Class Topics (클래스 홈페이지 참조)

- Part 1: Fundamental concepts and principles
  1) Invention of computers and digital logic design
  2) Abstractions to deal with complexity
  3) Data (versus code)
  4) Machines called computers
  5) Underlying technology and evolution since 1945
- Part 2: 빠른 컴퓨터를 위한 설계 (ISA design)
- Part 3: 빠른 컴퓨터를 위한 구현 (pipelining, cache)

Machines Called Computers

Part 4

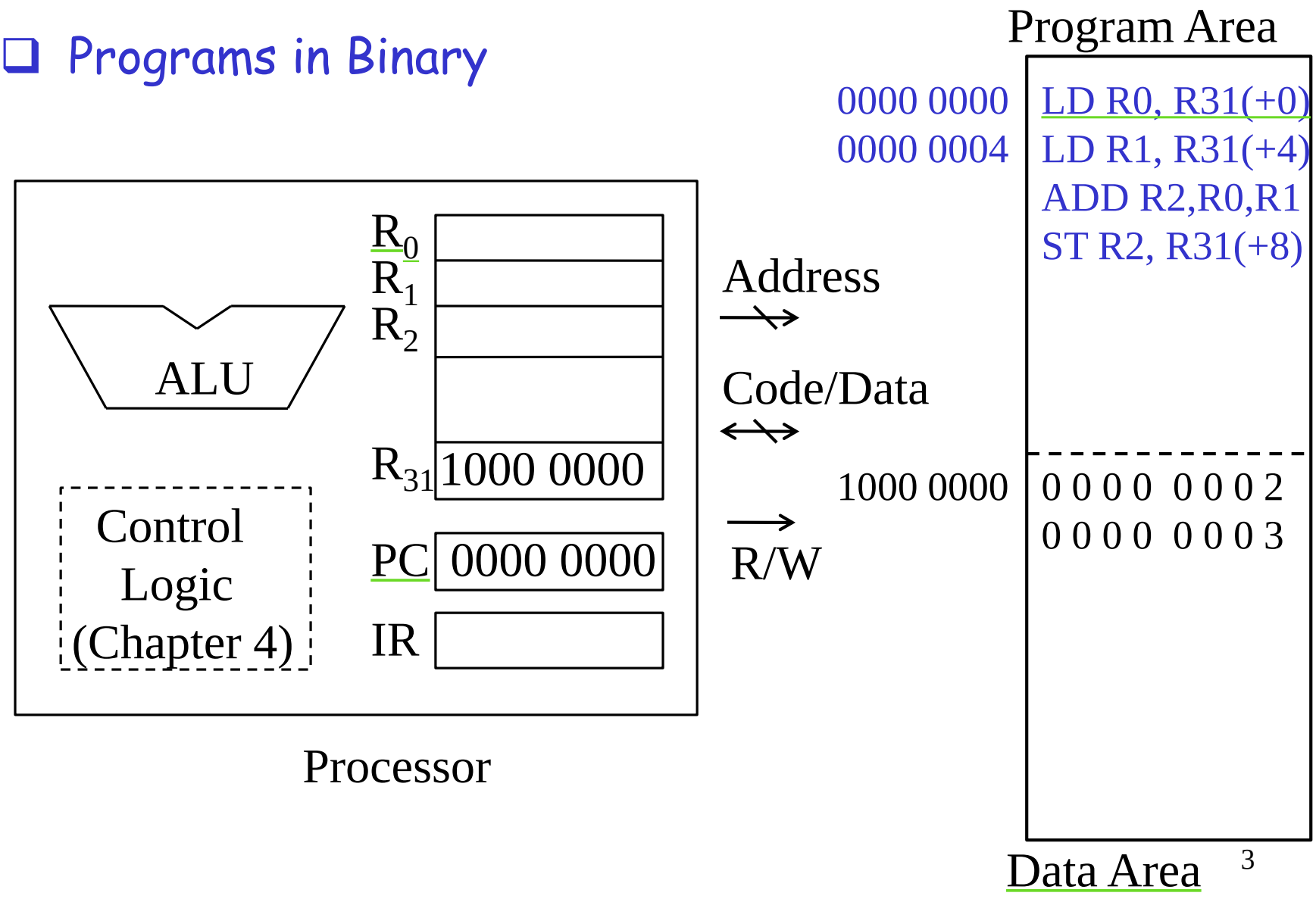
Fetch-decode-execute,
Stored Program Computers,
ISA (Instruction Set Architecture)
(Related to Textbook Chapter 2)

References:

1. Computer Organization and Design & Computer Architecture, Hennessy and Patterson (slides are adapted from those by the authors)

To Start Program Execution

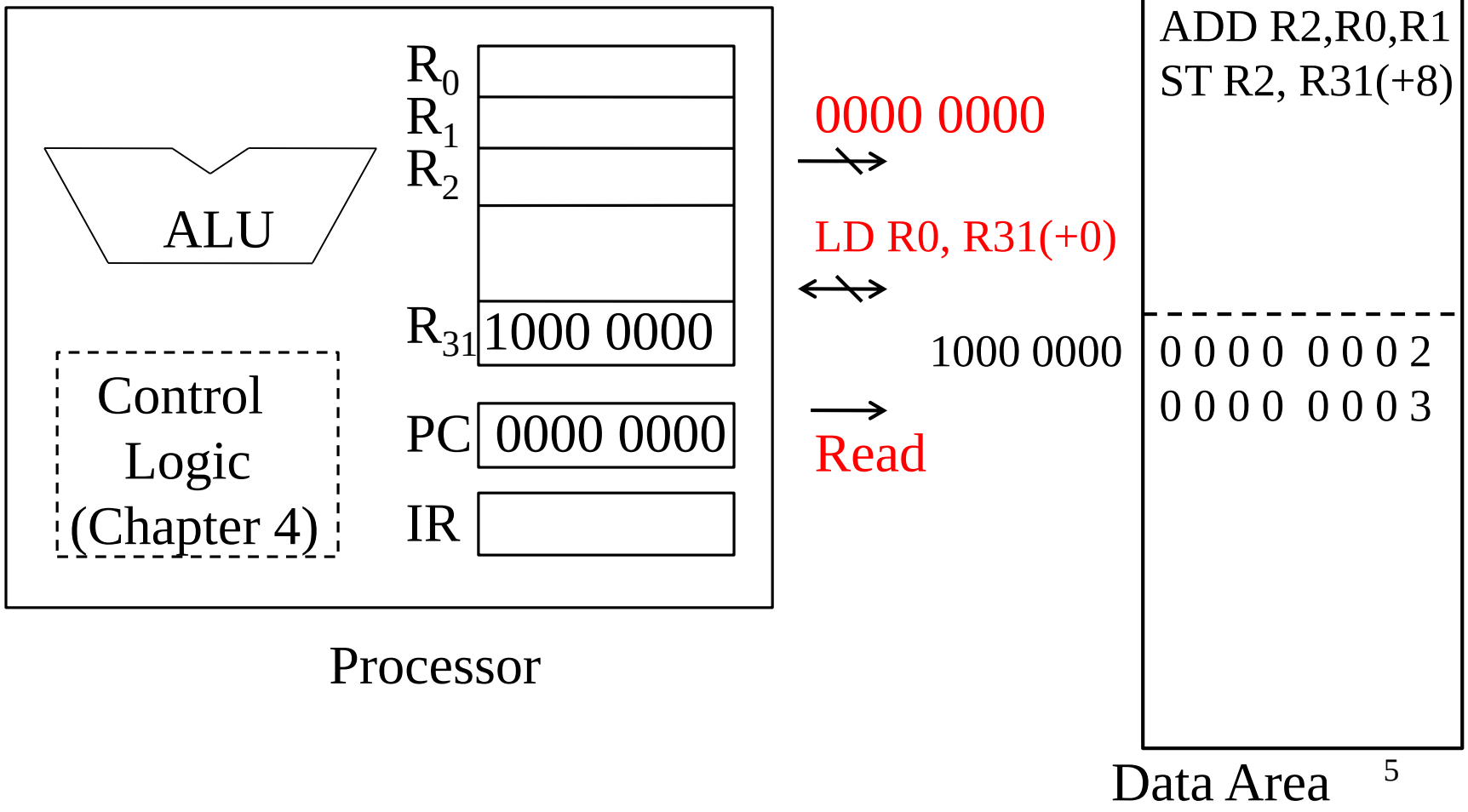
- Programs in Binary



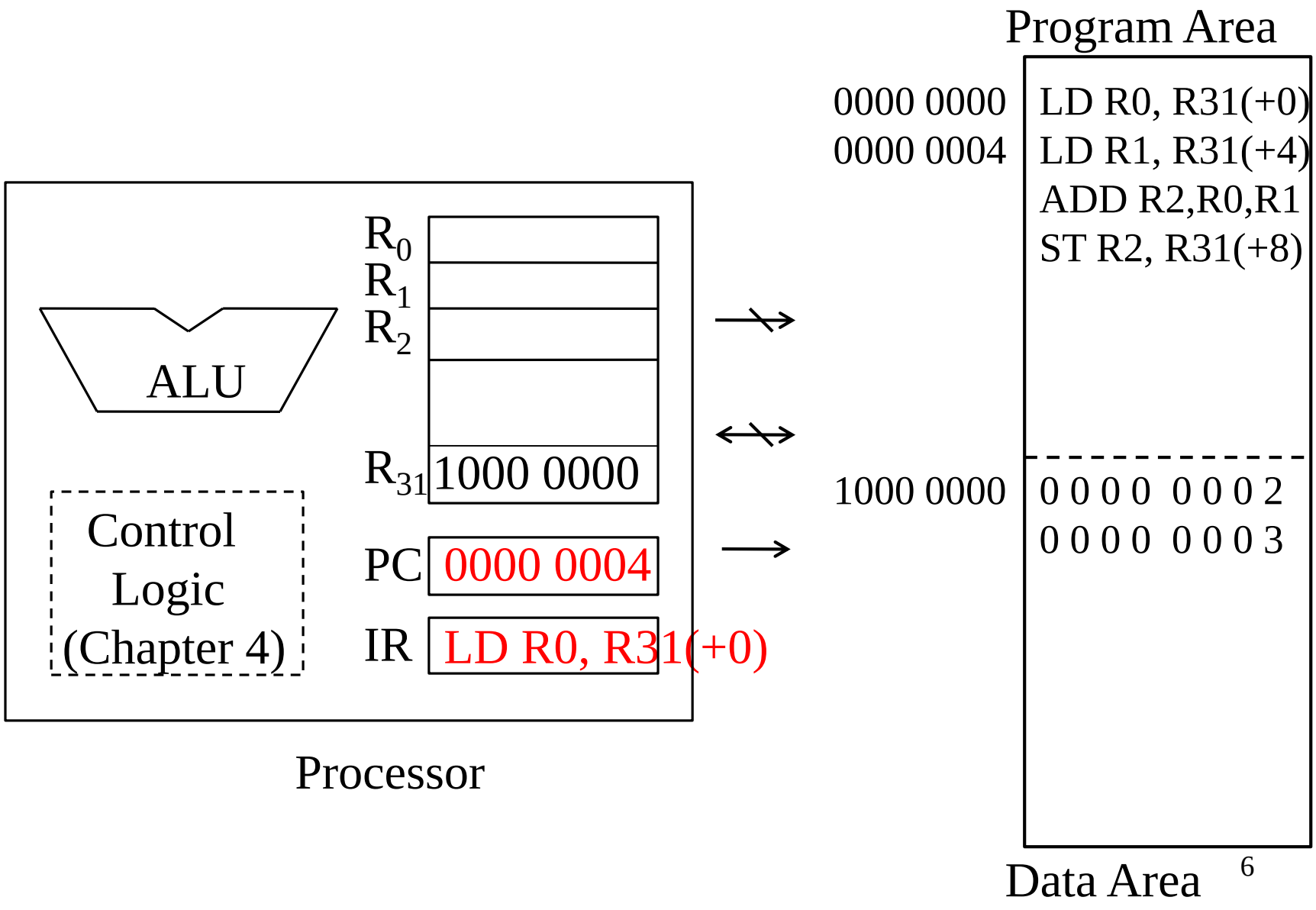
Fetch-Decode-Execute

- Set up for execution
  - Program at known location
    - PC (program counter) = $0000\ 0000_{HEX}$
      - † PC: address of instruction to execute next
  - Data at known location
    - R31 = $1000\ 0000_{HEX}$
- Program for adding two numbers
  - 4 machine instructions
  - Assume all instructions are 4-byte long (RISC style)

Step 1: Instruction Fetch (IF)



Step 1: Instruction Fetch (IF)



Step 2: Instruction Decode (ID)

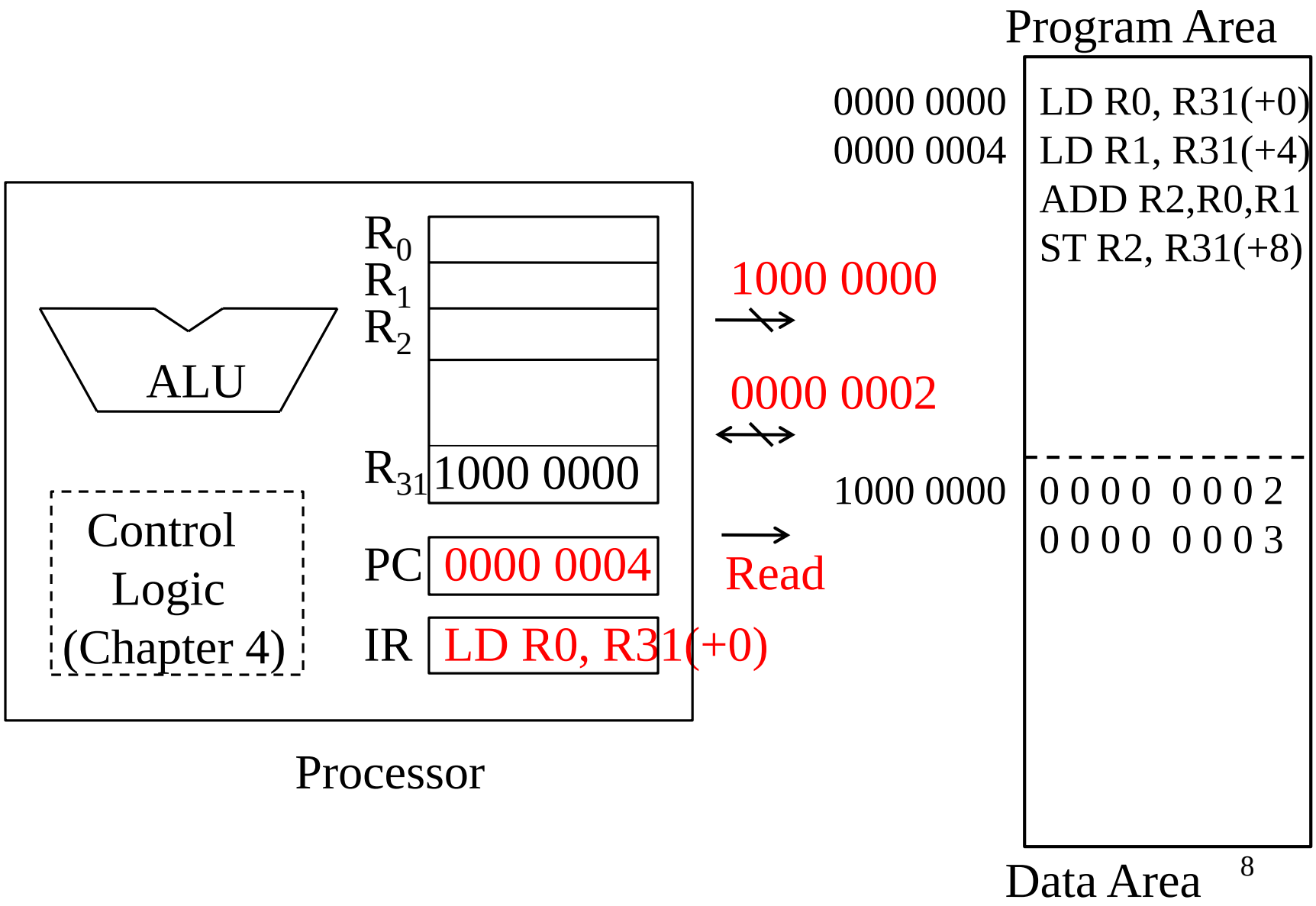
- Control logic of processor
  - Examine the content of IR (i.e., fetched instruction)
  - Understand what it is

    LD, R0, R31(0) // load instruction (memory read)

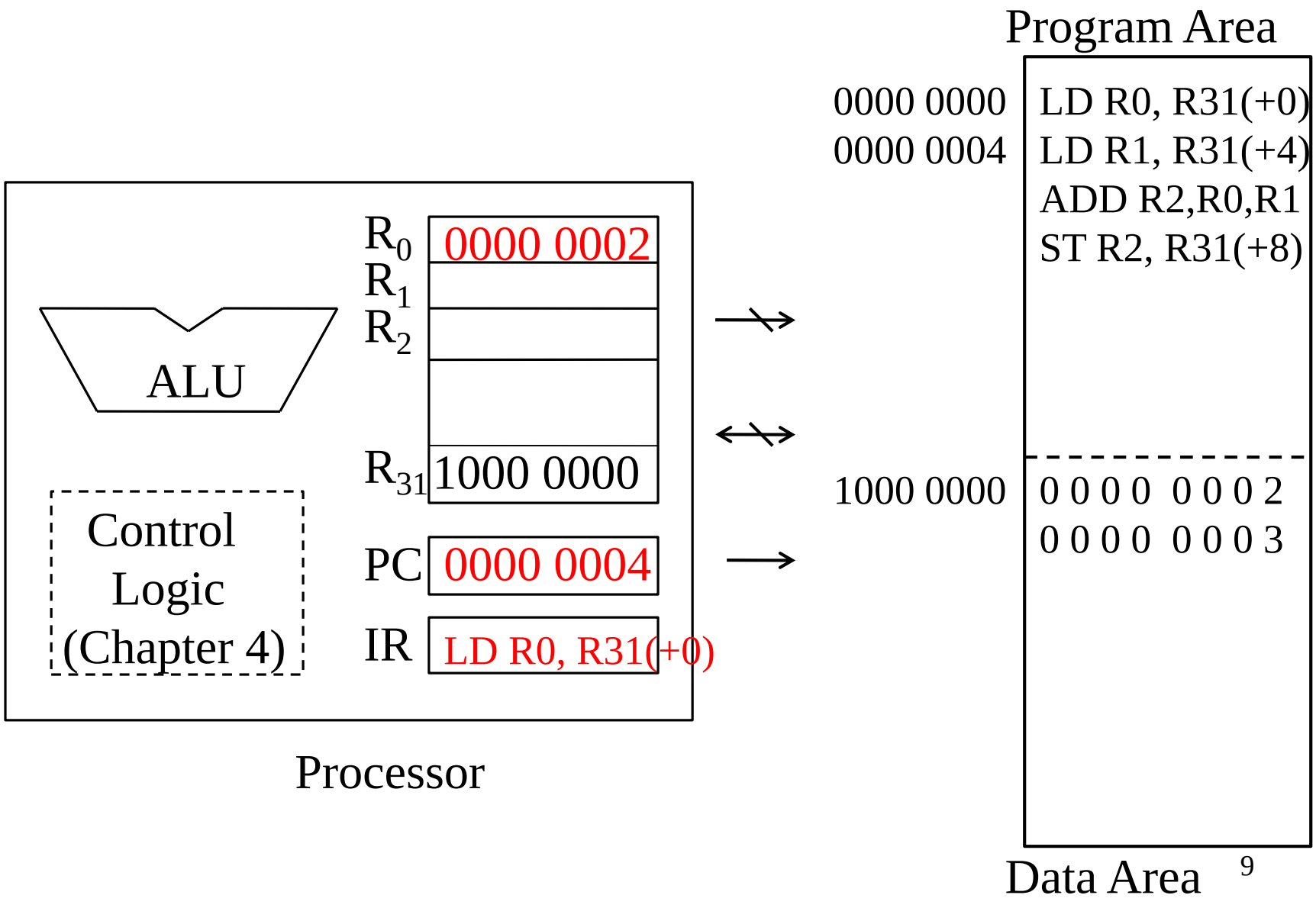
    R0 ← M[R31 + 0]

