Computer Architecture

Class Overview

1) What is the machine called computer?

2) What is Computer Science?

3) What is Computer Architecture?

4) 행정 사항

Tools

- ❑ 인간은 도구를 만든다
- ❑ Simple machines for farming, fishing, hunting
  - 동력원: 인간의 에너지
  - Transform the direction or magnitude of force

Image of hoe (괭이):

http://en.wikipedia.org/wiki/File:Peasant_in_the_vegetable_garden.JPG

Image of bow and arrow:

http://en.wikipedia.org/wiki/File:Aphaia_pediment_polychrome_model_W-XI_Glyptothek_Munich.jpg

Machines

- Steam engine, 산업혁명
  - 동력원: 화학에너지 (수력, 전기)
  - 효과: 힘 (운동에너지)
  - 기계 (자동장치) - 인간의 힘을 대신함
- Used in all kinds of machines: 자동차, 트랙터, 공장기계, …

Image of steam engine:
http://en.wikipedia.org/wiki/File:52_8134_Hoentrop_2012-09-16.jpg

Image of electric motor:
http://en.wikipedia.org/wiki/File:Motors01CJC.jpg

Machine Called Computer

- Computer
  - 동력원: 전기에너지, 효과: 계산, 논리적 처리
  - 자동장치 - 인간의 머리 (계산, 논리)를 대신함
    - 범용컴퓨터
    - All kinds of "smart" machines

Image of PC (범용컴퓨터):

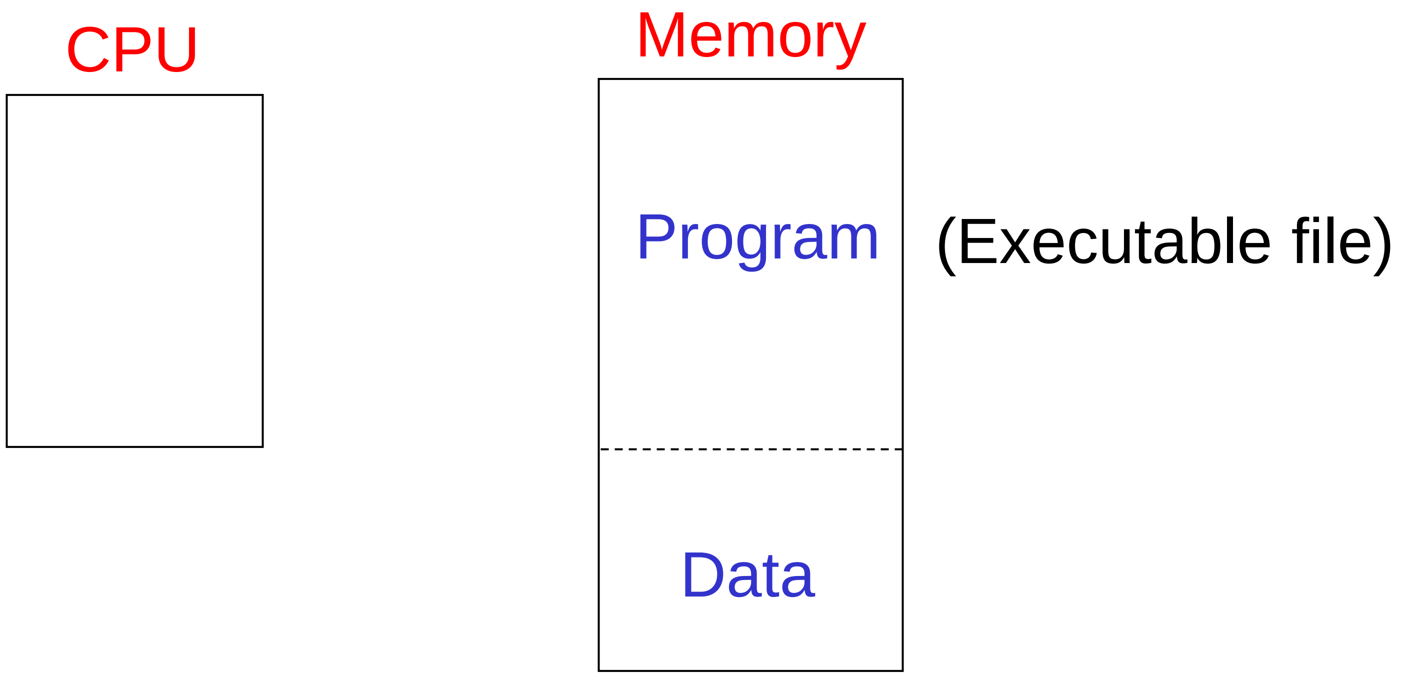
http://en.wikipedia.org/wiki/File:MSI_Laptop_computer.jpg

Image of robot (smart machine):

https://en.wikipedia.org/wiki/File:HONDA_ASIMO.jpg

Machine Called Computer

- What is computer? How does it work?



