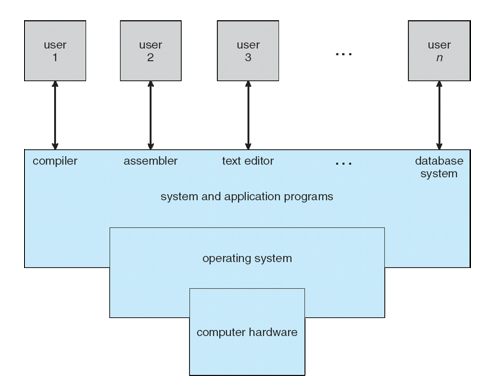
운영체제란?

* 컴퓨터 사용자와 컴퓨터 하드웨어의 중개 역할을 하는 프로그램
* 운영체제의 목표는?
  + 사용자 프로그램 실행 및 사용자 문제 해결 용이
  + 컴퓨터 시스템을 사용하기 편리하게 만들기
  + 효율적인 방법으로 컴퓨터 하드웨어 사용

컴퓨터 시스템 구조

* 컴퓨터 시스템의 4가지 요소
  + 하드웨어
    - 기본적인 계산 자원 제공
    - CPU, 메모리, I/O 장치들
  + 운영체제
    - 다양한 응용 프로그램 및 사용자 간의 하드웨어 사용 제어 및 조정
  + 응용프로그램
    - 사용자의 컴퓨팅 문제를 해결하기 위해 시스템 자원을 사용하는 방법 정의
    - 워드프로세서, 컴파일러, 웹 브라우저, 데이터베이스 시스템, 비디오 게임
  + 사용자
    - 사람들, 기계들, 다른 컴퓨터들
* 그림으로 보면



운영체제가 무엇을 하는지?

* 운영체제는 하드웨어를 제어하고 다양한 사용자들을 위한 다양한 애플리케이션 프로그램들 사이에서 그것의 사용을 조정한다.
* 컴퓨터 시스템을 하드웨어, 소프트웨어, 그리고 데이터로 구성된 것으로 볼 수 있다.
* 운영체제는 컴퓨터 시스템의 운영에 이러한 자원을 적절하게 사용할 수 있는 수단을 제공한다.
* 단순히 다른 프로그램들이 유용한 일을 할 수 있는 **환경**을 제공한다.
* 운영체제의 역할을 보다 완전하게 이해하기 위해, 우리는 다음의 두 가지 관점에서 운영체제에 대해 탐구한다.
  + 사용자
  + 시스템

사용자 관점

* 사용자의 컴퓨터 보기는 사용 중인 인터페이스에 따라 다르다.
  + 단일 사용자 컴퓨터
    - PC, 워크스테이션 등
    - 이러한 시스템은 한 명의 사용자가 자원을 독점하도록 설계되었다. 유저가 실시하고 있는 작업(또는 재생)의 최대화를 목표로 한다. 운영체제는 대부분 **사용하기 쉽고** **좋은 성능**을 위해 설계되었다.
  + 다중 사용자 컴퓨터
    - 메인 프레임, 컴퓨팅 서버 등
    - 이러한 사용자들은 자원을 공유하며 정보를 교환할 수 있다. 이러한 경우 운영 체제는 가용 CPU 시간, 메모리 및 I/O가 효율적으로 사용되며 개별 사용자가 에어 셰어보다 더 많이 사용하는 일이 없도록 자원 활용도를 극대화하도록 설계된다.
  + 핸드헬드 컴퓨터
    - 스마트폰, 태블릿 등
    - 모바일 컴퓨터의 사용자 인터페이스는 일반적으로 **터치 스크린**을 특징으로 한다. 이 시스템은 가용성과 배터리 수명에 최적화된 자원이 부족하다.
  + 임베디드 컴퓨터
    - 가정용 및 자동차용 컴퓨터
    - 사용자 인터페이스에는 숫자 키패드가 있을 수 있으며 표시등 조명을 켜거나 끌 수 있다. 운영체제는 주로 사용자의 개입 없이 실행되도록 설계된다.

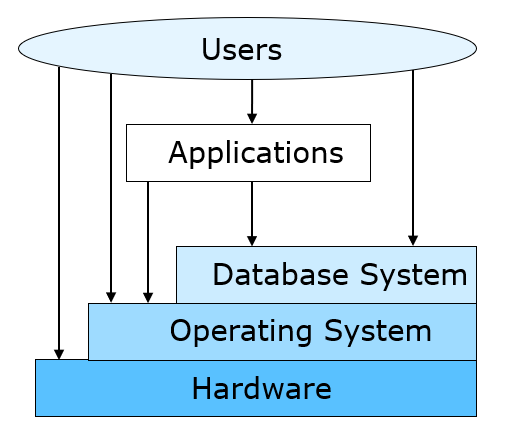
시스템 관점

* 컴퓨터의 관점에서 보면, 운영체제는 하드웨어와 가장 밀접하게 관련된 프로그램이다. 두 가지 관점이 있다.
  + **자원 분배**
    - 모든 자원 관리
    - 효율적인 자원사용과 공정한 자원사용에 대한 상충요청 결정
  + **프로그램 제어**
    - 컴퓨터의 오류 및 부적절한 사용을 방지하기 위한 프로그램 실행 제어

