Anthony Ventresque

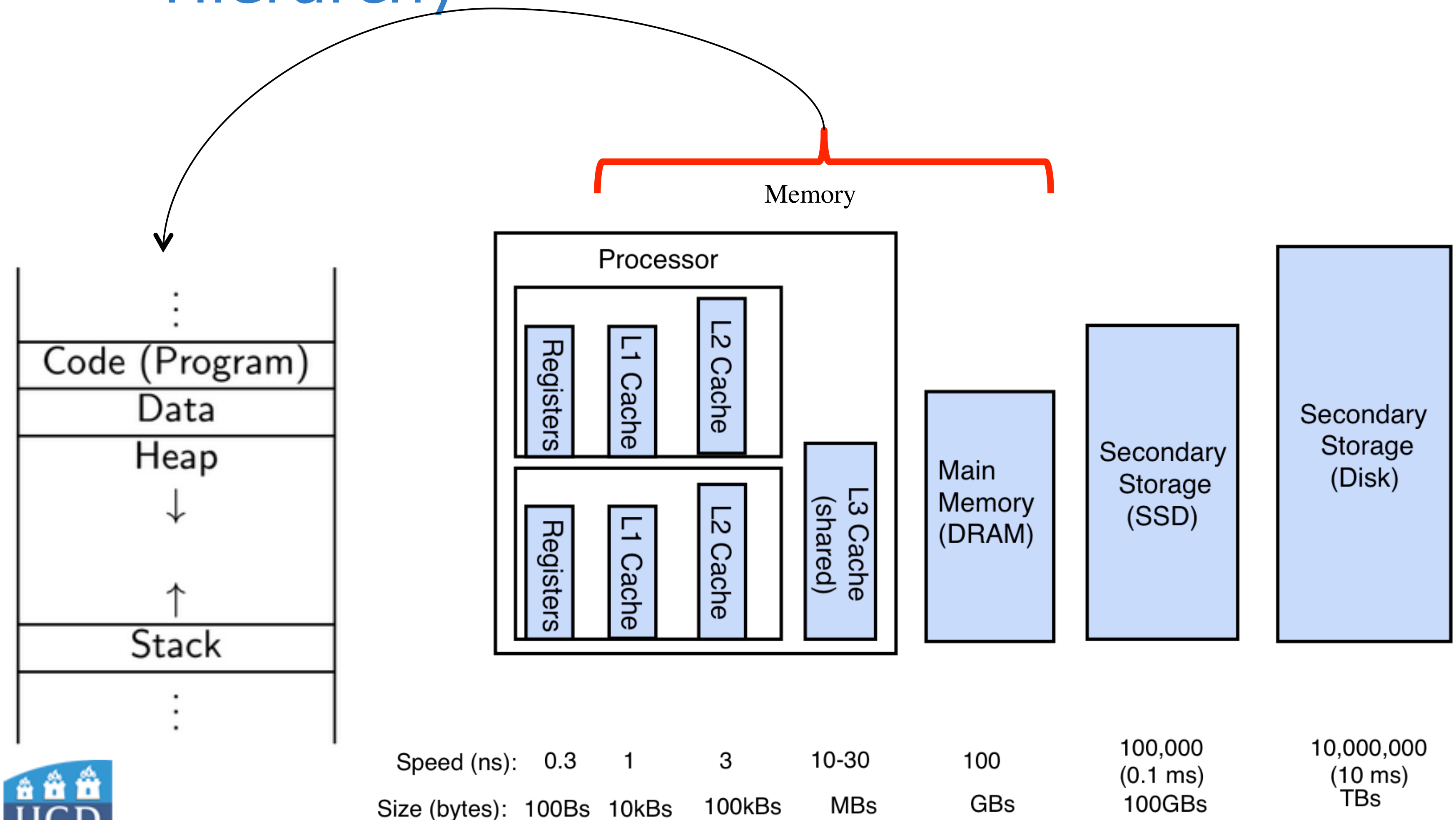
anthony.ventresque@ucd.ie

Summary of Memory Management and Virtual Memory Management

School of Computer Science, UCD

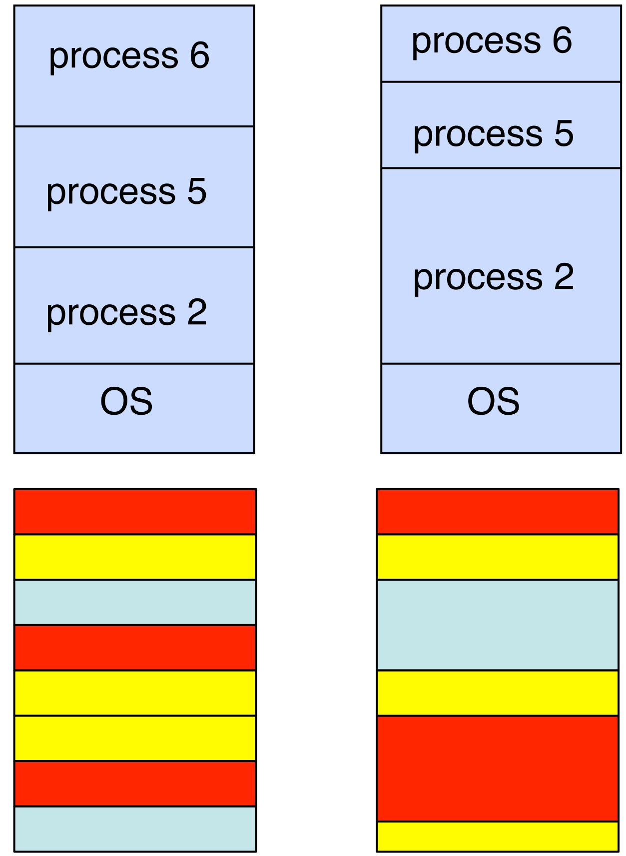
Scoil na Ríomheolaíochta, UCD

Management & Access to the Memory Hierarchy



General Classification

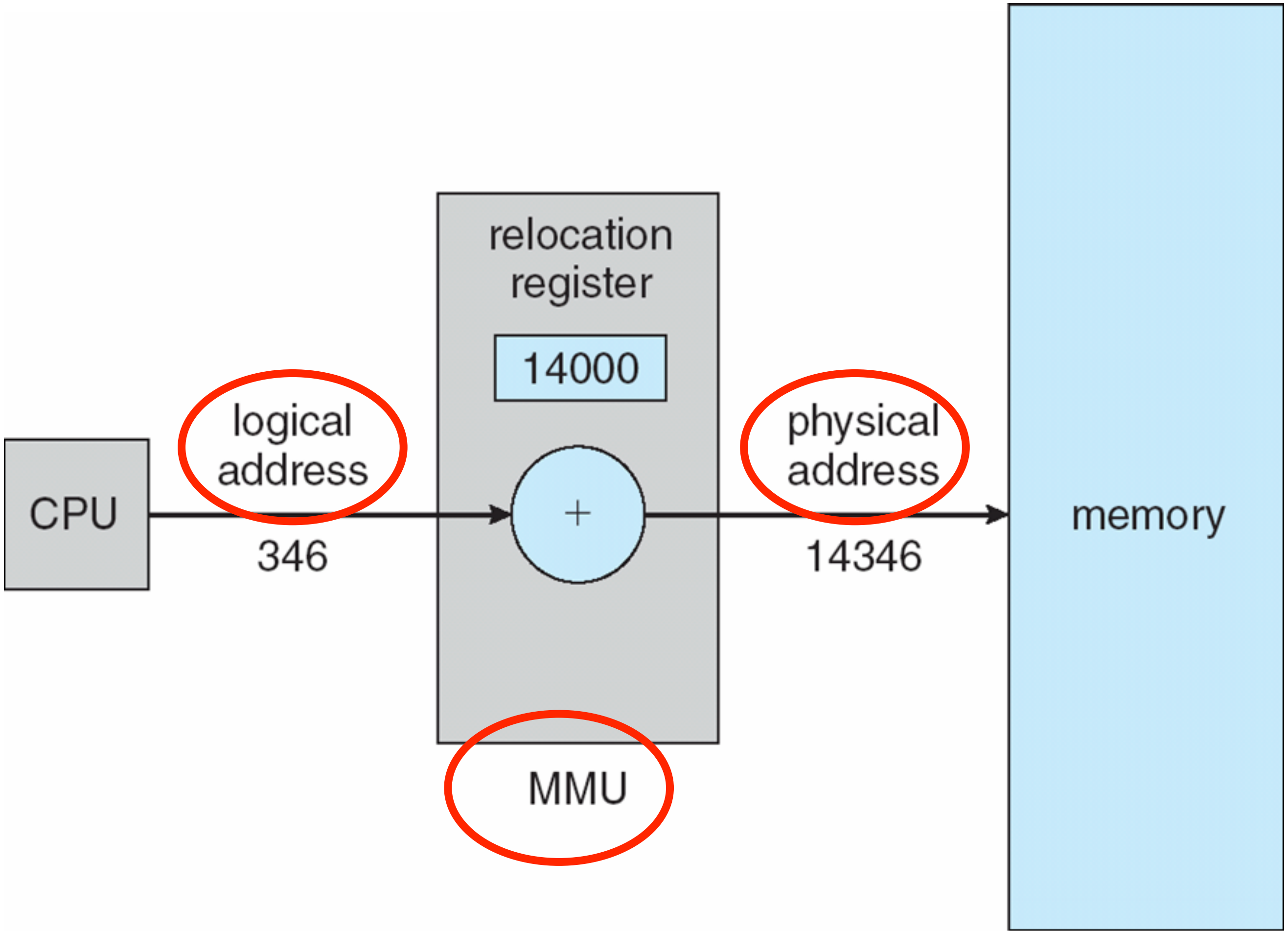
- Swapping
- Contiguous Allocation
  - Fixed
  - Variable
- Non-contiguous Allocation
  - Fixed: Paging
  - Variable: Segmentation
- Virtual Memory



