МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

НОВОСИБИРСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Факультет информационных технологий

Кафедра параллельных вычислений

ОТЧЕТ

О ВЫПОЛНЕНИИ ПРАКТИЧЕСКОЙ РАБОТЫ

«Вычисления задержки и темпа выдачи результатов векторного вещественного вычисления квадратного корня»

студента 3 курса, группы 20203

Синюкова Валерия Константиновича

Направление 09.03.01 – «Информатика и вычислительная техника»

Преподаватель:
доцент кафедры параллельных вычислений
Власенко Андрей Юрьевич

Новосибирск 2023

СОДЕРЖАНИЕ

Цель

Научиться оценивать производительность микропроцессора на заданных операциях.

Постановка задачи

Вычислить задержку и темп выдачи результатов инструкции векторного вещественного вычисления квадратного корня, то есть количество тактов работы ЦП, за которое выполняется одна инструкция в последовательности зависимых и независимых операций.

Описание работы

1) Была написана программа на языке C, в рамках которой выполняются две длинные последовательности одной и той же инструкции векторного вещественного вычисления квадратного корня, в одном случае операнды операций внутри последовательности не зависят друг от друга (вычисляется задержка), в другом – зависят (вычисляется темп выдачи результатов). Для измерения количества тактов необходимых для выполнения операций используется машинная команда rdtsc, для сериализации, то есть для предотвращения перемешивания инструкций вычислений квадратного корня и считывания счетчика тактов процессора использовалась машинная команда cpuid. С листингом данной программы вы можете ознакомиться в приложении №1.
2) Ассемблерный листинг вышеописанной программы был изучен и скорректирован таким образом, чтобы между инструкциями rdtsc находился только цикл, состоящий из инструкций, время работы которых измерялось, и время выполнения посторонних инструкций не влияло на результат, а также для того, чтобы изучаемые инструкции работали только с регистрами, а не с оперативной памятью. С ассемблерным листингом можно ознакомиться в приложении №2.
3) Данная программа была протестирована на ноутбуке с процессором Intel core i3-7100U с микроархитектурой Kaby Lake, а также на суперкомпьютере НГУ с процессором Intel Xeon E5-2680 v3 с микроархитектурой Haswell. Всего проводилось 10 независимых друг от друга запусков вышеописанной программы, для чего были написаны два bash-скрипта, с которыми вы можете ознакомиться в приложении №3 и приложении №4. Результаты полученных измерений представлены далее в виде таблиц (поля «мин» и «теория» будут пояснены позднее).

   Kaby Lake:

номер эксперимента	задержка (такт)	темп выдачи результатов (такт)
1	16,810	5,880
2	15,911	5,051
3	16,159	5,457
4	16,135	5,056

5	15,777	5,343
6	16,064	5,347
7	15,476	5,329
8	15,778	5,095
9	15,674	5,853
10	17,569	5,359
мин	**15,476**	**5,051**
теория	**15-16**	**4-6**

Haswell:

номер эксперимента	**задержка (такт)**	**темп выдачи результатов (такт)**
1	18,494	12,363125
2	23,732	12,51375
3	23,75	11,92
4	20,436	11,783125
5	23,15	12,640625
6	17,394	12,099375
7	23,506	12,904375
8	18,994	12,053125
9	17,556	12,3475
10	16,85	12,91
мин	**16,850**	**11,783**
теория	**16**	**8-14**

4) Из 10 полученных результатов были выбраны минимальные (поля «мин» в таблицах), они были сравнены с “теоретическими” (поля «теория» в таблицах), которые взяты из документа Instruction tables by Agner Fog. Technical University of Denmark для соответствующих микроархитектур. Как можно заметить, реально полученные результаты с хорошей точностью соотносятся с теоретическими.

Выводы

В результате данной практической работы была написана программа на языке C для измерения задержки и темпа выдачи результатов векторного вещественного вычисления квадратного корня. Тестирование данной программы было произведено на двух машинах с разными центральными процессорами, результаты тестирования были сравнены с теоретическими.

