서강대학교 SOGANG UNIVERSITY

운영체제입문
Introduction to Operating Systems

CSW3020

담당교수 장 나 은

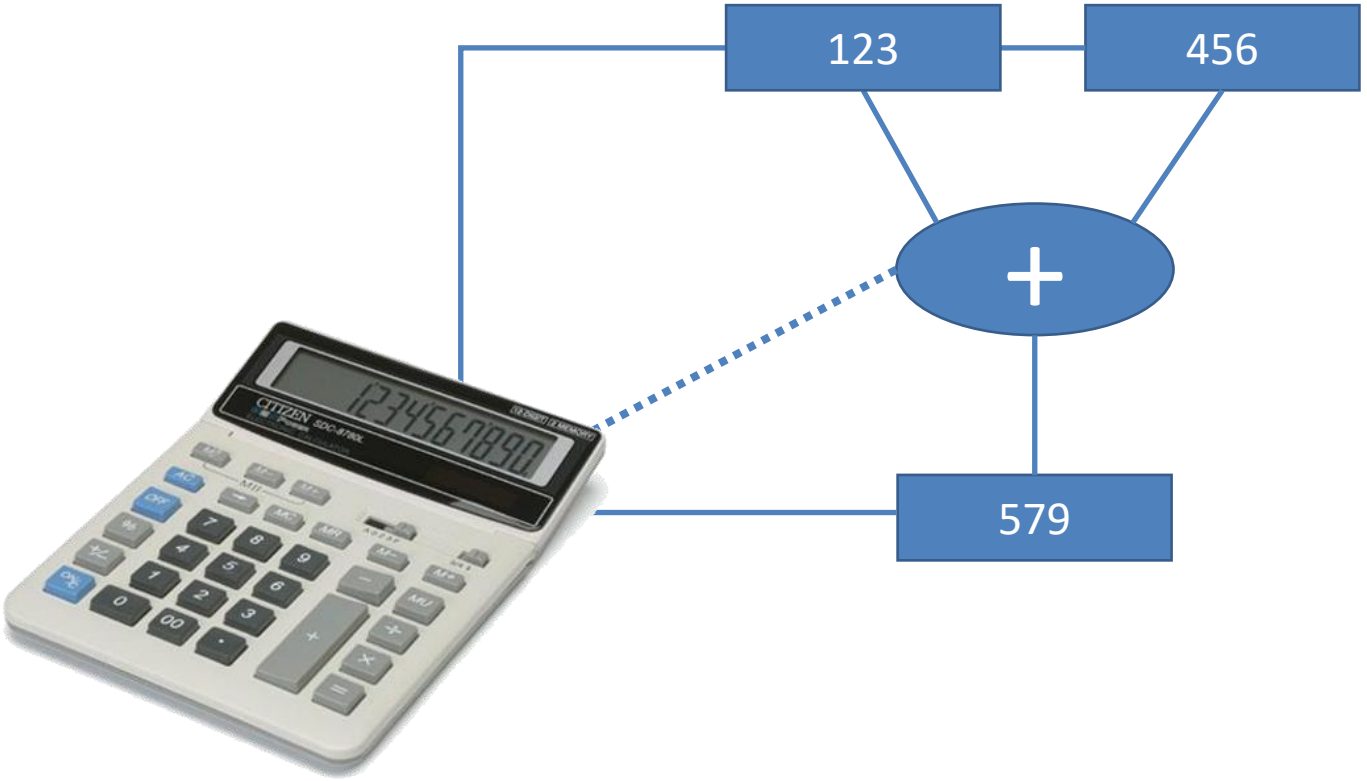
Contents

서강대학교 SOGANG UNIVERSITY

Chapter 1.
운영체제 개요

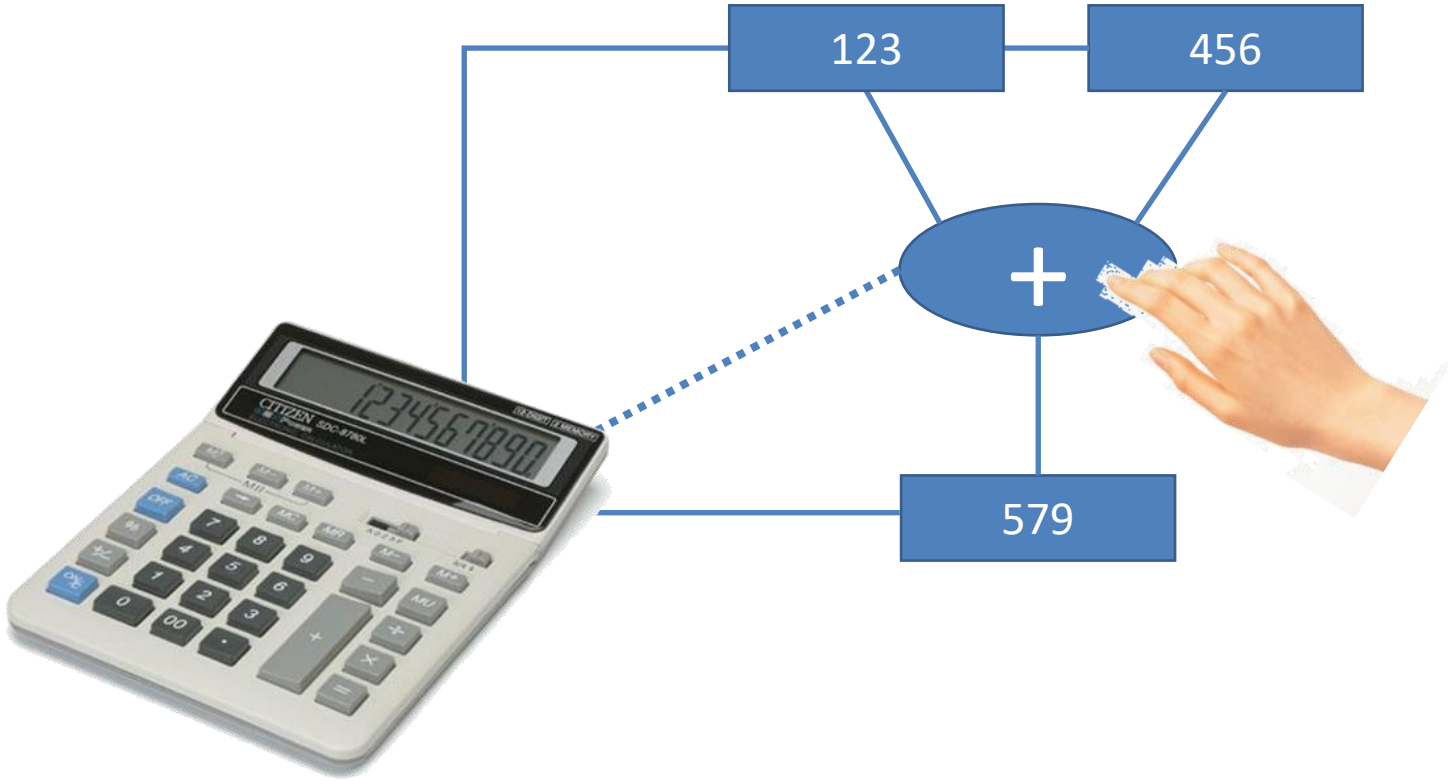
컴퓨터

- Computer => 계산기?