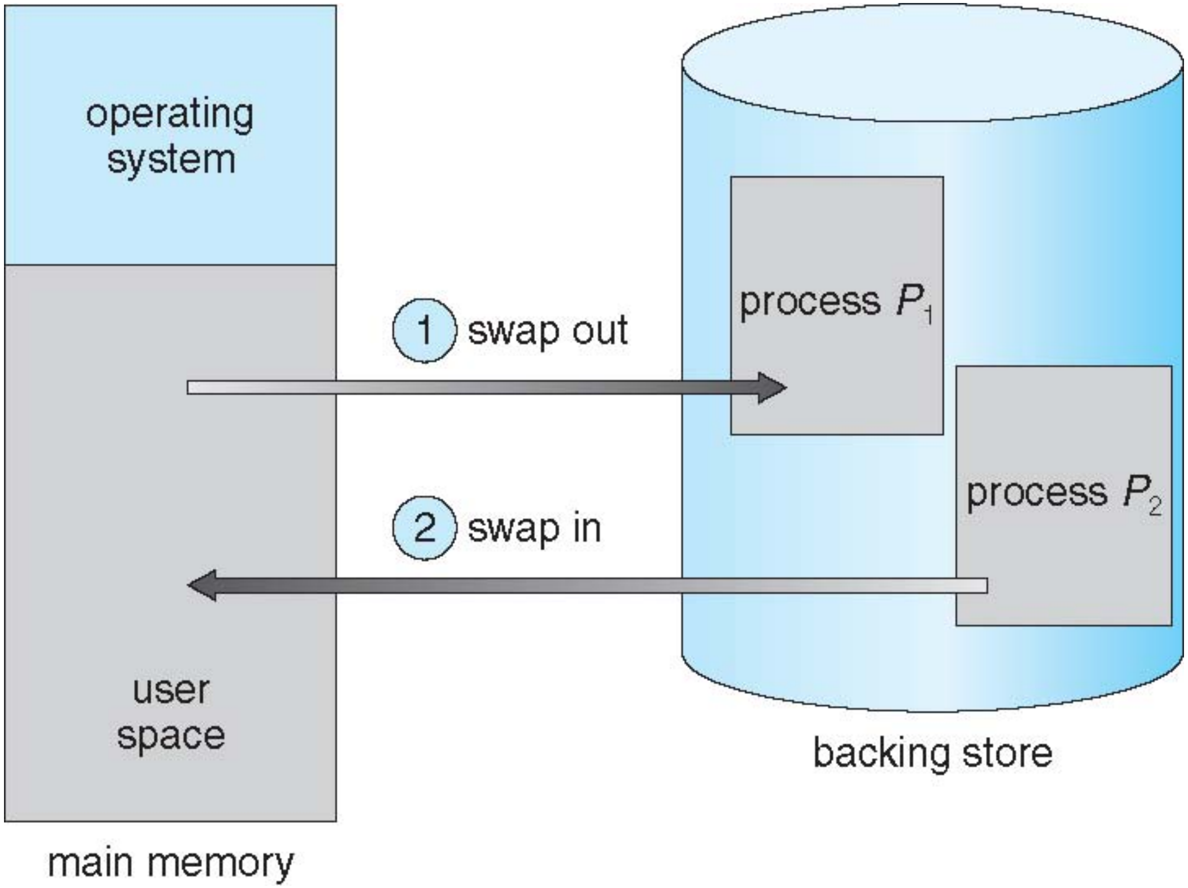
pages/frames

segments

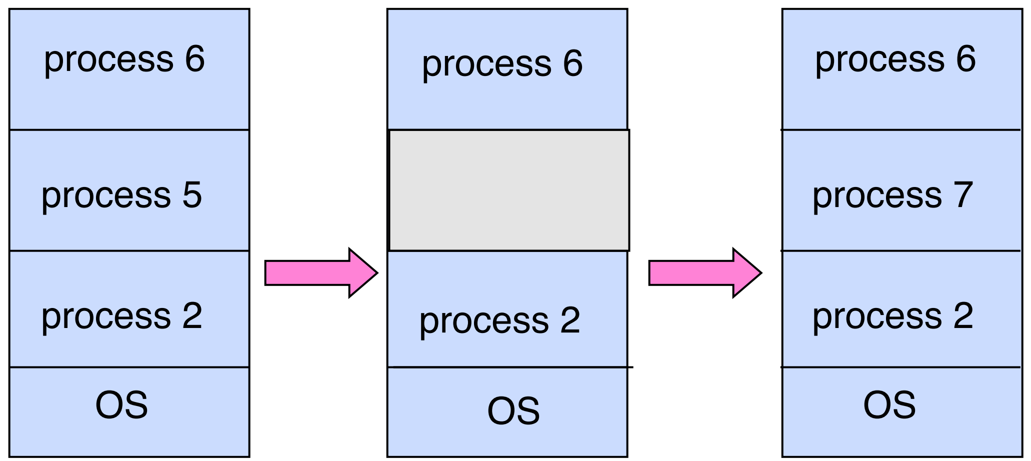
Logical vs. Physical Address Space



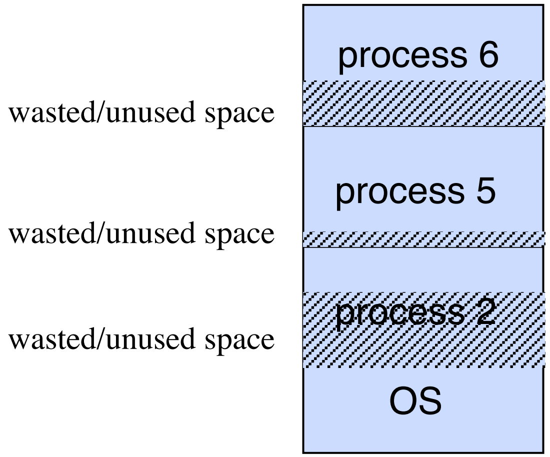
Loading and Swapping



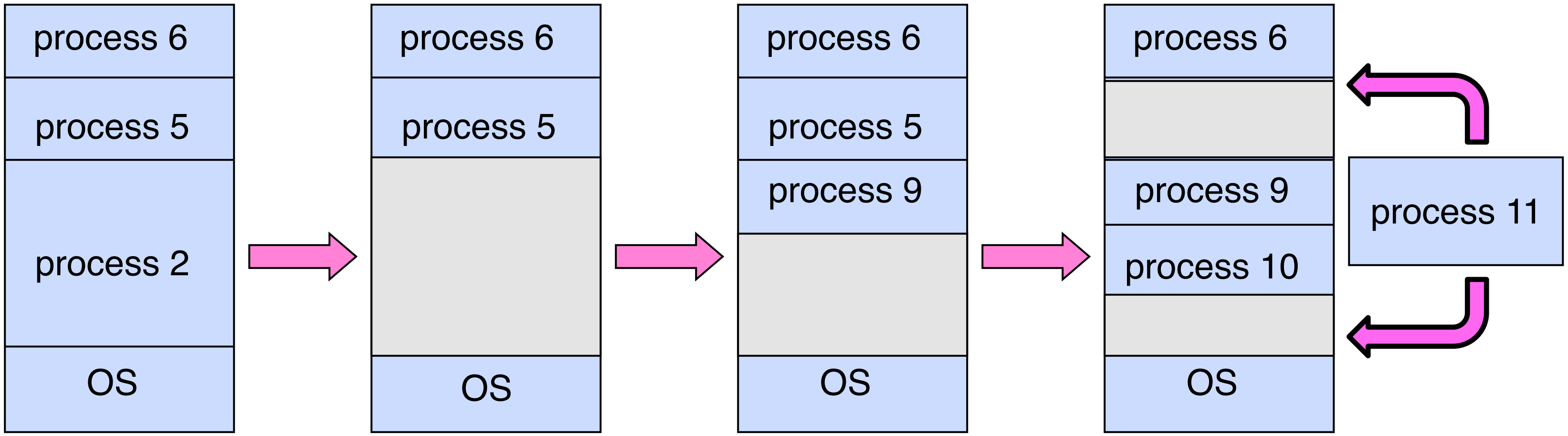
Contiguous Allocation – Fixed Partitions



