

폭주하는 인터럽트,
CPU는 어떻게 감당할까?

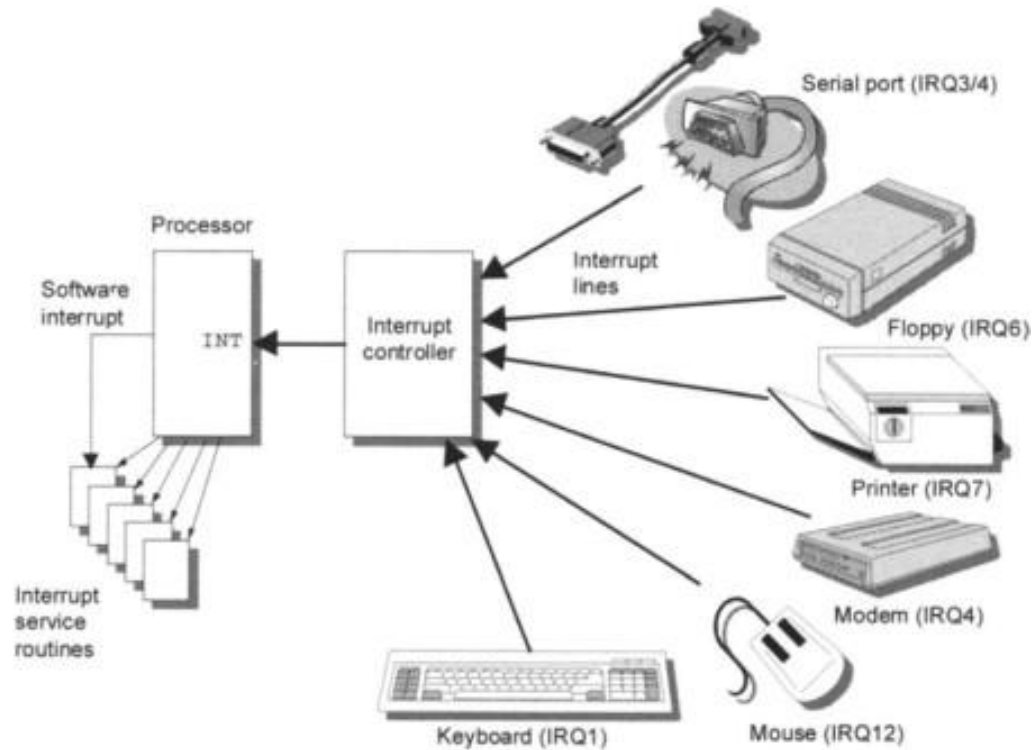
고종환

목차

1. 인터럽트란?
2. 소프트웨어 인터럽트 vs. 하드웨어 인터럽트
3. 하드웨어 인터럽트 부하 관리 방법
4. Q&A

1. 인터럽트란?

