Computer Architecture

Overview of Class

순서

1) What is a computer?

2) What is Computer Science?

3) What is Computer Architecture?

4) 행정 사항

도구 (Tools)

- 인간은 도구를 잘 만든다 (farming, hunting)
  - 동력원: 인간의 에너지
  - 도구는 힘이 효과적으로 사용되도록 함

기계 (Machines)

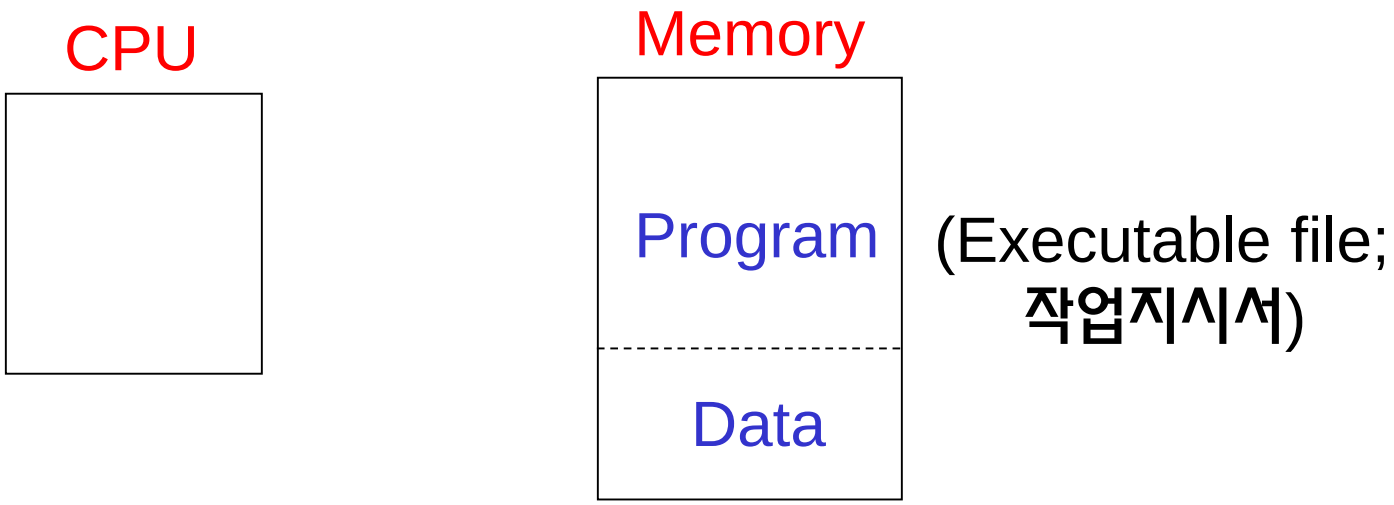
- Steam engines, 산업혁명 (17C 영국)
  - 기계 - 인간의 힘을 대신함; 동력원: 화학에너지
- 전기 현상 및 전기 기계 (19/20C; 2차 산업혁명)

Machines Called Computers

- 인간의 머리를 대신함 (20C 미국; 3차 산업혁명)
  - 동력원: 전기에너지, 효과: 계산, 논리적 처리
- 범용컴퓨터 vs. 임베디드시스템 ("smart" machines)

Machines Called Computers

- What is a computer? How does it work? (HW vs. SW)

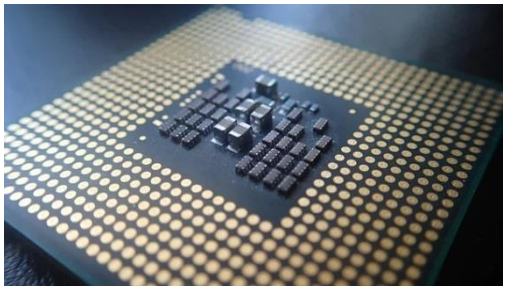


I/O: Monitor/keyboard, LAN-Internet, …

- Three fundamental abstractions
  - Computation (계산), memory (저장), communication (접속)