- ***Internal Fragmentation***: not all processes of the same size -> wasted space



Contiguous Allocation – Variable Partitions

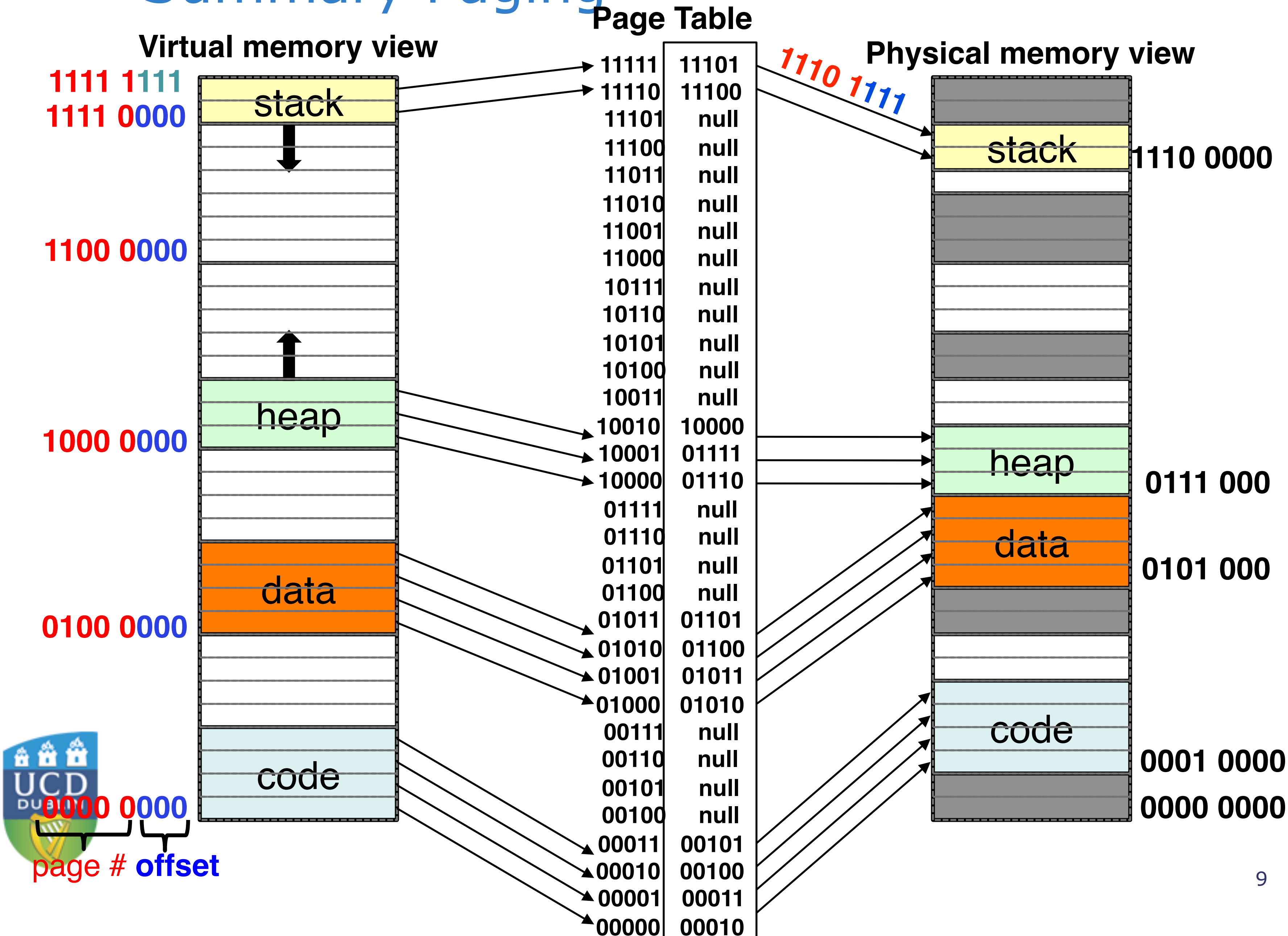


- External Fragmentation: wasted space between processes
  - First Fit
  - Best Fit
  - Worst Fit

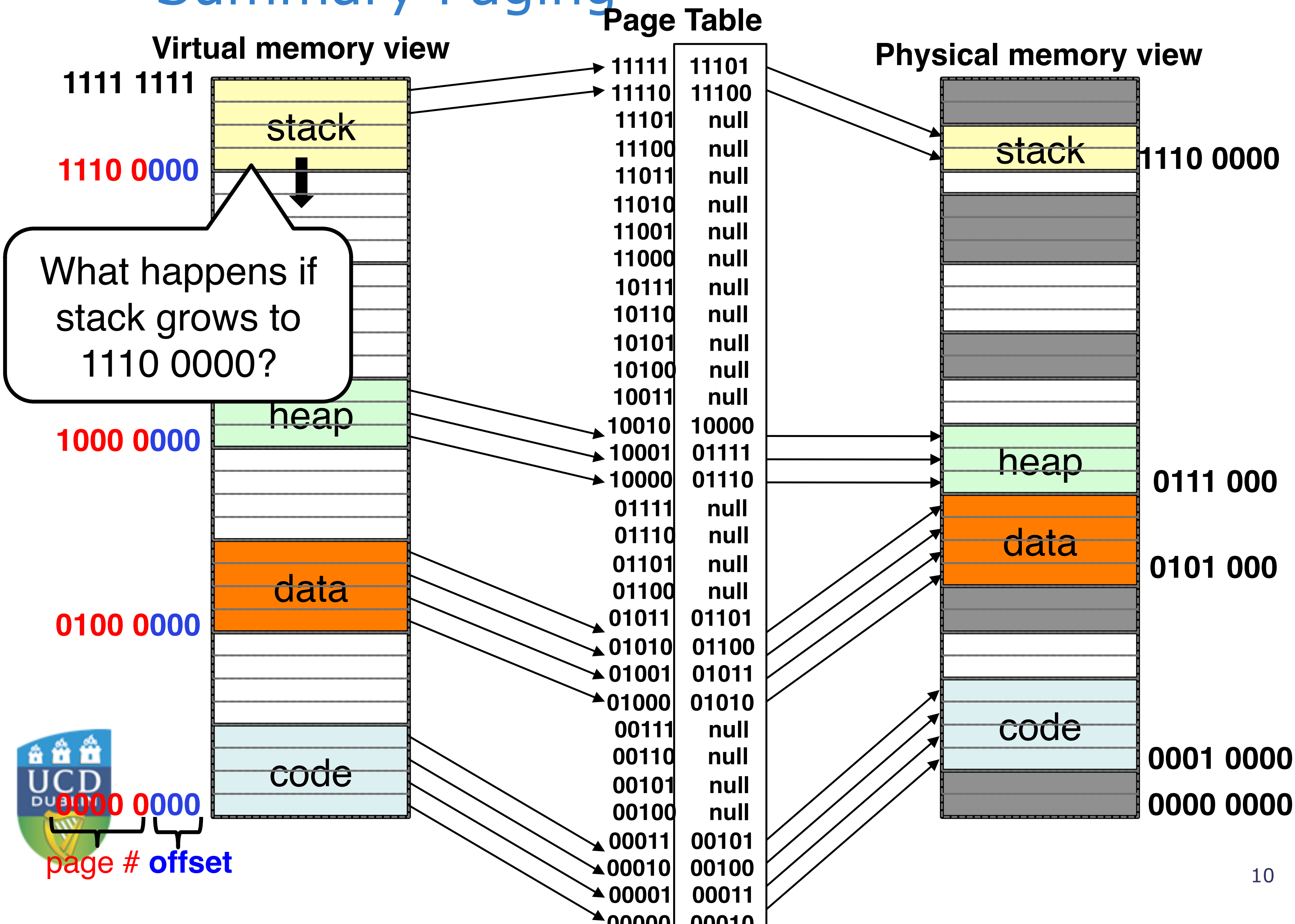
Non-contiguous Allocation (Fixed): Paging Technique