    R0 ← M[1000 0000]
  - Read memory address 1000 0000 and copy it to register 0

Step 3: Instruction Execute (EX)

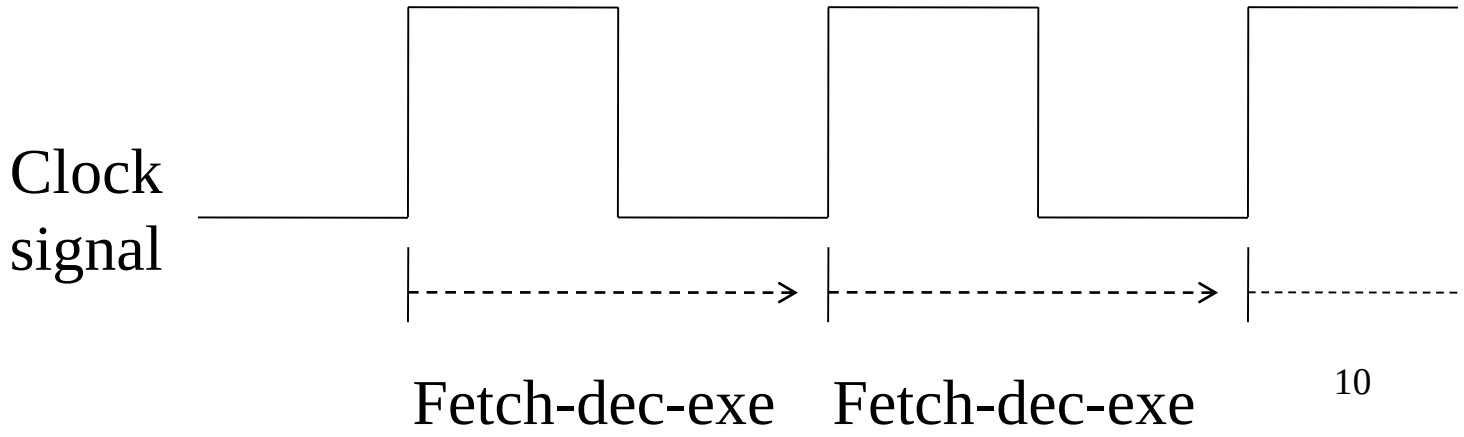


Step 3: Instruction Execute (EX)



Program Execution

- Three steps to execute instruction
  - Instruction fetch, decode and execute
- Changes in machine states after executing first instruction
  - PC: 0000 0000 → 0000 0004
  - R0 ← 0000 0002



More on Load Instruction

- Copy memory contents (instruction or data) to register

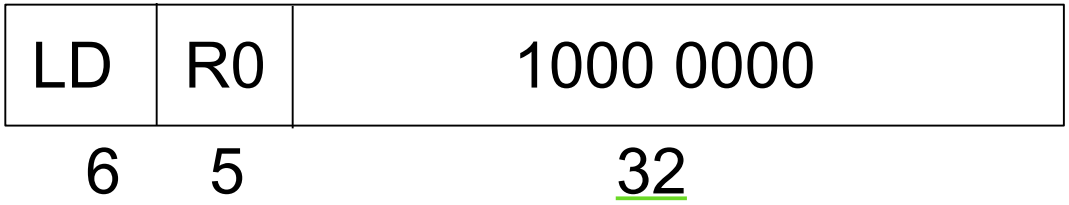
  LD, R0, R31(0) // memory read

  R0 ← M[R31 + 0] or M[1000 0000]

- Why not use absolute address ("LD, R0, 10000000")?

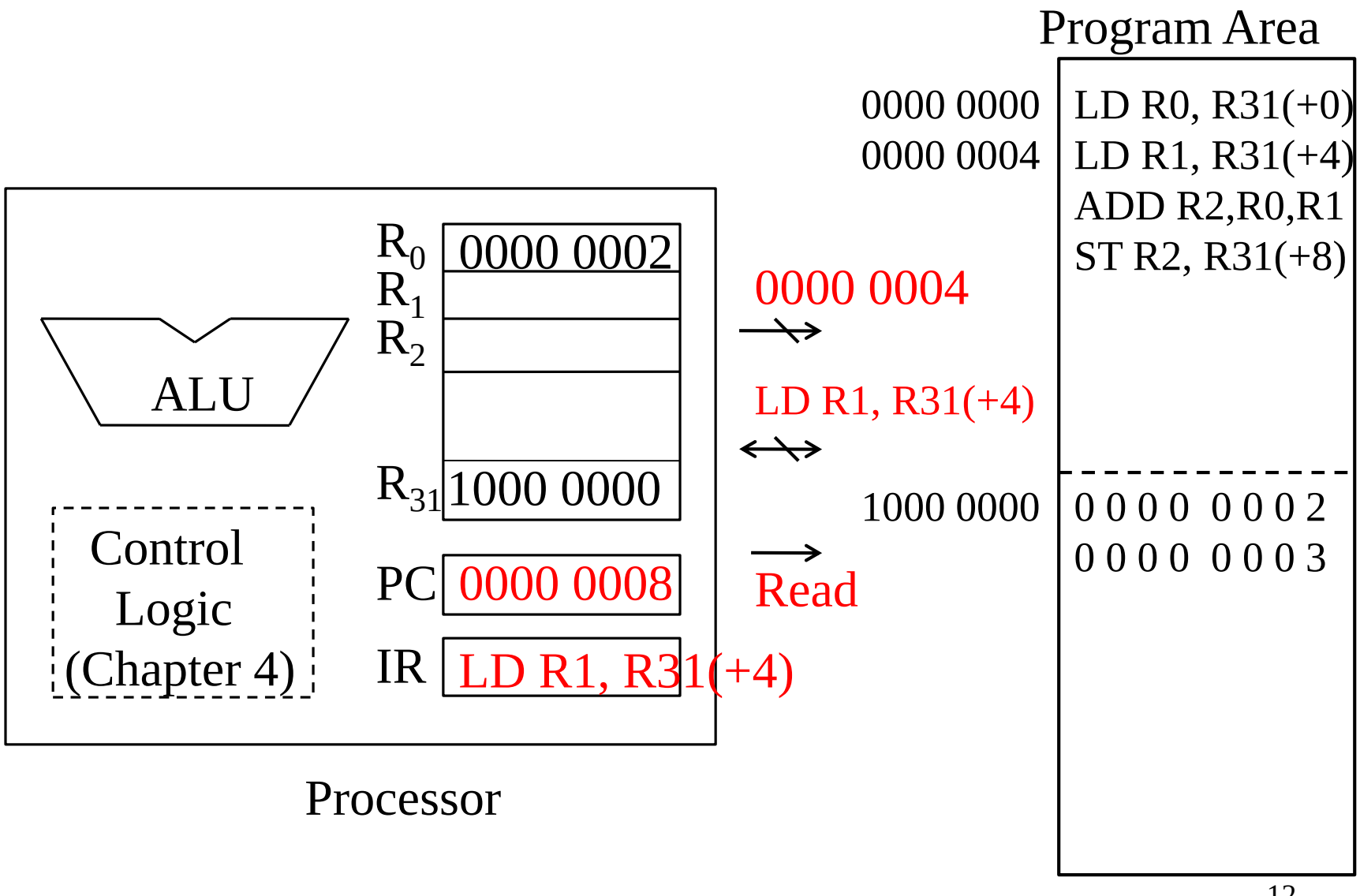
Instruction length

RISC style



† Store instruction: write register data to memory

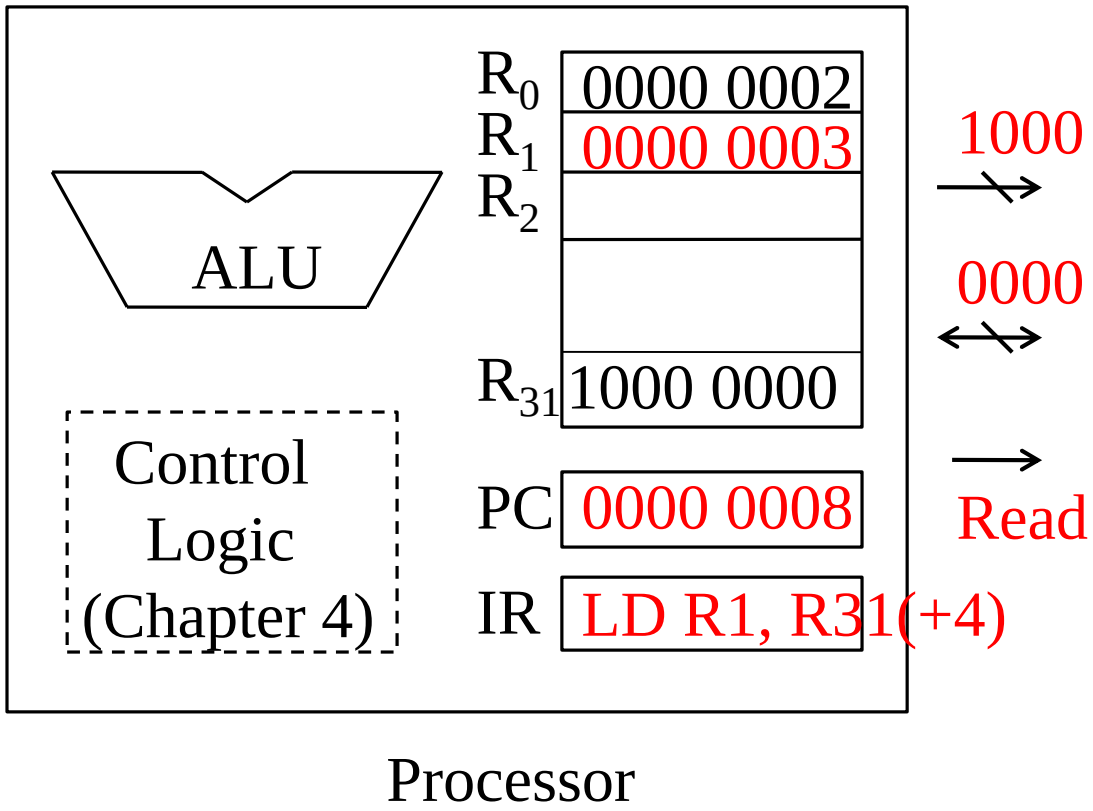
Fetch of 2nd Instruction

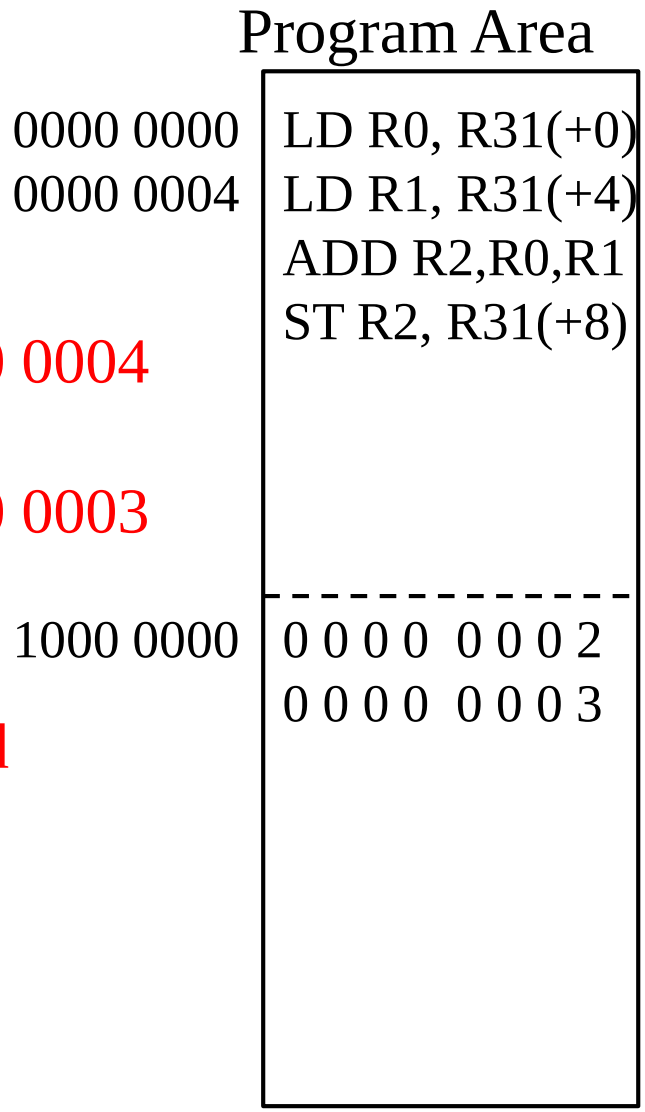


Decode and Execute 2nd Instruction

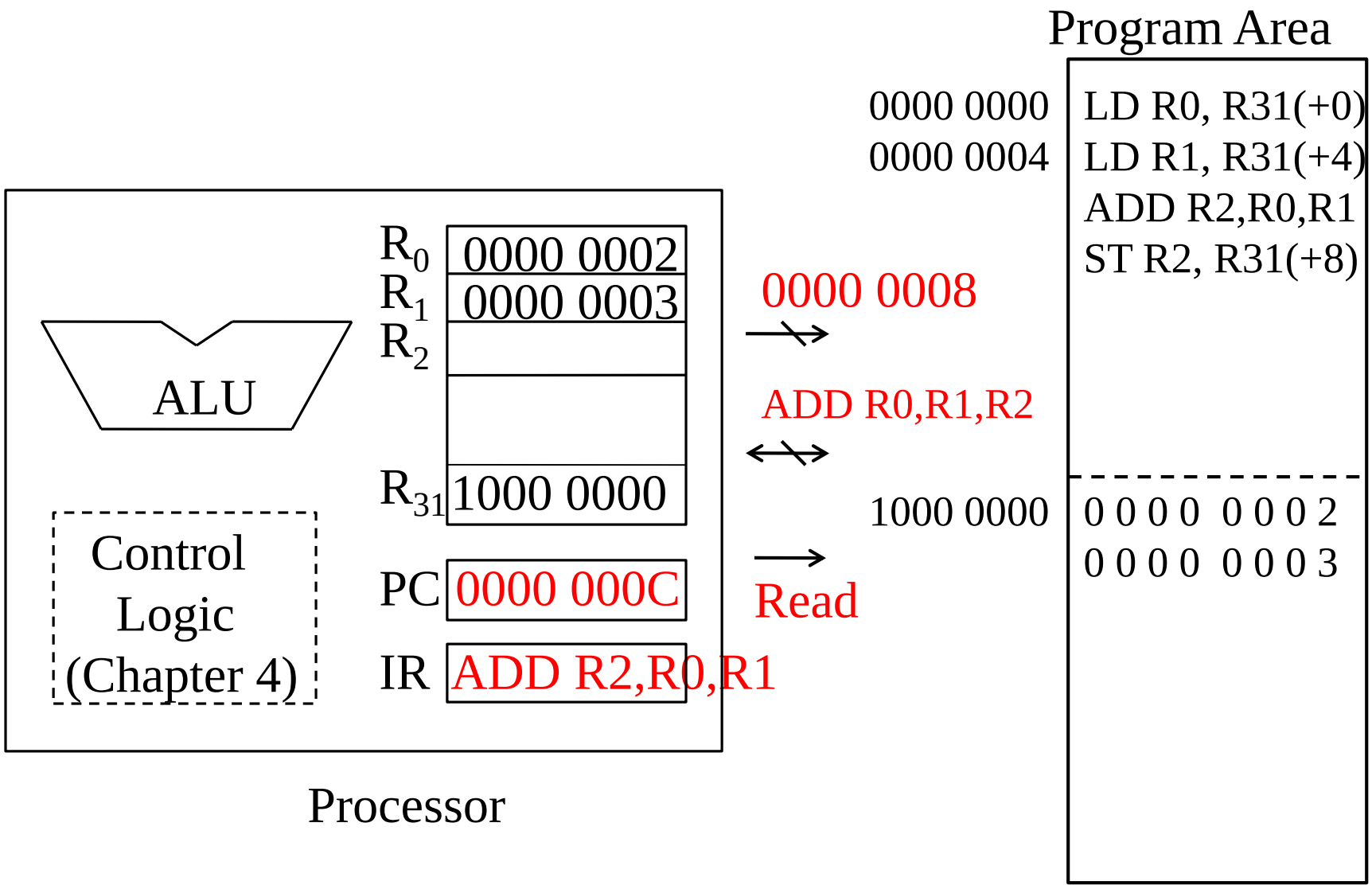
R1 ← M[R31 + 4]

R1 ← M[1000 0004]





