ФЕДЕРАЛЬНОЕ АГЕНТСТВО СВЯЗИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«СИБИРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ТЕЛЕКОММУНИКАЦИЙ И ИНФОРМАТИКИ»

Кафедра ВС

# Лабораторная работа №1 по дисциплине «Параллельные вычислительные технологии» по теме: «Оптимизация доступа к памяти, циклов и ветвлений»

Выполнил:
ст. гр. ИВ-823
Шиндель Э. Д.

Проверил:
Заведующий
кафедрой ВС
Курносов М. Г.

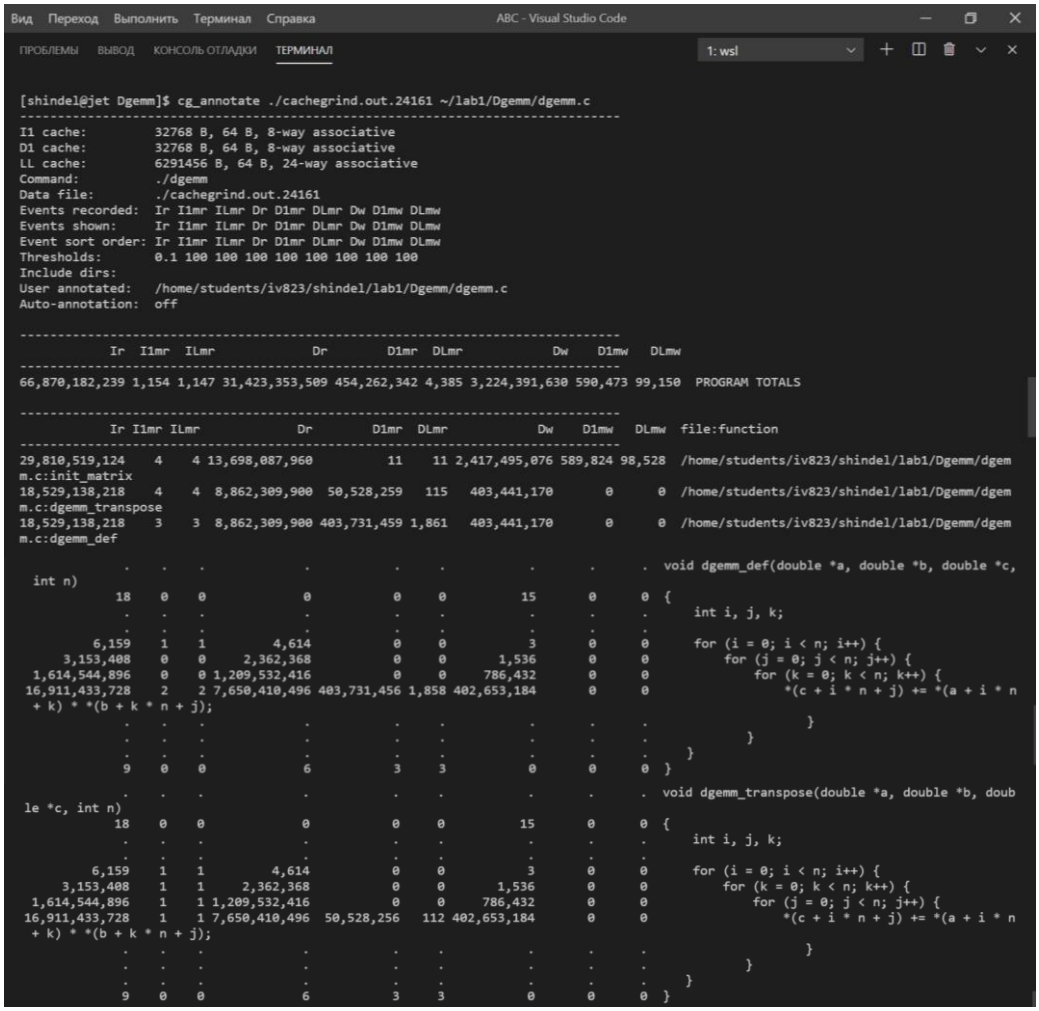
Новосибирск, 2020

## Содержание:

# Задание 1

| **dgemm_def** | **dgemm_transpose** | | **dgemm_block** | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| Time, s | Time, s | Speedup | BS = 2 | | BS = 4 | | BS = 8 | | BS = 16 | |
| | | | Time, s | Speedup | Time, s | Speedup | Time, s | Speedup | Time, s | Speedup |
| **1,727** | **1,299** | **1,33** | **1,457** | **1,19** | **1,077** | **1,6** | **0,966** | **1,788** | **0,941** | **1,84** |