CPU

Memory

Program

(Executable file)

Data

I/O: Monitor/keyboard, LAN-Internet, …

Hardware - Inside PC

Image of Motherboard:

http://en.wikipedia.org/wiki/File:Acer_E360_Socket_939_motherboard_by_Foxconn.svg

Block diagram of a modern motherboard:

http://en.wikipedia.org/wiki/File:Motherboard_diagram.svg

ENIAC (1943-1946)

First fully-electronic, general-purpose computer

Image of ENIAC:

http://en.wikipedia.org/wiki/File:Classic_shot_of_the_ENIAC.jpg

Image of ENIAC:

http://en.wikipedia.org/wiki/File:Eniac.jpg

History of Computers

- Pascal's mechanical calculator – oldest in record (1642)
  - Add and subtract two numbers directly
    - Multiply and divide by repetition

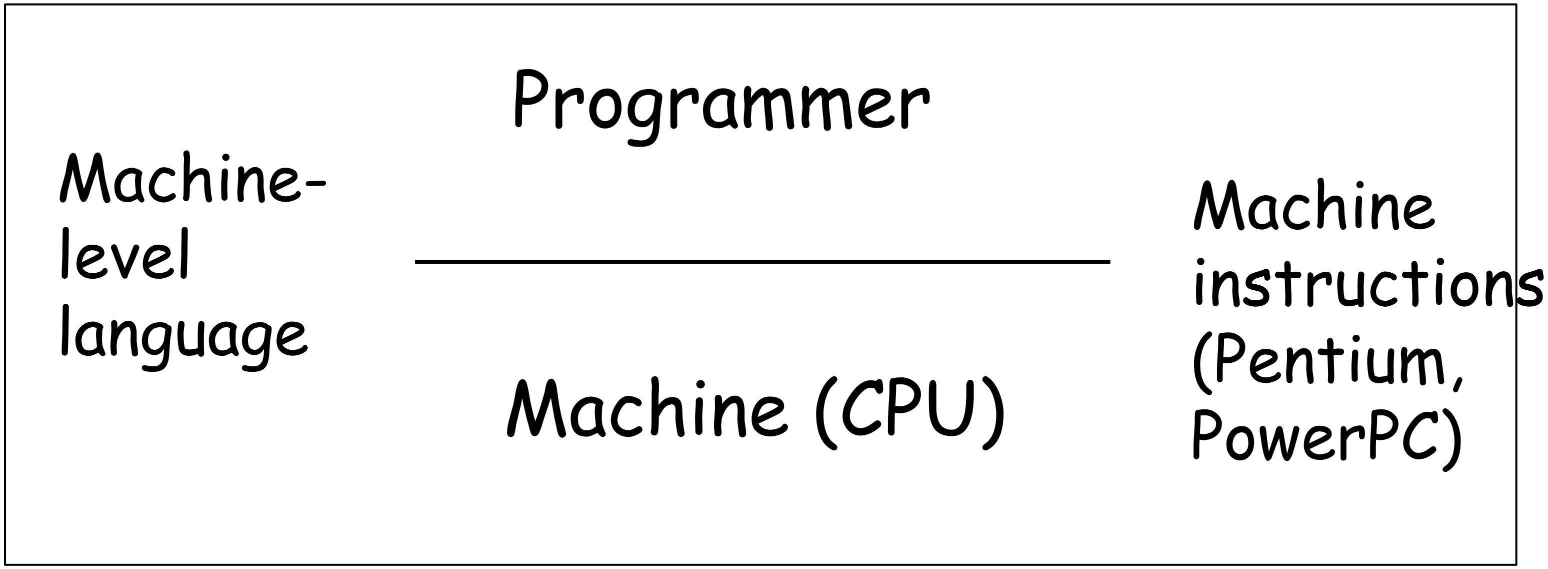
Image of Pascal's calculator: http://en.wikipedia.org/wiki/File:Arts_et_Metiers_Pascaline_dsc03869.jpg

History of Computers

- ❑ Pursuit of mechanical calculator since 17C
  - Add, subtract, multiply, divide
  - Used by engineers until 1970s
- ❑ 20C: more powerful, specialized, electric computers
  - 연립방정식 풀기, 복잡한 수학 함수 계산
  - 암호화 및 암호 해독 (2차대전)
  - 기업 업무용 계산 장치 (tabulating machines)
- ❑ ENIAC in 1945
  - First general-purpose electronic computer
    - Intended to be differential equation solver

Machine Called Computer

- ❑ Function determined by "program"
  - Sequence of machine instructions (HW/SW interface)



Machine-level language

Programmer

Machine (CPU)

Machine instructions (Pentium, PowerPC)

Machine Instruction Set

- ❑ Arithmetic and logic instructions (ALU)
  - add, sub, mult, div, and, or, not // ADD R1, R2, R3
- ❑ Data transfer instructions (for external memory, I/O)
  - load, store // LD R1, R31(#1)
- ❑ Jump instructions
  - jump if =, ≠, >, <, ≤, ≥

† With these, we have been computing for 70 years!

† With these, we can solve all problems we can imagine!

