федеральное государственное автономное образовательное учреждение
высшего образования «Национальный исследовательский университет
«Московский институт электронной техники»

Направление подготовки 09.03.04 «Программная инженерия»
Дисциплина «Электротехника»

Лабораторная РАБОТА №5
Использование ассемблерных вставок в программах на C++. Передача параметров во вставку. Перезаписываемые элементы

Работу выполнил студенты группы ПИН-24 Баранов Д.А. и Демочкина А.В.
Работу проверил ассистент Института СПИНТех Фомин Р.А.

Москва 2022

Цель работы: научиться вставлять в программы на языке высокого уровня ассемблерные фрагменты.

Для выполнения заданий выбран онлайн компилятор https://godbolt.org/.

ОС и разрядность ОС: GNU/Linux 64
Компилятор: Compiler Explorer x86-64 gcc 11.2
Архитектура: x86-64

Задание 1.

Вариант 2

2	$\begin{cases} z = (x+34)/y \\ w = (x+34)\%y \end{cases}$

Реализуйте расчёт беззнакового целочисленного выражения как ассемблерную вставку в программу на C/C++. При этом x, y, z, w — 32-битные локальные переменные функции $main()$ (или другой функции C/C++) и передаются во вставку как параметры (z и w — выходные, x и y — входные).

Входные параметры, соответствующие x и y (дайте им псевдонимы [X] и [Y]), должны располагаться в памяти. Затем, внутри вставки, при необходимости скопируйте [X] и [Y] в регистры общего назначения.

Проверьте расчёт, реализовав то же самое на C/C++.

```
#include <cstdio>
#include <iostream>
using namespace std;

int main() {
    cout << "Ассемблер: " << endl;
    unsigned int x =10, y = 3, z, w;
    asm (
    "movl   %[X], %%eax\n\t"
    "addl   $34, %%eax\n\t"
    "divl   %[Y]\n\t"
    :"=a"(z), "=d"(w)
    :[X]"m"(x), [Y]"m"(y)
    :"cc"
    );
    cout << "z = " << z << endl;
    cout << "w = " << w << endl << endl;

    cout << "C++: " << endl;
    z = (x + 34) / y;
    w = (x + 34) % y;
    cout << "z = " << z << endl;
    cout << "w = " << w << endl << endl;
    return 0;
}
```

```cpp
#include <cstdio>
#include <iostream>
using namespace std;

int main() {
    cout << "Ассемблер: " << endl;
    unsigned int x =10, y = 3, z, w;
    asm (
    "movl   %[X], %%eax\n\t"
    "addl   $34, %%eax\n\t"
    "divl   %[Y]\n\t"
    :"=a"(z), "=d"(w)
    :[X]"m"(x), [Y]"m"(y)
    :"cc"
    );
    cout << "z = " << z << endl;
    cout << "w = " << w << endl << endl;

    cout << "C++: " << endl;
    z = (x + 34) / y;
    w = (x + 34) % y;
    cout << "z = " << z << endl;
    cout << "w = " << w << endl << endl;
    return 0;
}
```

```
Ассемблер:
z = 14
w = 2

C++:
z = 14
w = 2
```

Помещение значений переменных x, y в память:

```
mov     DWORD PTR [rbp-12], 10
mov     DWORD PTR [rbp-16], 3
```

Отображение ассемблеровской вставки:

```
movl    DWORD PTR [rbp-12], %eax
addl    $34, %eax
divl    DWORD PTR [rbp-16]

mov     DWORD PTR [rbp-4], eax
mov     DWORD PTR [rbp-8], edx
```

Отображение программы, реализованной на C++:

```
        mov     eax, DWORD PTR [rbp-12]
        add     eax, 34
        mov     ecx, DWORD PTR [rbp-16]
        mov     edx, 0
        div     ecx
        mov     DWORD PTR [rbp-4], eax
        mov     eax, DWORD PTR [rbp-12]
        add     eax, 34
        mov     ecx, DWORD PTR [rbp-16]
        mov     edx, 0
        div     ecx
        mov     DWORD PTR [rbp-8], edx
```

Обращение к параметрам в памяти [X] и [Y]: %[X], %[Y]

Задание 2.
Реализуйте задание Л5.з1, располагая параметры [X] и [Y] в регистрах общего назначения ([X] в регистре A, [Y] — в выбираемом компилятором)

```
#include <cstdio>
#include <iostream>
using namespace std;

int main() {
    cout << "Ассемблер: " << endl;
    unsigned int x =10, y = 3, z, w;
    asm (
    "addl   $34, %%eax\n\t"
    "divl   %[Y]\n\t"
    :"=a"(z), "=d"(w)
    :[X]"a"(x), [Y]"r"(y)
    :"cc"
    );
    cout << "z = " << z << endl;
    cout << "w = " << w << endl << endl;

    return 0;
}
```

