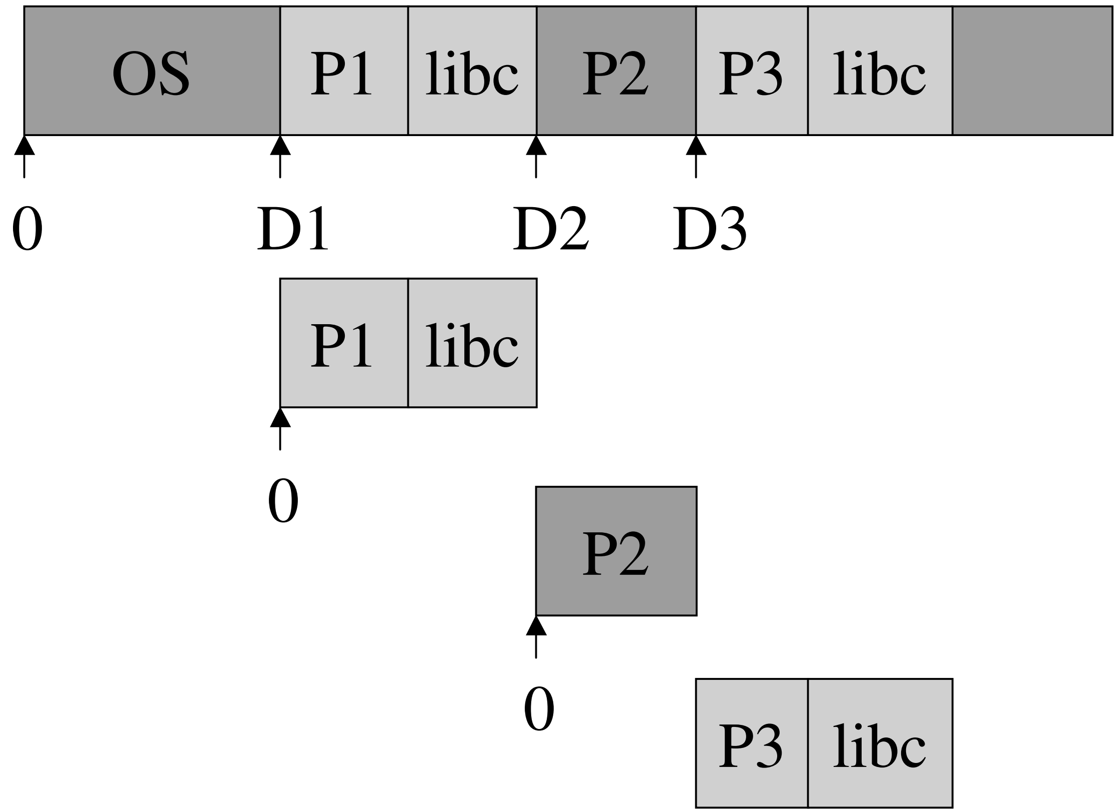
Towards Modern MMUs

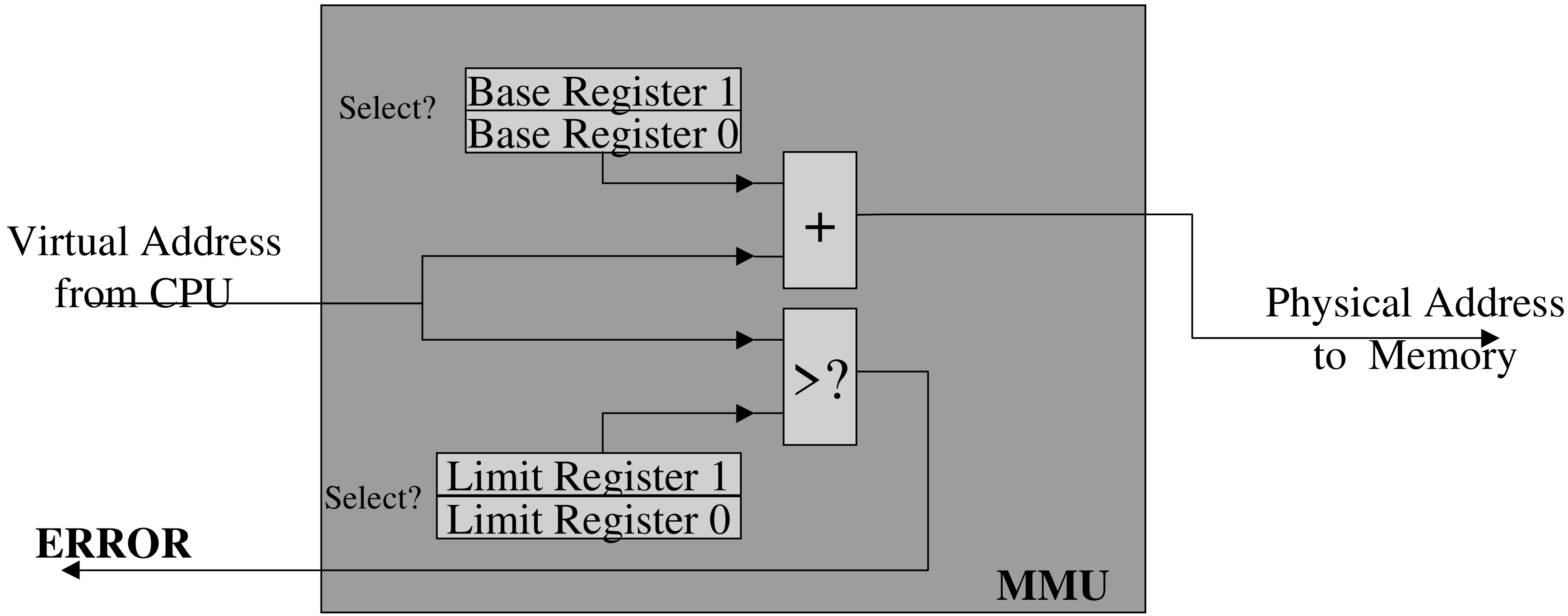
Sharing and Protection
in Modern
Operating Systems

