Computer Architecture
CSCI 4350

Introduction

Kwangsung Oh

kwangsungoh@unomaha.edu

http://faculty.ist.unomaha.edu/kwangsungoh

University of Nebraska Omaha

Today

- Class information
- Computer Architecture Overview
- Computer Performance

Class Information

- Instructor
  - Kwangsung Oh
  - Assistant professor at the dept. of CS
  - Distributed computing systems researcher
  - Five years working experience as a software engineer at Samsung

Class Information

- Time & Place
  - Tue/Thu 4:30 pm – 5:45 pm (Section 001)
  - Mon/Wed 10:30 am – 11:45 am (Section 002)
  - PKI 276
- Office Hours
  - Tue/Thu 3:20 pm – 4:20 pm
  - Mon/Wed 9:20 am – 10:20 am
  - By appointment
  - PKI 177D

Class Information

- Textbook
  - Patterson and Hennessy, *Computer Organization and Design: The hardware/software interface*, Fifth Edition
  - Bryant and O'Halloran, *Computer Systems A Programmer's Perspective*, Second Edition

Class Information

- Evaluation (tentative)
  - Homework: 35% (7 HWs)
  - Project: 25% (3 Projects)
  - Midterm (Oct/12): 20%
  - Final (Dec/14): 20%
  - Extra credit

Point	Grade
97 – 100	A+
94 – 96	A
90 – 93	A-
87 – 89	B+
84 – 86	B
80 – 83	B-
75 – 79	C+
70 – 74	C
65 – 69	C-
60 – 64	D+
55 – 59	D
50 – 54	D-

Estimated Schedule (Tentative)

• Introduction	1 week
• ISA (Instruction Set Architecture)	2 weeks
• Linking	1-2 weeks
• Arithmetic	2 weeks
• Pipelining	2 weeks
• Cache and Memory Hierarchy	2 weeks
• Virtual Memory	2 weeks

Collaboration

Finish all work by yourself

Attendance Policy

- No penalty until 4 non-excused absences
- 3 points penalty per absence after 4 non-excused absences
  - e.g., 7 classes missed; 9 points will be deducted from the overall grade
- A valid proof e.g., Dr's note

ABET Accreditation

- The CS Programs are currently pursuing accreditation by **ABET**, the Accreditation Board for Engineering and Technology. For accreditation purposes, this organization requires that we keep samples of student assignments, quizzes, and exams. Therefore, the instructor may, at random, retain a copy of your quiz, exam or assignment with your name removed.

Today

- Class information
- Computer architecture overview
- Computer performance

Which Code is faster?

```
void copyij(int src[2048][2048],
            int dst[2048][2048])
{
  int i,j;
  for (i = 0; i < 2048; i++)
    for (j = 0; j < 2048; j++)
      dst[i][j] = src[i][j];
}
```

```
void copyji(int src[2048][2048],
            int dst[2048][2048])
{
  int i,j;
  for (j = 0; j < 2048; j++)
    for (i = 0; i < 2048; i++)
      dst[i][j] = src[i][j];
}
```

4.3ms 2.0 GHz Intel Core i7 Haswell 81.8ms

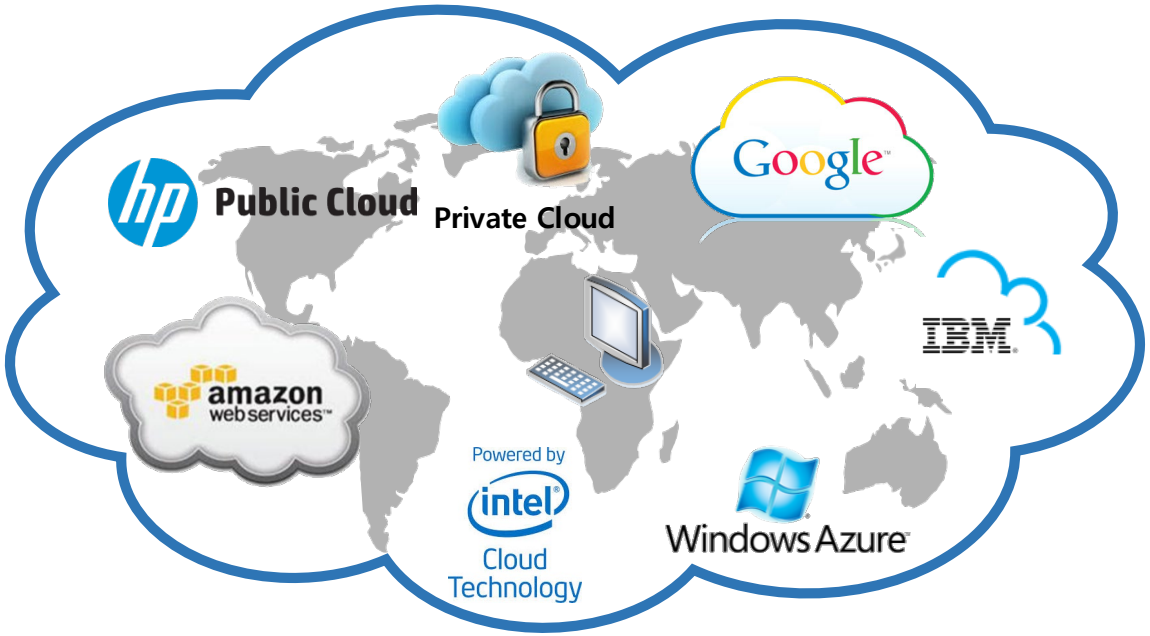
What are the values?

```
int a[2] = {1, 2}
double c = 4.0;
int b[2] = {3, 4}


a[0] = 1
a[1] = 2
a[2] = ?
a[3] = ?
a[4] = ?
a[5] = ?
```

