데이터 중단점

*이 샘플은 Xbox One XDK 2015 년 11 월 이상 버전과 호환됩니다.*

# 설명

이 샘플에서는 다양한 유형의 메모리 액세스를 감지하는 데 유용한 하드웨어 데이터 중단점을 만드는 방법을 보여줍니다. 그것들은 전적으로 실행 속도에 영향을 미치지 않는다는 것을 의미하는 프로세서 코어에 의해 처리됩니다. 이 사용점의 주요 단점은 프로세서 코어에서 4 개의 슬롯 만 사용할 수 있다는 것입니다.

Windows는 스레드에 로컬 중단점만 허용합니다. 즉, 해당 스레드가 실행 중일때만 중단점이 활성화됩니다. 하드웨어 중단점은 스레드에 바인딩되어 있으므로 다양한 코어에서 실행될 때 스레드를 따라갑니다. 필요한 경우 각 스레드는 동일한 중단점을 사용하여 설정할 수 있습니다.

# 목차

## DataBreak.cpp/h

* 하드웨어 데이터 중단점을 설정하고 지울 수 있는 독립 실행형 패키지
* SetHardwareBreakPointForThread
  + 해당 슬롯 및 해당 주소에서 중단점을 설정합니다.
  + 슬롯이 이미 사용 중이면 덮어 씁니다.
* ClearHardwareBreakPointForThread
  + 지정된 슬롯에서 중단점을 지웁니다.

# 구현 정보

하드웨어 브레이크 포인트를 설정하려면 디버그 레지스터의 내용을 조정해야 합니다. 주요 단점은 레지스터에 대한 액세스가 커널에서만 사용 가능하다는 것입니다. 디버그 레지스터를 설정하는 트릭은 스레드 컨텍스트를 조정하는 것입니다. 이렇게 하면 컨텍스트 전환 중에 스케줄러가 내용을 디버그 레지스터에 기록합니다.

스레드의 컨텍스트는 스레드가 일시 중단된 경우에만 변경할 수 있습니다. 이 때문에 대상 스레드가 현재 스레드인 경우 다른 스레드가 작업을 수행하는 데 필요합니다. 샘플에서는 컨텍스트 전환을 수행할 임시 작업자 스레드를 만듭니다.

발생한 예외는 구조화 된 예외 처리 **(\_try**, **\_except**) 또는 처리되지 않은 예외 필터를 통해 잡히게 됩니다. 그러나 캡처 예외 레코드는 **MiniDumpWriteDump**와 함께 사용되는 경우 무시됩니다. 그러나, **MiniDumpWriteDump**은 문제가 되는 코드가 호출 스택에 계속 남아있는 예외를 발생시킨 스레드에서 호출되고 있습니다. 처리되지 않은 예외 필터는 문제가 되는 스레드의 컨텍스트에서 호출됩니다.

## 중요 사항:

이 시스템에서 발생한 예외는 OS에서 특별히 처리하는 단일 단계 예외입니다. 디버거가 연결되면 디버거가 먼저 잡아냅니다. 기본적으로 Visual Studio에서는 코드를 무시하고 컨트롤을 제목 예외 처리기로 전달합니다. 기본적으로 KD는 코드 줄에서 중단되어 예외를 발생시킵니다. 디버거가 연결되어 있지 않으면 **EnableKernelDebugging** 행동을 정의합니다. 이 기능을 사용하면 ERA가 고정됩니다. 콘솔이 디버거가 연결되기를 기다리는 중입니다. 만약 **EnableKernelDebugging**을 활성화되지 않으면 제목 예외 핸들러가 호출됩니다.

**DataBreakThread** 함수는 DataBreak.cpp의 디버그 레지스터에 필요한 내용을 문서화합니다.

# 업데이트 기록

초기 출시 2016년 10월