

# Advanced Operating Systems

COMP9242  
Introduction



## Staff

- Lecturer in Charge
  - Gernot Heiser
- Lecturer
  - Kevin Elphinstone
- Various Support Staff
  - TBA



COMP9242 03s2

2

## Why are you here?

- You've done comp3231
  - Did well (minimum credit)
  - You would like delve deeper into issues of modern OS construction
  - You'd like more challenging projects to really get your hands dirty
- Curious about where research is heading in the field of operating systems.
- Thinking about doing a thesis project in operating systems



COMP9242 03s2

3

## What can you expect?

- Challenging Project
- Lectures in general:
  - Background required for project
  - Exposure to local research projects
  - An in-depth look at OS issues
    - Building upon the background in COMP3231
  - Exposure to recent and seminal research papers
  - Guest lectures by active researchers (PhD students and local researchers)



COMP9242 03s2

4

## Project Goal

Provide students with a deeper understanding of Operating Systems through practical experience.

- Approach: Participate in the design and implementation of a simple operating system (SOS).



COMP9242 03s2

5

## Project Aims

- Provide experience in OS **design** and development, including:
  - Microkernel based systems (L4::Pistachio).
    - User-level OS servers.
    - User-level page fault handlers.
  - Device drivers
  - Performance evaluation
  - Implications of cache architectures
  - Exposure to alternative OS Designs
- Demonstrate the importance of design
- Provide experience of being a team member in a large software project.



COMP9242 03s2

6

## Project Aims

- Expose students to a mostly realistic OS development environment.
  - Similar to professional OS and or embedded systems developer.
- Give an understanding of what's involved in constructing an entire OS on bare hardware.
- Give an understanding of the interaction between low-level software and hardware.
- Encourage you to undertake a thesis, or do research within Distributed Systems Group.

## Prerequisites

- Students are expected to be very competent C programmers.
- Students are expected to be familiar with
  - basic computer architecture concepts.
  - Assembly language (read-only)
  - Basic RISC processor characteristics (we'll use a MIPS R4600 for the project)

## Lectures

(subject to change)

- Introduction and Overview
- Introduction to the L4 Microkernel
  - L4 system calls and usage (to get you started on the project)
- A close look at selected OS issues
  - Protection, capabilities
  - Caching, and its implications for OS
  - Page tables for wide address spaces
  - SMP issues: locking, cache coherence, scheduling
  - File systems

## Lectures

(subject to change)

- Microkernels and User-level Servers
  - History and motivation for microkernel systems, Hydra, Mach, discussion, experiences; second-generation microkernel systems, L4, Exokernel, Spin; design and implementation of microkernel-based systems, including user-level page fault handling and device drivers
- Microkernel Implementation
  - A detailed look at a real microkernel (L4Ka::Pistachio).
- Persistent systems and Single-address-space operating systems
  - Concepts and examples; UNSW Mungi project

## Project/Lab Work

- Build a simple operating system (SOS) from the ground up
- Major component of the course
- Use L4Ka::Pistachio
  - ported to MIPS here by Carl van Schaik
- Develop and test on U4600 computers
  - R4600 based machines design and built by Kevin Elphinstone and Dave Johnson
  - “Clean” machine to get your hands dirty
- Can also use CPU simulator Simular
  - Developed locally
  - Demos must be on real hardware

## Project

- Some warm-up experiments
- Students will work in groups of two
- End goal:
  - To produce a small efficient operating system
- Project will have a series of due milestones
  - Demo to pass the milestone and be awarded marks
  - Help you manage your time
  - Avoid major problems

## Milestones

- Details released RSN (week 2)
- Late milestones
  - Less than one week late, 25% of the milestone mark is lost
  - More than one week late, but still less than two weeks late, 50% of the milestone mark lost

## Alternative Projects

- Special arrangements might be made for particular student to do alternative projects
  - Must be at least as challenging as the original project
  - Must convince us that you can actually do it.

