


TCSS 422: OPERATING SYSTEMS

Translation Lookaside Buffer, Smaller Tables, Multi-level Page Tables



Wes J. Lloyd
School of Engineering and Technology
University of Washington - Tacoma

December 2, 2021 TCSS422: Operating Systems [Fall 2021]
School of Engineering and Technology, University of Washington Tacoma

1

OBJECTIVES – 12/2

- **Questions from 11/30**
- Assignment 2 - Dec 3
- Quiz 3 – Synchronized Array - Dec 2
- Tutorial 2 – Pthread, locks, conditions tutorial - Nov 30
- Assignment 3 (as a Tutorial) - Dec 17
- Quiz 4 - Page Tables - Dec 13
- Chapter 18: Introduction to Paging
- Chapter 19: Translation Lookaside Buffer (TLB)
 - TLB Algorithm, Hit-to-Miss Ratios
- Chapter 20: Paging: Smaller Tables
 - Smaller Tables, Multi-level Page Tables, N-level Page Tables

December 2, 2021 TCSS422: Operating Systems [Fall 2021]
School of Engineering and Technology, University of Washington - Tacoma L16.2

2

ONLINE DAILY FEEDBACK SURVEY

- Daily Feedback Quiz in Canvas – Available After Each Class
- Extra credit available for completing surveys **ON TIME**
- Tuesday surveys: due by ~ Wed @ 11:59p
- Thursday surveys: due ~ Mon @ 11:59p

TCSS 422 A > Assignments

Spring 2021

Search for Assignment

Home

Announcements

Zoom

Syllabus

Assignments

Discussions

Upcoming Assignments

TCSS 422 - Online Daily Feedback Survey - 4/1
Available until Apr 5 at 11:59pm | Due Apr 5 at 10pm | -/1 pts

Quiz 0 - C background survey

December 2, 2021 TCSS422: Computer Operating Systems [Fall 2021] L16.3
School of Engineering and Technology, University of Washington - Tacoma

3

TCSS 422 - Online Daily Feedback Survey - 4/1

Quiz Instructions

Question 1 0.5 pts

On a scale of 1 to 10, please classify your perspective on material covered in today's class:

1	2	3	4	5	6	7	8	9	10
Mostly Review To Me				Equal New and Review					Mostly New to Me

Question 2 0.5 pts

Please rate the pace of today's class:

1	2	3	4	5	6	7	8	9	10
Slow				Just Right					Fast

December 2, 2021 TCSS422: Computer Operating Systems [Fall 2021] L16.4
School of Engineering and Technology, University of Washington - Tacoma

4

MATERIAL / PACE

- Please classify your perspective on material covered in today's class (25 respondents):
 - 1-mostly review, 5-equal new/review, 10-mostly new
 - **Average - 6.02 (↓ - previous 6.29)**

- Please rate the pace of today's class:
 - 1-slow, 5-just right, 10-fast
 - **Average - 5.46 (↓ - previous 5.67)**

December 2, 2021	TCSS422: Computer Operating Systems [Fall 2021] School of Engineering and Technology, University of Washington - Tacoma	L16.5
------------------	--	-------

5

FEEDBACK

- **Requesting more time on assignment 2, if possible.**
- **This is unrelated, but I was hoping if an extension can be granted for assignment 2**
- **This is unrelated, but I was hoping if an extension can be granted for assignment 2 due to conflicts with assignments from other classes**
- **Can we get extensions on the quiz and assignment 2 due to this week? A lot of us have been busy with other classes and could use the time to turn in a sufficient enough product. Thank you!**

December 2, 2021	TCSS422: Operating Systems [Fall 2021] School of Engineering and Technology, University of Washington - Tacoma	L16.6
------------------	---	-------

6

OBJECTIVES – 12/2

- Questions from 11/30
- **Assignment 2 - Dec 3**
- Quiz 3 – Synchronized Array - Dec 2
- Tutorial 2 – Pthread, locks, conditions tutorial - Nov 30
- Assignment 3 (as a Tutorial) - Dec 17
- Quiz 4 - Page Tables - Dec 13
- Chapter 18: Introduction to Paging
- Chapter 19: Translation Lookaside Buffer (TLB)
 - TLB Algorithm, Hit-to-Miss Ratios
- Chapter 20: Paging: Smaller Tables
 - Smaller Tables, Multi-level Page Tables, N-level Page Tables

December 2, 2021	TCSS422: Operating Systems [Fall 2021] School of Engineering and Technology, University of Washington - Tacoma	L16.7
------------------	---	-------

7

OBJECTIVES – 12/2

- Questions from 11/30
- Assignment 2 - Dec 3
- **Quiz 3 – Synchronized Array - Dec 2**
- Tutorial 2 – Pthread, locks, conditions tutorial - Nov 30
- Assignment 3 (as a Tutorial) - Dec 17
- Quiz 4 - Page Tables - Dec 13
- Chapter 18: Introduction to Paging
- Chapter 19: Translation Lookaside Buffer (TLB)
 - TLB Algorithm, Hit-to-Miss Ratios
- Chapter 20: Paging: Smaller Tables
 - Smaller Tables, Multi-level Page Tables, N-level Page Tables

December 2, 2021	TCSS422: Operating Systems [Fall 2021] School of Engineering and Technology, University of Washington - Tacoma	L16.8
------------------	---	-------

8

OBJECTIVES – 12/2

- Questions from 11/30
- Assignment 2 - Dec 3
- Quiz 3 – Synchronized Array - Dec 2
- **Tutorial 2 – Pthread, locks, conditions tutorial - Nov 30**
- Assignment 3 (as a Tutorial) - Dec 17
- Quiz 4 - Page Tables - Dec 13
- Chapter 18: Introduction to Paging
- Chapter 19: Translation Lookaside Buffer (TLB)
 - TLB Algorithm, Hit-to-Miss Ratios
- Chapter 20: Paging: Smaller Tables
 - Smaller Tables, Multi-level Page Tables, N-level Page Tables

December 2, 2021	TCSS422: Operating Systems [Fall 2021] School of Engineering and Technology, University of Washington - Tacoma	L16.9
------------------	---	-------

9

OBJECTIVES – 12/2

- Questions from 11/30
- Assignment 2 - Dec 3
- Quiz 3 – Synchronized Array - Dec 2
- Tutorial 2 – Pthread, locks, conditions tutorial - Nov 30
- **Assignment 3 (as a Tutorial) - Dec 17**
- Quiz 4 - Page Tables - Dec 13
- Chapter 18: Introduction to Paging
- Chapter 19: Translation Lookaside Buffer (TLB)
 - TLB Algorithm, Hit-to-Miss Ratios
- Chapter 20: Paging: Smaller Tables
 - Smaller Tables, Multi-level Page Tables, N-level Page Tables

December 2, 2021	TCSS422: Operating Systems [Fall 2021] School of Engineering and Technology, University of Washington - Tacoma	L16.10
------------------	---	--------

10

OBJECTIVES – 12/2

- Questions from 11/30
- Assignment 2 - Dec 3
- Quiz 3 – Synchronized Array - Dec 2
- Tutorial 2 – Pthread, locks, conditions tutorial - Nov 30
- Assignment 3 (as a Tutorial) - Dec 17
- **Quiz 4 - Page Tables - Dec 13**
- Chapter 18: Introduction to Paging
- Chapter 19: Translation Lookaside Buffer (TLB)
 - TLB Algorithm, Hit-to-Miss Ratios
- Chapter 20: Paging: Smaller Tables
 - Smaller Tables, Multi-level Page Tables, N-level Page Tables

December 2, 2021	TCSS422: Operating Systems [Fall 2021] School of Engineering and Technology, University of Washington - Tacoma	L16.11
------------------	---	--------

