Лабораторная работа № 2:
“Отладка кода”

МИЭТ

Выполнили:

студенты группы МП-30

Алимагадов К. А., Карпухин Г. К.

Вариант № 9

Зеленоград 2018 год

Задание №1

Разработайте программу на языке C++, вычисляющую три целых выражения от целого аргумента (в соответствии с вариантом).

А) y(x) = x * 2

Б) y(x) = x * 13

В) $y(x) = \begin{cases} x, x > 13 \\ -1, x \leq 13 \end{cases}$

```
int main()
{
	int x = 2, y;
	y = x * 2;
	y = x * 13;
	if (x > 13)
		y = x;
	else
		y = -1;
	return 0;
}
```

Задание №2

Запустите программу и, используя инструменты отладчика (в частности, дизассемблер), изучите ассемблерный код, соответсвующий вычислениям, чтобы в окне дизасемблера перед каждой группой команд, соответствующих одному оператору языка высокого уровня, явно отображался этот оператор).

Занесите ассамблерный код, соответствующий вычислению y(x), в отчёт (код, не связанный с вычислением y(x), копировать в отчёт не нужно!). Определите и прокомментируйте:

- **Обращение к переменным x и y;**
- **Арифметические и логические операции – сложение, вычитание, умножение, деление с остатком на 2n и т. д. (по возможности);**
- **Сравнения и передачу управления в ветвлениях.**

```
int x = 2, y;
00BF19FE  mov         dword ptr [x],2  // начиная с адреса x будет
записано 32-х битное число 2
	y = x * 2;
```

```
00BF1A05  mov         eax,dword ptr [x]  // в eax присваивается 32-х
битное число, хранящееся в x
00BF1A08  shl         eax,1  // сдвиг eax влево на 1 разряд
00BF1A0A  mov         dword ptr [y],eax  // начиная с адреса y будет
записано 32-х битное число, хранящееся в eax
     y = x * 13;
00BF1A0D  imul        eax,dword ptr [x],0Dh  // в eax присваивается
произведение 32-х битного числа, хранящегося в x, и 13
00BF1A11  mov         dword ptr [y],eax // начиная с адреса y будет
записано 32-х битное число, хранящееся в eax
     if (x > 13)
00BF1A14  cmp         dword ptr [x],0Dh  // сравнение x и 13
00BF1A18  jle         main+42h (0BF1A22h)  // условный переход по
адресу main+42h (0BF1A22h), если выполняется условие (x < 13)
          y = x;
00BF1A1A  mov         eax,dword ptr [x]  // в eax записывается 32-х
битное число, хранящееся в x.
00BF1A1D  mov         dword ptr [y],eax  // начиная с адреса y будет
записано 32-х битное число, хранящееся в eax
     else
00BF1A20  jmp         main+49h (0BF1A29h)  // безусловный переход по
адресу main+49h (0BF1A29h)
          y = -1;
00BF1A22  mov         dword ptr [y],0FFFFFFFFh // начиная с адреса y
будет записано 32-х битное число -1
     return 0;
00BF1A29  xor         eax,eax // xor(eax,eax) == 0
```

Задание №3

Внесите в программу из задания 1, а) изменения (либо, что предпочтительнее, добавьте новые фрагменты кода, выполняющие аналогичные вычисления для других переменных, используя макросы препроцессора или шаблоны C++).

- сделайте переменные глобальными;

- измените тип с int на char, short, long и long long;

- измените тип с int на long double.

Опишите в отчёте различия в ассемблерном коде.

Задание №4

Оформите вычисления из задания 1, а) как целую функцию от целого аргумента. Опишите в отчёте код вызова функции. Как передаётся аргумент? Как возвращается значение?

Задание №5

Измените тип аргумента и результата на вещественный. Опишите в отчёте код вызова функции. Как передаётся аргумент? Как возвращается значение?

Задание №6

Бонус (+2 балла)

Используйте в функции статическую переменную. Как выглядит обращение к ней?

Задание №7

Бонус(+2 балла за платформу):