## Assessment

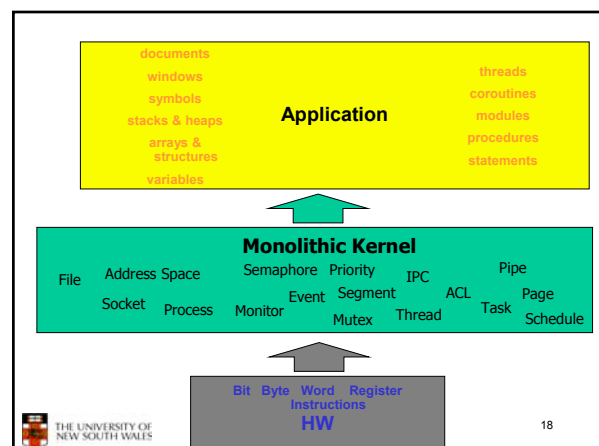
- 65% for project work
- 35% for final exam
  - A minimum of 14 (40%) required in final exam to pass
- Final Exam
  - 24hr take-home exam
  - Read and analyse two recent research papers and submit a critical report

## Textbook

- No particular textbook for course
  - See course web page for useful reference books
- Selected research papers referred to in the course

## L4 and Microkernels

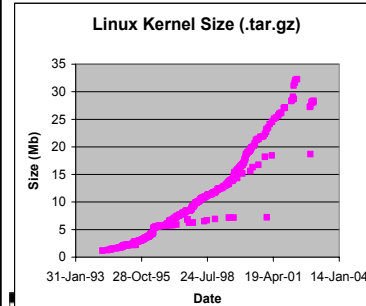
Background



## Monolithic Kernels - Advantages

- Kernel has access to everything, potentially:
  - All optimizations are possible
  - All techniques/mechanisms/concepts are implementable
- Can be extended by simply adding more code to the kernel

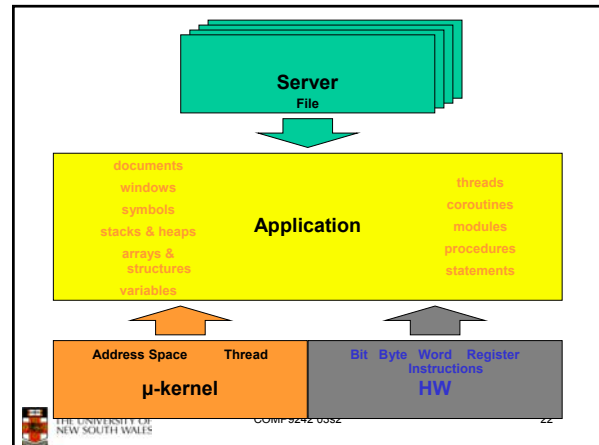
## Linux Kernel Evolution



For reference:  
Linux 2.4.18 = 2.7 million lines of code

## Approaches to tackling complexity

- Monolithic approaches
  - Layered Kernels
  - Modular Kernels
  - Object Oriented Kernels
- Alternatives
  - Extensible Kernels
  - Microkernels



## History

- monolithic kernels

## History

- monolithic kernels

- 1<sup>st</sup>- generation  $\mu$ -kernels
  - Mach *CMU, OSF* **External Pager**
  - Chorus *Inria, Chorus*
  - Amoeba *Vrije Universiteit*
  - (L3) *GMD* **User-Level Driver**

## Brief History

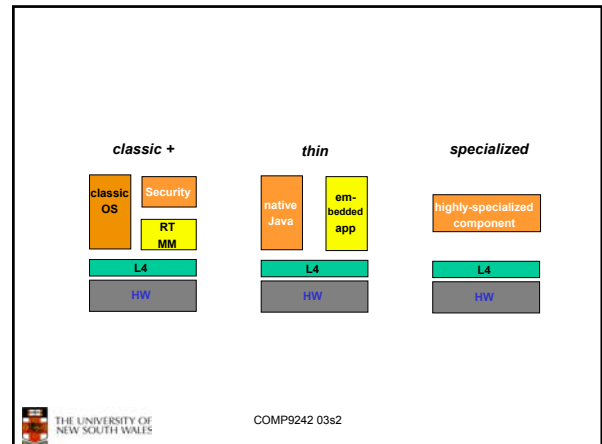
- monolithic kernels

- 1<sup>st</sup>- generation  $\mu$ -kernels

- Mach *CMU, OSF* **External Pager**
- Chorus *Inria, Chorus*
- Amoeba *Vrije Universiteit*
- (L3) *GMD* **User-Level Driver**