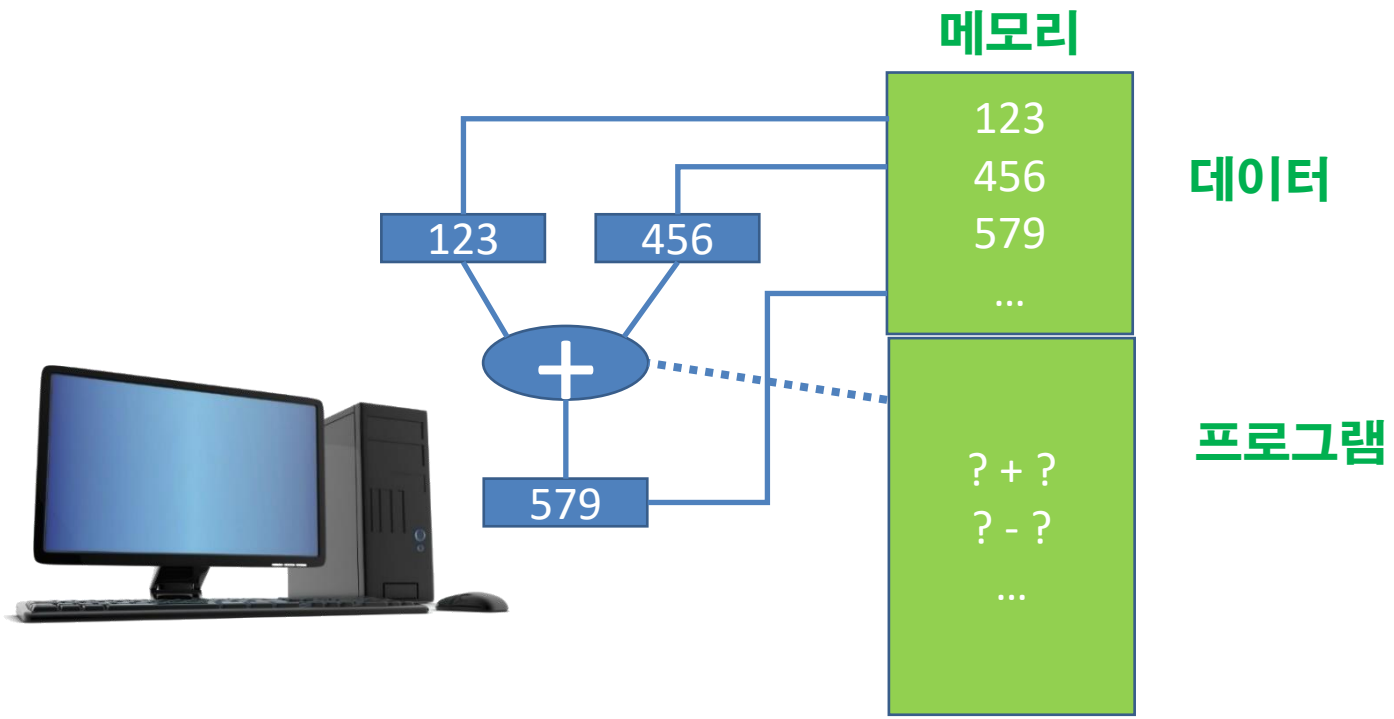
컴퓨터

- Stored Program Computer (프로그램 내장형 컴퓨터)
  - 데이터 뿐 아니라 프로그램을 메모리에 내장하여 실행



컴퓨터

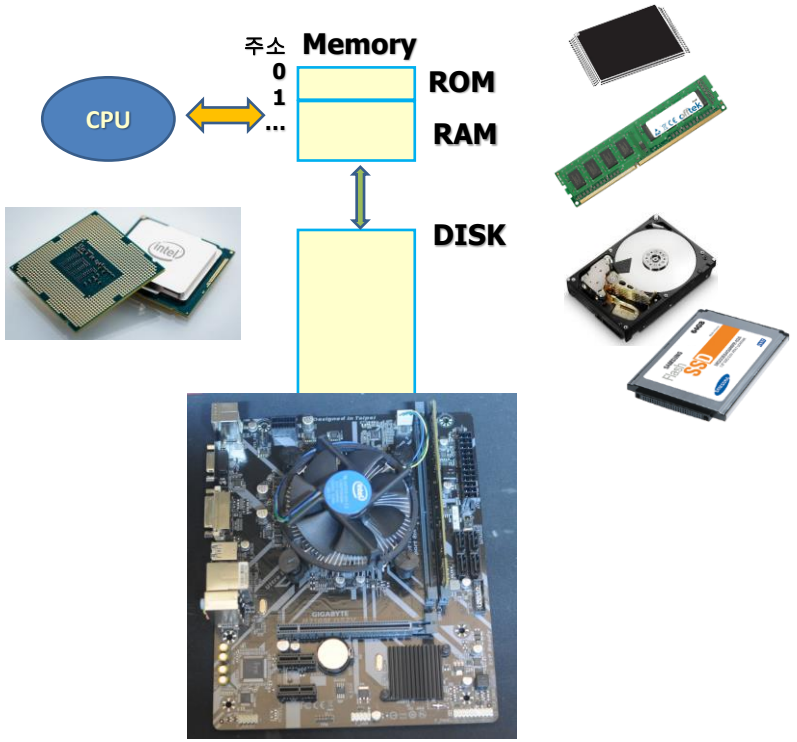
- 폰노이만의 프로그램 저장형 컴퓨터 (SPC: Stored Program Computer) 등장 => 오늘날의 실질적인 컴퓨터라 할 수 있음
  - 컴퓨터는 데이터 뿐 아니라 프로그램을 메모리에 저장하고 있다



컴퓨터 동작 메카니즘

- Stored Program Computer (프로그램 내장형 컴퓨터)
  - 데이터 뿐 아니라 프로그램을 메모리에 내장하여 실행
  - CPU, Memory, I/O장치, Disk, Display, …

- 컴퓨터가 처음 켜지면?
  - Bootstrap
  - Kernel
  - OS(System Program) Load
  - 인터럽트에 따라 실행



초기 컴퓨터란?

서강대학교 SOGANG UNIVERSITY

- 에니악(ENIAC:Electronic Numerical Integrator And Calculator) – 최초의 전자식 컴퓨터
- IBM 360 - 60년대 이후 메인컴퓨터
  - 초당 최대 34,500개의 명령을 수행
  - 메모리는 8~64 KB

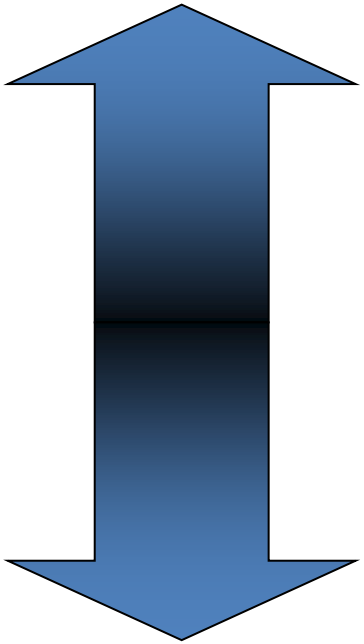
- 컴퓨터는 엄청 비싸다!
  - 프로그램은 메모리, CPU 등 리소스를 적게 사용하도록...
- 컴퓨터는 엄청 빠르다!
  - CPU가 쉬지 않고 계속 실행되도록 많은 프로그램을 함께 실행

운영체제의 출현 배경

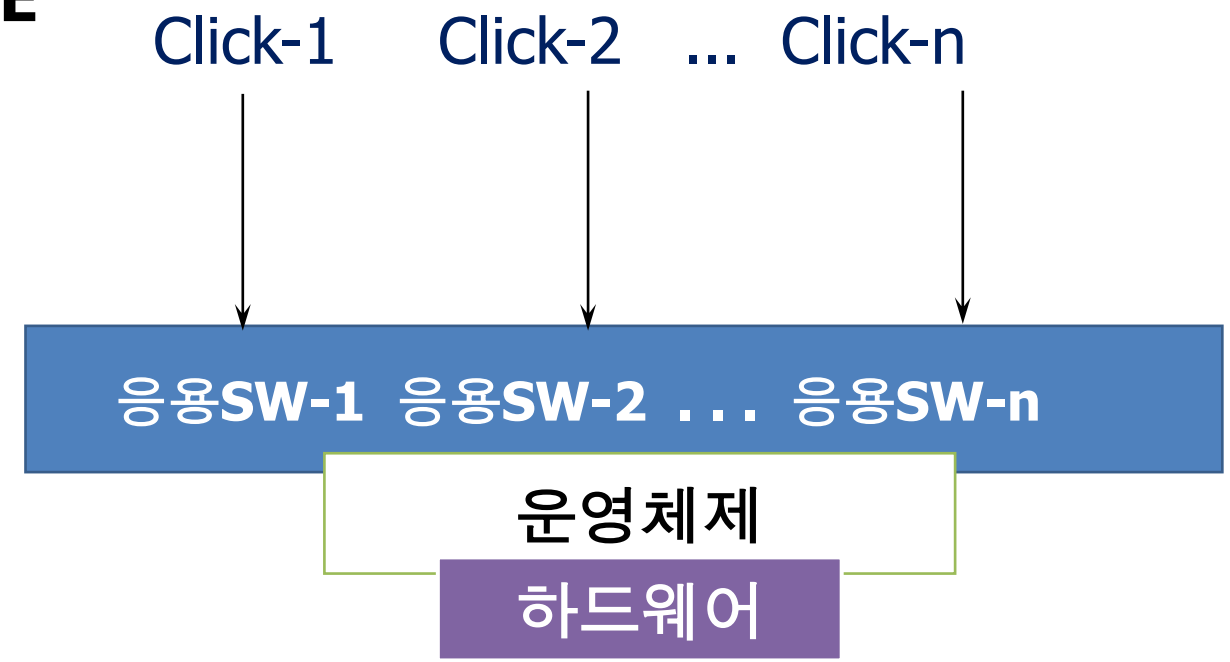
- 깡통컴퓨터: CPU + Memory, I/O
  - 어떻게 작동했나? OS가 없고 사람(Operator)에 의해…
- 일괄처리(Batch Processing)
  - 업무를 차례대로 하나씩 처리, 앞뒤로 시간낭비 (I/O 등)
- 다중프로그래밍(Multi-programming)
  - 업무처리가 중단될 때 다른업무 처리
  - 처리순서 조정 등 OS 개념 태동
- 시분할(Time-Sharing) 처리
  - CPU가 워낙 빨라 시간을 쪼개어 여러 업무를 동시에 처리
  - 프로그램들은 마치 단독으로 서비스 받는 느낌?
- 응용: 실시간, 다중처리, 클라우드 …

Motive!