Приложение №1. Листинг С-программы

```
#define N 2000
#define NUM_OF_ITERATIONS 1000
#define BUF_SIZE 256

#include <fcntl.h>
#include <xmmintrin.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>

void write_to_file(double value, int fd)
{
    char buf[BUF_SIZE];
    sprintf(buf,"%f;",value);
    int wc;
    if (strlen(buf) + 1 > (wc = write(fd,buf,strlen(buf) + 1)))
        if (wc < 0)
            perror("write()");
        else
            fprintf(stderr,"write to file error\n");
}

unsigned char matrix_production()
{
    double *A, *B, *C;
    A = (double *)malloc(N * N * sizeof(double));
    B = (double *)malloc(N * N * sizeof(double));
    C = (double *)calloc(N * N, sizeof(double));

    if (!A || !B || !C)
    {
        perror("mastrix production failed\n");
        return 1;
    }

    for (int i = 0; i < N; ++i)
    {
        for (int j = 0; j < N; ++j)
        {
            A[i * N + j] = i + j;
            B[i * N + j] = i - j;
        }
    }

    for (int i = 0; i < N; ++i)
    {
        double *c = C + i * N;
```

```
        for (int k = 0; k < N; ++k)
        {
            double *b = B + k * N;
            double a = A[i * N + k];
            for (int j = 0; j < N; ++j)
                c[j] += a * b[j];
        }
    }

    printf("proc heat: %f %f\n", C[N * (N / 2)], C[(N / 2) * (N / 2)]);

    free(A);
    free(B);
    free(C);
}

unsigned char heat_proc()
{
    return matrix_production();
}

unsigned long long get_ticks()
{
    unsigned long th, tl;
    __asm__ volatile("cpuid");
    __asm__ volatile("rdtsc\n"
                 : "=a"(tl), "=d"(th));
    unsigned long long ticks = (th << 32) | (4294967295UL & tl);
    printf("ticks: %lld\n", ticks);
    return ticks;
}

void measure_latency(int rtfd)
{
    double *result = (double *)_mm_malloc(sizeof(__m128d),
sizeof(__m128d));
    __m128d vect = _mm_set_pd(100000.0,400000.0);

    volatile unsigned long long startt = get_ticks();
    for (int i = 0; i < NUM_OF_ITERATIONS; ++i)
    {
        vect = _mm_sqrt_pd(vect);
        vect = _mm_sqrt_pd(vect);
        vect = _mm_sqrt_pd(vect);
        vect = _mm_sqrt_pd(vect);
        vect = _mm_sqrt_pd(vect);
        vect = _mm_sqrt_pd(vect);
        vect = _mm_sqrt_pd(vect);
        vect = _mm_sqrt_pd(vect);
        vect = _mm_sqrt_pd(vect);
```

```
        vect = _mm_sqrt_pd(vect);
        vect = _mm_sqrt_pd(vect);
        vect = _mm_sqrt_pd(vect);
        vect = _mm_sqrt_pd(vect);
        vect = _mm_sqrt_pd(vect);
        vect = _mm_sqrt_pd(vect);
        vect = _mm_sqrt_pd(vect);
    }
    volatile unsigned long long endt = get_ticks();

    _mm_store_pd(result, vect);
    printf("measure latency: %f\n", result[0]);
    double latency;
    printf("LATENCY: %f\n", latency = (double)(endt - startt) / (16 *
NUM_OF_ITERATIONS));
    write_to_file(latency,rtfd);
    free(result);
}

void measure_reciprocal_throughput(int rtfd)
{
    int number_of_ind_vects = 16;
    double *result = (double *)_mm_malloc(sizeof(__m128d),
sizeof(__m128d));
    __m128d *vects = (__m128d *)_mm_malloc(number_of_ind_vects *
sizeof(__m128d),sizeof(__m128d));

    for (int i = 0; i < number_of_ind_vects; ++i)
        vects[i] = _mm_set_pd(10 * i, 55.55 * i);

    volatile unsigned long long startt = get_ticks();
    for (int i = 0; i < NUM_OF_ITERATIONS; ++i)
    {
        vects[0] = _mm_sqrt_pd(vects[0]);
        vects[1] = _mm_sqrt_pd(vects[1]);
        vects[2] = _mm_sqrt_pd(vects[2]);
        vects[3] = _mm_sqrt_pd(vects[3]);
        vects[4] = _mm_sqrt_pd(vects[4]);
        vects[5] = _mm_sqrt_pd(vects[5]);
        vects[6] = _mm_sqrt_pd(vects[6]);
        vects[7] = _mm_sqrt_pd(vects[7]);
        vects[8] = _mm_sqrt_pd(vects[8]);
        vects[9] = _mm_sqrt_pd(vects[9]);
        vects[10] = _mm_sqrt_pd(vects[10]);
        vects[11] = _mm_sqrt_pd(vects[11]);
        vects[12] = _mm_sqrt_pd(vects[12]);
        vects[13] = _mm_sqrt_pd(vects[13]);
        vects[14] = _mm_sqrt_pd(vects[14]);
        vects[15] = _mm_sqrt_pd(vects[15]);
    }
    volatile unsigned long long endt = get_ticks();
```

```
    _mm_store_pd(result, vects[number_of_ind_vects / 2 +
number_of_ind_vects / 4]);
    printf("measure reciprocal throughput: %f\n", result[0]);
    double throughput;
    printf("RECIPROCAL THROUGHPUT: %f\n",throughput = (double)(endt -
startt) / (number_of_ind_vects * NUM_OF_ITERATIONS));
    write_to_file(throughput,rtfd);
    free(result);
    free(vects);
}

int main(int argc, char * argv[])
{
    if (2 != argc)
    {
        fprintf(stderr,"wrong number of arguments, expected file name,
where to put measurements results\n");
        return 1;
    }
    int results_table_fd = open(argv[1],O_RDWR | O_CREAT | O_APPEND,
0700);
    if (-1 == results_table_fd)
    {
        perror("open()");
        return 1;
    }

    if (heat_proc())
        return 1;
    measure_latency(results_table_fd);
    measure_reciprocal_throughput(results_table_fd);
    close(results_table_fd);
    return 0;
}
```