Эксперимент проводился на кластере Jet




Аннотированный исходный текст программы, сформированный с помощью Valgrind

```
[shindel@jet Dgemm]$ perf record -e cache-misses ./dgemm
dgemm_def = 1.734455 sec.
dgemm_transpose = 1.302567 sec.
[ perf record: Woken up 1 times to write data ]
[ perf record: Captured and wrote 0.184 MB perf.data (4804 samples) ]
[shindel@jet Dgemm]$
```

```
Вид  Переход  Выполнить  Терминал  Справка                    ABC - Visual Studio Code

ПРОБЛЕМЫ  ВЫВОД  КОНСОЛЬ ОТЛАДКИ  ТЕРМИНАЛ                    1: wsl

Samples: 3K of event 'cache-misses:u', 4000 Hz, Event count (approx.): 89341
dgemm_def  /home/students/iv823/shindel/lab1/Dgemm/dgemm
Percent       dgemm_def():
              };

              double A[N * N], B[N * N], C[N * N];

              void dgemm_def(double *a, double *b, double *c, int n)
              {
                push   %rbp
                mov    %rsp,%rbp
                mov    %rdi,-0x18(%rbp)
                mov    %rsi,-0x20(%rbp)
                mov    %rdx,-0x28(%rbp)
                mov    %ecx,-0x2c(%rbp)
                  int i, j, k;

                  for (i = 0; i < n; i++) {
                movl   $0x0,-0x4(%rbp)
              ↓ jmpq   f7
                      for (j = 0; j < n; j++) {
              1f:     movl   $0x0,-0x8(%rbp)
              ↓ jmpq   e7
                          for (k = 0; k < n; k++) {
              2b:     movl   $0x0,-0xc(%rbp)
              ↓ jmpq   d7
                              *(c + i * n + j) += *(a + i * n + k) * *(b + k * n + j);
 0.28         37:     mov    -0x4(%rbp),%eax
                      imul   -0x2c(%rbp),%eax
                      movslq %eax,%rdx
 0.06                 mov    -0x8(%rbp),%eax
 0.06                 cltq
                      add    %rdx,%rax
                      lea    0x0(,%rax,8),%rdx
 0.08                 mov    -0x28(%rbp),%rax
                      add    %rdx,%rax
 0.20                 movsd  (%rax),%xmm1
 8.22                 mov    -0x4(%rbp),%eax
                      imul   -0x2c(%rbp),%eax
                      movslq %eax,%rdx
                      mov    -0xc(%rbp),%eax
                      cltq
                      add    %rdx,%rax
                      lea    0x0(,%rax,8),%rdx
                      mov    -0x18(%rbp),%rax
                      add    %rdx,%rax
                      movsd  (%rax),%xmm2
 2.57                 mov    -0xc(%rbp),%eax
                      imul   -0x2c(%rbp),%eax
                      movslq %eax,%rdx
                      mov    -0x8(%rbp),%eax
 0.03                 cltq
                      add    %rdx,%rax
 0.03                 lea    0x0(,%rax,8),%rdx
                      mov    -0x20(%rbp),%rax
 0.03                 add    %rdx,%rax
                      movsd  (%rax),%xmm0
87.91                 mulsd  %xmm2,%xmm0
 2.40                 mov    -0x4(%rbp),%eax
                      imul   -0x2c(%rbp),%eax
                      movslq %eax,%rdx
                      mov    -0x8(%rbp),%eax
 0.06                 cltq
                      add    %rdx,%rax
                      lea    0x0(,%rax,8),%rdx
                      mov    -0x28(%rbp),%rax
                      add    %rdx,%rax
                      addsd  %xmm1,%xmm0
 0.03                 movsd  %xmm0,(%rax)
                          for (k = 0; k < n; k++) {
 0.03                 addl   $0x1,-0xc(%rbp)
              d7:     mov    -0xc(%rbp),%eax
                      cmp    -0x2c(%rbp),%eax
              ↑ jl     37
                      for (j = 0; j < n; j++) {
                      addl   $0x1,-0x8(%rbp)
              e7:     mov    -0x8(%rbp),%eax
                      cmp    -0x2c(%rbp),%eax
              ↑ jl     2b
                    for (i = 0; i < n; i++) {
                      addl   $0x1,-0x4(%rbp)
              f7:     mov    -0x4(%rbp),%eax
 0.03                 cmp    -0x2c(%rbp),%eax
              ↑ jl     1f
                                  }
                              }
                          }
                      }
                nop
                pop    %rbp
              ← retq
Press 'h' for help on key bindings
```

