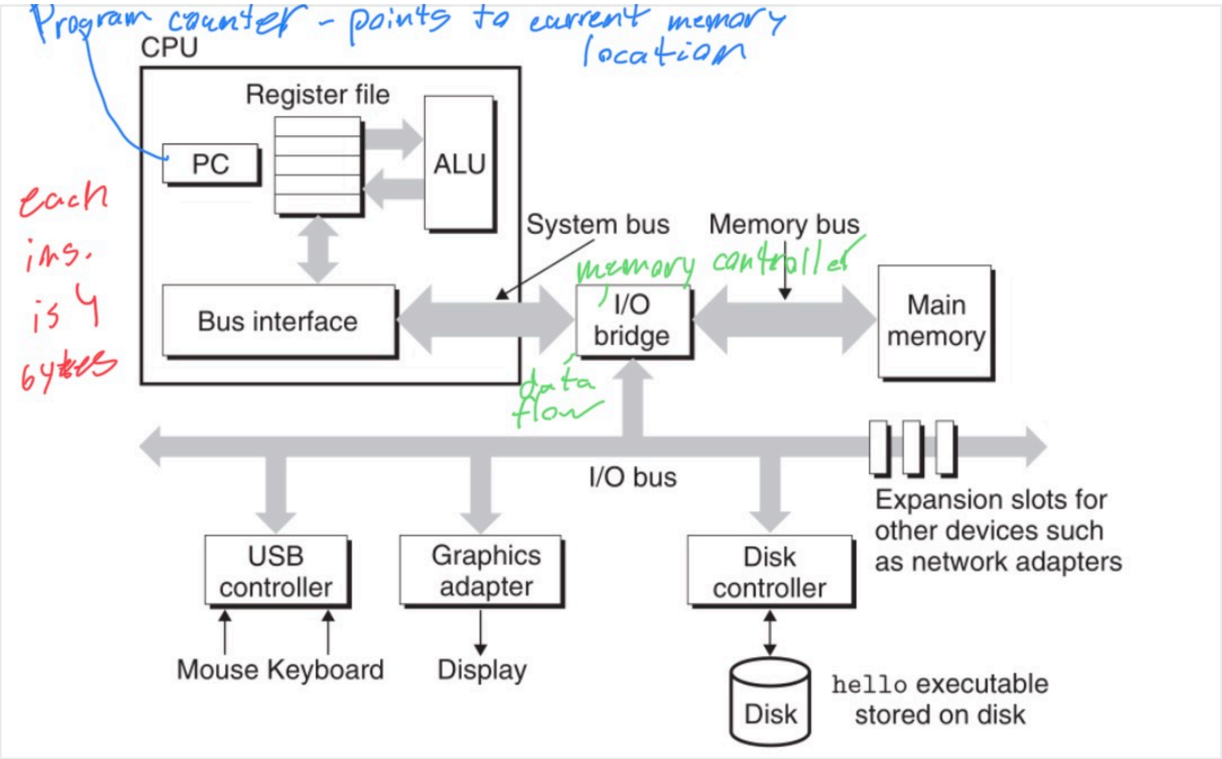
Slides 1



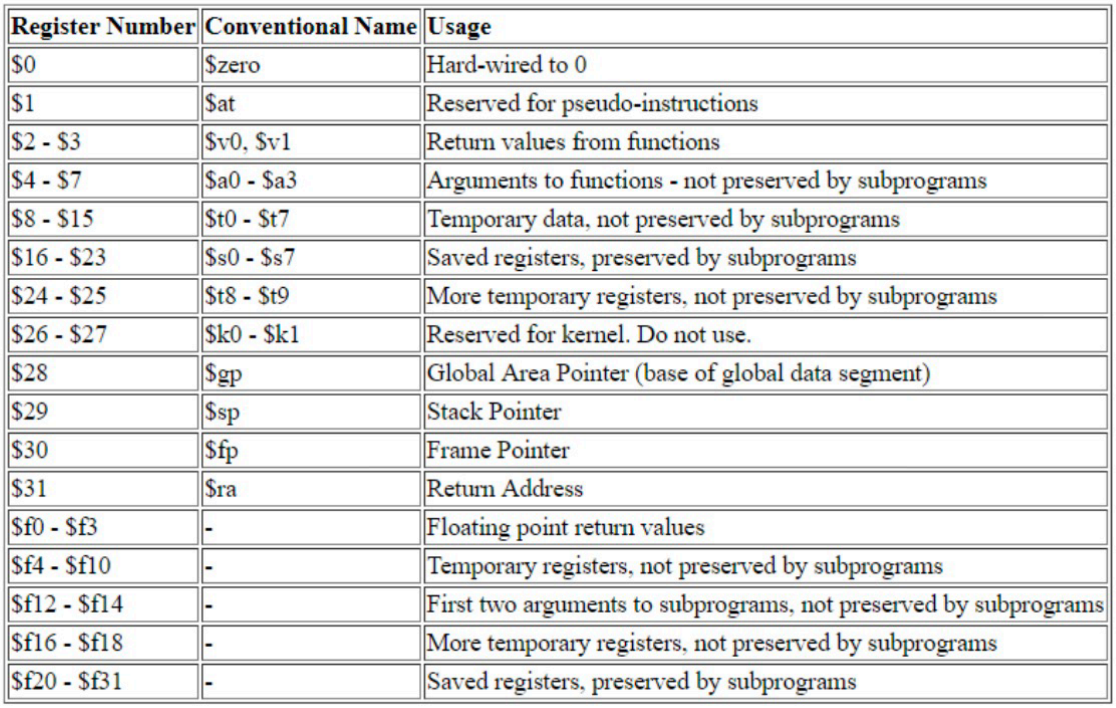
Program counter - points to current memory location