- Basic strategy in the ***paging*** technique:
  - The physical memory is partitioned into small equal fixed-size chunks (called ***frames***)
  - The logical memory (i.e. the address space of a process) is also divided into chunks of same size as the frames (called ***pages***)
- To execute a process, its pages are loaded from disk into available memory frames relying on a ***page table***
  - Frames associated to a process can be <u>noncontiguous</u>
- **Advantages:**
  - A program can be loaded if there are enough free frames, and there is no need for fitting algorithms
  - Moreover, we could just load the few pages relevant for execution (virtual memory: more about it in next lecture)

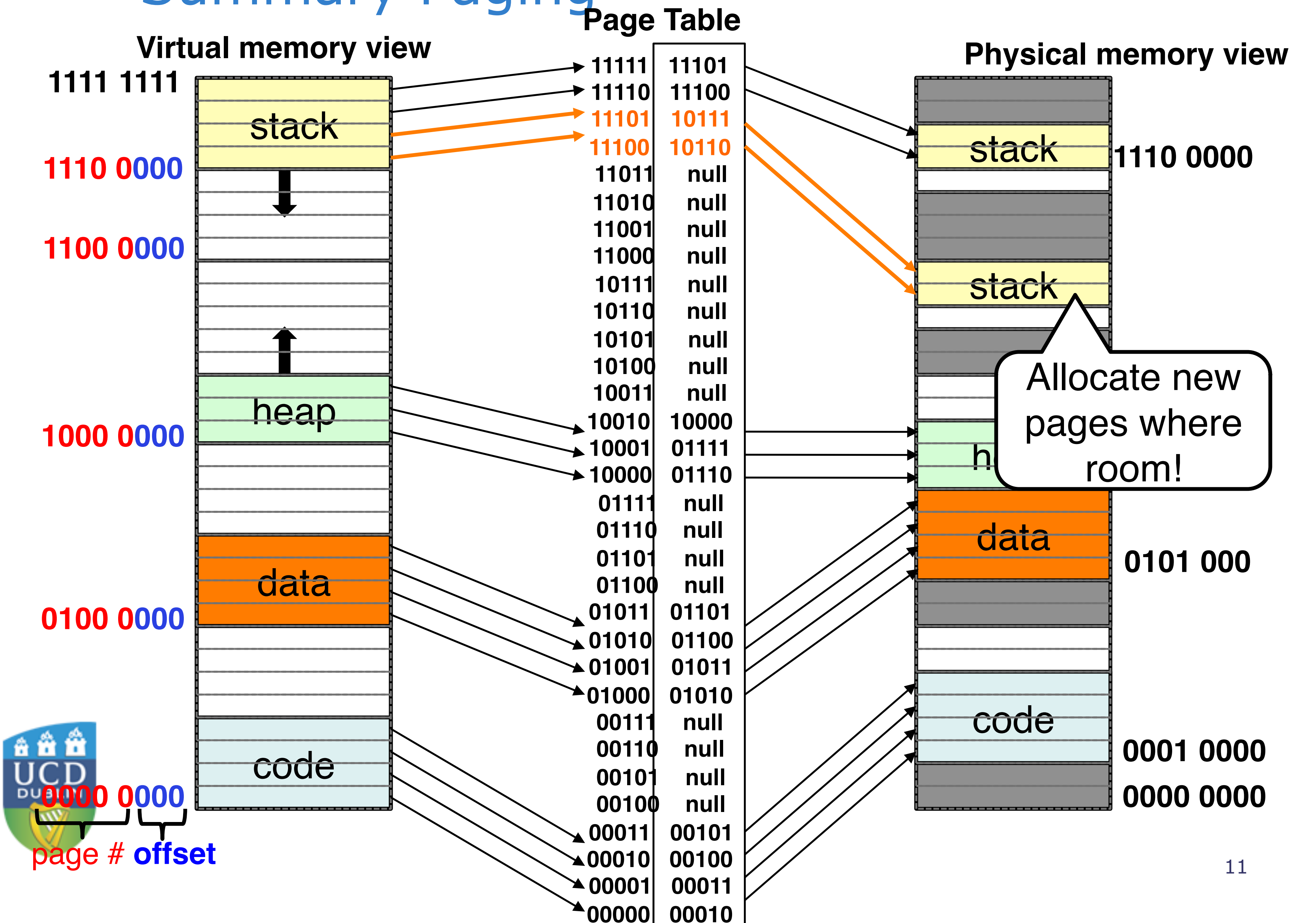
Summary Paging



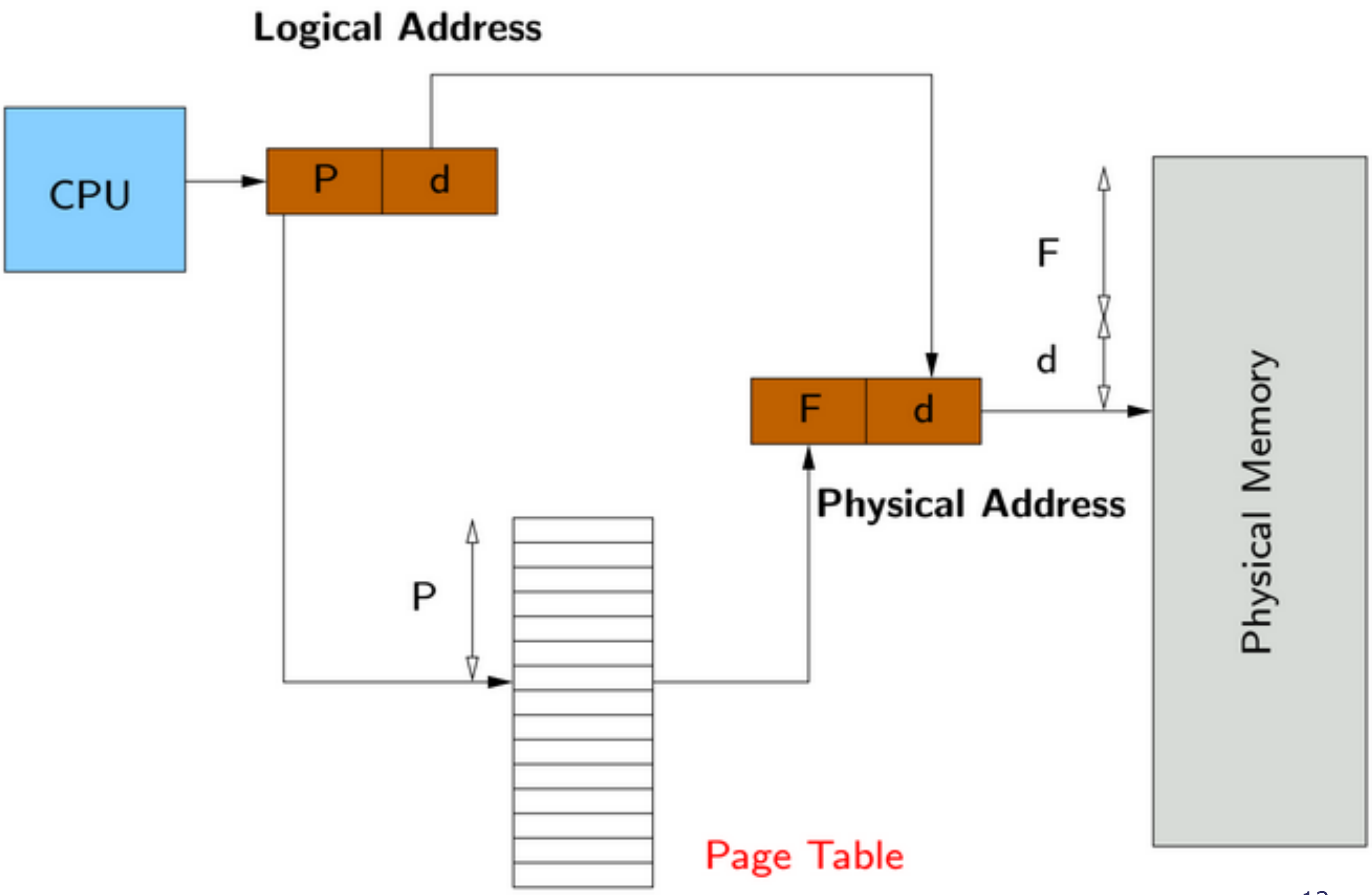
Summary Paging



Summary Paging



Paging Hardware

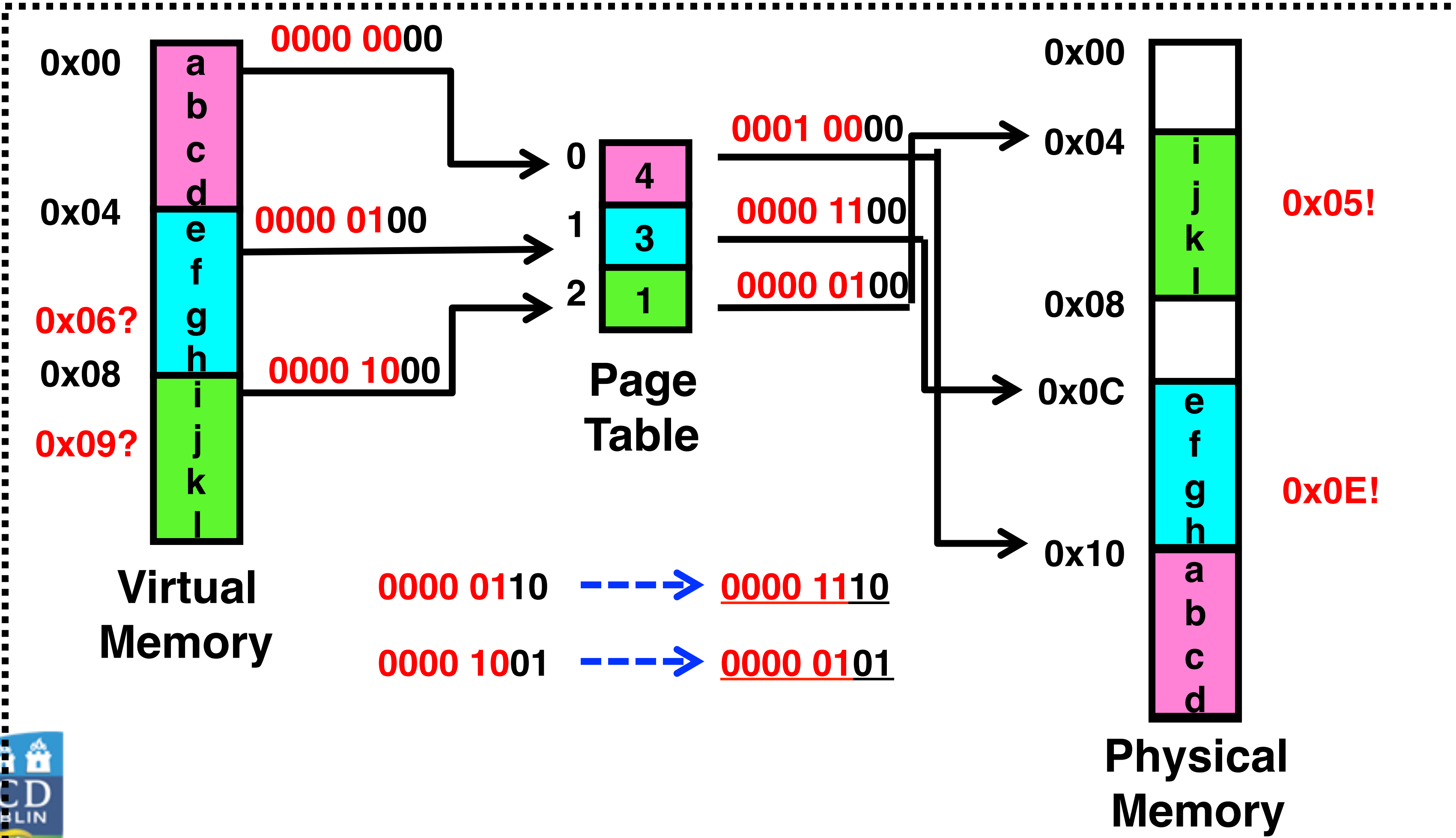


Paging Technique Features

- Fragmentation
  - **Internal:** only a fraction of the last page of a process
  - **External:** none (no need for compaction)
