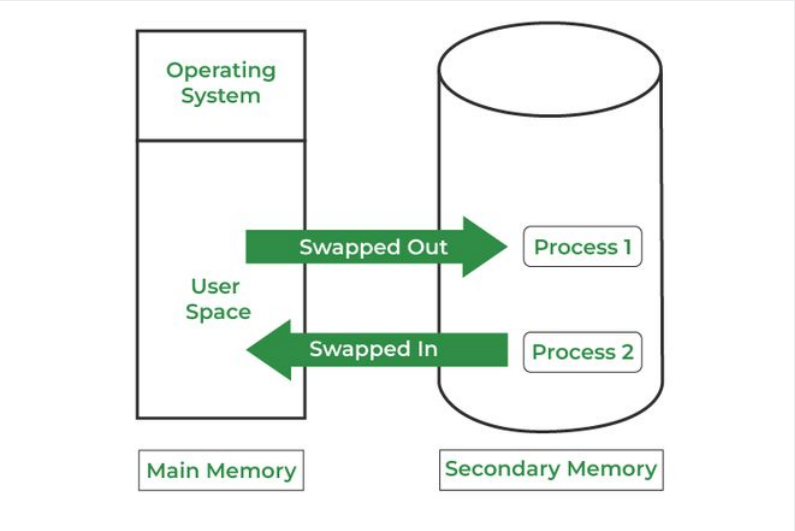
OSLab5

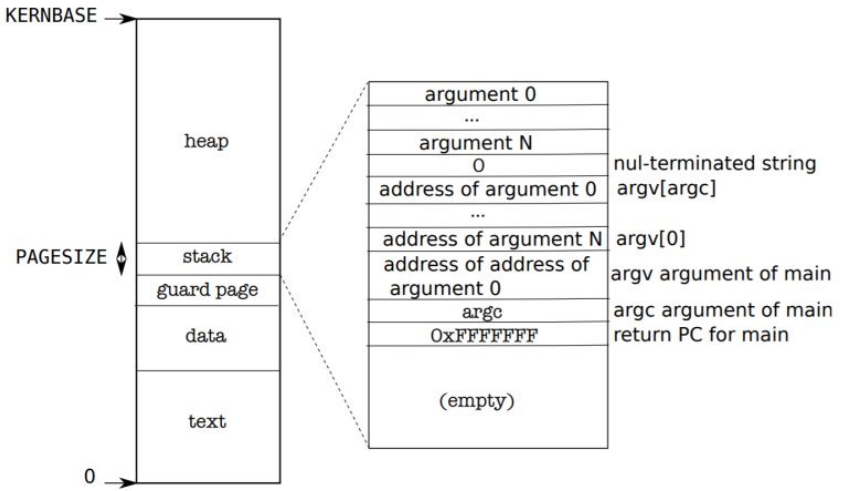
Memory Management

Mohammad Taha Fakharian, Pouriya Tajmehrabi



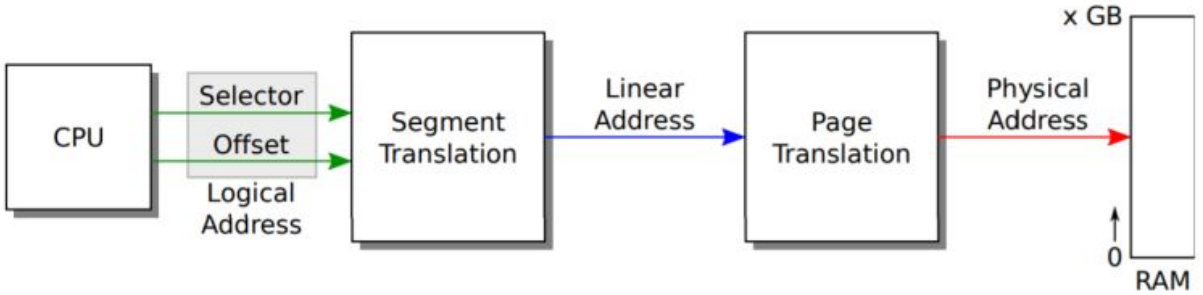
Memory Structure in XV6

- Code
- Heap
- Stack



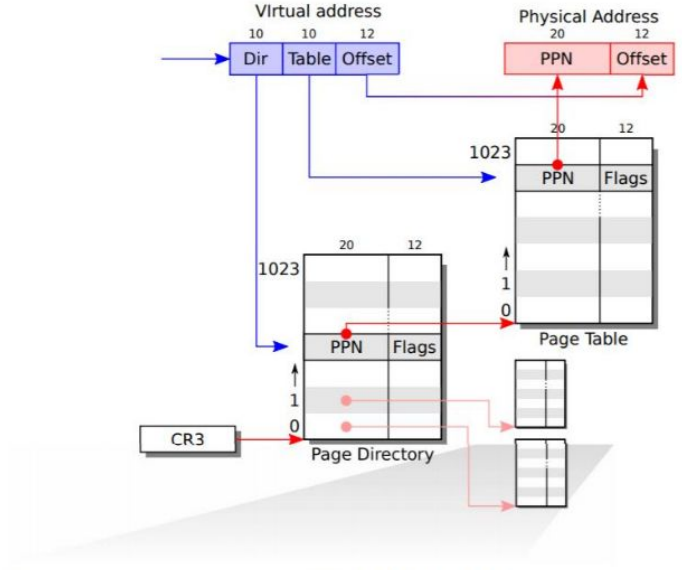
Memory Mapping

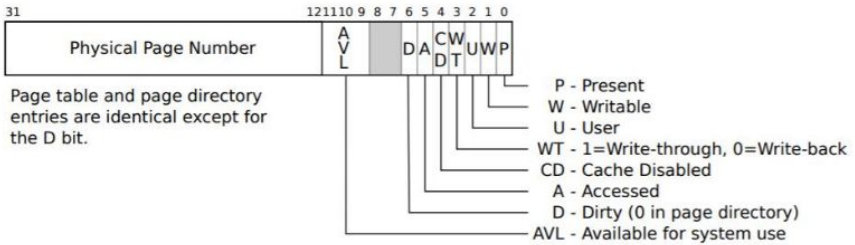
- Linear to virtual
- Virtual to physical
- No direct access



Paging

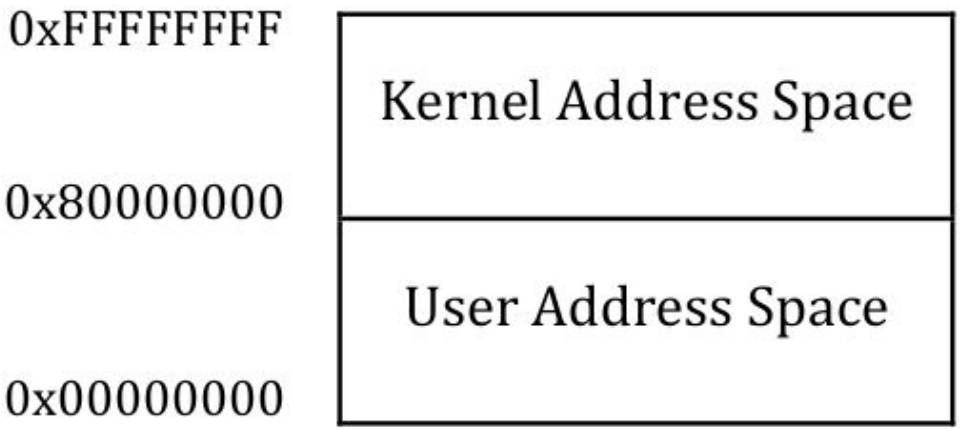
- Why?
  - Isolation
  - ABI simplicity
  - Memory swapping
- Page structure





Memory Layout in XV6

- Overall structure
- Useful functions to check
  - Kvmalloc
  - Setupkvm
  - mappages
  - exec



New Feature: Memory Sharing!

- Memory Sharing is when two processes access the same memory page
- Shared Pages are usually identified with an ID
- A system call usually enables one process to create a shared space and get a pointer to it
- Same system call can enable other processes to attach themselves to the shared page
- A method of process communication

Shared Memory Table

- Used to manage page physical counterpart addressing
- Tracks process reference counts
- Used to coordinate between processes
- Our system calls will rely on it to do their job

Implementing New System Calls

- open_sharedmem(int id, pointer):
  Returns a shared pointer through the second argument to the start of a shared page with id. ID is used to interact with shared memory page(allocates page if not shared before, attaches process if shared page already exists)
- close_sharedmem(int id):
  Searches for shared memory segment and closes it. If needed, reduces process reference count and clears shared memory table .

Important Notes

- Correctly use locks when interacting with shared memory to make sure you applications run reliably (eg. spinlocks)
- You need to design and implement the structure of shared memory table
- Don’t forget to use mappages when necessary
- Useful System calls you will need to use in your implementation: mappages, allocuvm,kvmalloc

Good Luck!

Feel free to contact us for any questions:

taha.fakharian@gmail.com

pouriyatajmehrabi1381@gmail.com