**1.** **Computer systems consist of CPU, Memories, HDD & IO Devices + Bus**

컴퓨터 시스템은 3가지, 혹은 더 자세하게는 5가지의 부분으로 이루어져 있다.

3가지로 표현을 하자면 Internal Device에 해당하는 CPU와 메모리, 그리고 그 외의 External Devices에 해당하는 디스크로 이루어져 있다.

5가지로 표현을 하자면 CPU, 메모리, 그리고 Network interfaces와 Character devices와 Block devices로 이루어져 있다.

본 강의에서는 External devices의 HDD에 가장 중점을 둘 것이기에, 최종적으로 컴퓨터 시스템은 CPU, 메모리, 하드디스크와 입출력 디바이스, 그리고 버스로 이루어져 있다.

**2.** **CPU & MM are the only “internal” devices**

CPU와 메모리만이 internal 디바이스에 해당한다. 이 두 개 외의 것은 모두 external devices이다.

**3.** **All others including HDD are “external” devices**

2번의 설명과 이어진다. CPU와 메모리만이 internal 디바이스이니, 이외의 것에 해당하는 HDD 또한 external devices이다.

**4.** **Storages are classified into two categories**

**4-1. “Internal storages” such as registers, caches, main memory are volatile.**

CPU 안에 존재하는 registers, 박스안에 존재하는 caches, 그리고 운영체제 강의에서 주로 다룰 메인 메모리(DRAM) 이 세 개의 장치는 컴퓨터 전원이 꺼지면 저장되었던 내용이 사라지는 휘발성의 특징을 가진다. Caches는 이번 운영체제 강의에서 공부할 범위에 해당하지 않는다.

**4-2. “External storages” such as HDD, SDD are non-volatile.**

그리고 External storages에 해당하는 HDD(운영체제 과목에서 주로 다룰 장치), SDD는 컴퓨터 전원이 꺼지더라도 저장되었던 내용이 사라지지 않는 비 휘발성의 특징을 가진다.

**5. All files are assumed to be stored in HDD**

4-2번의 설명과 이어진다. 운영체제 강의에서는 모든 파일이 하드디스크에 저장되어 있는 것으로 간주한다. 특별한 추가 설명이 있지 않는 한, SSD는 간주하지 않는다.

**6. All programs must be “loaded” from HDD into MM to be executed by CPU**

**(Load: Data movement upward / Store: Data movement downward)**

사실 6번의 문장에서 “by CPU”는 생략해도 Fact 적인 부분에서 아무런 문제가 없다. Registers는 애초에 CPU 안에 존재하기에, 이는 곧 CPU가 모든 프로그램을 실행한다는 의미가 된다.

메모리를 판단하는 기준에는 두 가지가 있는데 하나는 속도(Time Complexing)이고 나머지 하나는 크기(Size)이다. 여기서 주목할 점은 속도인데 CPU와 main memory(DRAM), 그리고 HDD의 데이터 전송 속도를 따져보면 각각 36Hz(3e+9Hz), 50~70nsec(1e+7Hz), 10msec(10Hz)에 해당한다.

속도가 가장 느린 HDD에서 main memory까지 Load가 이루어지며 최종적으로 CPU를 통해 프로그램이 실행이 되는 것이다.

**7. We assume the MM is Illinois Architecture (= von Neumann Architecture)**

모든 프로그램은 instruction과 데이터로 구성이 되어있다. 이 프로그램을 처리하는 방식에 있어서 두 가지 형태의 Machine이 존재한다.

첫 번째는 “von Neumann Machine”으로, 폰 노이만 구조라고도 불린다. 내장 메모리 순차처리 방식으로서, 그림과 같이 데이터 메모리와 프로그램 메모리(Code)가 구분되어 있지 않고 하나의 버스를 가져서 CPU에게 Data를 전송하고 Address를 받는 구조를 말한다. 우리가 사용하는 대부분의 컴퓨터는 이 방식을 따르고 있고, OS에서도 이 방식을 사용한다.

두 번째는 “Harvard Machine”으로, 명령과 데이터에 대한 별도의 저장 및 신호 경로가 존재하는 컴퓨터 아키텍처이다. von Neumann Machine를 사용하는 시스템에서는 명령과 데이터가 동일한 메모리가 저장되기 때문에, 데이터를 가져올 때 사용되는 동일한 경로를 통해 명령을 가져온다. 이는 곧 CPU가 동시에 명령어를 읽고 메모리에서 데이터를 쓰거나 읽는 것이 불가능하다는 걸 의미한다. 그러나 Harvard Machine를 사용하는 시스템에서는 CPU가 명령을 읽고 동시에 메모리 액세스를 하는 것이 가능하다. 또한 캐시가 없이도 명령을 가져오거나 데이터에 액세스하는 작업이 단일 메모리로 경합을 하지 않기 때문에, Harvard Machine을 사용하는 시스템이 훨씬 더 빠를 수 있다. 주로 아두이노나 우리가 사용하는 핸드폰에서 이 아키텍처를 사용한다.

