Computer Architecture (ENE1004)

▼ Lec 1

Lec 1: Computer Abstractions and Technology (컴퓨터 추상화 및 기술)

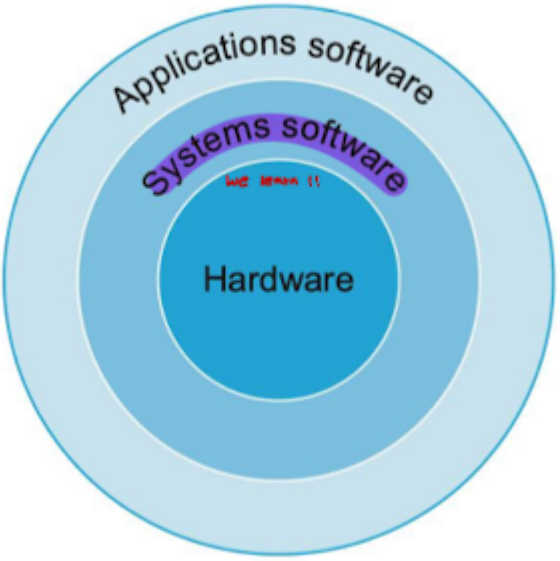
What You Can Learn ?

- By the time we end this course, you should be able to answer:
  - How are programs written in a high-level languages translated into the language of the hardware? (high-level 언어는 어떻게 하드웨어 언어로 변환 되는가?) How does the hardware execute the resulting program? (하드웨어는 결과 프로그램을 어떻게 실행하는가?)
    - 코드를 CPU(하드웨어)에 맞는 형태로 맞게 바꿔줘야함. → 이것이 compile임.
    - 우리는 Mips를 배움. 이것은 하드웨어 언어이고 Arm 언어와 유사하고 굉장히 쉬운것임.
  - What is the interface between the software and the hardware? How does software instruct the hardware to perform needed functions? (소프트웨어와 하드웨어 사이의 인터페이스는 무엇인가? 소프트웨어는 필요한 기능을 수행하도록 하드웨어에 어떻게 지시하는가?)
    - interface는 운영체제임. 이것이 없으면 hardware를 돌릴 수 없음. 이것에 대해서도 배울것.
  - What determines the performance of a program? How can a programmer improve the performance? (무엇이 프로그램의 성능을 결정하는가? 어떻게 프로그래머는 성능을 향상시키는가?)
    - 프로그램 성능은 프로그램 자체가 느리거나 하드웨어가 느리거나 둘의 조합이 안맞을 수 있음. 하드웨어를 이해해 이 성능을 키워볼 것.
- You can improve the performance of your program if you are aware of …

Our interest

Hardware or software component	How this component affects performance	Where is this topic covered?
Algorithm	Determines both the number of source-level statements and the number of I/O operations executed	Other books!
Programming language, compiler, and architecture	Determines the number of computer instructions for each source-level statement	Chapters 2 and 3
Processor and memory system	Determines how fast instructions can be executed	Chapters 4, 5, and 6
I/O system (hardware and operating system)	Determines how fast I/O operations may be executed	Chapters 4, 5, and 6

