


# TCCS 422: OPERATING SYSTEMS

## I/O Devices



Wes J. Lloyd  
Institute of Technology  
University of Washington - Tacoma

March 7, 2018 TCCS422: Operating Systems [Winter 2018]  
Institute of Technology, University of Washington - Tacoma

# OBJECTIVES

- Homework 3 Questions
- Quiz 5 Posted
- Feedback from 3/5
- Ch. 36
  - I/O Devices

March 7, 2018 TCCS422: Operating Systems [Winter 2018]  
Institute of Technology, University of Washington - Tacoma L16.2

# FEEDBACK FROM 3/5

- Could we go over one more example of a multi-level page table?
- It would be helpful to explicitly define what the input parameters are, what we can derive based on those, and what an expected answer would be.

March 7, 2018 TCCS422: Operating Systems [Winter 2018]  
Institute of Technology, University of Washington - Tacoma L16.3

# 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?

March 7, 2018 TCCS422: Operating Systems [Winter 2018]  
Institute of Technology, University of Washington - Tacoma L16.4

# 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)?

March 7, 2018 TCCS422: Operating Systems [Winter 2018]  
Institute of Technology, University of Washington - Tacoma L16.5

# 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

March 7, 2018 TCCS422: Operating Systems [Winter 2018]  
Institute of Technology, University of Washington - Tacoma L16.6

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

March 7, 2018

TCCS422: Operating Systems [Winter 2018]  
Institute of Technology, University of Washington - Tacoma

L16.7

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-  $512 / 16384 = .03125 \rightarrow 3.125\%$

March 7, 2018

TCCS422: Operating Systems [Winter 2018]  
Institute of Technology, University of Washington - Tacoma

L16.8

GRADING QUESTIONS


- For grading questions related to the programming assignments please first reach out to your grader, and then follow-up with any issues, questions
- Matthew Subido
  - Message on Canvas
- Ibrahim Diabate
  - message on Canvas

March 7, 2018

TCCS422: Operating Systems [Winter 2018]  
Institute of Technology, University of Washington - Tacoma

L16.9

CHAPTER 36:  
I/O DEVICES



March 7, 2018

TCCS422: Operating Systems [Winter 2018]  
Institute of Technology, University of Washington - Tacoma

L16.10

OBJECTIVES

- Chapter 36
  - Polling vs Interrupts
  - Programmed I/O (PIO)
  - Direct memory Access (DMA)
  - Port-mapped I/O (PMIO)
  - Memory-mapped I/O (MMIO)

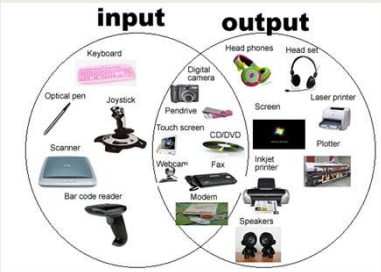
March 7, 2018

TCCS422: Operating Systems [Winter 2018]  
Institute of Technology, University of Washington - Tacoma

L16.11

I/O DEVICES

- Modern computer systems interact with a variety of devices



March 7, 2018

TCCS422: Operating Systems [Winter 2018]  
Institute of Technology, University of Washington - Tacoma

L16.12

COMPUTER SYSTEM ARCHITECTURE

Prototypical System Architecture

VERY FAST: CPU is attached to main memory via a Memory bus.  
FAST: High speed devices (e.g. video) are connected via a General I/O bus.  
SLOWER: Disks are connected via a Peripheral I/O bus.

March 7, 2018

TCS5422: Operating Systems [Winter 2018]  
Institute of Technology, University of Washington - Tacoma

L16.13

I/O BUSES

- Buses
  - Buses closer to the CPU are faster
  - Can support fewer devices
  - Further buses are slower, but support more devices
- Physics and costs dictate “levels”
  - Memory bus
  - General I/O bus
  - Peripheral I/O bus
- Tradeoff space: speed vs. locality

March 7, 2018

TCS5422: Operating Systems [Winter 2018]  
Institute of Technology, University of Washington - Tacoma

L16.14

CANONICAL DEVICE

- Consider an arbitrary canonical device

Registers: Status Command Data

Micro-controller(CPU)  
Memory (DRAM or SRAM or both)  
Other Hardware-specific Chips

interface  
internals

Canonical Device

- Two primary components
  - Interface (registers for communication)
  - Internals: Local CPU, memory, specific chips, firmware (embedded software)

March 7, 2018

TCS5422: Operating Systems [Winter 2018]  
Institute of Technology, University of Washington - Tacoma

L16.15

CANONICAL DEVICE:  
HARDWARE INTERFACE

- Status register
  - Maintains current device status
- Command register
  - Where commands for interaction are sent
- Data register
  - Used to send and receive data to the device

General concept:  
The OS interacts and controls device behavior  
by reading and writing the device registers.

