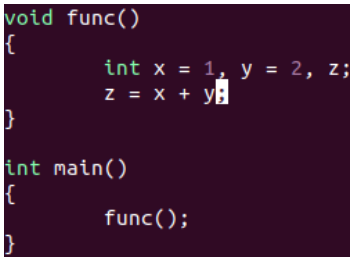
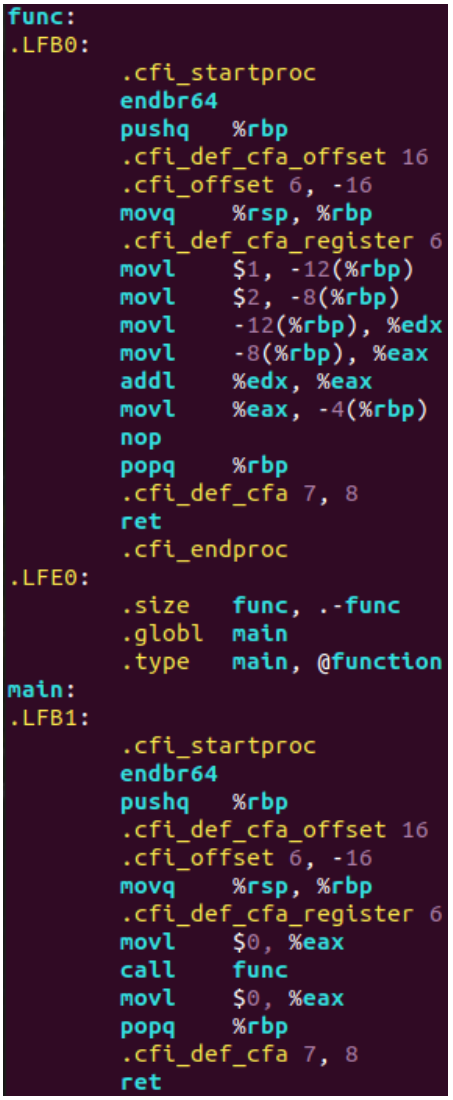
Lab 3/4

Lab 3

1. Functions

point	C	32bit	64bit
a. No args no ret			

```
void func()
{
        int x = 1, y = 2, z;
        z = x + y;
}

int main()
{
        func();
}
```

```
func:
.LFB0:
        .cfi_startproc
        endbr32
        pushl   %ebp
        .cfi_def_cfa_offset 8
        .cfi_offset 5, -8
        movl    %esp, %ebp
        .cfi_def_cfa_register 5
        subl    $16, %esp
        call    __x86.get_pc_thunk.ax
        addl    $_GLOBAL_OFFSET_TABLE_, %eax
        movl    $1, -12(%ebp)
        movl    $2, -8(%ebp)
        movl    -12(%ebp), %edx
        movl    -8(%ebp), %eax
        addl    %edx, %eax
        movl    %eax, -4(%ebp)
        nop
        leave
        .cfi_restore 5
        .cfi_def_cfa 4, 4
        ret
        .cfi_endproc
.LFE0:
        .size   func, .-func
        .globl  main
        .type   main, @function
main:
.LFB1:
        .cfi_startproc
        endbr32
        pushl   %ebp
        .cfi_def_cfa_offset 8
        .cfi_offset 5, -8
        movl    %esp, %ebp
        .cfi_def_cfa_register 5
        call    __x86.get_pc_thunk.ax
        addl    $_GLOBAL_OFFSET_TABLE_, %eax
        call    func
        movl    $0, %eax
        popl    %ebp
```

```
func:
.LFB0:
        .cfi_startproc
        endbr64
        pushq   %rbp
        .cfi_def_cfa_offset 16
        .cfi_offset 6, -16
        movq    %rsp, %rbp
        .cfi_def_cfa_register 6
        movl    $1, -12(%rbp)
        movl    $2, -8(%rbp)
        movl    -12(%rbp), %edx
        movl    -8(%rbp), %eax
        addl    %edx, %eax
        movl    %eax, -4(%rbp)
        nop
        popq    %rbp
        .cfi_def_cfa 7, 8
        ret
        .cfi_endproc
.LFE0:
        .size   func, .-func
        .globl  main
        .type   main, @function
main:
.LFB1:
        .cfi_startproc
        endbr64
        pushq   %rbp
        .cfi_def_cfa_offset 16
        .cfi_offset 6, -16
        movq    %rsp, %rbp
        .cfi_def_cfa_register 6
        movl    $0, %eax
        call    func
        movl    $0, %eax
        popq    %rbp
        .cfi_def_cfa 7, 8
        ret
```

b. No args, ret

```
int func()
{
        int x = 1, y = 2, z;
        z = x + y;
        return z;
}

int main()
{
        int a = func();
        print("%d", a);
}
```

```
func:
.LFB0:
        .cfi_startproc
        endbr32
        pushl   %ebp
        .cfi_def_cfa_offset 8
        .cfi_offset 5, -8
        movl    %esp, %ebp
        .cfi_def_cfa_register 5
        subl    $16, %esp
        call    __x86.get_pc_thunk.ax
        addl    $_GLOBAL_OFFSET_TABLE_, %eax
        movl    $1, -12(%ebp)
        movl    $2, -8(%ebp)
        movl    -12(%ebp), %edx
        movl    -8(%ebp), %eax
        addl    %edx, %eax
        movl    %eax, -4(%ebp)
        movl    -4(%ebp), %eax
        leave
        .cfi_restore 5
        .cfi_def_cfa 4, 4
        ret
        .cfi_endproc
.LFE0:
        .size   func, .-func
        .section        .rodata
.LC0:
        .string "%d"
        .text
        .globl  main
        .type   main, @function
main:
.LFB1:
        .cfi_startproc
        endbr32
        leal    4(%esp), %ecx
        .cfi_def_cfa 1, 0
        andl    $-16, %esp
        pushl   -4(%ecx)
        pushl   %ebp
        movl    %esp, %ebp
        .cfi_escape 0x10,0x5,0x2,0x75,0
        pushl   %ebx
        pushl   %ecx
        .cfi_escape 0xf,0x3,0x75,0x78,0x6
        .cfi_escape 0x10,0x3,0x2,0x75,0x7c
        subl    $16, %esp
        call    __x86.get_pc_thunk.bx
        addl    $_GLOBAL_OFFSET_TABLE_, %ebx
        call    func
        movl    %eax, -12(%ebp)
```

```
func:
.LFB0:
        .cfi_startproc
        endbr64
        pushq   %rbp
        .cfi_def_cfa_offset 16
        .cfi_offset 6, -16
        movq    %rsp, %rbp
        .cfi_def_cfa_register 6
        movl    $1, -12(%rbp)
        movl    $2, -8(%rbp)
        movl    -12(%rbp), %edx
        movl    -8(%rbp), %eax
        addl    %edx, %eax
        movl    %eax, -4(%rbp)
        movl    -4(%rbp), %eax
        popq    %rbp
        .cfi_def_cfa 7, 8
        ret
        .cfi_endproc
.LFE0:
        .size   func, .-func
        .section        .rodata
.LC0:
        .string "%d"
        .text
        .globl  main
        .type   main, @function
main:
.LFB1:
        .cfi_startproc
        endbr64
        pushq   %rbp
        .cfi_def_cfa_offset 16
        .cfi_offset 6, -16
        movq    %rsp, %rbp
        .cfi_def_cfa_register 6
        subq    $16, %rsp
        movl    $0, %eax
        call    func
        movl    %eax, -4(%rbp)
```

c.1 arg

```
int func(int d)
{
        int y = 2, z;
        z = d * y;
        return z;
}

int main()
{
        int x = 5;
        int a = func(x);
        printf("%d", a);
}
```

```
func:
.LFB0:
        .cfi_startproc
        endbr32
        pushl   %ebp
        .cfi_def_cfa_offset 8
        .cfi_offset 5, -8
        movl    %esp, %ebp
        .cfi_def_cfa_register 5
        subl    $16, %esp
        call    __x86.get_pc_thunk.ax
        addl    $_GLOBAL_OFFSET_TABLE_,
        movl    $2, -8(%ebp)
        movl    8(%ebp), %eax
        imull   -8(%ebp), %eax
        movl    %eax, -4(%ebp)
        movl    -4(%ebp), %eax
        leave
        .cfi_restore 5
        .cfi_def_cfa 4, 4
        ret
        .cfi_endproc
.LFE0:
        .size   func, .-func
        .section        .rodata
.LC0:
        .string "%d"
        .text
        .globl  main
        .type   main, @function
main:
.LFB1:
        .cfi_startproc
        endbr32
        leal    4(%esp), %ecx
        .cfi_def_cfa 1, 0
        andl    $-16, %esp
        pushl   -4(%ecx)
        pushl   %ebp
        movl    %esp, %ebp
        .cfi_escape 0x10,0x5,0x2,0x75,0
        pushl   %ebx
        pushl   %ecx
        .cfi_escape 0xf,0x3,0x75,0x78,0
        .cfi_escape 0x10,0x3,0x2,0x75,0
        subl    $16, %esp
        call    __x86.get_pc_thunk.bx
        addl    $_GLOBAL_OFFSET_TABLE_,
        movl    $5, -16(%ebp)
        pushl   -16(%ebp)
        call    func
```

```
func:
.LFB0:
        .cfi_startproc
        endbr64
        pushq   %rbp
        .cfi_def_cfa_offset 16
        .cfi_offset 6, -16
        movq    %rsp, %rbp
        .cfi_def_cfa_register 6
        movl    %edi, -20(%rbp)
        movl    $2, -8(%rbp)
        movl    -20(%rbp), %eax
        imull   -8(%rbp), %eax
        movl    %eax, -4(%rbp)
        movl    -4(%rbp), %eax
        popq    %rbp
        .cfi_def_cfa 7, 8
        ret
        .cfi_endproc
.LFE0:
        .size   func, .-func
        .section        .rodata
.LC0:
        .string "%d"
        .text
        .globl  main
        .type   main, @function
main:
.LFB1:
        .cfi_startproc
        endbr64
        pushq   %rbp
        .cfi_def_cfa_offset 16
        .cfi_offset 6, -16
        movq    %rsp, %rbp
        .cfi_def_cfa_register 6
        subq    $16, %rsp
        movl    $5, -8(%rbp)
        movl    -8(%rbp), %eax
        movl    %eax, %edi
        call    func
        movl    %eax, -4(%rbp)
```

d. many args

```c
int func(int a, int b, int c)
{
        int res;
        res = a * b + c;
        return res;
}

int main()
{
        int x = 5, y = 3, z = 7;
        int a = func(x, y, z);
        printf("%d", a);
}
```

```
func:
.LFB0:
        .cfi_startproc
        endbr32
        pushl   %ebp
        .cfi_def_cfa_offset 8
        .cfi_offset 5, -8
        movl    %esp, %ebp
        .cfi_def_cfa_register 5
        subl    $16, %esp
        call    __x86.get_pc_thunk.ax
        addl    $_GLOBAL_OFFSET_TABLE_, %eax
        movl    8(%ebp), %eax
        imull   12(%ebp), %eax
        movl    %eax, %edx
        movl    16(%ebp), %eax
        addl    %edx, %eax
        movl    %eax, -4(%ebp)
        movl    -4(%ebp), %eax
        leave
        .cfi_restore 5
        .cfi_def_cfa 4, 4
        ret
```

```
main:
.LFB1:
        .cfi_startproc
        endbr32
        leal    4(%esp), %ecx
        .cfi_def_cfa 1, 0
        andl    $-16, %esp
        pushl   -4(%ecx)
        pushl   %ebp
        movl    %esp, %ebp
        .cfi_escape 0x10,0x5,0x2,0x75,0
        pushl   %ebx
        pushl   %ecx
        .cfi_escape 0xf,0x3,0x75,0x78,0x6
        .cfi_escape 0x10,0x3,0x2,0x75,0x7c
        subl    $16, %esp
        call    __x86.get_pc_thunk.bx
        addl    $_GLOBAL_OFFSET_TABLE_, %ebx
        movl    $5, -24(%ebp)
        movl    $3, -20(%ebp)
        movl    $7, -16(%ebp)
        pushl   -16(%ebp)
        pushl   -20(%ebp)
        pushl   -24(%ebp)
        call    func
        addl    $12, %esp
        movl    %eax, -12(%ebp)
```