Computer?

From a Server to Cloud Computing



What is inside your PC?

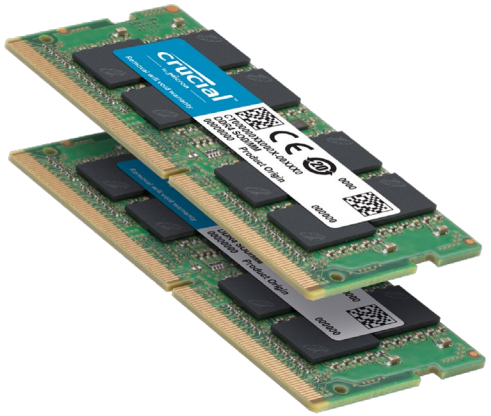
http://www.suekayton.com/careandfeeding.htm

After Cleaning

https://www.ebay.com/itm/324151964685

Components

https://www.engadget.com/2017/08/07/intel-18-core-i9-specs/



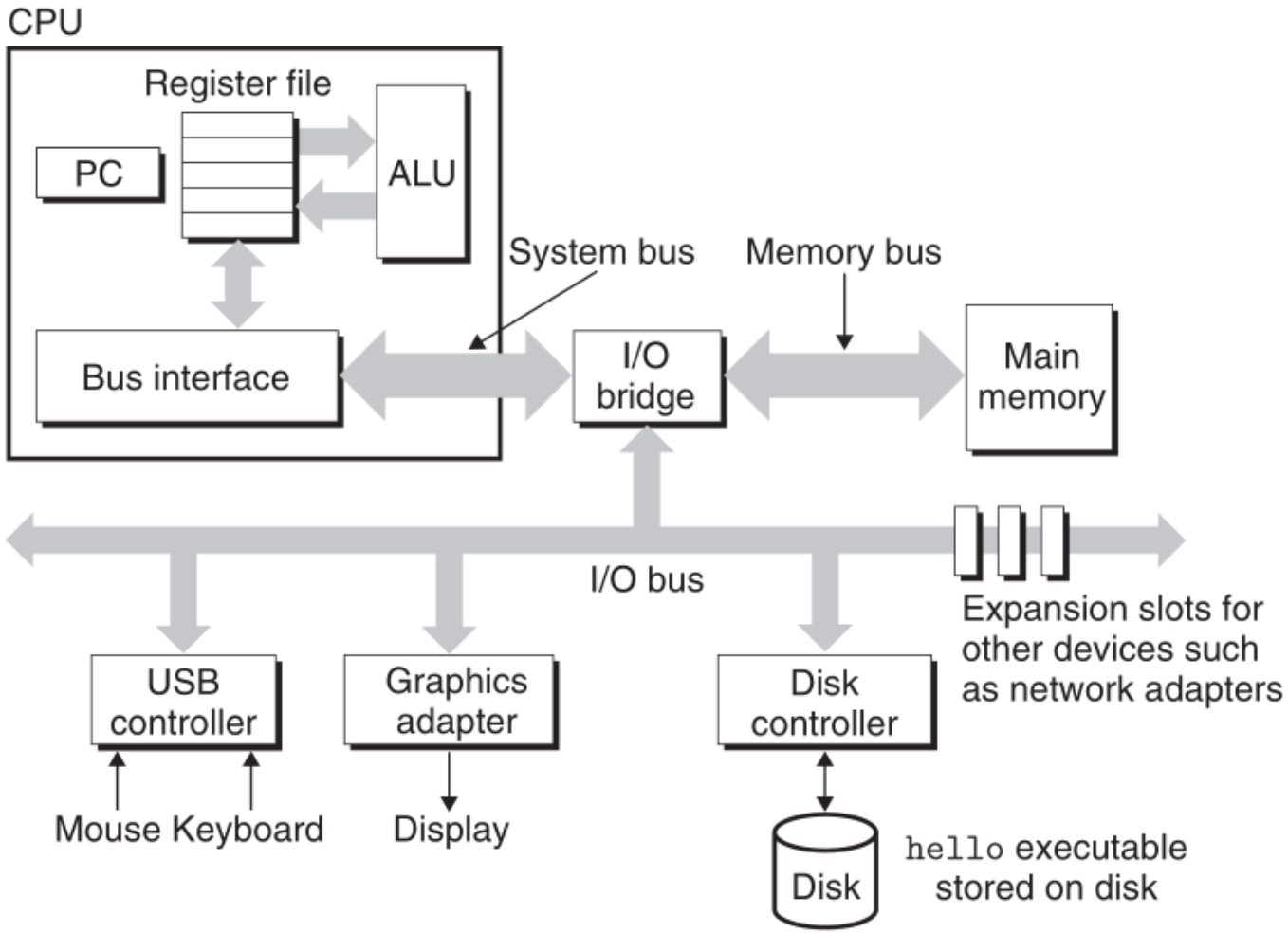
https://www.crucial.com/usa/en/memory-laptop

