Лабораторная работа 3 (0011 = 3)

Использование ассемблерных вставок в программах на С++.

Команды пересылки

Цель работы: научиться вставлять в программы на языке высокого уровня ассемблерные фрагменты. Ознакомиться с командами пересылки данных.

Все задания Л3 — в виде ассемблерных вставок в программу на C/C++. Как и для всего курса — если иное не указано явно, компилятор должен быть из коллекции GCC (среда Microsoft Visual Studio недопустима), подробнее в разделе «Компилятор, IDE, отладчик» регламента, РЛ.2.

Задание Л3.№1

Создайте массивы:
— Ms из 16-битных целых чисел $short/unsigned\ short$;
— Ml из 32-битных целых чисел $int/unsigned\ int$;
— Mq из 64-битных целых чисел $long\ long/unsigned\ long\ long$;
длина N и начальные значения аналогичны Л1.№3.

Реализуйте для каждого массива M вставку, записывающую непосредственное значение 18 в $M[i]$ для заданного $i \in [0, N{-}2]$ с использованием команды mov, где выражение $M[i]$ является выходным параметром вставки в памяти. Так как оба операнда mov здесь не имеют определённого размера (непосредственное значение и память), необходимо указывать для mov суффикс размера: $movw$, $movl$, $movq$.

Каждый M напечатайте до и после изменения в шестнадцатеричном представлении, элементы $M[i]$ разделяются пробелами, разные M — переводом строки.

```
Задание №1
======================================================================
Ms before:
FADE 1111101011011110 64222 -1314
FADE 1111101011011110 64222 -1314
FADE 1111101011011110 64222 -1314
FADE 1111101011011110 64222 -1314
FADE 1111101011011110 64222 -1314
Ml before:
ADE1A1DA 10101101111000011010000111011010 2917245402 -1377721894 -0x1.c343b4p-36 -2.565142e-11 -0.00
ADE1A1DA 10101101111000011010000111011010 2917245402 -1377721894 -0x1.c343b4p-36 -2.565142e-11 -0.00
ADE1A1DA 10101101111000011010000111011010 2917245402 -1377721894 -0x1.c343b4p-36 -2.565142e-11 -0.00
ADE1A1DA 10101101111000011010000111011010 2917245402 -1377721894 -0x1.c343b4p-36 -2.565142e-11 -0.00
ADE1A1DA 10101101111000011010000111011010 2917245402 -1377721894 -0x1.c343b4p-36 -2.565142e-11 -0.00
Mq before:
51F1AB1E 1010001111100011010101100011110 1374792478 +1374792478 +0x1.e3563cp+36 +1.297447e+11 +129744748544.00
51F1AB1E 1010001111100011010101100011110 1374792478 +1374792478 +0x1.e3563cp+36 +1.297447e+11 +129744748544.00
51F1AB1E 1010001111100011010101100011110 1374792478 +1374792478 +0x1.e3563cp+36 +1.297447e+11 +129744748544.00
51F1AB1E 1010001111100011010101100011110 1374792478 +1374792478 +0x1.e3563cp+36 +1.297447e+11 +129744748544.00
51F1AB1E 1010001111100011010101100011110 1374792478 +1374792478 +0x1.e3563cp+36 +1.297447e+11 +129744748544.00
Ms after:
FADE 1111101011011110 64222 -1314
FADE 1111101011011110 64222 -1314
0012 00010010 18 +18
FADE 1111101011011110 64222 -1314
FADE 1111101011011110 64222 -1314
Ml after:
ADE1A1DA 10101101111000011010000111011010 2917245402 -1377721894 -0x1.c343b4p-36 -2.565142e-11 -0.00
ADE1A1DA 10101101111000011010000111011010 2917245402 -1377721894 -0x1.c343b4p-36 -2.565142e-11 -0.00
00000012 00010010 18 +18 +0x1.2p-145 +2.522337e-44 +0.00
ADE1A1DA 10101101111000011010000111011010 2917245402 -1377721894 -0x1.c343b4p-36 -2.565142e-11 -0.00
ADE1A1DA 10101101111000011010000111011010 2917245402 -1377721894 -0x1.c343b4p-36 -2.565142e-11 -0.00
Mq after:
51F1AB1E 1010001111100011010101100011110 1374792478 +1374792478 +0x1.e3563cp+36 +1.297447e+11 +129744748544.00
51F1AB1E 1010001111100011010101100011110 1374792478 +1374792478 +0x1.e3563cp+36 +1.297447e+11 +129744748544.00
00000012 00010010 18 +18 +0x1.2p-145 +2.522337e-44 +0.00
51F1AB1E 1010001111100011010101100011110 1374792478 +1374792478 +0x1.e3563cp+36 +1.297447e+11 +129744748544.00
51F1AB1E 1010001111100011010101100011110 1374792478 +1374792478 +0x1.e3563cp+36 +1.297447e+11 +129744748544.00

======================================================================
```

