CE1006/CZ1006
Computer Organisation and Architecture

Computer Memory
Virtual Memory

Oh Hong Lye
Lecturer
SCSE, Nanyang Technological University.

Virtual Memory

- A memory management technique that maps memory addresses **used by program/code (virtual address)** to actual addresses in the physical memory (physical address).

- Virtual memory to physical memory address translation is carried out by a hardware known as **Memory Management Unit (MMU)**.

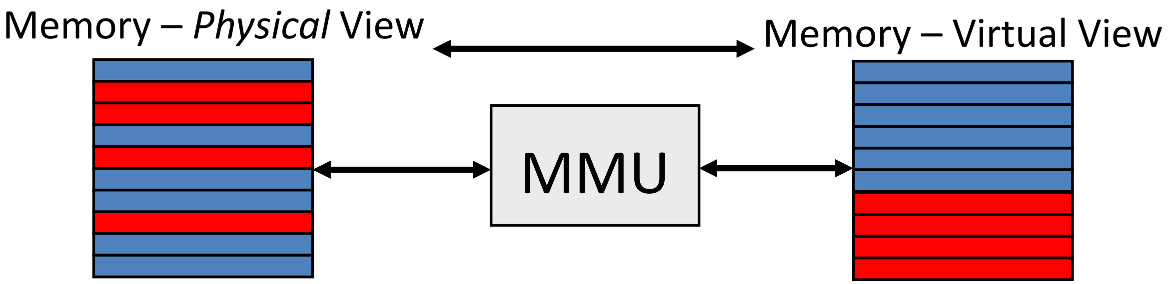
- A system with virtual memory may use **paging** or **segmentation**.

Advantages of Virtual Memory

- Allows the system to run programs that are **bigger** than the size of the main memory.
  - E.g. using the main storage (Hard Disk) to store all the program and only pull in the necessary code into the main memory when needed.

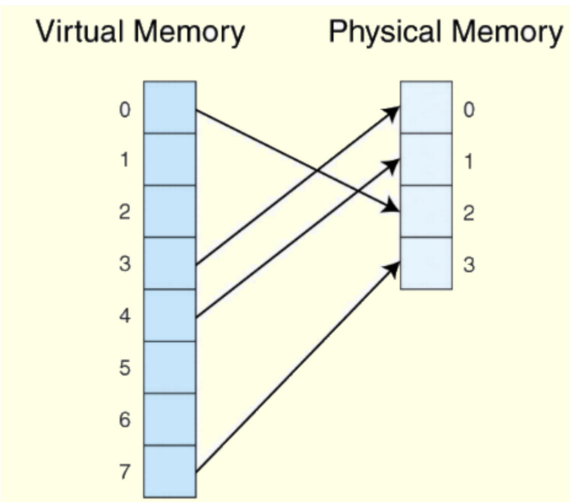
- Frees application from having to **manage a shared memory space** in the physical memory.

- A **virtual address space** seen by the application can appear to be **contiguous** when it is actually spread across fragmented blocks in the physical memory.



Paging Method

- In a system that uses paging, the main memory is partitioned into individually managed **page frames** (fixed size) and stores the complete code in the virtual memory, typically the secondary storage e.g. hard disk.
- Each page in the virtual memory is **copied** to the physical memory **as and when needed** according to the paging scheme used.
- **Frame** refers to the **physical page** numbers in the main memory.
- **Page** refers to the **virtual page** numbers used by program code.
- Information concerning the location of each page, whether on disk or in memory, is maintained in a data structure called a **page table** (shown below).
- There is one page table for each active process.



