**고급시스템프로그래밍 8주차 실시간수업 과제**

|  |
| --- |
| 제출일: 2021.04.29  이름: 김현용 |

1. 첨부의 mountain.tar에 대하여 다음을 수행하시오.
2. 본인이 사용하고 있는 컴퓨터 시스템에 대하여 다음을 조사하시오.

* CPU 종류(모델명)

1. Intel core i7-9750H

* CPU 주파수

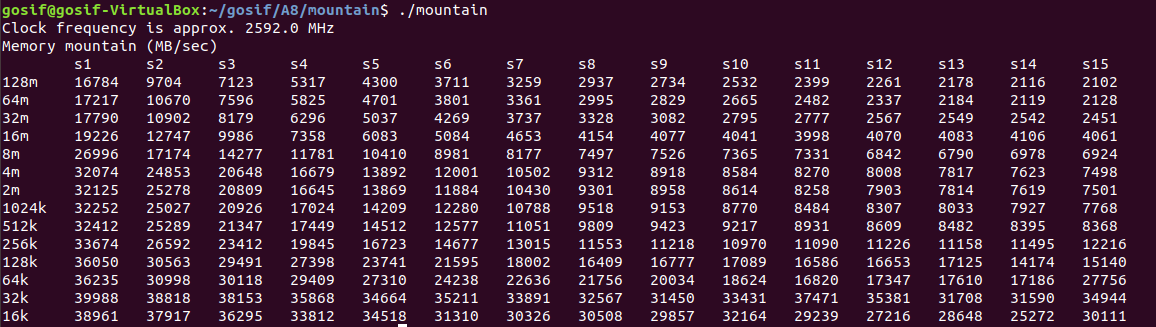
1. 2.6Ghz(기본), 4.5Ghz(최대)

* 코어 개수 및 스레드 수

1. 코어 개수 : 6개, 스레드 수 : 12개

* L1 Intr, L1 Data, L2, L3 캐쉬 사이즈 및 캐쉬 라인 사이즈 및 종류(예: direct mapped, set associative)

1. L1 I-Cache : 32KBytes\*6, 64-byte line size, 8-way set associative  
   L1 D-Cache : 32KBytes\*6, 64-byte line size, 8-way set associative  
   L2 Cache : 256KBytes\*6, 64-byte line size, 4-way set associative  
   L3 Cache : 12Mbytes, 64-byte line size, 16-way set associative
2. mountain 프로그램을 수행하고, 그 결과를 위 주어진 캐쉬 사이즈에 비추어 분석하시오.  
   A1-2)수행결과 및 설명



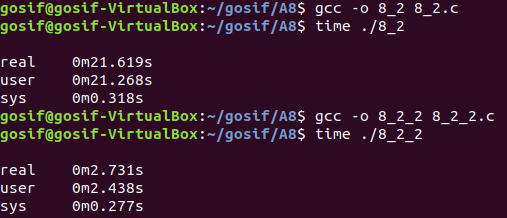
먼저 첫 줄에 2592.0MHz의 클럭 수가 나오는데, 조사한 대로 대략 2.6GHz가 나온다는 것을 알 수 있다.

CPU에서 캐쉬 메모리의 접근순서는 내 컴퓨터 기준으로 L1 I-Cache or L1 D-Cache -> L2 Cache -> L3 Cache 순이다 따라서 32k -> (32k\*6)192k -> (1.72\*4)6.912m -> 13.72m가 되는데 위의 표를 보면 알 수 있듯이 L1 I-Cache 한 개의 사이즈가 32k이므로 32k -> 64k 구간에서 급격히 감소하는 것을 알 수 있고, 128k->256k, 4m->8m, 8m->16m에서 급격히 감소하는 것을 확인할 수 있다.

1. 8\_2.c 프로그램에 대하여 물음에 답하시오.

|  |
| --- |
| #include <stdio.h>  #include <stdlib.h>  #define M 10000  #define TEST 10  void init\_array(double \*array, long n) {  unsigned long i;  for (i=0; i<n; i++)  array[i] = 1.0;  //array[i] = (rand()%100)/10.0;  return;  }  void mult (double\* a, double \*b, double \*c, int n) {  int i,j,k;  for (j=0; j<n; j++) {  for (i=0; i<n; i++) {  c[i] += a[i\*n+j]\*b[i\*n+j];  }  }  }  int main() {  double \*a = malloc(sizeof(double)\*M\*M);  double \*b = malloc(sizeof(double)\*M\*M);  double \*c = calloc(M, sizeof(double));  if (a==NULL || b == NULL || c == NULL)  exit(1);  init\_array(a, M\*M);  init\_array(b, M\*M);  for (int i = 0; i<TEST; i++)  mult(a,b,c,M);  } |

1. loop interchange 변환을 사용하여 i와 j 반복문의 순서를 바꿀 때 수행시간이 어떻게 되는지 조사하시오. (time 명령어를 사용)

  
mult 함수의 이중 for문에서 ji를 ij로 순서를 바꾼 결과 실행시간이 10배 이상 줄어드는 것을 확인할 수 있었다.

1. 1)의 프로그램 변환이 mult 함수에서 a, b, c 배열 접근의 공간 지역성 및 시간 지역성에 어떤 영향을 주는지 설명하시오.

c[i] += a[i\*n+j]\*b[i\*n+j]의 연산에서 for문 순서가 ji인 경우는 j = 0이고, i가 1씩 증가하기 때문에 a와 b 배열에서 a[1\*n+0] -> a[2\*n+0]의 식으로 공간 지역성에서 조금 먼 메모리를 접근하고, ij인 경우는 i = 0 이고, j가 1씩 증가하기 때문에 a와 b 배열에서 a[0+1] -> a[0+2]의 식으로 공간 지역성이 상대적으로 가까운 메모리를 접근하기 때문에 실행시간이 줄어 들었음을 알 수 있다.