Program to Add Two Numbers

```
1000    LOAD  R1, (2000)   //  load from address 2000 to R1
1004    LOAD  R2, (2004)   //  load from address 2004 to R2
1008    ADD  R3, R1, R2    //  add
100C    STORE  R3, (2008)  //  store result to address 2008
1010    HALT
        …
2000    25                 //  first operand
2004    31                 //  second operand
2008    -                  //  sum of two operands
```

Machine-level programming

C program:

```
int  a, b, c;
a = 25;
b = 31;
c = a + b;
```

High-level programming

Machine Called Computer

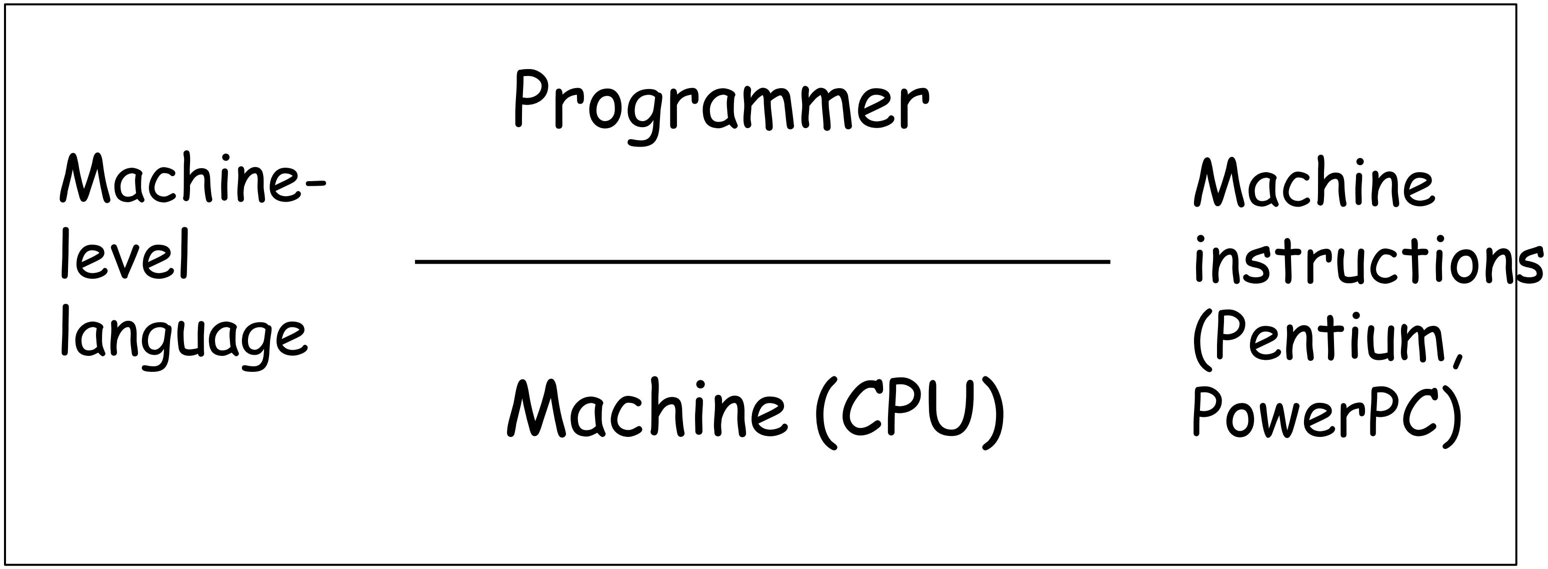
- ❑ Does it look intelligent?
  - Conceptually, a simple machine
- ❑ Real power of computers
  - Problem solving by programming (software development)
    - Automation and new useful tools
      - † The problem is solved forever!
  - Speed of light, no mistakes, never being tired
  - **인간의 생활 형태를 바꿈**
  - **기존 직업군의 소멸, 새로운 직업군의 탄생**
- ❑ 인간과 기계의 계산 및 저장 능력 비교

2) What is Computer Science?

(Computer Science and Engineering)

Computer Science

- Study of problem-solving with computational devices
- What kind of problem did we solve in 1945?
  - How to build computer (i.e., machine that compute)



