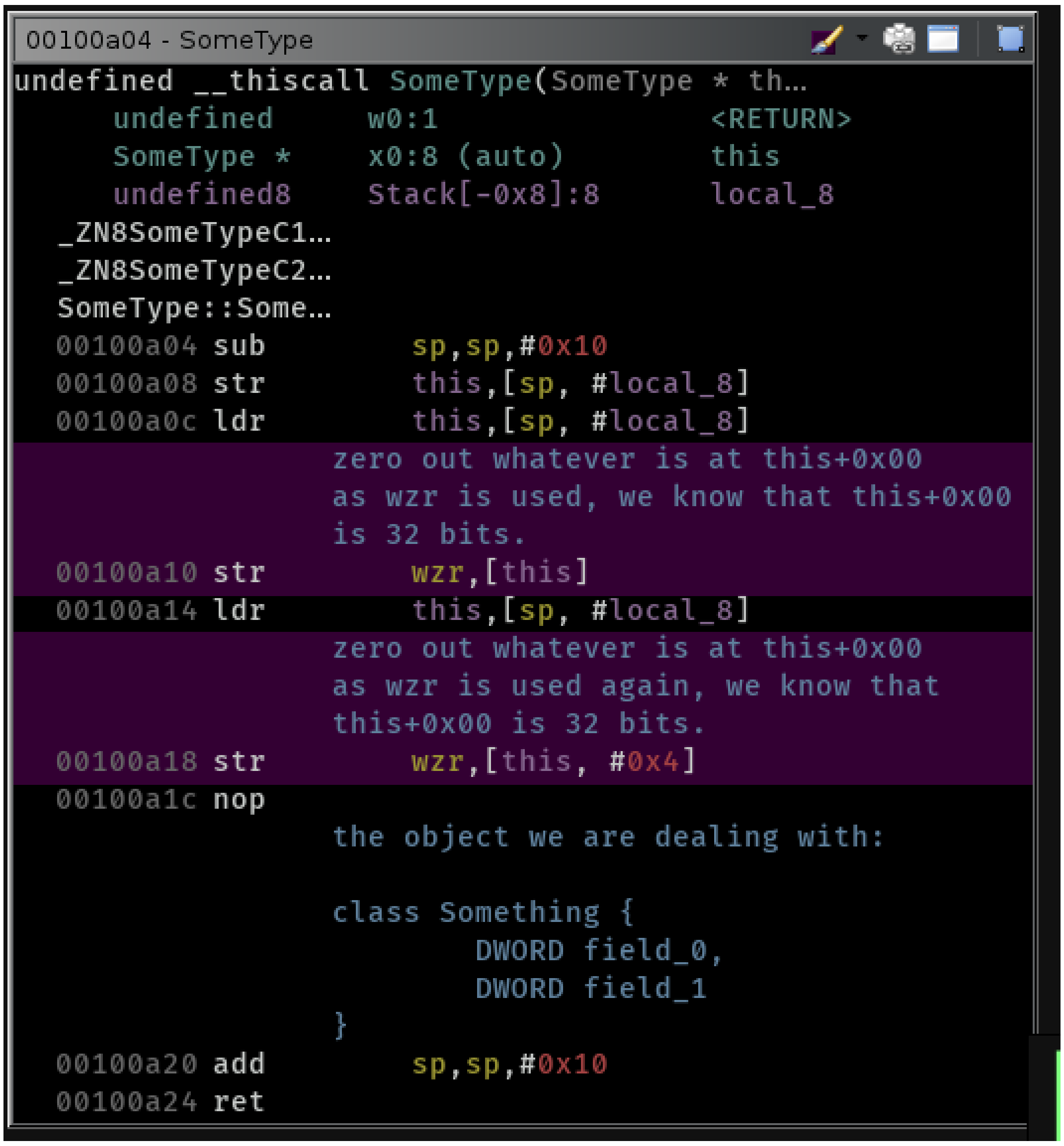
00_classes

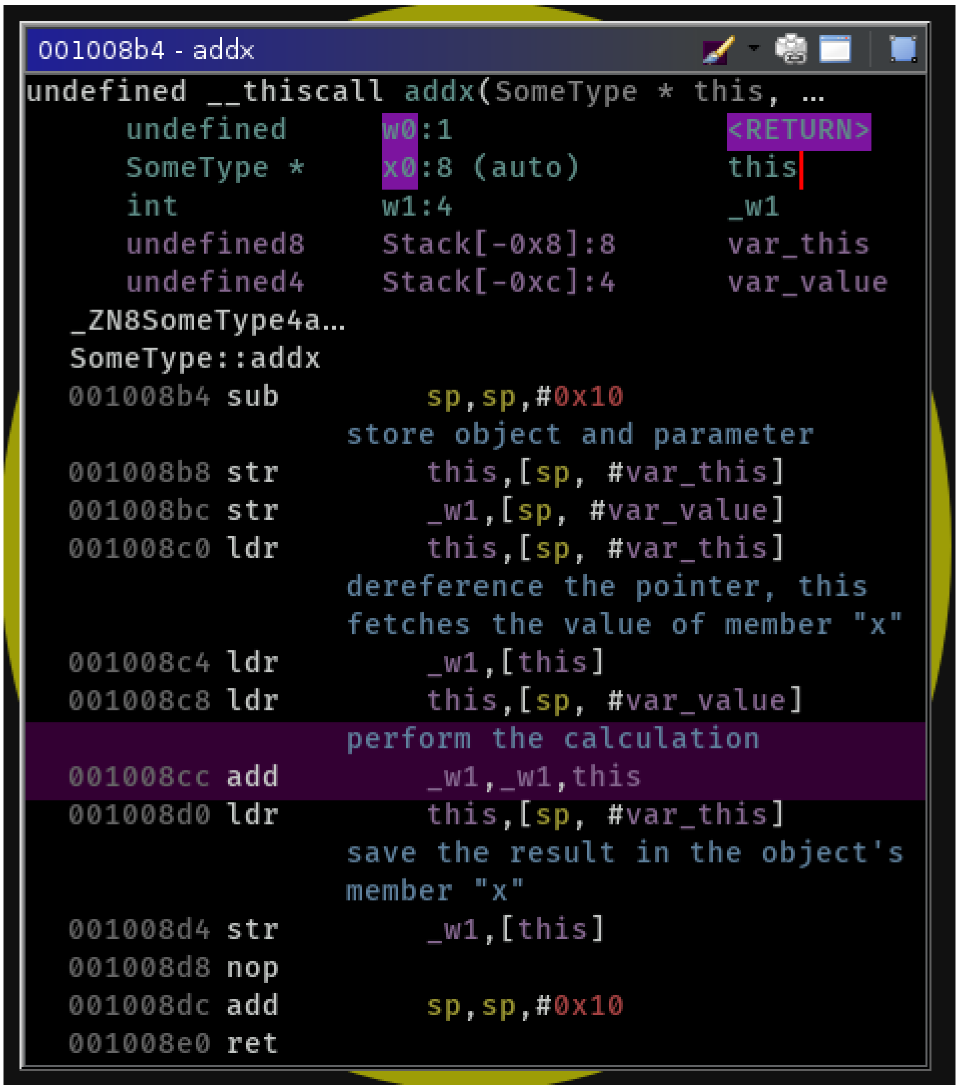
The first source code file we will have a look at a basic C++ class:

```cpp
/**
 * @file 00_classes.cpp
 * @author 0xca7 (0xca7.github.io)
 * @brief a simple C++ class to reverse
 * @version 0.1
 * @date 2022-12-27
 *
 * @copyright Copyright (c) 2022
 */

#include <iostream>
#include <stdio.h>

class SomeType {

        private:
                int x;
                int y;

        public:

                SomeType() {
                        x = 0; y = 0;
                }

                void addx(int x);
                void addy(int y);

                void show(void);
};

void
SomeType::addx(int x)
{
        this->x += x;
}

void
SomeType::addy(int y)
{
        this->y += y;
}

void
SomeType::show()
{
        printf("x: %d, y: %d\n", this->x, this->y);
}

int
main(void)
{

        SomeType s;

        s.addx(1);
        s.addy(1);
        s.show();

        return 0;
}
```

The first interesting part, starting from main, is the constructor, which is called via `SomeType s`. The assembly is shown below and allows us to reconstruct the class SomeType:

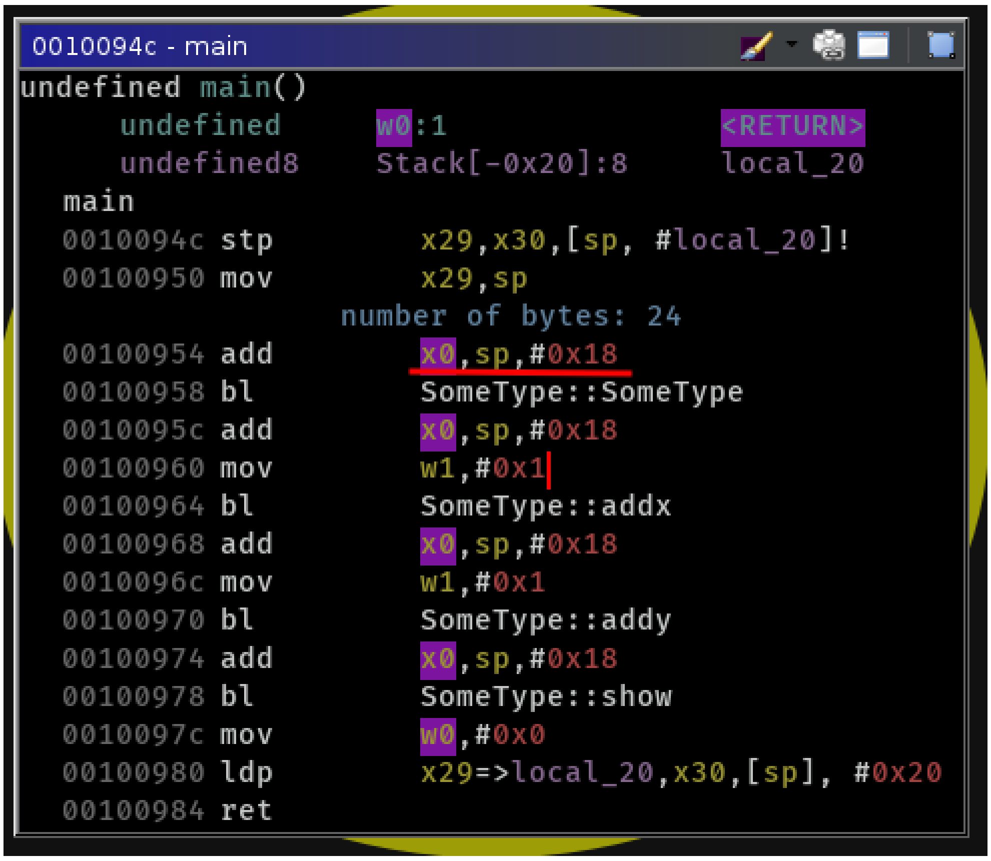

```
00100a04 - SomeType
undefined __thiscall SomeType(SomeType * th…
     undefined      w0:1               <RETURN>
     SomeType *     x0:8 (auto)        this
     undefined8     Stack[-0x8]:8      local_8
  _ZN8SomeTypeC1…
  _ZN8SomeTypeC2…
  SomeType::Some…
  00100a04 sub        sp,sp,#0x10
  00100a08 str        this,[sp, #local_8]
  00100a0c ldr        this,[sp, #local_8]
                  zero out whatever is at this+0x00
                  as wzr is used, we know that this+0x00
                  is 32 bits.
  00100a10 str        wzr,[this]
  00100a14 ldr        this,[sp, #local_8]
                  zero out whatever is at this+0x00
                  as wzr is used again, we know that
                  this+0x00 is 32 bits.
  00100a18 str        wzr,[this, #0x4]
  00100a1c nop
                  the object we are dealing with:

                  class Something {
                          DWORD field_0,
                          DWORD field_1
                  }
  00100a20 add        sp,sp,#0x10
  00100a24 ret
```

Next up: the add methods `addx` and `addy` - both are almost equivalent in assembly. Let's do `addx`.



```
001008b4 - addx
undefined __thiscall addx(SomeType * this, …
     undefined       w0:1            <RETURN>
     SomeType *      x0:8 (auto)     this
     int             w1:4            _w1
     undefined8      Stack[-0x8]:8   var_this
     undefined4      Stack[-0xc]:4   var_value
  _ZN8SomeType4a…
  SomeType::addx
  001008b4 sub        sp,sp,#0x10
                   store object and parameter
  001008b8 str        this,[sp, #var_this]
  001008bc str        _w1,[sp, #var_value]
  001008c0 ldr        this,[sp, #var_this]
                   dereference the pointer, this
                   fetches the value of member "x"
  001008c4 ldr        _w1,[this]
  001008c8 ldr        this,[sp, #var_value]
                   perform the calculation
  001008cc add        _w1,_w1,this
  001008d0 ldr        this,[sp, #var_this]
                   save the result in the object's
                   member "x"
  001008d4 str        _w1,[this]
  001008d8 nop
  001008dc add        sp,sp,#0x10
  001008e0 ret
```

Finally, this is the main function:



```
0010094c - main
undefined main()
     undefined       w0:1            <RETURN>
     undefined8      Stack[-0x20]:8  local_20
  main
  0010094c stp        x29,x30,[sp, #local_20]!
  00100950 mov        x29,sp
                   number of bytes: 24
  00100954 add        x0,sp,#0x18
  00100958 bl         SomeType::SomeType
  0010095c add        x0,sp,#0x18
  00100960 mov        w1,#0x1
  00100964 bl         SomeType::addx
  00100968 add        x0,sp,#0x18
  0010096c mov        w1,#0x1
  00100970 bl         SomeType::addy
  00100974 add        x0,sp,#0x18
  00100978 bl         SomeType::show
  0010097c mov        w0,#0x0
  00100980 ldp        x29=>local_20,x30,[sp], #0x20
  00100984 ret
```

As can be seen address `00100954` the object which is created resides at `sp+0x18`