운영체제의 정의

* OS가 **무엇인지**에 대한 보편적으로 받아들여지는 정의는 없다.
  + 운영체제는 사용 가능한 컴퓨팅 시스템을 만드는 문제를 해결할 수 있는 합리적인 방법을 제공하기 위해 존재한다.
  + 컴퓨터 시스템의 기본 목표는 사용자 프로그램을 실행하고 사용자 문제를 쉽게 해결하는 것이다.
  + 베어 하드웨어만으로는 특별히 사용하기 쉽지 않기 때문에, 어플리케이션 프로그램이 개발된다.
    - 이러한 프로그램에는 I/O 장치를 제어하는 것과 같은 특정한 공통적인 작업이 필요하다.
    - 자원을 통제하고 할당하는 공통적인 기능을 하나의 소프트웨어, 즉 운영체제로 통합한다.
* OS의 **일부**인 것에 대해 보편적으로 인정되는 정의는 없다.
  + 보다 일반적인 정의와 우리가 주로 따르는 것은 운영체제가 컴퓨터에서 항상 실행되는 프로그램 중 하나인데 보통 **커널**이라고 불린다.
  + 커널과 함께, 두 가지 다른 유형의 프로그램이 있다.
    - 운영체제와 관련되지만 반드시 커널의 일부가 아닌 시스템 프로그램이다.
    - 시스템 작동과 관련되지 않은 모든 프로그램을 포함하는 응용 프로그램.
  + 모바일 기기의 등장으로 인해 운영체제를 구성하는 기능이 증가하게 되었다.
  + 모바일 운영체제는 종종 핵심 커널뿐만 아니라 애플리케이션 개발자들에게 추가 서비스를 제공하는 소프트웨어 프레임워크인 **미들웨어**를 포함한다.
  + 예를 들어, 애플의 iOS와 구글의 Android라는 두 가지 가장 두드러진 모바일 운영체제는 각각 데이터베이스, 멀티미디어, 그래픽을 지원하는 미들웨어와 함께 핵심 커널을 특징으로 한다.

컴퓨터 시스템의 발전 과정



사용자 🡪 하드웨어

사용자 🡪 운영체제 🡪 하드웨어

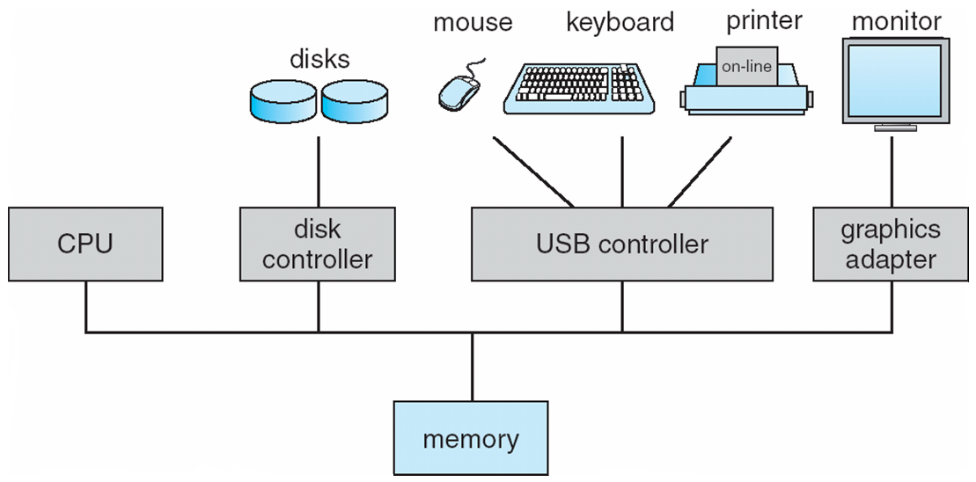
사용자 🡪 응용프로그램 🡪 운영체제 🡪 하드웨어

사용자 🡪 응용프로그램 🡪 데이터베이스 시스템 🡪 운영체제 🡪 하드웨어

컴퓨터 시스템 조직

* 현대의 범용 컴퓨터 시스템은 하나 이상의 CPU와 공유 메모리에 대한 접근을 제공하는 공통 버스를 통해 연결된 다수의 장치 제어기로 구성된다.
* 각 장치 컨트롤러는 특정 유형의 장치(예: 디스크 드라이브, 오디오 장치 또는 비디오 디스플레이)를 담당한다. 각 장치 컨트롤러에는 로컬 버퍼가 있다.
* CPU는 데이터를 메인 메모리에서 로컬 버퍼로, 로컬 버퍼에서 메인 메모리로 이동시킨다.
* CPU와 장치 컨트롤러는 병렬로 실행되어 메모리 사이클을 위해 경쟁할 수 있다. 공유 메모리에 대한 질서 있는 액세스를 보장하기 위해, 메모리 컨트롤러는 메모리에 대한 액세스를 동기화한다.

현대 컴퓨터 시스템



컴퓨터 시동

* **Bootstrap 프로그램**은 전원을 켜거나 재부팅할 때 로드된다.
  + 일반적으로 **펌웨어**[[1]](#footnote-1)로 알려진 ROM 또는 EPROM에 저장된다.
  + 시스템의 모든 측면을 초기화한다.
  + 운영체제 커널을 로드하고 실행을 시작한다.

