Operating Systems and Concurrency

Lecture 15: Memory Management IV

G52OSC

Geert De Maere and Isaac Triguero
{Geert.DeMaere,Isaac.Triguero}@Nottingham.ac.uk

University Of Nottingham
United Kingdom

2018

Recall

Last Lecture

- The **principles of paging** are:
  - Main memory is divided into small equal sized **frames**
  - Each process is divided into **pages** of equal size
  - A page table contains multiple “relocation registers” (**page table**) to map the pages on to frames
- The **benefits** of paging include:
  - Reduced **internal fragmentation**
  - No **external fragmentation**

Goals for Today

Overview

- Address Translation Implementation (revisited)
- Principles behind **virtual memory**
- Complex/large **page tables**

Recall

Memory as a linear array

- Memory can be seen as one **linear array** of **bytes**/words: *byte memory [N];*
- Address ranges from 0 through (N - 1)
- N address lines can be used to specify 2^N distinct addresses
- E.g. in a 16-bit machine, we can address up to 2^{16} distinct addresses. Nowadays, **each cell** of memory is typically a **byte**. Thus, we can address up to 64 kilobytes.
- If the memory is split into 16 blocks (=2^4); Block size = $2^{16}/2^4 = 2^{12}$ = 4 kilobytes

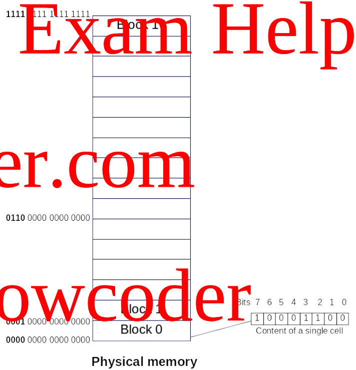
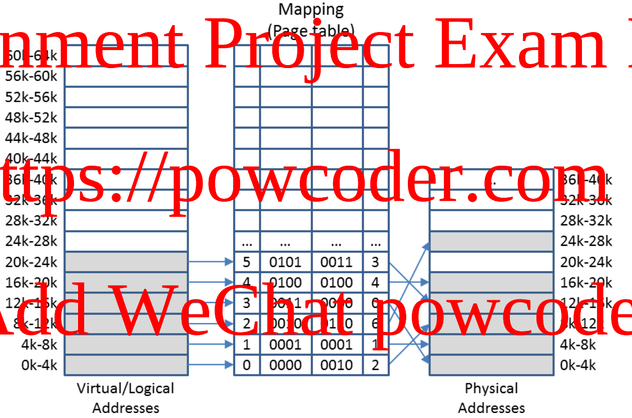


Figure: Memory as a linear array of bytes (16-bit)

Paging

Relocation: Address Translation



...	...	...	...
5	0101	0011	3
4	0100	0100	4
3	0011	0000	0
2	0010	0110	6
1	0001	0001	1
0	0000	0010	2

Figure: Address Translation

- A **logical (physical) address** is relative to the start of the **program (memory)** and consists of two parts:
  - The **right most** *m* **bits** that represent the **offset within the page (frame)**
    - e.g. 12 bits for the offset, allowing up to 4096 (2^{12}) bytes per page (frame)
  - The **left most** *n* **bits** that represent the **page (frame) number**
    - e.g. 4 bits for the page number allowing 16 (2^4) pages (frames)
- The **offset** within the page and frame **remains the same** (they are the same size)
- The page number to frame number mapping is held in the **page table**

