Computer Systems and Networks

ECPE 170 – Jeff Shafer – University of the Pacific

Design of a Simple Computer

Schedule

- **Today** – Basic computer / memory organization
- **Thursday** – Introduce new machine architecture – MARIE – and assembly programming language
- **Next Tuesday** – Continue with MARIE intro + exam review
- **Next Thursday – Exam 1**
  - Thursday, Sept 29th
  - Topics:
    - Chapter 2 (Data representations)
    - Chapter 3 (Digital logic)
    - Part of Chapter 4 (only up through today's material – nothing on MARIE)

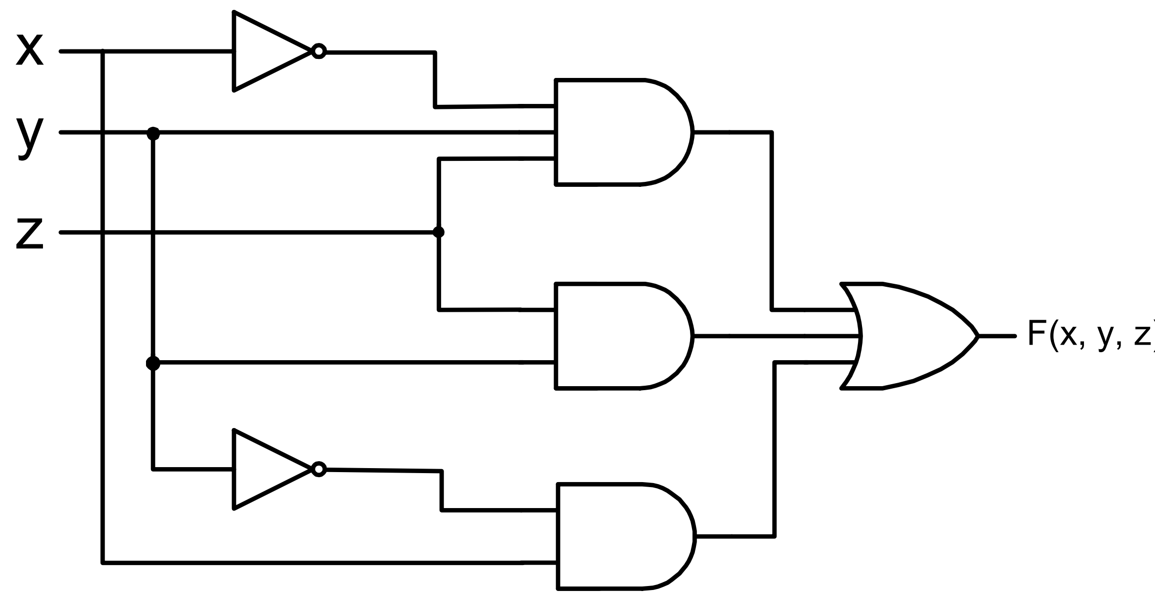
A Note on Grading…

- Grades for homework are posted in Sakai…
  - Goal is to return graded assignments within 1 week
- Partial credit?
  - **If you don't show any work…**
    and the answer is wrong…
    I can't give you any partial credit!