CONVENIENCE
(편의성)



EFFICIENCY
(효율성)



❖ 토의 – 편리하다는 뜻은? 효율적이라는 뜻은?

운영체제의 목적

- 사용자의 편의성 추구
- 자원의 효율적인 관리(자원 할당자)
- 다양한 입출력 장치의 운영과 통제

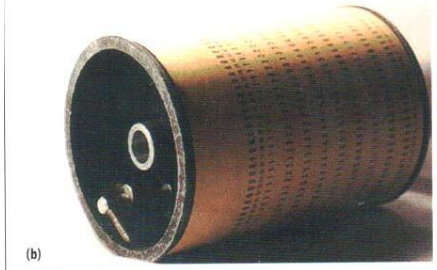
편의성 추구 vs 효율성 추구

- 무엇이 우선이면 좋을까요? 혹은 무엇이 우선이었을까?
- 운영체제의 변천 과정 속에서 파악해 봅시다.

초기 시스템

- 프로그램은 주로 10진수/8진수로 된 기계어이고,
  주로 자기테이프나 카드천공기(Card Punch)로 카드에 기록
- 프로그래머(또는 사용자)가 Sign-Up Sheet에 자기가 원하는 시간을 표시(예약)
- 프로그래머가 콘솔이나 스위치를 이용하여 모든 작업 수행
  - 이전 사용자의 프로그램 제거
  - 테이프나 카드덱의 내용 적재
  - 콘솔을 통하여 프로그램 수행/디버깅
  - 상대주소 아닌 절대주소 사용
  - 카드덱(Card Deck) 형태의 라이브러리 사용
- 무엇이 문제인가?
  - 활용도 비효율 (Sign-Up Sheet – 남는 시간 발생)
  - 준비 시간 과다 (Tape나 Punch-Card로 프로그램 적재)

(a)



(b)

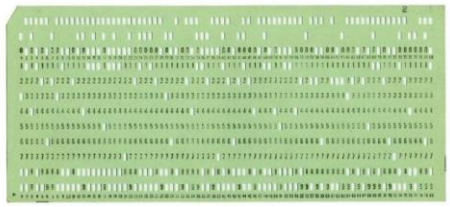
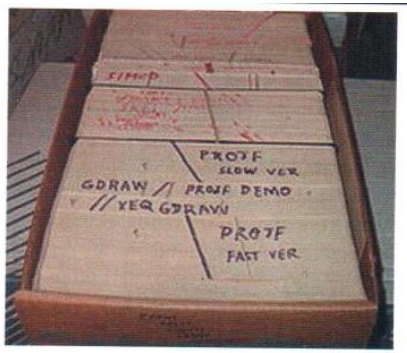


Figure 1. (a) An IBM 026 card punch machine, (b) program drum, and (c) standard 80-column punch card. The program drum is partially visible in the card punch machine, behind the vertical slot above the keyboard. (IBM 026 image courtesy of Sellam Ismail, Vintagetech.com; program drum image courtesy of Gabriel Robins, University of Virginia; and punch card image courtesy of Douglas W. Jones, University of Iowa.)

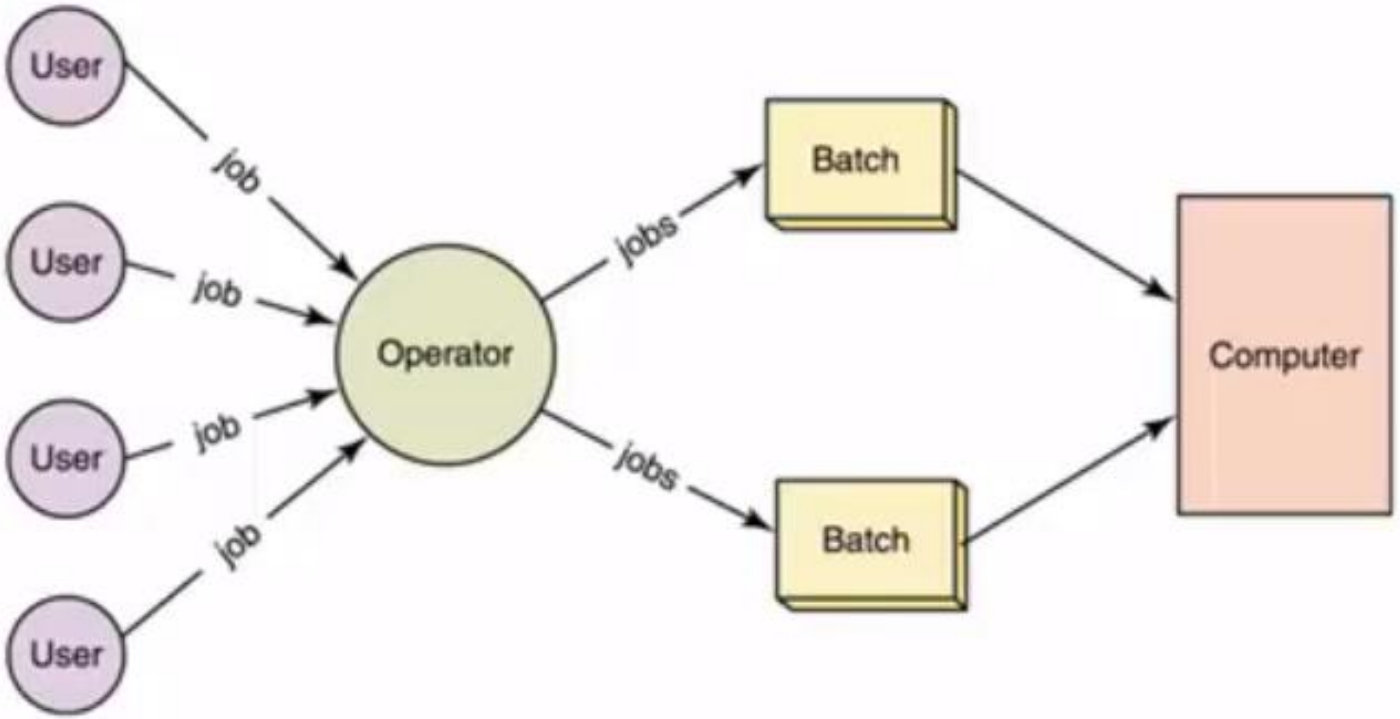


(a)

(b)

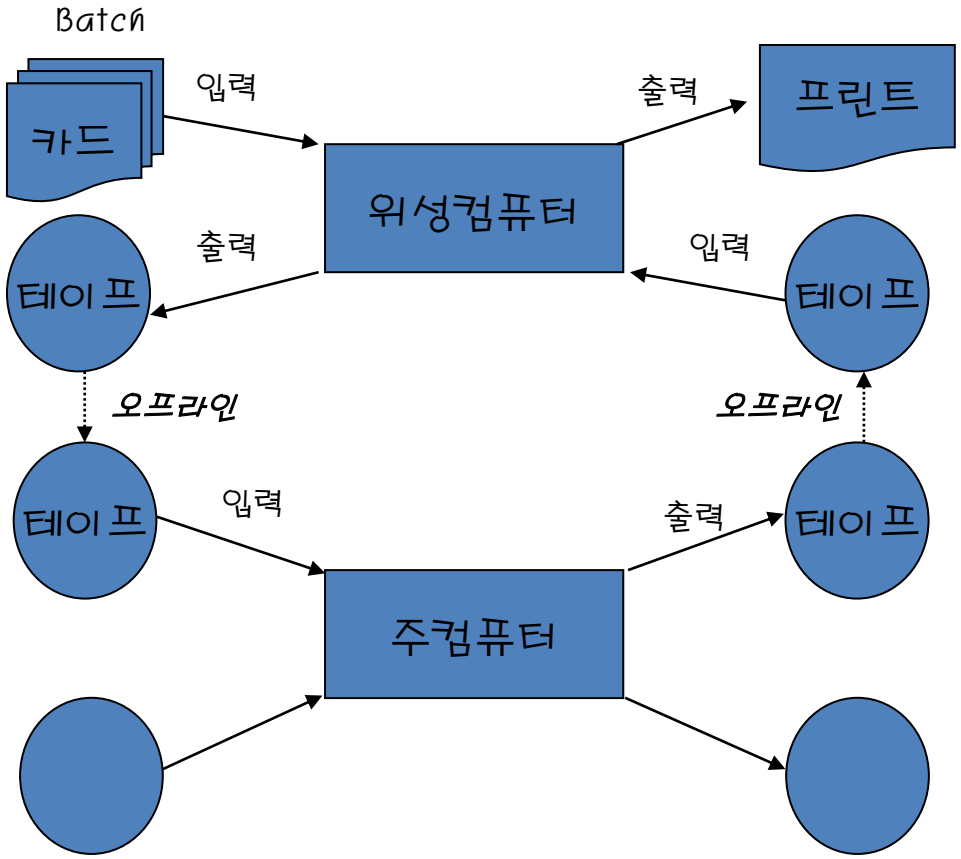
Figure 2. Early databases. Data centers often contained (a) an open box of punch cards and (b) magnetic tape that an unauthorized person could easily alter. (Punch card image courtesy of Arnold Reinhold; magnetic tape image courtesy of Gabriel Robins, University of Virginia.)