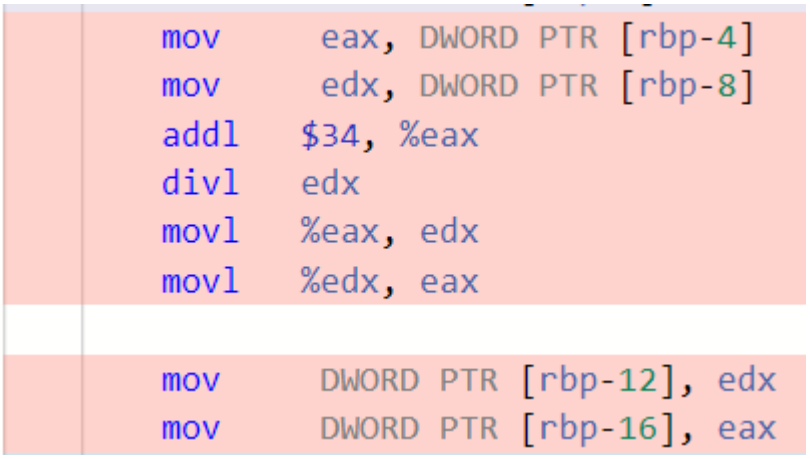
```
#include <cstdio>
#include <iostream>
using namespace std;

int main() {
    cout << "Ассемблер: " << endl;
    unsigned int x =10, y = 3, z, w;
    asm (
```

```
    "addl   $34, %%eax\n\t"
    "divl   %[Y]\n\t"
    :"=a"(z), "=d"(w)
    :[X]"a"(x), [Y]"r"(y)
    :"cc"
    );
    cout << "z = " << z << endl;
    cout << "w = " << w << endl << endl;

    return 0;
}
```

```
Ассемблер:
z = 14
w = 2
```



Параметр [X] расположен в регистре общего назначения eax. Параметр [Y] расположен в регистре, выбираемом компилятором (в регистре edx).

Обращение к параметрам [X] и [Y]: %[X], %[Y]

Задание 3.

```
#include <cstdio>
#include <iostream>
using namespace std;

int main() {
    cout << "Ассемблер: " << endl;
    unsigned int x =10, y = 3, z, w;
    asm (
    "xorl    %%edx, %%edx\n\t"
    "movl   (%[pX]), %%eax\n\t"
    "movl   (%[pY]), %%ebx\n\t"
    "add    $34, %%eax\n\t"
    "div    %%ebx"
    :"=a"(z), "=d"(w)
    :[pX]"r"(&x), [pY]"r"(&y)
    :"cc"
    );
    cout << "z = " << z << endl;
    cout << "w = " << w << endl << endl;

    return 0;
}
```

```
#include <cstdio>
#include <iostream>
using namespace std;

int main() {
    cout << "Ассемблер: " << endl;
    unsigned int x =10, y = 3, z, w;
    asm (
    "xorl    %%edx, %%edx\n\t"
    "movl   (%[pX]), %%eax\n\t"
    "movl   (%[pY]), %%ebx\n\t"
    "add    $34, %%eax\n\t"
    "div    %%ebx"
    :"=a"(z), "=d"(w)
    :[pX]"r"(&x), [pY]"r"(&y)
    :"cc"
    );
    cout << "z = " << z << endl;
    cout << "w = " << w << endl << endl;

    return 0;
}
```

```
Ассемблер:
z = 14
w = 2
```

```
    lea     rax, [rbp-12]
    lea     rdx, [rbp-16]
    xorl    %edx, %edx
    movl    (rax), %eax
    movl    (rdx), %ebx
    add     $34, %eax
    div     %ebx
    mov     DWORD PTR [rbp-4], eax
    mov     DWORD PTR [rbp-8], edx
```

Задание 4.