Hardware - Inside PC

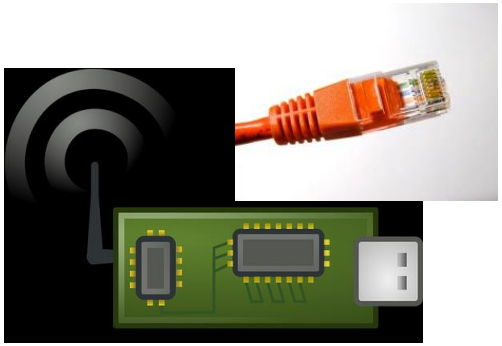
CPU



I/O

Motherboard

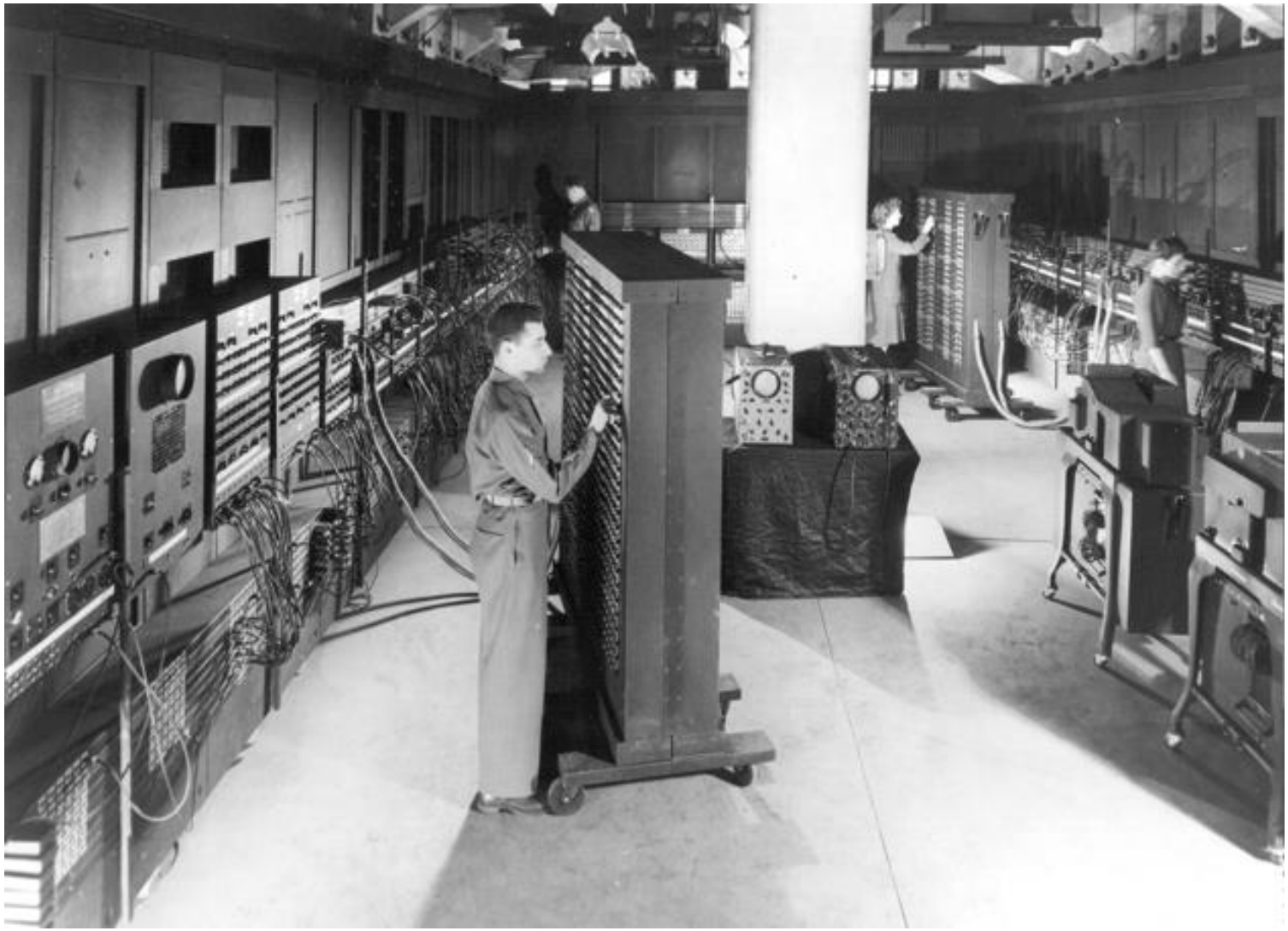
Memory



LAN-Internet

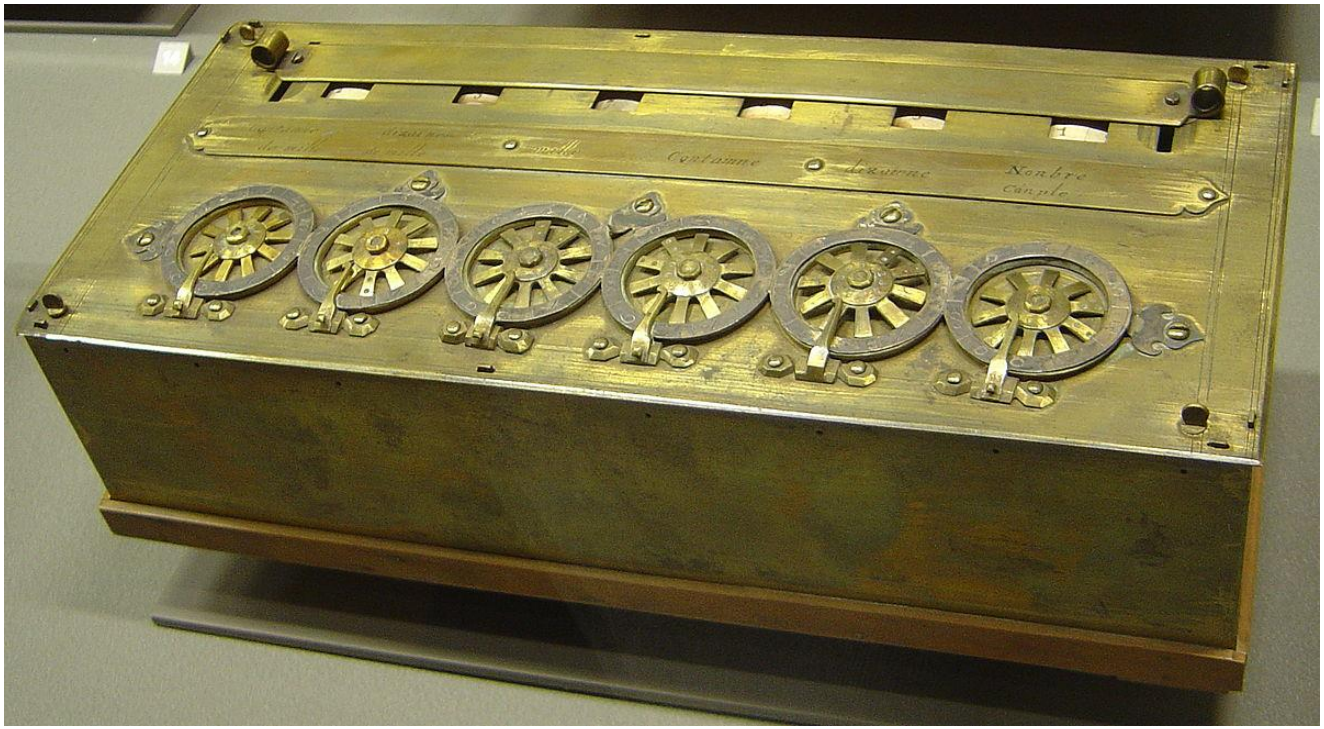
ENIAC (1943-1946)

First fully-electronic, "general-purpose" computer



History of Computers

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



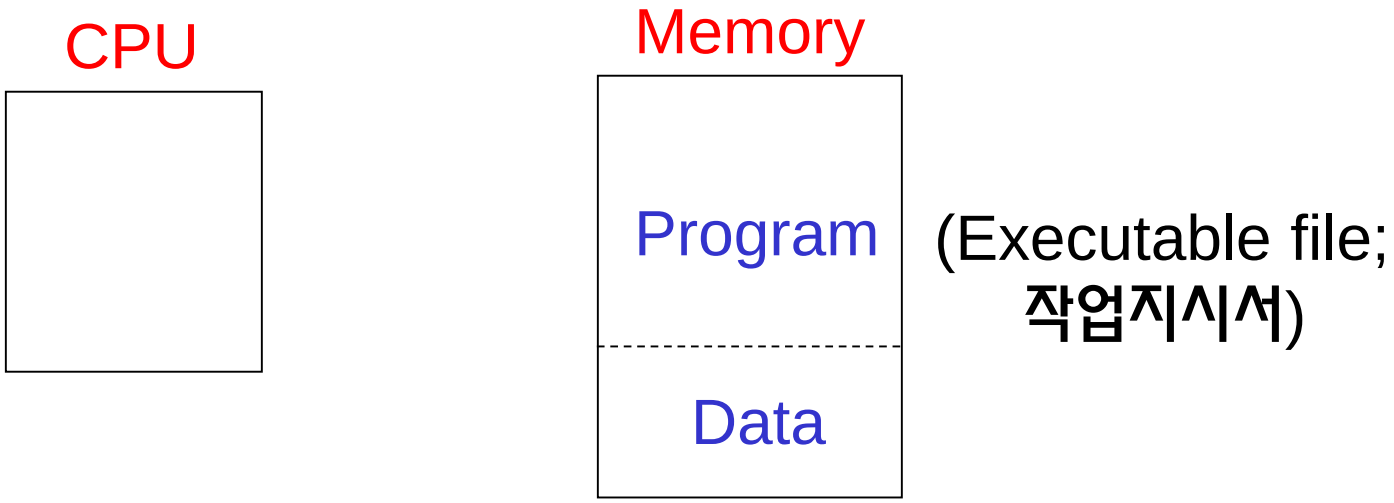
https://en.wikipedia.org/wiki/File:Arts_et_Metiers_Pascaline_dsc03869.jpg
Photographed by David Monniaux; no changes were made

History of Computers

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

Machines Called Computers (반복)

- What is a computer? How does it work? (HW vs. SW)

