ECE472 Advance Computer Architecture - Homework 3

Assignment 3

Rattanai Sawaspanich

For a 4KB page, and a 32 bit address space, calculate the amount of memory needed to store a process's page tables. Assume each entry in the page table requires 10 bytes. Show all calculations.

Given: Each entry in Page Table has a size of 10 Bytes. The Page frame has the size of 4 KByte. Each entry in the Frame Page takes up 32 bits (4 bytes)
Assumption: Each Byte conprises of 8 bits.
Solution:
32-bit address space can be used to refer total of 2^{32} Bytes
which is equipvalent to

$$\frac{2^{32}(byte)}{4096(byte/page)} = 1048576(page)$$

.
Therefore, the size of page table is $1048576(page) \times 10(byte/page) = 10485760(byte) = 10.485MB$

For a 4KB page, and a 64 bit address space, calculate the amount of memory needed to store a process's page tables. Assume each entry in the page table requires 10 bytes. Show all calculations.

Given: Each entry in Page Table has a size of 10 Bytes. The Page frame has the size of 4 KByte. Each entry in the Frame Page takes up 64 bits (8 bytes)
Assumption: Each Byte conprises of 8 bits.
Solution:
64-bit address space can be used to refer total of 2^{64} Bytes
which is equipvalent to

$$\frac{2^{64}(byte)}{4096(byte/page)} = 4.50 \times 10^{15}(page)$$

.
Therefore, the size of page table is $4.50 \times 10^{15}(page) \times 10(byte/page) = 4.50 \times 10^{16}(byte) = 45PB$

For a 8KB page, and a 32 bit address space, calculate the amount of memory needed to store a process's page tables. Assume each entry in the page table requires 10 bytes. Show all calculations.

Given: Each entry in Page Table has a size of 10 Bytes. The Page frame has the size of 8 KByte. Each entry in the Frame Page takes up 32 bits (4 bytes)
Assumption: Each Byte conprises of 8 bits.
Solution:
32-bit address space can be used to refer total of 2^{32} Bytes
which is equipvalent to

$$\frac{2^{32}(byte)}{8192(byte/page)} = 524288(page)$$

.
Therefore, the size of page table is $524288(page) \times 10(byte/page) = 5242880(byte) = 5.243MB$

For a 8KB page, and a 64 bit address space, calculate the amount of memory needed to store a process's page tables. Assume each entry in the page table requires 10 bytes. Show all calculations.

Given: Each entry in Page Table has a size of 10 Bytes. The Page frame has the size of 8 KByte. Each entry in the Frame Page takes up 64 bits (8 bytes)
Assumption: Each Byte conprises of 8 bits.
Solution:
64-bit address space can be used to refer total of 2^{64} Bytes
which is equipvalent to

$$\frac{2^{64}(byte)}{8192(byte/page)} = 2.25 \times 10^{15}(page)$$

.
Therefore, the size of page table is $2.25 \times 10^{15}(page) \times 10(byte/page) = 2.25 \times 10^{16}(byte) = 22.5PB$

Describe the concept of pipelining, and why it is useful.

Pipelining is a method of dividing process into independent stages and move objects through stages in sequence. At a given time, multiple objects are being processed. Fundamentally instruction pipelining is broken up into five independent stages: Fetch, Decode, Execute, Memory-Access(Read), and Memory-Update(Write). Pipelining a very useful technique because it allows most (if not all) the hardware resouces to be used; it improves an overall system performance. Although, pipelining is a nifty techinique but there are some trade-off . (Please take a look at next two question).

Describe the IA-32e paging structure, in detail.

Overview

Accroding to Intel Architectures Developer's Manual, there are four levels of system data structures including **the Page Map Level 4, A set of page direcotry pointer tables, Sets of page directories, Sets of Page tables.**

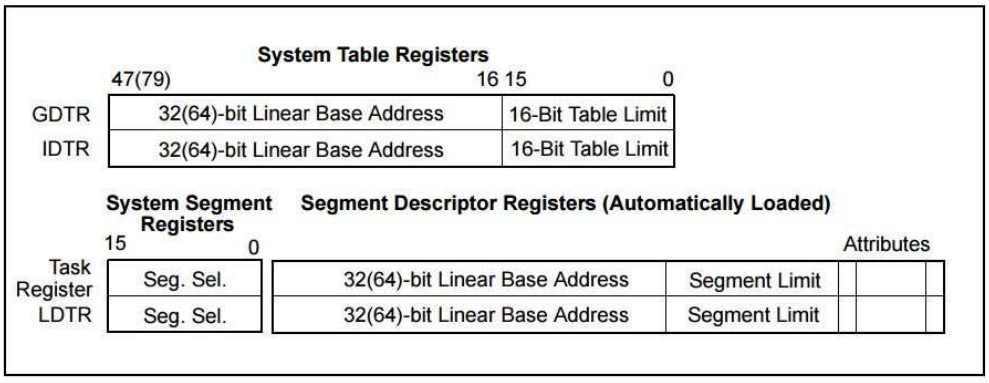
The Page Map Level4 (PML4)	An entry in PML4 contains a paddr of the base of a Page Directory Pointer Table. The base physical address of the PML4 is stored in CR3
A set of Page Directory Pointer Table	An entry in PDPT contains paddr point to base of Page Directory Table.
Sets of Page Directories	An entry in Page Directory table contains the physical address of the Page Table
Sets of Page Table	An entry in a Page Table contains the paddr of a Page (page frame).

