Operating Systems

1. History and Concepts

Hyunchan, Park

http://oslab.chonbuk.ac.kr

Department of Computer Science and Engineering

Chonbuk National University

DON'T KEEP CALM AND HAVE FUN

Contents

1. 컴퓨터의 기원: Hand-operated System
2. Batch System
    1. Automatic job sequencing
    2. Spooling Batch System
3. Multiprogramming
4. Timesharing
5. Multitasking
6. Other systems and Lineage of well-known operating systems

컴퓨터의 기원

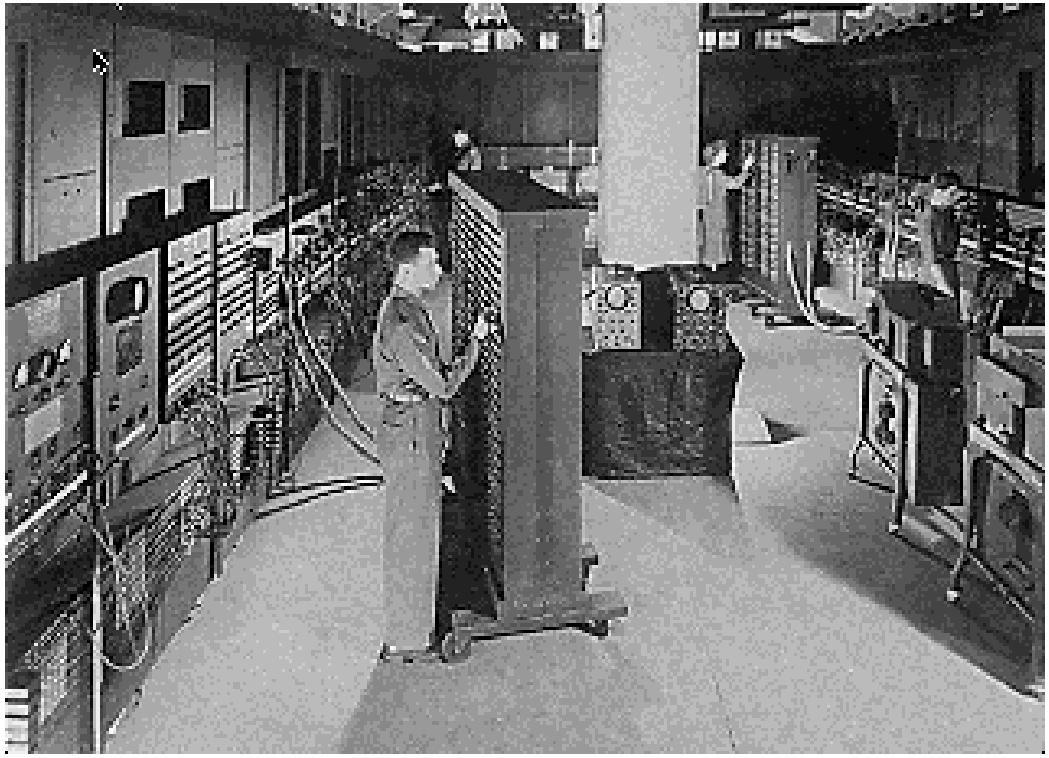
- Compute + er

- 2차 세계대전
  - 암호 해석
  - 미사일 탄도 분석
  - 물리 계산

OS history

- 급격한 하드웨어의 변화와 OS의 진보
  - 약 75년간의 HW 변화 (1945-2019)
    - 500 Flop/s (ENIAC) → 143.5 PFlop/s (Summit, IBM)
      - Flop/s: 초당 부동 소수점 연산
      - Kilo, Mega, Giga, Tera, Peta, Exa, Zetta, Yotta
    - 매년 약 1.5배 빨라짐
  - OS의 변화
    - Batch → Multiprogramming → Timesharing → Graphical UI → Smart Devices

Dawn of time
ENIAC: 1946



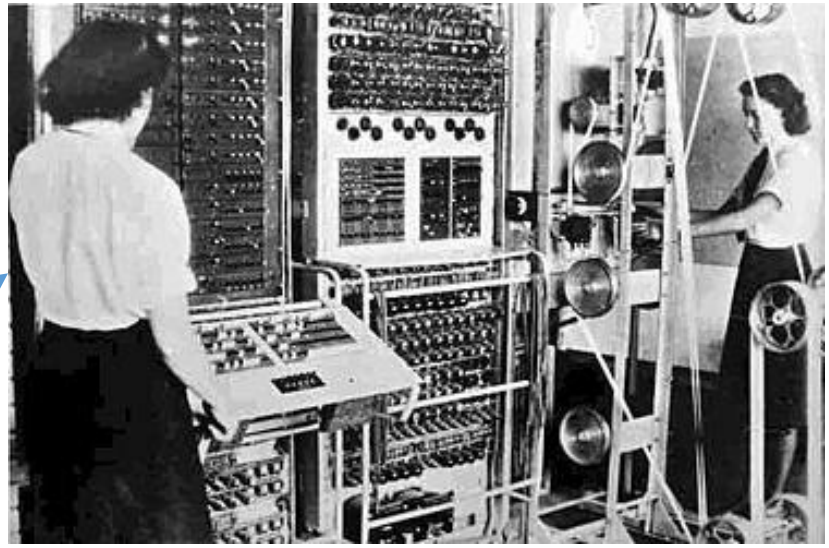
- “The machine designed by Dr. Eckert and Mauchly was a monstrosity. When it was finished, the ENIAC filled an entire room, weighed thirty tons, and consumed two hundred kilowatts of power.”