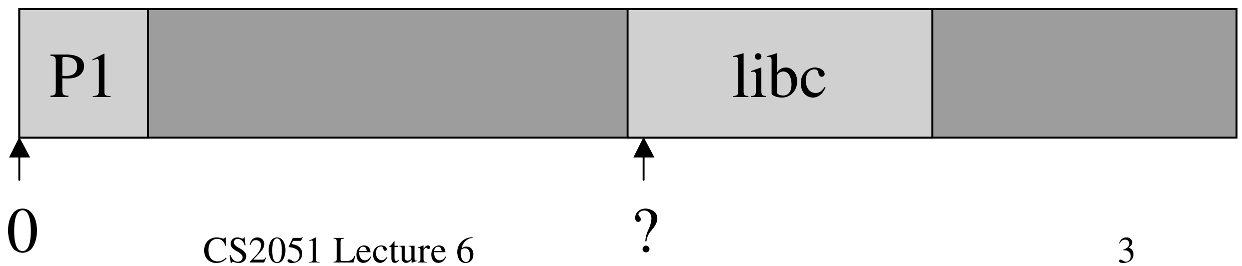
Towards Sharing?



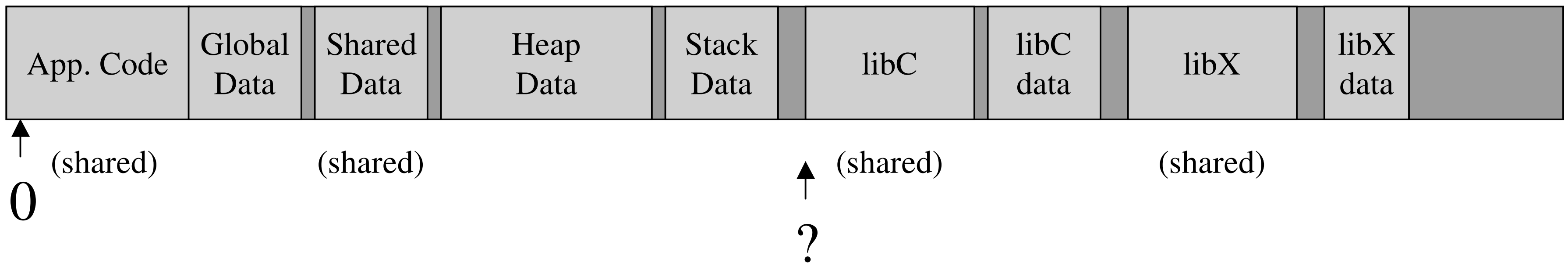
My linux box has ~50 processes running, each with libc (0.6Mb), libX (0.6Mb), libKDE, etc.