컴퓨터 시스템의 작동

* 일단 커널이 로드되고 실행되면, 그것은 시스템과 그 사용자에게 서비스를 제공하기 시작할 수 있다.
* 일부 서비스는 커널 외부에서, **시스템 프로세스**가 되기 위해 부팅 시 메모리에 로드되는 시스템 프로그램 또는 커널이 실행되는 동안 내내 실행되는 **시스템 데몬**에 의해 제공된다.
* UNIX에서, 첫 번째 시스템 과정은 **init**이고 그것은 많은 다른 데몬을 시작한다. 이 단계가 완료되면 시스템이 완전히 부팅되며, 시스템은 어떤 이벤트가 발생할 때까지 기다린다.
* 사건의 발생은 대개 **인터럽트**에 의해 신호가 잡힌다

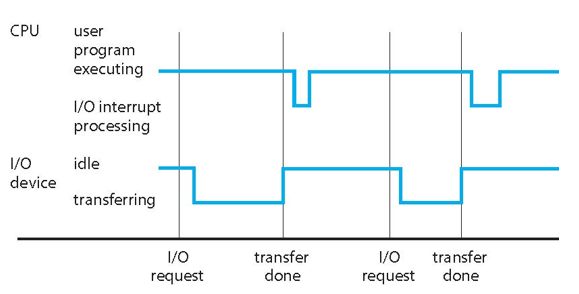
Interrupts

* 인터럽트에는 두 가지 유형이 있다.
  + **하드웨어**: 장치는 대개 시스템 버스를 통해 CPU로 신호를 보내서 인터럽트를 발생시킬 수 있다.
  + **소프트웨어**: 프로그램은 **시스템 호출**이라는 특별한 작업을 실행함으로써 인터럽트를 발생시킬 수 있다.
* 소프트웨어 생성 인터럽트(**트랩** 또는 **예외**라고도 함)는 오류(예: 0으로 나누기) 또는 사용자 요청(예: I/O 요청)에 의해 발생한다.
* 운영체제는 인터럽트에 의해 작동된다.

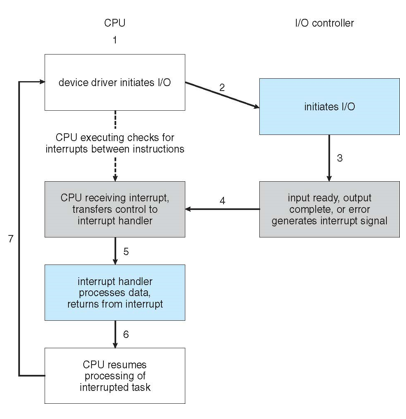
인터럽트의 공통 기능

* 인터럽트가 발생하면, 운영체제는 레지스터와 프로그램 카운터를 저장함으로써 CPU의 상태를 보존한다.
* 어떤 유형의 인터럽트가 발생했는지 결정하고 인터럽트 서비스 루틴으로 제어를 이전한다.
* 인터럽트 서비스 루틴은 각각 특정 인터럽트(예: 프린터, 디스크)를 처리하는 역할을 하는 루틴(모듈)의 모음입니다.
* 전송은 일반적으로 모든 서비스 루틴의 주소를 포함하는 **인터럽트 벡터**를 통해 이루어진다.
* 인터럽트 아키텍처는 중단 명령의 주소를 저장해야 한다.

