Virtual Memory

OS: The Memory Illusionist

- Segmentation and Paging
  - Code and data **appear to be** *contiguous* in RAM
- Virtual Memory
  - *In addition to contiguous view of code and data*
  - Your process **seems to have** access to terabyte* of memory (*much bigger* than the amount of installed RAM)
  - Your process **seems to reside** in RAM ***all* the time**

(Magician + Stage Props) ⇔ (Operating System + MMU)

Logical / *Virtual* address space

```
~ ssh eos lscpu
Architecture:                 x86_64
CPU op-mode(s):               32-bit, 64-bit
Byte Order:                   Little Endian
Address sizes:                39 bits physical, 48 bits virtual
CPU(s):                       8
```

```
→ ~ ./a.out
Address of main() at runtime is 0x562bb6130149      12 hex digits = 48 bits
→ ~ free --giga
              total        used        free      shared  buff/cache   available
Mem:             16           1          11           0           2          14
Swap:             4           0           4
```

- Logical/Virtual address (highlighted in yellow) has 12 hex digits or 48 bits
  - Amount of accessible memory is $2^{48} = 2^8 \times 2^{40}$ =256 Terabytes
- Total **physical** RAM is only 16 Gigabytes
  - 16 Gigabytes = 2^{34}

$$\frac{\text{Virtual Mem}}{\text{RAM size}} = \frac{2^{48}}{2^{34}} = 2^{14} \approx 16 \times 10^3 \text{ times}$$

How humans read a (printed) book

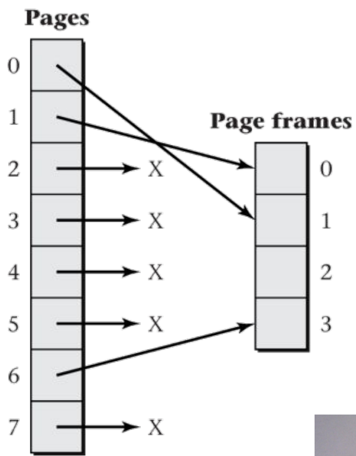
One page at a time?

One word at a time?

Observations

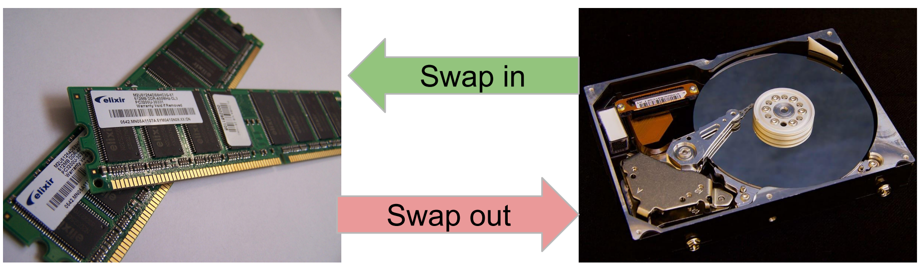
- **Not all pages** (of a process) have to be resident in RAM
  - For the CPU to work properly, it must have access to
    - The **current** instruction
    - The **current** set of data used by the instruction
      - Stack, heap, or data section
  - In a program that a huge array, the loop the manipulates it inspects only a small number of elements (two or three)
- Only pages currently needed by the CPU have to be resident in RAM
  - Other pages may reside on the *swap space/swap disk/paging disk*
- **Dynamic Loading** / **Demand Paging** allows the CPU to run a process that is only *partially resident* in memory

Components of Virtual Memory



Demand Paging (HW + SW)

Page Replacement Algorithm (SW)



Demand Paging

- Load (virtual) pages to RAM **only when** they are used/referenced
- Must coordinate loading with a (**lazy**) **pager daemon**
- To disambiguate: swapper vs. pager
  - Swapper: swap the **entire** process (slooooooower)
  - Paper: swap only pages of a process (faster)
- I/O operations to swap disk are usually faster than I/O to user filesystems
  - Swap disks are not organized into hierarchical directory structures, binary data from user pages are stored in a "flat" structure
  - No directory traversal required to access a page from the paging/swap disk