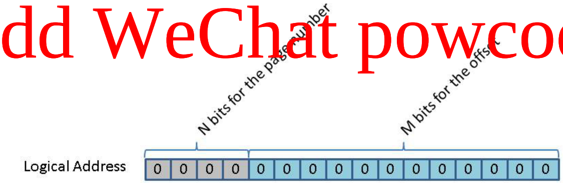


Figure: Logical Address

Paging

Address Translation: Implementation

- A **logical (physical) address** is relative to the start of the **program (memory)** and consists of two parts:
    - The **right most** *m* **bits** that represent the **offset within the page (frame)**
        - e.g. 12 bits for the offset, allowing up to 4096 (2^{12}) bytes per page (frame)
    - The **left most** *n* **bits** that represent the **page (frame) number**
        - e.g. 4 bits for the page number allowing 16 (2^4) pages (frames)
- The **offset** within the page and frame **remains the same** (they are the same size)
- The page number to frame number mapping is held in the **page table**

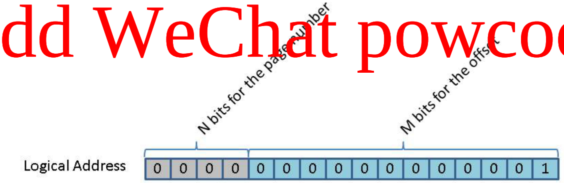


Figure: Logical Address

Paging

Address Translation: Implementation

- A **logical (physical) address** is relative to the start of the **program (memory)** and consists of two parts:
  - The **right most** *m* **bits** that represent the **offset within the page (frame)**
    - e.g. 12 bits for the offset, allowing up to 4096 (2^{12}) bytes per page (frame)
  - The **left most** *n* **bits** that represent the **page (frame) number**
    - e.g. 4 bits for the page number allowing 16 (2^4) pages (frames)
- The **offset** within the page and frame **remains the same** (they are the same size)
- The page number to frame number mapping is held in the **page table**

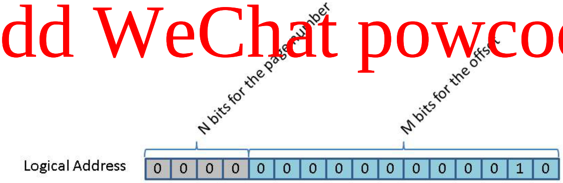


Figure: Logical Address

Paging

Address Translation: Implementation

- A **logical (physical) address** is relative to the start of the **program (memory)** and consists of two parts:
  - The **right most** *m* **bits** that represent the **offset within the page (frame)**
    - e.g. 12 bits for the offset, allowing up to 4096 (2^{12}) bytes per page (frame)
  - The **left most** *n* **bits** that represent the **page (frame) number**
    - e.g. 4 bits for the page number allowing 16 (2^4) pages (frames)
- The **offset** within the page and frame **remains the same** (they are the same size)
- The page number to frame number mapping is held in the **page table**

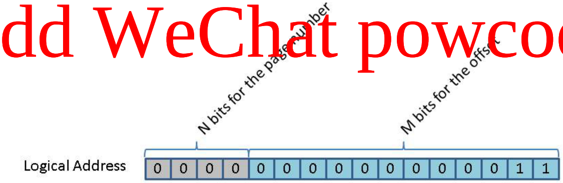


Figure: Logical Address

Paging

Address Translation: Implementation

- A **logical (physical) address** is relative to the start of the **program (memory)** and consists of two parts:
  - The **right most** *m* **bits** that represent the **offset within the page (frame)**
    - e.g. 12 bits for the offset, allowing up to 4096 (2^{12}) bytes per page (frame)
  - The **left most** *n* **bits** that represent the **page (frame) number**
    - e.g. 4 bits for the page number allowing 16 (2^4) pages (frames)
- The **offset** within the page and frame **remains the same** (they are the same size)
- The page number to frame number mapping is held in the **page table**

