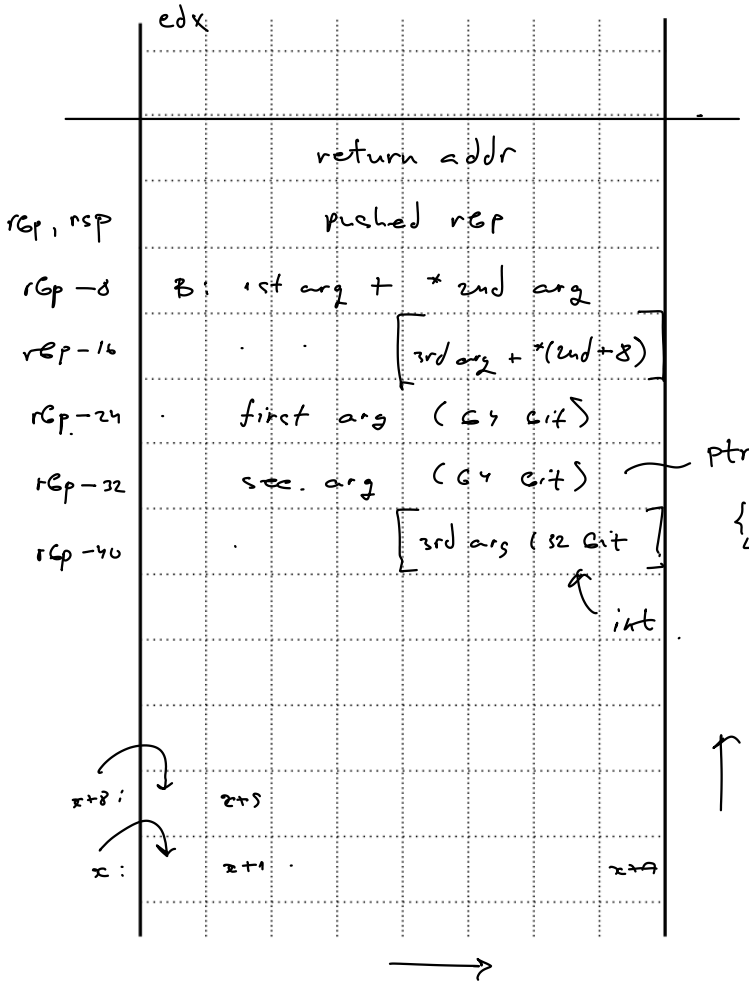
```
.intel_syntax noprefix
.text
.global foo
.type foo, @function
foo:
    push rbp
    mov rbp, rsp
    mov QWORD PTR -24[rbp], rdi
    mov QWORD PTR -32[rbp], rsi
    mov DWORD PTR -36[rbp], edx
    mov rax, QWORD PTR -32[rbp]
    mov rdx, QWORD PTR [rax]
    mov rax, QWORD PTR -24[rbp]
    add rax, rdx
    mov QWORD PTR -8[rbp], rax
    mov rax, QWORD PTR -32[rbp]
    mov edx, DWORD PTR 8[rax]
    mov eax, DWORD PTR -36[rbp]
    add eax, edx
    mov DWORD PTR -12[rbp], eax
    mov eax, DWORD PTR -12[rbp]
    movsx rdx, eax                  # MOVe with Sign eXtend
    mov rax, QWORD PTR -8[rbp]
    add rax, rdx
    pop rbp
    ret
```

(eaf, 128

rax := 2nd arg
rdx := * 2nd arg
rax := 1st arg

(rdi, rsi, rdx, rcx, r8, r9)




rax := 2nd arg

edx := *(2nd arg + 8)

eax := 3rd arg

A = 3rd arg + *(2nd arg + 8)

rdx := signExten(A).

