

Отчет

Лабораторная работа 4 (0100 = 4)

Использование ассемблерных вставок в программах на C++. Команды пересылки

Операционная система ОС Windows, 64-разрядная ОС.

Задание Л4.31.

*Как в задании Л1.34, создайте массивы M_8 из 16-битных целых чисел, M_4 из 32-битных целых чисел, M_2 из 64-битных целых чисел. Реализуйте для каждого массива M вставку, записывающую непосредственное значение 16 в $M[i]$ для заданного $i \in [0, N)$ 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*. Здесь и далее все целочисленные массивы до и после изменения выводите в шестнадцатеричном представлении.*

Программный код:

```
#include <stdio.h>

using namespace std;

// вывод целочисленных массивов
template <typename T>
void printArray(T *arr, int N, const char *mod) {
    for (int i = 0; i < N; i++) {
        printf(mod, arr[i], arr[i]);
    }
    printf("\n");
}

int main() {
    const int N = 5;

    // 16-битные целые числа
```

```

short Ms[N];

for (int i = 0; i < N; i++) {
    Ms[i] = 0xFADE;
}

// 32-битные целые числа
int MI[N];

for (int i = 0; i < N; i++) {
    MI[i] = 0xADEIAIDA;
}

// 64-битные целые числа
long long Mq[N];

for (int i = 0; i < N; i++) {
    Mq[i] = 0xCIA55IFIABIE;
}

printf("Massivy do izmeneniya:\n");
printArray(Ms, N, "%d, %04hX \n");
printArray(MI, N, "%d, %08X \n");
printArray(Mq, N, "%d, %016lX \n");

short i = 3;

asm volatile(
    "movw $16, %0\n"
    "movl $16, %1\n"
    "movq $16, %2\n"
    : "=m"(Ms[i]), "=m"(MI[i]), "=m"(Mq[i])
    :
    :);

```

```

printf("\nMassivy posle izmeneniya:\n");

printArray(Ms, N, "% 5d, %04hX \n");

printArray(Ml, N, "% 12d, %08X \n");

printArray(Mq, N, "% 12d, %016lX \n");

return 0;
}

```

Вывод:

```

Massivy do izmeneniya:
-1314, FADE
-1314, FADE
-1314, FADE
-1314, FADE
-1314, FADE

-1377721894, ADE1A1DA
-1377721894, ADE1A1DA
-1377721894, ADE1A1DA
-1377721894, ADE1A1DA
-1377721894, ADE1A1DA

1374792478, 0000C1A551F1AB1E
1374792478, 0000C1A551F1AB1E
1374792478, 0000C1A551F1AB1E
1374792478, 0000C1A551F1AB1E
1374792478, 0000C1A551F1AB1E

Massivy posle izmeneniya:
-1314, FADE
-1314, FADE
-1314, FADE
    16, 0010
-1314, FADE

-1377721894, ADE1A1DA
-1377721894, ADE1A1DA
-1377721894, ADE1A1DA
    16, 00000010
-1377721894, ADE1A1DA

    1374792478, 0000C1A551F1AB1E
    1374792478, 0000C1A551F1AB1E
    1374792478, 0000C1A551F1AB1E
        16, 0000000000000010
    1374792478, 0000C1A551F1AB1E

```

Задание Л4.32 (Вариант 2: массив M)

Реализуйте для одного из массивов M вставку, записывающую непосредственное (-1) в $M[i]$, где адрес начала массива M и индекс i передаются как входные параметры в регистрах.

Программный код:

```
#include <stdio.h>
```

```
#include <iostream>
```

```
using namespace std;
```

```
// вывод целочисленных массивов
```

```
template <typename T>
```

```
void printArray(T *arr, int N, const char *mod) {
```

```
    for (int i = 0; i < N; i++) {
```

```
        printf(mod, arr[i], arr[i]);
```

```
    }
```

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

```
}
```

```
int main() {
```

```
    const int N = 5;
```

```
    // 32-битные целые числа
```

```
    int MI[N];
```

```
    for (int i = 0; i < N; i++) {
```

```
        MI[i] = 0xADEIAIDA;
```

```
    }
```

```
    size_t i = 3;
```

```
    asm volatile(
```

```

"mov %0, %%rbx\n"
"mov %1, %%rcx\n"
"movl $-1, (%%rbx, %%rcx, 4)\n"
:
: "r"(Ml), "r"(i)
: "%rbx", "%rcx");