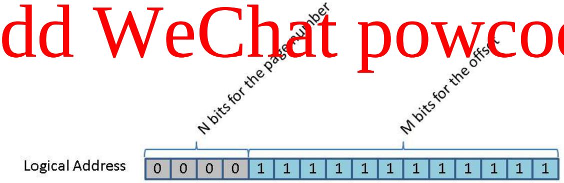


Figure: Logical Address

Paging

Address Translation: Implementation

- A **logical (physical) address** is relative to the start of the **program (memory)** and consists of two parts:
  - The **right most** *m* **bits** that represent the **offset within the page (frame)**
    - e.g. 12 bits for the offset, allowing up to 4096 (2^{12}) bytes per page (frame)
  - The **left most** *n* **bits** that represent the **page (frame) number**
    - e.g. 4 bits for the page number allowing 16 (2^4) pages (frames)
- The **offset** within the page and frame **remains the same** (they are the same size)
- The page number to frame number mapping is held in the **page table**

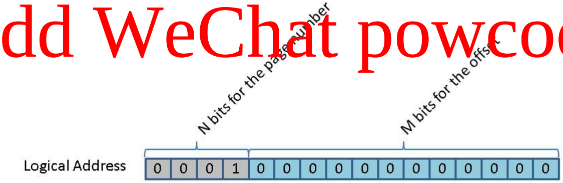


Figure: Logical Address

Paging

Address Translation: Implementation

- A **logical (physical) address** is relative to the start of the **program (memory)** and consists of two parts:
  - The **right most** *m* **bits** that represent the **offset within the page (frame)**
    - e.g. 12 bits for the offset, allowing up to 4096 (2^{12}) bytes per page (frame)
  - The **left most** *n* **bits** that represent the **page (frame) number**
    - e.g. 4 bits for the page number allowing 16 (2^4) pages (frames)
- The **offset** within the page and frame **remains the same** (they are the same size)
- The page number to frame number mapping is held in the **page table**

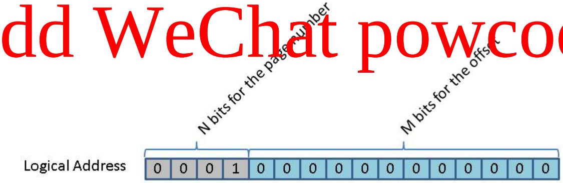


Figure: Logical Address

Paging

Address Translation: Implementation

- A **logical (physical) address** is relative to the start of the **program (memory)** and consists of two parts:
    - The **right most** *m* **bits** that represent the **offset within the page (frame)**
        - e.g. 12 bits for the offset, allowing up to 4096 (2^{12}) bytes per page (frame)
    - The **left most** *n* **bits** that represent the **page (frame) number**
        - e.g. 4 bits for the page number allowing 16 (2^4) pages (frames)
- The **offset** within the page and frame **remains the same** (they are the same size)
- The page number to frame number mapping is held in the **page table**

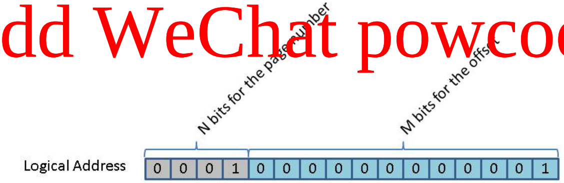


Figure: Logical Address

Paging

Address Translation: Implementation

- A **logical (physical) address** is relative to the start of the **program (memory)** and consists of two parts:
  - The **right most** *m* **bits** that represent the **offset within the page (frame)**
    - e.g. 12 bits for the offset, allowing up to 4096 (2^{12}) bytes per page (frame)
  - The **left most** *n* **bits** that represent the **page (frame) number**
    - e.g. 4 bits for the page number allowing 16 (2^4) pages (frames)
- The **offset** within the page and frame **remains the same** (they are the same size)
- The page number to frame number mapping is held in the **page table**