Dawn of time
Colossus: 1941? , ABC: 1942

- Colossus
  - 2차 세계 대전 당시, 독일의 암호문을 해독하는 영국군에서 제작
- ABC
  - Atanasoff-Berry Computer (John Vincent Atanasoff and Cliff Berry)
  - 진공관을 이용해 간단한 이진법 기반 계산과 Boolean 논리 연산을 수행하는 전기 컴퓨터
- The first electric programmable computer?
  - Many people think that it is the ENIAC
  - But officially, ABC is (legally by USA Judgement 1973)
  - However, it is the Colossus actually

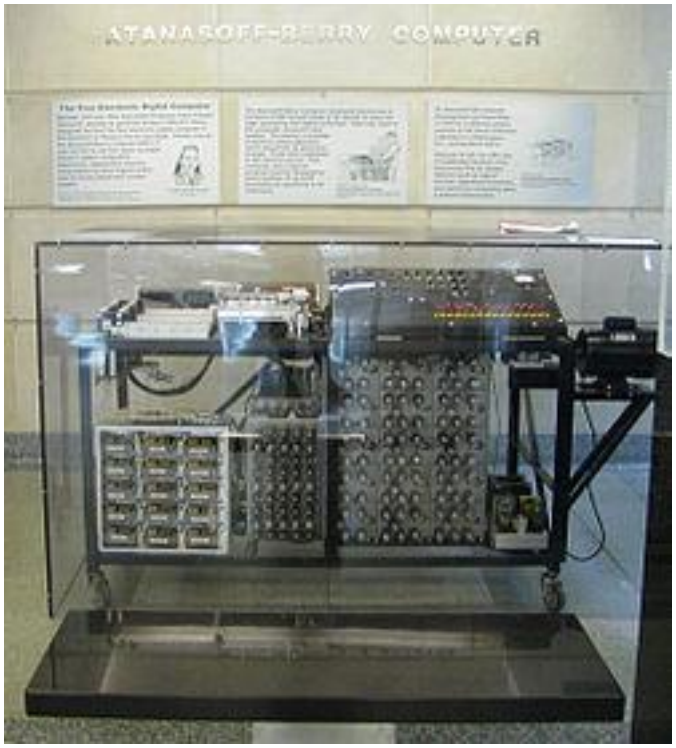
Dawn of time
Colossus: 1941? , ABC: 1942

Colossus Computer



Is she an operator?

Atanasoff–Berry computer

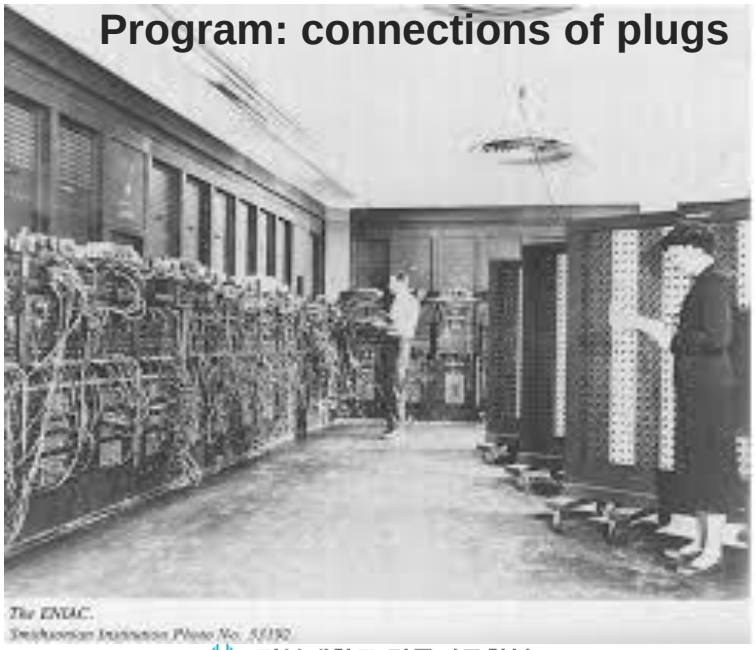


Hand-operated System

- 1950년대 초반
  - 당시 컴퓨터는 현재의 컴퓨터에 비해 매우 원시적임
  - 프로그램은 기계적인 스위치를 이용하여 1bit 단위로 컴퓨터에 입력되어 실행
    - 진공관 기반 – 커다란 방 하나를 차지하는 크기, 많은 열 방출

