CS 223 Computer Organization & Architecture

Lecture 22 [11.03.2020]

Review of Basic Computer Organization

John Jose

Assistant Professor

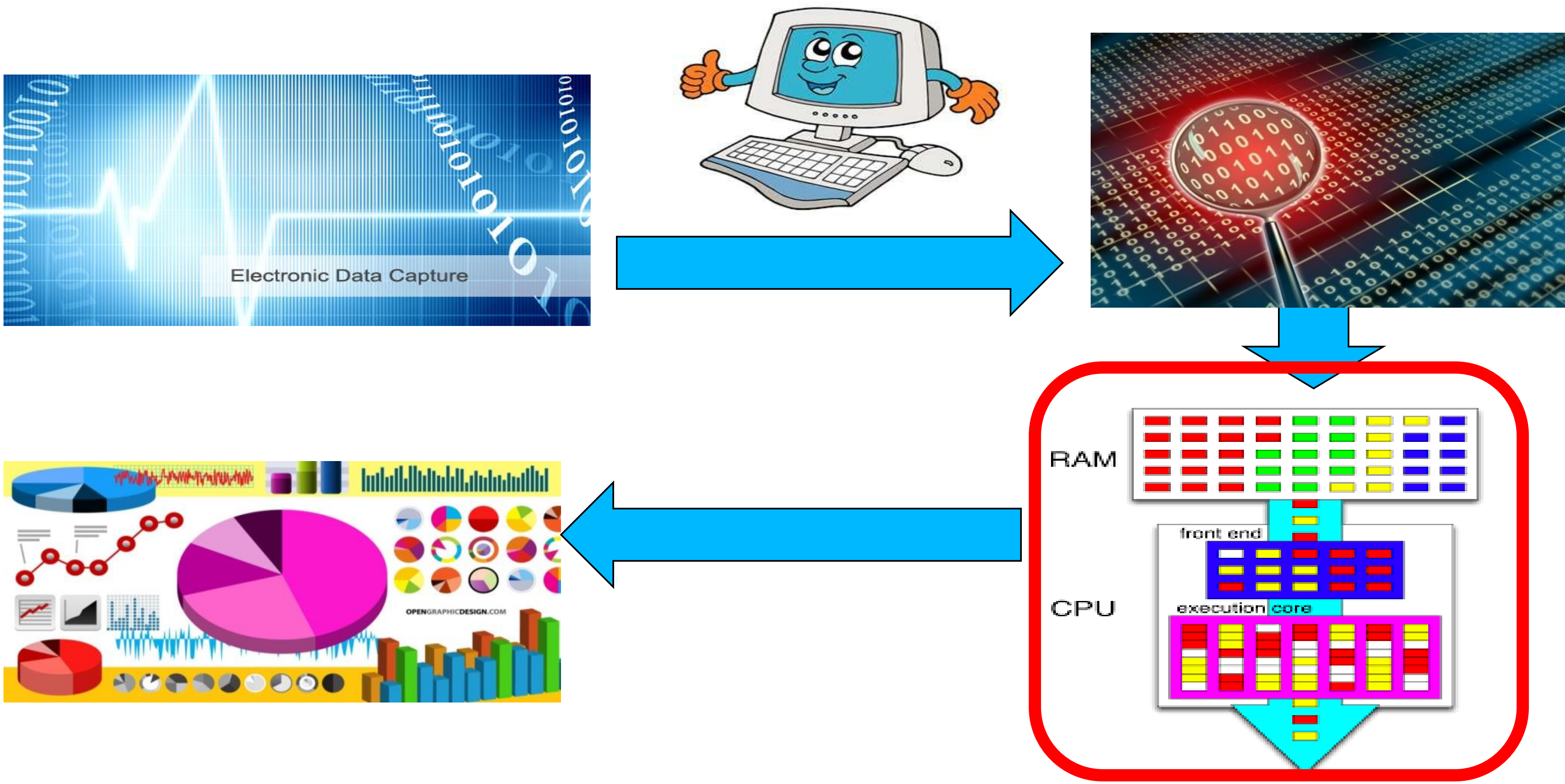
Department of Computer Science & Engineering

Indian Institute of Technology Guwahati, Assam.

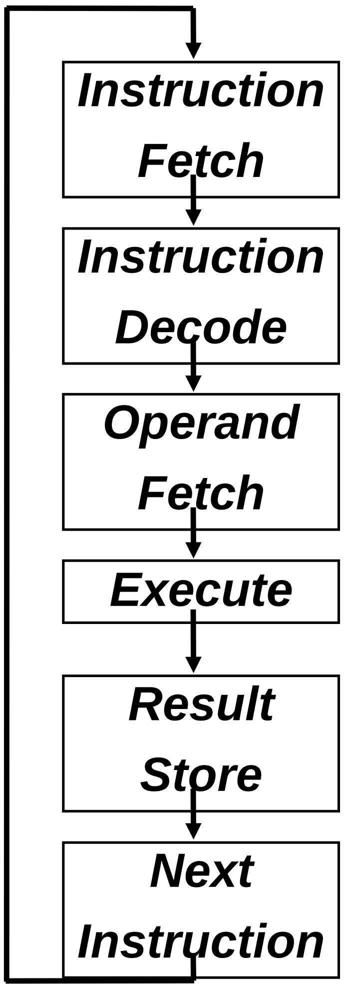
Smart Living Ahead

SmartPlanet

How is this all done ?