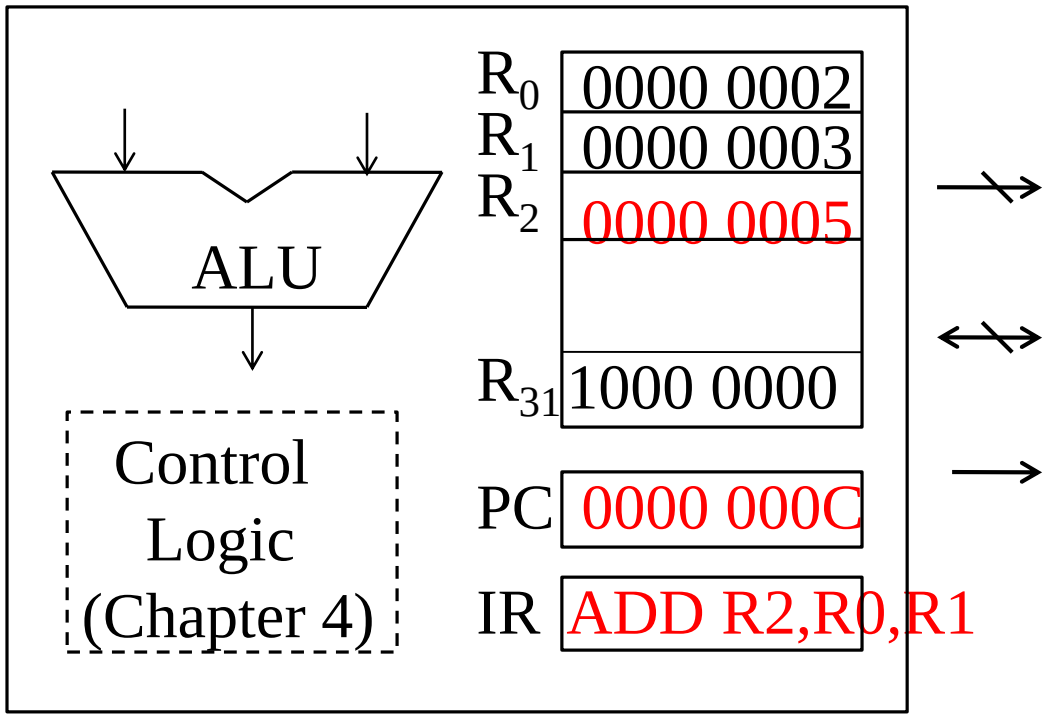
Fetch of 3rd Instruction

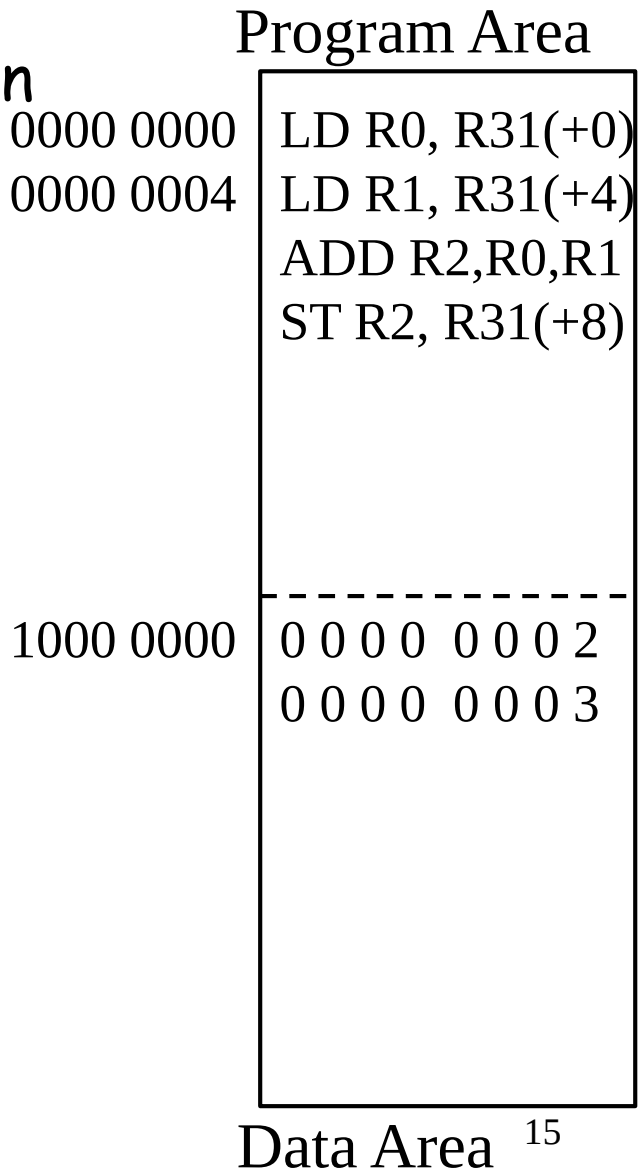


Decode and Execute 3rd Instruction

- ALU (processor internal) operation

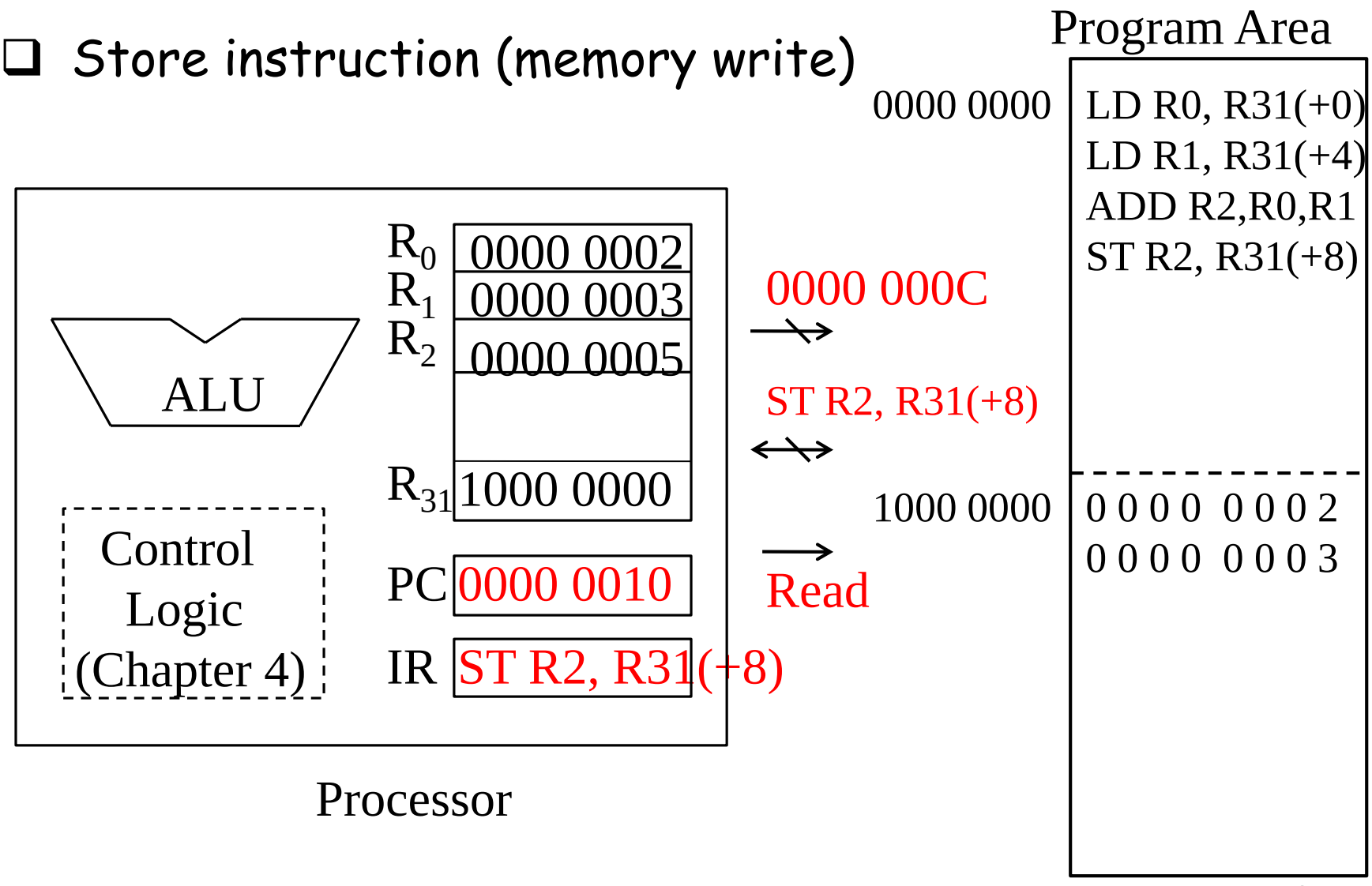
R2 ← R0 + R1





Fetch of 4th Instruction

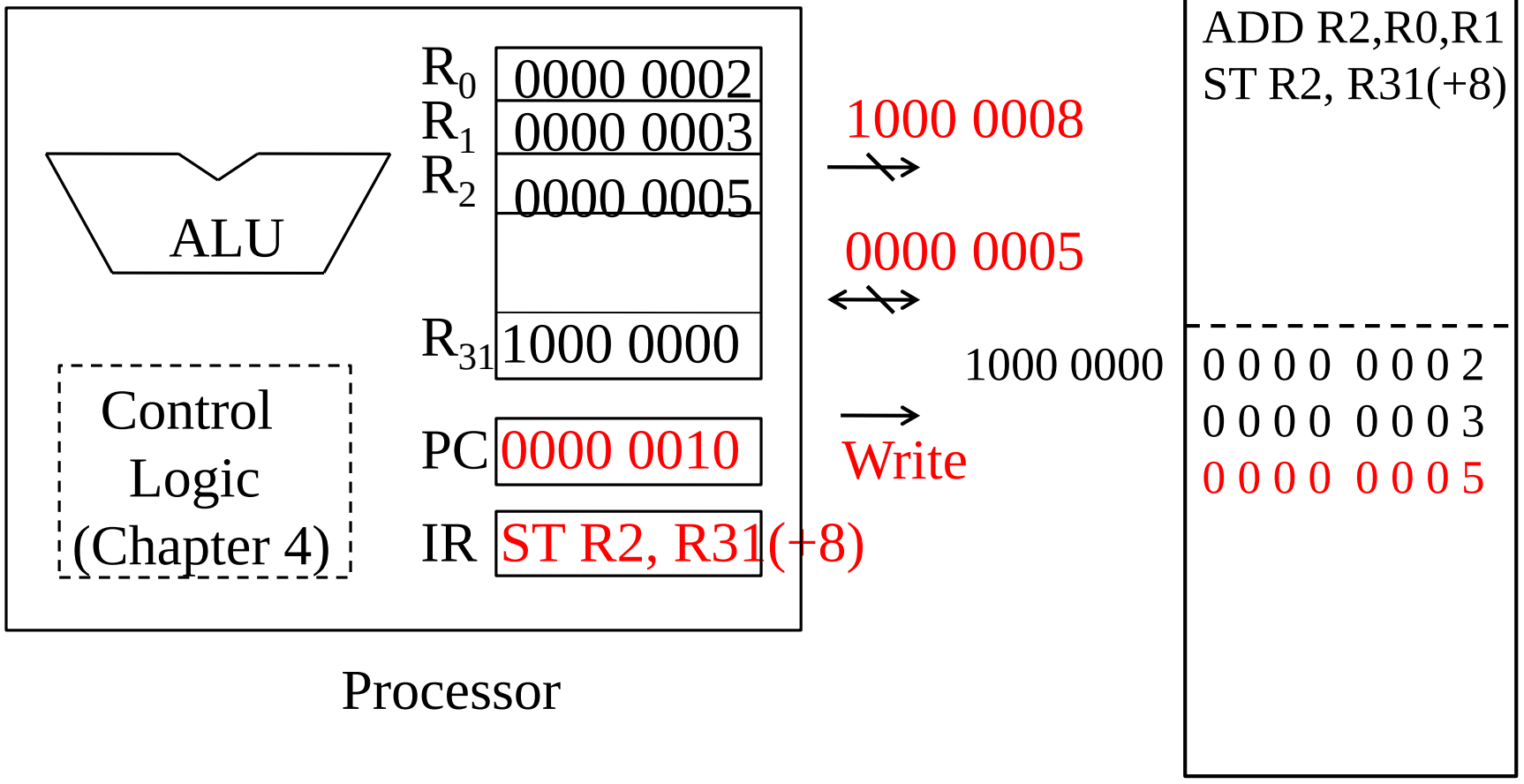
- Store instruction (memory write)



Decode and Execute 4th Instruction

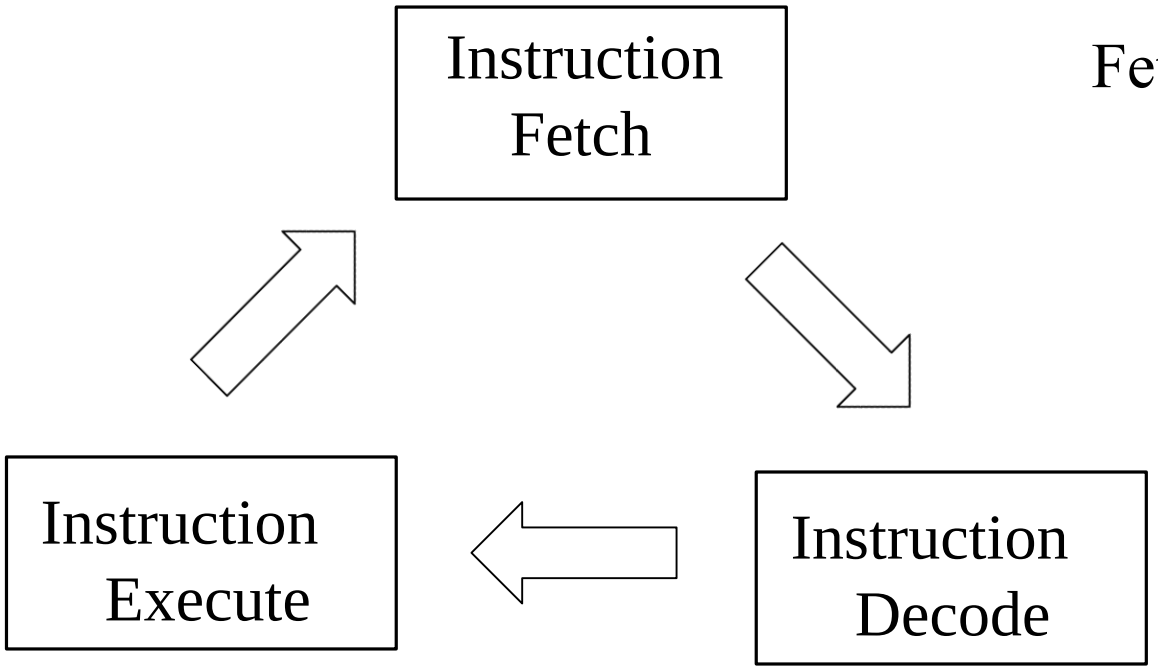
M[R31 + 8] ← R2

M[1000 0008] ← R2



Fetch-Decode-Execute Machines

- What is a processor? What is a computer?

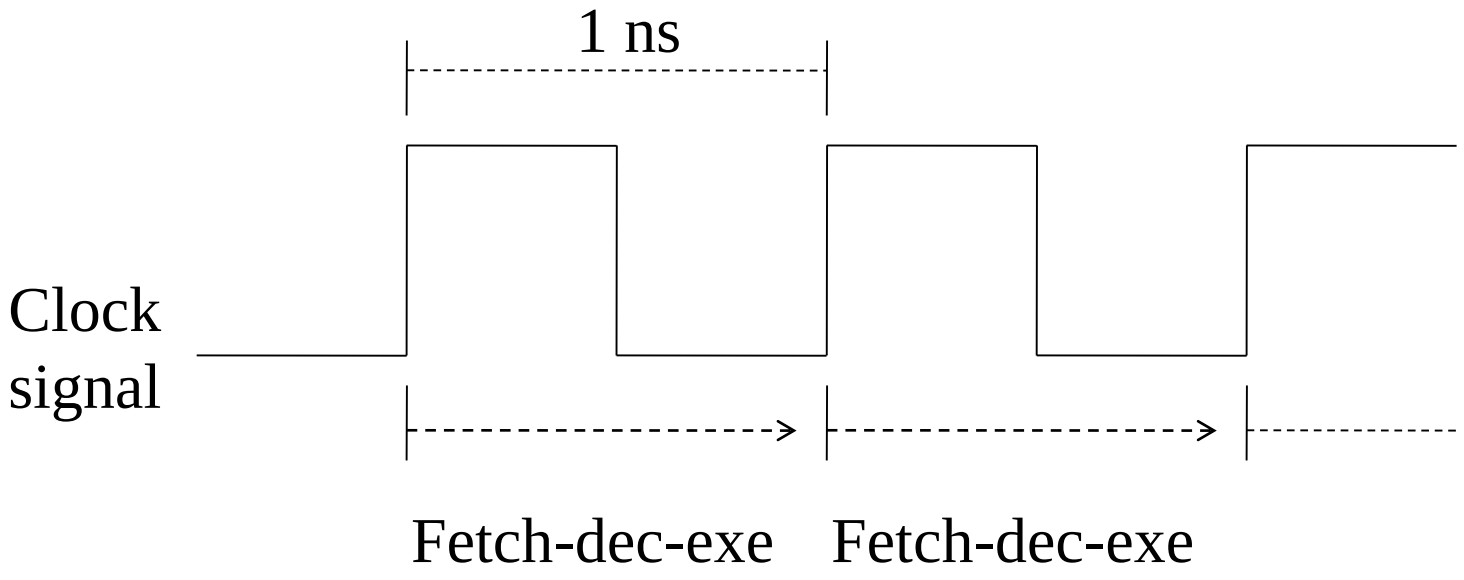


Fetch: IR ← M[PC]
PC ← PC+4

- 우리가 실제로 설계 및 구현함 (성능 고려) - Chapter 4

Time Behavior

- Imagine 1 GHz processor
  - Use 1 GHz clock; clock period of 1 ns - how long is it?
  - Simplified model:



† 인간과 기계의 계산 및 저장 능력 비교

ISA (Instruction Set Architecture)

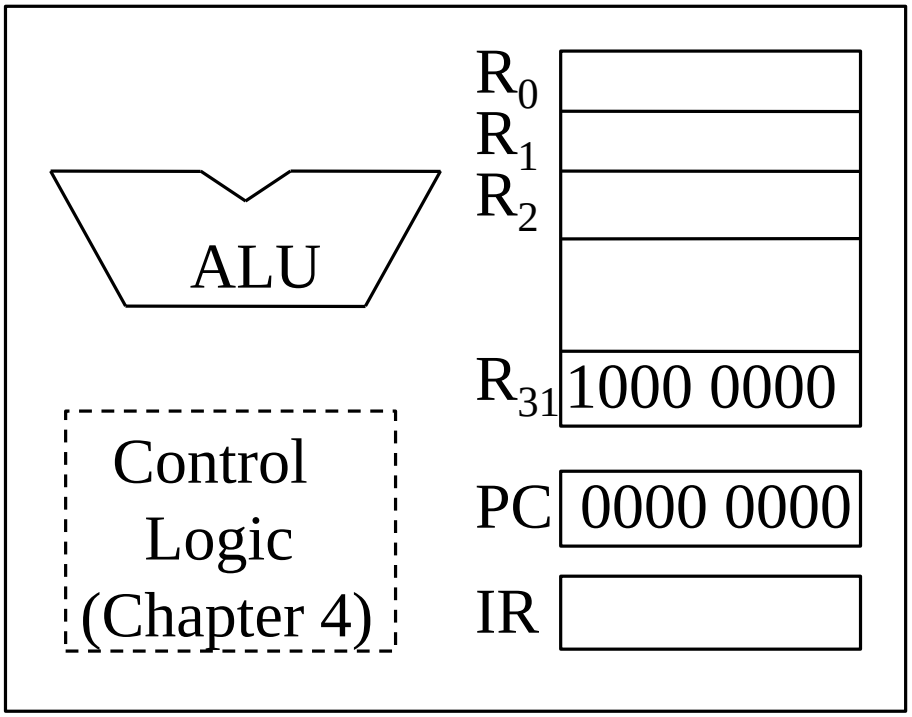
(몇백 개의 machine instructions)
(Machine 이 제공하는 서비스)
(우리가 machine 을 구매하는 이유)
(프로그래머는 이것을 써서 프로그램 만듬)
(Most important interface)
(Hardware/Software Interface)