**8. All “internal activities” are execution of programs by CPU using Main Memory**

**8-1. That is, if there are activities inside computer system, there must be certain programs being executed by CPU using the Main Memory**

여기서 “activities” 라는 단어의 의미는 “컴퓨터가 어떠한 일을 수행하고 있다.” 라는 의미이다. 컴퓨터 시스템 내부에서 활동이 있는 경우에, 메인 메모리를 이용해서 CPU가 프로그램을 실행시킨다.

축구 경기를 예제로 들어보자. 축구 경기에서는 골키퍼와 공격수, 수비수, 그리고 심판 등이 존재한다. 각각의 선수들은 경기내에서 수행해야 할 자신만의 역할이 있고, 또 그 역할은 배정받은 본인이 가장 잘 수행한다. 심판은 골대 앞에 서서 상대방의 공을 막으면 안 되고, 또 그 일에 대한 수행능력은 골키퍼보다 훨씬 떨어질 것이다.

컴퓨터에서도 마찬가지이다. CPU 내부에서 실행되는 각각의 “특정한 프로그램”이 실행되는 과정에 있어서 역할이 각각 나뉘어져 있어야만 “컴퓨터가 어떠한 일을 잘 수행하게” 되는 것이다.

**9. There are only two types of programs**

**9-1 User programs (& system programs) & OS programs**

**9-2 We assume system programs such as browsers, compilers, editors, and so on as user programs.**

프로그램에는 두 가지의 유형이 존재한다.

하나는 User programs (& system programs)이고, 나머지 하나는 OS programs이다. User programs 가 아닌 것들은 전부 OS programs로 생각하면 된다. User programs는 앞에 있는 그림에서 Application에 해당하는 부분이다. system programs는 브라우저, 컴파일러, 편집기 등에 해당하는데 이 프로그램 또한 User programs에 해당한다고 간주한다.

**10. That is, all programs except user programs are OS programs**

9번에서 설명했듯이, 프로그램에는 두 가지의 유형이 존재하는데 User programs와 OS programs이다. 따라서 system programs가 포함된 User programs를 제외한 모든 프로그램은 OS programs로 생각하면 된다.

**11. OS performs Process Management(=Mgt), Resource Mgt & Control Programs**

**11-1. Process Managements:**

**Start to execute, run, stop (exit, pause, kill(=terminate))**

**Scheduling, Communication, Synchronization, etc.**

**11-2. Resources:**

**Physical resources: CPU, Memories, IO Devices, etc.**

**Logical resources: Time, Files, etc.**

**11-3. Program Control**

**to prevent errors and improper use of the computer**

OS는 프로세스 관리, 리소스 관리, 그리고 제어 프로그램 관리의 역할을 한다.

프로세스 관리로는 실행 시작, 실행, 중지(일시 중지를 하거나 완전히 중지시키기), 스케줄링, 통신, 그리고 동기화와 같은 역할을 수행한다.

리소스 관리로는 두 가지의 유형이 있다. 물리적 리소스 관리로는 CPU, 메모리, 그리고 I/O 장치와 같은 눈에 보이는 리소스를 관리하고, 논리적 리소스 관리로는 시간이나 파일과 같이 눈에 보이지 않는 리소스를 관리한다.

제어 프로그램의 관리로는 컴퓨터의 오류와 적절하지 않은 사용을 방지하는 역할을 수행한다.

**12. We assume single-CPU / multi-user system in our class**

**12-1. Each user may have multiple processes.**

**12-2. Therefore, many processes exist, at a time, in a system!**

본 강의에서는 CPU는 하나, 그리고 사용자는 다중 사용자의 시스템을 가정하고 공부를 할 것이다.

각 사용자들은 여러 프로세스를 수행할 수 있고, 본 강의에서는 다중 사용자의 시스템을 가정한다고 했으니 시스템에는 한 번에 많은 프로세스가 존재한다.

어머니(CPU)께서 저녁 식사를 차리는 과정을 생각해보자. 저녁 식사로 만들 음식(프로그램)으로는 밥, 된장찌개, 김치, 오이무침, 그리고 감자조림이 있다. 각각의 음식(프로그램)을 만들기 위해서는 재료를 비롯한 조금의 노동력, 그리고 더욱 디테일한 부분(프로세스)들이 있을 것이다. 이와 동일하게, CPU가 어떠한 프로그램을 동작하기 위해서는 그 프로그램을 동작하는데 있어서 필요한 프로세스를 실행할 것이다. 이러한 과정을 통해 여러 개의 프로세스는 CPU를 공유한다.

**13. Computer system shared by processes must keep all users & programs happy**

**13-1. Happiness is from user convenience, fastness.**

**13-2. For this, OS tries for fair & efficient HW resource management.**

프로세스에 의해 공유되는 컴퓨터 시스템은 최대한 효율적으로 동작해서 모든 사용자와 프로그램을 “행복하게” 만들어야 한다. User에게는 최대한 편리하고, 신속함을 제공해야 하고 device와 같은 관리자에겐 효율적인 환경을 만들어줘야 한다. CPU는 한번에 하나의 프로세스만을 실행할 수 있기 때문에, 최선의 환경을 만들기 위해서는 최대한 효율적으로 동작해야만 한다. 이를 위해 OS는 공정하고 효율적인 HW 리소스 관리를 위해 노력하는 것이다.