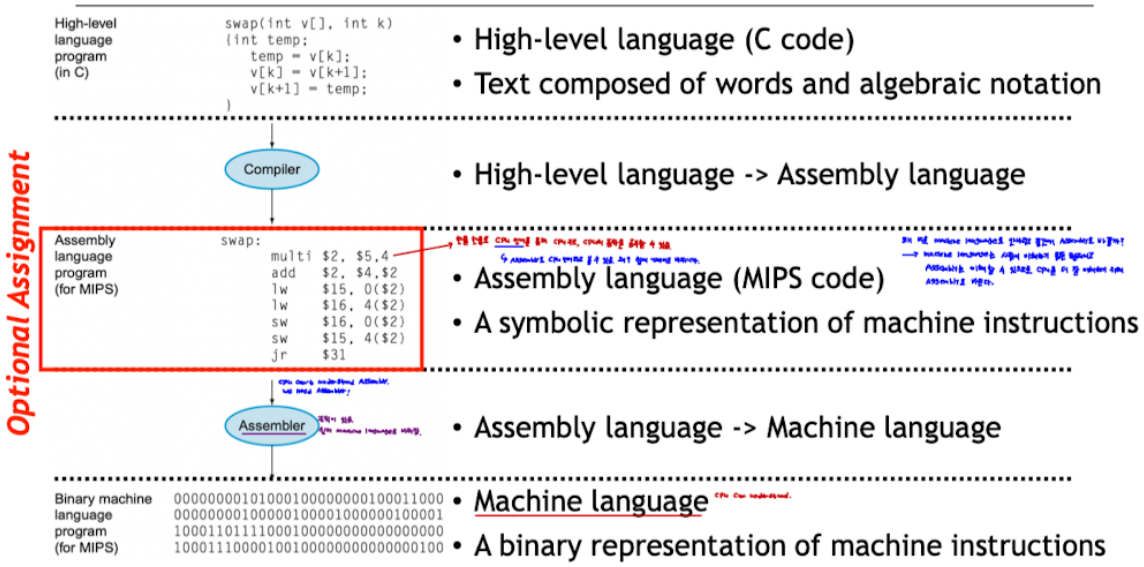
Review: Layers of Software



- System software helps your application software to run on hardware
- Two kinds of key system software
- (1) Operation system (실행 중 도움)
  - Allocating memory and storage (ex: 32바이트에 12바이트를 할당한다면 어디에 할당해야할까? → 운영체제가 결정함.)
  - Handling input/output operations
  - Coordinating concurrent applications
- (2) Compiler (실행 전 (언어 변경) 도움)
  - Translating a program written in a high-level language to instructions that the hardware can execute

- Your program is based on a high-level language: c = a + b;
- Hardware works based on binary values: 100111010

Review: from High-level Language to Machine Language



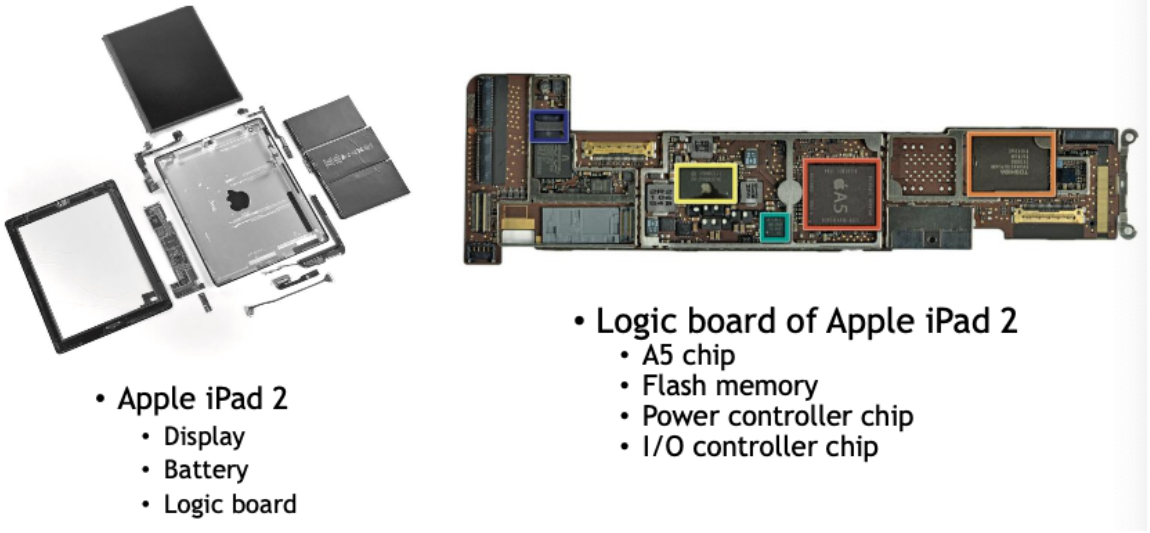
- Assembly language
  - CPU 언어의 한줄 한줄을 통해서 CPU 구조, CPU의 동작을 공부할 수 있음. (Assembly는 쉽게 기계어로 바뀌므로 CPU 언어라고 볼 수 있음.)
  - 왜 바로 Machine language로 안바꾸고 중간에 Assembly language로 바꿀까?
    - Machine language는 사람이 이해하기 힘든 형태이지만, Assembly language는 이해할 수 있으므로 CPU를 더 잘 이해하기 위해 중간에 Assembly language로 바꿈.
  - 하지만 CPU는 Assembly language는 이해할 수 없으므로 Assembler을 통해서 Machine language로 바꿔야한다.
  - Assembler에는 Assembly language를 쉽게 Machine language로 바꿔주는 규칙이 존재함.

