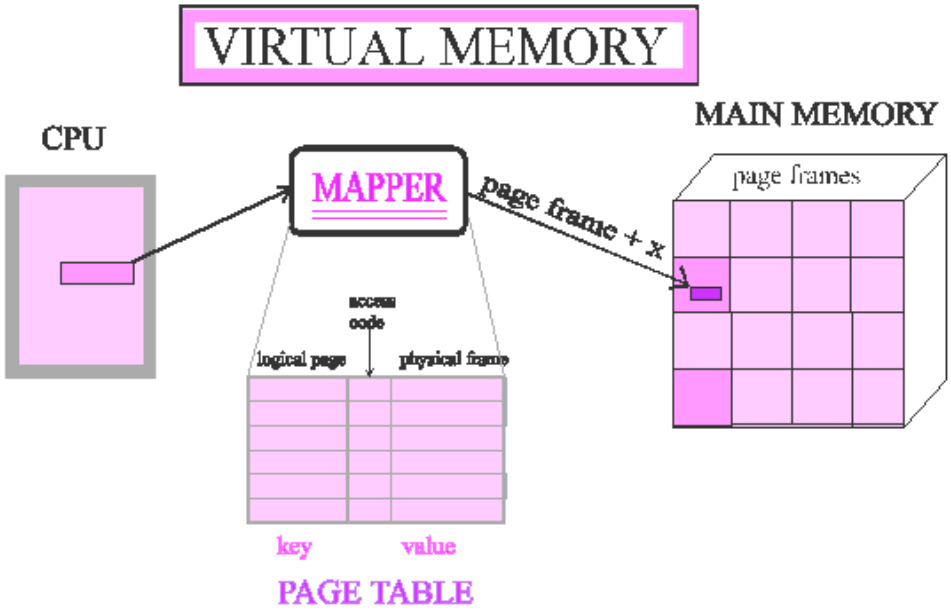
Use of multi-level page tables in virtual memory management

INTRODUCTION

What are virtual memory and page tables?

Virtual memory is a feature which simulates additional main memory such as RAM or disc storage. This allows the management of memory and loading multiple programs simultaneously as it virtually creates an infinite memory for each program. Virtual memory permits programs to use additional memory from the disc space as temporary storages. This is done using a mapping technique. This mapping technique is called page tables. This concept allows each virtual page to be mapped to a page frame independently.



DETAILS

When a program executes, particular data that refers to the program is usually loaded to the RAM. But considering the hardware cost RAM capacity is limited. This is the main reason why virtual memory play a big role in memory management. When considering this page tables, they have physical addresses mapped with a virtual addresses. Each of this Entry is called Page Table Entry (PTE). These physical addresses are the one which are to be used as temporary storages. So, when the program executes it refers virtual address which direct to the physical address mapped by page table.

How big is a typical page table?

Assume 32-bit address space, 4KB pages and 4 byte PTEs

Answer: 2 ^ (32 -log(4KB)) * 4 = 4 MB

- Page table size = Number of entries * size of each entry
- Number of entries = Number of virtual pages = 2^(bits for VPN)
- Bits for VPN = 32 – number of bits for page offset = 32 – log(4KB) = 32 –12 = 20
- Number of entries = 2^20 = 1 MB
- Page table size = Number of entries * 4 bytes = 4 MB

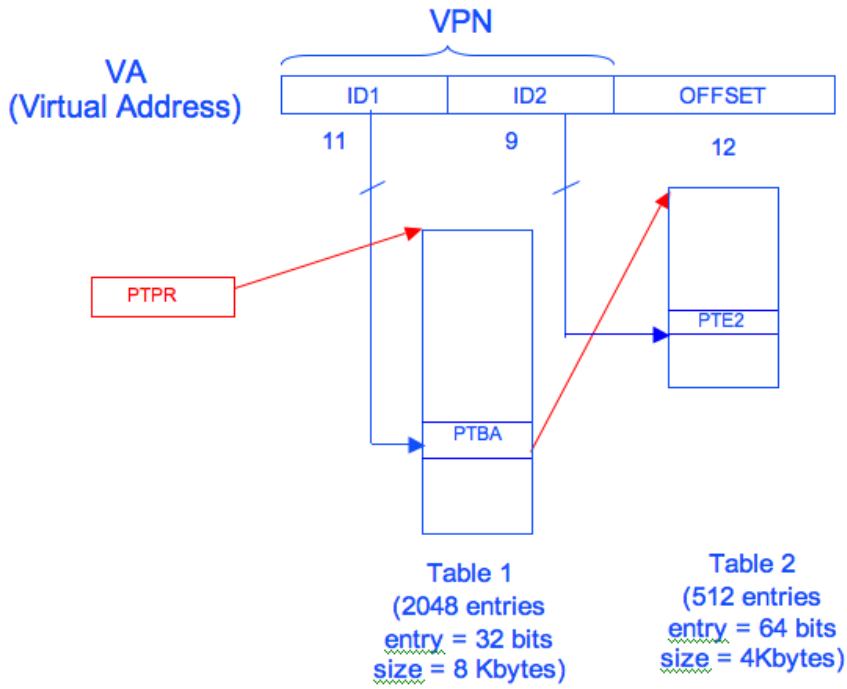
* VPN --> (Virtual Page Number)

It is not a big value of memory when we compare it with present day computer specifications.