each ins. is 4 bytes

memory controller

data flow

MIPS Registers



Register Number	Conventional Name	Usage
$0	$zero	Hard-wired to 0
$1	$at	Reserved for pseudo-instructions
$2 - $3	$v0, $v1	Return values from functions
$4 - $7	$a0 - $a3	Arguments to functions - not preserved by subprograms
$8 - $15	$t0 - $t7	Temporary data, not preserved by subprograms
$16 - $23	$s0 - $s7	Saved registers, preserved by subprograms
$24 - $25	$t8 - $t9	More temporary registers, not preserved by subprograms
$26 - $27	$k0 - $k1	Reserved for kernel. Do not use.
$28	$gp	Global Area Pointer (base of global data segment)
$29	$sp	Stack Pointer
$30	$fp	Frame Pointer
$31	$ra	Return Address
$f0 - $f3	-	Floating point return values
$f4 - $f10	-	Temporary registers, not preserved by subprograms
$f12 - $f14	-	First two arguments to subprograms, not preserved by subprograms
$f16 - $f18	-	More temporary registers, not preserved by subprograms
$f20 - $f31	-	Saved registers, preserved by subprograms

High-level Language

- Implementing a swap function in C

gets offset

add base + offset

-load

-store

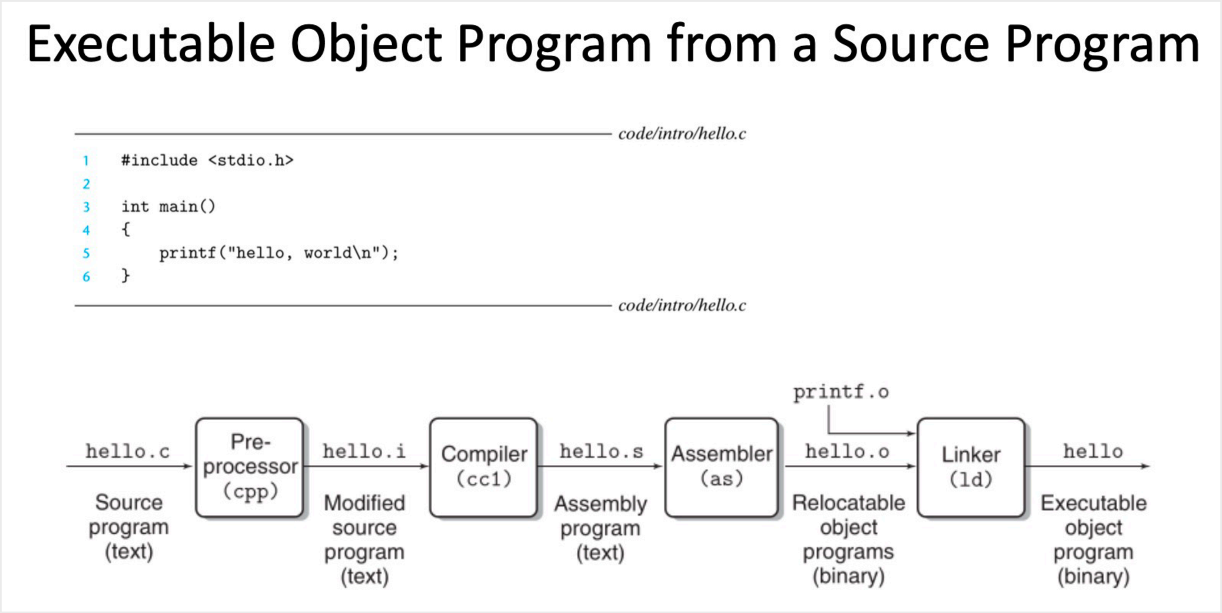
```
swap(int v[], int k) {
   int temp;
   temp = v[k];
   v[k] = v[k+1];
   v[k+1] = temp;
}
```