- Every logical address (i.e. CPU or process addresses) is divided into two parts
  - A ***page number*** and an ***offset*** within the page
- The page size is typically 2^n (e.g. 512 bytes – 16 MB); if the size of the logical address space is 2^m then
  - The $m - n$ high-order bits give the page number
  - The n low-order bits give the page offset
- A logical address is translated to a physical address using the processor hardware

Simple Page Table Example

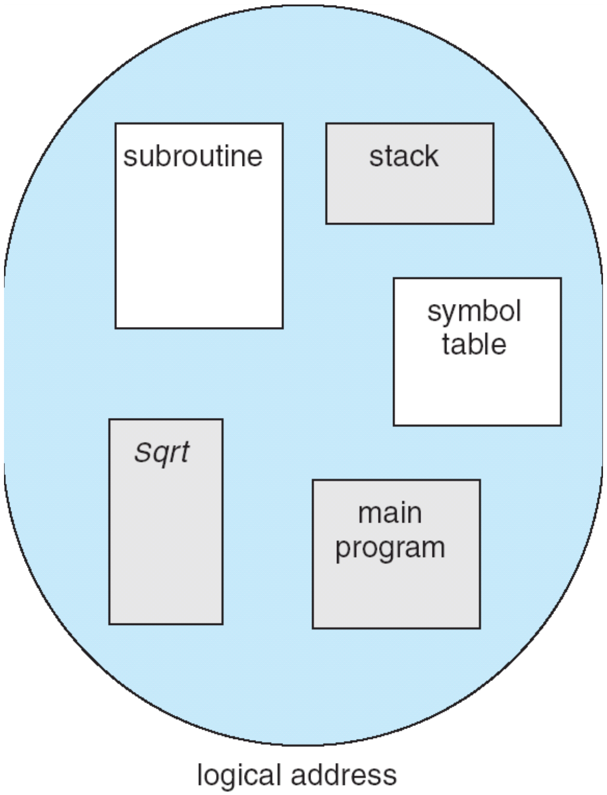
Example (4 byte pages)

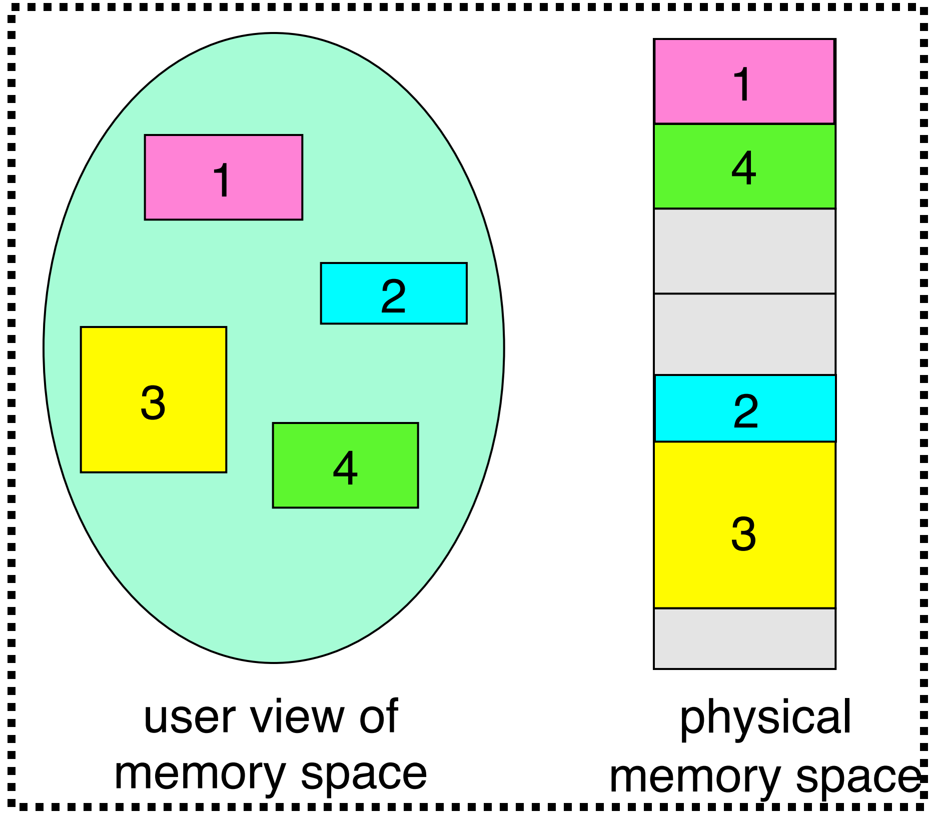


Paging Hardware Optimisation

- Page table usually big (i.e. 10^6 entries): kept in main memory
- But this is slow with respect to using registers: ***translation look-aside buffer (TLB)*** used to improve performance
  - A TLB is an associative high-speed memory, which associates a key (tag) with a value
  - When presented with a key, it compares it with all keys ***simultaneously***
  - It needs a replacement policy (what happens when it is full?)
  - TLB size: 64-1024 values

