Alex Yang (ahy3nz)
inlab9

```
main:
.LFB982:
        .cfi_startproc
        .cfi_personality 0,__gxx_personality_v0
        .cfi_lsda 0,.LLSDA982
        push    ebp
        .cfi_def_cfa_offset 8
        .cfi_offset 5, -8
        mov     ebp, esp
        .cfi_def_cfa_register 5
        push    ebx
        and     esp, -16
        sub     esp, 48
        lea     eax, [esp+28]
        mov     DWORD PTR [esp], eax
```

In assembly, the fields that are in both the parent class and the subclass are pushed as values onto the activation record. In this case, we are allocating 12 bytes for the string fields in dummy (name and address). After the prologue and allocating space on the stack for field

```
.LEHB9:
        .cfi_offset 3, -12
        call    _ZN8subclassC1Ev
```

This snippet will call subclass's constructor.

```
_ZN8subclassC2Ev:
.LFB975:
        .cfi_startproc
        .cfi_personality 0,__gxx_personality_v0
        .cfi_lsda 0,.LLSDA975
        push    ebp
        .cfi_def_cfa_offset 8
        .cfi_offset 5, -8
        mov     ebp, esp
        .cfi_def_cfa_register 5
        push    ebx
        sub     esp, 20
        mov     eax, DWORD PTR [ebp+8]
        mov     DWORD PTR [esp], eax
.LEHB2:
```

```
        .cfi_offset 3, -12
        call    _ZN11parentclassC2Ev
```

In subclass, space is allocated for fields in subclass, and then the parent class constructor is called.

```
_ZN11parentclassC2Ev:
.LFB967:
        .cfi_startproc
        .cfi_personality 0,__gxx_personality_v0
        .cfi_lsda 0,.LLSDA967
        push    ebp
        .cfi_def_cfa_offset 8
        .cfi_offset 5, -8
        mov     ebp, esp
        .cfi_def_cfa_register 5
        push    ebx
        sub     esp, 36
        lea     eax, [ebp-9]
        mov     DWORD PTR [esp], eax
        .cfi_offset 3, -12
        call    _ZNSaIcEC1Ev
        mov     eax, DWORD PTR [ebp+8]
        lea     edx, [ebp-9]
        mov     DWORD PTR [esp+8], edx
        mov     DWORD PTR [esp+4], OFFSET FLAT:.LC0
        mov     DWORD PTR [esp], eax
.LEHB0:
        call    _ZNSsC1EPKcRKSaIcE
```

In the parent class constructor, more space is allocated for fields in the parent class.
When initializing an object, first the class's constructor is called. This means allocating space on stack for the class's fields. Within the class's constructor, the parent class's constructor is called, where space is allocated for the parent class's fields.

```
ZN11parentclass7SetNameESs:
.LFB972:
        .cfi_startproc
        push    ebp
        .cfi_def_cfa_offset 8
        .cfi_offset 5, -8
        mov     ebp, esp
        .cfi_def_cfa_register 5
```

```
    sub     esp, 24
    mov     eax, DWORD PTR [ebp+8]
    mov     edx, DWORD PTR [ebp+12]
    mov     DWORD PTR [esp+4], edx
    mov     DWORD PTR [esp], eax
    call    _ZNSsaSERKSs
    leave
```

After loading and moving around registers and values in the stack, setName is called. However, the method is the parent class's set name. Since local parameters were stored in offsets from the base pointer, we access those parameters and store them elsewhere on the stack because we are now storing those values in the name field.

```
_ZN8subclass10setAddressESs:
.LFB980:
    .cfi_startproc
    push    ebp
    .cfi_def_cfa_offset 8
    .cfi_offset 5, -8
    mov     ebp, esp
    .cfi_def_cfa_register 5
    sub     esp, 24
    mov     eax, DWORD PTR [ebp+8]
    lea     edx, [eax+4]
    mov     eax, DWORD PTR [ebp+12]
    mov     DWORD PTR [esp+4], eax
    mov     DWORD PTR [esp], edx
    call    _ZNSsaSERKSs
    leave
```

setAddress is the next call in the main method. Between the setName and the setAddress call, however, more values are moved around. Some calls are loading addresses into registers for the subroutine, and others are moving registers into offsets of esp (the local parameters). In set address, we are moving a local parameter into a register, which is then moved into another position on the stack because we are changing the address field of the dummy object.
In both setName and setAddress, very little differentiation is done even though one is a parent method and one is the class's method. Regardless, we are doing similar operations by accessing locations on the stack as local parameters and storing them elsewhere in the stack where we store the fields.

```
LEHE14:
```

```
        lea     eax, [esp+40]
        mov     DWORD PTR [esp], eax
.LEHB15:
        call    _ZNSsD1Ev
.LEHE15:
        lea     eax, [esp+47]
        mov     DWORD PTR [esp], eax
        call    _ZNSaIcED1Ev
        lea     eax, [esp+28]
        mov     DWORD PTR [esp], eax
.LEHB16:
        call    _ZN8subclass5printEv
```

After setting address, some commands are performed to prepare for the print command.

```
_ZN8subclass5printEv:
.LFB981:
        .cfi_startproc
        push    ebp
        .cfi_def_cfa_offset 8
        .cfi_offset 5, -8
        mov     ebp, esp
        .cfi_def_cfa_register 5
        sub     esp, 24
        mov     eax, DWORD PTR [ebp+8]
        mov     DWORD PTR [esp], eax
        call    _ZN11parentclass5printEv
        mov     eax, DWORD PTR [ebp+8]
        add     eax, 4
        mov     DWORD PTR [esp+4], eax
        mov     DWORD PTR [esp], OFFSET FLAT:_ZSt4cout
        call
_ZStlsIcSt11char_traitsIcESaIcEERSt13basic_ostreamIT_T0_ES7_RKSbIS4_S5_T1_E
        mov     DWORD PTR [esp+4], OFFSET
FLAT:_ZSt4endlIcSt11char_traitsIcEERSt13basic_ostreamIT_T0_ES6_
        mov     DWORD PTR [esp], eax
        call    _ZNSolsEPFRSoS_E
        leave
```

After accessing fields located in locations on the stack based on registers and offsets from the base pointer, the parent print method is called first - this was specified in the c++ code.

```
_ZN11parentclass5printEv:
.LFB973:
        .cfi_startproc
        push    ebp
        .cfi_def_cfa_offset 8
        .cfi_offset 5, -8
        mov     ebp, esp
        .cfi_def_cfa_register 5
        sub     esp, 24
        mov     eax, DWORD PTR [ebp+8]
        mov     DWORD PTR [esp+4], eax
        mov     DWORD PTR [esp], OFFSET FLAT:_ZSt4cout
        call
_ZStlsIcSt11char_traitsIcESaIcEERSt13basic_ostreamIT_T0_ES7_RKSbIS4_S5_T1_E
        mov     DWORD PTR [esp+4], OFFSET
FLAT:_ZSt4endlIcSt11char_traitsIcEERSt13basic_ostreamIT_T0_ES6_
        mov     DWORD PTR [esp], eax
        call    _ZNSolsEPFRSoS_E
        leave
```

In the parent's print method, we are still accessing fields based on offsets from stack pointers. Then the actual cout is performed. After completing the parent's print method, the rest of the cout in the subclass's print method is performed.

In summary, constructors call the parent constructors first, and then the actual subclass's constructor. There is no distinction between parent class and subclass fields on the stack - they are just offsets from the stack pointer. Accessing these fields is the same as loading addresses specified by stack pointer offsets (as seen by the print methods).