Homework 5 – 3.30

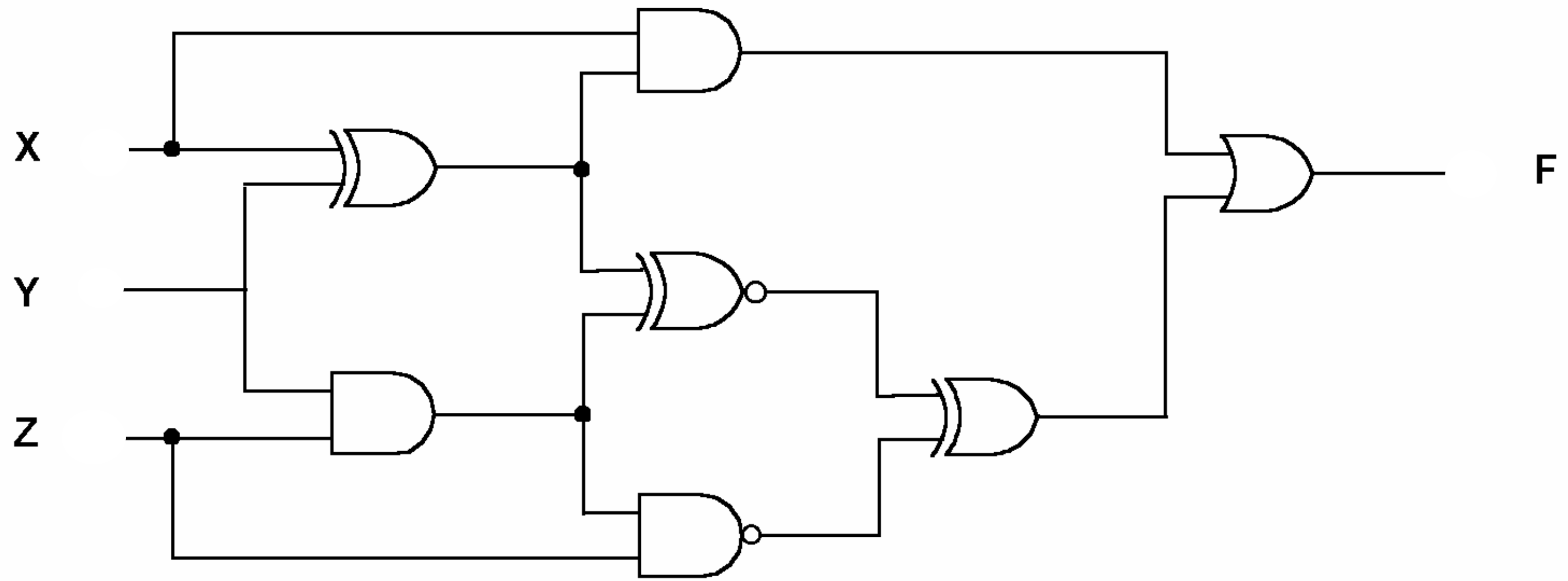
- **Draw the combinational circuit that implements**

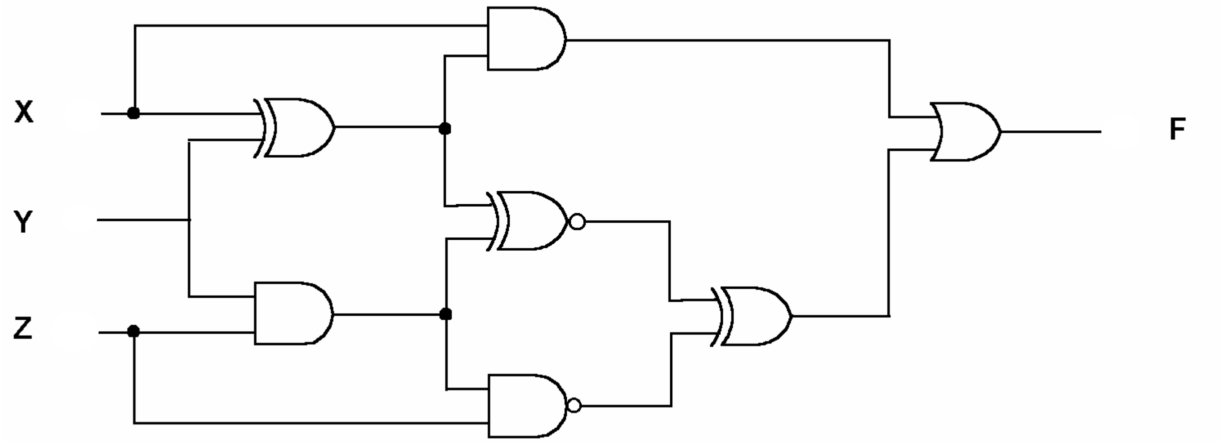
$$F(x, y, z) = \overline{x}yz + yz + x\overline{y}$$



Homework 5 – 3.34

- **Find the truth table that describes the circuit:**




X
Y
Z
F

Measures of Capacity and Speed

- Kilo- (K) = 1 thousand = 10^{3} and 2^{10}
- Mega- (M) = 1 million = 10^{6} and 2^{20}
- Giga- (G) = 1 billion = 10^{9} and 2^{30}
- Tera- (T) = 1 trillion = 10^{12} and 2^{40}
- Peta- (P) = 1 quadrillion = 10^{15} and 2^{50}
- Exa- (E) = 1 quintillion = 10^{18} and 2^{60}
- Zetta- (Z) = 1 sextillion = 10^{21} and 2^{70}
- Yotta- (Y) = 1 septillion = 10^{24} and 2^{80}

Whether a metric refers to a power of ten or a power of two typically depends upon what is being measured.

