Introduction to Computer Architecture

From Wikipedia, the free encyclopedia

In computer engineering, computer architecture is a set of rules and methods that describe the functionality, organization, and implementation of computer systems. Some definitions of architecture define it as describing the capabilities and programming model of a computer but not a particular implementation. In other definitions computer architecture involves instruction set architecture design, microarchitecture design, logic design, and implementation.

Computer Architecture

What is Computer Architecture — Parsed

Wikipedia

In computer engineering, **computer architecture** is a set of **rules and methods** that describe the **functionality**, **organization**, and **implementation** of computer systems. Some definitions of architecture define it as describing the **capabilities** and **programming model** of a computer but not a particular implementation. In other definitions computer architecture involves **instruction set architecture** design, **microarchitecture** design, **logic** design, and implementation.

Translation:

computer architecture = { **rules and methods** | describe

Functionality — system capabilities and programming model

Organization — instruction set architecture, microarchitecture

Implementation — logic design

}

Computer Architecture

What rules and methods?

Performance

- Short run time
  - Fast programs
- Low latency
  - No waiting between programs and operations
- Low energy consumption
  - Low electric bills
  - Long battery life
  - No overheating

Market factors

- Low cost (in relation to realistic demand for devices)
- Reliable manufacture and delivery
- Profitability

Computing Platform by Application

Workstation applications

Office, basic number crunching, graphics, gaming

A few sequential loop-oriented threads

Typical CPU — Intel x86 (2 to 16 cores)

Mobile applications

Low power version of workstation

Typical CPU — ARM (1 to 4 cores)

Online Transaction Processing (OLTP)

Banking, order processing, inventory, student information system

Thousands of independent SQL transactions with memory latency

Typical CPU — SPARC (64 to 256 cores)

Supercomputer applications

Heavy number crunching, data mining

Thousands of separable sequential loop-oriented threads

Typical CPU — IBM Power (up to 512 Kcores)

Mainframe + Virtualization + Cloud

Mainframe

- 120 CPU cores + 3840 GB RAM + 8 GB/s I/O + high reliability
- Replaces 10 to 1000 servers
- Complex partitioning
  - Allocate hardware subsystems as needed
  - Multiple independent operating systems

Server Virtualization

- Hardware resources partitioned by software running over OS
- Multiple guest operating systems over OS

Cloud computing

- Provider sells standard system interface as service
  - Infrastructure as Service, Platform as Service, Software as Service
- Customer sees system specified in contract
- Provider handles operations+administration+maintenance (OAM)

Introduction to Performance

Basic Definitions

Performance (ביצועים)

Processing speed

Performance measures

Response time (זמן תגובה)

Elapsed time from start to finish of a defined task

Run Time (זמן ריצה)

Response time for a start to finish program task

Latency (חביון – זמן המתנה)

Excess response time — depends on context

Throughput (תפוקה)

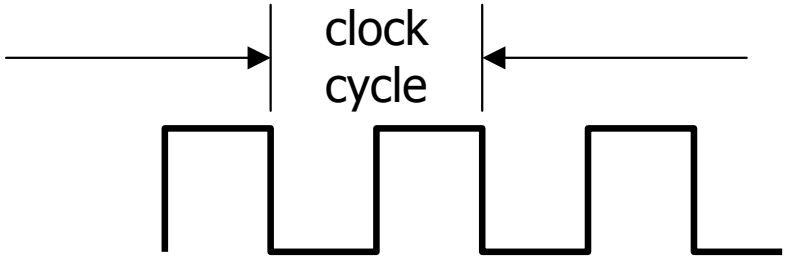
Number of defined tasks performed per unit time

Speedup (שיפור)

$$S = \frac{\text{old run time}}{\text{new run time}} \qquad S > 1 \Rightarrow \text{new run time} < \text{old run time}$$

Run Time and Clock Cycles

CPU is timed by periodic signal called a Clock



Clock Cycle (CC) measured in seconds per cycle

Clock Rate = cycles per second = Hz (Hertz)