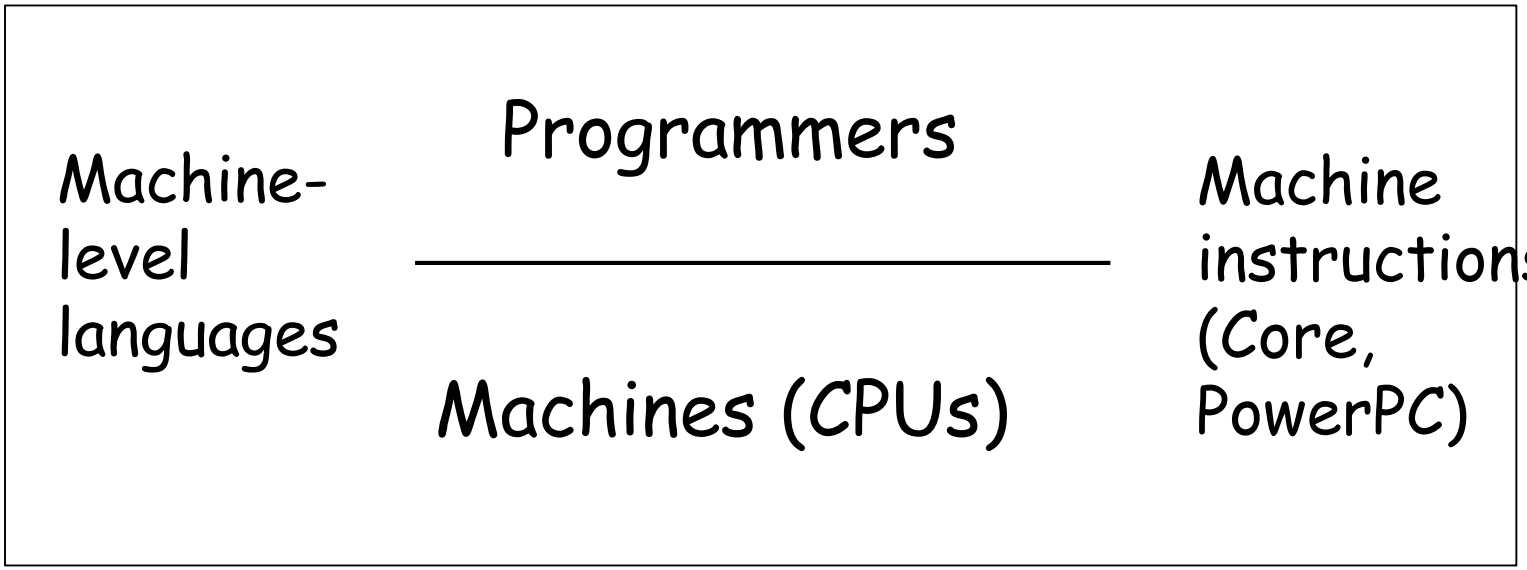


I/O: Monitor/keyboard, LAN-Internet, …

- Three fundamental abstractions
  - Computation (계산), memory (저장), communication (접속)

Machines Called Computers

- Functions determined by "programs"
  - Sequence of machine instructions



Machine Instruction Set (가볍게)

- 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 computational problems!