Compiler
[illegible] offset

```
swap:
       multi  $2, $5, 4
       add    $2, $4, $2
       lw     $15, 0($2)
       lw     $16, 4($2)
       sw     $16, 0($2)
       sw     $15, 4($2)
       jr     $31
```

load word

store word



Executable Object Program from a Source Program

code/intro/hello.c

```
#include <stdio.h>

int main()
{
    printf("hello, world\n");
}
```

code/intro/hello.c

printf.o

hello.c
Source program (text)

Pre-processor (cpp)

hello.i
Modified source program (text)

Compiler (cc1)

hello.s
Assembly program (text)

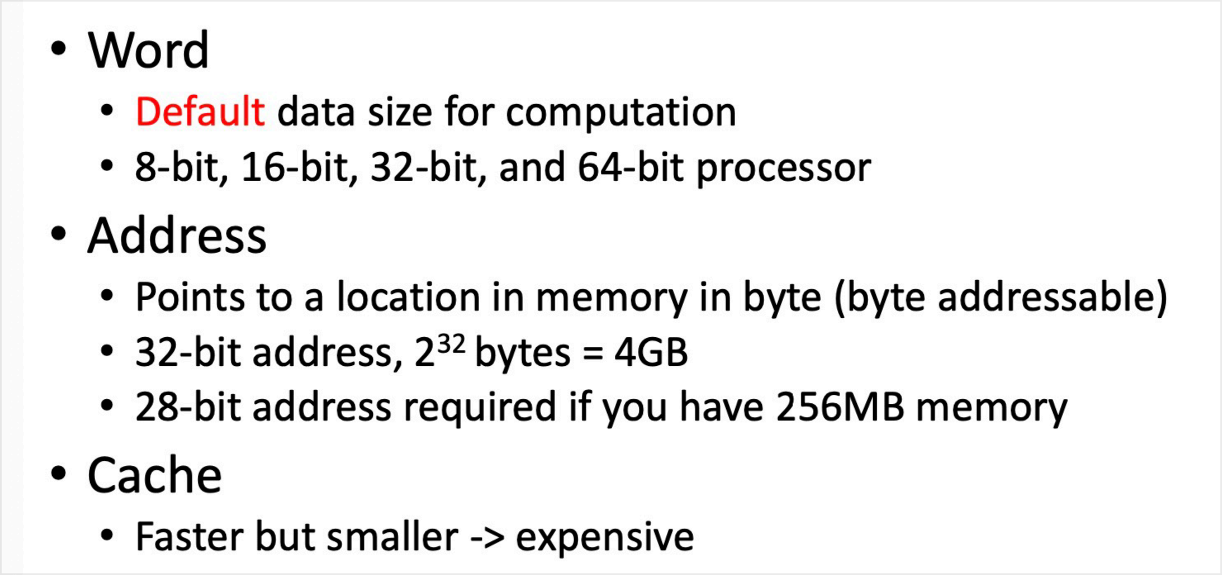
Assembler (as)

hello.o
Relocatable object programs (binary)