초기 일괄처리 시스템

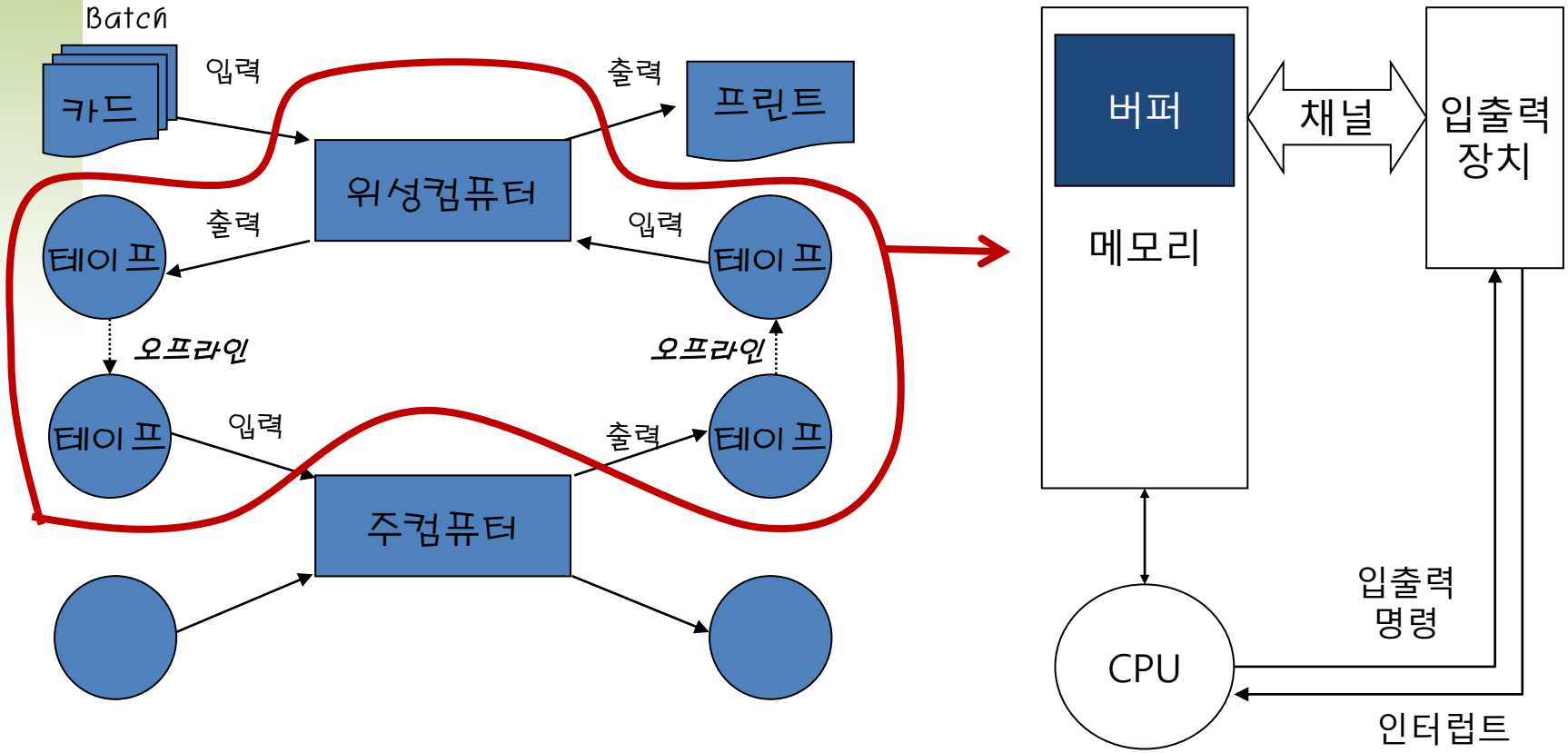

User
User
User
User
job
job
job
job
Operator
jobs
jobs
Batch
Batch
Computer

초기 일괄처리 시스템

- 운영자(operator)를 고용하여 사용자의 작업을 전문적으로 대행
- 사용자들이 요구하는 비슷한 작업들을 함께 묶어서 배치(batch)로 처리
- 배치는 별도의 오프라인 카드리더나 테이프에 수록되고, 처리 결과도 별도의 오프라인 테이프를 통해 프린터로 출력됨
- 배치를 취급하는 것을 사람 (운영자) 수행

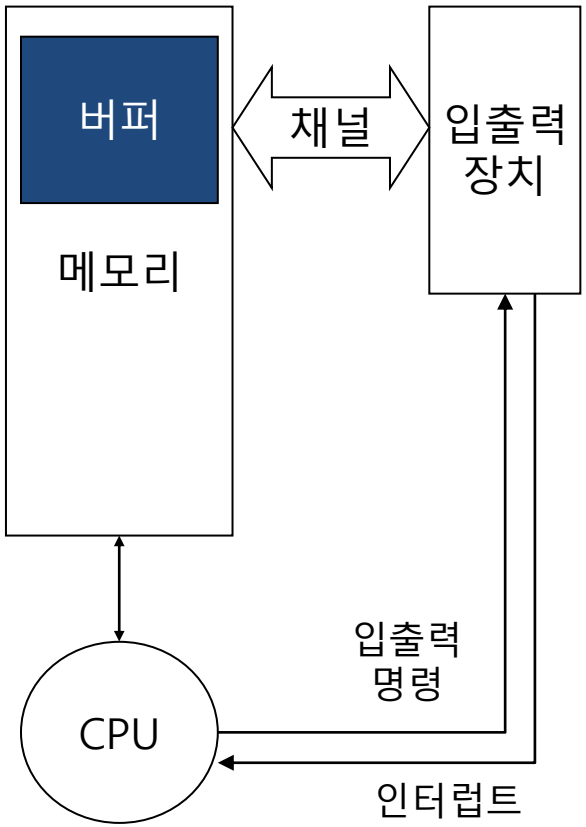


일괄처리 시스템


Batch
카드
입력
출력
프린트
위성컴퓨터
출력
입력
테이프
테이프
오프라인
오프라인
테이프
입력
출력
테이프
주컴퓨터
버퍼
메모리
채널
입출력
장치
CPU
입출력
명령
인터럽트

일괄 처리 시스템

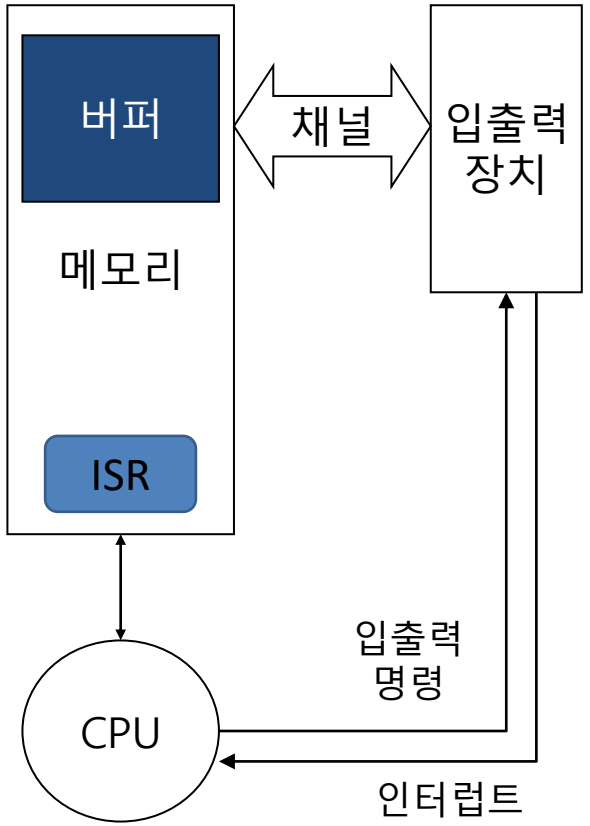
- 채널
  - 명령구조, 약간의 레지스터, 입출력 제어장치로 구성
  - CPU와 함께 메모리를 공유
  - CPU로부터 명령을 받아 CPU와 독립적으로 입출력 실행 (단, 메모리 사이클 경쟁 제어 필요 → DMA)
- 버퍼
  - CPU와 채널 입출력의 병렬 수행을 위하여 데이터 버퍼를 사용
  - 연산하는 동안 읽거나, 쓰는 것이 가능하게 되어 입출력 대기시간을 없앰