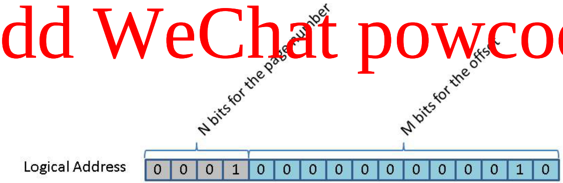


Figure: Logical Address

Paging

Address Translation: Implementation

- A **logical (physical) address** is relative to the start of the **program (memory)** and consists of two parts:
    - The **right most** *m* **bits** that represent the **offset within the page (frame)**
        - e.g. 12 bits for the offset, allowing up to 4096 (2^{12}) bytes per page (frame)
    - The **left most** *n* **bits** that represent the **page (frame) number**
        - e.g. 4 bits for the page number allowing 16 (2^4) pages (frames)
- The **offset** within the page and frame **remains the same** (they are the same size)
- The page number to frame number mapping is held in the **page table**

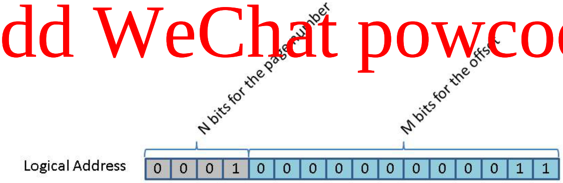
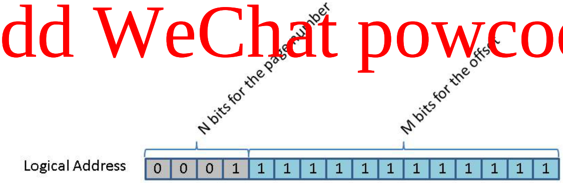


Figure: Logical Address

Paging

Address Translation: Implementation

- A **logical (physical) address** is relative to the start of the **program (memory)** and consists of two parts:
  - The **right most** *m* **bits** that represent the **offset within the page (frame)**
    - e.g. 12 bits for the offset, allowing up to 4096 (2^{12}) bytes per page (frame)
  - The **left most** *n* **bits** that represent the **page (frame) number**
    - e.g. 4 bits for the page number allowing 16 (2^4) pages (frames)
- The **offset** within the page and frame **remains the same** (they are the same size)
- The page number to frame number mapping is held in the **page table**



0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1

Figure: Logical Address

Paging

Address Translation: Implementation

- A **logical (physical) address** is relative to the start of the **program (memory)** and consists of two parts:

  - The **right most** *m* **bits** that represent the **offset within the page (frame)**
    - e.g. 12 bits for the offset, allowing up to 4096 (2^{12}) bytes per page (frame)
  - The **left most** *n* **bits** that represent the **page (frame) number**
    - e.g. 4 bits for the page number allowing 16 (2^4) pages (frames)
- The **offset** within the page and frame **remains the same** (they are the same size)

- The page number to frame number mapping is held in the **page table**

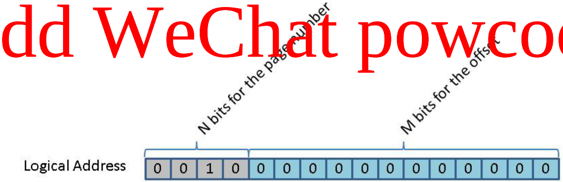
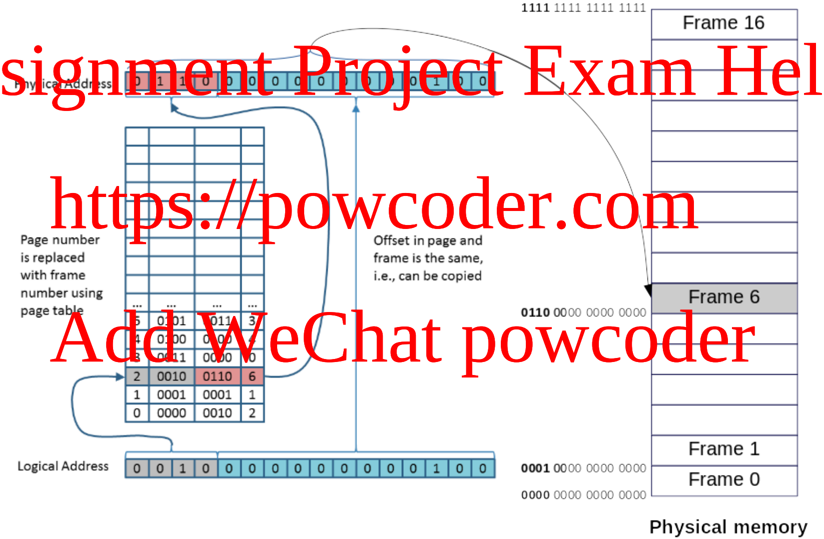