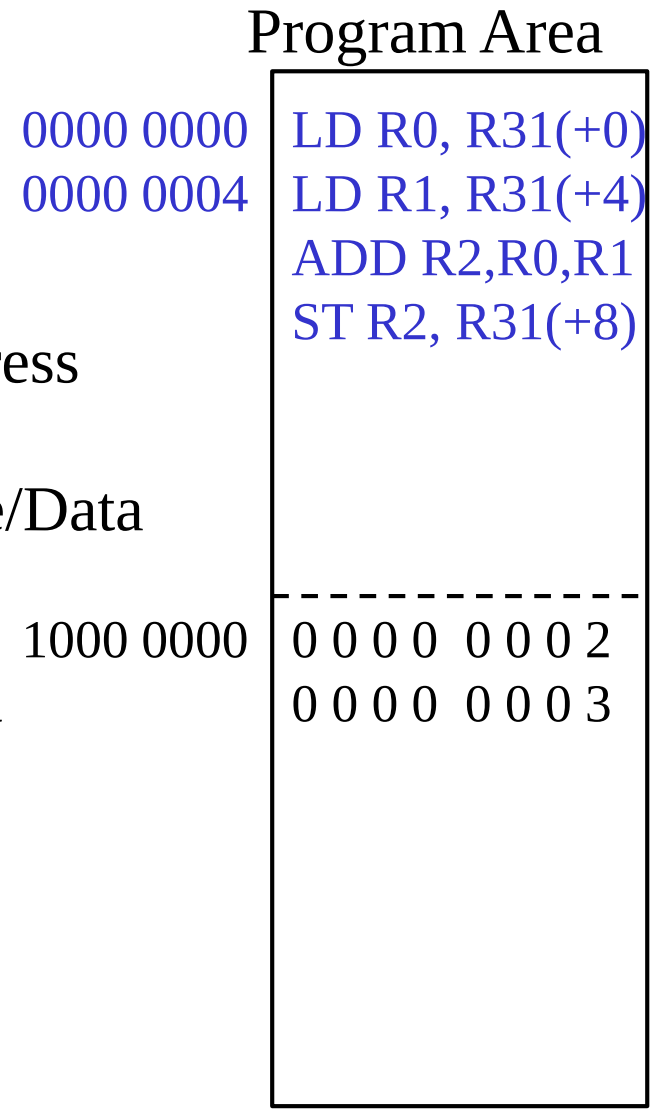
Final Puzzle of Machines Called Computers

- Computer organization
  - Digital logic design, abstraction
    - AND/OR/NOT gates → processors, memory, I/O
  - Programs and data in memory
  - Notion of address (for memory locations, I/O devices)
- Operation
  - Fetch-decode-execute cycle
- What is missing?
  - ISA (Instruction Set Architecture)
    - Kind of instructions needed to be a "computer"

Program Execution (반복)

- Programs in Binary





- I/O devices are just like memory
  - Load, store 의 대상

"IF" by Boole

- ALU instructions and memory access instructions
- IF: essential for problem solving, programming

IF (고객이 어린이) 입장료 = 정가 * 0.7

- Conditional jump instructions (PC 값 바꿈)

```
SUB  R1, R9, #10        //  R1 = age - 10
JUMP-POS  R1, #4        // if positive, PC ← PC + 4
MULT  R3, R3, #0.7      // if positive, skip (no discount)
```

- Six types of jump instructions: =, ≠, >, <, ≤, ≥

Instruction Set (Architecture)

- ALU instructions
  - 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 =, ≠, >, <, ≤, ≥

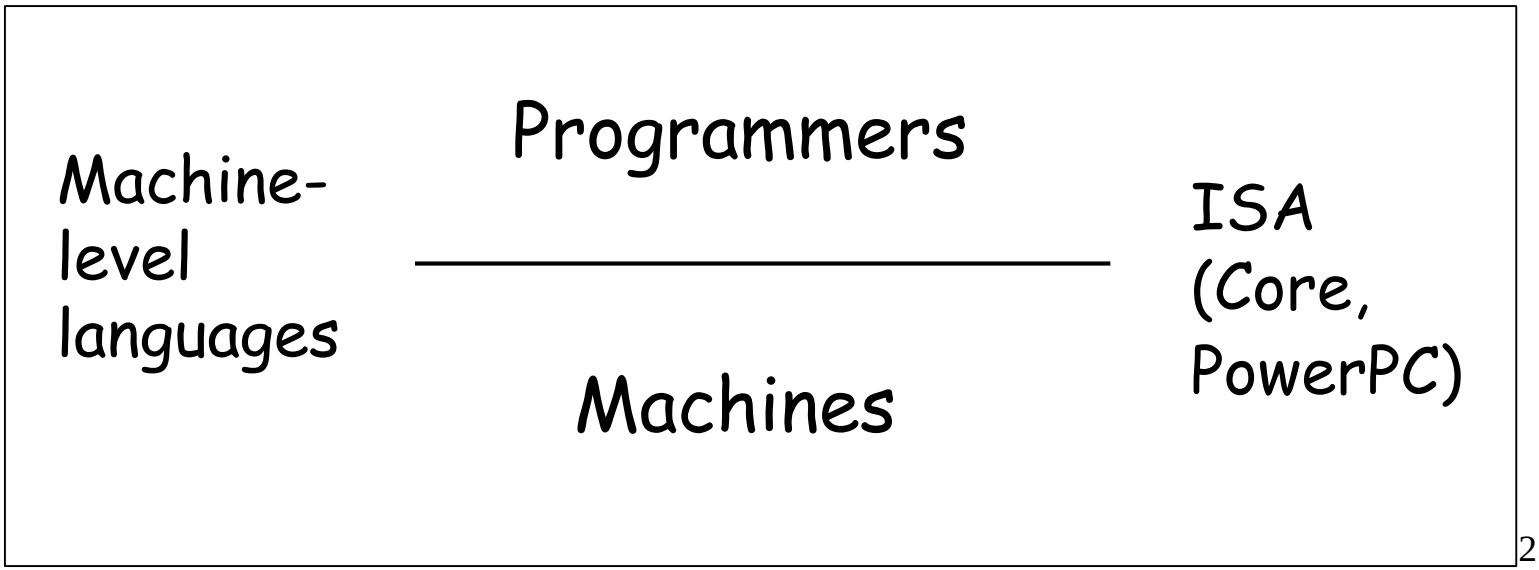
† With these, we have been computing for 70 years!

† Power of AND, OR, NOT, IF by Boole

  - "The Laws of Thought" and computers

Machines Called Computers (반복)

- Function determined by programs
- Doing useful work with computer requires
  - Problem solving by programming



Machines Called Computers

- Why do you buy it? For what service?
- What kind of interface or abstraction does it provide?
  - ISA
- What is a processor (or a machine)?
  - What implements an ISA
- Know it's fast; Does it look intelligent?
  - Once the software for solving differential equations is developed, then the problem is solved!
    - Reliable, very fast
  - Problem-solving by programming

컴퓨터의 한계

- 컴퓨터는 계산을 위해 만들어진 기계
- 컴퓨터가 빨라지고, 복잡하고 스마트한 소프트웨어 출현
  - 보다 복잡한 계산을 자동화 (스포츠 기사 작성, 주가 분석)
- 인공지능
  - Strong AI: 인간과 같이 생각하는 기계 (intelligence)
    - 아직은 (공상)과학의 영역, CS dream project
  - Weak AI: 이전보다 훨씬 스마트한 소프트웨어 (공학)
    - 알파고가 프로바둑기사를 누르다
    - 음성인식, 물체인식 (deep learning)

컴퓨터: 언어를 이해하는 기계? (가볍게)

- 기계에도 언어가 있는가?
  - 마이크로웨이브 오븐, 자동차
- 컴퓨터
  - Computer languages (기계적으로 처리 가능한 언어)를 이해
    - 모든 계산 및 논리적 처리 가능
- Natural languages (자연어, 인간의 언어: 한글, 영어; 진화)
  - 왜 그럴까? (창의성)
  - 오 맑은 햇빛, 너 참 아름답다 (감성)
- 다른 동물의 언어 (울음 소리?)

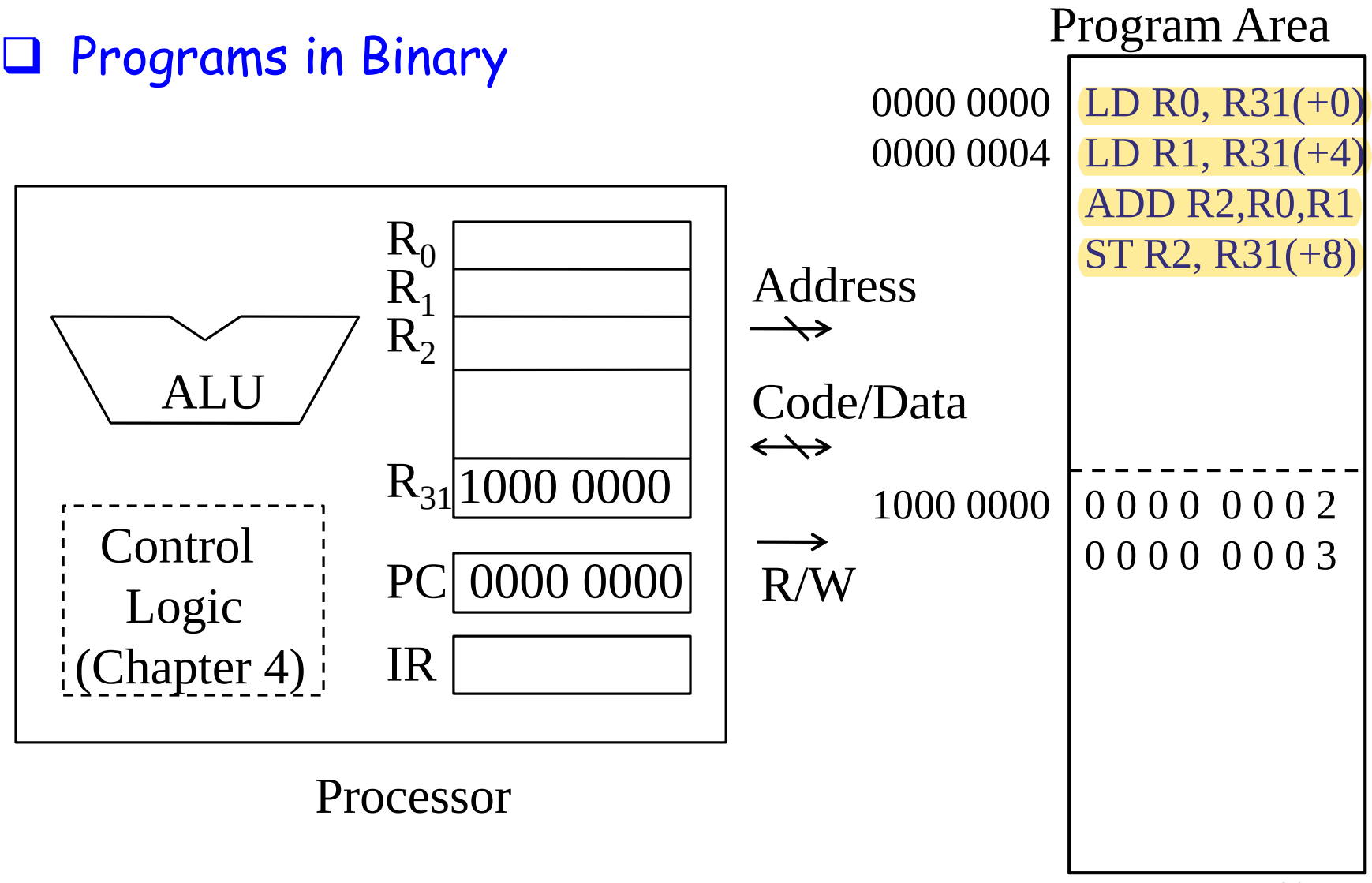
Programs in Binary

- Assembly and binary programming
- RISC-style machine instructions

  (related to Textbook Chapter 2)

Program Execution (반복)

- Programs in Binary

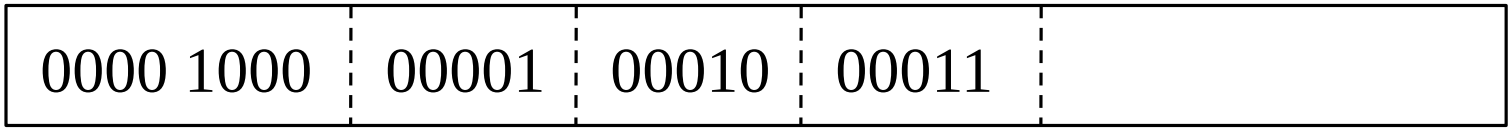


- I/O devices are just like memory

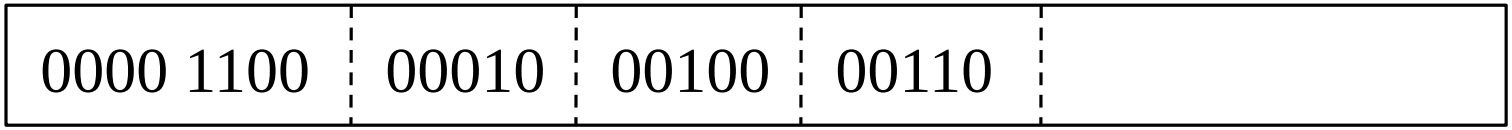
Assembly vs. Binary Languages (참고)

- ALU instructions: 32-bit long (opcode, operands)

ADD R1, R2, R3



OR R2, R4, R6

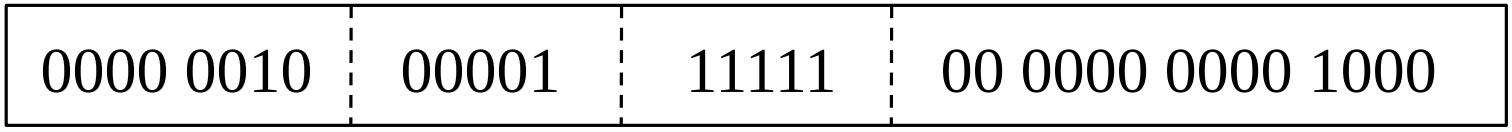


- The two are identical – both called machine languages
  - Simple 1:1 translation
  - Assembly language: mnemonic

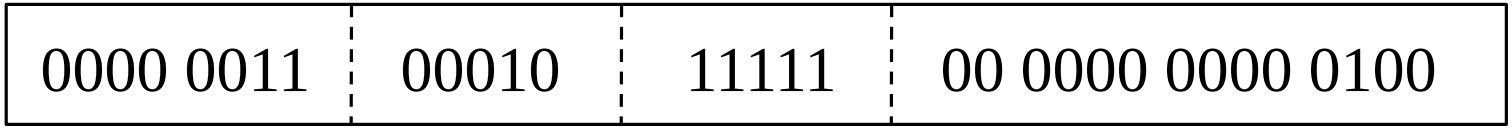
Assembly vs. Binary Language (참고)

- Load and store instructions: 32-bit long

LD R1, R31(8)



ST R2, R31(4)

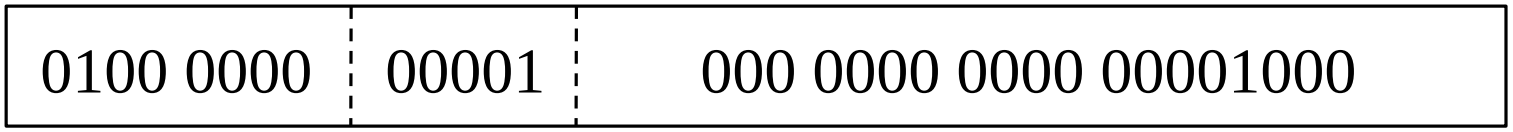


- Why not use absolute memory address?
- What is instruction decode?

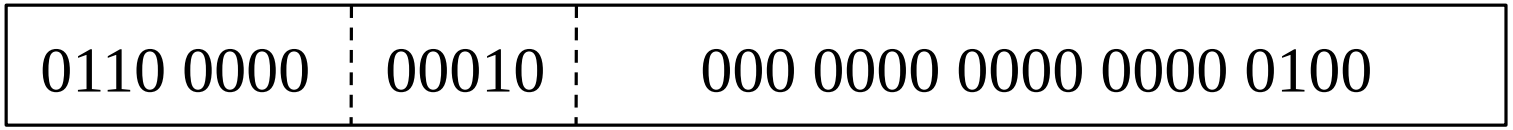
Assembly vs. Binary Language (참고)

- Jump instructions: 32-bit long

JUMP-NZ R1, 8



JUMP-POS R2, 4



- RISC-style machine instructions (Chapter 2)

Stored Program Concept

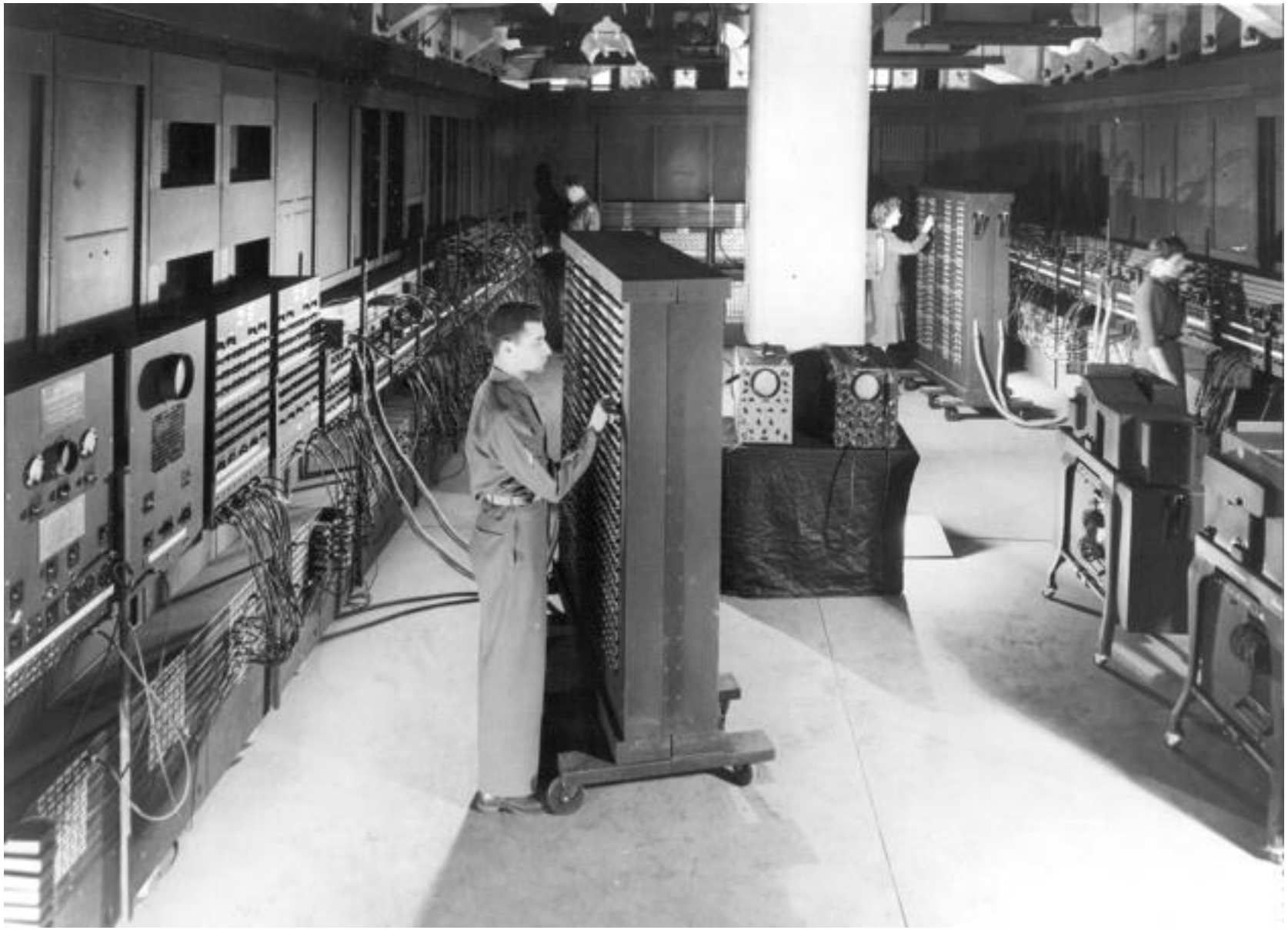
- Fetch-decode-execute
- Von Neumann architecture