Приложение №2. Ассемблерный листинг

```
      .file "main_sse.c"
      .text
      .section    .rodata.str1.1,"aMS",@progbits,1
.LC0:
      .string     "%f;"
.LC1:
      .string     "write to file error\n"
      .text
      .p2align 4
      .globl      write_to_file
      .type write_to_file, @function
write_to_file:
.LFB557:
      .cfi_startproc
      endbr64
      pushq %r12
      .cfi_def_cfa_offset 16
```

```
	.cfi_offset 12, -16
	movl  $256, %edx
	movl  $1, %esi
	leaq  .LC0(%rip), %rcx
	pushq %rbp
	.cfi_def_cfa_offset 24
	.cfi_offset 6, -24
	movl  %edi, %ebp
	pushq %rbx
	.cfi_def_cfa_offset 32
	.cfi_offset 3, -32
	subq  $272, %rsp
	.cfi_def_cfa_offset 304
	movq  %fs:40, %rax
	movq  %rax, 264(%rsp)
	xorl  %eax, %eax
	movq  %rsp, %r12
	movl  $1, %eax
	movq  %r12, %rdi
	call  __sprintf_chk@PLT
	movq  %r12, %rdi
	call  strlen@PLT
	movq  %r12, %rsi
	movl  %ebp, %edi
	leaq  1(%rax), %rbx
	movq  %rbx, %rdx
	call  write@PLT
	cltq
	cmpq  %rax, %rbx
	ja    .L6
.L1:
	movq  264(%rsp), %rax
	subq  %fs:40, %rax
	jne   .L7
	addq  $272, %rsp
	.cfi_remember_state
	.cfi_def_cfa_offset 32
	popq  %rbx
	.cfi_def_cfa_offset 24
	popq  %rbp
	.cfi_def_cfa_offset 16
	popq  %r12
	.cfi_def_cfa_offset 8
	ret
	.p2align 4,,10
	.p2align 3
.L6:
	.cfi_restore_state
	movq  stderr(%rip), %rdi
	leaq  .LC1(%rip), %rdx
	movl  $1, %esi
	xorl  %eax, %eax
	call  __fprintf_chk@PLT
	jmp   .L1
.L7:
	call  __stack_chk_fail@PLT
	.cfi_endproc
.LFE557:
	.size write_to_file, .-write_to_file
	.section    .rodata.str1.1
.LC3:
	.string     "mastrix production failed\n"
```

```
.LC5:
      .string     "proc heat: %f %f\n"
      .text
      .p2align 4
      .globl      matrix_production
      .type matrix_production, @function
matrix_production:
.LFB558:
      .cfi_startproc
      endbr64
      pushq %r13
      .cfi_def_cfa_offset 16
      .cfi_offset 13, -16
      movl  $32000000, %edi
      pushq %r12
      .cfi_def_cfa_offset 24
      .cfi_offset 12, -24
      pushq %rbp
      .cfi_def_cfa_offset 32
      .cfi_offset 6, -32
      call  malloc@PLT
      movl  $32000000, %edi
      movq  %rax, %r13
      call  malloc@PLT
      movl  $8, %esi
      movl  $4000000, %edi
      movq  %rax, %r12
      call  calloc@PLT
      testq %r13, %r13
      movq  %rax, %rbp
      sete  %al
      testq %r12, %r12
      sete  %dl
      orb   %dl, %al
      jne   .L9
      testq %rbp, %rbp
      je    .L9
      movdqa      .LC4(%rip), %xmm5
      movq  %r13, %r9
      movq  %r12, %rcx
      xorl  %esi, %esi
      movdqa      .LC2(%rip), %xmm6
      movq  %r13, %rdx
.L10:
      movd  %esi, %xmm4
      xorl  %eax, %eax
      movdqa      %xmm6, %xmm1
      pshufd      $0, %xmm4, %xmm3
.L11:
      movdqa      %xmm1, %xmm2
      paddd %xmm5, %xmm1
      movdqa      %xmm2, %xmm0
      paddd %xmm3, %xmm0
      cvtdq2pd    %xmm0, %xmm4
      pshufd      $238, %xmm0, %xmm0
      movups      %xmm4, (%rdx,%rax)
      cvtdq2pd    %xmm0, %xmm0
      movups      %xmm0, 16(%rdx,%rax)
      movdqa      %xmm3, %xmm0
      psubd %xmm2, %xmm0
      cvtdq2pd    %xmm0, %xmm2
      pshufd      $238, %xmm0, %xmm0
```

```
      movups      %xmm2, (%rcx,%rax)
      cvtdq2pd    %xmm0, %xmm0
      movups      %xmm0, 16(%rcx,%rax)
      addq  $32, %rax
      cmpq  $16000, %rax
