Chapter

01

운영체제의 시작과 발전

1. 운영체제 개념
2. 운영체제의 태동
3. 운영체제의 발전

강의 목표

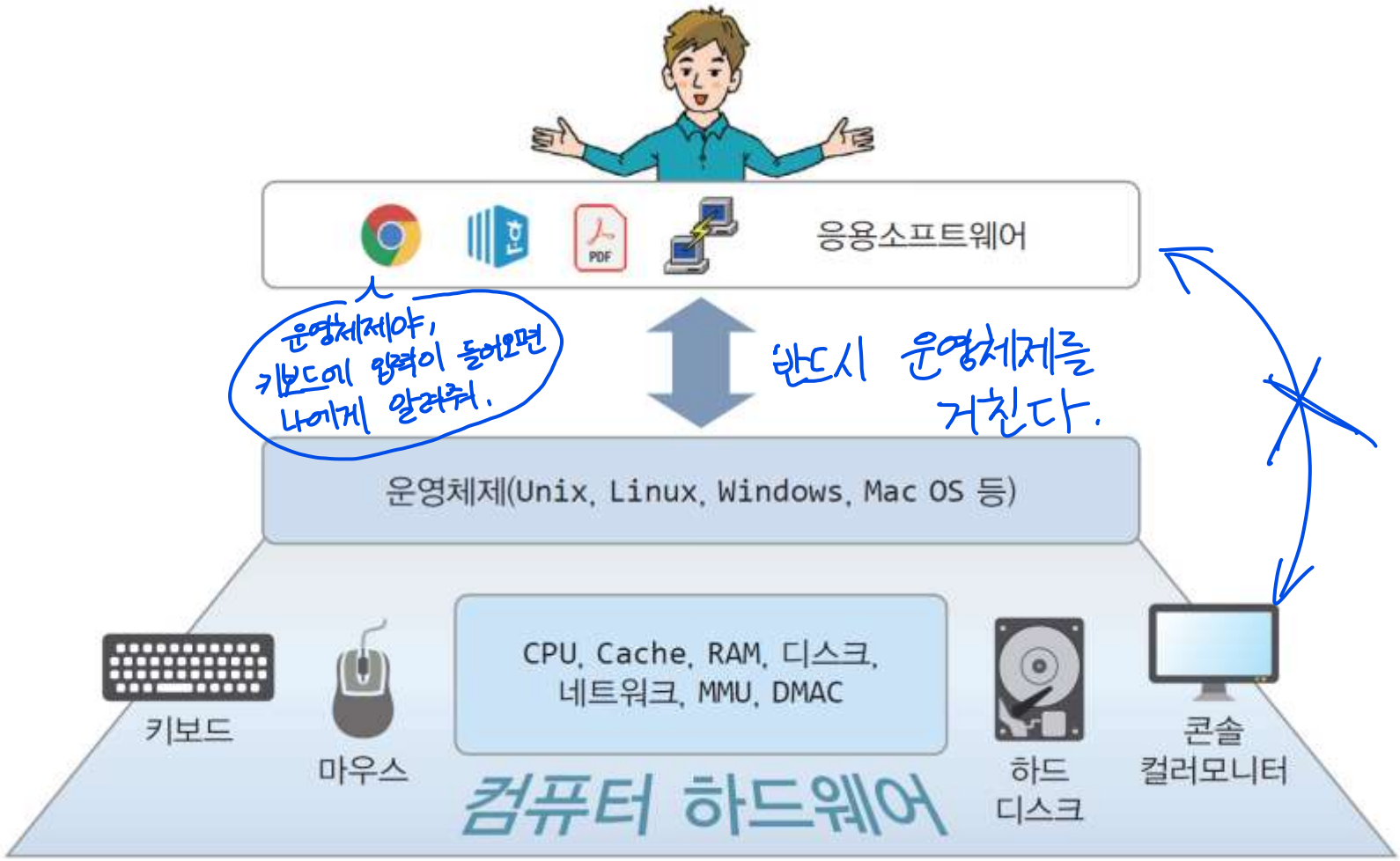
1. 운영체제의 정의와 목표를 알고 기능을 간단히 이해한다.
2. 운영체제가 없던 시절에서 운영체제가 생겨나는 태동 과정을 앎으로써 운영체제의 발단과 역할을 이해한다.
3. 원시 운영체제 이후 운영체제의 발전 과정을 이해한다.
4. 다중프로그래밍의 출현이 가져온 컴퓨터 시스템과 운영체제의 발전과 영향을 이해한다.
5. 운영체제의 종류와 특징을 간단히 이해한다.

1. 운영체제 개념

운영체제 정의

- 운영체제 정의들
  - 컴퓨터 사용자와 컴퓨터 하드웨어 사이에서 중계 역할을 하면서, 프로그램을 실행을 관리하고 제어하는 시스템 소프트웨어
    - 운영체제는 컴퓨터가 아니다
    - 운영체제는 실체가 있는 소프트웨어
  - 컴퓨터가 켜질 때 처음으로 적재되어 나머지 모든 프로그램의 실행을 제어하고 사용자의 요청을 처리해주는 소프트웨어
  - 컴퓨터의 **자원**을 **독점**적으로 **관리**하는 특별한 **소프트웨어**

컴퓨터와 운영체제, 그리고 사용자



운영체제의 정의에서 핵심 단어

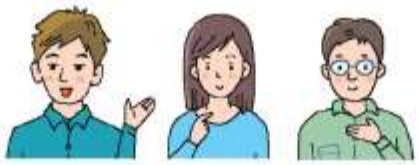
1. 운영체제는 컴퓨터의 모든 자원(resource) 관리
   - 자원
     - 하드웨어 자원 – CPU, 캐시, 메모리, 키보드, 마우스, 디스플레이,하드디스크,프린터
     - 소프트웨어 자원 – 응용프로그램
     - 데이터 자원 – 파일, 데이터베이스 등
2. 운영체제는 자원에 대한 독점(exclusive) 권한 소유
   - 자원에 대한 모든 관리 권한 운영체제에게 있음
     - 자원 할당, 자원 공유, 자원 액세스, 자원 입출력 등
     - 예) 파일 생성 – 디스크의 빈 공간 관리, 파일 저장 위치 관리, 파일 입출력 등
3. 운영체제는 관리자(supervisor)
   - 실행중인 프로그램 관리, 메모리 관리,
   - 파일과 디스크 장치 관리, 입출력 장치 관리, 사용자 계정 등 관리 등
4. 운영체제는 소프트웨어(software)
   - 커널(kernel)이라고 불리는 핵심 코드와,
   - UI를 비롯한 도구 프로그램들(tool/utility),
     - 예: 탐색기(explorer), 작업 관리자(task manager), 제어판(control panel) 등
   - 장치를 제어하는 디바이스 드라이버들로 구성

운영체제의 목적과 기능

- 운영체제의 목적
  - 사용자의 컴퓨터 사용 편리성
  - 컴퓨터의 자원 관리 효율성 메모리를 적게 사용하면서, 많은 프로그램이 실행되게 함

- 운영체제의 기능
  - CPU/프로세스 관리(process management)
  - 메모리 관리(memory management)
  - 파일 시스템 관리(file system management)
  - 장치 관리(device management)
  - 네트워크 관리
  - 보안 관리
  - 기타 관리
    - 사용자 관리 – 사용자 계정 관리
    - 통계 – CPU, 메모리, 네트워크의 사용 시간, 사용자의 접속 시간 등
    - 오류 발견 및 대응
    - 부팅(booting)

운영체제의 기능

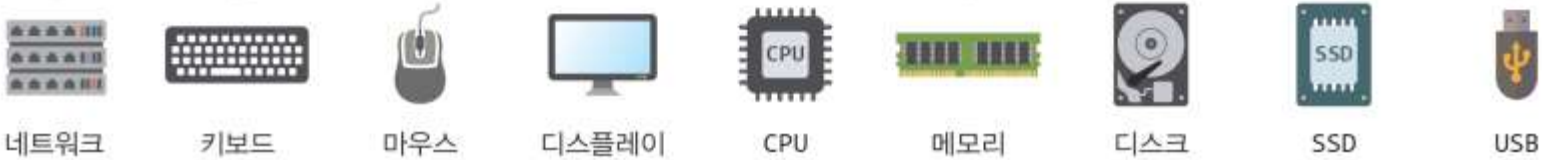


응용소프트웨어

실행중인 프로그램

보안 관리	네크워크 관리	장치 관리	프로세스 관리	메모리 관리	파일 시스템 관리	기타 관리
• 외부 침입으로부터 보호	• 네트워크 입출력 • 분산처리 관리	• 장치로부터 읽기 • 장치에 쓰기 • 장치 제어	• 프로세스 적재 • 프로세스 실행 • 프로세스 종료 • 새 프로세스 생성 • 프로세스 스케줄 • 프로세스 동기화	• 메모리 할당/반환 • 메모리 보호 • 디스크 영역으로 메모리 영역 확장 (가상 메모리)	• 파일 생성/삭제 • 파일 열기/닫기 • 파일 읽기/쓰기 • 파일 저장 위치 관리 • 파일 속성 등 정보 관리 • 디스크의 빈 영역 등 저장 장치 관리	• 사용자 계정 • 통계 • 오류 발견 및 대응 • 부팅