Реализуйте задание Л5.з1 для 16-битных x, y, z, w.

```
#include <cstdio>
#include <iostream>
using namespace std;

int main() {
    cout << "Ассемблер: " << endl;
    unsigned short x =10, y = 3, z, w;
    asm (
    "movw   %[X], %%ax\n\t"
    "addw   $34, %%ax\n\t"
    "divw   %[Y]\n\t"
    :"=a"(z), "=d"(w)
    :[X]"m"(x), [Y]"m"(y)
    :"cc"
    );

    cout << "z = " << z << endl;
    cout << "w = " << w << endl << endl;
}
```

```
#include <cstdio>
#include <iostream>
using namespace std;

int main() {
    cout << "Ассемблер: " << endl;
    unsigned short x =10, y = 3, z, w;
    asm (
    "movw   %[X], %%ax\n\t"
    "addw   $34, %%ax\n\t"
    "divw   %[Y]\n\t"
    :"=a"(z), "=d"(w)
    :[X]"m"(x), [Y]"m"(y)
    :"cc"
    );

    cout << "z = " << z << endl;
    cout << "w = " << w << endl << endl;
}
```

Ассемблер:
z = 14
w = 2

```
movw    WORD PTR [rbp-6], %ax
addw    $34, %ax
divw    WORD PTR [rbp-8]

mov     WORD PTR [rbp-2], ax
mov     WORD PTR [rbp-4], dx
```

Задание 5.

На языке C/C++ выделите память под массив M (статический или динамический) из N 32-битных целых чисел и инициализируйте M значениями 0x44332211.

Реализуйте для заданного $k \in [0, N)$ запись значения $x \neq 0$ на место элемента $M[k]$, используя компоненты эффективного адреса (*Base*, *Index*, 2 *Scale*).

```
#include <cstdio>
#include <iostream>
#include <random>
using namespace std;

const int N = 10;

int main() {
    srand(time(NULL));
    int M[N];
    size_t i = rand()%N;
    for (int j = 0; j < N; j++)
        M[j] = 0x44332211;

    cout << "До: ";
    for (int j = 0; j < N; j++)
        cout << hex << showbase << M[j] << " ";
    cout << endl;

    asm(
    "movl $65535, (%[M], %[I], 4)\n"
    : [I] "+r"(i)
    : [M] "d"(M)
    : "memory"
    );

    cout << "После: ";
    for (int j = 0; j < N; j++)
        cout << hex << showbase << M[j] << " ";
    cout << endl;
}
```

```cpp
#include <cstdio>
#include <iostream>
#include <random>
using namespace std;

const int N = 10;

int main() {
    srand(time(NULL));
    int M[N];
    size_t i = rand()%N;
    for (int j = 0; j < N; j++)
        M[j] = 0x44332211;

    cout << "До: ";
    for (int j = 0; j < N; j++)
        cout << hex << showbase << M[j] << " ";
    cout << endl;

    asm(
    "movl $65535, (%[M], %[I], 4)\n"
    : [I] "+r"(i)
    : [M] "d"(M)
    : "memory"
    );

    cout << "После: ";
    for (int j = 0; j < N; j++)
        cout << hex << showbase << M[j] << " ";
    cout << endl;
}
```

1 запуск:

```
До: 0x44332211 0x44332211 0x44332211 0x44332211 0x44332211 0x44332211 0x44332211 0x44332211 0x44332211 0x44332211
После: 0x44332211 0x44332211 0x44332211 0x44332211 0x44332211 0xffff 0x44332211 0x44332211 0x44332211 0x44332211
```

2 запуск:

```
До: 0x44332211 0x44332211 0x44332211 0x44332211 0x44332211 0x44332211 0x44332211 0x44332211 0x44332211 0x44332211
После: 0xffff 0x44332211 0x44332211 0x44332211 0x44332211 0x44332211 0x44332211 0x44332211 0x44332211 0x44332211
```

3 запуск:

```
До: 0x44332211 0x44332211 0x44332211 0x44332211 0x44332211 0x44332211 0x44332211 0x44332211 0x44332211 0x44332211
После: 0x44332211 0x44332211 0x44332211 0x44332211 0x44332211 0x44332211 0x44332211 0x44332211 0xffff 0x44332211
```

Задание 6.

Реализуйте для заданного $j \in [0, N)$, $j \neq k$ запись значения FF в старший байт элемента $M[j]$, используя все компоненты эффективного адреса.

```cpp
#include <cstdio>
#include <iostream>
#include <random>
using namespace std;

const int N = 10;

int main() {
    srand(time(NULL));
    int M[N];
    size_t i = rand()%N;
    size_t j = rand()%N;
    while (j == i)
        j = rand()%N;
    for (int j = 0; j < N; j++)
        M[j] = 0x44332211;

    cout << "До: ";
    for (int k = 0; k < N; k++)
        cout << hex << showbase << M[k] << " ";
    cout << endl;

    asm(
    "movl $65535, (%[M], %[I], 0x4)\n"
    "movb $0xFF, 0x3(%[M], %[J], 0x4)\n"
    : [I] "+r"(i), [J] "+r"(j)
    : [M] "d"(M)
    : "memory"
    );

    cout << "После: ";
    for (int k = 0; k < N; k++)
        cout << hex << showbase << M[k] << " ";
    cout << endl;
}
```

```cpp
#include <cstdio>
#include <iostream>
#include <random>
using namespace std;

const int N = 10;

int main() {
    srand(time(NULL));
    int M[N];
    size_t i = rand()%N;
    size_t j = rand()%N;
    while (j == i)
        j = rand()%N;
    for (int j = 0; j < N; j++)
        M[j] = 0x44332211;

    cout << "До: ";
```

```
    for (int k = 0; k < N; k++)
        cout << hex << showbase << M[k] << " ";
    cout << endl;

    asm(
    "movl $65535, (%[M], %[I], 0x4)\n"
    "movb $0xFF, 0x-1(%[M], %[J], 0x4)\n"
    : [I] "+r"(i), [J] "+r"(j)
    : [M] "d"(M)
    : "memory"
    );

    cout << "После: ";
    for (int k = 0; k < N; k++)
        cout << hex << showbase << M[k] << " ";
    cout << endl;
}
```