Introduction

- **Chapter 4 in textbook**
- Course to date
  - Chapter 2 – **Representing** numbers/letters in a computer-friendly format
  - Chapter 3 - Creating **digital circuits** that implement Boolean functions and store data
- Next goal
  - Combine these basic components to build a simple (but functional) computer
  - Program that computer (in **assembly language**)

CPU Basics

- Steps to run a program?
  - **Fetch** instruction from memory
  - **Decode** instruction to determine operation
  - **Execute** instruction
- Many components are needed to accomplish this

CPU Basics

- Two main components: datapath and control unit
- **Datapath**
  - Arithmetic-logic unit
  - Storage units (registers)
  - Connected by a data bus that also reaches main memory
- **Control Unit**
  - Responsible for sequencing operations
    - What does hardware do first?
    - What does hardware do second?
    - What does the ALU do?

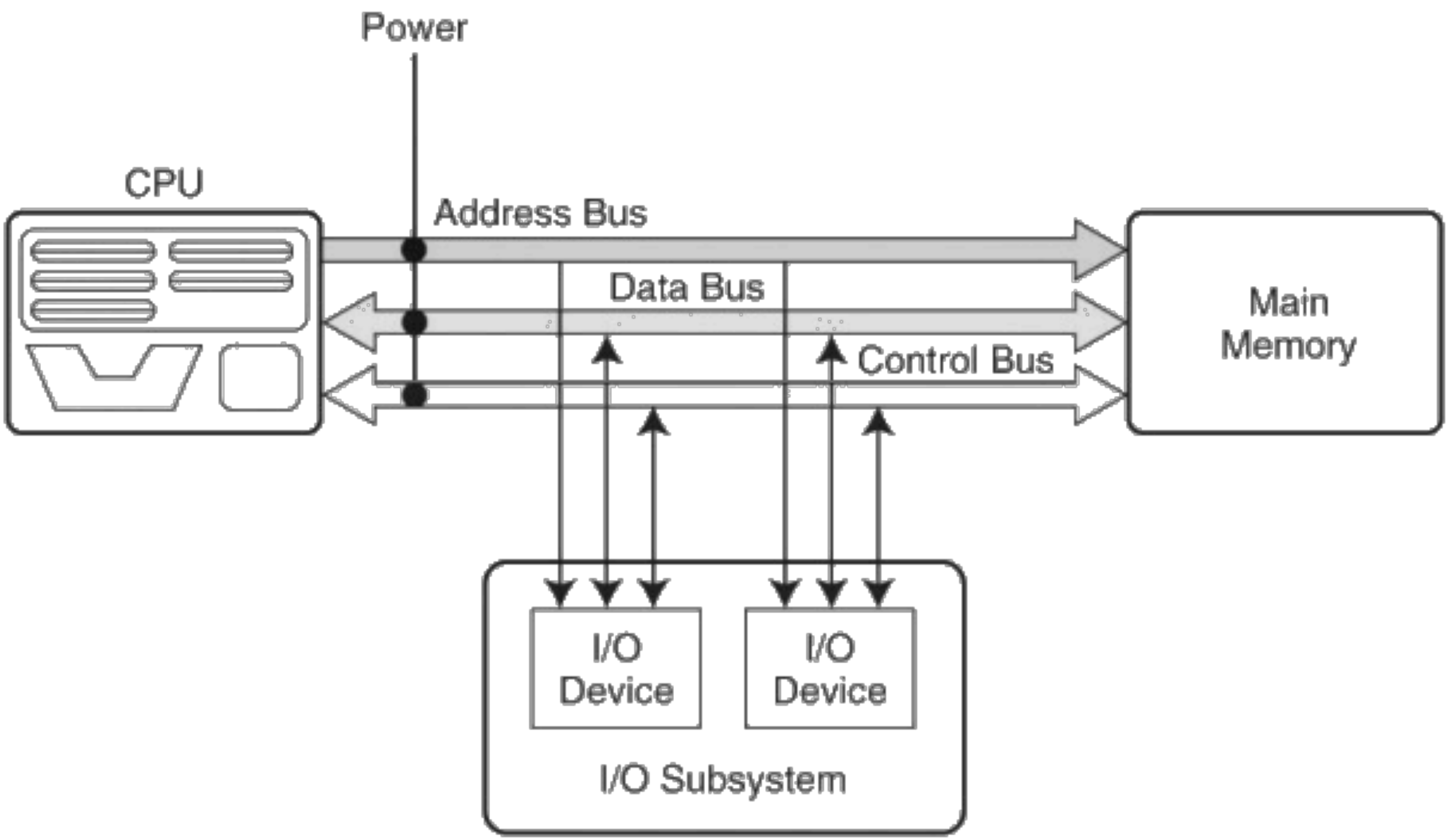
Registers

- Registers hold data that can be readily accessed by the CPU
  - **Much faster** than main memory
- Implemented using D flip-flops
  - A 32-bit register requires 32 D flip-flops