https://www.flipkart.com/zebronics-zeb-g31-motherboard/p/itme6fxvyzfndnwz

Components

- Processor(s)
  - CPU (Central Processing Unit)
- Memory
  - Caches, Main memory, SSD, HDD, Blue ray …
- Input
  - Keyboard, Mouse, Touch Screen, Scanner …
- Output
  - Monitor, Printer, Speaker …
- Mother Board (Chipset)

The Organization of Computer



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

Assembly Language

- Arithmetic
  - ADD, SUB, MUL, DIV …
- Data transfer
  - LOAD and STORE
- Logical
  - AND, OR, NOR, SLL …
- Conditional Branch
  - BEQ, BNE, SLT …

MIPS assembly language

Category	Instruction	Example	Meaning	Comments
Arithmetic	add	add $s1,$s2,$s3	$s1 = $s2 + $s3	Three register operands
	subtract	sub $s1,$s2,$s3	$s1 = $s2 – $s3	Three register operands
	add immediate	addi $s1,$s2,20	$s1 = $s2 + 20	Used to add constants
Data transfer	load word	lw $s1,20($s2)	$s1 = Memory[$s2 + 20]	Word from memory to register
	store word	sw $s1,20($s2)	Memory[$s2 + 20] = $s1	Word from register to memory
	load half	lh $s1,20($s2)	$s1 = Memory[$s2 + 20]	Halfword memory to register
	load half unsigned	lhu $s1,20($s2)	$s1 = Memory[$s2 + 20]	Halfword memory to register
	store half	sh $s1,20($s2)	Memory[$s2 + 20] = $s1	Halfword register to memory
	load byte	lb $s1,20($s2)	$s1 = Memory[$s2 + 20]	Byte from memory to register
	load byte unsigned	lbu $s1,20($s2)	$s1 = Memory[$s2 + 20]	Byte from memory to register
	store byte	sb $s1,20($s2)	Memory[$s2 + 20] = $s1	Byte from register to memory
	load linked word	ll $s1,20($s2)	$s1 = Memory[$s2 + 20]	Load word as 1st half of atomic swap
	store condition. word	sc $s1,20($s2)	Memory[$s2+20]=$s1;$s1=0 or 1	Store word as 2nd half of atomic swap
	load upper immed.	lui $s1,20	$s1 = 20 * 2^{16}	Loads constant in upper 16 bits
Logical	and	and $s1,$s2,$s3	$s1 = $s2 & $s3	Three reg. operands; bit-by-bit AND
	or	or $s1,$s2,$s3	$s1 = $s2 | $s3	Three reg. operands; bit-by-bit OR
	nor	nor $s1,$s2,$s3	$s1 = ~ ($s2 | $s3)	Three reg. operands; bit-by-bit NOR
	and immediate	andi $s1,$s2,20	$s1 = $s2 & 20	Bit-by-bit AND reg with constant
	or immediate	ori $s1,$s2,20	$s1 = $s2 | 20	Bit-by-bit OR reg with constant
	shift left logical	sll $s1,$s2,10	$s1 = $s2 << 10	Shift left by constant
	shift right logical	srl $s1,$s2,10	$s1 = $s2 >> 10	Shift right by constant
Conditional branch	branch on equal	beq $s1,$s2,25	if ($s1 == $s2) go to PC + 4 + 100	Equal test; PC-relative branch
	branch on not equal	bne $s1,$s2,25	if ($s1!= $s2) go to PC + 4 + 100	Not equal test; PC-relative
	set on less than	slt $s1,$s2,$s3	if ($s2 < $s3) $s1 = 1; else $s1 = 0	Compare less than; for beq, bne
	set on less than unsigned	sltu $s1,$s2,$s3	if ($s2 < $s3) $s1 = 1; else $s1 = 0	Compare less than unsigned
	set less than immediate	slti $s1,$s2,20	if ($s2 < 20) $s1 = 1; else $s1 = 0	Compare less than constant
	set less than immediate unsigned	sltiu $s1,$s2,20	if ($s2 < 20) $s1 = 1; else $s1 = 0	Compare less than constant unsigned
Unconditional jump	jump	j 2500	go to 10000	Jump to target address
	jump register	jr $ra	go to $ra	For switch, procedure return
	jump and link	jal 2500	$ra = PC + 4; go to 10000	For procedure call