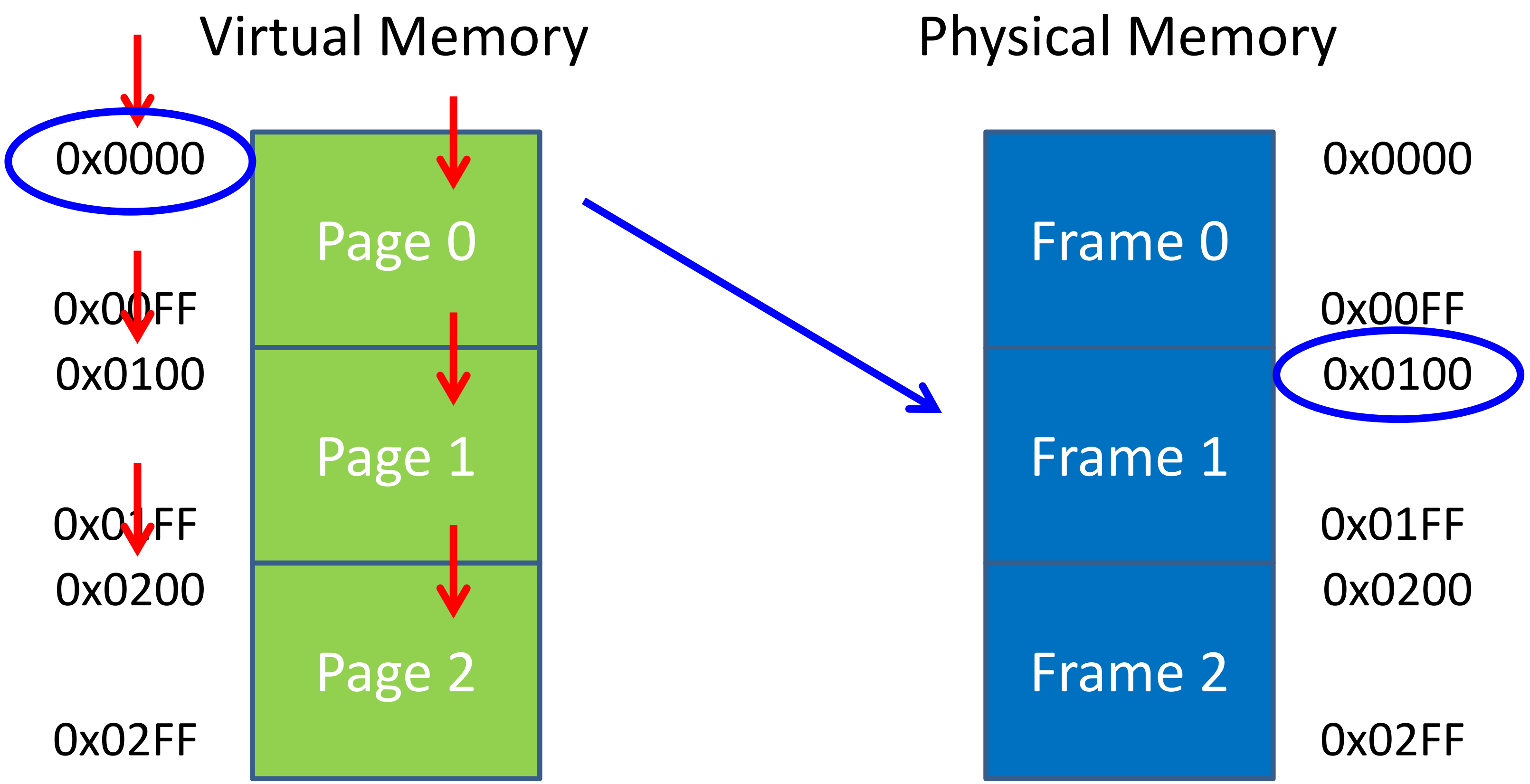
Page Table

Page	Frame #	Valid Bit
0	2	1
1	-	0
2	-	0
3	0	1
4	1	1
5	-	0
6	-	0
7	3	1

Virtual Page 0 -> Frame 2
Virtual Page 3 -> Frame 0
Virtual Page 4 -> Frame 1
Virtual Page 7 -> Frame 3

