Imperial College London

Lecture 10

Virtual Memory

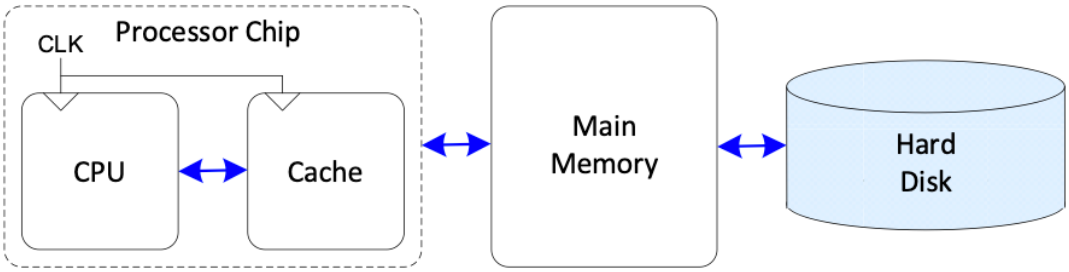
Peter Cheung
Imperial College London

URL: www.ee.imperial.ac.uk/pcheung/teaching/EE2_CAS/
E-mail: p.cheung@imperial.ac.uk

This lecture is about how virtual memory is managed in a process and a computer system.

Virtual Memory – What is it?

- Gives the illusion of memory bigger than physical size
- Main memory (DRAM) acts as cache for hard disk



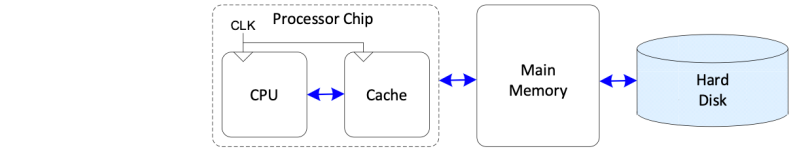
Based on: *"Digital Design and Computer Architecture (RISC-V Edition)"* by Sarah Harris and David Harris (H&H),

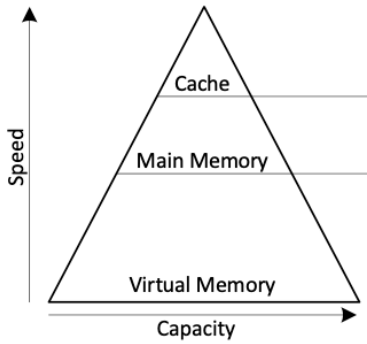
Caches makes the main memory, which is off-chip, appears to be on-chip and therefore much faster. It provides an interface mechanism between the CPU and the main memory.

Virtual memory is a different method that give the illusion that the main memory is larger than it actually is. In this way, the main memory act as cache for the main memory which provides a larger address space than what is possible with the main memory.

For example, a 32-bit byte addressing processor may only have a maximum of physical address space of 4GB. If one needs more than that, the hard disk could be used to provide that extra storage up to whatever is the size of the disk.

Memory Hierarchy - revisited





Technology	Price / GB	Access Time (ns)	Bandwidth (GB/s)
SRAM	$100	0.2 - 3	100+
DRAM	$3	10 - 50	30
SSD	$0.10	20,000	0.05 - 3
HDD	$0.03	5,000,000	0.001 - 0.1

- **Physical Memory:** DRAM (Main Memory)
- **Virtual Memory:** Hard drive
  - Slow, Large, Cheap

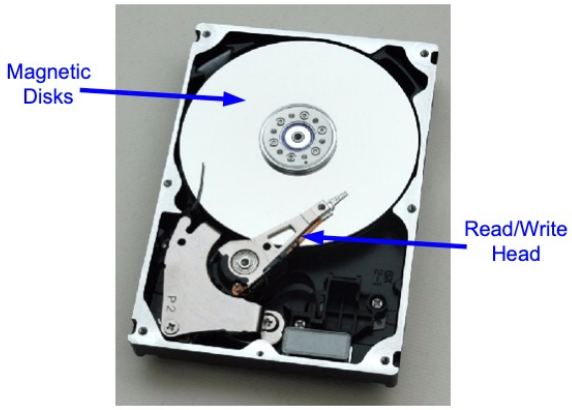
Based on: *"Digital Design and Computer Architecture (RISC-V Edition)"* by Sarah Harris and David Harris (H&H),

Since the cost per GB of hard disk is many times lower than main memory (DRAMs), having the ability to make hard disk appear like main memory is cost effective. However, hard disks, even the fast Solid State Disk (SSD) is much slower than DRAMs.