Hand-operated System

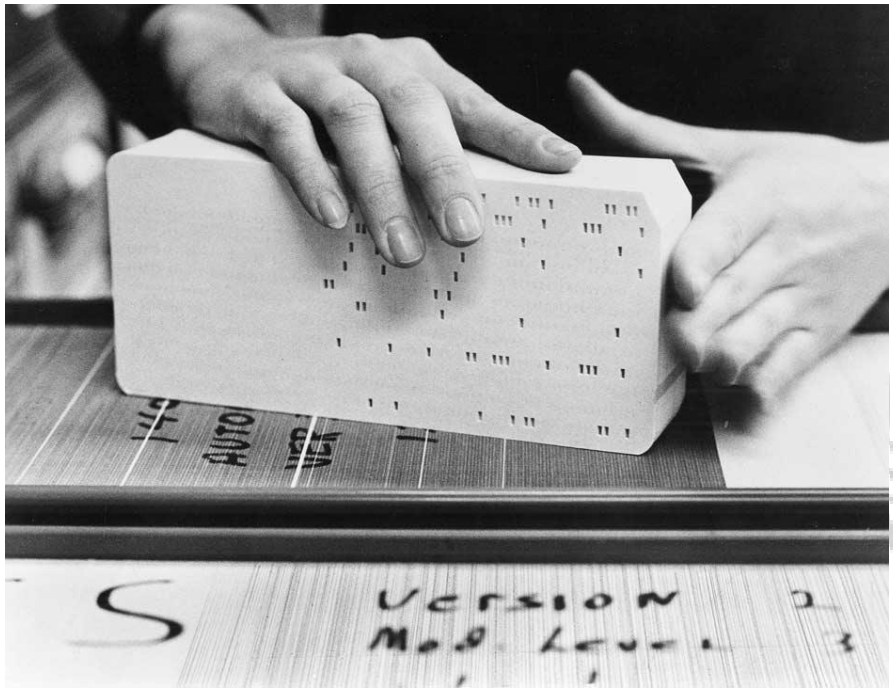
- 1950년대 중반
  - 플러그 보드(Plug-board)에 와이어링을 통해 컴퓨터의 기능을 제어
  - 모든 프로그램은 기계어로 쓰여짐
  - 프로그래밍 언어 및 운영체제라는 존재가 없음
    - 영속적인 저장장치가 없음 – 매번 프로그램 다시 입력



Hand-operated System

- 1960년대 초반
  - 펀치카드가 등장
  - 프로그래밍한 카드로 컴퓨터 구동
    - 플러그 보드 대체

Program: punch card



Mainframe

- 중앙집중형 대형 컴퓨터
  - 사용자들은 터미널을 이용해서 접속
  - HW, SW 결합된 형태로 공급
  - 일괄 처리 (batch), 시분할 (time sharing) 지원
- IBM 시스템 360 (1964)
  - 개발에 50억 달러 투자
  - 1965년 세계 최초 항공사 온라인 예약 시스템
  - 1966년 미국 의료보험 카드 온라인 처리
  - 1968년 우리나라 조사통계국 인구조사결과 분석

IBM system 360: the very first mainframe

Mainframe – 일괄처리(Batch)

- Business machinery로써 쓰이면서 가치가 발생
  - 계산을 하는데 주로 사용되기 시작함
  - 교량(Bridge) 설계

- 일괄 처리- 아주 단순한 OS개념
  - 일단 시작한 "job"은 끝나야 다음 job이 수행됨
  - Punch card를 제출하면, 메모리에 적재, 수행의 순서로 진행
  - 결과를 받기까지 중간에 사람의 개입 불가 (No user interaction)
  - 사람이 job scheduling을 수행 (문제점 1: job 전환에 오랜 시간 걸림)

- 문제점 2: CPU는 빈번히 idle 상태로 전환됨
  - 기계적인 I/O 장치와 전기 장치인 CPU사이에 현격한 속도 차가 존재

Automatic job sequencing – 좀 더 나은 OS

- 사람의 관여 없이 여러 개의 프로그램을 컴퓨터에서 순차적으로 실행

- 이전 작업이 종료되자 마자 다음 작업을 실행하기 때문에, 일괄처리(batch) 보다 성능이 향상됨
  - 일괄처리
    - 사람이 직접 스케줄링
  - Automatic job sequencing
    - 스케줄링을 담당하는 소프트웨어에서 프로그램 실행