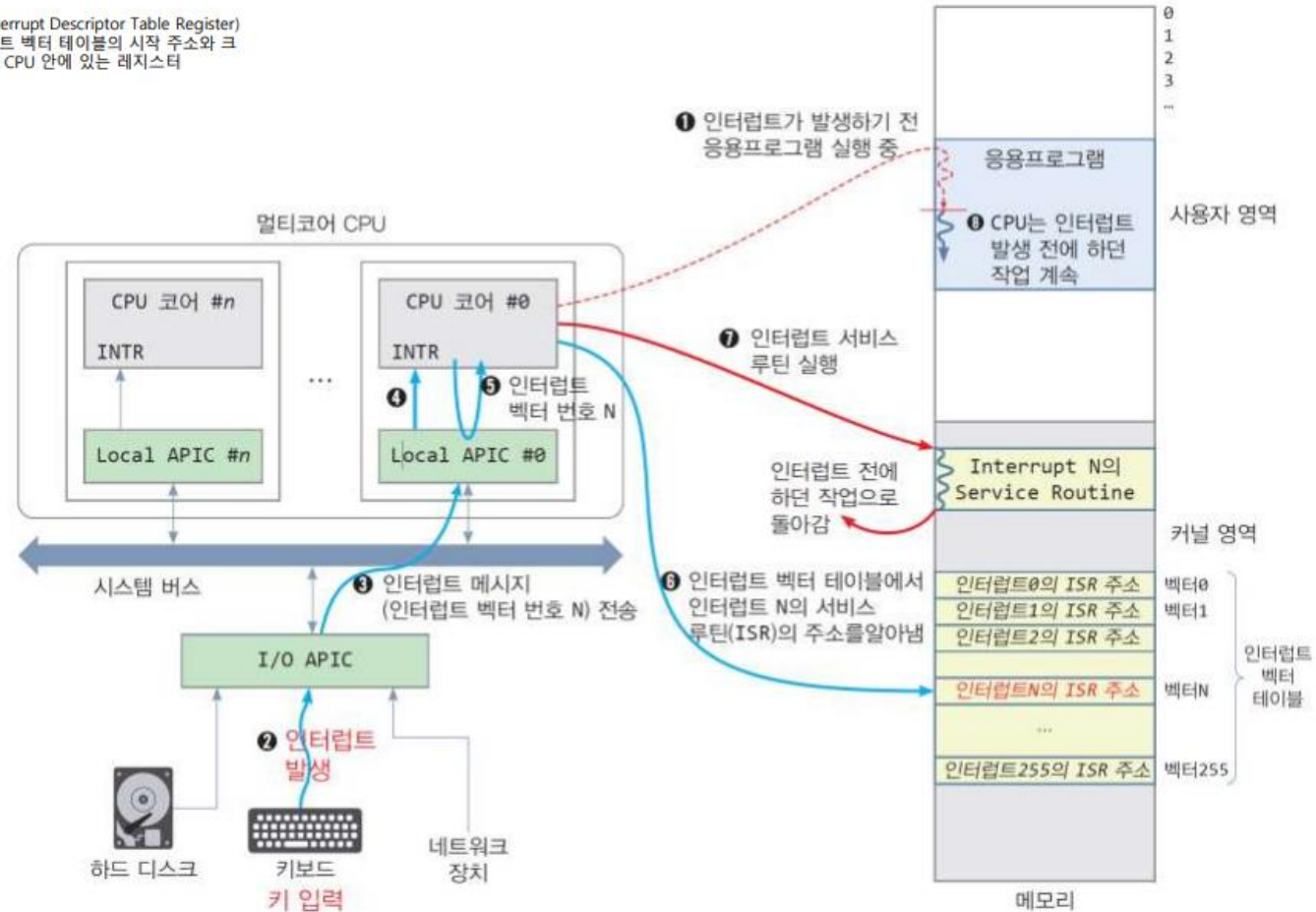
인터럽트의 개념



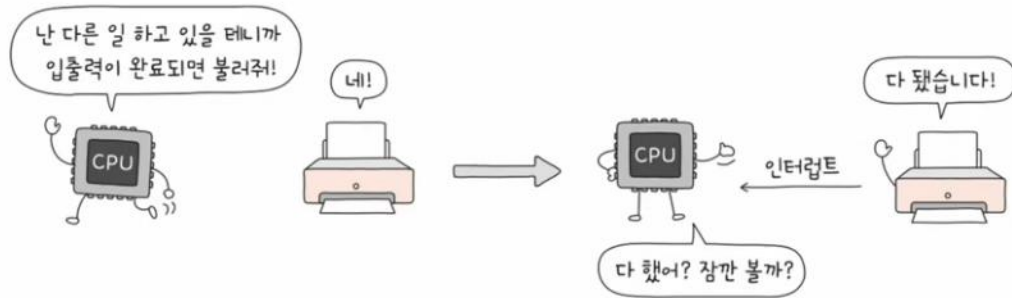
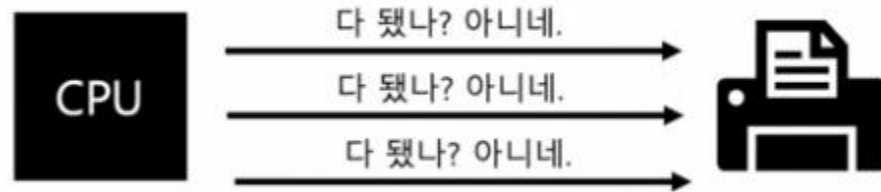
- CPU가 다른 요청을 처리하기 위해 현재 작업을 잠시 멈추는 것
- 외부 장치의 중요한 비동기적인 요청을 처리
- **Interrupt controller:** 여러 하드웨어 장치들의 인터럽트 요청을 받아 우선순위를 결정하고, CPU에 INT 신호를 보낸다.
- **ISR(Interrupt Service Routine):** 인터럽트 발생 시 CPU가 실행하는 특별한 프로그램. 인터럽트를 일으킨 원인에 맞춰 해당 문제를 해결하거나 필요한 작업을 수행하는 코드로 구성됨.