The memory space for the CPU that is on the hard disk is known as "Virtual" memory. The memory space offered by the main memory is called "Physical" memory.

Two common types of Disk memory

Hard Disk Drive



Solid State Drive

Relatively fast read, often slower write

Takes milliseconds to *seek* correct location on disk

Based on: "*Digital Design and Computer Architecture (RISC-V Edition)*" by Sarah Harris and David Harris (H&H),

In the past, computers generally use hard disk drives to put the virtual memory. It is large, cheap and slow.

In modern computers, hard disk drives with spinning platters are replaced by much faster (but more expensive) Solid State Disk (SSD) drives. These are made from integrated circuit, no moving parts, reprogrammable and non-volatile. While it costs 3X to 5X as compared to a hard disk per GB, it also offers much better performance. It is arguably the best cost-effective upgrade one can perform on an older laptop – replace the hard disk drive with a SSD drive!

Virtual Memory Address

- **Virtual addresses**
  - Programs use **virtual addresses**
  - Entire virtual address space **stored on a hard drive**
  - **Subset** of virtual address data in DRAM
  - CPU **translates v**irtual addresses into ***physical addresses*** (DRAM)
  - Data not in DRAM **fetched** from hard drive
- **Memory Protection**
  - Each program has own **virtual to physical mapping**
  - Two programs can use same virtual address for different data
  - Programs don't need to be aware others are running
  - One program (or virus) can't corrupt memory used by another

Based on: "*Digital Design and Computer Architecture (RISC-V Edition)*" by Sarah Harris and David Harris (H&H),

Addresses in any program is NOT real, but virtual. The whole of virtual memory contents (instructions and data) are stored on disk. Part of these are loaded into main memory from disk as required.

The CPU translate virtual addresses to physical addresses each time memory is accessed. Data/instructions then get fetch from hard disk to main memory, then to cache etc.

In this way, the same CPU can run different programs (threads) with either time multiplexing or using different CPU cores. Each program has its own virtual address space co-existing with the virtual address spaces of other programs.

Virtual addresses are mapped to physical addresses space. Two programs can have identical virtual address space, but map to DIFFERENT physical addresses. One program CANNOT read or write into another program's address space. This avoids corruption of other programs data or instructions.

Such mapping from virtual to physical addresses relies on both hardware and software implementations. The hardware is usually called Memory Management Unit (MMU). The software implementation resides in the operating system.

Virtual Memory Terminologies

Physical memory acts as cache for virtual memory

Cache	Virtual Memory
Block	Page
Block Size	Page Size
Block Offset	Page Offset
Miss	Page Fault
Tag	Virtual Page Number

- **Page size:** amount of memory transferred from hard disk to DRAM at once
- **Address translation:** determining physical address from virtual address
- **Page table:** lookup table used to translate virtual addresses to physical addresses

Based on: "*Digital Design and Computer Architecture (RISC-V Edition)*" by Sarah Harris and David Harris (H&H),

The physical memory in effect acts as cache for the virtual memory, which resides on the hard disk.

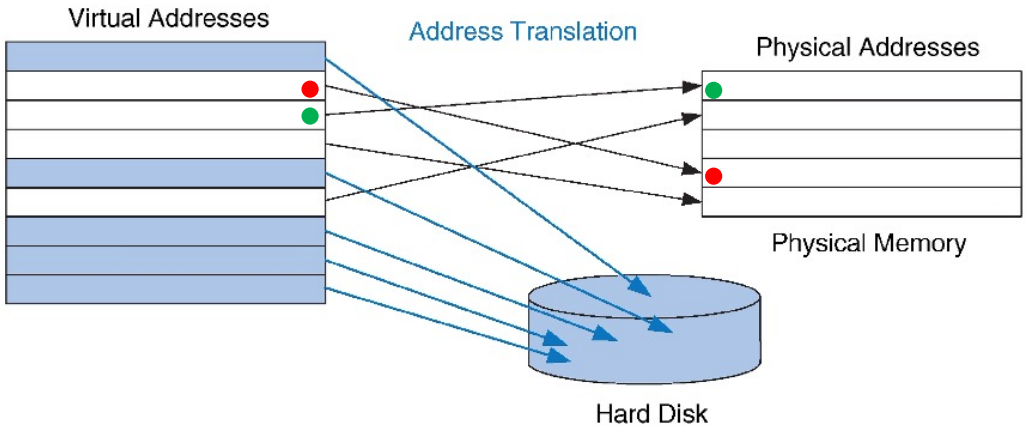
There are a lot of similarity between cache and virtual memory. Instead of blocks, virtual memory is organized into pages. A VM page is usually much larger than a cache block.