Figure: Logical Address

Paging

Address Translation: Implementation



...	...	...	...
5	0101	0011	3
4	0100	0100	4
3	0011	0000	0
2	0010	0110	6
1	0001	0001	1
0	0000	0010	2

Figure: Address Translation

Paging

Relocation: Address Translation

- Steps in **address translation**:
  1. **Extract the page number** from logical address
  2. Use page number as an index to **retrieve the frame number** in the page table
  3. **Add the “logical offset within the page”** to the start of the physical frame
- **Hardware implementation** of address translation
  1. The CPU’s **memory management unit** (MMU) intercepts logical addresses
  2. MMU uses a page table as above
  3. The resulting **physical address** is put on the **memory bus**

Virtual Memory

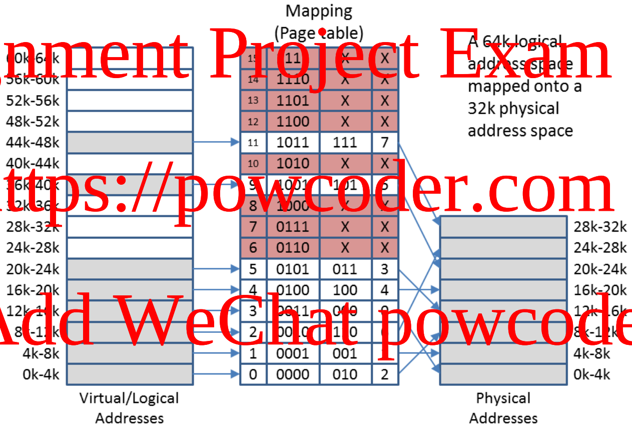
Principle of Locality

- Are there any other **benefits of paging**?
  - Code execution and data manipulation are usually **restricted to a small subset** (i.e. limited number of pages) at any point in time
  - I.e. **code** and **data references** within a process are usually **clustered** ⇒ This is called the **principle of locality**

- **Not all pages** have to be **loaded** in memory at the same time ⇒ **virtual memory**
  - Loading an entire set of pages for an entire program/data set into memory is **wasteful**
  - Desired blocks could be **loaded on demand**

Virtual Memory

An Example



A 64k logical address space mapped onto a 32k physical address space

60k-64k	15	1111	X	X
56k-60k	14	1110	X	X
52k-56k	13	1101	X	X
48k-52k	12	1100	X	X
44k-48k	11	1011	111	7
40k-44k	10	1010	X	X
36k-40k	9	1001	101	5
32k-36k	8	1000	X	X
28k-32k	7	0111	X	X
24k-28k	6	0110	X	X
20k-24k	5	0101	011	3
16k-20k	4	0100	100	4
12k-16k	3	0011	000	0
8k-12k	2	0010	110	6
4k-8k	1	0001	001	1
0k-4k	0	0000	010	2

Figure: Virtual Memory

- The **resident set** refers to the pages that are loaded in main memory
- A **page fault** is generated if the processor accesses a page that is **not in memory**

  - A page fault results in an **interrupt** (process enters **blocked state**)
  - An **I/O operation** is started to bring the missing page into main memory
  - A **context switch** (may) take place
  - An **interrupt signals** that the I/O operation is complete (process enters **ready state**)