- I/O에 의해 CPU가 유휴 상태로 전환되는 문제는 해결X

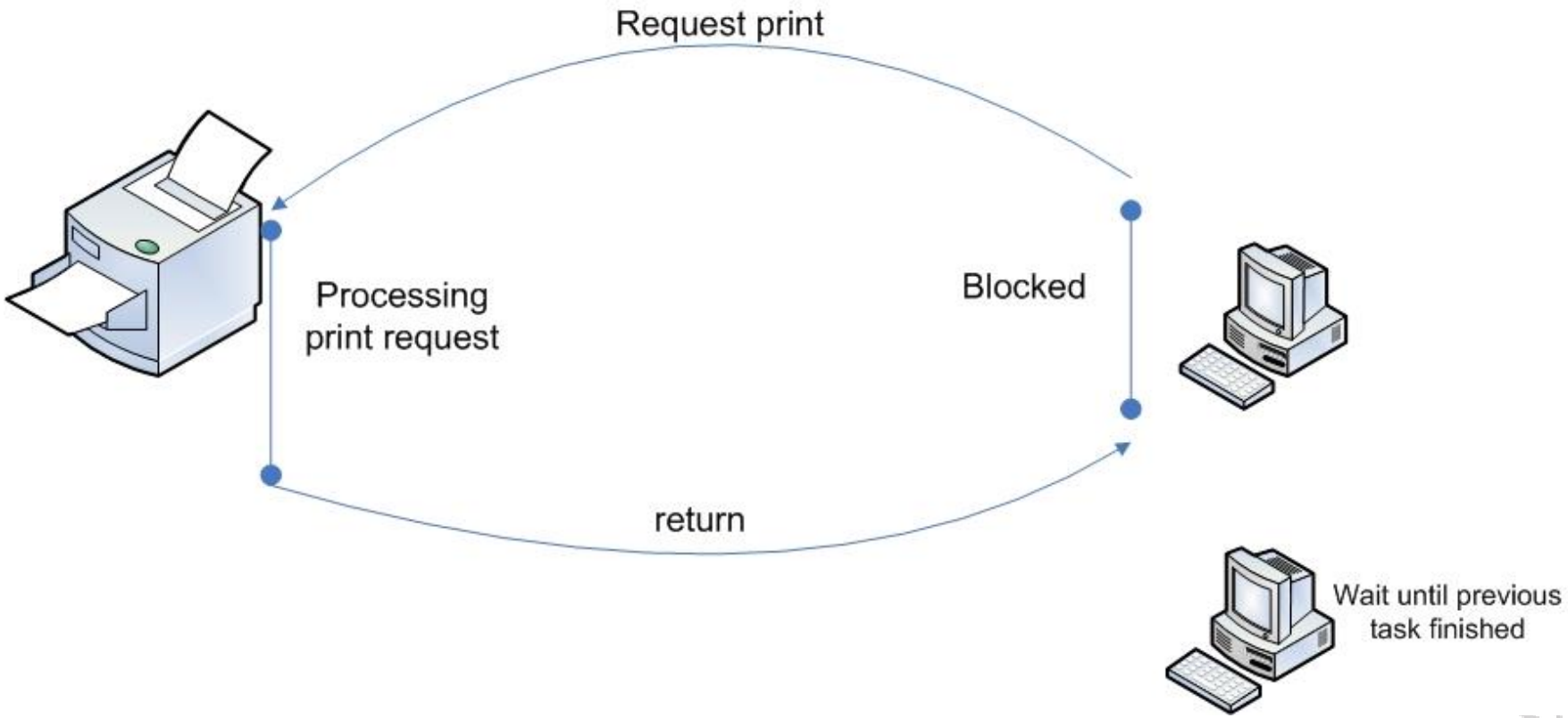
문제점 1: Job switching

- 해결방안: Automatic job sequencing

- 사람의 관여 없이 여러 개의 프로그램을 컴퓨터에서 순차적으로 실행
  - 필요 기술: 멀티 프로그래밍 (multi programming)

- 이전 작업이 종료되자 마자 다음 작업을 실행하기 때문에, 일괄처리(batch) 보다 성능이 향상됨
  - 일괄처리
    - 사람이 직접 스케줄링
  - Automatic job sequencing
    - 스케줄링을 담당하는 소프트웨어에서 프로그램 실행