Execution Cycle



Obtain instruction from program storage

Determine required actions and instruction size

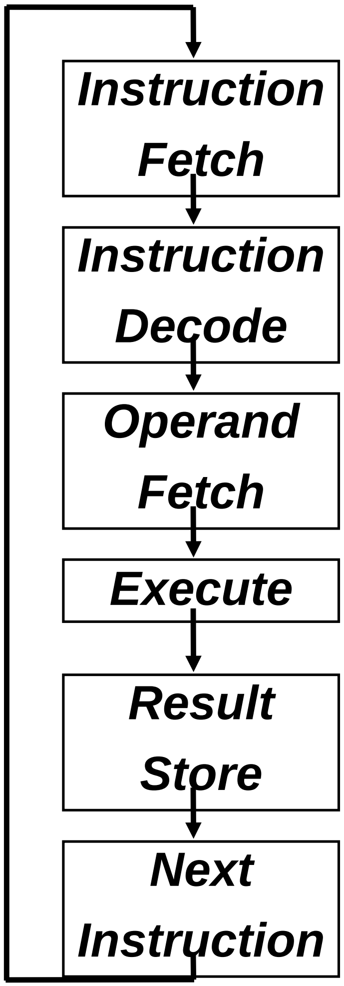
Locate and obtain operand data

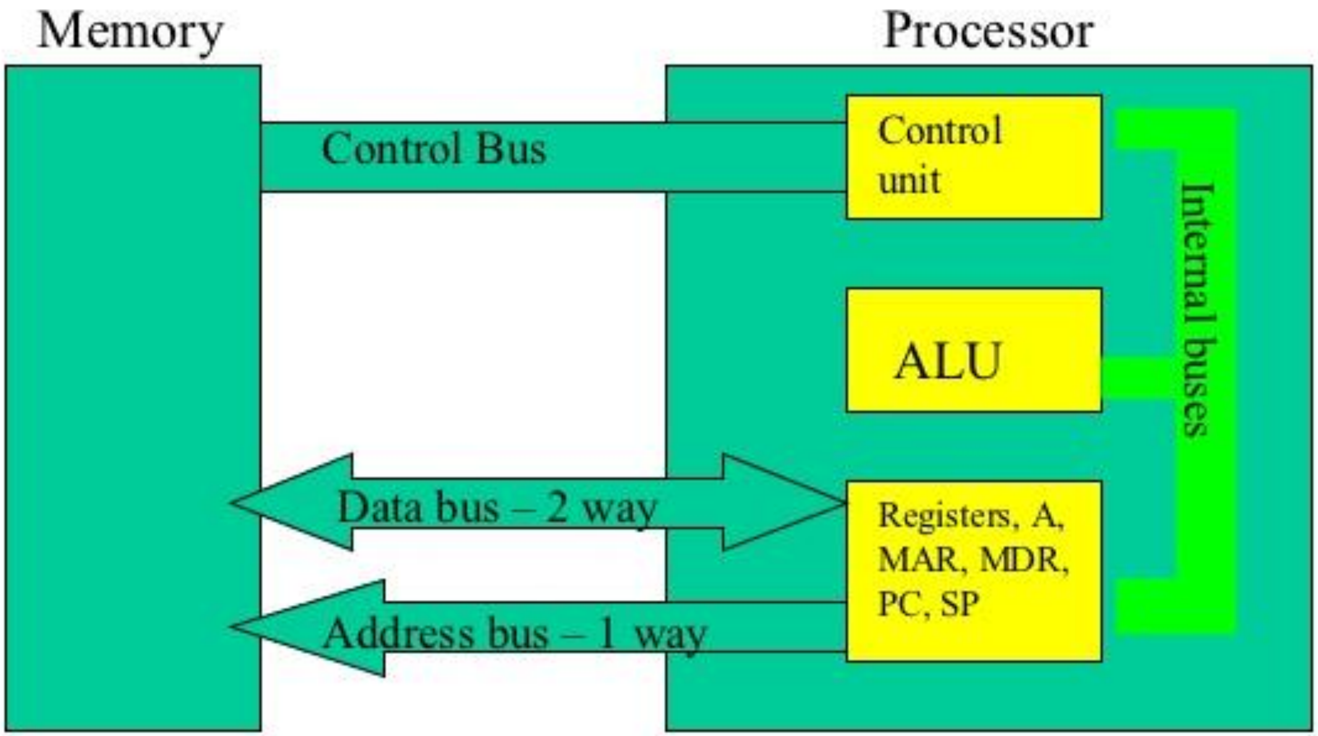
Compute result value or status

Deposit results in storage for later use

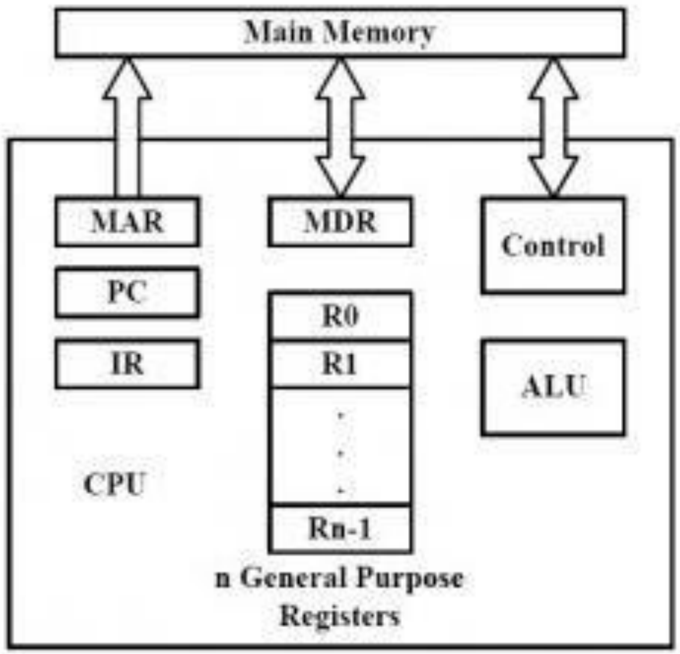
Determine successor instruction

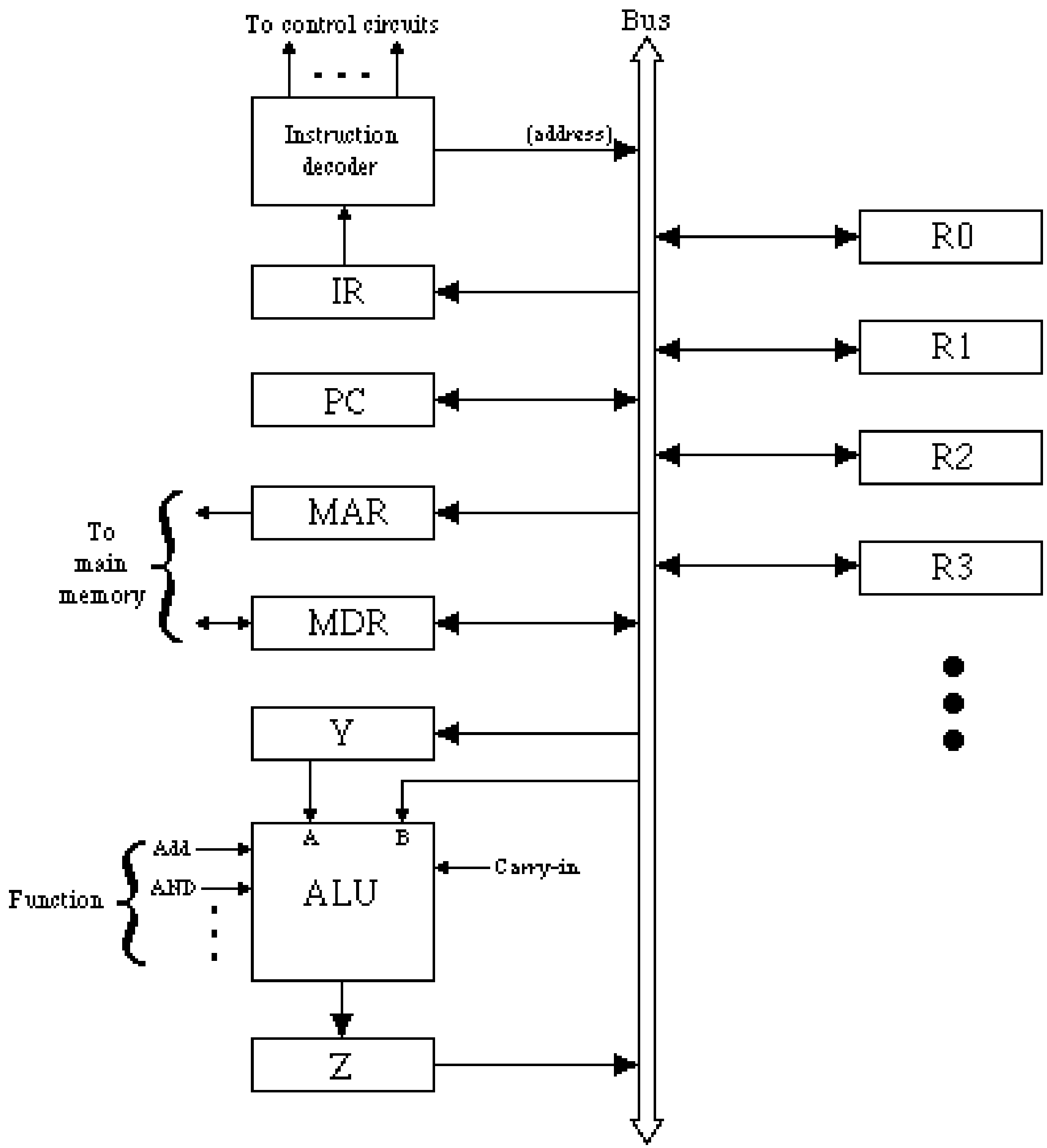
Processor Memory Interaction





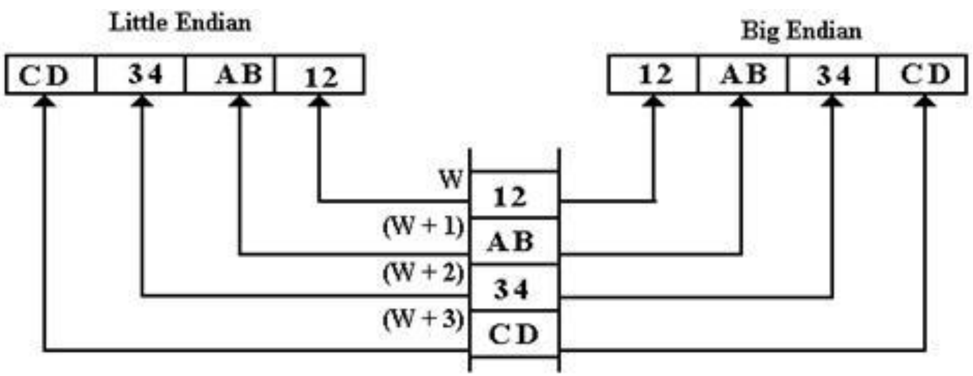
Processor Memory Interaction

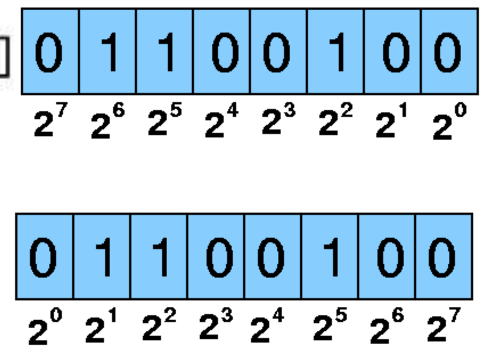




Byte Ordering

❖ Big Endian vs Little Endian

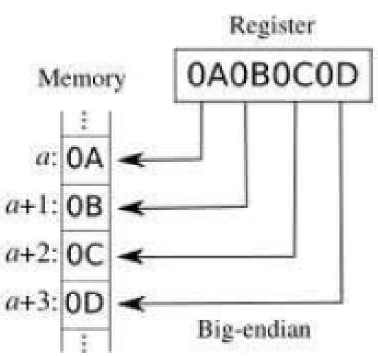


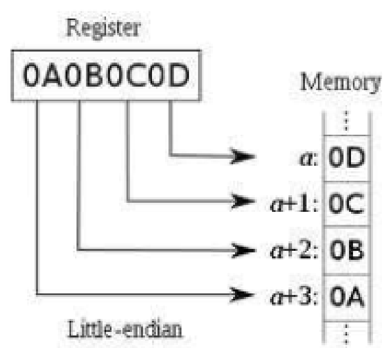