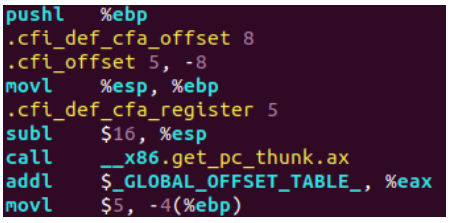
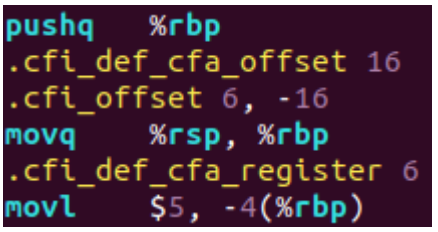
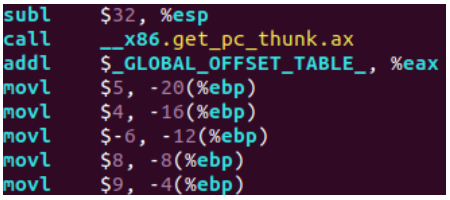
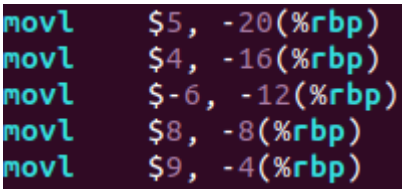
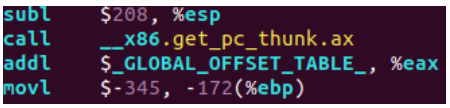
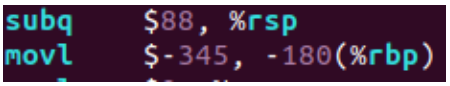
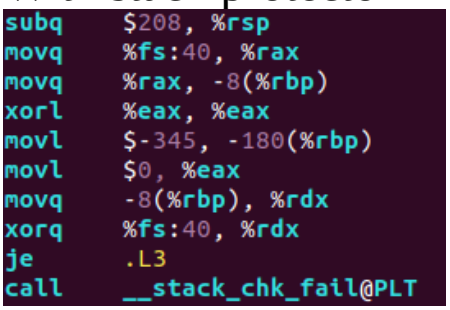
```
func:
.LFB0:
        .cfi_startproc
        endbr64
        pushq   %rbp
        .cfi_def_cfa_offset 16
        .cfi_offset 6, -16
        movq    %rsp, %rbp
        .cfi_def_cfa_register 6
        movl    %edi, -20(%rbp)
        movl    %esi, -24(%rbp)
        movl    %edx, -28(%rbp)
        movl    -20(%rbp), %eax
        imull   -24(%rbp), %eax
        movl    %eax, %edx
        movl    -28(%rbp), %eax
        addl    %edx, %eax
        movl    %eax, -4(%rbp)
        movl    -4(%rbp), %eax
        popq    %rbp
        .cfi_def_cfa 7, 8
        ret
main:
.LFB1:
        .cfi_startproc
        endbr64
        pushq   %rbp
        .cfi_def_cfa_offset 16
        .cfi_offset 6, -16
        movq    %rsp, %rbp
        .cfi_def_cfa_register 6
        subq    $16, %rsp
        movl    $5, -16(%rbp)
        movl    $3, -12(%rbp)
        movl    $7, -8(%rbp)
        movl    -8(%rbp), %edx
        movl    -12(%rbp), %ecx
        movl    -16(%rbp), %eax
        movl    %ecx, %esi
        movl    %eax, %edi
        call    func
        movl    %eax, -4(%rbp)
```

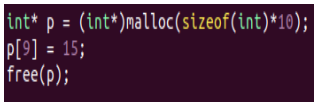
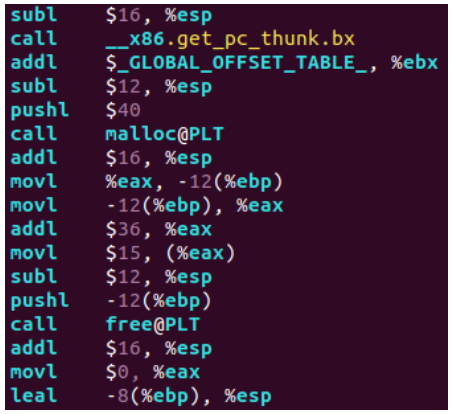
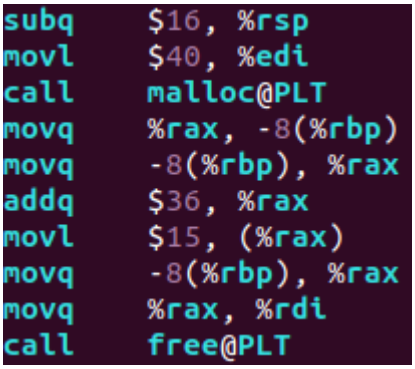
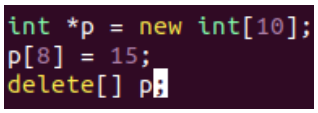
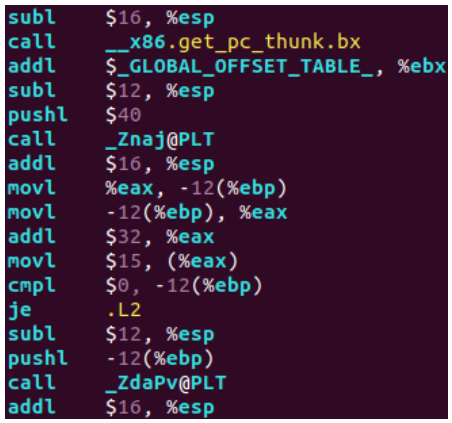
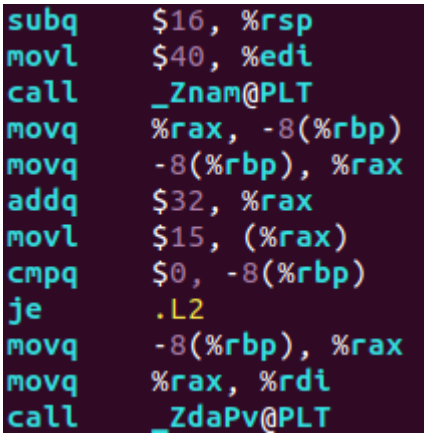
General observations:

- register %eax is used as a return value
- 32bit and 64bit systems operate the setup and cleanup of stack frame differently(i.g. 32bit uses leave(equals movl + popl), 64bit operate with just popq
- another feature of 64bit system is that, unlike 32bit system, which allocate memory for local variables using, i.g., subl $16, %esp in both main and leaf function, there is a “red zone” of 128 bytes below %rsp. These 128 bytes belong to

the function as long as it’s a leaf function. Thus, all of local variables of a leaf function fit into the red zone, so no adjustment of %rsp needed (no instructions such as subq $16, %rsp).

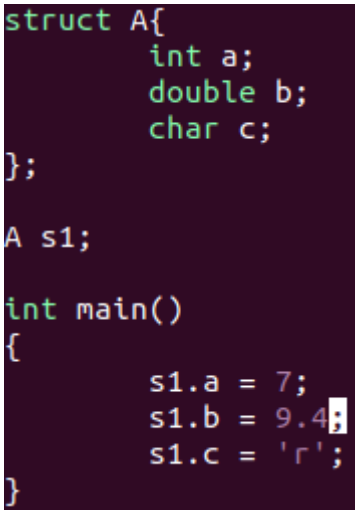
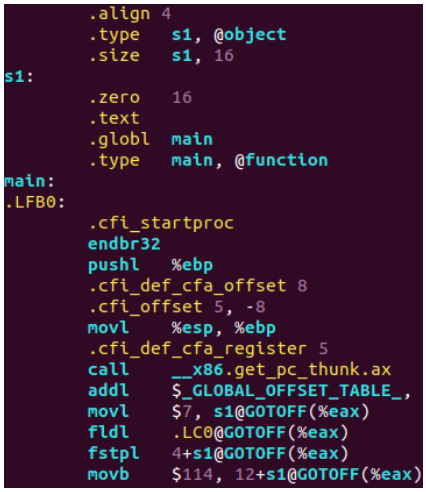
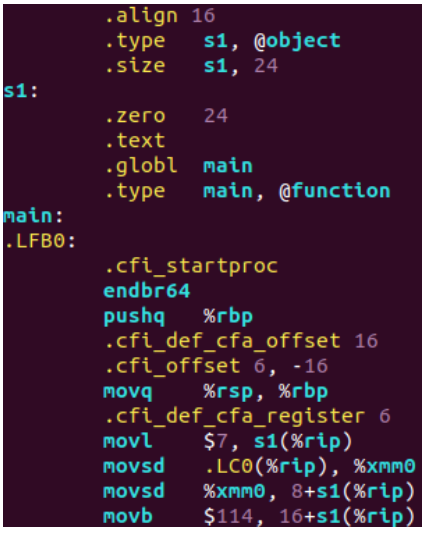
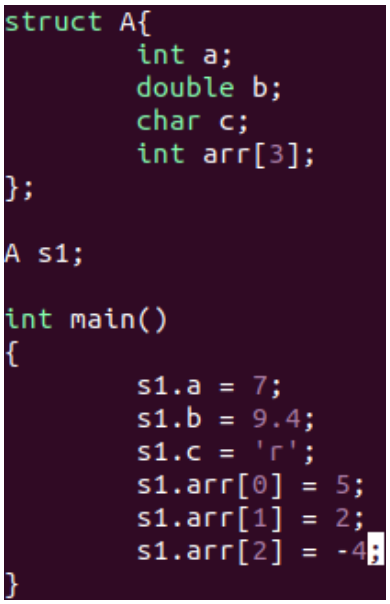
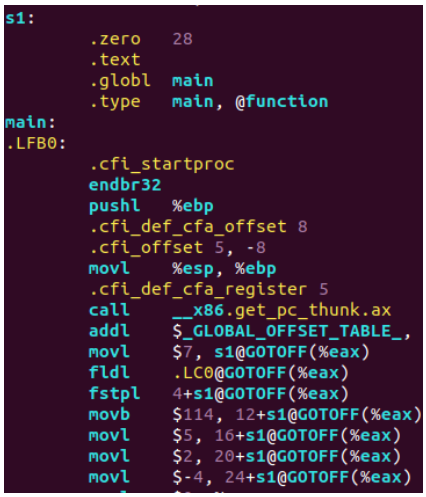
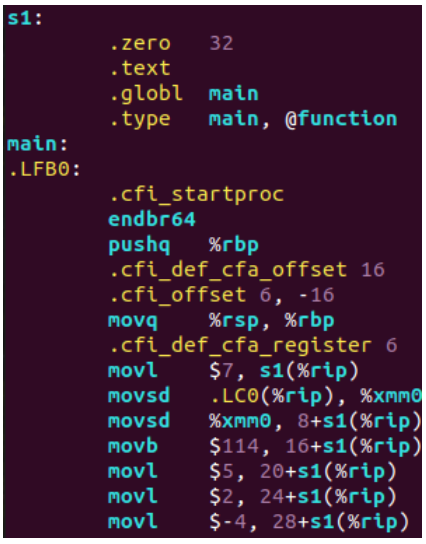
2. Local variables.

point	C	32bit	64bit
a. 1 loc var	int x = 5		
b. 5 loc var	int a = 5, b = 4, c = -6, d = 8, i = 9;		
c. static array	int arr[50]; arr[7] = -345;		Without stack protector  With stack protector 

d. dynamic array(C)			
d. dynamic array(C++)			 _Znam@PLT = new[] _ZdaPV@PLT = delete[]

