Khaula Molapo

Computer Architecture Week 10

1. Thrashing

Thrashing happens when a computer's memory is overloaded with too many active processes, causing the system to spend most of its time swapping data between RAM and the hard drive instead of doing real work. This happens when the working set of one or more processes — meaning the set of memory pages they currently need — is larger than the available physical memory. As a result, pages are constantly being replaced, leading to very high page fault rates and low CPU usage.

The main causes of thrashing include running too many programs at once, not having enough RAM, or poor memory management by the operating system. The system becomes extremely slow, as every small task involves loading data from disk. The operating system can detect thrashing by monitoring CPU activity and page faults — if CPU use drops while disk activity rises sharply, it's a sign of thrashing. To fix it, the OS can reduce the number of active processes, increase the working set size, or add more physical memory. Avoiding thrashing ensures smoother performance and faster program execution.

2. Paging vs. Segmentation

Paging and segmentation are two ways the operating system manages memory. Paging divides memory into fixed-size blocks called pages, while segmentation divides it into variable-sized segments based on logical parts of a program, like code, data, or stack.

Paging helps avoid external fragmentation and is simple to handle, but it can cause internal fragmentation since a page might not be fully used. Segmentation is easier for programmers to understand and can save space because segments fit data more naturally. However, segmentation suffers from external fragmentation, making memory harder to allocate over time.

Paging became the dominant system because it works better with modern hardware and supports virtual memory more efficiently. It allows fast memory access using a Translation Lookaside Buffer (TLB) and simplifies memory management, making it the main choice in today's operating systems.

3. EMAT Calculation

Given:
- TLB hit ratio = 99%
- TLB search time = 10 ns
- Memory access time = 100 ns
- Page fault rate = 0.1% (0.001)
- Page fault handling time = 20 ms = 20,000,000 ns

Step 1: EMAT without page fault
EMAT = (Hit ratio × (TLB + Memory)) + (Miss ratio × (TLB + 2 × Memory))
= (0.99 × 110) + (0.01 × 210) = 108.9 + 2.1 = 111 ns

Step 2: Include page faults
EMAT = (1 − 0.001) × 111 + (0.001 × 20,000,000)
= 0.999 × 111 + 20,000 ≈ 20,110 ns

Even though the page fault rate is very low, the large cost of handling one (20 ms) makes a huge difference. This shows why reducing page faults is critical for fast memory performance.