Big Endian
= 0x64 = 100

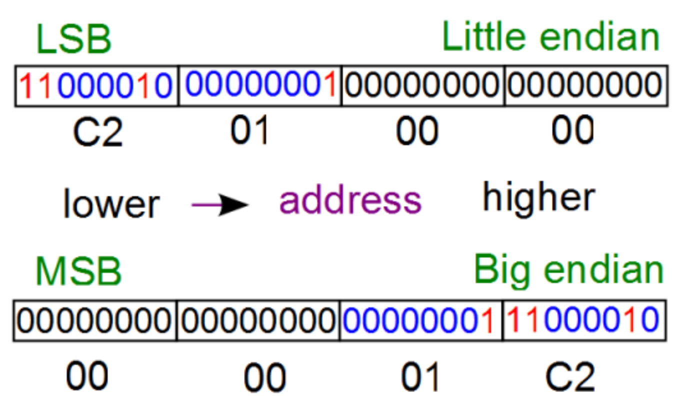
Little Endian
= 0x26 = 38

Big Endian vs. Little Endian

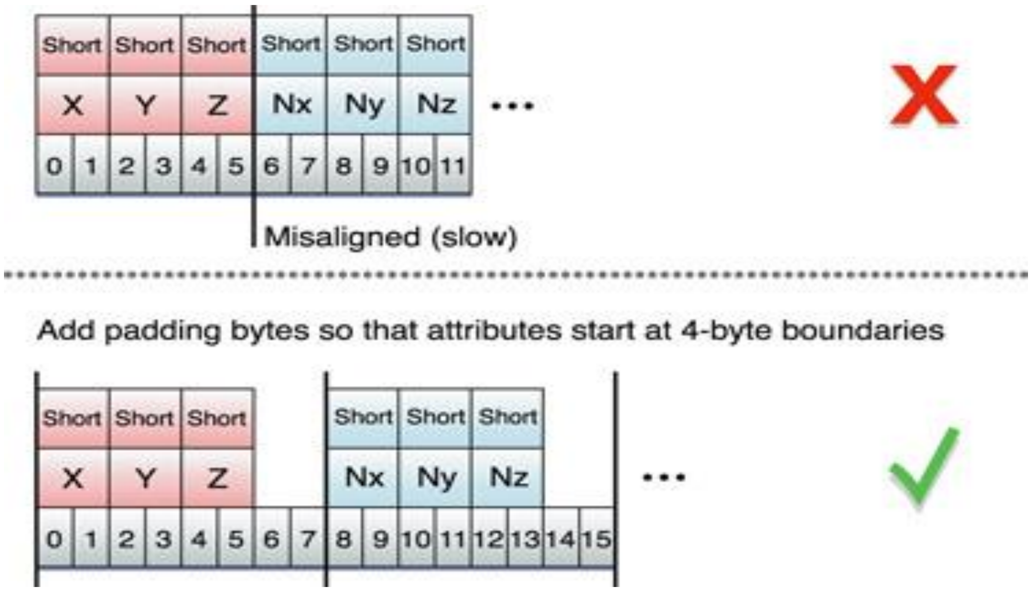


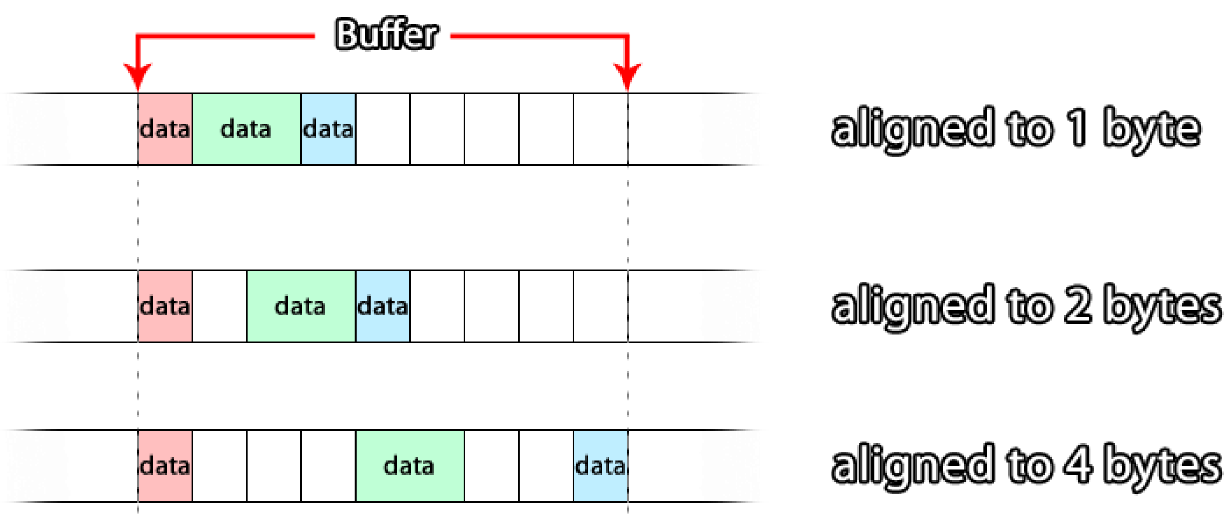


Int i = 450 = $2^8 + 2^7 + 2^6 + 2$ = x000001C2



Byte / Word Alignment





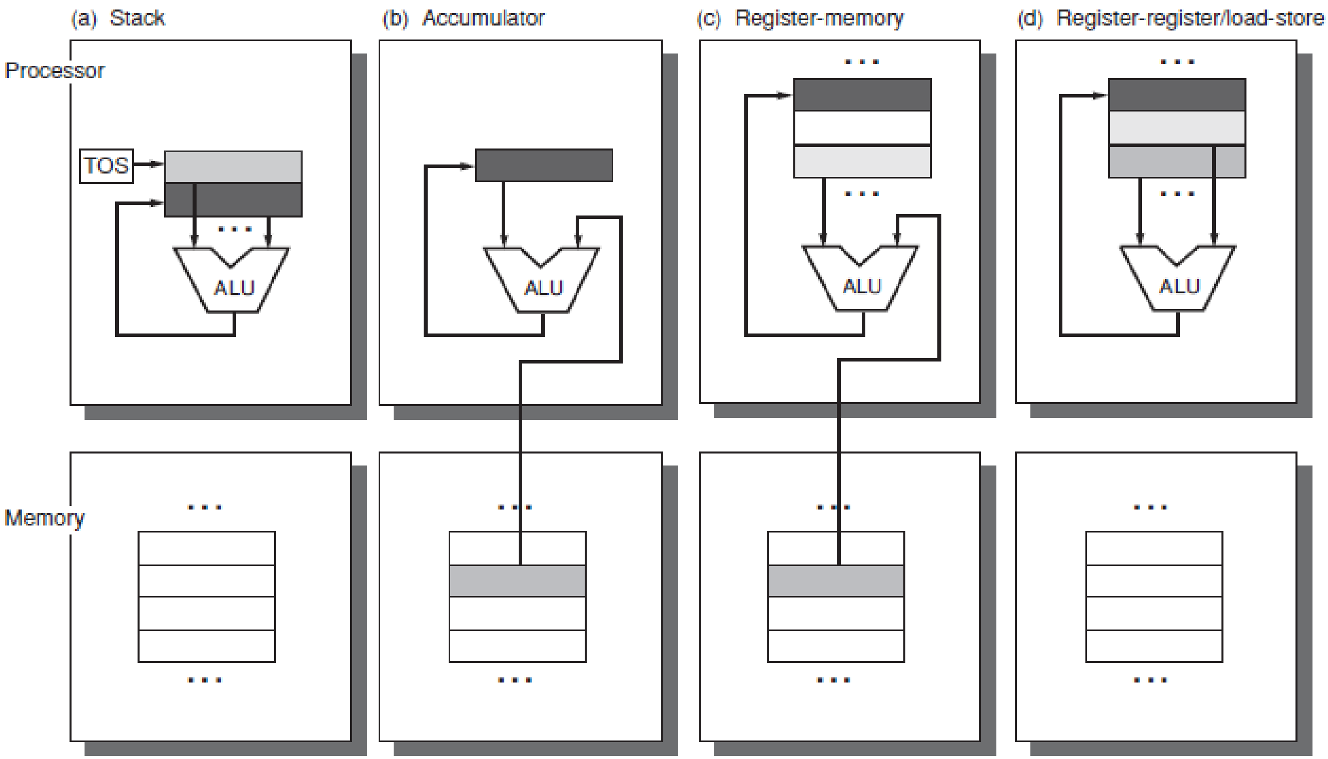
Byte /Word Alignment

<table>
<tr><th></th><th colspan="8">Value of 3 low-order bits of byte address</th></tr>
<tr><th>Width of object</th><th>0</th><th>1</th><th>2</th><th>3</th><th>4</th><th>5</th><th>6</th><th>7</th></tr>
<tr><td>1 byte (byte)</td><td>Aligned</td><td>Aligned</td><td>Aligned</td><td>Aligned</td><td>Aligned</td><td>Aligned</td><td>Aligned</td><td>Aligned</td></tr>
