Лабораторная работа №2
Вариант 6

Задание 1

```
int main()
{
    int x = 6;
    int y = x / 2;
    y = x / 3;
    y = x > 13 ? x : -1;
    return 0;
}
```

Задание 2

```
    int x = 6;
```

```
  movl $6, -8(%rbp)
```

```
    int y = x / 2;
```

присваивание

```
  movl -8(%rbp), %eax
  movl %eax, %edx
```

Расширенный сдвиг

```
  shrl $31, %edx
```

Сложение

```
  addl %edx, %eax
```

арифметический сдвиг

```
 sarl %eax
```

Присваивание

```
 movl %eax, -4(%rbp)
```

```
    y = x / 3;
```

присваивание

```
  movl -8(%rbp), %eax
  movslq %eax, %rdx
```

Знаковое умножение с указанием результата

```
  imulq $1431655766, %rdx, %rdx
```

логический сдвиг вправо

```
  shrq $32, %rdx
```

арифметический сдвиг (логический, но с другим заполнением)

```
  sarl $31, %eax
```

присваивание

```
  movl %edx, %ecx
```

вычитание

```
  subl %eax, %ecx
```

присваивание

```
  movl %ecx, %eax
  movl %eax, -4(%rbp)
```

```
 if (x > 13)
{
   y = x;
}
else
{
   y = -1;
}
```