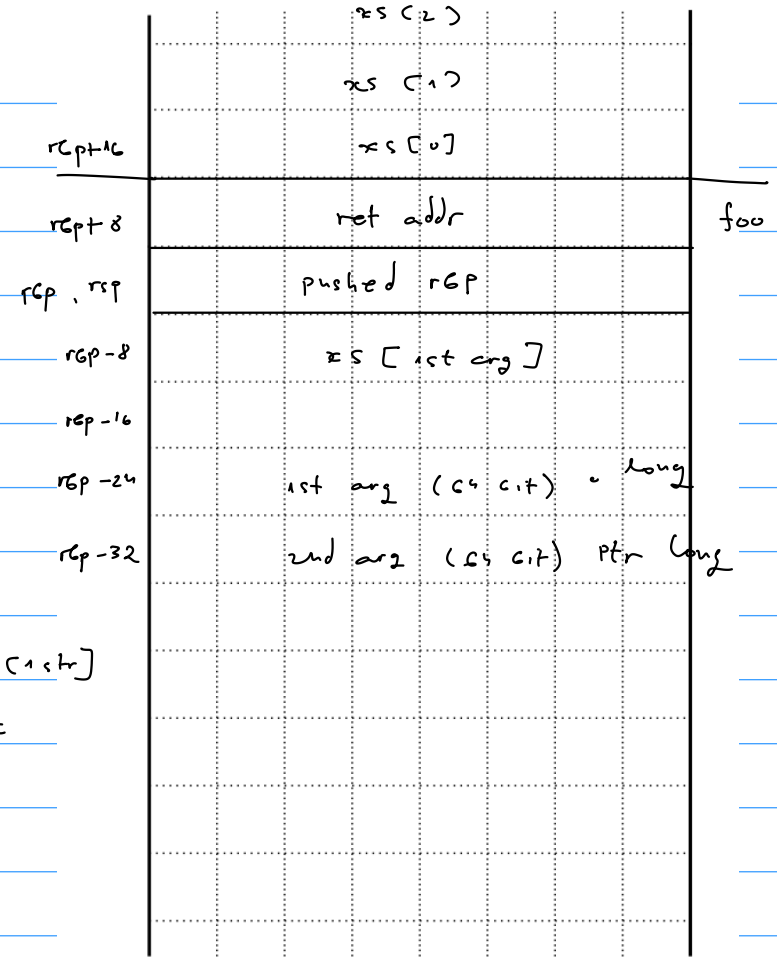
rax := A + B

```
long foo(long x, struct A *p, int y) {
    long A = y + p -> a2;
    long B = x + p -> a1;
    retur A + B;
}.
```

```
struct A {
    long a1;
    int a2;
}.
```

(rdi, rsi, rdx, rcx, r8, r9)

```
	.intel_syntax noprefix
	.text
	.global foo
	.type foo, @function
foo:
	push rbp
	mov rbp, rsp
	mov QWORD PTR -24[rbp],.rdi
	mov QWORD PTR -32[rbp],.rsi
	mov rax, QWORD PTR -24[rbp]
	mov rax, QWORD PTR 16[rbp+rax*8]
	mov QWORD PTR -8[rbp], rax
	mov rax, QWORD PTR -32[rbp]
	mov rdx, QWORD PTR [rax]
	mov rax, QWORD PTR -8[rbp]
	add rdx, rax
	mov rax, QWORD PTR -32[rbp]
	mov QWORD PTR [rax], rdx
	mov rax, QWORD PTR -8[rbp]
	pop rbp
	ret
```




*2nd += xs[1st]  rax: = 1st

rax += 2nd

rdx: = *2nd.  rax: = xs[1st]

rdx: = *2nd + xs[1st] = A

*2nd = A  txs [1st]

rax: = xs [1st arg]

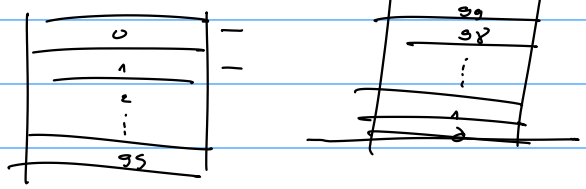
```
struct A {
    long xs [3];
};

long foo (struct A a, long idx, long *p) {
    long val = a.xs [idx];
    *p += val;
    return val;
}
```