High-Level Programming

```
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

compile

C program:

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

High-level programming

Machines Called Computers

- 지난 70년간 우리의 일상을 바꾸어 옴
- What makes computers so powerful?
  - Problem solving by programming
    - Software development and automation
      - † The problem is solved forever!
    - Speed of light, no mistakes, never being tired
  - New life-changing tools (e.g., Internet)
  - 인간의 생활 형태를 바꿈
    - 기존 직업군의 소멸, 새로운 직업군의 탄생

2) What is Computer Science?

(Problem Solving by Programming)

Fundamental Paradigm

Problem recognition
(what to solve)

Solution methods
(how to solve, 알고리즘/수학)

Software tool development
(programming skills)

목표:
problem-solving
(창의성,전문지식)

수단:
programming
(구현 기술)

- Problem solving and programming: tightly coupled

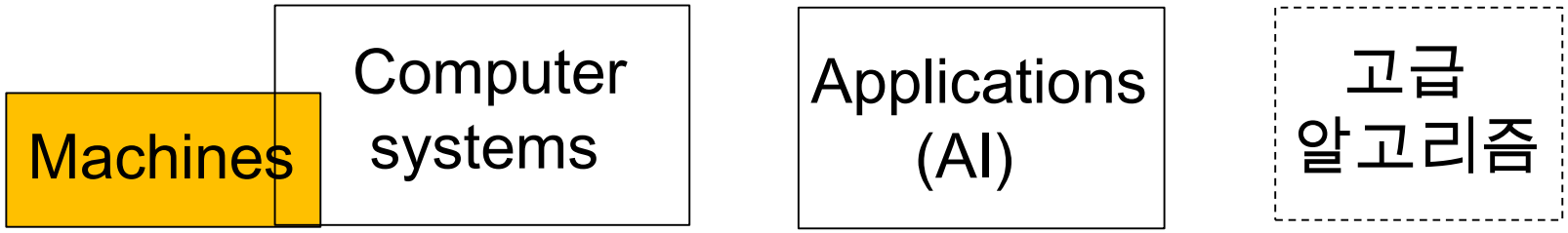
CS 전공 교육

- CS 입문 (1/2 학년): 프로그래밍
  - 프로그래밍 개념/언어, 생각의 표현, 알고리즘, 수학적 사고/지식
- CS 3/4 학년: 전공 problems 및 solution methods
  - 컴퓨터시스템 (어떤 응용에서도 필요한 infra 기술)
    - Architecture, OS, DB, network, security, …
  - 다양한 응용 시스템
    - 인간의 모든 활동 분야, AI, machine learning, …
- 창조

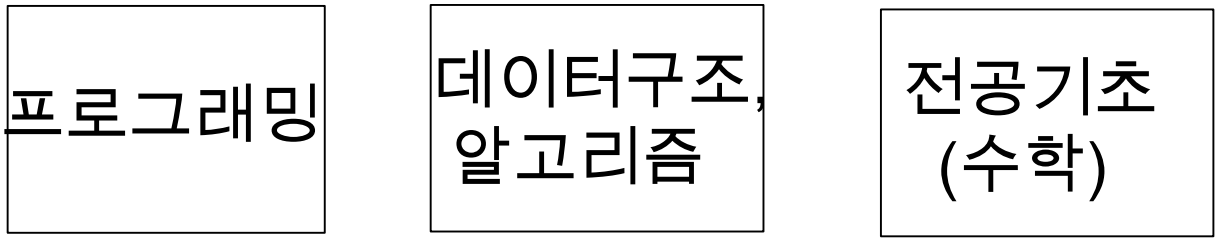
CS 전공 교육

창조 (ideas)

전공: problem-solving based on CS theory and mathematics



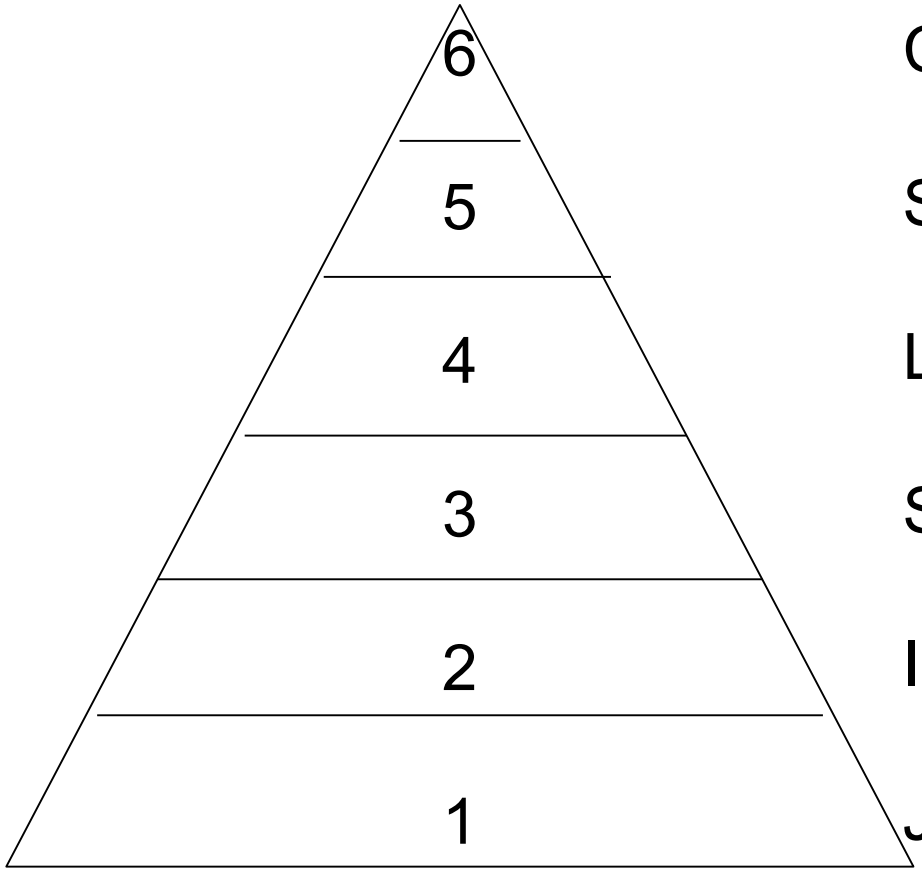
CS 입문: programming skills and mathematics



CS 전공 교육

- CS 1/2 학년: programming skill
  - 쉬운 문제에서 시작하여 난이도/복잡성 높여감
  - SW projects 중요, 전공 동아리 활동 중요 (비전공 프로그래머)
- CS 3/4 학년: problem-solving
  - OS 사례: OS 개발 과정에서 무슨 문제에 부딪쳤고 이를 어떻게 해결하였나 (algorithms and design patterns)
    - 대학원 수준: 더 좋은 해결 방법은? 새로 해결할 문제는?
  - 전공은 왜 배우나
    - 대부분의 고급 problem-solving 에서 이들 기술 필요