Instruction requires 1 or more clock cycles to process

$$\text{Run time} = \text{clock cycles to run program} \times \text{seconds per clock cycles}$$

$$= \frac{\text{clock cycles to run program}}{\text{clock cycles per second}}$$

Higher clock rate $\Rightarrow$ shorter run time

Fewer clock cycles (at constant clock rate) $\Rightarrow$ shorter run time

Speedup and Clock Rate

$$S = \frac{T^{old}}{T^{new}}$$

$$= \frac{\text{program clock cycles}^{old} \times \text{seconds per clock cycle}^{old}}{\text{program clock cycles}^{new} \times \text{seconds per clock cycle}^{new}}$$

$$= \frac{\text{program clock cycles}^{old} \Big/ \text{clock rate}^{old}}{\text{program clock cycles}^{new} \Big/ \text{clock rate}^{new}}$$

$$= \frac{\text{program clock cycles}^{old}}{\text{program clock cycles}^{new}} \times \frac{\text{clock rate}^{new}}{\text{clock rate}^{old}}$$

Speedup follows from

- Higher clock rate
- Fewer program clock cycles
  - Improvements to code
  - Structural improvements in hardware

Factors Affecting Run Time

CPU hardware

Hardware → average clock cycles (CC) required per instruction

Memory (RAM + cache)

Quantity and organization affects data availability

Internal communication and I/O

Speed and organization affects data availability

Operating system efficiency

CPU devotes less time to dense OS code

OS manages tasks/threads to keep hardware busy

Compiler

Converts high level language to machine code

Optimized code runs faster

Special hardware

Dedicated processors (graphics, memory management)

Application code

Efficient algorithms, data structures, parallelization

Examples of Factors Affecting Performance

CPU Hardware Example — Multiple Core Processors

N Core Symmetric Multiprocessor (SMP)

N complete CPUs on one chip

Divide work into **THREADS** running in **PARALLEL** on N processors

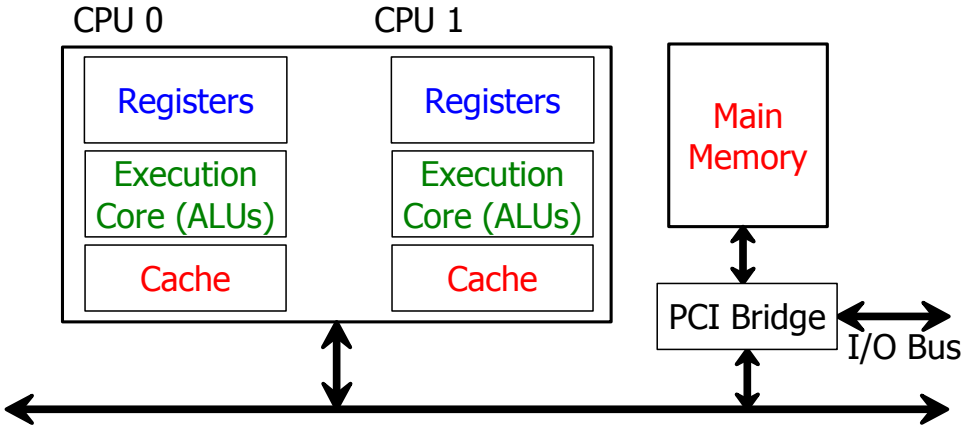
Each CPU has multiple Execution Units (EU)