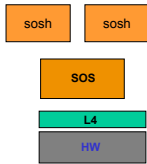
- 2<sup>nd</sup>- generation  $\mu$ -kernels

- (Spin) *U Washington*
- Exokernel *MIT*
- L4 *GMD / IBM / UKa* **User-Level Address Space**



COMP9242 03s2

## Project



COMP9242 03s2

## The Great Promise

- coexistence of different
  - APIs
  - file systems
  - OS personalities
- flexibility
- extensibility
- simplicity
- maintainability
- security
- safety



COMP9242 03s2

28

## The Great Promise *The Big Disaster*

- coexistence of different
    - APIs
    - file systems
    - OS personalities
  - flexibility
  - extensibility
  - simplicity
  - maintainability
  - security
  - safety
- SLOW**  
**• UNFLEXIBLE**  
**• LARGE**

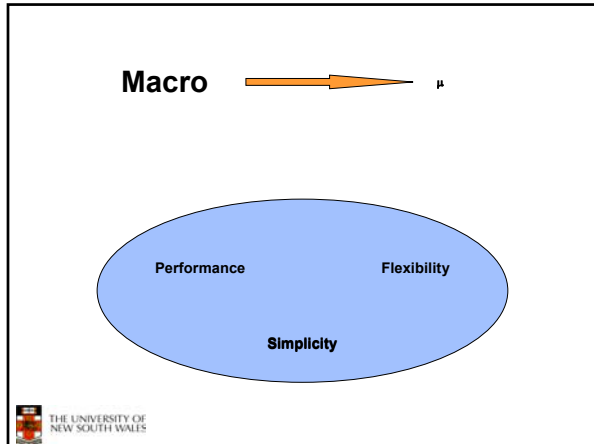


COMP9242 03s2

29

Macro   $\mu$





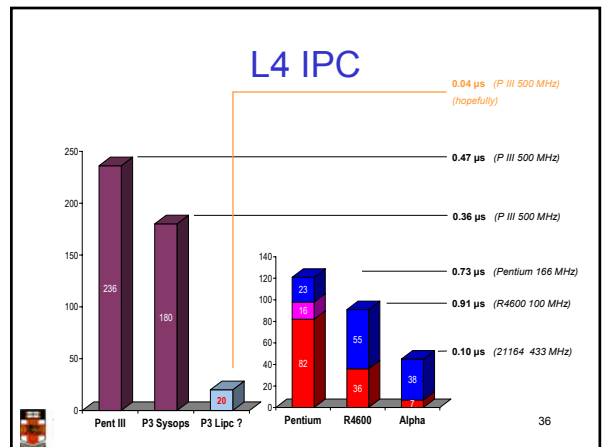
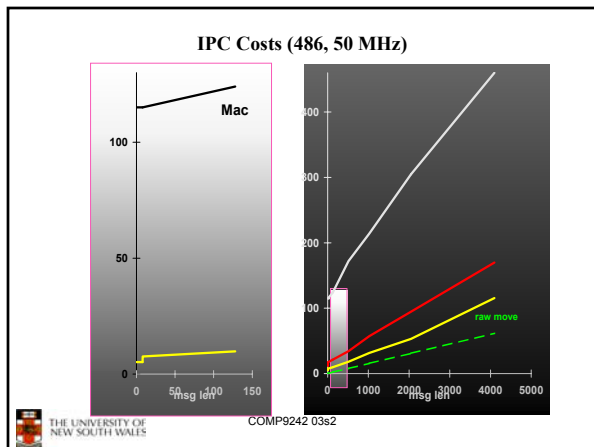
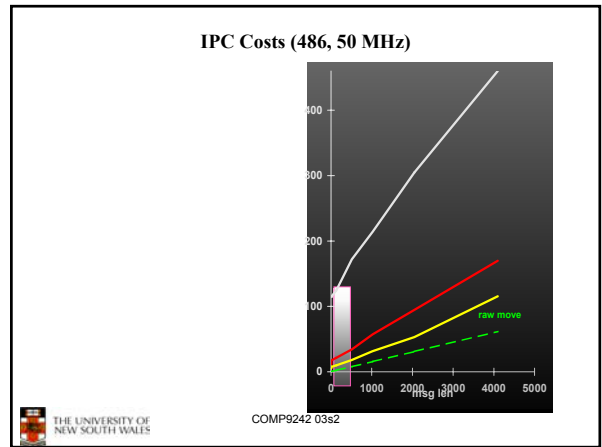
# The 100- $\mu$ s Disaster