- static array in 64bit shift %rsp to a less amount, than the size of array, perhaps, because of the red zone.
- In 32bit system size of dynamic array is pushed in stack; in 64bit is moved to %edi
- %eax/%rax initially refers to the first element of dynamic array(it must be so because after we call malloc/new[], the result of function, i.e. pointer to the first element, is written in %eax)

3. Structures

point	C++	32bit	64bit
b. global struct			
c. static array as a member			

d. struct as an arg in function

```
struct A{
        int a;
        double b;
        char c;
        int arr[3];
};

void func(A &s)
{
        s.a = 2;
        s.c = 'q';
        s.arr[2] = 90;
}

int main()
{
        A s1;
        s1.c = 't';
        func(s1);
}
```

```
        .globl  _Z4funcR1A
        .type   _Z4funcR1A, @function
_Z4funcR1A:
.LFB0:
        .cfi_startproc
        endbr32
        pushl   %ebp
        .cfi_def_cfa_offset 8
        .cfi_offset 5, -8
        movl    %esp, %ebp
        .cfi_def_cfa_register 5
        call    __x86.get_pc_thunk.ax
        addl    $_GLOBAL_OFFSET_TABLE_, %eax
        movl    8(%ebp), %eax
        movl    $2, (%eax)
        movl    8(%ebp), %eax
        movb    $113, 12(%eax)
        movl    8(%ebp), %eax
        movl    $90, 24(%eax)
        nop
        popl    %ebp
        .cfi_restore 5
        .cfi_def_cfa 4, 4
        ret
        .cfi_endproc
.LFE0:
        .size   _Z4funcR1A, .-_Z4funcR1A
        .globl  main
        .type   main, @function
main:
.LFB1:
        .cfi_startproc
        endbr32
        pushl   %ebp
        .cfi_def_cfa_offset 8
        .cfi_offset 5, -8
        movl    %esp, %ebp
        .cfi_def_cfa_register 5
        subl    $32, %esp
        call    __x86.get_pc_thunk.ax
        addl    $_GLOBAL_OFFSET_TABLE_, %eax
        movb    $116, -16(%ebp)
        leal    -28(%ebp), %eax
        pushl   %eax
        call    _Z4funcR1A
        addl    $4, %esp
```

```
        .globl  _Z4funcR1A
        .type   _Z4funcR1A, @function
_Z4funcR1A:
.LFB0:
        .cfi_startproc
        endbr64
        pushq   %rbp
        .cfi_def_cfa_offset 16
        .cfi_offset 6, -16
        movq    %rsp, %rbp
        .cfi_def_cfa_register 6
        movq    %rdi, -8(%rbp)
        movq    -8(%rbp), %rax
        movl    $2, (%rax)
        movq    -8(%rbp), %rax
        movb    $113, 16(%rax)
        movq    -8(%rbp), %rax
        movl    $90, 28(%rax)
        nop
        popq    %rbp
        .cfi_def_cfa 7, 8
        ret
        .cfi_endproc
.LFE0:
        .size   _Z4funcR1A, .-_Z4funcR1A
        .globl  main
        .type   main, @function
main:
.LFB1:
        .cfi_startproc
        endbr64
        pushq   %rbp
        .cfi_def_cfa_offset 16
        .cfi_offset 6, -16
        movq    %rsp, %rbp
        .cfi_def_cfa_register 6
        subq    $32, %rsp
        movb    $116, -16(%rbp)
        leaq    -32(%rbp), %rax
        movq    %rax, %rdi
        call    _Z4funcR1A
```

e. struct as a return value

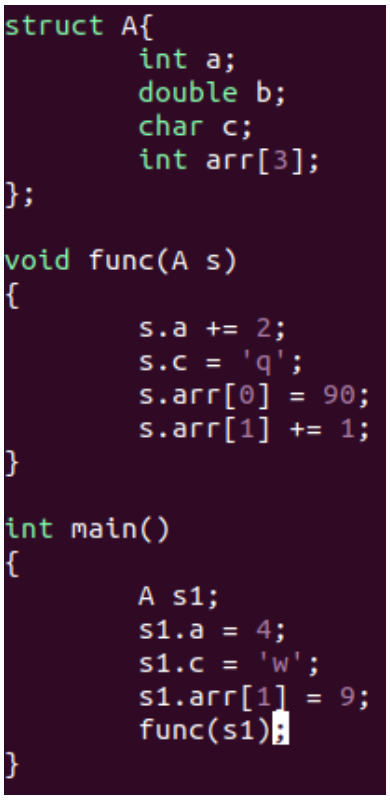
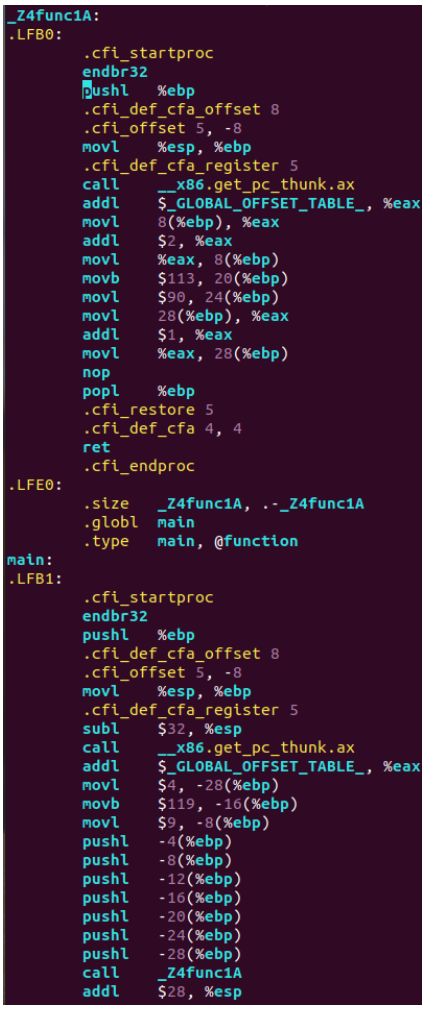
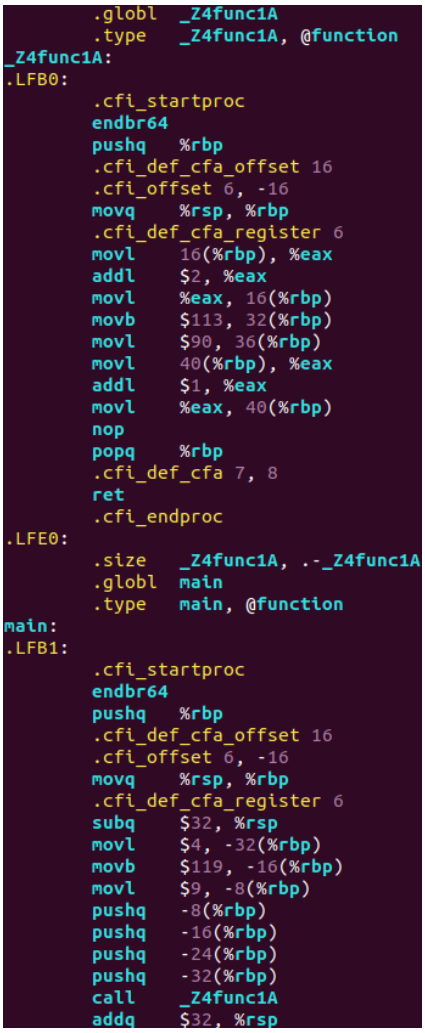
```
struct A{
        int a;
        double b;
        char c;
        int arr[3];
};

A func()
{
        A s;
        s.a = 2;
        s.c = 'q';
        s.arr[2] = 90;
        return s;
}

int main()
{
        A s1 = func();
}
```

```
        .globl  _Z4funcv
        .type   _Z4funcv, @function
_Z4funcv:
.LFB0:
        .cfi_startproc
        endbr32
        pushl   %ebp
        .cfi_def_cfa_offset 8
        .cfi_offset 5, -8
        movl    %esp, %ebp
        .cfi_def_cfa_register 5
        call    __x86.get_pc_thunk.ax
        addl    $_GLOBAL_OFFSET_TABLE_, %eax
        movl    8(%ebp), %eax
        movl    $2, (%eax)
        movl    8(%ebp), %eax
        movb    $113, 12(%eax)
        movl    8(%ebp), %eax
        movl    $90, 24(%eax)
        nop
        movl    8(%ebp), %eax
        popl    %ebp
        .cfi_restore 5
        .cfi_def_cfa 4, 4
        ret     $4
        .cfi_endproc
.LFE0:
        .size   _Z4funcv, .-_Z4funcv
        .globl  main
        .type   main, @function
main:
.LFB1:
        .cfi_startproc
        endbr32
        pushl   %ebp
        .cfi_def_cfa_offset 8
        .cfi_offset 5, -8
        movl    %esp, %ebp
        .cfi_def_cfa_register 5
        subl    $32, %esp
        call    __x86.get_pc_thunk.ax
        addl    $_GLOBAL_OFFSET_TABLE_, %eax
        leal    -28(%ebp), %eax
        pushl   %eax
        call    _Z4funcv
```

```
        .globl  _Z4funcv
        .type   _Z4funcv, @function
_Z4funcv:
.LFB0:
        .cfi_startproc
        endbr64
        pushq   %rbp
        .cfi_def_cfa_offset 16
        .cfi_offset 6, -16
        movq    %rsp, %rbp
        .cfi_def_cfa_register 6
        movq    %rdi, -8(%rbp)
        movq    -8(%rbp), %rax
        movl    $2, (%rax)
        movq    -8(%rbp), %rax
        movb    $113, 16(%rax)
        movq    -8(%rbp), %rax
        movl    $90, 28(%rax)
        nop
        movq    -8(%rbp), %rax
        popq    %rbp
        .cfi_def_cfa 7, 8
        ret
        .cfi_endproc
.LFE0:
        .size   _Z4funcv, .-_Z4funcv
        .globl  main
        .type   main, @function
main:
.LFB1:
        .cfi_startproc
        endbr64
        pushq   %rbp
        .cfi_def_cfa_offset 16
        .cfi_offset 6, -16
        movq    %rsp, %rbp
        .cfi_def_cfa_register 6
        subq    $32, %rsp
        leaq    -32(%rbp), %rax
        movq    %rax, %rdi
        call    _Z4funcv
```

- in 32bit system Global offset table is used to dynamically access structures(in this task) . In a. and b. @GOTOFF is used for global structures – perhaps it stands for GOT Offset, meaning making an offset from GOT address. In 64bit system register %rip was responsible for that.