When data is not found on the page stored in main memory, and the system has to go to the VM on disk to fetch it, this is called a Page Fault instead of a Cache Miss.

When mapping a virtual address to a physical address, we identify the Page Number as part of the virtual address. This is similar to the Tag on cache memory.

In order to translate a virtual address to a physical address, we use a Page Table, which is a lookup table the maps a virtual page to a physical page, as explained in the next few slides.

Virtual to Physical Memory Mapping



Most accesses hit in physical memory

But programs have the **large capacity** of virtual memory

Based on: "*Digital Design and Computer Architecture (RISC-V Edition)*"
by Sarah Harris and David Harris (H&H),

This is a diagram showing that the virtual address space of a typical program has some of its contents mapped to the hard disk, and some to physical memory. The mapping is the address translation.

Unit of mapping is the a Page (which is often say, 4KB in size).

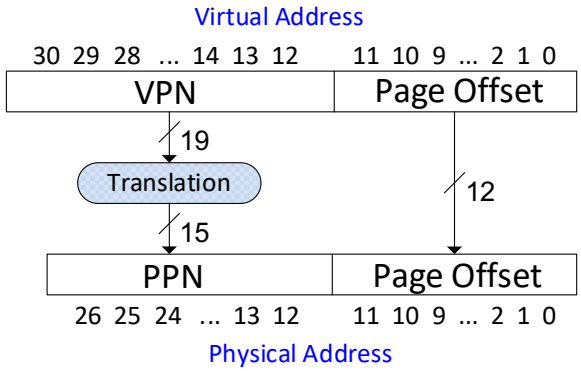
Note that the locations of consecutive pages in Virtual Memory can be ANYWHERE in physical memory. For example, the GREEN page and the RED page in VM is map to different part of the physical memory, despite them being consecutive in the virtual memory space.

Due to the large Page Size and the property of temporal and spatial locality in memory accesss of a typical program, most accesses to the VM space are found in the physical memory. In most cases, Page Misses are rare and therefore the fact that VM space exists on disk is often not notices. This is particularly true with SSD, which is much faster than hard disk.

When a Page has to be fetch from disk to be placed in physical memory, the old page is vacated and the new Page written. This is known as "swapping". A well designed system to minimum such page swapping.

Address Translation Example

- **System:**
  - Virtual memory size: 2 GB = 2^{31} bytes
  - Physical memory size: 128 MB = 2^{27} bytes
  - Page size: 4 KB = 2^{12} bytes



- **Organization:**
  - Virtual address: **31** bits
  - Physical address: **27** bits
  - Page offset: **12** bits
  - # Virtual pages = $2^{31}/2^{12}$ = 2^{19} (VPN = **19** bits)
  - # Physical pages = $2^{27}/2^{12}$ = 2^{15} (PPN = **15** bits)

Based on: "*Digital Design and Computer Architecture (RISC-V Edition)*" by Sarah Harris and David Harris (H&H),

This slide shows how to translate a virtual address into a physical address.

Assumptions of the system:

Virtual address space is 2GB, therefore the virtual address is 31 bits (2^{31} = 2G).

The actual physical memory space is only 128MB, 16 times smaller than the virtual address space. The physical address is therefore 27 bits.

Each VM Page is 4kB, i.e. 12 bit page offset (or address within a page).

The bottom 12 bits of the virtual address and physical address are the same.