Аннотированный исходный текст программы, сформированный с помощью профилировщика perf

# Задание 2

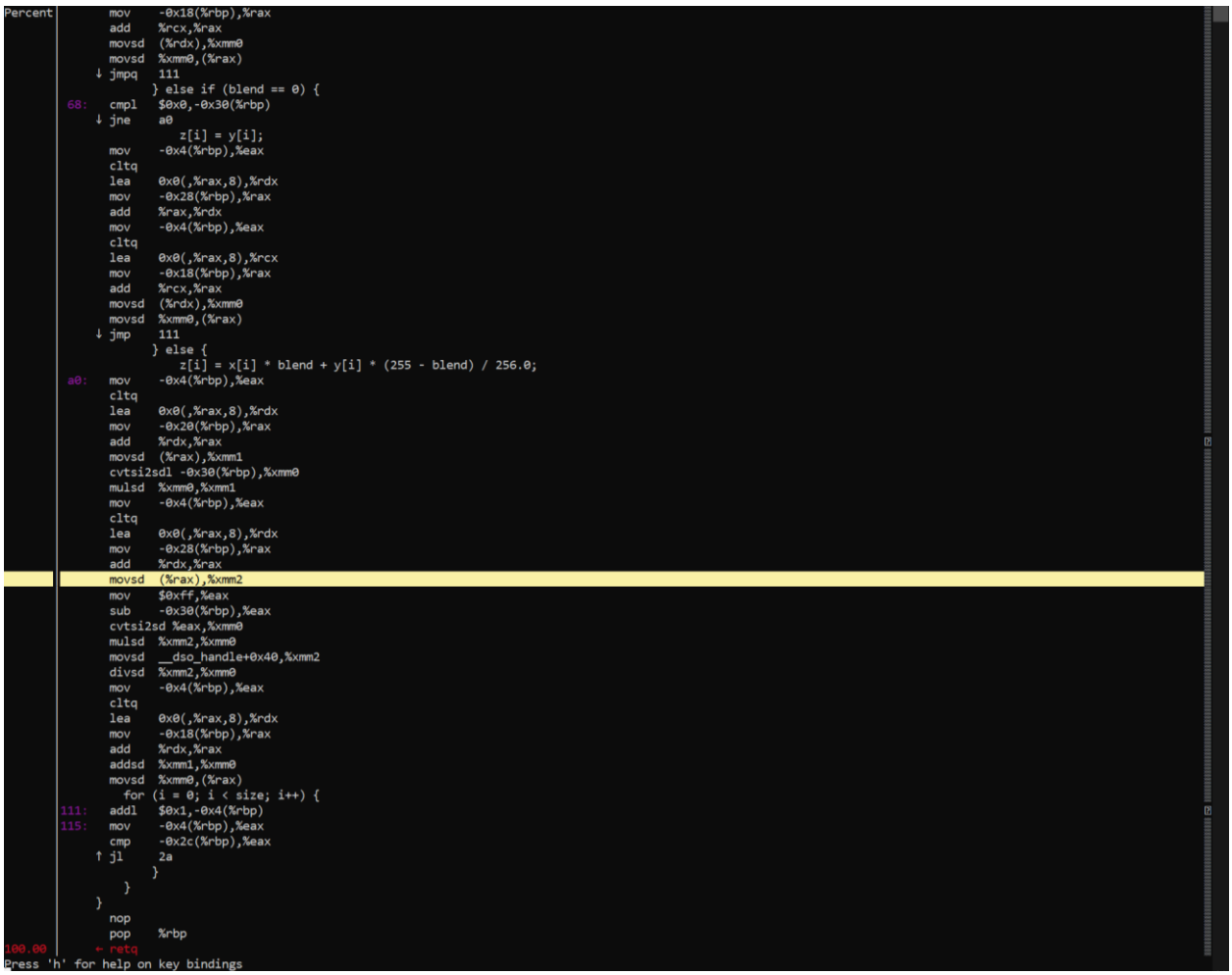
| n | Время выполнения функции blend_map | Время выполнения функции blend_map_opt | Ускорение (speedup) |
|---|---|---|---|
| 100 000 | 0,000557 | 0,00039 | **1,43** |
| 1 000 000 | 0,00603 | 0,004227 | **1,43** |

```
[shindel@jet Branch]$ perf stat -e branch-misses ./branch
hpctimer: Initializing timer...
hpctimer: TSC ticks per second: 2493682740 (2.49 GHz)
First run (sec.): 0.015098
Mean of 20 runs (sec.): 0.004170

 Performance counter stats for './branch':

            4358      branch-misses:u

      3.100905374 seconds time elapsed

      0.092524000 seconds user
      0.008045000 seconds sys
```

