

Computer Science II – Part 2
Computer Systems Organization

Prof. Dr. Dieter Hogrefe
Dr. Xiaoming Fu
Kevin Scott, M.A.

Telematics group
University of Göttingen, Germany

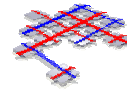


Table of Content

- **Introduction**
- **Processors**
 - CPU Organization
 - Instruction Execution
- **Memories**
 - Main memory
 - Secondary memory
- **Computer Architectures**
- **I/O**
- **Computer Buses**

SS 2003 Computer Science II

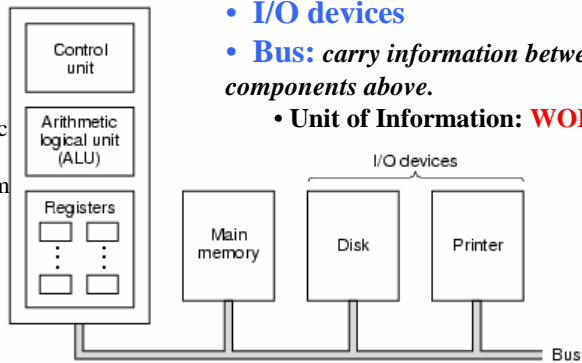
2

A Computer System: von Neumann Architecture

CPU:

- Control part
- ALU (Arithmetic Logic Unit)
- Registers, e.g., Program Counter (PC)

Central processing unit (CPU)



- **Processor:** *brain of the computer*
- **Memory**
- **I/O devices**
- **Bus:** *carry information between 3 components above.*

- **Unit of Information: WORD**

SS 2003 Computer Science II

3

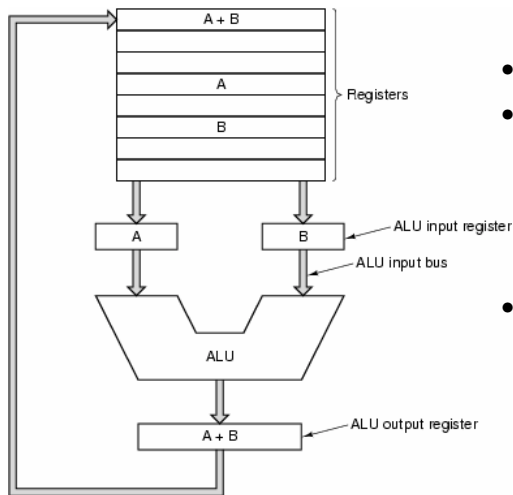
Table of Content

- Introduction
- **Processors**
 - CPU Organization
 - Instruction Execution
- Memories
 - Main memory
 - Secondary memory
- Computer Architectures
- I/O
- Computer Buses

SS 2003 Computer Science II

4

Inside a CPU...



- This shows **data path**
- Consists of:
 - ALU (Arithmetic Logic Unit)
 - Registers
 - Buses connecting pieces
- This demonstrates how an instruction is performed

SS 2003 Computer Science II

5

Instructions

- Digital computers can solve problems for people by carrying out *instructions* given to it.
 - A sequence of instructions describing how to perform a certain task is called a *program*.
 - Computer hardware can recognize and directly execute a limited set of instructions
 - Inside a computer, CPU executes instructions
- Two categories of instructions
 - *Register-memory instructions*
 - *Register-register instructions*

SS 2003 Computer Science II

6

Instruction Execution: the *Fetch-Decode-Execute Cycle*

1. *Fetch* the next instruction from memory into the instruction register.
2. Change the *program counter (PC)* to point to the following instruction.
3. *Determine the type* of instruction just fetched.
4. If the instruction uses a word in memory, determine where it is. (*Addressing*)
5. Fetch the word, if needed, into a CPU register.
6. *Execute* the instruction.
7. Go to step 1 to begin executing the next instruction.