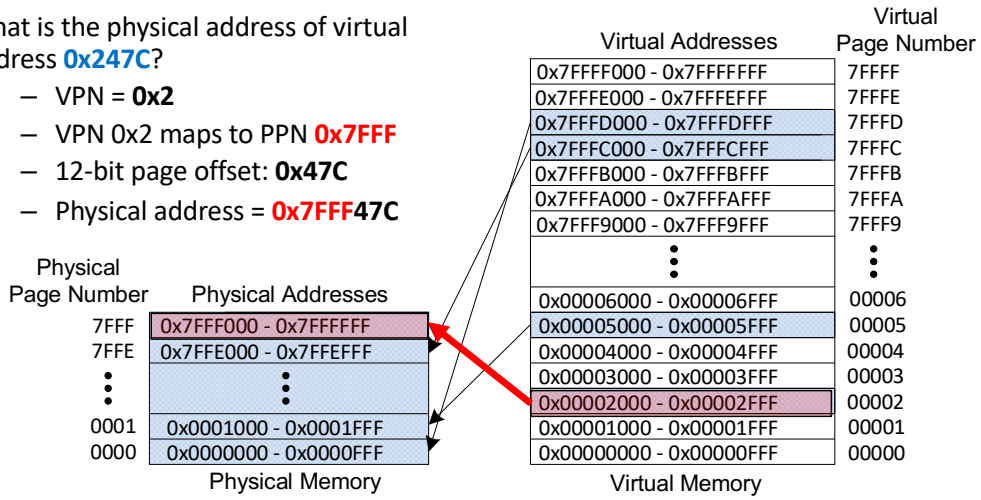
The top 19 bits of the virtual address is the Virtual Page Number (VPN). This goes through a translation lookup table, to find the 15-bit Physical Page Number (PPN) in the physical address.

Virtual Memory Example

- **19-bit virtual page numbers (VPN)**
- **15-bit physical page numbers (PPN)**

What is the physical address of virtual address **0x247C**?

- VPN = **0x2**
- VPN 0x2 maps to PPN **0x7FFF**
- 12-bit page offset: **0x47C**
- Physical address = **0x7FFF47C**



Based on: "*Digital Design and Computer Architecture (RISC-V Edition)*" by Sarah Harris and David Harris (H&H),

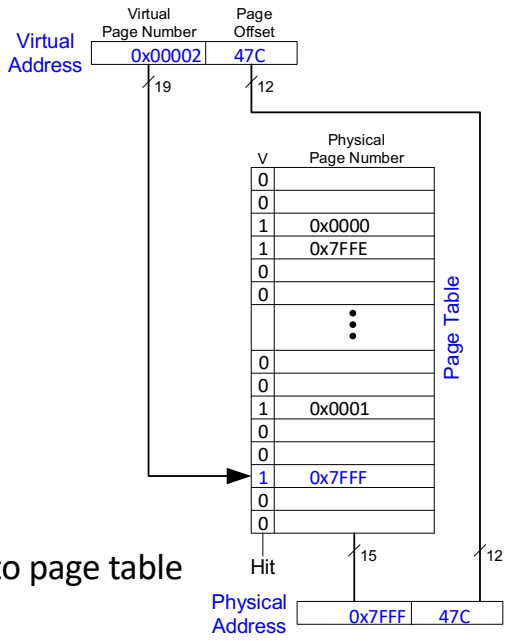
Here is an example. The virtual address is 0x247C. Therefore the VPN is 2, i.e. it is virtual address page 2. The Page Offset is 0x47C.

There is a translation table somewhere. Page 2 is third up from the bottom: page 0, page 1, then page 2. This entry in the lookup table is responsible for translating any virtual address between 0x2000 and 0x2FFF to physical address. The table contains the Physical Page Number, say, 0x7FFF.

Then virtual address 0x247C is translated to the physical address 0x7FFF47C.

Address Translation using Page Table

- **Page table**
  - One entry for **each** virtual page
  - Entry fields:
    - **Valid bit:** 1 if page in physical memory
    - **Physical page number:** where the page is located



Virtual Page Number (VPN) is index into page table

Based on: "*Digital Design and Computer Architecture (RISC-V Edition)*" by Sarah Harris and David Harris (H&H),

Here is the details of the Page Table that performs the translation.

