CSCE 613
Operating Systems

ABSTRACT

Design Document – For 2 Page Size Support

Tharun Battula
82400197

The current Design Analysis (4KB frame size)

32 bit addressing

10 bits	10 bits	12 bits
Page Directory Index	Page Table Index	Address Offset

This currently supports 4KB frame sizes with 2 level paging
If we convert this 2 level paging single level paging it will be mapped to 1 level paging

- i.e. if we combine last 10 bits plus 12 bits,
- it would result in 22 bits offset effectively – 4MB frame size.

10 bits	22 bits
Page Directory Index	Address Offset

Question: A design supporting 4KB and 16MB page sizes in same paging architecture.

Solution:

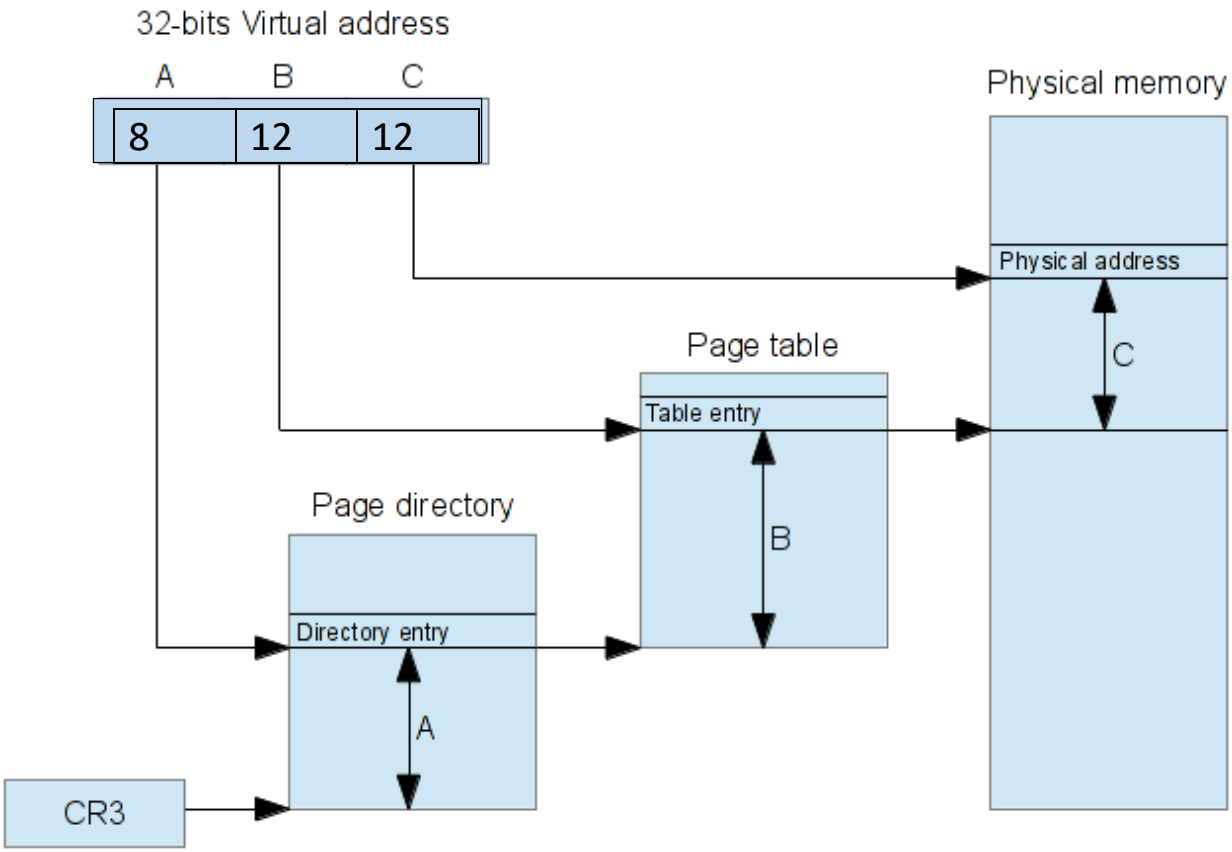
32 bit addressing, with the same set,
First level Page Directory look up remains same
Second level page lookup wont exist for 16MB paging
Offset will be 24bit for 16 MB page.