인터럽트 처리 과정

IDTR(Interrupt Descriptor Table Register)
2. 인터럽트 벡터 테이블의 시작 주소와 크
1을 가진 CPU 안에 있는 레지스터



폴링 vs. 인터럽트



폴링 (Polling)

- CPU가 **일정 간격으로** 장치 상태를 확인하는 방식
- CPU가 상태 확인에 많은 시간을 소비하여 다른 작업 수행에 제약 발생

인터럽트 (Interrupt)

- 장치가 **준비되었음을 CPU에 신호로 알려주는 방식**
- CPU 자원을 효율적으로 사용 가능 → 반응 속도도 빠름

2. 소프트웨어 인터럽트 vs. 하드웨어 인터럽트

소프트웨어 인터럽트란?

- 실행 중인 프로그램의 요청에 의해 의도적으로 발생하는 인터럽트



- 예시
 - 시스템 콜: 운영체제의 기능을 요청
 - 예외 처리: 프로그램의 비정상적인 상황 처리
 - 디버깅 중단점: 디버깅을 위해 프로그램 실행을 일시 중단

소프트웨어 vs. 하드웨어 인터럽트

	소프트웨어 인터럽트	하드웨어 인터럽트
발생 원인	프로그램 내부 명령어에 의해 의도적으로 발생	외부 하드웨어 장치에 의해 비동기적으로 발생
발생 시점	프로그램 실행 중에 특정 시점에 발생	예측 불가능한 시점에 발생
처리 방식	인터럽트 컨트롤러를 거치지 않고 스스로 처리	외부 신호를 받아 CPU가 처리
주요 역할	사용자 프로그램이 OS의 기능을 사용하도록 함	CPU와 외부 장치 간의 통신 및 데이터 입출력 관리