Рис. 1: Вывод результата вставки значения в массив

Листинг:

Файл task3_1.c:

```c
#ifndef task3_1_H
#define task3_1_H

#define N 5

// Макрос для вывода массива с параметризацией массива и делегата вывода
#define PRINT_ARRAY(arr, printer) \
do { \
    for (int i = 0; i < N; i++) { \
        printer(&arr[i]); \
    } \
} while(0)

#include <stdio.h>

void print16(void *p);
void print32(void *p);
void print64(void *p);

void run_task3_1()
{
    printf("\nЗадание №1\n");
    printf("================================================================================\n");

    // Инициализация массивов (значения аналогичны Л1.№3, например)
    unsigned short Ms[N] = {0xFADE, 0xFADE, 0xFADE, 0xFADE, 0xFADE};
    unsigned int Ml[N] = {0xADE1A1DA, 0xADE1A1DA, 0xADE1A1DA, 0xADE1A1DA, 0xADE1A1DA};
    unsigned long long Mq[N] = {0xC1A551F1AB1EULL, 0xC1A551F1AB1EULL, 0xC1A551F1AB1EULL,
0xC1A551F1AB1EULL, 0xC1A551F1AB1EULL};

    int i = 2; // индекс вставки, 0 <= i <= N-2

    // Вывод до вставки
    printf("Ms before:\n");
    PRINT_ARRAY(Ms, print16);

    printf("Ml before:\n");
    PRINT_ARRAY(Ml,print32);

    printf("Mq before:\n");
    PRINT_ARRAY(Mq,print32);

    // Вставка значения 18 в Ms[i] с помощью movw
    __asm__ volatile (
        "movw $18, %0"
        : "=m" (Ms[i])
    );

    // Вставка значения 18 в Ml[i] с помощью movl
    __asm__ volatile (
        "movl $18, %0"
        : "=m" (Ml[i])
    );

    // Вставка значения 18 в Mq[i] с помощью movq
    __asm__ volatile (
        "movq $18, %0"
        : "=m" (Mq[i])
    );

    // Вывод после вставки
    printf("Ms after:\n");
    PRINT_ARRAY(Ms, print16);

    printf("Ml after:\n");
    PRINT_ARRAY(Ml,print32);

    printf("Mq after:\n");
    PRINT_ARRAY(Mq,print32);

    printf("\n================================================================================\n");
}

#endif
```

Задание Л3.№2

Реализуйте для одного из массивов M (по варианту согласно таблице Л3.1) вставку, записывающую непосредственное (−1) в $M[i]$, где адрес начала массива M и индекс i передаются как входные параметры в РОН.

Используйте компоненты эффективного адреса ($Base$, $Index$, 2 $Scale$). Разрядность компонент $Base$ и $Index$ должна быть одинаковой, поэтому для переносимости вставки необходимо объявить переменную i не как int (4 байта как для 32-, так и для 64-битного режимов), а как $size\_t$ (размер равен размеру указателя).

Заданный M напечатайте до и после изменения аналогично Л3.№1.

