2022-2 멀티코어프로그래밍 중간고사

1. 1번
   1. X다른 코어 왔다갔다 할 수도 있고 하나에서 할수도 있음
   2. X메모리는 스레드와 상관없다.
   3. O성능향상이 없다. 있다면 프로그램에 문제가 있는거
   4. X프로세스가 다르면 스레드나 프로세스 문맥교환 똑같다. 효율적일 이유가 없다.
   5. X다른 스레드라도 같은 page table을 쓴다. 엔트리는 다를 수 있다. 똑 같은 가상 메모리를 사용하므로
   6. X스레드에서 스레드 생성 가능. 제한 없다. 메인 스레드만 스레드를 생성할 수 있는 것이 아님.
2. 없어지지 않음. 시분할 운영체제에서 CONTEXT SWITCH가 일어나기 때문에 임계영역 실행도중 다른 스레드가 실행될 수 있기 때문이다.
3. sum = sum +2는 atomic한 결과가 아니기 때문에 Data Race가 발생해서 틀린 결과가 나올 수 있다.
4. 4번
   1. O Data Race가 있기 때문에 컴파일러 최적화 문제가 발생하는 것임.
   2. X 캐시도 없고, 버퍼도 없고, 파이프라인도 없고,,, 한 CPU가 존재했었음 이는 메모리 일관성 문제가 발생하지 않음
   3. X 우리가 실습하는 것은 모두 데이터가 레벨3 캐시에 들어가 있음. 같은 CPU이므로. 그런데도 중간값 문제가 발생했다.
   4. O 소프트웨어로 구현했을 때 논블럭킹으로 구현할 수가 없음. 그래서 명령어로 구현하는 것임
   5. O 다른 스레드가 CAS에 성공하면 다시 루프를 돌아야
   6. X 가능은 하다. 하지만 비효율적이라 하드웨어의 도움을 받는다.
   7. O 작업이 골고루 더 분산이 되어서. 헤테로는 스레드마다 역할이 있어 병목현상 발생 시 분배되지 않는다.
5. 복수의 스레드가 하나의 캐시 라인을 반복적으로 접근하면서 생기는 현상. 반복되는 캐시 라인 INVALIDATION으로 전체적인 프로그램 실행 성능이 떨어지는 현상. (False Sharing만 cache thrashing을 일으키는게 아님.)
6. 10만분의 1 확률로 lock이 깨짐. 메모리 일관성이 깨지는 문제를 해결해야 함. 그러려면 fence를 넣어야 함.
7. 7번
   1. O 다른 스레드에서 delete 한걸 읽고 쓰고 하니까 문제임
   2. O 알아서 사라지니까 문제될거 없다.
   3. X 논블러킹이라는 보장이 없다. 블로킹 알고리즘으로 바뀔 수는 있다.
   4. X 가능
   5. X 존재하지도 않음.
8. Get\_ptr\_mark
   1. unsigned long long t = next;

\*removed = (t & 1) == 1;

return reinterpret\_cast<LF\_NODE\*>((t >> 1) << 1);

next를 미리 저장해놓고 가져다 쓸 것!

* 1. succ = curr->next.get\_ptr\_mark(&removed);

1. cas2
   1. 잘 실행된다. CAS2를 통해서 수정이 필요한 값들과 변동 없음을 확인해야 하는 값들을 모두 관리할 수 있기 때문이다. 검색을 하고 한방에 다 CAS를 해서. 가정이 동시에 할수 있는 CAS라고 되어 있다.
   2. 하드웨어에서 구현하고 있지 않기 때문에.
2. Lock free가 성능감소가 적다. RANGE가 감소하면 ADD/REMOVE 충돌확률이 커지고 mutex로 인한 성능 저하가 두드러지기 때문이다. 게으른 동기화의 성능이 더 많이 감소한다.