Abstractions, Trends and Performance

Image from: http://home.deib.polimi.it/mottola/

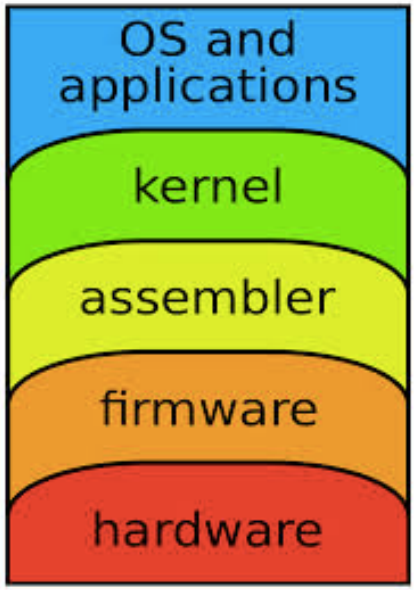
@carlosjaimebh

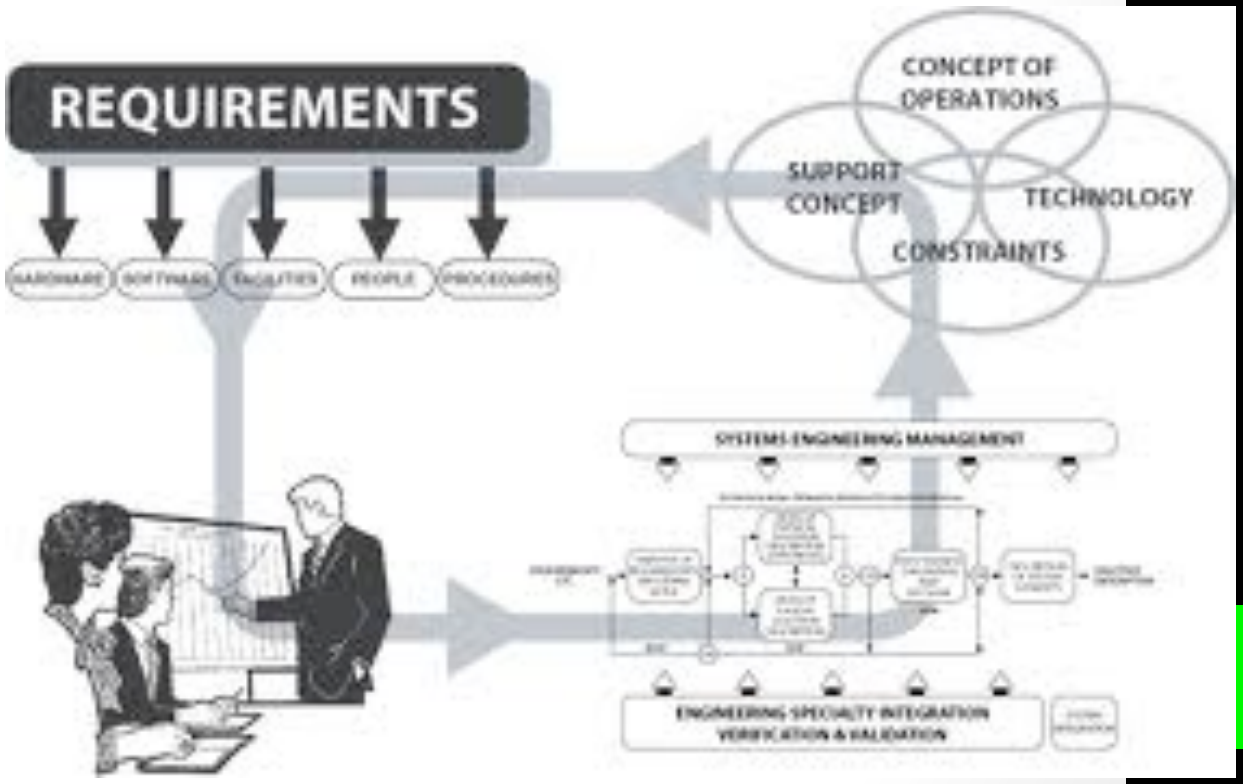
Designer/Architect of Computer Systems

- The task is a complex one: Determinate what attributes are important for a new computer:
  - MAXIMIZE PERFORMANCE
  - Energy Eficiency
  - Low Cost and Power
  - Availability

And the Role for a Systems Engineer?

- Take decisions, suggest to acquire or use new techology in accordance with the previous attributes and requirements.



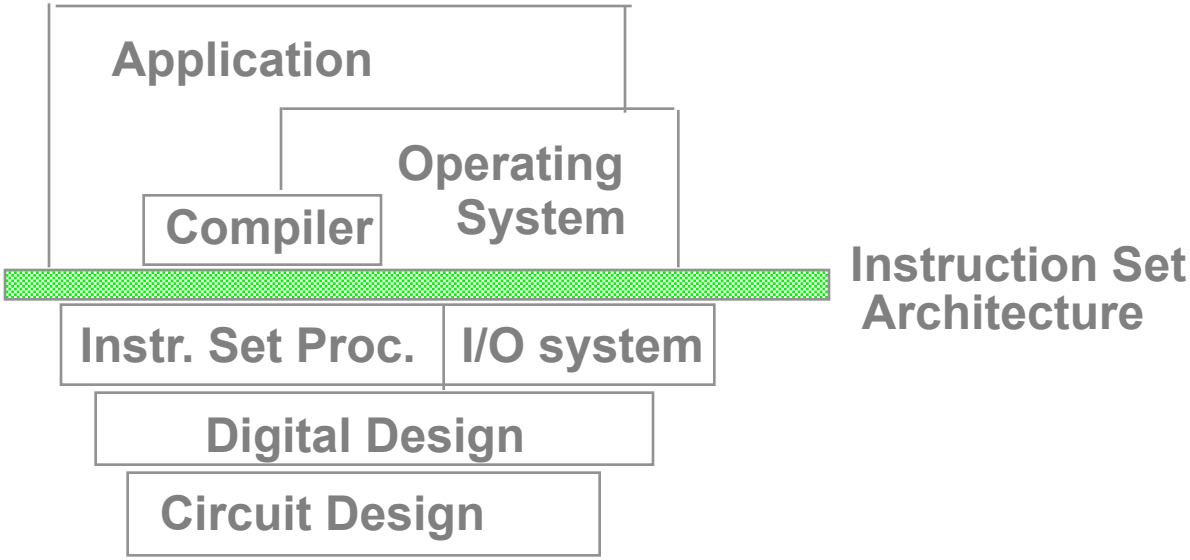


Abstraction

- **Abstraction** is a process by which concepts are derived from the usage and classification of literal ("real" or "concrete") concepts, first principles, or other methods.
- Abstractions may be formed by reducing the information content of a concept or an observable phenomenon, typically to retain only information which is relevant for a particular purpose.

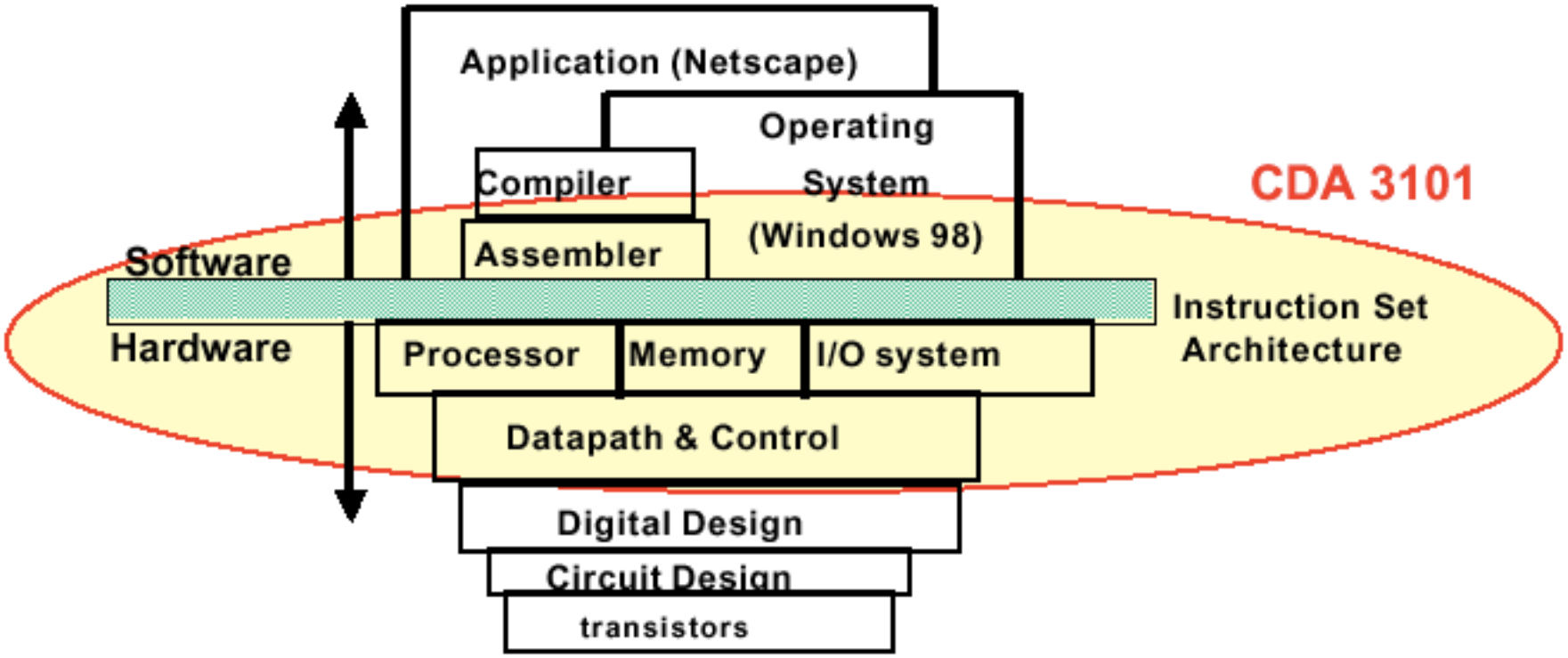
Instruction Set Architecture (ISA)

° Co-ordination of *levels of abstraction*



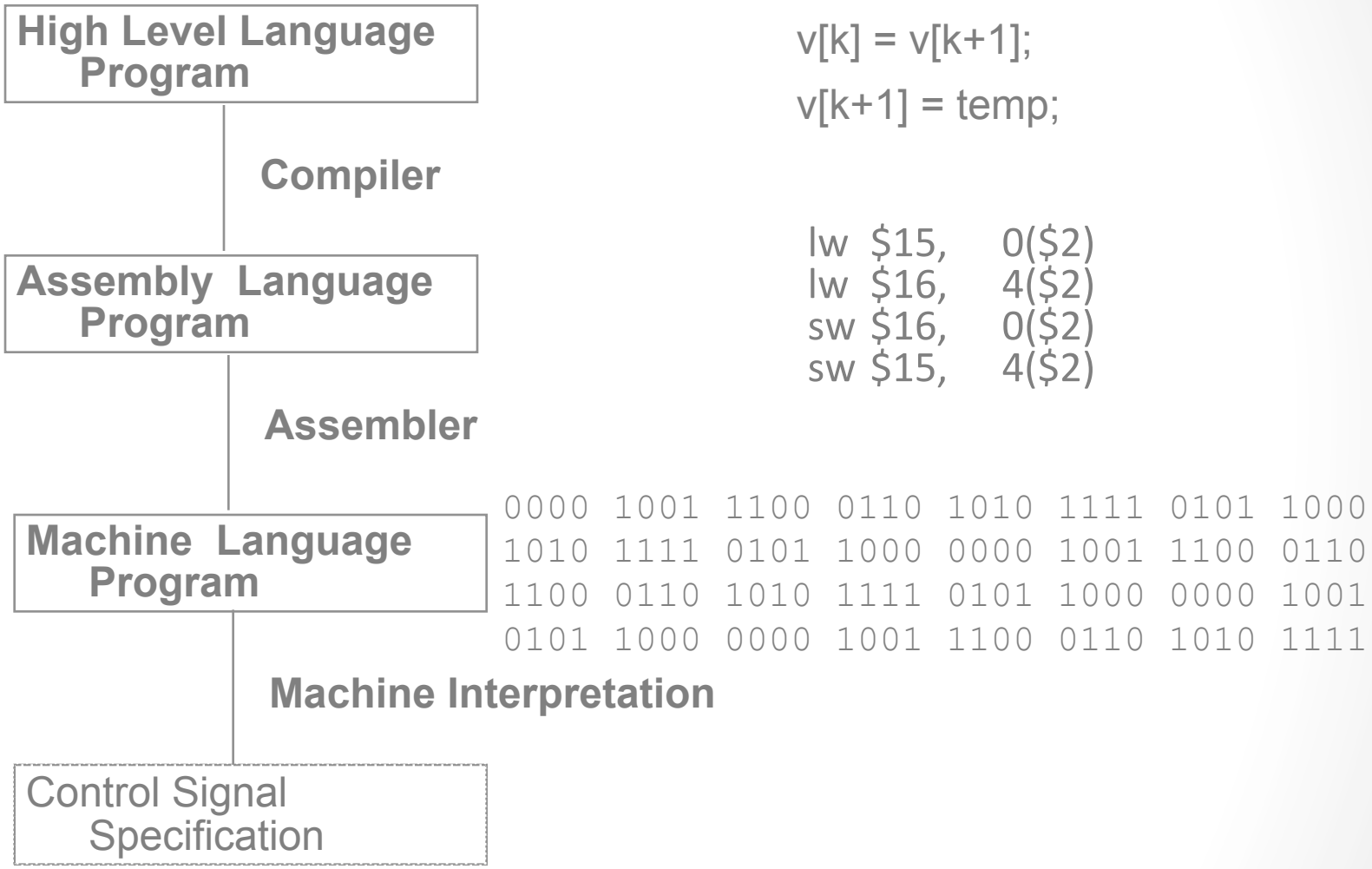
° Under a set of rapidly changing *Forces*

ISA Levels



http://www.cise.ufl.edu/~mssz/CompOrg/CDAintro.html

Review: Levels of Representation

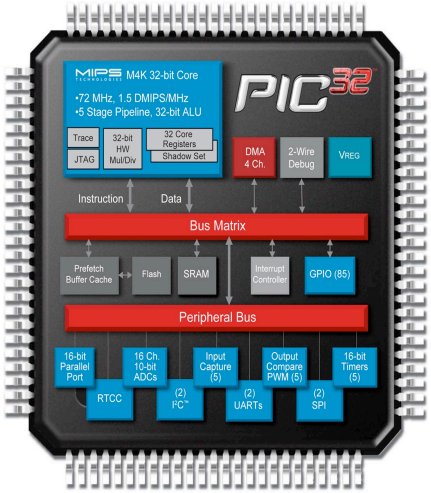


80X86 Architecture

- X86 denotes a family of ISAs based on the Intel® 8086CPU.