      jne   .L11
      addl  $1, %esi
      addq  $16000, %rdx
      addq  $16000, %rcx
      cmpl  $2000, %esi
      jne   .L10
      movq  %rbp, %r8
      xorl  %r10d, %r10d
      leaq  16000(%r12), %r11
.L17:
      movq  %r12, %rdx
      movl  $2, %ecx
.L14:
      movsd -16(%r9,%rcx,8), %xmm4
      movsd -8(%r9,%rcx,8), %xmm3
      leaq  16000(%rdx), %rsi
      xorl  %eax, %eax
      unpcklpd    %xmm4, %xmm4
      unpcklpd    %xmm3, %xmm3
.L13:
      movupd      16(%rdx,%rax), %xmm0
      movupd      16(%rsi,%rax), %xmm1
      movupd      16(%r8,%rax), %xmm6
      movupd      (%rsi,%rax), %xmm2
      mulpd %xmm4, %xmm0
      movupd      (%r8,%rax), %xmm5
      mulpd %xmm3, %xmm1
      mulpd %xmm3, %xmm2
      addpd %xmm6, %xmm0
      addpd %xmm1, %xmm0
      movupd      (%rdx,%rax), %xmm1
      mulpd %xmm4, %xmm1
      movups      %xmm0, 16(%r8,%rax)
      addpd %xmm5, %xmm1
      addpd %xmm2, %xmm1
      movups      %xmm1, (%r8,%rax)
      addq  $32, %rax
      cmpq  $16000, %rax
      jne   .L13
      movslq      %ecx, %rax
      addq  $2, %rcx
      addq  $32000, %rdx
      cmpq  $2000, %rcx
      jne   .L14
      leaq  1(%rax), %rdi
      leaq  15984(%r9), %rsi
      imulq $16000, %rax, %rcx
      leaq  0(%rbp,%r10,8), %rdx
      imulq $16000, %rdi, %rdi
      addq  %r12, %rcx
      addq  %r11, %rdi
.L16:
      movsd (%rsi), %xmm1
      xorl  %eax, %eax
      unpcklpd    %xmm1, %xmm1
      .p2align 4,,10
      .p2align 3
```

```
.L15:
      movupd      (%rcx,%rax), %xmm0
      movupd      (%rdx,%rax), %xmm7
      mulpd %xmm1, %xmm0
      addpd %xmm7, %xmm0
      movups      %xmm0, (%rdx,%rax)
      addq  $16, %rax
      cmpq  $16000, %rax
      jne   .L15
      addq  $16000, %rcx
      addq  $8, %rsi
      cmpq  %rdi, %rcx
      jne   .L16
      addq  $2000, %r10
      addq  $16000, %r8
      addq  $16000, %r9
      cmpq  $4000000, %r10
      jne   .L17
      movsd 16000000(%rbp), %xmm0
      movl  $1, %edi
      movl  $2, %eax
      movsd 8000000(%rbp), %xmm1
      leaq  .LC5(%rip), %rsi
      call  __printf_chk@PLT
      movq  %r13, %rdi
      call  free@PLT
      movq  %r12, %rdi
      call  free@PLT
      movq  %rbp, %rdi
      call  free@PLT
.L8:
      popq  %rbp
      .cfi_remember_state
      .cfi_def_cfa_offset 24
      popq  %r12
      .cfi_def_cfa_offset 16
      popq  %r13
      .cfi_def_cfa_offset 8
      ret
.L9:
      .cfi_restore_state
      leaq  .LC3(%rip), %rdi
      call  perror@PLT
      movl  $1, %eax
      jmp   .L8
      .cfi_endproc
.LFE558:
      .size matrix_production, .-matrix_production
      .p2align 4
      .globl      heat_proc
      .type heat_proc, @function
heat_proc:
.LFB559:
      .cfi_startproc
      endbr64
      xorl  %eax, %eax
      jmp   matrix_production
      .cfi_endproc
.LFE559:
      .size heat_proc, .-heat_proc
      .section    .rodata.str1.1
.LC6:
```

```
	.string	"ticks: %lld\n"
	.text
	.p2align 4
	.globl	get_ticks
	.type get_ticks, @function
get_ticks:
.LFB560:
	.cfi_startproc
	endbr64
	pushq %r12
	.cfi_def_cfa_offset 16
	.cfi_offset 12, -16
#APP
# 74 "main_sse.c" 1
	cpuid
# 0 "" 2
# 75 "main_sse.c" 1
	rdtsc

# 0 "" 2
#NO_APP
	movl  %eax, %eax
	salq  $32, %rdx
	movl  $1, %edi
	leaq  .LC6(%rip), %rsi
	orq   %rax, %rdx
	xorl  %eax, %eax