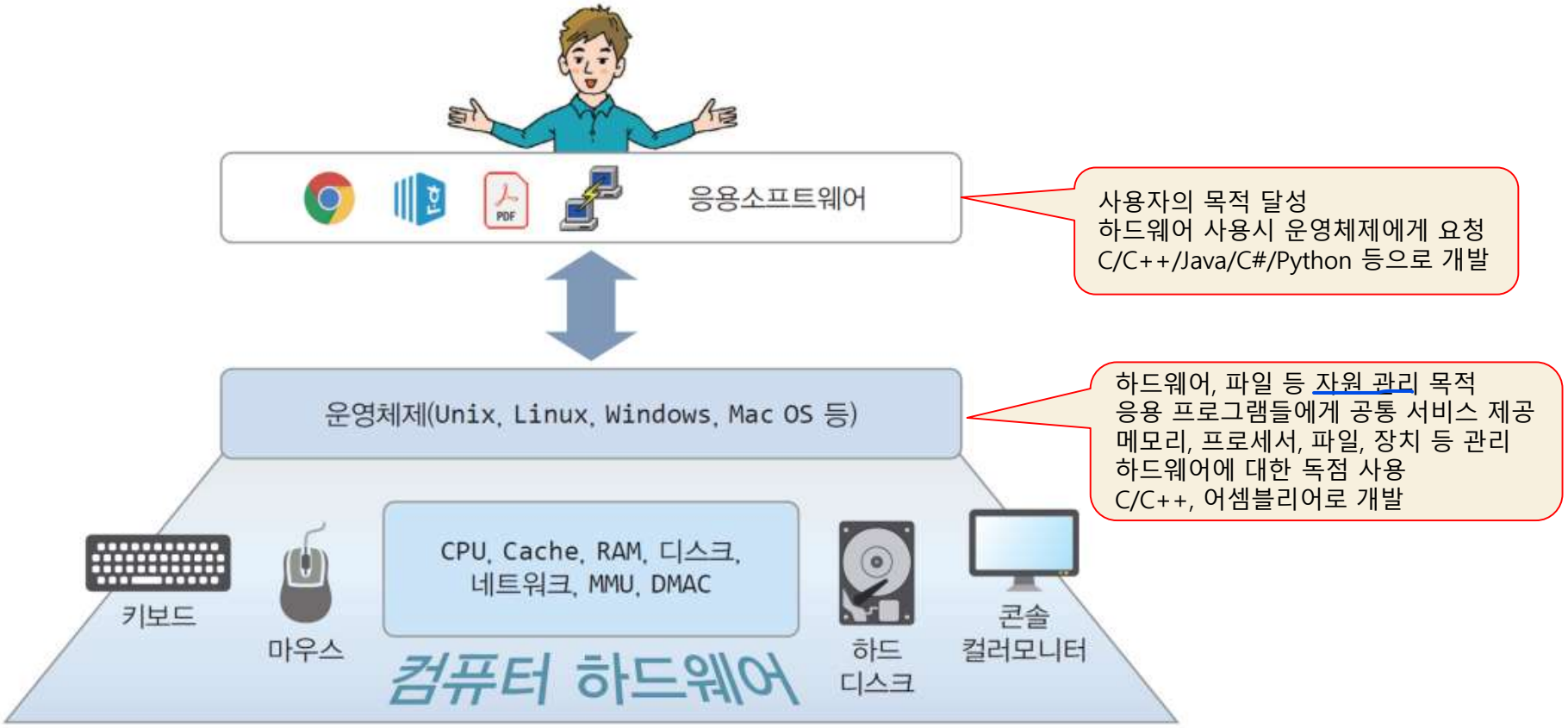
운영체제



네트워크 키보드 마우스 디스플레이 CPU 메모리 디스크 SSD USB

프로그램을 적재

운영체제와 응용소프트웨어의 차이



운영체제와 응용소프트웨어

	운영체제	응용소프트웨어
목적	컴퓨터 하드웨어나 응용소프트웨어 등 자원 관리	사용자들의 특정 작업을 보다 편리하게 처리할 목적으로 만들어진 소프트웨어(예: 게임, 웹서핑, 문서작성, 채팅 등)
기능	프로세스, 메모리, 파일 시스템, 입출력 장치 등 자원 관리와 사용자 관리	소프트웨어를 만든 특정 목적만 수행
개발 언어	C/C++, 어셈블리어	C/C++뿐 아니라 Java, Python, C# 등 다양한 언어
실행	부팅 시 메모리에 적재되어 상주하여 컴퓨터를 끌 때까지 실행	사용자가 명령을 통해 실행시키거나 종료시킴
자원에 대한 접근 권한	컴퓨터의 모든 자원에 대해 배타적 독점 사용 권한	컴퓨터 자원을 사용하고자 할 때 반드시 운영체제에게 요청

2. 운영체제의 태동

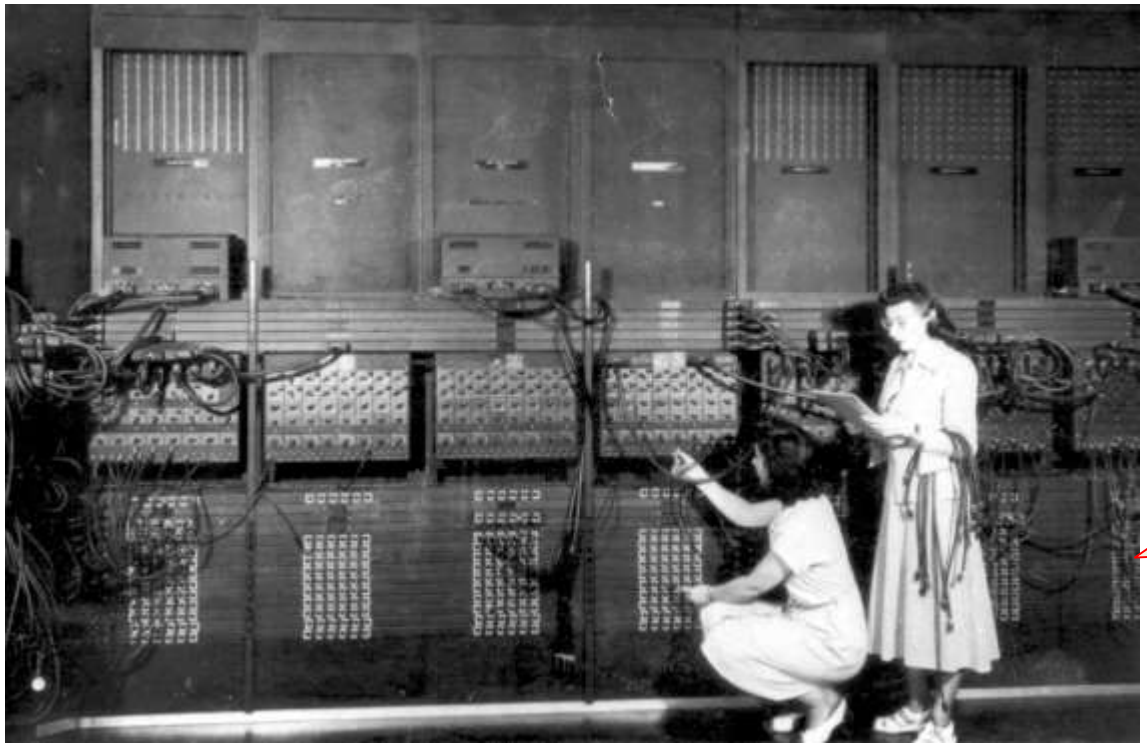
1. 고정 프로그래밍 방식 - 1940년대
2. 내장 프로그래밍 방식 – 1945년 이후
3. 프로그램 로딩 시대
4. 로더의 필요성 부각 - 1954년 IBM 701 개발 후
5. 원시 운영체제 GM OS의 탄생 - 1955년
6. 최초의 운영체제 GM-NAA I/O 개발 - 1956~1957년

1. 고정 프로그램 컴퓨터 – 1940년대

- 고정 프로그램 방식(fixed program computer)
  - 1940년대, 전자식 디지털 컴퓨터가 만들어지기 시작하는 시대
  - 운영체제에 대한 개념 없음
  - 소프트웨어와 하드웨어의 분리 개념 없음
    - 모든 것이 하드웨어로 제작
  - 프로그래밍
    - 종이에, 프로그램을 구현하는 스위치와 전선 연결도 작성
    - 배선판(plugboard)에 전선 연결, 프로그램을 기계에 고착화
    - 하나의 명령을 구성하기 위해 여러 가닥의 전선 연결
    - 프로그램 전체 구축에 수천 개의 전선 연결, 며칠 소요
    - 새로운 프로그램을 작성할 때(구축할 때) 큰 고통
  - 사례
    - 1941년 독일에서 만든 Z3 computer
    - 1944년 영국에서 독일군의 암호를 해독하기 위해 만든 Colossus
    - 1943~1945년 미국에서 만든 최초의 전자식 컴퓨터 ENIAC(Electronic Numerical Integrator And Computer)

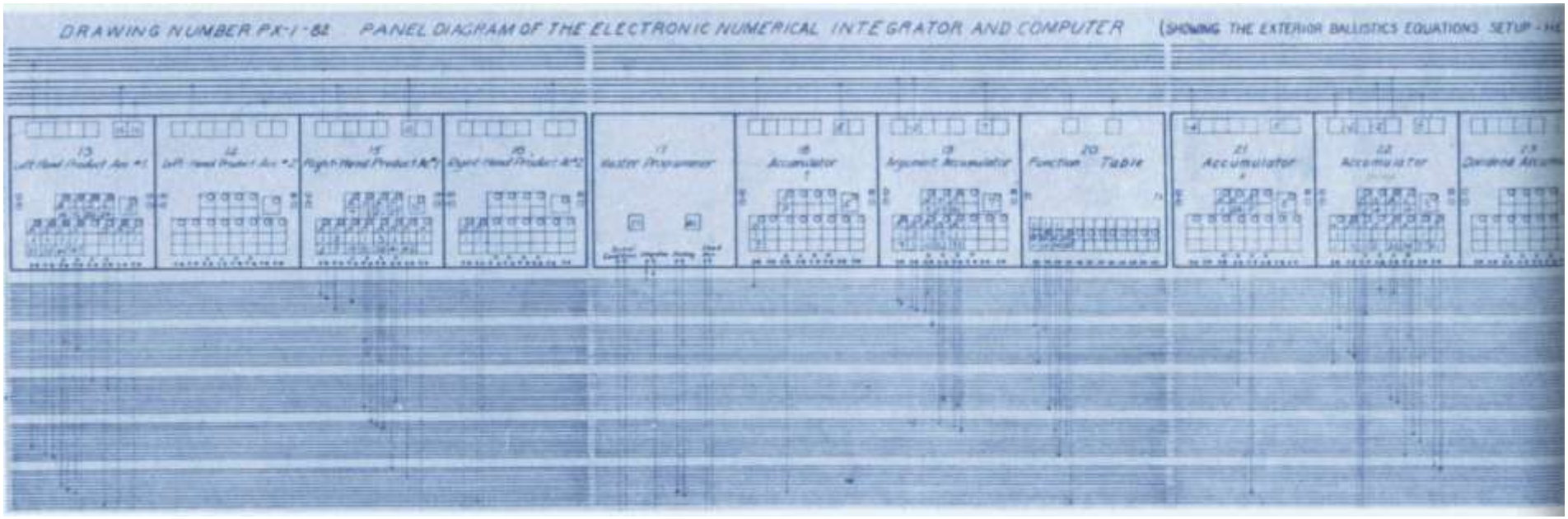
ENIAC 컴퓨터의 사례

- 방 하나를 메울 30톤 크기, 진공관 약 18000개로 구성



ENIAC 컴퓨터에 여성 프로그래머가 전선을 연결하는 식으로 프로그램을 입력하는 모습

프로그램 연결도



(출처: http://www.Columbia.edu/cu/computinghistory/eniac-program.gif)