Software Developer Levels



Consulting

Senior staff (fellow, advisory)

Lead (principal)

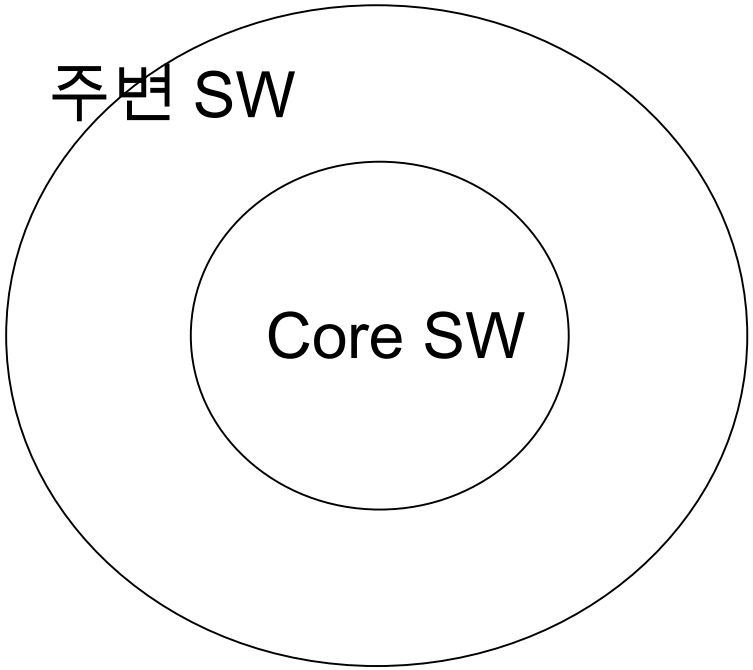
Senior

Intermediate

Junior

- New hires are generally Level 1 or 2, moving quickly to 3
  - Stay there for a while, most people not make it past 4

성취도 (전공지식 및 숙련도) 따른 대우



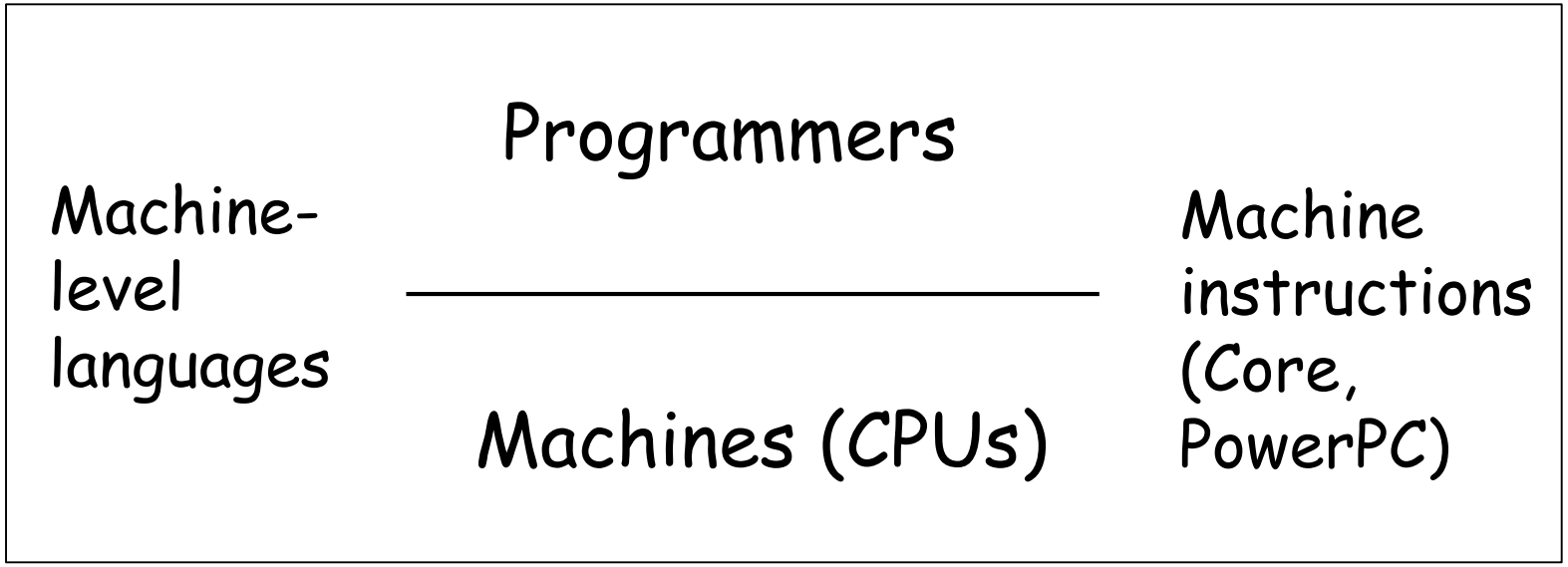
- 창의성, 전문지식, 난이도, 숙련도

한양 CS Vision

- 한양 CS 동문의 problem-solving
  - 세계의 SW 판도를 흔든다 - 소프트웨어 창업의 메카
- 재학생
  - 나는 무슨 문제를 풀고 있나? (경시대회/창업 석권)
    - Self driven; 학부, 교수진은 계기와 환경 제공
- 성공의 핵심: 개인/팀 소프트웨어 프로젝트
  - 스스로 무슨 문제를 어떻게 풀 것인지 고민하고
  - 그 결과를 이용하여 SW 구현 기술 연마

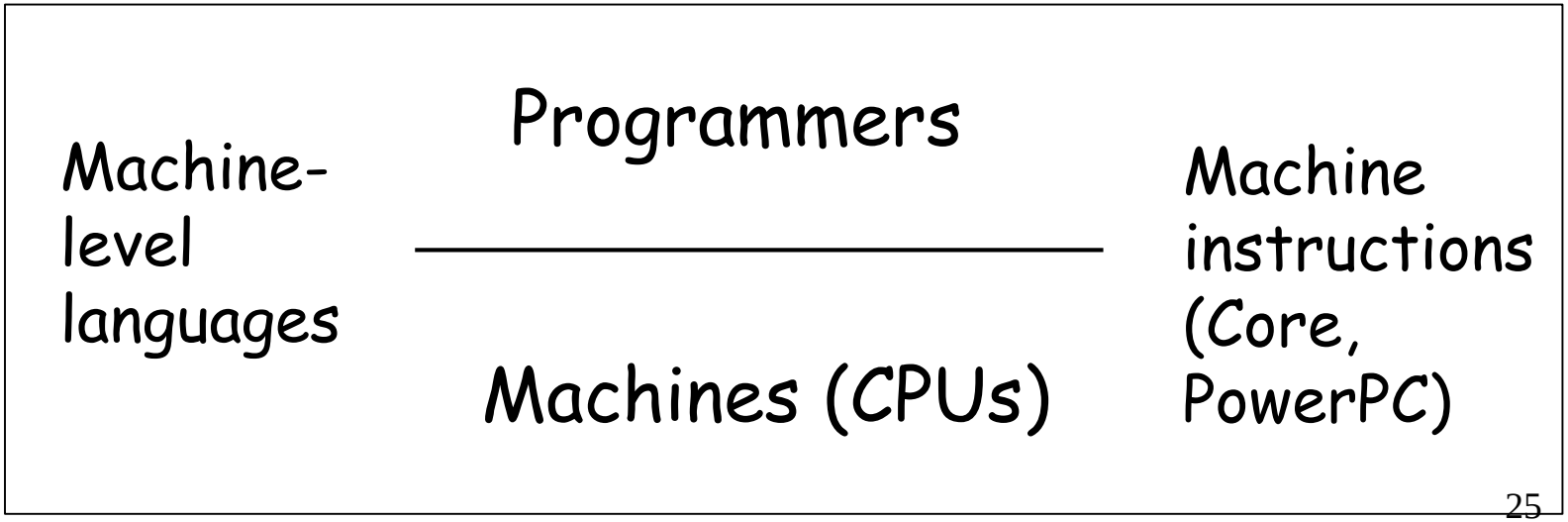
Computer Science and Engineering

- Study of problem-solving with computational devices
- What kinds of problems did we solve in 1945?
  - How to build computers (i.e., machines that compute)

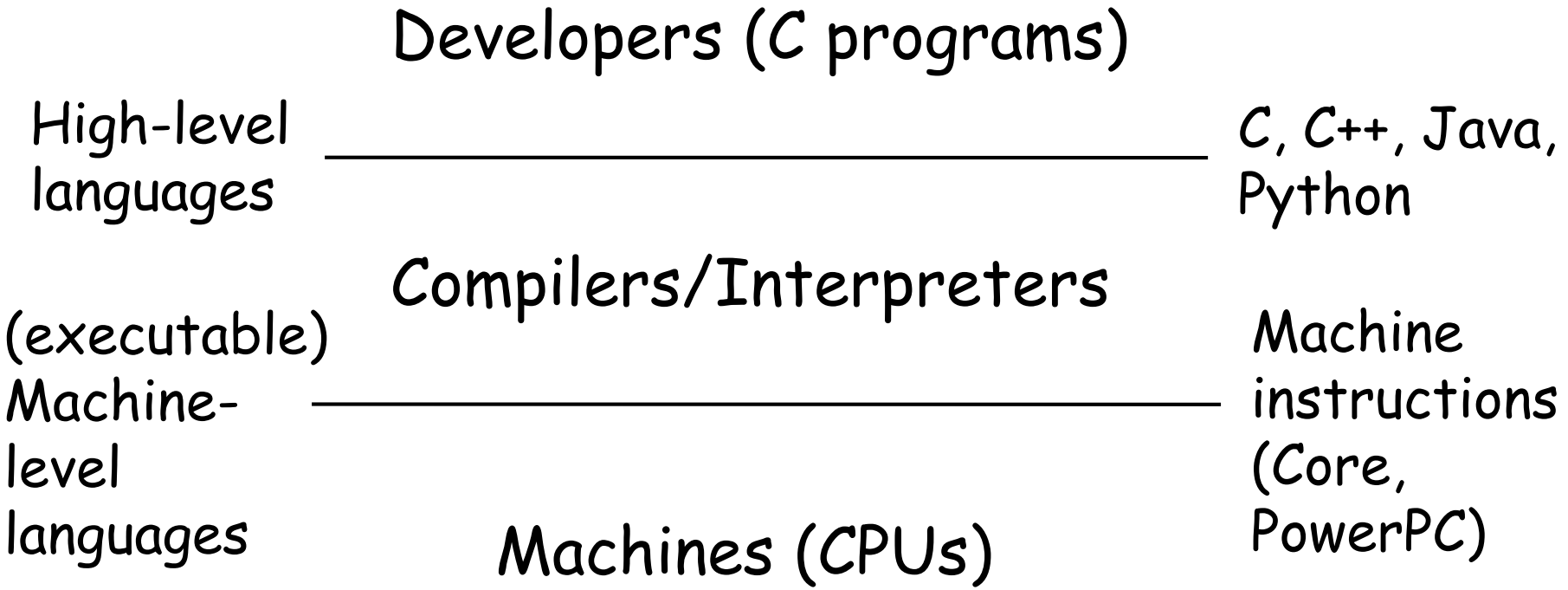


Programming

- Telling computers what to do
- Machines provide low-level languages
  - "The Hardware/Software Interface"
  - Productive?



High-Level Programming for Productivity

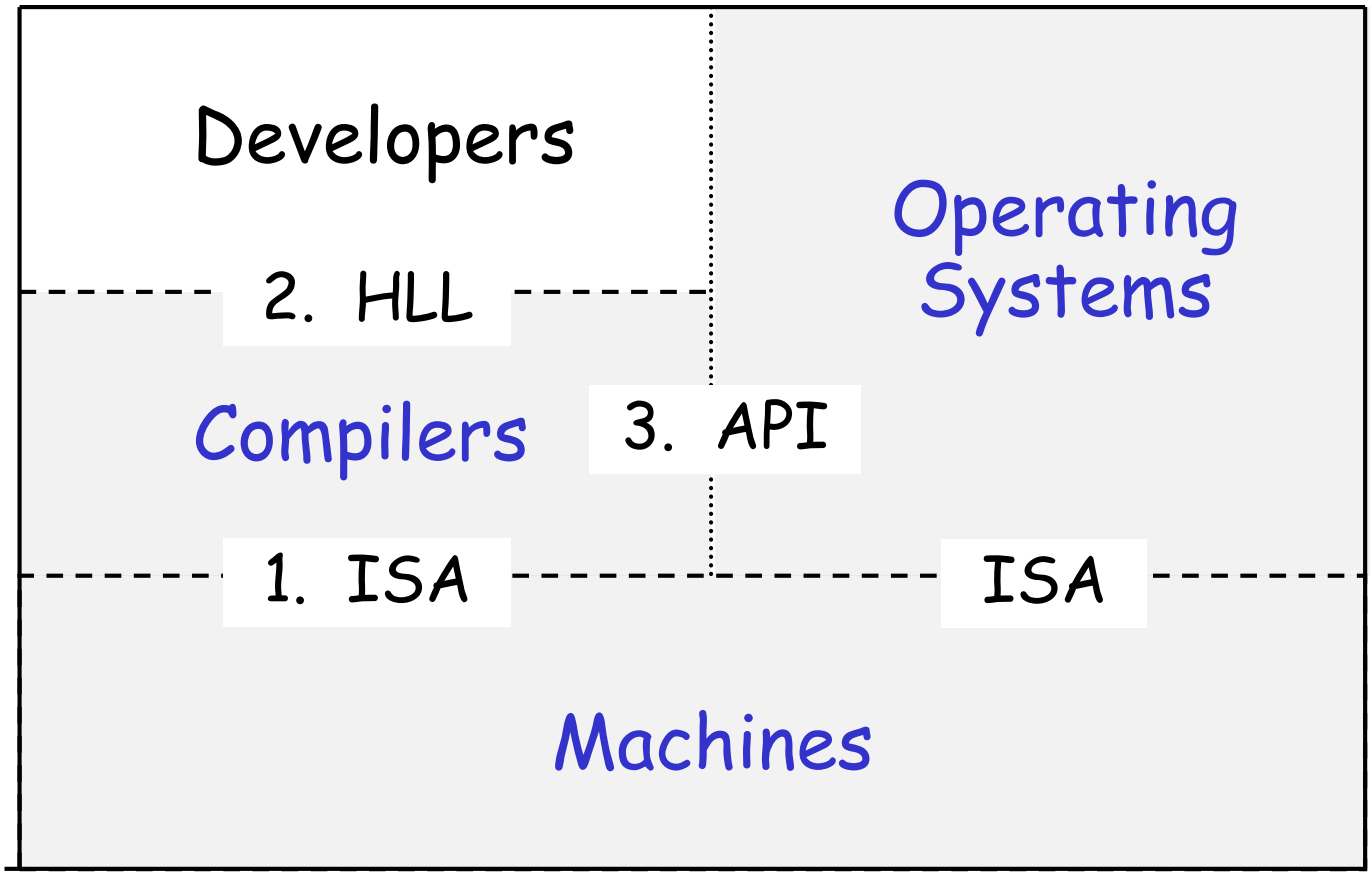


What is CSE?

- Study of problem-solving with computational devices
- What kinds of problems did we solve?
  - 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 (가볍게)

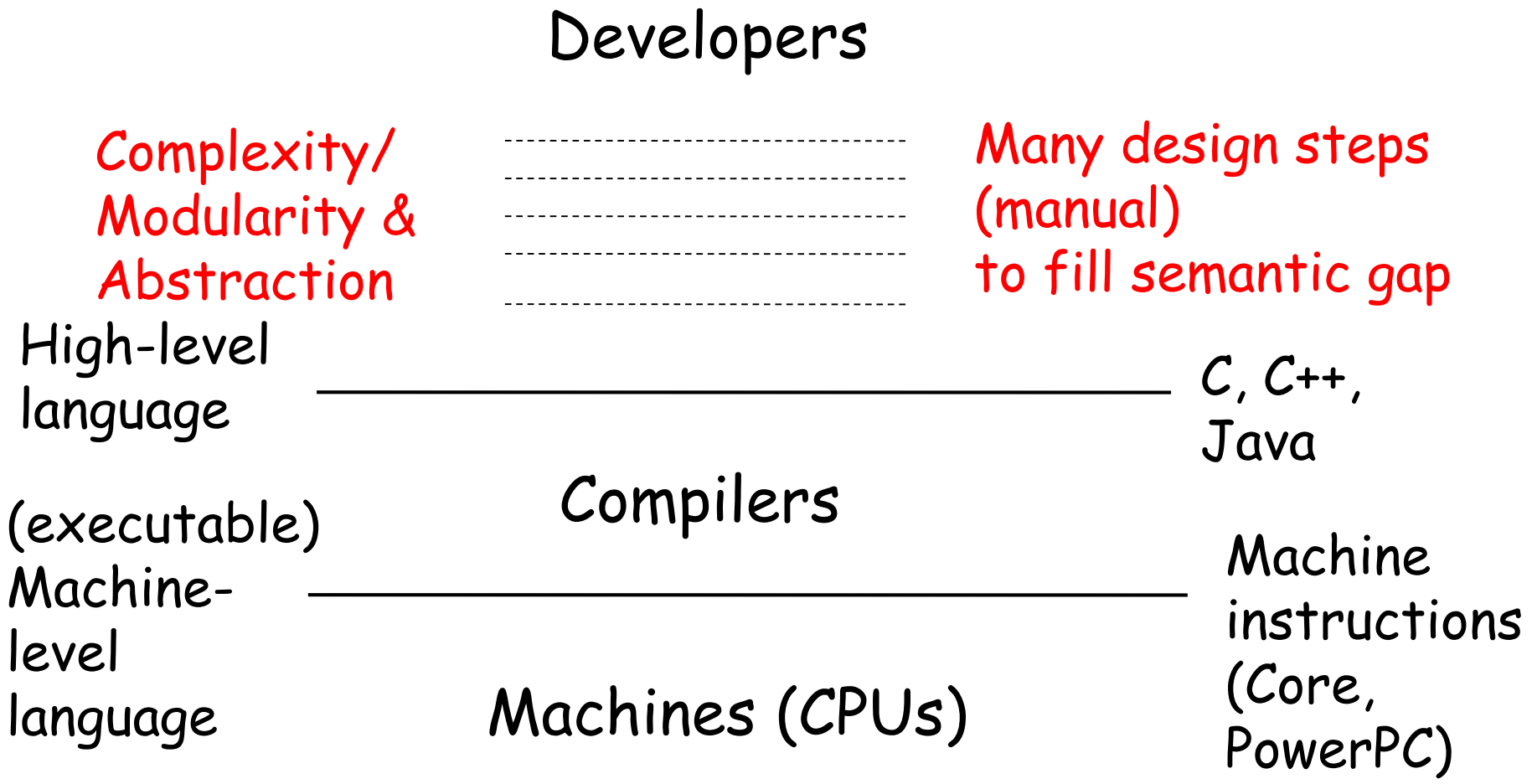
- Three key products and their services
- Three "core" CSE subjects (computer systems; 전공핵심)



What is CSE?

- Given machines/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
    - Deep learning, SNS data, bioinformatics

Million Lines of Source Code



- Most intellectually-challenging tasks

Why the name Computer Science?

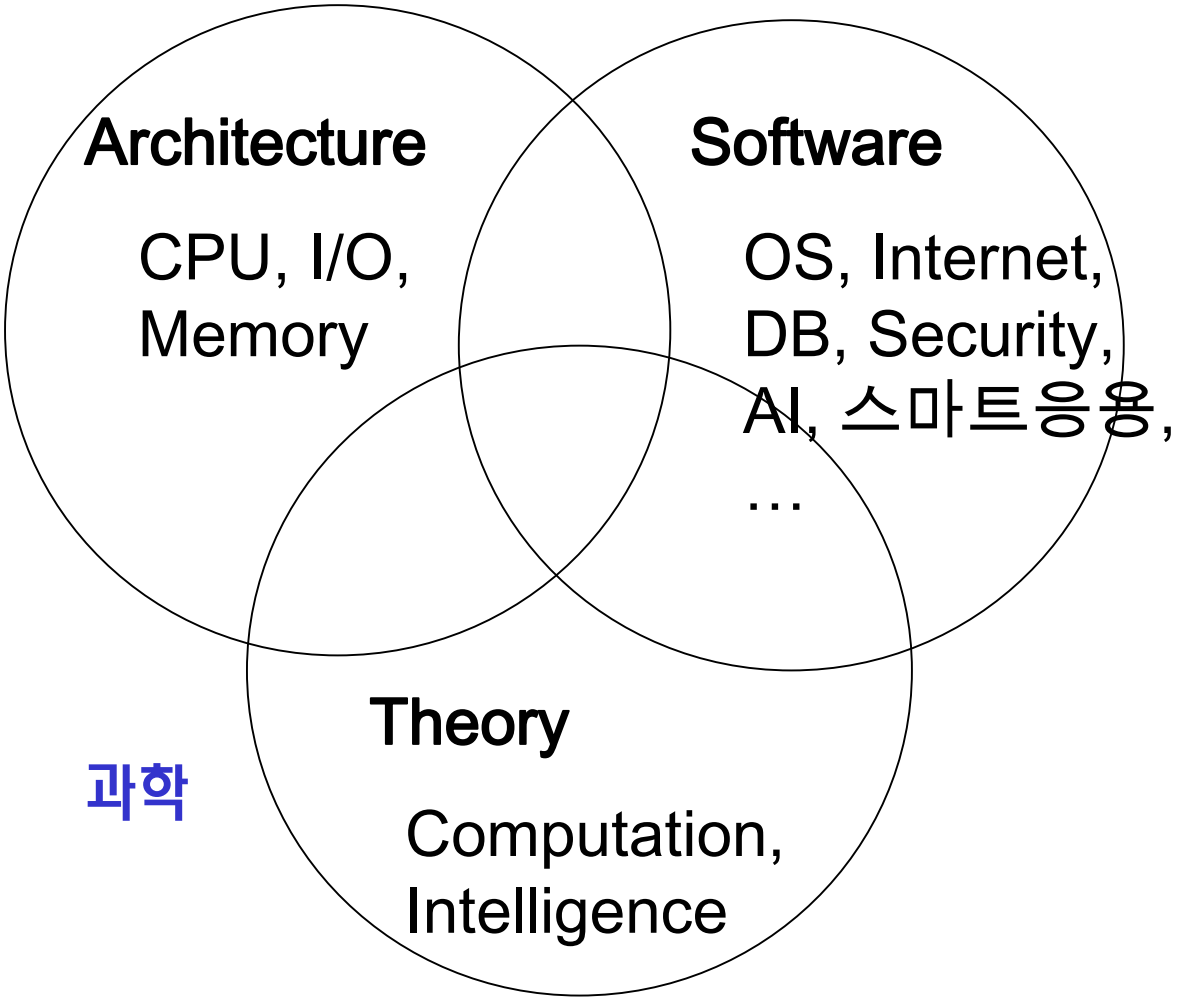