4. Pointers and references

point	C++	32bit	64bit
a. struct			

```
struct A{
        int a;
        double b;
        char c;
        int arr[3];
};

void func(A s)
{
        s.a += 2;
        s.c = 'q';
        s.arr[0] = 90;
        s.arr[1] += 1;
}

int main()
{
        A s1;
        s1.a = 4;
        s1.c = 'w';
        s1.arr[1] = 9;
        func(s1);
}
```

```
_Z4func1A:
.LFB0:
        .cfi_startproc
        endbr32
        pushl   %ebp
        .cfi_def_cfa_offset 8
        .cfi_offset 5, -8
        movl    %esp, %ebp
        .cfi_def_cfa_register 5
        call    __x86.get_pc_thunk.ax
        addl    $_GLOBAL_OFFSET_TABLE_, %eax
        movl    8(%ebp), %eax
        addl    $2, %eax
        movl    %eax, 8(%ebp)
        movb    $113, 20(%ebp)
        movl    $90, 24(%ebp)
        movl    28(%ebp), %eax
        addl    $1, %eax
        movl    %eax, 28(%ebp)
        nop
        popl    %ebp
        .cfi_restore 5
        .cfi_def_cfa 4, 4
        ret
        .cfi_endproc
.LFE0:
        .size   _Z4func1A, .-_Z4func1A
        .globl  main
        .type   main, @function
main:
.LFB1:
        .cfi_startproc
        endbr32
        pushl   %ebp
        .cfi_def_cfa_offset 8
        .cfi_offset 5, -8
        movl    %esp, %ebp
        .cfi_def_cfa_register 5
        subl    $32, %esp
        call    __x86.get_pc_thunk.ax
        addl    $_GLOBAL_OFFSET_TABLE_, %eax
        movl    $4, -28(%ebp)
        movb    $119, -16(%ebp)
        movl    $9, -8(%ebp)
        pushl   -4(%ebp)
        pushl   -8(%ebp)
        pushl   -12(%ebp)
        pushl   -16(%ebp)
        pushl   -20(%ebp)
        pushl   -24(%ebp)
        pushl   -28(%ebp)
        call    _Z4func1A
        addl    $28, %esp
```

```
        .globl  _Z4func1A
        .type   _Z4func1A, @function
_Z4func1A:
.LFB0:
        .cfi_startproc
        endbr64
        pushq   %rbp
        .cfi_def_cfa_offset 16
        .cfi_offset 6, -16
        movq    %rsp, %rbp
        .cfi_def_cfa_register 6
        movl    16(%rbp), %eax
        addl    $2, %eax
        movl    %eax, 16(%rbp)
        movb    $113, 32(%rbp)
        movl    $90, 36(%rbp)
        movl    40(%rbp), %eax
        addl    $1, %eax
        movl    %eax, 40(%rbp)
        nop
        popq    %rbp
        .cfi_def_cfa 7, 8
        ret
        .cfi_endproc
.LFE0:
        .size   _Z4func1A, .-_Z4func1A
        .globl  main
        .type   main, @function
main:
.LFB1:
        .cfi_startproc
        endbr64
        pushq   %rbp
        .cfi_def_cfa_offset 16
        .cfi_offset 6, -16
        movq    %rsp, %rbp
        .cfi_def_cfa_register 6
        subq    $32, %rsp
        movl    $4, -32(%rbp)
        movb    $119, -16(%rbp)
        movl    $9, -8(%rbp)
        pushq   -8(%rbp)
        pushq   -16(%rbp)
        pushq   -24(%rbp)
        pushq   -32(%rbp)
        call    _Z4func1A
        addq    $32, %rsp
```

b. struct pointer

```
struct A{
        int a;
        double b;
        char c;
        int arr[3];
};

void func(A *s)
{
        s->a += 2;
        s->c = 'q';
        s->arr[0] = 90;
        s->arr[1] += 1;
}

int main()
{
        A s1;
        s1.a = 4;
        s1.c = 'w';
        s1.arr[1] = 9;
        func(&s1);
        printf("%d", s1.a);
}
```

```
_Z4funcP1A:
.LFB0:
        .cfi_startproc
        endbr32
        pushl   %ebp
        .cfi_def_cfa_offset 8
        .cfi_offset 5, -8
        movl    %esp, %ebp
        .cfi_def_cfa_register 5
        call    __x86.get_pc_thunk.ax
        addl    $_GLOBAL_OFFSET_TABLE_, %eax
        movl    8(%ebp), %eax
        movl    (%eax), %eax
        leal    2(%eax), %edx
        movl    8(%ebp), %eax
        movl    %edx, (%eax)
        movl    8(%ebp), %eax
        movb    $113, 12(%eax)
        movl    8(%ebp), %eax
        movl    $90, 16(%eax)
        movl    8(%ebp), %eax
        movl    20(%eax), %eax
        leal    1(%eax), %edx
        movl    8(%ebp), %eax
        movl    %edx, 20(%eax)
        nop
        popl    %ebp
        .cfi_restore 5
        .cfi_def_cfa 4, 4
        ret
        .cfi_endproc
.LFE0:
        .size   _Z4funcP1A, .-_Z4funcP1A
        .section        .rodata
.LC0:
        .string "%d"
        .text
        .globl  main
        .type   main, @function
main:
.LFB1:
        .cfi_startproc
        endbr32
        leal    4(%esp), %ecx
        .cfi_def_cfa 1, 0
        andl    $-16, %esp
        pushl   -4(%ecx)
        pushl   %ebp
        movl    %esp, %ebp
        .cfi_escape 0x10,0x5,0x2,0x75,0
        pushl   %ebx
        pushl   %ecx
```

```
        subl    $32, %esp
        call    __x86.get_pc_thunk.bx
        addl    $_GLOBAL_OFFSET_TABLE_, %ebx
        movl    $4, -36(%ebp)
        movb    $119, -24(%ebp)
        movl    $9, -16(%ebp)
        leal    -36(%ebp), %eax
        pushl   %eax
        call    _Z4funcP1A
        addl    $4, %esp
        movl    -36(%ebp), %eax
        subl    $8, %esp
```

```
_Z4funcP1A:
.LFB0:
        .cfi_startproc
        endbr64
        pushq   %rbp
        .cfi_def_cfa_offset 16
        .cfi_offset 6, -16
        movq    %rsp, %rbp
        .cfi_def_cfa_register 6
        movq    %rdi, -8(%rbp)
        movq    -8(%rbp), %rax
        movl    (%rax), %eax
        leal    2(%rax), %edx
        movq    -8(%rbp), %rax
        movl    %edx, (%rax)
        movq    -8(%rbp), %rax
        movb    $113, 16(%rax)
        movq    -8(%rbp), %rax
        movl    $90, 20(%rax)
        movq    -8(%rbp), %rax
        movl    24(%rax), %eax
        leal    1(%rax), %edx
        movq    -8(%rbp), %rax
        movl    %edx, 24(%rax)
        nop
        popq    %rbp
        .cfi_def_cfa 7, 8
        ret
        .cfi_endproc
.LFE0:
        .size   _Z4funcP1A, .-_Z4funcP1A
        .section        .rodata
.LC0:
        .string "%d"
        .text
        .globl  main
        .type   main, @function
main:
.LFB1:
        .cfi_startproc
        endbr64
        pushq   %rbp
        .cfi_def_cfa_offset 16
        .cfi_offset 6, -16
        movq    %rsp, %rbp
        .cfi_def_cfa_register 6
        subq    $32, %rsp
        movl    $4, -32(%rbp)
        movb    $119, -16(%rbp)
        movl    $9, -8(%rbp)
        leaq    -32(%rbp), %rax
        movq    %rax, %rdi
        call    _Z4funcP1A
        movl    -32(%rbp), %eax
```

c. struct reference

```
struct A{
        int a;
        double b;
        char c;
        int arr[3];
};

void func(A &s)
{
        s.a += 2;
        s.c = 'q';
        s.arr[0] = 90;
        s.arr[1] += 1;
}

int main()
{
        A s1;
        s1.a = 4;
        s1.c = 'w';
        s1.arr[1] = 9;
        func(s1);
        printf("%d", s1.a);
}
```

```
_Z4funcR1A:
.LFB0:
        .cfi_startproc
        endbr32
        pushl   %ebp
        .cfi_def_cfa_offset 8
        .cfi_offset 5, -8
        movl    %esp, %ebp
        .cfi_def_cfa_register 5
        call    __x86.get_pc_thunk.ax
        addl    $_GLOBAL_OFFSET_TABLE_, %eax
        movl    8(%ebp), %eax
        movl    (%eax), %eax
        leal    2(%eax), %edx
        movl    8(%ebp), %eax
        movl    %edx, (%eax)
        movl    8(%ebp), %eax
        movb    $113, 12(%eax)
        movl    8(%ebp), %eax
        movl    $90, 16(%eax)
        movl    8(%ebp), %eax
        movl    20(%eax), %eax
        leal    1(%eax), %edx
        movl    8(%ebp), %eax
        movl    %edx, 20(%eax)
        nop
        popl    %ebp
        .cfi_restore 5
        .cfi_def_cfa 4, 4
        ret
        .cfi_endproc
.LFE0:
        .size   _Z4funcR1A, .-_Z4funcR1A
        .section        .rodata
.LC0:
        .string "%d"
        .text
        .globl  main
        .type   main, @function
main:
.LFB1:
        .cfi_startproc
        endbr32
        leal    4(%esp), %ecx
        .cfi_def_cfa 1, 0
        andl    $-16, %esp
        pushl   -4(%ecx)
        pushl   %ebp
        movl    %esp, %ebp
        .cfi_escape 0x10,0x5,0x2,0x75,0
        pushl   %ebx
        pushl   %ecx
```

```
        subl    $32, %esp
        call    __x86.get_pc_thunk.bx
        addl    $_GLOBAL_OFFSET_TABLE_, %ebx
        movl    $4, -36(%ebp)
        movb    $119, -24(%ebp)
        movl    $9, -16(%ebp)
        leal    -36(%ebp), %eax
        pushl   %eax
        call    _Z4funcR1A
        addl    $4, %esp
        movl    -36(%ebp), %eax
        subl    $8, %esp
```

```
_Z4funcR1A:
.LFB0:
        .cfi_startproc
        endbr64
        pushq   %rbp
        .cfi_def_cfa_offset 16
        .cfi_offset 6, -16
        movq    %rsp, %rbp
        .cfi_def_cfa_register 6
        movq    %rdi, -8(%rbp)
        movq    -8(%rbp), %rax
        movl    (%rax), %eax
        leal    2(%rax), %edx
        movq    -8(%rbp), %rax
        movl    %edx, (%rax)
        movq    -8(%rbp), %rax
        movb    $113, 16(%rax)
        movq    -8(%rbp), %rax
        movl    $90, 20(%rax)
        movq    -8(%rbp), %rax
        movl    24(%rax), %eax
        leal    1(%rax), %edx
        movq    -8(%rbp), %rax
        movl    %edx, 24(%rax)
        nop
        popq    %rbp
        .cfi_def_cfa 7, 8
        ret
        .cfi_endproc
.LFE0:
        .size   _Z4funcR1A, .-_Z4funcR1A
        .section        .rodata
.LC0:
        .string "%d"
        .text
        .globl  main
        .type   main, @function
main:
.LFB1:
        .cfi_startproc
        endbr64
        pushq   %rbp
        .cfi_def_cfa_offset 16
        .cfi_offset 6, -16
        movq    %rsp, %rbp
        .cfi_def_cfa_register 6
        subq    $32, %rsp
        movl    $4, -32(%rbp)
        movb    $119, -16(%rbp)
        movl    $9, -8(%rbp)
        leaq    -32(%rbp), %rax
        movq    %rax, %rdi
        call    _Z4funcR1A
```

- in a. all struct members are pushed into stack; in b. and c. (pointers and references) struct members are not pushed into stack, only registers and stack frame(via i(%rbp)) are used.