문제점 2 - I/O in the middle of job



CPU idling in batch system

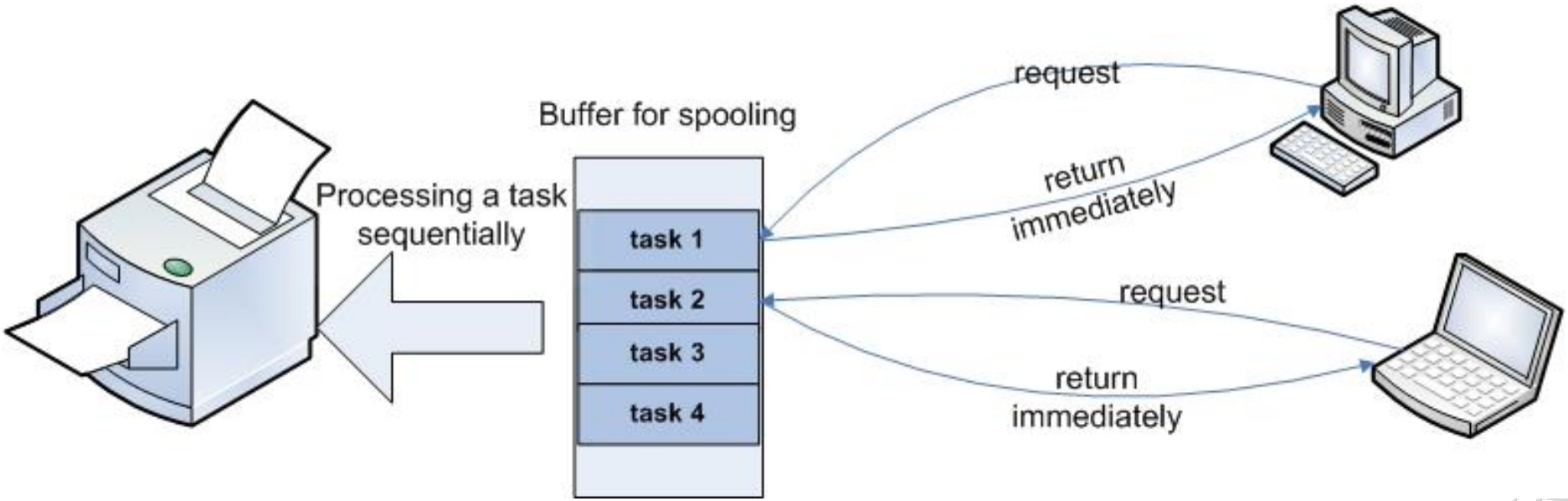
문제점 2 – 해결방안: Spooling

- Simultaneous Peripheral Operation On-Line

- I/O와 computation을 동시에 진행할 수 있음
  - 예) 프린터 spooling
    - 인쇄할 문서를 디스크나 메모리의 버퍼에 로드
    - 프린터는 버퍼에서 자신의 처리 속도로 인쇄할 데이터를 가져옴
    - 프린터가 인쇄하는 동안 컴퓨터는 다른 작업을 수행 할 수 있음

- Spooling을 통해서 사용자는 여러 개의 인쇄 작업을 프린터에 순차적으로 요청할 수 있음.
  - 이전 작업의 종료를 기다리지 않고, 버퍼에 인쇄 작업을 로드하여 자신의 인쇄 작업을 요청함

Spooling 예시



< Spooling Batch System >

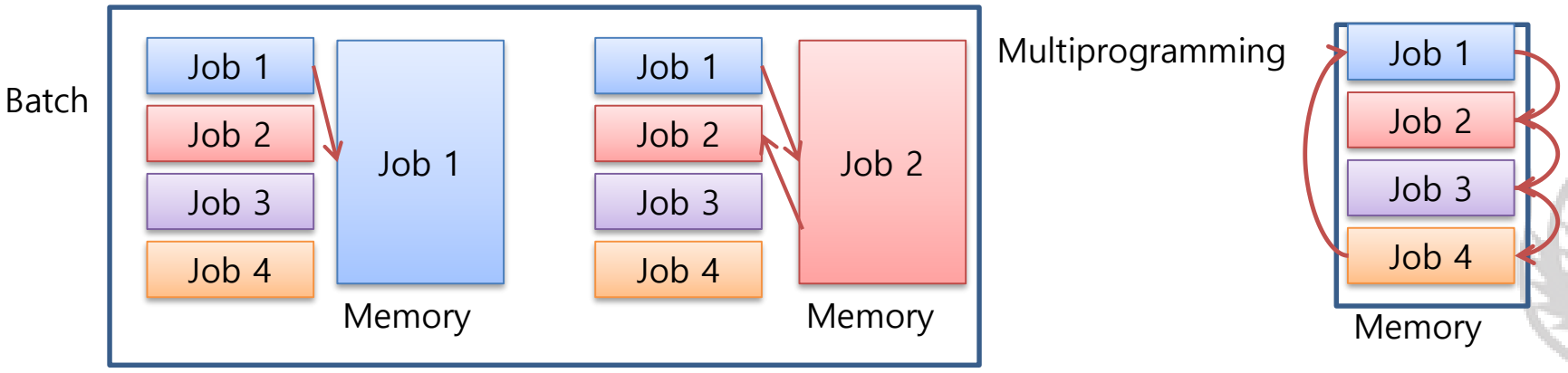
Spool 문제점

- IO 처리를 요청해두고, CPU 작업은 재개할 수 있다는 장점
  - 프린팅, 디스크 기록
  - Asynchronous 작업에는 적합함

- 그러나 CPU 작업이 IO 처리 결과에 의존적일 경우에는?
  - Synchronous 작업
    - 일반적으로 read() 작업이 해당함
    - If (fgetc() == 'Y') then CLASS DISMISSED else NOT
  - 스풀링의 도움을 받을 수 없고, 작업을 중단해야 함