The Page Table has one entry to every virtual memory page. The top 19 bits of the virtual address specifies the VPN. The entry in the Page Table consists of a Valid bit (V), which is set 1 only if it contains a valid Physical Page Number, meaning that the contents of the page resides in physical memory and are valid to use.

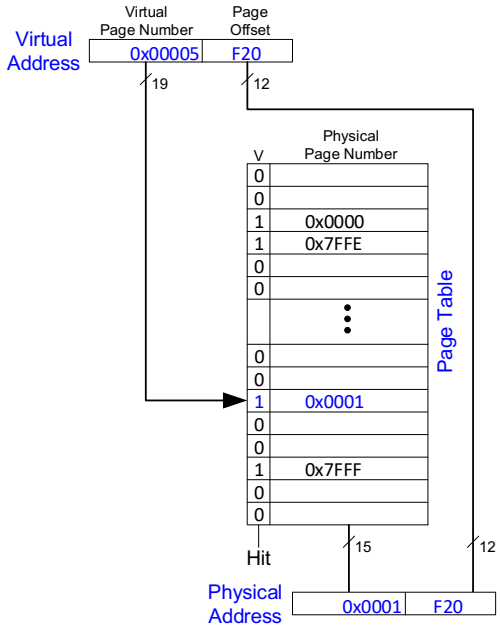
In this case, page 2 entry in the Page Table is value, and contains 15-bit PPN of 0x7FFF. In this way, the physical address is found and used and HIT is asserted.

Address Translation Example 1

Virtual address is **0x5F20**

What is the physical address?

- VPN = **5**
- Entry 5 in page table VPN 5 => physical page **1**
- Physical address: **0x1F20**



Based on: "*Digital Design and Computer Architecture (RISC-V Edition)*" by Sarah Harris and David Harris (H&H),

Take another example, virtual address is 0x**5**F20. VPN is 5.

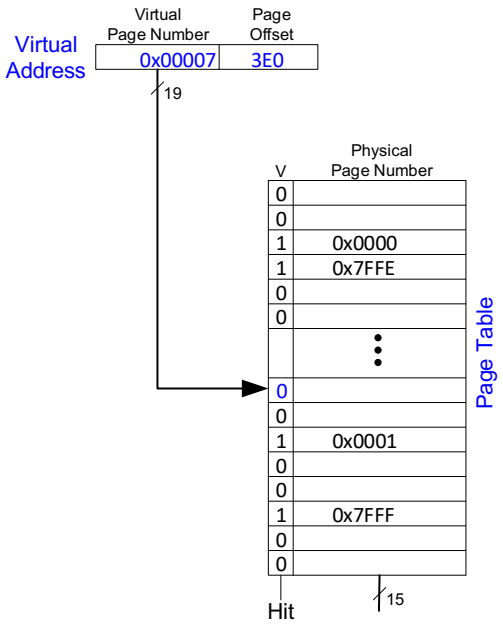
In the Page Table, entry for VPN=5 is valid and has the value of 0x0001. Therefore the physical address is 0x**1**F20.

Address Translation Example 2

Virtual address is **0x73E4**

What is the physical address?

- VPN = **7**
- Entry 7 is invalid
- Virtual page must be ***paged*** into physical memory from disk



Based on: "*Digital Design and Computer Architecture (RISC-V Edition)*" by Sarah Harris and David Harris (H&H),

Here is a final example. The virtual address is 0x73E4. Therefore VPN = 7.

Entry for 7 has valid bit V=0. A Page Fault occurs. 4kB of contents is now read from disk to physical memory (the location is determined by the OS), and the PPN is written into the table.

Page Table Challenges

- Page table is **large**
  - usually located in physical memory
- Load/store requires **2 main memory accesses**:
  - one for translation (page table read)
  - one to access data (after translation)
- Cuts memory performance in half
  - ***Unless we get clever…***

- **Cache most recent translations**
- Reduces number of memory accesses for *most* loads/stores from 2 to **1**

Based on: "*Digital Design and Computer Architecture (RISC-V Edition)*" by Sarah Harris and David Harris (H&H),

In general, the page table is large. In our example, the page table has 2^{19} entries. This table is used for every memory access, therefore it usually resides in the main physical memory. Even then, each load/store operation to virtual address requires up to 2 main memory access. The first to read the translation table, and the second to access the data.