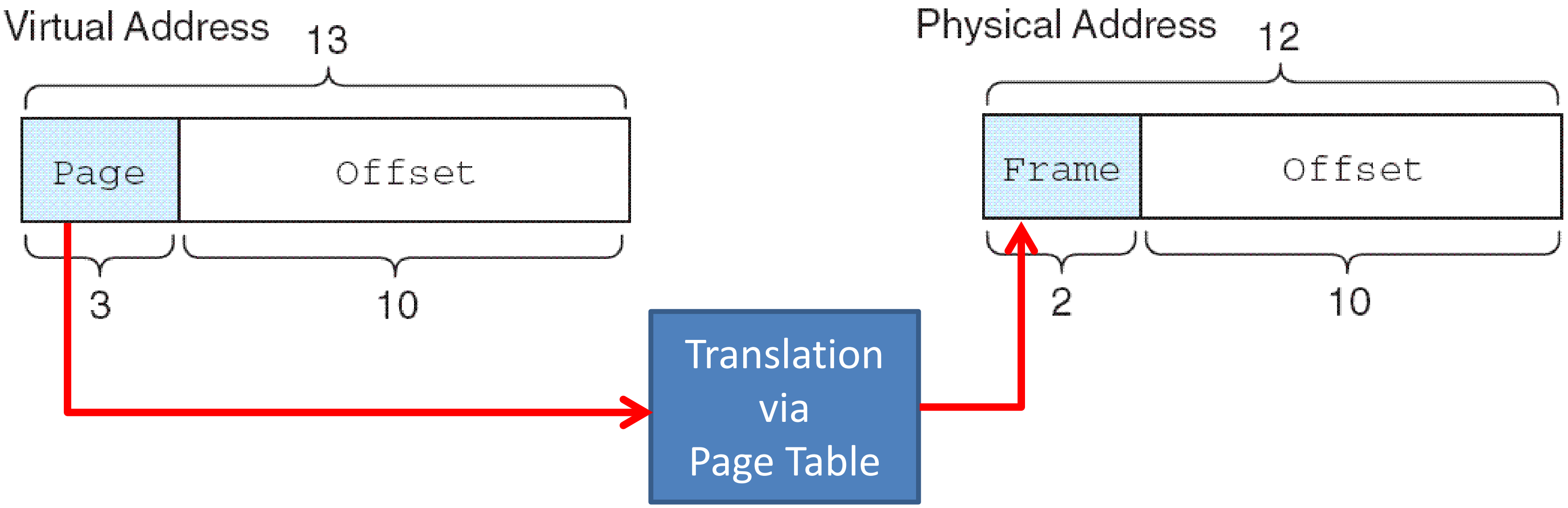
Address vs Page vs Frame Number

Page Size = Frame Size = 256 Bytes = 0x100 Bytes



Paging Example

- Consider a system with a virtual address space of 8K and a physical address space of 4K, and the system uses byte addressing.
- If we use 1Kbyte per page, we can have 8 virtual pages mapping to 4 page frames.
- A virtual address has 13 bits ($8K = 2^{13}$) with 3 bits for the page field and 10 for the offset, because the page size is 1024.
- A physical memory address requires 12 bits, the first two bits for the page frame and the trailing 10 bits the offset.



Paging Example

- Suppose we have the page table shown below.
- What happens when CPU generates address $5459_{10} = 1010101010011_2 = 1553_{16}$?
- What happens when CPU generates address $4000_{10} = 0111110100000_2 = 0FA0_{16}$?

Page Table

Page	Frame	Valid Bit
0	–	0
1	3	1
2	0	1
3	–	0
4	–	0
5	1	1
6	2	1
7	–	0

Addresses

Page	Base 10	Base 16
0 :	0 – 1023	0 – 3FF
1 :	1024 – 2047	400 – 7FF
2 :	2048 – 3071	800 – BFF
3 :	3072 – 4095	C00 – FFF
4 :	4096 – 5119	1000 – 13FF
5 :	5120 – 6143	1400 – 17FF
6 :	6144 – 7167	1800 – 1BFF
7 :	7168 – 8191	1C00 – 1FFF

Paging Example

- In the previous slide, when accessing address 0x1553 (which is in virtual page 5), the page table shows the valid bit is 1. From the page table, physical frame 1 will be used for subsequent address translation needed to access the actual data.

- When accessing address 0xFA0 (virtual page 3), the page table shows that the valid bit is zero. If the valid bit is zero in the page table entry for the virtual address, this means that the page is not in memory and must be fetched from disk.
  - This is a page fault.
  - If necessary, a page is evicted from memory and is replaced by the page retrieved from disk, and the valid bit is set to 1.

- For both cases above, the data is then accessed by adding the offset to the physical frame number (address translation is simply replacing the virtual page number with the corresponding physical frame number in the page table).