<tr><td>2 bytes (half word)</td><td colspan="2">Aligned</td><td colspan="2">Aligned</td><td colspan="2">Aligned</td><td colspan="2">Aligned</td></tr>
<tr><td>2 bytes (half word)</td><td></td><td colspan="2">Misaligned</td><td colspan="2">Misaligned</td><td colspan="2">Misaligned</td><td>Misaligned</td></tr>
<tr><td>4 bytes (word)</td><td colspan="4">Aligned</td><td colspan="4">Aligned</td></tr>
<tr><td>4 bytes (word)</td><td></td><td colspan="4">Misaligned</td><td colspan="3">Misaligned</td></tr>
<tr><td>4 bytes (word)</td><td colspan="2"></td><td colspan="4">Misaligned</td><td colspan="2">Misaligned</td></tr>
<tr><td>4 bytes (word)</td><td colspan="3"></td><td colspan="4">Misaligned</td><td>Misaligned</td></tr>
<tr><td>8 bytes (double word)</td><td colspan="8">Aligned</td></tr>
<tr><td>8 bytes (double word)</td><td></td><td colspan="7">Misaligned</td></tr>
<tr><td>8 bytes (double word)</td><td colspan="2"></td><td colspan="6">Misaligned</td></tr>
<tr><td>8 bytes (double word)</td><td colspan="3"></td><td colspan="5">Misaligned</td></tr>
<tr><td>8 bytes (double word)</td><td colspan="4"></td><td colspan="4">Misaligned</td></tr>
<tr><td>8 bytes (double word)</td><td colspan="5"></td><td colspan="3">Misaligned</td></tr>
<tr><td>8 bytes (double word)</td><td colspan="6"></td><td colspan="2">Misaligned</td></tr>
<tr><td>8 bytes (double word)</td><td colspan="7"></td><td>Misaligned</td></tr>
</table>

Instruction Set Architecture

- ❖ **Instruction vs Program vs Software**
- ❖ **Opcode, Operand**
- ❖ **Classification of ISA**
  - ❖ **Stack architecture**
  - ❖ **Accumulator architecture**
  - ❖ **Register-Memory architecture**
  - ❖ **Register-Register/Load Store architecture**

Instruction Set Architecture



Stack	Accumulator	Register (register-memory)	Register (load-store)
`Push A`	`Load A`	`Load R1,A`	`Load R1,A`
`Push B`	`Add B`	`Add R3,R1,B`	`Load R2,B`
`Add`	`Store C`	`Store R3,C`	`Add R3,R1,R2`
`Pop C`			`Store R3,C`

The code sequence for C = A + B for four classes of instruction sets.