일괄 처리 시스템

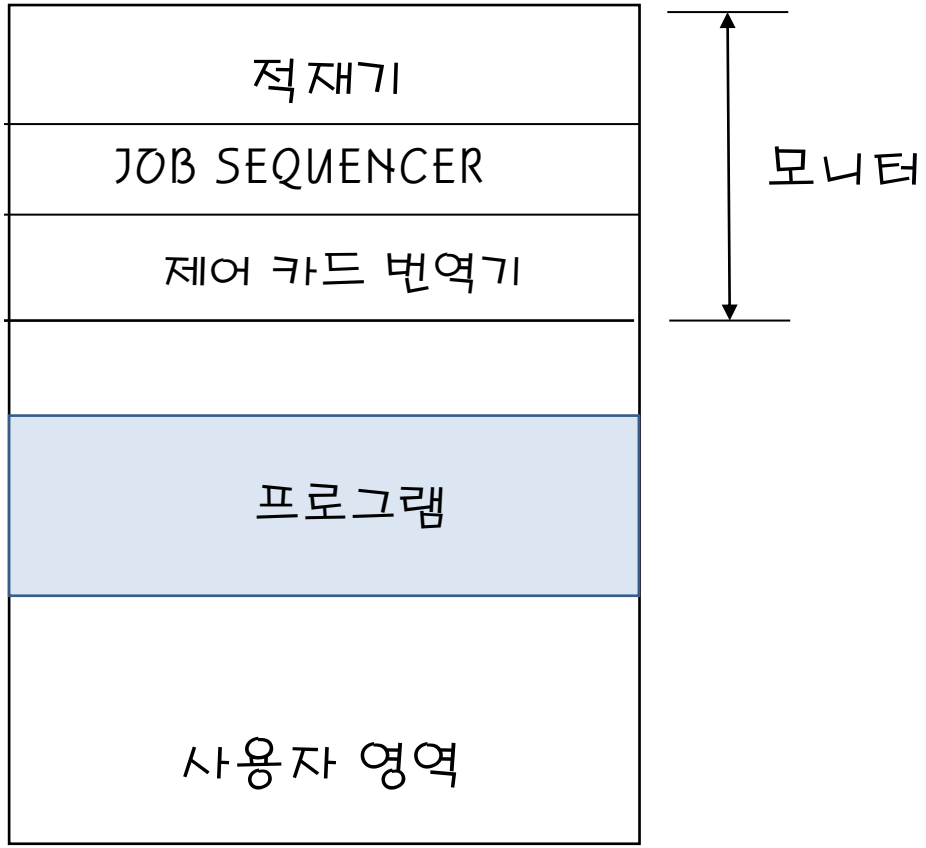
인터럽트

- 사용자 프로그램이 아니라 하드웨어에 의하여 자동으로 메모리 특정 부분에 있는 함수(ISR, Interrupt Service Routine)를 호출하는 개념
- 입출력의 완료와 예외동작 처리 (파일종료,테이프 끝, 패리티 오류 등)
- 채널을 통한 입출력 버퍼링을 CPU와는 독립적으로 수행



일괄 처리 시스템

- 상주모니터(Resident Monitor)
  - 인터럽트 처리기 또는 입출력 관리자를 메모리에 영구적으로 상주시켜야 할 필요성 대두
  - 작업 제어 명령어, 적재기, 작업 순서 제어기를 상주시켜 컴퓨터의 운영을 좀더 자동화 시킴



일괄 처리 시스템

서강대학교 SOGANG UNIVERSITY

- 보호
  - 잘못된 사용자 프로그램이 상주모니터 영역에 접근하여 덮어쓰는 일이 빈번히 발생함에 따라 상주모니터 보호의 필요성 대두
    - 모든 프로그램이 입출력을 직접 하지 않고 상주모니터의 루틴을 호출하도록 하고, (→ 시스템 콜 태동!)
    - 미리 작성된 테이블에 기록된 허용치를 참조하여 각 연산(instruction) 수행 시 메모리 접근 범위를 제한

일괄처리 시스템

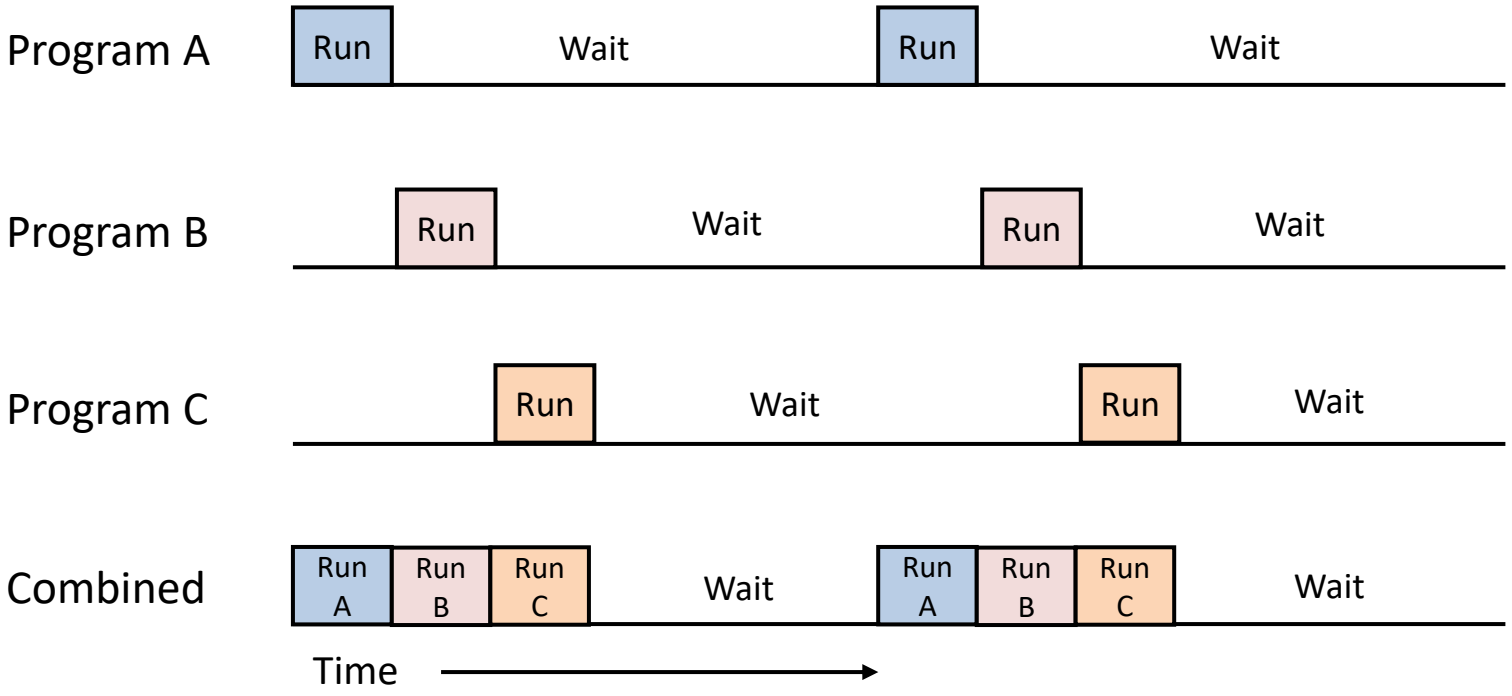
- 장점
  - 하나의 작업이 CPU를 독점하므로 해당 작업으로 볼 때는 처리 속도가 가장 빠름
  - 사용자와의 대화가 필요하지 않은 응용 프로그램 수행에 활용 가능
    - 수치 계산, 대용량 데이터 처리 등
- 단점
  - 작업이 적재된 후 일정기간(반환시간)이 지난 후에야 그 결과를 볼 수 있기 때문에, 대화형 응용 프로그램에는 사용하기 어렵다.

만약 프로그램 수행 중에 사용자로부터의 Y/N 확인이 필요한 프로그램이 있다고 가정할 때, 만약 사용자가 식사를 하러 갔다면 어떤 일이 벌어질까요?