Data Bus

- Data bus moves data between CPU components
  - A bus is a **set of wires** (8, 16, 32, 64, …)
  - One bit per wire per clock cycle
- Bus components
  - **Data lines**
    - Move data
  - **Address lines**
    - Determine location of data (either source or destination)
  - **Control lines**
    - Determine direction of data flow

Example Bus


Power
CPU
Address Bus
Data Bus
Control Bus
Main Memory
I/O Device
I/O Device
I/O Subsystem

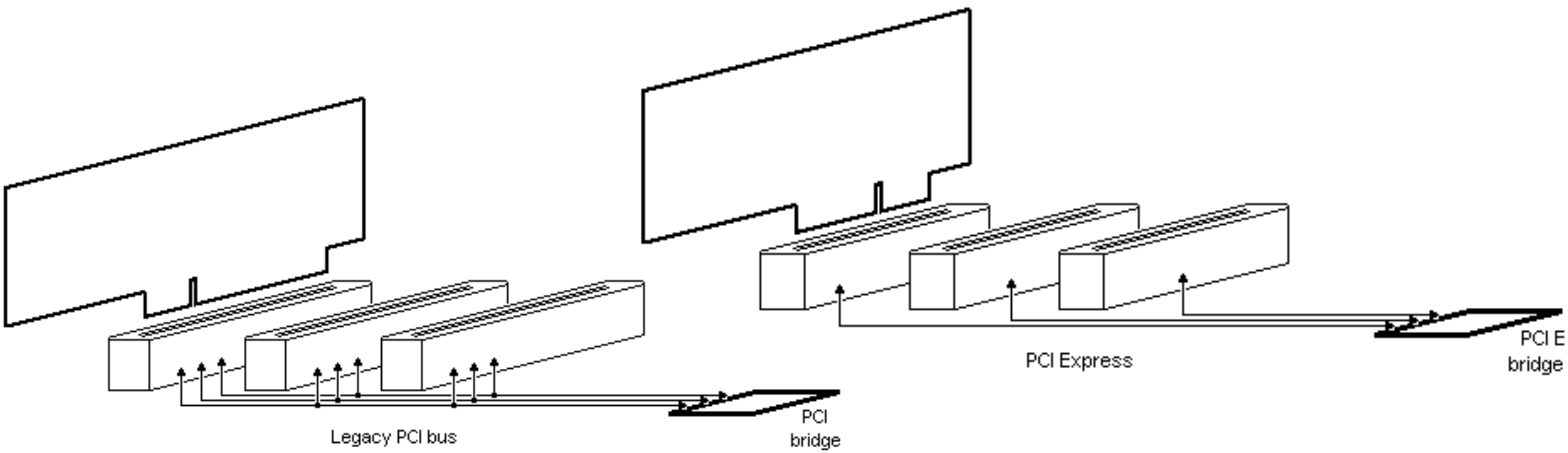
Point-to-Point vs Multipoint

Multipoint Bus

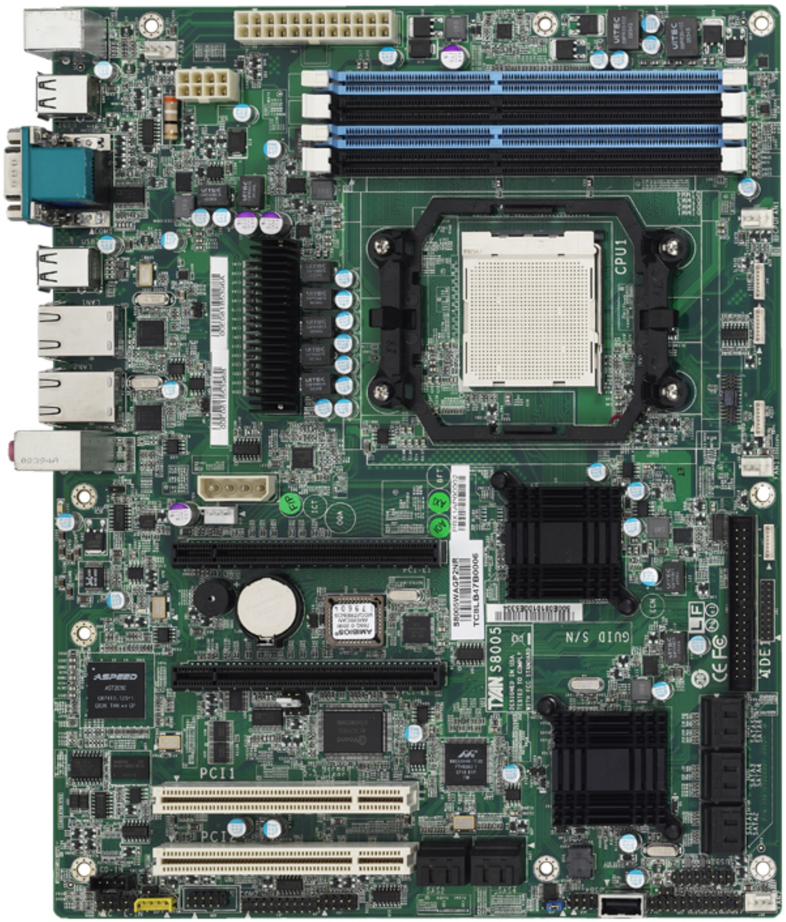
- Connect two components via **shared** wires
- Example: PCI bus