Modern Digital Computers

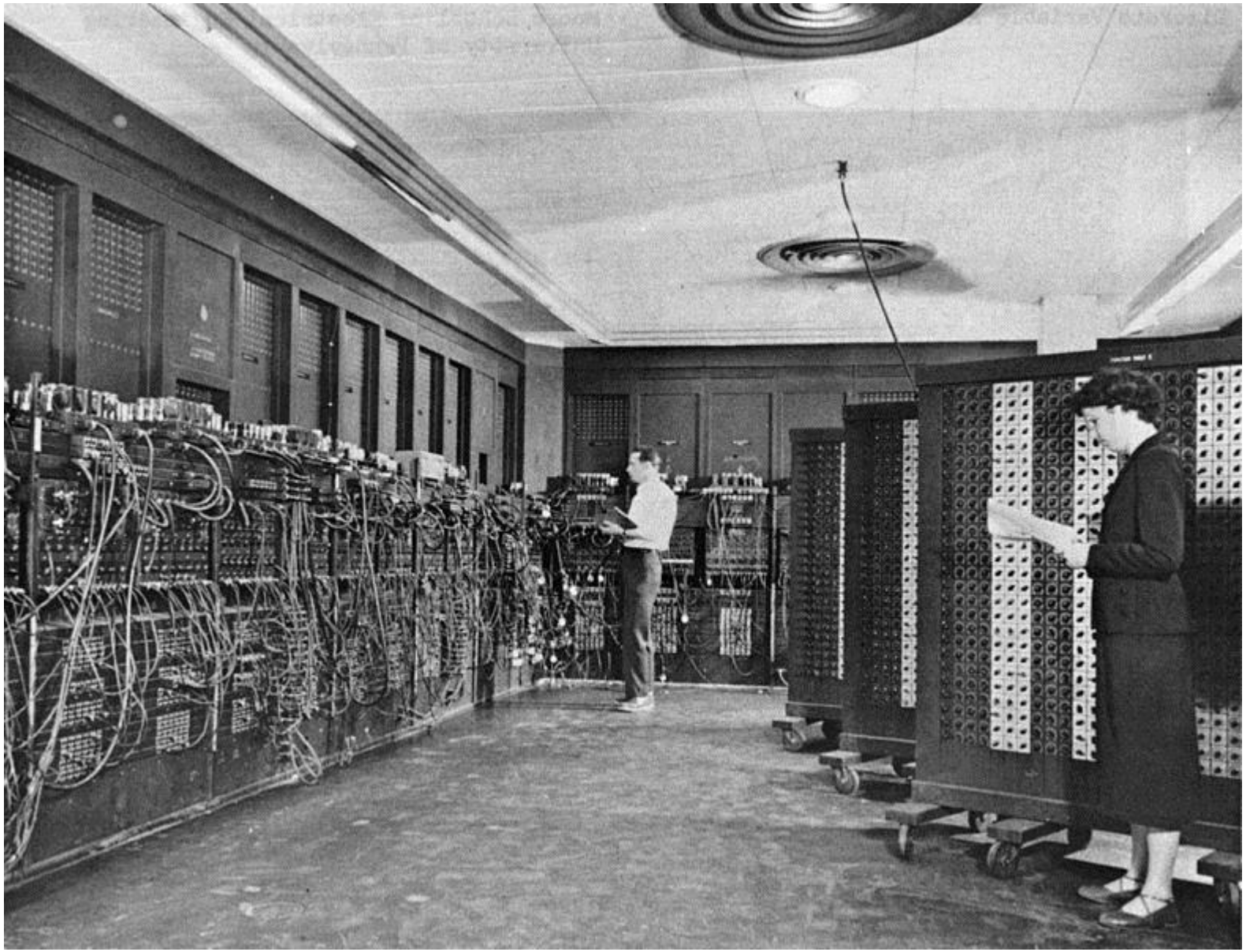
- Gradual evolutions to meet human computational need
  - Capabilities, design techniques, supporting technology

- ENIAC (1943-1946)
  - First fully-electronic, general-purpose computer
  - U. Pennsylvania., Eckert and Mauchly
  - Programs not in memory, vacuum tube

ENIAC (1943-1946)



ENIAC (1943-1946)



Modern Digital Computers

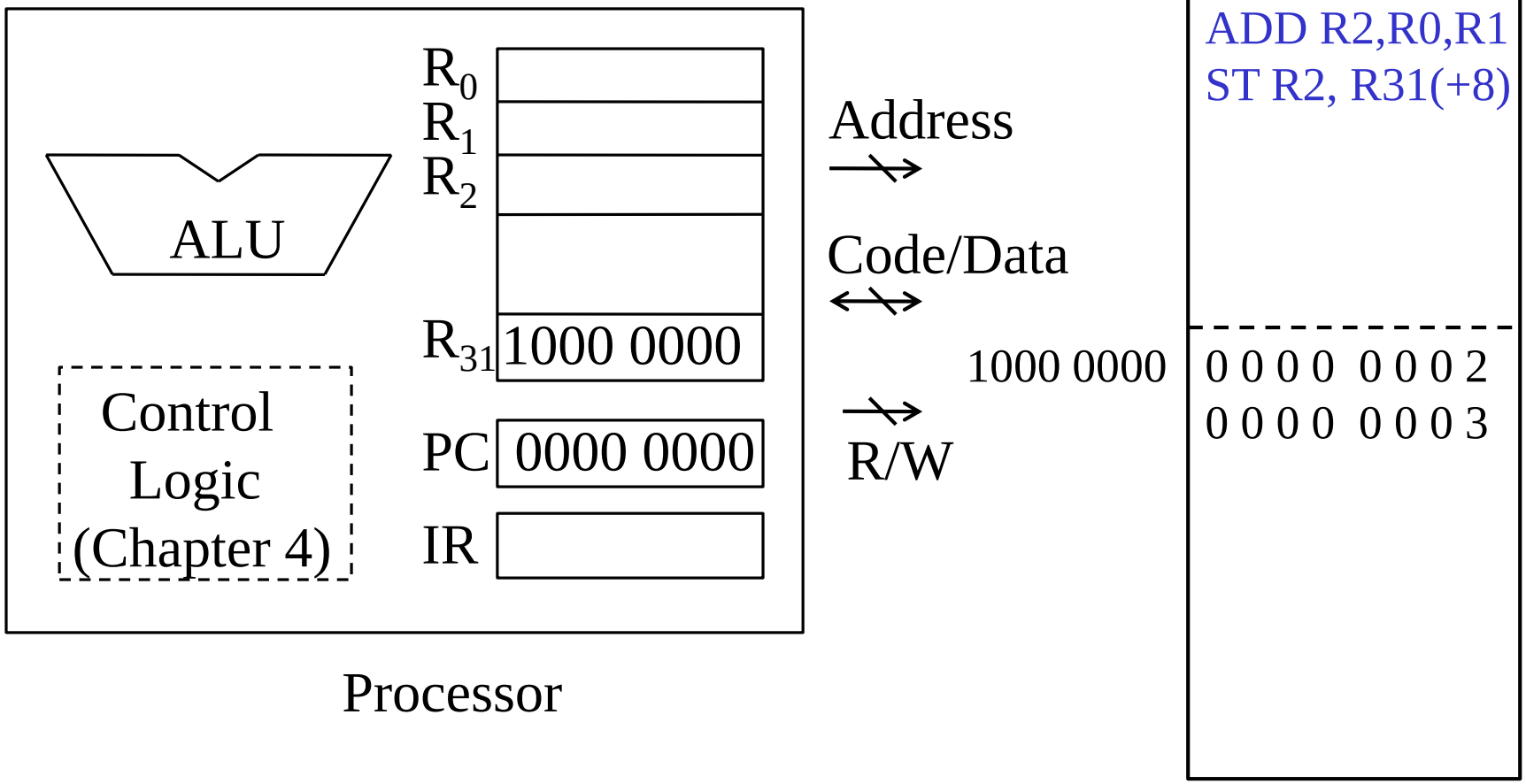
- What was a brilliant final touch?
  - Stored program concept in 1945 by von Neumann
  - Natural consequence: fetch-decode-execute

† Von Neumann architecture/bottleneck

† C. Babbage's idea in early 19C

Stored Program Concept (반복)

- Programs in memory
  - Fetch, decode, execute



† I/O devices are just like memory

Stored Program Concept

- Notion of machine instructions
  - Computation as a sequence of instructions
  - Treat programs like data
    - Fetch-decode-execute
- Flexible, general-purpose machines
- Facilitate high-level programming

Modern Digital Computers

- Completed by stored program concept in 1945
  - First stored program computers
    - UNIVAC I (1951), EDVAC (1952)
    - Earlier, smaller-scale British/US computers
- 70 years of evolution for performance (CA, engineering)
- Search continues (CA, science)
  - Non-von Neumann architectures
  - Alternate forms of computing
    - Biological, optical, quantum, dataflow, …
  - Possibly, new definition of "computing"

What is Computer Science?

(Problem Solving by Programming)

* 이 부분은 첫 주 수업자료의 반복, 부연임

Fundamental Paradigm (반복)

Problem recognition
(what to solve)

Solution methods
(how to solve, 알고리즘/수학)

Software tool development
(programming skills)

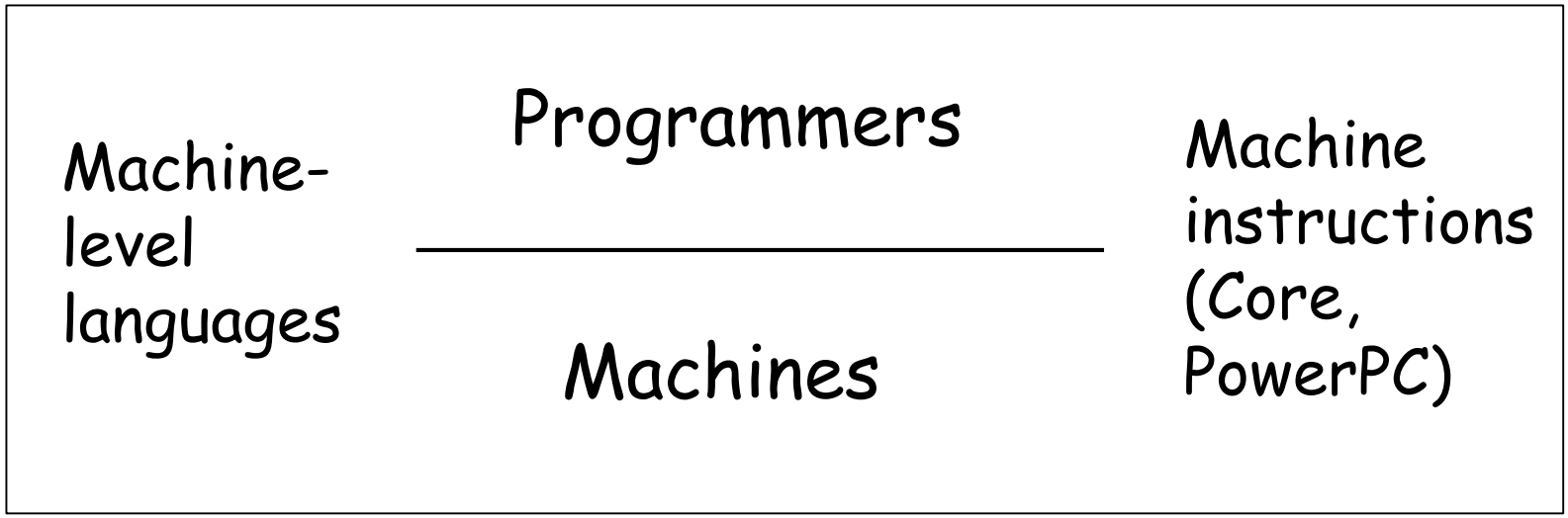
목표:
problem-solving
(창의성,전문지식)

수단:
programming
(구현 기술)

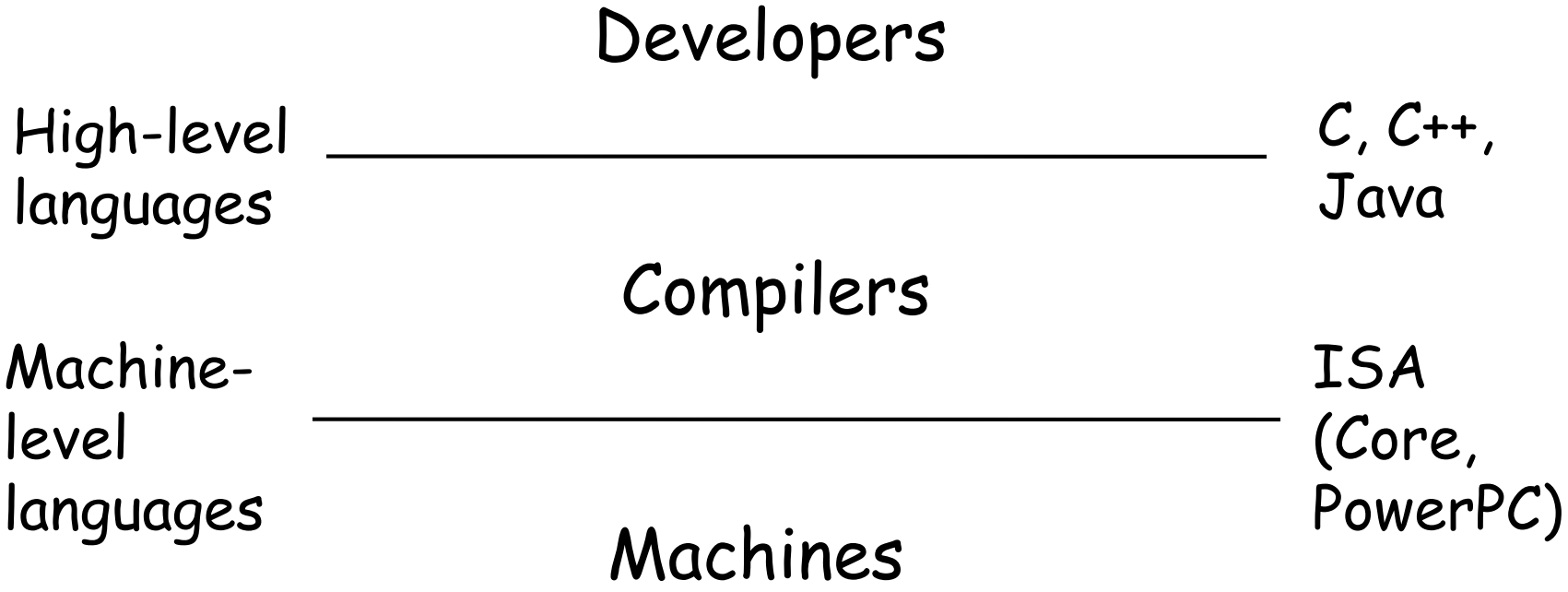
- Problem solving and programming: tightly coupled

Problem-Solving by Programming (반복)

- Telling computers what to do
- Machines provide low-level languages
  - Productive?



Two Major Interfaces in CS



- Computer languages vs. natural languages?
- Abstractions supported by ISA and HLL adequate?
  - Turing Award

Programming Languages and USA Dominance

Procedural:

- Fortran, 1957
- Algol, 1958
- Cobol, 1960
- Basic, 1964
- Pascal, 1970
- C, 1973

Object-oriented:

- Simula, 1967
- Smalltalk, 1980
- C++, 1985
- Perl, 1987
- Python, 1990
- Java, 1995
- Ruby, 1995

Functional: Lisp (1958)

Logic: Prolog (1972)

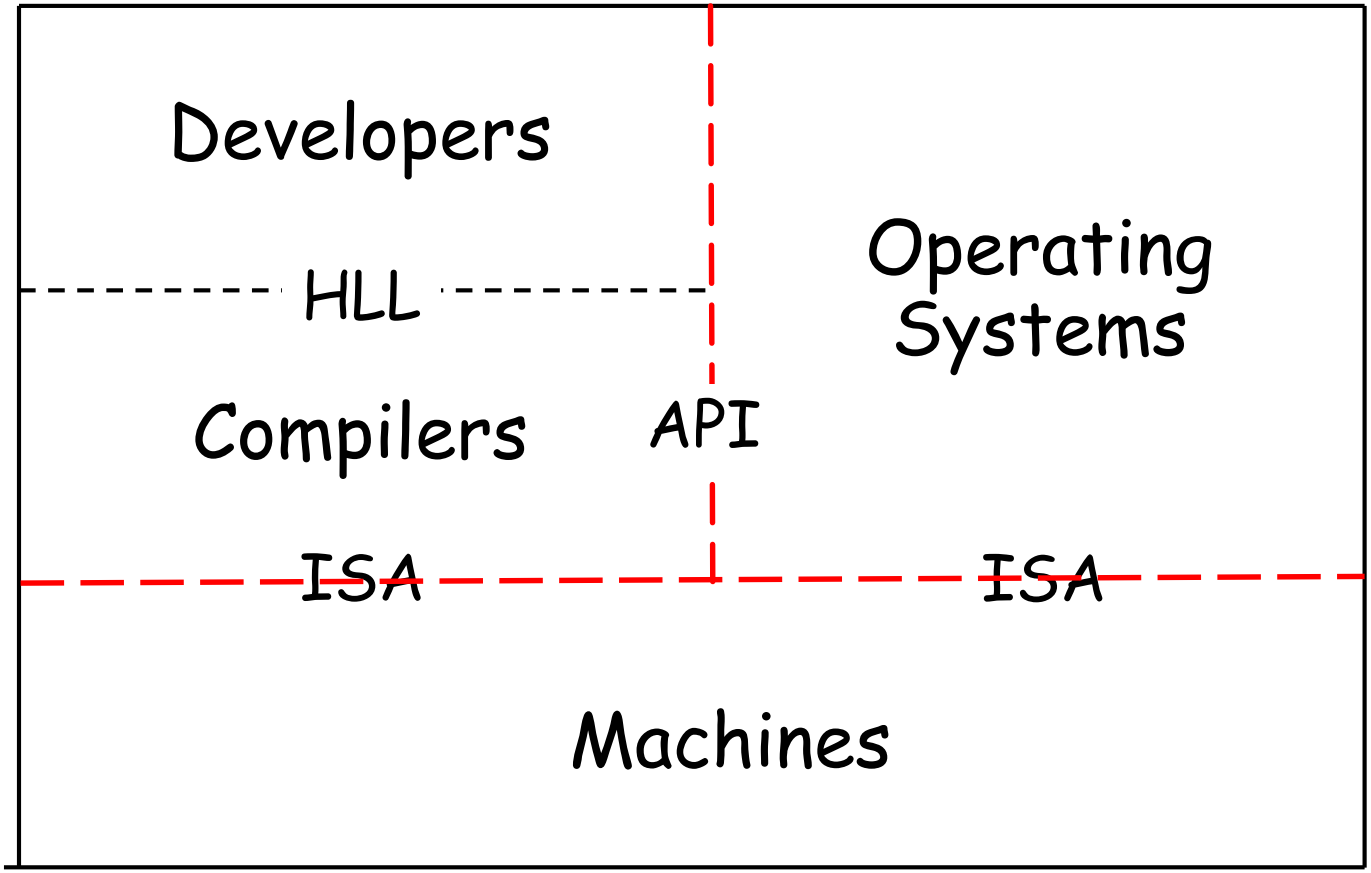
- What about processors, OS, database, Internet, …