Swapping

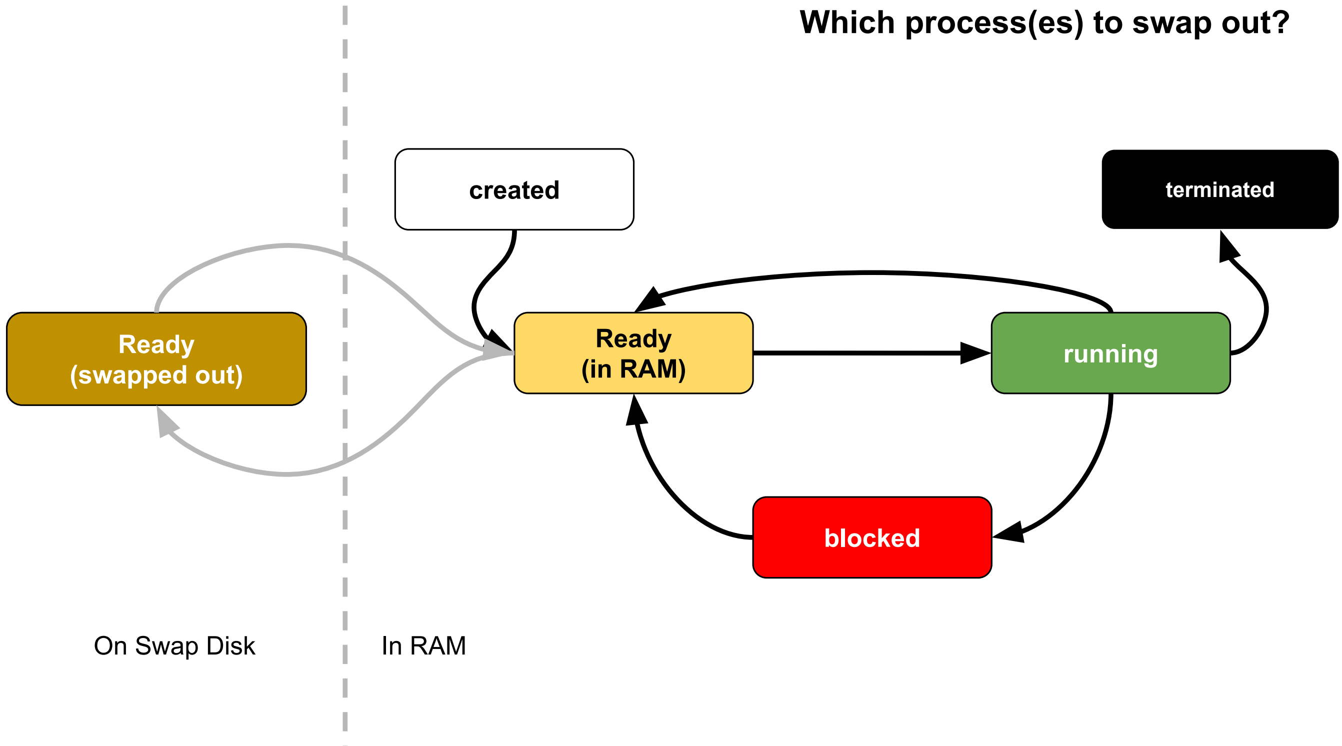
- A process must be resident in RAM to run
- When more memory is needed, the Medium Term Scheduler may begin to swap out processes
  - Processes in the ready queue are good candidates for swapping out
  - When a process is (being) swapped out
    - The **entire current process image** is dumped to the swap space
    - All memory areas owned by the process are released
- Swapping allows the system to host several processes whose total memory requirement exceeds the physical RAM size
- **Swap Space/Swap Disk**: a designated disk used for storing the *binary process image* of swapped out processes