NOTE: The addressses in each data layer are 64 bits physical address value. In order for the system to be able to keep track of all the addresses, four system-descriptor registers are required: GDTR, IDTR, LDTR, and TR. The registers are expanded in hardware to handle the 64 bit addressing.

Furthermore, the paging system arhitecture also follows the Operating System Instruction, Performance-monitoring counter and Internal Caches and Buffers. The Performance-noitoring counter are event counters that counts processor event. Generally, a processor has multiple internal caches and buffers. The caches are used to store both data and instructions. The buffers are used to stored decoded addresses to the program segment and write operations waiting to be processed.

Memory Management Registers

GDTR, LDTR, IDTR, and TR are four system memory-management registers that contain physical addresses of the data structures which control segmented memory mamangement. The figure below shows the format of the register layout.



Global Descriptor Table Rekgister (GDTR)
GDTR register contains base physical address (64 bits in IA-32e mode) and the 16-bits table limit (offset).

The physical address points to byte 0 of GDT and the table limit (an offset) specifies the number of byte in the table.

Local Descriptor Table Register (LDTR)
LDTR contains the 16-bit segment selector: base address, segment limit, and descriptor attributes for the LDT. The base address specifies the linear address of byte 0 of LDT segment annd the segment limit specifies the number of byte in thesegment.

IDTR Interrupt Descriptor Table Register
IDTR register contains base address and 16-bit table limit for the IDT. The base address specifies the linear address of byte 0 of the IDT and the table limit specifies the number of bytes in the table.

Task Register (TR)
TR holds 16-bits segment selector, base address, segment limit, and descriptor attributes for the TSS of teh current task. The selector references the TSS descriptor in the GDT. The base address specifies the address of byte 0 of the TSS and the segment limit specifies the number of bytes in the TSS.

Note: All the register listed above are reset (set to defualt value of 0) upon power up.

Paging Modes

Paging Mode	PG in CR0	PAE in CR4	LME in IA32_EFER	Lin.-Addr. Width	Phys.-Addr. Width[1]	Page Sizes	Supports Execute-Disable?	Supports PCIDs and protection keys?
None	0	N/A	N/A	32	32	N/A	No	No
32-bit	1	0	0[2]	32	Up to 40[3]	4 KB 4 MB[4]	No	No
PAE	1	1	0	32	Up to 52	4 KB 2 MB	Yes[5]	No
IA-32e	1	1	1	48	Up to 52	4 KB 2 MB 1 GB[6]	Yes[5]	Yes[7]

Note: Every pageing strucuture is 4096 Bytes in size (4KB) and comprised a number of individual entires.

If CR0.PG = 0, the paging is disable and the rest of the parameters are ignored. The logical proccessor treats all virtual address as if they were physical physical addresses.

If CR0.PG = 1, the paging is enable if the protection is also enable (CR0.PE = 1). There are three modes of paging: 32-bit paging, PAE paging, and IA-32e paging.

32-bit paging [CR0.PG = 1 and CR4.PAE = 0]
In the 32-bit paging mode, each entry is 32 bits (4 bytes); there are 1024 entires in each strucutre.

PAE paging [CR4.PAE = 1 and IA32-EFER.LME = 0]
In PAE paging mode, normally each entry is 64 bits (8 bytes); there are 512 entires in each structure. Except when the mode is specified to have each entry size of 32 bytes; there are **four** 64-bit entries.

IA-32e paging [CR0.PG = 1, CR4.PAE = 1, and IA32-EFER.LME = 0]
In IA-32e paging mode, each entry is 64 bits (8 bytes); there are 512 entires in each structure.

Note: For a specific configuration i.e. page size, supervisor mode, execution rights. Pleaes take a look at **Paging-Mode Modifiers** in Intel 64 and IA-32 Architectures Software Developer's Manual Vol.3 section 4.1.3.