- Strong science flavor (과학, 새로운 지식을 추구)
  - Theory of computation (since early 20C)
  - Artificial intelligence (with invention of ENIAC)

- Application of computers

  (engineering flavor; 공학, 새로운/유용한 도구)
  - Smarter software tools
  - Powerful machines

Computer Science and Engineering

IT (Information Technology; 공학)



과학

- ICT (Information and Communication Technology)

Engineering (IT) Big Picture

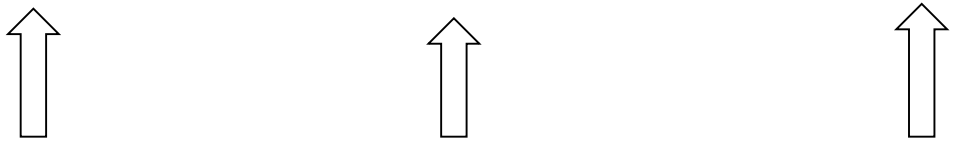
보다 빠른 machines + 보다 스마트한 software

Architecture (computers)

Algorithms (computation)

컴퓨터 전문지식 기반의

Problem Solving by Programming



Creativity, logical reasoning

Algorithms, Math.

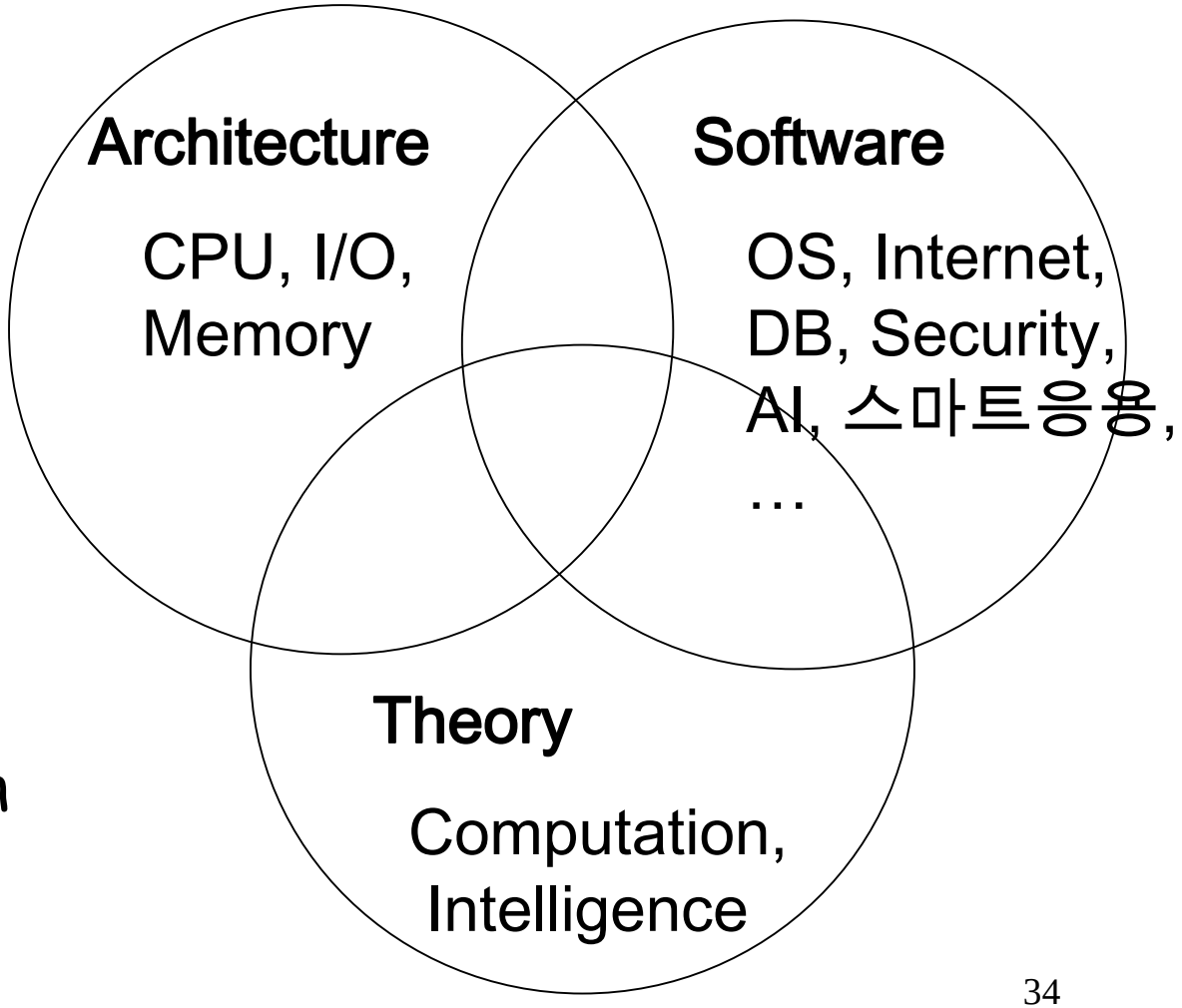
Programming skill

- Interplay between architecture and algorithms

Software Convergence

- CSE

- SW 융합
  - Management
  - Finance
  - Law
  - Automotive
  - Education
  - Transportation
  - Silver, …



소프트웨어 융합

- CS used to solve "infrastructure" problems
  - Infra software: OS, 인터넷, 웹, DB, 보안, AI, …
- Realization in 21C
  - 소프트웨어를 이용한 자동화가 어떤 영역에도 적용 가능
  - 모든 산업과 우리의 일상을 바꾸어 놓음
    - "Software is eating the world"

  † 4차 산업 혁명 (엄밀한 정의는?)
- 누가 주도할 것인가 - 문제 해결 능력을 갖춘 SW 인재

CS 교육 목표

- 일류 소프트웨어 개발자 양성?
- 못지 않게 중요한 것은: "큰 교육"
  - 유익한 사회 구성원 (말/글/행동)
  - 전공 공부를 통한 전문인으로서의 자신감과 도전정신
- CS 전공자의 진로
  - IT 정통한 사업가, 투자자, 교육자, 관리자, 경영자
  - IT 정통한 산업 디자이너, 법률가, 관료, …

To remember:

I am a computer scientist!

I am a problem solver!

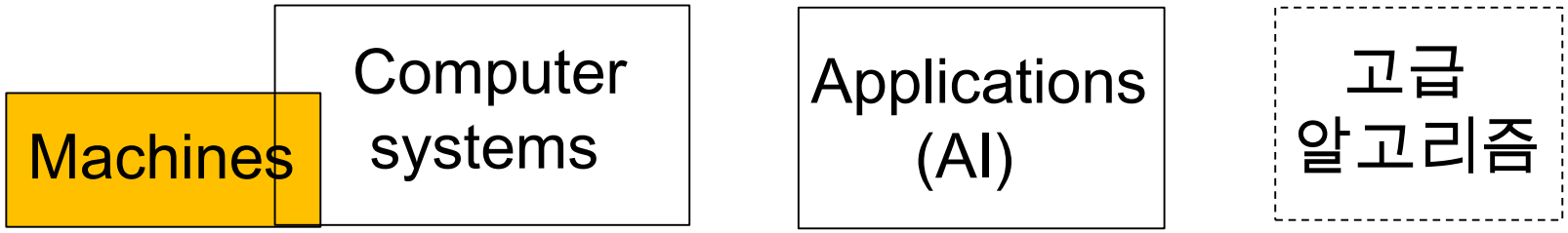
3) What is Computer Architecture?

This class is about "(fast) machines"

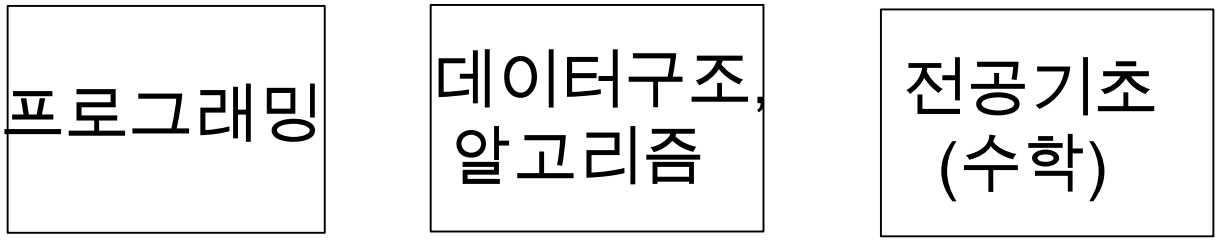
CS 전공 교육 (반복)

창조 (ideas)

전공: problem-solving based on CS theory and mathematics