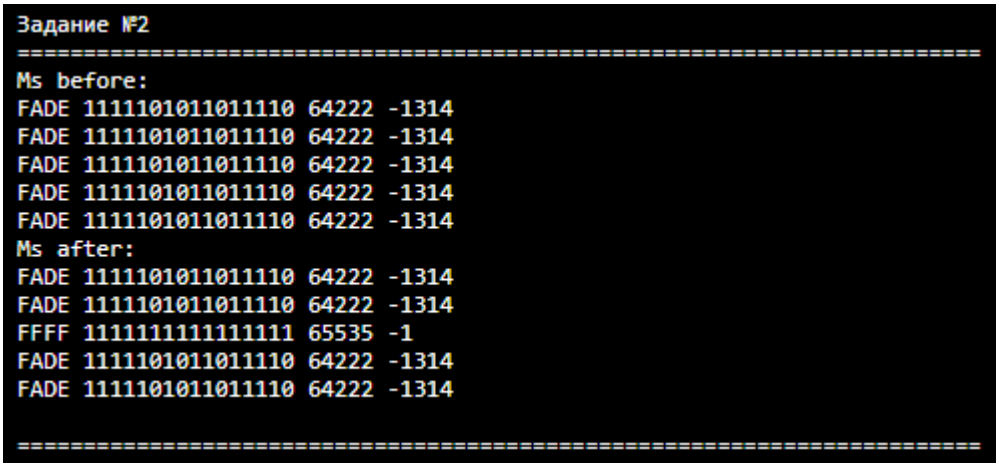

Рис. 2: результат выполнения вставки

Листинг

Файл task3_2.c:

```c
#ifndef task3_2_H
#define task3_2_H

#define N 5

// Макрос для вывода массива с параметризацией массива и делегата вывода
#define PRINT_ARRAY(arr, printer) \
do { \
    for (int i = 0; i < N; i++) { \
        printer(&arr[i]); \
    } \
} while(0)

#include <stdio.h>

void print16(void *p);
void print32(void *p);
void print64(void *p);

void run_task3_2()
{
    printf("\nЗадание №2\n");
    printf("================================================================================\n");

    // Инициализация массивов
    unsigned short Ms[N] = {0xFADE, 0xFADE, 0xFADE, 0xFADE, 0xFADE};

    size_t i = 2; // индекс вставки

    printf("Ms before:\n");
    PRINT_ARRAY(Ms, print16);

    // Ассемблерная вставка с использованием Base, Index, Scale=2 для Ms (16-бит)
    // Эффективный адрес: base = &Ms[0], index = i, scale = 2 (sizeof(short))
    // Запишем -1 (0xFFFF) в Ms[i]

    unsigned short val = 0xFFFF;

    __asm__ volatile (
        "movw %[val], (%[base], %[index], 2)"
        :
        : [val] "r" (val),
          [base] "r" (Ms),
          [index] "r" (i)
        : "memory"
    );

    printf("Ms after:\n");
    PRINT_ARRAY(Ms, print16);

    printf("\n================================================================================\n");
}

#endif
```

Задание Л3.№3

Реализуйте вставку, записывающую непосредственное значение 0x55 в заданный байт $Mq[i]$ (по варианту согласно таблице Л3.2; младший байт считайте нулевым) с использованием одной команды mov ($movb$) и всех компонент эффективного адреса $Disp(Base, Index, 2\ Scale)$; адрес начала массива Mq и индекс i передаются как входные параметры в РОН.

Mq напечатайте до и после изменения аналогично Л3.№1

```
Задание №3
=====================================================================
Mq before:
C1A551F1AB1E0000 10101011000111100000000000000000 13953649118877188096 -4493094954832363520 -0x1.551f1ab1ep+27 -1.788459e+08 -178845909.56
C1A551F1AB1E0000 10101011000111100000000000000000 13953649118877188096 -4493094954832363520 -0x1.551f1ab1ep+27 -1.788459e+08 -178845909.56
C1A551F1AB1E0000 10101011000111100000000000000000 13953649118877188096 -4493094954832363520 -0x1.551f1ab1ep+27 -1.788459e+08 -178845909.56
C1A551F1AB1E0000 10101011000111100000000000000000 13953649118877188096 -4493094954832363520 -0x1.551f1ab1ep+27 -1.788459e+08 -178845909.56
C1A551F1AB1E0000 10101011000111100000000000000000 13953649118877188096 -4493094954832363520 -0x1.551f1ab1ep+27 -1.788459e+08 -178845909.56
Mq after:
C1A551F1AB1E0000 10101011000111100000000000000000 13953649118877188096 -4493094954832363520 -0x1.551f1ab1ep+27 -1.788459e+08 -178845909.56
C1A551F1AB1E0000 10101011000111100000000000000000 13953649118877188096 -4493094954832363520 -0x1.551f1ab1ep+27 -1.788459e+08 -178845909.56
C1A551F1AB550000 10101011010101010000000000000000 13953649118880792576 -4493094954828759040 -0x1.551f1ab55p+27 -1.788459e+08 -178845909.67
C1A551F1AB1E0000 10101011000111100000000000000000 13953649118877188096 -4493094954832363520 -0x1.551f1ab1ep+27 -1.788459e+08 -178845909.56
C1A551F1AB1E0000 10101011000111100000000000000000 13953649118877188096 -4493094954832363520 -0x1.551f1ab1ep+27 -1.788459e+08 -178845909.56

=====================================================================
```