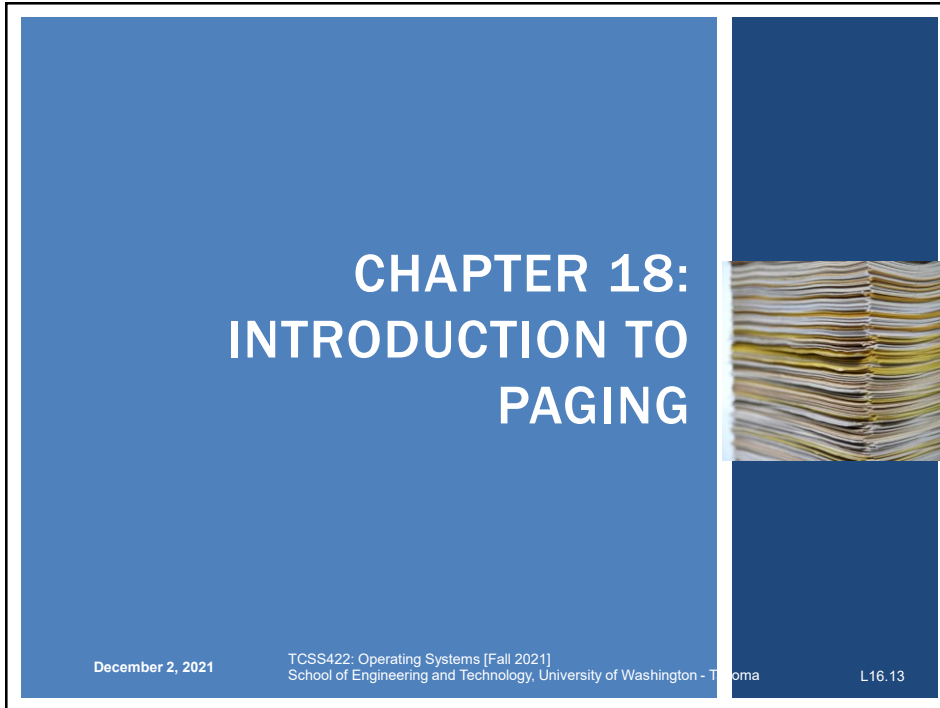
11

OBJECTIVES – 12/2

- Questions from 11/30
- Assignment 2 - Dec 3
- Quiz 3 – Synchronized Array - Dec 2
- Tutorial 2 – Pthread, locks, conditions tutorial - Nov 30
- Assignment 3 (as a Tutorial) - Dec 17
- Quiz 4 - Page Tables - Dec 13
- **Chapter 18: Introduction to Paging**
- Chapter 19: Translation Lookaside Buffer (TLB)
 - TLB Algorithm, Hit-to-Miss Ratios
- Chapter 20: Paging: Smaller Tables
 - Smaller Tables, Multi-level Page Tables, N-level Page Tables

December 2, 2021	TCSS422: Operating Systems [Fall 2021] School of Engineering and Technology, University of Washington - Tacoma	L16.12
------------------	---	--------

12

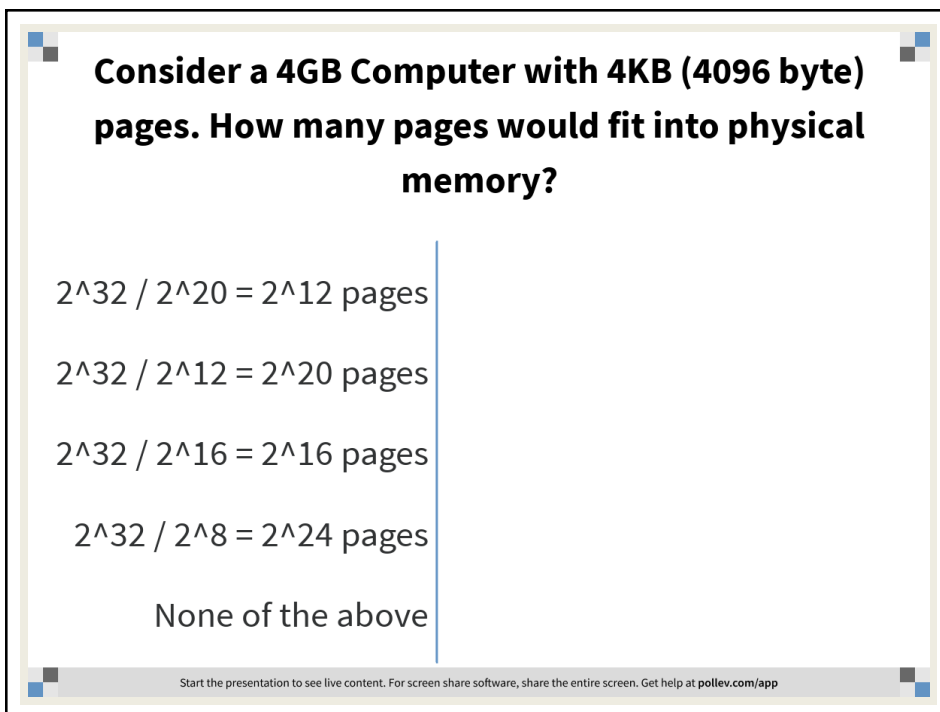


The slide features a blue background with a stack of papers on the right side. The text is centered and reads:

CHAPTER 18: INTRODUCTION TO PAGING

December 2, 2021 TCSS422: Operating Systems [Fall 2021]
School of Engineering and Technology, University of Washington - Tacoma L16.13

13



Consider a 4GB Computer with 4KB (4096 byte) pages. How many pages would fit into physical memory?

$2^{32} / 2^{20} = 2^{12}$ pages

$2^{32} / 2^{12} = 2^{20}$ pages

$2^{32} / 2^{16} = 2^{16}$ pages

$2^{32} / 2^8 = 2^{24}$ pages

None of the above

Start the presentation to see live content. For screen share software, share the entire screen. Get help at polllev.com/app

14

For the 4GB computer example, how many bits are required for the VPN?

- 24 VPN bits (indexes 2^{24} locations)
- 16 VPN bits (indexes 2^{16} locations)
- 20 VPN bits (indexes 2^{20} locations)
- 12 VPN bits (indexes 2^{12} locations)
- None of the above

December 2, 2021 TCSS422: Operating Systems (Fall 2021) L16.5

15

For the 4GB computer example, how many bits are available for page status bits?

- 32 - 12 VPN bits = 20 status bits
- 32 - 24 VPN bits = 8 status bits
- 32 - 16 VPN bits = 16 status bits
- 32 - 20 VPN bits = 12 status bits
- None of the above

Start the presentation to see live content. For screen share software, share the entire screen. Get help at polllev.com/app

16

For the 4GB computer, how much space does this page table require? (number of page table entries x size of page table entry)

2^{20} entries x 4b = 4 MB

2^{12} entries x 4b = 16 KB

2^{16} entries x 4b = 256 KB

2^{24} entries x 4b = 64 MB

None of the above

December 2, 2021 TCSS422: Operating Systems [Fall 2021] L16.7

17

For the 4GB computer, how many page tables (for user processes) would fill the entire 4GB of memory?

$4 \text{ GB} / 16 \text{ KB} = 65,536$

$4 \text{ GB} / 64 \text{ MB} = 256$

$4 \text{ GB} / 256 \text{ KB} = 16,384$

$4 \text{ GB} / 4 \text{ MB} = 1,024$

None of the above

Start the presentation to see live content. For screen share software, share the entire screen. Get help at polllev.com/app

18

PAGING SYSTEM EXAMPLE

- Consider a 4GB Computer:
 - With a 4096-byte page size (4KB)
 - How many pages would fit in physical memory?

- Now consider a page table:
 - For the page table entry, how many bits are required for the VPN?
 - If we assume the use of 4-byte (32 bit) page table entries, how many bits are available for status bits?
 - How much space does this page table require?
of page table entries x size of page table entry
 - How many page tables (for user processes) would fill the entire 4GB of memory?