CE1006/CZ1006
Computer Organisation and Architecture

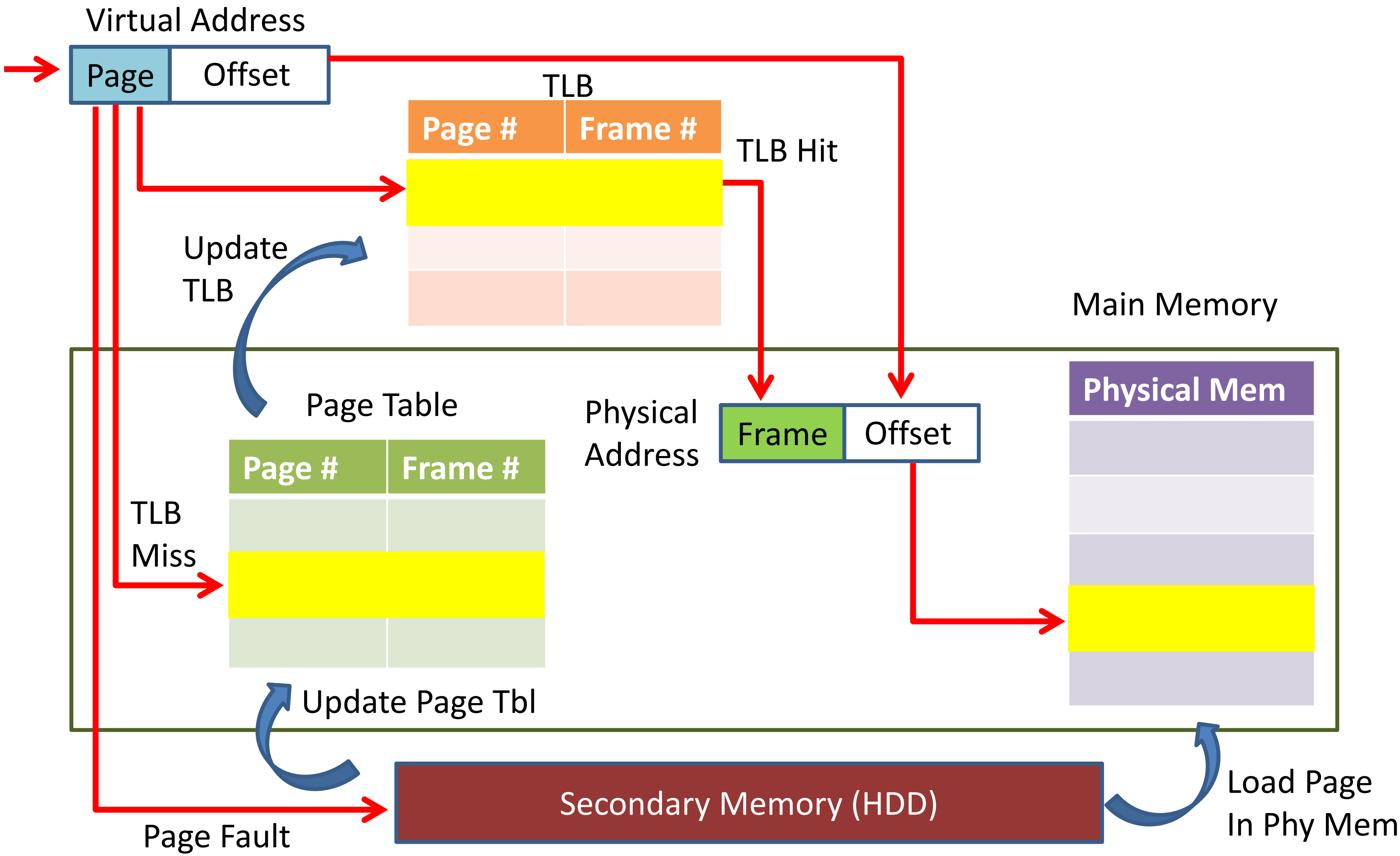
Computer Memory
Translation Lookaside Buffer (TLB)

Oh Hong Lye
Lecturer
SCSE, Nanyang Technological University.

Translation Lookaside Buffer (TLB)

- Page Table is located in Main Memory so access is slow.
- To speed up the page translation process, a page table cache (TLB) is used to stored a list of most recent page lookup values.
- Each TLB consist of a virtual Page number and its corresponding frame number.

TLB Lookup Process



CE1006/CZ1006

Computer Organisation and Architecture

Computer Memory
Segmentation and Fragmentation

Oh Hong Lye

Lecturer

SCSE, Nanyang Technological University.

Segmentation Method

- Another approach to virtual memory is the use of **segmentation**.
- Instead of dividing memory into equal-sized pages, virtual address space is divided into **variable-length** segments, often under the control of the programmer.
- A segment is located through its entry in a **segment table**, which contains the segment's memory location and a bounds limit that indicates its **size**.
- After a **page fault**, the operating system searches for a location in memory large enough to hold the segment that is retrieved from disk.

Fragmentation

- **Memory fragmentation** occurs when small, unusable clusters of memory addresses are created. Both paging and segmentation can cause fragmentation.
- **Paging** is subject to **internal fragmentation** because a process may not need the entire range of addresses contained within the page. Thus, there may be many pages containing unused fragments of memory.
- **Segmentation** is subject to **external fragmentation**, which occurs when contiguous chunks of memory become broken up as segments are allocated and de-allocated over time.