Assignment 1

Tejaswi Konduri (800965883)

Machine Used: Mamba

Composition: Number of CPU Cores = 8,

Number of Processors = 16,

CPU GHz = 3600/1024 = 3.51,

2 Ports,

2 FMA,

256/32 = 8 single precision.

Processor: Intel® Xeon® Processor E5-2667 v3 (20M Cache, 3.20 GHz)

Architecture: We have a Haswell architecture on Mamba compute nodes.

https://ark.intel.com/products/83361/Intel-Xeon-Processor-E5-2667-v3-20M-Cache-3_20-GHz

Formula to calculate the max operations performed per second

= Socket * Core * Frequency * Instructions * Operations/Instruction

Question: What is the maximum number of floating point operations this machine can perform per second?

Answer: ~ 1.6 TFlops/sec.

Question: What is the maximum number of integer operations this machine can perform per second?

Answer: ~ 1.3 TFlops/sec.

Question: Write a code that gets peak Flops

Answer:

```
#include <stdio.h>
#include <immintrin.h>
#include <omp.h>
#include <time.h>
#include <sys/time.h>

int main()
{
  //variable initialization...
  int iThreadCnt = 1, iLoopCntr = 1000*1000*1000;
  struct timeval start, end;
  float fTimeTaken = 0;
  int iCntr = 0;

  register __m256 vec1 = _mm256_setr_ps(4.0, 5.0, 13.0, 6.0, 4.0, 5.0, 13.0, 6.0);
  register __m256 vec2 = _mm256_setr_ps(9.0, 3.0, 6.0, 7.0, 4.0, 5.0, 13.0, 6.0);
  register __m256 vec3 = _mm256_setr_ps(1.0, 1.0, 1.0, 1.0, 4.0, 5.0, 13.0, 6.0);
  register __m256 vec4 = _mm256_setr_ps(4.0, 5.0, 13.0, 6.0, 4.0, 5.0, 13.0, 6.0);
  register __m256 vec5 = _mm256_setr_ps(9.0, 3.0, 6.0, 7.0, 4.0, 5.0, 13.0, 6.0);
  register __m256 vec6 = _mm256_setr_ps(1.0, 1.0, 1.0, 1.0, 4.0, 5.0, 13.0, 6.0);
  register __m256 vec7 = _mm256_setr_ps(4.0, 5.0, 13.0, 6.0, 4.0, 5.0, 13.0, 6.0);
  register __m256 vec8 = _mm256_setr_ps(9.0, 3.0, 6.0, 7.0, 4.0, 5.0, 13.0, 6.0);
  register __m256 vec9 = _mm256_setr_ps(1.0, 2.0, 3.0, 4.0, 4.0, 5.0, 13.0, 6.0);

  //start timer..
  gettimeofday(&start, NULL);
```

```
 //Parallel block
 #pragma omp parallel for default(shared)
 for (iCntr=0; iCntr < iLoopCntr; iCntr++)
 {
     if(iCntr == 0)
     {
          iThreadCnt = omp_get_num_threads();
     }
     __m256 result1 = _mm256_fmadd_ps(vec1, vec2, vec3);
     __m256 result2 = _mm256_fmadd_ps(vec4, vec5, vec6);
     __m256 result3 = _mm256_fmadd_ps(vec7, vec8, vec9);
     __m256 result4 = _mm256_fmadd_ps(result1, result2, result3);
     asm("");
 }


 //end timer..
 gettimeofday(&end, NULL);


 fTimeTaken = ((end.tv_sec * 1000000 + end.tv_usec) - (start.tv_sec * 1000000 + start.tv_usec));
//time in micro-sec


 printf("Number of flops = %f per sec.\n", (2 * 4 * (float)iLoopCntr * (float)iThreadCnt * (256./32.)
* 1000000) / fTimeTaken);
 // 2 for the no. of operations performed in one Fused Multiply add
 // 4 for the no. of FMAs performed inside the for loop


 return 0;
}
```