Sharing a Single Library





Programs have more, smaller Regions



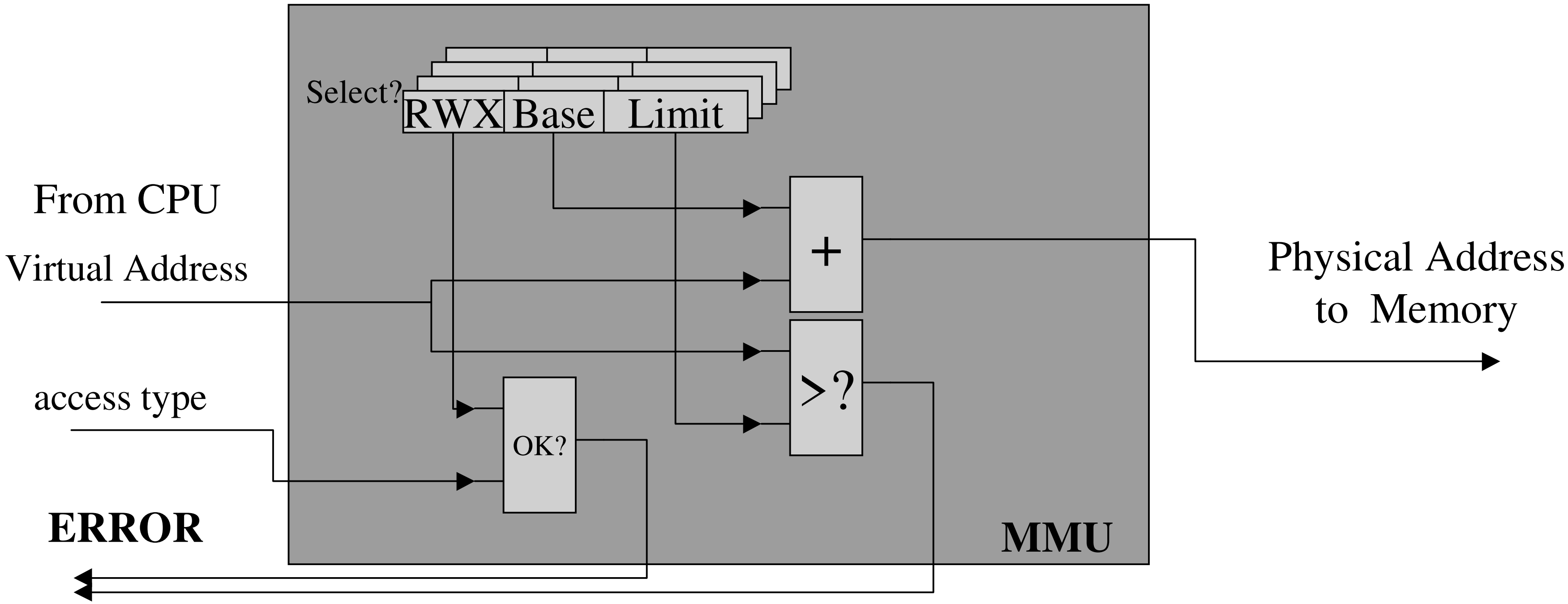
Each region has different sharing requirements

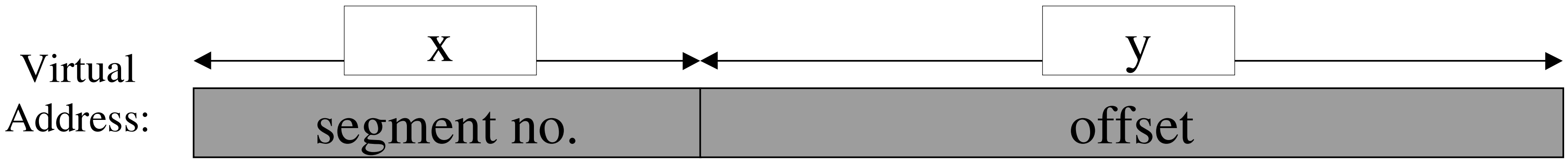
– MMU translates each region differently