FIGURE 2.1 MIPS assembly language revealed in this chapter. This information is also found in Column 1 of the MIPS Reference Data Card at the front of this book.

Programming Languages

- High-level language (Human readable)
  - Python, Java Script, C, C++, java …
- Low-level language (Still human readable)
  - Assembly
- Machine language
  - Instruction Set

Machine Language (MIPS)

- Implementing a swap function in binary

```
000000 00101 00010 00000 00100 011000
000000 00100 00010 00010 00000 100001
100011 01111 00010 00000 00000 000000
100011 10000 10010 00000 00000 000100
101011 10000 10010 00000 00000 000000
101011 01111 00010 00000 00000 000100
000000 11111 00000 00000 00000 001000
```

High-level Language

- Implementing a swap function in C

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


```
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
```

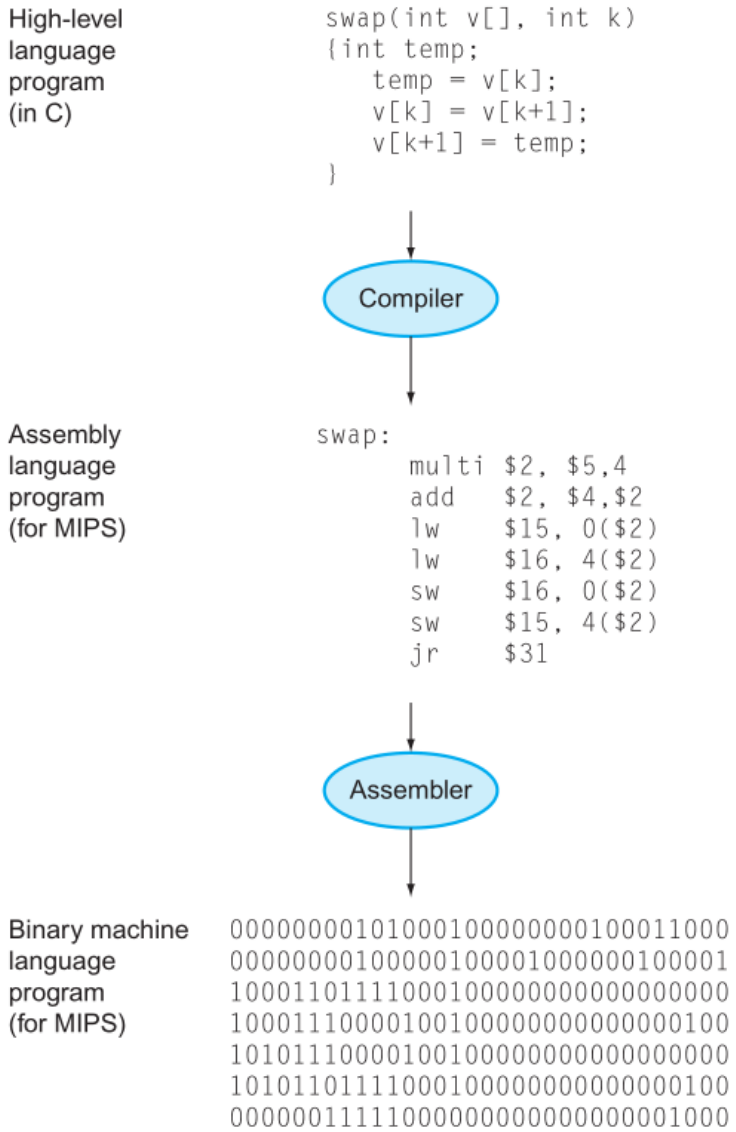
Assembly Language

- Implementing a swap function in assembly

```
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
```


```
000000 00101 00010 00000 00100 011000
000000 00100 00010 00010 00000 100001
100011 01111 00010 00000 00000 000000
100011 10000 10010 00000 00000 000100
101011 10000 10010 00000 00000 000000
101011 01111 00010 00000 00000 000100
000000 11111 00000 00000 00000 001000
```

