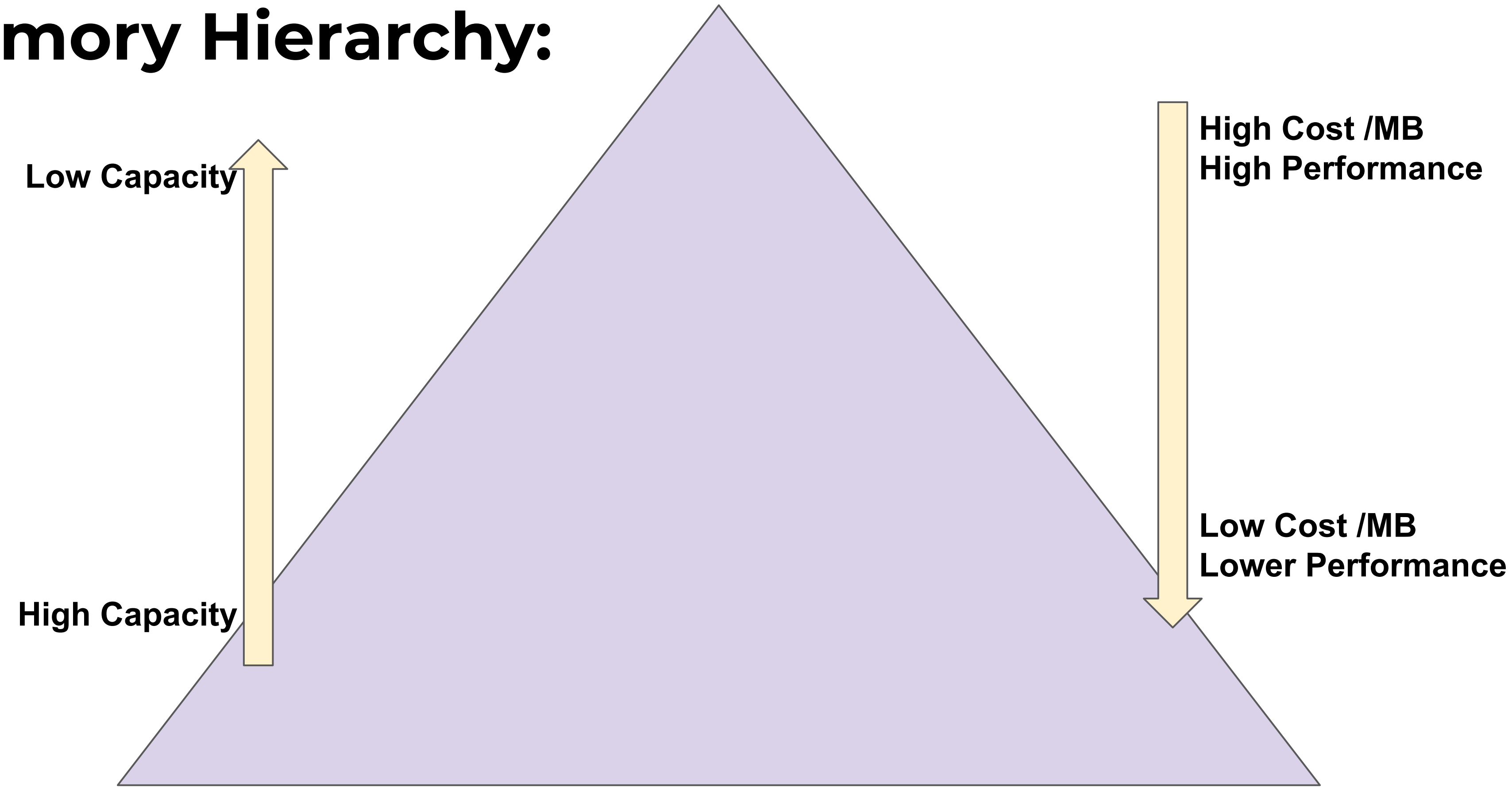
Memory Hierarchy

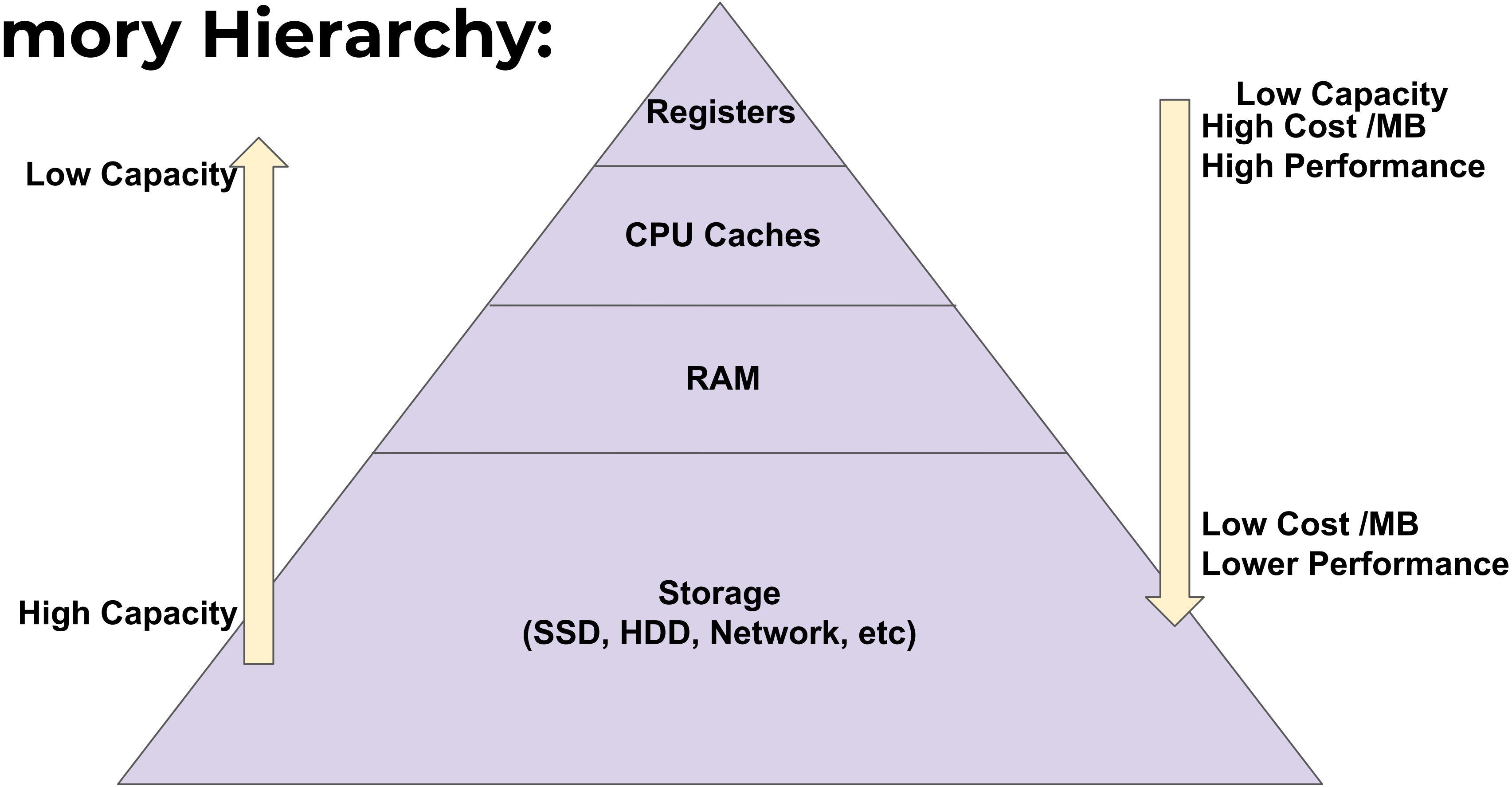
CS 423 - The University of Illinois

Wade Fagen-Ulmschneider

Memory Hierarchy:



Memory Hierarchy:



Memory Considerations

- ★ We have a **limited amount of fast** resources.
- ★ We have an **abundance of slow** resources.
- ★ How do we create an **allusion** of an **abundance of fast** resources?