	movq  %rdx, %r12
	call  __printf_chk@PLT
	movq  %r12, %rax
	popq  %r12
	.cfi_def_cfa_offset 8
	ret
	.cfi_endproc
.LFE560:
	.size get_ticks, .-get_ticks
	.section	.rodata.str1.1
.LC8:
	.string	"measure latency: %f\n"
.LC10:
	.string	"LATENCY: %f\n"
	.text
	.p2align 4
	.globl	measure_latency
	.type measure_latency, @function
measure_latency:
.LFB561:
	.cfi_startproc
	endbr64
	pushq %r13
	.cfi_def_cfa_offset 16
	.cfi_offset 13, -16
	movl  $16, %edx
	movl  $16, %esi
	pushq %r12
	.cfi_def_cfa_offset 24
	.cfi_offset 12, -24
	movl  %edi, %r12d
	pushq %rbp
	.cfi_def_cfa_offset 32
	.cfi_offset 6, -32
	xorl  %ebp, %ebp
```

```
      pushq %rbx
      .cfi_def_cfa_offset 40
      .cfi_offset 3, -40
      subq  $56, %rsp
      .cfi_def_cfa_offset 96
      movq  %fs:40, %rax
      movq  %rax, 40(%rsp)
      xorl  %eax, %eax
      leaq  32(%rsp), %rdi
      call  posix_memalign@PLT
      testl %eax, %eax
      cmove 32(%rsp), %rbp
#APP
# 74 "main_sse.c" 1
      cpuid
# 0 "" 2
# 75 "main_sse.c" 1
      rdtsc

# 0 "" 2
#NO_APP
      salq  $32, %rdx
      movl  %eax, %eax
      movl  $1, %edi
      movq  %rdx, %rbx
      leaq  .LC6(%rip), %r13
      orq   %rax, %rbx
      movq  %r13, %rsi
      xorl  %eax, %eax
      movq  %rbx, %rdx
      movq  %rbx, 24(%rsp)
      movl  $1000, %eax
      movapd      .LC7(%rip), %xmm0
      .p2align 4,,10
      .p2align 3
.L30:
      sqrtpd      %xmm0, %xmm0
      sqrtpd      %xmm0, %xmm0
      sqrtpd      %xmm0, %xmm0
      sqrtpd      %xmm0, %xmm0
      sqrtpd      %xmm0, %xmm0
      sqrtpd      %xmm0, %xmm0
      sqrtpd      %xmm0, %xmm0
      sqrtpd      %xmm0, %xmm0
      sqrtpd      %xmm0, %xmm0
      sqrtpd      %xmm0, %xmm0
      sqrtpd      %xmm0, %xmm0
      sqrtpd      %xmm0, %xmm0
      sqrtpd      %xmm0, %xmm0
      sqrtpd      %xmm0, %xmm0
      sqrtpd      %xmm0, %xmm0
      sqrtpd      %xmm0, %xmm0
      subl  $1, %eax
      jne   .L30
      movaps      %xmm0, (%rsp)
#APP
# 74 "main_sse.c" 1
      cpuid
# 0 "" 2
# 75 "main_sse.c" 1
      rdtsc
```

```
# 0 "" 2
#NO_APP
      salq  $32, %rdx
      movl  %eax, %eax
      movq  %r13, %rsi
      movl  $1, %edi
      movq  %rdx, %rbx
      orq   %rax, %rbx
      xorl  %eax, %eax
      movq  %rbx, %rdx
      call  __printf_chk@PLT
      movapd      (%rsp), %xmm0
      movq  %rbx, 32(%rsp)
      leaq  .LC8(%rip), %rsi
      movl  $1, %edi
      movl  $1, %eax
      movaps      %xmm0, 0(%rbp)
      call  __printf_chk@PLT
      movq  32(%rsp), %rax
      movq  24(%rsp), %rdx
      subq  %rdx, %rax
      js    .L31
      pxor  %xmm0, %xmm0
      cvtsi2sdq   %rax, %xmm0
.L32:
      movl  $1, %edi
      leaq  .LC10(%rip), %rsi
      movl  $1, %eax
      divsd .LC9(%rip), %xmm0
      movsd %xmm0, (%rsp)
      call  __printf_chk@PLT
      movsd (%rsp), %xmm0
      movl  %r12d, %edi
      call  write_to_file
      movq  40(%rsp), %rax
      subq  %fs:40, %rax
      jne   .L38
      addq  $56, %rsp
      .cfi_remember_state
      .cfi_def_cfa_offset 40
      movq  %rbp, %rdi
      popq  %rbx
      .cfi_def_cfa_offset 32
      popq  %rbp
      .cfi_def_cfa_offset 24
      popq  %r12
      .cfi_def_cfa_offset 16
      popq  %r13
      .cfi_def_cfa_offset 8
      jmp   free@PLT
.L31:
      .cfi_restore_state
      movq  %rax, %rdx