CS 입문: programming skills and mathematics



수업 내용 및 목표

- 컴퓨터라는 기계는?
  - 개념, 구조, 동작, 원리
- 빠른 컴퓨터를 만들려면?
  - 설계
  - 구현

3단계 수업 세부 목표

- 컴퓨터, 컴퓨터구조, 컴퓨터 사이언스
  - 기본적이고 핵심적인 개념과 원리
- 컴퓨터의 성능평가 방법, 이에 따라 설계된 RISC (MIPS) 인스트럭션 셋
- 인스트럭션 셋의 구현 (또는 프로세서 설계)
  - 성능을 높이기 위한 파이프라인 및 캐시 메모리 설계방법

† Many SW-HW Interactions

- How programs run on computers
- Processor design to serve SW better
- Performance programming, exploiting HW features

Topics and Textbook (홈페이지 참조)

- Part 1: Fundamental concepts and principles
- Part 2: ISA (HW-SW interface) design
  - Ch. 1: computer performance
  - Ch. 2: languages of computers: ISA
    - What is a good ISA? RISC-style ISA (MIPS)
    - How do programs run on computers?
  - Ch. 3: data representation and ALU
- Part 3: implementation of ISA (internal design)
  - Ch. 4: processors
  - Ch. 5: memory systems
- Short introduction to parallel processors

Why Architecture Class?

- Essential knowledge for all CSE majors
  - What is the machine called computer?
    - Principles, structure and operation
    - Efficient design and implementation
  - Many HW-SW interactions
- Software (smart use of machines) require knowledge of machines
  - What problems can we solve?
  - How best can we solve problems?
    - Algorithms for performance, reliability, power, …
- Architect: how can we build fast machines?

Why Architecture Class?

Applications	performance
System software	embedded, OS, compilers
Machines	design/implementation (architects)

Architecture Line of Classes

- 컴퓨터라는 기계의 하드웨어는 어떻게 만드나?
  - Digital logic design (AND, OR, NOT)
    - Hardware: CPU, memory, I/O
- 컴퓨터라는 기계를 사용해 봄으로써 무엇인지 이해한다

  (often special-purpose embedded systems)
  - 마이크로프로세서응용 또는 어셈블리프로그래밍