3. 하드웨어 인터럽트 부하 관리 방법

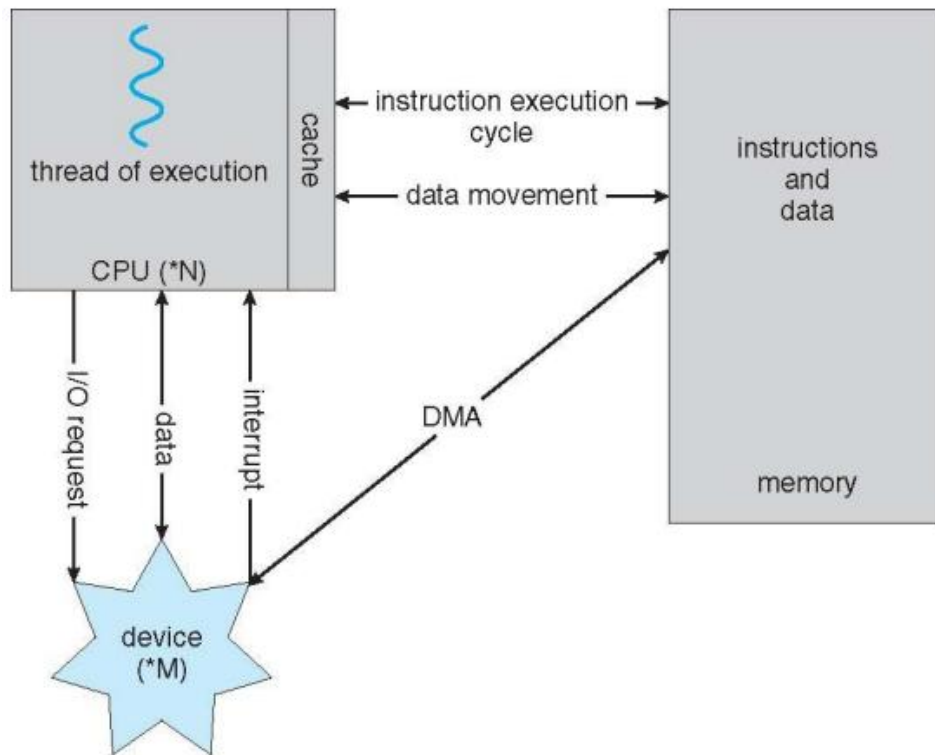
CPU 리소스를 잡아먹는 인터럽트

1. 타이머 인터럽트 — 밀리초 단위로 주기적으로 발생 (OS 스케줄링 핵심)
2. 네트워크 인터럽트 — 대량 패킷 수신 시 초당 수천 번 발생 가능
3. 저장장치 I/O 인터럽트 — SSD/HDD 읽기·쓰기 완료 시 빈번 발생
4. 그래픽/GPU 인터럽트 — 고주사율·고프레임 작업 시 빈번
5. 오디오 인터럽트 — 실시간 스트리밍·녹음 시 짧은 주기로 발생

인터럽트 오버헤드를 어떻게 감당할까?

- 현대 컴퓨터가 하드웨어 인터럽트를 최적화하는 방식
 - DMA (Direct Memory Access)
 - 인터럽트 병합 (Interrupt Coalescing)
 - APIC (Advanced Programmable Interrupt Controller)
 - 인터럽트 친화성 (IRQ Affinity)
 - NAPI (New API)

DMA (Direct Memory Access)



- CPU의 개입 없이 주변 장치가 메인 메모리에 직접 데이터를 읽거나 쓸 수 있도록 해주는 기술.
- 대량의 데이터 전송 시, 데이터 전송이 완료된 경우에만 **DMA 컨트롤러**가 단 한번의 인터럽트를 보냄.

인터럽트 병합 (Interrupt Coalescing)