      andl  $1, %eax
      pxor  %xmm0, %xmm0
      shrq  %rdx
      orq   %rax, %rdx
      cvtsi2sdq   %rdx, %xmm0
      addsd %xmm0, %xmm0
      jmp   .L32
.L38:
      call  __stack_chk_fail@PLT
```

```
	.cfi_endproc
.LFE561:
	.size measure_latency, .-measure_latency
	.section	.rodata.str1.8,"aMS",@progbits,1
	.align 8
.LC26:
	.string	"measure reciprocal throughput: %f\n"
	.section	.rodata.str1.1
.LC27:
	.string	"RECIPROCAL THROUGHPUT: %f\n"
	.text
	.p2align 4
	.globl	measure_reciprocal_throughput
	.type measure_reciprocal_throughput, @function
measure_reciprocal_throughput:
.LFB562:
	.cfi_startproc
	endbr64
	pushq %r14
	.cfi_def_cfa_offset 16
	.cfi_offset 14, -16
	movl  $16, %edx
	movl  $16, %esi
	pushq %r13
	.cfi_def_cfa_offset 24
	.cfi_offset 13, -24
	pushq %r12
	.cfi_def_cfa_offset 32
	.cfi_offset 12, -32
	movl  %edi, %r12d
	pushq %rbp
	.cfi_def_cfa_offset 40
	.cfi_offset 6, -40
	xorl  %ebp, %ebp
	pushq %rbx
	.cfi_def_cfa_offset 48
	.cfi_offset 3, -48
	movq  %rbp, %r13
	subq  $48, %rsp
	.cfi_def_cfa_offset 96
	movq  %fs:40, %rax
	movq  %rax, 40(%rsp)
	xorl  %eax, %eax
	leaq  32(%rsp), %r14
	movq  %r14, %rdi
	call  posix_memalign@PLT
	movl  $256, %edx
	movl  $16, %esi
	movq  %r14, %rdi
	testl %eax, %eax
	cmove 32(%rsp), %r13
	call  posix_memalign@PLT
	pxor  %xmm0, %xmm0
	testl %eax, %eax
	cmove 32(%rsp), %rbp
	movaps	%xmm0, 0(%rbp)
	movapd	.LC11(%rip), %xmm0
	movaps	%xmm0, 16(%rbp)
	movapd	.LC12(%rip), %xmm0
	movaps	%xmm0, 32(%rbp)
	movapd	.LC13(%rip), %xmm0
	movaps	%xmm0, 48(%rbp)
```

```
	movapd	.LC14(%rip), %xmm0
	movaps	%xmm0, 64(%rbp)
	movapd	.LC15(%rip), %xmm0
	movaps	%xmm0, 80(%rbp)
	movapd	.LC16(%rip), %xmm0
	movaps	%xmm0, 96(%rbp)
	movapd	.LC17(%rip), %xmm0
	movaps	%xmm0, 112(%rbp)
	movapd	.LC18(%rip), %xmm0
	movaps	%xmm0, 128(%rbp)
	movapd	.LC19(%rip), %xmm0
	movaps	%xmm0, 144(%rbp)
	movapd	.LC20(%rip), %xmm0
	movaps	%xmm0, 160(%rbp)
	movapd	.LC21(%rip), %xmm0
	movaps	%xmm0, 176(%rbp)
	movapd	.LC22(%rip), %xmm0
	movaps	%xmm0, 192(%rbp)
	movapd	.LC23(%rip), %xmm0
	movaps	%xmm0, 208(%rbp)
	movapd	.LC24(%rip), %xmm0
	movaps	%xmm0, 224(%rbp)
	movapd	.LC25(%rip), %xmm0
	movaps	%xmm0, 240(%rbp)
	movapd	0(%rbp), %xmm15
	movapd	16(%rbp), %xmm14
	movapd	32(%rbp), %xmm13
	movapd	48(%rbp), %xmm12
	movapd	64(%rbp), %xmm11
	movapd	80(%rbp), %xmm10
	movapd	96(%rbp), %xmm9
	movapd	112(%rbp), %xmm8
	movapd	128(%rbp), %xmm7
	movapd	144(%rbp), %xmm6
	movapd	160(%rbp), %xmm5
	movapd	176(%rbp), %xmm4
	movapd	192(%rbp), %xmm3
	movapd	208(%rbp), %xmm2
	movapd	224(%rbp), %xmm1
	movapd	240(%rbp), %xmm0
	.p2align 4,,10
	.p2align 3
#APP
# 74 "main_sse.c" 1
	cpuid
# 0 "" 2
# 75 "main_sse.c" 1
	rdtsc

# 0 "" 2
#NO_APP
	salq	$32, %rdx
	movl	%eax, %eax
	movl	$1, %edi
	movq	%rdx, %rbx
	leaq	.LC6(%rip), %r14
	orq	%rax, %rbx
	movq	%r14, %rsi
	xorl	%eax, %eax
	movq	%rbx, %rdx
	movq	%rbx, 24(%rsp)
	movl	$1000, %eax
```

```
.L42:
      sqrtpd      %xmm15, %xmm15
      sqrtpd      %xmm14, %xmm14
      sqrtpd      %xmm13, %xmm13
      sqrtpd      %xmm12, %xmm12
      sqrtpd      %xmm11, %xmm11
      sqrtpd      %xmm10, %xmm10
      sqrtpd      %xmm9, %xmm9
      sqrtpd      %xmm8, %xmm8
      sqrtpd      %xmm7, %xmm7
      sqrtpd      %xmm6, %xmm6
      sqrtpd      %xmm5, %xmm5
      sqrtpd      %xmm4, %xmm4
