Лабораторная работа 2.
Отладка кода

Цель работы: научиться использовать инструменты отладки современных IDE; исследовать генерируемый компилятором код.

Л2.1. Задание на лабораторную работу

Задание 1. Разработайте программу на языке C++, вычисляющую три целых выражения от целого аргумента (в соответствии с вариантом).

а) $y(x) = 3 * x$

```
#include <stdlib.h>
int main(void)
{
  int x=20;
  int y=3*x;
}
```

б) $y(x) = x - 21$

```
#include <stdlib.h>
int main(void)
{
  int x=20;
  int y=x-21;
}
```

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

```
#include <stdlib.h>
int main(void)
{
  int x=20;
  int y;
  if (x>13)
          y= x;
  else
          y=-1;
}
```

Задание 2. Запустите программу и, используя инструменты отладчика (в частности, дизассемблер), изучите ассемблерный код, соответствующий вычислениям (для Code::Blocks – переключитесь на синтаксис AT&T и включите Mixed mode, чтобы в окне дизассемблера перед каждой группой команд, соответствующих одному оператору языка высокого уровня, явно отображался этот оператор).

Занесите ассемблерный код, соответствующий вычислению $y(x)$, в отчёт (код, не связанный с вычислением $y(x)$, копировать в отчёт не нужно!).

а) $y(x) = 3 * x$

```
mov    -0x4(%rbp),%edx
mov    %edx,%eax
add    %eax,%eax
add    %edx,%eax
mov    %eax,-0x8(%rbp)
```

б) $y(x) = x - 21$

```
mov    -0x4(%rbp),%eax
sub    $0x15,%eax
mov    %eax,-0x8(%rbp)
```

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

```
cmpl   $0xd,-0x4(%rbp)
jle    0x401eb5 <main()+52>
mov    -0x4(%rbp),%eax
jmp    0x401eba <main()+57>
mov    $0xffffffff,%eax
mov    %eax,-0x8(%rbp)
```

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

```
x=20;
movl   $0x14,0x804a018
mov -0x804a018, %eax
sar $0x1f, %eax
mov %eax, %edx
xor -0x8(%ebp), %edx
mov %edx, -0x4(%ebp)
sub %eax, -0x4(%ebp)
mov %eax, 0x804a018
mov $0x0, %eax
```

Измените тип с int на char, short, long и long long

char

```
;9  :         char x = 20;
0x401e8e  movb   $0x14,-0x1(%rbp)
;10 :         char y;
;11 :              y = x * 3;
0x401e92  movzbl -0x1(%rbp),%edx
0x401e96  mov    %edx,%eax
0x401e98  add    %eax,%eax
0x401e9a  add    %edx,%eax
0x401e9c  mov    %al,-0x2(%rbp)
```

short

```
;9  :         short x = 20;
0x401e8e  movw   $0x14,-0x2(%rbp)
;10 :         short y;
;11 :              y = x * 3;
0x401e94  movzwl -0x2(%rbp),%edx
0x401e98  mov    %edx,%eax
0x401e9a  add    %eax,%eax
0x401e9c  add    %edx,%eax
0x401e9e  mov    %ax,-0x4(%rbp)
```

long long

```
;9  :         long long x = 20;
0x401e8e  movq   $0x14,-0x8(%rbp)
;10 :         long long y;
;11 :              y = x * 3;
0x401e96  mov    -0x8(%rbp),%rdx
0x401e9a  mov    %rdx,%rax
0x401e9d  add    %rax,%rax
0x401ea0  add    %rdx,%rax
0x401ea3  mov    %rax,-0x10(%rbp)
```

Задание 4. Оформите вычисления из задания 1, а) как целую функцию от целого аргумента.

```
int foo(int x)
{
    return 3*x;
}

int main(void)
{
  int x=20;
  int y=foo(x);
}
```

Опишите в отчёте код вызова функции.

```
;14 :              y = foo(x);
0x401ea7  mov    -0x4(%rbp),%eax
0x401eaa  mov    %eax,%ecx
0x401eac  callq  0x401e81 <foo(int)>
0x401eb1  mov    %eax,-0x8(%rbp)
```

Задание 5. Измените тип аргумента и результата на вещественный.
Опишите в отчёте код вызова функции.

```
;14 :              y = foo(x);
0x401ec5  mov    -0x4(%rbp),%eax
0x401ec8  mov    %eax,%ecx
0x401eca  callq  0x401e81 <foo(int)>
0x401ecf  movsd  %xmm0,-0x18(%rbp)
0x401ed4  mov    -0x18(%rbp),%rax
0x401ed8  mov    %rax,-0x10(%rbp)
```

Задание 6. Бонус (+2 балла). Используйте в функции статическую переменную.
Как выглядит обращение к ней?

```
;14 :              y = foo();
0x401ea0  callq  0x401e81 <foo()>
0x401ea5  mov    %eax,-0x4(%rbp)
```

```
static int x = 20;
;7  :     int foo() {
0x401e81  push   %rbp
0x401e82  mov    %rsp,%rbp
;8  :         return x*3;
0x401e85  mov    0xe185(%rip),%edx        # 0x410010
0x401e8b  mov    %edx,%eax
0x401e8d  add    %eax,%eax
0x401e8f  add    %edx,%eax
;9  :     }
0x401e91  pop    %rbp
0x401e92  retq
```

Вопросы:

1. Уметь пользоваться окнами просмотра переменных и содержимого памяти в отладчике используемой вами IDE.

+

2. Чем различается размещение в памяти локальных, глобальных и статических переменных?

Локальные переменные подпрограмм находятся в сегменте стека, также оптимизирующие компиляторы могут помещать часть целочисленных переменных в регистры общего назначения.

Глобальные переменные программы, доступные в любой её точке и статические переменные, отличающиеся от глобальных только областью видимости, расположены в сегменте данных. Те глобальные и статические переменные, которые не были инициализированы при объявлении, отделяются в специальный сегмент BSS.

3. Чем различается работа с целыми числами разной разрядности?

Для большей разрядности может быть необходимо обрабатывать значение по частям.

4. Чем различается работа с целыми и вещественными числами?

С фиксированной запятой: целая и дробная части
С плавающей запятой: $X = N^p \mu$

5. Как в функции передаются целые параметры (в исследуемом компиляторе и платформе)?

```
mov    -0x8(%ebp),%eax
```

6. Как в функции передаются вещественные параметры (в исследуемом компиляторе и платформе)?

```
mov    (%esp),%eax
```