ALU operates on integers

FPU operates on float / double

Vector processor operates on long registers

Dual Core Processor



OS assigns THREADS to each core

If program threads are separable

If data structures are not too entangled

CPU Hardware Example —Vector Processor

Vector Processor

SIMD — **S**ingle **I**nstruction **M**ultiple **D**ata

Performs same operation on 4, 8, or 16 bytes in parallel

No carry/borrow between bytes

Example

64-bit Source and Destination registers

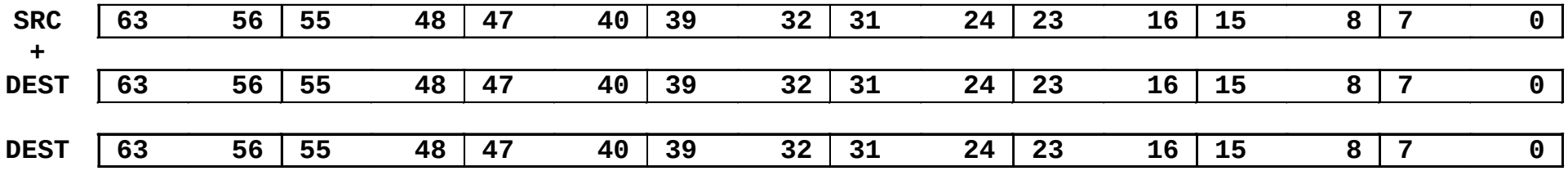
PARALLEL_ADD on 8 pairs of byte operands

$$\mathbf{SRC_{0\ldots7} + DEST_{0\ldots7} = DEST_{0\ldots7}}$$

$$\mathbf{SRC_{8\ldots15} + DEST_{8\ldots15} = DEST_{8\ldots15}}$$

$$\mathbf{\ldots}$$

$$\mathbf{SRC_{56\ldots63} + DEST_{56\ldots63} = DEST_{56\ldots63}}$$



Memory Example — Hybrid Data Structure

Graphic array

200 vertex points = 25 groups of 8 words

Hybrid Data Structure for efficient vector processing

Coordinates and colors

- Stored in separate data structures
- Structures handled in **CONCURRENT THREADS** on separate CPUs

Coordinates

```
struct { float x[8], y[8], z[8] ;
} H_xyz[25] ;
```

8-word group loaded and processed as **vector** on CPU 0

Each loop updates 8 x-coordinates, then 8 y's, then 8 z's

Colors

```
struct { float r[8], g[8], b[8] ;
} H_rgb[25] ;
```

8-word group loaded and processed as **vector** on CPU 1

Each loop updates 8 reds, then 8 greens, then 8 blues

Memory Example — Color Data Structure

Addressing in 32-bit processors

Processor sends 32-bit aligned address A (multiple of 4)

Reads 4-byte word — bytes from addresses **`A, A+1, A+2, A+3`**

Access to individual byte requires reading entire dword

24-bit True Color

3 color bytes — Red, Blue, Green

2^8 = 256 levels per color (**`0x00 – 0xFF`**)

Most 24-bit colors split between dwords

Access to pixel color ⇒ 2 memory cycles

dword			dword			dword					
R	G	B	R	G	B	R	G	B	R	G	B
1 cycle			2 cycles			2 cycles			1 cycles		

32-bit True Color

Pad 24-bit color with blank byte

Align color data on 32-bit addresses

One memory cycle per pixel

dword				dword			
R	G	B	—	R	G	B	—

Compiler Efficiency Example

C code compiled inefficiently for Intel 8086 processor

```
main()
{
      int i,j;
      for (i = 0; i < 10; i++){
            j = 2 * i;
      }
}
```

```
0000  MOV     WORD PTR [BP-02],0000 ; i = 0
0005  CMP     WORD PTR [BP-02],+0A
0009  JGE     0018                          ; break on i ≥ 10
000B  MOV     AX,[BP-02]                    ; AX ← i
000E  SHL     AX,1                          ; AX ← 2 * AX
0010  MOV     [BP-04],AX                    ; j ← AX
0013  INC     WORD PTR [BP-02]              ; i++
0016  JMP     0005                          ; loop
0018  RET
```

Page from Intel 8086 Manual



80186/80188 HIGH-INTEGRATION 16-BIT MICROPROCESSORS,
COPYRIGHT © INTEL CORPORATION, 1995

INSTRUCTION SET SUMMARY

Clock Cycles per Instruction →

Function	Format				80186 Clock Cycles
DATA TRANSFER					
MOV = Move:					
Register to Register/Memory	1 0 0 0 1 0 0 w	mod reg r/m			2/12
Register/memory to register	1 0 0 0 1 0 1 w	mod reg r/m			2/9
Immediate to register/memory	1 1 0 0 0 1 1 w	mod 000 r/m	data	data if w=1	12/13
Immediate to register	1 0 1 1 w reg	data	data if w=1		3/4
Memory to accumulator	1 0 1 0 0 0 0 w	addr-low	addr-high		8
Accumulator to memory	1 0 1 0 0 0 1 w	addr-low	addr-high		9
Register/memory to segment register	1 0 0 0 1 1 1 0	mod 0 reg r/m			2/9
Segment register to register/memory	1 0 0 0 1 1 0 0	mod 0 reg r/m			2/11