5. Heavy structures

a. struct as an arg

```cpp
const int n = 10000000;
struct A{
        int arr1[n];
        int arr2[n];
};

void func(A s)
{
        for(int i = 0; i < n; i ++)
        {
                s.arr1[i] = i - 1;
                s.arr2[i] = i;
        }
}

int main()
{
        A s1;
        func(s1);
}
```

```asm
_Z4func1A:
.LFB0:
        .cfi_startproc
        endbr32
        pushl   %ebp
        .cfi_def_cfa_offset 8
        .cfi_offset 5, -8
        movl    %esp, %ebp
        .cfi_def_cfa_register 5
        subl    $16, %esp
        call    __x86.get_pc_thunk.ax
        addl    $_GLOBAL_OFFSET_TABLE_, %eax
        movl    $0, -4(%ebp)
.L3:
        cmpl    $9999999, -4(%ebp)
        jg      .L4
        movl    -4(%ebp), %eax
        leal    -1(%eax), %edx
        movl    -4(%ebp), %eax
        movl    %edx, 8(%ebp,%eax,4)
        movl    -4(%ebp), %eax
        leal    10000000(%eax), %edx
        movl    -4(%ebp), %eax
        movl    %eax, 8(%ebp,%edx,4)
        addl    $1, -4(%ebp)
        jmp     .L3
.L4:
        nop
        leave
        .cfi_restore 5
        .cfi_def_cfa 4, 4
        ret
        .cfi_endproc
.LFE0:
        .size   _Z4func1A, .-_Z4func1A
        .globl  main
        .type   main, @function
main:
.LFB1:
        .cfi_startproc
        endbr32
        leal    4(%esp), %ecx
        .cfi_def_cfa 1, 0
        andl    $-16, %esp
        pushl   -4(%ecx)
        pushl   %ebp
        movl    %esp, %ebp
        .cfi_escape 0x10,0x5,0x2,0x75,0
        pushl   %ebx
        pushl   %ecx
        leal    -79998976(%esp), %eax
.LPSRL0:
        subl    $4096, %esp
        orl     $0, (%esp)
        cmpl    %eax, %esp
        jne     .LPSRL0
        subl    $1024, %esp
        call    __x86.get_pc_thunk.ax
        addl    $_GLOBAL_OFFSET_TABLE_, %eax
        subl    $80000000, %esp
        movl    %esp, %edx
        movl    %edx, %ecx
        leal    -80000008(%ebp), %edx
        movl    $80000000, %ebx
        subl    $4, %esp
        pushl   %ebx
        pushl   %edx
        pushl   %ecx
        movl    %eax, %ebx
        call    memcpy@PLT
        addl    $16, %esp
        call    _Z4func1A
        addl    $80000000, %esp
```

```asm
_Z4func1A:
.LFB0:
        .cfi_startproc
        endbr64
        pushq   %rbp
        .cfi_def_cfa_offset 16
        .cfi_offset 6, -16
        movq    %rsp, %rbp
        .cfi_def_cfa_register 6
        movl    $0, -4(%rbp)
.L3:
        cmpl    $9999999, -4(%rbp)
        jg      .L4
        movl    -4(%rbp), %eax
        leal    -1(%rax), %edx
        movl    -4(%rbp), %eax
        cltq
        movl    %edx, 16(%rbp,%rax,4)
        movl    -4(%rbp), %eax
        cltq
        leaq    10000000(%rax), %rdx
        movl    -4(%rbp), %eax
        movl    %eax, 16(%rbp,%rdx,4)
        addl    $1, -4(%rbp)
        jmp     .L3
.L4:
        nop
        popq    %rbp
        .cfi_def_cfa 7, 8
        ret
        .cfi_endproc
.LFE0:
        .size   _Z4func1A, .-_Z4func1A
        .globl  main
        .type   main, @function
main:
.LFB1:
        .cfi_startproc
        endbr64
        pushq   %rbp
        .cfi_def_cfa_offset 16
        .cfi_offset 6, -16
        movq    %rsp, %rbp
        .cfi_def_cfa_register 6
        leaq    -79998976(%rsp), %r11
.LPSRL0:
        subq    $4096, %rsp
        orq     $0, (%rsp)
        cmpq    %r11, %rsp
        jne     .LPSRL0
        subq    $1024, %rsp
        subq    $80000000, %rsp
        movq    %rsp, %rax
        movq    %rax, %rcx
        leaq    -80000000(%rbp), %rax
        movl    $80000000, %edx
        movq    %rax, %rsi
        movq    %rcx, %rdi
        call    memcpy@PLT
        call    _Z4func1A
        addq    $80000000, %rsp
```

b. struct as a return value

```
const int n = 100000;
struct A{
        int arr1[n];
        int arr2[n];
};

A func()
{
        A s;
        for(int i = 0; i < n; i ++)
        {
                s.arr1[i] = i - 1;
                s.arr2[i] = i;
        }
        return s;
}

int main()
{
        A s1;
        s1 = func();
        printf("%d", s1.arr2[1090]);
}
```

```
_Z4funcv:
.LFB0:
        .cfi_startproc
        endbr32
        pushl   %ebp
        .cfi_def_cfa_offset 8
        .cfi_offset 5, -8
        movl    %esp, %ebp
        .cfi_def_cfa_register 5
        subl    $16, %esp
        call    __x86.get_pc_thunk.ax
        addl    $_GLOBAL_OFFSET_TABLE_, %eax
        movl    $0, -4(%ebp)
.L3:
        cmpl    $99999, -4(%ebp)
        jg      .L5
        movl    -4(%ebp), %eax
        leal    -1(%eax), %ecx
        movl    8(%ebp), %eax
        movl    -4(%ebp), %edx
        movl    %ecx, (%eax,%edx,4)
        movl    8(%ebp), %eax
        movl    -4(%ebp), %edx
        leal    100000(%edx), %ecx
        movl    -4(%ebp), %edx
        movl    %edx, (%eax,%ecx,4)
        addl    $1, -4(%ebp)
        jmp     .L3
.L5:
        nop
        movl    8(%ebp), %eax
        leave
        .cfi_restore 5
        .cfi_def_cfa 4, 4
        ret     $4
```

```
main:
.LFB1:
        .cfi_startproc
        endbr32
        leal    4(%esp), %ecx
        .cfi_def_cfa 1, 0
        andl    $-16, %esp
        pushl   -4(%ecx)
        pushl   %ebp
        movl    %esp, %ebp
        .cfi_escape 0x10,0x5,0x2,0x75,0
        pushl   %ebx
        pushl   %ecx
        .cfi_escape 0xf,0x3,0x75,0x78,0x6
        .cfi_escape 0x10,0x3,0x2,0x75,0x7c
        leal    -1597440(%esp), %eax
.LPSRL0:
        subl    $4096, %esp
        orl     $0, (%esp)
        cmpl    %eax, %esp
        jne     .LPSRL0
        subl    $2560, %esp
        call    __x86.get_pc_thunk.bx
        addl    $_GLOBAL_OFFSET_TABLE_, %ebx
        leal    -1600008(%ebp), %eax
        pushl   %eax
        call    _Z4funcv
        leal    -800008(%ebp), %eax
        leal    -1600008(%ebp), %edx
        movl    $800000, %ecx
        subl    $4, %esp
        pushl   %ecx
        pushl   %edx
        pushl   %eax
        call    memcpy@PLT
        addl    $16, %esp
        movl    -395648(%ebp), %eax
```

```
_Z4funcv:
.LFB0:
        .cfi_startproc
        endbr64
        pushq   %rbp
        .cfi_def_cfa_offset 16
        .cfi_offset 6, -16
        movq    %rsp, %rbp
        .cfi_def_cfa_register 6
        movq    %rdi, -24(%rbp)
        movl    $0, -4(%rbp)
.L3:
        cmpl    $99999, -4(%rbp)
        jg      .L5
        movl    -4(%rbp), %eax
        leal    -1(%rax), %ecx
        movq    -24(%rbp), %rax
        movl    -4(%rbp), %edx
        movslq  %edx, %rdx
        movl    %ecx, (%rax,%rdx,4)
        movq    -24(%rbp), %rax
        movl    -4(%rbp), %edx
        movslq  %edx, %rdx
        leaq    100000(%rdx), %rcx
        movl    -4(%rbp), %edx
        movl    %edx, (%rax,%rcx,4)
        addl    $1, -4(%rbp)
        jmp     .L3
.L5:
        nop
        movq    -24(%rbp), %rax
        popq    %rbp
        .cfi_def_cfa 7, 8
        ret
        .cfi_endproc
.LFE0:
        .size   _Z4funcv, .-_Z4funcv
        .section        .rodata
.LC0:
        .string "%d"
        .text
        .globl  main
        .type   main, @function
main:
.LFB1:
        .cfi_startproc
        endbr64
        pushq   %rbp
        .cfi_def_cfa_offset 16
        .cfi_offset 6, -16
        movq    %rsp, %rbp
        .cfi_def_cfa_register 6
        leaq    -1597440(%rsp), %r11
```

```
        subq    $4096, %rsp
        orq     $0, (%rsp)
        cmpq    %r11, %rsp
        jne     .LPSRL0
        subq    $2560, %rsp
        leaq    -1600000(%rbp), %rax
        movq    %rax, %rdi
        call    _Z4funcv
        leaq    -800000(%rbp), %rax
        leaq    -1600000(%rbp), %rcx
        movl    $800000, %edx
        movq    %rcx, %rsi
        movq    %rax, %rdi
        call    memcpy@PLT
        movl    -395640(%rbp), %eax
```

