Computer Organization

KR Chowdhary
Professor & Head
Email: kr.chowdhary@gmail.com
webpage: krchowdhary.com

Department of Computer Science and Engineering
MBM Engineering College, Jodhpur

November 14, 2013

The course

1. Introduction: Overview of basic digital building blocks; truth tables; basic structure of a digital computer.
2. Number representation: Integer - unsigned, signed (sign magnitude, 1's complement, 2's complement, r_s' complement); Characters - ASCII coding, other coding schemes; Real numbers - fixed and floating point, IEEE754 representation.
3. Assembly language programming for some processor.
4. Basic building blocks for the ALU: Adder, Subtractor, Shifter, Multiplication and division circuits.
5. CPU Subblock: Datapath - ALU, Registers, CPU buses; Control path - microprogramming (only the idea), hardwired logic; External interface.

6. Memory Subblock: Memory organization; Technology - ROM, RAM, EPROM, Flash, etc. Cache; Cache coherence protocol for uniprocessor (simple).
7. I/O Subblock: I/O techniques - interrupts, polling, DMA; Synchronous vs. Asynchronous I/O; Controllers.
8. Peripherals: Disk drives; Printers - impact, dot matrix, ink jet, laser; Plotters; Keyboards; Monitors.
9. Advanced Concepts: Pipelining; Introduction to Advanced Processors.

Books and References:

- Computer Organization, fifth edition: Carl Hamacher, Zvonko Vranesic, Safwat Zaky, McGrawHill, Indian Edition
- Computer Architecture and Organization: J.P. Hayes, McGrawHill
- Computer Architecture and organization: William Stallings
- Computer Architecture: H. Patterson, Elsevier
- Net, Wikipedia, OCW MIT
- http://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-823-computer-system-architecture-fall-2005/lecture-notes/
- http://ocw.mit.edu/courses/electrical-engineering-and-computer-science/

Class test, attendance, Midsem, endsem evaluation:

- 15% Quizes, Assignments,
- 15% first midsem, 15% II midsem,
- 10% Project
- 40% endsem
- 5% Attendance

Outline of Introduction

- What is a computer?

  A computer is just a digital system

  - Consists of combinational and sequential logic
  - A big, finite state machine
  - A computer just does what software tells it to do
- Software is a series of instructions
  1. What instructions does a computer need?
  2. What kinds of instructions are there?
  3. How do we represent instructions?
- What is computer architecture?

  Architecture is the attributes of a computer seen by the machine language programmer
- Why are there different types of computers?
- How do we tell computers what to do?

What is Computer Architecture?

- Strictly speaking, a computer architecture specifies what the hardware looks like (its interface), so that we can write software to run on it
- Exactly what instructions does it have, Number of register storage locations, etc
- Computer architecture includes:
  1. Instruction set
  2. Instruction format
  3. operation codes
  4. addressing modes
  5. all registers and memory locations that may be directly manipulated or tested by a machine language program
  6. formats for data representation

- It is: CPU, ALU, Control unit, memory, and the Buses for connection between these components
- Various functional blocks:

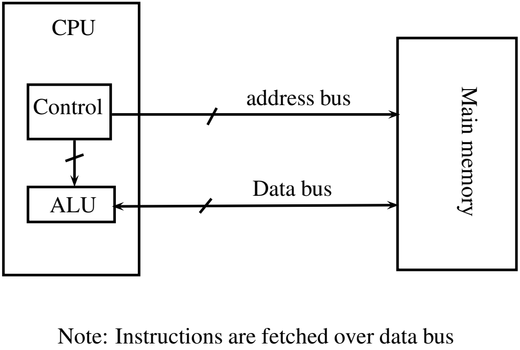


Figure: Functional block diagram of computer

- Blocks:

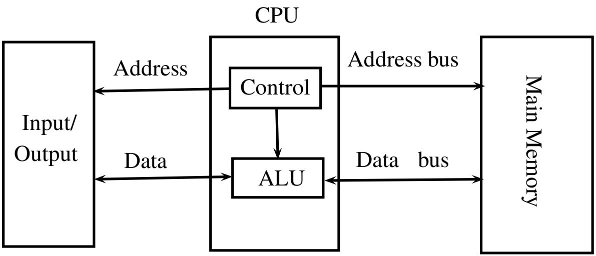


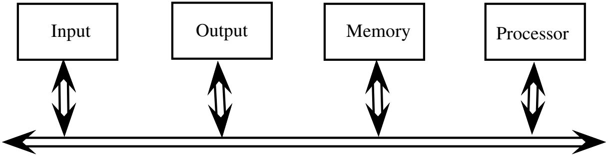
Figure: Functional block diagram of computer with IO

- Connections are buses, size of buses: 8, 16, 32, 64. (older systems: 8, 12, 24, 40, etc.)
- Von Neumann Model (Arithmetic and Boolean logic, memory: R/W, Execution Control: branches and Jumps).
- Bottlenecks of Von Neumann architecture ?: 1) excessively dependent on addresses (every memory access must start by

Single bus architecture Functional blocks

- low cost and flexibility for attaching IO devices.
- only one transfer at a time
- How to handle Speed mismatch in devices ?