Multiprogramming

- 2개 이상의 작업을 동시에 실행
  - 운영체제는 여러 개의 작업을 메모리에 동시에 유지
  - 현재 실행중인 작업이 I/O를 할 경우 다음 작업을 순차적 실행
  - 스케줄링 고려사항 – first-come, first-served
- 장점: CPU 활용률 극대화
  - CPU가 쉬는 시간이 없음



Issues with multiprogramming

- 다른 job이 수행되기 위해서는 현재 수행되는 job이 I/O를 해야 함 - voluntary yield 에 의존
  - 의도적으로 I/O 하지 않거나, 마지막에 몰아서 하면?

- 공평성을 유지할 필요 발생
  - 누구나 컴퓨터를 오래 많이 쓰고 싶어한다

- High priority로 수행할 필요도 생김

- 단순한 automatic job scheduling으로는 해결 안됨

Timesharing

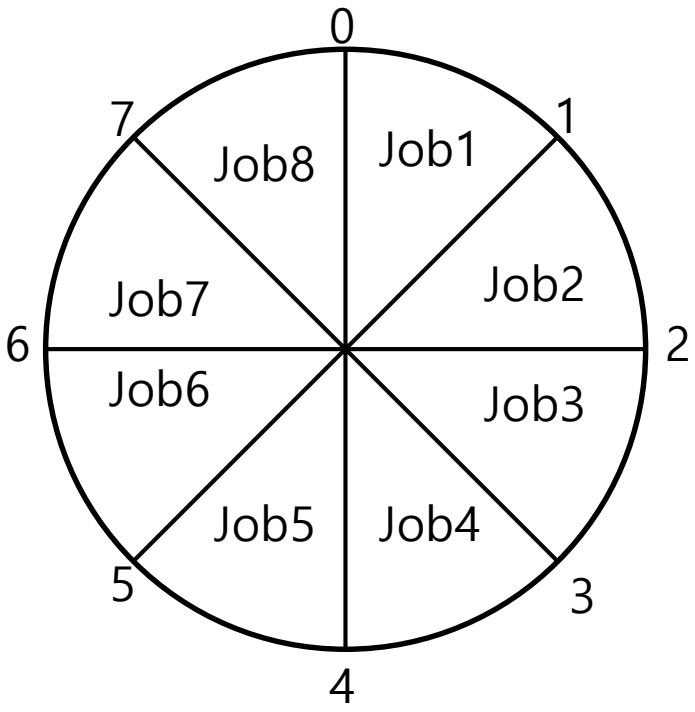
- CPU의 실행 시간을 타임 슬라이스(time slice)로 나누어 실행
  - 10ms (mill-seconds, 밀리초)

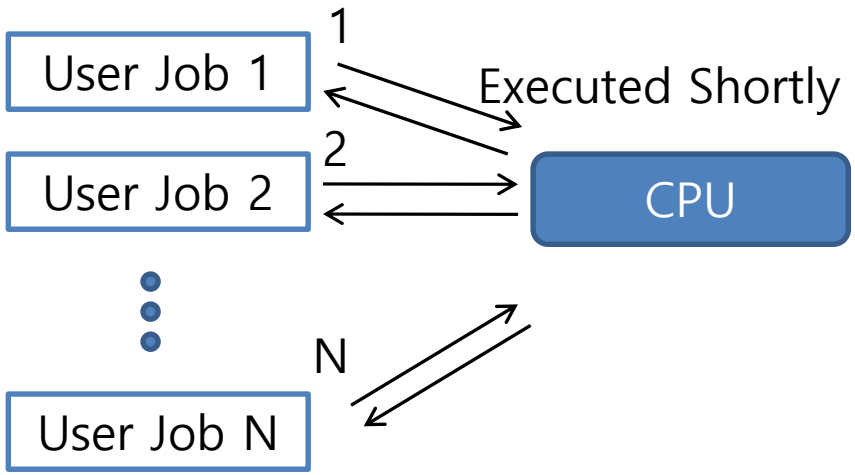
- 모든 프로그램은 타임 슬라이스 동안 CPU를 점유하고,
  그 시간이 끝나면 CPU를 양보(relinquish)

- 여러 개의 작업들은 CPU 스위칭을 통해서 동시에 실행됨

- CPU스위칭이 매우 빈번하게 일어남
  - 사용자는 실행중인 프로그램에 관여(interact)가 가능
  - 문제점: 성능 저하. 레지스터와 캐시를 비우고 새롭게 로드해야 하기 때문

Timesharing (cont.)