c. struct as local var

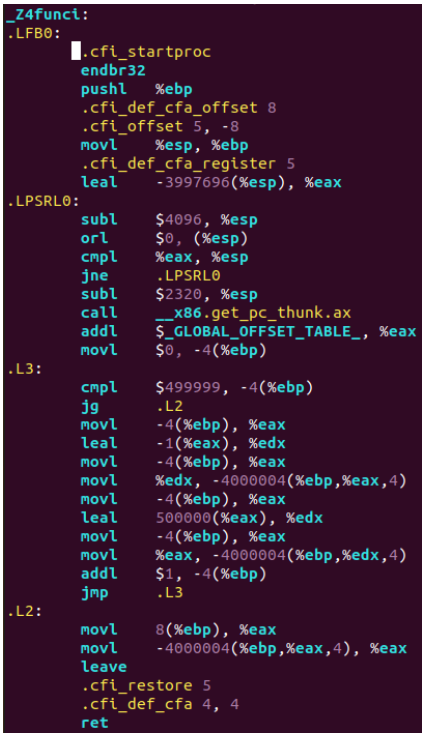
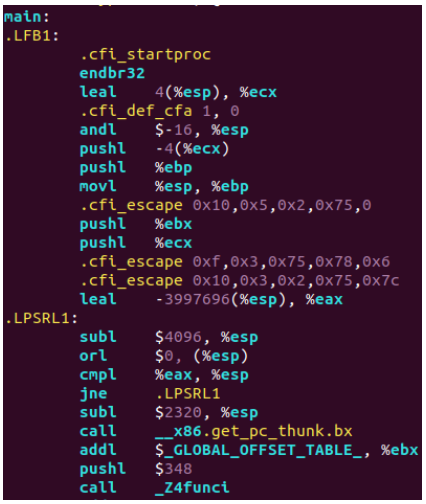
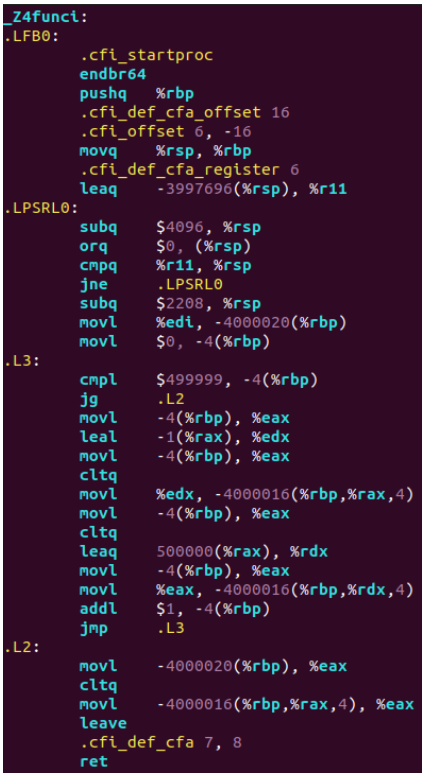
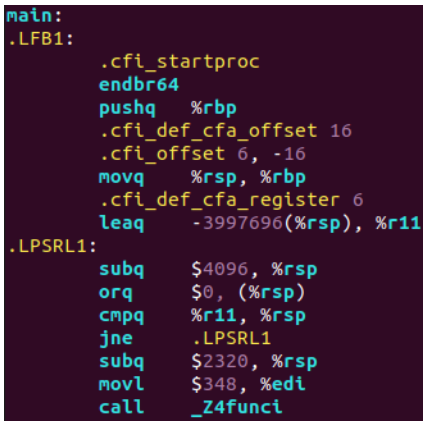
```
const int n = 100000;
struct A{
        int arr1[n];
        int arr2[n];
};

int func(int k)
{
        A s;
        for(int i = 0; i < n; i ++)
        {
                s.arr1[i] = i - 1;
                s.arr2[i] = i;
        }
        return s.arr1[k];
}

int main()
{
        A s1;
        int n = func(348);
        printf("%d",n);
}
```

```
_Z4funci:
.LFB0:
        .cfi_startproc
        endbr32
        pushl   %ebp
        .cfi_def_cfa_offset 8
        .cfi_offset 5, -8
        movl    %esp, %ebp
        .cfi_def_cfa_register 5
        leal    -798720(%esp), %eax
.LPSRL0:
        subl    $4096, %esp
        orl     $0, (%esp)
        cmpl    %eax, %esp
        jne     .LPSRL0
        subl    $1296, %esp
        call    __x86.get_pc_thunk.ax
        addl    $_GLOBAL_OFFSET_TABLE_, %eax
        movl    $0, -4(%ebp)
.L3:
        cmpl    $99999, -4(%ebp)
        jg      .L2
        movl    -4(%ebp), %eax
        leal    -1(%eax), %edx
        movl    -4(%ebp), %eax
        movl    %edx, -800004(%ebp,%eax,4)
        movl    -4(%ebp), %eax
        leal    100000(%eax), %edx
        movl    -4(%ebp), %eax
        movl    %eax, -800004(%ebp,%edx,4)
        addl    $1, -4(%ebp)
        jmp     .L3
.L2:
        movl    8(%ebp), %eax
        movl    -800004(%ebp,%eax,4), %eax
        leave
        .cfi_restore 5
        .cfi_def_cfa 4, 4
        ret
```

```
main:
.LFB1:
        .cfi_startproc
        endbr32
        leal    4(%esp), %ecx
        .cfi_def_cfa 1, 0
        andl    $-16, %esp
        pushl   -4(%ecx)
        pushl   %ebp
        movl    %esp, %ebp
        .cfi_escape 0x10,0x5,0x2,0x75,0
        pushl   %ebx
        pushl   %ecx
        .cfi_escape 0xf,0x3,0x75,0x78,0x6
        .cfi_escape 0x10,0x3,0x2,0x75,0x7c
        leal    -798720(%esp), %eax
.LPSRL1:
        subl    $4096, %esp
        orl     $0, (%esp)
        cmpl    %eax, %esp
        jne     .LPSRL1
        subl    $1296, %esp
        call    __x86.get_pc_thunk.bx
        addl    $_GLOBAL_OFFSET_TABLE_, %ebx
        pushl   $348
        call    _Z4funci
        addl    $4, %esp
```

```
_Z4funci:
.LFB0:
        .cfi_startproc
        endbr64
        pushq   %rbp
        .cfi_def_cfa_offset 16
        .cfi_offset 6, -16
        movq    %rsp, %rbp
        .cfi_def_cfa_register 6
        leaq    -798720(%rsp), %r11
.LPSRL0:
        subq    $4096, %rsp
        orq     $0, (%rsp)
        cmpq    %r11, %rsp
        jne     .LPSRL0
        subq    $1184, %rsp
        movl    %edi, -800020(%rbp)
        movl    $0, -4(%rbp)
.L3:
        cmpl    $99999, -4(%rbp)
        jg      .L2
        movl    -4(%rbp), %eax
        leal    -1(%rax), %edx
        movl    -4(%rbp), %eax
        cltq
        movl    %edx, -800016(%rbp,%rax,4)
        movl    -4(%rbp), %eax
        cltq
        leaq    100000(%rax), %rdx
        movl    -4(%rbp), %eax
        movl    %eax, -800016(%rbp,%rdx,4)
        addl    $1, -4(%rbp)
        jmp     .L3
.L2:
        movl    -800020(%rbp), %eax
        cltq
        movl    -800016(%rbp,%rax,4), %eax
        leave
        .cfi_def_cfa 7, 8
        ret
```

```
main:
.LFB1:
        .cfi_startproc
        endbr64
        pushq   %rbp
        .cfi_def_cfa_offset 16
        .cfi_offset 6, -16
        movq    %rsp, %rbp
        .cfi_def_cfa_register 6
        leaq    -798720(%rsp), %r11
.LPSRL1:
        subq    $4096, %rsp
        orq     $0, (%rsp)
        cmpq    %r11, %rsp
        jne     .LPSRL1
        subq    $1296, %rsp
        movl    $348, %edi
        call    _Z4funci
```

d. changing size(relative to c.)	Size = 500000	 	 

- just big numbers of allocated memory appear
- also it seems that %rsp cannot be reduced by a bigger number than 4096, thus there is a cycle .LPSRL1, which decrements %rsp only by 4096. After that %rsp is once again decremented for lacking bytes.

6. Recursion

C

```c
const int n = 100000;

int recursion(int a)
{
        int arr[n];
        arr[4] = a;
        if(arr[4] < 19)
        {
                return recursion(arr[4] + 1);
        }
        else return arr[4];
}

int main()
{
        int c = recursion(0);
}
```

32bit

```
recursion:
.LFB0:
        .cfi_startproc
        endbr32
        pushl   %ebp
        .cfi_def_cfa_offset 8
        .cfi_offset 5, -8
        movl    %esp, %ebp
        .cfi_def_cfa_register 5
        pushl   %ebx
        subl    $20, %esp
        .cfi_offset 3, -12
        call    __x86.get_pc_thunk.ax
        addl    $_GLOBAL_OFFSET_TABLE_, %eax
        movl    %esp, %eax
        movl    %eax, %ebx
        movl    $100000, %eax
        subl    $1, %eax
        movl    %eax, -12(%ebp)
        movl    $100000, %eax
        leal    0(,%eax,4), %edx
        movl    $16, %eax
        subl    $1, %eax
        addl    %edx, %eax
        movl    $16, %ecx
        movl    $0, %edx
        divl    %ecx
        imull   $16, %eax, %eax
        movl    %eax, %edx
        andl    $-4096, %edx
        movl    %esp, %ecx
        subl    %edx, %ecx
        movl    %ecx, %edx
.L2:
        cmpl    %edx, %esp
        je      .L3
        subl    $4096, %esp
        orl     $0, 4092(%esp)
        jmp     .L2
.L3:
        movl    %eax, %edx
        andl    $4095, %edx
        subl    %edx, %esp
        movl    %eax, %edx
        andl    $4095, %edx
        testl   %edx, %edx
        je      .L4
        andl    $4095, %eax
        subl    $4, %eax
        addl    %esp, %eax
        orl     $0, (%eax)
```

64bit

```
recursion:
.LFB0:
        .cfi_startproc
        endbr64
        pushq   %rbp
        .cfi_def_cfa_offset 16
        .cfi_offset 6, -16
        movq    %rsp, %rbp
        .cfi_def_cfa_register 6
        pushq   %rbx
        subq    $40, %rsp
        .cfi_offset 3, -24
        movl    %edi, -36(%rbp)
        movq    %rsp, %rax
        movq    %rax, %rbx
        movl    $100000, %eax
        cltq
        subq    $1, %rax
        movq    %rax, -24(%rbp)
        movl    $100000, %eax
        cltq
        movq    %rax, %r10
        movl    $0, %r11d
        movl    $100000, %eax
        cltq
        movq    %rax, %r8
        movl    $0, %r9d
        movl    $100000, %eax
        cltq
        leaq    0(,%rax,4), %rdx
        movl    $16, %eax
        subq    $1, %rax
        addq    %rdx, %rax
        movl    $16, %esi
        movl    $0, %edx
        divq    %rsi
        imulq   $16, %rax, %rax
        movq    %rax, %rdx
        andq    $-4096, %rdx
        movq    %rsp, %rcx
        subq    %rdx, %rcx
        movq    %rcx, %rdx
.L2:
        cmpq    %rdx, %rsp
        je      .L3
        subq    $4096, %rsp
        orq     $0, 4088(%rsp)
        jmp     .L2
```

```
.L4:
        movl    %esp, %eax
        addl    $3, %eax
        shrl    $2, %eax
        sall    $2, %eax
        movl    %eax, -16(%ebp)
        movl    -16(%ebp), %eax
        movl    8(%ebp), %edx
        movl    %edx, 16(%eax)
        movl    -16(%ebp), %eax
        movl    16(%eax), %eax
        cmpl    $18, %eax
        jg      .L5
        movl    -16(%ebp), %eax
        movl    16(%eax), %eax
        addl    $1, %eax
        subl    $12, %esp
        pushl   %eax
        call    recursion
        addl    $16, %esp
        jmp     .L6
.L5:
        movl    -16(%ebp), %eax
        movl    16(%eax), %eax
.L6:
        movl    %ebx, %esp
        movl    -4(%ebp), %ebx
        leave
        .cfi_restore 5
        .cfi_restore 3
        .cfi_def_cfa 4, 4
        ret
```

```
main:
.LFB1:
        .cfi_startproc
        endbr32
        leal    4(%esp), %ecx
        .cfi_def_cfa 1, 0
        andl    $-16, %esp
        pushl   -4(%ecx)
        pushl   %ebp
        movl    %esp, %ebp
        .cfi_escape 0x10,0x5,0x2,0x75,0
        pushl   %ecx
        .cfi_escape 0xf,0x3,0x75,0x7c,0x6
        subl    $20, %esp
        call    __x86.get_pc_thunk.ax
        addl    $_GLOBAL_OFFSET_TABLE_, %eax
        subl    $12, %esp
        pushl   $0
        call    recursion
        addl    $16, %esp
        movl    %eax, -12(%ebp)
        movl    $0, %eax
        movl    -4(%ebp), %ecx
```

```
.L3:
        movq    %rax, %rdx
        andl    $4095, %edx
        subq    %rdx, %rsp
        movq    %rax, %rdx
        andl    $4095, %edx
        testq   %rdx, %rdx
        je      .L4
        andl    $4095, %eax
        subq    $8, %rax
        addq    %rsp, %rax
        orq     $0, (%rax)
.L4:
        movq    %rsp, %rax
        addq    $3, %rax
        shrq    $2, %rax
        salq    $2, %rax
        movq    %rax, -32(%rbp)
        movq    -32(%rbp), %rax
        movl    -36(%rbp), %edx
        movl    %edx, 16(%rax)
        movq    -32(%rbp), %rax
        movl    16(%rax), %eax
        cmpl    $18, %eax
        jg      .L5
        movq    -32(%rbp), %rax
        movl    16(%rax), %eax
        addl    $1, %eax
        movl    %eax, %edi
        call    recursion
        jmp     .L6
.L5:
        movq    -32(%rbp), %rax
        movl    16(%rax), %eax
.L6:
        movq    %rbx, %rsp
        movq    -8(%rbp), %rbx
        leave
        .cfi_def_cfa 7, 8
        ret
```

```
main:
.LFB1:
        .cfi_startproc
        endbr64
        pushq   %rbp
        .cfi_def_cfa_offset 16
        .cfi_offset 6, -16
        movq    %rsp, %rbp
        .cfi_def_cfa_register 6
        subq    $16, %rsp
        movl    $0, %edi
        call    recursion
        movl    %eax, -4(%rbp)
        movl    $0, %eax
        leave
        .cfi_def_cfa 7, 8
        ret
```