CISC vs RISC architecture

	CISC (Complex Instruction Set Computer)	**RISC (Reduced Instruction Set Computer)**
1	Emphasis on hardware	Emphasis on software
2	Includes multi-clock complex instructions	Single-clock, reduced instruction only
3	Small code sizes	Typically larger code sizes
4	Many addressing modes	Few addressing modes.
5	An easy compiler design	A complex compiler design.
6	Pipelining does not function correctly here because of complexity in instructions.	Pipelining is not a major problem and this option speeds up the processors.

Measuring Performance

- **When can we say one computer / architecture design is better than others?**
  - **Desktop PC – (execution time of a program)**
  - **Server (transactions / unit time)**

- **When can we say X is n times faster than Y ?**
  - **Execution time$_Y$ / Execution time$_X$ =n**
  - **Throughput $_x$/ Trhoughput $_y$ =n**

Measuring Performance

- **Typical performance metrics:**
  - Response time
  - Throughput
- **Speedup of X relative to Y**
  - Execution time$_Y$ / Execution time$_X$
- **Execution time**
  - Wall clock time: includes all system overheads
  - CPU time: only computation time
- **Benchmarks**
  - Kernels (e.g. matrix multiply)
  - Toy programs (e.g. sorting)
  - Synthetic benchmarks (e.g. Dhrystone)
  - Benchmark suites (e.g. SPEC06, EEMBC, TPC-C)

Benchmark Suite

SPEC CPU2006 Programs

CINT2006 (Integer) — 12 programs

Benchmark	Language	Descriptions
400.perlbench	C	PERL Programming Language
401.bzip2	C	Compression
403.gcc	C	C Compiler
429.mcf	C	Combinatorial Optimization
445.gobmk	C	Artificial Intelligence: go
456.hmmer	C	Search Gene Sequence
458.sjeng	C	Artificial Intelligence: chess
462.libquantum	C	Physics: Quantum Computing
464.h264ref	C	Video Compression
471.omnetpp	C++	Discrete Event Simulation
473.astar	C++	Path-finding Algorithms
483.Xalancbmk	C++	XML Processing

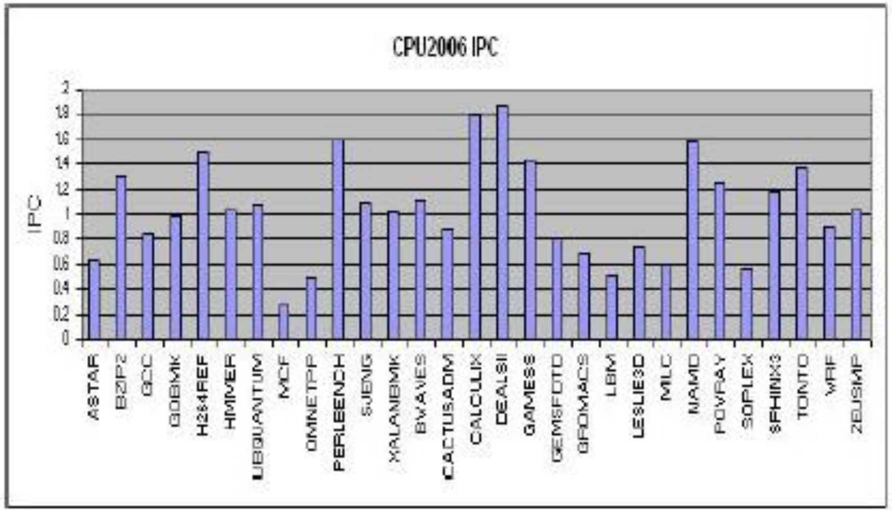
CFP2006 (Floating Point) — 17 programs

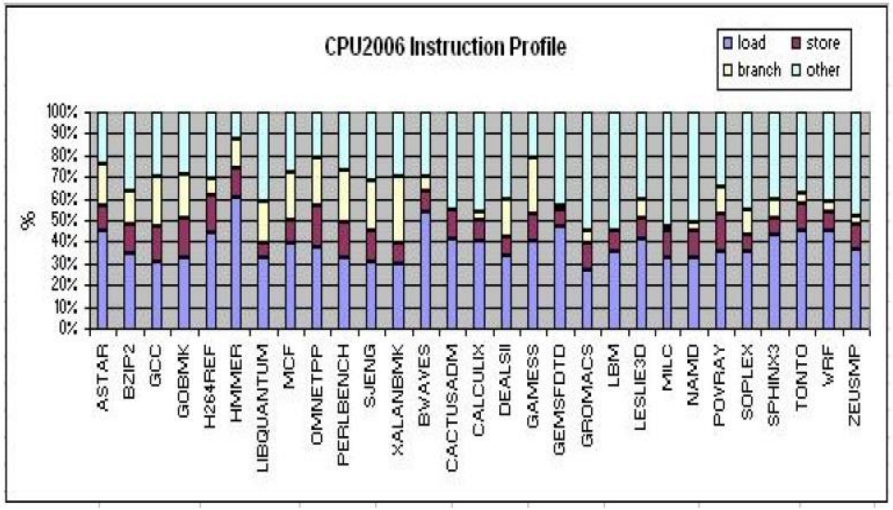
Benchmark	Language	Descriptions
410.bwaves	Fortran	Fluid Dynamics
416.gamess	Fortran	Quantum Chemistry
433.milc	C	Physics: Quantum Chromodynamics
434.zeusmp	Fortran	Physics/CFD
435.gromacs	C/Fortran	Biochemistry/Molecular Dynamics
436.cactusADM	C/Fortran	Physics/General Relativity
437.leslie3d	Fortran	Fluid Dynamics
444.namd	C++	Biology/Molecular Dynamics
447.dealII	C++	Finite Element Analysis
450.soplex	C++	Linear Programming, Optimization
453.povray	C++	Image Ray-tracing
454.calculix	C/Fortran	Structural Mechanics
459.GemsFDTD	Fortran	Computational Electromagnetics
465.tonto	Fortran	Quantum Chemistry
470.lbm	C	Fluid Dynamics
481.wrf	C/Fortran	Weather Prediction
482.sphinx3	C	Speech recognition