```

```
printArray(Ml, N, "%d, %08X \n");
```

```
return 0;
```

```
}
```

Вывод:

```

-1377721894, ADE1A1DA
-1377721894, ADE1A1DA
-1377721894, ADE1A1DA
-1, FFFFFFFF
-1377721894, ADE1A1DA

```

Задание Л4.з3. (Вариант 5: седьмой (старший) байт)

Реализуйте вставку, записывающую непосредственное значение 0xBB в заданный байт $Mq[i]$ (по варианту согласно таблице Л4.2; 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 регистрах.

Программный код:

```

#include <stdio.h>

#include <iostream>

using namespace std;

// вывод целочисленных массивов

template <typename T>

```

```

void printArray(T *arr, int N, const char *mod) {
    for (int i = 0; i < N; i++) {
        printf(mod, arr[i], arr[i]);
    }
    printf("\n");
}

int main() {
    const int N = 5;

    // 64-битные целые числа
    long long Mq[N];
    for (int i = 0; i < N; i++) {
        Mq[i] = 0xC1A551F1AB1E;
    }
    size_t i = 3;

    asm volatile(
        "mov %0, %%rbx\n"
        "mov %1, %%rcx\n"
        "movb $0xBB, 7(%%rbx,%%rcx,8)\n"
        :
        : "r"(Mq), "r"(i)
        : "%rbx", "%rcx");

    printArray(Mq, N, "%d, %016lX \n");

    return 0;
}

```

Вывод:

```
1374792478, 0000c1a551f1ab1e
1374792478, 0000c1a551f1ab1e
1374792478, 0000c1a551f1ab1e
1374792478, bb00c1a551f1ab1e
1374792478, 0000c1a551f1ab1e
```

Задание Л4.34.

Реализуйте вставку, записывающую в $M[i]$ значение x (M по варианту согласно таблице Л4.1; размер переменной x равен размеру элемента M), где значение x передаётся как входной параметр в памяти, M и i — как входные параметры в регистрах. Так как команда x86 не может адресовать два операнда в памяти, прямая пересылка $x \rightarrow M[i]$ невозможна; используйте промежуточный регистр (таблица Л4.3).

Программный код:

```
#include <stdio.h>

#include <iostream>

using namespace std;

// вывод целочисленных массивов

template <typename T>
void printArray(T *arr, int N, const char *mod) {
    for (int i = 0; i < N; i++) {
        printf(mod, arr[i], arr[i]);
    }
    printf("\n");
}

int main() {
    const int N = 5;

    // 32-битные целые числа

    int M[N];
```

```

for (int i = 0; i < N; i++) {
    MI[i] = 0xADE1A1DA;
}

int x = 0xABCDABCD;

size_t i = 3;

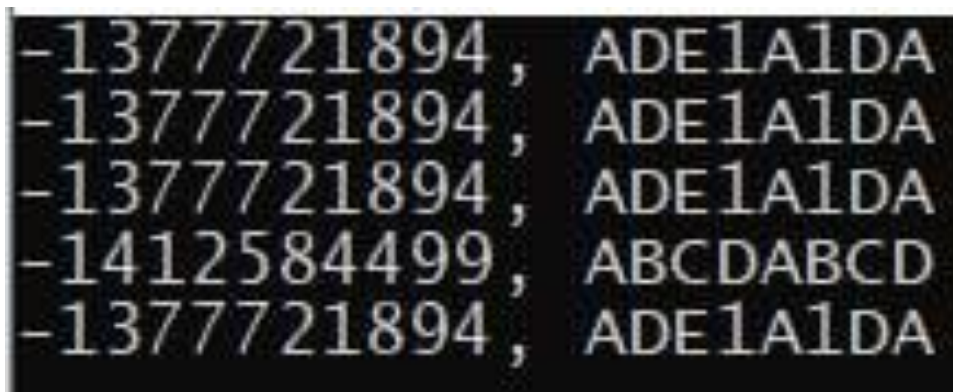
asm volatile(
    "mov %0, %%rbx\n"
    "mov %1, %%rcx\n"
    "movl %2, %%edi\n"
    "movl %%edi, (%%rbx, %%rcx, 4)\n"
    :
    : "r"(MI), "r"(i), "m"(x)
    : "%edi", "%rbx", "%rcx");

printArray(MI, N, "%d, %08X \n");

return 0;
}

```

Вывод:



```

-1377721894, ADE1A1DA
-1377721894, ADE1A1DA
-1377721894, ADE1A1DA
-1412584499, ABCDABCD
-1377721894, ADE1A1DA

```


Задание Л4.35.

