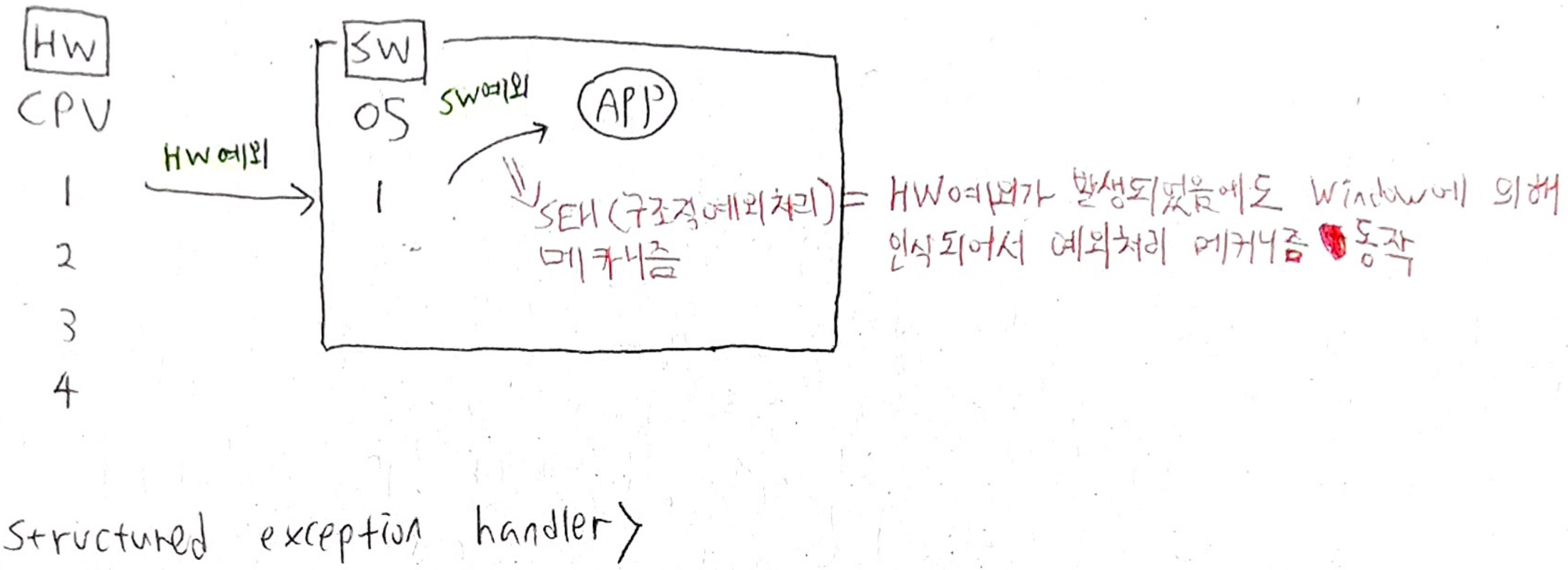
17장 구조적 예외처리(SEH) 기법

-1/22 이충현

CPU
1 예외 → 1' Handler 예외
2 예외 → 2' Handler 예외
3 예외 → 3' 〃
4 예외 → 4' 〃

OS
SW
결정권 넘어!