Interpreter

- How about write a program to imitate the execution procedure?
 - Sure, **interpreter** does this!
- An interpreter fetches, examines, and executes the instructions of another program.
- Let's look at an example...>

```

1. public class Interp {
2.     static int PC;        // program counter, holds addr of next instruction
3.     static int AC;        // accumulator, a register for doing arithmetic
4.     static int instr;     // a register, holds current instruction
5.     static int instr_type; //the instruction type: ADD, or Boolean OR, AND...
6.     static int data_loc;  //data addr, -1 if no data
7.     static boolean run_bit = true; //stop flag

8.     public static void interpret(int memory[], int starting_address) {
9.         // This procedure interprets programs for a simple computer with
10.        // instructions having one memory operand. The ADD instruction adds
11.        // an integer in memory to the AC, for example.

12.        PC = starting_address;
13.        while (run_bit) {
14.            instr = memory[PC]; //fetch next instruction
15.            PC = PC + 1;        //move to next instruction address
16.            instr_type = get_instr_type(instr); // determine instruction type
17.            data_loc = find_data(instr, instr_type); // locate data (return -1 if no)
18.            if(data_loc >= 0) // check whether there is operand
19.                data = memory[data_loc]; // fetch the data
20.            execute(instr_type, data); // execute instruction
21.        }
22.    }
23.
24.    private static int get_instr_type(int addr) {...}
25.    private static int find_data(int instr, int type) {...}
26.    private static void execute(int type, int data) {...}
27. }

```

Describing CPU operation using java: an interpreter for a simple computer

SS 2003 Computer Science II

9

Software v.s. Hardware

- Without interpreter: hardware (logic circuits) directly executes instructions written in machine language L.
- With an interpreter (software): instructions are translated to hardware and let the latter perform
- What's the benefit of using software to replace some function of hardware?
 - Do not always build new hardware when a more complex instruction comes!
 - This vital technical advantage is also an economic issue!

SS 2003 Computer Science II

10

Table of Content

- Introduction
- Processors
 - CPU Organization
 - Instruction Execution
- **Memories**
 - **Main memory**
 - **Secondary memory**
- Computer Architectures
- I/O
- Computer Buses

SS 2003 Computer Science II

11

Memory

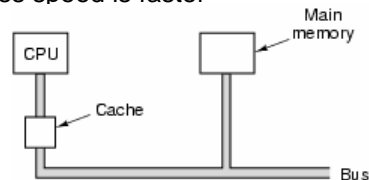
- Memories as part of computer, store programs and data
- Basic unit: bit (do you know *why?*)
 - A k-bit memory will have 2^k different combinations of content
- How memories are organized?
 - A memory consists of certain number of cells (locations)
 - Each has a number, called its "**address**"
 - A cell is the smallest addressable unit. **Byte**: 8-bit cells
 - **Word** is certain number of bytes, a unit that a machine can operate with.
 - 32-bit machine, 64-bit machine...

SS 2003 Computer Science II

12

Main Memory v.s. Cache Memory

- Main memories locate outside CPU but interconnect with CPU by bus
- What if a CPU is faster than memory?
 - CPU may wait for "fetch" operation from memory in a CPU cycle!
 - And this (faster CPU, slower memory) is very typical case
- Solution: build fast, "local" memory inside the CPU
 - Cache memory
 - Advantage: overall access speed is faster



SS 2003 Computer Science II

13

RAMs and ROMs

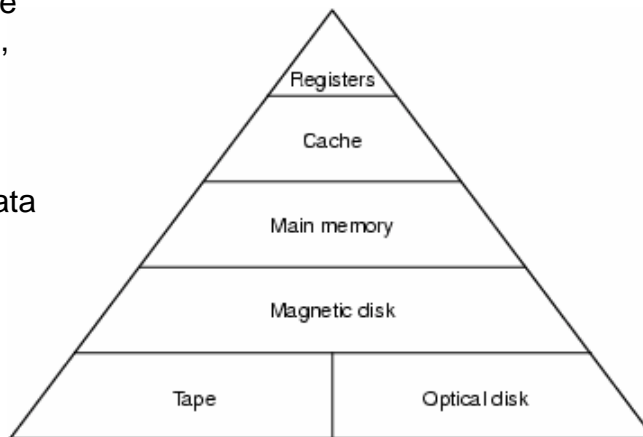
- RAM (Random Access Memory): can both read and write: for main memory. Two types:
 - Static RAM (SRAM): based on D flip-flops. Fast
 - Dynamic RAM (DRAM): array of cells. Large capacity
 - Recently, SDRAM (Synchronous DRAM): driven by a single synchronous clock. For large caches
- ROM (Read-Only Memory): content cannot be changed once built. Variants:
 - PROM (Programmable ROM)
 - EPROM (Erasable PROM) and EEPROM
 - Flash memory