ALMA

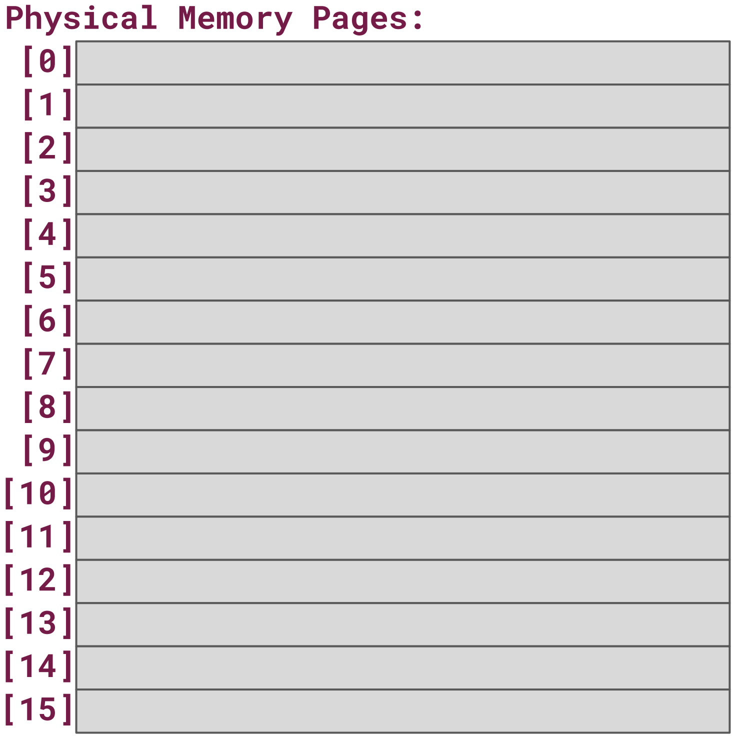
Memory Overlays

CS 423 - The University of Illinois

Wade Fagen-Ulmschneider

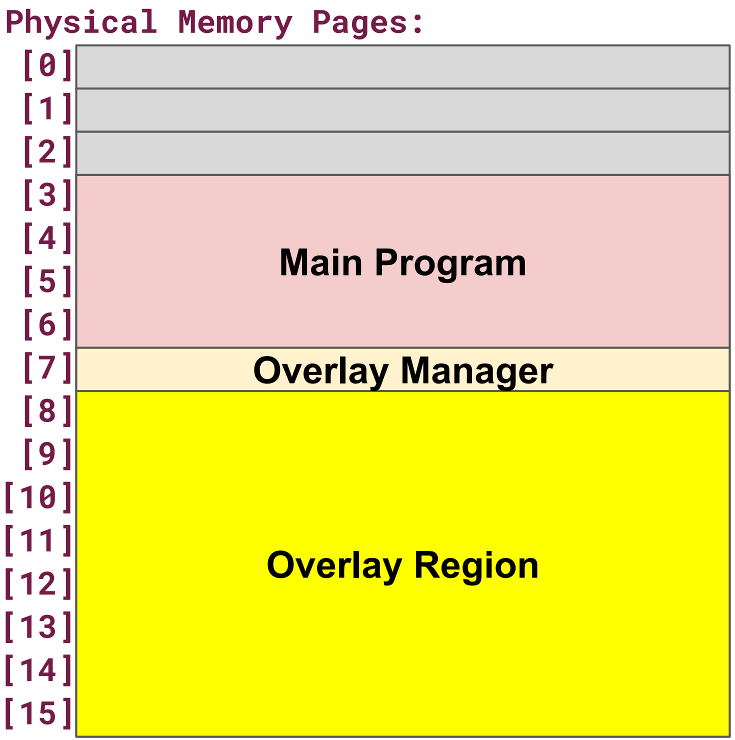
Overlays

- ★ **Memory overlays** are fixed-sized segments of data used when a program exceeds the available memory.
- ★ Simple, minimal complexity; implemented at compile-time.
- ★ Still used in embedded systems.



Overlays

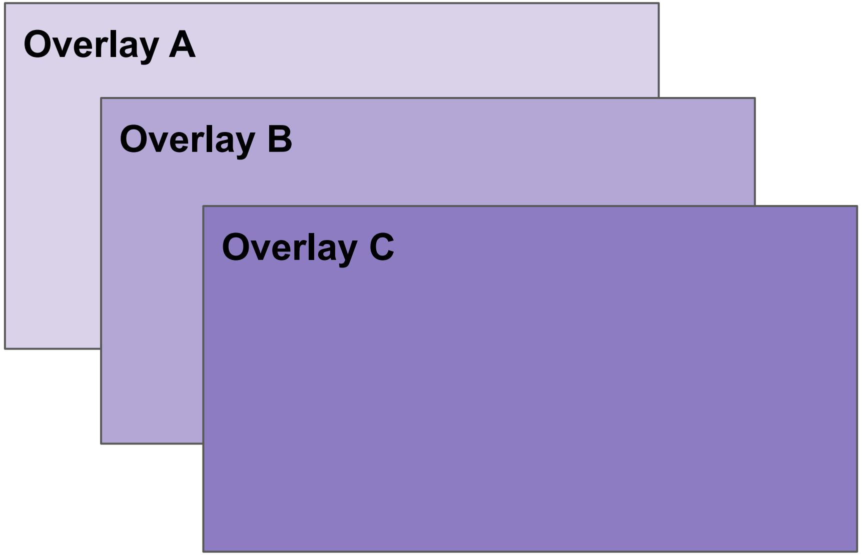
- ★ **Memory overlays** are fixed-sized segments of data used when a program exceeds the available memory.
- ★ Simple, minimal complexity; implemented at compile-time.
- ★ Still used in embedded systems.



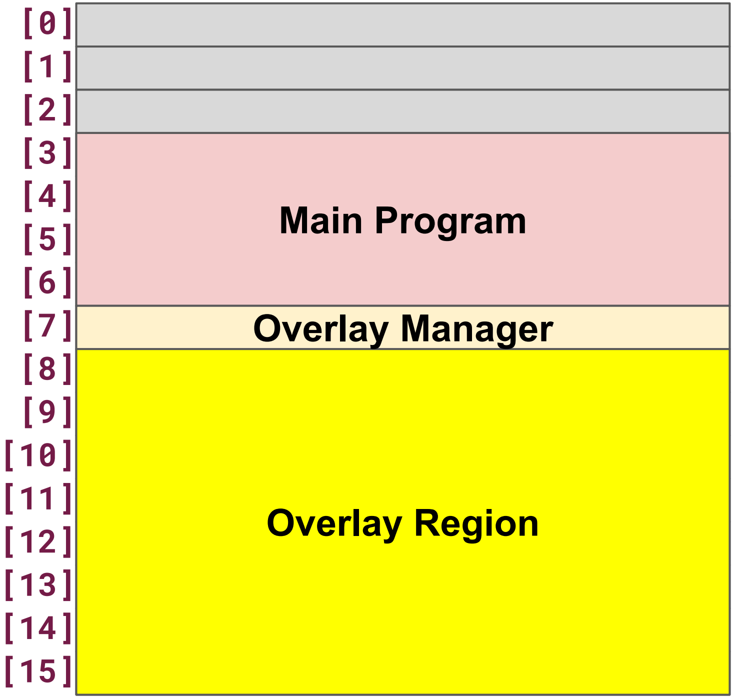
Overlays

Program's Overlays:

(Stored in secondary storage)



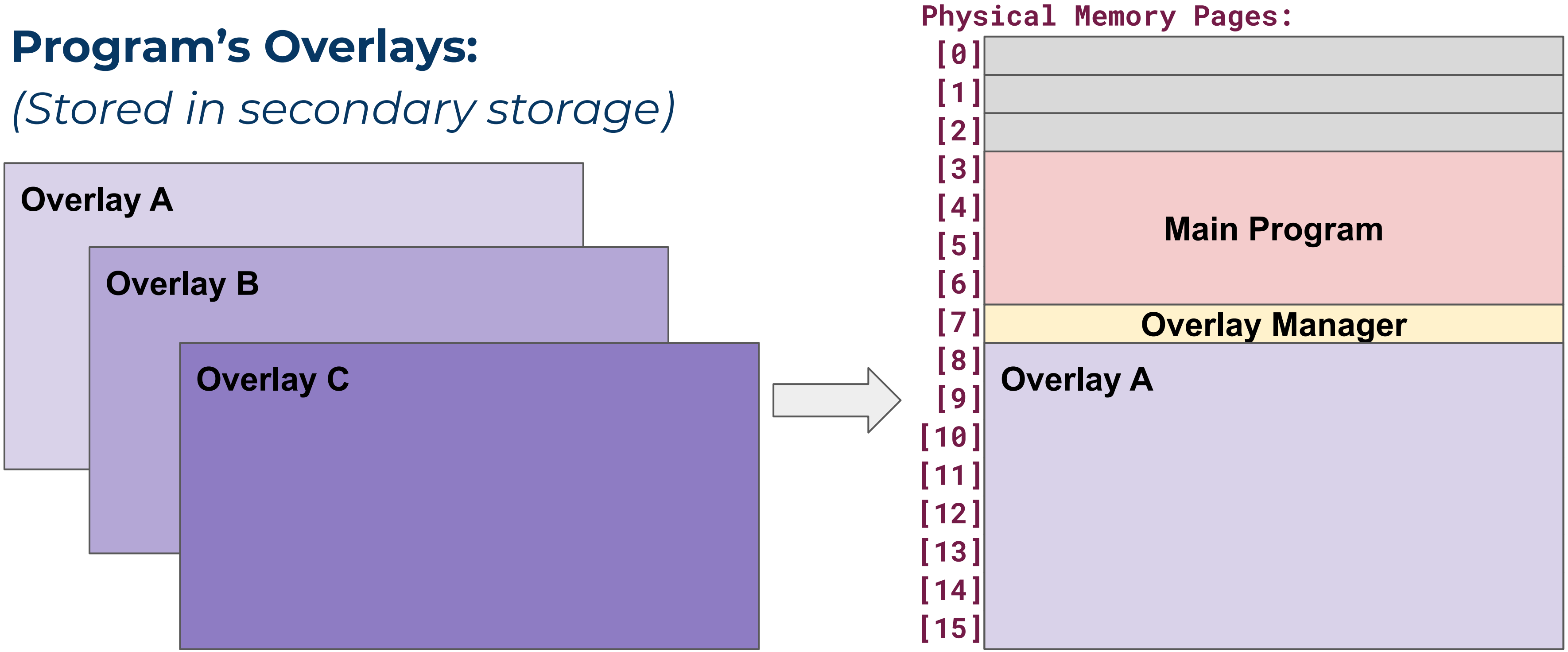
Physical Memory Pages:



Overlays

Program's Overlays:

(Stored in secondary storage)



Overlays

- ★ Systems may have multiple overlays and overlays are loaded in before they're required by the program code.
  - ○ All modern compilers/linkers support overlays.
  - ○ Compiled code target a specific overlay (ex: 2 x64 KB overlays).
- ★ **Disadvantages**:
  - ○ Fixed size segments (ex: 64 KB),
  - ○ Platform-specific (must compile for different segment sizes),
  - ○ Raw access to RAM; limited process isolation.

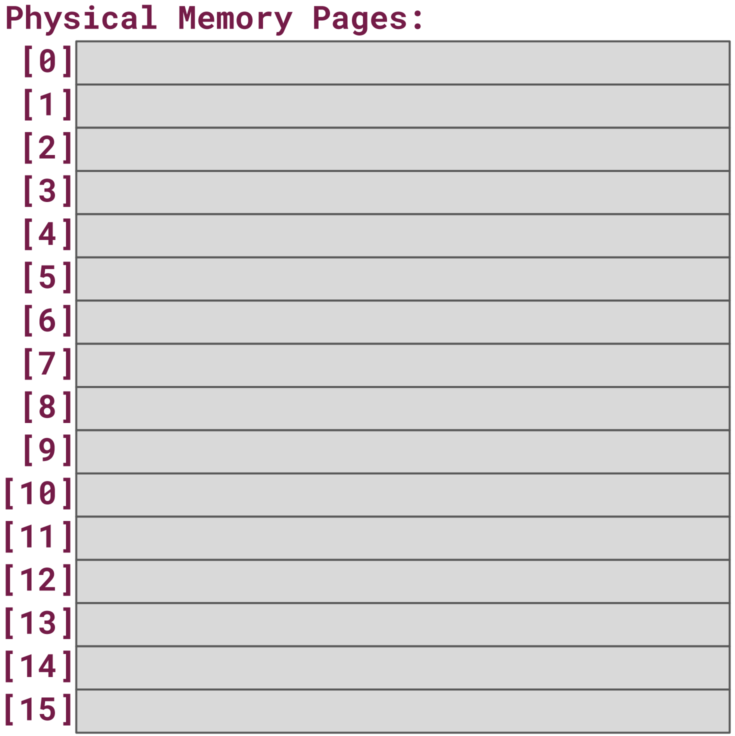
Fixed Partitions

CS 423 - The University of Illinois

Wade Fagen-Ulmschneider

Fixed Partitions

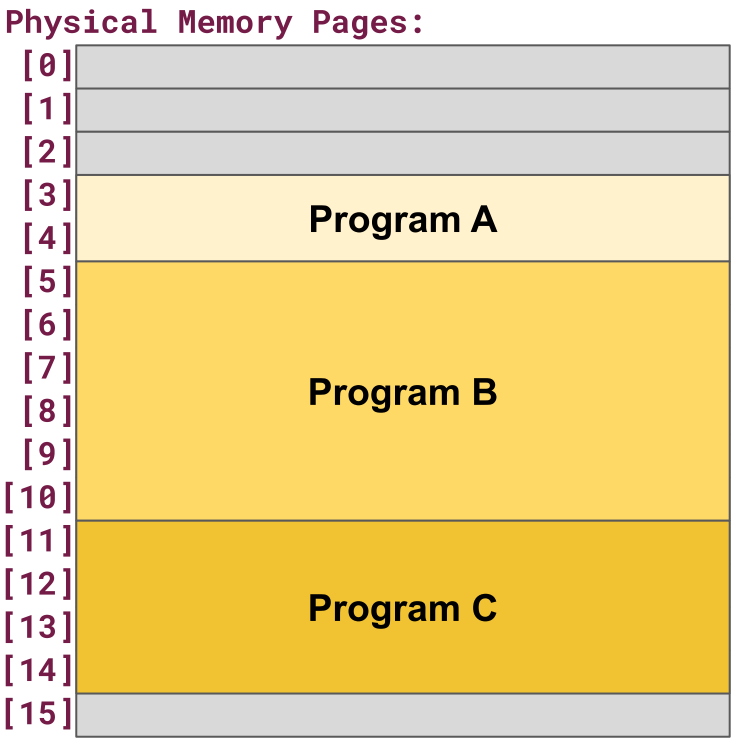
★ **Fixed Partitions** allocate a fixed amount of physical RAM to every process in a fixed location.



Fixed Partitions

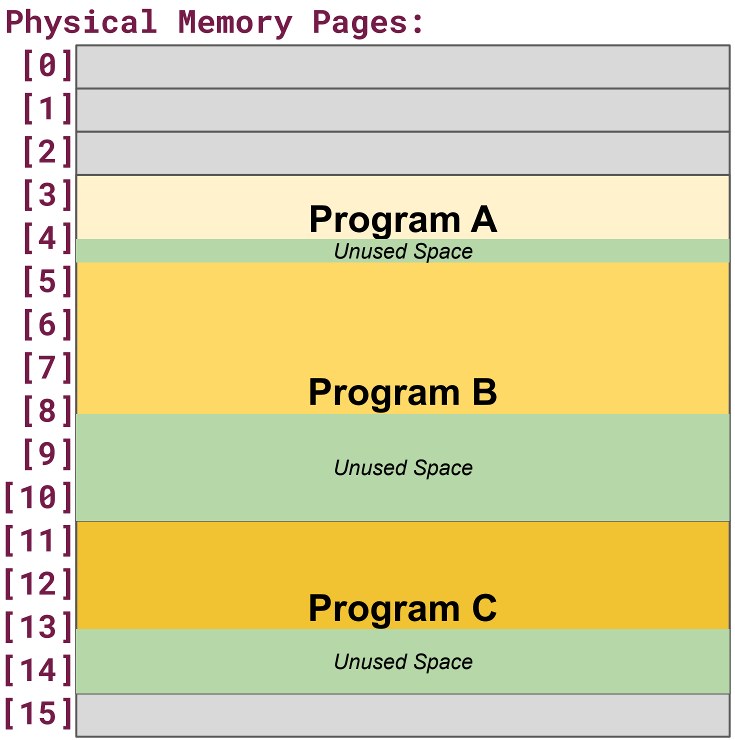
★ **Fixed Partitions** allocate a fixed amount of physical RAM to every process in a fixed location.

★ On creation, each process declares the **maximum** memory space it may need.
  - OS allocates a sequential amount of space for the process.



Fixed Partitions

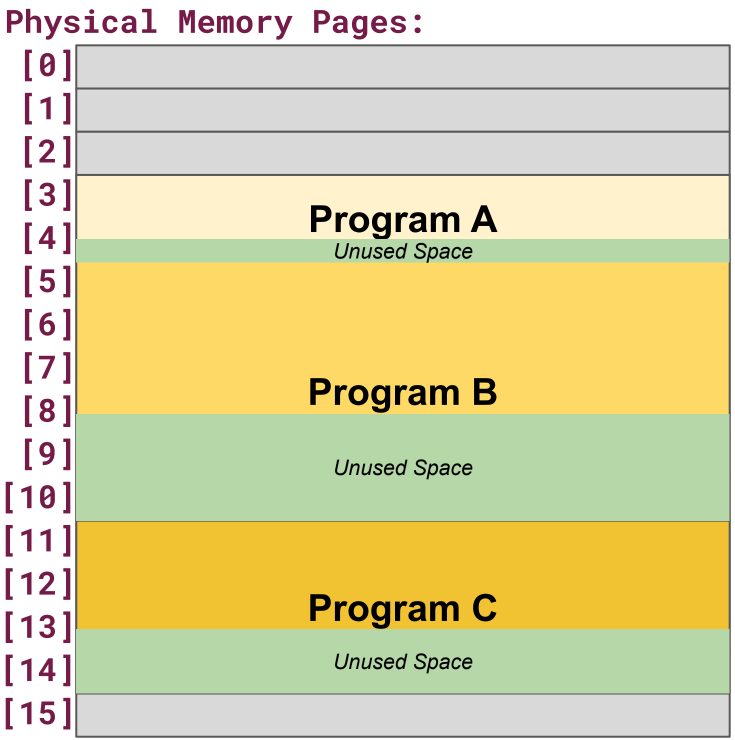
★ At any moment in time, a program may use only a part of its partition.