Programmer

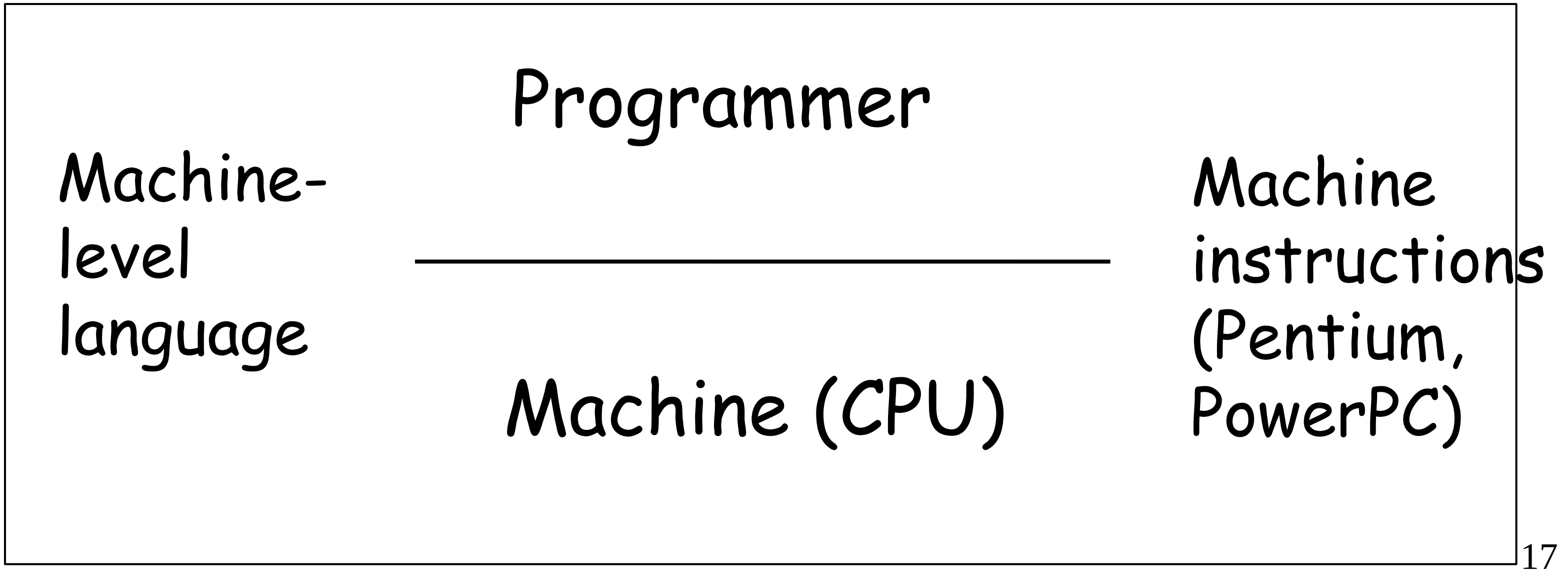
Machine-level language

Machine (CPU)

Machine instructions (Pentium, PowerPC)

Programming

- Telling computer what to do
- Machine provide low-level language
  - "The Hardware/Software Interface"
  - Productive?



Programmer

Machine-level language

Machine (CPU)

Machine instructions (Pentium, PowerPC)

High-Level Programming for Productivity

Developers (C program)

High-level language

사용법

C, C++, Java

Compiler 물건

(executable) Machine-level language

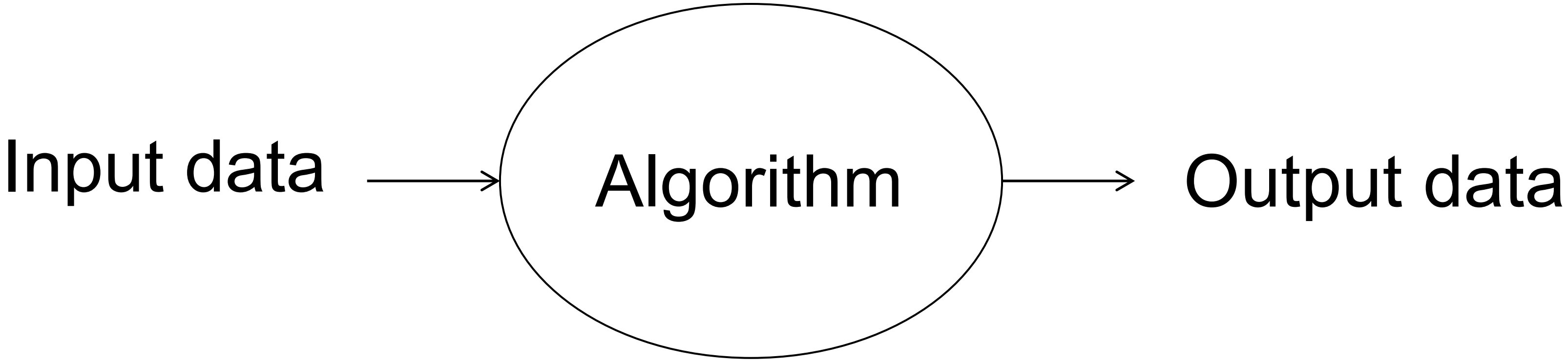
사용법

Machine instructions (Pentium, PowerPC)

Machine (CPU) 물건

Algorithm; Problem-Solving (참고)

- Well-defined procedure to solve particular problem
  - Input data, output data
  - Correctness, efficiency