THE UNIVERSITY OF NEW SOUTH WALES

# The 100- $\mu$ s Disaster

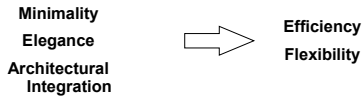
25 MHz 386  $\longleftrightarrow$  50 MHz 486  $\longleftrightarrow$  90 MHz Pentium  $\longleftrightarrow$  133 MHz Alpha

THE UNIVERSITY OF NEW SOUTH WALES



### Thesis:

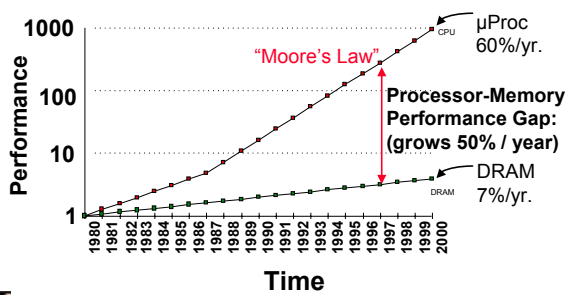
- A  $\mu$ -Kernel does the Job
  - if Properly Designed
  - if Carefully Implemented



### When analyzing IPC performance,

Cycles are not the only the to consider!!

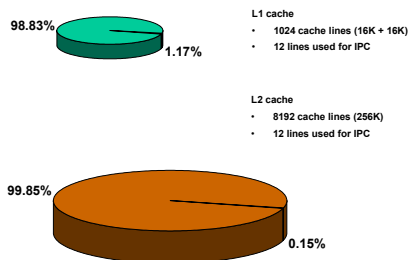
### Processor-DRAM Gap (latency)



### Today's Situation: Microprocessor

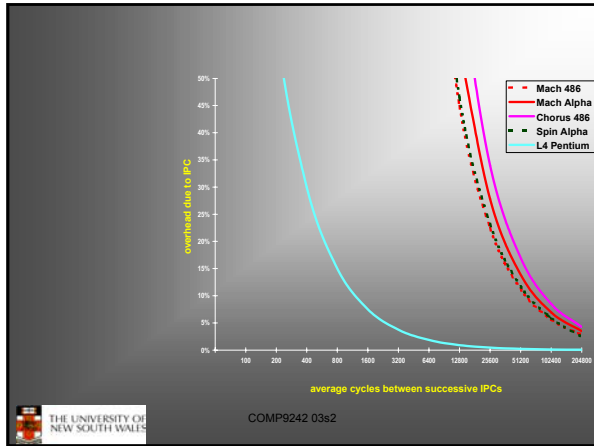
- Microprocessor-DRAM performance gap
  - time of a full cache miss in instructions executed
  - 1st Alpha (7000): 340 ns/5.0 ns = 68 clks x 2 or 136
  - 2nd Alpha (8400): 266 ns/3.3 ns = 80 clks x 4 or 320
  - 3rd Alpha (t.b.d.): 180 ns/1.7 ns = 108 clks x 6 or 648
  - 1/2X latency x 3X clock rate x 3X Instr/clock  $\Rightarrow \approx 5X$

