. Page sizes of 4K to 64K is used in any modern computing system. However, embedded system goes in different direction with page size of 1K (which is OK since such system does not uses electromechanical hard disk, but uses solid state hard disk – which uses flash memory technology).

• The page fault rate is further reduced by using fully associative page placement – i.e. any page can be placed into any available free page in main memory. Further the page fault is handled by software using interrupt mechanism. OS deploys sophisticated page replacement algorithm (LRU is one of them). This is OK since serving a page fault is so large, handling it in software does not slow down the page fault service significantly. On the other hand, page fault handling through software enables complex replacement algorithm to further reduce the page fault. For the same reason, for write operation, write back is the preferred mechanism to sync main memory page content and corresponding page in disk.

## Virtual Memory – Finding a page




13

- Since full associative placement, it is impractical to use full search in lookup for physical page location. Instead, a mapping table is used which is indexed by the virtual page number. The table contains control bits (such as valid, dirty , reference, etc.) and corresponding physical page number. This mapping table is known as page table. In this example, the 32-bit virtual memory space is divided into 1M pages with 4K page size. Thus for each active program, OS will maintain one separate page table per program with 1M entry. This page table is stored into the reserved space of the main memory. Processor also provides a special register to load pointer to the page table for the processor. This register is called 'Page Table Register'. At the time of context switching, this register is also saved along with other register values and program counter.

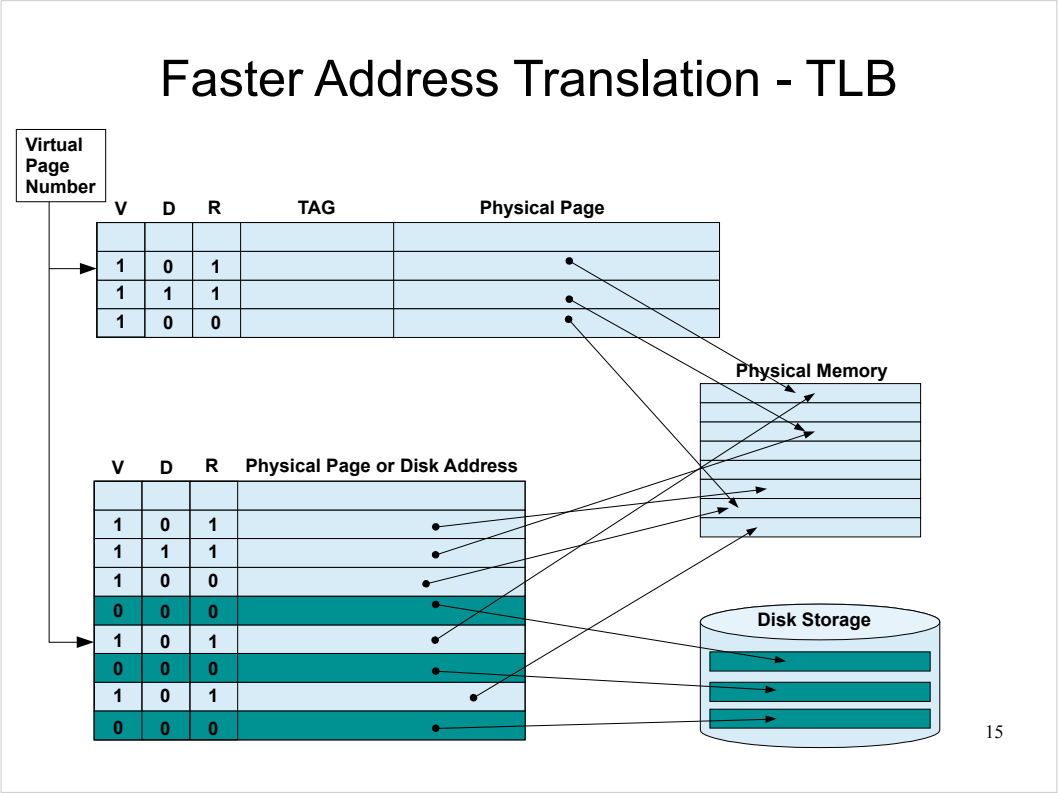
- Before starting a program (i.e. making a program active) OS creates copies of pages in the swap space for the program. The main memory is loaded with relevant page to start the program execution. Also a page table is created in the reserved space of the main memory for this new process. The page table register is loaded with pointer to this page table. The page table content points to either main memory address or disk location from where the page can be loaded. Once setup is done, program counter is set to address of the first instruction of the program.

## Virtual Memory – handling page fault

- Upon page fault, execution transfer is given to OS by interrupt mechanism.
- OS keeps track of
  - Copies and location of pages of a process in the disk.
  - Each individual physical pages and its recent usage history.
- Upon page fault service request OS performs the following steps.
  - Deploy LRU to choose a physical page to be replaced.
  - First the replaced physical page is stored back into disk (if it is dirty) and then new page from disk is written into physical page.
  - Process's page table is updated with new physical location.
  - Control is returned back to actual program.