Fixed Partitions

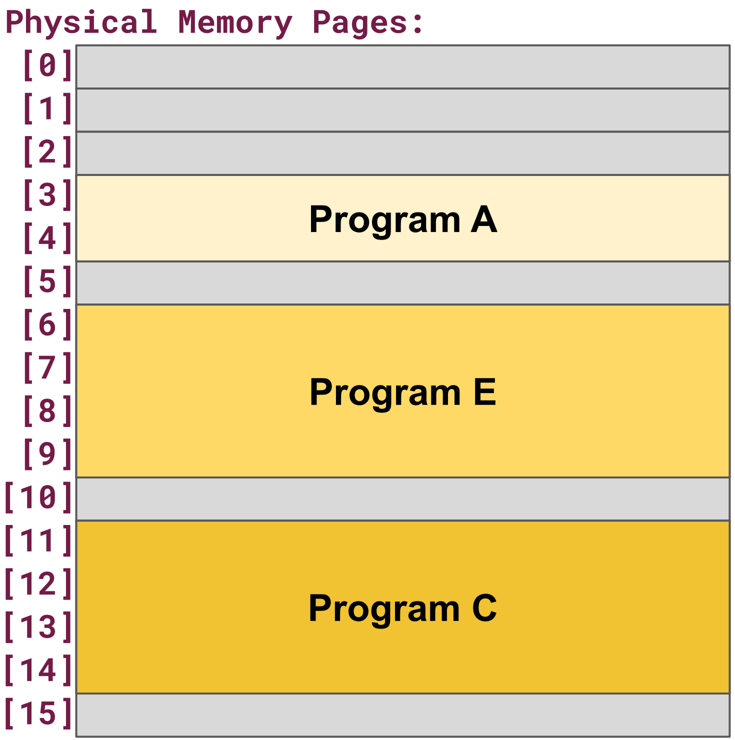
★ At any moment in time, a program may use only a part of its partition.

★ The unused space is **internal fragmentation** -- OS allocated the space, but process does not utilize it fully.



Fixed Partitions

- ★ Additionally, the RAM will become fragmented with various sized holes as processes enter/exit.
  - *Some processes creation may be blocked until a partition is available.*
- ★ Raw access to RAM; limited process isolation.

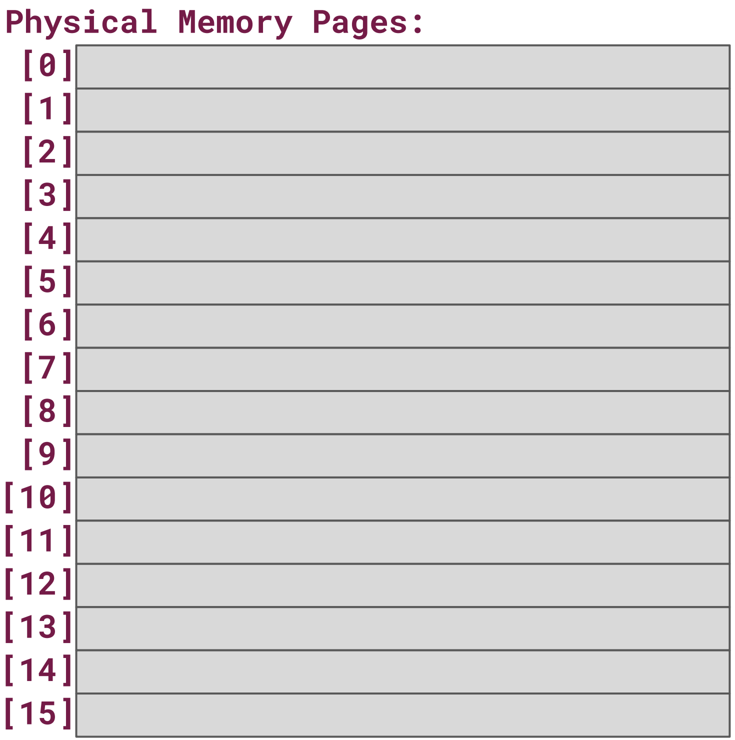


Relocation and Variable Partitions

CS 423 - The University of Illinois
Wade Fagen-Ulmschneider

Relocation

- ★ **Reallocation** provides a translation from a “offset (logical) address” to the physical address through the reallocation register.



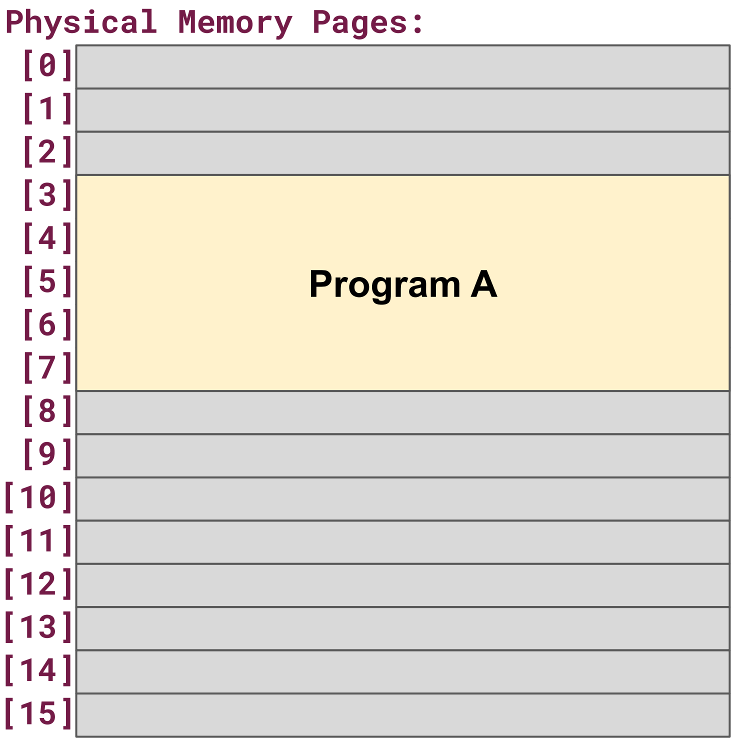
Relocation

- ★ All programs will address their memory from 0x0 ⇒ 0x{MAX}.

- ★ The "offset" or "logical" address would be translated into the physical address by relocating the request:

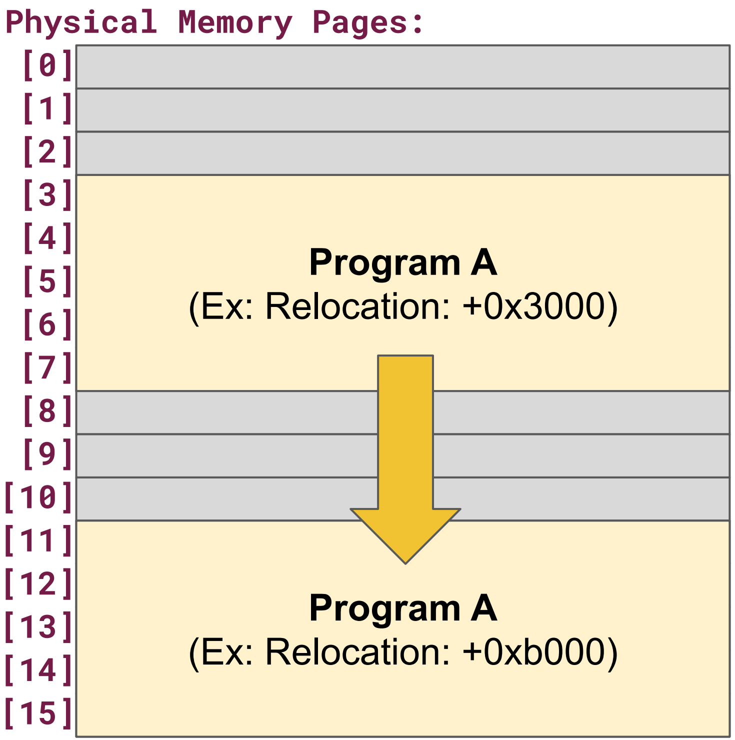
0x 3ac ⇒ **0x3c3a**3ac

+Relocation Register



Relocation

★ By changing the value of the relocation register, each process can now be moved around within RAM.



Relocation

- ★ First system with a “translation” between a “logical address” and the “physical address” in RAM.
  - **Disadvantage:** Still requires sequential memory to be committed.