Non-contiguous Allocation (Variable): Segmentation



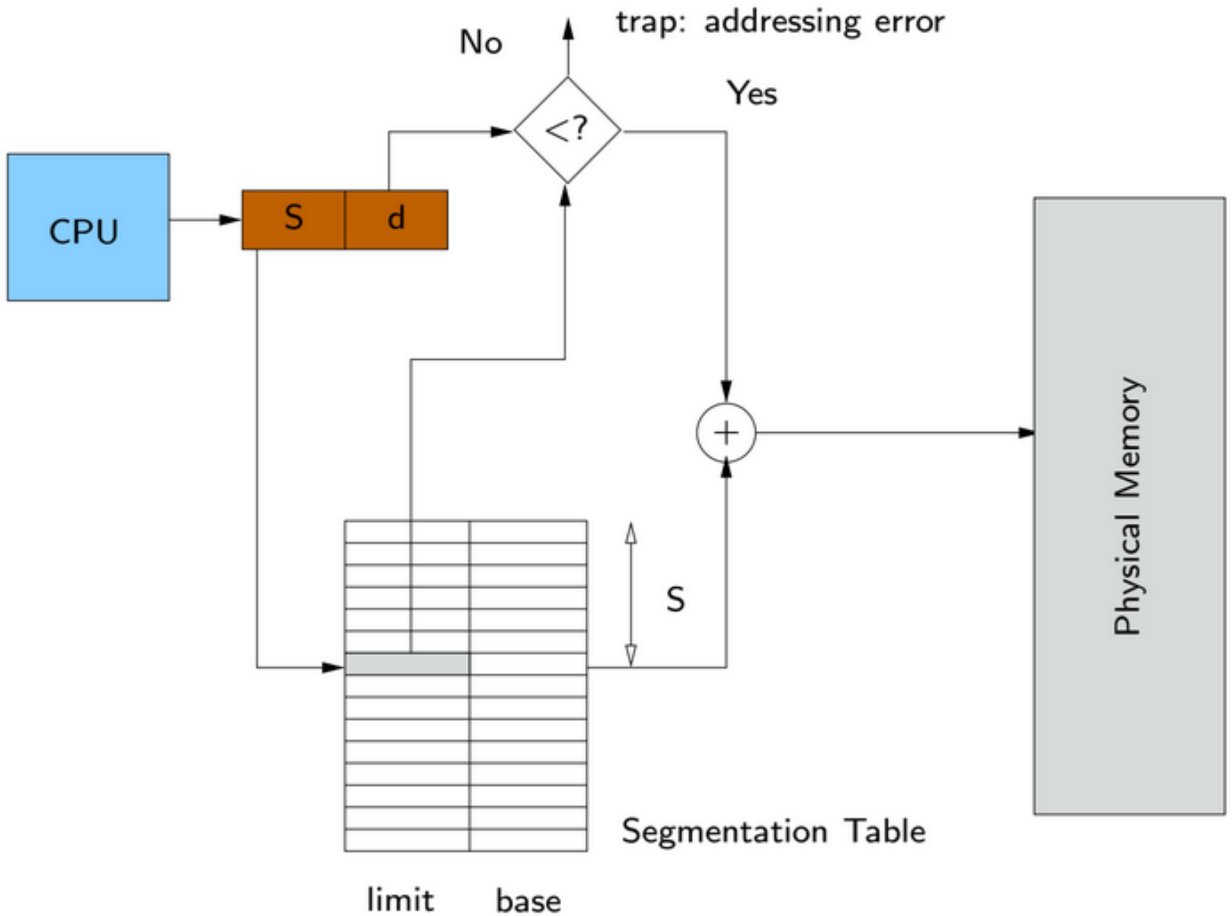


- Logical View: multiple separate segments
  - Typical: Code, Data, Stack
  - Others: memory sharing, etc
- Each segment is given region of contiguous memory
  - Has a base and limit
  - Can reside anywhere in physical memory

Segmentation Technique

- **Basic strategy:** the logical memory is divided into a number of segments, each of possibly different length
  - Logical address consists now of a ***segment number*** and an ***offset*** within the segment
  - More complex relationship between logical and physical address
- Entries in the segmentation table include the base and limit registers for a segment
  - Association of protection with the segments
- Fragmentation:
  - **Internal**: none
  - **External:** not solved, but less severe than variable partitioning because of the smaller pieces a process is divided into