December 2, 2021	TCSS422: Operating Systems [Fall 2021] School of Engineering and Technology, University of Washington - Tacoma	L16.19
------------------	---	--------

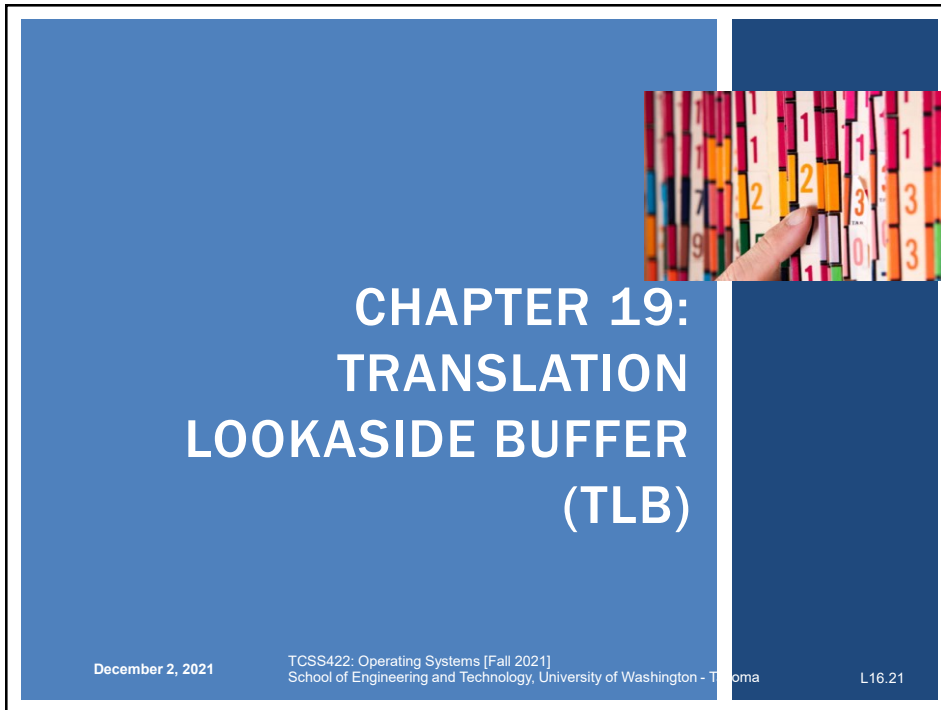
19

OBJECTIVES – 12/2

- Questions from 11/30
- Assignment 2 - Dec 3
- Quiz 3 – Synchronized Array - Dec 2
- Tutorial 2 – Pthread, locks, conditions tutorial - Nov 30
- Assignment 3 (as a Tutorial) - Dec 17
- Quiz 4 - Page Tables - Dec 13
- Chapter 18: Introduction to Paging
- **Chapter 19: Translation Lookaside Buffer (TLB)**
 - TLB Algorithm, Hit-to-Miss Ratios
- Chapter 20: Paging: Smaller Tables
 - Smaller Tables, Multi-level Page Tables, N-level Page Tables

December 2, 2021	TCSS422: Operating Systems [Fall 2021] School of Engineering and Technology, University of Washington - Tacoma	L16.20
------------------	---	--------

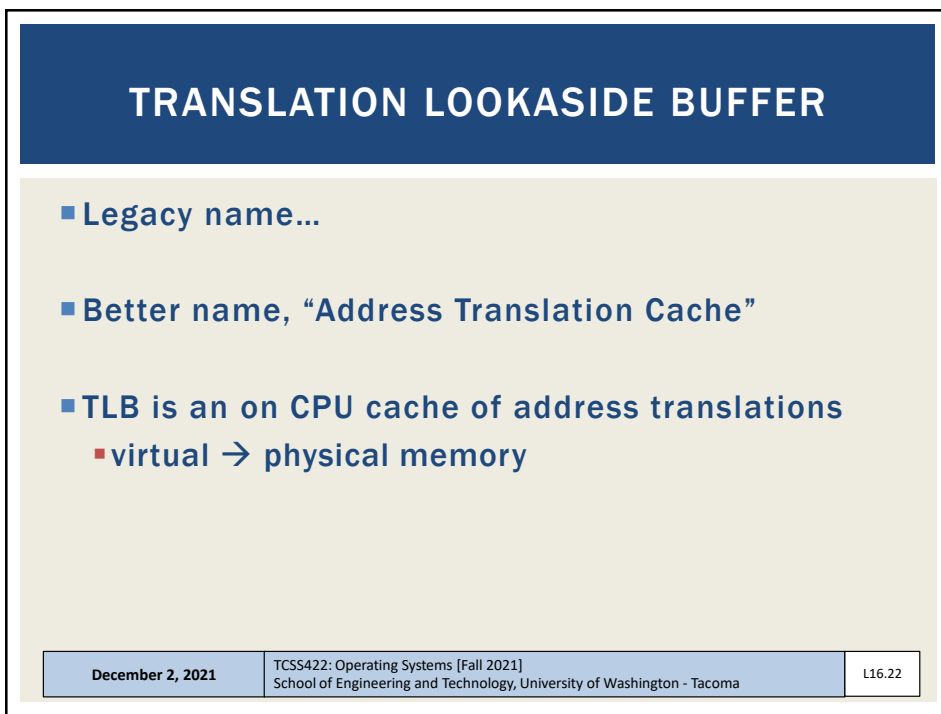
20



CHAPTER 19: TRANSLATION LOOKASIDE BUFFER (TLB)

December 2, 2021 TCSS422: Operating Systems [Fall 2021]
School of Engineering and Technology, University of Washington - Tacoma L16.21

21



TRANSLATION LOOKASIDE BUFFER

- Legacy name...
- Better name, “Address Translation Cache”
- TLB is an on CPU cache of address translations
 - virtual → physical memory

December 2, 2021 TCSS422: Operating Systems [Fall 2021]
School of Engineering and Technology, University of Washington - Tacoma L16.22

22

COUNTING MEMORY ACCESSES

- **Example: Use this Array initialization Code**

```

int array[1000];
...
for (i = 0; i < 1000; i++)
    array[i] = 0;
                    
```

- **Assembly equivalent:**

```

0x1024 movl $0x0, (%edi,%eax,4)
0x1028 incl %eax
0x102c cmpl $0x03e8,%eax
0x1030 jne 0x1024
                    
```

December 2, 2021	TCSS422: Operating Systems [Fall 2021] School of Engineering and Technology, University of Washington - Tacoma	L16.23
------------------	---	--------

23

VISUALIZING MEMORY ACCESSES: FOR THE FIRST 5 LOOP ITERATIONS

- **Locations:**
 - Page table
 - Array
 - Code

- **50 accesses for 5 loop iterations**

December 2, 2021	TCSS422: Operating Systems [Fall 2021] School of Engineering and Technology, University of Washington - Tacoma	L16.24
------------------	---	--------

24

TRANSLATION LOOKASIDE BUFFER - 2

- **Goal:**
Reduce access to the page tables
- **Example:**
50 RAM accesses for first 5 for-loop iterations
- **Move lookups** from RAM to TLB by caching page table entries

The figure consists of three vertically stacked graphs sharing a common x-axis labeled 'Memory Access' from 0 to 50.