Сравнение

```
cmpl $13, -8(%rbp)
```

Если да

```
 jle .L2
 movl -8(%rbp), %eax
 movl %eax, -4(%rbp)
```

Не идем в else

```
 jmp .L3
```

else

```
.L2:
 movl $-1, -4(%rbp)
```

Задание 3

```
template<typename T>
T func(T x)
{
   return x / 2;
}

char _char = 15;
short _short = 15;
long _long = 15;
long long _long_long = 15;
long double _long_double = 15;

int main()
{
   char y_char = func(_char);
   short y_short = func(_short);
   long y_long = func(_long);
   long long y_long_long = func(_long_long);
   long double y_long_double = func(_long_double);

   return 0;
}
```

```
_char:
  .byte 15
_short:
  .value 15
_long:
  .quad 15
_long_long:
  .quad 15
_long_double:
  .long 0
  .long 4026531840
  .long 16386
  .long 0
main:
  endbr64
  pushq %rbp
  movq %rsp, %rbp
  subq $48, %rsp
  movzbl _char(%rip), %eax
  movsbl %al, %eax
  movl %eax, %edi
  call _Z4funcIcET_S0_
  movb %al, -35(%rbp)
  movzwl _short(%rip), %eax
  cwtl
  movl %eax, %edi
  call _Z4funcIsET_S0_
  movw %ax, -34(%rbp)
  movq _long(%rip), %rax
  movq %rax, %rdi
  call _Z4funcIlET_S0_
  movq %rax, -32(%rbp)
  movq _long_long(%rip), %rax
  movq %rax, %rdi
  call _Z4funcIxET_S0_
  movq %rax, -24(%rbp)
  fldt _long_double(%rip)
  leaq -16(%rsp), %rsp
  fstpt (%rsp)
  call _Z4funcIeET_S0_
  addq $16, %rsp
  fstpt -16(%rbp)
  movl $0, %eax
  leave
  ret
_Z4funcIcET_S0_:
  endbr64
  pushq %rbp
  movq %rsp, %rbp
  movl %edi, %eax
  movb %al, -4(%rbp)
```

```
 movzbl -4(%rbp), %eax
 movl %eax, %edx
 shrb $7, %dl
 addl %edx, %eax
 sarb %al
 popq %rbp
 ret
_Z4funcIsET_S0_:
 endbr64
 pushq %rbp
 movq %rsp, %rbp
 movl %edi, %eax
 movw %ax, -4(%rbp)
 movzwl -4(%rbp), %eax
 movl %eax, %edx
 shrw $15, %dx
 addl %edx, %eax
 sarw %ax
 popq %rbp
 ret
_Z4funcIlET_S0_:
 endbr64
 pushq %rbp
 movq %rsp, %rbp
 movq %rdi, -8(%rbp)
 movq -8(%rbp), %rax
 movq %rax, %rdx
 shrq $63, %rdx
 addq %rdx, %rax
 sarq %rax
 popq %rbp
 ret
_Z4funcIxET_S0_:
 endbr64
 pushq %rbp
 movq %rsp, %rbp
 movq %rdi, -8(%rbp)
 movq -8(%rbp), %rax
 movq %rax, %rdx
 shrq $63, %rdx
 addq %rdx, %rax
 sarq %rax
 popq %rbp
 ret
_Z4funcIeET_S0_:
 endbr64
 pushq %rbp
 movq %rsp, %rbp
 fldt 16(%rbp)
 fldt .LC1(%rip)
 fdivrp %st, %st(1)
 popq %rbp
```

```
  ret
.LC1:
  .long 0
  .long 2147483648
  .long 16384
  .long 0
0:
1:
2:
3:
4:
```

```
    char y_char = func(_char);
```

```
  movzbl _char(%rip), %eax
  movsbl %al, %eax
  movl %eax, %edi
  call _Z4funcIcET_S0_
  movb %al, -35(%rbp)


_Z4funcIcET_S0_:
  endbr64
  pushq %rbp
  movq %rsp, %rbp
  movl %edi, %eax
  movb %al, -4(%rbp)
  movzbl -4(%rbp), %eax
  movl %eax, %edx
  shrb $7, %dl
  addl %edx, %eax
  sarb %al
  popq %rbp
  ret
```

```
    short y_short = func(_short);
```

```
  movzwl _short(%rip), %eax
  cwtl
  movl %eax, %edi
  call _Z4funcIsET_S0_
  movw %ax, -34(%rbp)

_Z4funcIlET_S0_:
  endbr64
  pushq %rbp
  movq %rsp, %rbp
  movq %rdi, -8(%rbp)
  movq -8(%rbp), %rax
  movq %rax, %rdx
  shrq $63, %rdx
```

```
addq %rdx, %rax
sarq %rax
popq %rbp
ret
```

```
  long y_long = func(_long);
```

```
movq _long(%rip), %rax
movq %rax, %rdi
call _Z4funcIlET_S0_
movq %rax, -32(%rbp)

_Z4funcIlET_S0_:
endbr64
pushq %rbp
movq %rsp, %rbp
movq %rdi, -8(%rbp)
movq -8(%rbp), %rax
movq %rax, %rdx
shrq $63, %rdx
addq %rdx, %rax
sarq %rax
popq %rbp
ret
```

```
  long long y_long_long = func(_long_long);
```

```
movq _long_long(%rip), %rax
movq %rax, %rdi
call _Z4funcIxET_S0_
movq %rax, -24(%rbp)

_Z4funcIxET_S0_:
endbr64
pushq %rbp
movq %rsp, %rbp
movq %rdi, -8(%rbp)
movq -8(%rbp), %rax
movq %rax, %rdx
shrq $63, %rdx
addq %rdx, %rax
sarq %rax
popq %rbp
ret
```

```
  long double y_long_double = func(_long_double);
```

```
fldt _long_double(%rip)
leaq -16(%rsp), %rsp
fstpt (%rsp)
call _Z4funcIeET_S0_
addq $16, %rsp
```

```
 fstpt -16(%rbp)
_Z4funcIeET_S0_:
 endbr64
 pushq %rbp
 movq %rsp, %rbp
 fldt 16(%rbp)
 fldt .LC1(%rip)
 fdivrp %st, %st(1)
 popq %rbp
 ret
```

Для каждого типа была определена функция. Теперь в переменную не просто идет запись с помощью mov, есть предобработка. Глобальные переменные вынеслись вверх.