HW예외 ⇓ SW예외 ⇓



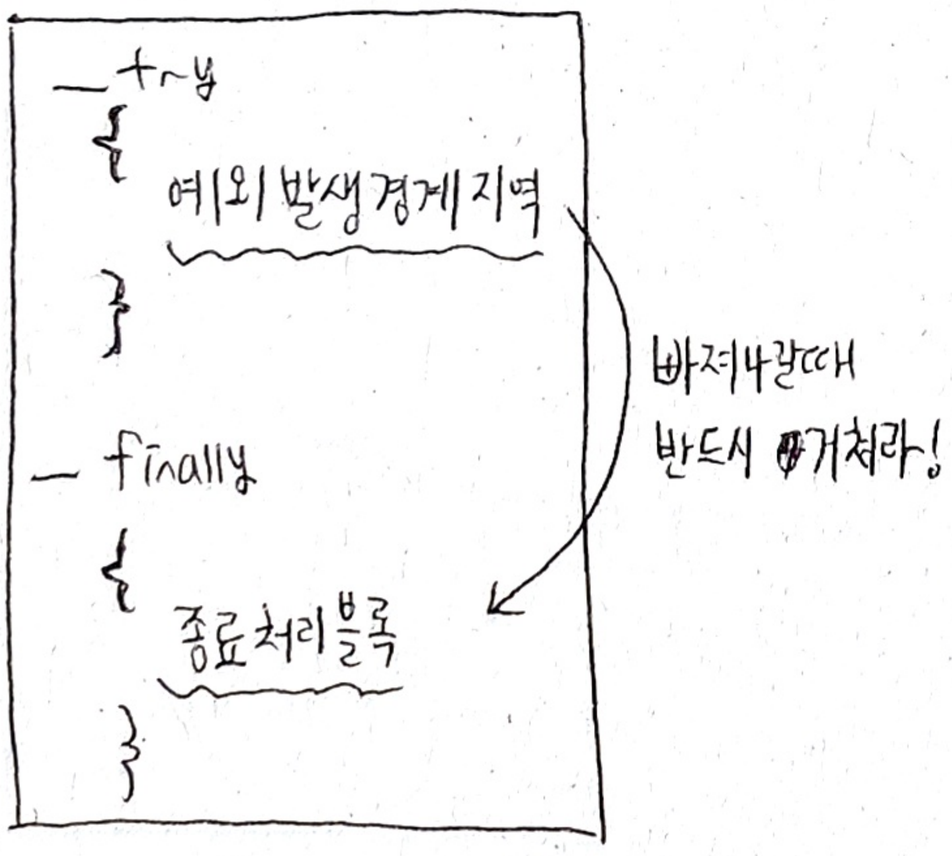
<Structured exception handler>

프로그램의 실제 흐름, 그 흐름에 대한 예외처리 영역을 구분! (분리)

- 컴파일 타임 오류 : 에러 (고쳐야 된다)
- 런타임 오류 : 일반적으로 예외 (처리되어야 된다)

<종료 핸들러>

- 예외 핸들러의 확장
- 파일에 개방과 이에 따른 종료 상황시 사용 (open/close)
- 메모리 할당 및 반환시 사용 (동적)



<예외 핸들러>

- 예외상황 발생시 선별적 실행

```
try
{
  // 예외 발생 경계지역
}

except(예외처리방식)
{
  //예외처리 위한 코드 지역
}
```

필터

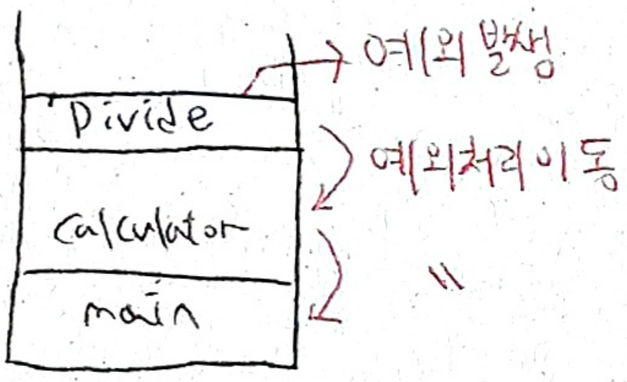
Exception_execute_Handler → 예외발생 이후의 나머지 문장 생략한다

_continue_execution → except블록 실행하지 않고 예외발생지점부터 다시 실행

_continue_search

예외발생시 그냥 pop up 해라
예외발생위치와 처리하려는 위치가 다르다
(별로 좋지 않는 방법)

- 처리되지 않은 예외의 이동



<Stack Memory>

- main에서도 예외처리못하면 window(os)가 프로그램 종료!

- 예외 핸들러는 중복이 가능하다.
  중복