During 4KB Page Table: With 2 level hierarchy

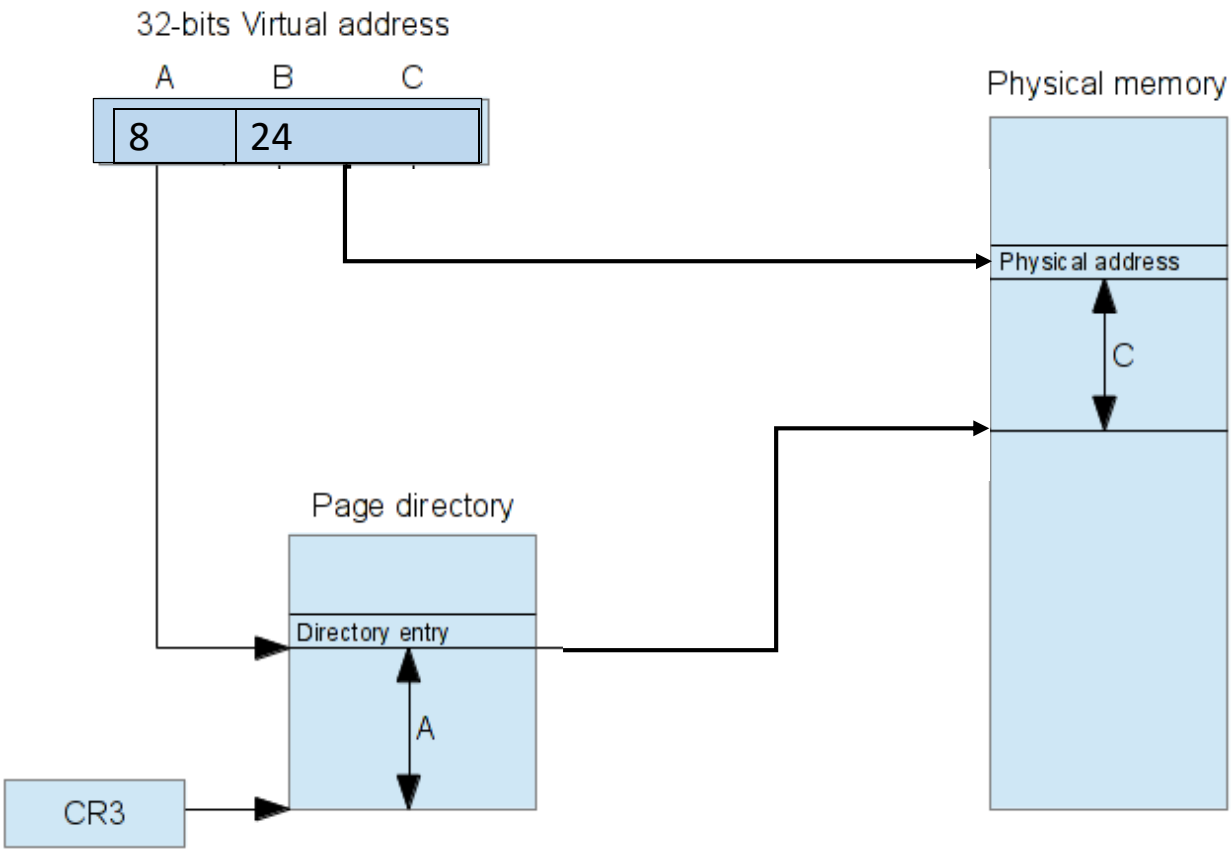
8 bits	12 bits	12 bits
Page Directory Index	Page Table Index	Address Offset

During 16MB Page Table: with 1 level hierarchy

8 bits	24 bit
Page Directory Index	Address Offset



The proposed idea with 2 level paging scheme will suffice for 4KB frame size.



The same architecture idea with 1 level paging scheme will convert to 16MB frame size.

The Design with Hardware Support

There needs to be control in the hardware for switching to different configuration. Assuming the switchablity.

After paging is enabled hardware needs to translate virtual to physical translation for the page address and offsets need to be looked up.

- In the Page Director, an entry points to 16MB of larger page or 4096 different pages of 4KB sizes.

- If we assuming hardware has a way of knowing lookup addresses based on size. Design is simple since the direct lookup (skipping level page lookup possible)
- If hardware doesn't have support, then we will have to store the linear address pointers for 4096 small 4KB frame sizes to replicate behavior of contiguous 16MB memory.

Pool frame sizes and Sizes for storing the info

The design needs different types of contiguous memory for the requirement of maintaining 16MB, 4KB dual support.

- 12 bit offset address → 4KB frame sizes = Contiguous addresses
- 24 bit offset address → 16MB frame sizes = Contiguous addresses
- 12 bit offset page table → 4096 entries of 4 bytes pointers → 16KB
- 8 bits offset page directory → 256 entries of 4bytes pointers → 1KB

1. 4KB frame size is already implemented not a problem
2. 16MB frame needs an extra pool with different allocation, contiguous memory of 16MB is need for this purpose.
3. 16KB frame size table needed for storing internal second level info
4. Page Directory information can be stored in Kernel Memory 1KB < 4KB hence we can use a frame from kernel memory pool

There can be separate pools constructed in organizing physical memory
one for smaller 4KB sizes and another for larger page 16MB sizes. Some smaller section of pool only for 16KB tables(or 4 contiguous 4KB frames are needed)

16MB pool → this will be difficult to implement in our current 32MB system, with limited frames.

Design Pros:
Large page size are useful in case of high performance computing applications where data higher granularity and huge chunks of data helps

- Higher virtual memory
- Reduced TLB misses in case of huge memory
- Memory prefetching also helpful

Cons:

- TLB bottlenecks
- Debugging Issues
- Fragmentation issues
- Alignment problems

Changes in the code for the design:

- Frame Pool.c => Different pool organizations should be supported
    - Implementation in the current allocations needs to be changed to use accordingly instead of using hard coded address.
    - Bit manipulations need to be changed
    - Alignement in physical address for 16MB , 16KB, 4KB needs to be handled
- Page Table.c
    - Data structures can be reused as it is, most of the page table structure remain as it is.
    - Info frames always allocate from – 4KB,16KB Pools
    - Data frames always allocate from – Process Pool
    - Shift operations calculating the left shift and right shift operations. Masking operations need to be done accordingly instead of hardcoded values.
- Using the extra bits in the page directory entry, code needs to be updated handling page-fault exceptions according to memory request
- ENTRIES PER PAGE changes from 1024 to 64, can be calculated based on frame size
- Bit manipulation
    - Recursive lookup last entry index calculation need to be updated based on selected frame size.

References:

1. Wikipedia
2. osdev.net – tutorials and wiki pages
3. http://www.rohitab.com/discuss/topic/31139-tutorial-paging-memory-mapping-with-a-recursive-page-directory/