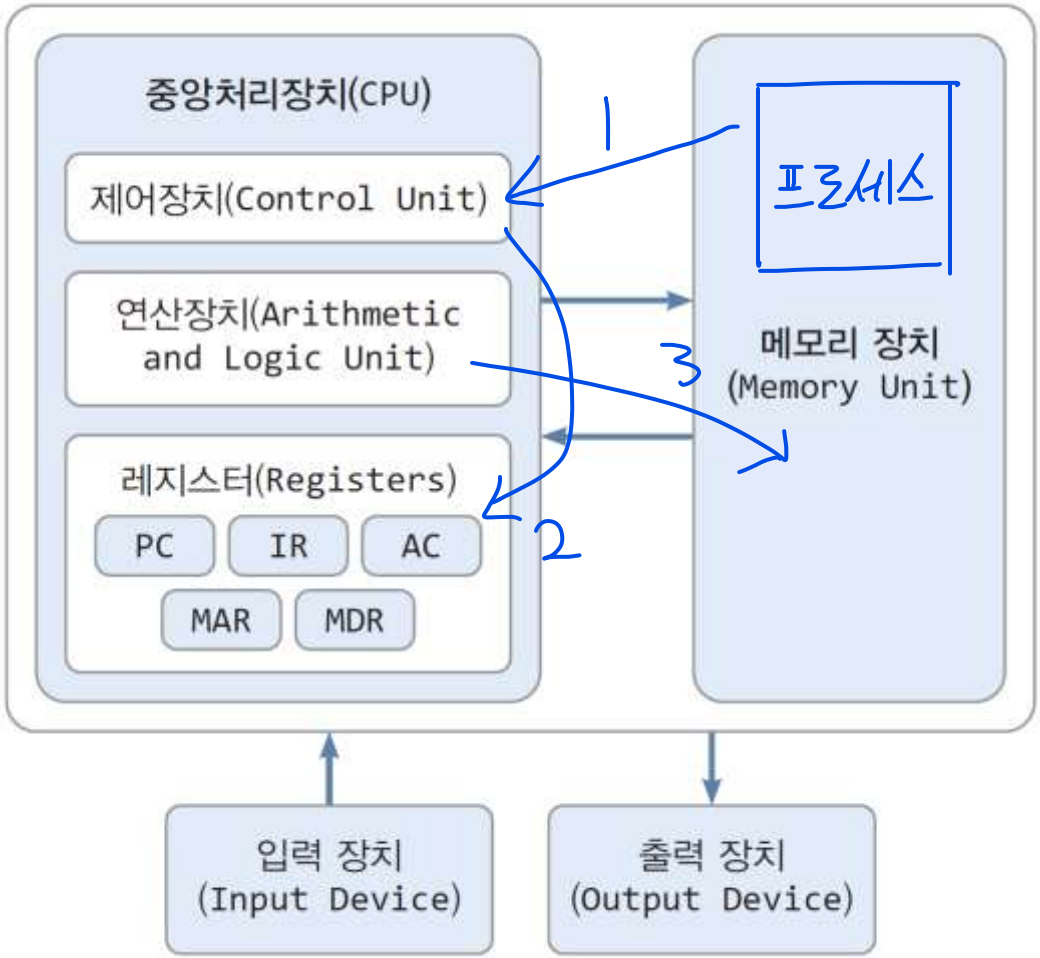
ENIAC 컴퓨터의 프로그램의 셋업(연결선)을 그려놓은 패널 다이어그램

2. 내장 프로그래밍 방식 등장 - 1945년~

- 내장 프로그래밍 방식(stored program)
  - 1945년 폰노이만에 의해 제안
  - 1951년 EDVAC 컴퓨터를 만들 때 적용
  - 오늘날 컴퓨터의 구조가 됨

- 내장 프로그램 컴퓨터의 의미
  - CPU와 메모리 분리
  - 소프트웨어와 하드웨어 분리
  - 실행할 프로그램을 메모리에 담고, CPU가 프로그램을 실행하는 방식
    - 고정 프로그래밍 방식에 비해 획기적인 변화
    - 하드웨어의 변화 없이, 실행시키려는 프로그램만 메모리에 적재
  - 프로그램은 입력 장치를 통해 메모리에 적재
    - 펀치 카드에 구멍을 뚫어 프로그램 작성
    - 카드 리더기로 프로그램을 메모리에 읽어 들임

폰노이만이 제안한 내장 프로그램 컴퓨터 구조



3. 프로그램 로더의 발견 – 운영체제 개념 시작(1950년대)

- 프로그램 로딩 시대
  - 운영체제 개념의 시작
- IBM 701 메인 프레임
  - 1954년 IBM에서 만든 첫 번째 내장 프로그래밍 컴퓨터
  - IBM의 첫 번째 범용 컴퓨터
  - 판매하지 않고 대여만
  - 기계만 대여하고 어떤 소프트웨어도 제공하지 않았음
    - IBM 701을 빌린 고객(기업)이 모든 것을 해야 했음

IBM 701 mainframe computer

IBM 701 메인프레임 컴퓨터의 컨트롤 패널

IBM 701을 사용하는 방법은 간단히 다음과 같다.

1. 프로그램 개발자들(이들은 곧 오퍼레이터이다)은 이름을 적어 대기명단에 올린 후 자신의 차례가 되기를 기다린다.

2. 자신의 차례가 되면 모든 것을 스스로 해야 한다. 주어진 시간은 15분으로 알려져 있다. 컴퓨터실에 들어가, 카드 입력 장치, 프린터, 테이프 장치 등을 확인하고, 컨트롤 패널 앞에 선다. 프로그램의 입력, 실행, 출력 모두 개발자가 컨트롤 패널을 통해 제어한다. 컨트롤 패널 상에 있는 조작 스위치를 세팅한다.

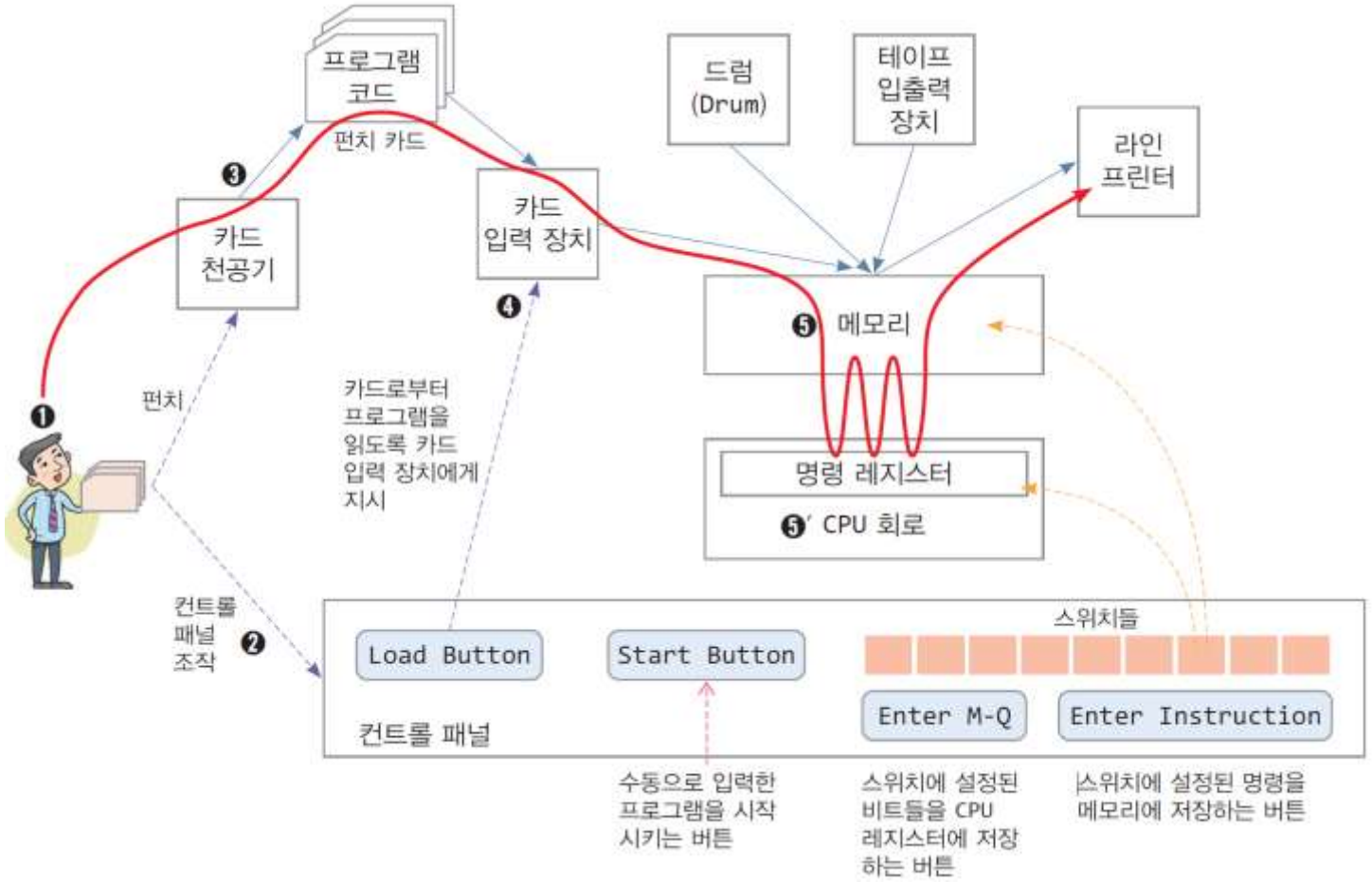
3. 카드 천공기를 이용하여 종이에 적어간 프로그램을 카드에 펀칭한다. 카드에는 모두 기계어 즉 이진수로 펀치된다(그 후 어셈블리어로 펀치되도록 개선).

4. 펀치된 카드를 카드 리더기에 쌓고 컨트롤 패널의 'LOAD' 버튼을 눌러 컴퓨터를 가동(부트스트랩)시킨다.

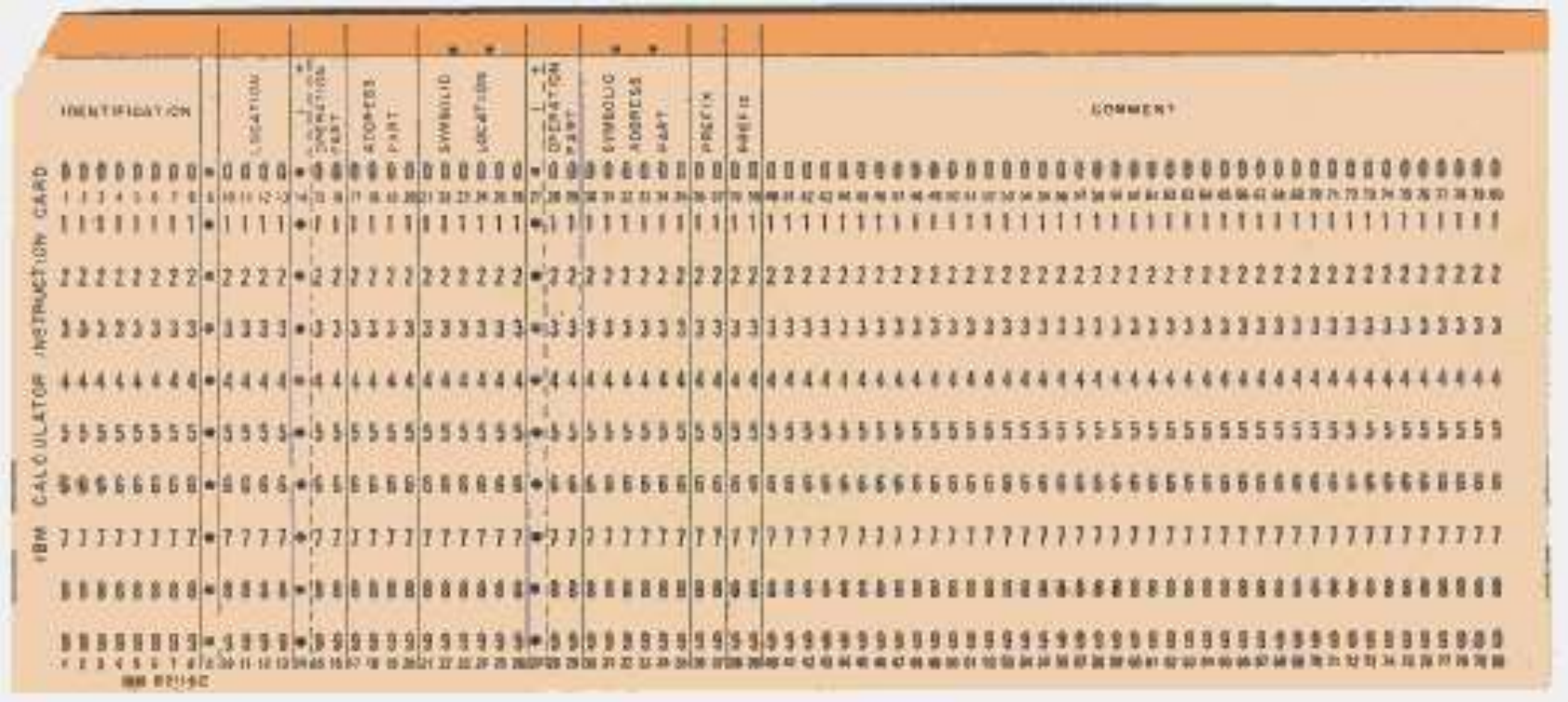
5. LOAD 버튼은 한 장의 카드만 읽는 것으로 끝난다. 카드 한 장에는 최대 24개의 명령을 작성할 수 있다. LOAD 버튼이 눌러지면, 카드 리더기를 이용하여 첫 번째 카드에 작성된 프로그램(여러 기계어 명령들)을 메모리 0번지부터 순서대로 적재 시킨 후, 0번지부터 실행시킨다.