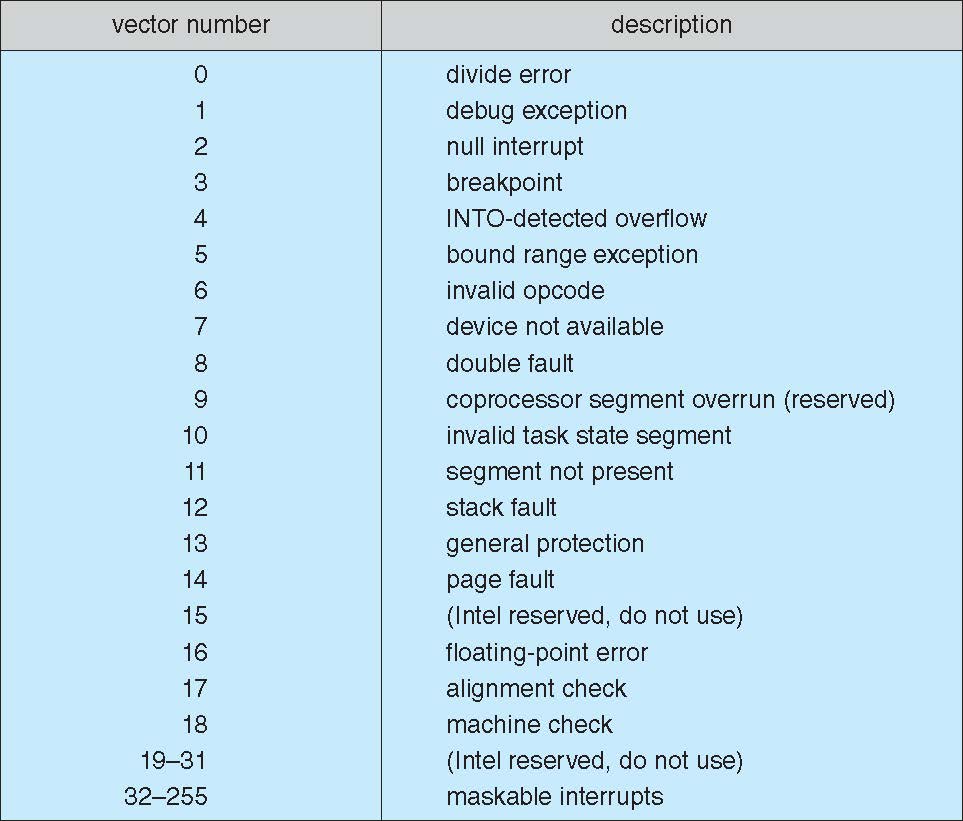
Interrupt Timeline



Interrupt-driven I/O cycle



Intel Pentium processor event-vector table



저장소 구조

* 메인 메모리: CPU가 직접 접근할 수 있는 유일한 대형 저장 매체
  + **랜덤 액세스**
  + 전형적으로 **휘발성**
* 2차 저장소: 대용량 **비휘발성** 저장 용량을 제공하는 메인 메모리의 확장
  + 하드 디스크: 자기 기록 재료로 덮인 견고한 금속 또는 유리 플래터
    - 디스크 표면은 논리적으로 **트랙**으로 나뉘는데, 이것은 **섹터**로 세분된다.
    - **디스크 컨트롤러**는 장치와 컴퓨터 사이의 논리적 상호작용을 결정한다.
  + **솔리드 스테이트 디스크(SSD)**: 하드 디스크보다 빠른 속도, 비휘발성
    - 다양한 기술
    - 인기가 높아지다

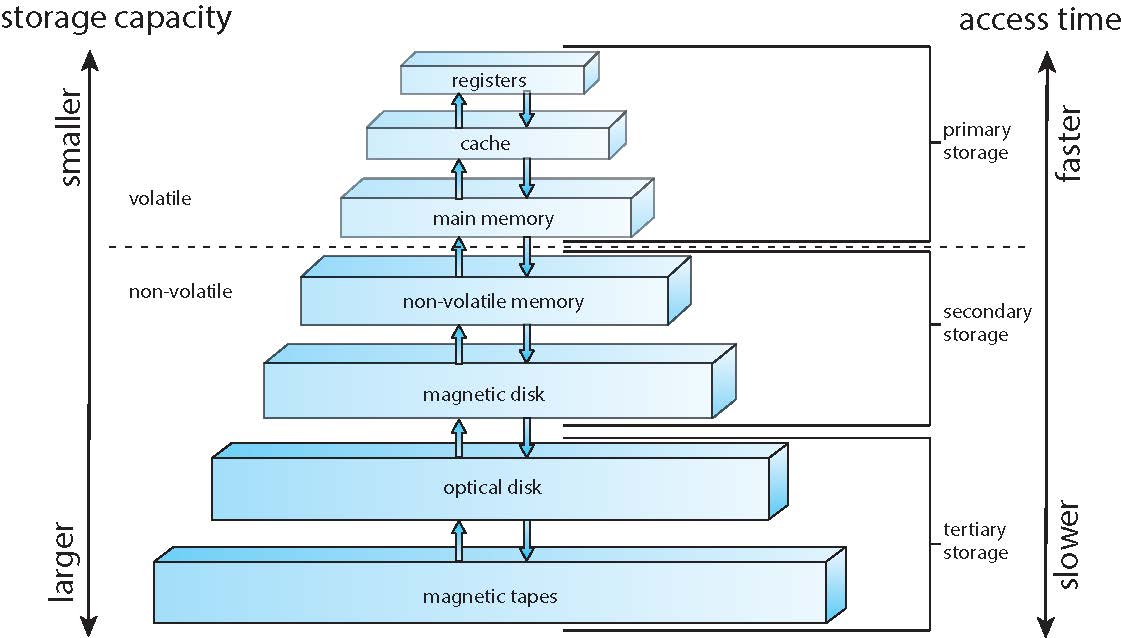
저장소 정의

* 컴퓨터 저장의 기본 단위는 **비트**다. 비트는 0과 1의 두 값 중 하나를 포함할 수 있다. 컴퓨터의 다른 모든 저장소는 비트의 집합에 기초한다.
* 1바이트는 8비트이고 대부분의 컴퓨터에서는 가장 작은 편리한 저장장치이다.
* 덜 일반적인 용어는 주어진 컴퓨터 아키텍처의 기본 데이터 단위인 **워드**이다. 한 워드는 하나 이상의 바이트로 구성되어 있다.
* 컴퓨터 스토리지는 대부분의 컴퓨터 처리량과 함께 일반적으로 바이트와 바이트 모음으로 측정되고 조작된다.
* **KB = 10의 3승**
* **MB = 10의 6승**
* **GB = 10의 9승**
* **TB = 10의 12승**
* **PB = 10의 15승**.
* 컴퓨터 제조업체들은 종종 이 수치를 반올림하며 메가바이트는 100만 바이트, 기가바이트는 10억 바이트라고 말한다. 네트워킹 측정은 이 일반적인 규칙에서 예외로 한다; 그것들은 비트 단위로 주어진다.

스토리지 계층

* 계층구조로 정리된 스토리지 시스템
  + 속도
  + 비용.
  + 휘발성
* **캐싱**: "느린" 스토리지에서 보다 빠른 스토리지 시스템으로 정보를 복사한다.
  + 메인 메모리는 2차 저장용 캐시로 볼 수 있다.
* I/O를 관리할 각 장치 컨트롤러용 **장치 드라이버**
  + 컨트롤러와 커널 간의 균일한 인터페이스 제공