      sqrtpd      %xmm3, %xmm3
      sqrtpd      %xmm2, %xmm2
      sqrtpd      %xmm1, %xmm1
      sqrtpd      %xmm0, %xmm0
      subl  $1, %eax
      jne   .L42
#APP
# 74 "main_sse.c" 1
      cpuid
# 0 "" 2
# 75 "main_sse.c" 1
      rdtsc

# 0 "" 2
#NO_APP
      movaps      %xmm15, 0(%rbp)
      movaps      %xmm14, 16(%rbp)
      movaps      %xmm13, 32(%rbp)
      movaps      %xmm12, 48(%rbp)
      movaps      %xmm11, 64(%rbp)
      movaps      %xmm10, 80(%rbp)
      movaps      %xmm9, 96(%rbp)
      movaps      %xmm8, 112(%rbp)
      movaps      %xmm7, 128(%rbp)
      movaps      %xmm6, 144(%rbp)
      movaps      %xmm5, 160(%rbp)
      movaps      %xmm4, 176(%rbp)
      movaps      %xmm3, 192(%rbp)
      movaps      %xmm2, 208(%rbp)
      movaps      %xmm1, 224(%rbp)
      movaps      %xmm0, 240(%rbp)
      salq  $32, %rdx
      movl  %eax, %eax
      movq  %r14, %rsi
      movl  $1, %edi
      movq  %rdx, %rbx
      orq   %rax, %rbx
      xorl  %eax, %eax
      movq  %rbx, %rdx
      call  __printf_chk@PLT
      movq  %rbx, 32(%rsp)
      movl  $1, %edi
      movapd      192(%rbp), %xmm0
      leaq  .LC26(%rip), %rsi
      movl  $1, %eax
      movaps      %xmm0, 0(%r13)
      call  __printf_chk@PLT
      movq  32(%rsp), %rax
      movq  24(%rsp), %rdx
```

```
      subq  %rdx, %rax
      js    .L43
      pxor  %xmm0, %xmm0
      cvtsi2sdq   %rax, %xmm0
.L44:
      leaq  .LC27(%rip), %rsi
      movl  $1, %edi
      movl  $1, %eax
      divsd .LC9(%rip), %xmm0
      movsd %xmm0, 8(%rsp)
      call  __printf_chk@PLT
      movsd 8(%rsp), %xmm0
      movl  %r12d, %edi
      call  write_to_file
      movq  %r13, %rdi
      call  free@PLT
      movq  40(%rsp), %rax
      subq  %fs:40, %rax
      jne   .L51
      addq  $48, %rsp
      .cfi_remember_state
      .cfi_def_cfa_offset 48
      movq  %rbp, %rdi
      popq  %rbx
      .cfi_def_cfa_offset 40
      popq  %rbp
      .cfi_def_cfa_offset 32
      popq  %r12
      .cfi_def_cfa_offset 24
      popq  %r13
      .cfi_def_cfa_offset 16
      popq  %r14
      .cfi_def_cfa_offset 8
      jmp   free@PLT
.L43:
      .cfi_restore_state
      movq  %rax, %rdx
      andl  $1, %eax
      pxor  %xmm0, %xmm0
      shrq  %rdx
      orq   %rax, %rdx
      cvtsi2sdq   %rdx, %xmm0
      addsd %xmm0, %xmm0
      jmp   .L44
.L51:
      call  __stack_chk_fail@PLT
      .cfi_endproc
.LFE562:
      .size measure_reciprocal_throughput, .-measure_reciprocal_throughput
      .section    .rodata.str1.8
      .align 8
.LC28:
      .string     "wrong number of arguments, expected file name, where to put
measurements results\n"
      .section    .rodata.str1.1
.LC29:
      .string     "open()"
      .section    .text.startup,"ax",@progbits
      .p2align 4
      .globl      main
      .type main, @function
main:
```

```
.LFB563:
      .cfi_startproc
      endbr64
      pushq %rbp
      .cfi_def_cfa_offset 16
      .cfi_offset 6, -16
      cmpl  $2, %edi
      je    .L53
      movq  stderr(%rip), %rdi
      leaq  .LC28(%rip), %rdx
      movl  $1, %esi
      xorl  %eax, %eax
      call  __fprintf_chk@PLT
      movl  $1, %eax
.L52:
      popq  %rbp
      .cfi_remember_state
      .cfi_def_cfa_offset 8
      ret
.L53:
      .cfi_restore_state
      movq  8(%rsi), %rdi
      movl  $448, %edx