Multitasking

- 여러 개의 태스크(task)들이 CPU와 같은 자원을 공유하도록 하는 방법

- 하나의 작업(job) 은 동시에 실행할 수 있는 태스크로 나뉘어 질 수 있음
  - 예) 유닉스의 프로세스는 fork()시스템 콜을 이용해서 여러 개의 자식 프로세스를 생성할 수 있음

- Multitasking은 사용자가 여러 개의 프로그램을 실행 할 수 있도록 하며, CPU가 유휴 상태일 때는 background 작업을 실행 가능하도록 함

Issues with multitasking system

- 복잡한 메모리 관리 시스템이 필요
  - 동시에 여러 개의 프로그램이 메모리에 상주됨
  - 메모리 관리 및 보호 시스템 필요

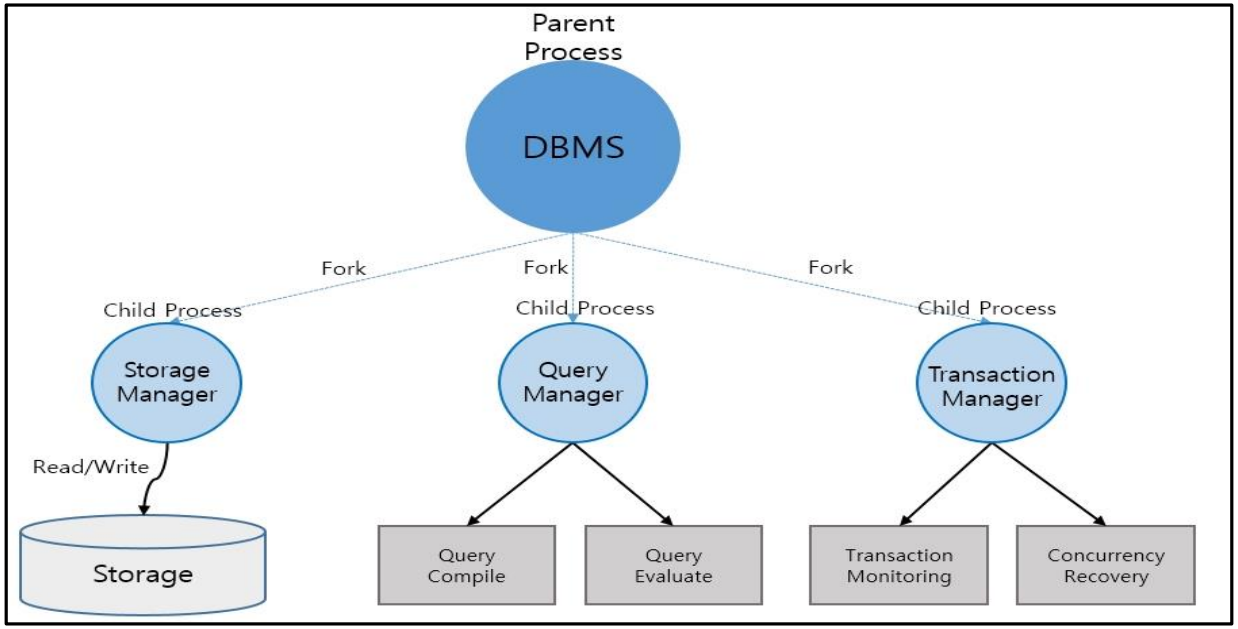
- 적절한 응답 시간을 제공해야 함
  - Job들은 메모리와 디스크로 swap in/out 될 수 있음

- (virtually) Concurrent execution 제공해야 함
  - Fine-grained CPU 스케줄링이 필요
    - 조밀한 타임 슬라이스 제공: 만약 10s 간격으로 동영상을 재생해주면?

- 필요에 따라서 job들간의 orderly execution 이 필요
  - 동기화, deadlock

Example of DBMS Multitasking

- Fork를 이용한 Child Process 생성
  - 멀티태스킹 : 여러 프로그램이 동시에 수행(Concurrent Execution)
  - 하나의 태스크가 다른 태스크(Child Process)를 만들 수 있는 기능을 의미함
  - 자신이 필요한 기능을 Child Process형태로 만들어 서로 협력을 통한 작업 수행