SS 2003 Computer Science II

14

Secondary Memory

- Main memories are fast but expensive, have only limited capacity.
- We need to store large amount of data
- → Hierarchy of memories!
 - Tradeoff between speed, cost and capacity



SS 2003 Computer Science II

15

Error checking for memories

- Data stored in large memories are likely to be corrupted!
- Thus, error-checking code are necessary
- **Parity code**: a simple error-checking code
 - Counts 1s in a word
 - If the count number is odd/even: the word has odd/even parity
 - A bit is written into the memory within a word
 - If the word does not fit the parity check, it causes an error
- A bit parity code can only detect one-bit error
 - Complex codes are available

SS 2003 Computer Science II

16

Table of Content

- Introduction
- Processors
 - CPU Organization
 - Instruction Execution
- Memories
 - Main memory
 - Secondary memory
- **Computer Architectures**
- I/O
- Computer Buses

SS 2003 Computer Science II

17

Computer Architectures

- **Computer architecture = Instruction Set Architecture + Machine Organization**
- Computer Set Architecture (ISA) describes the structure as seen by a programmer
- Machine organization describes the functional units, their interconnect and arrangements.

SS 2003 Computer Science II

18

ISAs

- ISAs: interface between software and hardware
- Some ISAs

IBM 360	197x
SUN SPARC	1987
DEC Alpha	1992
SGI MIPS	1986
Intel x86	1978

SS 2003 Computer Science II

19

Machine organization

- With a same ISA, machines can be built using different
 - pipeline length,
 - number of functional units,
 - cache size and organisation,
 - sophistication of instruction issue unit (ability to issue/start more than one instruction at the same time), etc.
- However, despite different performance (speed), a same instruction could be run by the same ISA in the exactly same way

SS 2003 Computer Science II

20

RISC v.s. CISC

- Many modern CPUs support varying length of instructions, and store back the result in one of the operand.
 - The size of instruction set is large, thus this kind of CPU is called *Complex Instruction Set Computer (CISC)*.
 - E.g., Intel x86, M68xxx, IBM 360, ...
- *Reduced Instruction Set Computer (RISC)*
 - ISA runs instructions having same size, e.g., 32bit
 - All ALU instructions' operands are *registers*.
 - Memory can only be accessed by *LOAD/STORE* instructions
 - e.g., MIPS 2000
- Why introduced RISC? → same size of instructions requires less bits for opcodes, thus less instructions are necessary
- Why CISC CPUs are still being developed?

Table of Content

- Introduction
- Processors
 - CPU Organization
 - Instruction Execution
- Memories
 - Main memory
 - Secondary memory
- Computer Architecture
- **I/O**
- Computer Buses

Input/Output Devices

- A von Neumann architecture computer's major components:
 - CPU
 - Memories
 - Input/Output (I/O)
- Now we study I/O devices such as keyboard, printers, scanners, and modems.

I/O Device Examples

<u>Device</u>	<u>Behavior</u>	<u>Partner</u>	<u>Data Rate (KB/sec)</u>
Keyboard	Input	Human	0.01
Mouse	Input	Human	0.02
Line Printer	Output	Human	1.00
Floppy disk	Storage	Machine	50.00
Laser Printer	Output	Human	100.00
Optical Disk	Storage	Machine	500.00
Magnetic Disk	Storage	Machine	5,000.00
Network-LAN	Input or Output	Machine	20 – 1,000.00
Graphics Display	Output	Human	30,000.00

I/O System Performance

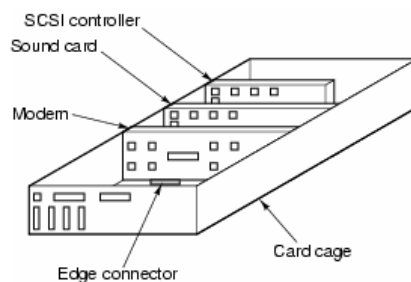
- I/O System performance depends on many aspects of the system (“*limited by weakest link in the chain between OS and device*”):
 - The CPU
 - The memory system:
 - Internal and external caches
 - Main Memory
 - The underlying interconnection (buses)
 - The I/O controller
 - The I/O device
 - The speed of the I/O software (Operating System)
 - The efficiency of the software’s use of the I/O devices
- Two common performance metrics:
 - Throughput: I/O bandwidth
 - Response time: Latency