Each region has different protection requirements

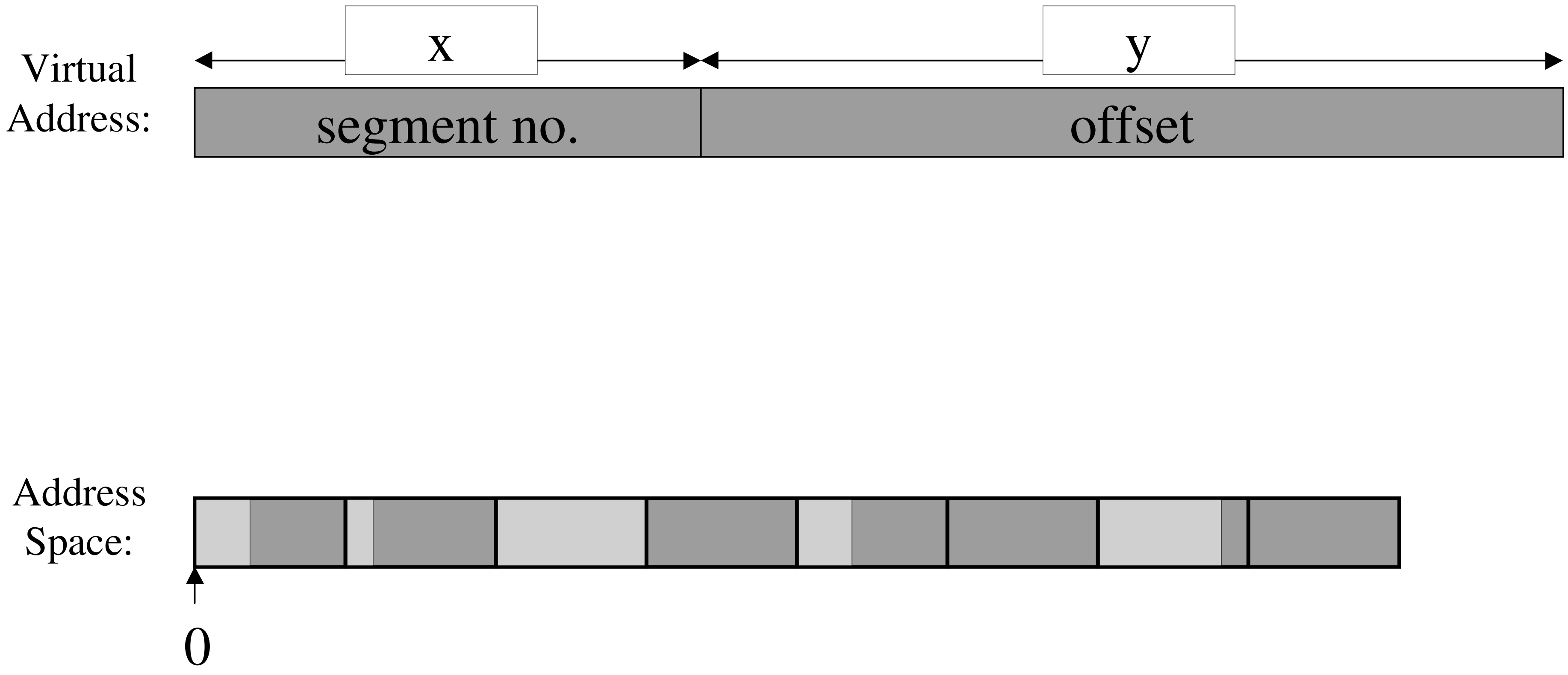
– provide separate R/W/X bits per region

Sharing and Protection





Segmented Address Space



Relieve Memory Pressure

- Not all memory needed at once:
  - application's error handling code
  - libraries and data
- Allow certain segments to be held on disk
  - how to load segment from disk to memory
- Only OS knows current main memory pressure
  - OS loads segment on demand, rejects segments when required

Virtual Memory Concept

- Segments are logically distinct program areas, of variable size
- Segments are natural unit of sharing
- Segments are natural unit of protection
- MMU implements
  - Presence bit, generates exception (when?)
  - Modified bit, set by MMU (when?)
- Disk, Main Memory shared transparently by OS