

# Example

```

long arith
(long x, long y, long z)
{
    long t1 = x+y;
    long t2 = z+t1;
    long t3 = x+4;
    long t4 = y * 48;
    long t5 = t3 + t4;
    long rval = t2 * t5;
    return rval;
}

```

arith:

```

leaq    (%rdi,%rsi), %rax
addq    %rdx, %rax
leaq    (%rsi,%rsi,2), %rdx
salq    $4, %rdx
leaq    4(%rdi,%rdx), %rcx
imulq   %rcx, %rax
ret

```

## Interesting Instructions

- **leaq**: address computation
- **salq**: shift
- **imulq**: multiplication
  - But, only used once instead of twice

# Understanding arith

```

long arith
(long x, long y, long z)
{
    long t1 = x+y;
    long t2 = z+t1;
    long t3 = x+4;
    long t4 = y * 48;
    long t5 = t3 + t4;
    long rval = t2 * t5;
    return rval;
}

```

arith:

```

leaq    (%rdi,%rsi), %rax    # t1
addq    %rdx, %rax          # t2
leaq    (%rsi,%rsi,2), %rdx
salq    $4, %rdx           # t4
leaq    4(%rdi,%rdx), %rcx  # t5
imulq   %rcx, %rax         # rval
ret

```

| Register | Use(s)              |
|----------|---------------------|
| %rdi     | Argument <b>x</b>   |
| %rsi     | Argument <b>y</b>   |
| %rdx     | Argument <b>z</b>   |
| %rax     | <b>t1, t2, rval</b> |
| %rdx     | <b>t4</b>           |
| %rcx     | <b>t5</b>           |