March 7, 2018

TCS5422: Operating Systems [Winter 2018]  
Institute of Technology, University of Washington - Tacoma

L16.16

OS DEVICE INTERACTION

- Common example of device interaction

```
while ( STATUS == BUSY)  ← Poll-Is device available?  
; //wait until device is not busy  
write data to data register  ← Command parameterization  
write command to command register  ← Send command  
Doing so starts the device and executes the command  
while ( STATUS == BUSY)  ← Poll – Is device done?  
; //wait until device is done with your request
```

March 7, 2018

TCS5422: Operating Systems [Winter 2018]  
Institute of Technology, University of Washington - Tacoma

L16.17

POLLING

- OS checks if device is **READY** by repeatedly checking the **STATUS** register
  - Simple approach
  - CPU cycles are wasted without doing meaningful work
  - Ok if only a few cycles, for rapid devices that are often **READY**
  - BUT polling, as with “spin locks” we understand is inefficient

CPU utilization by polling

March 7, 2018

TCS5422: Operating Systems [Winter 2018]  
Institute of Technology, University of Washington - Tacoma

L16.18

INTERRUPTS VS POLLING

- For longer waits, put process waiting on I/O to sleep
- Context switch (C/S) to another process
- When I/O completes, fire an interrupt to initiate C/S back
  - Advantage: better multi-tasking and CPU utilization
  - Avoids: unproductive CPU cycles (polling)

Diagram of CPU utilization by interrupt

March 7, 2018

TCCS422: Operating Systems [Winter 2018]  
Institute of Technology, University of Washington - Tacoma

L16.19

INTERRUPTS VS POLLING - 2

What is the tradeoff space ?

- Interrupts are not always the best solution
  - How long does the device I/O require?
  - What is the cost of context switching?

If device I/O is fast → polling is better.

If device I/O is slow → interrupts are better.

March 7, 2018

TCCS422: Operating Systems [Winter 2018]  
Institute of Technology, University of Washington - Tacoma

L16.20

INTERRUPTS VS POLLING - 3

- One solution is a two-phase hybrid approach
  - Initially poll, then sleep and use interrupts
- Livelock problem
  - Common with network I/O
  - Many arriving packets generate *many many* interrupts
  - Overloads the CPU!
  - No time to execute code, just interrupt handlers !
- Livelock optimization
  - Coalesce multiple arriving packets (for different processes) into fewer interrupts
  - Must consider number of interrupts a device could generate

March 7, 2018

TCCS422: Operating Systems [Winter 2018]  
Institute of Technology, University of Washington - Tacoma

L16.21

DEVICE I/O

- To interact with a device we must send/receive DATA
- There are two general approaches:
  - Programmed I/O (PIO)
  - Direct memory access (DMA)

March 7, 2018

TCCS422: Operating Systems [Winter 2018]  
Institute of Technology, University of Washington - Tacoma

L16.22

Transfer Modes			
Mode	#	Maximum transfer rate (MB/s)	cycle time
PIO	0	3.3	600 ns
	1	5.2	383 ns
	2	8.3	240 ns
	3	11.1	180 ns
	4	16.7	120 ns
Single-word DMA	0	2.1	960 ns
	1	4.2	480 ns
	2	8.3	240 ns
Multi-word DMA	0	4.2	480 ns
	1	13.3	150 ns
	2	16.7	120 ns
	3 <sup>[34]</sup>	20	100 ns
	4 <sup>[34]</sup>	25	80 ns
Ultra DMA	0	16.7	240 ns + 2
	1	25.0	160 ns + 2
	2 (Ultra ATA/33)	33.3	120 ns + 2
	3	44.4	90 ns + 2
	4 (Ultra ATA/66)	66.7	60 ns + 2
	5 (Ultra ATA/100)	100	40 ns + 2
	6 (Ultra ATA/133)	133	30 ns + 2
	7 (Ultra ATA/167) <sup>[35]</sup>	167	24 ns + 2

From [https://en.wikipedia.org/wiki/Parallel\\_ATA](https://en.wikipedia.org/wiki/Parallel_ATA)

PROGRAMMED I/O (PIO)

- Spend CPU time to perform I/O
- CPU is involved with the data movement (input/output)
- PIO is slow – CPU is occupied with meaningless work