인터럽트 병합

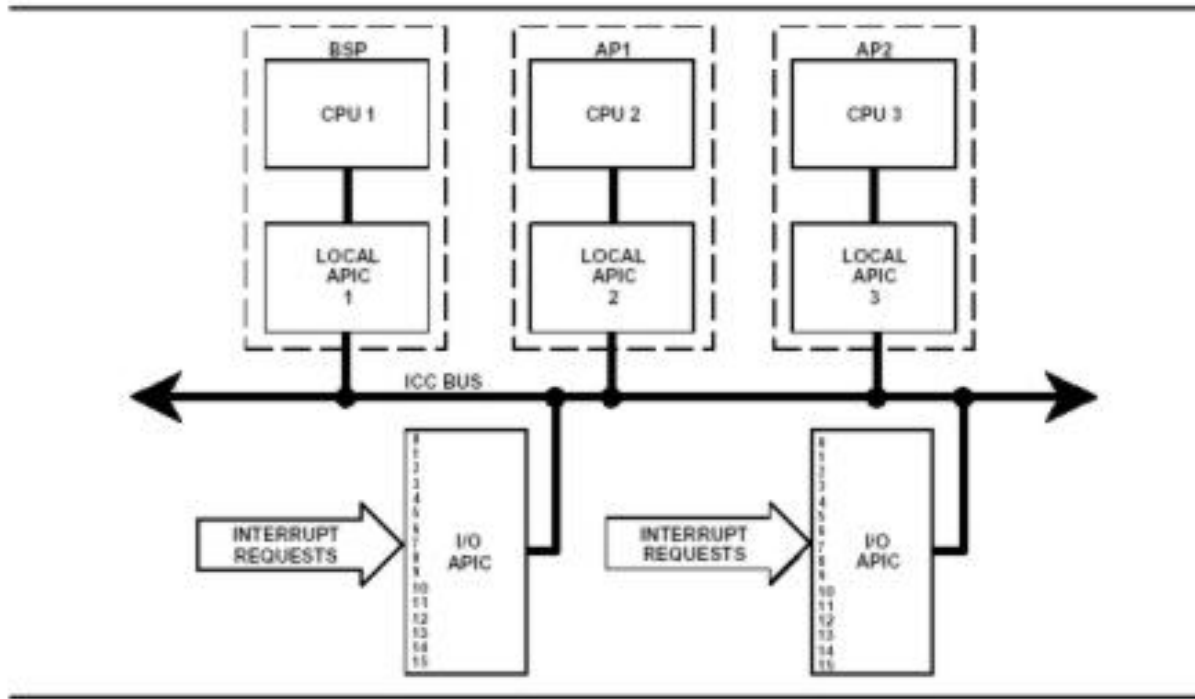
마지막 업데이트: 2025년 1월 20일

호스트 시스템에 너무 많은 인터럽트가 발생하는 것을 방지하기 위해, 패킷을 수집하여 여러 패킷에 대해 하나의 인터럽트를 생성합니다. 이를 *인터럽트 병합*이라고 합니다.

수신 작업의 경우, 인터럽트는 일반적으로 호스트 CPU에 패킷이 장치의 입력 큐에 도착했음을 알립니다. 어댑터에 인터럽트 조절 로직이 없으면 각 수신 패킷에 대해 인터럽트가 발생할 수 있습니다. 그러나 수신 패킷 속도가 증가함에 따라 장치 드라이버는 한 패킷의 처리를 완료하고 수신 큐에 더 많은 패킷이 있는지 확인한 후 드라이버를 종료하고 인터럽트를 해제합니다. 드라이버는 처리할 패킷이 더 있음을 확인하고 패킷 속도가 증가함에 따라 인터럽트당 여러 패킷을 처리하게 됩니다. 이는 부하가 증가함에 따라 시스템 효율이 향상됨을 의미합니다.

하지만 일부 어댑터는 수신 인터럽트 생성 시점을 더욱 세부적으로 제어할 수 있는 추가 기능을 제공합니다. 이를 인터럽트 병합 또는 인터럽트 조절 로직이라고 하며, 여러 패킷을 수신하고 여러 패킷에 대해 하나의 인터럽트를 생성할 수 있도록 합니다. 첫 번째 패킷이 도착하면 타이머가 시작되고, 그 후 n 마이크로초 또는 m 개의 패킷이 도착할 때까지 인터럽트가 지연됩니다. 이러한 방법은 어댑터와 장치 드라이버에서 사용자가 제어할 수 있는 기능에 따라 다릅니다.

APIC (Advanced Programmable Interrupt Controller)

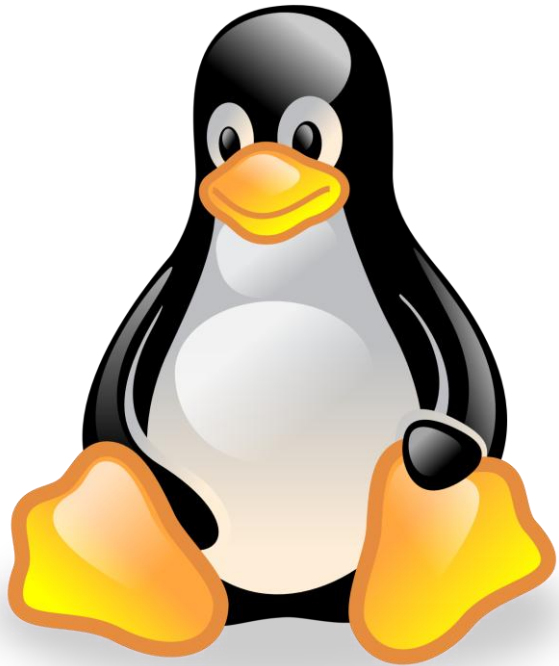


- 다중 코어 시스템에서 인터럽트를 효율적으로 관리하기 위한 고급 인터럽트 컨트롤러
- 각 CPU 코어에 인터럽트를 분산시키거나 특정 코어에만 인터럽트 할당 가능