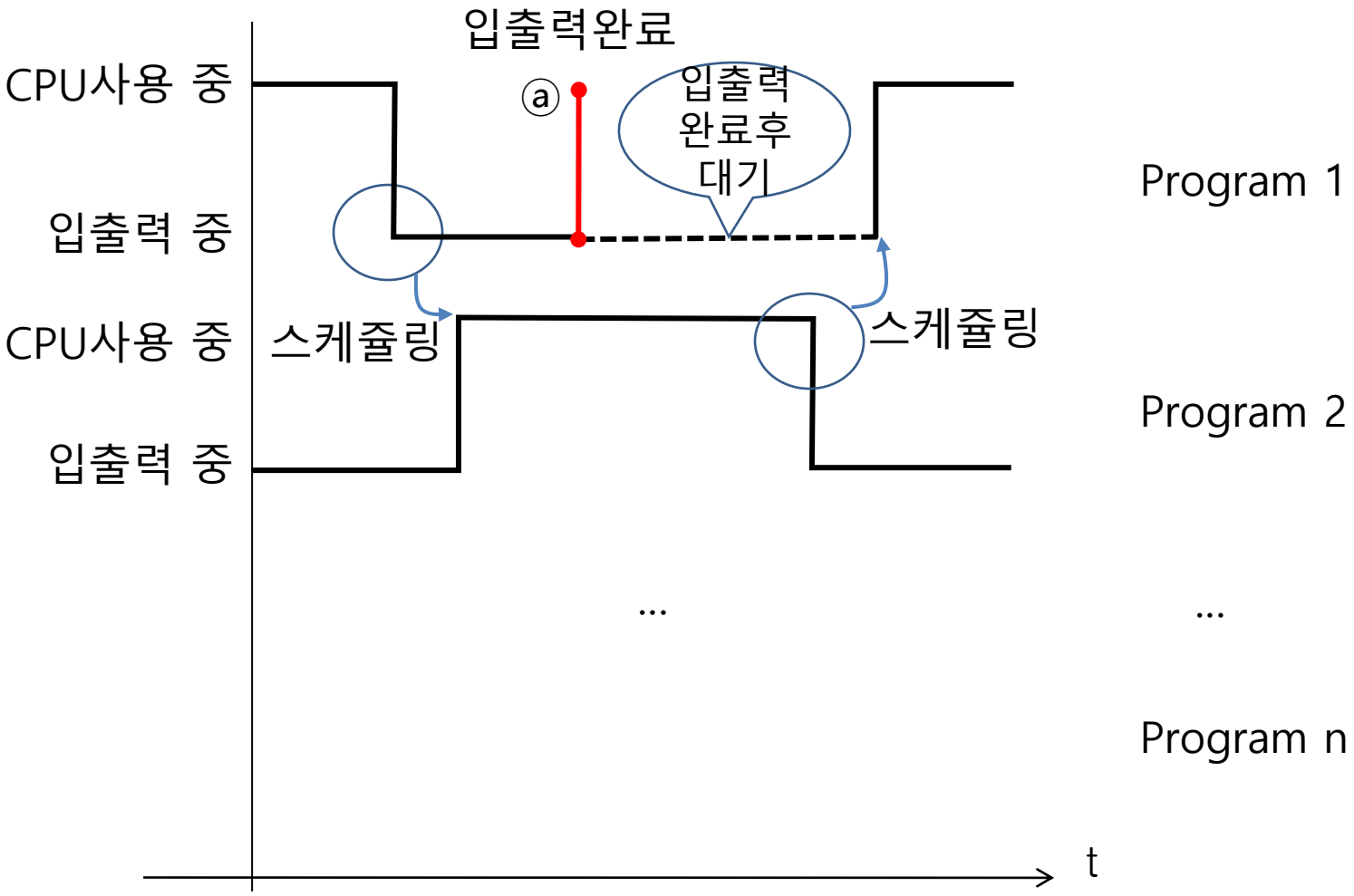
Multiprogramming(다중프로그래밍)

- 한 시점에 여러 프로그램을 사용자 영역에 탑재
  - 시스템에 들어오는 모든 작업은 일단 작업 풀(디스크 사용)에 적재됨
  - 작업풀 내의 작업은 운영체제의 정책에 따라 선택되어 메모리에 탑재
- 탑재된 작업 중 하나를 선택하여 실행
- 실행하다 보면 궁극적으로 입출력이 일어남(키보드 입력을 기다리거나 출력이 끝나기를 기다리는 상태에 도달)
- **한 프로그램이 입출력을 하는 동안 다른 프로그램에 CPU를 할당하여 실행**

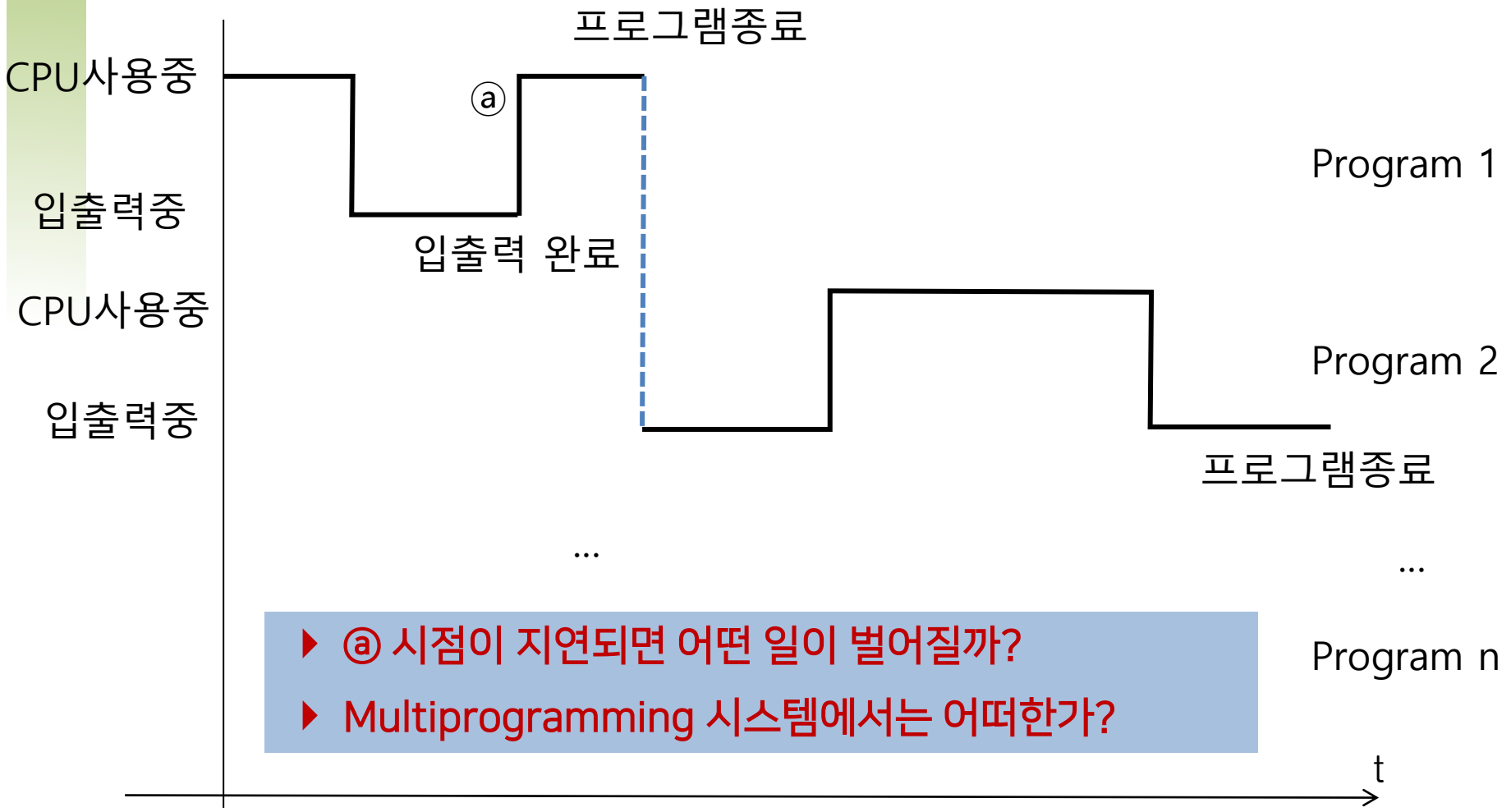
Multiprogramming(다중프로그래밍)



Multiprogramming(다중프로그래밍)



일괄처리의 경우라면…



- ⓐ 시점이 지연되면 어떤 일이 벌어질까?
- Multiprogramming 시스템에서는 어떠한가?

Multiprogramming(다중프로그래밍)

- 모티브가 된 중요한 힌트
  - 대부분 프로그램의 실행 시간에서 CPU의 사용 시간은 극히 일부분이고 나머지는 입출력 시간이다.