### Cache Working Sets



### Other Complications

- P4 trace cache
  - A cache of recently translated  $\mu$ -ops
  - Flushed on every page-table switch
- Virtual Caches
  - Need to be flushed on address space switch



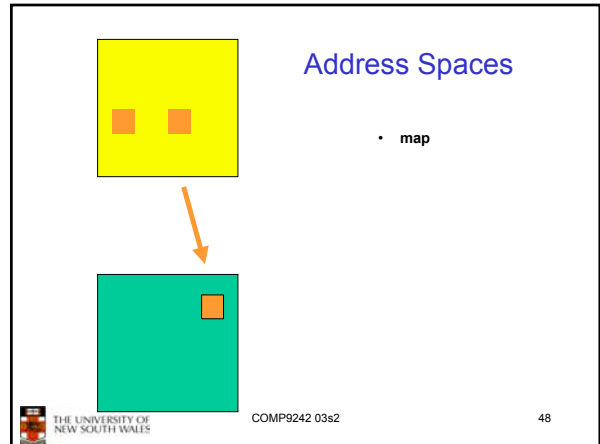
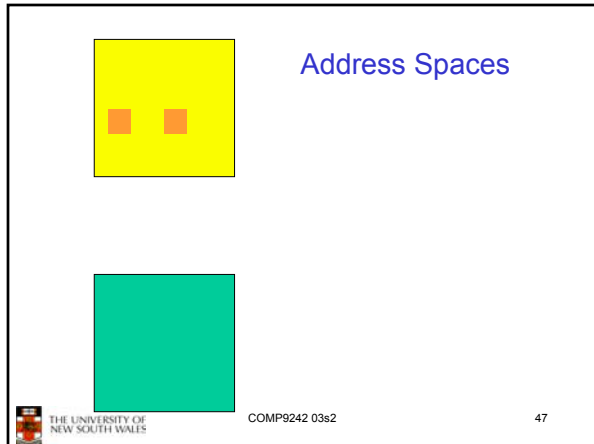
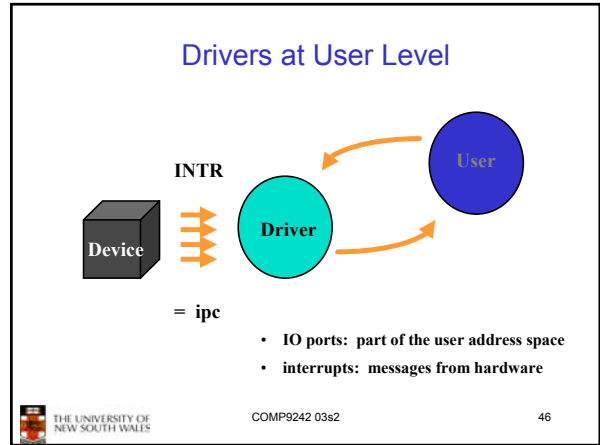
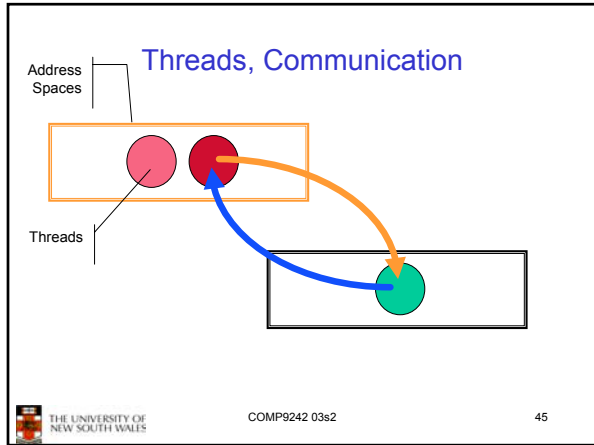
A  $\mu$ -kernel does no real work.  
 $\mu$ -Kernel services are only required to overcome  $\mu$ -kernel constraints.

