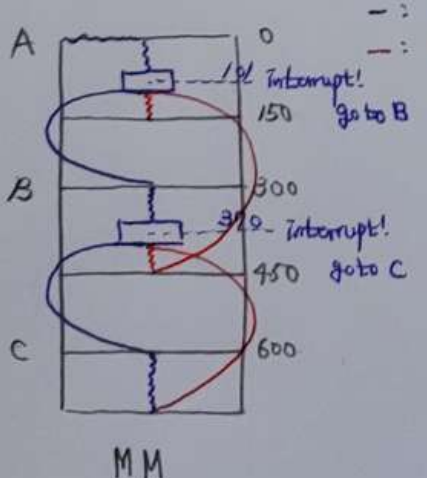
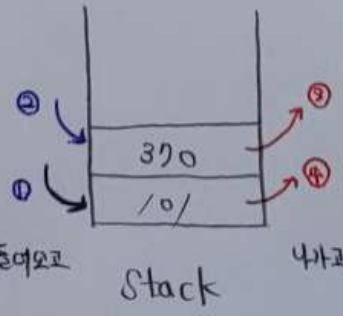
2019 11 19 | 컴퓨터 공학과 이은동 운영체제 - Interrupt Servicing procedure

컴퓨터 내부의 CPU가 현재 어떠한 Action을 취하고 있는 중에, CPU에게 interrupt signal이 도착한다. 그러면 CPU는 자신이 하던 일을 마저 다 끝마치고 나서, 그 interrupt를 확인한다. CPU에게 도착한 interupt를 처리하기 전에, 먼저 OS는 interrupt를 해결하고 CPU가 다시 수행해야할 자리의 adress를 stack에다가 저장을 해둔다. ex) push(stk-ptr, PC) 예를 들어, CPU가 100번지 자리에서 일을 하던 중에 interrupt가 발생하면 stack에는 CPU가 다음에 실행해야할 자리의 주소값인 (PC값) 101번지를 stack에다가 저장하는 것이다. CPU가 interrupt를 처리하는 데에는 크게 방식이 존재한다. 첫번째는 polling 방식으로, Device controller 들에게 누가 interrupt를 보냈는지 직접 CPU가 확인하는 방식이다. 이는 주로 작은 system에서 사용한다. 두번째는 Vectored (= array) interrupt 방식이다. 모든 interrupt 에는 각각의 고유한 unique id가 부여되는데, CPU는 이 interrupt starting address 들을 모아둔 vector를 register에서 확인한 뒤, I id에 알맞는 adress로 가서 interrupt를 처리한 뒤, stack에 저장해 두었던 주소값을 pop 하여 (PC = pop(Stk-ptr)) 보내 하고 있었던 일의 다음 주소부터 이어서 일을 할 수 있는 것이다. 컴퓨터를 실행하다 보면 정말 여러종류의 interrupt가 시도때도 없이 발생하는 걸 알 수 있는데, 이러한 interrupt 들을 처리하는데 3가지의 방식이 존재한다. ①들어온 순서대로 interrupt를 실행하거나, ②뒤에 들어오는 interrupt들은 버려버리거나, ③혹은 interrupt에 우선순위를 두고 계층적으로 실행하는 방식이 존재한다. 보통은 가장 효율적으로 동작할 수 있는 ③을 선택하는데, 이 방식은 기본적으로 Preempt - Resume 방식을 이용한다. 현재 하던일을 처두고 다른 (새로운 일) 일을 하러갔다가, 새로운 일이 끝나면 현재하던 일을 돌아가는 것이다.

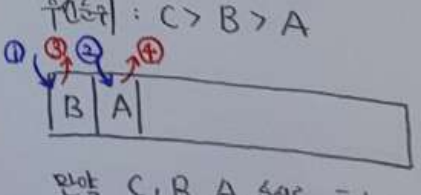
A를 실행하던 중 interrupt가 걸려 B로 I 일을 수행하러 갔다가, 또 그중에 Interrupt가 걸려 C로 그 일을 수행하고, 순차적으로 되돌아오는 과정과 Stack의 구조 과정을 그림으로 나타내면 다음과 같다.



— : interrupt 처리!
— : 들어오는것



또한 우선순위가 낮은 interrupt가 들어오면 버리지 않고 "Queue"에 save 한다.



만약 C, B, A 순서로 interrupt가 들어왔을때를 가정! (단 이미 실행중)

Queue