6. 수동으로 프로그램을 입력하고 실행시키기 위해 LOAD 버튼 대신 컨트롤 패널에서 Enter Instruction 버튼, Start Button 등을 눌러 실행시킬 수도 있

IBM 701의 프로그램 입력 및 실행 과정



IBM 701에서 프로그램 작성에 사용된 펀치 카드



출처 : Douglas John의 punched card history, http://landley.net/history/mirror/pre/punchedcards/history.html

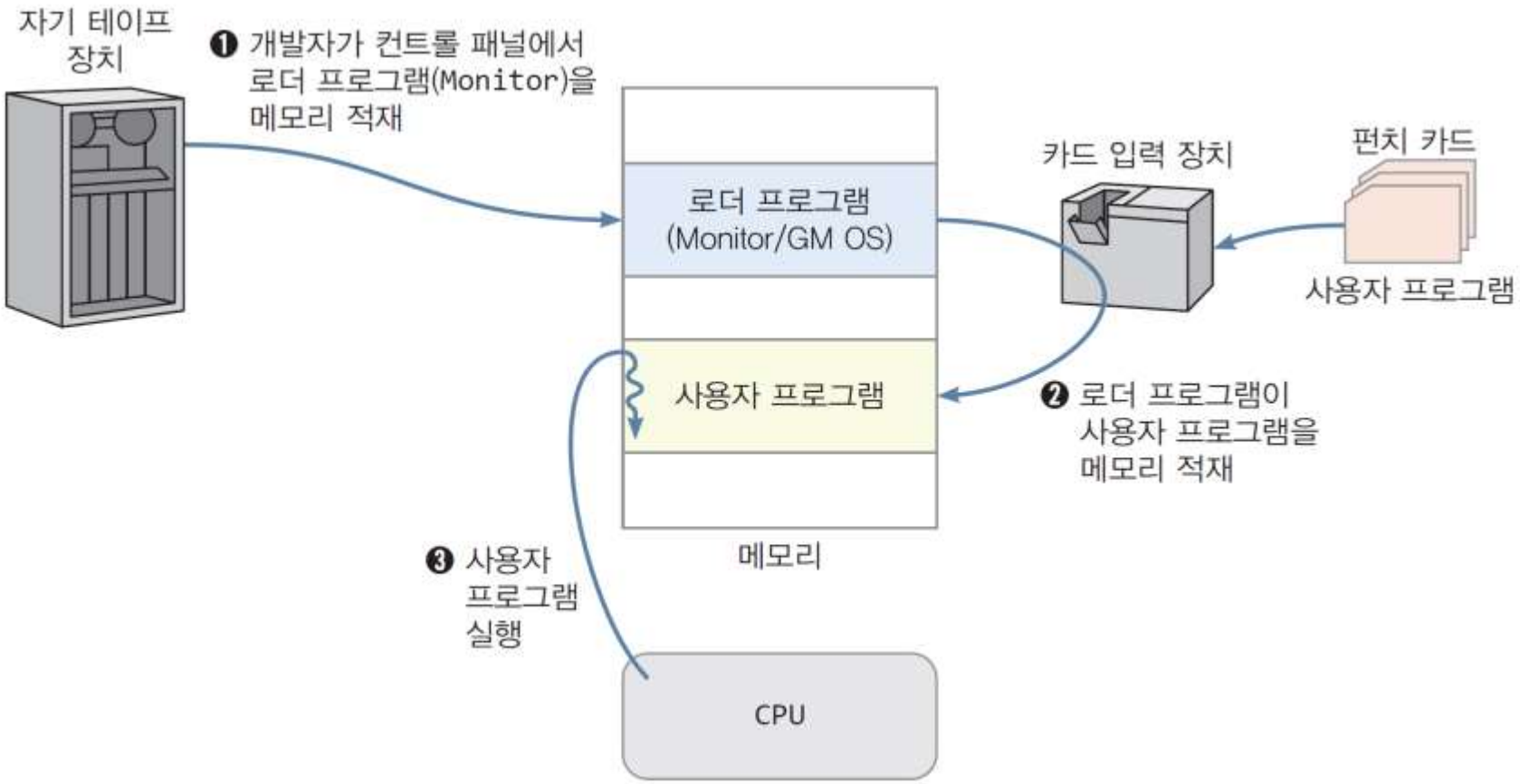
로더 프로그램의 필요 – 운영체제의 싹

- IBM 701 개발자의 일반적인 개발 형태
  - 첫번째 카드
    - 다음 카드에 작성된 프로그램을 메모리에 적재하는 코드만 작성
      - 왜냐하면, IBM 701은 첫번째 카드만 메모리에 적재하는 기능 뿐이고,
      - 한 장의 카드에 24개의 명령만 작성 가능, 24개보다 긴 프로그램 작성에 2장 이상의 카드 필요
  - 두번째 카드부터
    - 목적하는 프로그램 작성
- 로더 프로그램 필요
  - 개발자가 뒷 카드들을 메모리에 적재하는 프로그램을 첫번째 카드에 작성하는 반복되는 시간 낭비를 줄일 필요
    - 이 코드를 로더(loader)라고 부름
    - 로더는 모든 사용자(개발자)에게 공통으로 필요
- **로더가 운영체제로 발전**
  - 오늘날 운영체제의 가장 기본적인 기능?
    - 사용자의 명령을 받아 저장 장치에 담긴 프로그램을 메모리에 적재하는 기능

4. 원시 운영체제 GM OS 탄생 - 1955년

- GM OS
  - 1955년, IBM701의 고객 GM(General Motors)에 의해 개발
    - 펀치 카드에 담긴 프로그램을 메모리에 적재하는 로더 프로그램 개발
  - 핵심 개념
    - 로더 프로그램을 사용자 프로그램에서 분리
    - 사용자가 자신의 프로그램을 적재하는 셀프-로더 프로그램 작성의 번거로움 해소
    - 로더 프로그램은 시스템 내 테이프에 저장, 필요할 때마다 실행
    - 그 당시, 로더 프로그램을 모니터(Monitor)라고 불렀음
    - 후세대, General Motors Operating System(GM OS)라고 불렀음
- GM OS는 원시적인 최초의 운영체제
  - 사용자 프로그램에서 로더 프로그램 분리하고,
  - 로더 프로그램을 시스템에 한 개만 두고,
  - 사용자 프로그램을 실행할 때 작동하여,
  - 사용자 프로그램을 읽어 실행시켜준다는 점에서

원시 운영체제 GM OS


자기 테이프 장치
❶ 개발자가 컨트롤 패널에서 로더 프로그램(Monitor)을 메모리 적재
로더 프로그램 (Monitor/GM OS)
사용자 프로그램
메모리
카드 입력 장치
펀치 카드
사용자 프로그램
❷ 로더 프로그램이 사용자 프로그램을 메모리 적재
❸ 사용자 프로그램 실행
CPU

5. 최초의 운영체제 GM-NAA I/O 탄생 - 1956~1957년

- 1955년 GM은 IBM 701의 처리 속도를 높인 IBM 704 주문
- IBM 704 컴퓨터 활용에 3가지 문제점 발견과 해결 시도
  1. 개발자들은 여전히 대기 번호를 뽑고 자신의 차례를 기다린다는 점
  2. 많은 시간 비싼 컴퓨터를 놀리고 있다는 점
     - 개발자가 프로그램을 실행시키기 위한 시스템 셋업하는 동안, 컴퓨터는 놀게 됨
  3. 입출력 루틴을 개발자 스스로 작성
     - 카드 입력 장치나 테이프 장치를 제어하는 프로그램 코드