저장 장치의 계층



입출력 구조

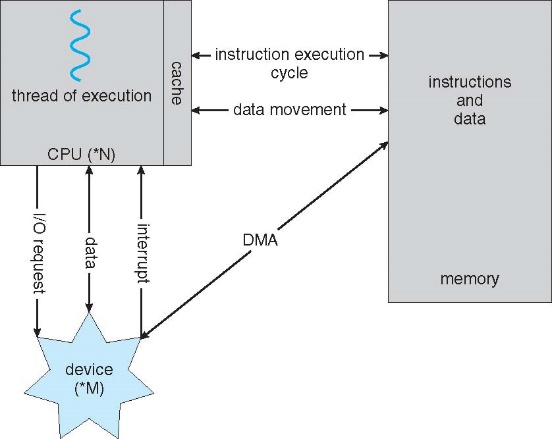
* 범용 컴퓨터 시스템은 공통 버스를 통해 연결된 CPU와 다중 장치 제어기로 구성된다.
* 각 장치 컨트롤러는 특정 유형의 장치를 담당한다. 두 개 이상의 장치를 부착할 수 있다. 예를 들어, 7개 이상의 장치를 **소형 컴퓨터 시스템 인터페이스(SCSI)** 컨트롤러에 연결할 수 있다.
* 기기 컨트롤러는 일부 로컬 버퍼 저장장치와 특수 목적 레지스터 세트를 유지한다.
* 장치 컨트롤러는 자신이 제어하는 주변 장치와 로컬 버퍼 스토리지 간에 데이터를 이동시킬 책임이 있다.
* 일반적으로 운영체제는 각 기기 컨트롤러에 대한 장치 드라이버를 가지고 있다. 이 장치 드라이버는 장치 컨트롤러를 이해하고 나머지 운영체제는 장치에 균일한 인터페이스를 제공한다.
* I/O 작동을 시작하기 위해 장치 드라이버는 장치 컨트롤러 내에 적절한 레지스터를 로드한다.
* 기기 컨트롤러는 차례로 이 레지스터의 내용을 검사하여 어떤 조치를 취할 것인지(예: 키보드에서 문자를 "읽기") 결정한다.
* 컨트롤러는 장치에서 로컬 버퍼로 데이터 전송을 시작한다. 데이터 전송이 완료되면 장치 컨트롤러는 인터럽트를 통해 장치 드라이버에 작동을 완료했음을 알린다.
* 그런 다음 장치 드라이버는 제어장치를 운영체제로 되돌리고, 작업이 판독값일 경우 데이터나 포인터를 데이터로 되돌릴 수 있다.
* 다른 작동의 경우 장치 드라이버는 상태 정보를 반환한다.

직접 메모리 액세스 구조

* 인터럽트 구동 I/O는 소량의 데이터 이동에는 괜찮지만 디스크 I/O와 같은 대용량 데이터 이동에 사용할 경우 높은 오버헤드를 발생시킬 수 있다.
* 이 문제를 해결하기 위해 **DMA(Direct Memory Access)**를 사용한다.
  + I/O 장치의 버퍼, 포인터 및 카운터를 설정한 후, 장치 컨트롤러는 CPU의 개입 없이 자체 버퍼 스토리지로 또는 직접 데이터 블록을 전송한다.
  + 장치 드라이버에게 작업이 완료되었음을 알리기 위해 블록당 하나의 인터럽트만 생성된다. 기기 컨트롤러가 이러한 작업을 수행하는 동안 CPU는 다른 작업을 수행할 수 있다.
* 일부 고급 시스템은 버스 구조보다 스위치를 사용한다. 이러한 시스템에서, 복수의 구성요소는 공유 버스의 사이클을 위해 경쟁하는 것이 아니라 다른 구성요소와 동시에 대화할 수 있다. 이 경우 DMA는 더욱 효과적이다. 다음 슬라이드의 그림은 컴퓨터 시스템의 모든 구성요소의 상호작용을 보여준다.

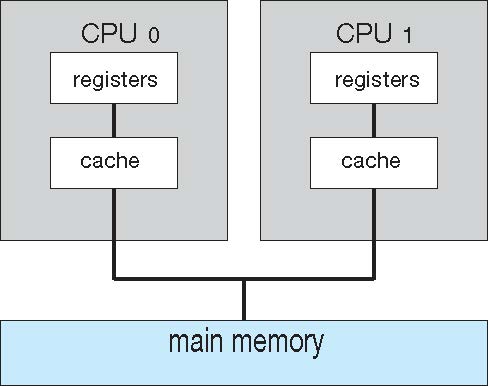
현대 컴퓨터의 작동 방식

* 폰 노이만 건축과 컴퓨터 시스템의 모든 요소들의 상호작용을 묘사한다.



컴퓨터 시스템 구조

* **단일 범용 프로세서**
  + 대부분의 시스템은 특수 목적 프로세서를 가지고 있다.
* **멀티프로세서** 시스템
  + **병렬 시스템**이라고도 하며, **단단히 결합된 시스템**
  + 장점은 다음과 같다.
    - **처리량 증가**
    - **규모의 경제**
    - **신뢰성 향상**: 우아하게 분해/고장감소
  + 두 가지 유형:
    - **대칭 다중 처리**: 각 프로세서가 모든 작업을 수행한다.

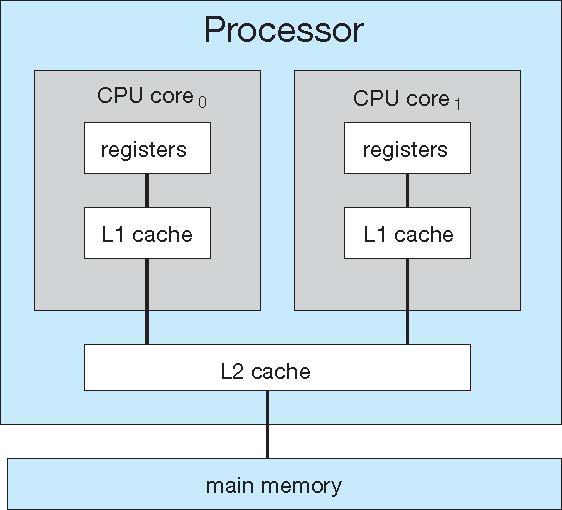


* + - **비대칭 다중 처리**: 각 프로세서에 특정 작업이 할당된다.