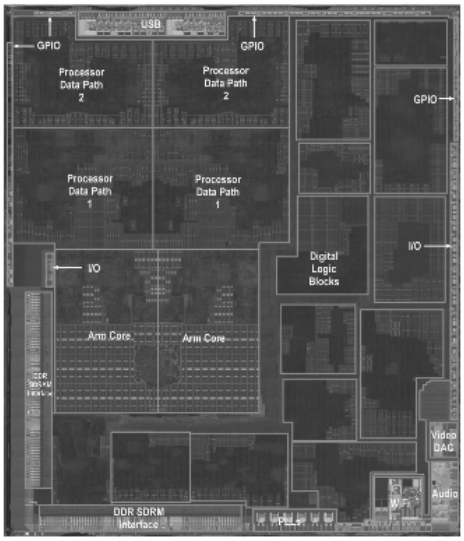
Review: from High-Level Language to Machine Language

- Regarding to the conversion, there are many terms.
  - Compiling, linking, buliding, ... ; inclusion relationship, used interchangeably

    - Some compilers cut out the intermediate process and produce binary directly
- Why is the focus of curriculum on high-level language?
    - High-level languages can offer several important benefits
        - It allows programmers to think in a more natural language, using English words and algebraic notation, resulting in programs that look much more like text
        - It improves programmer productivity; it takes less time to develop programs
        - It allows programs to be independent of the computer on which they were developed
        - PLDI (https://pldi23.sigplan.org/track/pldi-2023-pldi)
    - Enhanced compilers can produce very efficient assembly code (machine cod)
      optimized for the target machine
        - Compiler research has a long history
        - CGO (https://conf.researchr.org/home/cgo-2024)

Example: Device Components





- Inside the A5 chip
  - Two ARM processors
  - DRAM interfaces, and others
  - Integrated circuit, chip, package
- Processor (central processor unit, CPU)
  - (1) Datapath: a collection of functional units for processing data
  - (2) Control: tells each functional unit what to do
  - Chapter 4 will cover the two components in detail
- Memory
  - SRAM within chip
  - DRAM outside of chip

- Storage
  - Non-volatile: it keeps data after power loss
  - Hard disk drives (HDDs) or Solid state drives (SSDs) based on flash memory
  - Slow, low cost per bit
- Main memory
  - Volatile: data is lost when power is removed
  - Dynamic random access memory (DRAM)
  - Faster than storage, high cost per bit
- Cache memory (built from SRAM)
  - Volatile as well
  - Static random access memory (SRAM)
  - Faster than DRAM, more expensive than DRAM
- Chapter 5 will cover the memory systems in detail

Review: Two performance Metrics

- We say one computer has better performance than another; what do we mean?
  - Performance of airplanes (cruising speed vs passenger 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

- Response time (execution time) - s, ms, us, ns, ...

    - The time between the start and the completion of a task
    - Important to individual users
    - Used in evaluating the performance of embedded and laptop computers
- Throughput: 처리량 (bandwidth: 대역폭) – MB/s, GB/s, ...
    - The total amount of work done in a given time
    - Important to datacenter managers
    - Used in evaluating the performance of servers
- Improving (decreasing) response time usually improves (increases) throughput

Review: Measuring Execution Time

- Relative Performance
    - If computer A runs a program in 10 seconds and computer B runs the same program in 15 seconds, how much faster is A than B?
    - is *n* times faster than B if execution_timeB/execution_timeA = *n*
    - 15/10 = 1.5; so, A is 1.5 times faster than B
- Two types of execution time
- (1) Elapsed time
    - The total time to complete a task
    - It includes disk accesses, memory accesses, I/O activities, operating system overhead
- (2) CPU time
    - The time the CPU spends computing for this task
    - It does not include time spent for other activities
    - User CPU time + system CPU time