Question: Write a code that gets peak Iops

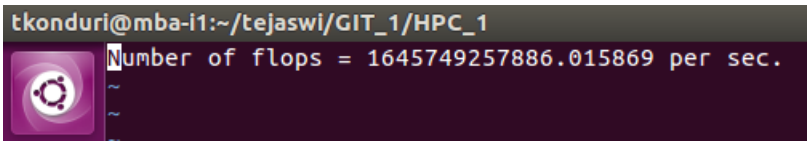
Answer:

```
#include <stdio.h>
#include <immintrin.h>
#include <omp.h>
#include <time.h>
#include <sys/time.h>

int main()
{
  //variable initialization...
  int iThreadCnt = 1, iLoopCntr = 1000*1000*1000;
  struct timeval start, end;
  float fTimeTaken = 0;
  int iCntr = 0;

  //start timer..
  gettimeofday(&start, NULL);

  //initializing variables...
  register  __m256i vec1 = _mm256_set_epi32(1, 2, 3, 4, 5, 6, 7, 8);
  register  __m256i vec2 = _mm256_set_epi32(9, 3, 6, 7, 9, 3, 6, 7);
  register  __m256i vec3 = _mm256_set_epi32(1, 1, 1, 1, 1, 1, 1, 1);
  register  __m256i vec4 = _mm256_set_epi32(4, 5, 3, 6, 4, 5, 1, 6);

  #pragma omp parallel for default(shared)
```

```
 for (iCntr=0; iCntr < iLoopCntr; iCntr++)
 {
     if(iCntr == 0)
     {
          iThreadCnt = omp_get_num_threads();
     }
     __m256i result1 = _mm256_add_epi32(vec1, vec2);
     __m256i result2 = _mm256_add_epi32(vec3, vec4);
     __m256i result3 = _mm256_sub_epi32(result1, result2);
     __m256i result4 = _mm256_add_epi32(result1, result2);
     asm("");
 }

 //end timer..
 gettimeofday(&end, NULL);

 fTimeTaken = ((end.tv_sec * 1000000 + end.tv_usec) - (start.tv_sec * 1000000 + start.tv_usec));
//time in micro-sec

 printf("Number of iops = %f per sec.\n", (4 * (float)iLoopCntr * (float)iThreadCnt * (256./32.) *
1000000) / fTimeTaken);
 // 4 for the no. of operations performed inside the for loop...

 return 0;
}
```

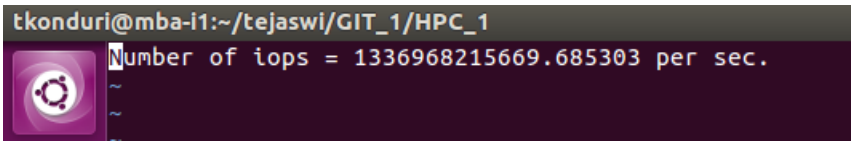
Question: How many Flops did the code achieve?

Answer: My Code achieved: **~ 1.6 TFlops/sec**



Question: How many Iops did the code achieve?

Answer: My Code achieved: **~ 1.3 TFlops/sec**



Question: Does that match expectation? Where does the discrepancy come from?

Answer: Yes, the output matches the expectation.

Question: Can you do better?

Answer: The code is able to achieve the maximum level expected.

Being said that, we are not suing the last variable result4 (in program to calculate Flops), so the compiler might optimize that, so that might be a pain point.

Question: Write a short report that explain your techniques, explanation and findings.

Answer:

To calculate the maximum FLOPS that can be performed on the machine, the steps I followed were:

1) Starting the timer
2) Using #pragma omp parallel for default(shared).
   I am using OpenMP here to run the 'for' loop code in parallel fashion among various threads and by default the variables will be shared across all the threads.
3) For the first time I am checking the number of threads are spawned in the processor. It would the stored in the iThreadCnt variable.
4) Then inside the for loop I am performing 4 FMA operations, where each operation will perform fused Multiplication and Addition in just one clock cycle.
5) Here, the 4th FMA operation is using the results of the first three FMA operations as their inputs.

6) I am then stopping the timer outside the for loop.
7) Then I am calculating the time taken in fTimeTaken variable.
8) Then finally, I am calculating the total FLOPS taken per second.
   Formula for FLOPS/sec = (No. of operations performed) * (iLoopCntr) * (iThreadCnt) * (256./32.) / (fTimeTaken).

To calculate the maximum IOPS that can be performed on the machine, the steps I followed were:

1) Starting the timer
2) Using #pragma omp parallel for default(shared).
   I am using OpenMP here to run the 'for' loop code in parallel fashion among various threads and by default the variables will be shared across all the threads.
3) For the first time I am checking the number of threads are spawned in the processor. It would the stored in the iThreadCnt variable.
4) Then inside the for loop I am performing 2 ADD operations followed by a SUBTRACT operation and finally followed by another ADD operation, where each operation will take one clock cycle.
5) Here, the 4^{th} ADD operation is using the results of the first 2 ADD operations as their inputs.
6) I am then stopping the timer outside the for loop.
7) Then I am calculating the time taken in fTimeTaken variable.
8) Then finally, I am calculating the total FLOPS taken per second.
   Formula for IOPS/sec = (No. of operations performed) * (iLoopCntr) * (iThreadCnt) * (256./32.) / (fTimeTaken).

Question: Submit an archive of the report and code. Or give a link to the git repository that stores that.

Answer: I have submitted the .zip file containing the source codes, scripts and the Report on Canvas.