* 메모리 베리어

x,y,r1,r2 모두 0이고, Thread\_1(),Thread\_2() 함수를 이용함

텍스트이(가) 표시된 사진

자동 생성된 설명

만약 Thread\_1()을 task 하면, y=1, r1=0, Thread\_2()를 task하면 x=1, r2=1

텍스트이(가) 표시된 사진

자동 생성된 설명

Task를 하는 순간 r1과 r2는 0이 될 수가 없어보임

텍스트이(가) 표시된 사진

자동 생성된 설명

그러나 실제로는 빠져나오는 경우가 있는데, 이는 하드웨어가 최적화를 실행하기 떄문,

y = 1;

r1 = x; 이 식을

r1 = x;

y = 1; 이 순서로 마음대로 바꿔버림

* 이때에 순서를 고정하는 방법 : 메모리 베리어
* 메모리 베리어 : 코드 재배치를 막고, 가시성

텍스트이(가) 표시된 사진

자동 생성된 설명

Thread 함수에 Thread.MenoryBarrier를 사용하면 정상적으로 무한루프를 돌게됨 (코드 재배치 막기)

MemoryBarrier를 사용아면 앞뒤로 변수의 값을 실제 메모리에 저장하기 때문에, 가시성이 좋아진다.

텍스트이(가) 표시된 사진

자동 생성된 설명

예제