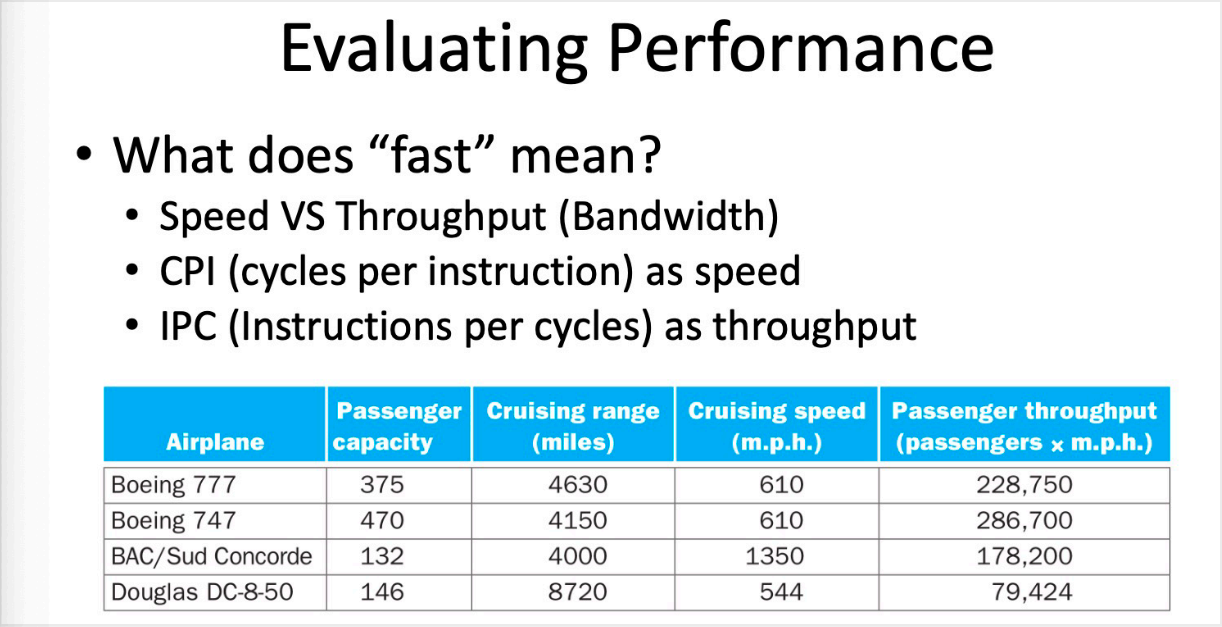
Linker (ld)

hello
Executable object program (binary)

Microarchitecture: implementation to support ISA
ISA: the interface.



- Word
  - Default data size for computation
  - 8-bit, 16-bit, 32-bit, and 64-bit processor
- Address
  - Points to a location in memory in byte (byte addressable)
  - 32-bit address, 2^{32} bytes = 4GB
  - 28-bit address required if you have 256MB memory
- Cache
  - Faster but smaller -> expensive