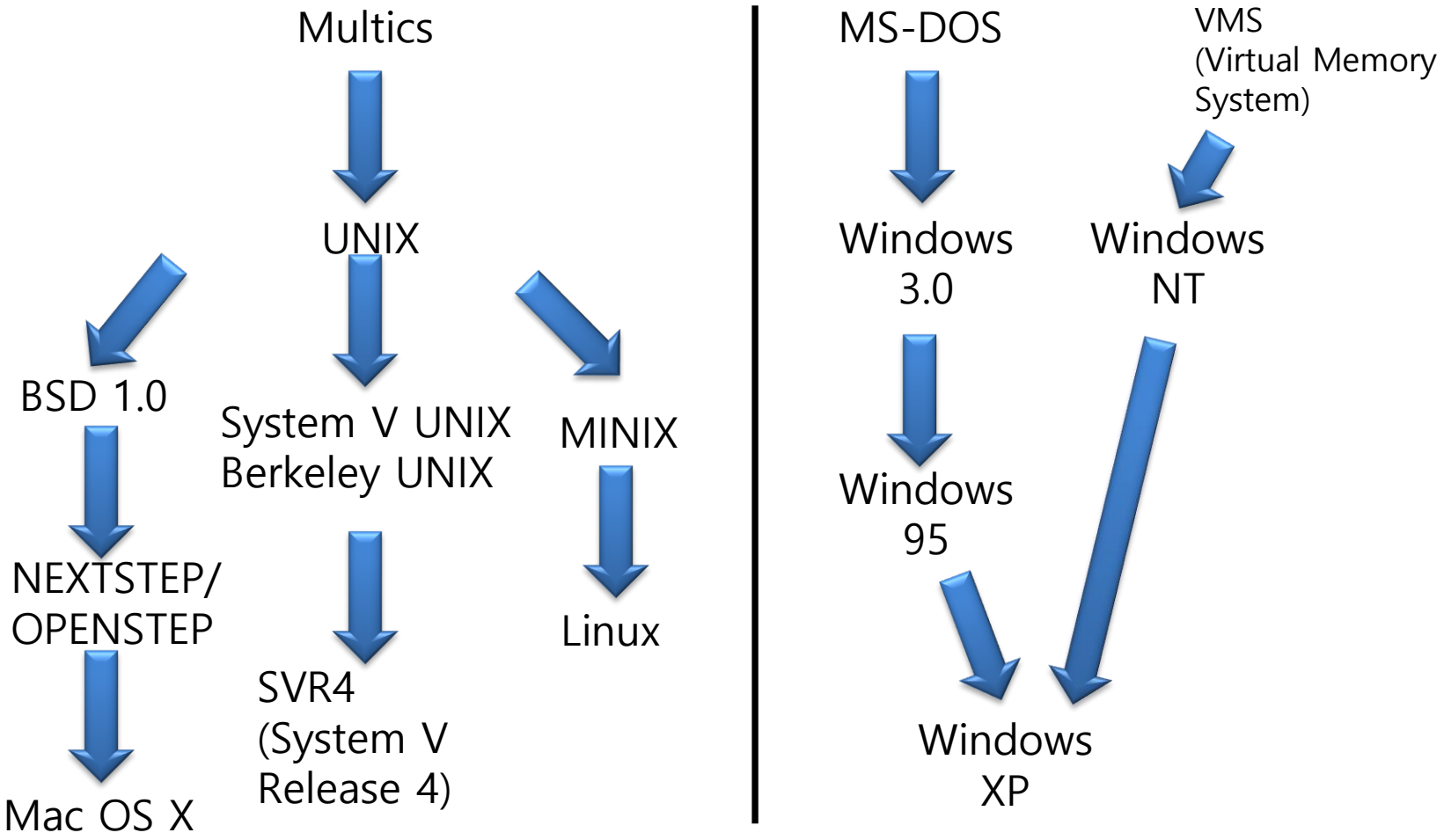
각 기법의 비교

- Multiprogramming: w/o voluntary yield
  - 장점: CPU 활용률 극대화
  - 단점: 작업들 간의 공평성 보장 불가
- Timesharing: with an interrupt of OS
  - 장점: OS의 정책에 따른 정확한 스케줄링 가능
  - 단점: 작업 간의 스위칭으로 인한 성능 오버헤드 발생
- Multitasking: based on timesharing
  - 하나의 job을 여러 task 로 구성하는 개념을 제공
    - 여러 task가 CPU를 번갈아가며 점유한다는 점에서, multiprogramming과 timesharing을 모두 multitasking의 한 형태로 보는 관점도 존재
  - 장점: 복잡한 프로그램의 설계를 가능하게 함

그 외 시스템 들

- Multiprocessor systems
  - Symmetric vs. asymmetric multiprocessor
- 분산시스템
  - LAN/WAN으로 연결
  - Client-server model, peer-to-peer model
- Clustered Systems
  - 공동의 목적을 위해 여러 개의 시스템 네트워크를 통해 작업을 수행
- Embedded systems
  - 특정 목적을 위한 운영체제 및 소프트웨어가 하드웨어에 탑재된 형태의 시스템
  - MP3 player, smart phone, PDAs
- Real-time systems
  - 시스템에서 수행하는 작업의 완료 시간(deadline)이 정해짐
  - Soft real-time, hard real-time

Lineage of well-known operating systems



References

- Course materials of
  - CS162, John Kubiatowicz, Berkeley Univ., 2015
  - CSE451, Gary Kimura & Mark Zbikowski, Washington Univ., 2007