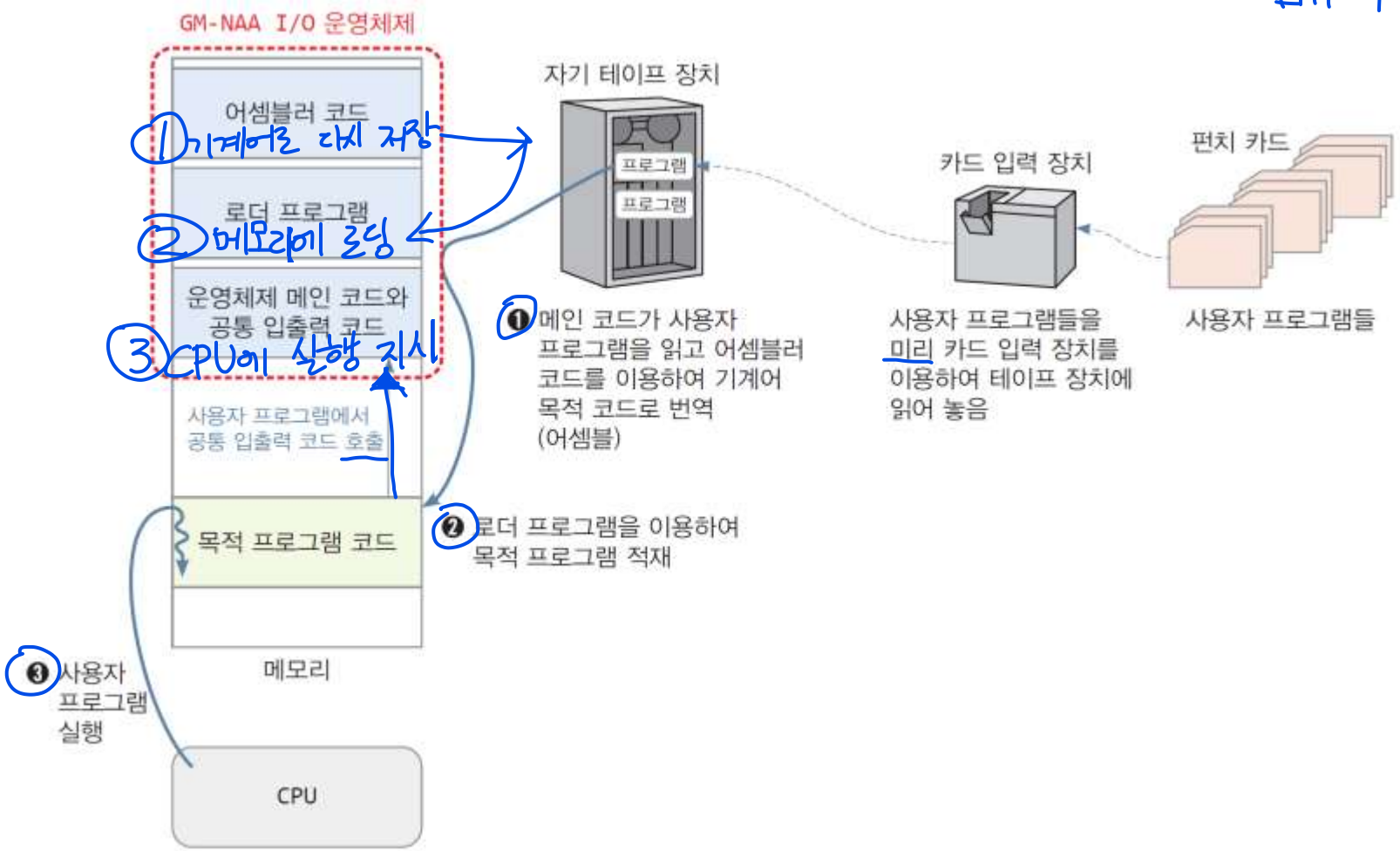
- 최초의 운영체제 GM-NAA I/O 개발
  - 1956년 GM과 NAA(North American Aviation)의 공동 개발
    - 고가의 IBM 704 컴퓨터를 보다 효율적으로 사용하기 위해
    - IBM 701의 모니터 프로그램(GM OS)을 확장하여 구현
  - 운영체제로서의 모습 갖춤
    - 배치 방식(batch operating system)으로 작동
      - 개발자들이 작성하여 쌓아놓은 작업들을 순서대로 하나씩 메모리에 적재, 한 번에 하나의 작업 실행
    - GM-NAA I/O 프로그램을 메모리에 상주(오늘날 운영체제 방식)
    - 입출력 장치들을 제어하는 루틴들을 라이브러리 형식으로 갖추고 프로그램 사이에 공유
      - 카드 입력 장치의 입출력, 테이프 입출력, 프린터 출력 등
      - 개발자는 입출력 코드를 작성할 필요 없음

GM-NAA-I/O 운영체제의 구조와 기능

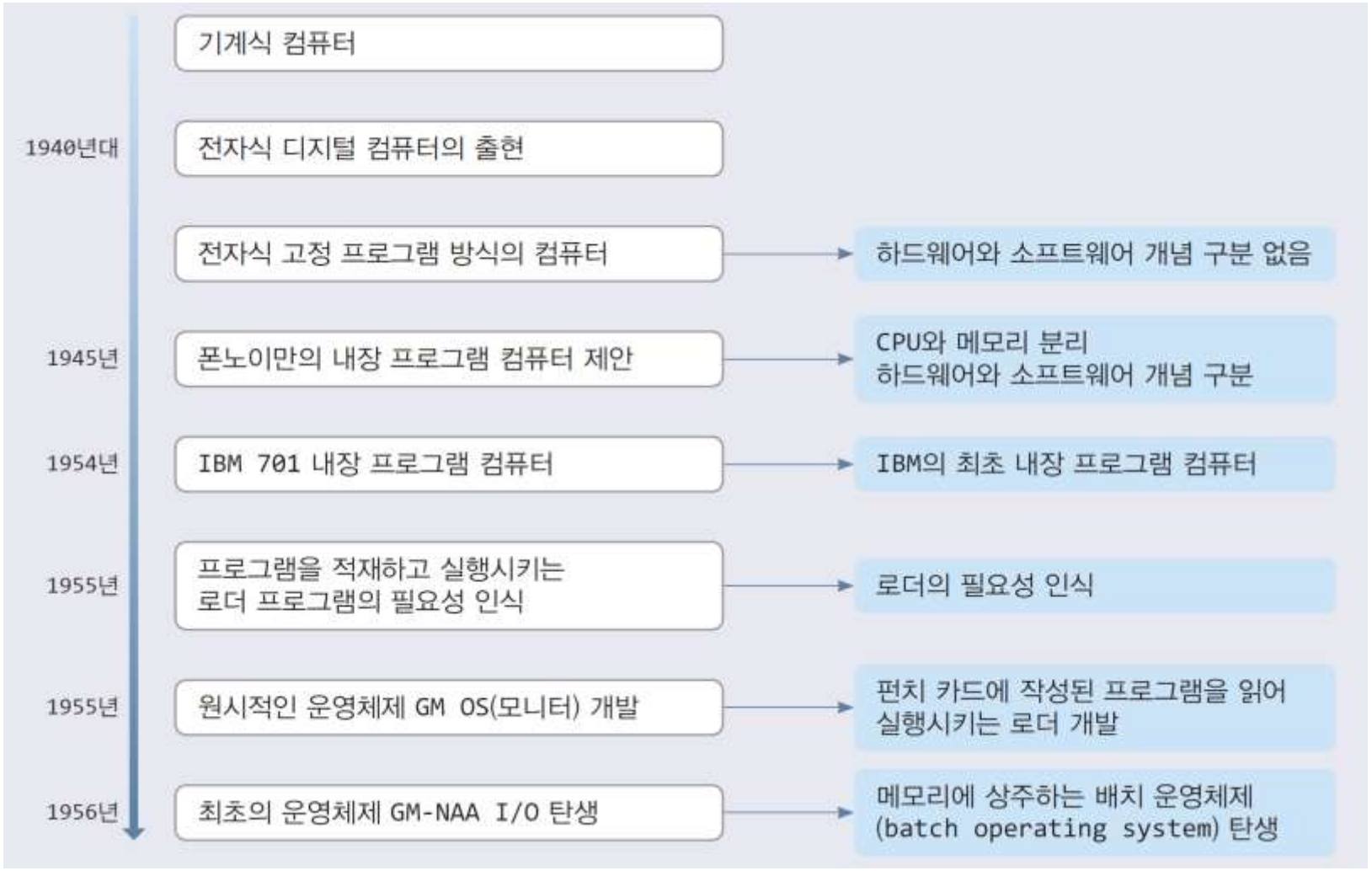
- GM-NAA-IO의 구조
  - 어셈블러 코드
    - 사용자가 작성한 어셈블리어 프로그램을 기계어 코드로 번역
  - 로더 프로그램
    - 사용자 프로그램을 하나씩 메모리에 적재
  - 운영체제 메인 코드와 공통 입출력 코드
    - 운영체제 메인 코드 – 운영체제 시작 코드
    - 공통 입출력 코드 – 장치 입출력을 다루는 프로그램 코드
      - 라이브러리화 되어 모든 사용자 프로그램(목적 프로그램)이 실행 시 호출하여 활용
      - 라이브러리 개념 등장

IBM 704에서 GM-NAA-I/O 운영체제의 작동

배치방식



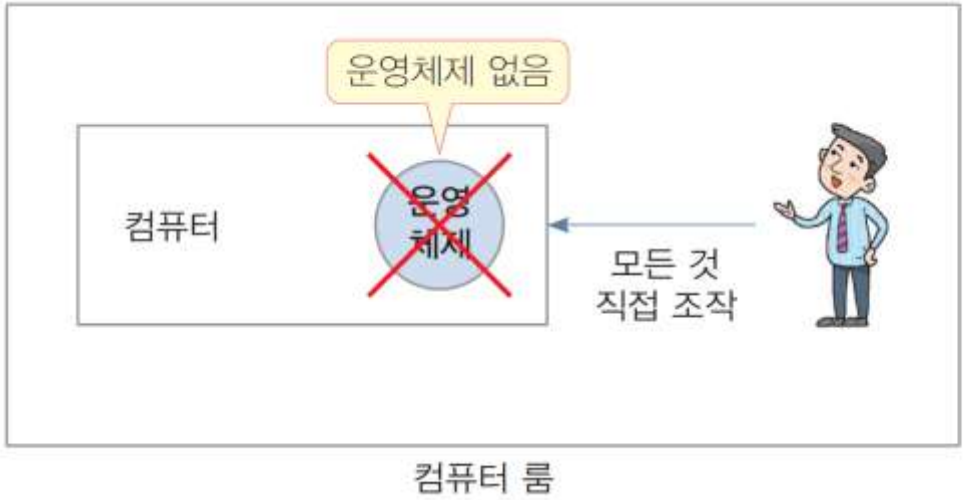
운영체제 태동의 역정


기계식 컴퓨터
1940년대
전자식 디지털 컴퓨터의 출현
전자식 고정 프로그램 방식의 컴퓨터
하드웨어와 소프트웨어 개념 구분 없음
1945년
폰노이만의 내장 프로그램 컴퓨터 제안
CPU와 메모리 분리
하드웨어와 소프트웨어 개념 구분
1954년
IBM 701 내장 프로그램 컴퓨터
IBM의 최초 내장 프로그램 컴퓨터
1955년
프로그램을 적재하고 실행시키는
로더 프로그램의 필요성 인식
로더의 필요성 인식
1955년
원시적인 운영체제 GM OS(모니터) 개발
펀치 카드에 작성된 프로그램을 읽어
실행시키는 로더 개발
1956년
최초의 운영체제 GM-NAA I/O 탄생
메모리에 상주하는 배치 운영체제
(batch operating system) 탄생

3. 운영체제의 발전

운영체제 태동 시절

- 1세대 컴퓨터 시절 - 운영체제 암흑 시대
- 운영체제의 개념 없음
  - 개발자가 펀치 카드에 프로그램 작성, 입력, 실행
  - 컴퓨터는 한 번에 한 개의 작업만 실행
  - 컴퓨터는 셋업하는 동안 많은 시간이 유휴(노는,idle) 상태