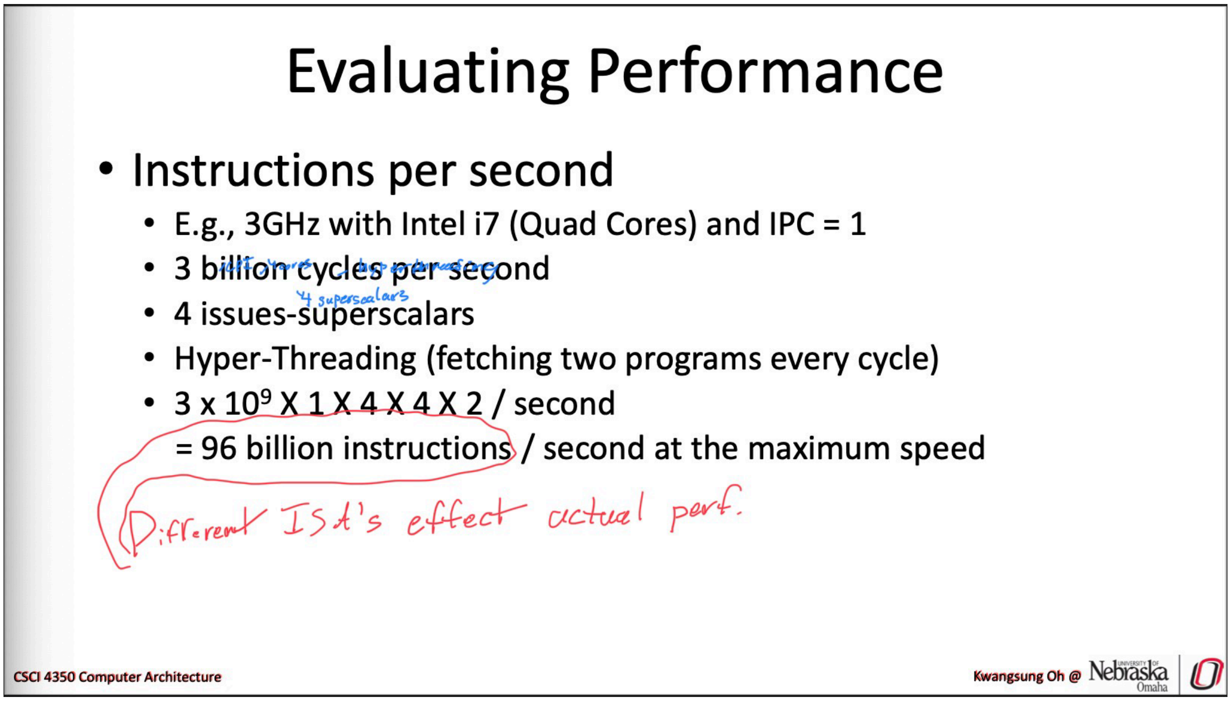


Evaluating Performance

- What does “fast” mean?
  - Speed VS Throughput (Bandwidth)
  - CPI (cycles per instruction) as speed
  - IPC (Instructions per cycles) as throughput

Airplane	Passenger capacity	Cruising range (miles)	Cruising speed (m.p.h.)	Passenger throughput (passengers x m.p.h.)
Boeing 777	375	4630	610	228,750
Boeing 747	470	4150	610	286,700
BAC/Sud Concorde	132	4000	1350	178,200
Douglas DC-8-50	146	8720	544	79,424

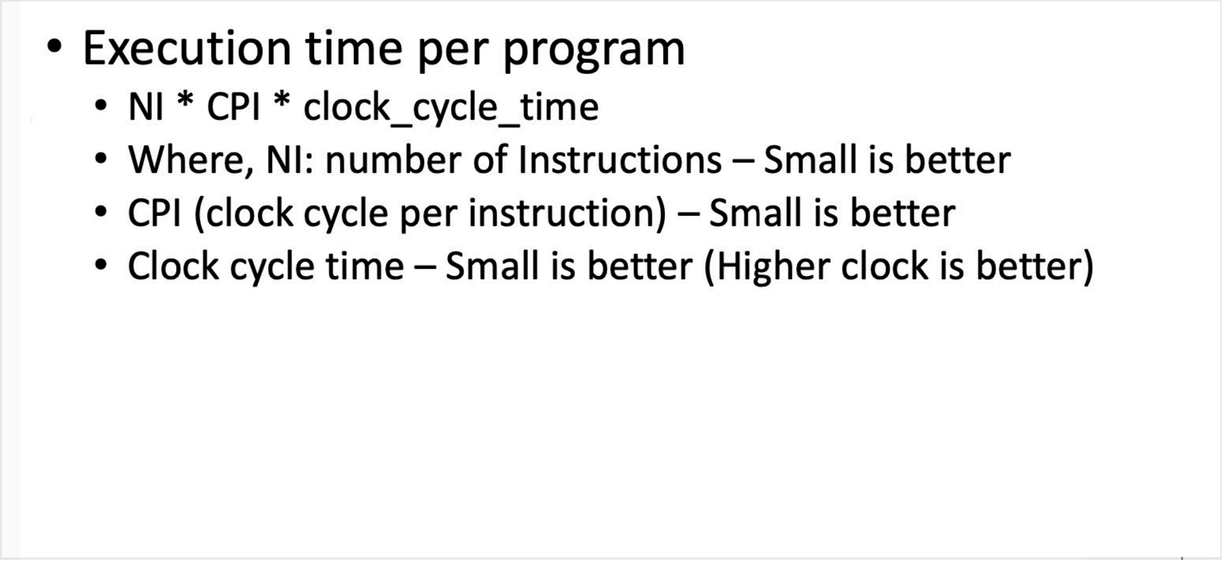
Improve throughput with: pipelining, hyper threading, clock speed