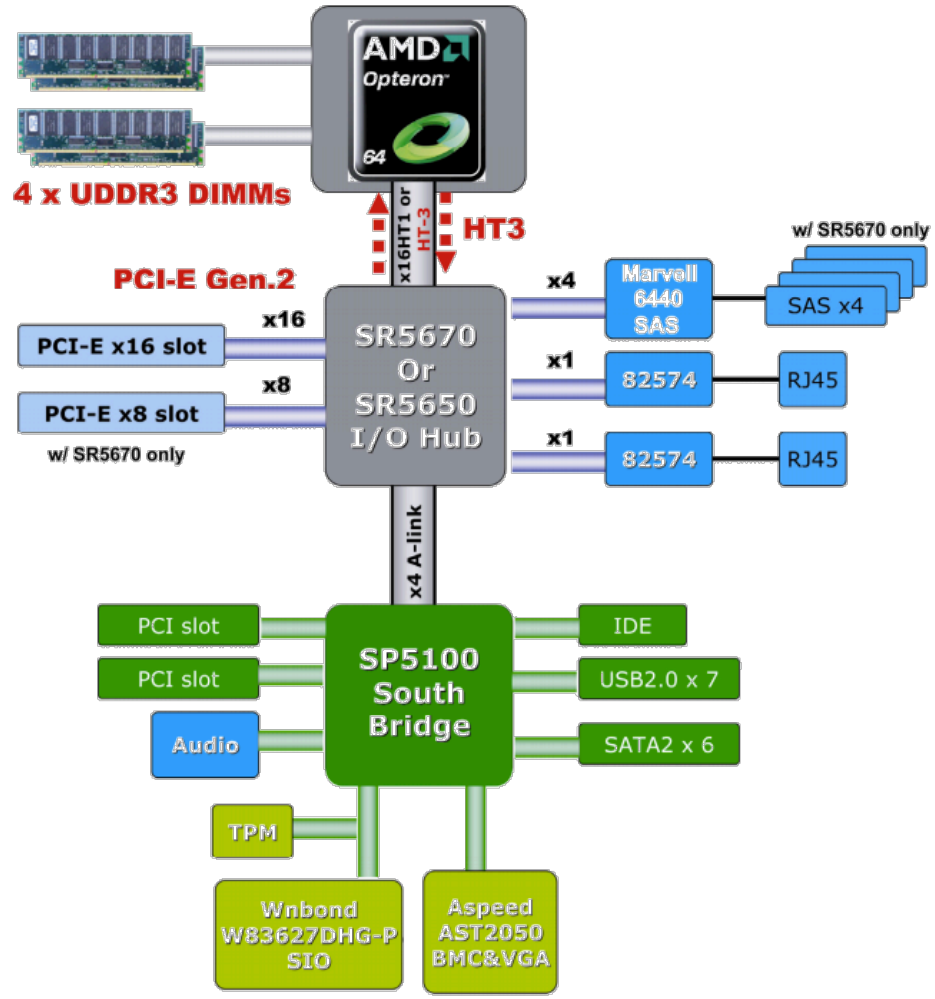
Point-to-Point Bus

- Connect multiple components via **dedicated** wires
- Example: PCI-e bus



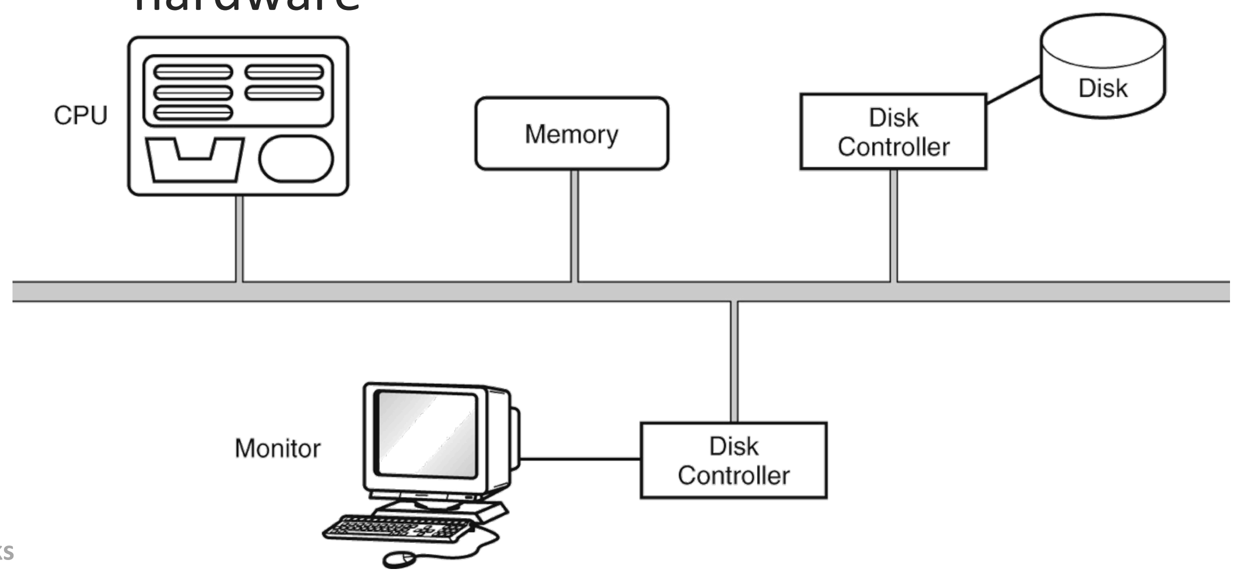
Modern AMD Opteron System





Multipoint Bus

- Multipoint bus is a shared resource
  - When can I access this shared resource?
  - What can others access it?
  - Controlled through protocols implemented in hardware



Clocks

- Computer components must be carefully synchronized
  - Use a **clock** (think of a "drummer", rather than a "watch")
- Fixed number of clock cycles are required to carry out each data movement or computational operation
- Clock frequency determines the speed with which all operations are carried out.
  - Measured in megahertz or gigahertz
  - Clock cycle time is the reciprocal of clock frequency
    - An 800 MHz clock has a cycle time of 1.25 ns.

Clocks

- **Clock speed does not (*directly*) equal CPU performance!**
- CPU time required to run a program:

$$\text{CPU Time} = \frac{\text{seconds}}{\text{program}} = \frac{\text{instructions}}{\text{program}} * \frac{\text{avg.cycles}}{\text{instruction}} * \frac{\text{seconds}}{\text{cycle}}$$

- How can we decrease CPU time? Many ways!
  - Reduce the number of instructions in a program
  - Reduce the number of cycles per instruction
  - Reduce the number of nanoseconds per clock cycle

The Input/Output Subsystem

- A computer communicates with the outside world through its input/output (I/O) subsystem
- Two different ways I/O devices can function
  - **Memory-mapped**: the I/O device behaves like main memory from the CPU's point of view.
  - **Instruction-based**: the CPU has a specialized I/O instruction set
- Modern devices are typically memory-mapped
  - But CPUs still have legacy I/O instructions…

