2주차

2022년 3월 11일 금요일 오후 8:37

L2,L3

OS: 하드웨어와 유저 사이의 interface

1. **process management**

2. **Processor management**
   a. Scheduling
   b. 할당 관리
3. **Memory management**
   a. 메모리 할당방법
   b. 주기억장치
   c. Multi user , multi tasking 시스템
4. **File management**
   a. 사용자 및 시스템의 파일 관리
   b. 디렉터리 구조 지원
   c. 파일 관리 기능

Cache는 개념이다 buffer

Overlapped CPU & I/O operations

CPU와 device들은 bus를 통해 연결 되어있다

컴퓨터의 cpu는 부모 cpu가 되어 각 device마다 가지고 있는 cpu에게 일 할당한다
컴퓨터 cpu와 device간의 소통을 위해 systme bus를 사용한다

Device controller는 cput의 attention이 필요할때 bus를 통해 interrupt signal전송한다

컴퓨터 cpu와 모든 device들이 직접적으로 연결 되어있지 않은 이유

-> 모든 device들이 cpu와 통신하는건 아니기 때문이다 device끼리 통신할때도 있다
->I/O device와 cpu가 사용하는 data format 길이가 다르다
->이 차이를 없애기 위해 controller 라는 interface사용해서 연결한다

Multiprogrammed batch systems

일괄 처리 시스템 모아두었다가 한번에 처리 하는것을 의미한다

여러 프로그램들을 한번에 처리 하는것을 의미
CPU 사용률 극대화가 목적

단점: 다른 프로세스는 계속 대기 해야하는 단점이 있다
->그래서 time sharing system 적용한다

Time sharing system

여러 사용자가 자원을 동시에 사용
시간을 나누어서 사용하는 시스템이다

New problems : Need preemptive(우선순위)
즉 우선 순위에 따른 처리가 필요함
그리고 과도한 time Sharing 현상인 thrashing을 피하는게 필요하다

-> personal computing 지급됨
더 이상 cpu 활용율이 고려 대상이 아님 os가 단순해짐

More Recent Developments

1. **parallel OS**
   a. Clock과 memory 공유한다
   b. 성능 향상
2. **Distributed OS**
   a. No shared memory , no shared clock
   b. Loosely coupled system -> network로 여러대의 컴퓨터 연결한다 -> 클라우드 시스템

c. 각 노드들은 각자 OS를 가지고 있는 개인 PC이다
d. 높은 확장성 , 고신뢰성
e. 구축 및 관리가 어려움

3. **Real time OS**
a. 작업처리에 제한시간(deadline)을 갖는 시스템
b. Hard real time task : 무기 제어 ,발전소 제어 [시간안에 처리해야하는 경우]
c. Soft real time task : 동영상 재생 등 실시간 영상 [큰 재앙까지는 아닌 정도]
d. IOT , 비행 제어에서 쓰인다

I/O

CPU와 device controller는 common bus를 사용한다

Software polling synchronous I/O

Polling : 한 프로그램이나 장치에서 다른 프로금이나 장치들이 어떤 상태에서 있는지를 지속적으로 체크하는 전송제어 방식으로 , 대체로 그들이 아직도 접속 되어있는지와 데이터 전송을 원하는지 등을 확인한다

CPU가 I/O operation을 시작하면, 계속해서 I/O operation이 끝날때까지 device를 체크한다.

Device Controller는 다른 device와 소통하기 위해 자신만의 register를 가지고 있다.
1. Input register, output register(Data register) - for data
2. Control register - CPU가 보낸 I/O 명령을 임시저장하는 역할
3. Status register - Data Register의 상태를 표시 받을 준비가 됐는지 안됐는지

Synchronous I/O

Clock 신호 맞춰서 일 진행
I/O 가 끝날때까지 cpu는 기다린다

Asynchronous I/O

Brude rate 시간 bps 맞추어서 통신하는 방법
I/O 종료 여부 관계 없이 제어권이 user program에게 돌아온다

CPU가 interrupt 받았을때

Interrupt handler가 interrupt vector(ISR 주소)사용해서 ISR수행 (interrupt service routine)

인터럽트 벡터: 벡터 테이블, 커널에 존재, 서비스 루틴(커널속 코드) 짜두고 그 주소만 벡터테이블에 넣어놓고 포인팅할 수 있도록 함 (입력-> 주소 알아냄-> 거기로가서 명령어 수행)
인터럽트 아키텍쳐는 인터럽트된 지시에 대한 주소를 저장하고 있어야함. 칩에 0~15의 전기신호가 발생하면 이것이 cpu속으로 들어가 cpu에게 인터럽트가 발생했음을 알림.

Handler는 넘어올때 register 저장하고 다시 돌아갈때 restore한다

인터럽트 5번 누름 -> 벡터 테이블로감 -> 주소 알아냄 -> 디바이스로 감 -> 적힌 명령어대로 버퍼에서 유저 프로세서로 넘김
이 벡터에서 해당 루틴의 번지가 기억된 순번을 지적하여 그 기능을 요청하면 그곳에 기억된 번지로 분기하여 인터럽트를 수행한 후 서브루틴을 호출할 때와 같이 원래의 순서로 복귀될 수 있도록 한다.

DMA

Direct memory access
->CPU거치지 않고 memory를 통해 data 통신 할수있다는 개념

높은 속도의 I/O deivce의 정보 전송을 memory speed처럼 빠르게 하게 해준다.
OS가 이 I/O를 위해 특별히 memory에 buffer를 만들어서 쓰라고 준다.
I/O Device는 CPU와 Asynchronous하게 작동되고, 끝나면 CPU에게 인터럽트를 준다.
Device controller는 buffer에 있는 data block을 CPU의 개입 없이 main memory로 바로 전송한다.

Storage structures -> buffer

메모리 계층 구조
Cache 개념 -> Locality of Reference buffer의 핵심 원리

공간 지역성
시간 지역성
Cache hit or miss