Optymalizacja Kodu Na Różne Architektury
Ćwiczenie 3

1 Wprowadzenie

Rozważmy następującą procedurę uruchamiającą eliminację Gaussa bez pivotingu

Listing 1: Bazowa wersja kodu

```
//requires additional changes to the code to make it work

#include <stdio.h>
#include <stdlib.h>
#include <sys/time.h>
#include <time.h>

static double gtod_ref_time_sec = 0.0;

/* Adapted from the bl2_clock() routine in the BLIS library */

double dclock()
{
  double the_time, norm_sec;
  struct timeval tv;
  gettimeofday( &tv, NULL );
  if ( gtod_ref_time_sec == 0.0 )
    gtod_ref_time_sec = ( double ) tv.tv_sec;
  norm_sec = ( double ) tv.tv_sec - gtod_ref_time_sec;
  the_time = norm_sec + tv.tv_usec * 1.0e-6;
  return the_time;
}

int ge(double ** A, int SIZE)
```

```
{
  int i,j,k;
  for (k = 0; k < SIZE; k++) {
    for (i = k+1; i < SIZE; i++) {
      for (j = k+1; j < SIZE; j++) {
         A[i][j] = A[i][j]-A[k][j]*(A[i][k]/A[k][k]);
      }
    }
  }
  return 0;
}

int main( int argc, const char* argv[] )
{
  int i,j,k,iret;
  double dtime;
  int SIZE = 1500;
  double matrix[SIZE][SIZE]; // TODO - make near optimal dynamic allocation
  srand(1);
  for (i = 0; i < SIZE; i++) {
    for (j = 0; j < SIZE; j++) {
      matrix[i][j] = rand();
    }
  }
  printf("call GE");
  dtime = dclock();
  iret = ge(matrix);
  dtime = dclock()-dtime;
  printf("Time: %le \n", dtime);

  double check=0.0;
  for (i = 0; i < SIZE; i++) {
    for (j = 0; j < SIZE; j++) {
      check = check + matrix[i][j];
    }
  }
  printf( "Check: %le \n", check);
  fflush( stdout );


  return iret;
}
```

2 Ćwiczenia

Przy kolejnych krokach proszę wykonywać kopie aktualnych wersji stanu kodu. Będą potrzebne przy ostatnim punkcie oraz na kolejnych zajęciach. Proszę na bieżąco notować czasy wykonywania się obliczeń.

2.1

Proszę wprowadzić zaznaczone w kodzie modyfikacje a następnie skompilować procedurę bez optymalizacji. Dla zadanego rozmiaru macierzy w kolejnych uruchomieniach wartość zmiennej **check** nie powinna ulegać zmianie.

```
maciekw@Banach:~lab3> gcc ge1.c
maciekw@Banach:~lab3> ./a.out
call GETime: 5.806514e+00
Check: -4.605781e+16
maciekw@Banach:~/studenci/OORA/skrypty/lab3>
```

2.2

Proszę wykonać kopię procedury

```
maciekw@Banach:~lab3> cp ge1.c ge2.c
```

Proszę umieścić w rejestrach liczniki pętli.
Proszę skompilować procedurę z optymalizacją.
Jaki jest czas działania?

2.3

Proszę wykonać kopię procedury

```
maciekw@Banach:~lab3> cp ge2.c ge3.c
```

Proszę umieścić w rejestrze powtarzającą się w najbardziej zagnieżdżonym miejscu wartość.

```
multiplier = (A[i][k]/A[k][k]);
```

Proszę skompilować procedurę z optymalizacją.
Jaki jest czas działania?

2.4

Proszę wykonać kopię procedury

```
maciekw@Banach:~lab3> cp ge3.c ge4.c
```

Proszę ręcznie rozwinąć najbardziej zagnieżdżoną pętlę do 8 iteracji.

```
for (j = k+1; j < SIZE; ) {
  if (j < max(SIZE - BLKSIZE, 0)) {
    //PROSZE UZUPELNIC
    j += BLKSIZE;
  }
  else{
    A[i][j]=A[i][j]-A[k][j]*multiplier;
    j++;
  }
```

Proszę skompilować procedurę z optymalizacją.
Jaki jest czas działania?

2.5

Proszę wykonać kopię procedury

```
maciekw@Banach:~lab3> cp ge4.c ge5.c
```

Proszę zamienić deklarację macierzy na macierz jednowymiarową indeksowaną poprzez makro (patrz laboratorium 2).
Proszę skompilować procedurę z optymalizacją.
Jaki jest czas działania?

2.6

Proszę wykonać kopię procedury

```
maciekw@Banach:~lab3> cp ge5.c ge6.c
```

Proszę wprowadzić operacje wetorowe SSE3 (patrz laboratorium 2).

```
#include <x86intrin.h>
...
register __m128d mm_multiplier;
register __m128d tmp0, tmp1,...;
tmp0 = _mm_loadu_pd(...);
...
tmp0 = _mm_loadu_pd(A + IDX(i,j,SIZE));
tmp1 = _mm_loadu_pd(A + IDX(k,j,SIZE));
```

```
...
tmp1 = _mm_mul_pd(tmp1,mm_multiplier);
...
tmp0 = _mm_sub_pd(tmp0,tmp1);
...
_mm_storeu_pd(A + IDX(i,j,SIZE),tmp0);
```

Proszę skompilować procedurę z optymalizacją.
Jaki jest czas działania?

2.7

Proszę wykonać kopię procedury

```
maciekw@Banach:~lab3> cp ge6.c ge7.c
```

Proszę wprowadzić 256-bitowe operacje wetorowe AVX (patrz laboratorium 2).

```
#include <immintrin.h>
register __m256d ...
tmp0 = _mm256_loadu_pd(A + IDX(i,j,SIZE));
...
```

Proszę skompilować procedurę z optymalizacją.

```
maciekw@Banach:~lab3> gcc -mavx ge7.c
```

Jaki jest czas działania?

2.8

Proszę wszystkie uzyskane na Laboratoriach kody skompilować z opcją optymalizacji **-O2** oraz przetestować opcje **-march=native** oraz **-mfma**.

```
maciekw@Banach:~/studenci/OORA/skrypty/lab3> gcc -mavx -O2 ge7.c
maciekw@Banach:~/studenci/OORA/skrypty/lab3> ./a.out
call GETime: 3.880930e-01
Check: -4.605781e+16
maciekw@Banach:~lab3> gcc -mavx -O2 -march=native ge7.c
maciekw@Banach:~lab3> ./a.out
call GETime: 3.098970e-01
Check: -4.605781e+16
maciekw@Banach:~lab3> gcc -mavx -O2 -mfma ge7.c
maciekw@Banach:~lab3> ./a.out
call GETime: 3.976870e-01
Check: -4.605781e+16
maciekw@Banach:~lab3> gcc -mavx -O2 -march=native -mfma ge7.c
maciekw@Banach:~lab3> ./a.out
```

```
call GETime: 2.800250e-01
Check: -4.605781e+16
maciekw@Banach:~lab3>
```

Jaki jest ich czas działania?

3 Podsumowanie

Które metody optymalizacji dały najlepsze rezultaty?
Dlaczego? Jakie mechanizmy za nimi stały?