Diagram of CPU utilization

March 7, 2018

TCCS422: Operating Systems [Winter 2018]  
Institute of Technology, University of Washington - Tacoma

L16.24

PPIO DEVICES

- Legacy serial ports
- Legacy parallel ports
- PS/2 keyboard and mouse
- Legacy MIDI, joysticks
- Old network interfaces

March 7, 2018

TCCS422: Operating Systems [Winter 2018]  
Institute of Technology, University of Washington - Tacoma

L16.25

DIRECT MEMORY ACCESS (DMA)

- Copy data in memory by offloading to a “DMA controller”
- Many devices (including CPUs) have DMA controllers
- Give DMA memory address, size, and copy instruction
- DMA performs I/O independent of the CPU

The diagram illustrates CPU utilization by DMA. It shows three horizontal bars representing the CPU, DMA, and Disk. The CPU bar has 12 segments: 10 are labeled '1' (task 1) and 2 are labeled '2' (task 2). The DMA bar has 3 segments labeled 'C' (copy data from memory). The Disk bar has 5 segments labeled '1' (task 1). A legend indicates that '1' represents task 1, '2' represents task 2, and 'C' represents copy data from memory.

March 7, 2018

TCCS422: Operating Systems [Winter 2018]  
Institute of Technology, University of Washington - Tacoma

L16.26

DEVICE INTERACTION

- Two primary methods
- Port mapped I/O (PMIO)
- Memory mapped I/O (MMIO)

March 7, 2018

TCCS422: Operating Systems [Winter 2018]  
Institute of Technology, University of Washington - Tacoma

L16.27

PORT MAPPED I/O (PMIO)

- Device specific CPU I/O Instructions
- Follows a CISC model: extra instructions
- x86-x86-64: `in` and `out` instructions
- `outb`, `outw`, `outl`
- 1, 2, 4 byte copy from EAX → device’s I/O port

March 7, 2018

TCCS422: Operating Systems [Winter 2018]  
Institute of Technology, University of Washington - Tacoma

L16.28

MEMORY MAPPED I/O (MMIO)

- Device’s memory is mapped to CPU memory
- Tenet of RISC CPUs: instructions are eliminated, CPU is simpler
- Old days: 16-bit CPUs didn’t have a lot of spare memory space
- Today’s CPUs: 32-bit (4GB addr space) & 64-bit (128 TB addr space)
- Regular CPU instructions used to access device: mapped to memory
- Devices monitor CPU address bus and respond to their addresses
- I/O device address areas of memory are **reserved** for I/O
  - Must not be available for normal memory operations.

March 7, 2018

TCCS422: Operating Systems [Winter 2018]  
Institute of Technology, University of Washington - Tacoma

L16.29

DEVICE INTERACTION

- The OS must interact with a variety of devices
- Example: for DISK I/O consider the variety of disks:
- SCSI, IDE, USB flash drive, DVD, etc.
- Device drivers use abstraction to provide general interfaces for vendor specific hardware
- In Linux: block devices

March 7, 2018

TCCS422: Operating Systems [Winter 2018]  
Institute of Technology, University of Washington - Tacoma

L16.30

FILE SYSTEM ABSTRACTION

- Layers of I/O abstraction in Linux
- C functions (open, read, write) issue **block read and write** requests to the generic block layer

The diagram illustrates the File System Stack, showing the flow of data from the user space to the kernel space. The stack consists of the following layers (from top to bottom):

- Application** (user space)
- POSIX API** (open, read, write, close, etc.)
- File System** (kernel space)
- Generic Block Interface** (block read/write)
- Generic Block Layer**
- Specific Block Interface** (protocol-specific read/write)
- Device Driver** (SCSI, ATA, etc.)

The stack is labeled "The File System Stack" at the bottom.

March 7, 2018

TCCS422: Operating Systems [Winter 2018]  
Institute of Technology, University of Washington - Tacoma

L16.31

FILE SYSTEM ABSTRACTION ISSUES

- Too much abstraction**
  - Many devices provide special capabilities
  - Example: SCSI Error handling
  - SCSI devices provide extra detail which are lost to the OS
- Buggy device drivers**
  - 70% of OS code is in device drivers
  - Device drivers are required for every device plugged in
  - Drivers are often 3<sup>rd</sup> party, which is not quality controlled at the same level as the OS (Linux, Windows, MacOS, etc.)

March 7, 2018

TCCS422: Operating Systems [Winter 2018]  
Institute of Technology, University of Washington - Tacoma

L16.32

QUESTIONS