Реализуйте вставку, записывающую в $M[i]$ значение x аналогично Л4.34, но во вставку передаётся адрес

Программный код

```
#include <stdio.h>

#include <iostream>

using namespace std;

// вывод целочисленных массивов
template <typename T>
void printArray(T *arr, int N, const char *mod) {
    for (int i = 0; i < N; i++) {
        printf(mod, arr[i], arr[i]);
    }
    printf("\n");
}

int main() {
    const int N = 5;

    // 32-битные целые числа
    int M[N];

    for (int i = 0; i < N; i++) {
        M[i] = 0xADEIAIDA;
    }

    int x = 0xABCDABCD;

    int *px = &x;

    size_t i = 3;
```

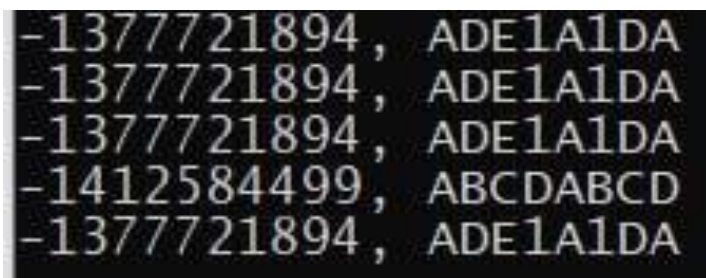
```

asm volatile(
    "mov %0, %%rbx\n"
    "mov %1, %%rcx\n"
    "mov %2, %%rdi\n"
    "movl (%%%rdi), %%eax\n"
    "movl %%eax, (%%rbx, %%rcx, 4)\n"
    :
    : "r"(Ml), "r"(i), "m"(px)
    : "%rax", "%rbx", "%rcx", "%rdi");

printArray(Ml, N, "%d, %08X \n");
return 0;
}

```

Вывод



```

-1377721894, ADE1A1DA
-1377721894, ADE1A1DA
-1377721894, ADE1A1DA
-1412584499, ABCDABCD
-1377721894, ADE1A1DA

```

Задание Л4.36.

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 Л4.4; переменные x , y , z , w передаются во вставку как параметры (z и w — выходные, x и y — входные)

Программный код:

```

#include <stdio.h>

using namespace std;

int main() {
    long long x = 20, y = 5, z, w;

```

asm volatile(

"movq %2, %%rax \n"

"movq %3, %%rbx \n"

"addq %%rbx, %%rax \n"

"movq %%rax, %0 \n"

"movq %2, %%rax \n"

"movq %3, %%rbx \n"

"subq %%rbx, %%rax \n"

"movq %%rax, %1 \n"

: "=m"(z), "=m"(w)

: "r"(x), "r"(y)

: "%rax", "%rbx");

printf("x = %d\n", x);

printf("y = %d\n", y);

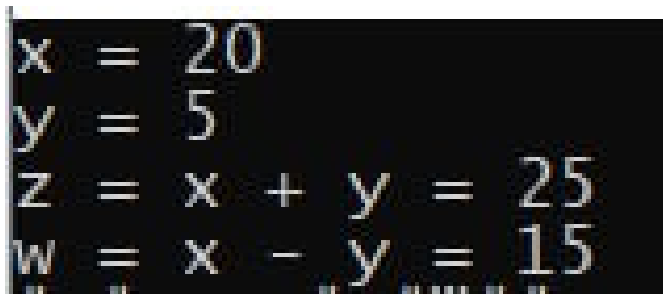
printf("z = x + y = %d\n", z);

printf("w = x - y = %d\n", w);

return 0;

}

Вывод:



```
x = 20
y = 5
z = x + y = 25
w = x - y = 15
```

Задание Л4.37. (Вариант 5 – Mfl , xmm4)

Как в задании Л1.34, создайте массивы Mfl из 64-битных чисел с плавающей запятой и Mfs из 32-битных чисел с плавающей запятой. Реализуйте вставку, записывающую в $M[i]$ значение x с плавающей запятой аналогично Л4.34 (M по варианту согласно таблице Л4.5; размер переменной x равен размеру элемента M ; x , M и i — параметры вставки), используя команды AVX $vmovsd/vmovss$ или их SSE-аналоги $movsd/movss$. Используйте промежуточный регистр $xmm\ i$.