Рис. 3: результат выполнения вставки

Листинг:

Файл task3_3.c:

```c
void run_task3_3()
{
    printf("\nЗадание №3\n");
    printf("========================================================================\n");

   unsigned long long Mq[N] = {
        0xC1A551F1AB1E0000ULL, 0xC1A551F1AB1E0000ULL,
        0xC1A551F1AB1E0000ULL, 0xC1A551F1AB1E0000ULL,
        0xC1A551F1AB1E0000ULL
    };

    const int i = 2; // индекс элемента для изменения
    const int byte_offset = 2;  // №2 — третий байт

    printf("Mq before:\n");
    PRINT_ARRAY(Mq, print64);

   // Вычисляем адрес нужного байта
    unsigned char *addr = (unsigned char *)Mq + i * sizeof(unsigned long long) + byte_offset;

    // Используем movb для записи 0x55 в третий байт Mq[i]
    __asm__ volatile (
        "movb $0x55, (%[addr])"
        :
        : [addr] "r" (addr)
        : "memory"
    );

    printf("Mq after:\n");
    PRINT_ARRAY(Mq, print64);

    printf("\n========================================================================\n");
}
```

Задание Л3.№4.

Реализуйте вставку, записывающую в $M[i]$ (M по варианту согласно таблице Л3.1) значение переменной x; размер x равен размеру элемента M. Значение x передаётся как входной параметр в памяти, адрес M и индекс i — как входные параметры в РОН.

Так как команда x86/amd64 не может адресовать два операнда в памяти, прямая пересылка $x \to M[i]$ невозможна; используйте промежуточный регистр (по варианту согласно таблице Л3.3).

Вариант 3: Регистр D ($rdx/edx/dx/dl$)

M напечатайте до и после изменения аналогично Л3.№1

```
Задание №4
========================================================================
Ml before:
DEADBEEF 11011110101011011011111011101111 3735928559 -559038737 -0x1.5b7ddep+62 -6.259853e+18 -6259853398707798016.00
DEADBEEF 11011110101011011011111011101111 3735928559 -559038737 -0x1.5b7ddep+62 -6.259853e+18 -6259853398707798016.00
DEADBEEF 11011110101011011011111011101111 3735928559 -559038737 -0x1.5b7ddep+62 -6.259853e+18 -6259853398707798016.00
DEADBEEF 11011110101011011011111011101111 3735928559 -559038737 -0x1.5b7ddep+62 -6.259853e+18 -6259853398707798016.00
DEADBEEF 11011110101011011011111011101111 3735928559 -559038737 -0x1.5b7ddep+62 -6.259853e+18 -6259853398707798016.00
Ml after:
DEADBEEF 11011110101011011011111011101111 3735928559 -559038737 -0x1.5b7ddep+62 -6.259853e+18 -6259853398707798016.00
DEADBEEF 11011110101011011011111011101111 3735928559 -559038737 -0x1.5b7ddep+62 -6.259853e+18 -6259853398707798016.00
DEADBEEF 11011110101011011011111011101111 3735928559 -559038737 -0x1.5b7ddep+62 -6.259853e+18 -6259853398707798016.00
12345678 10010001101000101011001111000 305419896 +305419896 +0x1.68acfp-91 +5.690457e-28 +0.00
DEADBEEF 11011110101011011011111011101111 3735928559 -559038737 -0x1.5b7ddep+62 -6.259853e+18 -6259853398707798016.00

========================================================================
```

Рис. 4: результат выполнения вставки

Листинг:

Файл task3_4.c:

```c
void run_task3_4()
{
    printf("\nЗадание №4\n");
    printf("========================================================================\n");

   unsigned int Ml[N] = {0xDEADBEEF, 0xDEADBEEF, 0xDEADBEEF, 0xDEADBEEF, 0xDEADBEEF};

    size_t i = 3; // индекс для записи
    unsigned int x = 0x12345678; // значение для вставки

    printf("Ml before:\n");
    PRINT_ARRAY(Ml, print32);

    // Используем промежуточный регистр edx (регистр D для 32-бит)
    __asm__ volatile (
        "movl %[x], %%edx\n\t"            // загрузить x в edx
        "movl %%edx, (%[base], %[index], 4)" // записать edx в M[i], scale=4 (sizeof(unsigned int))
        :
        : [x] "r" (x),
          [base] "r" (Ml),
          [index] "r" (i)
        : "edx", "memory"
    );

    printf("Ml after:\n");
    PRINT_ARRAY(Ml, print32);

    printf("\n========================================================================\n");
}
```

Задание Л3.№5.