멀티코어 시스템

* 현재 대부분의 CPU 설계는 하나의 칩에 여러 개의 컴퓨팅 코어를 포함하고 있다. 그러한 멀티프로세서 시스템은 **멀티코어**라고 불린다.
* 멀티코어 시스템은 다음과 같은 이유로 단일 코어를 가진 다중 칩보다 더 효율적일 수 있다.
  + 칩간 통신보다 온칩 통신이 빠르다.
  + 코어가 여러 개인 한 칩은 여러 개의 싱글 코어 칩보다 훨씬 적은 전력을 사용하는데, 이는 모바일 기기뿐만 아니라 노트북에서도 중요한 이슈다.
* 참고: 다중 코어 시스템이 다중 프로세서 시스템인 반면 모든 다중 프로세서 시스템이 다중 코어인 것은 아니다.

두 개의 코어가 동일한 칩에 배치된 듀얼 코어



클러스터 시스템

* 멀티프로세서 시스템처럼 여러 시스템이 함께 동작한다.
  + 보통 **스토리지 영역 네트워크(SAN)**를 통해 스토리지 공유
  + 장애에서 살아남는 **고가용성** 서비스 제공
    - **비대칭 클러스터링**에는 열대기 모드의 기계가 하나 있다.
    - **대칭 클러스터링**은 애플리케이션을 실행하는 여러 노드를 가지며, 서로 모니터링한다.
  + **고성능 컴퓨팅(HPC)**을 위한 클러스터도 있다.
    - 응용 프로그램은 **병행** 사용을 위해 작성되어야 한다.
  + 상충하는 작업을 피하기 위해 **DLM(Distributed Lock Manager)**을 배포했다.

1.08.pdf

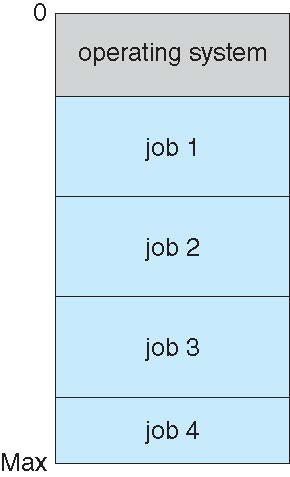
멀티프로그래밍 시스템

* 단일 사용자는 CPU와 I/O 장치를 항상 바쁘게 할 수 없다.
* **멀티프로그래밍**은 작업(코드와 데이터)을 구성하므로 CPU는 항상 실행할 작업을 가지고 있다.
* 시스템 내 전체 직업의 하위 집합은 기억 속에 보관된다.
* 배치 시스템:
  + 선택한 작업 한 개 및 **작업 스케줄**을 통해 실행
  + 기다려야 할 때(예를 들어 I/O의 경우), OS가 다른 작업으로 전환된다.
* 대화형 시스템:
  + 배치 시스템의 논리적 확장: CPU가 작업을 너무 자주 전환하여 사용자가 실행 중인 각 작업과 **상호작용**을 할 수 있게 함으로써 인터랙티브 컴퓨팅을 만들어낸다.

상호작용 시스템

* **응답 시간**은 1초 미만이어야 한다.
* 각 사용자는 메모리에서 실행 중인 프로그램을 적어도 한 개 가지고 있다. 그런 프로그램을 **프로세스**이라고 한다.
* 여러 프로세스가 동시에 실행될 준비가 되면 **CPU 스케줄링**이 필요하다.
* 프로세스가 메모리에 맞지 않으면 **스와핑**이 실행되도록 안팎으로 이동한다.
* **가상 메모리**는 완전히 메모리에 있지 않은 프로세스의 실행을 허용한다.