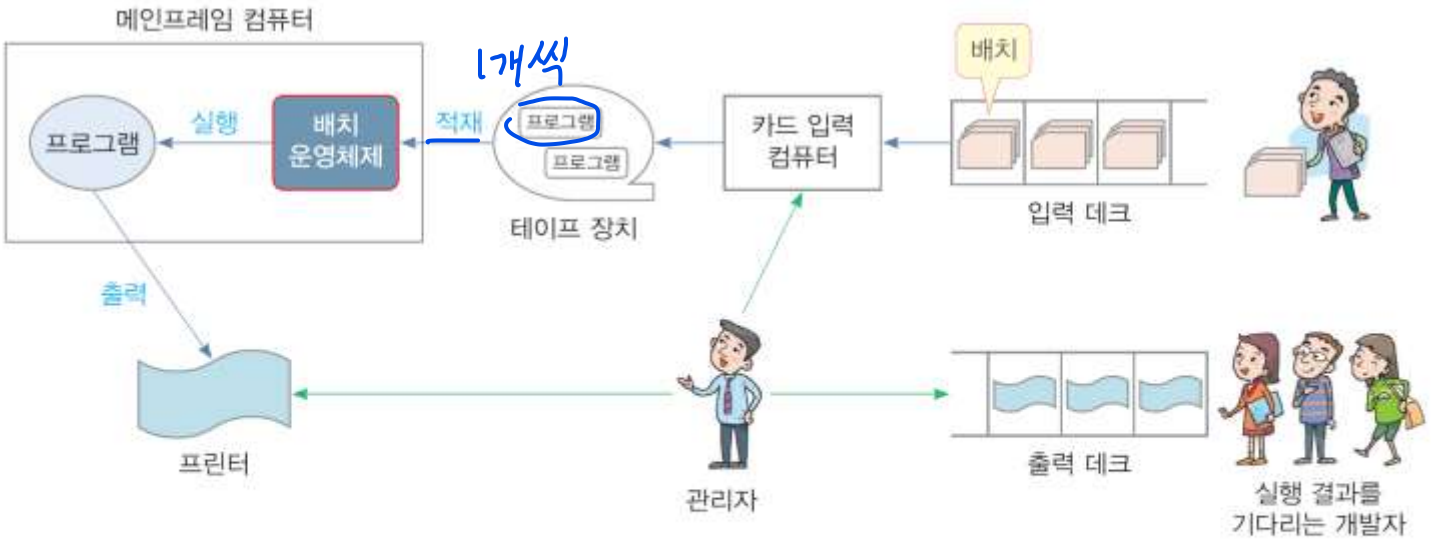
컴퓨터 룸

대기표를 받고
기다리는 개발자

개발자가 프로그램 작성,
컴퓨터 켜기, 프로그램 적재,
실행, 출력 모든 것을 제어

배치 운영체제

- 출현 배경
  - 컴퓨터의 노는 시간(idle 시간, 유휴시간)을 줄여 컴퓨터의 활용률 향상
- 배치 운영체제 컴퓨터 시스템
  - 개발자와 관리자의 구분
  - 개발자는 펀치 카드를 입력 데크에 두고 결과 기다림
  - 배치 운영체제는 자동으로 테이프 장치에 대기중인 프로그램을 **한 번에 하나씩 적재하고, 실행**



- 관리자는 비교적 저렴한 카드 입력 컴퓨터를 이용하여 펀치 카드를 테이프 장치에 적재
- 배치 운영체제는 테이프 장치에 대기 중인 작업을 한 번에 하나씩 읽어 들여 실행
- 프로그램이 출력한 결과는 프린터에 출력
- 관리자는 출력된 프린트 용지를 개발자 별로 나누어 출력 데크에 쌓아 놓음

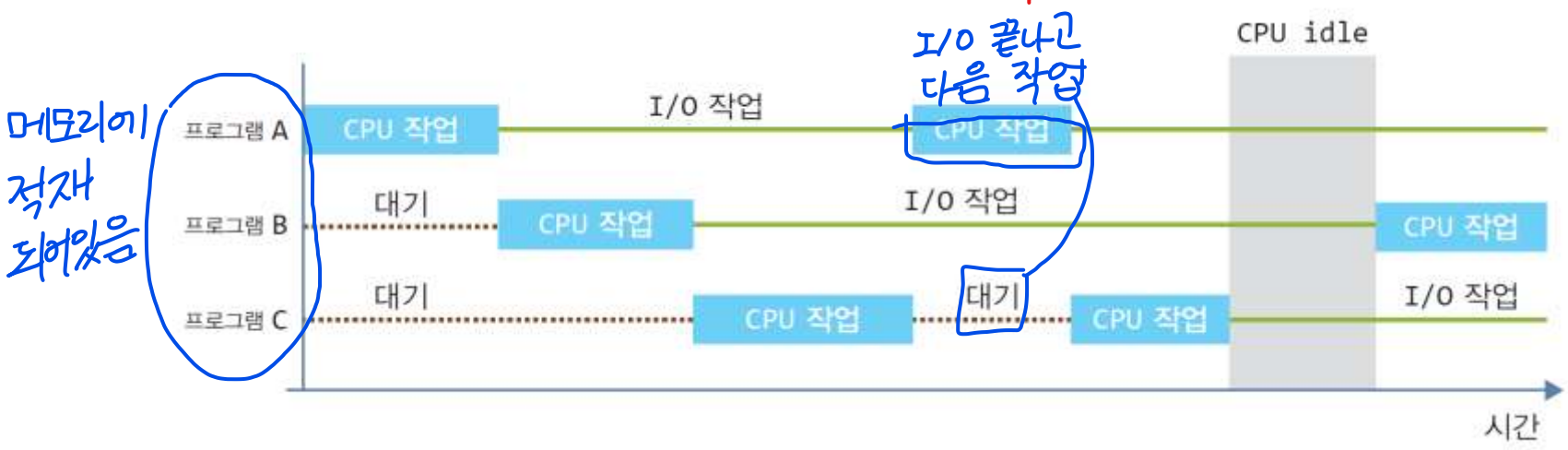
다중프로그래밍(Multiprogramming) 운영체제

- 출현 배경
  - 1960년대 중반
    - CPU 등 하드웨어 속도 개선, 컴퓨터 가격 증가
  - 프로그램의 실행 형태로 인한 CPU의 유휴시간(idle 시간) 발생
    - 프로그램 실행 형태: CPU 작업 – I/O 작업 – CPU 작업 – I/O 작업의 반복
    - 배치 작업은 1번에 1개의 프로그램만 실행하므로,
    - I/O 작업이 이루어지는 동안 CPU는 놀면서 대기, CPU의 많은 시간 낭비
  - CPU의 유휴시간을 줄일 필요

    -> CPU 활용률 증가 -> 처리율 증가(더 많은 사용자 프로그램 실행)

- 다중프로그래밍 기법 출현
  - 미리 여러 프로그램을 메모리에 적재
    - 메모리가 수용할 만큼 다수의 프로그램 적재
  - 프로그램 실행 도중 I/O가 발생하면,
  - CPU에게 메모리에 적재된 다른 프로그램 실행시킴
- 정의
  - 다중프로그래밍은 여러 프로그램을 메모리에 올려놓고, CPU가 한 프로그램을 실행하다 I/O가 발생하면, 입출력이 완료될 때까지 CPU가 메모리에 적재된 다른 프로그램을 실행하는 식으로 CPU의 노는 시간을 줄이는 기법이다.