Program Timing for 8086

Program contains

Setup/takedown instructions (run once)

Loop control instructions

ALU instructions

Instruction timings are given in 8086 manual (in clock cycles)

	Instruction	8086 Clock Cycles (CC)
	MOV WORD PTR [BP-02],0000	MOV imm to r/m 4/13
start:	CMP WORD PTR [BP-02],+0A	CMP r/m,imm 3/10
	JGE stop	Jcc (not taken/taken) 4/13
	MOV AX,[BP-02]	MOV r/m to reg 2/9
	SHL AX,1	Shift reg 2
	MOV [BP-04],AX	MOV reg to r/m 2/12
	INC WORD PTR [BP-02]	INC r/m 3/15
	JMP start	JMP 14
stop:	RET	RET 16

Program Run Time

	Instruction	8086 Clock Cycles (CC)
	MOV WORD PTR [BP-02],0000	13 CC (runs once)
start:	CMP WORD PTR [BP-02],+0A	10 CC on each loop
	JGE stop	4 CC on all loops but last 13 CC on last
	MOV AX,[BP-02]	9 CC on all loops but last
	SHL AX,1	2 CC on all loops but last
	MOV [BP-04],AX	12 CC on all loops but last
	INC WORD PTR [BP-02]	15 CC on all loops but last
	JMP start	14 CC on all loops but last
stop:	RET	16 CC (runs once)

N = number of loop iterations

Total clock cycles = $13 + N \times 10 + (N - 1) \times (4 + 9 + 2 + 12 + 15 + 14) + 13 + 16$

$= 66 \times N - 14$

For N = 11 (stop on i = 10), Total CC = 712

Example — More Efficient Compilation

Store Variables in Registers — Not Memory

	Instruction	8086 Clock Cycles (CC)
	MOV SI,0000	4 CC (runs once)
start:	CMP SI,+0A	3 CC on each loop
	JGE stop	4 CC on all loops but last 13 CC on last
	MOV AX,SI	2 CC on all loops but last
	SHL AX,1	2 CC on all loops but last
	MOV DI,AX	2 CC on all loops but last
	INC SI	3 CC on all loops but last
	JMP start	14 CC on all loops but last
stop:	RET	16 CC (runs once)

Total clock cycles $= 4 + N \times 3 + (N - 1) \times (4 + 2 + 2 + 2 + 3 + 14) + 13 + 16$

$= 30 \times N + 6$

For N = 11 (stop on i = 10), Total CC = 337

Using register variables requires large number of registers

$$S = \frac{712}{337} = 2.11$$

Example — Even More Efficient Compilation

Rebuild Loop

	Instruction	8086 Clock Cycles
	MOV SI,0000	MOV imm to reg 4
start:	MOV AX,SI	MOV reg to reg 2
	SHL AX,1	SHIFT reg 2
	MOV DI,AX	MOV reg to reg 2
	INC SI	INC reg 3
	CMP SI,+0A	CMP reg,imm 3/10
	JL start	Jcc (not taken/taken) 4/13
stop:	RET	RET 16

Total clock cycles = $4 + N \times (2 + 2 + 2 + 3 + 3) + (N - 1) \times 13 + 4 + 16$

$= 25 \times N + 11$

For N = 10 (stop on i = 10), Total CC = 261

$$S = \frac{712}{261} = 2.73$$

Measuring Performance

Benchmarks

Definition

Collection of programs for measurement and comparison of system performance

Requirements

Standard and scientific

- Consistent result on repeated tests
- Consistent result by anyone repeating tests