This is obviously not efficient each if the main memory speed is high. Therefore we build a hardware cache on the processor chip to store the most recent translations, such that most access now only take 1 memory operation instead of 2.

Translation Lookaside Buffer (TLB)

- **Page table accesses:** high temporal locality
  - Large page size, so consecutive loads/stores likely to access same page
- **TLB**
  - **Small:** accessed in < 1 cycle
  - Typically **16 - 512 entries**
  - **Fully associative**
  - **> 99%** hit rates typical
  - **Reduces number of memory accesses** for most loads/stores from 2 to 1

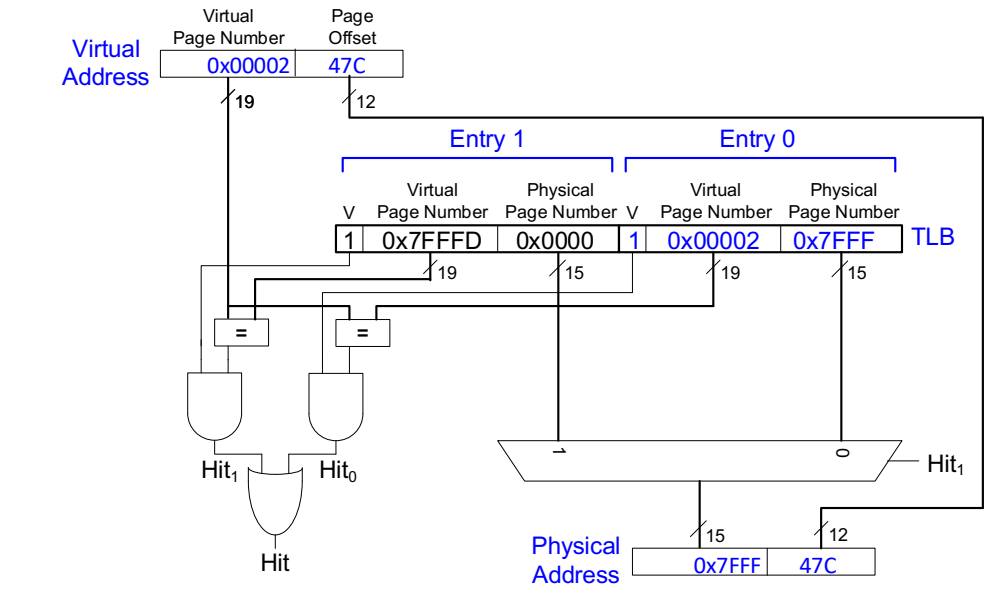
Based on: "*Digital Design and Computer Architecture (RISC-V Edition)*" by Sarah Harris and David Harris (H&H),

Due to temporal locality and the large page size, generally successive access to memory are very likely to be fro the same page, using the same VPN to PPN mapping. Therefore we have a local cache to store this translation table.

This lookup table cache is called a "Translation Lookaside Buffer" (TLB).

TLB are usually quite small – 16 to 512 entries. It is fast and could be access within a memory access cycle. It is fully associative meaning that the mapping can be to any physical memory page. Usually the TLB hit rate is over 99%, thus making most memory access to 1 per read/write to physical memory.

A 2-entry TLB



Based on: "*Digital Design and Computer Architecture (RISC-V Edition)*" by Sarah Harris and David Harris (H&H),

This diagram shows a 2-entry TLB – it looks very similar to cache, except that it stores the PPN instead of memory contents.

Virtual memory Summary

- **Multiple processes** (programs) run at once
- Each process has its **own page table**
- Each process can use **entire virtual address space**
- A process can only access a **subset of physical pages**: those mapped in its own page table

- Virtual memory increases **capacity**
- A subset of virtual pages in physical memory
- **Page table** maps virtual pages to physical pages – address translation
- A **TLB** speeds up address translation
- Different page tables for different programs provides **memory protection**

Based on: "*Digital Design and Computer Architecture (RISC-V Edition)*" by Sarah Harris and David Harris (H&H),

Here are the key lessons on Virtual Memory.