인터럽트 친화성 (IRQ Affinity)

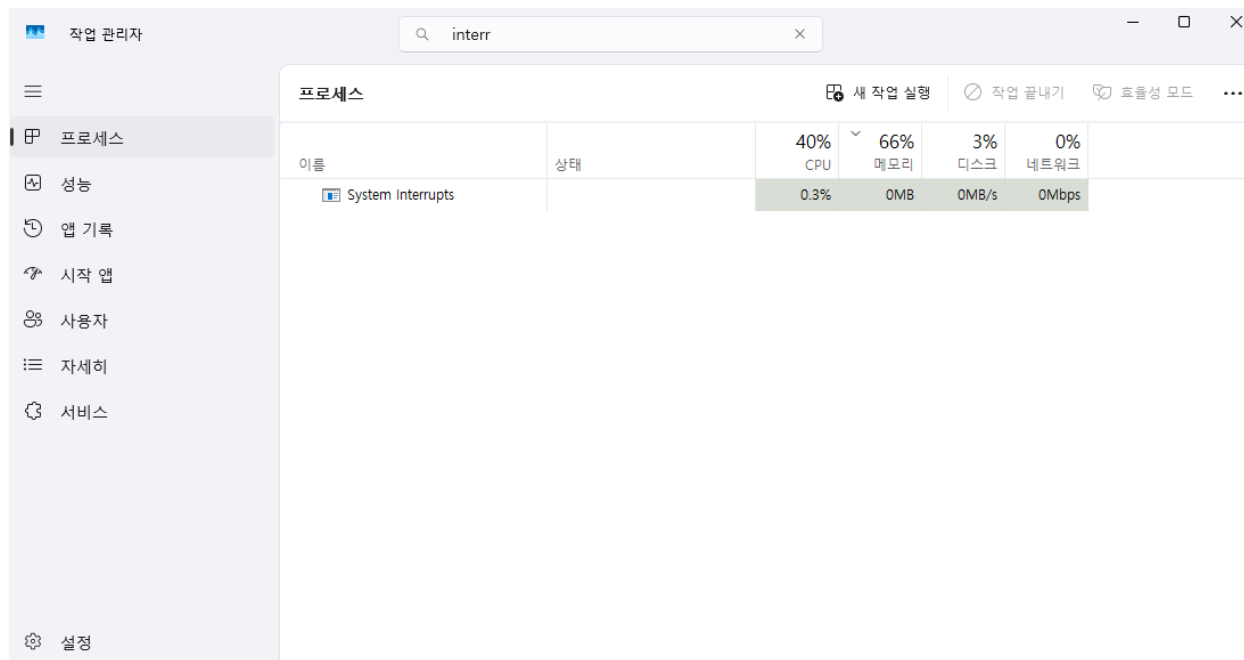
- 특정 종류의 인터럽트를 특정 코어에 할당하는 기능
 - ex) 특정 네트워크 카드의 인터럽트는 항상 한 코어에서만 처리
- > CPU 캐시 미스를 줄여 캐시 효율 향상

NAPI (New API)



- 네트워크 패킷 처리의 효율을 높이기 위한 리눅스 커널 기술
- 기존 방식: 패킷이 도착할 때마다 인터럽트를 발생
- 개선: 인터럽트 발생 시, 드라이버가 패킷을 폴링하는 방식으로 처리

인터럽트 부하 확인하기



The screenshot shows the Windows Task Manager window with the 'Processes' tab selected. A search bar at the top contains the text 'interr'. The 'System Interrupts' process is highlighted in the list. The left sidebar shows navigation options: '작업 관리자' (Task Manager), '프로세스' (Processes), '성능' (Performance), '앱 기록' (App History), '시작 앱' (Startup Apps), '사용자' (Users), '자세히' (More details), '서비스' (Services), and '설정' (Settings).

프로세스		새 작업 실행				작업 끝내기	효율성 모드	...
이름	상태	40% CPU	66% 메모리	3% 디스크	0% 네트워크			
System Interrupts		0.3%	0MB	0MB/s	0Mbps			

4. Q & A