Paging Translation

The processor uses the higher portion of a logical address to identify a series of paging-structure entires. The last of these entries contains the physical address of the region to which the virtual address translates. (a Page Frame). The lower portion of the logical address is a Page Offset that specifies the address within the page frame. Each paging-structure contains a physical address that either the address of another paging structure (referencing) or the address of a Page Frame (mapping a page).

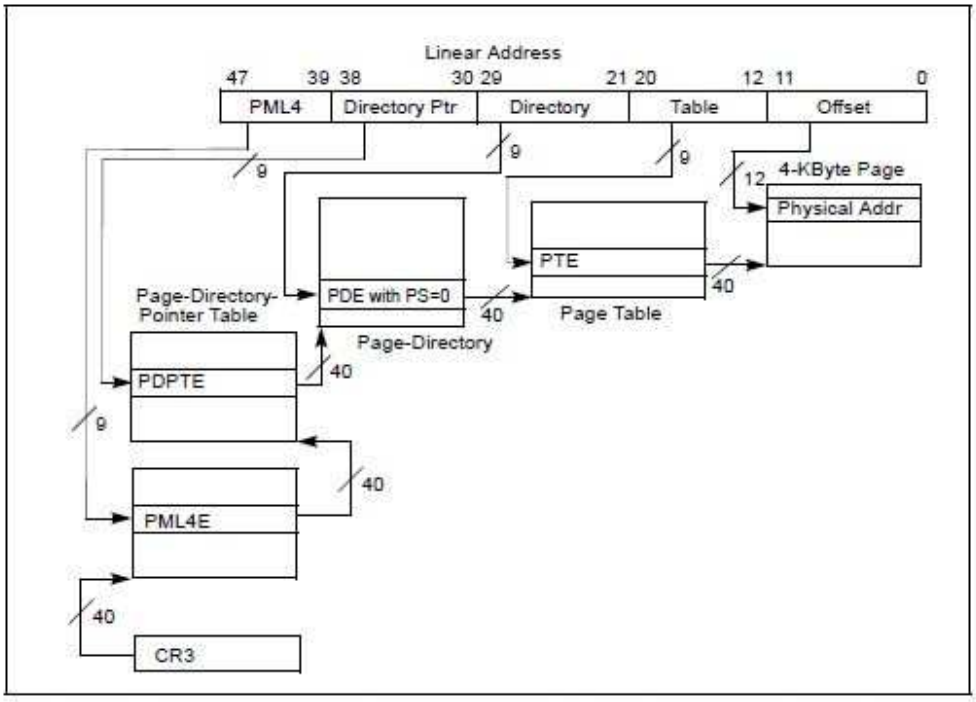


Figure 4-8. Linear-Address Translation to a 4-KByte Page using IA-32e Paging

For an example in case of IA-32e Paging mode, the processor starts at CR3 register looking into bit M-1:12 for a physical address of the 4KB aligned PML4 table. Then the processor would resolve to PML4 entry, look into bit M-1:12 for address of the page-directory-pointer table (PDPTE). In PDPTE, the processor looks for the address of page director (PDE), bit M-1:12, and the PDE offset, bit M-1:30. Afterwards, the processor resolves to the PDE and select a byte with the PDE offset to obtain an address to 2MB-page frame and address of page table (offset). In 2MB-page frame, the processor would look for the address of 4KB page frame and the offset from the original logical address to resolve for the translated physical address.

Note: For more detail of the bit layout, please take a look at P.2084 of Intel SDM Vol.3

What are some examples of pipeline hazards, and what are some ways of dealing with them?

Complexity

- The more pipelining stages, the more prone there is to the system errors. It is because of the complexity of how the instructions are breaking up and how to the data path layout. For a piplining to be resilient, the system requires to have a better error handler.
- The edge cases can be very tricky in a pipelining architecture. There are multiple blocks of instructions and each of them have their own unique edge-cases that may cause a system failure or an unpredictable result.

Non-Determinism

- You cannot guarantee the actual execution time for each instruction because the run time of the current instruction also depending on the empty stages there are from the previous instructions. Although, you can still find an accurate average run time for an instruction.
- Pipelining cannot really be used in a system that requires a synchronization in a high clock frequency. It is due to the propagatoin delay at the hardware level; an electron moving through a wire takes time.
- Branching in a pipelining is a nightmare because if the destination stage is currently in used, the stage will be flushed in replace with the instruction. Thus, the system takes a severe performance panalty.

Instruction Ordering

- Intel has an out-of-order instruction execution. With pipelining the complexity of the system just goes off the roof. Moreover, the order of the execution (even though it is out of order) needs to be re-arrange to optimize the performance of the pipeline.
- Typically, a program would have to Read/Write a section of data. In pipelining, it is possible for a program to accidentally get a piece of an outdated data becaues a piece of data is being processed on a different stage in the pipeline.