- ★ **Overhead**: Single offset is needed to translate the page; the offset can be adjusted by the OS as needed. ***(Low overhead!)***

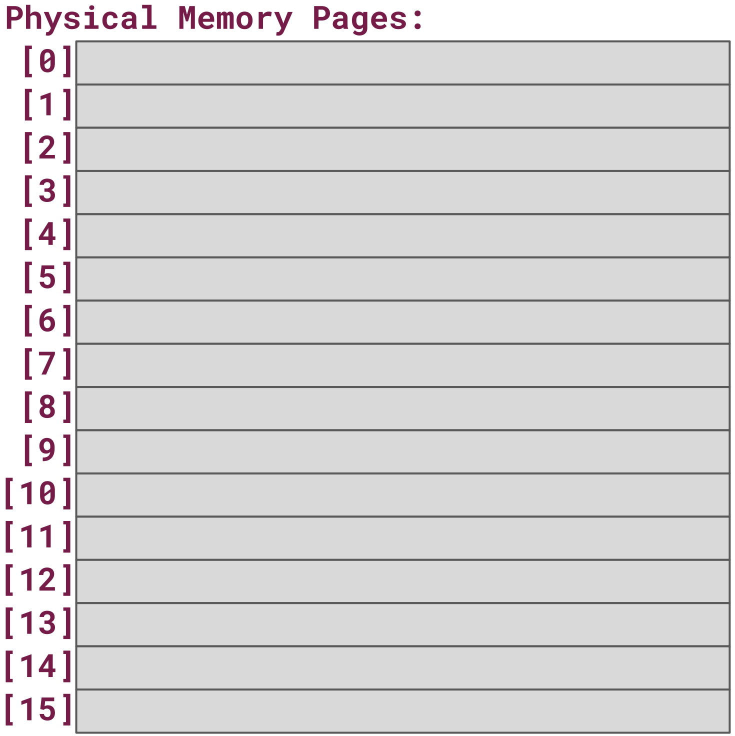

ALMA

Paging and Virtual Memory

CS 423 - The University of Illinois
Wade Fagen-Ulmschneider

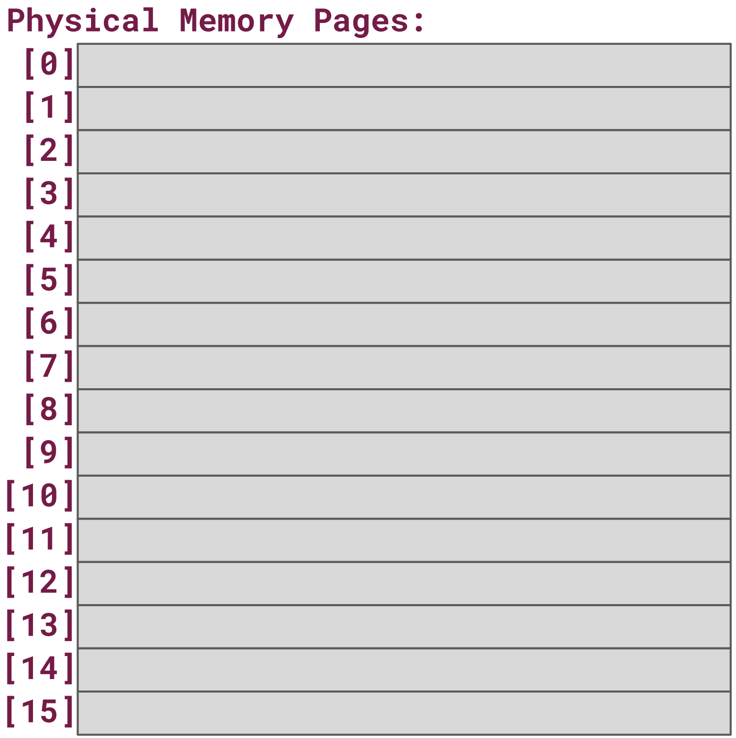
Paging

★ **Paging** is an extension of segmentation, where we divide all data on our system into **fixed-sized pages**.



Paging

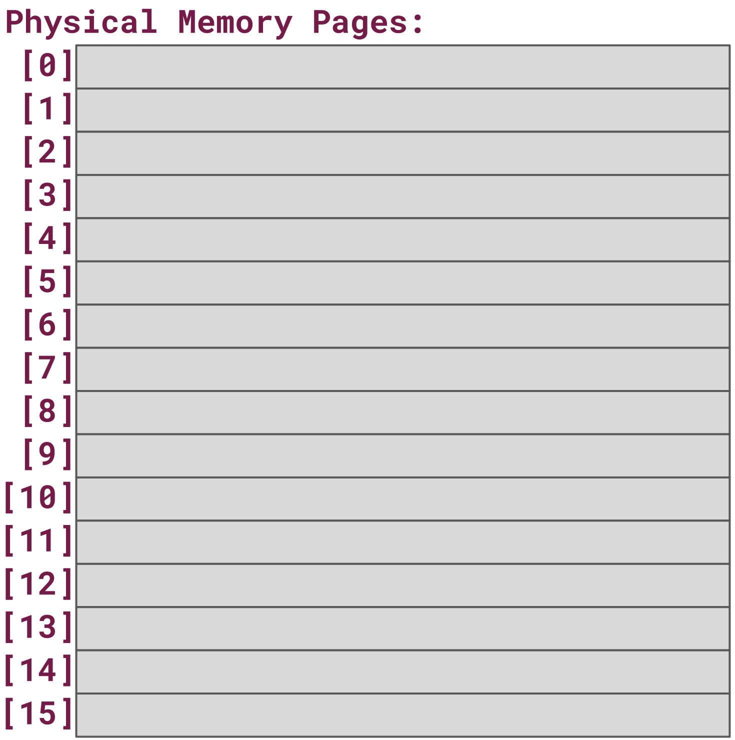
★ **Paging** is an extension of segmentation, where we divide all data on our system into **fixed-sized pages**.
  - Small enough to have minimal internal fragmentation.
  - Large enough to have minimal external fragmentation and overhead.



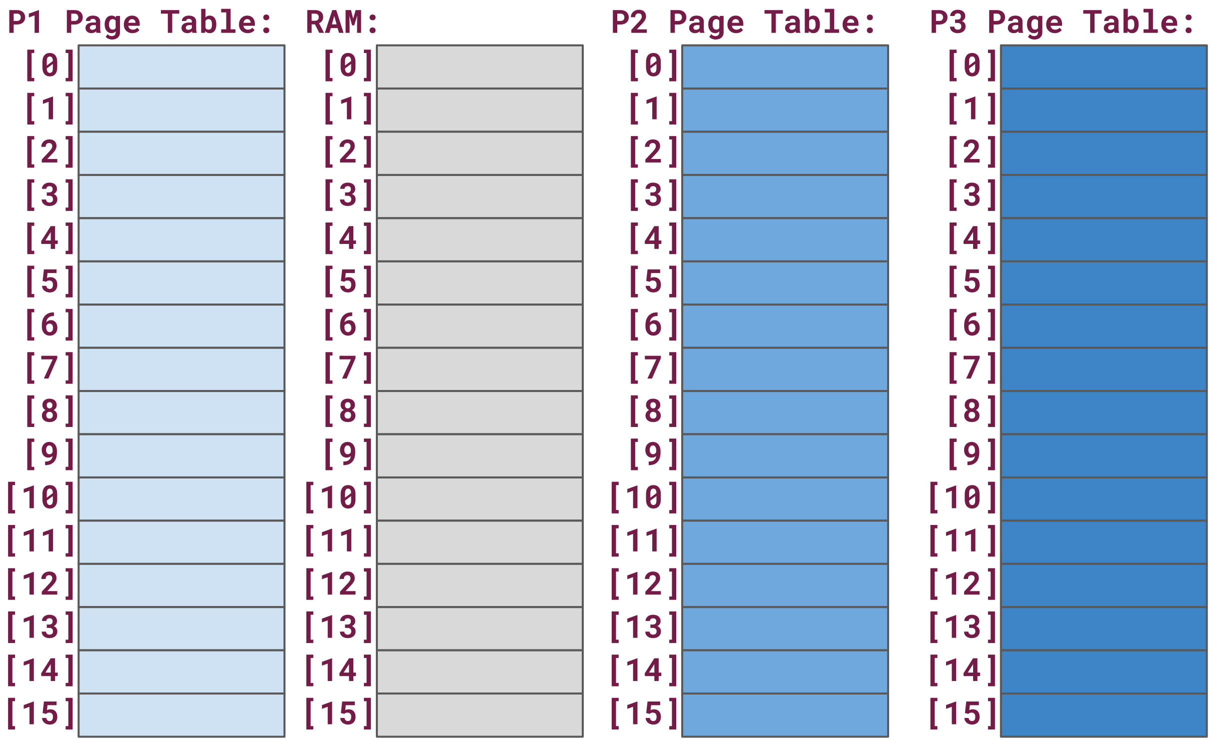
Paging

- ★ **Linux: `getconf PAGESIZE`**
  - Reports the size of the page on a system.
  - Most systems use 2^{12}, or 4096 B /page.
    - *I've started to see 2^{16} (64 KiB) used in the wild more and more.*

```
$ getconf PAGESIZE
4096
```