Preemptive Scheduling ↔ CPU

Swapping ↔ RAM

Swapped Out Processes

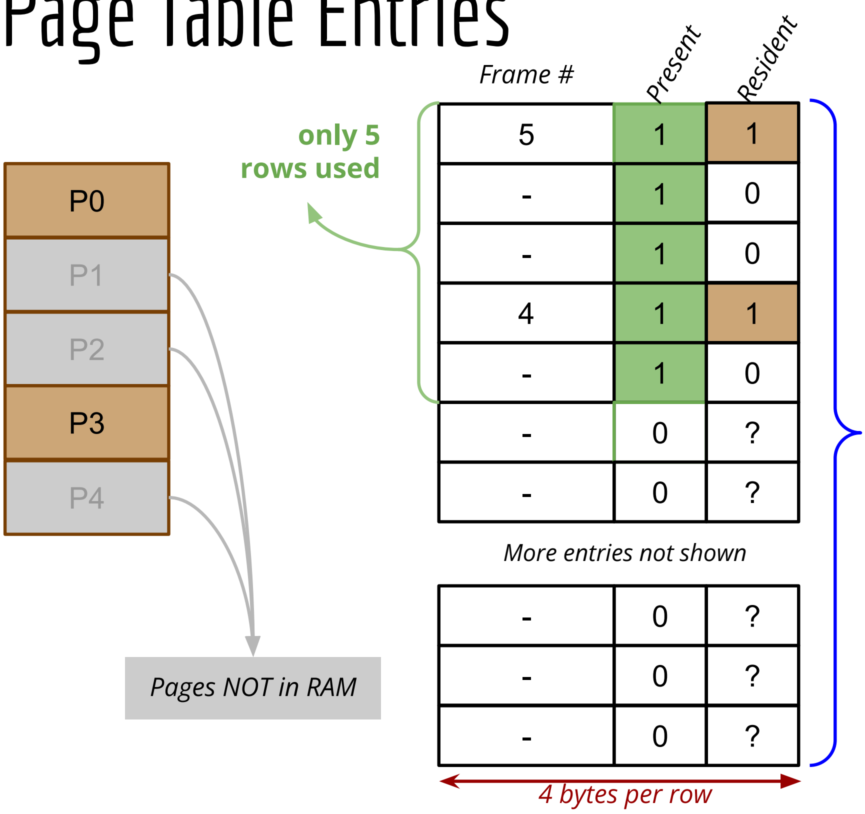
Which process(es) to swap out?



Swapping-Related Issues

- The OS maintains two ready queues
  - Processes which are ready and **resident in RAM**
  - Processes which are ready but **swapped out**
- When a process is swapped (back) in, it may resume execution at **a different physical address**
- A process to be swapped out should be completely IDLE
  - No pending I/O (because pending I/O requires target buffer to be resident in RAM)

