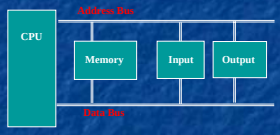
CPU How It Works

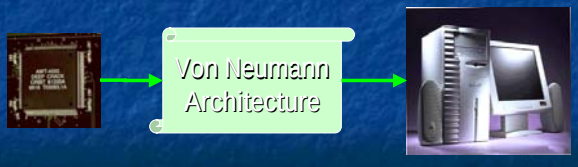
Generic Block Diagram



2

Hardware

The Von Neumann Architecture



Designing Computers

- All computers more or less based on the same basic design, the Von Neumann Architecture!

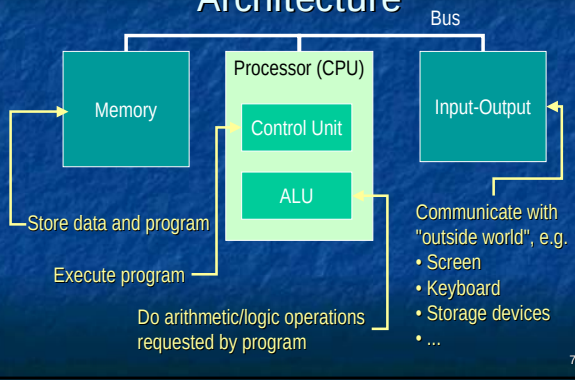
5

The Von Neumann Architecture

- Model for designing and building computers, based on the following three characteristics:
  1) The computer consists of four main sub-systems:
     - Memory
     - ALU (Arithmetic/Logic Unit)
     - Control Unit
     - Input/Output System (I/O)
  2) Program is stored in memory during execution.
  3) Program instructions are executed sequentially.

6

The Von Neumann Architecture



7

Memory Subsystem

- Memory, also called RAM (Random Access Memory),
  - Consists of many memory cells (storage units) of a fixed size.
    Each cell has an address associated with it: 0, 1, ...
  - All accesses to memory are to a specified address.
    A cell is the minimum unit of access (fetch/store a complete cell).
  - The time it takes to fetch/store a cell is the same for all cells.
- When the computer is running, both
  - Program
  - Data (variables) are stored in the memory.

8

Memory Size / Speed

- Typical memory in a personal computer (PC):
  - 64MB - 256MB
- Memory sizes:
  - Kilobyte (KB) = 2^{10} =1,024 bytes ~ 1 thousand
  - Megabyte(MB) = 2^{20} =1,048,576 bytes ~ 1 million
  - Gigabyte(GB) = 2^{30} = 1,073,741,824 bytes ~ 1 billion
- Memory Access Time (read from/ write to memory)
  - 50-75 nanoseconds (1 nsec. = 0.000000001 sec.)
- RAM is
  - volatile (can only store when power is on)
  - relatively expensive

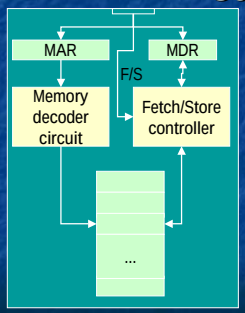
9

Operations on Memory

- Fetch (address):
  - Fetch a copy of the content of memory cell with the specified address.
  - Non-destructive, copies value in memory cell.
- Store (address, value):
  - Store the specified value into the memory cell specified by address.
  - Destructive, overwrites the previous value of the memory cell.
- The memory system is interfaced via:
  - Memory Address Register (MAR)
  - Memory Data Register (MDR)
  - Fetch/Store signal

10

Structure of the Memory Subsystem



- Fetch(address)
  - Load address into MAR.
  - Decode the address in MAR.
  - Copy the content of memory cell with specified address into MDR.
- Store(address, value)
  - Load the address into MAR.
  - Load the value into MDR.
  - Decode the address in MAR
  - Copy the content of MDR into memory cell with the specified address.

11

Input/Output Subsystem

- Handles devices that allow the computer system to:
  - Communicate and interact with the outside world
    - Screen, keyboard, printer, ...
  - Store information (mass-storage)
    - Hard-drives, floppies, CD, tapes, ...
- Mass-Storage Device Access Methods:
  - Direct Access Storage Devices (DASDs)
    - Hard-drives, floppy-disks, CD-ROMs, ...
  - Sequential Access Storage Devices (SASDs)
    - Tapes (for example, used as backup devices)