- Top Graph (Page Table):** Shows 'Page Table(PA)' on the y-axis (1024 to 1224). It has two series: 'Page Table[39]' (squares) and 'Page Table[1]' (circles). 'Page Table[39]' has 5 accesses at indices 0, 10, 20, 30, 40. 'Page Table[1]' has 45 accesses at indices 1-49.
- Middle Graph (Array):** Shows 'Array(VA)' on the y-axis (40000 to 40100). It has one series 'mov' (squares) with 5 accesses at indices 0, 10, 20, 30, 40.
- Bottom Graph (Code):** Shows 'Code(VA)' on the y-axis (1024 to 1124). It has three series: 'mov' (squares), 'inc' (circles), 'cmp' (triangles), and 'jne' (diamonds). 'mov' has 5 accesses at indices 0, 10, 20, 30, 40. 'inc' has 10 accesses at indices 1-10. 'cmp' has 10 accesses at indices 11-20. 'jne' has 10 accesses at indices 21-30.

December 2, 2021

TCSS422: Operating Systems [Fall 2021]
School of Engineering and Technology, University of Washington - Tacoma

L16.25

25

TRANSLATION LOOKASIDE BUFFER (TLB)

- Part of the CPU's Memory Management Unit (MMU)
- Address translation cache

The diagram illustrates the flow of address translation:

- A **CPU** provides a **Logical Address**.
- The **MMU** performs a **TLB Lookup**.
- If it's a **TLB Hit**, the **MMU** outputs a **Physical Address**.
- If it's a **TLB Miss**, the **MMU** consults the **Page Table** (containing 'all v to p entries') to find the **Physical Address**.
- The **Physical Address** is used to access **Physical Memory**, which is organized into **Page 0, Page 1, Page 2, ..., Page n**.

December 2, 2021

TCSS422: Operating Systems [Fall 2021]
School of Engineering and Technology, University of Washington - Tacoma

L16.26

26

TRANSLATION LOOKASIDE BUFFER (TLB)

- Part of the CPU's Memory Management Unit (MMU)
- Address translation cache

**The TLB is an address translation cache
Different than L1, L2, L3 CPU memory caches**

Address Translation with MMU

Physical Memory

December 2, 2021	TCSS422: Operating Systems [Fall 2021] School of Engineering and Technology, University of Washington - Tacoma	L16.27
------------------	---	--------

27

OBJECTIVES – 12/2

- Questions from 11/30
- Assignment 2 - Dec 3
- Quiz 3 – Synchronized Array - Dec 2
- Tutorial 2 – Pthread, locks, conditions tutorial - Nov 30
- Assignment 3 (as a Tutorial) - Dec 17
- Quiz 4 - Page Tables - Dec 13
- Chapter 18: Introduction to Paging
- Chapter 19: Translation Lookaside Buffer (TLB)
 - **TLB Algorithm** Hit-to-Miss Ratios
- Chapter 20: Paging: Smaller Tables
 - Smaller Tables, Multi-level Page Tables, N-level Page Tables

December 2, 2021	TCSS422: Operating Systems [Fall 2021] School of Engineering and Technology, University of Washington - Tacoma	L16.28
------------------	---	--------

28

TLB BASIC ALGORITHM

- For: array based page table
- Hardware managed TLB

```
1: VPN = (VirtualAddress & VPN_MASK ) >> SHIFT
2: (Success , TlbEntry) = TLB_Lookup(VPN)
3:  if(Success == True){ // TLB Hit
4:  if(CanAccess(TlbEntry.ProtectBits) == True ){
5:      Offset = VirtualAddress & OFFSET_MASK
6:      PhysAddr = (TlbEntry.PFN << SHIFT) | Offset
7:      AccessMemory( PhysAddr )
8:  }else RaiseException(PROTECTION_ERROR)
```

Generate the physical address to access memory

December 2, 2021	TCSS422: Operating Systems [Fall 2021] School of Engineering and Technology, University of Washington - Tacoma	L16.29
------------------	---	--------

29

TLB BASIC ALGORITHM - 2

```
11:  else{ //TLB Miss
12:      PTEAddr = PTBR + (VPN * sizeof(PTE))
13:      PTE = AccessMemory(PTEAddr)
14:      (...) // Check for, and raise exceptions...
15:
16:      TLB_Insert( VPN , PTE.PFN , PTE.ProtectBits)
17:      RetryInstruction()
18:  }
19: }
```

Retry the instruction... (requery the TLB)

December 2, 2021	TCSS422: Operating Systems [Fall 2021] School of Engineering and Technology, University of Washington - Tacoma	L16.30
------------------	---	--------

30

TLB – ADDRESS TRANSLATION CACHE

- **Key detail:**
- For a TLB miss, we first access the page table in RAM to populate the TLB... we then requery the TLB
- **All address translations go through the TLB**

December 2, 2021	TCSS422: Operating Systems [Fall 2021] School of Engineering and Technology, University of Washington - Tacoma	L16.31
------------------	---	--------

31

OBJECTIVES – 12/2

- Questions from 11/30
- Assignment 2 - Dec 3
- Quiz 3 – Synchronized Array - Dec 2
- Tutorial 2 – Pthread, locks, conditions tutorial - Nov 30
- Assignment 3 (as a Tutorial) - Dec 17
- Quiz 4 - Page Tables - Dec 13
- Chapter 18: Introduction to Paging
- Chapter 19: Translation Lookaside Buffer (TLB)
 - TLB Algorithm, **Hit-to-Miss Ratios**
- Chapter 20: Paging: Smaller Tables
 - Smaller Tables, Multi-level Page Tables, N-level Page Tables

December 2, 2021	TCSS422: Operating Systems [Fall 2021] School of Engineering and Technology, University of Washington - Tacoma	L16.32
------------------	---	--------

32

OBJECTIVES – 5/25

- Questions from 5/25
- Assignment 2
- Activity – Memory Segmentation (available in Canvas)
- Tutorial 2 – Pthread, locks, conditions tutorial
- Quiz 4 - Page Tables - Dec 13
- Chapter 18: Introduction to Paging
- Chapter 19: Translation Lookaside Buffer (TLB)
 - TLB Algorithm, Hit-to-Miss Ratios
- Chapter 20: Paging: Smaller Tables
 - Smaller Tables, Multi-level Page Tables, N-level Page Tables

December 2, 2021	TCSS422: Operating Systems [Fall 2021] School of Engineering and Technology, University of Washington - Tacoma	L16.33
------------------	---	--------

33

TLB EXAMPLE

```

0:   int sum = 0 ;
1:   for( i=0; i<10; i++){
2:       sum+=a[i];
3:   }
```

- Example:
- Program address space: 256-byte
 - Addressable using 8 total bits (2^8)
 - 4 bits for the VPN (16 total pages)
- Page size: 16 bytes
 - Offset is addressable using 4-bits
- Store an array: of (10) 4-byte integers

VPN	OFFSET			
	00	04	08	12 16
VPN = 00				
VPN = 01				
VPN = 03				
VPN = 04				
VPN = 05				
VPN = 06				
VPN = 07	a[3]	a[4]	a[5]	a[6]
VPN = 08	a[7]	a[8]	a[9]	
VPN = 09				
VPN = 10				
VPN = 11				
VPN = 12				
VPN = 13				
VPN = 14				
VPN = 15				

December 2, 2021	TCSS422: Operating Systems [Fall 2021] School of Engineering and Technology, University of Washington - Tacoma	L16.34
------------------	---	--------

34

TLB EXAMPLE - 2

```

0:   int sum = 0 ;
1:   for( i=0; i<10; i++){
2:       sum+=a[i];
3:   }
    
```

- Consider the code above:
- Initially the TLB does not know where a[] is
- Consider the accesses:
- a[0], a[1], a[2], a[3], a[4], a[5], a[6], a[7], a[8], a[9]
- How many pages are accessed?
- What happens when accessing a page not in the TLB?