Количество ветвлений в bland_map

```
[shindel@jet Branch]$ perf record -e branch-misses ./branch
hpctimer: Initializing timer...
hpctimer: TSC ticks per second: 2493692612 (2.49 GHz)
First run (sec.): 0.013947
Mean of 20 runs (sec.): 0.006307
[ perf record: Woken up 1 times to write data ]
[ perf record: Captured and wrote 0.001 MB perf.data (16 samples) ]
[shindel@jet Branch]$
```

```
Samples: 2  of event 'branch-misses:u', 4000 Hz, Event count (approx.): 28
blend_map  /home/students/iv823/shindel/lab1/Branch/branch
Percent


    Disassembly of section .text:

    0000000000400567 <blend_map>:
    blend_map():
    };

    double x[n], y[n], z[n];

    void blend_map(double *z, double *x, double *y, int size, int blend)
    {
      push   %rbp
      mov    %rsp,%rbp
      mov    %rdi,-0x18(%rbp)
      mov    %rsi,-0x20(%rbp)
      mov    %rdx,-0x28(%rbp)
      mov    %ecx,-0x2c(%rbp)
      mov    %r8d,-0x30(%rbp)
        int i = 0;
      movl   $0x0,-0x4(%rbp)

        for (i = 0; i < size; i++) {
      movl   $0x0,-0x4(%rbp)
    ↓ jmpq   115
            if (blend == 255) {
 2a:  cmpl   $0xff,-0x30(%rbp)
    ↓ jne    68
               z[i] = x[i];
      mov    -0x4(%rbp),%eax
      cltq
      lea    0x0(,%rax,8),%rdx
      mov    -0x20(%rbp),%rax
      add    %rax,%rdx
      mov    -0x4(%rbp),%eax
      cltq
      lea    0x0(,%rax,8),%rcx
```




Аннотированный исходный код программы bland_map

```
[shindel@jet Branch]$ perf stat -e branch-misses ./branch
hpctimer: Initializing timer...
hpctimer: TSC ticks per second: 2493697800 (2.49 GHz)
First run (sec.): 0.014062
Mean of 20 runs (sec.): 0.004258

 Performance counter stats for './branch':

            4239      branch-misses:u

       3.101422863 seconds time elapsed

       0.091947000 seconds user
       0.009093000 seconds sys
```