- 컴퓨터구조
  - Fast machines
    - 사용법 (instruction set) 설계 및 내부 구현
  - Interactions between hardware and software

타 학과생: 수강신청 조언

- "Digital logic design" 교과목을 선수강하지 않고 "컴퓨터구조론"를 수강하는 사례
  - 이 교과목을 따라갈 수 없음
- 이 학생들을 위한 매우 중요한 조언
  - "Computer Systems: A Programmer's Perspective" 라는 책을 대신 공부
  - 또는 이 책을 교재로 사용하는 "시스템프로그래밍" 교과목 수강

4) This Class (행정 사항)

Administration

- Textbook
  - Computer Organization and Design – The Hardware and Software Interface, Hennessy & Patterson
    - 5th Edition, (MIPS edition)
  - See the online companion materials in class homepage
- Undergraduate or first-year graduate level class
- 필수선수과목
  - 프로그래밍 능력 (프로그래밍 최소 2과목, 자료구조)
  - Digital logic design

Administration

- Class homepage updated weekly
  http://csl.hanyang.ac.kr/2020/ca (과제물 공지, 시험 공지)
- 한양인 (HY-in): 블랙보드
  - 수업자료 공지 (암호?)
  - 과제물 제출 (electronic submission)
    - 전반부: small weekly homework
    - 후반부: RISC processor design 등의 실습 과제
- 필기 시험: 중간/기말
- Tentative grading plan: 시험 90%, 과제물 10%

Administration

- Let's go through the class homepage and the first homework

Homework #1 (see Class Homepage)

- Based on discussions in first class, write a report (about 3 pages of dense report) to answer following questions:

  1) what is the machine called computer?

  2) what is Computer Science and Engineering?

  3) what kinds of training does CSE department provide?

  4) what are key topics in Computer Architecture class?

- Submit electronically to Blackboard
  - Due date: see Blackboard

- Study lecture notes - you should be able to give a lecture with them

End of Lecture 1

- Next lecture
  - Topic 0-1: invention of computers and digital logic design (see class homepage)