Input data

Algorithm

Output data

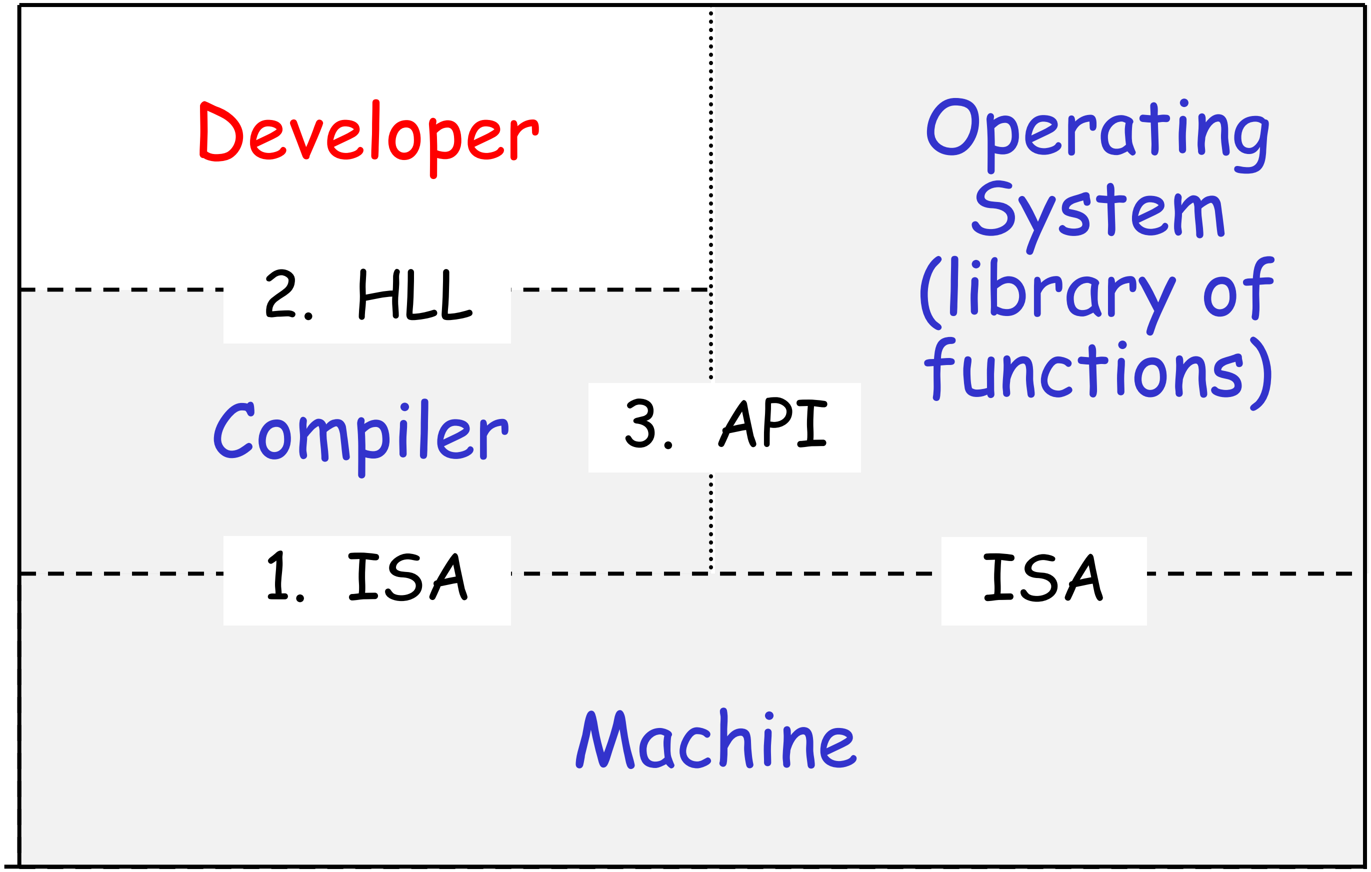
- Organization and processing of information

What is CSE?

- Study of problem-solving with computational devices
- What kind of problems did we solve?
  - How can we build a machine that computes?
    - How to kill: solve differential equations
  - How can we boost productivity in programming?
    - High-level programming languages
  - How can we make the machine easier to use?
    - OS (운영체제; collection of many algorithms)