Реализуйте вставку, записывающую в $M[i]$ значение x аналогично Л3.№4, но во вставку передаётся адрес $\&x$.

```
Задание №5
======================================================================
M1 before:
DEADBEEF 11011110101011011011111011101111 3735928559 -559038737 -0x1.5b7ddep+62 -6.259853e+18 -6259853398707798016.00
DEADBEEF 11011110101011011011111011101111 3735928559 -559038737 -0x1.5b7ddep+62 -6.259853e+18 -6259853398707798016.00
DEADBEEF 11011110101011011011111011101111 3735928559 -559038737 -0x1.5b7ddep+62 -6.259853e+18 -6259853398707798016.00
DEADBEEF 11011110101011011011111011101111 3735928559 -559038737 -0x1.5b7ddep+62 -6.259853e+18 -6259853398707798016.00
DEADBEEF 11011110101011011011111011101111 3735928559 -559038737 -0x1.5b7ddep+62 -6.259853e+18 -6259853398707798016.00
M1 after:
DEADBEEF 11011110101011011011111011101111 3735928559 -559038737 -0x1.5b7ddep+62 -6.259853e+18 -6259853398707798016.00
DEADBEEF 11011110101011011011111011101111 3735928559 -559038737 -0x1.5b7ddep+62 -6.259853e+18 -6259853398707798016.00
DEADBEEF 11011110101011011011111011101111 3735928559 -559038737 -0x1.5b7ddep+62 -6.259853e+18 -6259853398707798016.00
12345678 10010001101000101011001111000 305419896 +305419896 +0x1.68acfp-91 +5.690457e-28 +0.00
DEADBEEF 11011110101011011011111011101111 3735928559 -559038737 -0x1.5b7ddep+62 -6.259853e+18 -6259853398707798016.00

======================================================================
```

Рис. 5: результат выполнения вставки

Листинг:

Файл task3_5.c:

```c
void run_task3_5()
{
    printf("\nЗадание №5\n");
    printf("========================================================================\n");

    unsigned int M1[N] = {0xDEADBEEF, 0xDEADBEEF, 0xDEADBEEF, 0xDEADBEEF, 0xDEADBEEF};

    size_t i = 3; // индекс для записи
    unsigned int x = 0x12345678; // значение для вставки

    printf("M1 before:\n");
    PRINT_ARRAY(M1, print32);

    __asm__ volatile (
        "movl (%[x_addr]), %%edx\n\t"         // загрузить значение по адресу x_addr в edx
        "movl %%edx, (%[base], %[index], 4)"  // записать edx в M[i]
        :
        : [x_addr] "r" (&x),
          [base] "r" (M1),
          [index] "r" (i)
        : "edx", "memory"
    );

    printf("M1 after:\n");
    PRINT_ARRAY(M1, print32);

    printf("\n========================================================================\n");
}

```

Задание Л3.№6.

Реализуйте вставку, рассчитывающую для целочисленных x и y значения $z = x + y$ и $w = x - y$ при помощи команд add и sub. Разрядность указана в таблице Л3.4; переменные x, y, z, w передаются во вставку как параметры (z и w — выходные, x и y — входные). Значения x, y, z, w напечатайте до и после вставки.

```
Задание №6
========================================================================
Before:
x = 9ABCDEF0 10011010101111001101111011110000 2596069104 -1698898192 -0x1.79bdep-74 -7.811515e-23 -0.00
y = 87654321 10000111011001010100001100100001 2271560481 -2023406815 -0x1.ca8642p-113 -1.724777e-34 -0.00
z = 00000000 00000000 0 +0 +0x0p+0 +0.000000e+00 +0.00
w = 00000000 00000000 0 +0 +0x0p+0 +0.000000e+00 +0.00
After:
x = 9ABCDEF0 10011010101111001101111011110000 2596069104 -1698898192 -0x1.79bdep-74 -7.811515e-23 -0.00
y = 87654321 10000111011001010100001100100001 2271560481 -2023406815 -0x1.ca8642p-113 -1.724777e-34 -0.00
z = 22222211 10001000100010001000100010001 572662289 +572662289 +0x1.444422p-59 +2.197313e-18 +0.00
w = 789ABCDF 11110001001101010111100111011111 2023406815 +2023406815 +0x1.3579bep+114 +2.510764e+34 +2510763990937783280564630869508096O.00

========================================================================
```