SS 2003 Computer Science II

25

Physical Structure of a Computer

- Motherboard (mainboard)
 - Contains CPU, sockets to memories, I/O boards
- I/O devices
 - Video display & card
 - Keyboard
 - Mouse
 - Sound card
 - Modem
 - Mouse
 - ISDN card
 - Network Interface Card (NIC)
 - More? ---wireless Card, ...

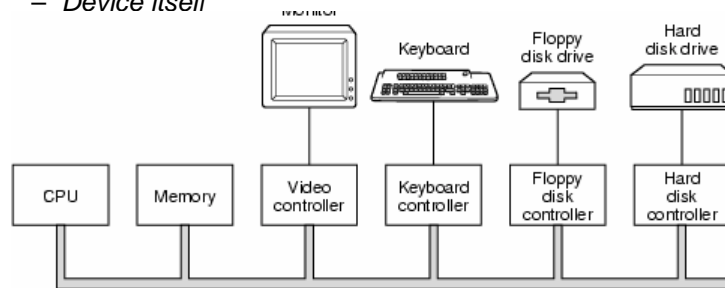


SS 2003 Computer Science II

26

Logical Structure of a Simple Computer

- CPU, memory and *I/O devices*
- Bus, connecting these components
- Each I/O device has two parts:
 - *Controller*
 - *Device itself*



SS 2003 Computer Science II

27

Controller

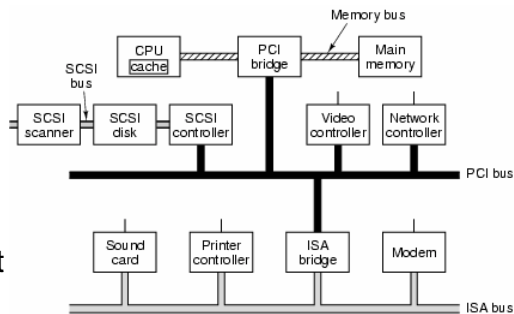
- Function: to control its I/O device and handle bus access for it.
- *DMA* (Direct Memory Access): direct access from memory, without bothering CPU
- After I/O data access is finished, an **interrupt** will be triggered:
 - CPU has to suspend its current program, and run a special procedure --- "interrupt handler"
 - Examples of interrupt handler: check for errors, take special action, inform OS that I/O is finished
 - After finishing interrupt handler, CPU continues with the suspended program

SS 2003 Computer Science II

28

Standards for PC Bus' Operation

- Industry Standard Architecture (ISA)
 - Differs from *Instruction Set Architecture (ISA)*!
- PS/2
- Extended ISA (EISA)
- Peripheral Component Interconnect (PCI)



A typical modern PC with a PCI bus and an ISA bus:

- Modem and sound card are **ISA** devices
- SCSI controller is a **PCI** device

Keyboard

- Each keyboard button associate with an electromagnetic or mechanical sensor
- Detect the depression and release of buttons and generate interrupts accordingly
- Keyboard interrupt handler is triggered upon the interrupt
- Handling of Multikey sequences involving SHIFT, CTRL, and ALT (including CTRL-ALT-DEL) keys is done in software.

Display

- A terminal = keyboard + monitor
 - Some workstations integrated both, PCs separate them
- A screen could be a *CRT (Cathode Ray Tube) monitor* or an *LCD (Liquid Crystal Display) display*
 - converts binary electrical signals to a visual display
 - This display consists of bright and dark spots.



A Terminal

Mouse

- When a mouse moves a certain minimum distance (e.g., 0.01 inch), it sends a sequence of *3 bytes* to the computer
 - 1st byte contains the signed integer telling how many units the mouse has moved in the *x-direction* since the last time.
 - 2nd byte give the same information for *y motion*.
 - The 3rd byte contains the *current state* of the mouse buttons.
- Then it displays an arrow on the screen corresponding to where the mouse is

Other I/O Devices

- Printers
- Modems
- ISDN devices
- Network Interface Cards (NICs)
- Sound cards
- Video cameras...