Memory Organization

- Imagine computer memory as a linear array of addressable storage cells (i.e. an array of registers)
- Addressability
  - What is the smallest amount of memory I can access?
  - Byte-address or word-addressable
    - A word might be 2, 4, or 8 bytes…
- Memory is constructed from RAM chips
  - Each chip referred to in terms of length × width
  - Example: if the memory word size of the machine is 16 bits, then a 4M × 16 RAM chip gives us 4×2^{20} memory locations, each of which is 16 bits wide

Memory Organization

- How does the computer access a memory location corresponds to a particular address?
- We observe that 4M can be expressed as $2^2 \times 2^{20} = 2^{22}$ words
- The memory locations for this memory are numbered 0 through $2^{22}-1$.
- Thus, the memory bus of this system requires *at least* 22 address lines
    - Address lines "count" from 0 to $2^{22} - 1$ in binary

Memory Organization

- How does the number of addresses relate to the memory width?
  - If memory is **word-addressable**, then the number of locations directly gives the number of address lines
  - If memory is **byte-addressable**, we need **extra address lines** to specify which byte within each word

Memory Organization

- Typically multiple RAM chips are used
  - Access is more efficient when memory is organized into banks of chips with the addresses **interleaved** across the chips
- Low-order interleaving
  - Low order bits of the address specify which memory bank contains the address of interest
- High-order interleaving
  - High order address bits specify the memory bank

Memory Interleaving

Module 0	Module 1	Module 2	Module 3	Module 4	Module 5	Module 6	Module 7
0	1	2	3	4	5	6	7
8	9	10	11	12	13	14	15
16	17	18	19	20	21	22	23
24	25	26	27	28	29	30	31

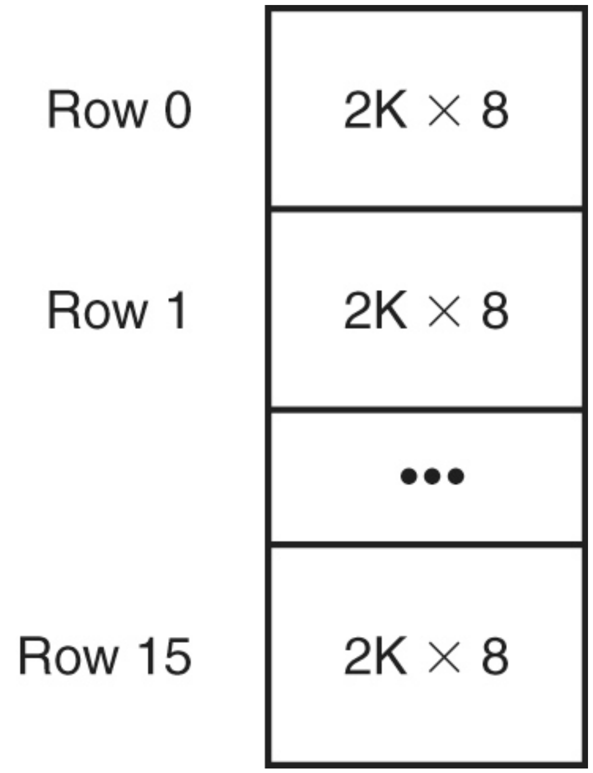
Low-Order Interleaving

Module 0	Module 1	Module 2	Module 3	Module 4	Module 5	Module 6	Module 7
0	4	8	12	16	20	24	28
1	5	9	13	17	21	25	29
2	6	10	14	18	22	26	30
3	7	11	15	19	23	27	31