Запустите тестовую программу (программы), используя платформу и/или компилятор, отличные от GNU/Linux и GCC. Результат с пояснениями внести в конспект.

```
int x1, y;

template <typename T>
T First(T const x)
{
	return x * 2;
}

template <typename T>
T Second(T const x)
{
	return x * 13;
}

template <typename T>
T Third(T const x)
{
	if (x < 0)
	{
		return T(7);
	}
	else
		return T(13);
}

int main()
{
	x1 = 2;
	char y2, x2 = 2;
```

```
      short y3, x3 = 2;
      long y4, x4 = 2;
      long long y5, x5 = 2;
      long double y6, x6 = 2;
      static int y7, x7 = 2;
y = First<int>(x1);
{
      return x * 2;
012C181E  mov         eax,dword ptr [x]  // начиная с адреса x будет
записано 32-х битное число в eax
012C1821  shl         eax,1  // Сдвиг eax на 1 бит влево
}

y2 = First<char>(x2);
{
      return x * 2;
012C179E  movsx       eax,byte ptr [x]  // начиная с адреса x будет
записано 8-ми битное число в eax
012C17A2  shl         eax,1 // Сдвиг eax на 1 бит влево
}
y3 = First<short>(x3);
{
      return x * 2;
012C17DE  movsx       eax,word ptr [x]  // начиная с адреса x будет
записано 16-ти битное число в eax
012C17E2  shl         eax,1  // Сдвиг eax на 1 бит влево
}

y4 = First<long>(x4);

{
      return x * 2;
012C185E  mov         eax,dword ptr [x]  // начиная с адреса x будет
записано 32-х битное число в eax
012C1861  shl         eax,1  // Сдвиг eax на 1 бит влево
}

y5 = First<long long>(x5);
{
      return x * 2;
012C18EE  push        0  // помещает 0 в стек
012C18F0  push        2  // помещает 2 в в стек
012C18F2  mov         eax,dword ptr [ebp+0Ch]  // начиная с адреса
ebp+0Ch будет записано 32-х битное число в eax
012C18F5  push        eax  // помещает значение eax в стек
012C18F6  mov         ecx,dword ptr [x]  // начиная с адреса x будет
записано 32-х битное число в ecx
012C18F9  push        ecx  // помещает значение ecx в в стек
012C18FA  call        __allmul (012C1285h)  // передаёт управление по
адресу
}
```

```
y6 = First<long double>(x6);
{
     return x * 2;
012C189E  movsd       xmm0,mmword ptr [x]  // начиная с адреса x будет
записано 64-х битное число в xmm0
012C18A3  mulsd       xmm0,mmword ptr [__real@4000000000000000
(012C7B30h)]  // записывает в xmm0 произведение xmm0 на число,
хранящееся по адресу (012C7B30h)
012C18AB  movsd       mmword ptr [ebp-0C8h],xmm0  // начиная с адреса
ebp-0C8h будет записано 64-х битное число в ecx
012C18B3  fld         qword ptr [ebp-0C8h]  // Загрузить вещественное
число в вершину стека из области памяти.
}
```

Разница в ассемблерном коде наблюдается в зависимости от типа данных, который передаётся в функцию. К примеру, при работе работе с char используется byte ptr, который указывает на 8 битный участок памяти, а при работе с short word ptr, который указвает на 16 битный участок памяти.

```
int y8, x8 = 2;
double y9, x9 = 2;
```

Задание №4

```
y8 = First<int>(x8);
002E1A82  mov         eax,dword ptr [x8]  // начиная с адреса x8 будет
записано 32-х битное число в eax
002E1A85  push        eax  // eax помещается в стек
002E1A86  call        First<int> (02E13D9h)

{
002E19E0  push        ebp
002E19E1  mov         ebp,esp
002E19E3  sub         esp,0C0h
002E19E9  push        ebx
002E19EA  push        esi
002E19EB  push        edi
002E19EC  lea         edi,[ebp-0C0h]
002E19F2  mov         ecx,30h
002E19F7  mov         eax,0CCCCCCCCh
002E19FC  rep stos    dword ptr es:[edi]
     return x * 2;
002E19FE  mov         eax,dword ptr [x]
002E1A01  shl         eax,1
}
002E1A03  pop         edi
002E1A04  pop         esi
002E1A05  pop         ebx
002E1A06  mov         esp,ebp
```

```
002E1A08  pop         ebp
002E1A09  ret // возвращение значения
```

Задание №5

```
y9 = First<double>(x9);
002E1A91  sub         esp,8  // в esp записывается результат (esp – 8)
002E1A94  movsd       xmm0,mmword ptr [x9] // Значение, записанное по
адресу x9, записывается в xmm0
002E1A99  movsd       mmword ptr [esp],xmm0  // значение xmm0
записывается по адресу [esp]
002E1A9E  call        First<double> (02E13D4h)
{
002E1A10  push        ebp
002E1A11  mov         ebp,esp
002E1A13  sub         esp,0C8h
002E1A19  push        ebx
002E1A1A  push        esi
002E1A1B  push        edi
002E1A1C  lea         edi,[ebp-0C8h]
002E1A22  mov         ecx,32h
002E1A27  mov         eax,0CCCCCCCCh
002E1A2C  rep stos    dword ptr es:[edi]
     return x * 2;
002E1A2E  movsd       xmm0,mmword ptr [x]
002E1A33  mulsd       xmm0,mmword ptr [__real@4000000000000000
(02E7B30h)]
002E1A3B  movsd       mmword ptr [ebp-0C8h],xmm0
002E1A43  fld         qword ptr [ebp-0C8h]
}
002E1A49  pop         edi
002E1A4A  pop         esi
002E1A4B  pop         ebx
002E1A4C  mov         esp,ebp
002E1A4E  pop         ebp
002E1A4F  ret  // возвращение значения
```

Задание №6

```
y7 = First<int>(x7);
01053D44  mov         eax,dword ptr ds:[0105A018h] // В eax
записывается значение расположенное по адресу [0105A018h]
01053D49  push        eax  // значение eax записывается в стек
01053D4A  call        First<int> (01051163h)

{
010516F0  push        ebp
010516F1  mov         ebp,esp
010516F3  sub         esp,0C0h
010516F9  push        ebx
```

```
010516FA  push        esi
010516FB  push        edi
010516FC  lea         edi,[ebp-0C0h]
01051702  mov         ecx,30h
01051707  mov         eax,0CCCCCCCCh
0105170C  rep stos    dword ptr es:[edi]
     return x * 2;
0105170E  mov         eax,dword ptr [x]  // начиная с адреса x будет
записано 32-х битное число в eax
01051711  shl         eax,1  // Сдвиг eax на 1 бит влево
}
01051713  pop         edi
01051714  pop         esi
01051715  pop         ebx
01051716  mov         esp,ebp
01051718  pop         ebp
01051719  ret  // возвращение значения
return 0;
012C54B1  xor         eax,eax  // xor(eax,eax) == 0
}
```

Программы были запущены в Visual Studio 2017.