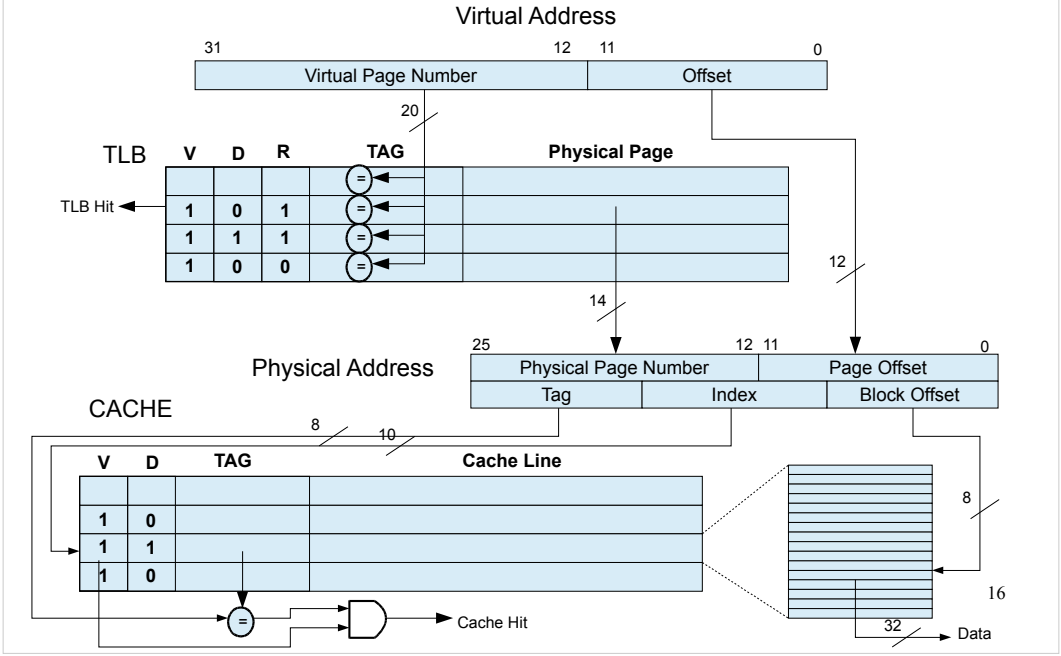
14

- The page fault occur if the valid bit is 0 in the page table. If that happens, control is transferred to OS through interrupt mechanism. OS keeps track of all the physical pages and its recent usage history. If a new page has to be loaded, incase of a page fault occurs, OS considers if there is a free page which can be used. If it finds one, it loads the page from disk into that physical page location and update the page table entry. If there is no free page available on the main memory, OS determines which page to be replaced. This replacement mechanism can deploy complex algorithm (which performs better than even LRU). Once such replacement page is identified, if the page is dirty, the page is written back into disk before it is replaced with new page. If a page is replaced, corresponding page table entry (may be from other process) is also updated to indicate that the page is now residing only in the disk. Once the page fault interrupt is served, control is returned back to the actual program.




- Notice that the page table is residing in main memory. Therefore any virtual memory address reference will have to be referencing main memory at least once on each reference. Recall that the main memory is slower, therefore it will slow down overall memory access. To fix this issue, cache approach is taken for the page table as well. A portion of page table resided in the cache memory, which deploys same tag based mechanism as in cache. This table is called a '**Translation look-aside buffer**' or TLB in short. In every virtual page lookup, system looks at the TLB. If it is present, mapping is done right there from the TLB. If it is not present, then it means either it is a true page fault, or the entry is missing in TLB. If actual page table look up determines that this is a true page fault, the fault is handled by OS. If it is not a true fault, corresponding entry is loaded into the TLB, by replacing one of the existing entry. Replacement must consider 'dirty' bit and write the entry back to the TLB. TLB also keeps track of which page has been referred recently. This bit used in replacement algorithm. TLB usually deploys full associativity with a a random replacement algorithm. However, if there is a set of not referenced entry in TLB, only they are considered in the random replacement.

# Putting it together – Virtual to Physical Memory




- The processor issues a virtual address. The virtual address is translated into physical address by TLB. This physical address is then used to determine cache line index and tag. Cache access happens using this information as discussed earlier.

# CS147 - Lecture 23

Kaushik Patra
(kaushik.patra@sjsu.edu)

17

- Virtual Memory – Concept
- Virtual Memory - Implementation

*Reference Books / Source:*

1) *Chapter 4 of 'Computer Organization & Architecture' by Stallings*
2) *Chapter 7 of 'Computer Organization & Design' by Patterson and Hennessy/*