Рис. 6: результат выполнения вставки

Листинг:

Файл task3_6.c:

```c
void run_task3_6()
{
    printf("\nЗадание №6\n");
    printf("================================================================================\n");

    uint64_t x = 0x123456789ABCDEF0;
    uint64_t y = 0x0FEDCBA987654321;
    uint64_t z = 0;
    uint64_t w = 0;

    printf("Before:\n");
    printf("x = ");
    print32(&x);
    printf("y = ");
    print32(&y);
    printf("z = ");
    print32(&z);
    printf("w = ");
    print32(&w);

    __asm__ volatile (
        "mov %[x], %%rax\n\t"   // rax = x
        "add %[y], %%rax\n\t"   // rax = x + y
        "mov %%rax, %[z]\n\t"   // z = rax

        "mov %[x], %%rbx\n\t"   // rbx = x
        "sub %[y], %%rbx\n\t"   // rbx = x - y
        "mov %%rbx, %[w]\n\t"   // w = rbx
        : [z] "=r" (z), [w] "=r" (w)
        : [x] "r" (x), [y] "r" (y)
        : "rax", "rbx"
    );

    printf("After:\n");
    printf("x = ");
    print32(&x);
    printf("y = ");
    print32(&y);
    printf("z = ");
    print32(&z);
    printf("w = ");
    print32(&w);

    printf("\n================================================================================\n");
}
```

Задание Л3.№7.

Определите, доступны ли на выбранной платформе расширения AVX и SSE, используя команду $cpuid$ или документацию на процессор.

Как в задании Л1.№3, создайте массивы: — Mfl из 64-битных чисел с плавающей запятой $double$; — Mfs из 32-битных чисел с плавающей запятой $float$; длина N и начальные значения аналогичны Л1.№3. Реализуйте вставку, записывающую в $M[i]$ (M по варианту согласно таблице Л3.5) значение переменной x с плавающей запятой (размер x равен размеру элемента M), используя команды AVX $vmovsd/vmovss$ или их SSE-аналоги $movsd/movss$. Значение x передаётся как входной параметр в памяти, адрес M и индекс i — как входные параметры в РОН. Используйте промежуточный регистр xmm j, где номер регистра j ∈ [0, 5] рассчитывается по варианту как (№ – 1)%6.

```
Задание №7
======================================================================
Before:
4005555555555555 1010101010101010101010101010101 4613187218303178069 +4613187218303178069 +0x1.5555555555555p+1 +2.666667e+00 +2.67
4005555555555555 1010101010101010101010101010101 4613187218303178069 +4613187218303178069 +0x1.5555555555555p+1 +2.666667e+00 +2.67
4005555555555555 1010101010101010101010101010101 4613187218303178069 +4613187218303178069 +0x1.5555555555555p+1 +2.666667e+00 +2.67
4005555555555555 1010101010101010101010101010101 4613187218303178069 +4613187218303178069 +0x1.5555555555555p+1 +2.666667e+00 +2.67
4005555555555555 1010101010101010101010101010101 4613187218303178069 +4613187218303178069 +0x1.5555555555555p+1 +2.666667e+00 +2.67
After:
4005555555555555 1010101010101010101010101010101 4613187218303178069 +4613187218303178069 +0x1.5555555555555p+1 +2.666667e+00 +2.67
4005555555555555 1010101010101010101010101010101 4613187218303178069 +4613187218303178069 +0x1.5555555555555p+1 +2.666667e+00 +2.67
4023C0CA4588F633 1000101100010001111011000110011 4621749617594791475 +4621749617594791475 +0x1.3c0ca4588f633p+3 +9.876543e+00 +9.88
4005555555555555 1010101010101010101010101010101 4613187218303178069 +4613187218303178069 +0x1.5555555555555p+1 +2.666667e+00 +2.67
4005555555555555 1010101010101010101010101010101 4613187218303178069 +4613187218303178069 +0x1.5555555555555p+1 +2.666667e+00 +2.67

======================================================================
```