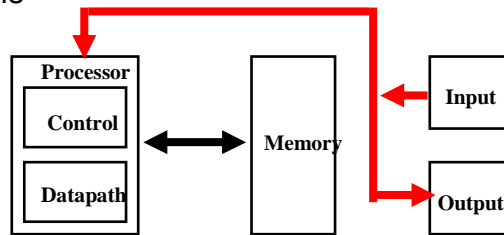
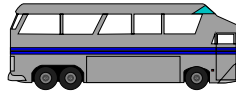
Table of Content

- Introduction
- Processors
 - CPU Organization
 - Instruction Execution
- Memories
 - Main memory
 - Secondary memory
- Computer Architecture
- I/O
- **Computer Buses**

What is a bus?

A Bus Is:

- shared communication link
- single set of wires used to connect multiple subsystems

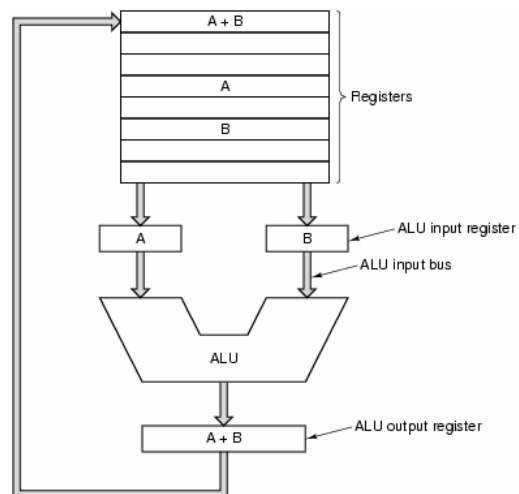


- Just to transport information, not a space for storing information!

SS 2003 Computer Science II

35

Remember the data path: Inside CPU



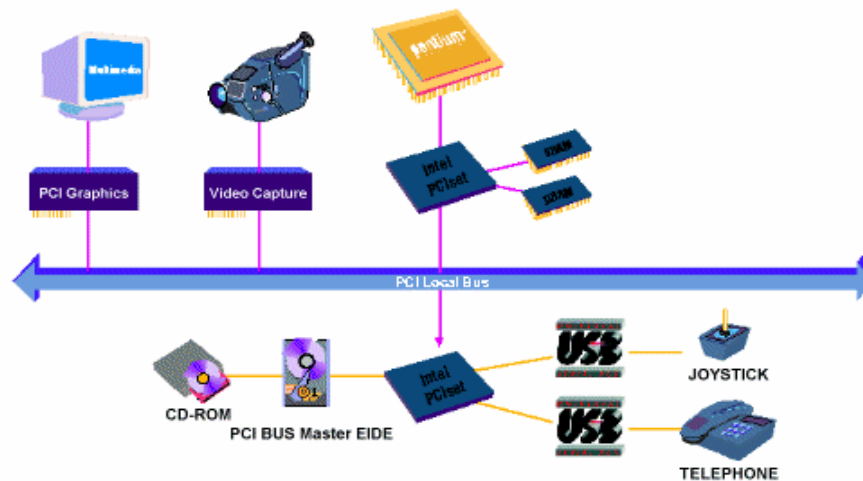
SS 2003 Computer Science II

36

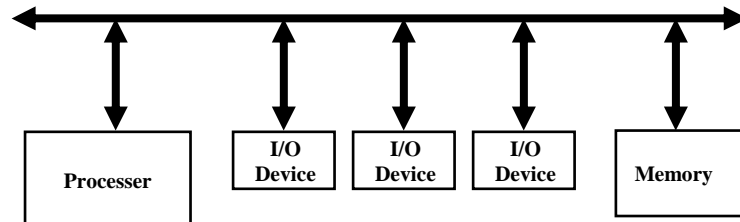
One Bus or More?