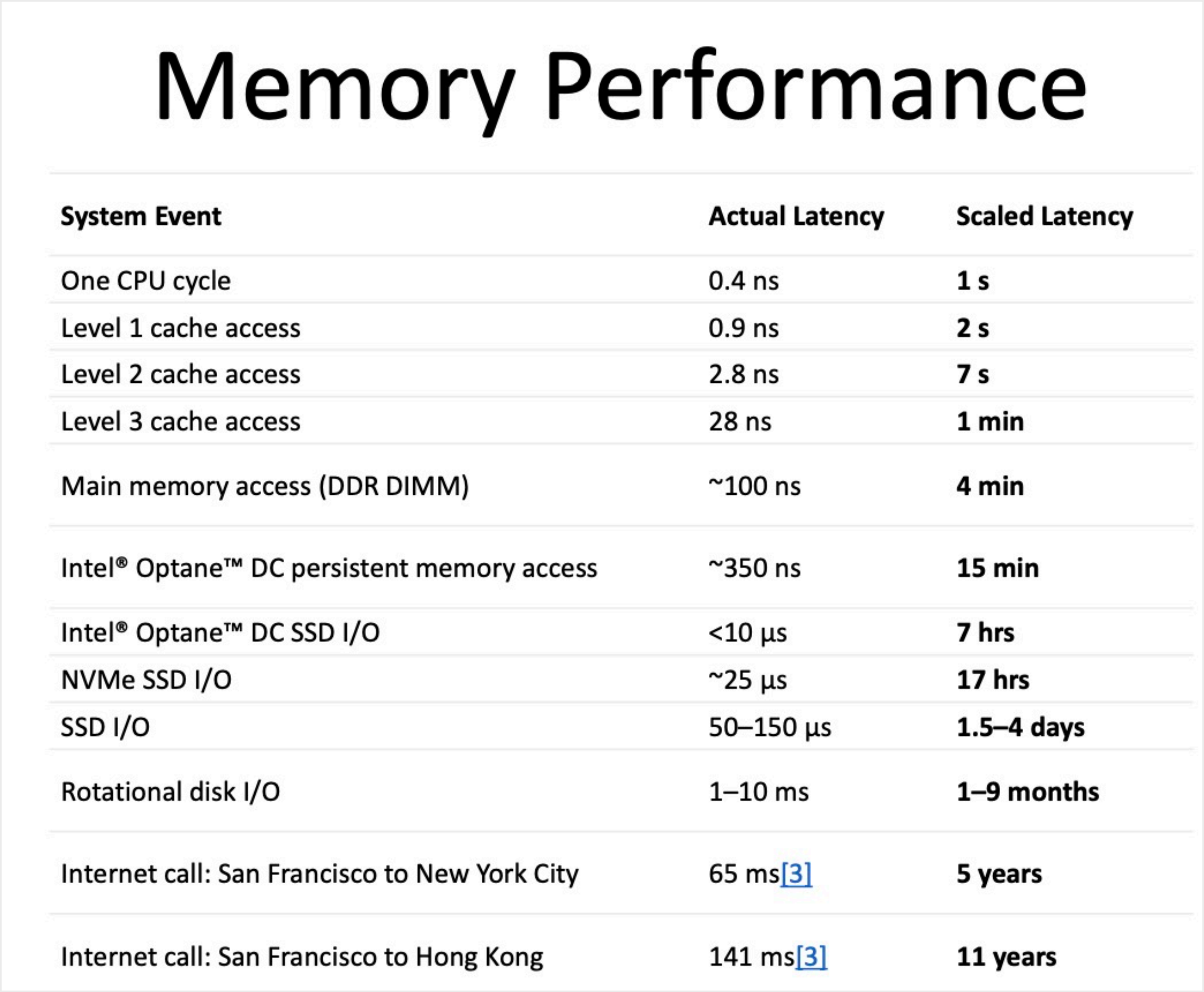
Evaluating Performance

- Instructions per second
  - E.g., 3GHz with Intel i7 (Quad Cores) and IPC = 1
  - 3 billion cycles per second
    IPC1 4cores hyperthreading
    4 superscalars
  - 4 issues-superscalars
  - Hyper-Threading (fetching two programs every cycle)
  - 3×10^9 X 1 X 4 X 4 X 2 / second
    = 96 billion instructions / second at the maximum speed

Different ISA's effect actual perf.



- Execution time per program
  - NI * CPI * clock_cycle_time
  - Where, NI: number of Instructions – Small is better
  - CPI (clock cycle per instruction) – Small is better
  - Clock cycle time – Small is better (Higher clock is better)



Memory Performance

System Event	Actual Latency	Scaled Latency
One CPU cycle	0.4 ns	**1 s**
Level 1 cache access	0.9 ns	**2 s**
Level 2 cache access	2.8 ns	**7 s**
Level 3 cache access	28 ns	**1 min**
Main memory access (DDR DIMM)	~100 ns	**4 min**
Intel® Optane™ DC persistent memory access	~350 ns	**15 min**
Intel® Optane™ DC SSD I/O	<10 μs	**7 hrs**
NVMe SSD I/O	~25 μs	**17 hrs**
SSD I/O	50–150 μs	**1.5–4 days**
Rotational disk I/O	1–10 ms	**1–9 months**
Internet call: San Francisco to New York City	65 ms[3]	**5 years**
Internet call: San Francisco to Hong Kong	141 ms[3]	**11 years**

4 ghz dual core

3-way scalar, hyper threaded