MIPS Architecture

- Microprocessor without Interlocked Pipeline Stages is a RISC_ISA computer developed by MIPS Technologies, formerly used in Embedded Systems or video game consoles (Sony® PlayStation®) .



Some Aspects of ISA

1. Class of ISA
2. Memory Addressing
3. Addresing Modes
4. Types of Sizes of Operands
5. Operations
6. Control Flow Instructions
7. Encoding and ISA

1. Class of ISA

- All ISA are classified as general-purpose register architecture (Operands are either registers or memory locations)
  - 80x86 has a 16 general-purpose registers (16 floating-point data)
  - MIPS has 32 general-purpose registers (32 floating-point reigisters)
- Two Popular versions:
  - Register Memory ISAs (80x86)
  - Load Store ISAs (ARM, MIPS)

1. Class of ISA: An Example

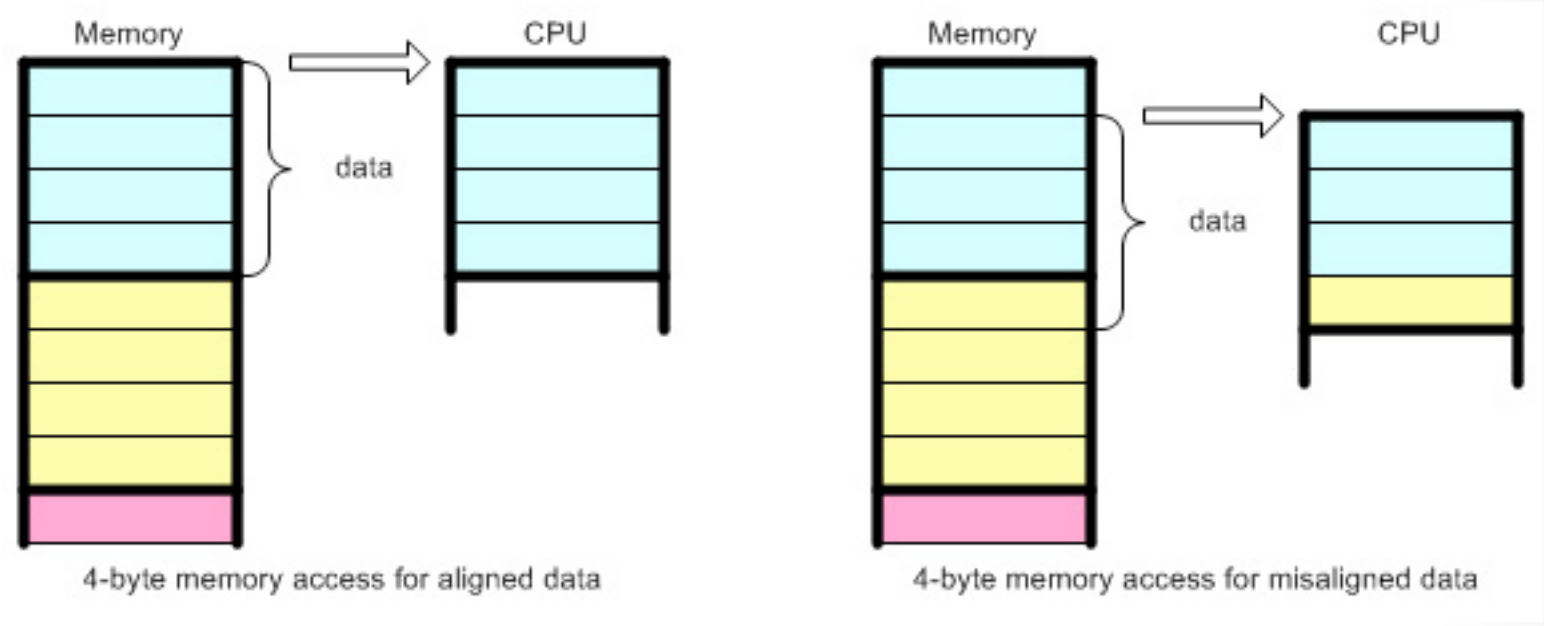
MIPS registers and usage conventions

Name	Register Number	Usage	Preserved on call
$zero	0	the constant value 0	n.a.
$at	1	reserved for the assembler	n.a.
$v0-$v1	2-3	value for results and expressions	no
$a0-$a3	4-7	arguments (procedures/functions)	yes
$t0-$t7	8-15	temporaries	no
$s0-$s7	16-23	saved	yes
$t8-$t9	24-25	more temporaries	no
$k0-$k1	26-27	reserved for the operating system	n.a.
$gp	28	global pointer	yes
$sp	29	stack pointer	yes
$fp	30	frame pointer	yes
$ra	31	return address	yes

2. Memory Addressing

- All desktop and servers computers use byte addressing to access memory operands.
  - Some architectures (ARMS, MIPS) require that objects must be aligned.
  - 80x86 does not require alignement, but accesses are generally faster if operands are aligned

2. Memory Addressing
Aligned and Missalinged examples



3. Addressing Modes

- Addressing Modes specify the address of a memory object, registers and constant operands.
  - MIPS: Register, Immediate (Constants), Displacement.
  - 80x86: Support the previous three + three variations of displacement:

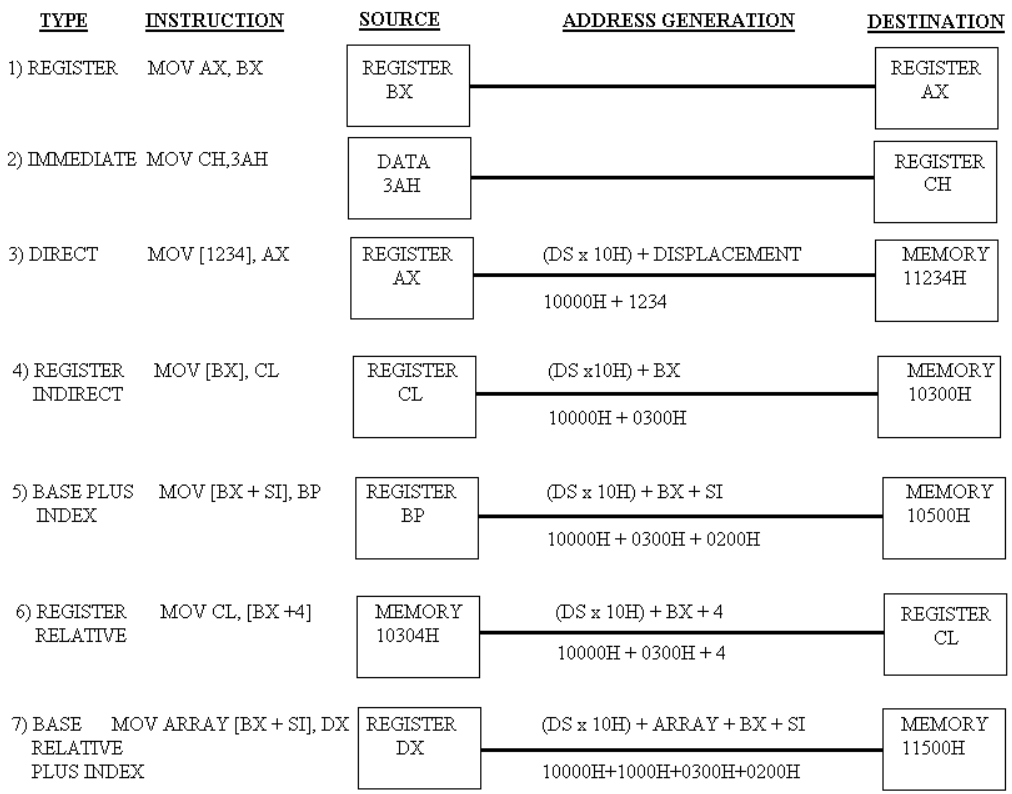
    no register (absolute)

    two registers (based indexed with displacement)

    two registers (based with scaled index and displacement)
  - ARM: The three MIPS registers + PC-Relative addressing, the sum of two registers and the sum of two registers multiplied by the size of the operand in bytes.

3. Addressing Modes : An Example

8086 ADDRESS MODES



ASSUME: BX = 0300H; SI = 0200H; ARRAY = 1000H; DS = 1000H.

4. Types and Sizes of Operands

- 8-bit (ASCII character)
- 16-bit (Unicode Character or half word)
- 32-bit (Integer or word)
- 64-bit (Double word or long integer)
- IEE 754 Floating Point in 32-bit (Single Precision) and 64-bit (Double precision)
- 80-bit Floating Point (Extended Double Precision Only for 80x86)

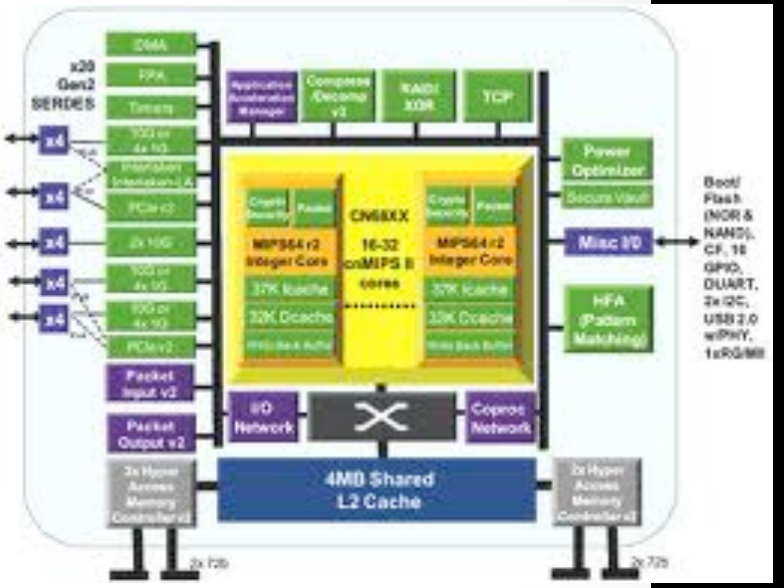
5. Operations

- Data Transfer
  - Moves data between registers and memory, or between the integer and special registers.
- Arithmetic Logical
  - Operations on Integer or Logical data in GPRs (General Purpose Registers)
- Control
  - Conditional branches and jumps
- Floating Point
  - Floating Point Operations on Double Precision and Single Precisions Formats