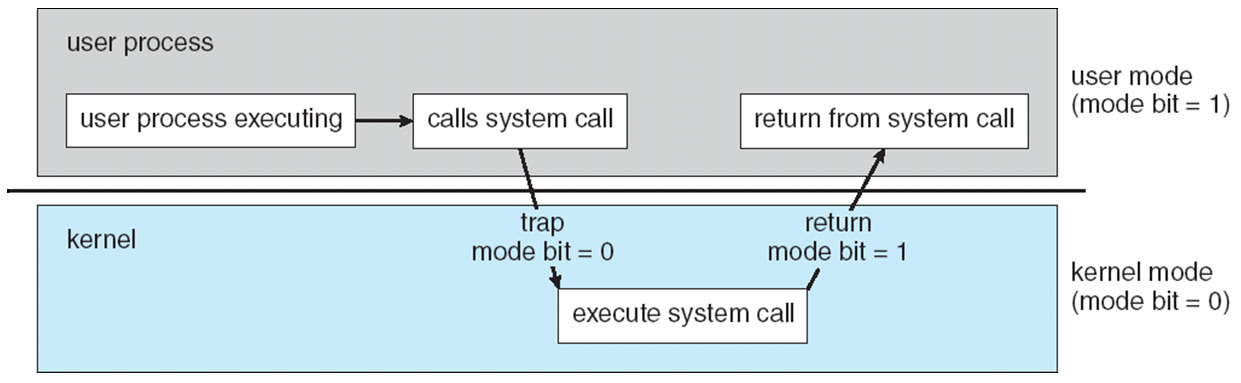
멀티프로그래밍 시스템의 메모리 레이아웃



작동 모드

* OS가 자신과 다른 시스템 구성요소를 보호하도록 하는 메커니즘
* 두 가지 모드:
  + **사용자 모드**
  + **커널 모드**
* 하드웨어 제공 모드 비트(0 또는 1)
  + 시스템이 사용자 코드 또는 커널 코드를 실행 중일 때 구별할 수 있는 기능 제공
  + 커널모드에서만 실행 가능한 **특권**으로 지정된 일부 명령
  + 사용자가 OS에 사용자 모드에서 커널 모드로 일부 기능 변경을 수행하도록 요청하는 시스템을 호출한다.
  + 시스템 호출에서 복귀하면 모드가 사용자 모드로 재설정된다.

사용자에서 커널 모드로 전환



Timer

* 프로세스가 무한 루프(프로세스 호깅 자원)에 있지 않도록 하드웨어 장치인 타이머 사용.
  + 타이머는 물리적 시계에 의해 감소되는 카운터다.
  + 타이머는 일정 시간이 지나면 컴퓨터를 중단하도록 설정되어 있다.
  + 운영 체제가 카운터를 설정함(권한 명령)
  + 카운터가 값 0에 도달하면 인터럽트가 생성된다.
  + OS는 할당된 시간을 초과하는 제어 또는 프로그램 종료를 위한 프로세스를 예약하기 전에 카운터의 값을 설정한다.

프로세스 관리

* 프로세스는 실행 중인 프로그램이다. 그것은 시스템 내의 작업 단위다. 프로그램은 **수동적인 실체(passive entity)**, 과정은 **능동적인 실체(active entity)**다.
* 프로세스는 작업을 수행하기 위한 자원이 필요하다.
  + CPU, 메모리, 입출력, 파일 등
  + 초기화 데이터
* 프로세스 종료는 재사용 가능한 자원의 재확보를 요구한다.
* 스레드는 프로세스 내에서 CPU 활용의 기본 단위다.
* 단일 스레드 공정. 지침은 순차적으로, 한 번에 하나씩, 완성될 때까지 실행된다.
  + 프로세스에는 실행할 다음 명령의 위치를 지정하는 **프로그램 카운터**가 하나 있다.
* 다중 스레드 프로세스에는 스레드당 하나의 프로그램 카운터가 있다.
* 일반적으로 시스템은 많은 프로세스, 일부 사용자, 일부 운영 체제를 하나 이상의 CPU에서 동시에 실행한다.
  + 스레드 간에 CPU를 멀티플렉싱하여 동시성 확보

프로세스 관리 활동

* 운영체제는 프로세스 관리와 관련하여 다음과 같은 활동을 담당한다.
  + 사용자 프로세스와 시스템 프로세스 모두 생성 및 삭제
  + 프로세스 일시 중단 및 재개
  + 프로세스 동기화를 위한 메커니즘 제공
  + 프로세스 커뮤니케이션을 위한 메커니즘 제공
  + 교착 상태 처리를 위한 메커니즘 제공

메모리 관리

* 프로그램을 실행하려면 지침의 모든(또는 일부)가 메모리에 있어야 한다.
* 프로그램에 필요한 데이터의 모든(또는 일부)는 메모리에 있어야 한다.
* 메모리 관리는 메모리에 무엇이 있는지 그리고 언제인지 결정한다.
  + 사용자에 대한 CPU 활용 및 컴퓨터 응답 최적화
* 메모리 관리 활동
  + 현재 사용되는 메모리의 어떤 부분과 누구를 통해 사용되는지를 추적하는 것
  + 메모리 내부 및 외부로 이동할 프로세스(또는 그 일부) 및 데이터 결정
  + 필요에 따라 메모리 공간 할당 및 할당 해제

스토리지 관리

* OS는 정보 저장에 대한 획일적이고 논리적인 관점을 제공한다.
* 물리적 속성을 논리적 스토리지로 추상화 단위: **file**
* 파일은 여러 개의 다른 저장 매체에 저장된다.
  + 디스크
  + 플래시 메모리
  + 테이프
* 각 매체는 장치 드라이버(디스크 드라이브, 테이프 드라이브)에 의해 제어된다.
  + 변동 속성에는 액세스 속도, 용량, 데이터 전송 속도, 액세스 방법(순차 또는 임의)이 포함된다.

파일 시스템 관리

* 일반적으로 디렉토리로 정리된 파일
* 누가 무엇을 액세스할 수 있는지 결정하기 위해 대부분의 시스템에 대한 액세스 제어
* OS 활동에는 다음이 포함된다.
  + 파일 및 디렉터리 만들기 및 삭제
  + 파일 및 디렉토리를 조작하는 기본 사항
  + 파일을 2차 저장소에 매핑
  + 안정적인(비휘발성) 저장 매체에 파일 백업

2차 저장소 관리

* 일반적으로 주메모리에 맞지 않는 데이터나 "긴" 기간 동안 보관해야 하는 데이터를 저장하는 데 사용되는 디스크
* 적절한 관리가 중요하다.
* 컴퓨터 작동의 전체 속도는 디스크 서브시스템과 그 알고리즘에 달려 있다.
* OS 활동
  + 자유공간관리
  + 보관할당
  + 디스크 스케줄링
* 어떤 저장장치는 빠를 필요가 없다.
  + 3차 저장소에는 optical storage, 마그네틱 테이프가 포함된다.
  + OS 또는 애플리케이션에 의해 여전히 관리되어야 함

