CS4617 Computer Architecture

Lecture 6: Virtual Memory

Dr J Vaughan

September 24, 2014

Memory management

- Memory is a resource that is essential for the execution of instructions
- Execution model states that instructions are fetched from memory in the fetch phase instruction cycle
- Some instruction operands are also fetched from memory

Single contiguous allocation

- One process in memory
- Code, data, stack
- Some wasted memory because process does not fit exactly in available memory
- If process code & data too large for memory, use overlays and swapping

Multiprogramming

- One process spends time in blocked state
- Processor time wasted until process returns to ready state
- Solution: increase number of processes in ready state to raise probability of finding a ready process when current process enters blocked state
- Memory must be shared between a number of processes

Fixed partitioning

- Divide memory into a fixed number of regions called *partitions*
- Degree of multiprogramming = number of partitions
- Some memory wasted in each partition
- Probability that a process will not fit completely in a partition is increased
- Protection becomes an issue
- Base and Limit registers

Variable partitioning

- Degree of multiprogramming is variable
- Holes increase as processes are created and terminate
- Memory becomes fragmented
- Solution to fragmentation is hole coalescing and compaction
- Compaction requires dynamic relocation
- Allocation is still contiguous

Paging

- Plug-and-play approach to solving the fitting problem
- Memory divided into fixed-length page frames
- Process code and data divided into pages of same length as a page frame
- Pages plug into page frames
- Memory address developed by a running process is divided into two fields, page number and word number
- $Process\ address = Page\ Number | Word\ Number$

Example: Paging

- 32-bit address
- Bits 31..12 = 20-bit Page number, p
- Bits 11..0 = 12-bit Word number, w
- Word number is an offset or displacement within a page
- In this example, pages are 4KB long and there are 1M pages
- Common page lengths are 1K, 2K, 4K
- Process must have all its pages in memory in order to execute
- Degree of multiprogramming is limited by number of available page frames

Paging (continued)

- Allocation is non-contiguous
- Page 0 can reside in Frame 7, Page 1 in Frame 4, Page 2 in Frame 6
- Address translation mechanism must be provided to convert Page Number to Frame Number
- This is the Page Table (PT)
- Processes are translated to run in memory beginning at location 0
- Page Table provides dynamic relocation
- Static relocation still needed to deal with static linking

Demand paging

- Paging alone cannot cope with processes larger than available number of page frames
- Principle of Locality applies
- On any one execution of a program, process will not need all its pages
- In any time interval of execution, process will only reference a subset of its pages within a relatively narrow address range
- The subset of referenced pages changes intermittently
- Therefore, process does not need to load all its pages in order to make progress with execution

Virtual Memory

- All pages of a process exist on secondary storage (disk)
- Pages that are needed for execution are copied into main memory
- Therefore, process address range is not limited by physical main memory
- Executing process generates a *Virtual address*
- Translation mechanism produces a *Physical address*
- Tracks whether page is in primary or secondary storage
- Page is loaded into main memory on demand

Working Set

- Set of pages needed by a process in a time interval = Working Set
- Working set changes in address values and size from time to time
- Process can progress its execution if its working set is in memory
- If working set is not in memory, due to degree of multiprogramming being too large, thrashing can occur

Controlling the degree of multiprogramming

- Degree of multiprogramming needs to be controlled: *admission scheduling*
- Working set concept is good, but difficult to implement in practice
- When process requests a page that is not in main memory, an interrupt called a *page fault* occurs
- Page fault rate is low when processes are making progress
- Page fault rate increases rapidly as thrashing is imminent
- Control degree of multiprogramming based on page fault rate

Fields in a page table entry (PTE)

- Page number p
- Frame number f
- Reference bit
- Dirty bit
- Secondary storage address

Replacement

- When a page is loaded, it is placed in a free page frame and the page table is updated
- If no page frame is free, a resident page must be replaced
- The best page to replace is that one which will not be referenced for the longest time in the future

Replacement in practice

- Locality permits the inference that recent past history is a good indicator of near future performance
- So the best page to replace is the one that is Least Recently Used (LRU)
- Frequency of reference in the current time interval is easier to track, so Least Frequently Used (LFU) is a good approximation to LRU
- The Reference Bit in the PTE is used in implementing a variety of page replacement algorithms that approximate LFU
- If a page has been written to since being loaded, the Dirty Bit in its PTE is set and it must be copied to secondary storage before being replaced

The Page Table

- Returning to the example where $|p| = 20$ bits and $|w| = 12$ bits
- p is an index into the PT
- Size of PT = 1M entries
- Each PT entry comprises frame number, judgement bits and secondary storage address
- Assume $|PTE| = 32$ bits
- PT must be paged: it occupies $2^{20} \times 2^2 / 2^{12} = 2^{10} = 1024 = 1K$ pages