What is CSE? (반복)

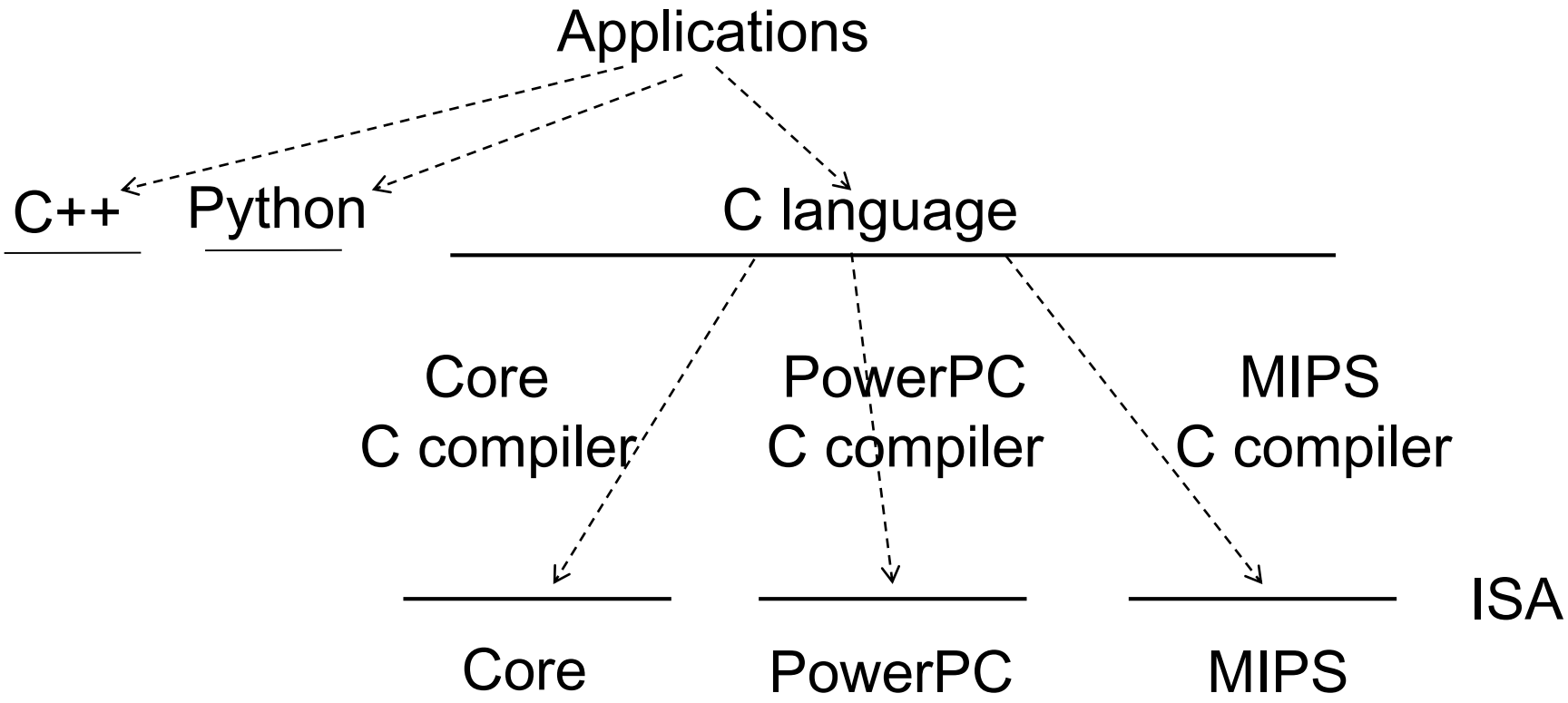
- Study of problem-solving with computational devices
- What kinds of problems did we solve?
  - How can we boost productivity in programming?
    - High-level programming languages
  - How can we make the machine easier to use?
    - OS (운영체제; collection of many algorithms)

Platform Dependency (독점성)

- Program execution environments
  - Commonly ISA plus OS API (e.g., Wintel platform)

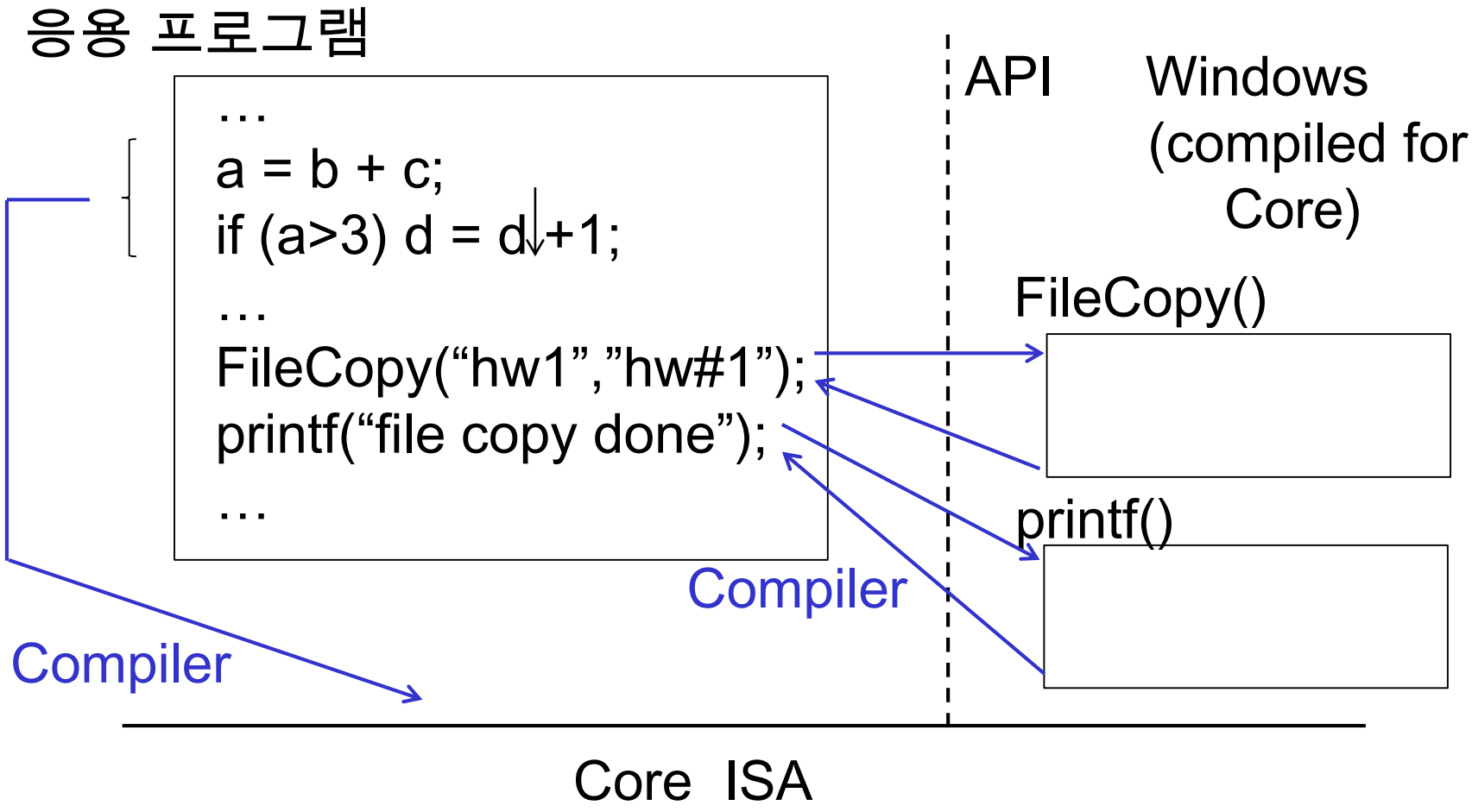


ISA (Processor) Dependency



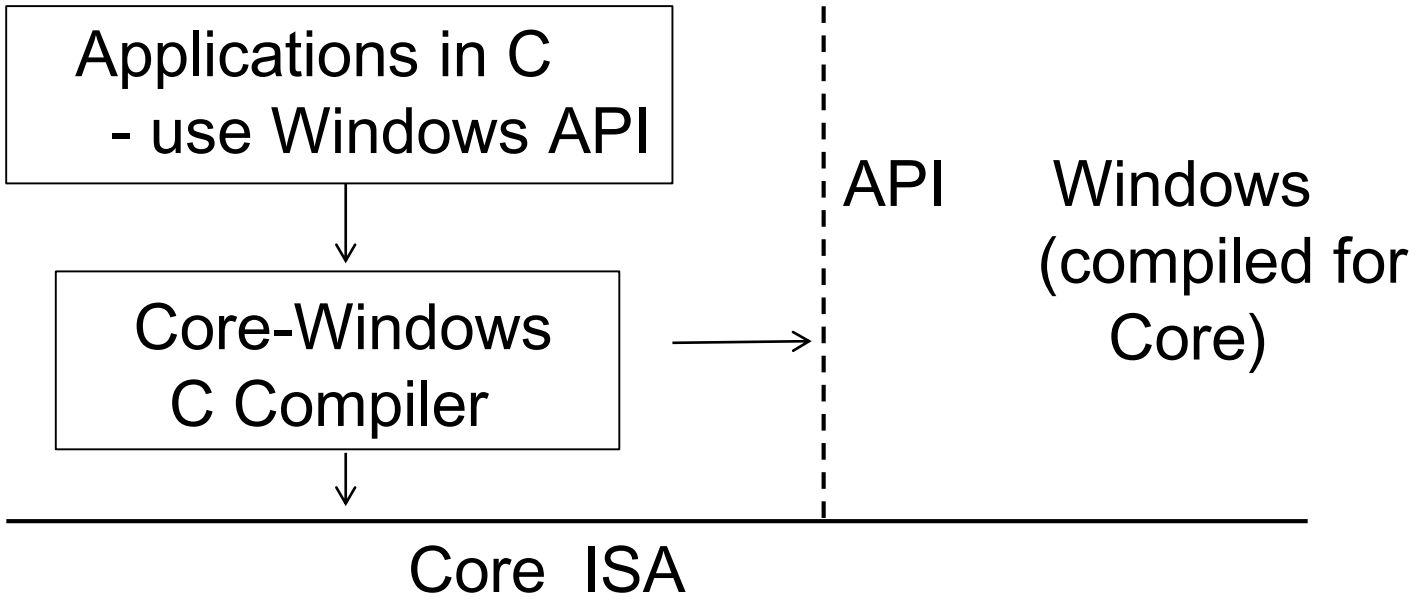
- You buy compiled code (e.g., Word for Core)
- When upgrading your PC, stuck with Core (독점성)

ISA and OS Dependency



- Buy code compiled for Core and Windows
  - Stuck with Wintel platform

ISA and OS Dependency



- Windows programming (or Linux programming)
  - Determine ISA and OS (and language) first
- All applications: Processor and OS dependent (독점성)

Platform Wars

- Desktops: Windows and Intel (Apple)
- Smartphones: Apple, Android
- Servers: Linux

What is CSE? (반복)

- 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

The Name Computer Science

(Computer Science and Engineering)

Why the name Computer Science?

(반복)

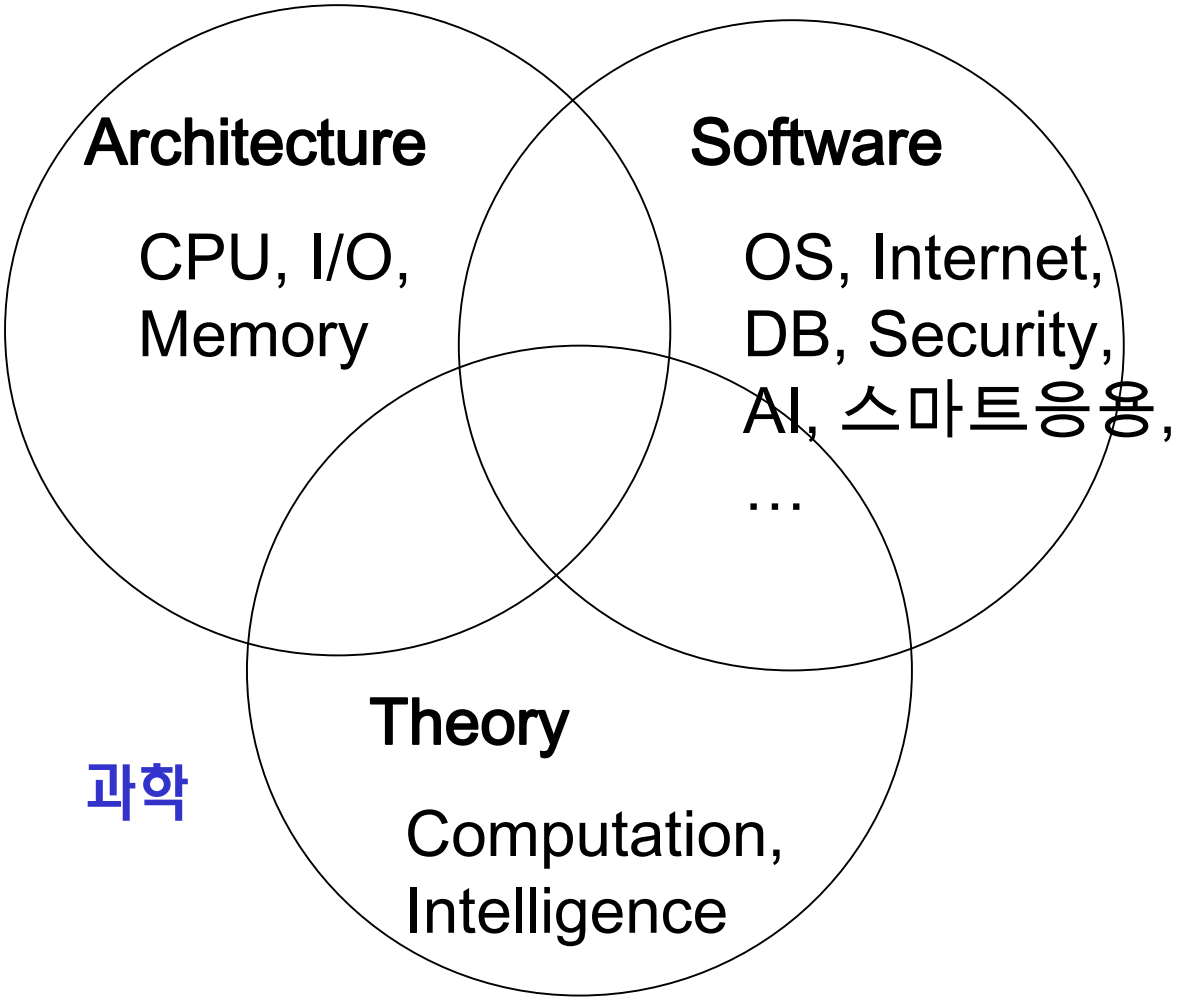
- Strong science flavor (과학, 새로운 지식을 추구)
  - Theory of computation (since early 20C)
  - Artificial intelligence (with invention of ENIAC)

- Application of computers

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

Computer Science and Engineering (반복)

IT (Information Technology; 공학)



과학

- ICT (Information and Communication Technology)

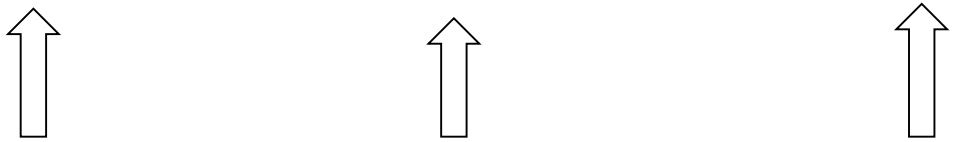
Engineering (IT) Big Picture (반복)

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

How did it change our lives?

(정보혁명, 소프트웨어융합)

Evolutions Turning into Infrastructure

- 1950 – 1970
  - Built many "big" computers (IT gold rush in USA)
- 1970s
  - Minicomputers, personal computers (Silicon Valley)
- 1990s
  - Internet, web, electronic commerce
- 2000s
  - Smartphones (mobile commerce)
- 2010s
  - What's next?