5. Operations:

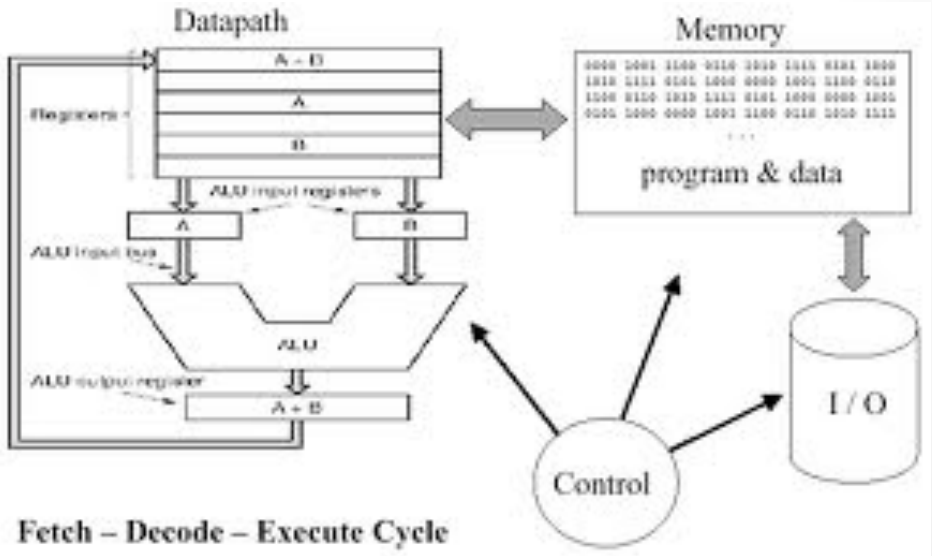
Subset of the Instructions in MIPS64

Instruction type/opcode	Instruction meaning
Data transfers	*Move data between registers and memory, or between the integer and FP or special registers; only memory address mode is 16-bit displacement + contents of a GPR*
LB,LBU,SB	Load byte, load byte unsigned, store byte (to/from integer registers)
LH,LHU,SH	Load half word, load half word unsigned, store half word (to/from integer registers)
LW,LWU,SW	Load word, load word unsigned, store word (to/from integer registers)
LD,SD	Load double word, store double word (to/from integer registers)
L.S,L.D,S.S,S.D	Load SP float, load DP float, store SP float, store DP float
MFC0,MTC0	Copy from/to GPR to/from a special register
MOV.S,MOV.D	Copy one SP or DP FP register to another FP register
MFC1,MTC1	Copy 32 bits to/from FP registers from/to integer registers
Arithmetic/logical	*Operations on integer or logical data in GPRs; signed arithmetic trap on overflow*
DADD,DADDI,DADDU,DADDIU	Add, add immediate (all immediates are 16 bits); signed and unsigned
DSUB,DSUBU	Subtract; signed and unsigned
DMUL,DMULU,DDIV, DDIVU,MADD	Multiply and divide, signed and unsigned; multiply-add; all operations take and yield 64-bit values
AND,ANDI	And, and immediate
OR,ORI,XOR,XORI	Or, or immediate, exclusive or, exclusive or immediate
LUI	Load upper immediate; loads bits 32 to 47 of register with immediate, then sign-extends
DSLL,DSRL,DSRA,DSLLV, DSRLV,DSRAV	Shifts: both immediate (DS__) and variable form (DS__V); shifts are shift left logical, right logical, right arithmetic
SLT,SLTI,SLTU,SLTIU	Set less than, set less than immediate; signed and unsigned
Control	*Conditional branches and jumps; PC-relative or through register*
BEQZ,BNEZ	Branch GPRs equal/not equal to zero; 16-bit offset from PC + 4
BEQ,BNE	Branch GPR equal/not equal; 16-bit offset from PC + 4
BC1T,BC1F	Test comparison bit in the FP status register and branch; 16-bit offset from PC + 4
MOVN,MOVZ	Copy GPR to another GPR if third GPR is negative, zero
J,JR	Jumps: 26-bit offset from PC + 4 (J) or target in register (JR)
JAL,JALR	Jump and link: save PC + 4 in R31, target is PC-relative (JAL) or a register (JALR)
TRAP	Transfer to operating system at a vectored address
ERET	Return to user code from an exception; restore user mode
Floating point	*FP operations on DP and SP formats*
ADD.D,ADD.S,ADD.PS	Add DP, SP numbers, and pairs of SP numbers
SUB.D,SUB.S,SUB.PS	Subtract DP, SP numbers, and pairs of SP numbers
MUL.D,MUL.S,MUL.PS	Multiply DP, SP floating point, and pairs of SP numbers
MADD.D,MADD.S,MADD.PS	Multiply-add DP, SP numbers, and pairs of SP numbers
DIV.D,DIV.S,DIV.PS	Divide DP, SP floating point, and pairs of SP numbers
CVT._._	Convert instructions: CVT.x.y converts from type x to type y, where x and y are L (64-bit integer), W (32-bit integer), D (DP), or S (SP). Both operands are FPRs.
C.__.D,C.__.S	DP and SP compares: "__" = LT,GT,LE,GE,EQ,NE; sets bit in FP status register



6. Control Flow Instructions

- Conditional Branches
- Unconditional Jumps
- Procedure Calls
- Returns



Fetch – Decode – Execute Cycle

7. Encoding an ISA

- Fixed Length
  - ARM and MIPS (32-bits long)
- Variable Length
  - 80x86 (Ranking from 1 to 18 bytes)

- I.E. MIPS instruction encoding formats:
  - R-type (6-bit opcode, 5-bit rs, 5-bit rt, 5-bit rd, 5-bit shamt, 6-bit function code)

31-26	25-21	20-16	15-11	10-6	5-0
opcode	*rs*	*rt*	*rd*	shamt	function

  - I-type (6-bit opcode, 5-bit rs, 5-bit rt, 16-bit immediate)

31-26	25-21	20-16	15-0
opcode	*rs*	*rt*	*imm*

  - J-type (6-bit opcode, 26-bit pseudo-direct address)

31-26	25-0
opcode	*pseudodirect jump address*

Designing and Organization

- Implementation of a computer has two components: organization and hardware.
  - Organization: High level aspects (memory system, memory interconection, design of the CPU)

    Microarchitecture

    - Example: two processors with the same ISA but different organization are AMD Opteron adn Intel Core i7.
  - Hardware: Specifics of a Computer (Detailed logic design and the packaging technology of the computer)

Summary of Some Functional Requerimients an Architect Faces

Functional requirements	Typical features required or supported
Application area	*Target of computer*
Personal mobile device	Real-time performance for a range of tasks, including interactive performance for graphics, video, and audio; energy efficiency (Ch. 2, 3, 4, 5; App. A)
General-purpose desktop	Balanced performance for a range of tasks, including interactive performance for graphics, video, and audio (Ch. 2, 3, 4, 5; App. A)
Servers	Support for databases and transaction processing; enhancements for reliability and availability; support for scalability (Ch. 2, 5; App. A, D, F)
Clusters/warehouse-scale computers	Throughput performance for many independent tasks; error correction for memory; energy proportionality (Ch 2, 6; App. F)
Embedded computing	Often requires special support for graphics or video (or other application-specific extension); power limitations and power control may be required; real-time constraints (Ch. 2, 3, 5; App. A, E)
Level of software compatibility	*Determines amount of existing software for computer*
At programming language	Most flexible for designer; need new compiler (Ch. 3, 5; App. A)
Object code or binary compatible	Instruction set architecture is completely defined—little flexibility—but no investment needed in software or porting programs (App. A)
Operating system requirements	*Necessary features to support chosen OS (Ch. 2; App. B)*
Size of address space	Very important feature (Ch. 2); may limit applications
Memory management	Required for modern OS; may be paged or segmented (Ch. 2)
Protection	Different OS and application needs: page vs. segment; virtual machines (Ch. 2)
Standards	*Certain standards may be required by marketplace*
Floating point	Format and arithmetic: IEEE 754 standard (App. J), special arithmetic for graphics or signal processing
I/O interfaces	For I/O devices: Serial ATA, Serial Attached SCSI, PCI Express (App. D, F)
Operating systems	UNIX, Windows, Linux, CISCO IOS
Networks	Support required for different networks: Ethernet, Infiniband (App. F)
Programming languages	Languages (ANSI C, C++, Java, Fortran) affect instruction set (App. A)

From Hennesy and Patterson

Trends in Technology

- Integrated Circuit Logic Technology
  - Transistor density increases by about 35%/ year, quadrupling somewhat over four years (Moore's Law)
- Semiconductor DRAM

CA:AQA Edition	Year	DRAM growth rate	Characterization of impact on DRAM capacity
1	1990	60%/year	Quadrupling every 3 years
2	1996	60%/year	Quadrupling every 3 years
3	2003	40%–60%/year	Quadrupling every 3 to 4 years
4	2007	40%/year	Doubling every 2 years
5	2011	25%–40%/year	Doubling every 2 to 3 years

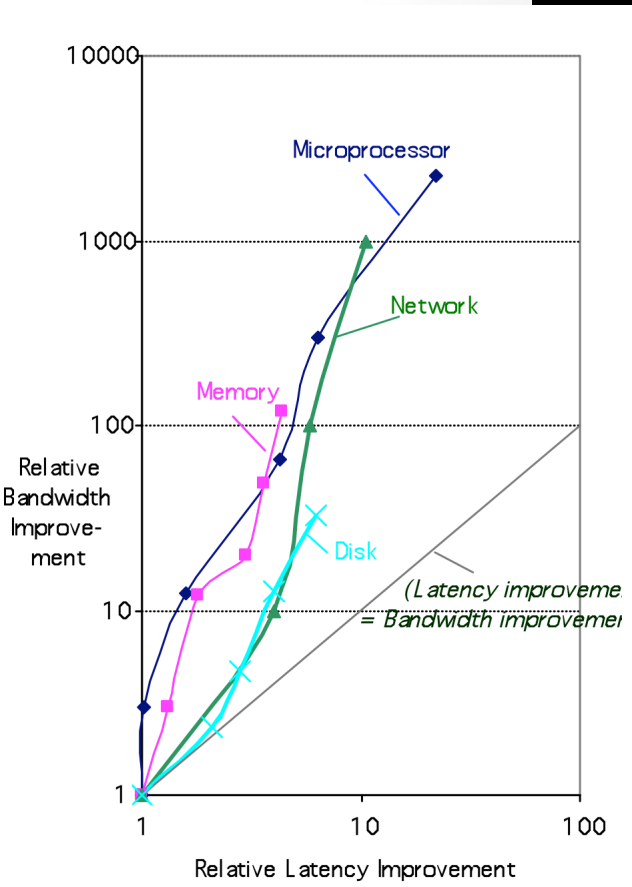
Trends in Technology

- Semiconductor Flash
  - Electrically erasable programmable read-only memory (Flash Memory)
- Magnetic Disk Technology
  - Disks are 15 to 25 times cheaper per bit than flash.
- Network Technology
  - Network performance depends both on the performance of switches and on the performance of the transmission system.

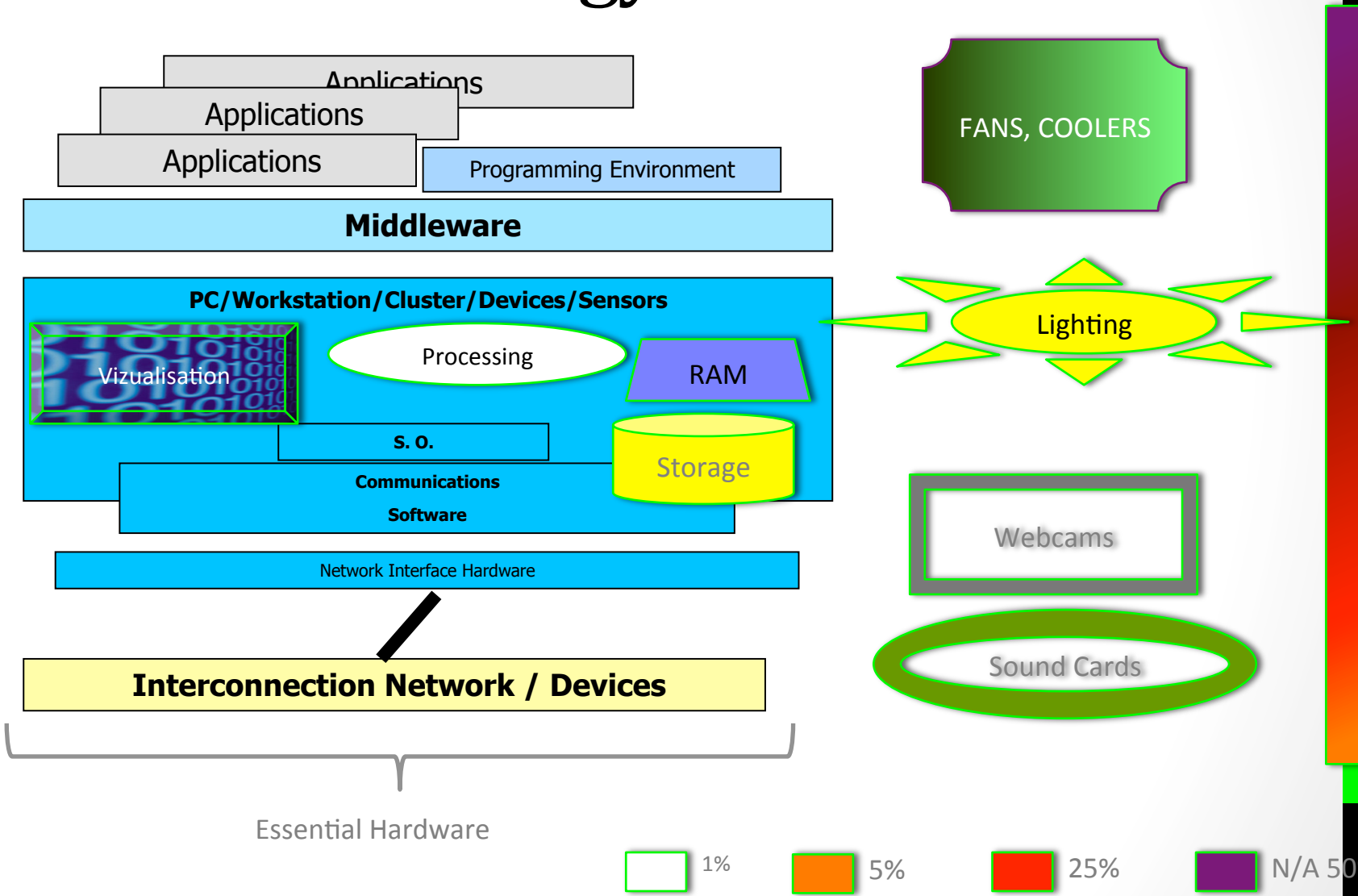
Performance Trends:

Bandwidth over Latency

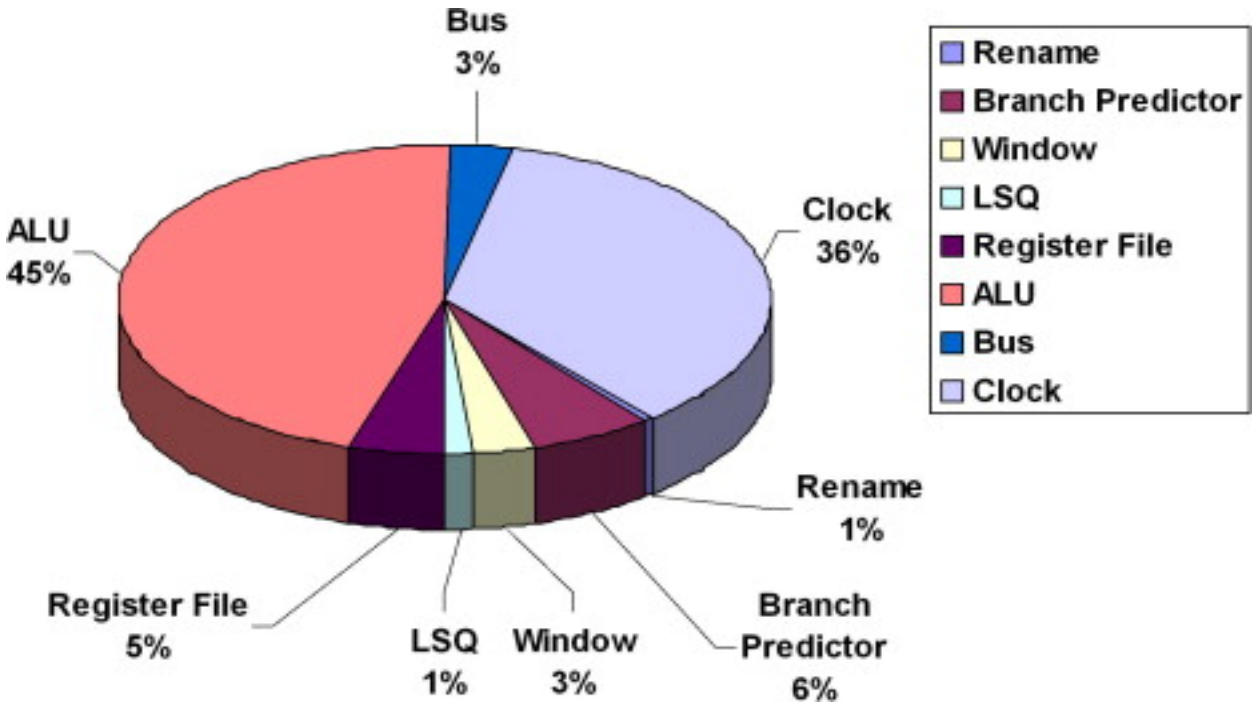
Microprocessor	16-bit address/ bus, microcoded	32-bit address/ bus, microcoded	5-stage pipeline, on-chip I & D caches, FPU	2-way superscalar, 64-bit bus	Out-of-order 3-way superscalar	Out-of-order superpipelined, on-chip L2 cache	Multicore OOO 4-way on chip L3 cache, Turbo
Product	Intel 80286	Intel 80386	Intel 80486	Intel Pentium	Intel Pentium Pro	Intel Pentium 4	Intel Core i7
Year	1982	1985	1989	1993	1997	2001	2010
Die size (mm^2)	47	43	81	90	308	217	240
Transistors	134,000	275,000	1,200,000	3,100,000	5,500,000	42,000,000	1,170,000,000
Processors/chip	1	1	1	1	1	1	4
Pins	68	132	168	273	387	423	1366
Latency (clocks)	6	5	5	5	10	22	14
Bus width (bits)	16	32	32	64	64	64	196
Clock rate (MHz)	12.5	16	25	66	200	1500	3333
Bandwidth (MIPS)	2	6	25	132	600	4500	50,000
Latency (ns)	320	313	200	76	50	15	4
Memory module	DRAM	Page mode DRAM	Fast page mode DRAM	Fast page mode DRAM	Synchronous DRAM	Double data rate SDRAM	DDR3 SDRAM
Module width (bits)	16	16	32	64	64	64	64
Year	1980	1983	1986	1993	1997	2000	2010
Mbits/DRAM chip	0.06	0.25	1	16	64	256	2048
Die size (mm^2)	35	45	70	130	170	204	50
Pins/DRAM chip	16	16	18	20	54	66	134
Bandwidth (MBytes/s)	13	40	160	267	640	1600	16,000
Latency (ns)	225	170	125	75	62	52	37
Local area network	Ethernet	Fast Ethernet	Gigabit Ethernet	10 Gigabit Ethernet	100 Gigabit Ethernet		
IEEE standard	802.3	803.3u	802.3ab	802.3ac	802.3ba		
Year	1978	1995	1999	2003	2010		
Bandwidth (Mbits/sec)	10	100	1000	10,000	100,000		
Latency (µsec)	3000	500	340	190	100		
Hard disk	3600 RPM	5400 RPM	7200 RPM	10,000 RPM	15,000 RPM	15,000 RPM	
Product	CDC Wren I 94145-36	Seagate ST41600	Seagate ST15150	Seagate ST39102	Seagate ST373453	Seagate ST3600057	
Year	1983	1990	1994	1998	2003	2010	
Capacity (GB)	0.03	1.4	4.3	9.1	73.4	600	
Disk form factor	5.25 inch	5.25 inch	3.5 inch	3.5 inch	3.5 inch	3.5 inch	
Media diameter	5.25 inch	5.25 inch	3.5 inch	3.0 inch	2.5 inch	2.5 inch	
Interface	ST-412	SCSI	SCSI	SCSI	SCSI	SAS	
Bandwidth (MBytes/s)	0.6	4	9	24	86	204	
Latency (ms)	48.3	17.1	12.7	8.8	5.7	3.6	



Performance Trends Power and Energy



Performance Trends Power and Energy



The distribution of the energy consumed in the processor components.

From Nikolaos Kroupis , Dimitrios Soudris, **FILESPPA: Fast Instruction Level Embedded System Power and Performance Analyzer**

Microprocessors and Microsystems Volume 35, Issue 3 2011 329 - 342

Performance Trends
Power and Energy:

Techniques to improve energy efficiency

1. Do nothing well: Turn off the clock of inactive modules to save energy and dynamic power.
2. Dynamic Voltage-Frequency Scaling (DVFS)
3. Design for Typical Case (Scheduling of Activity)
4. Overclocking (Turbo Mode)

Trends in Cost

- Impact of Time, Volume and Commoditization
  - Learning Curve
  - Manufacturing Costs
  - Transport
  - Market (Cost vs Price)
  - Operation Costs
  - **Dependability**
    - **Service Level Agreements (Infrastructure)**
    - **Service Level Objects (Networking, Power)**

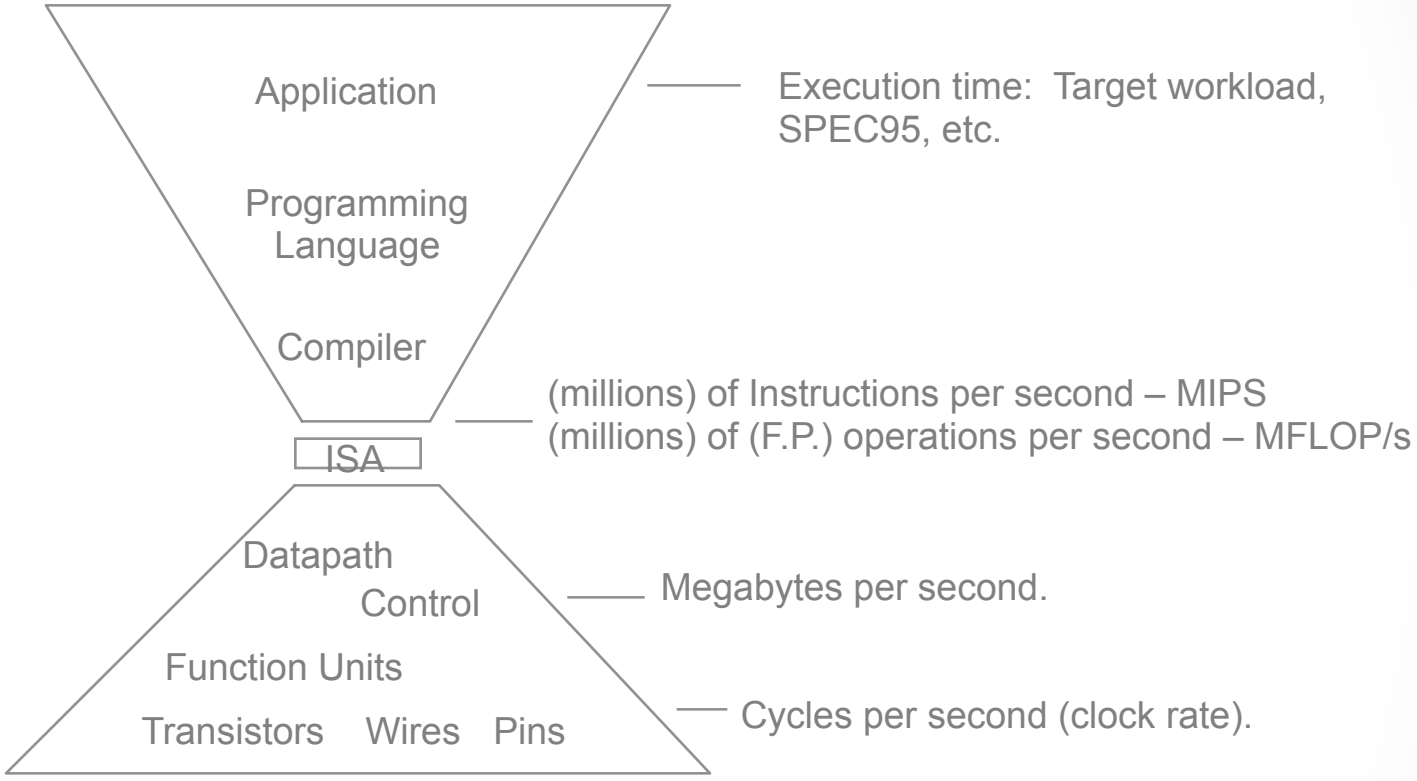
Performance

- A Computer System exists to IMPROVE PERFORMANCE
  - High Speed
  - Data Treatment
  - Availability
  - Capacity
  - Low Latency
  - High Bandwdith

Measuring Performance

- In order to compare how fast computers can process data, we have to measure their performance.
- There are a number of measurements of performance.
- Clock speed, MIPS, FLOPS & Benchmark tests are all used. Some are a better measure than others.

Metrics of Computer Performance



Each metric has a purpose, and each can be misused.

Computer Performance

- *Response Time* (*elapsed time, latency*):
  - how long does it take for *my* job to run?
  - how long does it take to execute (start to finish) *my* job?
  - how long must *I* wait for the database query?
- *Throughput*:
  - how *many* jobs can the machine run at once?
  - what is the *average* execution rate?
  - how *much* work is getting done?

Individual user concerns…

Systems manager concerns…

- *If we upgrade a machine with a new processor what do we increase?*
- *If we add a new machine to the lab what do we increase?*

Execution Time

- *Elapsed Time*
  - counts everything (*disk and memory accesses, waiting for I/O, running other programs, etc.*) from start to finish
  - a useful number, but often not good for comparison purposes

    elapsed time = CPU time + wait time (I/O, other programs, etc.)

- *CPU time*
  - doesn't count waiting for I/O or time spent running other programs
  - can be divided into *user CPU time* and *system CPU time* (OS calls)

    CPU time = user CPU time + system CPU time

    ⇒ elapsed time = user CPU time + system CPU time + wait time

- Our focus: *user CPU time* (*CPU execution time* or, simply, *execution time*)
  - time spent executing the lines of code that are *in our program*

Definition of Performance

- For some program running on machine X:

$$\text{Performance}_X = 1 \,/\, \text{Execution time}_X$$

- *X is n times faster than Y* means:

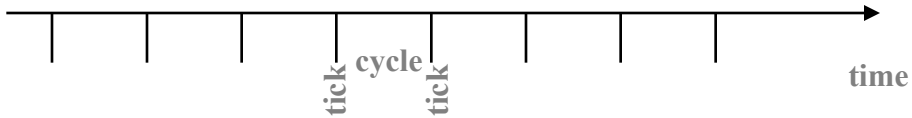
$$\text{Performance}_X \,/\, \text{Performance}_Y = n$$

Clock Cycles

- Instead of reporting execution time in seconds, we often use *cycles*. In modern computers hardware events progress cycle by cycle: in other words, each event, e.g., multiplication, addition, etc., is a sequence of cycles

$$\frac{\text{seconds}}{\text{program}} = \frac{\text{cycles}}{\text{program}} \times \frac{\text{seconds}}{\text{cycle}}$$

- *Clock ticks* indicate start and end of cycles:



- *cycle time* = time between ticks = seconds per cycle
- *clock rate* (*frequency*) = cycles per second (1 Hz. = 1 cycle/sec, 1 MHz. = 10^6 cycles/sec)
- *Example*: A 200 Mhz. clock has a cycle time

$$\frac{1}{200 \times 10^6} \times 10^9 = 5 \text{ nanoseconds}$$

Clock Speed

- The clock signal is carried by one of the lines on the control bus.
- One single pulse is called a ‘clock cycle’.
- Measured in Megahertz (MHz) & Gigahertz (GHz). 1 MHz = 1 million pulses per second. 1 GHZ = 1000 MHz.

Processor Clock Speed

- CPU clock speeds are compared at http://www.cpubenchmark.net/common_cpus.html

Performance Equation I

$$\frac{\text{seconds}}{\text{program}} = \frac{\text{cycles}}{\text{program}} \times \frac{\text{seconds}}{\text{cycle}}$$

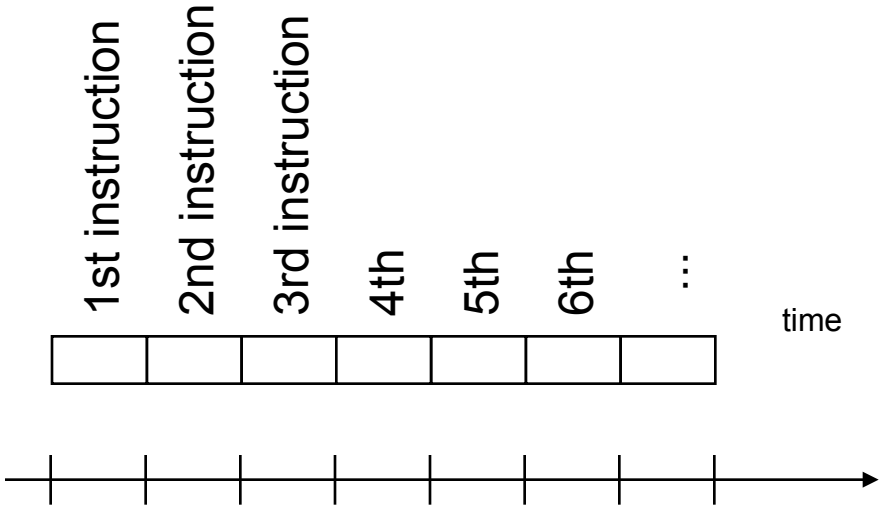
equivalently

$$\text{CPU execution time for a program} = \text{CPU clock cycles for a program} \times \text{Clock cycle time}$$

- So, to improve performance one can either:
  - reduce the number of cycles for a program, or
  - reduce the clock cycle time, or, equivalently,
  - increase the clock rate

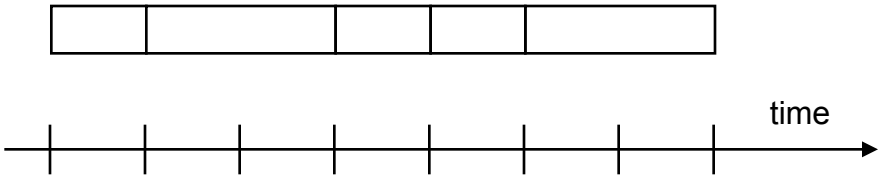
How many cycles are required for a program?