C++	32bit	64bit

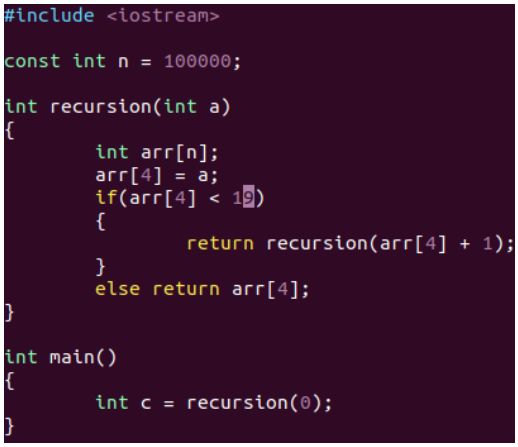
```cpp
#include <iostream>

const int n = 100000;

int recursion(int a)
{
        int arr[n];
        arr[4] = a;
        if(arr[4] < 19)
        {
                return recursion(arr[4] + 1);
        }
        else return arr[4];
}

int main()
{
        int c = recursion(0);
}
```

```
_Z9recursioni:
.LFB1519:
        .cfi_startproc
        endbr32
        pushl   %ebp
        .cfi_def_cfa_offset 8
        .cfi_offset 5, -8
        movl    %esp, %ebp
        .cfi_def_cfa_register 5
        leal    -397312(%esp), %eax
.LPSRL0:
        subl    $4096, %esp
        orl     $0, (%esp)
        cmpl    %eax, %esp
        jne     .LPSRL0
        subl    $2712, %esp
        call    __x86.get_pc_thunk.ax
        addl    $_GLOBAL_OFFSET_TABLE_, %eax
        movl    %gs:20, %eax
        movl    %eax, -12(%ebp)
        xorl    %eax, %eax
        movl    8(%ebp), %eax
        movl    %eax, -399996(%ebp)
        movl    -399996(%ebp), %eax
        cmpl    $18, %eax
        jg      .L2
        movl    -399996(%ebp), %eax
        addl    $1, %eax
        subl    $12, %esp
        pushl   %eax
        call    _Z9recursioni
        addl    $16, %esp
        jmp     .L4
.L2:
        movl    -399996(%ebp), %eax
.L4:
        movl    -12(%ebp), %edx
        xorl    %gs:20, %edx
        je      .L5
        call    __stack_chk_fail_local
.L5:
        leave
```

```
main:
.LFB1520:
        .cfi_startproc
        endbr32
        leal    4(%esp), %ecx
        .cfi_def_cfa 1, 0
        andl    $-16, %esp
        pushl   -4(%ecx)
        pushl   %ebp
        movl    %esp, %ebp
        .cfi_escape 0x10,0x5,0x2,0x75,0
        pushl   %ecx
        .cfi_escape 0xf,0x3,0x75,0x7c,0x6
        subl    $20, %esp
        call    __x86.get_pc_thunk.ax
        addl    $_GLOBAL_OFFSET_TABLE_, %eax
        subl    $12, %esp
        pushl   $0
        call    _Z9recursioni
        addl    $16, %esp
        movl    %eax, -12(%ebp)
        movl    $0, %eax
        movl    -4(%ebp), %ecx
        .cfi_def_cfa 1, 0
        leave
        .cfi_restore 5
        leal    -4(%ecx), %esp
        .cfi_def_cfa 4, 4
        ret
```

```
_Z9recursioni:
.LFB1522:
        .cfi_startproc
        endbr64
        pushq   %rbp
        .cfi_def_cfa_offset 16
        .cfi_offset 6, -16
        movq    %rsp, %rbp
        .cfi_def_cfa_register 6
        leaq    -397312(%rsp), %r11
.LPSRL0:
        subq    $4096, %rsp
        orq     $0, (%rsp)
        cmpq    %r11, %rsp
        jne     .LPSRL0
        subq    $2704, %rsp
        movl    %edi, -400004(%rbp)
        movl    -400004(%rbp), %eax
        movl    %eax, -399984(%rbp)
        movl    -399984(%rbp), %eax
        cmpl    $18, %eax
        jg      .L2
        movl    -399984(%rbp), %eax
        addl    $1, %eax
        movl    %eax, %edi
        call    _Z9recursioni
        jmp     .L4
.L2:
        movl    -399984(%rbp), %eax
.L4:
        leave
```

```
main:
.LFB1523:
        .cfi_startproc
        endbr64
        pushq   %rbp
        .cfi_def_cfa_offset 16
        .cfi_offset 6, -16
        movq    %rsp, %rbp
        .cfi_def_cfa_register 6
        subq    $16, %rsp
        movl    $0, %edi
        call    _Z9recursioni
        movl    %eax, -4(%rbp)
        movl    $0, %eax
        leave
        .cfi_def_cfa 7, 8
        ret
```

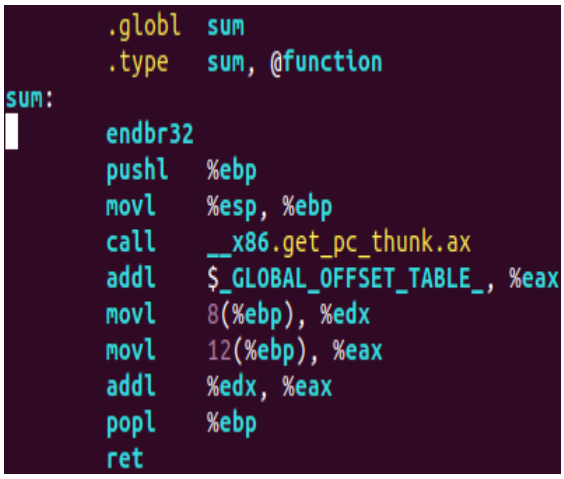
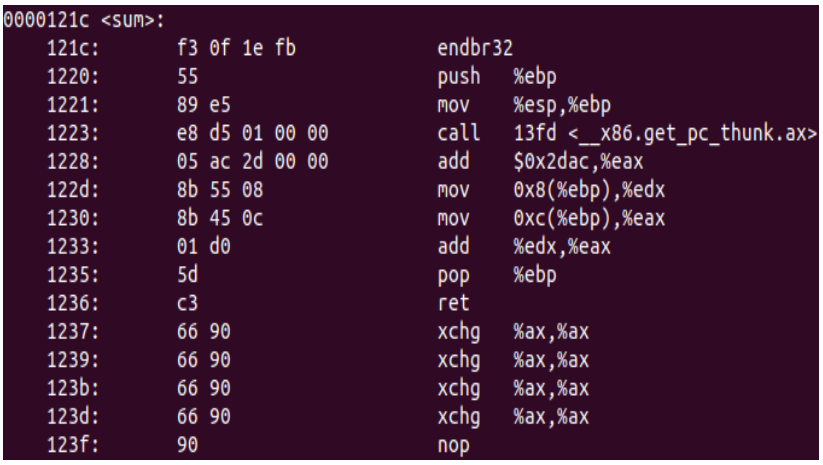
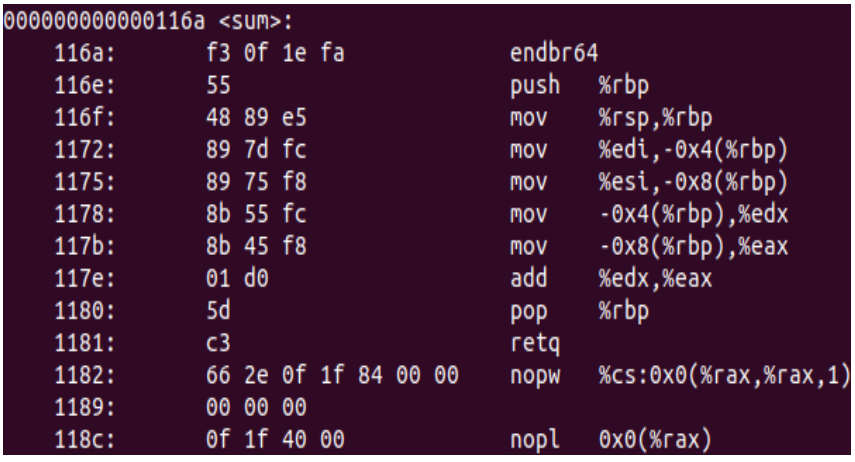
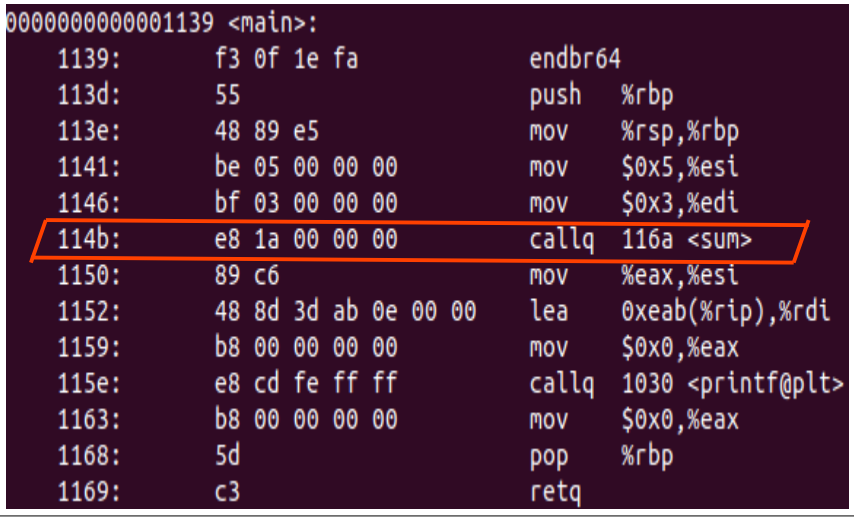
- Differences between 32bit and 64bit are not that significant, just the amount of memory allocated for arrays(differs by approximately 0-20 bytes)
- 

```
#include <iostream>

const int n = 100000;

int recursion(int a)
{
        int arr[n];
        arr[4] = a;
        if(arr[4] < 19)
        {
                return recursion(arr[4] + 1);
        }
        else return arr[4];
}

int main()
{
        int c = recursion(0);
}
```

  program executes correctly, when arr[4] < 19, so for 20 recursion calls. If I write arr[4] < 20 – stack overflow occurs. So the estimated stack size is between 20*100000*4 bytes ≈ 7800KB and 8200KB. So assuming that stack size is a good looking number, it’s around 8MB.

Lab 4

1. Static libraries

bit	Assembly(sum.s)	Binary(library part after linking)
32		
64		 

```
        .globl  sum
        .type   sum, @function
sum:
        endbr32
        pushl   %ebp
        movl    %esp, %ebp
        call    __x86.get_pc_thunk.ax
        addl    $_GLOBAL_OFFSET_TABLE_, %eax
        movl    8(%ebp), %edx
        movl    12(%ebp), %eax
        addl    %edx, %eax
        popl    %ebp
        ret
```

```
0000121c <sum>:
    121c:       f3 0f 1e fb             endbr32
    1220:       55                      push   %ebp
    1221:       89 e5                   mov    %esp,%ebp
    1223:       e8 d5 01 00 00          call   13fd <__x86.get_pc_thunk.ax>
    1228:       05 ac 2d 00 00          add    $0x2dac,%eax
    122d:       8b 55 08                mov    0x8(%ebp),%edx
    1230:       8b 45 0c                mov    0xc(%ebp),%eax
    1233:       01 d0                   add    %edx,%eax
    1235:       5d                      pop    %ebp
    1236:       c3                      ret
    1237:       66 90                   xchg   %ax,%ax
    1239:       66 90                   xchg   %ax,%ax
    123b:       66 90                   xchg   %ax,%ax
    123d:       66 90                   xchg   %ax,%ax
    123f:       90                      nop
```

```
.globl sum
sum:
        endbr64
        pushq   %rbp
        movq    %rsp, %rbp
        movl    %edi, -4(%rbp)
        movl    %esi, -8(%rbp)
        movl    -4(%rbp), %edx
        movl    -8(%rbp), %eax
        addl    %edx, %eax
        popq    %rbp
        ret
```

```
000000000000116a <sum>:
    116a:       f3 0f 1e fa             endbr64
    116e:       55                      push   %rbp
    116f:       48 89 e5                mov    %rsp,%rbp
    1172:       89 7d fc                mov    %edi,-0x4(%rbp)
    1175:       89 75 f8                mov    %esi,-0x8(%rbp)
    1178:       8b 55 fc                mov    -0x4(%rbp),%edx
    117b:       8b 45 f8                mov    -0x8(%rbp),%eax
    117e:       01 d0                   add    %edx,%eax
    1180:       5d                      pop    %rbp
    1181:       c3                      retq
    1182:       66 2e 0f 1f 84 00 00    nopw   %cs:0x0(%rax,%rax,1)
    1189:       00 00 00
    118c:       0f 1f 40 00             nopl   0x0(%rax)
```

```
0000000000001139 <main>:
    1139:       f3 0f 1e fa             endbr64
    113d:       55                      push   %rbp
    113e:       48 89 e5                mov    %rsp,%rbp
    1141:       be 05 00 00 00          mov    $0x5,%esi
    1146:       bf 03 00 00 00          mov    $0x3,%edi
    114b:       e8 1a 00 00 00          callq  116a <sum>
    1150:       89 c6                   mov    %eax,%esi
    1152:       48 8d 3d ab 0e 00 00    lea    0xeab(%rip),%rdi
    1159:       b8 00 00 00 00          mov    $0x0,%eax
    115e:       e8 cd fe ff ff          callq  1030 <printf@plt>
    1163:       b8 00 00 00 00          mov    $0x0,%eax
    1168:       5d                      pop    %rbp
    1169:       c3                      retq
```