Therefore,  $\mu$ -kernels have to be infinitely fast!

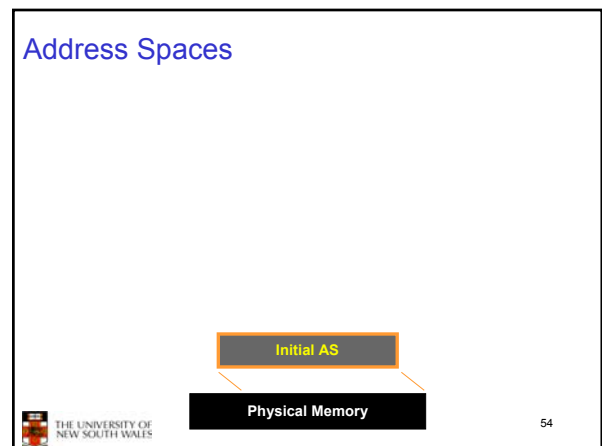
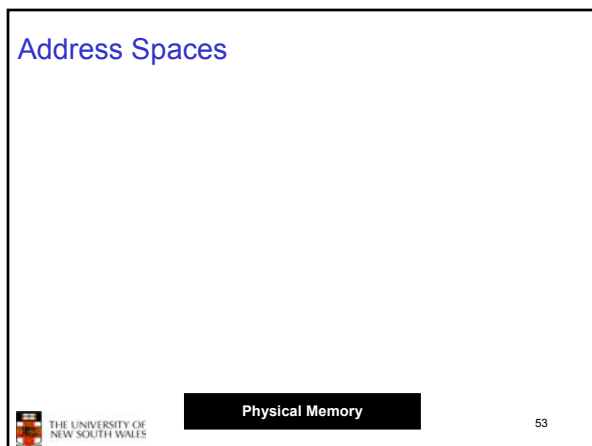
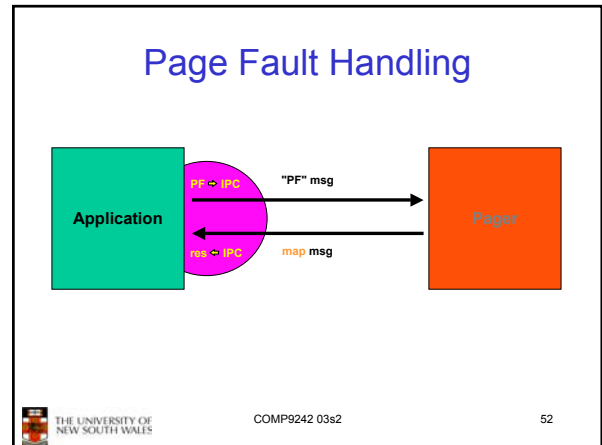
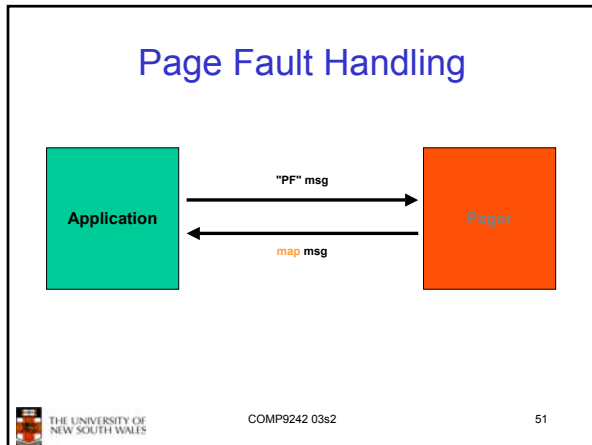
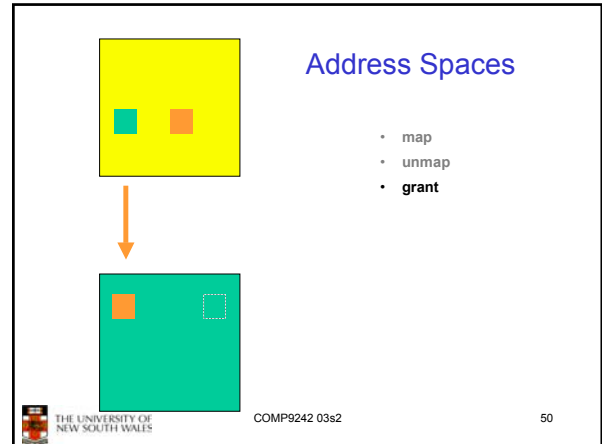
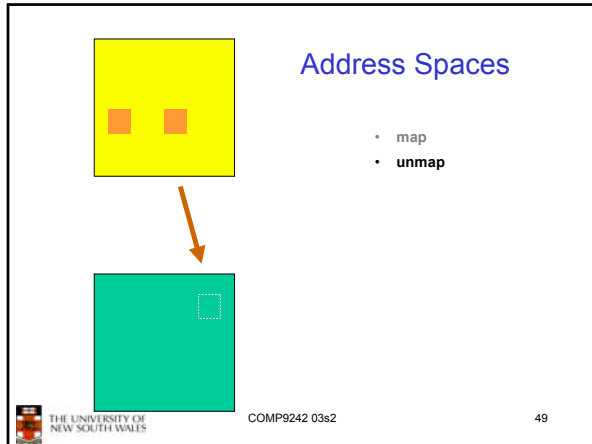
Minimality is the key!