1 запуск:

```
До: 0x44332211 0x44332211 0x44332211 0x44332211 0x44332211 0x44332211 0x44332211 0x44332211 0x44332211 0x44332211
После: 0xffff 0x44332211 0x44332211 0x44332211 0x44332211 0x44332211 0x44332211 0x44332211 0x44332211 0xff332211
```

2 запуск:

```
До: 0x44332211 0x44332211 0x44332211 0x44332211 0x44332211 0x44332211 0x44332211 0x44332211 0x44332211 0x44332211
После: 0x44332211 0xff332211 0x44332211 0x44332211 0x44332211 0x44332211 0x44332211 0x44332211 0x44332211 0xffff
```

3 запуск:

```
До: 0x44332211 0x44332211 0x44332211 0x44332211 0x44332211 0x44332211 0x44332211 0x44332211 0x44332211 0x44332211
После: 0x44332211 0x44332211 0x44332211 0x44332211 0x44332211 0xffff 0xff332211 0x44332211 0x44332211 0x44332211
```

Л5.1. Дополнительные бонусные и штрафные баллы

—3 балла за неуказание в списке перезаписываемых элементов вставки явно или неявно модифицируемых (кроме параметров) в ней *регистров, памяти, флагов*, даже если это не влечёт некорректного расчёта в рассматриваемой программе.

Примечание: устаревшие версии компилятора могут выдавать сообщение о невозможности указания в списке перезаписываемых некоторых регистров (*esi*, *edi*, регистры расширений и т. п.). В этом (и только этом) случае регистр там не указывается, но должен сохраняться/восстанавливаться вручную.

Вопросы.

1. Каким ключевым словом открывается ассемблерная вставка?

asm либо __asm__

2. Где описываются выходные параметры ассемблерных вставок расширенного синтаксиса GCC? Что означают символы =, =&, + в начале строки ограничений выходного параметра?

__asm__ (вставка : список_выходных_операндов : список_входных_операндов : список_разрушаемых_регистров);

=/=&/+: инициализация и совмещение регистровых выходных параметров. Если при этом ограничение выходного параметра `[X]` в регистре ρ начинается с:

=, то регистр ρ никак не инициализируется перед вставкой, а *в том же регистре ρ может быть размещён какой-либо из входных параметров* (скорее всего в регистре ρ *будет* размещён один из входных параметров, так как компилятор стремится минимизировать общее число задействованных во вставке регистров);

=&, то регистр ρ никак не инициализируется перед вставкой (аналогично =), но (в отличие от =) в регистре ρ не может быть размещён никакой другой параметр (см. раздел 4.3.6);

+, то регистр ρ инициализируется перед вставкой значением `cvariablename` (значение `cvariablename` копируется в ρ), и в регистре ρ не может быть размещён никакой другой параметр.

3. Где описываются входные параметры?

__asm__ (вставка : список_выходных_операндов : список_входных_операндов : список_разрушаемых_регистров);

4. Где указывается список перезаписываемых [clobbers] во вставке регистров (кроме параметров)? Какая строка соответствуют изменению флагов *flags*? Какая строка соответствуют изменению памяти (кроме параметров)?

__asm__ (вставка : список_выходных_операндов : список_входных_операндов : список_разрушаемых_регистров);

5. Где описываются метки ЯВУ, на которые может быть передано управление из вставки?

```
asm [volatile] goto (
    "команды␣и␣директивы␣ассемблера"
    "как␣последовательная␣текстовая␣строка"
    :: <входные параметры> : <перезаписываемые элементы> :
    <метки>
    );
```

6. Какое ключевое слово нужно указать после asm, чтобы запретить компилятору оптимизировать вставку?

asm [volatile] (...