InfoLab

IC621: Distributed and Parallel Computing

Lab 01 : SIMD

Min-Soo Kim

DGIST 대구경북과학기술원
Daegu Gyeongbuk Institute of Science & Technology

Intel Advanced Vector Extensions(AVX)

- **Environments**
  - Intel Sandy Bridge supports AVX instruction set
    - e.g., Intel i7 2600 CPU
  - gcc (> 4.6) supports AVX instruction set
- **Example code**
  - computing `sqrt(sqrt(sqrt(x)))` for 10M elements
  - three modes: naïve, SSE 4.2 (128 bits), AVX (256 bits)
- **Compilation**
  - `g++ simdTest.cpp -O3 -march=native -o simdTest`
- **Further study**
  - change the code so as to do inner-product between two vectors of 10 elements for 1M times

```
1.    #include <immintrin.h>
2.    #include <iostream>
3.    #include <math.h>
4.    #include <sys/time.h>

5.    using namespace std;

6.    // ------------------------------------------------------------------------
7.    // This function returns the current time, expressed as seconds since the Epoch
8.    // ------------------------------------------------------------------------
9.    double getCurrentTime(){
10.     struct timeval curr;
11.     struct timezone tz;
12.     gettimeofday(&curr, &tz);
13.     double tmp = static_cast<double>(curr.tv_sec) * static_cast<double>(1000000)
14.                + static_cast<double>(curr.tv_usec);
15.     return tmp*1e-6;
16.   }
```

```
17.   // ------------------------------------------------------------------------
18.   // Main routine
19.   // ------------------------------------------------------------------------
20.   int main() {

21.     srand48(0);             // seed PRNG
22.     double e,s;             // timestamp variables
23.     float *a, *b;           // data pointers
24.     float *pA,*pB;          // work pointer
25.     __m128 rA,rB;           // variables for SSE
26.     __m256 rA_AVX, rB_AVX; // variables for AVX

27.     // define vector size
28.     const int vector_size = 10000000;

29.     // allocate aligned blocks of memory
30.     a = (float*) _mm_malloc (vector_size*sizeof(float),32);
31.     b = (float*) _mm_malloc (vector_size*sizeof(float),32);

32.     // initialize vectors //
33.     for(int i=0;i<vector_size;i++) {
34.       a[i]=fabs(drand48());
35.       b[i]=0.0f;
36.     }
```

```
37.   // ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
38.   // Naive implementation
39.   // ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
40.     s = getCurrentTime();
41.     for (int i=0; i<vector_size; i++){
42.       b[i] = sqrtf(sqrtf(sqrtf(a[i])));
43.     }
44.     e = getCurrentTime();
45.     cout << (e-s)*1000 << " ms" << ", b[42] = " << b[42] << endl;

46.   // ----------------------------------------------------------------------------
47.     for(int i=0;i<vector_size;i++) {
48.       b[i]=0.0f;
49.     }
50.   // ----------------------------------------------------------------------------
```

```
51.   // ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
52.   // SSE2 implementation
53.   // ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
54.     pA = a; pB = b;

55.     s = getCurrentTime();
56.     for (int i=0; i<vector_size; i+=4){
57.       rA   = _mm_load_ps(pA); // loads four single-precision, floating-point values
58.       rB   = _mm_sqrt_ps(_mm_sqrt_ps(_mm_sqrt_ps(rA)));
59.       _mm_store_ps(pB,rB);
60.       pA += 4;
61.       pB += 4;
62.     }
63.     e = getCurrentTime();
64.     cout << (e-s)*1000 << " ms" << ", b[42] = " << b[42] << endl;

65.   // ------------------------------------------------------------------------------
66.     for(int i=0;i<vector_size;i++) {
67.       b[i]=0.0f;
68.     }
69.   // ------------------------------------------------------------------------------
```

```
70.   // ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
71.   // AVX implementation
72.   // ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
73.     pA = a; pB = b;

74.     s = getCurrentTime();
75.     for (int i=0; i<vector_size; i+=8){
76.       rA_AVX   = _mm256_load_ps(pA);
77.       rB_AVX   = _mm256_sqrt_ps(_mm256_sqrt_ps(_mm256_sqrt_ps(rA_AVX)));
78.       _mm256_store_ps(pB,rB_AVX);
79.       pA += 8;
80.       pB += 8;
81.     }
82.     e = getCurrentTime();
83.     cout << (e-s)*1000 << " ms" << ", b[42] = " << b[42] << endl;

84.     _mm_free(a);
85.     _mm_free(b);

86.     return 0;
87.   }
```