- N개의 프로그램의 실행 시간을 각각 t1, t2, …tn 이라 할 때,
  - 일괄처리 또는 uniprogramming 을 사용할 경우 전체 시간은

    t1+t2+…+tN

  - 반면, 다중프로그래밍인 경우 **대략**

    max(t1, t2,…,tN)

Multiprogramming

- 한 프로세스의 입출력 시에 다른 프로세스를 처리할 수 있게 되므로, CPU가 항상 일을 하고 있게 됨

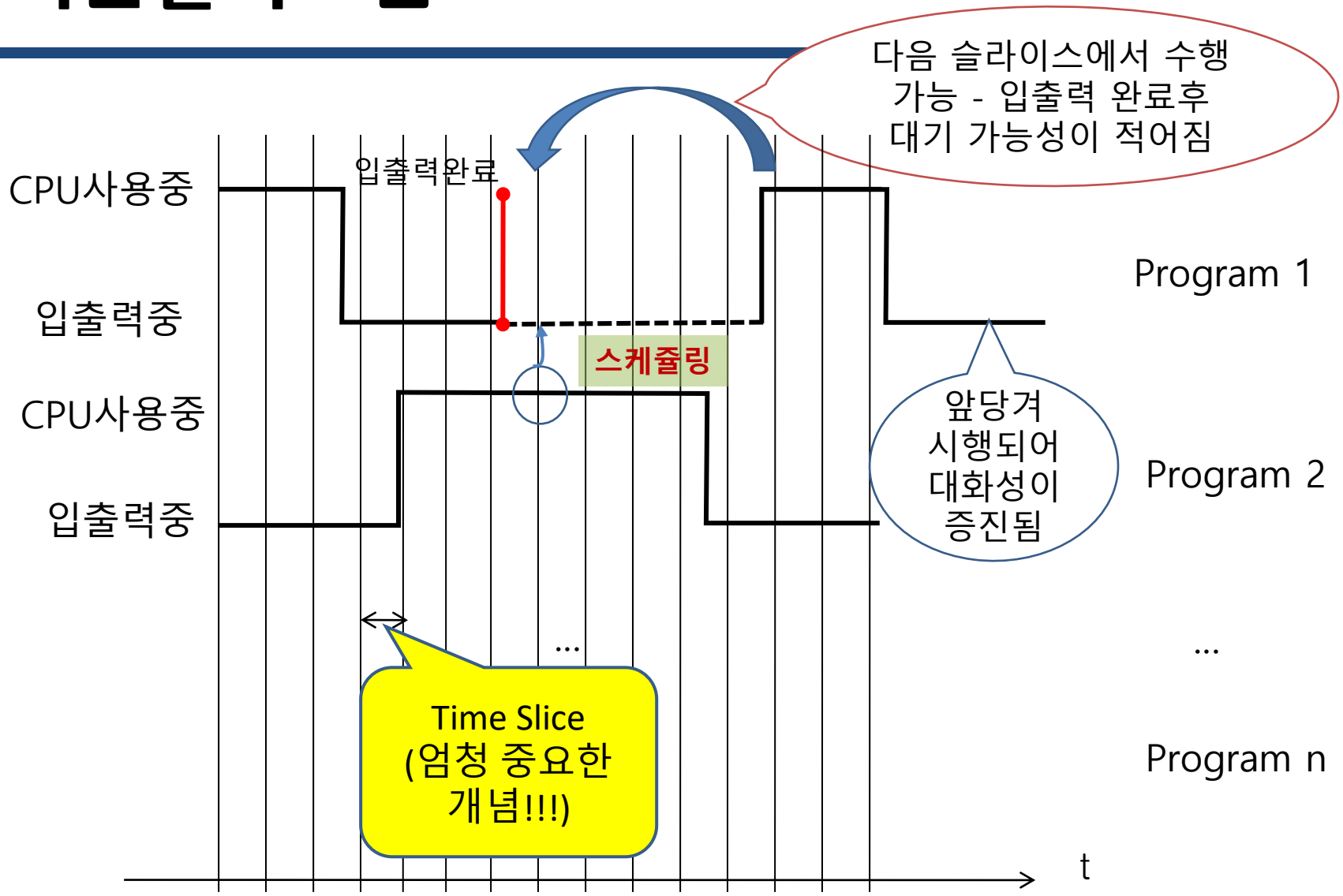
- 또한, 디스크를 이용한 Buffering 과 Spooling으로 입출력과 CPU 수행의 중복 정도를 높일 수 있게됨
  - Input Spooling은 Job Scheduling에 사용
  - Output Spooling은 산발적인 프린트 출력을 모아서 프로세스가 끝난 후에 출력

- 새롭게 대두되는 이슈
  - Job Scheduling
  - 메모리 경영 – 여러 작업이 메모리 상에 존재

시분할 시스템

- 일괄처리 시스템의 결정적 단점
  - 일정 시간이 지나야만 결과를 출력을 받아 볼 수 있음
  - 작업이 실행되고 있는 동안 사용자와 작업간의 대화가 불가능
    - 대화식 프로그램 개발(컴파일→수행→디버깅의 순환)이 불가능
    - 대화식 응용 프로그램(예: 워드프로세서 등) 수행에 부적합
- 다중 프로그램의 한계
  - 프로그램 입출력 시간을 CPU의 효율성 증진에 활용
  - 사용자와의 빈번한 대화성(편리성) 증진에 한계
    (@시점 이후 대기시간이 다른 프로그램으로부터 영향을 받음)
- 1960년 MIT의 CTSS가 시초
  - CTSS(Compatible Time Sharing System)
- 하드웨어 여건 부족으로 1970년대에 상용화

시분할시스템


다음 슬라이스에서 수행 가능 - 입출력 완료후 대기 가능성이 적어짐
입출력완료
CPU사용중
입출력중
CPU사용중
입출력중
스케쥴링
앞당겨 시행되어 대화성이 증진됨
Program 1
Program 2
...
Program n
...
Time Slice (엄청 중요한 개념!!!)
t

시분할 시스템

- 대화식 응용프로그램의 경우 CPU 사용 시간은 소량인 반면 빈번한 대화(입출력)을 요구

- 시분할 시스템으로 빈번한 입출력이 가능하게 됨에 따라 대화성이 증대가 실현됨
- 여러 사용자가 개인용 모니터를 통하여 한 시스템에 동시에 연결하여 동시 사용이 가능해짐
  - 운영체제는 각 사용자 사이를 재빠르게 전환함으로써 사용자가 마치 컴퓨터를 독점하고 있는 듯한 착각을 만들어 줌

시분할 시스템

- 여러 사용자의 터미널과 시스템 간에 온라인 통신 마련하여 동시 접속
  - 입력: 키보드
  - 출력: 개인용 모니터(초기에는 RS232 시리얼 라인을 이용한 흑백 CRT 모니터)
  - 주요 프로그램: Vi, Emacs, 포트란컴파일러, C컴파일러 등
- 여러 사용자가 운영체제나 프로그램에 직접 명령을 주고 즉시 응답을 받을 수 있음