Page Table Entries



Present bit: page is part of the process

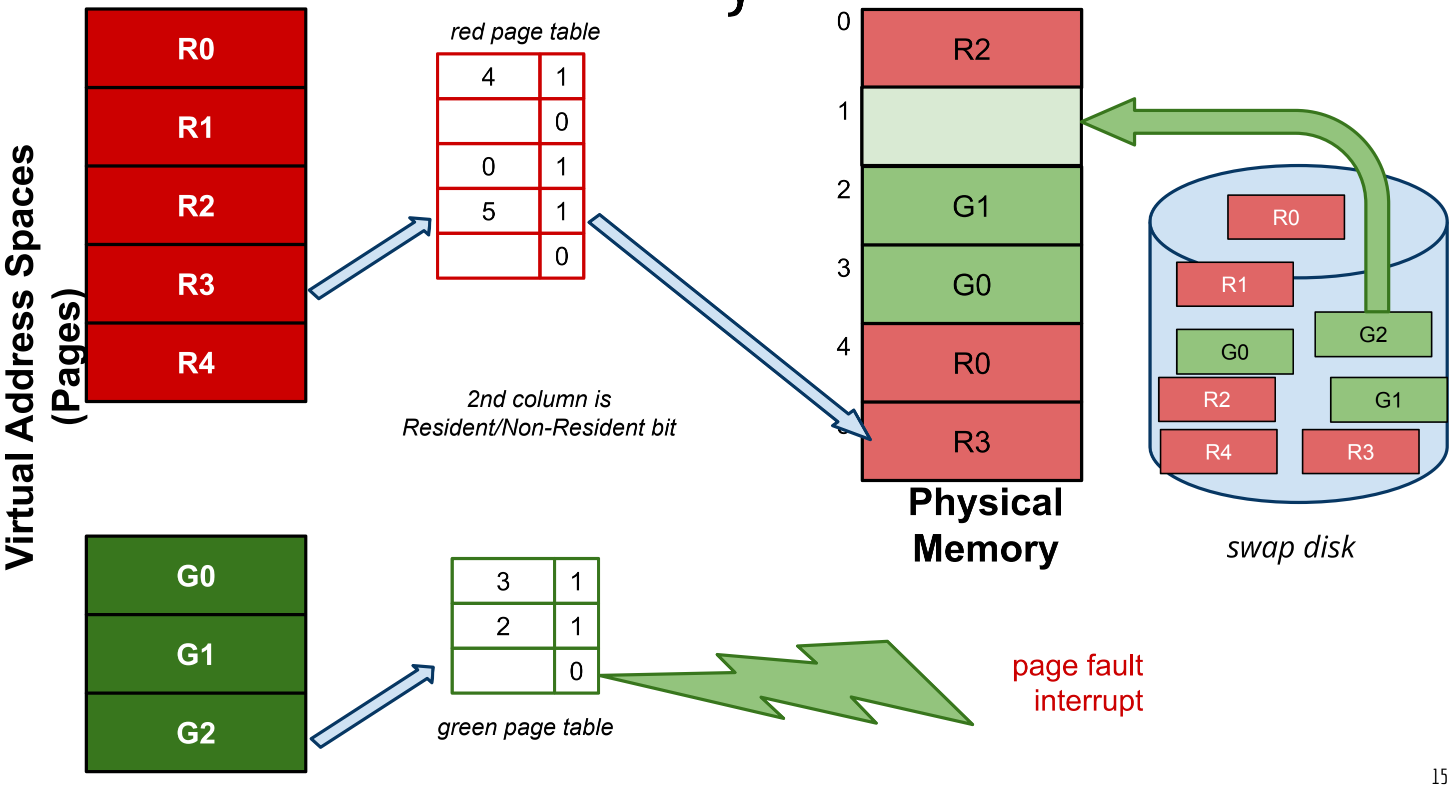
Resident bit: page is physically in RAM

1024 rows total

Assuming page size 4K (4096 bytes)
And 4 bytes per page table entry

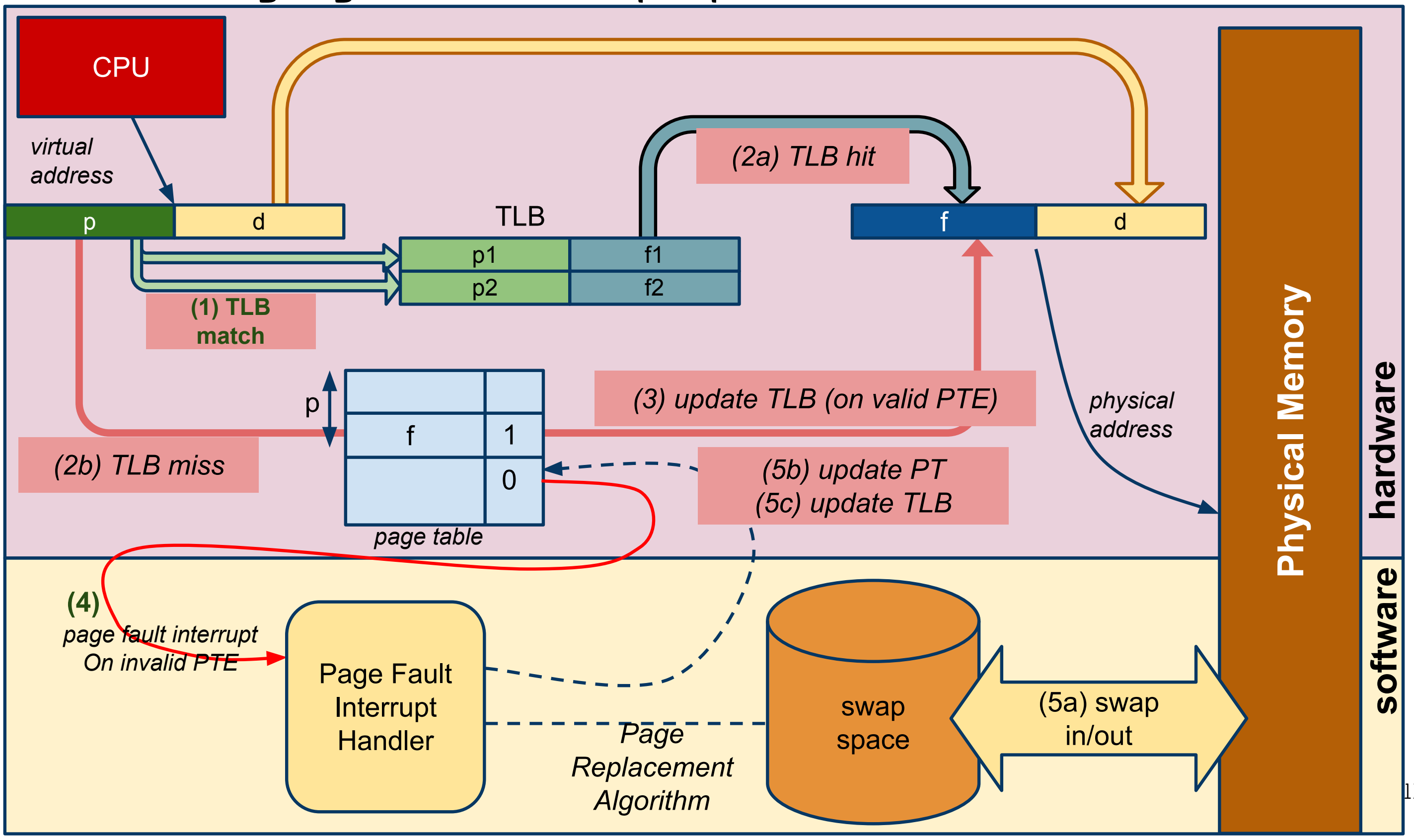
A physical page can hold 1024 entries / rows

