[2019-09-18 멀티코어 수업]

DMB - Data Memory Barrier

C++11의 atomic\_thread\_fence를 활용할 것.

atomic\_thread\_fence(std::moe

seq – Sequancial

cst – consistency

< \_asm mfence를 활용한 피터슨 알고리즘 >

[ 실행결과 ]

Threads [1] Sum = 100000000 Exec\_time = Time = 608ms

Threads [2] Sum = 100000000 Exec\_time = Time = 3125ms

프로그램 순서대로 메모리를 읽고 쓰지 않는다.

volatile은 컴파일에 요청하는 것이지 CPU에 요청하는 것이 아니다.

메모리일관성 PPT 2-27p Type-E는 피터슨알고리즘의 문제.

#include <iostream>

#include <thread>

using namespace std;

const int MAX = 50'000'000;

volatile int x, y;

volatile int trace\_x[MAX], trace\_y[MAX];

void thread\_x()

{

for(int i = 0; i<MAX; ++i)

{

x = i;

trace\_y[x] = y;

}

}

void thread\_y()

{

for (int i = 0; i < MAX; ++i)

{

y = i;

trace\_x[y] = x;

}

}

int main()

{

int count = 0;

thread t1{ thread\_x };

thread t2{ thread\_y };

t1.join();

t2.join();

for (int i = 0; i < (MAX - 1); ++i)

{

if (trace\_x[i] != trace\_x[i + 1]) continue;

int x = trace\_x[i];

if (trace\_y[x] != trace\_y[x + 1]) continue;

if (trace\_y[x] == i)

++count;

}

cout << "Number of Error = " << count << endl;

}

위 코드는 메모리의 일관성이 꺠지는 현상을 확인해보는 프로그램.

[ 실행 결과 ]

Number of Error = 554879

메모리 일관성이 55만번이나 깨졌다.

< Cache Boundary>

int a[64];

int temp = reinterpret\_cast<int>(&a[31]);

temp = (temp / 64) \* 64;

temp = temp - 2;

b = reinterpret\_cast<int\*>(temp);

\*b = 0;

[ 실행 결과 ]

Number of Error = 1180950

에러가 난 값을확인해보면 ffff ffff, 0000 0000인데 중간값인 ffff 0000, 0000 ffff가 보이는 것이다.

0하고 1이 바뀌는 과정에서 한번에 바뀌지 못해서 발생하는 문제이다

캐시에는 데이터 옆에 태그가 붙어있음. 어디서 가져온지에 대한 정보가 해당 태그에 있다.

태그의 크기가 데이터보다 크면 그것 또한 낭비이다.

컴파일러가 모든 integer주소는 4의 배수로 만들어준다.