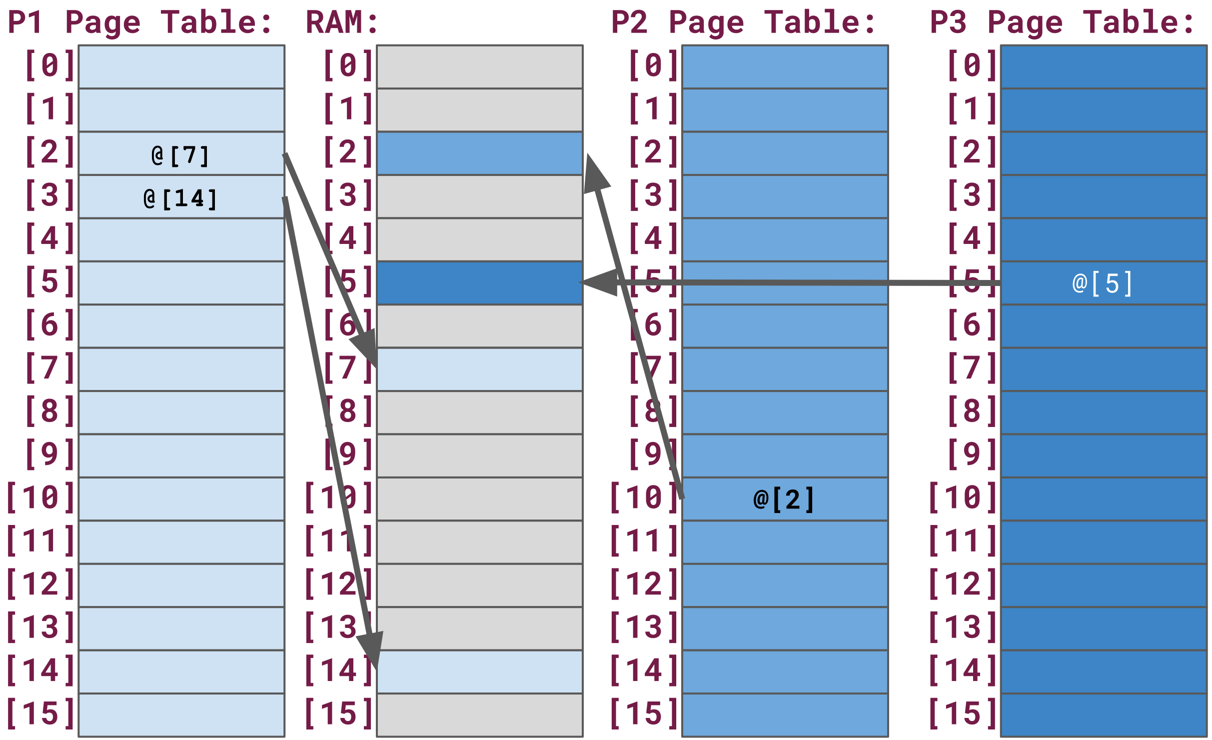
Page Tables


P1 Page Table:
RAM:
P2 Page Table:
P3 Page Table:
[0]
[1]
[2]
[3]
[4]
[5]
[6]
[7]
[8]
[9]
[10]
[11]
[12]
[13]
[14]
[15]

Page Table

★ Every process has its own **page table**.
  - Page table provides a translation between a “virtual address” used by the program and the “physical address” on RAM.
  - No user process will ever see the real physical address!

Page Tables



P1 Page Table:
RAM:
P2 Page Table:
P3 Page Table:
[0]
[1]
[2]
[3]
[4]
[5]
[6]
[7]
[8]
[9]
[10]
[11]
[12]
[13]
[14]
[15]
@[7]
@[14]
@[2]
@[5]

Page Table

★ Every "virtual address" now has two components:

- **Page Offset:** Where, within the page, is the data I'm addressing?
- **Page Number:** What index is our page within our virtual page table?

Page Table

- ★ If our page is 2^{12} bytes in size, the lowest 12 bits of a memory address it the **page offset**.
- ★ The remaining bits is the **page number**.

Page Table

- ★ If our page is 2^{12} bytes in size, the lowest 12 bits of a memory address it the **page offset**.

- ★ The remaining bits is the **page number**.

Memory Address: **`0x 32ac 51c16`**

```
Page Number: 0x 32ac51
Page Offset: 0x c16  (lowest 12 bits)
```

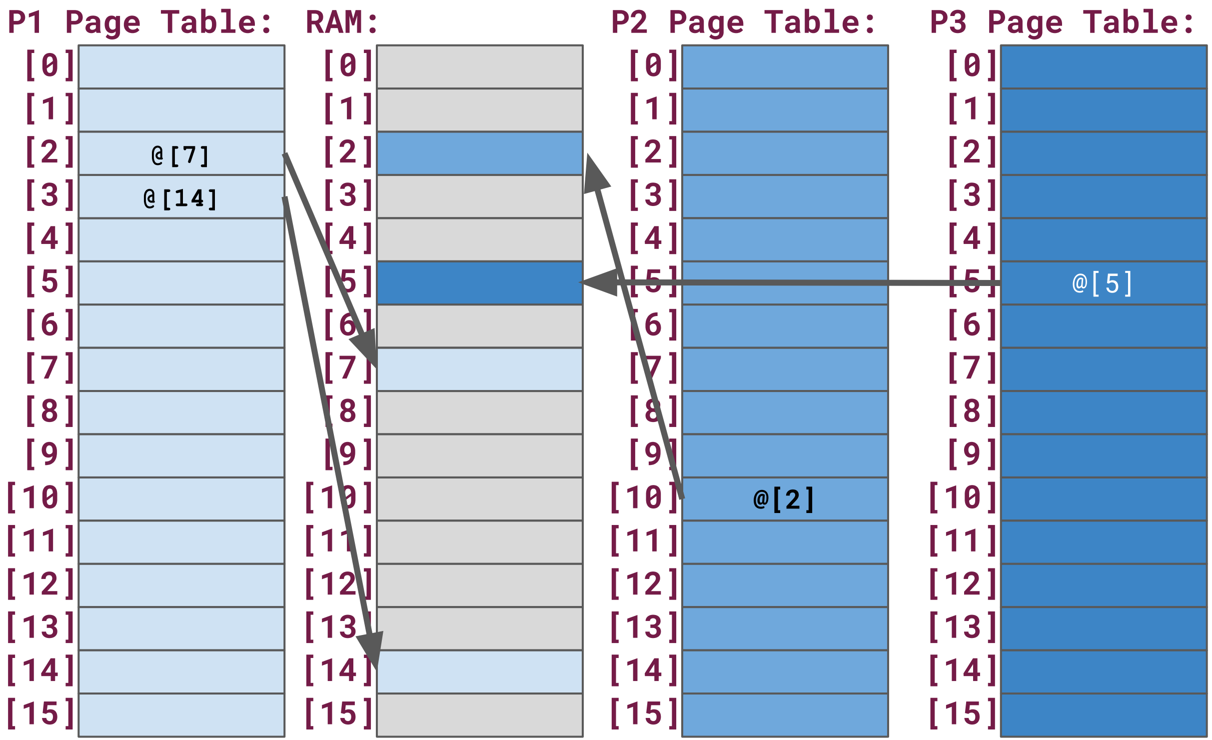
Page Table

Memory Address: **0x 32ac 51c16**

```
Page Number: 0x 32ac51
Page Offset: 0x c16  (lowest 12 bits)
```

★ The page table entry at **0x32ac51** will provide the translation to the physical address in RAM.

Page Tables



P1 Page Table:
RAM:
P2 Page Table:
P3 Page Table:
[0]
[1]
[2]
[3]
[4]
[5]
[6]
[7]
[8]
[9]
[10]
[11]
[12]
[13]
[14]
[15]
@[7]
@[14]
@[2]
@[5]

Page Table

- **Advantages:**
    - Processes can have the **allusion of sequential memory** even though the pages may be located in *(translated to)* various locations throughout RAM.

    - **Pages do not have to always be “present” in RAM**; can point to data on storage and load it in RAM when needed.
        - Need a mechanism to load pages in when needed.

    - Processes have no direct access to RAM; allows OS to provide protections to RAM.

Page Table

- ★ **Disadvantages:**
  - Overhead:
    - Consider 4 GiB of RAM divided into 4 KiB pages:

      4 GiB / 4 KiB == 1 MiB pages (!!)

      ...each process has its own 1 MiB pages!
  - Complexity: Non-trivial to translate addresses.