MMU with Virtual Memory



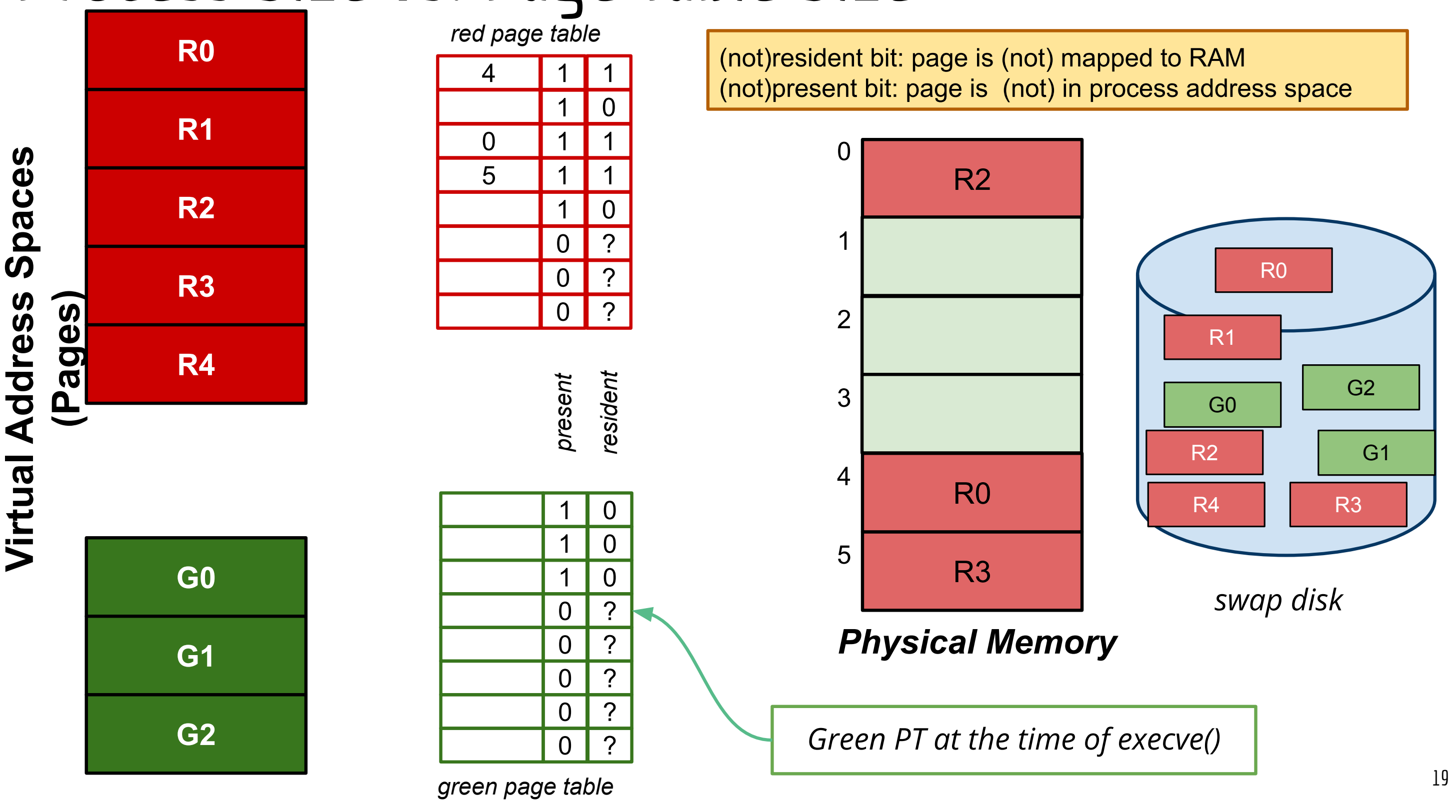
SegFault vs. Page Fault

Demand Paging + TLB + Swap Space



More Page Table Details

Process Size vs. Page Table Size



red page table

	present	resident
4	1	1
	1	0
0	1	1
5	1	1
	1	0
	0	?
	0	?
	0	?

green page table

	present	resident
	1	0
	1	0
	1	0
	0	?
	0	?
	0	?
	0	?
	0	?