- Could assume that # of cycles = # of instructions



- *This assumption is incorrect!* Because:
  - Different instructions take different amounts of time (cycles)
  - Why…?

How many cycles are required for a program?



- Multiplication takes more time than addition
- Floating point operations take longer than integer ones
- Accessing memory takes more time than accessing registers
- *Important point*: changing the cycle time often changes the number of cycles required for various instructions because it means changing the hardware design. More later…

Example

- Our favorite program runs in 10 seconds on computer A, which has a 400Mhz. clock.
- We are trying to help a computer designer build a new machine B, that will run this program in 6 seconds. The designer can use new (or perhaps more expensive) technology to substantially increase the clock rate, but has informed us that this increase will affect the rest of the CPU design, causing machine B to require 1.2 times as many clock cycles as machine A for the same program.

- *What clock rate should we tell the designer to target?*

Terminology

- A given program will require:
  - some number of instructions (machine instructions)
  - some number of cycles
  - some number of seconds
- We have a vocabulary that relates these quantities:
  - *cycle time* (seconds per cycle)
  - *clock rate* (cycles per second)
  - (*average*) *CPI* (cycles per instruction)

    a floating point intensive application might have a higher average CPI
  - *MIPS* (millions of instructions per second)

    this would be higher for a program using simple instructions

Performance Measure

- *Performance is determined by execution time*

- Do any of these other variables equal performance?
  - # of cycles to execute program?
  - # of instructions in program?
  - # of cycles per second?
  - average # of cycles per instruction?
  - average # of instructions per second?

- *Common pitfall* : thinking one of the variables is indicative of performance when it really isn't

Performance Equation II

CPU execution time for a program = Instruction count for a program × average CPI × Clock cycle time

- *Derive the above equation from Performance Equation I*

Other Ways to Understand Computer Performance

What your research supposedly looks like:

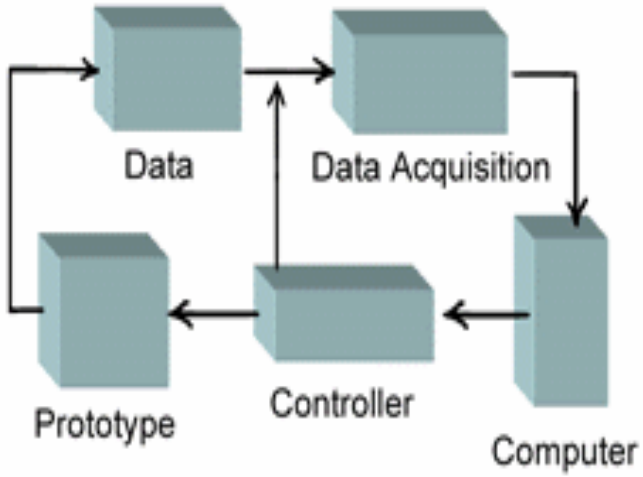


Figure 1. Experimental Diagram

What your research *actually* looks like:

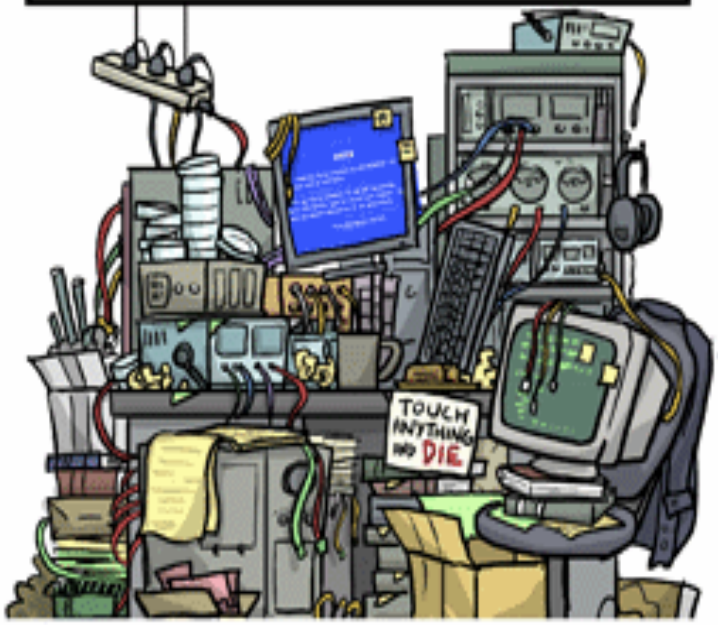


Figure 2. Experimental Mess

Computer Performance Evaluation: Cycles Per Instruction (CPI)

- Most computers run synchronously utilizing a CPU clock running at a constant clock rate:

  where: Clock rate = 1 / clock cycle

- A computer machine instruction is comprised of a number of elementary or micro operations which vary in number and complexity depending on the instruction and the exact CPU organization and implementation.
  - A micro operation is an elementary hardware operation that can be performed during one clock cycle.
  - This corresponds to one micro-instruction in microprogrammed CPUs.
  - Examples: register operations: shift, load, clear, increment, ALU operations: add , subtract, etc.
- Thus a single machine instruction may take one or more cycles to complete termed as the Cycles Per Instruction (CPI).

Computer Performance Measures: Program Execution Time

- For a specific program compiled to run on a specific machine "A", the following parameters are provided:
  - The total instruction count of the program.
  - The average number of cycles per instruction (average CPI).
  - Clock cycle of machine "A"
- How can one measure the performance of this machine running this program?
  - Intuitively the machine is said to be faster or has better performance running this program if the total execution time is shorter.
  - Thus the inverse of the total measured program execution time is a possible performance measure or metric:

$$\text{Performance}_A = 1 \; / \; \text{Execution Time}_A$$

How to compare performance of different machines?

What factors affect performance? How to improve performance?

Comparing Computer Performance Using Execution Time

- To compare the performance of two machines "A", "B" running a given program:

$$\text{Performance}_A = 1 / \text{Execution Time}_A$$
$$\text{Performance}_B = 1 / \text{Execution Time}_B$$

- Machine A is n times faster than machine B means:

  $n = \text{Performance}_A / \text{Performance}_B = \text{Execution Time}_B / \text{Execution Time}_A$

- Example:

  For a given program:

  Execution time on machine A: $\text{Execution}_A = 1$ second

  Execution time on machine B: $\text{Execution}_B = 10$ seconds

  $\text{Performance}_A / \text{Performance}_B = \text{Execution Time}_B / \text{Execution Time}_A$

  $= 10 / 1 = 10$

  The performance of machine A is 10 times the performance of machine B when running this program, or: Machine A is said to be 10 times faster than machine B when running this program.

CPU Execution Time: The CPU Equation

- A program is comprised of a number of instructions, I
  - Measured in: instructions/program
- The average instruction takes a number of cycles per instruction (CPI) to be completed.
  - Measured in: cycles/instruction, CPI
- CPU has a fixed clock cycle time C = 1/clock rate
  - Measured in: seconds/cycle
- CPU execution time is the product of the above three parameters as follows:

$$\text{CPU time} = \frac{\text{Seconds}}{\text{Program}} = \frac{\text{Instructions}}{\text{Program}} \times \frac{\text{Cycles}}{\text{Instruction}} \times \frac{\text{Seconds}}{\text{Cycle}}$$

$$T = I \times CPI \times C$$

CPU Execution Time

For a given program and machine:

CPI = Total program execution cycles / Instructions count

→ CPU clock cycles = Instruction count x CPI

CPU execution time =

= CPU clock cycles x Clock cycle

= Instruction count x CPI x Clock cycle

= I x CPI x C

CPU Execution Time: Example

- A Program is running on a specific machine with the following parameters:
  - Total instruction count: 10,000,000 instructions
  - Average CPI for the program: 2.5 cycles/instruction.
  - CPU clock rate: 200 MHz.
- What is the execution time for this program:

$$\text{CPU time} = \frac{\text{Seconds}}{\text{Program}} = \frac{\text{Instructions}}{\text{Program}} \times \frac{\text{Cycles}}{\text{Instruction}} \times \frac{\text{Seconds}}{\text{Cycle}}$$

CPU time = Instruction count x CPI x Clock cycle

= 10,000,000 x 2.5 x 1 / clock rate

= 10,000,000 x 2.5 x 5×10^{-9}

= .125 seconds

Aspects of CPU Performance

$$\text{CPU time} = \frac{\text{Seconds}}{\text{Program}} = \frac{\text{Instructions}}{\text{Program}} \times \frac{\text{Cycles}}{\text{Instruction}} \times \frac{\text{Seconds}}{\text{Cycle}}$$

	instr. count	CPI	clock rate
Program			
Compiler			
Instr. Set Arch.			
Organization			
Technology			

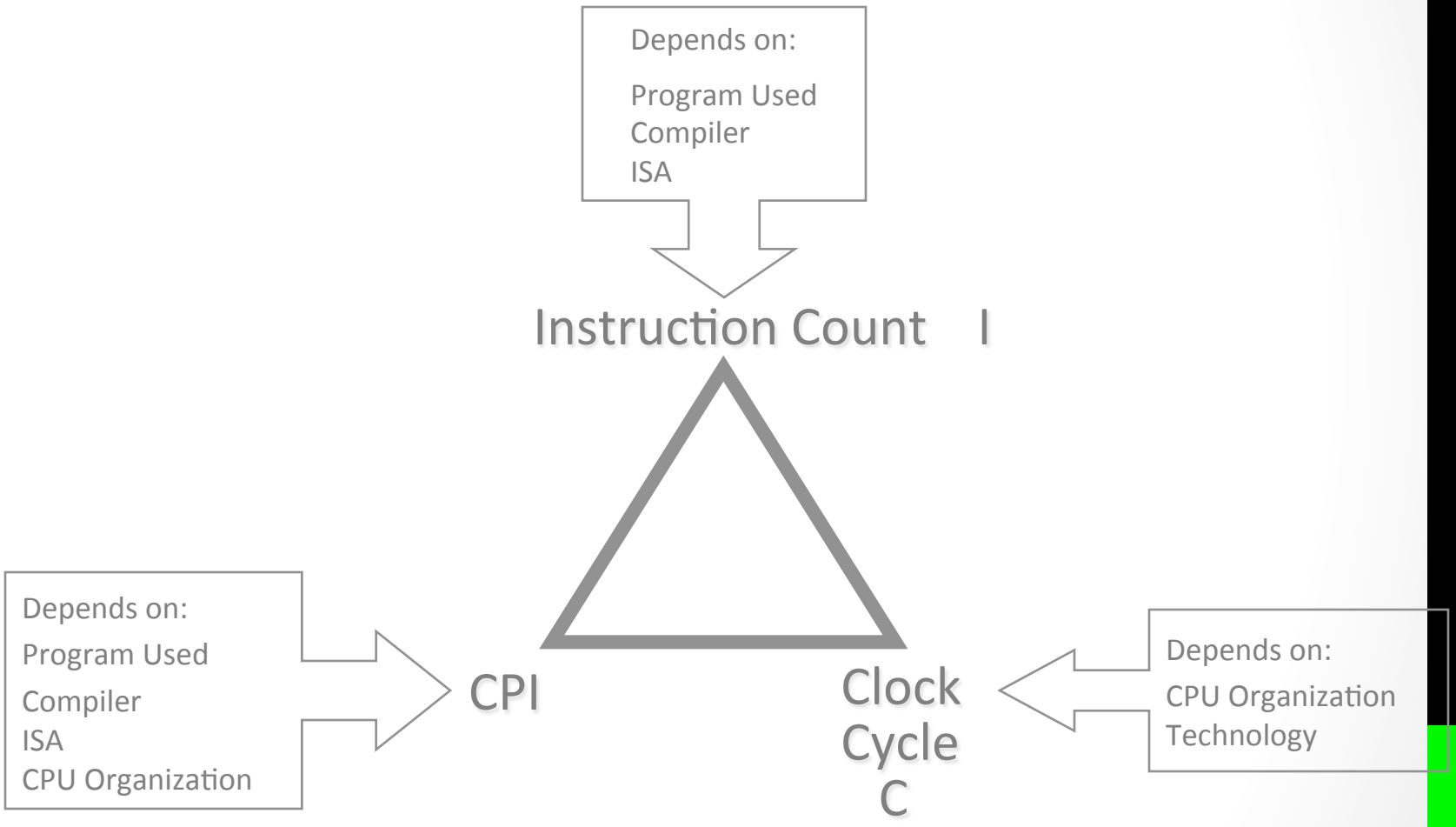
Aspects of CPU Performance

$$\text{CPU time} = \frac{\text{Seconds}}{\text{Program}} = \frac{\text{Instructions}}{\text{Program}} \times \frac{\text{Cycles}}{\text{Instruction}} \times \frac{\text{Seconds}}{\text{Cycle}}$$

	instr count	CPI	clock rate
Program	X	(x)	
Compiler	X	(x)	
Instr. Set.	X	X	
Organization		X	X
Technology			X

Aspects of CPU Execution Time

CPU Time = Instruction count x CPI x Clock cycle



CPU Execution Time: Example

- A Program is running on a specific machine with the following parameters:
  - Total instruction count: 10,000,000 instructions
  - Average CPI for the program: 2.5 cycles/instruction.
  - CPU clock rate: 200 MHz.
- What is the execution time for this program:

$$\text{CPU time} = \frac{\text{Seconds}}{\text{Program}} = \frac{\text{Instructions}}{\text{Program}} \times \frac{\text{Cycles}}{\text{Instruction}} \times \frac{\text{Seconds}}{\text{Cycle}}$$

CPU time = Instruction count x CPI x Clock cycle

= 10,000,000 x 2.5 x 1 / clock rate

= 10,000,000 x 2.5 x 5×10^{-9}

= .125 seconds

Performance Comparison: Example

- From the previous example: A Program is running on a specific machine with the following parameters:
  - Total instruction count: 10,000,000 instructions
  - Average CPI for the program: 2.5 cycles/instruction.
  - CPU clock rate: 200 MHz.
- Using the same program with these changes:
  - A new compiler used: New instruction count 9,500,000

    New CPI: 3.0
  - Faster CPU implementation: New clock rate = 300 MHZ
- What is the speedup with the changes?

