Assignment 3

Lokesh Mohanty (SR no: 21014)

August 2022

1 Computer & Compiler Details

1.1 Basic Information

Architecture	x86_64
CPU op-mode(s)	32-bit, 64-bit
Address sizes	48 bits physical, 48 bits virtual
Byte Order	Little Endian
CPU(s)	16

1.2 CPU Details

Vendor ID	AuthenticAMD
Model name	AMD Ryzen 7 PRO 5875U with Radeon Graphics
CPU family	25
Model	80
Thread(s) per core	2
Core(s) per socket	8
Socket(s)	1
Stepping	0
Frequency boost	enabled
CPU(s) scaling MHz	44%
CPU max MHz	4546.8750
CPU min MHz	1600.0000

1.3 Cache

L1d cache	256 KiB (8 instances)
L1i cache	256 KiB (8 instances)
L2 cache	4 MiB (8 instances)
L3 cache	16 MiB (1 instance)

1.4 Compiler Details

Compiler	gcc (GCC)
Version	10.2.1 20201203

2 Program Code

2.1 Function to multiply 2 matrices

```
void matmul(float C[][N], float A[][N], float B[][N]) {
  for (int i = 0; i < N; i++) {
    for (int k = 0; k < N; k++) {
      for (int j = 0; j < N; j++) {
        C[i][j] += A[i][k] * B[k][j];
      }
    }
  }
}
```

2.2 Function to verify correctness of matrix multiplication

```
bool verifyMatmul(float C[][N], float A[][N], float B[][N]) {
  for (int count = 0; count < 10; count++) {
    int i = rand() % N, j = rand() % N;
    float actualValue = 0;
    for (int k = 0; k < N; k++) actualValue += A[i][k] * B[k][j];
    if (C[i][j] != actualValue) return false;
  }
  return true;
}
```

2.3 Function to generate input matrices

```
void generateMatrix(float matrix[][N]) {
  for (int i = 0; i < N; i++) {
    for (int j = 0; j < N; j++) {
      matrix[i][j] = rand() % N;
    }
  }
}
```

2.4 Function to measure matrix multiplication time

```
typedef void (*MatMul)(float [][N], float [][N], float [][N]);
void time(MatMul func, float C[][N], float A[][N], float B[][N]) {
  struct timeval start, end, timeTaken;

  gettimeofday(&start, NULL);
  func(C, A, B);
  gettimeofday(&end, NULL);

  bool isMatmulValid = verifyMatmul(C, A, B);
  printf("Is Valid Matmul: %s\n",
         isMatmulValid ? "true" : "false");
  timeTaken.tv_usec = end.tv_usec - start.tv_usec;
  timeTaken.tv_sec = end.tv_sec - start.tv_sec;
  if (timeTaken.tv_usec < 0) {
    timeTaken.tv_usec = 1000000 + timeTaken.tv_usec;
    timeTaken.tv_sec--;
  }
  printf("Time Taken: %ld.%lds\n",
         timeTaken.tv_sec, timeTaken.tv_usec);
}
```

2.5 Main Function

```
int main() {
  // using static to prevent segmentation fault for large N
  static float A[N][N], B[N][N], C[N][N] = {0.0};

  generateMatrix(A);
  generateMatrix(B);
  time(matmul, C, A, B);
  return 0;
}
```

2.6 Imports and Macros

```
#include <stdio.h>
#include <stdlib.h>
#include <sys/time.h>
#define N 1024
```

2.7 Bash script for profiling

```
#!/bin/bash

OPTIONS=("-O2"
         "-O2 -floop-interchange"
         "-O2 -fpeel-loops -funroll-loops"
         "-O2 -floop-interchange -fpeel-loops -funroll-loops"
         "-O2 -ftree-vectorize -fopt-info-vec"
         "-O2 -floop-interchange -fpeel-loops -funroll-loops -ftree-vectorize -fopt-info-vec")

for option in "${OPTIONS[@]}"; do
    gcc $option Ass3.c
    echo $option
    # remove pagecache
    sync
    sudo sh -c 'echo 1 > /proc/sys/vm/drop_caches'
    # time for i in {1..10}; do ./a.out; done
    for i in {1..10}; do
        perf stat -e cycles,instructions,cache-misses,cache-references ./a.out;
    done
done
```

3 Results & Observations

3.1 Results

With Basic optimizations
Flags: `-O2`

Time	$0.388s$
Cycles	1.693×10^9
Instructions	7.531×10^9
Instructions/Cycle	4.45

With loop interchange
Flags: `-O2 -floop-interchange`

Time	$0.393s$
Cycles	1.683×10^9
Instructions	7.531×10^9
Instructions/Cycle	4.47

With loop unrolling
Flags: `-O2 -fpeel-loops -funroll-loops`

Time	$0.385s$
Cycles	1.680×10^9
Instructions	4.707×10^9
Instructions/Cycle	2.90

With both loop interchange and unrolling
Flags: `-O2 -floop-interchange -fpeel-loops -funroll-loops`

Time	$0.386s$
Cycles	1.693×10^9
Instructions	4.707×10^9
Instructions/Cycle	2.79

With vectorization
Flags: `-O2 -ftree-vectorize -fopt-info-vec`

Time	$0.097s$
Cycles	0.373×10^9
Instructions	2.172×10^9
Instructions/Cycle	5.82

With loop interchange, loop unrolling and vectorization
Flags: `-O2 -floop-interchange -fpeel-loops -funroll-loops -ftree-vectorize -fopt-info-vec`

Time	$0.091s$
Cycles	0.395×10^9
Instructions	1.465×10^9
Instructions/Cycle	3.71

3.2 Observations

From the results we can see that we get the best optimization when we use a combination of loop-interchange, loop-unroll and vectorization optimization flags