- Threads
- Address Spaces

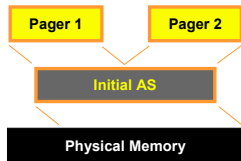
IPC Mapping



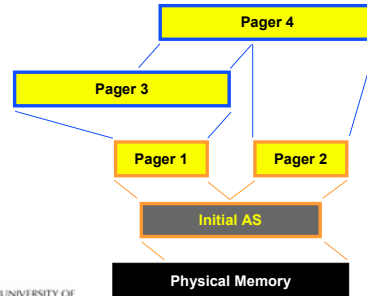




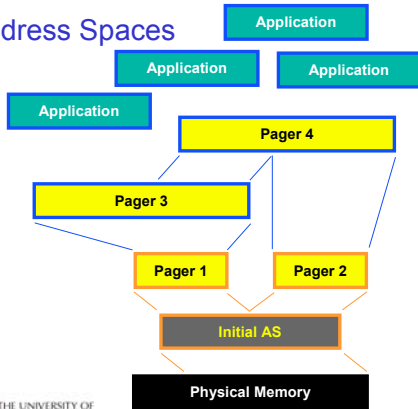
## Address Spaces



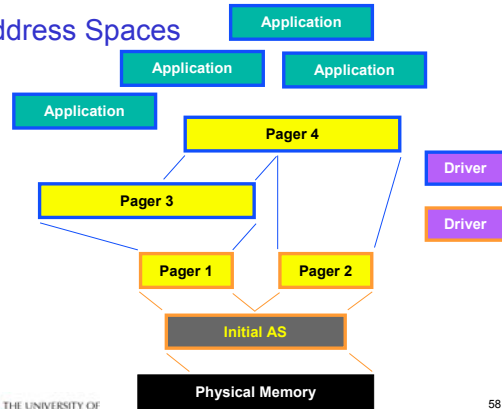
## Address Spaces



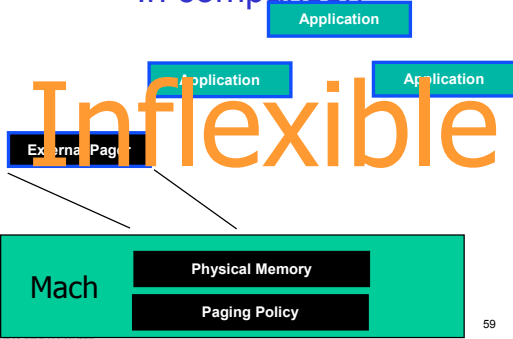
## Address Spaces



## Address Spaces



## Mach Virtual Memory In comparison



## Size Comparison