  Use Buffer registers for devices for handling speed mismatch



Single bus structure

Technology is the dominant factor in computer design

- 0-Mechanical / Electromechanical
  - Liebniz's calculator (1685), Joseph Jacquard loom (1805)
  - Charles Babbage's difference and analytical engines (1833, 1837, 1853)
  - Herman Hollerith's census tabulator (1890)
  - Howard Aiken's (Harvard) Mark I (1944)
  - John Von Neumann: stored prog. concept (prog+data in same memory)(1940s)
- 1: Vacuum tube
  - ENIAC (1946), UNIVAC (1951)
  - IAS machine (1952)
  - IBM 701 (1953), IBM 709 (1958)
- 2: Transistor
  - DEC PDP-1 / 4 / 7 / 9 / 15
  - DEC PDP-5 / 8 / 12
  - DEC PDP-6 / 10
  - IBM 7090 / 7094 /
  - IBM 1401, IBM 1620, CDC1604, CDC 6600

Computer Generations contd.

- 3 - Integrated circuit

  IBM 360/370
  DEC PDP-8/I, DEC PDP-11/40, DEC VAX 11/780
- 4 - Very Large Scale Integration (VLSI) / Microprocessor

  Intel 4004, Intel 8008, Intel 8080 / 8085, Zilog Z80 / Z8 / Z8000
  Intel 8086 / 8088, Intel 80186 / 80286 / 80386 / 80486
  Intel IA-32: Pentium, PII, PIII, p4, Celeron, Xeon...
  Motorola 6800, / 68010 / 68020 / 68030/68040/68060 ...
  DEC PDP-11/03, DEC MicroVAX
  SPARC-1 / SPARC-2 / SuperSPARC, HyperSPARC, UltraSPARC, IBM RISCSystem-6000, Power series
  DEC Alpha,
- 5 - Homogeneous parallel processors

System Software

1. Receiving and interpreting user commands
2. Managing the storage, file i/o
3. Running standard programs, like spreadsheet, word,..
4. Controlling IO devices
5. Translation of programs
6. Linking, loading etc.

   Who does perform all these?

   The System Software

User program and OS routine sharing processor

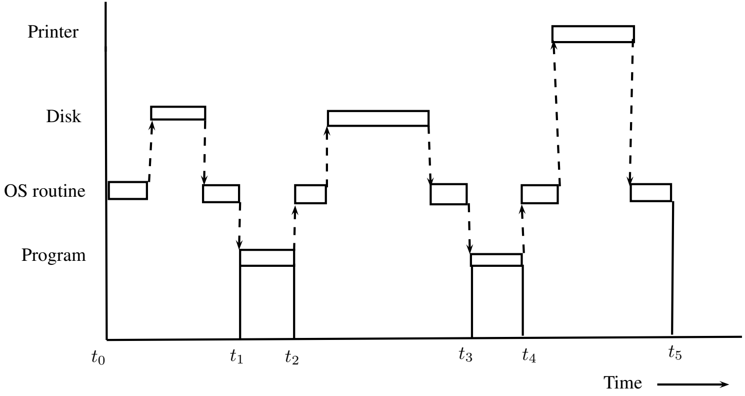


Figure: User prog. and os sharing cpu in time division multiplexing

$t_0 - t_1$: application program loading, $t_1 - t_2$: app. prog. runs, $t_2 - t_3$: loads data file, $t_3 - t_4$: runs appl., $t_4 - t_5$: prints results. at t_5 another prog. starts.

How to use the resources better?

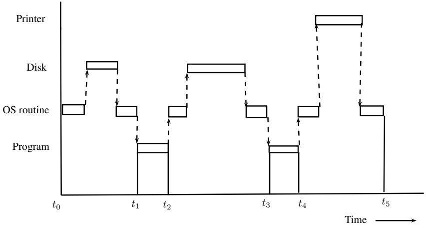


Figure: User prog. and os sharing cpu

- $t_4 - t_5$: cpu and disk are free. $t_0 - t_1, t_2 - t_3$: cpu and printer free.
- If OS concurrently execute several programs, better utilization of resources possible. This pattern is called multiprogramming or multitasking.

Performance

- = How quickly can it execute programs: function of HW + Machine language instructions.
- Does compiler effect it: Yes. How?
- Total time for execution in our exercise=$t_5 - t_0$ (elapsed time): it is effected by: speed of processor, disk, printer.
- Processor time: dependents on HW (CPU+Mem+Cache)
- Processor Clock: $P =$ clock period; rate or freq. $R = \frac{1}{P}$
- Let N: actual number of instructions to be executed., S: avg. basic steps needed per instructions.
- Total execution time, for prog. in sec. $T = \frac{N \times S}{R}$
- To $\downarrow T \Rightarrow \downarrow N, \downarrow S, \uparrow R$

Advanced Processors

- Instruction: *ADD R1, R2, R3*: ($R_3 \leftarrow R_1 + R_2$)
- Instruction cycle time = Fetch time + decode time + execute time

$$t_i = t_f + t_d + t_e$$

- Next Instruction can be read, while previous addition takes place.
- This results to overlapping execution (called pipelining). Ideally $|S| = 1$.
- Higher degree of execution possible by multiple instruction-pipelines ((called superscalar arch.)