- **CPU의 효율성과 사용자에게 빠른 응답시간을 동시에 추구**

시분할 시스템

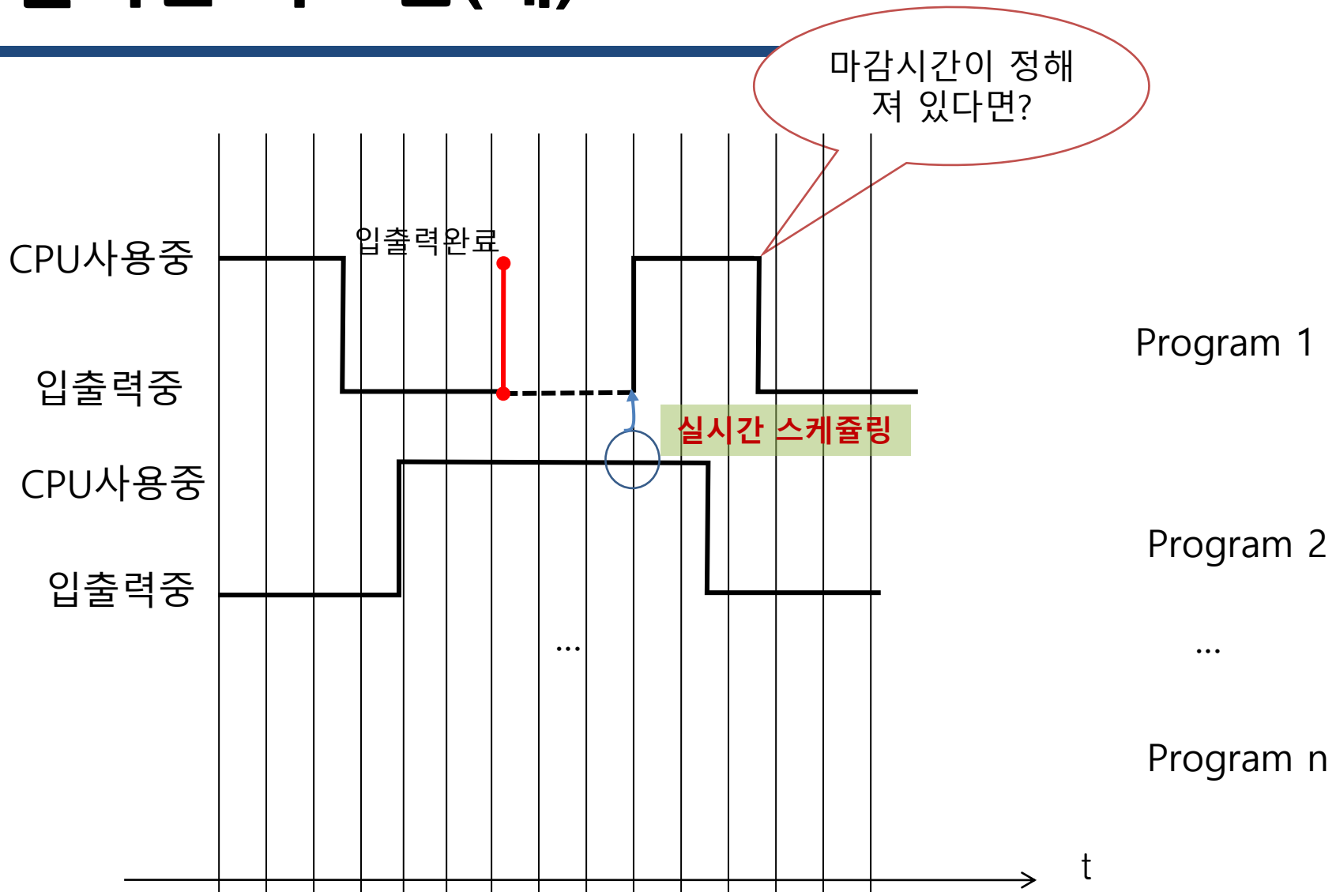
서강대학교 SOGANG UNIVERSITY

- 다중 프로그래밍의 진일보로서 보다 복잡해 짐
  - 타임 슬라이스
    - 사용자에게 빠른 응답시간을 주기 위하여 보다 작은 단위로 CPU 시간 나누어 할당(Time Slice)
  - 가상메모리
    - 여러 프로그램의 동시 적재로 메모리 부족을 야기. 가상메모리 기법 탄생으로 실제 메모리보다 더 큰 프로그램의 수행도 가능해짐
  - 온라인 파일 시스템, 디스크 스케쥴링, CPU 스케쥴링, 프로세스간 통신, 동기화 및 교착상태 처리 등 다양한 이슈 대두

실시간 시스템

- 프로세서의 동작이나 자료의 흐름에 대한 엄격한 시간 제한적(deadline) 연산이 필요한 경우 사용
  - 예: 센서로부터 직접 입력 받아 기계장치를 제어해야하는 경우: 공장 생산 라인, 과학실험 제어, 산업제어, 항공기/미사일 제어, 로봇, 동영상 처리 등
- Hard Real-time Vs. Soft Real-time
  - Hard Real-time(경성 실시간성):
    - 데드라인 위반사건 발생시 재앙적 사건이 발생 – 무기 제어, 원자력 발전소.
    - 데드라인 준수에 대한 보장성과 예측성이 필수
    - 가상메모리 등 고급 기능이나 모든 부가적 자원 관리를 최소화하고 하드웨어와 밀착된 S/W 개발
  - Soft Real-time(연성 실시간성):
    - 데드라인은 존재하지만 위반에도 크게 심각하지 않은 경우 – 동영상, 멀티미디어 동기
    - 실시간 작업과 일반 작업 간의 우선순위 제어로 해결하는 경향

실시간 시스템(예)


마감시간이 정해져 있다면?
입출력완료
CPU사용중
입출력중
실시간 스케쥴링
CPU사용중
입출력중
...
Program 1
Program 2
...
Program n
t

컴퓨터의 I/O 동작

- I/O 장치와 CPU는 동시 실행 (Concurrent)
- 특정한 장치마다 각각의 장치제어기(Device Controller)가 있다
- 각 장치제어기는 자체 버퍼메모리를 가지고 있다
- CPU는 주기억장치와 장치제어기의 버퍼 간 데이터 이동 실행
- I/O는 장치와 제어기 버퍼간에 발생
- 장치제어기가 I/O 작업을 마치면 Interrupt 신호를 통해 CPU에게 알려 줌

인터럽트의 공통 기능들

서강대학교 SOGANG UNIVERSITY

- 인터럽트가 발생하면 CPU는 수행중인 프로세스를 대기상태로 만들고 Interrupt Service Routine (or Interrupt Handler: IH)를 호출, IH는 Interrupt Vector를 통하여 해당 처리루틴을 호출한다.
- Interrupt Handler는 운영체제의 한 부분
- A trap (or an exception) is a software-generated interrupt caused ether by an error or a user request

인터럽트 종류

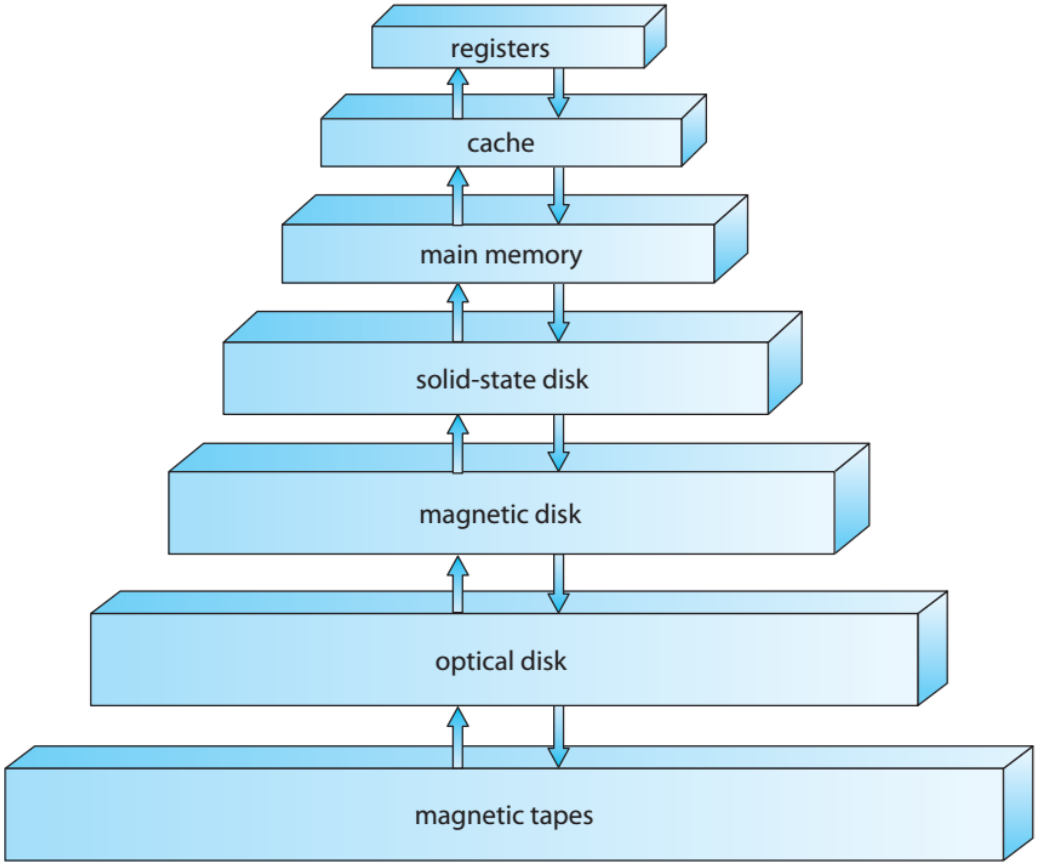
- Program (Trap or Software Interrupt)
  - arithmetic overflow
  - division by zero
  - execute illegal instruction
  - reference outside user's memory space
  - by user's request (e.g. system call or monitor call)
- Timer
- I/O
- Hardware failure

스토리지(Storage) 구조

- 레지스터: CPU 내에 정보저장을 위한 고정된 메모리
- 주기억장치(Main Memory): CPU에서 프로그램 실행 중 직접 접근이 가능하며 프로그램 주소공간을 가지고 있음. 컴퓨터가 종료되면 기억 내용이 모두 지워짐. RAM
- 보조기억장치(Secondary Memory): 주기억장치의 확장 메모리로 사용 전 주기억 장치에 적재되어야 함. 컴퓨터가 종료되어도 내용이 보존 됨. Magnetic Disk(HDD), SSD(Solid State Disk), CD/DVD 등

스토리지 계층(Hierarchy)

- Storage systems organized in hierarchy.
  - Speed
  - Cost
  - Volatility



메모리 캐싱(Cashing)

- 빈번하게 사용되는 데이터를 보다 고속의 메모리에 담아놓고 사용
- Cache Management 정책 필요
- 메모리 계층의 한 레벨로 들어갈 수 있음. 하위 중복된 메모리 내용과 동일한 내용으로 유지되어야 함
  - 캐시메모리를 잘 사용하면 데이터 접근 시간을 획기적으로 줄일 수 있음 (자주 사용되고 변경이 적은 데이터가 유리)
  - 잘못 사용되면 캐시데이터 갱신 시간이 많아져 더 느려질 수도 있음 (자주 사용되지 않으면서 변경이 자주 발생하면 캐시 하위 데이터와의 일관성 유지를 위한 작업이 발생 함)

Direct Memory Access 구조

- 인터럽트 구동방식의 I/O는 적은양의 데이터 전송은 문제가 없으나 디스크 I/O등 많은양의 입출력에 너무 잦은 인터럽트 발생 등으로 오버헤드가 발생
- DMA는 CPU 개입없이 메모리와 버퍼 간 블록 단위의 큰 데이터를 한번에 전송. 즉, 매 바이트 전송마다 인터럽트를 발생시키는 대신 한번의 인터럽트로 블록 전체를 전송 함
- 장치제어기가 전송 작업 수행동안 CPU는 다른 작업을 수행

End of Chapter 1