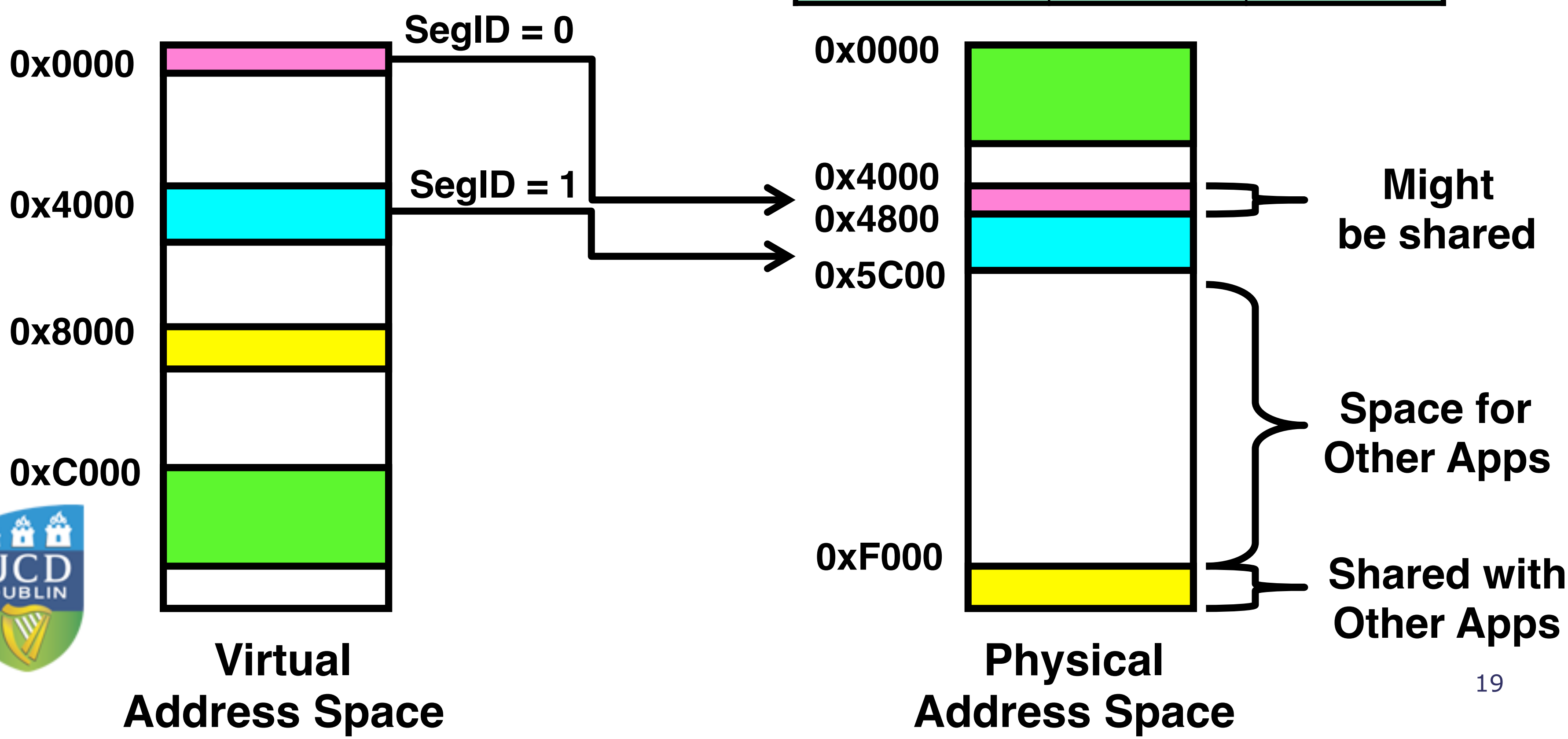
Segmentation Hardware



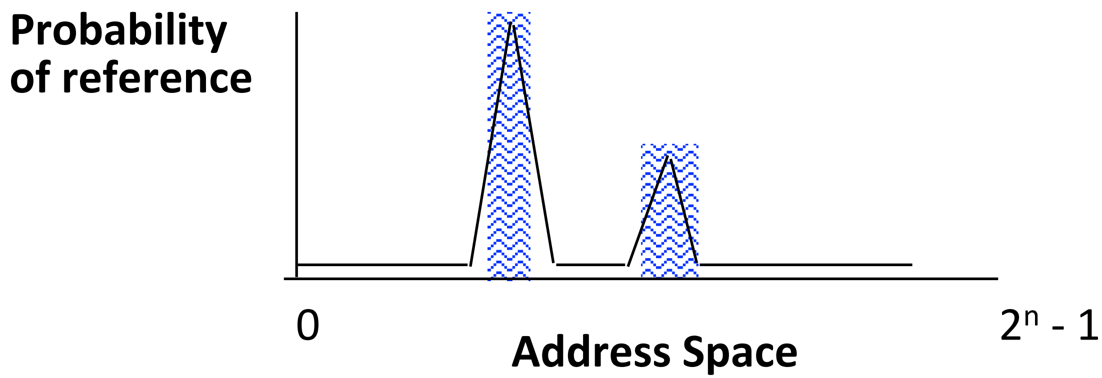
Example: 4 Segments (16 bit addresses)

Virtual Address Format

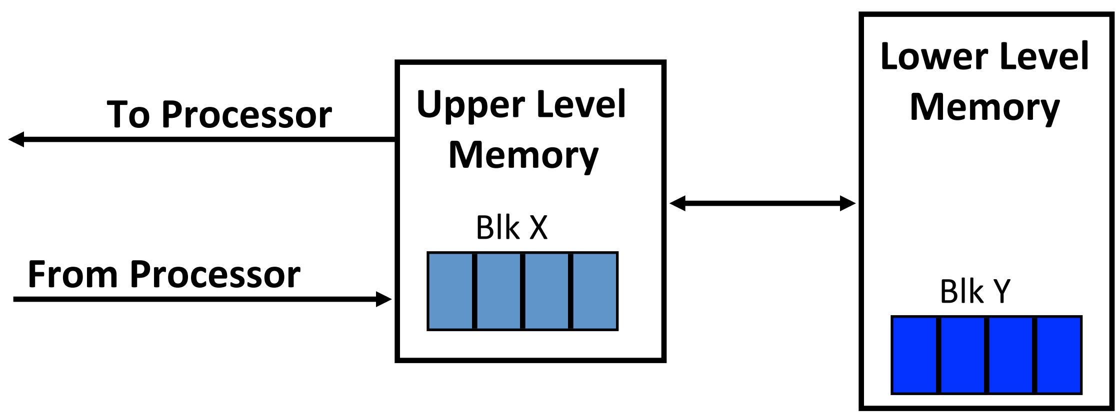
Seg ID #	Base	Limit
0 (code)	0x4000	0x0800
1 (data)	0x4800	0x1400
2 (shared)	0xF000	0x1000
3 (stack)	0x0000	0x3000



Caching and Locality

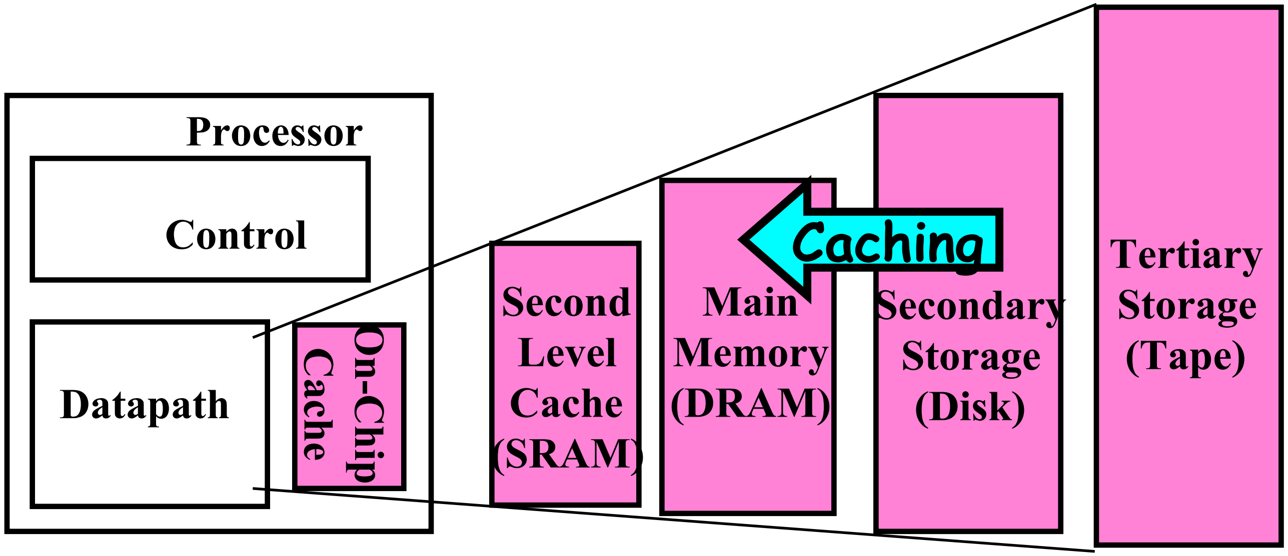


- ***Temporal Locality*** (Locality in Time):
  - Keep recently accessed data items closer to processor
- ***Spatial Locality*** (Locality in Space):
  - Move contiguous blocks to the upper levels

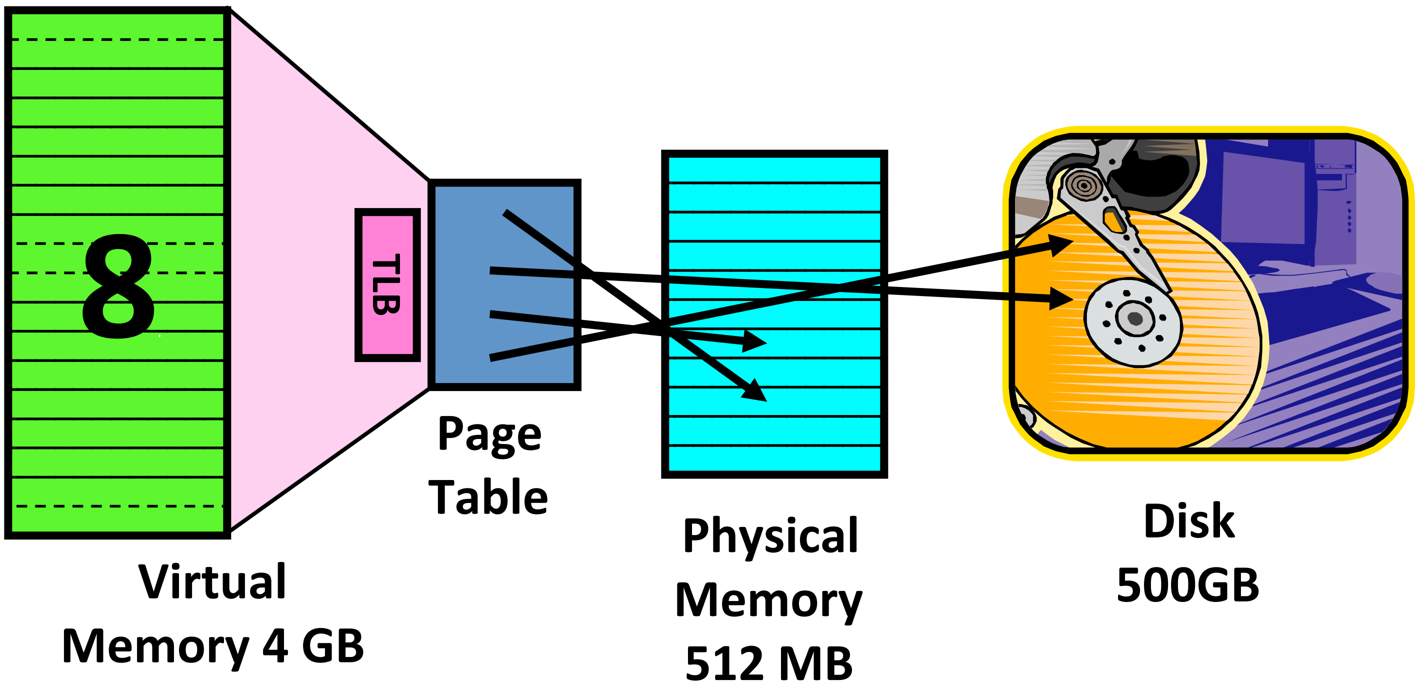


Demand Paging

- Modern programs require a lot of physical memory
  - Memory per system growing faster than 25%-30%/year
- But they don't use all their memory all of the time
  - 90-10 rule: programs spend 90% of their time in 10% of their code
  - Wasteful to require all of user's code to be in memory
- Solution: use main memory as cache for disk



