COSC 3750

Memory Management, Paging

Kim Buckner

University of Wyoming

Mar. 10, 2022

Previously

- memory

Paging

- Physical memory is broken down into blocks called frames
- Logical memory into pages of the same size.
- Backing store is divided into same size blocks.
- Every address $= page + offset$.

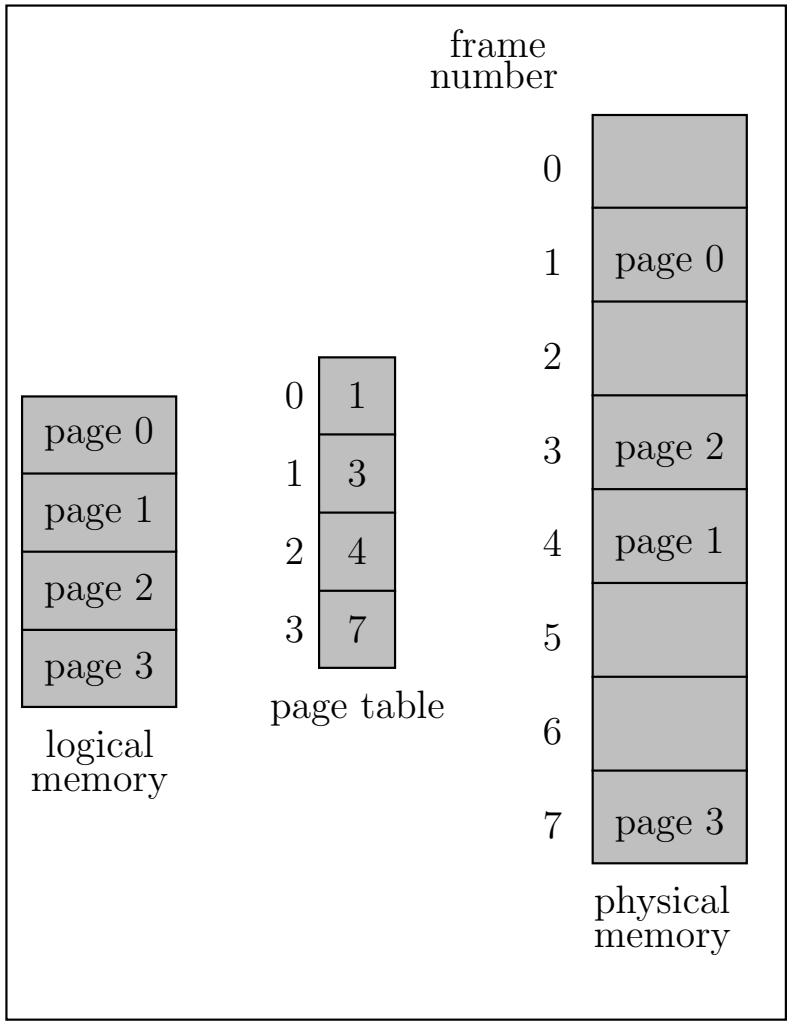
(more . . .)

- Page number is the index into page table, maps number to real address.
- Page size determined by hardware, power of 2, like 4096.
- Hardware support is provided to map logical addresses to physical.

(more . . .)

- Every address consists of a page and an offset. The page is mapped to a frame and offset is added to it.
- There are normally a power of 2 number of entries in the page table so that if there are 2^m bytes in memory and a page is 2^n then the high-order $m - n$ bits of a logical address are the page and the low-order bits are the offset.

Page mapping



Paging continued

- There is no external fragmentation in a paging scheme but there is internal fragmentation up to page size - 1
- Smaller page sizes would reduce internal fragmentation but increase the size of the page table putting a heavier drain on system resources.
- Also disk I/O is more efficient for larger pages.

(more . . .)

- Typical page sizes are 4k to 8k, some systems support multiple page sizes depending what is stored on them.
- Processes are examined for size on entry and the size determines how many pages must be made available.
- The dispatcher sets up the page table for the process which means that context switch time is higher for paging.

(more . . .)

- O/S maintains a "frame table" which tells which are free and which are allocated to what process.
- All addresses must be translated for system calls.

Hardware support

- Can use a special set of registers – ex. PDP 11
- Most use a "page table base register" (PTBR) and the table is kept in memory but this can be very slow.
- Solution is a small fast hardware cache called the "translation look-aside buffer" (TLB).

(more . . .)

- This is an associative memory and is small and expensive, entries usually between 64 and 1024, all keys are compared simultaneously.
- If it's in the TLB, only about 10% overhead.
- If a miss, we do a memory access for the entry and add to the TLB.

(more . . .)

- If TLB is full then replace an entry, usually ”least recently used (LRU), but can range to anything, including random.
- The TLB must be flushed on a context switch if it does not support “address space identifiers” (ASID).

(more . . .)

- Page table entries contain more than address info, such as a bit that marks the page as read-only.
- Also can add a bit that marks a page as valid or invalid.

Structure

- Normally hierarchical. Many options.
- Can be up to 4MB for the page table.
- Maybe use a two-level scheme.
  - Outer table indexes an page table of pages which address memory.
- Hash a page table.
- Clustered page table.

(more . . .)

- Inverted page table - only for the whole system and it contains pid/page entries.
  - And can then hash it.
- And can have shared pages.

Segmentation – per process

- Code segment.
- Data segment.
- Main segment, etc.
- And can add paging.
- Many systems are working toward a mix of paging and segmentation.