VPN	OFFSET			
	00	04	08	12
VPN = 00				
VPN = 01				
VPN = 03				
VPN = 04				
VPN = 05				
VPN = 06		a[0]	a[1]	a[2]
VPN = 07	a[3]	a[4]	a[5]	a[6]
VPN = 08	a[7]	a[8]	a[9]	
VPN = 09				
VPN = 10				
VPN = 11				
VPN = 12				
VPN = 13				
VPN = 14				
VPN = 15				

December 2, 2021
TCSS422: Operating Systems [Fall 2021]
School of Engineering and Technology, University of Washington - Tacoma
L16.35

35

TLB EXAMPLE - 3

```

0:   int sum = 0 ;
1:   for( i=0; i<10; i++){
2:       sum+=a[i];
3:   }
    
```

- For the accesses: a[0], a[1], a[2], a[3], a[4], a[5], a[6], a[7], a[8], a[9]
- How many are hits?
- How many are misses?
- What is the hit rate? (%)
 - 70% (3 misses one for each VP, 7 hits)

VPN	OFFSET			
	00	04	08	12
VPN = 00				
VPN = 01				
VPN = 03				
VPN = 04				
VPN = 05				
VPN = 06		a[0]	a[1]	a[2]
VPN = 07	a[3]	a[4]	a[5]	a[6]
VPN = 08	a[7]	a[8]	a[9]	
VPN = 09				
VPN = 10				
VPN = 11				
VPN = 12				
VPN = 13				
VPN = 14				
VPN = 15				

December 2, 2021
TCSS422: Operating Systems [Fall 2021]
School of Engineering and Technology, University of Washington - Tacoma
L16.36

36

TLB EXAMPLE - 4

```

0:   int sum = 0 ;
1:   for( i=0; i<10; i++){
2:       sum+=a[i];
3:   }
    
```

- **What factors affect the hit/miss rate?**
 - **Page size**
 - **Data/Access locality** (how is data accessed?)
 - **Sequential array access vs. random array access**
 - **Temporal locality**
 - **Size of the TLB cache** (how much history can you store?)

VPN	OFFSET				
	00	04	08	12	16
VPN = 00					
VPN = 01					
VPN = 03					
VPN = 04					
VPN = 05					
VPN = 06		a[0]	a[1]	a[2]	
VPN = 07	a[3]	a[4]	a[5]	a[6]	
VPN = 08	a[7]	a[8]	a[9]		
VPN = 09					
VPN = 10					
VPN = 11					
VPN = 12					
VPN = 13					
VPN = 14					
VPN = 15					

December 2, 2021
TCSS422: Operating Systems [Fall 2021]
School of Engineering and Technology, University of Washington - Tacoma
L16.37

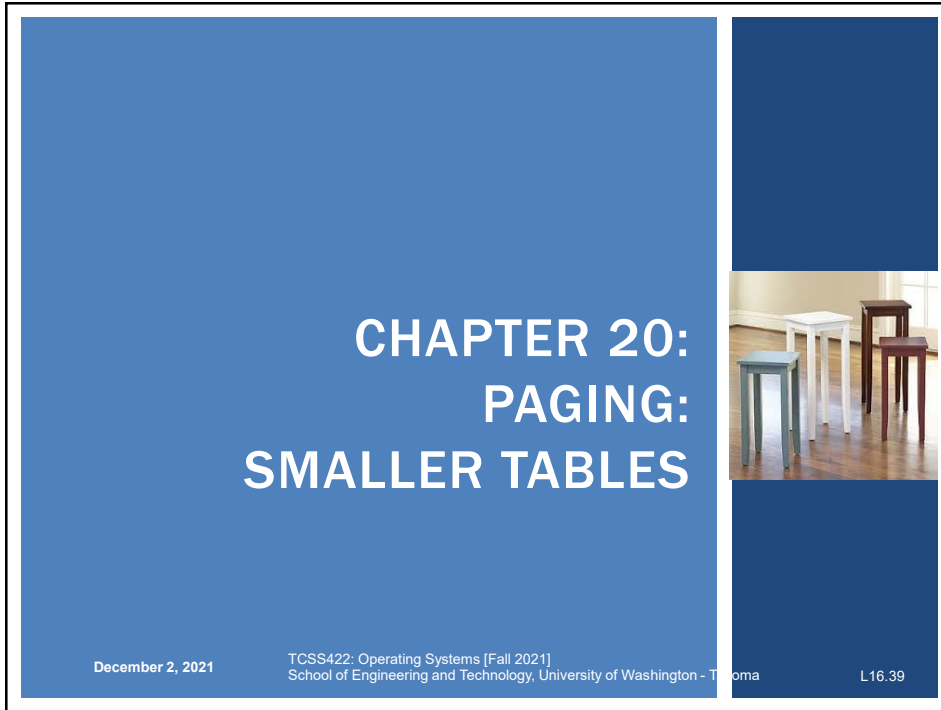
37

OBJECTIVES - 12/2

- Questions from 11/30
- Assignment 2 - Dec 3
- Quiz 3 - Synchronized Array - Dec 2
- Tutorial 2 - Pthread, locks, conditions tutorial - Nov 30
- Assignment 3 (as a Tutorial) - Dec 17
- Quiz 4 - Page Tables - Dec 13
- Chapter 18: Introduction to Paging
- Chapter 19: Translation Lookaside Buffer (TLB)
 - TLB Algorithm, Hit-to-Miss Ratios
- **Chapter 20: Paging: Smaller Tables**
 - Smaller Tables, Multi-level Page Tables, N-level Page Tables

December 2, 2021
TCSS422: Operating Systems [Fall 2021]
School of Engineering and Technology, University of Washington - Tacoma
L16.38

38



CHAPTER 20:
PAGING:
SMALLER TABLES

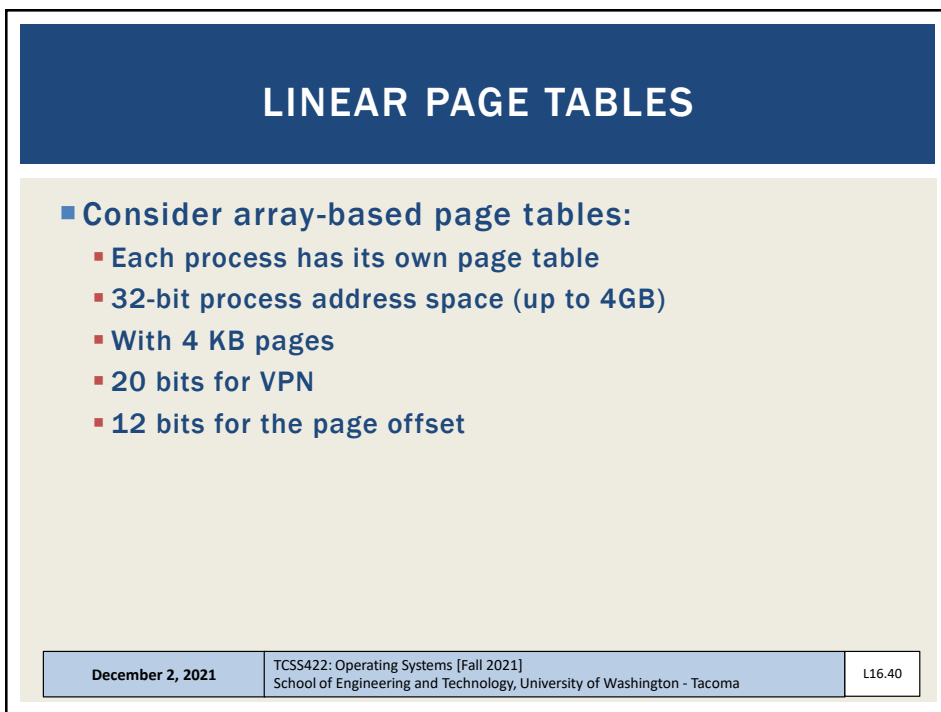
December 2, 2021

TCSS422: Operating Systems [Fall 2021]
School of Engineering and Technology, University of Washington - Tacoma

L16.39

The slide features a blue background with a white text box containing the chapter title. To the right, there is a vertical strip with a dark blue top and bottom section and a central photograph of several small, colorful stools in a room.