Demand Paging

Processing Page Faults

1. Trap operating system
   - Save registers/process state
   - Analyse interrupt (i.e., identify page fault)
   - Validate page reference, determine page location
   - Issue disk I/O: queueing, seek, latency, transfer
2. Context switch (optional)
3. Interrupt for I/O completion
   - Store process state/registers
   - Analyse interrupt from disk
   - Update page table (page in memory)
   - Wait for original process to be scheduled
4. Context switch to original process

Virtual Memory

The Benefits

- Being able to maintain **more processes** in main memory through the use of virtual memory **improves CPU utilisation**
  - Individual processes take up less memory since they are only partially loaded
- Virtual memory allows the **logical address space** (i.e processes) to be **larger than physical address space** (i.e main memory)
  - 64 bit machine ⇒ 2^{64} logical addresses (theoretically)

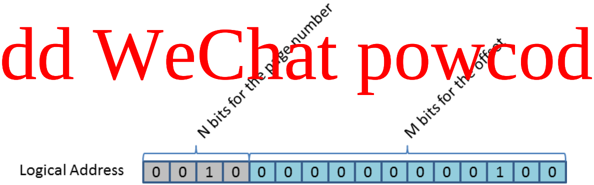


Figure: Logical Address ⇒ physical address

Virtual Memory

The Benefits

- Being able to maintain **more processes** in main memory through the use of virtual memory **improves CPU utilisation**
    - Individual processes take up less memory since they are only partially loaded
- Virtual memory allows the **logical address space** (i.e processes) to be **larger than physical address space** (i.e. main memory)
    - 64-bit machine ⇒ 2^{64} logical addresses (theoretically)

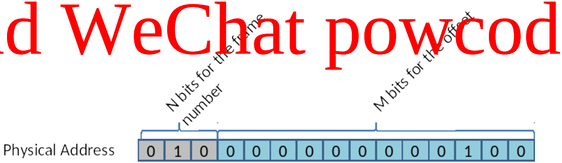
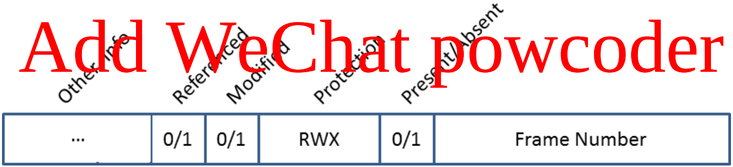


Figure: Logical Address ⇒ physical address

Virtual Memory

Page Tables Revisited: Contents of a Page Entry

- A “**present/absent bit**” that is set if the page/frame is in memory
- A “**modified bit**” that is set if the page/frame has been modified (only modified pages have to be written back to disk when evicted)
- A “**referenced bit**” that is set if the page is in use
- **Protection and sharing bits**: read, write, execute or combinations thereof



Other Info	Referenced	Modified	Protection	Present/Absent	
...	0/1	0/1	RWX	0/1	Frame Number

Figure: Page Table Entry

Virtual Memory

Page Tables Revisited: Page Table Size

- On a **16 bit machine**, the total address space is 2^{16}
  - Assuming that 10 bits are used for the offset (2^{10})
  - 6 bits can be used to number the pages
  - i.e. $2^6 = 64$ pages can be maintained

- In a **32 bit machine**, total address space is 2^{32}
  - Assuming pages of 2^{12} bits (4KB)
  - 20 bits can be used to number the pages
  - i.e. 2^{20} pages (approx. 1 million) can be maintained

- On a **64 bit machine** …

- How do we deal with **the increasing size of page tables**, i.e., where do we store them?
  - Their size prevents them from being **stored in registers**
  - They have to be stored in (virtual) **main memory**:
    - **Multi-level** page tables
    - **Inverted page tables** (for large virtual address spaces)
- How can we maintain **acceptable speeds**?: address translation happens at every memory reference, it has to be fast!
  - Accessing main memory results in **memory stalls**