- Q:
 - Is one bus is enough to transport its passengers (instructions and data)?
- A:
 - External bus (system bus), could be ONE. E.g., early simple PC
 - Even for PCs, modern ones use several buses, at least:
 - I/O bus for the I/O devices
 - Memory bus for CPU \leftrightarrow memory

Buses



Advantages of Buses

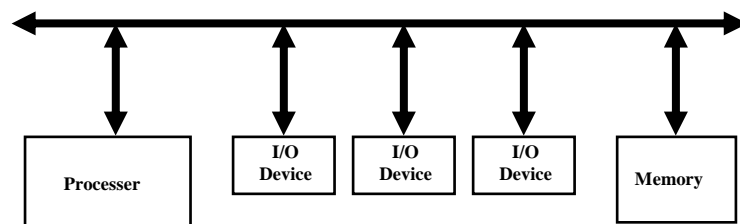


- Versatility:
 - New devices can be added easily
 - Peripherals can be moved between computer systems that use the same bus standard
- Low Cost:
 - A single set of wires is shared in multiple ways

SS 2003 Computer Science II

39

Disadvantage of Buses

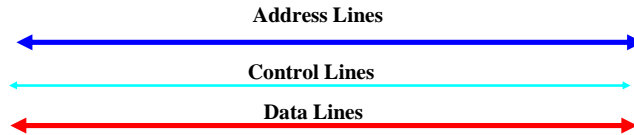


- It creates a communication bottleneck
 - The bandwidth of that bus can limit the maximum I/O throughput
- The maximum bus speed is largely limited by:
 - The **length** of the bus
 - The **number** of devices on the bus
 - The need to support a range of devices with:
 - Widely varying latencies
 - Widely varying data transfer rates

SS 2003 Computer Science II

40

The General Organization of a Bus



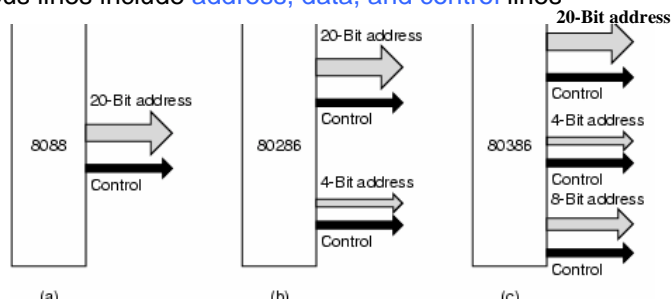
- **Address lines:**
 - Indicate where is the location of information
- **Control lines:**
 - Signal requests and acknowledgments
 - Indicate what type of information is on the data lines
- **Data lines** carry information between the source and the destination:
 - Data
 - Complex commands

SS 2003 Computer Science II

41

Bus Width

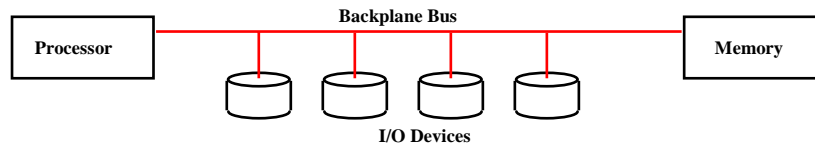
- **Bus width:** number of lines for addressing memory or I/O locations
 - E.g., IBM PC: 8088 CPU, 20-bit address bus → allow to address $2^{20}=1\text{MB}$ of memory
 - Bus lines include **address, data, and control** lines



SS 2003 Computer Science II

42

A Computer System with One Bus

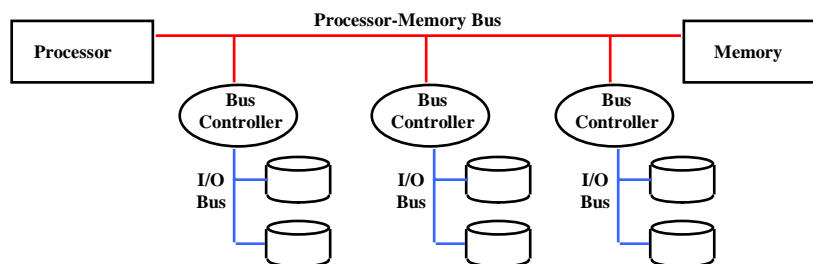


- A single bus is used for:
 - Processor to memory communication
 - Communication between I/O devices and memory
- Advantages: Simple and low cost
- Disadvantages: slow and the bus can become a major bottleneck
- Example: IBM PC - AT