Paging

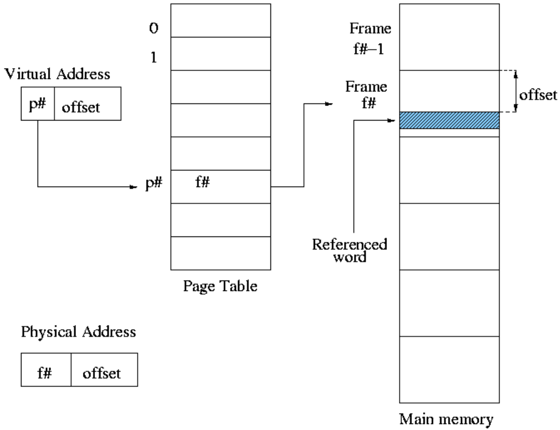
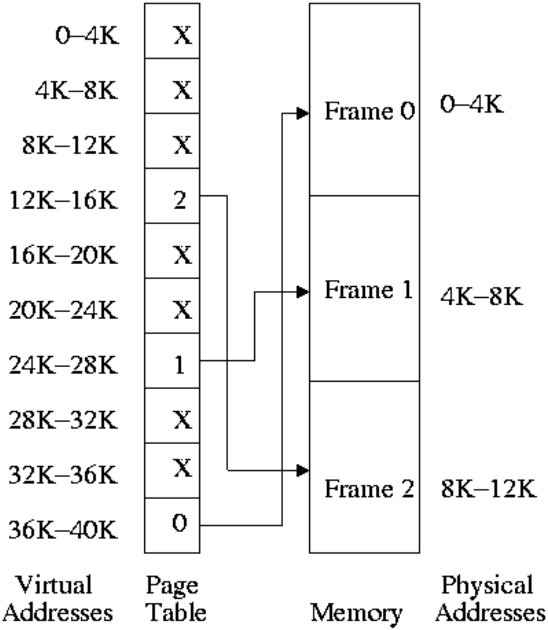


Figure: Paging

Paging


0–4K
4K–8K
8K–12K
12K–16K
16K–20K
20K–24K
24K–28K
28K–32K
32K–36K
36K–40K
X
X
X
2
X
X
1
X
X
0
Frame 0
Frame 1
Frame 2
0–4K
4K–8K
8K–12K
Virtual Addresses
Page Table
Memory
Physical Addresses

Paging

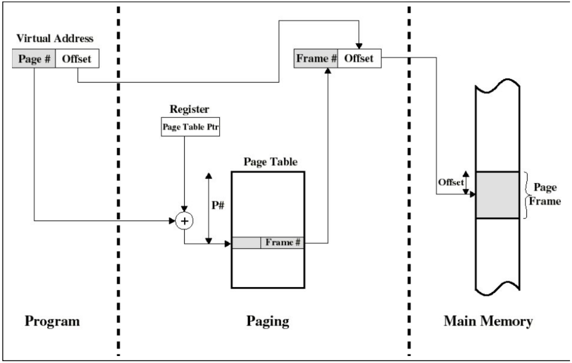


Figure: Paging

Paging

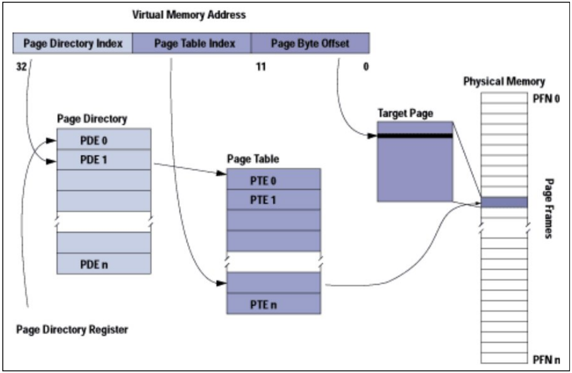


Figure: Paging

Paging

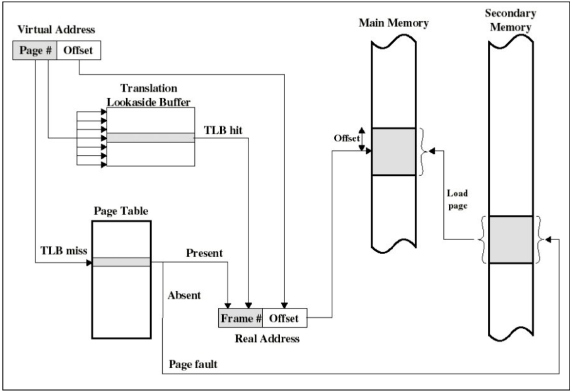


Figure: Paging