Illusion of Infinite Memory



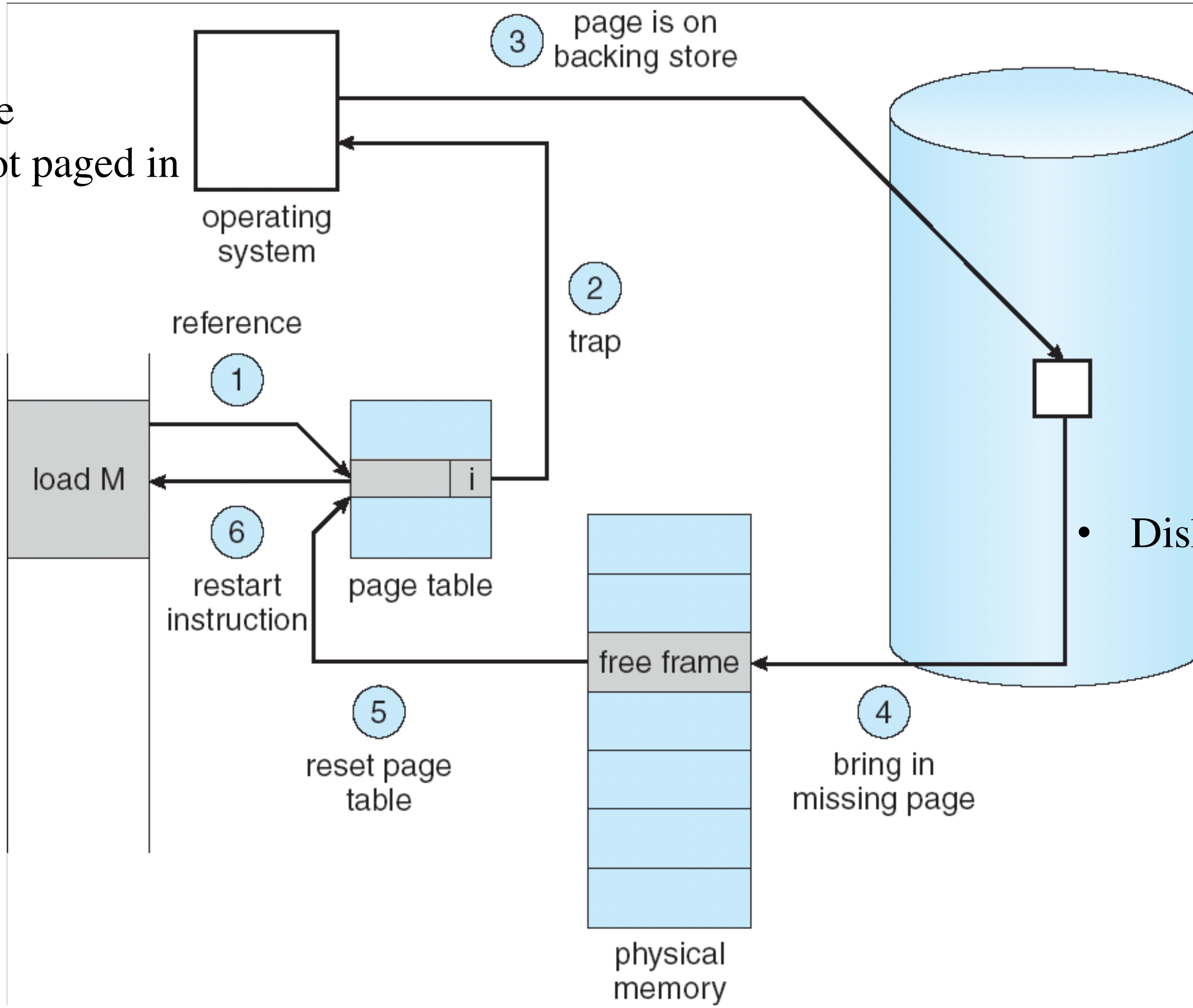
- Disk is larger than physical memory ⇒
  - In-use virtual memory can be bigger than physical memory
  - Combined memory of running processes much larger than physical memory
    - More programs fit into memory, allowing more concurrency
- Principle: ***Transparent Level of Indirection*** (page table)
  - Supports flexible placement of physical data
    - Data could be on disk or somewhere across network
  - Variable location of data transparent to user program
    - Performance issue, not correctness issue

Virtual Memory Features

- VM is commonly implemented by ***demand paging***
  - When a program is loaded, the OS brings into main memory only a few pages of it (including its starting point)
  - Further pages are then brought to memory or swapped to disk as needed
  - The ***resident set*** is the portion of the process that is in main memory at a given time
- VM system supported by
  - **Hardware:** paging mechanism, which generates ***page faults*** when pages in disk are referenced
  - **Software:** page swapping management (OS algorithm)

Steps in Handling a Page Fault

- if invalid
  - terminate
- if valid but not paged in
  - page in



- Disk operation

Optimal/Minimum (Min)

- Replaces page that will not be referenced forlongest period of time
- Minimum number of page faults, but impossible to implement (knowledge of future events required)
- Standard yardstick used to gauge other algorithms
- Example Suppose we have the same reference stream:
  - **A B C A B D A D B C B**

Ref:	A	B	C	A	B	D	A	D	B	C	B
Page: 1	A									C	
2		B									
3			C			D					

- Where will D be brought in? Look for page not referenced farthest in future
- MIN: 5 faults

LRU (Last Recently Used)

- Replaces the page that has not been referenced for the longest period of time
- By the principle of locality: it is likely that this page will not be referenced in the near future either
- Almost as good as the optimal policy, but difficult to implement (overheads associated to time keeping)
- Example: A B C D A B C D A B C D

Ref: / Page:	A	B	C	D	A	B	C	D	A	B	C	D
1	A			D			C			B		
2		B			A			D			C	
3			C			B			A			D

First In First Out (FIFO)

- Frames traversed as a circular buffer, triggered by replacements
- Pages are removed in round-robin style
- Rationale: a page fetched long ago may be now out of use (when main memory is composed by many frames)
- Simple to implement, but some replacements will not be good
- Example: Suppose we have 3 page frames, 4 virtual pages, and following reference stream:
  - A B C A B D A D B C B

Ref: Page:	A	B	C	A	B	D	A	D	B	C	B
1	A					D				C	
2		B					A				
3			C						B		

- FIFO: 7 faults.
- When referencing D, replacing A is bad choice, since need A again right away

Conclusion

- **Cache:** A repository for copies that can be accessed more quickly than original
- **Virtual Memory:** Illusion supported by system hardware and software that a process has a vast and linear expanse of available memory
- **Principle of Locality:** Program likely to access a relatively small portion of the address space at any instant of time.
  - ***Temporal Locality, Spatial Locality***
- VM is commonly implemented by ***demand paging***
- **Working Set:** Set of pages touched by a process recently
- **Resident Set:** Portion of process that is in main memory at a given time
- **Thrashing:** A process is busy swapping pages in and out
  - Process will thrash if working set doesn't fit in memory
  - Need to swap out a process

Conclusion (cont’d)

- A good replacement policy should exploit the principle of locality of references
- Replacement policies
  - **FIFO:** Place pages on queue, replace page at end
  - **MIN:** Replace page that will be used farthest in future (optimal)
  - **LRU:** Replace page used farthest in past
- Multi-Level Tables
  - Virtual address mapped to series of tables
  - Permit sparse population of address space
- Inverted Page table
  - Size of page table related to physical memory size