SS 2003 Computer Science II

43

A Two-Bus System

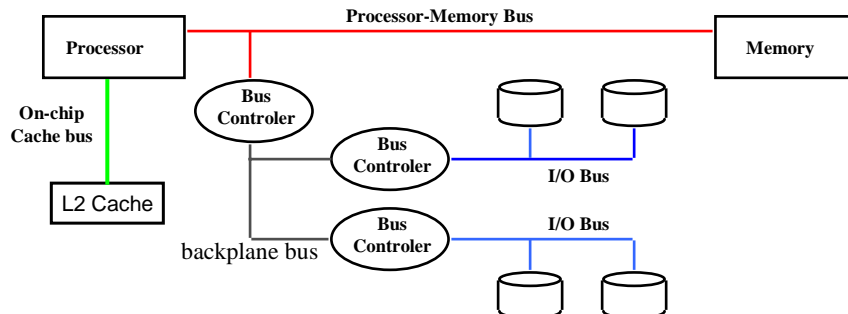


- I/O buses tap into the processor-memory bus via bus adaptors:
 - Processor-memory bus: mainly for processor-memory traffic
 - I/O buses: provide expansion slots for I/O devices
- Apple Macintosh-II
 - NuBus: Processor, memory, and a few selected I/O devices
 - SCCI Bus: the rest of the I/O devices

SS 2003 Computer Science II

44

A Three-Bus System (+ CPU cache)



- A small number of backplane buses tap into the processor-memory bus
 - Processor-memory bus is only used for processor-memory traffic
 - I/O buses are connected to the backplane bus
- Advantage: loading on the processor bus is greatly reduced

Bus protocol

- Bus protocol – rules about:
 - how the bus works
 - which all devices attached must obey
- A large number of bus protocols:
 - IBM PC bus (PC/XT), *ISA bus* (PC/AT), *EISA* (80386), *PCI bus* (many PCs), *SCSI bus* (many PCs & workstations), *Nubus* (Macintosh), *USB – Universal Serial Bus* (modern PCs)...
 - How can these numerous incompatible systems work together? *Economic issue again*

How buses work?

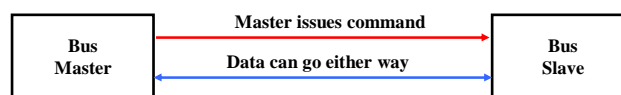
- Master – slave model
 - Master: active, can initiate bus transfer
 - Slave: passive, wait for request
- E.g.,
 - CPU can be mastering data transfer with I/O controller
 - No memory can be master

Master	Slave	Example
CPU	Memory	Fetching instructions and data
CPU	I/O device	Initiating data transfer
I/O	Memory	DMA (Direct Memory Access)

SS 2003 Computer Science II

47

Master versus Slave



- A **bus transaction** includes two parts:
 - (after a master is determined)
 - Master issues the command (and address) – request
 - Transferring the data (can be either way) – action
- Master is the one who starts the bus transaction by:
 - issuing the command (and address)
- Slave is the one who responds to the address by:
 - Sending data to the master if the master ask for data
 - Receiving data from the master if the master wants to send data

SS 2003 Computer Science II

48

Bus: other issues

- *Bus arbitration*: to prevent multiple devices to serve as master at the same time
 - E.g., an I/O device may need to read & write memory and cause interrupts, while CPU is normally a master
- An *interrupt controller* indexes the multiple interrupts, assigning them with priorities.

Summary

- Von Neumann computers are built up from 3 types of components:
 - CPU: *fetch* instructions from a memory, *decode* and then *execute* them
 - Memory:
 - main memory: store programs currently executed. Fast speed (cache); Error-correction
 - Secondary: slower but larger capacity. IDE/SCSI/floppy Disks, CD-ROM,...
 - I/O: transfer information into and out of the computer. Terminal, mouse, printer, modem...
- Bus connects these 3 components
 - “Master-slave” model
 - Bus width determines the address space can be accessed
 - Several is possible. (E)ISA, PCI, USB...