From High-Level to Instruction

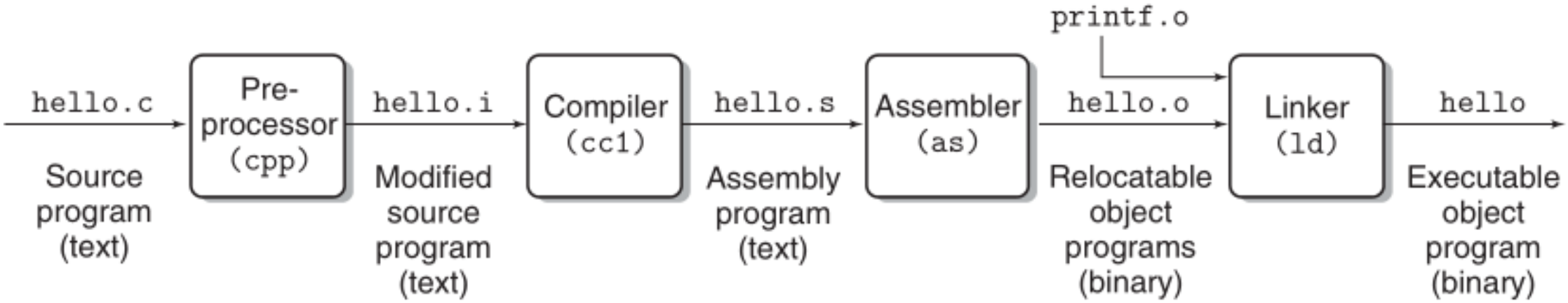


Executable Object Program from a Source Program

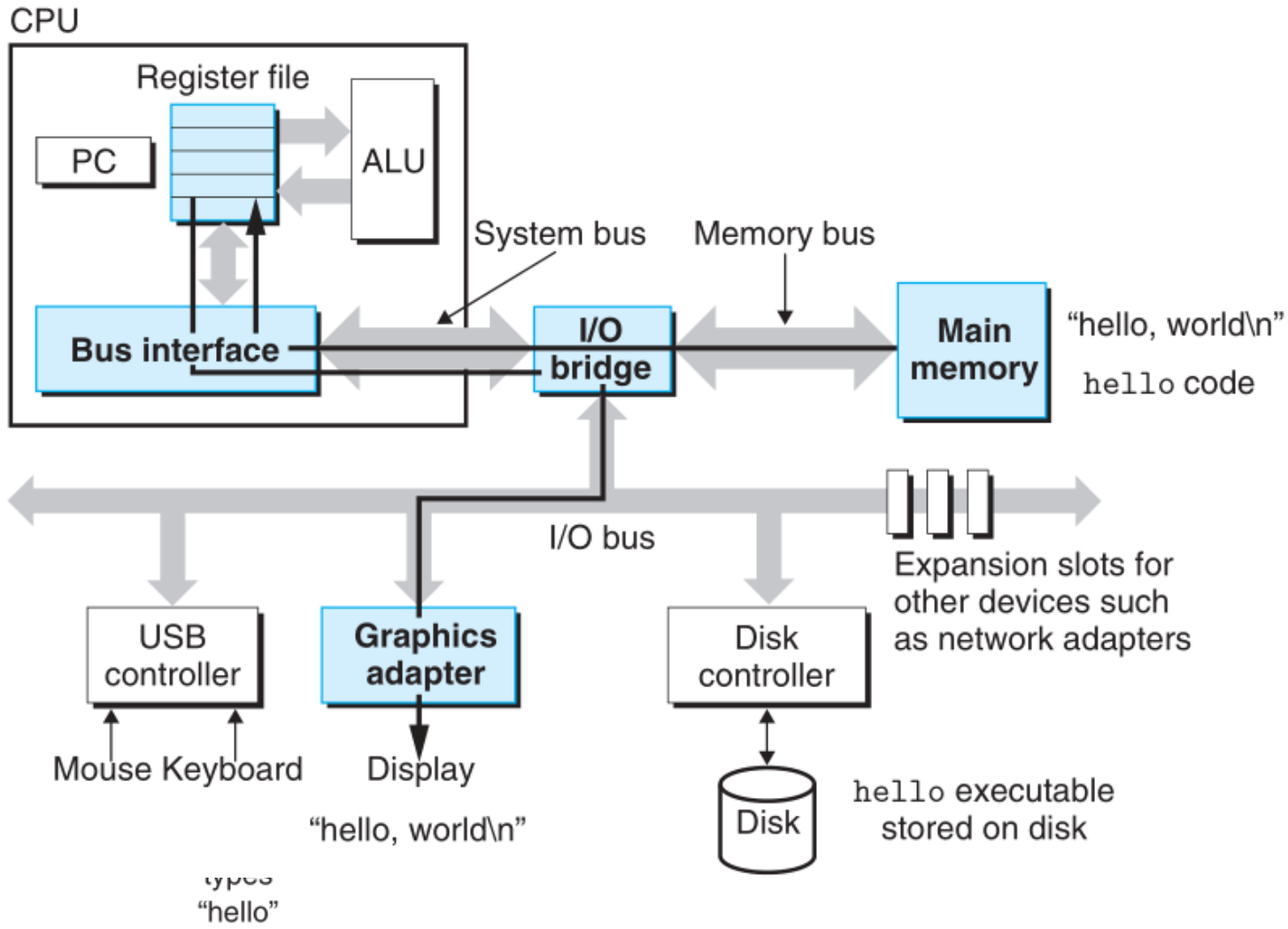
code/intro/hello.c

```
1   #include <stdio.h>
2
3   int main()
4   {
5       printf("hello, world\n");
6   }
```