High-Order Interleaving

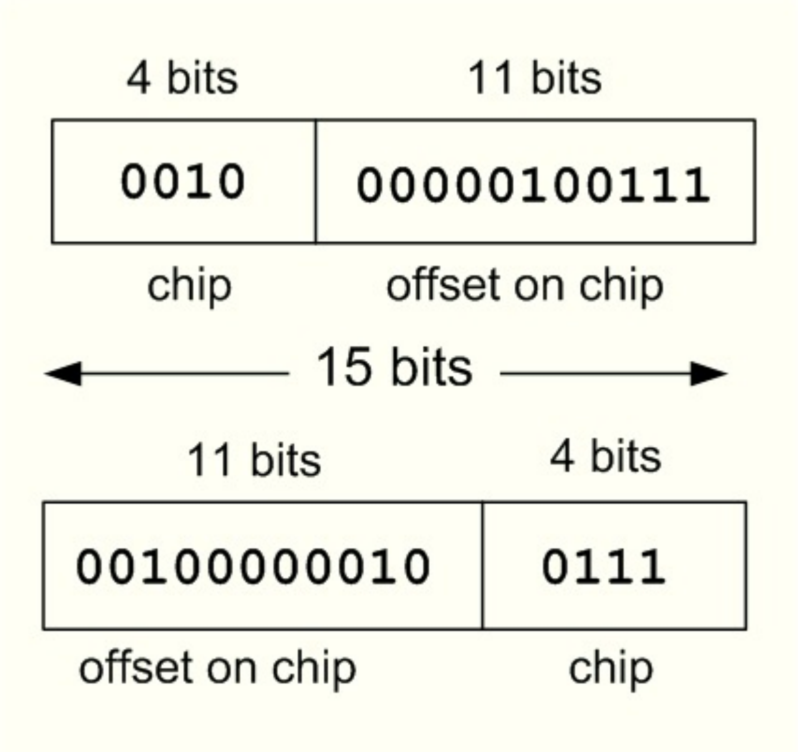
Memory Organization

- Example: Computer memory composed of 16 2K x 8 bit chips
- Total memory locations?
  - 32K = $2^5 \times 2^{10} = 2^{15}$
  - 15 bits are needed for each address
- Wiring
  - 4 bits will select which chip (out of 16)
  - 11 bits will select a particular byte inside the selected chip



Memory Organization

- In high-order interleaving the high-order 4 bits select the chip
- In low-order interleaving the low-order 4 bits select the chip



Example – Memory Organization

- Example: Suppose we build a 8M x 32 word-addressable main memory using 512K x 8 RAM chips.
- **How many RAM chips are necessary?**
  - 8M/512K * 32/8 = 16 * 4 = 64
- **How many RAM chips are there per word?**
  - 32/8 = 4 chips per word
- **How many address bits are needed per RAM chip?**
  - 512K addresses = 2^{10+9} = 19 address bits
- **How many banks will there be?**
  - 8M/512K = 16 banks

Example – Memory Organization

- Example: Suppose we build a 8M x 32 word-addressable main memory using 512K x 8 RAM chips.
- **How many address bits are needed for all memory?**
  - Word addressable: 8M addresses = $8*2^{20} = 2^{20+3}$ = 23 address bits
  - Byte-addressable, 8M addresses * 4 bytes per word = 2^{20+3+2} = 25 address bits
- **If high-order interleaving is used, where would address 247193_{16} be located?**
  - **Bank 4** (247193_{16} = 010 0100 0111 0001 1001 0011_2)
- **If low-order interleaving is used, where would address 247193_{16} be located?**
  - **Bank 3** (247193_{16} = 010 0100 0111 0001 1001 0011_2)

Exercise – Memory Organization

- Exercise: Build a 1M x 16 word-addressable main memory using 128K x 4 RAM chips.
  1. **How many address bits are needed per RAM chip?**
  2. **How many RAM chips are there per word?**
  3. **How many RAM chips are necessary?**
  4. **How many address bits are needed for all memory?**
  5. **How many address bits would be needed if it were byte addressable?**
  6. **How many banks will there be?**
  7. **What bank would contain address 47129_{16} with (a) high-order interleaving or (b) low-order interleaving?**

Solution to Exercise

1. Each RAM chip has 128K locations: $2^7 * 2^{10}$ = **17 bits**

2. Each RAM chip location stores 4 bits, but we need 16:
   1. **4 chips needed per word**

3. Each RAM chip has 128K locations, but we need 1M locations:
   1. 1M/128K = 8 (times 4 chips per word) = **32 RAM chips** (8 rows, 4 columns)

4. Memory is 1M: 2^20 = **20 bits for all of memory**

5. Byte addressable adds 1 more bit here (to select either the lower 8 or upper 8 of the 16 bit long word): **21 bits**

6. **8 banks** of memory, where each bank has 4 chips

7. Address is 20 bits long, bank is upper 3 bits (2^3=8):
   47129(16) = 0100 0111 0001 0010 1001 (2)
   With high-order interleaving, bank is **#2**
   With low-order interleaving, bank is **#1**

Interrupts

- High priority events (requiring immediate handling) can alter normal program flow
  - I/O requests
  - Arithmetic errors (division by 0)
  - Invalid instructions
- CPU is notified of the high-priority event via an **interrupt**
  - Nonmaskable interrupts are high-priority interrupts that cannot be ignored
- Each interrupt is associated with a procedure (subroutine) that tells the CPU what to do
  - Copy data from the NIC?
  - Give the video card a new frame to display?