Количество ветвлений в bland_map_opt

# Задание 3

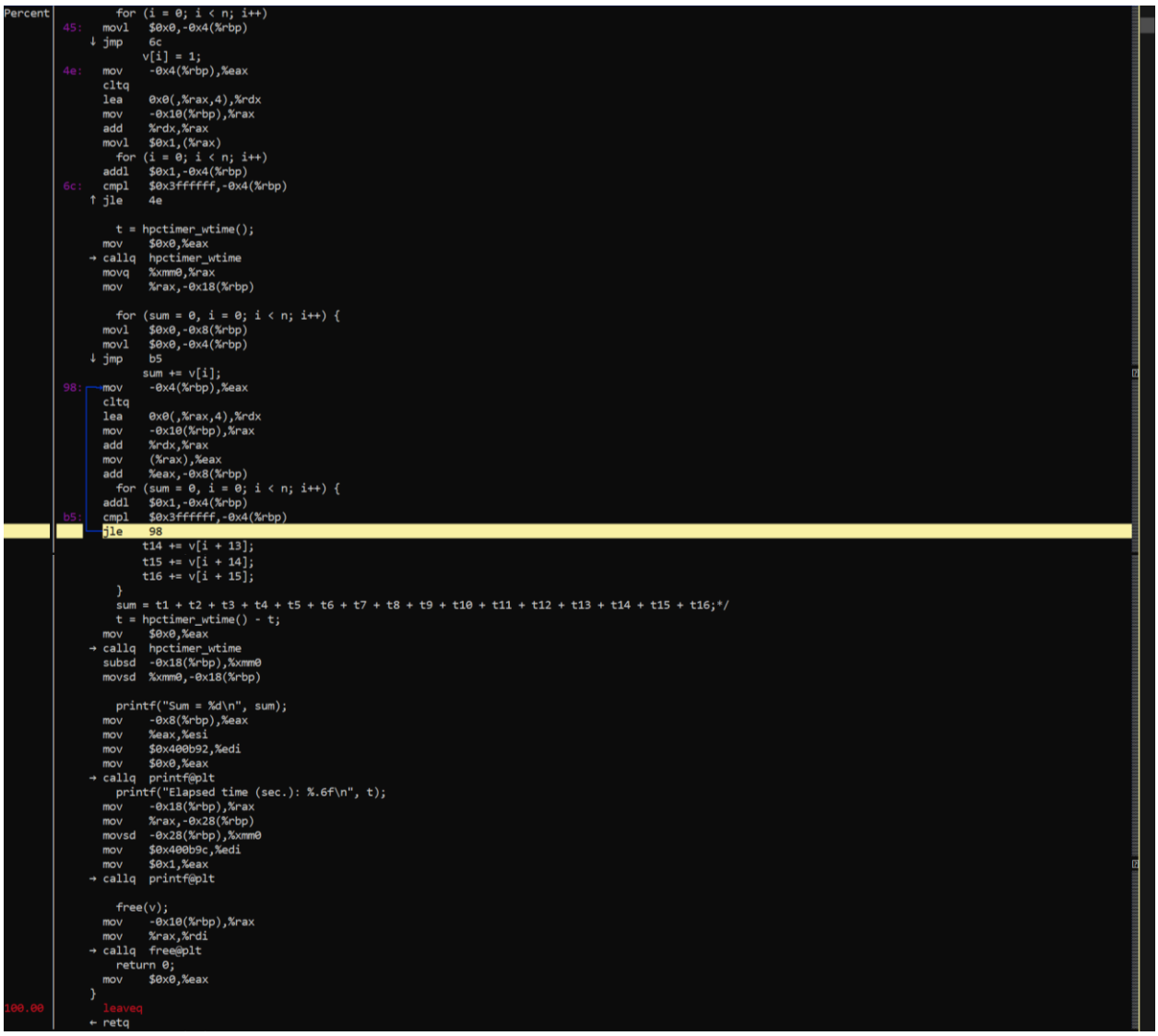
| n | Глубина раскрутки цикла | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | -- | 2 | | 4 | | 8 | | 16 | |
| | Time | Time | Speedup | Time | Speedup | Time | Speedup | Time | Speedup |
| 16 MiB | 0,073 | 0,048 | 1,52 | 0,039 | 1,87 | 0,037 | 1,97 | 0,035 | 2,09 |
| 64 MiB | 0,285 | 0,189 | 1,51 | 0,157 | 1,82 | 0,147 | 1,94 | 0,138 | 2,07 |

```
[shindel@jet Loop]$ perf stat -e branch-misses ./loop
hpctimer: Initializing timer...
hpctimer: TSC ticks per second: 2493700077 (2.49 GHz)
Sum = 67108864
Elapsed time (sec.): 0.286126

 Performance counter stats for './loop':

            4350      branch-misses:u

       3.812489618 seconds time elapsed

       0.539931000 seconds user
       0.191267000 seconds sys
```