39



LINEAR PAGE TABLES

- Consider array-based page tables:
 - Each process has its own page table
 - 32-bit process address space (up to 4GB)
 - With 4 KB pages
 - 20 bits for VPN
 - 12 bits for the page offset

December 2, 2021

TCSS422: Operating Systems [Fall 2021]
School of Engineering and Technology, University of Washington - Tacoma

L16.40

The slide has a dark blue header with the title 'LINEAR PAGE TABLES'. The main content is on a light beige background with a bulleted list. The footer is a light blue bar with three sections: date, course information, and slide number.

40

LINEAR PAGE TABLES - 2

- Page tables stored in RAM
- Support potential storage of 2^{20} translations
= 1,048,576 pages per process @ 4 bytes/page
- Page table size 4MB / process

Page table size = $\frac{2^{32}}{2^{12}} * 4Byte = 4MByte$

- Consider 100+ OS processes
 - Requires 400+ MB of RAM to store process information

December 2, 2021	TCSS422: Operating Systems [Fall 2021] School of Engineering and Technology, University of Washington - Tacoma	L16.41
------------------	---	--------

41

LINEAR PAGE TABLES - 2

- Page tables stored in RAM
- Support potential storage of 2^{20} translations
= 1,048,576 pages per process @ 4 bytes/page
- Page table size 4MB / process

**Page tables are too big and
consume too much memory.
Need Solutions ...**

- Consider 100+ OS processes
 - Requires 400+ MB of RAM to store process information

December 2, 2021	TCSS422: Operating Systems [Fall 2021] School of Engineering and Technology, University of Washington - Tacoma	L16.42
------------------	---	--------

42

OBJECTIVES – 12/2

- Questions from 11/30
- Assignment 2 - Dec 3
- Quiz 3 – Synchronized Array - Dec 2
- Tutorial 2 – Pthread, locks, conditions tutorial - Nov 30
- Assignment 3 (as a Tutorial) - Dec 17
- Quiz 4 - Page Tables - Dec 13
- Chapter 18: Introduction to Paging
- Chapter 19: Translation Lookaside Buffer (TLB)
 - TLB Algorithm, Hit-to-Miss Ratios
- Chapter 20: Paging: Smaller Tables
 - **Smaller Tables**, Multi-level Page Tables, N-level Page Tables

December 2, 2021	TCSS422: Operating Systems [Fall 2021] School of Engineering and Technology, University of Washington - Tacoma	L16.43
------------------	---	--------

43

PAGING: USE LARGER PAGES

- **Larger pages** = 16KB = 2^{14}
- 32-bit address space: 2^{32}
- 2^{18} = 262,144 pages

$$\frac{2^{32}}{2^{14}} * 4 = 1MB \text{ per page table}$$

- Memory requirement cut to $\frac{1}{4}$
- However pages are huge
- Internal fragmentation results
- 16KB page(s) allocated for small programs with only a few variables

December 2, 2021	TCSS422: Operating Systems [Fall 2021] School of Engineering and Technology, University of Washington - Tacoma	L16.44
------------------	---	--------

44

PAGE TABLES: WASTED SPACE

■ Process: 16KB Address Space w/ 1KB pages

Page Table
Virtual Address Space

A 16KB Address Space with 1KB Pages

PFN	valid	prot	present	dirty
10	1	r-x	1	0
-	0	-	-	-
-	0	-	-	-
-	0	-	-	-
15	1	rw-	1	1
...
-	0	-	-	-
3	1	rw-	1	1
23	1	rw-	1	1

A Page Table For 16KB Address Space

December 2, 2021
TCSS422: Operating Systems [Fall 2021]
School of Engineering and Technology, University of Washington - Tacoma
L16.45

45

PAGE TABLES: WASTED SPACE

■ Process: 16KB Address Space w/ 1KB pages

Page Table
Virtual Address Space

A 16KB Address Space with 1KB Pages

PFN	valid	prot	present	dirty
0	0	-	-	-
-	0	-	-	-
-	0	-	-	-
-	0	-	-	-
15	1	rw-	1	1
...
-	0	-	-	-
3	1	rw-	1	1
23	1	rw-	1	1

A Page Table For 16KB Address Space

Most of the page table is unused and full of wasted space. (73%)

December 2, 2021
TCSS422: Operating Systems [Fall 2021]
School of Engineering and Technology, University of Washington - Tacoma
L16.46

46

**WE WILL RETURN AT
2:40PM**



December 2, 2021 TCSS422: Operating Systems [Fall 2021]
School of Engineering and Technology, University of Washington - Tacoma L16.47

47

OBJECTIVES – 12/2

- Questions from 11/30
- Assignment 2 - Dec 3
- Quiz 3 – Synchronized Array - Dec 2
- Tutorial 2 – Pthread, locks, conditions tutorial - Nov 30
- Assignment 3 (as a Tutorial) - Dec 17
- Quiz 4 - Page Tables - Dec 13
- Chapter 18: Introduction to Paging
- Chapter 19: Translation Lookaside Buffer (TLB)
 - TLB Algorithm, Hit-to-Miss Ratios
- Chapter 20: Paging: Smaller Tables
 - Smaller Tables, **Multi-level Page Tables**, N-level Page Tables

December 2, 2021 TCSS422: Operating Systems [Fall 2021]
School of Engineering and Technology, University of Washington - Tacoma L16.48

48

MULTI-LEVEL PAGE TABLES

- Consider a page table:
- 32-bit addressing, 4KB pages
- 2^{20} page table entries
- Even if memory is sparsely populated the *per process* page table requires:

$$\text{Page table size} = \frac{2^{32}}{2^{12}} * 4\text{Byte} = 4\text{MByte}$$

- Often most of the 4MB *per process* page table is empty
- Page table must be placed in 4MB contiguous block of RAM

▪ MUST SAVE MEMORY!

December 2, 2021	TCSS422: Operating Systems [Fall 2021] School of Engineering and Technology, University of Washington - Tacoma	L16.49
------------------	---	--------

49

MULTI-LEVEL PAGE TABLES - 2

- Add level of indirection, the “page directory”

Linear Page Table

PBTR [201]

valid	prot	PFN
1	rx	12
1	rx	13
0	-	-
1	rw	100
0	-	-
0	-	-
0	-	-
0	-	-
0	-	-
0	-	-
1	rw	86
1	rw	15

PFN201

Multi-level Page Table

PBTR [200]

valid	PFN
1	201
0	-
0	-
1	203

The Page Directory

valid	prot	PFN
1	rx	12
1	rx	13
0	-	-
1	rw	100

PFN201

[Page 1 of PT: Not Allocated]

valid	prot	PFN
0	-	-
0	-	-
1	rw	86
1	rw	15

PFN204

Linear (Left) And Multi-Level (Right) Page Tables

December 2, 2021	TCSS422: Operating Systems [Fall 2021] School of Engineering and Technology, University of Washington - Tacoma	L16.50
------------------	---	--------

50

MULTI-LEVEL PAGE TABLES - 2

- Add level of indirection, the “page directory”

Linear Page Table

PBTR 201

Multi-level Page Table

PBTR 200

**Two level page table:
 2²⁰ pages addressed with
 two level-indexing
 (page directory index, page table index)**

0	-	-	PFN203
0	-	-	
1	rw	86	
1	rw	15	

0	-	-	PFN204
0	-	-	
1	rw	86	
1	rw	15	

Linear (Left) And Multi-Level (Right) Page Tables

December 2, 2021
TCSS422: Operating Systems [Fall 2021]
 School of Engineering and Technology, University of Washington - Tacoma
L16.51

51

MULTI-LEVEL PAGE TABLES - 3

- Advantages
 - Only allocates page table space in proportion to the address space actually used
 - Can easily grab next free page to expand page table
- Disadvantages
 - Multi-level page tables are an example of a time-space tradeoff
 - Sacrifice address translation time (now 2-level) for space
 - Complexity: multi-level schemes are more complex