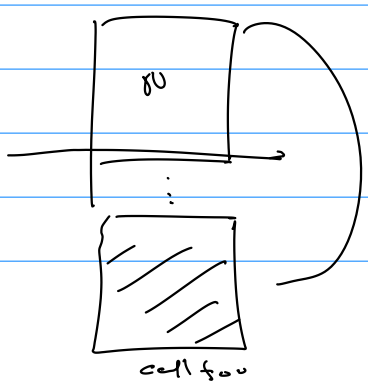
int xs[100]  struct X {

foo (xs)  int xs [100];

} s;

400, foo(s)



```
        .intel_syntax noprefix
        .text
        .globl foo
        .type foo, @function
foo:
        push rbp
        mov rbp, rsp
        sub rsp, 16
        mov QWORD PTR -8[rbp], rdi
        cmp QWORD PTR -8[rbp], 0
        je label_1
        mov rax, QWORD PTR -8[rbp]
        sub rax, 1
        push QWORD PTR 88[rbp]
        push QWORD PTR 80[rbp]
        push QWORD PTR 72[rbp]
        push QWORD PTR 64[rbp]
        push QWORD PTR 56[rbp]
        push QWORD PTR 48[rbp]
        push QWORD PTR 40[rbp]
        push QWORD PTR 32[rbp]
        push QWORD PTR 24[rbp]
        push QWORD PTR 16[rbp]
        mov rdi, rax
        call foo
        add rsp, 80
        # imul reg64, mem64
        # signed multiply
        # reg64 ← truncate(reg64 * mem64)
        imul rax, QWORD PTR -8[rbp]
        jmp label_2
label_1:
        mov rax, QWORD PTR 16[rbp]
label_2:
        leave
        ret
```

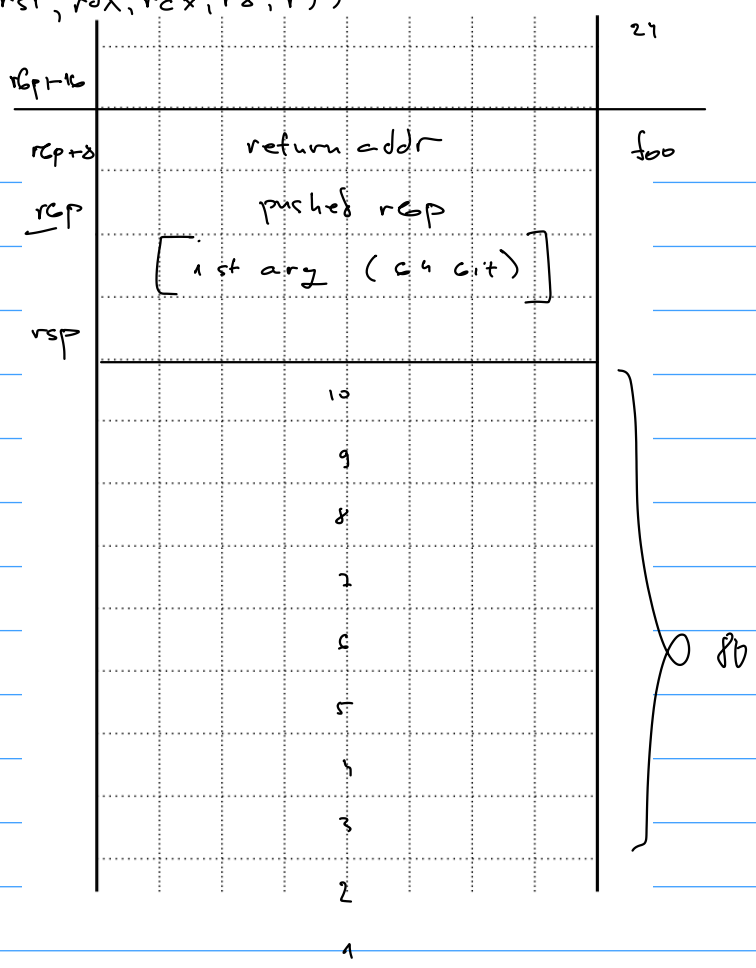
if (1st == 0) goto label

10 x 8

label:

foo(n-1,

(rdi, rsi, rdx, rcx, r8, r9)




```
struct A {
    long xs[10];
};

long foo(long n, struct A a) {
    if (n == 0) {
        return a.xs[0];
    } else {
        return n * foo(n-1, a);
    }
}
```