Cashing

* 컴퓨터의 여러 계층에서 수행되는 중요한 원칙(하드웨어, 운영 체제, 소프트웨어)
* 사용 중인 정보를 저속 저장장치에서 고속 저장장치로 일시적으로 복사했다.
* 정보가 있는지 확인하기 위해 먼저 더 빠른 저장소(캐시)를 체크했다.
  + 만약 있다면, 캐시에서 직접 사용되는 정보(빠른 속도)
  + 만약 없다면, 데이터는 캐쉬에 복사되어 거기서 사용된다.
* 캐시는 캐시되는 스토리지보다 작다.
  + 캐시 관리 중요 설계 문제
  + 캐시 크기 및 교체 정책

다양한 수준의 저장성능

1_11.pdf

* 스토리지 계층 수준 간 이동은 명시적이거나 암묵적일 수 있다.

가상화

* 운영체제가 다른 OS 내에서 애플리케이션을 실행하도록 허용
  + 거대하고 성장 중인 산업
* 소스 CPU 유형이 대상 유형과 다를 때 사용되는 **에뮬레이션**(즉, PowerPC에서 Intel x86으로)
  + 일반적으로 가장 느린 방법
  + 컴퓨터 언어가 네이티브 코드로 컴파일되지 않은 경우 🡪 Interpretation
* **가상화**: CPU를 위해 기본적으로 컴파일된 OS, **게스트** OS를 실행하는 OS도 기본적으로 컴파일됨
  + 기본 WinXP **호스트** OS에서 WinXP 게스트, 각 애플리케이션을 실행하는 VMware를 고려해 보십시오.
  + **VMM(가상 머신 관리자)**은 가상화 서비스를 제공한다.

노트북과 데스크탑에서의 가상화

* VMM은 사용자가 네이티브 호스트 이외의 운영체제를 위해 작성된 애플리케이션을 실행할 수 있도록 한다.
  + 맥 OS X 호스트 윈도우를 게스트로 실행하는 애플 노트북
  + 다중 시스템 없이 여러 OS용 앱 개발
  + 다중 시스템 없이 응용 프로그램 테스트
  + 데이터 센터 내 컴퓨팅 환경 실행 및 관리

가상화 아키텍처 구조

1_20.pdf

커널 데이터 구조

* **1_13.pdf**많은 표준 프로그래밍 데이터 구조와 유사함
* **1_14.pdfSingly linked list**
* 1_15.pdf**Doubly linked list**
* **Circular linked list**
* **해시함수**는 **해시맵**을 생성할 수 있다.

1_17.pdf

* **비트맵**: n개 항목의 상태를 나타내는 n개의 이진수 문자열

컴퓨팅 환경 – 분산된

* 서로 네트워크로 연결된 별도의 이기종 시스템 모음
  + **네트워크**는 통신 경로, **TCP/IP**가 가장 일반적이다.
    - Local Area Network (LAN)
    - Wide Area Network (WAN)
    - Metropolitan Area Network (MAN)
    - Personal Area Network (PAN)
* **네트워크 운영체제**는 네트워크를 통해 시스템 간에 데이터를 공유할 수 있는 기능을 제공한다.
  + 통신 방식은 시스템이 메시지를 교환할 수 있게 해준다.
  + 단일체계의 환각

컴퓨팅 환경 – 클라이언트/서버

* 스마트 PC로 대체되는 덤프 단말기
* **클라이언트**에서 생성된 요청에 응답하는 많은 시스템들이 지금 **서버**를 사용하고 있다.
* **Compute-server 시스템**
  + 클라이언트에게 서비스를 요청하기 위한 인터페이스를 제공한다(예: 데이터베이스).
* **파일-서버 시스템**
  + 클라이언트가 파일을 저장하고 검색할 수 있는 인터페이스를 제공한다.

1_18.pdf

컴퓨팅 환경 – 실시간 시스템

* 가장 널리 보급된 컴퓨터 형태의 실시간 임베디드 시스템
  + 매우 상당한, 특별한 목적, 제한된 목적의 OS, **실시간 OS**
  + 확장 사용
* 다른 많은 특별한 컴퓨터 환경들 또한
  + 일부는 OS를 사용하고 일부는 OS 없이 작업을 수행한다.
* 실시간 OS는 잘 정의된 고정 시간 제약조건을 가지고 있다.
  + 제약 조건 내에서 처리가 이루어져야 한다.
  + 제약조건이 충족된 경우에만 작동 수정

오픈 소스 운영체제

* 2진 **폐쇄형 소스**가 아닌 소스 코드 형식으로 제공되는 운영 체제
* **복사 방지** 및 **DRM(Digital Rights Management)** 이동으로 이동
* "copyleft" **GNU Public License(GPL)**를 가지고 있는 **FSF(Free Software Foundation)**로부터 시작
* **GNU/Linux** 및 **BSD UNIX(Mac OS X**의 코어 포함**)** 등이 그 예다.
* VMware Player(윈도우즈에서 무료), Virtualbox(오픈 소스 및 여러 플랫폼에서 무료)와 같은 VMM을 사용할 수 있다. http://www.virtualbox.com
  + 탐색을 위한 게스트 운영 체제 실행에 사용

1. 펌웨어: 전자기기 등의 기본적인 제어 및 구동을 맡는 운영 체제의 일종 [↑](#footnote-ref-1)