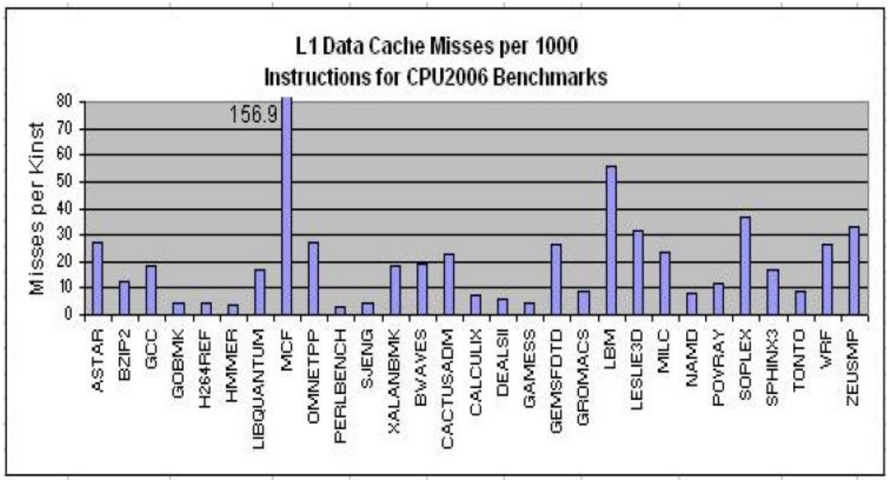
Target Programs application domain: Engineering and scientific computation

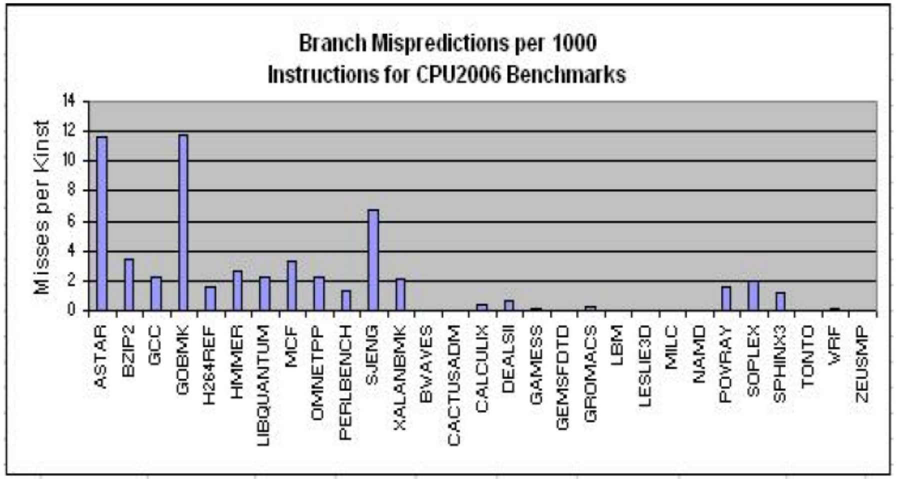
Source: http://www.spec.org/cpu2006/

Benchmark based evaluation









SPEC Ratio

$$\text{SPECRatio}_A = \frac{\text{Execution time}_{\text{reference}}}{\text{Execution time}_A}$$

$$\frac{\text{SPECRatio}_A}{\text{SPECRatio}_B} = \frac{\dfrac{\text{Execution time}_{\text{reference}}}{\text{Execution time}_A}}{\dfrac{\text{Execution time}_{\text{reference}}}{\text{Execution time}_B}} = \frac{\text{Execution time}_B}{\text{Execution time}_A} = \frac{\text{Performance}_A}{\text{Performance}_B}$$

$$GeometricMean\,(a_1, a_2, a_3, \ldots, a_N) = \sqrt[N]{\prod_{i}^{N} a_i}$$

johnjose@iitg.ac.in
http://www.iitg.ac.in/johnjose/