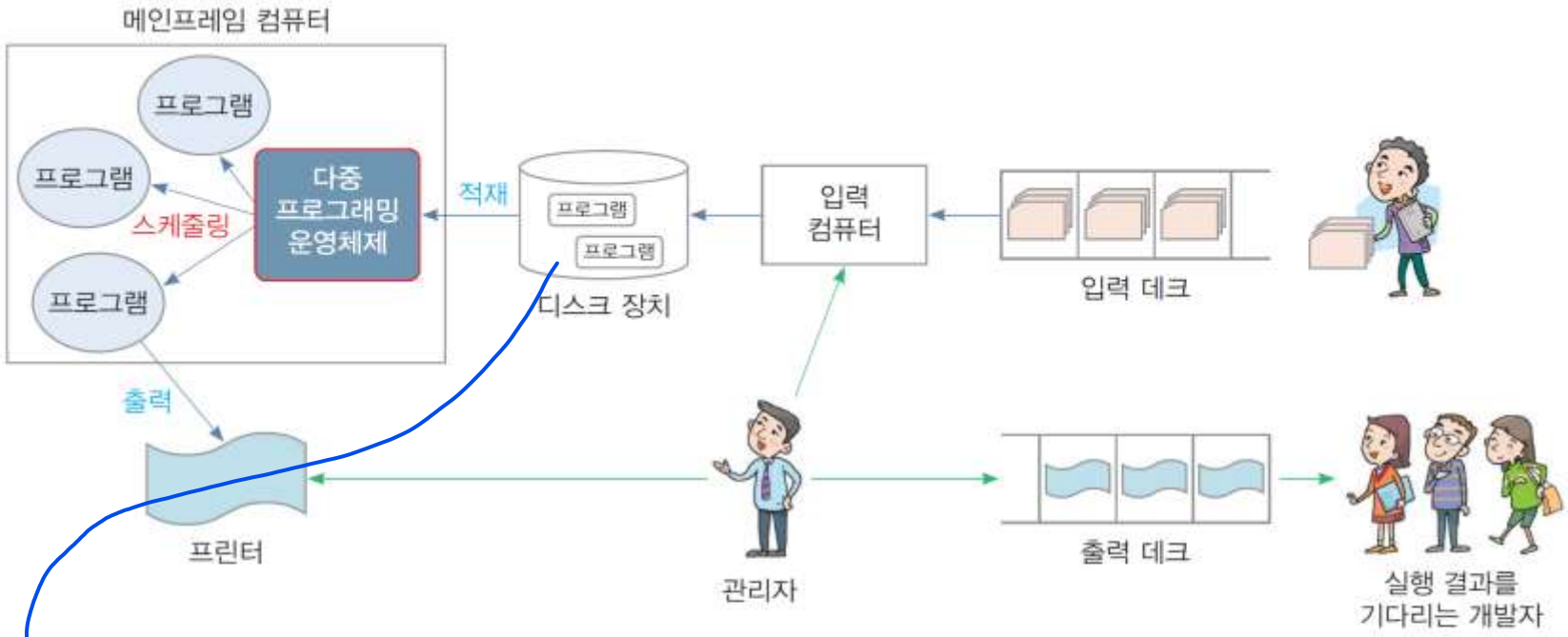
다중프로그래밍 기법



다중프로그래밍 기법으로 3개의 프로그램이 실행되는 과정

CPU의 활용

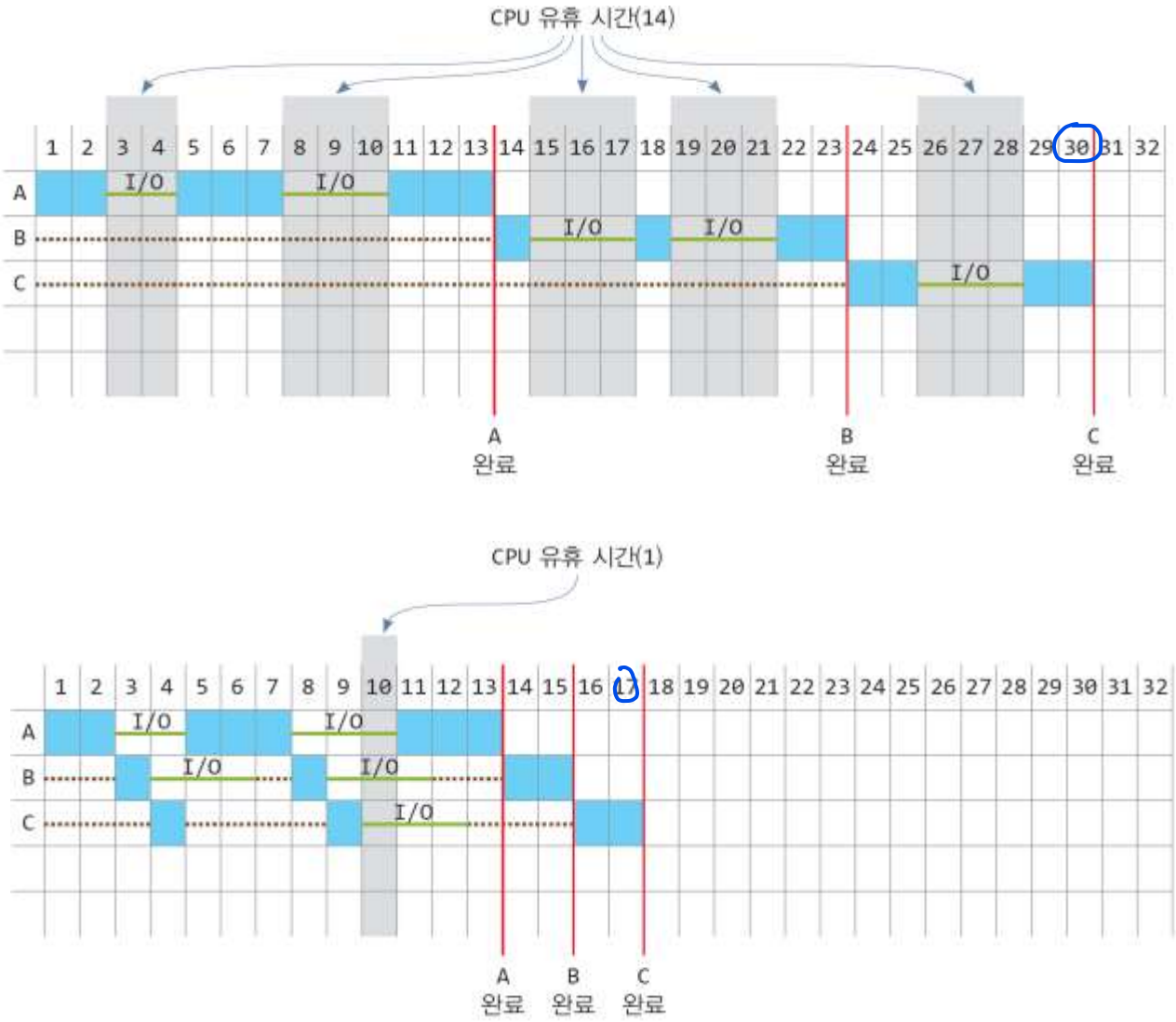
다중프로그래밍 운영체제를 사용하는 시스템



- 메모리에 빠르게 적재하기 위해 테이프 장치 대신 디스크 장치 사용
- 운영체제는 메모리 크기를 고려하여 디스크 장치에 대기 중인 적당한 개수의 프로그램 적재
- 한 프로그램의 실행이 끝날 때마다 디스크에서 대기 중인 프로그램 적재
- 관리자는 출력된 프린트 용지를 사용자 별로 나누어 출력 데크에 쌓아 놓음
- 프로그램이 I/O를 실행하면, 운영체제는 I/O가 완료될 때까지 메모리에 적재된 다른 프로그램을 선택하고 CPU가 실행
- 시스템의 구조는 배치 시스템의 구조와 거의 유사 (조금 다름)

스케줄링: I/O가 발생했다. 다음인 무엇을 실행시키지?

탐구 1-1 배치 운영체제와 다중프로그래밍 운영체제의 실행 비교



(a) 배치 운영체제

(b) 다중프로그래밍 운영체제

탐구 1-1의 배치 시스템과 다중프로그래밍 시스템의 성능 비교

	배치 운영체제	다중프로그래밍 운영체제
총 실행 시간	30	17
CPU 유휴 시간	14	1
CPU 활용률	16/30 = 0.53 = 53%	16/17 = 0.94 = 94%
작업 처리율	3/30 = 0.1작업/시간	3/17 = 0.176작업/시간

CPU 일한시간 / 총 실행시간

잘 활용됨

완료한 프로그램

다중프로그래밍 도입으로 인한 이슈

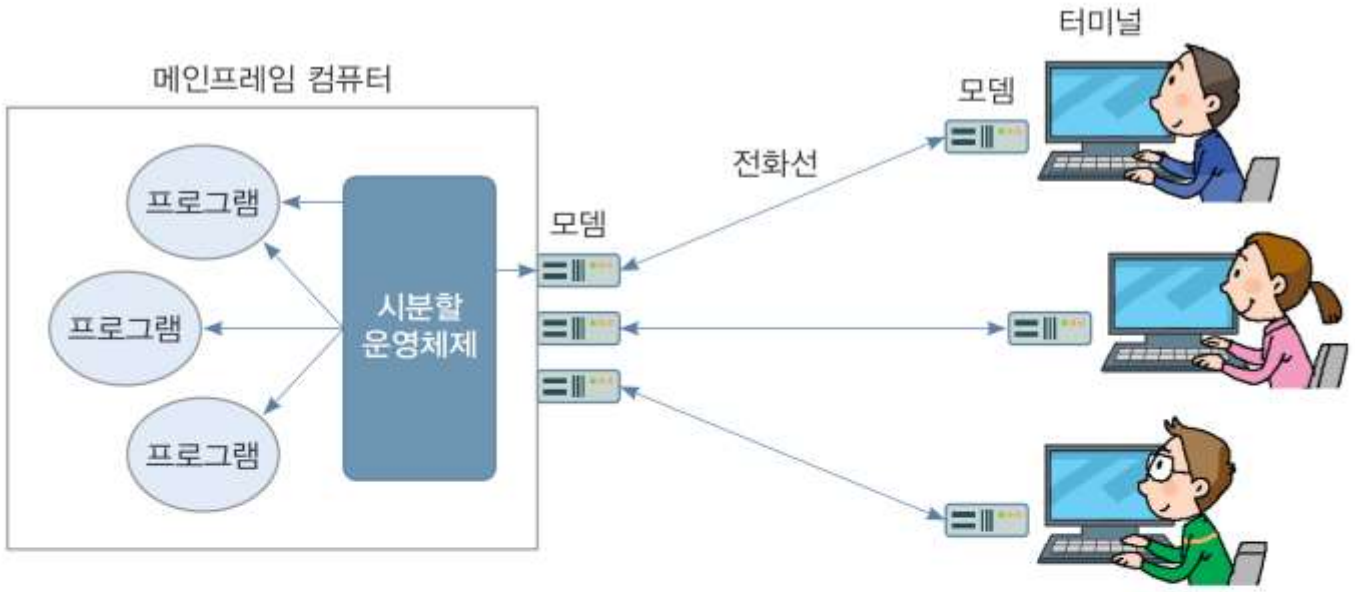
- 큰 메모리 이슈
  - 여러 프로그램을 동시에 메모리에 올려놓기 위해서는 메모리의 크기 늘릴 필요
- 프로그램의 메모리 할당 및 관리 이슈
  - 몇 개의 프로그램 적재? 메모리 어디에 적재? 프로그램 당 할당하는 메모리 크기?
- 메모리 보호 이슈
  - 프로그램이 다른 프로그램의 영역을 침범하지 못하게 막는 방법 필요
- CPU 스케줄링과 컨텍스트 스위칭
  - 실행시킬 프로그램 선택하는 스케줄링 필요
  - 프로그램의 실행 상태를 저장할 컨텍스트 정의
  - 컨텍스트 스위칭 필요
- 인터럽트 개념 도입
  - 운영체제가 I/O 장치로부터 입출력 완료를 전달받는 방법 필요
- 동기화
  - 여러 프로그램이 동일한 자원을 동시에 액세스할 때 발생하는 문제 해결 충돌 막기
- 교착 상태 해결
  - 프로세스들이 상대가 가진 자원을 서로 요청하면서 무한대기하는 교착상태 해결

시분할 다중프로그래밍(Time Sharing Multiprogramming) 운영체제

- 출현 배경
  - 다중프로그래밍 운영체제와 거의 동시에 연구 시작
  - 배치 처리와 당시 다중프로그래밍의 다음 2가지 문제점 인식
    - 비 대화식 처리방식(non-interactive processing)
    - 느린 응답시간, 오랜 대기 시간
      - 프로그램을 제출하고 하루 후에 결과 보기, 사용자의 즉각적인 대응 없음

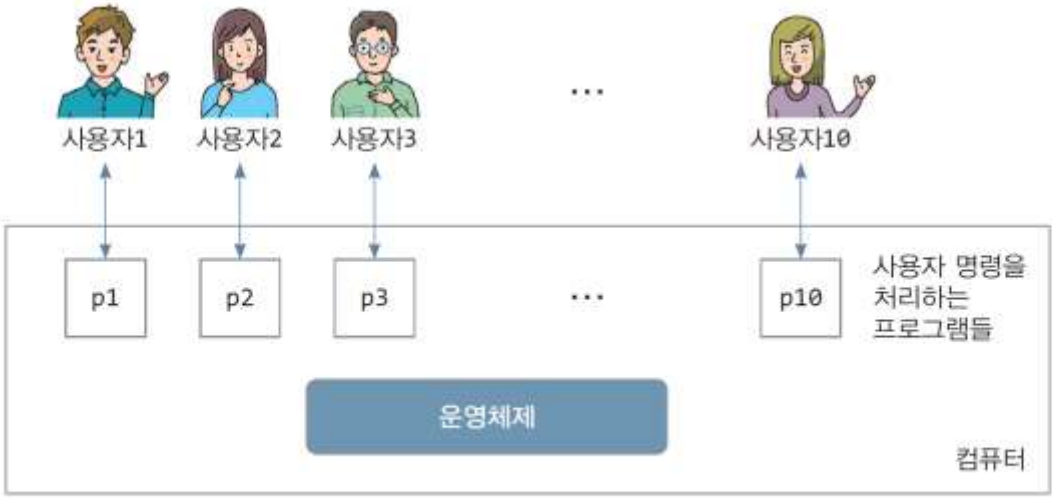
- 시분할 운영체제의 시작
  - 1959년 MIT 대학, John McCarty 교수에 의해
  - 빠른 프로그래밍 디버깅 필요
    - McCarty 교수의 당면한 문제였음
  - 사용자에게 빠른 응답을 제공하는 대화식 시스템 제안
    - 터미널이란? 키보드+모니터+전화선+모뎀으로 구성된 장치
    - 사용자는 자신의 터미널을 이용하여 메인 컴퓨터에 원격 접속
    - 운영체제는 시간을 나누어 돌아가면서 각 터미널의 명령 처리
  - CTSS(Compatible Time Sharing System) 시분할 시스템 개발
    - 1962년 MIT