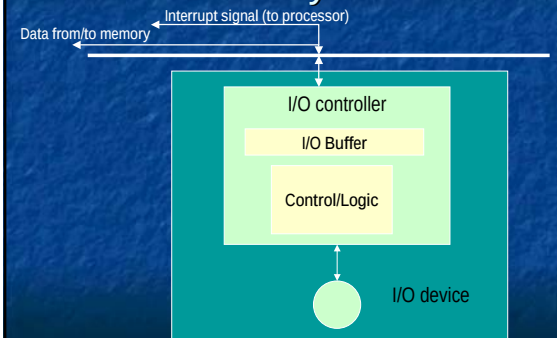
12

I/O Controllers

- Speed of I/O devices is slow compared to RAM
  - RAM ~ 50 nsec.
  - Hard-Drive ~ 10msec. = (10,000,000 nsec)
- Solution:
  - I/O Controller, a special purpose processor:
    - Has a small memory buffer, and a control logic to control I/O device (e.g. move disk arm).
    - Sends an interrupt signal to CPU when done read/write.
  - Data transferred between RAM and memory buffer.
  - Processor free to do something else while I/O controller reads/writes data from/to device into I/O buffer.

13

Structure of the I/O Subsystem



14

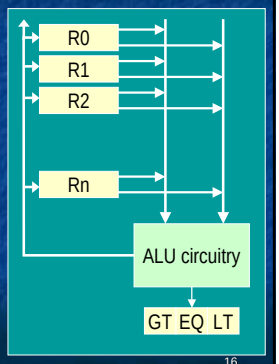
The ALU Subsystem

- The ALU (Arithmetic/Logic Unit) performs
  - mathematical operations (+, -, x, /, ...)
  - logic operations (=, <, >, and, or, not, ...)
- In today's computers integrated into the CPU
- Consists of:
  - Circuits to do the arithmetic/logic operations.
  - Registers (fast storage units) to store intermediate computational results.
  - Bus that connects the two.

15

Structure of the ALU

- Registers:
  - Very fast local memory cells, that store operands of operations and intermediate results.
  - CCR (condition code register), a special purpose register that stores the result of <, = , > operations
- ALU circuitry:
  - Contains an array of circuits to do mathematical/logic operations.
- Bus:
  - Data path interconnecting the registers to the ALU circuitry.



16

The Control Unit

- Program is stored in memory
  - as machine language instructions, in binary
- The task of the control unit is to execute programs by repeatedly:
  - Fetch from memory the next instruction to be executed.
  - Decode it, that is, determine what is to be done.
  - Execute it by issuing the appropriate signals to the ALU, memory, and I/O subsystems.
  - Continues until the HALT instruction

17

Machine Language Instructions

- A machine language instruction consists of:
  - Operation code, telling which operation to perform
  - Address field(s), telling the memory addresses of the values on which the operation works.
- Example: ADD X, Y (Add content of memory locations X and Y, and store back in memory location Y).
- Assume: opcode for ADD is 9, and addresses X=99, Y=100

Opcode (8 bits)	Address 1 (16 bits)	Address 2 (16 bits)
00001001	0000000001100011	0000000001100100

18

How does this all work together?

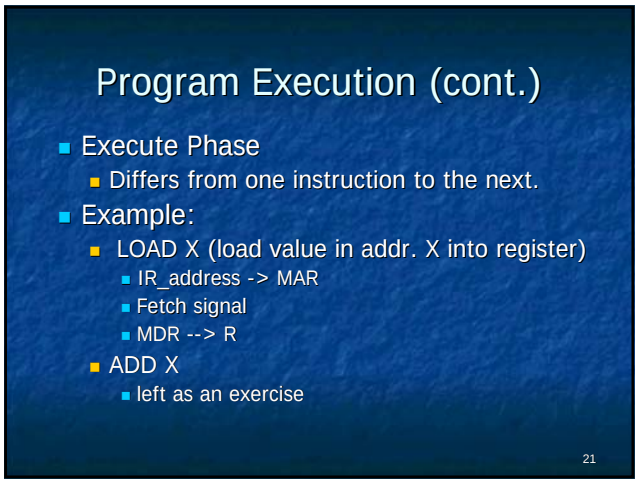
- Program Execution:
  - PC is set to the address where the first program instruction is stored in memory.
  - Repeat until HALT instruction or fatal error
    - Fetch instruction
    - Decode instruction
    - Execute instruction
    - End of loop

19

Program Execution (cont.)

- Fetch phase
  - PC --> MAR (put address in PC into MAR)
  - Fetch signal (signal memory to fetch value into MDR)
  - MDR --> IR (move value to Instruction Register)
  - PC + 1 --> PC (Increase address in program counter)
- Decode Phase
  - IR -> Instruction decoder (decode instruction in IR)
  - Instruction decoder will then generate the signals to activate the circuitry to carry out the instruction

20


Program Execution (cont.)
Execute Phase
Differs from one instruction to the next.
Example:
LOAD X (load value in addr. X into register)
IR_address -> MAR
Fetch signal
MDR --> R
ADD X
left as an exercise
21