† Imagine all software developed along the way

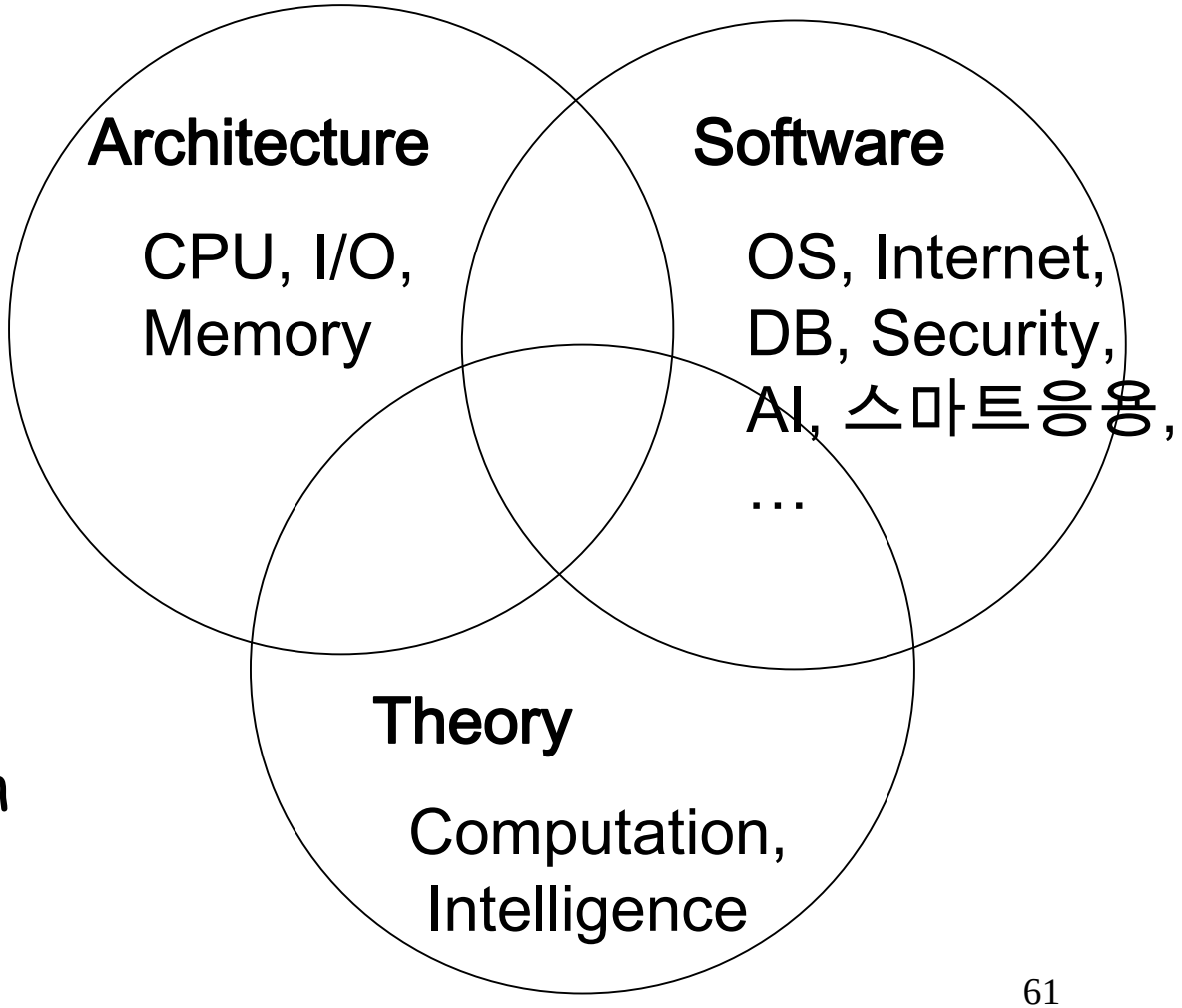
- How they changed our living, and USA dominance

Information Revolution (정보혁명)

- Social perspective (정보의 유통)
  - Control of information, propagation speed
- Electronic commerce (상품과 서비스의 유통)
  - Eliminate distance; must be globally competitive
  - Only one economy and only one market
- Automation and new business models (직업군의 변화)
  - Knowledge is the leading factor in production
- Part of Scientific/Technical Revolution since 17C

Software Convergence (반복)

- CSE



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

소프트웨어 융합 (반복)

- 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

Why Software Is Eating The World

By Marc Andreessen

The Wall Street Journal, Aug. 20, 2011

Trend

- 점점 더 많은 기업이 소프트웨어 기반으로 돌아가고 고객에게 온라인 형태의 서비스 제공
- Silicon-Valley 스타일의 기술회사
  - 분야별로 기존의 전통 기업으로 부터 주도권을 빼앗아 감
- Software companies
  - Amazon: 서점, 소매점/백화점 대체
  - Netflix: 비디어 대여점 대체
  - Apple iTunes: 전통 음원 회사는 뒷전
  - Zynga: videogame maker, 전통 오락회사 잠식

Software Eating the World

- Software companies (계속)
  - Pixar: 애니메이션 영화
  - Shutterfly (사진): 온라인 archiving/sharing, Kodak(?)
  - Google: 온라인 직접 광고 시장
  - Skype (telecom)
  - LinkedIn (astest growing recruiting company)
  - Automobile (예, Benz: 연구개발 비용의 90%가 SW로)
  - Wal-Mart (재고관리 및 분배)
  - FedEx (트럭 및 비행기의 운용)
  - Airlines (표 가격 책정 및 비행 경로 설계)

Software Eating the World

- Software companies (계속)
  - Financial services: done in software (FinTech, PayPal)
  - National defense
    - Robot soldier, SW powered drone, cyber war, intelligence gathering
  - Healthcare and education
    - Next up for software-based transformation
  -
  -

Why is this happening now?

- 소프트웨어로 글로벌 온라인 서비스 제공하기 위한 기반기술 완성
  - 60 years since invention of computers
    - Problem solving by programming
  - 20 years since modern Internet
  - 5 years since smartphone ("mobile")
- Fully digitally-wired online global economy
  - 수십억의 잠재 고객 (smartphones, broadband Internet)
- Amazon 클라우드 이용한 기본 사업 비용: 200만원/월

† Cloud computing: 전기/수도와 같이 사용량에 따른 비용 지불

Future

- Creative destruction will continue
- USA dominance
  - Great research universities (우수 인력)
  - Pro-risk business culture (벤처 정신)
  - Deep pools of innovation-seeking equity capital (자본)
  - Reliable business and contract law (제도와 시스템)
- 기회의 양극화
  - Every promising software company starved for talents
  - Many people lack the education and skills required to participate in software revolution

소프트웨어 융합

- 누가 주도할 것인가?
  - CS 전공자 또는 각 분야 전문가?
  - 문제해결능력을 갖춘 SW 인재
- 한국: 2018년 부터 초중등 교육에 필수 프로그래밍 교육
- 한양대학교: 모든 학부생에게 기초 소프트웨어 교육 실시

How did it change our lives?

Blown to Bits

(Your Life, liberty, and Happiness after the Digital Explosion)

Hal Abelson, Ken Ledeen, Harry Lewis

Privacy Lost, Abandoned

- George Orwell's 1984
  - "BIG BROTHER IS WATCHING YOU"
- "Wherever we go, we leave digital footprints"
  - Surveillance cameras, cell phone companies, GPS, credit card companies, banks, web clicks, Emails, SNS, automobile black boxes, …
- 저장된 디지털 정보를 이용하여 사람의 인생을 재구성 가능
- 소비자는 어떤 IT 장치가 유용해지면, 이 장치가 나를 감시하는데 사용될 수 있음을 무시함
  - 그런 IT 장치 늘어남에 따라, 프라이버시의 벽이 허물어짐

Privacy Lost, Abandoned

- 효율, 편리성, 경제성 등의 이유로 프라이버시 포기
  - 모르거나, 생각 않거나, 어쩔 수 없다고 포기함
- 또는 적극적으로 개인의 사생활 노출을 사업화
- Who is Big Brother?
  - Government protecting us
  - Corporations
- 정보 민주주의 또는 정보 과두체체?
- 웹과 검색 엔진: 사이버 세상을 보여 주는 렌즈
  - Can hide as well as reveal

How will it change our lives?

(What's next?)

새로운 발명이 기반구조로 정착 (반복)

- 1950 – 1970: built many "big" computers
- 1970s: 미니컴퓨터, 퍼스널 컴퓨터 (Silicon Valley)
- 1990s: 인터넷, 웹, 전자상거래
- 2000s: 스마트폰 (mobile)
- 2010s: What's next?
  - 소프트웨어 융합 또는 4차 산업혁명 (IoT, AI)

Internet of Things (사물인터넷)

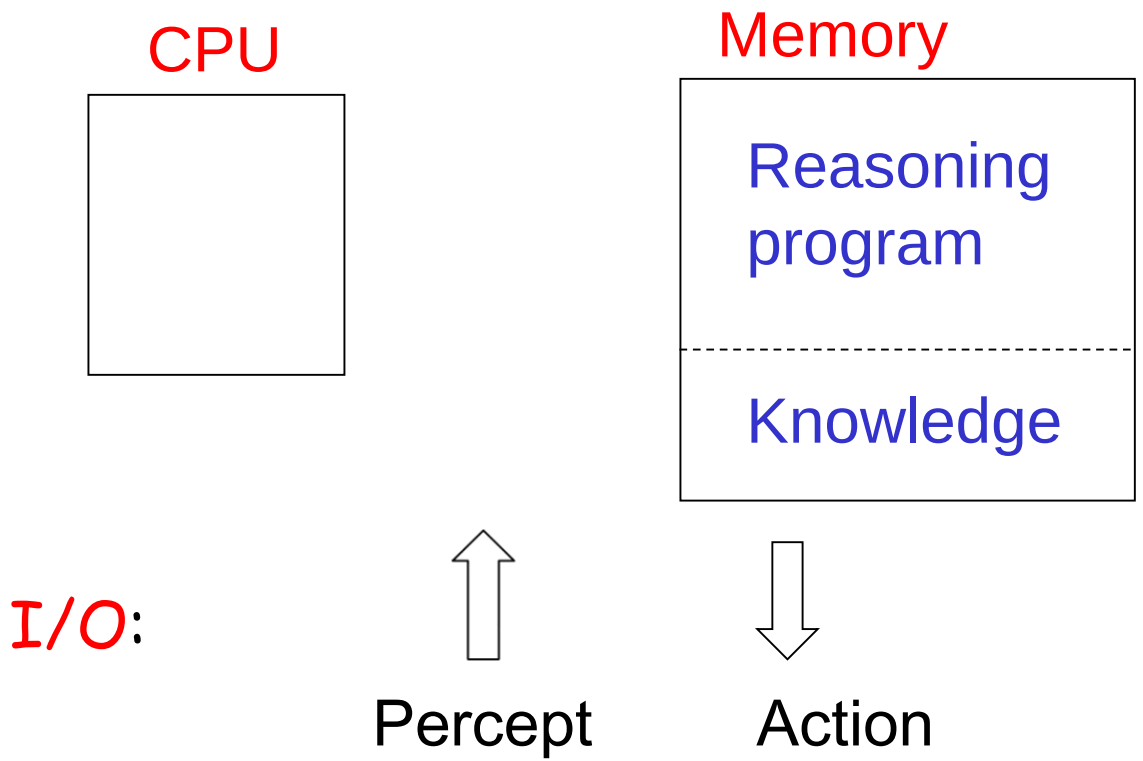
- 전통적 유무선 인터넷: 사람과 사람을 연결
- Things
  - CCTV, 자동차, 스마트워치, 가전, 공장 기계, 로봇, …
  - 스마트 임베디드 시스템: 기계 + 센서/액터 + 컴퓨터/SW + 인터넷
- SW 융합: 기계-기계 협업, 기계-사람 협업
  - 스마트한 운송, 가전, 홈, 건강관리, 공장, 시티, 재난관리, 에너지관리
- 미래 정보사회를 위한 인프라
  - 인간 사회 전 분야의 자동화

용어 주의: 4차 산업혁명과 인공지능

- 4차 산업혁명 (?)
  - 로봇, AI, IoT, 자율운송, 3D 인쇄, 나노, 생명, …
    - 1차 산업혁명: 증기 엔진, 대량생산
    - 2차 산업혁명: 전기 응용 시스템
    - 3차 선업혁명: 컴퓨터 및 인터넷
- 인공지능: strong AI vs. weak AI
  - 인공지능(?)이 우리의 직업을 앗아 간다
    - 소프트웨어에 의한 자동화

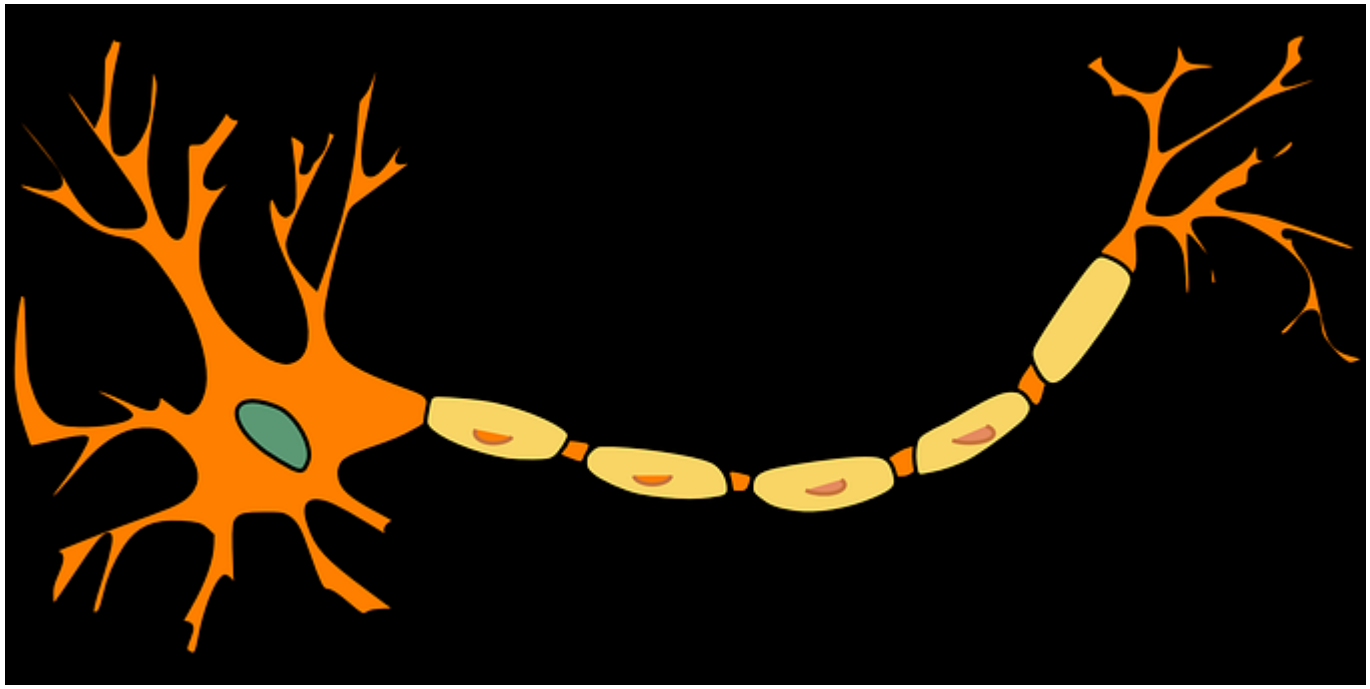
AI: Knowledge Based Approach(반복)

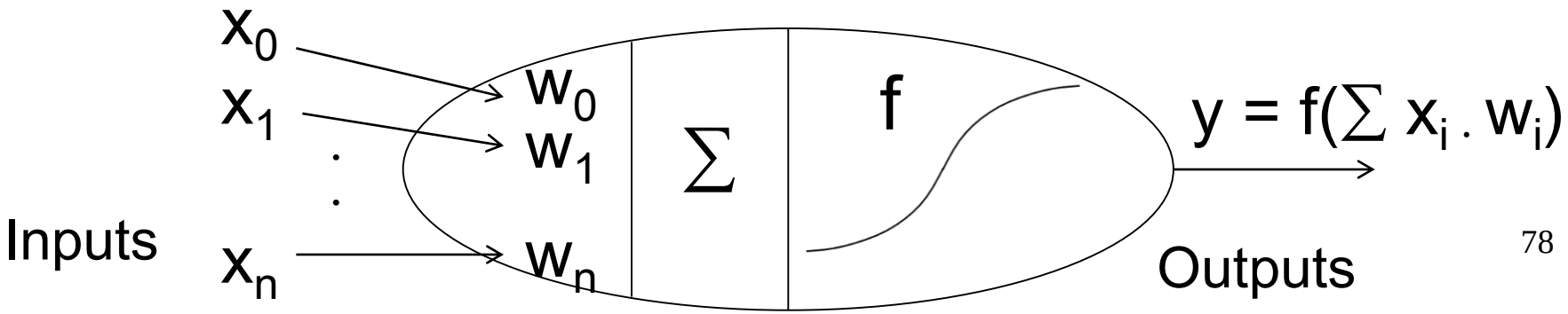
- Given computers, can we develop intelligent machines?
- Knowledge representation and reasoning (bio-inspired)



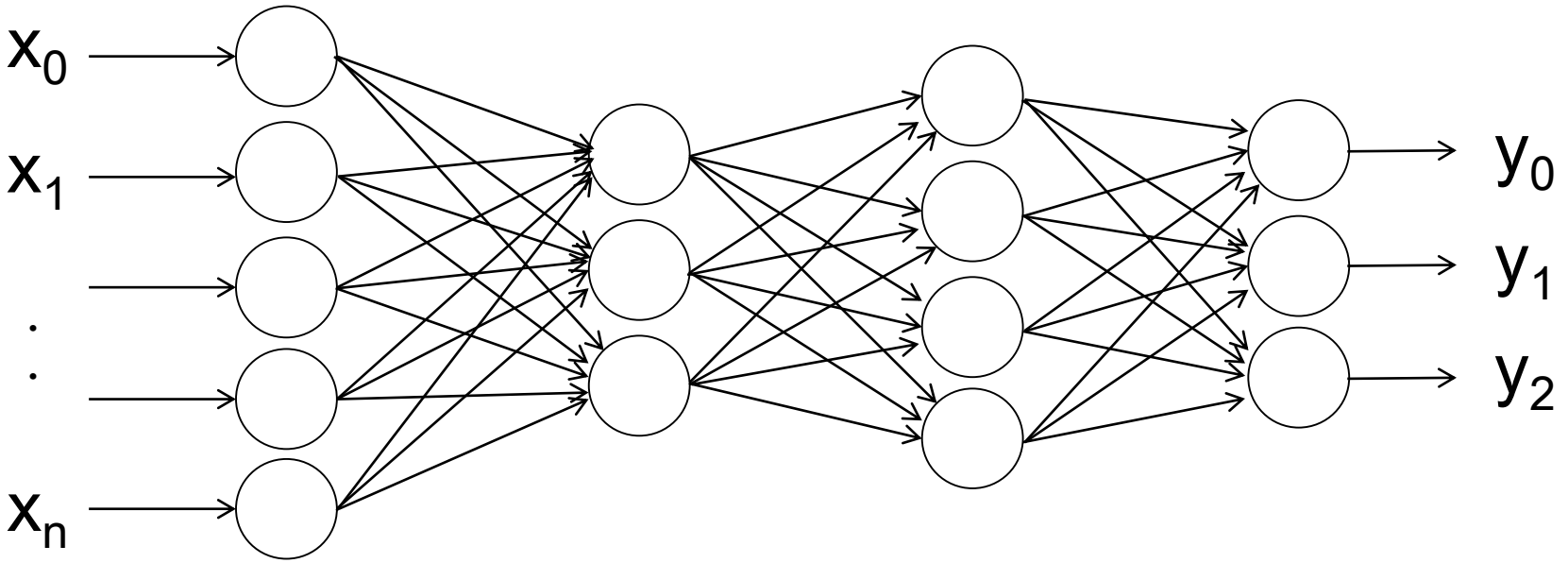
AI: Neural Networks

- 10^{11} neurons in human brain; 7,000 synapses on average





AI: Deep Learning



- 축적된 "big" 데이터로 부터 유용한 알고리즘을 추출: 영상/언어 처리
- 뉴럴 네트워크를 이용하는 머신 러닝의 한 방법
  - 뇌 신경망의 동작을 모사 (bio-inspired)

Artificial Intelligence (AI)

- AI as engineering techniques (for smart applications)
  - Knowledge-based approach
  - Deep learning (cf. machine learning and data science)
  - Planning
  - Uncertain knowledge reasoning
  - Search
- AI as science
  - Open questions: intelligence? thinking? consciousness?