Three Major Interfaces

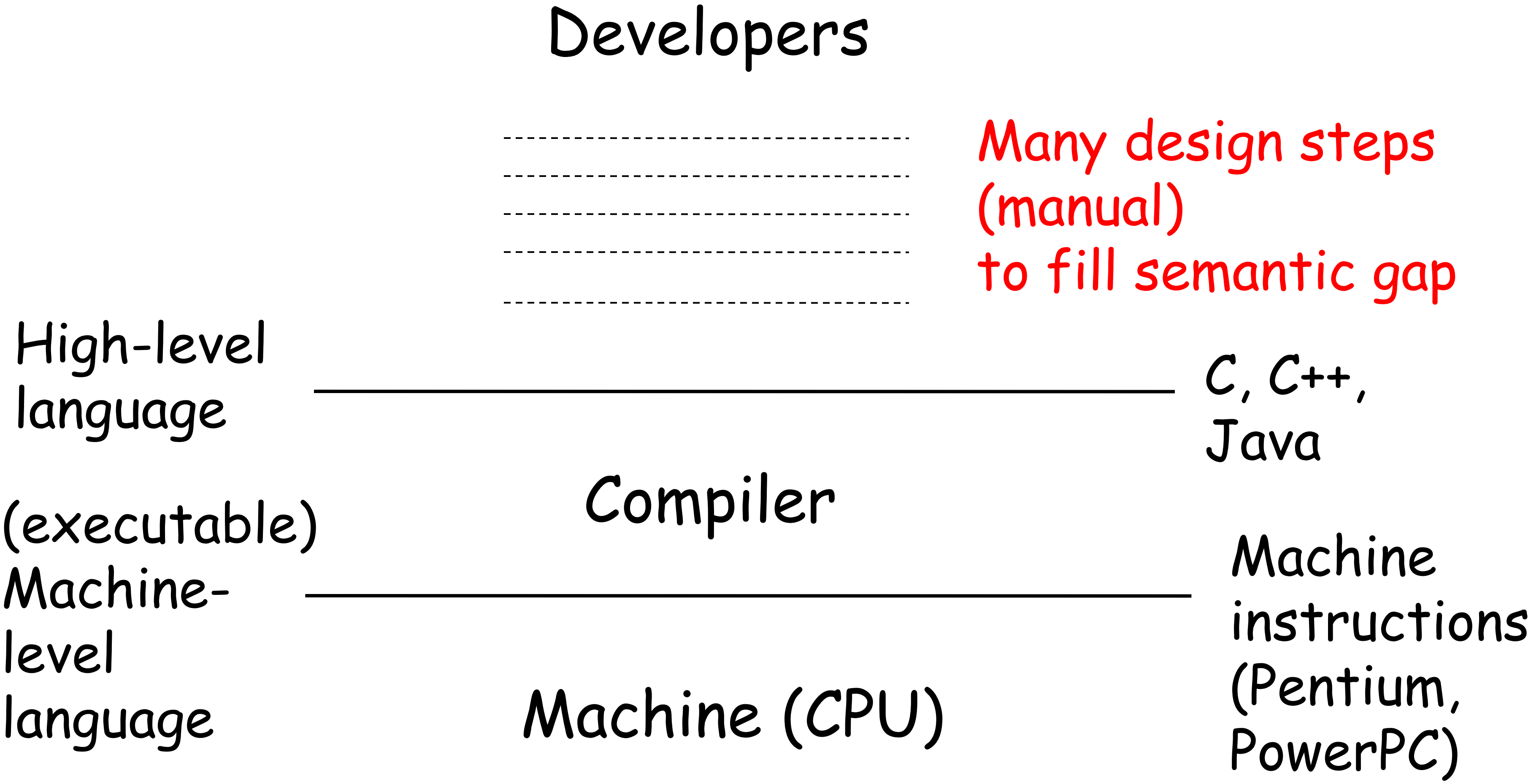
- ❑ Three key products and their services
- ❑ Three "core" CSE subjects (computer system; 전공핵심)



What is CSE?

- ❑ Given Pentium/C, what kinds of problems did we solve?
  - How to send Apollo to moon (과학계산)
  - How to manage the information on things (database)
  - How to connect all computers in the world (Internet)
    - Given Internet, how to share information (web)
  - Given the web, how to find what I want (search engine)
  - Given web, how to sell my products (e-commerce)
  - How to make documentation/publishing easier (Word)
  - Big data challenge
    - Buying/accessing record, SNS data, bioinformatics

Million Lines of Source Code (참고)



Developers

Many design steps
(manual)
to fill semantic gap

High-level
language

C, C++,
Java

Compiler

(executable)
Machine-
level
language

Machine
instructions
(Pentium,
PowerPC)

Machine (CPU)

Computer Science and Engineering

- Science vs. engineering
  - Science pursue a major new piece of knowledge
  - Engineering is about tools
    - Accumulation of knowledge facilitate engineering
- Science nature
  - Recognition of problems, establish mathematical approaches
- Engineering nature
  - Software tool development
    - Smaller-scale problem-solving by many engineers

† Problems: Nobel-prize scale vs. every day programming

Fundamental Paradigm

Problem recognition
(what to solve, creativity)