December 2, 2021
TCSS422: Operating Systems [Fall 2021]
 School of Engineering and Technology, University of Washington - Tacoma
L16.52

52

EXAMPLE

- 16KB address space, 64byte pages
- How large would a one-level page table need to be?
- 2^{14} (address space) / 2^6 (page size) = $2^8 = 256$ (pages)

0000 0000	code
0000 0001	code
...	(free)
	(free)
	heap
	heap
	(free)
	(free)
	stack
1111 1111	stack

Flag	Detail
Address space	16 KB
Page size	64 byte
Virtual address	14 bit
VPN	8 bit
Offset	6 bit
Page table entry	2^8 (256)

A 16-KB Address Space With 64-byte Pages

13	12	11	10	9	8	7	6	5	4	3	2	1	0
←-----								-----→					
								Offset					

December 2, 2021
TCSS422: Operating Systems [Fall 2021]
School of Engineering and Technology, University of Washington - Tacoma
L16.53

53

EXAMPLE - 2

- 256 total page table entries (64 bytes each)
- 1,024 bytes page table size, stored using 64-byte pages
 = $(1024/64) = 16$ page directory entries (PDEs)
- Each page directory entry (PDE) can hold 16 page table entries (PTEs) *e.g. lookups*
- 16 page directory entries (PDE) x 16 page table entries (PTE)
 = 256 total PTEs
- **Key idea: the page table is stored using pages too!**

December 2, 2021
TCSS422: Operating Systems [Fall 2021]
School of Engineering and Technology, University of Washington - Tacoma
L16.54

54

PAGE DIRECTORY INDEX

- Now, let's split the page table into two:
 - 8 bit VPN to map 256 pages
 - 4 bits for page directory index (PDI - 1st level page table)
 - 6 bits offset into 64-byte page

The diagram shows a 14-bit virtual address represented as a row of 14 boxes numbered 13 down to 0. Bits 13, 12, 11, and 10 are grouped under a bracket labeled 'Page Directory Index'. Bits 9, 8, 7, and 6 are grouped under a bracket labeled 'VPN'. Bits 5, 4, 3, 2, 1, and 0 are grouped under a bracket labeled 'Offset'. A larger bracket below the entire row is labeled '14-bits Virtual address'.

December 2, 2021
TCSS422: Operating Systems [Fall 2021]
 School of Engineering and Technology, University of Washington - Tacoma
L16.55

55

PAGE TABLE INDEX

- 4 bits page directory index (PDI - 1st level)
- 4 bits page table index (PTI - 2nd level)

The diagram shows a 14-bit virtual address represented as a row of 14 boxes numbered 13 down to 0. Bits 13, 12, 11, and 10 are grouped under a bracket labeled 'Page Directory Index'. Bits 9, 8, 7, and 6 are grouped under a bracket labeled 'Page Table Index'. Bits 5, 4, 3, 2, 1, and 0 are grouped under a bracket labeled 'Offset'. A larger bracket below the entire row is labeled '14-bits Virtual address'.

- To dereference one 64-byte memory page,
 - We need one page directory entry (PDE)
 - One page table Index (PTI) - can address 16 pages

December 2, 2021
TCSS422: Operating Systems [Fall 2021]
 School of Engineering and Technology, University of Washington - Tacoma
L16.56

56

EXAMPLE - 3

- For this example, how much space is required to store as a single-level page table with any number of PTEs?
 - 16KB address space, 64 byte pages
 - 256 page frames, 4 byte page size
 - 1,024 bytes required (*single level*)
- How much space is required for a two-level page table with only 4 page table entries (PTEs) ?
 - Page directory = 16 entries x 4 bytes (1 x 64 byte page)
 - Page table = 4 entries x 4 bytes (1 x 64 byte page)
 - 128 bytes required (2 x 64 byte pages)
 - Savings = using just 12.5% the space !!!

December 2, 2021

TCSS422: Operating Systems [Fall 2021]
School of Engineering and Technology, University of Washington - Tacoma

L16.57

57

32-BIT EXAMPLE

- Consider: 32-bit address space, 4KB pages, 2^{20} pages
- Only 4 mapped pages
- Single level: 4 MB (we've done this before)
- Two level: (old VPN was 20 bits, split in half)
 - Page directory = 2^{10} entries x 4 bytes = 1 x 4 KB page
 - Page table = 4 entries x 4 bytes (mapped to 1 4KB page)
 - 8KB (8,192 bytes) required
 - Savings = using just .78 % the space !!!
- 100 sparse processes now require < 1MB for page tables

December 2, 2021

TCSS422: Operating Systems [Fall 2021]
School of Engineering and Technology, University of Washington - Tacoma

L16.58

58

OBJECTIVES – 12/2

- Questions from 11/30
- Assignment 2 - Dec 3
- Quiz 3 – Synchronized Array - Dec 2
- Tutorial 2 – Pthread, locks, conditions tutorial - Nov 30
- Assignment 3 (as a Tutorial) - Dec 17
- Quiz 4 - Page Tables - Dec 13
- Chapter 18: Introduction to Paging
- Chapter 19: Translation Lookaside Buffer (TLB)
 - TLB Algorithm, Hit-to-Miss Ratios
- Chapter 20: Paging: Smaller Tables
 - Smaller Tables, Multi-level Page Tables, N-level Page Tables

December 2, 2021	TCSS422: Operating Systems [Fall 2021] School of Engineering and Technology, University of Washington - Tacoma	L16.59
------------------	---	--------

59

MORE THAN TWO LEVELS

- Consider: page size is $2^9 = 512$ bytes
- Page size 512 bytes / Page entry size 4 bytes
- VPN is 21 bits

Flag	Detail
Virtual address	30 bit
Page size	512 byte
VPN	21 bit
Offset	9 bit

December 2, 2021	TCSS422: Operating Systems [Fall 2021] School of Engineering and Technology, University of Washington - Tacoma	L16.60
------------------	---	--------

60

MORE THAN TWO LEVELS - 2

- Page table entries per page = $512 / 4 = 128$
- 7 bytes - for page table index (PTI)

← Page Directory Index
← Page Table Index
← offset

VPN
offset

Flag	Detail
Virtual address	30 bit
Page size	512 byte
VPN	21 bit
Offset	9 bit
Page entry per page	128 PTEs → $\log_2 128 = 7$

December 2, 2021
TCCS422: Operating Systems [Fall 2021]
School of Engineering and Technology, University of Washington - Tacoma
L16.61

61

MORE THAN TWO LEVELS - 3

- To map 1 GB address space ($2^{30}=1\text{GB RAM}$, 512-byte pages)
- $2^{14} = 16,384$ page directory entries (PDEs) are required
- When using 2^7 (128 entry) page tables...
- Page size = 512 bytes / 4 bytes per addr

← Page Directory Index
← Page Table Index
← offset

VPN
offset

Flag	Detail
Virtual address	30 bit
Page size	512 byte
VPN	21 bit
Offset	9 bit
Page entry per page	128 PTEs → $\log_2 128 = 7$

December 2, 2021
TCCS422: Operating Systems [Fall 2021]
School of Engineering and Technology, University of Washington - Tacoma
L16.62

62

MORE THAN TWO LEVELS - 3

- To map 1 GB address space ($2^{30}=1\text{GB RAM}$, 512-byte pages)
- $2^{14} = 16,384$ page directory entries (PDEs) are required
- When using 2^7 (128 entry) page tables...
- Page size = 512 bytes / 4 bytes per addr

Can't Store Page Directory with 16K pages, using 512 bytes pages. Pages only dereference 128 addresses (512 bytes / 32 bytes)