Test system in realistic way

- Reflect statistically representative use of
  - Instruction types
  - Data types
  - Loop length
  - OS and compiler conditions

Summarize data so comparisons make sense

SPEC Benchmark

Programs for system performance measurement + comparison

- Standard + repeatable
- Test system for realistic conditions
- Summary score for easy comparison
- Results posted at **http://www.spec.org/**

Specific test suites

- Cint — CPU integer instructions
- Cfp — CPU FP instructions
- Performance as file server, web server, mail server, graphics

Updated every few years to reflect realistic conditions

- Based on current statistical distributions of computing tasks
- Current CPU test version — 2017
  - Previous version — 2006

Reports speedup

- Run time compared with a standard machine

How SPEC Works

User runs n programs on test machine

Records run-time conditions

Records program run-time in seconds

$$T_i^{test} \quad , \quad i = 1, 2, \ldots, n$$

SPEC provides run-times on reference machine

Sun Fire V490

2100 MHz UltraSPARC-IV+ processor

Powerful symmetric multiprocessing (SMP) server (2006 – 2014)

$$T_i^{ref}$$

User calculates speedup for each program

$$S_i = \frac{T_i^{ref}}{T_i^{test}} \quad , \quad i = 1, 2, \ldots, n$$

User calculates geometric mean of speedups

$$S(\text{test machine on ref}) = \left[\prod_{i=1}^{n} \left(\frac{T_i^{ref}}{T_i^{test}} \right) \right]^{\frac{1}{n}}$$

$$S(\text{machine A compared to machine B}) = \frac{S(\text{machine A on ref})}{S(\text{machine B on ref})}$$

Typical Reference Run Times

Cint2017 Programs

Program	Language	KLOC	Application	Ref Run Time
600.perlbench_s	C	362	Perl interpreter	1773
602.gcc_s	C	1,304	GNU C compiler	3982
605.mcf_s	C	3	Route planning	4709
620.omnetpp_s	C++	134	Discrete Event simulation - computer network	1630
623.xalancbmk_s	C++	520	XML to HTML conversion via XSLT	1413
625.x264_s	C	96	Video compression	1770
631.deepsjeng_s	C++	10	Artificial Intelligence: alpha-beta tree search (Chess)	1434
641.leela_s	C++	21	Artificial Intelligence: Monte Carlo tree search (Go)	1706
648.exchange2_s	Fortran	1	Artificial Intelligence: recursive solution generator (Sudoku)	2948
657.xz_s	C	33	General data compression	6188

KLOC = 1000 lines of code

Typical SPEC Report — 1

SPEC(R) CPU2017 Integer Speed Result
ASUSTeK Computer Inc.
ASUS RS700-E9(Z11PP-D24) Server System
(2.70 GHz, Intel Xeon Gold 6150)

CPU2017 License: 9016 — **Test date: Dec-2017**
Test sponsor: ASUSTeK Computer Inc. — **Hardware availability: Jul-2017**
Tested by: ASUSTeK Computer Inc. — **Software availability: Sep-2017**

Base Benchmarks	Base Thrds	Base Run Time	Base Ratio	Peak Thrds	Peak Run Time	Peak Ratio
600.perlbench_s	72	286	6.22	72	239	7.42
602.gcc_s	72	423	9.42	72	413	9.65
605.mcf_s	72	426	11.1	72	421	11.2
620.omnetpp_s	72	257	6.35	72	248	6.58
623.xalancbmk_s	72	150	9.46	72	140	10.1
625.x264_s	72	150	11.8	72	150	11.8
631.deepsjeng_s	72	280	5.11	72	282	5.08
641.leela_s	72	393	4.34	72	392	4.36
648.exchange2_s	72	220	13.4	72	220	13.4
657.xz_s	72	280	22.1	72	277	22.3
SPECspeed2017_int_base			8.87			
SPECspeed2017_int_peak						9.16