      movl  $1090, %esi
      xorl  %eax, %eax
      call  open@PLT
      movl  %eax, %ebp
      cmpl  $-1, %eax
      je    .L58
      xorl  %eax, %eax
      call  matrix_production
      movl  %eax, %r8d
      movl  $1, %eax
      testb %r8b, %r8b
      jne   .L52
      movl  %ebp, %edi
      call  measure_latency
      movl  %ebp, %edi
      call  measure_reciprocal_throughput
      movl  %ebp, %edi
      call  close@PLT
      xorl  %eax, %eax
      popq  %rbp
      .cfi_remember_state
      .cfi_def_cfa_offset 8
      ret
.L58:
      .cfi_restore_state
      leaq  .LC29(%rip), %rdi
      call  perror@PLT
      movl  $1, %eax
      popq  %rbp
      .cfi_def_cfa_offset 8
      ret
      .cfi_endproc
.LFE563:
      .size main, .-main
      .section    .rodata.cst16,"aM",@progbits,16
      .align 16
.LC2:
      .long 0
      .long 1
```

```
	.long 2
	.long 3
	.align 16
.LC4:
	.long 4
	.long 4
	.long 4
	.long 4
	.align 16
.LC7:
	.long 0
	.long 1092119040
	.long 0
	.long 1090021888
	.section	.rodata.cst8,"aM",@progbits,8
	.align 8
.LC9:
	.long 0
	.long 1087324160
	.section	.rodata.cst16
	.align 16
.LC11:
	.long 1717986918
	.long 1078707814
	.long 0
	.long 1076101120
	.align 16
.LC12:
	.long 1717986918
	.long 1079756390
	.long 0
	.long 1077149696
	.align 16
.LC13:
	.long -858993460
	.long 1080349900
	.long 0
	.long 1077805056
	.align 16
.LC14:
	.long 1717986918
	.long 1080804966
	.long 0
	.long 1078198272
	.align 16
.LC15:
	.long 0
	.long 1081170944
	.long 0
	.long 1078525952
	.align 16
.LC16:
	.long -858993460
	.long 1081398476
	.long 0
	.long 1078853632
	.align 16
.LC17:
	.long -1717986919
	.long 1081626009
	.long 0
	.long 1079083008
```

```
	.align 16
.LC18:
	.long 1717986918
	.long 1081853542
	.long 0
	.long 1079246848
	.align 16
.LC19:
	.long 858993459
	.long 1082081075
	.long 0
	.long 1079410688
	.align 16
.LC20:
	.long 0
	.long 1082219520
	.long 0
	.long 1079574528
	.align 16
.LC21:
	.long 1717986918
	.long 1082333286
	.long 0
	.long 1079738368
	.align 16
.LC22:
	.long -858993460
	.long 1082447052
	.long 0
	.long 1079902208
	.align 16
.LC23:
	.long 858993459
	.long 1082560819
	.long 0
	.long 1080049664
	.align 16
.LC24:
	.long -1717986919
	.long 1082674585
	.long 0
	.long 1080131584
	.align 16
.LC25:
	.long 0
	.long 1082788352
	.long 0
	.long 1080213504
	.ident	"GCC: (Ubuntu 11.3.0-1ubuntu1~22.04) 11.3.0"
	.section	.note.GNU-stack,"",@progbits
	.section	.note.gnu.property,"a"
	.align 8
	.long 1f - 0f
	.long 4f - 1f
	.long 5
0:
	.string	"GNU"
1:
	.align 8
	.long 0xc0000002
	.long 3f - 2f
2:
```

```
      .long 0x3
3:
      .align 8
4:
```

Приложение №3. Bash-скрипт для запуска на ПК

```
#!/bin/bash

result_file_name=table.csv

>$result_file_name

for ((j = 0; j < 10; j++))
do
      ./main $result_file_name
      printf "\n">>$result_file_name
      printf "\n"
done
```

Приложение №4. Bash-скрипт для запуска на суперкомпьютере НГУ

```
#!/bin/bash

#PBS -l select=1:ncpus=1:mem=100m
#PBS -l walltime=00:01:00
#PBS -q xl230g9q
#PBS -m n

cd $PBS_O_WORKDIR

result_file_name=clu_table.csv

cat /proc/cpuinfo

gcc main_sse.s -o main_sse

>$result_file_name

for ((j = 0; j < 10; j++))
do
      ./main_sse $result_file_name
      printf "\n">>$result_file_name
      printf "\n"
done
```