Virtual address	30 bit	
Page size	512 byte	
VPN	21 bit	
Offset	9 bit	
Page entry per page	128 PTEs	→ $\log_2 128 = 7$

December 2, 2021
TCSS422: Operating Systems [Fall 2021]
 School of Engineering and Technology, University of Washington - Tacoma
L16.63

63

MORE THAN TWO LEVELS - 3

- To map 1 GB address space ($2^{30}=1\text{GB RAM}$, 512-byte pages)
- $2^{14} = 16,384$ page directory entries (PDEs) are required
- When using 2^7 (128 entry) page tables...
- Page size = 512 bytes / 4 bytes per addr

**Need three level page table:
 Page directory 0 (PD Index 0)
 Page directory 1 (PD Index 1)
 Page Table Index**

Virtual address	30 bit	
Page size	512 byte	
VPN	21 bit	
Offset	9 bit	
Page entry per page	128 PTEs	→ $\log_2 128 = 7$

December 2, 2021
TCSS422: Operating Systems [Fall 2021]
 School of Engineering and Technology, University of Washington - Tacoma
L16.64

64

MORE THAN TWO LEVELS - 4

- We can now address 1GB with “fine grained” 512 byte pages
- Using multiple levels of indirection

The diagram shows a 32-bit Virtual Page Number (VPN) divided into four sections. From left to right: bits 30-24 are labeled 'PD Index 0' (7 bits); bits 23-16 are labeled 'PD Index 1' (7 bits); bits 15-9 are labeled 'Page Table Index' (7 bits); and bits 8-0 are a 23-bit offset. The entire 32-bit field is labeled 'VPN' at the bottom.

- Consider the implications for address translation!
- How much space is required for a virtual address space with 4 entries on a 512-byte page? (let's say 4 32-bit integers)
- PD0 1 page, PD1 1 page, PT 1 page = 1,536 bytes
- Memory Usage = $1,536 \text{ (3-level)} / 8,388,608 \text{ (1-level)} = .0183\% !!!$

December 2, 2021	TCSS422: Operating Systems [Fall 2021] School of Engineering and Technology, University of Washington - Tacoma	L16.65
------------------	---	--------

65

ADDRESS TRANSLATION CODE

```

// 5-level Linux page table address lookup
//
// Inputs:
// mm_struct - process's memory map struct
// vpage - virtual page address

// Define page struct pointers
pgd_t *pgd;
p4d_t *p4d;
pud_t *pud;
pmd_t *pmt;
pte_t *pte;
struct page *page;
    
```

December 2, 2021	TCSS422: Operating Systems [Fall 2021] School of Engineering and Technology, University of Washington - Tacoma	L16.66
------------------	---	--------

66

ADDRESS TRANSLATION - 2

```

pgd = pgd_offset(mm, vpage);
if (pgd_none(*pgd) || pgd_bad(*pgd))
    return 0;
p4d = p4d_offset(pgd, vpage);
if (p4d_none(*p4d) || p4d_bad(*p4d))
    return 0;
pud = pud_offset(p4d, vpage);
if (pud_none(*pud) || pud_bad(*pud))
    return 0;
pmd = pmd_offset(pud, vpage);
if (pmd_none(*pmd) || pmd_bad(*pmd))
    return 0;
if (!(pte = pte_offset_map(pmd, vpage)))
    return 0;
if (!(page = pte_page(*pte)))
    return 0;
physical_page_addr = page_to_phys(page);
pte_unmap(pte);
return physical_page_addr; // param to send back
    
```

pgd_offset():
 Takes a vpage address and the mm_struct for the process, returns the PGD entry that covers the requested address...


p4d/pud/pmd_offset():
 Takes a vpage address and the pgd/p4d/pud entry and returns the relevant p4d/pud/pmd.

pte_unmap()
 release temporary kernel mapping for the page table entry

December 2, 2021	TCSS422: Operating Systems [Fall 2021] School of Engineering and Technology, University of Washington - Tacoma	L16.67
------------------	---	--------

67

INVERTED PAGE TABLES



- Keep a single page table for each physical page of memory
- Consider 4GB physical memory
- Using 4KB pages, page table requires 4MB to map all of RAM
- Page table stores
 - Which process uses each page
 - Which process virtual page (from process virtual address space) maps to the physical page
- All processes share the same page table for memory mapping, kernel must isolate all use of the shared structure
- Finding process memory pages requires search of 2^{20} pages
- Hash table: can index memory and speed lookups

December 2, 2021	TCSS422: Operating Systems [Fall 2021] School of Engineering and Technology, University of Washington - Tacoma	L16.68
------------------	---	--------

68

MULTI-LEVEL PAGE TABLE EXAMPLE

- Consider a 16 MB computer which indexes memory using 4KB pages
- **(#1)** For a single level page table, how many pages are required to index memory?
- **(#2)** How many bits are required for the VPN?
- **(#3)** Assuming each page table entry (PTE) can index any byte on a 4KB page, how many offset bits are required?
- **(#4)** Assuming there are 8 status bits, how many bytes are required for each page table entry?

December 2, 2021

TCSS422: Operating Systems [Fall 2021]
School of Engineering and Technology, University of Washington - Tacoma

L16.69

69

MULTI LEVEL PAGE TABLE EXAMPLE - 2

- **(#5)** How many bytes (or KB) are required for a single level page table?
- Let's assume a simple HelloWorld.c program.
- HelloWorld.c requires virtual address translation for 4 pages:
 - 1 - code page 1 - stack page
 - 1 - heap page 1 - data segment page
- **(#6)** Assuming a two-level page table scheme, how many bits are required for the Page Directory Index (PDI)?
- **(#7)** How many bits are required for the Page Table Index (PTI)?

December 2, 2021

TCSS422: Operating Systems [Fall 2021]
School of Engineering and Technology, University of Washington - Tacoma

L16.70

70

MULTI LEVEL PAGE TABLE EXAMPLE - 3

- Assume each page directory entry (PDE) and page table entry (PTE) requires 4 bytes:
 - 6 bits for the Page Directory Index (PDI)
 - 6 bits for the Page Table Index (PTI)
 - 12 offset bits
 - 8 status bits
- (#8)** How much **total** memory is required to index the HelloWorld.c program using a two-level page table when we only need to translate 4 total pages?
- HINT:** we need to allocate one Page Directory and one Page Table...
- HINT:** how many entries are in the PD and PT

December 2, 2021

TCSS422: Operating Systems [Fall 2021]
School of Engineering and Technology, University of Washington - Tacoma

L16.71

71

MULTI LEVEL PAGE TABLE EXAMPLE - 4

- (#9)** Using a single page directory entry (PDE) pointing to a single page table (PT), if all of the slots of the page table (PT) are in use, what is the total amount of memory a two-level page table scheme can address?
- (#10)** And finally, for this example, as a percentage (%), how much memory does the 2-level page table scheme consume compared to the 1-level scheme?
- HINT:** two-level memory use / one-level memory use

December 2, 2021

TCSS422: Operating Systems [Fall 2021]
School of Engineering and Technology, University of Washington - Tacoma

L16.72

72


ANSWERS

- #1 - 4096 pages
- #2 - 12 bits
- #3 - 12 bits
- #4 - 4 bytes
- #5 - $4096 \times 4 = 16,384$ bytes (16KB)
- #6 - 6 bits
- #7 - 6 bits
- #8 - 256 bytes for Page Directory (PD) (64 entries x 4 bytes)
256 bytes for Page Table (PT) **TOTAL = 512 bytes**
- #9 - 64 entries, where each entry maps a 4,096 byte page
With 12 offset bits, can address 262,144 bytes (256 KB)
- #10- $512/16384 = .03125 \rightarrow 3.125\%$

December 2, 2021	TCSS422: Operating Systems [Fall 2021] School of Engineering and Technology, University of Washington - Tacoma	L16.73
------------------	---	--------

73

QUESTIONS



74