Virtual Memory

Page Tables Revisited: Multi-level Page Tables

- **Solution**: Page the page table!
- We keep tree-like structures to hold page tables
- Divide the page number into
  - An index to a page table of second level
  - A page within a second level page table
- No need to keep all page tables in memory all time

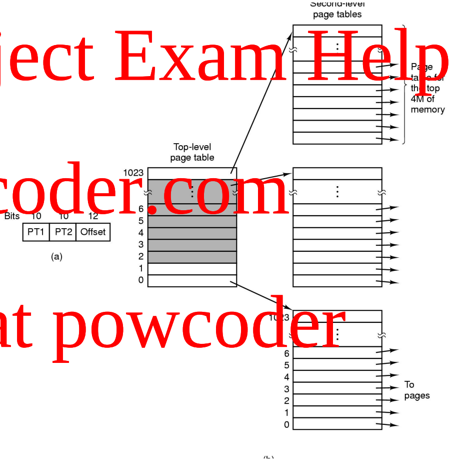


Figure: Multi-level page tables (from Tanenbaum)

Virtual Memory

Page Tables Revisited: Multi-level Page Tables

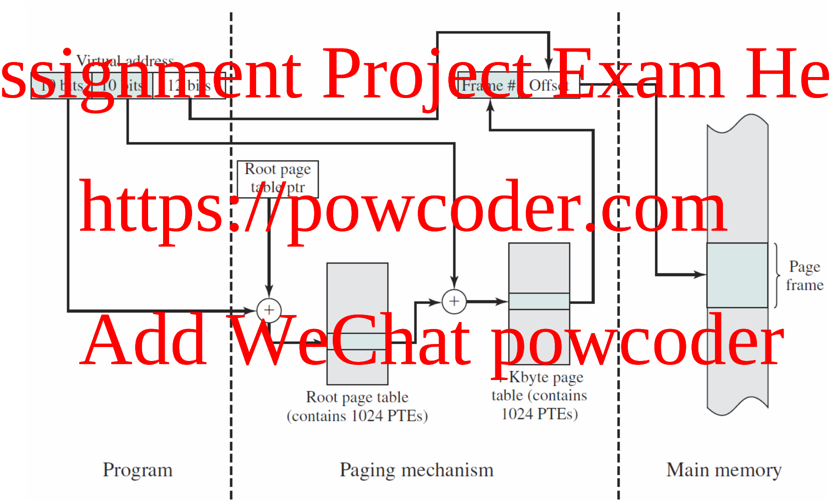


Figure: Multi-level Address Translation (from Stallings)

Virtual Memory

Page Tables Revisited: Access Speed

- **Memory organisation** of multi-level page tables:
  - The **root page table** is always maintained in memory
  - Page tables themselves are maintained in **virtual memory** due to their size
- Assume that a **fetch** from main memory takes T nano seconds
  - With a **single page table level**, access is $2 \times T$
  - With **two page table levels**, access is $3 \times T$
  - …

Paging

Address Translation: Exercises

- Given a 4KB page/frame size, and a 16-bit address space, calculate:
  - Number **M** of bits for offset within a page.
  - Number **N** of bits for representing pages. So, number of pages?
- What is the physical address for 0, 8192, 20500 using this page table?

Pages		Frames	
0	0000	0010	2
1	0001	0001	1
2	0010	0110	6
3	0011	0000	0
4	0100	0100	4
5	0101	0011	3
6	0110	X	X
7	0111	X	X
8	1000	X	X
9	1001	0101	5
10	1010	X	X
11	1011	0111	7
12	1100	X	X

Table: Page Table

`https://b.socrative.com/login/student/`

Room name: G52OSC

Summary

Take-Home Message[1]

- **Paging** splits logical and physical address spaces into small **pages/frames** to reduce internal and external fragmentation

- **Virtual memory** exploits the principle of **locality** and allows for processes to be **loaded only partially** into memory, **large logical address spaces** require “different” approaches

[1] Tanenbaum Section 3.3