Задание №4

```
int func(int x)
{
   return x / 2;
}

int main()
{
   int x = 15;
   int y = func(x);
   return 0;
}
```

```
_Z4funci:
 endbr64
 pushq %rbp
 movq %rsp, %rbp
 movl %edi, -4(%rbp)
 movl -4(%rbp), %eax
 movl %eax, %edx
 shrl $31, %edx
 addl %edx, %eax
 sarl %eax
 popq %rbp
 ret
main:
 endbr64
 pushq %rbp
 movq %rsp, %rbp
 subq $16, %rsp
 movl $15, -8(%rbp)
 movl -8(%rbp), %eax
 movl %eax, %edi
 call _Z4funci
 movl %eax, -4(%rbp)
 movl $0, %eax
 leave
 ret
```

```
0:
1:
2:
3:
4:
```

Код вызова функции:

```
   int y = func(x);
```

```
  movl -8(%rbp), %eax
  movl %eax, %edi
  call _Z4funci
  movl %eax, -4(%rbp)
```

Передача аргументов происходит по адресу:

```
  movl -8(%rbp), %eax
```

```
_Z4funci:
  endbr64
  pushq %rbp
  movq %rsp, %rbp
  movl %edi, -4(%rbp)
  movl -4(%rbp), %eax
  movl %eax, %edx
  shrl $31, %edx
  addl %edx, %eax
  sarl %eax
  popq %rbp
  ret
```

Возврат по адресу:

```
  movl %eax, -4(%rbp)
```

Задание № 5

```
float func(int x)
{
   return (float)x / 2;
}

int main()
{
   int x = 15;
   float y = func(x);
   return 0;
}
```

```
_Z4funci:
  endbr64
  pushq %rbp
  movq %rsp, %rbp
```

```
  movl %edi, -4(%rbp)
  movl -4(%rbp), %eax
  movl %eax, %edx
  shrl $31, %edx
  addl %edx, %eax
  sarl %eax
  popq %rbp
  ret
main:
  endbr64
  pushq %rbp
  movq %rsp, %rbp
  subq $16, %rsp
  movl $15, -8(%rbp)
  movl -8(%rbp), %eax
  movl %eax, %edi
  call _Z4funci
  movl %eax, -4(%rbp)
  movl $0, %eax
  leave
  ret
0:
1:
2:
3:
4:
```

Аргумент передается и возвращается по значению.
Видно, что изменился первый оператор перемещения. movl -> movq (4 байта -> 8 байт).

Задание №6

```
int func(int x)
{
   return x / 2;
}

int main()
{
   const int x = 15;
   int y = func(x);
   return 0;
}
```

```
_Z4funci:
  endbr64
  pushq %rbp
  movq %rsp, %rbp
  movl %edi, -4(%rbp)
  movl -4(%rbp), %eax
  movl %eax, %edx
  shrl $31, %edx
  addl %edx, %eax
```

```
  sarl %eax
  popq %rbp
  ret
main:
  endbr64
  pushq %rbp
  movq %rsp, %rbp
  subq $16, %rsp
  movl $15, -8(%rbp)
  movl $15, %edi
  call _Z4funci
  movl %eax, -4(%rbp)
  movl $0, %eax
  leave
  ret
0:
1:
2:
3:
4:
```

Теперь мы обращаемся к переменной без посредников.

Задание №7

icc x86-64 12.1.18

```
int func(int x)
{
   return x / 2;
}

int main()
{
   const int x = 15;
   int y = func(x);
   return 0;
}
```

```
push     rbp                                  #2.1
mov      rbp, rsp                              #2.1
sub      rsp, 16                               #2.1
mov      DWORD PTR [-16+rbp], edi                    #2.1
mov      eax, DWORD PTR [-16+rbp]                    #3.12
mov      edx, 0                                #3.16
test     eax, eax                              #3.16
setl     dl                                   #3.16
add      edx, DWORD PTR [-16+rbp]                    #3.16
sar      edx, 1                                #3.16
mov      eax, edx                               #3.16
leave                                         #3.16
ret                                          #3.16
```

```
main:
push    rbp                                 #7.1
mov     rbp, rsp                            #7.1
sub     rsp, 16                             #7.1
mov     DWORD PTR [-16+rbp], 15             #8.17
mov     eax, 15                             #9.13
mov     edi, eax                            #9.13
call    func(int)                           #9.13
mov     DWORD PTR [-12+rbp], eax            #9.13
mov     eax, DWORD PTR [-12+rbp]            #9.13
mov     DWORD PTR [-8+rbp], eax             #9.11
mov     eax, 0                              #10.12
leave                                       #10.12
ret                                         #10.12
```

Изменился способ передачи аргумента в функцию. Теперь он передает по стеку.