Operating Systems Notes

0.1 Operating System Structure

1 Memory Managment

Programs must be brought from disk into mmemory and then placed into a process.

The CPU can only access data from memory, not disk.

Register access takes at most 1 CPU clock, but Main Memory can take mayn cycles, causing a *Stall*.

1.1 Base and Limit Registers

A set of *base* and *limit registers* define the logical adress space. the CPU must chech every memory access is valid between the base and the limit for that user. Failure causes a trap to the OS monitor

1.2 Virtual Address Space

Logical/Virtual addresses are independent of physical memory.

Hardware translates virtual addresses into physical ones.

Logical/Virtual addresses a process can reference is called the address space.

1.3 Memory Managment Unit (MMU)

Effectively is a hash function from logical address to physical address.

a MMU prevents the need for swapped out process to be swapped back into the same physical addresses.

Swapping is not typically supported on mobile devices, more likely to to overwite least used data.

1.4 Partitioning

Main memory is usually broken up into two partitions; The OS and user process.

Each process is contained within a single contiguous section on memory.

Realocation registers are used to protect users processes from one another and from canging the OS code.

Some old techniques include:

- Fixed Partitions - simple but causes fragmentation often

- Variable Partitions - no internal fragmentation, but can leave holes in the physical memory

Dynamic Storage-Allocation is possible using First-fit, Best-fir and Worst-fit in terms of hole filling.

2 Paging

2.1 Address Translation Scheme

The Address generated by the CPU is divided into:

- Page Number - used as an index into the page table, containing the base address if each page in physical memory
- Page Offset - combined with the base address to define the physical memory address

2.2 Page Tables

Page Tables are kept in main memory.

The *Page-table base register (PTBR)* points to the page table

The *Page-table length register (PTLR)* indicates the size oof the page table

In this scheme each instruction requires two memory accesses, one for the page table and one of the instruction, this can be solved by using *translation look-aside buffers (TLBs)*.

Some TLBs store address-space identifiers(*ASIDs*) in each TLB -

- uniquely identifies each process
- provides address-space protection

On a TLB miss, value os loaded into the TLB for faster access next time.

2.3 Memory Protection

A Valid/Invalid bit can be attatched to each entry in a page table.

Valid indicates that the associated page is in the logical address space, and thus is a legal page.

We can use this or a *page-table length register (PTLR)* which can contain more info (Thnaks for the detail here Michael).

Any violations result in a trap to the kernal.

2.4 Hierarchical Page Tables

Break up the logical address space into multiple page tables. A simple example is a teo-leveled page table, known as *forward-mapped page table*.