CS

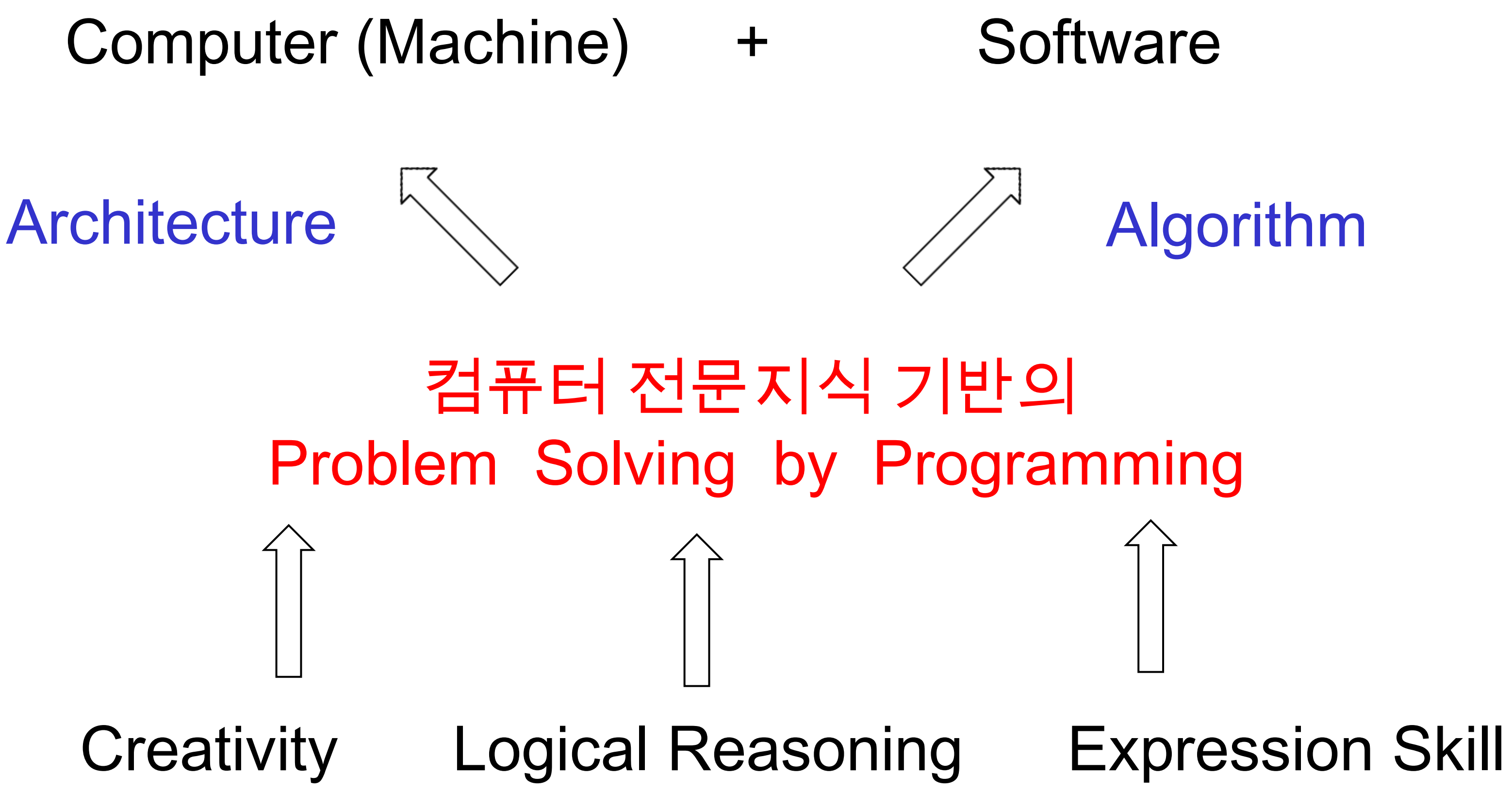
Solution method
(how to solve, logical thinking, efficiency)

Software tool development
(programming skills)

programming

- ❑ CS is not about programming
  - 프로그래밍은 problem solving을 위한 수단

CSE: Big Picture



Computer (Machine) + Software

Architecture

Algorithm

컴퓨터 전문지식 기반의

Problem Solving by Programming

Creativity

Logical Reasoning

Expression Skill

- ❑ Think like a computer scientist!
- ❑ Interplay between architecture and algorithm

컴퓨터공학부 전공 교육

- ❑ 1/2 학년
  - Programming languages
  - 기초 공통 알고리즘 (흔히 나오는 problems 및 solutions)
- ❑ 3/4 학년
  - 전공별 알고리즘 (problems 및 solutions)
    - OS, database, network, compiler, AI
- ❑ 실전 (사회 진출)
  - 위의 지식을 이용한 실전 problem solving by programming
- ❑ 대학원 (academic)
  - 새로운 problem formulation 및 새로운 solution methods
  - 연구 트랙 또는 세분화된 영역 전문가

CSE Success Guide

- ❑ 승부의 분기점 (현실은?)
  - 학부 1/2 학년
    - Problem solving by programming 훈련
    - 프로그래밍 언어, 이론과 자료 구조 익힘
- ❑ 성공의 핵심: 실전형 개인 프로젝트
  - Self-defined problem, solution, programming
- ❑ 실전 프로젝트의 중요성
  - Think about 10-year old young musician
- ❑ Attitude
  - 내가 내 책상에서 소프트웨어를 만들 줄 알면, 아무도 나를 막을 수 없다

To remember:

I am a computer scientist!

I am a problem solver!

Job carnivalization!

3) What is Computer Architecture?