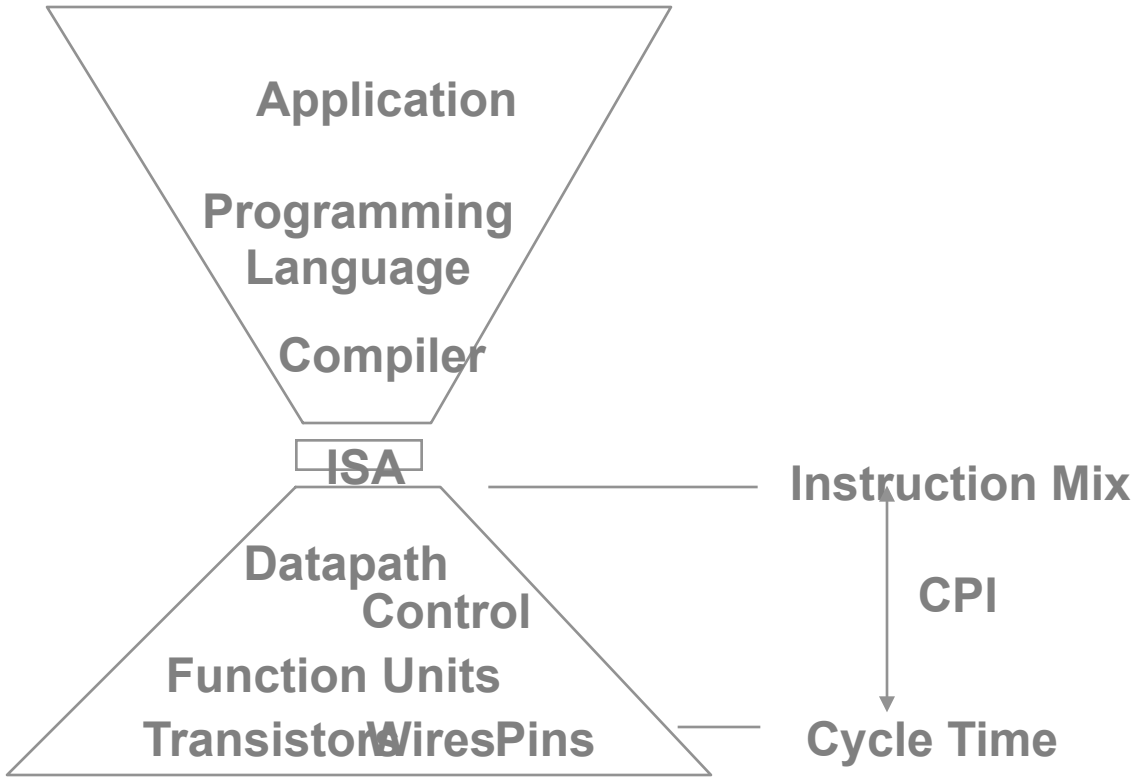
$$\text{Speedup} = \frac{\text{Old Execution Time}}{\text{New Execution Time}} = \frac{I_{old} \times CPI_{old} \times \text{Clock cycle}_{old}}{I_{new} \times CPI_{new} \times \text{Clock Cycle}_{new}}$$

$$\text{Speedup} = (10{,}000{,}000 \times 2.5 \times 5\times10^{-9}) / (9{,}500{,}000 \times 3 \times 3.33\times10^{-9})$$

$$= .125 / .095 = 1.32$$

or 32 % faster after changes.

Organizational Trade-offs



CPI

"Average cycles per instruction"

CPI = (CPU Time * Clock Rate) / Instruction Count
= Clock Cycles / Instruction Count

$$\text{CPU time} = \text{ClockCycleTime} * \sum_{i=1}^{n} \text{CPI}_i * \text{I}_i$$

$$\text{CPI} = \sum_{i=1}^{n} \text{CPI}_i * \text{F}_i \quad \text{where} \quad \text{F}_i = \frac{\text{I}_i}{\text{Instruction Count}}$$

"instruction frequency"

Invest Resources where time is Spent!

CPI Example I

- Suppose we have two implementations of the same instruction set architecture (ISA). For some program:
  - machine A has a clock cycle time of 10 ns. and a CPI of 2.0
  - machine B has a clock cycle time of 20 ns. and a CPI of 1.2

- *Which machine is faster for this program, and by how much?*
- *If two machines have the same ISA, which of our quantities (e.g., clock rate, CPI, execution time, # of instructions, MIPS) will always be identical?*

CPI Example II

- A compiler designer is trying to decide between two code sequences for a particular machine.
- Based on the hardware implementation, there are three different classes of instructions: Class A, Class B, and Class C, and they require 1, 2 and 3 cycles (respectively).
- The first code sequence has 5 instructions:

  2 of A, 1 of B, and 2 of C

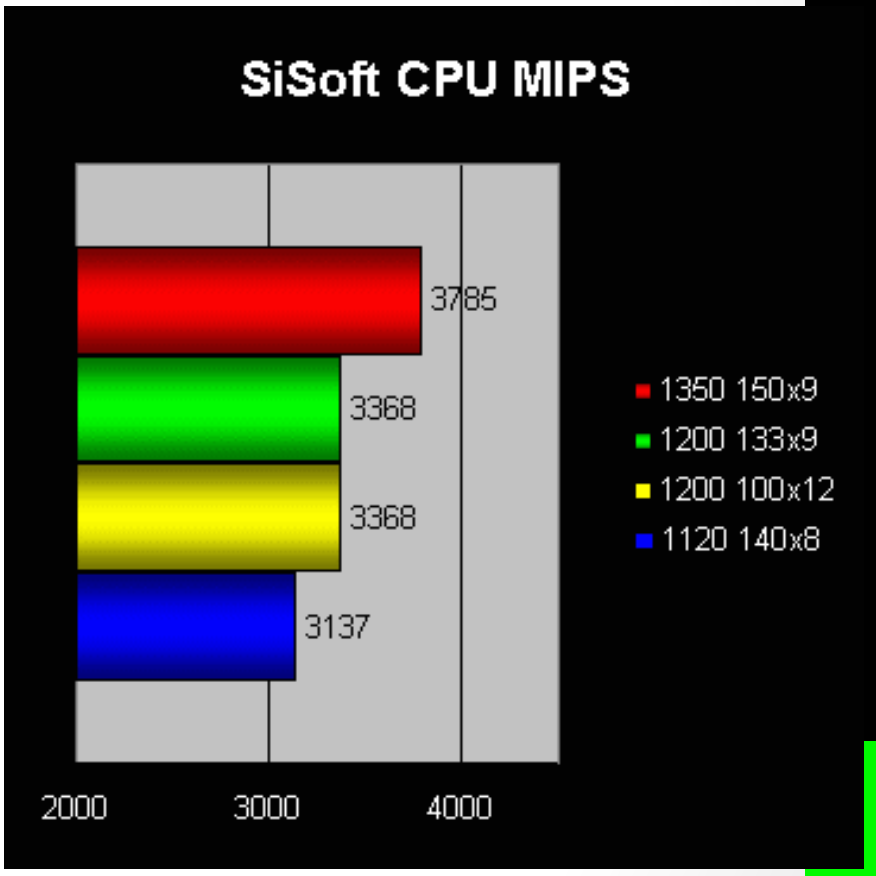
  The second sequence has 6 instructions:

  4 of A, 1 of B, and 1 of C.

- *Which sequence will be faster? How much? What is the CPI for each sequence?*

MIPS

- Stands for Millions of Instructions per second.
- Is a measure of how many machine code instructions a processor execute per second.



Computer Performance Measures : MIPS (Million Instructions Per Second)

- For a specific program running on a specific computer MIPS is a measure of how many millions of instructions are executed per second:

MIPS = Instruction count / (Execution Time x 10^6)

= Instruction count / (CPU clocks x Cycle time x 10^6)

= (Instruction count x Clock rate) / (Instruction count x CPI x 10^6)

= Clock rate / (CPI x 10^6)

- Faster execution time usually means faster MIPS rating.
- Problems with MIPS rating:
  - No account for the instruction set used.
  - Program-dependent: A single machine does not have a single MIPS rating since the MIPS rating may depend on the program used.
  - Easy to abuse: Program used to get the MIPS rating is often omitted.
  - Cannot be used to compare computers with different instruction sets.
  - A higher MIPS rating in some cases may not mean higher performance or better execution time. i.e. due to compiler design variations.

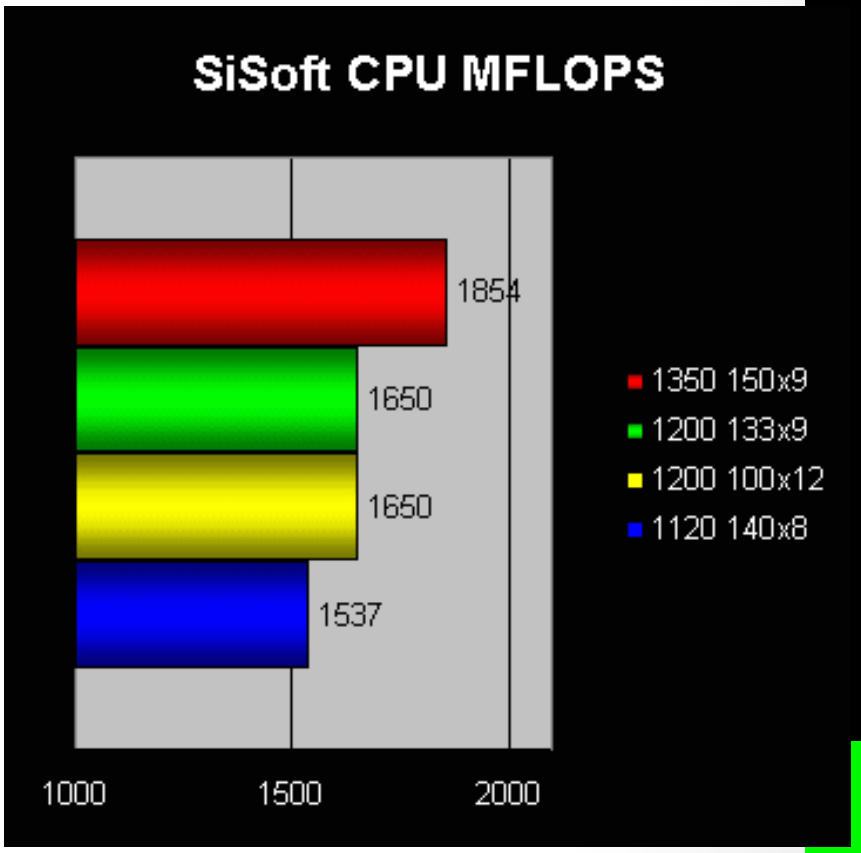
MIPS Example

- Two different compilers are being tested for a 500 MHz. machine with three different classes of instructions: Class A, Class B, and Class C, which require 1, 2 and 3 cycles (respectively). Both compilers are used to produce code for a large piece of software.
- Compiler 1 generates code with 5 billion Class A instructions, 1 billion Class B instructions, and 1 billion Class C instructions.
- Compiler 2 generates code with 10 billion Class A instructions, 1 billion Class B instructions, and 1 billion Class C instructions.

- *Which sequence will be faster according to MIPS?*
- *Which sequence will be faster according to execution time?*

FLOPS

- Stands for ‘Floating Point Operations Per Second.
- Seen as a reliable indicator of performance.
- It’s a measure of the arithmetical calculating speed of a computer.



Computer Performance Measures : MFOLPS (Million FLOating-Point Operations Per Second)

- A floating-point operation is an addition, subtraction, multiplication, or division operation applied to numbers represented by a single or a double precision floating-point representation.
- MFLOPS, for a specific program running on a specific computer, is a measure of millions of floating point-operation (megaflops) per second:

MFLOPS = Number of floating-point operations / (Execution time x 10^6)

- MFLOPS is a better comparison measure between different machines than MIPS.
- Program-dependent: Different programs have different percentages of floating-point operations present. i.e compilers have no floating- point operations and yield a MFLOPS rating of zero.
- Dependent on the type of floating-point operations present in the program.