BUT, each program needs its own page table. If a computer run 100 programs 400MB of memory is going to be allocated only for page tables. If page table is not in the RAM there is no way to find it. This is not good…!!!

This can be solved using page tables to represent another page tables by its entries. This is the concept which is called multiple page tables.

Multiple Page Tables



Above figure shows a two level page table. According to that first 11 bits of virtual address is referred to find the page table. 9 bits after that is referred to find the page table entry from the page table which was pointed by the first 11 bits.

Let's see that how this structure behave efficiently.

How big is multiple page table in above example?

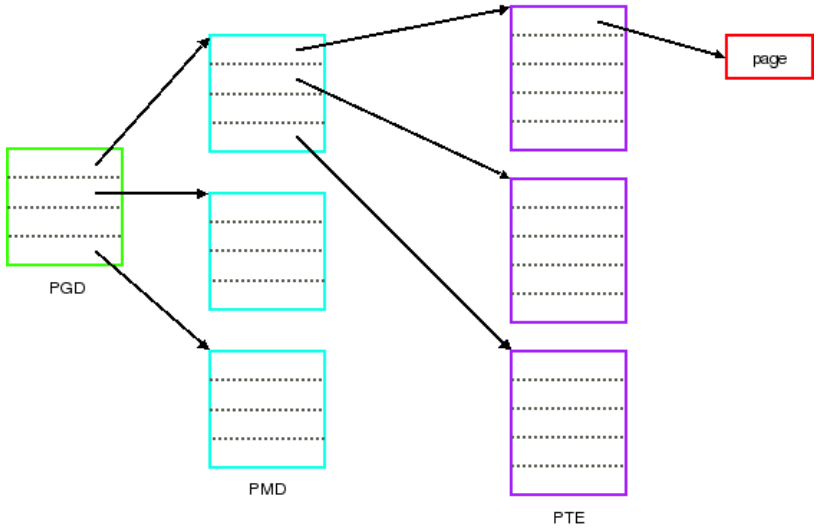
Assume that table 1 got 2048 entries of 32 bit size and table 2 got 512 entries of 64 bit size

Answer : 12 kB

- Size of table 1 = (2048*32)/8 = 8192 bits = 8 kBytes
- Size of table 2 = (512*32)/8 = 4096 bits = 4 kBytes
- Page table size = 8kB + 4kB = 12kB

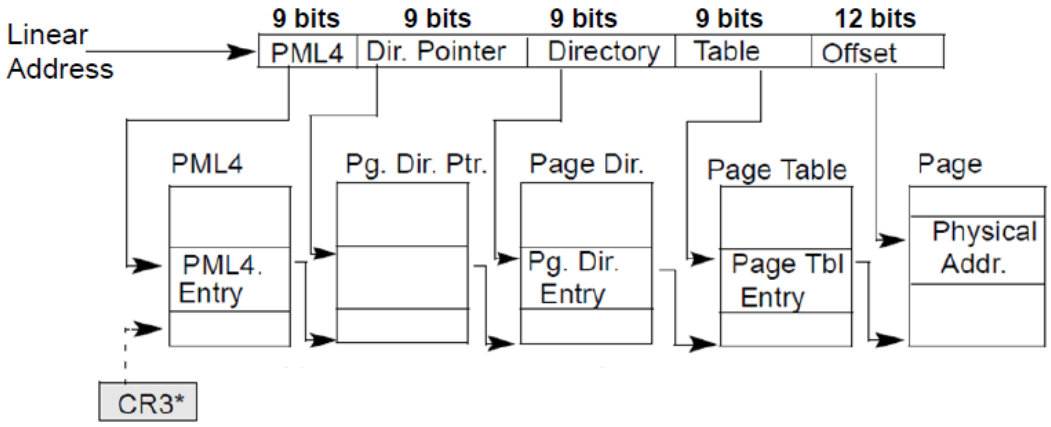
If computer run 100 programs it need 1200kB memory in order to save page tables. This is better…!!!

Efficiency of this mechanism can be improved by using multiple levels.



Above figure shows how the page table mapping is done in 4 level page table. This is actually a tree structure. Leaves of this tree structure represent the page tables.

In 64 bit x86 system page tables are implemented in 5 levels which direct towards a huge tree structure.



CONCLUSION

- Problem: Simple linear page tables require too much contiguous memory
- Multi - level page table is the answer
- Tree structure is the best one for efficiently organizing page tables
- x86 architecture which is popular in present uses this tree structure based multi – level page tables

REFERENCES

- https://www.techopedia.com/definition/4773/virtual-memory-vm
- https://compas.cs.stonybrook.edu/~nhonarmand/courses/fa17/cse306/slides/06-paging.pdf

GROUP 14 E/14/403

E/14/108

E/14/244