시분할 운영체제를 가진 시스템

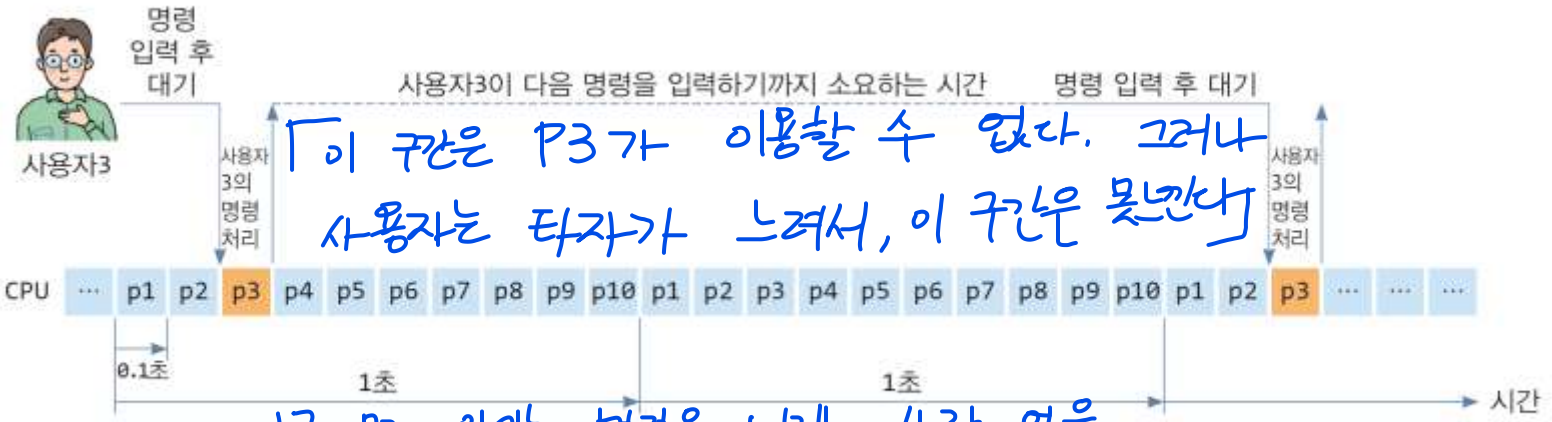


- 터미널은 모니터와 키보드, 모뎀으로 구성된 간단한 입출력 장치이며, 전화선으로 메인프레임과 연결
- 메인프레임 컴퓨터에는 터미널(사용자)마다 사용자의 명령을 받아 처리하는 프로그램 실행
- 시분할이란 각 프로그램에게 고정된 시간(time slice)만큼 CPU를 할당하여 번갈아 실행시키는 기법
- 사용자의 키 입력 속도에 비해 컴퓨터의 속도가 비교할 수 없이 빠르기 때문에 시분할 처리 가능
- 사용자가 느리기 때문에 시간을 나누어 CPU가 여러 프로그램을 실행한다고 하더라도 사용자는 응답이 늦게 온다고 여기지 않는다. 사용자는 명령을 내리기 위해 생각하거나, 이전 결과를 분석하거나, 커피를 마시거나, 화장실을 가거나 하는 등 많은 시간을 지체하기 때문이다.

시분할 시스템에서 각 프로그램(사용자)에게 0.1초 시간씩 CPU 할당하는 사례

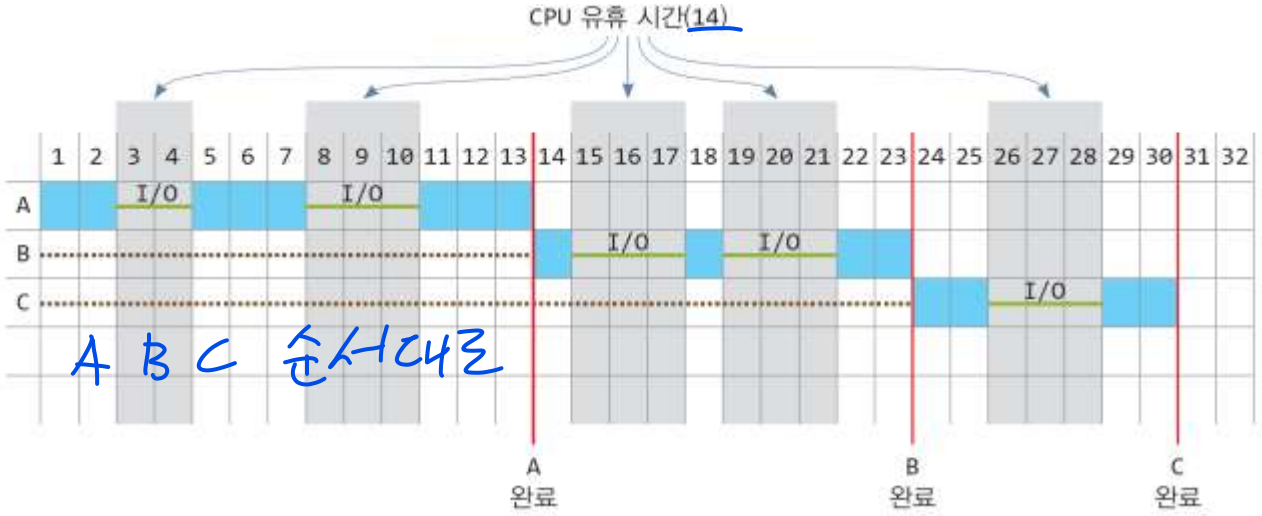


(a) 시분할 시스템에서 10명의 사용자가 원격으로 연결된 상황



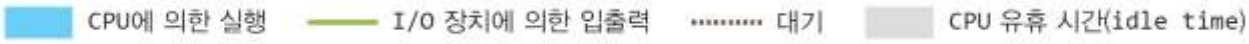
(b) 사용자 3의 명령 입력과 처리, 그리고 다음 명령을 처리하는데 걸리는 시간

탐구 1-2 배치 운영체제와 시분할 다중프로그래밍 운영체제의 실행 비교



(a) 배치 운영체제

(b) 시분할 운영체제



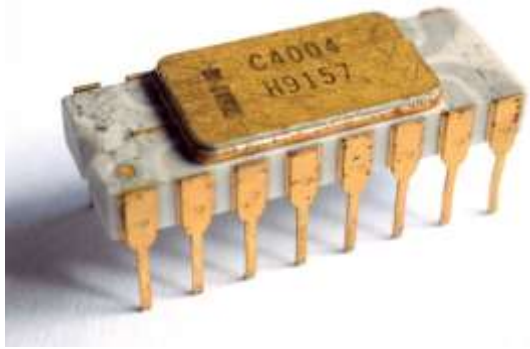
배치 운영체제, 다중프로그래밍 운영체제, 시분할 운영체제의 성능 비교

다중프로그래밍 OS 보다 '항상' 좋은건 아님

	배치 운영체제	다중프로그래밍 운영체제	시분할 운영체제
총 실행 시간	30	17	16
CPU 유휴 시간	14	1	0
CPU 활용률	16/30 = 0.53 = 53%	16/17 = 0.176 = 94%	16/16 = 100%
처리율	3/30 = 0.1작업/시간	3/17 = 0.176작업/시간	3/16 = 0.1875작업/시간

개인용 운영체제

- 출현 배경
  - 메인 프레임에서, 성능이 향상된 미니 컴퓨터 시대로 바뀜
    - 미니 컴퓨터 역시 한 방을 가득 채울 수준의 크기였음
  - 미니 컴퓨터에서의 시분할 시스템 사용의 불편함
    - 응답 속도 저하 – 많은 사용자로 인해 응답 속도 저하
    - 공간 제약 – 터미널이 있는 전산실에서만 컴퓨터 사용 가능
      - 개인이 집에서 터미널 장치를 사용한 것을 아니었음
  - 개인용 컴퓨터 필요성
    - 원격 접속 없이, 가정에서, 혼자 사용
- 개인용 컴퓨터 등장
  - 마이크로프로세서 CPU 장치 개발
    - 1971년, Intel 4004 처음 시장에 등장
    - 개인용 컴퓨터 등장
  - 메인프레임이나 미니컴퓨터에 비해 저렴, 소형
- 개인용 운영체제 등장
  - 1980년 개인용 운영체제 MS-DOS 등장



임베디드 운영체제

- 임베디드(embedded, 내장형) 컴퓨터
  - 자동차, 비행기 제어 시스템, 공장, 디지털 TV, ATM기, 네비게이터, 엘리베이터, 블루레이 플레이어를 비롯한 미디어 재생기, POS 단말기, 교통신호시스템, 셋톱 박스, 게임기, 유무선 공유기 등 가전제품이나, 산업 현장의 기계들, 상용 제품 등에 내장되어 이들 장치들의 목적을 지원하는 **소형 컴퓨터**
- 임베디드 운영체제(Embedded Operating System)
  - 임베디드 컴퓨터에서 장치들을 제어하고 작동시키는 기능을 수행하며, 장치를 제어하는 프로그램이 원활이 실행되도록 하는 목적
  - 사례
    - WinCE, 여러 종류의 임베디드 리눅스

모바일 운영체제

- 모바일 컴퓨터
  - 하드웨어의 급속한 발전으로 휴대 가능한 크기로 들고 다닐 수 있는 모바일 장치 혹은 모바일 컴퓨터 출현
  - 스마트폰, 태블릿, 스마트 시계와 같은 입는 컴퓨터(wearable computer) 등 어디에서나 휴대가능한 컴퓨터
  - 터치스크린, 블루투스 장치, 전화기, 무선네트워크 장치, GPS, 사진 및 동영상 촬영이 가능한 카메라, 음성 인식, 녹음기, 근거리 통신 장치, 적외선 장치, 지문 인식기, 배터리 등의 장치 내장
- 모바일 운영체제
  - 모바일 컴퓨터 내 장치들을 구동시키고, 이들을 활용하는 다양한 응용프로그램을 실행할 수 있도록 특별히 설계된 운영체제

운영체제의 종류

- 데스크톱 운영체제
  - PC나 노트북 등 책상에 놓고 사용하는 데스크톱 컴퓨터를 위한 운영체제
    - 개인의 문서 편집, 웹 서핑, 게임, 프로그램 개발, 음악 감상 등 범용 사용
    - 비전문가라도 사용하기 쉽고 다양한 종류의 응용프로그램을 쉽게 활용하도록 하는데 목적
  - Windows, Mac OS, Linux가 전체 시장 지배
    - Windows: 80~90%, Mac OS : 10~20%, Linux : 나머지
- 서버 컴퓨터 운영체제
  - 네트워크에 연결하고 24시간 실행되는 컴퓨터, 보안 중요
    - Unix 계열의 linux, FreeBSD, Windows Server, Mac OS Server
- 모바일 운영체제
  - 모바일 전화기, 스마트폰, 태블릿 컴퓨터 등 다양한 이동용 혹은 휴대용 장치에서 실행되도록 만들어진 운영체제, 절전과 보안 중요
    - Android, iOS, BlackBerry, Bada, Symbian, Windows Mobile
- 임베디드 운영체제
  - 임베디드 컴퓨터에서 장치들을 제어하고 작동시키는 기능
    - WinCE, 여러 종류의 임베디드 리눅스
- 실시간 운영체제
  - 실시간 애플리케이션(혹은 태스크)을 각각 정해진 데드라인(deadline) 시간 내에 처리되도록 보장하는 것을 목표
    - PSOS, VxWorks, VRTX, RT-Linux, Lynx