Количество ветвлений в исходной программе

```
[shindel@jet Loop]$ perf record -e branch-misses ./loop
hpctimer: Initializing timer...
hpctimer: TSC ticks per second: 2493688980 (2.49 GHz)
Sum = 67108864
Elapsed time (sec.): 0.287896
[ perf record: Woken up 1 times to write data ]
[ perf record: Captured and wrote 0.001 MB perf.data (9 samples) ]
[shindel@jet Loop]$
```

```
Disassembly of section .text:

0000000004006c7 <main>:
main():
#include "hpctimer.h"

enum { n = 64 * 1024 * 1024 };

int main()
{
  push   %rbp
  mov    %rsp,%rbp
  sub    $0x30,%rsp
    int *v, i, sum;
    double t;

    if ( (v = malloc(sizeof(*v) * n)) == NULL) {
  mov    $0x10000000,%edi
→ callq  malloc@plt
  mov    %rax,-0x10(%rbp)
  cmpq   $0x0,-0x10(%rbp)
↓ jne    45
        fprintf(stderr, "No enough memory\n");
  mov    stderr@@GLIBC_2.2.5,%rax
  mov    %rax,%rcx
  mov    $0x11,%edx
  mov    $0x1,%esi
  mov    $0x400b80,%edi
→ callq  fwrite@plt
        exit(EXIT_FAILURE);
  mov    $0x1,%edi
→ callq  exit@plt
    }
```




Аннотированный исходный код программы

```
[shindel@jet Loop]$ perf stat -e branch-misses ./loop
hpctimer: Initializing timer...
hpctimer: TSC ticks per second: 2493691285 (2.49 GHz)
Sum = 67108864
Elapsed time (sec.): 0.136914

 Performance counter stats for './loop':

            4318      branch-misses:u

      3.595893667 seconds time elapsed

      0.399477000 seconds user
      0.194745000 seconds sys
```