- First obvious remark is 32bit system addresses have 8 digits, 64bit – 16.
- sum function is called in main by its address(116a)
- If we look at binaries of created libraries before linking, we can notice that function and other symbols still have there names definition, in other word functions and variables are not bound to any specific address, thus symbols associated with 00000000(32bit):

```
File: libSUM.a(sum.o)

Symbol table '.symtab' contains 5 entries:
   Num:    Value          Size Type    Bind   Vis      Ndx Name
     0: 0000000000000000     0 NOTYPE  LOCAL  DEFAULT  UND
     1: 0000000000000000     0 SECTION LOCAL  DEFAULT    1
     2: 0000000000000000     0 SECTION LOCAL  DEFAULT    2
     3: 0000000000000000     0 SECTION LOCAL  DEFAULT    3
     4: 0000000000000000     0 NOTYPE  GLOBAL DEFAULT    1 sum
```

This is a **relocatable object file**.

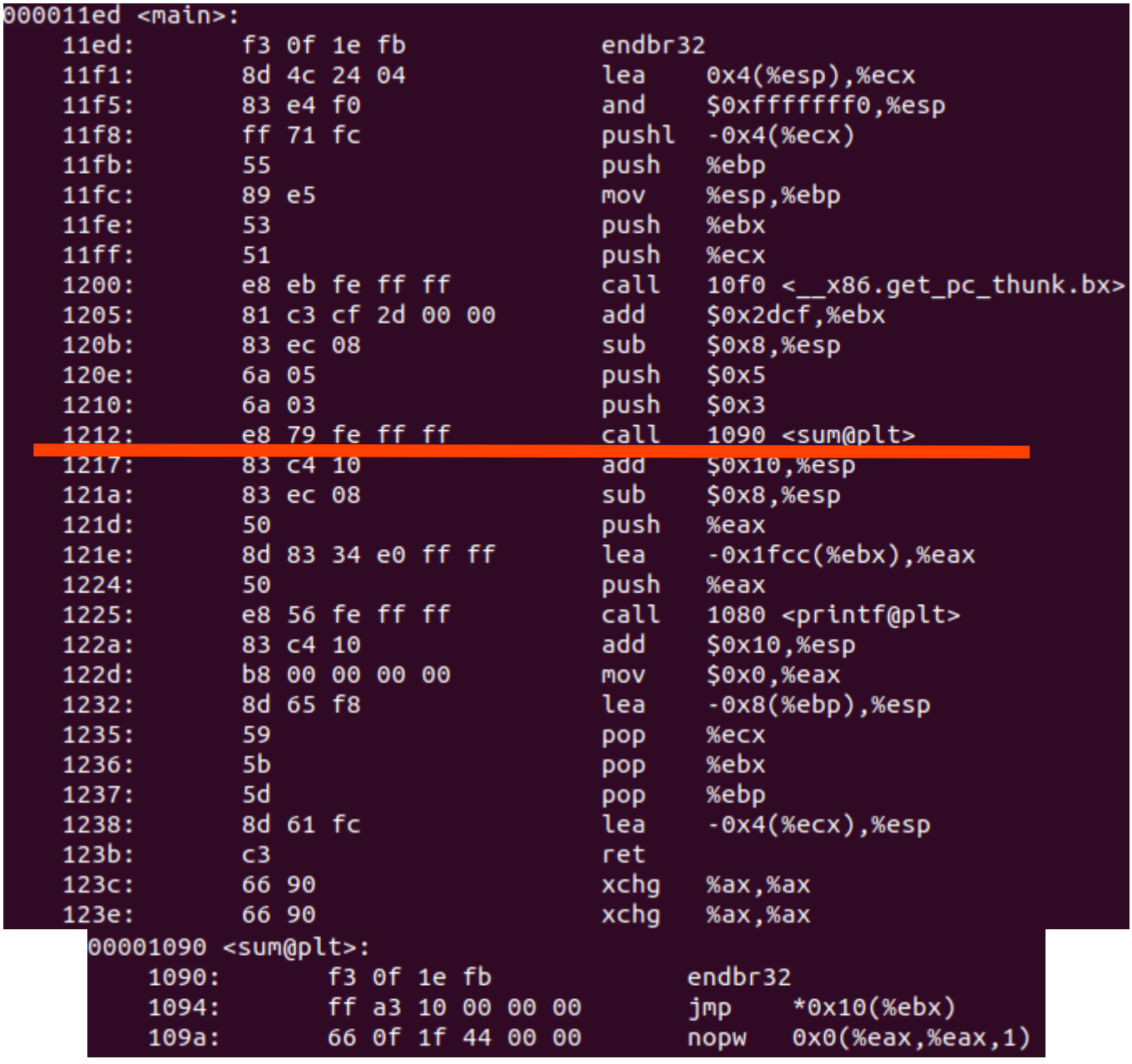
After linking all object files (and libraries), an e**xecutable object file** is created in which all symbols are associated with real address.

```
    54: 000000000000116a     0 NOTYPE  GLOBAL DEFAULT   14 sum
    55: 0000000000002000     4 OBJECT  GLOBAL DEFAULT   16 _IO_stdin_used
    56: 0000000000001190   101 FUNC    GLOBAL DEFAULT   14 __libc_csu_init
    57: 0000000000004018     0 NOTYPE  GLOBAL DEFAULT   24 _end
    58: 0000000000001050    47 FUNC    GLOBAL DEFAULT   14 _start
    59: 0000000000004010     0 NOTYPE  GLOBAL DEFAULT   24 __bss_start
    60: 0000000000001139    49 FUNC    GLOBAL DEFAULT   14 main
```

Row 54 is referenced to sum function and has an address.

In total: After the compiler and assembler generate the relocatable object file, the data start at address 0. The linker then relocates these sections by associating each with a location with in memory.

2. Dynamic libraries

bit	Binary(library part after linking)
32	

```
000011ed <main>:
    11ed:       f3 0f 1e fb             endbr32
    11f1:       8d 4c 24 04             lea    0x4(%esp),%ecx
    11f5:       83 e4 f0                and    $0xfffffff0,%esp
    11f8:       ff 71 fc                pushl  -0x4(%ecx)
    11fb:       55                      push   %ebp
    11fc:       89 e5                   mov    %esp,%ebp
    11fe:       53                      push   %ebx
    11ff:       51                      push   %ecx
    1200:       e8 eb fe ff ff          call   10f0 <__x86.get_pc_thunk.bx>
    1205:       81 c3 cf 2d 00 00       add    $0x2dcf,%ebx
    120b:       83 ec 08                sub    $0x8,%esp
    120e:       6a 05                   push   $0x5
    1210:       6a 03                   push   $0x3
    1212:       e8 79 fe ff ff          call   1090 <sum@plt>
    1217:       83 c4 10                add    $0x10,%esp
    121a:       83 ec 08                sub    $0x8,%esp
    121d:       50                      push   %eax
    121e:       8d 83 34 e0 ff ff       lea    -0x1fcc(%ebx),%eax
    1224:       50                      push   %eax
    1225:       e8 56 fe ff ff          call   1080 <printf@plt>
    122a:       83 c4 10                add    $0x10,%esp
    122d:       b8 00 00 00 00          mov    $0x0,%eax
    1232:       8d 65 f8                lea    -0x8(%ebp),%esp
    1235:       59                      pop    %ecx
    1236:       5b                      pop    %ebx
    1237:       5d                      pop    %ebp
    1238:       8d 61 fc                lea    -0x4(%ecx),%esp
    123b:       c3                      ret
    123c:       66 90                   xchg   %ax,%ax
    123e:       66 90                   xchg   %ax,%ax
00001090 <sum@plt>:
    1090:       f3 0f 1e fb             endbr32
    1094:       ff a3 10 00 00 00       jmp    *0x10(%ebx)
    109a:       66 0f 1f 44 00 00       nopw   0x0(%eax,%eax,1)
```

64

```
0000000000001169 <main>:
    1169:       f3 0f 1e fa             endbr64
    116d:       55                      push   %rbp
    116e:       48 89 e5                mov    %rsp,%rbp
    1171:       be 05 00 00 00          mov    $0x5,%esi
    1176:       bf 03 00 00 00          mov    $0x3,%edi
    117b:       e8 f0 fe ff ff          callq  1070 <sum@plt>
    1180:       89 c6                   mov    %eax,%esi
    1182:       48 8d 3d 7b 0e 00 00    lea    0xe7b(%rip),%rdi
    1189:       b8 00 00 00 00          mov    $0x0,%eax
    118e:       e8 cd fe ff ff          callq  1060 <printf@plt>
    1193:       b8 00 00 00 00          mov    $0x0,%eax
    1198:       5d                      pop    %rbp
    1199:       c3                      retq
    119a:       66 0f 1f 44 00 00       nopw   0x0(%rax,%rax,1)
0000000000001070 <sum@plt>:
    1070:       f3 0f 1e fa             endbr64
    1074:       f2 ff 25 55 2f 00 00    bnd jmpq *0x2f55(%rip)        # 3fd0 <sum>
    107b:       0f 1f 44 00 00          nopl   0x0(%rax,%rax,1)
0000000000003fb0 <_GLOBAL_OFFSET_TABLE_>:
    3fb0:       b0 3d                   mov    $0x3d,%al
        ...
    3fc6:       00 00                   add    %al,(%rax)
    3fc8:       30 10                   xor    %dl,(%rax)
    3fca:       00 00                   add    %al,(%rax)
    3fcc:       00 00                   add    %al,(%rax)
    3fce:       00 00                   add    %al,(%rax)
    3fd0:       40 10 00                adc    %al,(%rax)
```

```
sonamkrlm@sonamkrlm-IdeaPad-5-15ALC05:~/programming/assembler/Lab4$ readelf -h libSUM_SHARED.so |grep Type
  Type:                              DYN (Shared object file)
sonamkrlm@sonamkrlm-IdeaPad-5-15ALC05:~/programming/assembler/Lab4$ readelf -h 1_shared.o |grep Type
  Type:                              DYN (Shared object file)
```

- Both dynamic library and executable are shared object files
- In main there is a call for <sum@plt>, which means that library code is not copied(as with static libraries), but accessed via address. In line 1074(64bit, where underlined with red color) there is another call for a sum function, which is located in GOT. Using GOT allows shared libraries to be relocated to a different memory address at startup and avoid memory address conflicts with the main program or other shared libraries.

To sum up(similarities/differences between static and dynamic libraries):

- **both connected to program via linker(static/dynamic, resulting in an executable)**
- **static library's code accessed (inserted) at a compile time**
- **dynamic library's code accessed at a runtime as well as memory is being loaded with it**
- **static libraries upgrades require recompiling of all parts of program, whereas dynamic libraries upgrades independently**
- **static libraries are easy to install, with dynamics ones problems can occur**