Программный код:

```
#include <stdio.h>

#include <iostream>

using namespace std;

// вывод целочисленных массивов

template <typename T>
void printArray(T *arr, int N, const char *mod) {
    for (int i = 0; i < N; i++) {
        printf(mod, arr[i], arr[i]);
    }
    printf("\n");
}

int main() {
    const int N = 5;

    // 64-битные целые числа с плавающей запятой

    double Mfl[N];

    for (int i = 0; i < N; i++) {
        Mfl[i] = 5.0 / 3.0;
    }
```

```

double x = 2.0 / 3.0;

size_t i = 3;

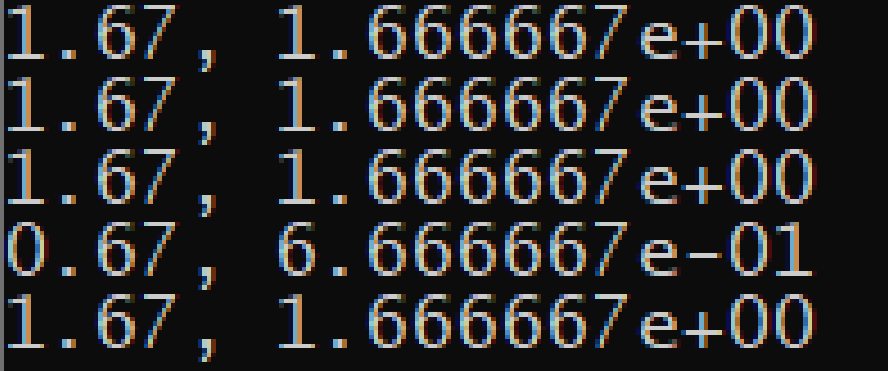
asm volatile(
    "mov %0, %%rbx\n"
    "mov %1, %%rcx\n"
    "movsd %2, %%xmm4\n"
    "movsd %%xmm4, (%%rbx, %%rcx, 8)\n"
    :
    : "r"(Mfl), "r"(i), "m"(x)
    : "%xmm4", "%rbx", "%rcx");

printArray(Mfl, N, "%.2lf, %le \n");

return 0;
}

```

Вывод:



```

1.67, 1.666667e+00
1.67, 1.666667e+00
1.67, 1.666667e+00
0.67, 6.666667e-01
1.67, 1.666667e+00

```

Задание Л4.38.

Реализуйте вставку, записывающую в $M[i]$ значение с плавающей запятой, равное целочисленному значению x . Преобразование целочисленного x к

нужному виду выполните при помощи команд AVX *vcvtsi2sd/vcvtsi2ss* или их SSE-аналогов *cvtsi2sd/cvtsi2s*

Программный код:

```
#include <stdio.h>

#include <iostream>

using namespace std;

// вывод целочисленных массивов
template <typename T>
void printArray(T *arr, int N, const char *mod) {
    for (int i = 0; i < N; i++) {
        printf(mod, arr[i], arr[i]);
    }
    printf("\n");
}

int main() {
    const int N = 5;

    // 64-битные целые числа с плавающей запятой
    double Mfl[N];

    for (int i = 0; i < N; i++) {
        Mfl[i] = 5.0 / 3.0;
    }

    long long x = 0xC1A551F1AB1E;
    size_t i = 3;
```

```
asm volatile(
    "mov %0, %%rbx\n"
    "mov %1, %%rcx\n"
    "cvttsi2sd %2, %%xmm4\n"
    "movsd %%xmm4, (%%rbx, %%rcx, 8)\n"
    :
    : "r"(Mfl), "r"(i), "m"(x)
    : "%xmm4", "%rbx", "%rcx");
```

```
printArray(Mfl, N, "%0.2lf, %le \n");
```

```
return 0;
```

```
}
```

Вывод:

```
1.67, 1.666667e+00
1.67, 1.666667e+00
1.67, 1.666667e+00
1374792478.00, 1.374792e+09
1.67, 1.666667e+00
```