ALMA

Page Faults and Page Evictions

CS 423 - The University of Illinois

Wade Fagen-Ulmschneider

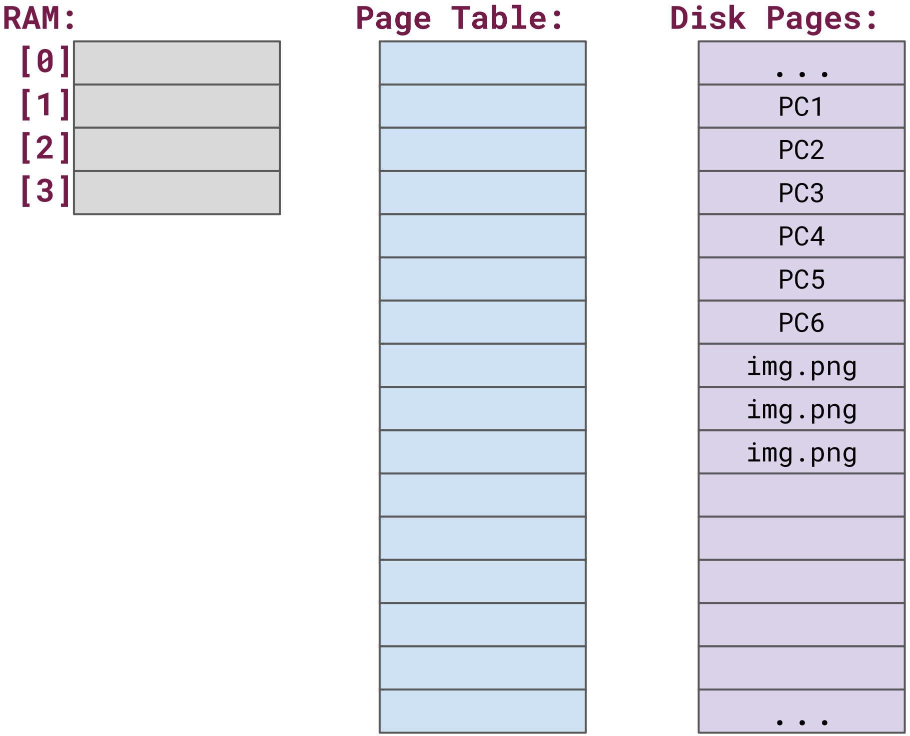
Virtual Memory

1: Load Program

2: Run PC1
- malloc 4 KB

3: Run PC2:
- malloc 10 KB
- Open img.png
- Read all of image

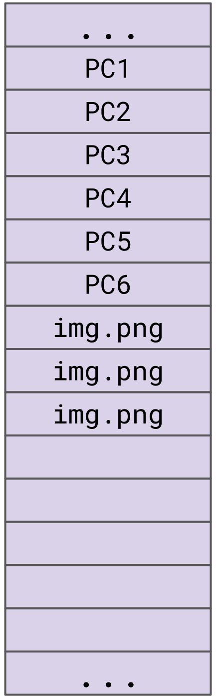
4: Run PC3
- Access OG 4 KB
- Finish program



Virtual Memory Analysis

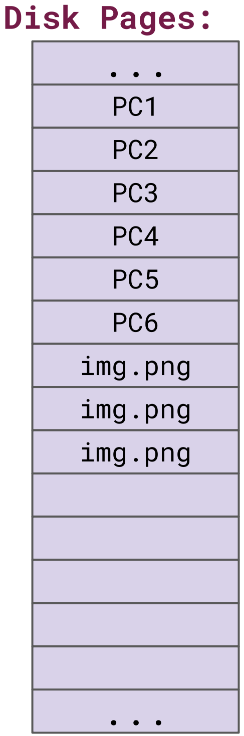
What is the range of possible file sizes for img.png?

Disk Pages:

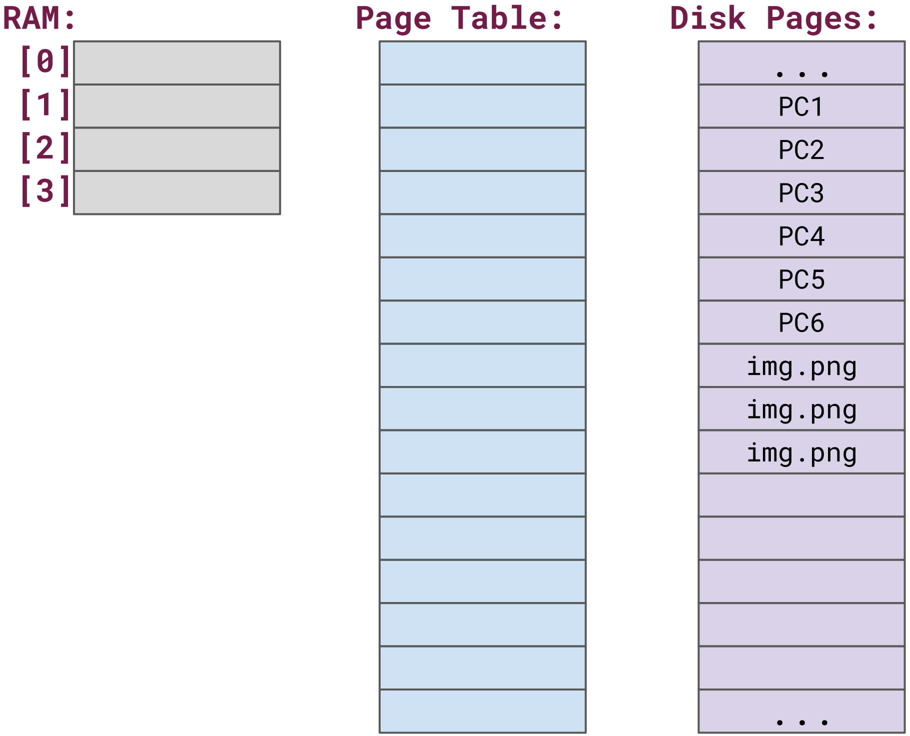


Virtual Memory Analysis

What is the range of possible file sizes for ./programCode (“PC”)?