† Knowledge-based and deep learning started as science

Classes of Computers

(Large-Scale Servers)

Two Types of Computers

- General-purpose computers (범용컴퓨터)
  - 다양한 종류의 프로그램을 실행함 (PC, 기업 서버)
- Embedded systems (내장형시스템)
  - Machines 과 결합하여 스마트한 자동형 기계 형성
    - 항공기, 우주선, 자동차, 청소기, drone, 로봇, ...
  - Many different types, so many of them
    - 프로그램은 HW에 특화되어 한 가지로 고정
  - 컴퓨터는 머리 역할하며 기계에 안에 내장됨

† Special-purpose computers, dedicated computers

Classes of Computers

- Servers
  - Large workload: single complex application or many small jobs (supercomputers, web servers)
  - Software from another source, but customized
- PCs (or desktops)
  - Good performance for single user at low cost
  - Third-party software
- Embedded computers
  - Fixed applications integrated into a single system
  - Widest range, not seen as computers, cost and power

The Smartphone Era

- Personal mobile devices (PMDs)
  - Smartphones or tablets (maybe wearables)
  - Battery operated, wireless connectivity to Internet
  - Downloaded software ("apps")
  - Touch-sensitive screen or even speech input
- Servers
  - Giant data centers with 100,000 computers
  - A portion of application on PMD and a portion in the cloud (e.g., web search, SNS)

† Cloud computing, warehouse-scale servers, supercomputers

Cloud Computing

- H사 전산실의 경우: 자체 인력, 예산
  - Hardware: 중대형 서버들
  - Software: OS, database, …
  - 자체 응용 소프트웨어 유지 보수
- IBM: 우리한테 외주 주시오
  - HW only, HW + SW, or everything
  - 인터넷 저편 어디의 IBM 서비스를 web 을 통해 사용
    - Will my data be safe?
- 비슷한 예: web hard, email service, …

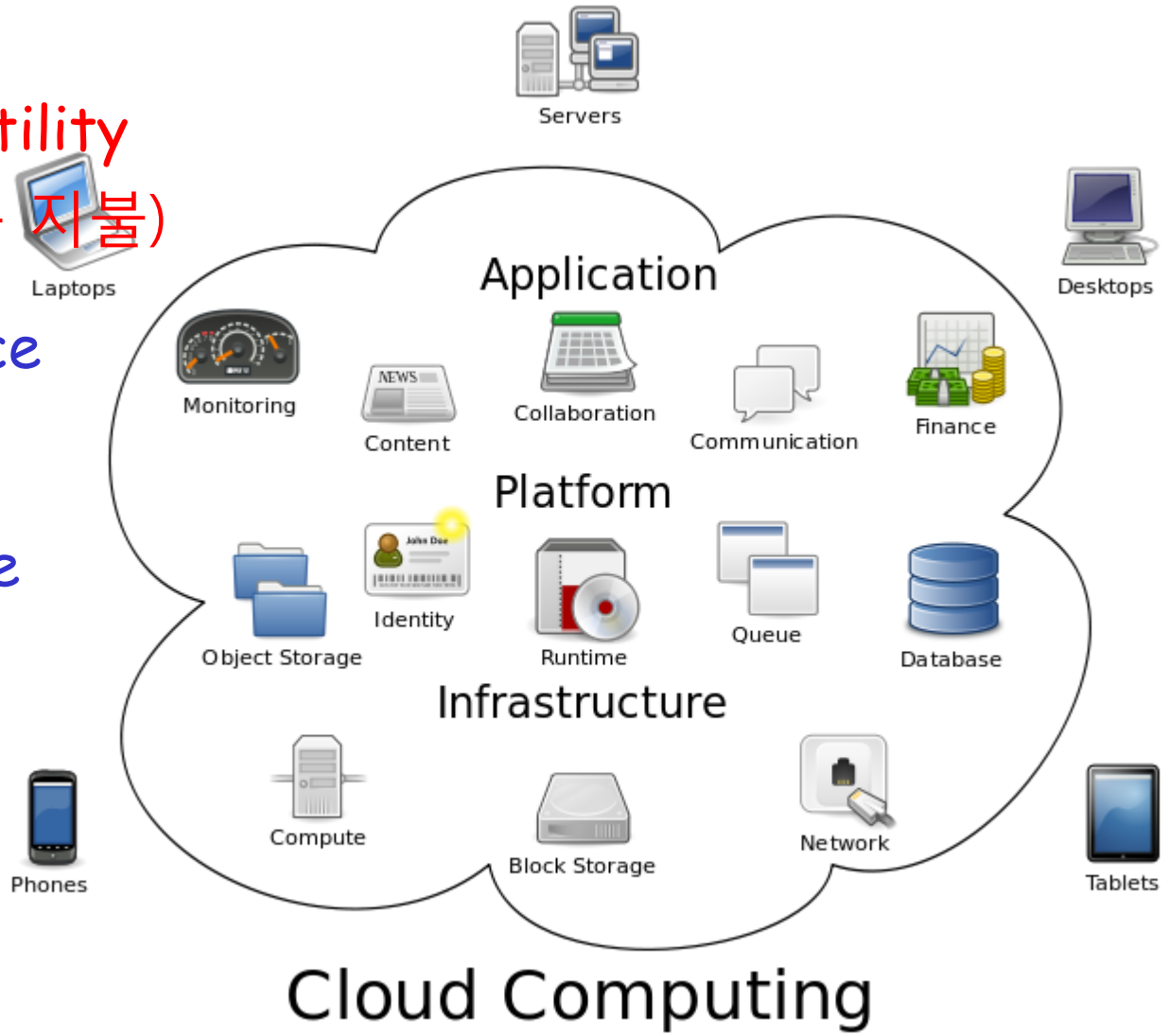
Cloud Computing

- Cloud computing:
  computers as utility
  (사용량에 따른 비용 지불)

Software as a Service (SaaS)

Platform as a Service (PaaS)

Infrastructure as a Service (IaaS)



https://en.wikipedia.org/wiki/Cloud_computing#/media/File:Cloud_computing.svg
Created by Sam Johnston; no changes were made; CC BY-SA 3.0

Large Data Centers

- Computing resources delivered as service over Internet
  - Used by millions of users
- Warehouse-scale computers (economy of scale)
  - Giant data centers with 100,000 computers
  - Space, cooling, networking, storage
- Physical design standard: rack mount computers
  - 19" wide
  - 1.75" high – rack unit or unit (U)
  - Most popular: 42 U high
- Standard container filled with racks and interconnection

19-inch Rack with 42 1U Servers

(Source: Computer Organization and Design, Hennessy and Patterson)

Sun Fire x4150 1U Server

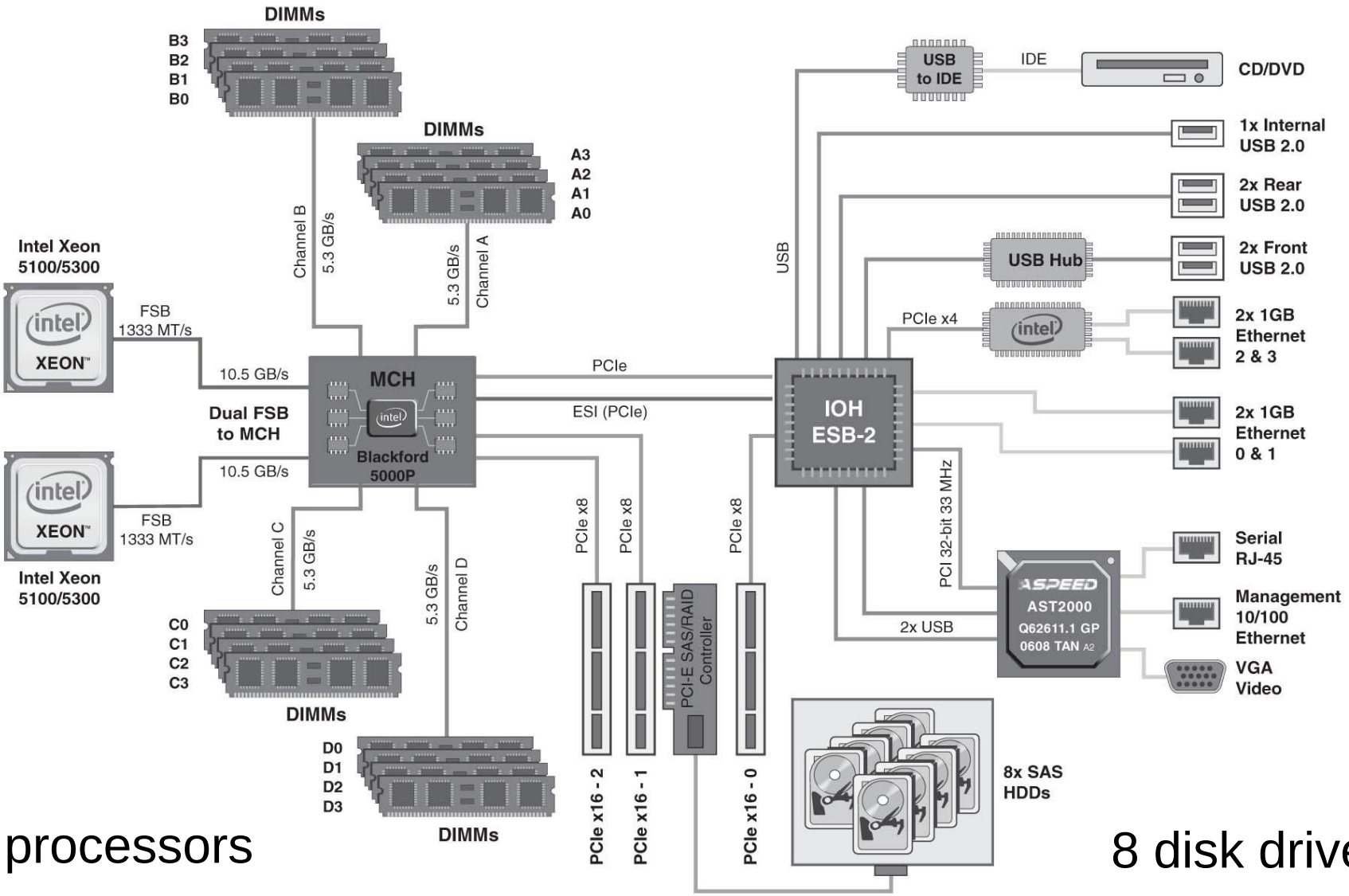
(Source: Computer Organization and Design, Hennessy and Patterson)

Front

Rear

Inside of Sun Fire x4150

(Source: Computer Organization and Design, Hennessy and Patterson)



2 processors

8 disk drives

Container Data Centers

- Power supply, networking, cooling

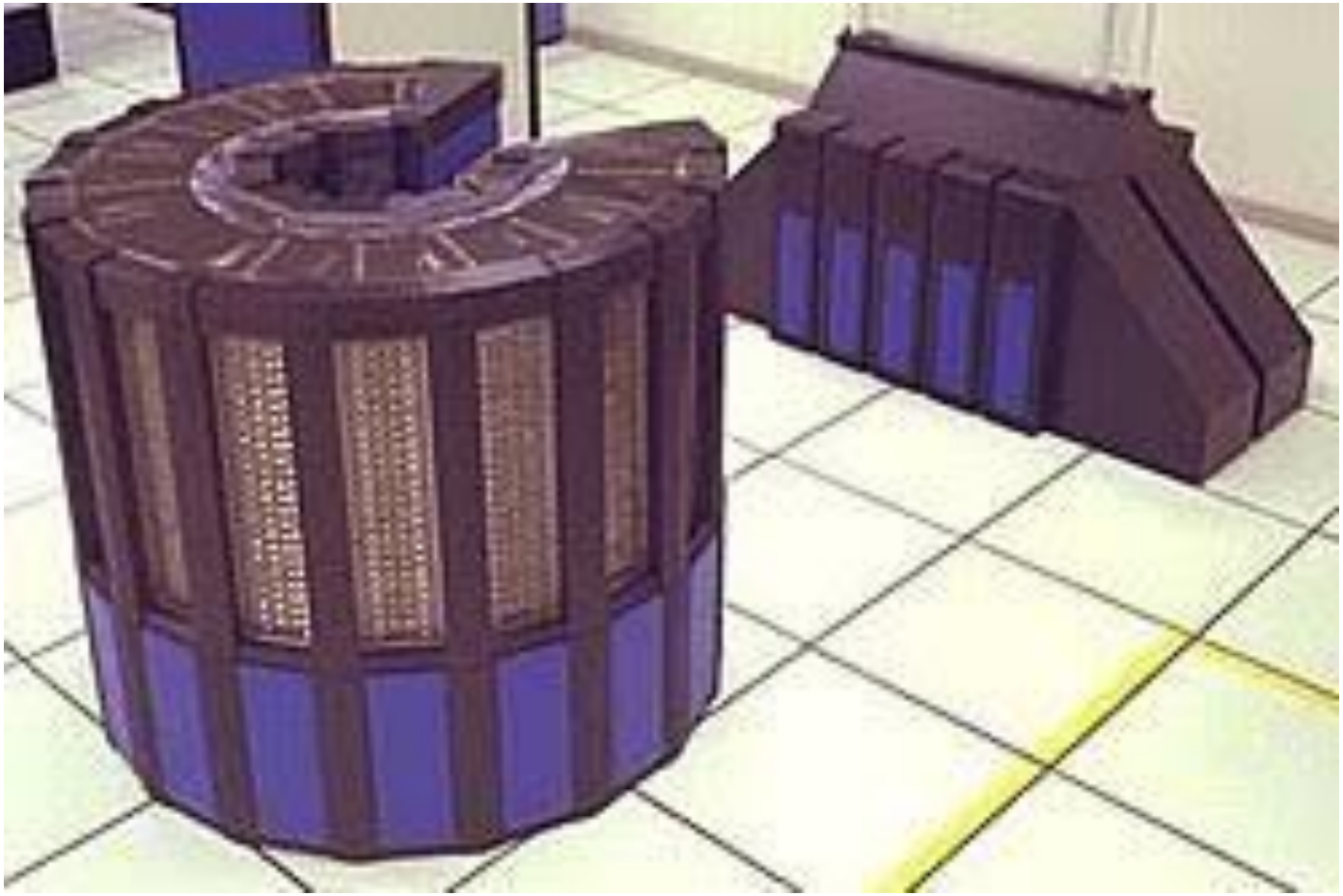
Large-Scale Data Centers vs. Supercomputers

Supercomputers

- Desire for highest performance possible
  - IBM, NEC, Fujitsu, Europe
  - US startups in late 1980s
    - Cray, Thinking machines, Kendal Square Research
- Financial difficulty at the end of cold war
- Demand for supercomputers strong and growing in commercial applications
  - Commercial aircraft and automobile design
  - Chemical engineering, weather forecasting, and so on
  - Industry must deal with cost/performance

Supercomputers

- Cray in 1980s
  - Fastest hardware with best existing technologies

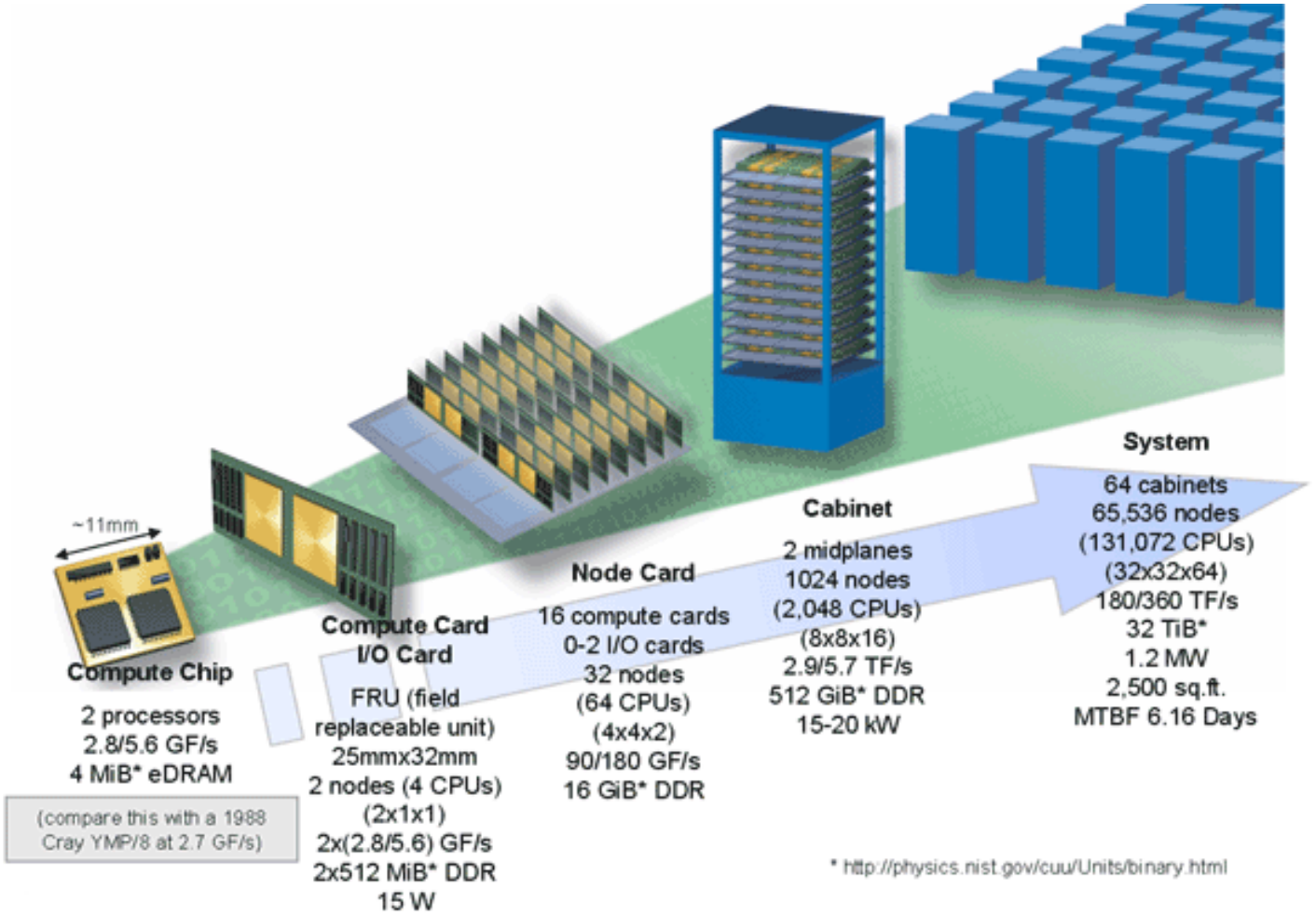


Supercomputers

- 2012 fastest supercomputer: IBM Sequoia Blue Gene/Q
  - 1,572,864 processor cores, 1.6 petaFLOPS
  - 91 refrigerator-sized server racks
- Why not possible with Cray style of design?

Supercomputers

- LLNL Blue Gene L installation
  - Massively-parallel computer (like protein folding)



Summary

- Machines called computers
  - Basic computer organization
  - Operational principle: fetch-decode-execute
  - Service provided by processor (or computer)
    - Instruction Set Architecture
  - Completion of modern digital computers in 1945
- Computer Science and Engineering
  - How IT has changed our lives
- Classes of computers
  - Large-scale data centers and supercomputers

Homework #3 (see Class Homepage)

1) Write a summary report about the materials discussed in Topics 0-3 and 0-4 (at least 5 pages of detailed report)

   - 문장으로 써도 좋고 파워포인트 형태의 개조식 정리도 좋음

❑ Submit electronically to Blackboard

** also, read chapter 1 of textbook

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

Class Topics (클래스 홈페이지 참조)

- Part 1: Fundamental concepts and principles
  1) Invention of computers and digital logic design
  2) Abstractions to deal with complexity
  3) Data (versus code)
  4) Machines called computers
  5) Underlying technology and evolution since 1945
- Part 2: 빠른 컴퓨터를 위한 설계 (ISA design)
- Part 3: 빠른 컴퓨터를 위한 구현 (pipelining, cache)