(This Class)

Topics of This Class

- ❑ First class
  - Overview of computers, CS, computer architecture
- ❑ Issue 1: machine called computer (fundamental concepts)
  - Part 1: 어떻게 컴퓨터라는 복잡한 기계 만들 수 있었나,
    digital logic design의 의미
  - Part 2: "abstraction" to deal with complexity
  - Part 3: data (vs. code)
  - Part 4: 컴퓨터라는 기계의 동작 원리 (fetch-execute),
    기계가 제공하는 서비스 (ISA, HW/SW interface)
  - (skip) Part 5: ENIAC 에 이르기까지의 300년의 여정
  - Part 6: ENIAC 이후의 IT Gold Rush 및 evolution

Topics of This Class

- ❑ Issue 2: 컴퓨터라는 기계 (processor)의 external I/F 설계
  - What is a good ISA?
  - Today's RISC-style ISA (MIPS)
  - Hardware-software interaction
    - How do programs run on machine?
- ❑ Issue 3: 설계한 ISA의 효율적인 구현
  - Given an ISA, what is a good implementation?
    - Data path, control
    - Pipelining, cache memory
- ❑ Short introduction to parallel processors

Topics and Textbook (홈페이지 참조)

- Issue 1: Fundamental concepts and principles
- Issue 2: ISA (HW-SW interface) design
  - Ch. 1: computer performance
  - Ch. 2: language of computer; ISA
    - What is a good ISA? Today’s RISC-style ISA (MIPS)
    - How do programs run on computer?
  - Ch. 3: data representation and ALU
- Issue 3: implementation of ISA (internal design)
  - Ch. 4: processor
  - Ch. 5: memory system
- Short introduction to parallel processors

Why Architecture Class?

- Essential knowledge for all CSE majors
  - What is the machine called computer?
    - Principles, structure and operation
    - Service: ISA (hardware and software interface)
  - How programs run on computer
    - HW-SW interactions
- Architecture perspective
  - Quest for faster machine
- Software perspective
  - Effective use of machine (performance programming)

Architect (HW) Perspective (부연설명)

- ❑ How can I make the machine faster?
  - Design issues and implementation techniques
- ❑ I am interested in embedded systems or SoC (system-on-chip)
- ❑ Interplay between machine and software
  - Software simulation of new architectural ideas
    - Performance simulation of machine
  - Programs are customers
  - VLSI chip design by programming

Programmer Perspective (부연 설명)

- ❑ How best can I utilize the machine? (Programming for performance)
  - Processor in your PC
    - Single core or multicore?
    - What are features that programmers must know?
  - Unresolved challenge under parallel revolution
    - Parallel programming
- ❑ Interested in core library, system software, embedded systems?

Intel Desktop Products (참고)

- Multicore processor era

	Core i7	Core i5	Core i3	Pentium	Celeron
# cores/threads					
Hyper-Threading					
Turbo Boost					
AVX					
CPU overclocking					

Programming Challenge (참고)

- ❑ Industry-grade software
  - Large and complex
    - Collection of many algorithms
  - Correct
  - Reliable
  - Fast
  - Elegant
  - Energy efficient
  - Solve an important problem
  - Provide interface (GUI, API)

Architecture Line of Classes

- ❑ Digital logic design
  - Given: AND, OR, NOT
  - Design: decoder, register, memory, …, ALU, processor
  - VHDL/Verilog design environments
- ❑ Microprocessors
  - Read databook of real CPU, do assembly programming
  - Hands-on experience of machine
- ❑ Computer architecture (this class)
  - Core CSE subject
  - Undergraduate or first-year graduate level

Architecture Classes (부연 설명)

- ❑ 컴퓨터라는 기계는 원리적으로 어떻게 만드나?
  - Digital logic design
    - Hardware: CPU, memory, I/O
- ❑ 컴퓨터라는 기계를 사용해 봄으로써 무엇인지 이해한다
  - 마이크로프로세서응용 또는 어셈블리프로그래밍
- ❑ 컴퓨터구조
  - 사용법 (ISA) 설계 및 내부 설계
  - Design for performance

Architecture Classes (소프트웨어 전공)

- ❑ 소프트웨어 전공
  - 우리는 소프트웨어: 세 과목은 많다, 두 과목으로
  - 첫 번째 과목
    - 컴퓨터구성및어셈블리프로그래밍 채택
      - Digital Logic Design + assembly programming
  - 두 번째 과목
    - 컴퓨터구조론 (this class)

Graduate-Level Architecture (참고)

- ❑ Advanced architecture courses
  - Computer architecture: A Quantitative Approach, by Hennessy and Patterson
  - Parallel architectures and concurrent programming
- ❑ Read papers in top 5 conferences on architecture
  - e.g., Int. Symposium on Computer Architecture (ISCA)
- ❑ 새로운 powerful machine 설계 방법
  - Require knowledge on digital electronics circuits, VLSI systems design

Must understand the interplay:

- Quest for powerful machine
- Smart use of machine

4) This Class (행정 사항)

Administration

- ❑ Textbook
  - Computer Organization and Design – The Hardware and Software Interface, 5th Ed., Hennessy & Patterson, Morgan-Kaufmann
    - English version only
- ❑ Undergraduate or first-year graduate level class
- ❑ Prerequisite
  - Digital logic design, C programming

Administration

- ❑ Class homepage updated weekly
  - 수업자료, 과제물, 시험 공지
    - † 위치와 암호
- ❑ 시험
  - 중간/기말 또는 3회로 나누어: 수업시간 이용
- ❑ Homework (submit electronically to “HY-in”)
  - 전반부: small weekly homework
  - 후반부: RISC processor design 등의 실습 과제
- ❑ Tentative grading plan
  - 시험: 90%, 과제물: 10%

Any Question?