code/intro/hello.c



Running Program in Shell



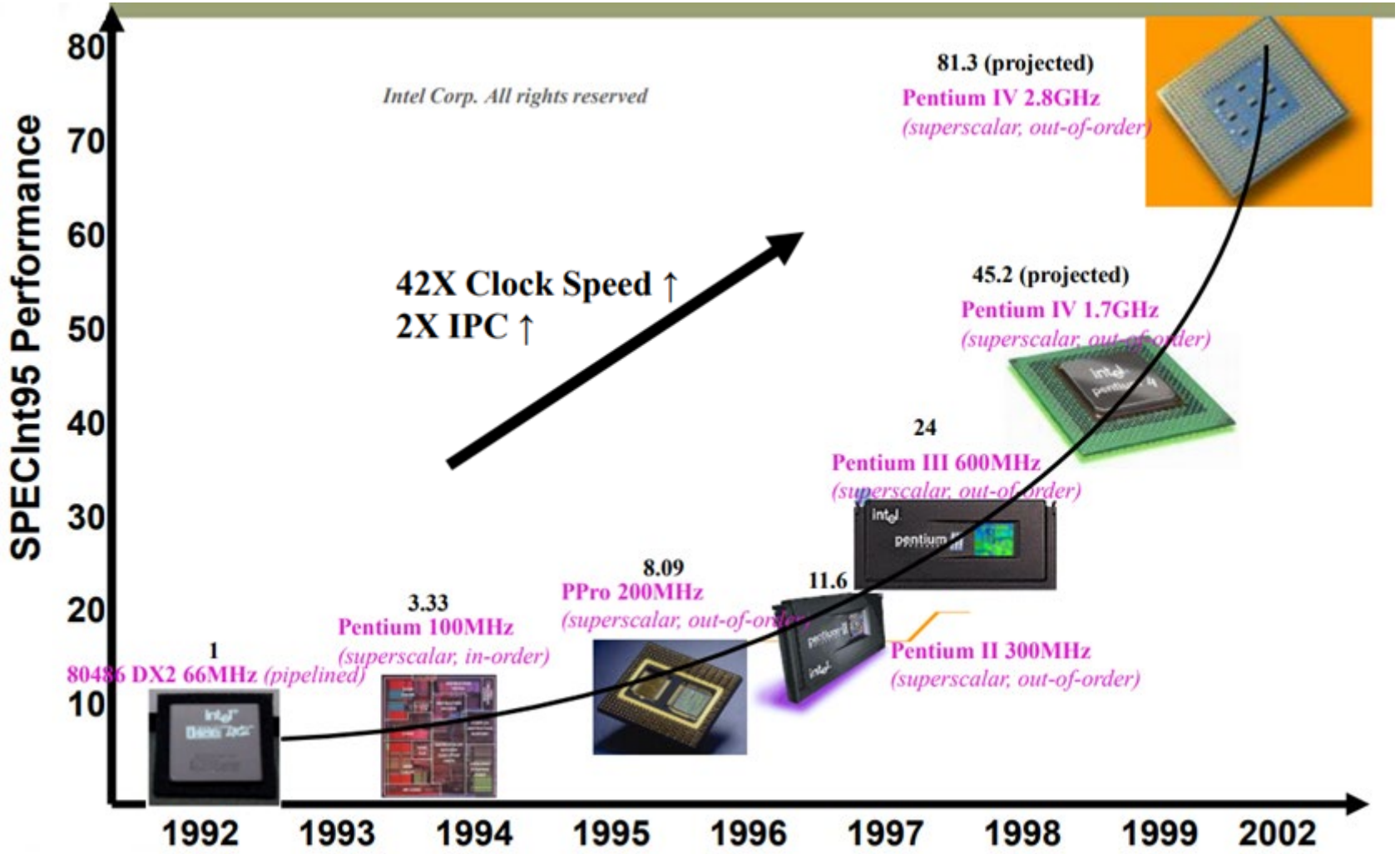
Terminology

- CISC (Complex Instruction Set Computer)
  - Complex instruction
  - Different sizes of instructions
  - Computations on memory
  - x86, …
- RISC (Reduced Instruction Set Computer)
  - Simple instruction
  - Small number of instruction in ISA
  - Computation on registers only
  - MIPS, ARM, PowerPC …