Рис. 7: результат выполнения вставки

Листинг:

Файл task3_7.c:

```c
void run_task3_7()
{
    printf("\nЗадание №7\n");
    printf("========================================================================\n");

    // Вариант 17: x = 8/3 для float/double
    double Mfl[N];

    size_t i = 2;
    double x = 9.87654321;

    for (size_t i = 0; i < N; i++) {
        Mfl[i] = 8.0 / 3.0;
    }

    printf("Before:\n");
    PRINT_ARRAY(Mfl, print64);

    __asm__ volatile (
        "vmovsd (%[x_addr]), %%xmm4\n\t"             // загрузить double x в xmm4
        "vmovsd %%xmm4, (%[base], %[index], 8)\n\t"  // записать xmm4 в M[i], scale=8 (double)
        :
        : [x_addr] "r" (&x),
          [base] "r" (Mfl),
          [index] "r" (i)
        : "xmm4", "memory"
    );

    printf("After:\n");
    PRINT_ARRAY(Mfl, print64);

    printf("\n========================================================================\n");
}
```

Задание Л3.№8.

Реализуйте вставку, записывающую в $M[i]$ (M по варианту согласно таблице Л3.5) значение с плавающей запятой, равное целочисленному значениюx. Преобразование целочисленного x к нужному виду выполните при помощи команд AVX $vcvtsi2sd/vcvtsi2ss$ или их SSE-аналогов $cvtsi2sd/cvtsi2ss$.

```
Задание №8
======================================================================
Before:
4005555555555555 1010101010101010101010101010101 4613187218303178069 +4613187218303178069 +0x1.5555555555555p+1 +2.666667e+00 +2.67
4005555555555555 1010101010101010101010101010101 4613187218303178069 +4613187218303178069 +0x1.5555555555555p+1 +2.666667e+00 +2.67
4005555555555555 1010101010101010101010101010101 4613187218303178069 +4613187218303178069 +0x1.5555555555555p+1 +2.666667e+00 +2.67
4005555555555555 1010101010101010101010101010101 4613187218303178069 +4613187218303178069 +0x1.5555555555555p+1 +2.666667e+00 +2.67
4005555555555555 1010101010101010101010101010101 4613187218303178069 +4613187218303178069 +0x1.5555555555555p+1 +2.666667e+00 +2.67
After:
4005555555555555 1010101010101010101010101010101 4613187218303178069 +4613187218303178069 +0x1.5555555555555p+1 +2.666667e+00 +2.67
4005555555555555 1010101010101010101010101010101 4613187218303178069 +4613187218303178069 +0x1.5555555555555p+1 +2.666667e+00 +2.67
43D008F03291623E 1100101001000101100010001111110 4886415423430550078 +4886415423430550078 +0x1.008f03291623ep+62 +4.621750e+18 +4621749617594791936.00
4005555555555555 1010101010101010101010101010101 4613187218303178069 +4613187218303178069 +0x1.5555555555555p+1 +2.666667e+00 +2.67
4005555555555555 1010101010101010101010101010101 4613187218303178069 +4613187218303178069 +0x1.5555555555555p+1 +2.666667e+00 +2.67

======================================================================
```

Рис. 8: результат выполнения вставки

Листинг:

Файл task3_8.c:

```c
void run_task3_8()
{
    printf("\nЗадание №8\n");
    printf("========================================================================\n");

    // Вариант 17: x = 8/3 для float/double
    double Mfl[N];

    size_t i = 2;
    double x = 9.87654321;

    for (size_t i = 0; i < N; i++) {
        Mfl[i] = 8.0 / 3.0;
    }

    printf("Before:\n");
    PRINT_ARRAY(Mfl, print64);

    __asm__ volatile (
        "vcvtsi2sdq %[x], %%xmm4, %%xmm4\n\t"          // int64 x -> double xmm4
        "vmovsd %%xmm4, (%[base], %[index], 8)\n\t"    // store xmm4 to M[i]
        :
        : [x] "r" (x),
          [base] "r" (Mfl),
          [index] "r" (i)
        : "xmm4", "memory"
    );

    printf("After:\n");
    PRINT_ARRAY(Mfl, print64);

    printf("\n========================================================================\n");
}
```