Virtual Memory



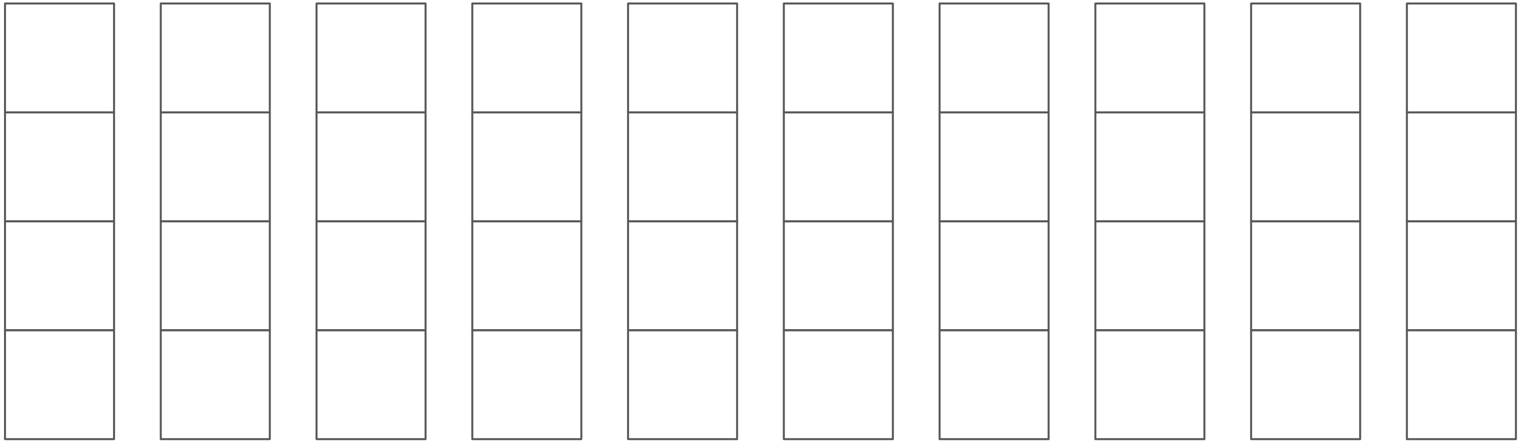
1: Load Program

2: Run PC1
- malloc 4 KB

3: Run PC2:
- malloc 10 KB
- Open img.png
- Read all of image

4: Run PC3
- Access OG 4 KB
- Finish program

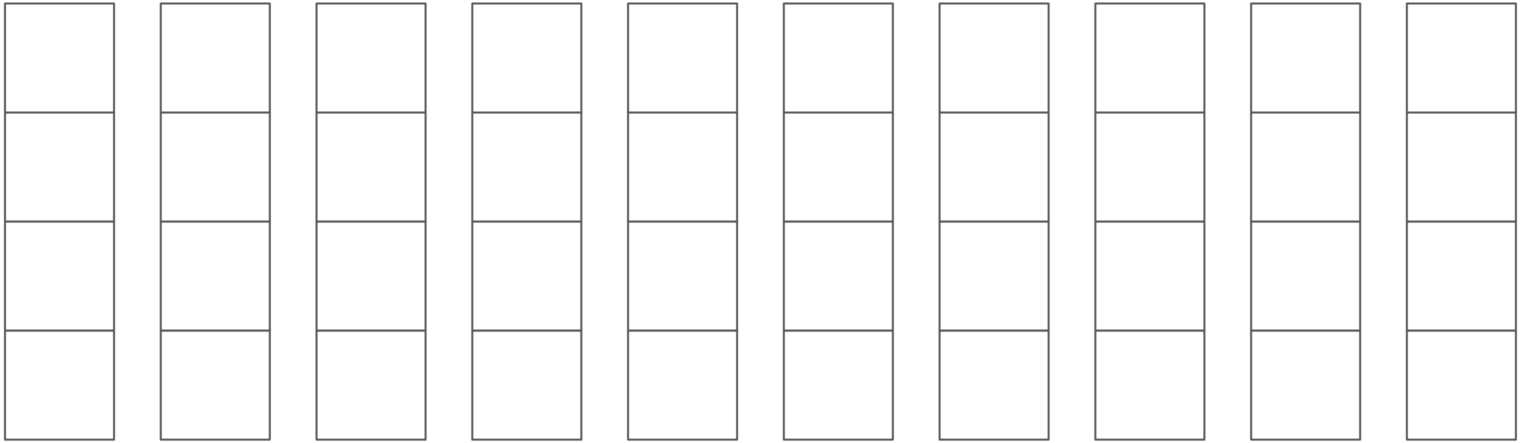
Page Eviction/Replacement Strategies:



Page Access: 17 33 40 17 43 8 99 33 99 17

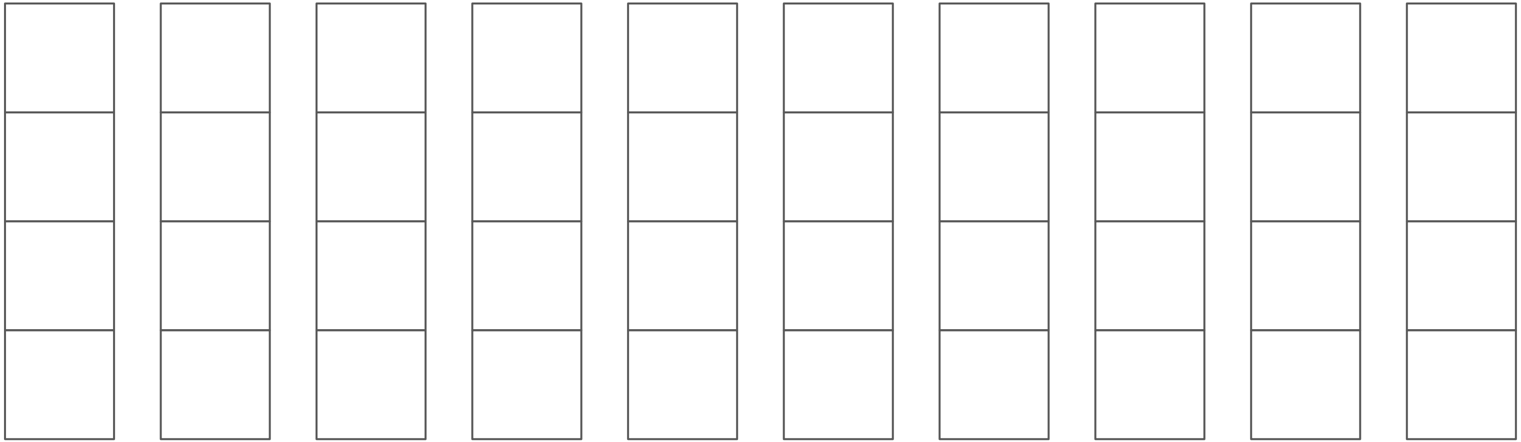
Page Faults

Page Eviction/Replacement Strategies:



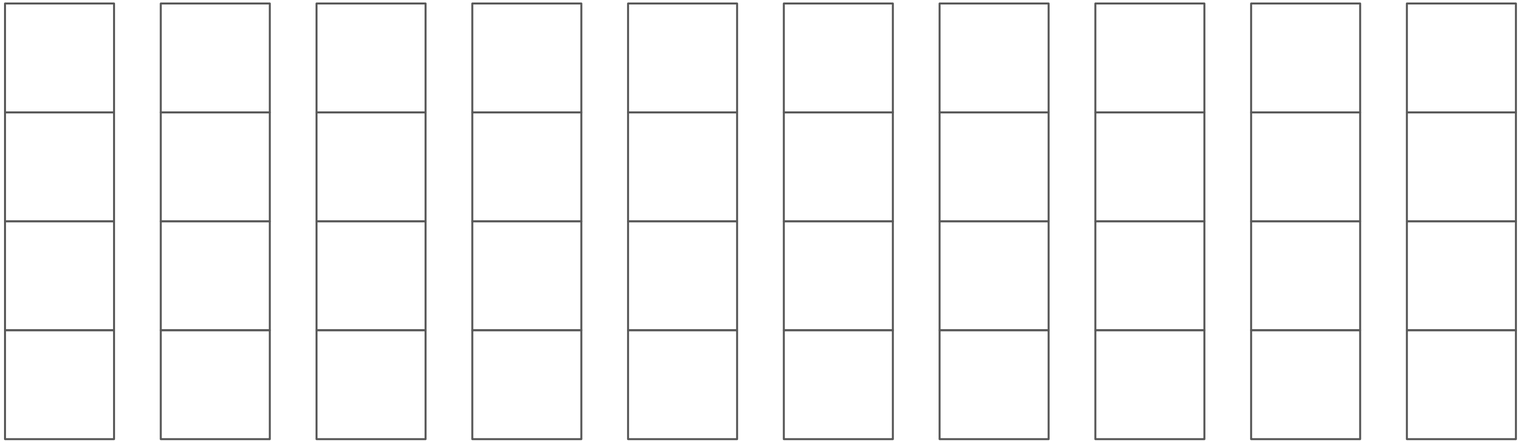
Page Access: 17 33 40 17 43 8 99 33 99 17

Page Eviction/Replacement Strategies:



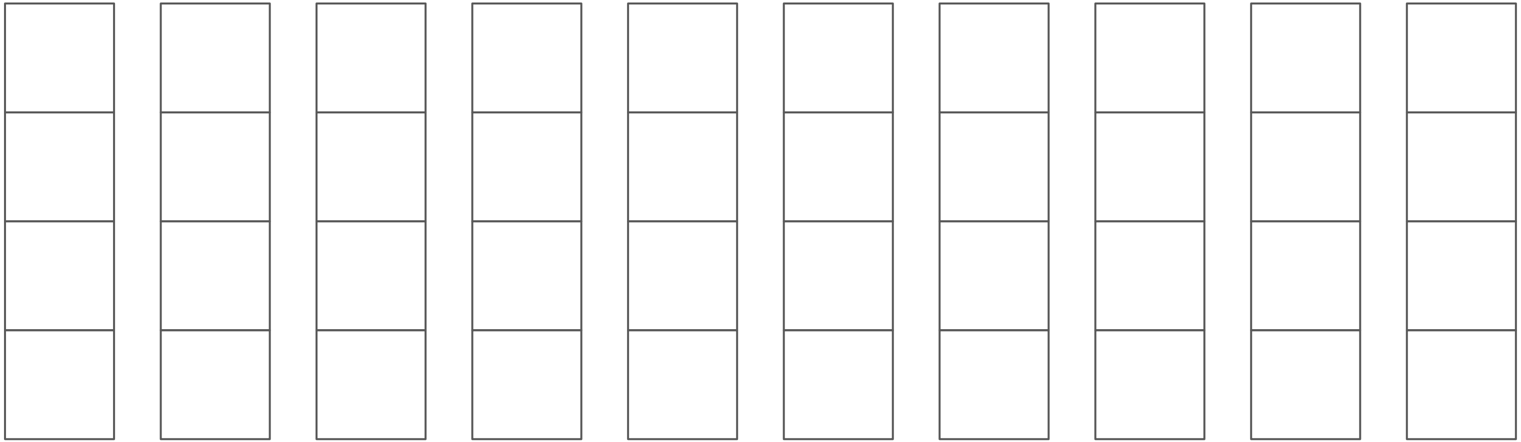
Page Access: 17 33 40 17 43 8 99 33 99 17

Page Eviction/Replacement Strategies:



Page Access: 17 33 40 17 43 8 99 33 99 17

Page Eviction/Replacement Strategies:



Page Access: 17 33 40 17 43 8 99 33 99 17