Base = standard configuration **Peak = specialist configuration**

Typical SPEC Report — 2

```
                          HARDWARE
                         --------
      CPU Name: Intel Xeon Gold 6150
      Max MHz.: 3700
       Nominal: 2700
       Enabled: 36 cores, 2 chips
     Orderable: 1, 2 chip(s)
      Cache L1: 32 KB I + 32 KB D on chip per core
            L2: 1 MB I+D on chip per core
            L3: 24.75 MB I+D on chip per chip
         Other: None
        Memory: 768 GB (24 x 32 GB 2Rx4 PC4-2666V-R)
       Storage: 1 x 240 GB SATA SSD
         Other: None
                          SOFTWARE
                          --------
            OS: Red Hat Enterprise Linux Server release 7.3 (x86_64)
                Kernel 3.10.0-514.el7.x86_64
      Compiler: C/C++: Version 18.0.0.128 of Intel C/C++ Compiler;
                Fortran: Version 18.0.0.128 of Intel Fortran Compiler
      Parallel: Yes
      Firmware: Version 0601 released Oct-2017
    File System: xfs
   System State: Run level 3 (multi-user)
  Base Pointers: 64-bit
  Peak Pointers: 32/64-bit
         Other: jemalloc: jemalloc memory allocator library V5.0.1
```

Some Cint2017 Results

Processor	Clock (GHz)	Total Chips	Total Cores	Total Threads	Cint 2017 Base
Intel Xeon E-2278G (Lenovo)	3.4	1	8	16	13.2
Intel Xeon E-2278G (Asus)	3.4	1	8	8	12.4
Intel Xeon E-2278G (Fujitsu)	3.4	1	8	16	12.1
Intel Xeon Gold 6250 (Lenovo)	3.9	2	16	32	12.6
Intel Xeon Gold 6256 (Lenovo)	3.6	2	24	48	12.6
Intel Core i9-9900K	3.6	1	6	6	11.2
Intel Core i7-9700K	3.6	1	8	8	10.6

Some Comments on Cint2017 Results

Auto parallel

High level Cint code not explicitly threaded for parallel processing

Auto parallel compiler creates parallel threads using heuristics

Provides limited speed up (or even degradation)

Intel Xeon E-2278G with 3.4 GHz clock

Fastest CPU in Cint2017 tests

16 threads slightly faster than 8 threads

- Communication between more threads can slow processing

16 threads from different manufacturer slightly slower

Intel Xeon Gold 6256 with 3.6 GHz clock

Cint with 48 threads slightly slower than E-2278G

With 3.6 GHz clock, expect 3.6 GHz / 3.4 GHz = 1.06 faster

48 threads slower than 16 threads

Some Cfp2017 Results

Processor	Clock (GHz)	Total Chips	Total Cores	Total Threads	Cfp 2017 Base	Cint 2017 Base
Intel Xeon Platinum 8280L	2.7	8	224	224	244	10.5
Intel Xeon Platinum 8280L	2.7	4	112	112	232	10.5
Intel Xeon Platinum 8156	3.6	2	8	8	53.8	8.26
Intel Core i9-9900K	3.6	1	6	6	33.2	11.2
Intel Core i7-9700K	3.6	1	8	8	32.3	10.6

Actual Sources of Performance Improvement

1978 Clock speed of 8086 is 4 MHz

2008 Xeon (clock speed of 4 GHz) is 100,000 times faster

- Clock speedup = 4 GHz / 4 MHz = 1000
- Structural speedup = 100,000 / 1000 = 100
  - Reducing waiting time between operations
  - Performing operations in parallel

No more clock speedup

- Pentium 4 clock rate (4 GHz) = 4 x Pentium III clock (1 GHz)
- Clock speedup 1 GHz $\rightarrow$ 4 GHz required structural slowdown
  - Pentium 4 at 1 GHz slower than Pentium III at 1 GHz
  - Run Pentium III at 4 GHz $\Rightarrow$ melt CPU
- Clock speed $\rightarrow$ physical limit of about 10 GHz
  - Signal takes clock cycle to cross Pentium 4 at speed of light
- Future speedup comes from structural improvements
  - More cores
  - Better architectures