Terminology

- ISA (Instruction Set Architecture)
  - Machine instructions and visible machine states
  - X86, IA64, MIPS…
- Microarchitecture
  - Implementation: according to the ISA
  - Pipelining, super-scalar, caches, branch prediction…
  - Invisible to programmers

Intel Microprocessors



Terminology

- 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

Today

- Class information
- Computer architecture overview
- Computer performance

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 × 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

Evaluating Performance

- Instructions per second
  - E.g., 3GHz with Intel i7 (Quad Cores) and IPC = 1
  - 3 billion cycles per second
  - 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

Evaluating Performance

- 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)

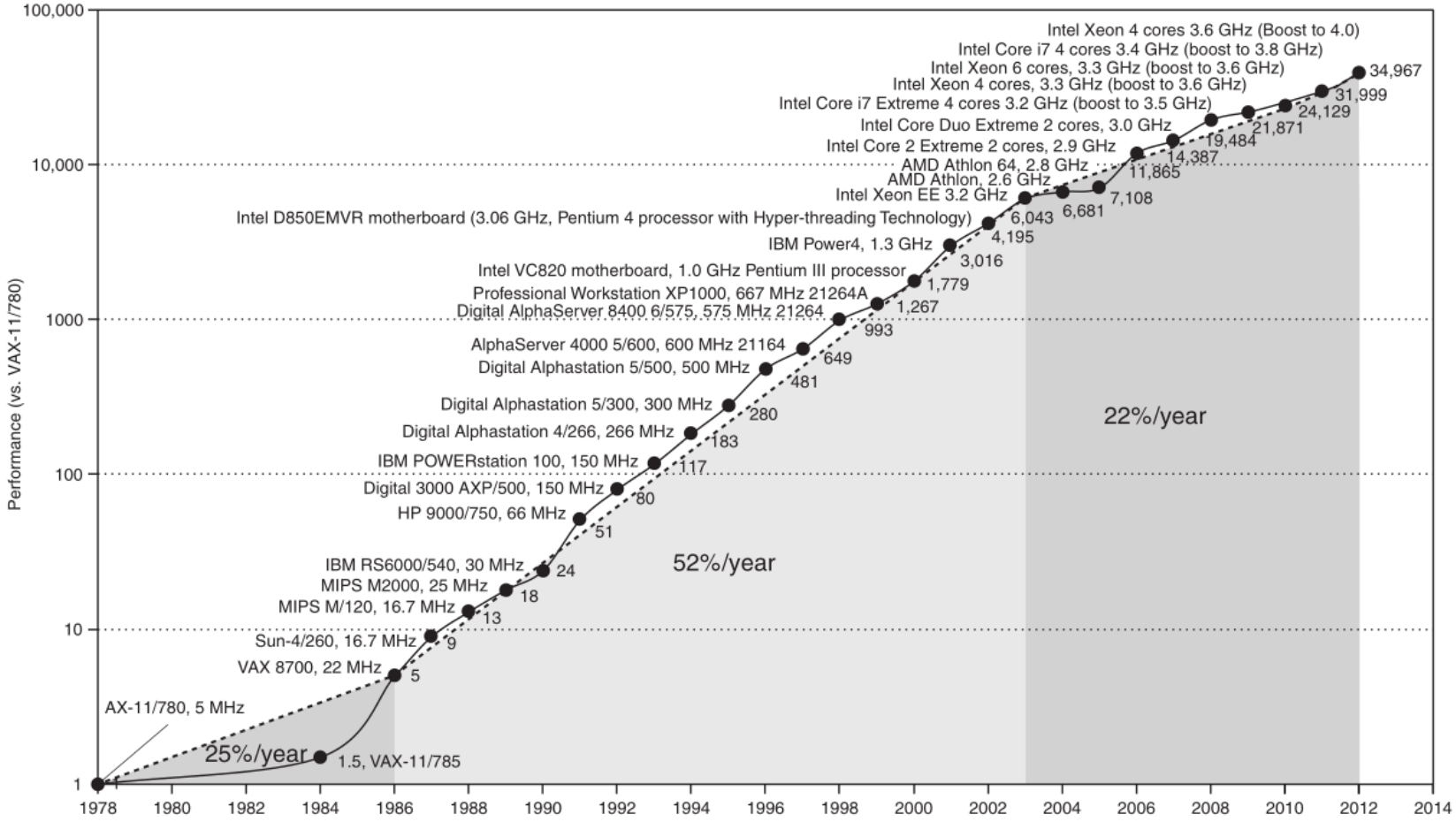
Memory Performance

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**

https://www.prowesscorp.com/computer-latency-at-a-human-scale/

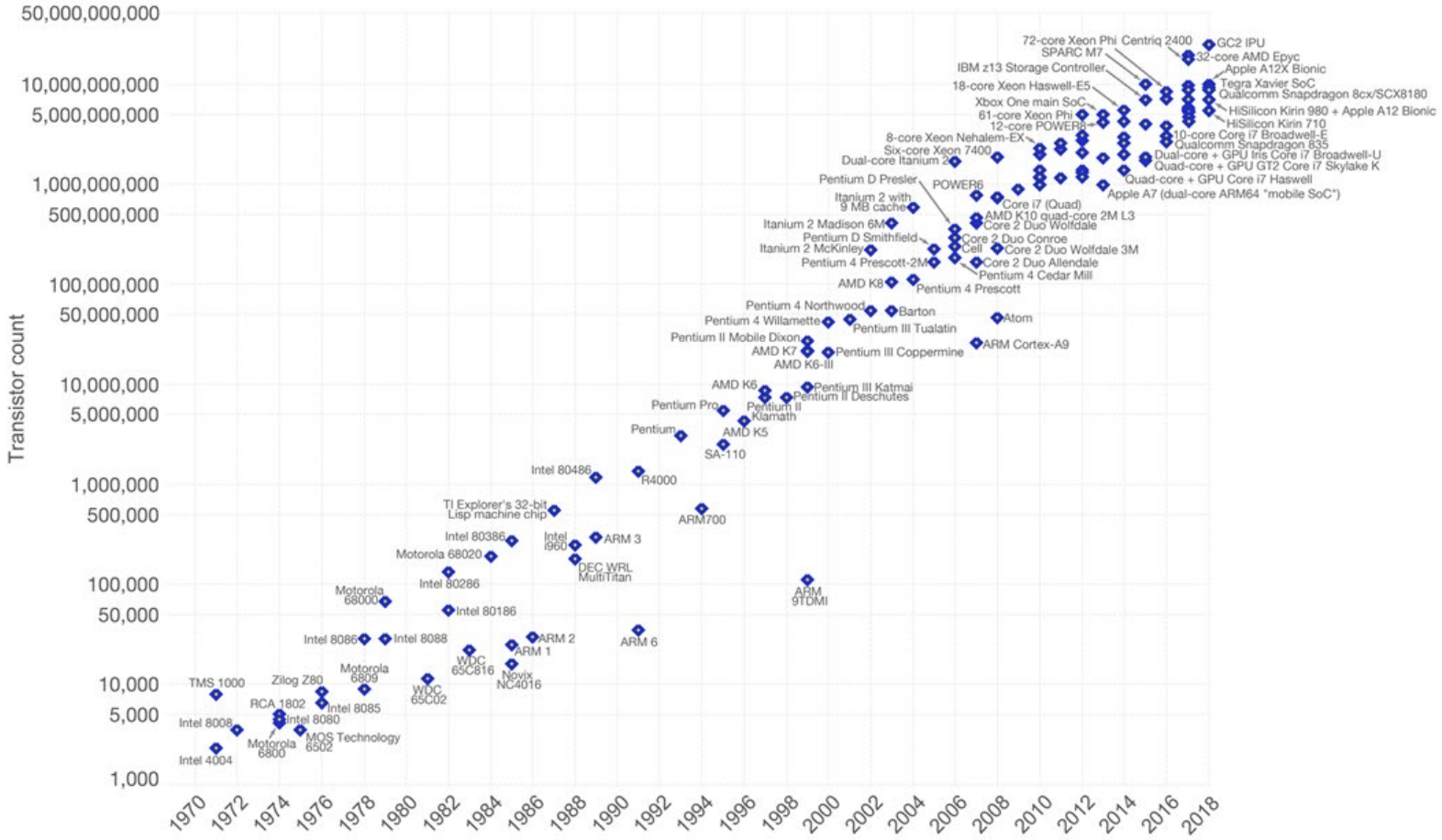
How to Compare Performance?

- SPECint (Integer processing power)



Evaluating Performance

- Moore's law: doubled transistors every two years



https://en.wikipedia.org/wiki/Moore%27s_law