Performance Enhancement Calculations: Amdahl's Law

- The performance enhancement possible due to a given design improvement is limited by the amount that the improved feature is used
- Amdahl's Law:

Performance improvement or speedup due to enhancement E:

$$\text{Speedup(E)} = \frac{\text{Execution Time without E}}{\text{Execution Time with E}} = \frac{\text{Performance with E}}{\text{Performance without E}}$$

- Suppose that enhancement E accelerates a fraction F of the execution time by a factor S and the remainder of the time is unaffected then:

$$\text{Execution Time with E} = ((1-F) + F/S) \times \text{Execution Time without E}$$

Hence speedup is given by:

$$\text{Speedup(E)} = \frac{\text{Execution Time without E}}{((1 - F) + F/S) \times \text{Execution Time without E}} = \frac{1}{(1 - F) + F/S}$$

Pictorial Depiction of Amdahl's Law

Enhancement E accelerates fraction F of execution time by a factor of S

Before:
Execution Time without enhancement E:

After:
Execution Time with enhancement E:

$$\text{Speedup(E)} = \frac{\text{Execution Time without enhancement E}}{\text{Execution Time with enhancement E}} = \frac{1}{(1 - F) + F/S}$$

Performance Enhancement Example

- For the RISC machine with the following instruction mix given earlier:

Op	Freq	Cycles	CPI(i)	% Time
ALU	50%	1	.5	23%
Load	20%	5	1.0	45%
Store	10%	3	.3	14%
Branch	20%	2	.4	18%

CPI = 2.2

- If a CPU design enhancement improves the CPI of load instructions from 5 to 2, what is the resulting performance improvement from this enhancement:

Fraction enhanced = F = 45% or .45

Unaffected fraction = 100% - 45% = 55% or .55

Factor of enhancement = 5/2 = 2.5

Using Amdahl's Law:

$$\text{Speedup(E)} = \frac{1}{(1 - F) + F/S} = \frac{1}{.55 + .45/2.5} = 1.37$$

An Alternative Solution Using CPU Equation

Op	Freq	Cycles	CPI(i)	% Time
ALU	50%	1	.5	23%
Load	20%	5	1.0	45%
Store	10%	3	.3	14%
Branch	20%	2	.4	18%

CPI = 2.2

- If a CPU design enhancement improves the CPI of load instructions from 5 to 2, what is the resulting performance improvement from this enhancement:

Old CPI = 2.2

New CPI = .5 x 1 + .2 x 2 + .1 x 3 + .2 x 2 = 1.6

$$\text{Speedup(E)} = \frac{\text{Original Execution Time}}{\text{New Execution Time}} = \frac{\text{Instruction count x old CPI x clock cycle}}{\text{Instruction count x new CPI x clock cycle}}$$

$$= \frac{\text{old CPI}}{\text{new CPI}} = \frac{2.2}{1.6} = 1.37$$

Which is the same speedup obtained from Amdahl's Law in the first solution.

Performance Enhancement Example

- A program runs in 100 seconds on a machine with multiply operations responsible for 80 seconds of this time. By how much must the speed of multiplication be improved to make the program four times faster?

$$\text{Desired speedup} = 4 = \frac{100}{\text{Execution Time with enhancement}}$$

→ Execution time with enhancement = 25 seconds

25 seconds = (100 - 80 seconds) + 80 seconds / n

25 seconds = 20 seconds + 80 seconds / n

→ 5 = 80 seconds / n

→ n = 80/5 = 16

Hence multiplication should be 16 times faster to get a speedup of 4.

Performance Enhancement Example

- For the previous example with a program running in 100 seconds on a machine with multiply operations responsible for 80 seconds of this time. By how much must the speed of multiplication be improved to make the program five times faster?

$$\text{Desired speedup} = 5 = \frac{100}{\text{Execution Time with enhancement}}$$

→ Execution time with enhancement = 20 seconds

20 seconds = (100 - 80 seconds) + 80 seconds / n

20 seconds = 20 seconds + 80 seconds / n

→ 0 = 80 seconds / n

No amount of multiplication speed improvement can achieve this.

Extending Amdahl's Law To Multiple Enhancements

- Suppose that enhancement E_i accelerates a fraction F_i of the execution time by a factor S_i and the remainder of the time is unaffected then:

$$Speedup = \frac{\text{Original Execution Time}}{\left((1 - \sum_i F_i) + \sum_i \frac{F_i}{S_i}\right) X \text{Original Execution Time}}$$

$$Speedup = \frac{1}{\left((1 - \sum_i F_i) + \sum_i \frac{F_i}{S_i}\right)}$$

Note: All fractions refer to original execution time.

Amdahl's Law With Multiple Enhancements: Example

- Three CPU performance enhancements are proposed with the following speedups and percentage of the code execution time affected:

  $Speedup_1 = S_1 = 10$ $Percentage_1 = F_1 = 20\%$

  $Speedup_2 = S_2 = 15$ $Percentage_1 = F_2 = 15\%$

  $Speedup_3 = S_3 = 30$ $Percentage_1 = F_3 = 10\%$

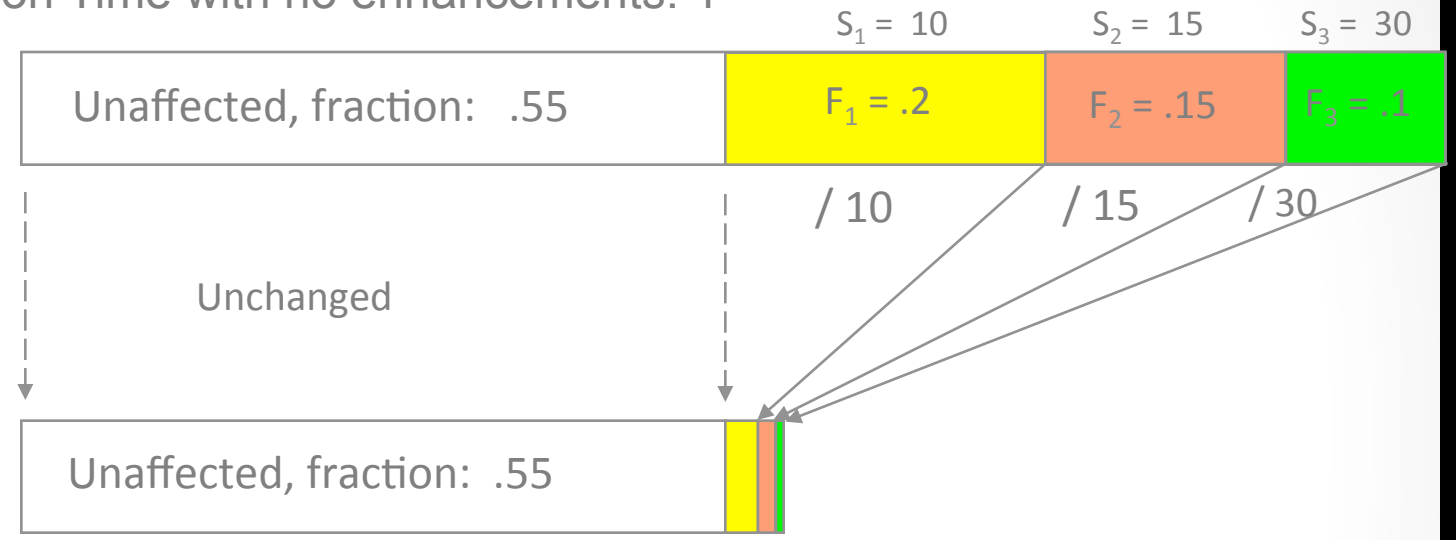
- While all three enhancements are in place in the new design, each enhancement affects a different portion of the code and only one enhancement can be used at a time.
- What is the resulting overall speedup?

$$Speedup = \frac{1}{\left((1 - \sum_i F_i) + \sum_i \frac{F_i}{S_i}\right)}$$

- Speedup = 1 / [(1 - .2 - .15 - .1) + .2/10 + .15/15 + .1/30)]
  = 1 / [.55 + .0333]
  = 1 / .5833 = 1.71

Pictorial Depiction of Example

Before:
Execution Time with no enhancements: 1



After:
Execution Time with enhancements: .55 + .02 + .01 + .00333 = .5833

Speedup = 1 / .5833 = 1.71

Note: All fractions refer to original execution time.

Benchmarks

- Performance best determined by running a real application
  - use programs typical of expected workload
  - or, typical of expected class of applications
    e.g., compilers/editors, scientific applications, graphics, etc.

- Small benchmarks
  - nice for architects and designers
  - easy to standardize
  - can be abused!

- Benchmark suites
  - Perfect Club: set of application codes
  - Livermore Loops: 24 loop kernels
  - Linpack: linear algebra package
  - SPEC: mix of code from industry organization

Benchmark Tests

- Benchmark tests simply time how long a computer system takes to complete a standard set of application based tasks i.e. reformatting a 100 page 'Word' document.
- http://www.passmark.com/ is one make of benchmark test software.

Choosing Programs To Evaluate Performance

Levels of programs or benchmarks that could be used to evaluate performance:

- Actual Target Workload: Full applications that run on the target machine.

- Real Full Program-based Benchmarks:

  Select a specific mix or suite of programs that are typical of targeted applications or workload (e.g SPEC95, SPEC CPU2000).

- Small "Kernel" Benchmarks:

  Key computationally-intensive pieces extracted from real programs.

  - Examples: Matrix factorization, FFT, tree search, etc.

  Best used to test specific aspects of the machine.

- Microbenchmarks:

  Small, specially written programs to isolate a specific aspect of performance characteristics: Processing: integer, floating point, local memory, input/output, etc.

Types of Benchmarks

Pros		Cons
• Representative	Actual Target Workload	• Very specific. • Non-portable. • Complex: Difficult to run, or measure.
• Portable. • Widely used. • Measurements useful in reality.	Full Application Benchmarks	• Less representative than actual workload.
• Easy to run, early in the design cycle.	Small “Kernel” Benchmarks	• Easy to “fool” by designing hardware to run them well.
• Identify peak performance and potential bottlenecks.	Microbenchmarks	• Peak performance results may be a long way from real application performance

SPEC (System Performance Evaluation Corporation)

- Sponsored by industry but independent and self-managed – trusted by code developers and machine vendors
- Clear guides for testing, see www.spec.org
- Regular updates (benchmarks are dropped and new ones added periodically according to relevance)
- Specialized benchmarks for particular classes of applications
- Can still be abused…, by selective optimization!

SPEC History

- First Round: SPEC CPU89
  - 10 programs yielding a single number
- Second Round: SPEC CPU92
  - SPEC CINT92 (6 integer programs) and SPEC CFP92 (14 floating point programs)
  - compiler flags can be set differently for different programs
- Third Round: SPEC CPU95
  - new set of programs: SPEC CINT95 (8 integer programs) and SPEC CFP95 (10 floating point)
  - single flag setting for all programs
- Fourth Round: SPEC CPU2000
  - new set of programs: SPEC CINT2000 (12 integer programs) and SPEC CFP2000 (14 floating point)
  - single flag setting for all programs
  - programs in C, C++, Fortran 77, and Fortran 90

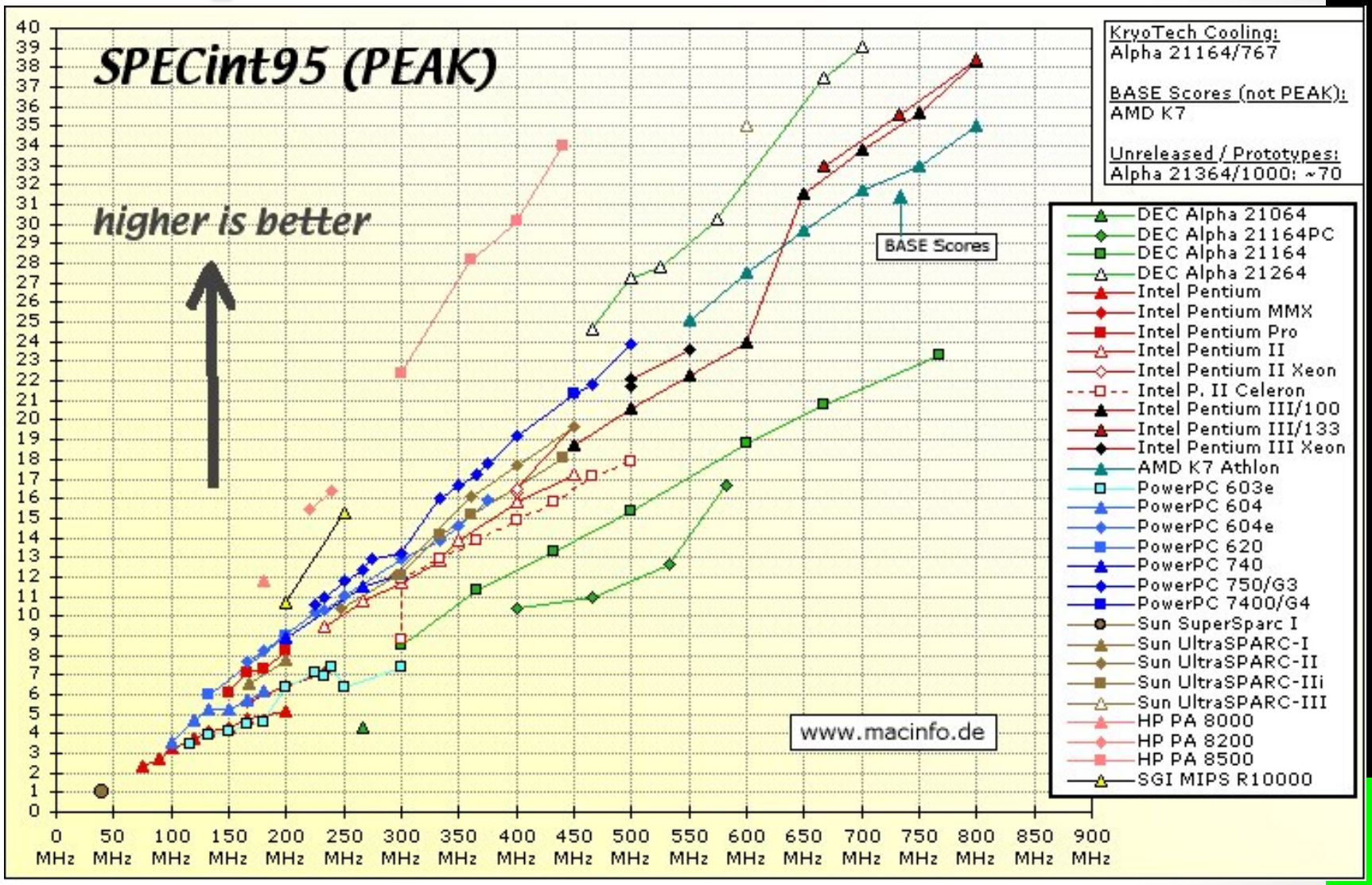
CINT2000 (Integer component of SPEC CPU2000)