Количество ветвлений в программе с развёрнутым циклом

# Листинг

```
/*
 * dgemm.c: DGEMM - Double-precision General Matrix Multiply.
 *
 */

#include <stdio.h>
#include <stdlib.h>

#include "hpctimer.h"

enum {
    N = 512,
    NREPS = 3
};

double A[N * N], B[N * N], C[N * N];

void dgemm_def(double *a, double *b, double *c, int n)
{
    int i, j, k;

    for (i = 0; i < n; i++) {
        for (j = 0; j < n; j++) {
            for (k = 0; k < n; k++) {
                *(c + i * n + j) += *(a + i * n + k) * *(b + k * n + j);
            }
```

```
        }
    }
}

void dgemm_transpose(double *a, double *b, double *c, int n)
{
    int i, j, k;

    for (i = 0; i < n; i++) {
        for (k = 0; k < n; k++) {
            for (j = 0; j < n; j++) {
                *(c + i * n + j) += *(a + i * n + k) * *(b + k * n + j);
            }
        }
    }
}

void dgemm_block(double *a, double *b, double *c, int n, int BS)
{
    int i, j, k, i0, j0, k0;
    double *a0, *b0, *c0;

    for (i = 0; i < n; i += BS) {
        for (j = 0; j < n; j += BS) {
            for (k = 0; k < n; k += BS) {
                for (i0 = 0, c0 = (c + i * n + j), a0 = (a + i * n + k);
                                        i0 < BS; ++i0, c0 += n, a0 += n) {
```

```
                    for (k0 = 0, b0 = (b + k * n + j);
                                                k0 < BS; ++k0, b0 += n) {
                        for (j0 = 0; j0 < BS; ++j0) {
                            c0[j0] += a0[k0] * b0[j0];
                        }
                    }
                }
            }
        }
    }

void init_matrix(double *a, double *b, double *c, int n)
{
    int i, j, k;

    for (i = 0; i < n; i++) {
        for (j = 0; j < n; j++) {
            for (k = 0; k < n; k++) {
                *(a + i * n + j) = 1.0;
                *(b + i * n + j) = 2.0;
                *(c + i * n + j) = 0.0;
            }
        }
    }
}
```

```
void print_matrix(double *a, int n)
{
    int i, j;

    printf("Matrix:\n");
    for (i = 0; i < n; i++) {
        for (j = 0; j < n; j++) {
            printf("%12.2f", *(a + i * n + j));
        }
        printf("\n");
    }
}

int main(int argc, char **argv)
{
    int i;
    double t1, t2, tsum = 0.0;

    for (i = 0; i < 3; i++) {
        init_matrix(A, B, C, N);
        t1 = hpctimer_getwtime();
        dgemm_def(A, B, C, N);
        t1 = hpctimer_getwtime() - t1;
    tsum += t1;
    }
    t1 = tsum / 3;
    tsum = 0.0;
```

```
    for (i = 0; i < 3; i++) {
        init_matrix(A, B, C, N);
    t2 = hpctimer_getwtime();
        dgemm_transpose(A, B, C, N);
        t2 = hpctimer_getwtime() - t2;
    tsum += t2;
    }
    t2 = tsum / 3;
    tsum = 0.0;

    /*int BS = 16;
    for (i = 0; i < 3; i++) {
        init_matrix(A, B, C, N);
        t3 = hpctimer_getwtime();
        dgemm_block(A, B, C, N, BS);
        t3 = hpctimer_getwtime() - t3;
        tsum += t3;
    }
    t3 = tsum / 3;

    printf("dgemm_def = %.6f sec.\ndgemm_transpose = %.6f sec.\ndgemm_block =
 %.6f sec (BS = %d)\n", t1, t2, t3, BS);*/
    printf("dgemm_def = %.6f sec.\ndgemm_transpose = %.6f sec.\n", t1, t2);

    return 0;
}
```

```
*
 * branch.c:
 *
 */
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

#include "hpctimer.h"

enum {
    n = 1000000,
    nreps = 20
};

double x[n], y[n], z[n];

void blend_map(double *z, double *x, double *y, int size, int blend)
{
    int i = 0;

    for (i = 0; i < size; i++) {
        if (blend == 255) {
            z[i] = x[i];
        } else if (blend == 0) {
            z[i] = y[i];
        } else {
```

```
            z[i] = x[i] * blend + y[i] * (255 - blend) / 256.0;
        }
    }
}

void blend_map_opt(double *dest, double *a, double *b, int size, int blend)
{
    int i = 0;

    if (blend == 255) {
        for (i = 0; i < size; i++) {
            z[i] = x[i];
    }
    } else if (blend == 0) {
        for (i = 0; i < size; i++) {
        z[i] = y[i];
    }
    } else {
    for (i = 0; i < size; i++) {
            z[i] = x[i] * blend + y[i] * (255 - blend) / 256.0;
        }
    }
}

int main()
{
    double tfirst, t;
```

```
    int i;

    /* First run: warmup */
    tfirst = hpctimer_wtime();
    //blend_map(z, x, y, n, 0);
    blend_map_opt(z, x, y, n, 0);
    tfirst = hpctimer_wtime() - tfirst;

    /* Measures */
    t = hpctimer_wtime();
    for (i = 0; i < nreps; i++) {
        //blend_map(z, x, y, n, 0);
        blend_map_opt(z, x, y, n, 0);
    }
    t = (hpctimer_wtime() - t) / nreps;

    printf("First run (sec.): %.6f\n", tfirst);
    printf("Mean of %d runs (sec.): %.6f\n", nreps, t);

    return 0;
}
```

```
/*
 * loop.c:
 *
 */
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

#include "hpctimer.h"

enum { n = 64 * 1024 * 1024 };

int main()
{
    int *v, i, sum;
    double t;

    if ( (v = malloc(sizeof(*v) * n)) == NULL) {
        fprintf(stderr, "No enough memory\n");
        exit(EXIT_FAILURE);
    }

    for (i = 0; i < n; i++)
        v[i] = 1;

    t = hpctimer_wtime();
```

```
    for (sum = 0, i = 0; i < n; i++) {
        sum += v[i];
    }
    /*int t1, t2, t3, t4, t5, t6, t7, t8, t9, t10, t11, t12, t13, t14, t15,
t16;
    t1 = t2 = t3 = t4 = t5 = t6 = t7 = t8 = t9 = t10 = t11 = t12 = t13 = t14
                                                        = t15 = t16 = 0;
    for (sum = 0, i = 0; i < n; i += 16) {
        t1 += v[i];
        t2 += v[i + 1];
        t3 += v[i + 2];
        t4 += v[i + 3];
        t5 += v[i + 4];
        t6 += v[i + 5];
        t7 += v[i + 6];
        t8 += v[i + 7];
        t9 += v[i + 8];
        t10 += v[i + 9];
        t11 += v[i + 10];
        t12 += v[i + 11];
        t13 += v[i + 12];
        t14 += v[i + 13];
        t15 += v[i + 14];
        t16 += v[i + 15];
    }
    sum = t1 + t2 + t3 + t4 + t5 + t6 + t7 + t8 + t9 + t10 + t11 + t12 + t13
                                                    + t14 + t15 + t16;*/
```

```
    t = hpctimer_wtime() - t;


    printf("Sum = %d\n", sum);
    printf("Elapsed time (sec.): %.6f\n", t);


    free(v);
    return 0;
}
```