Program	Language	What It Is
164.gzip	C	Compression
175.vpr	C	FPGA Circuit Placement and Routing
176.gcc	C	C Programming Language Compiler
181.mcf	C	Combinatorial Optimization
186.crafty	C	Game Playing: Chess
197.parser	C	Word Processing
252.eon	C++	Computer Visualization
253.perlbmk	C	PERL Programming Language
254.gap	C	Group Theory, Interpreter
255.vortex	C	Object-oriented Database
256.bzip2	C	Compression
300.twolf	C	Place and Route Simulator

SPEC95 Programs

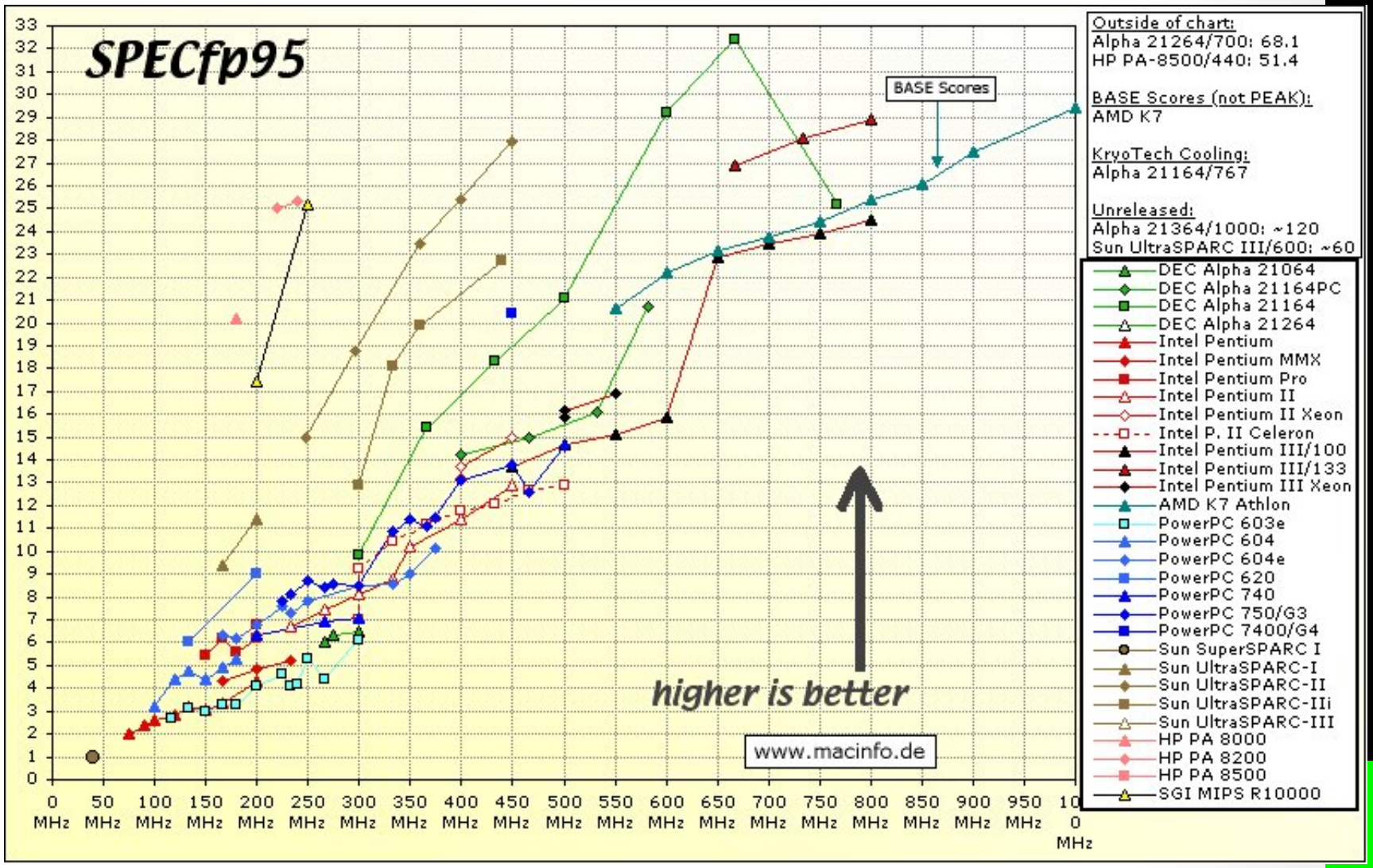
	Benchmark	Description
Integer	go	Artificial intelligence; plays the game of Go
	m88ksim	Motorola 88k chip simulator; runs test program
	gcc	The Gnu C compiler generating SPARC code
	compress	Compresses and decompresses file in memory
	li	Lisp interpreter
	ijpeg	Graphic compression and decompression
	perl	Manipulates strings and prime numbers in the special-purpose programming language Perl
	vortex	A database program
Floating Point	tomcatv	A mesh generation program
	swim	Shallow water model with 513 x 513 grid
	su2cor	quantum physics; Monte Carlo simulation
	hydro2d	Astrophysics; Hydrodynamic Naiver Stokes equations
	mgrid	Multigrid solver in 3-D potential field
	applu	Parabolic/elliptic partial differential equations
	trub3d	Simulates isotropic, homogeneous turbulence in a cube
	apsi	Solves problems regarding temperature, wind velocity, and distribution of pollutant
	fpppp	Quantum chemistry
	wave5	Plasma physics; electromagnetic particle simulation

Sample SPECint95 Results



Source URL: http://www.macinfo.de/bench/specmark.html

Sample SPECfp95 Results



Source URL: http://www.macinfo.de/bench/specmark.html

SPEC CPU2000 Programs

	Benchmark	Language	Descriptions
CINT2000 (Integer)	164.gzip	C	Compression
	175.vpr	C	FPGA Circuit Placement and Routing
	176.gcc	C	C Programming Language Compiler
	181.mcf	C	Combinatorial Optimization
	186.crafty	C	Game Playing: Chess
	197.parser	C	Word Processing
	252.eon	C++	Computer Visualization
	253.perlbmk	C	PERL Programming Language
	254.gap	C	Group Theory, Interpreter
	255.vortex	C	Object-oriented Database
	256.bzip2	C	Compression
	300.twolf	C	Place and Route Simulator
CFP2000 (Floating Point)	168.wupwise	Fortran 77	Physics / Quantum Chromodynamics
	171.swim	Fortran 77	Shallow Water Modeling
	172.mgrid	Fortran 77	Multi-grid Solver: 3D Potential Field
	173.applu	Fortran 77	Parabolic / Elliptic Partial Differential Equations
	177.mesa	C	3-D Graphics Library
	178.galgel	Fortran 90	Computational Fluid Dynamics
	179.art	C	Image Recognition / Neural Networks
	183.equake	C	Seismic Wave Propagation Simulation
	187.facerec	Fortran 90	Image Processing: Face Recognition
	188.ammp	C	Computational Chemistry
	189.lucas	Fortran 90	Number Theory / Primality Testing
	191.fma3d	Fortran 90	Finite-element Crash Simulation
	200.sixtrack	Fortran 77	High Energy Nuclear Physics Accelerator Design
	301.apsi	Fortran 77	Meteorology: Pollutant Distribution

Source: http://www.spec.org/osg/cpu2000/

Top 20 SPEC CPU2000 Results (As of March 2002)

Top 20 SPECint2000

#	MHz	Processor	int peak	int base
1	1300	POWER4	814	790
2	2200	Pentium 4	811	790
3	2200	Pentium 4 Xeon	810	788
4	1667	Athlon XP	724	697
5	1000	Alpha 21264C	679	621
6	1400	Pentium III	664	648
7	1050	UltraSPARC-III Cu	610	537
8	1533	Athlon MP	609	587
9	750	PA-RISC 8700	604	568
10	833	Alpha 21264B	571	497
11	1400	Athlon	554	495
12	833	Alpha 21264A	533	511
13	600	MIPS R14000	500	483
14	675	SPARC64 GP	478	449
15	900	UltraSPARC-III	467	438
16	552	PA-RISC 8600	441	417
17	750	POWER RS64-IV	439	409
18	700	Pentium III Xeon	438	431
19	800	Itanium	365	358
20	400	MIPS R12000	353	328

Top 20 SPECfp2000

MHz	Processor	fp peak	fp base
1300	POWER4	1169	1098
1000	Alpha 21264C	960	776
1050	UltraSPARC-III Cu	827	701
2200	Pentium 4 Xeon	802	779
2200	Pentium 4	801	779
833	Alpha 21264B	784	643
800	Itanium	701	701
833	Alpha 21264A	644	571
1667	Athlon XP	642	596
750	PA-RISC 8700	581	526
1533	Athlon MP	547	504
600	MIPS R14000	529	499
675	SPARC64 GP	509	371
UltraSPARC-III		482	427
1400	Athlon	458	426
1400	Pentium III	456	437
500	PA-RISC 8600	440	397
450	POWER3-II	433	426
500	Alpha 21264	422	383
400	MIPS R12000	407	382

Source: http://www.aceshardware.com/SPECmine/top.jsp

CFP2000 (Floating point component of SPEC CPU2000)

Program	Language	What It Is
168.wupwise	Fortran 77	Physics / Quantum Chromodynamics
171.swim	Fortran 77	Shallow Water Modeling
172.mgrid	Fortran 77	Multi-grid Solver: 3D Potential Field
173.applu	Fortran 77	Parabolic / Elliptic Differential Equations
177.mesa	C	3-D Graphics Library
178.galgel	Fortran 90	Computational Fluid Dynamics
179.art	C	Image Recognition / Neural Networks
183.equake	C	Seismic Wave Propagation Simulation
187.facerec	Fortran 90	Image Processing: Face Recognition
188.ammp	C	Computational Chemistry
189.lucas	Fortran 90	Number Theory / Primality Testing
191.fma3d	Fortran 90	Finite-element Crash Simulation
200.sixtrack	Fortran 77	High Energy Physics Accelerator Design
301.apsi	Fortran 77	Meteorology: Pollutant Distribution

SPEC CPU2000 reporting

- Refer SPEC website www.spec.org for documentation
- Single number result – geometric mean of normalized ratios for each code in the suite
- Report precise description of machine
- Report compiler flag setting

Factors Affecting Performance – Data Bus

- Data bus width determines how much data can be transferred from memory to processor in one clock cycle.
- Increasing the data bus width will increase the quantity of data which the bus can carry at any one time.

Factors Affecting Performance – Cache Memory

- Faster for processor to access data in cache than main memory;
- Cache is made up from ‘static RAM’
- The internal system bus linking cache memory and the processor can be up to 256 bits wide.

Factors Affecting Performance – Peripheral transfer speed

- All peripherals operate at slower speed than the processor.
- This can have a major affect on performance.
- Selecting drives and peripherals with the fastest transfer rate can improve system performance.

Factors Affecting Performance – Peripheral transfer speed

CD Transfer rate	Transfer rate in Kilobytes per second	Time taken to read a 10 Megabyte file
52 X	7800	1.31 seconds
32 X	4800	2.13 seconds

The difference between these times may not look much to our eyes but they are significant in terms of computer performance.

Other Factors

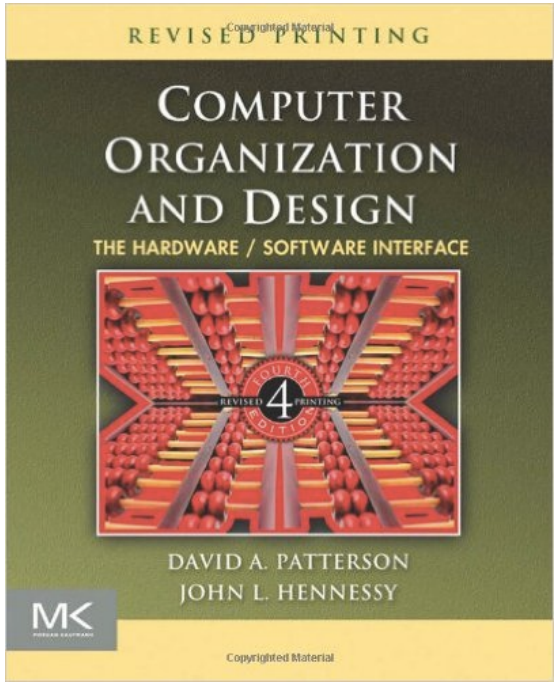
- Increasing clock speed.
- Adding more Main Memory
- Increasing VRAM
- Adding more processors

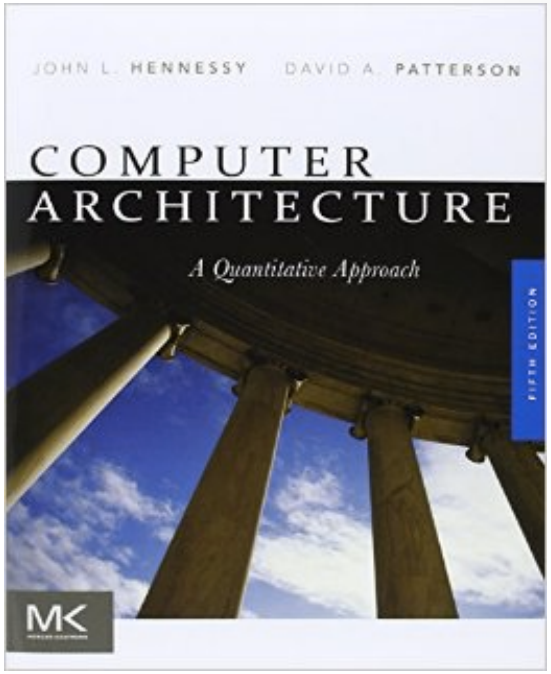
Computer Performance

Tactic	Effect on Performance
Increase clock speed	Increase
Increase data bus width	Increase
Increase Cache memory	Increase
Increase Address Bus	None
Number of processors	Increase
Increase RAM	Slight Increase
Increase VRAM	Increase graphics performance
Increase data transfer rate	Increase

More Details…

Patterson and Hennesy, Computer Organization and Design (The Hardware, Software Interface)





Patterson and Hennesy, Computer Architecture; A Quantitative Approach

More Questions?