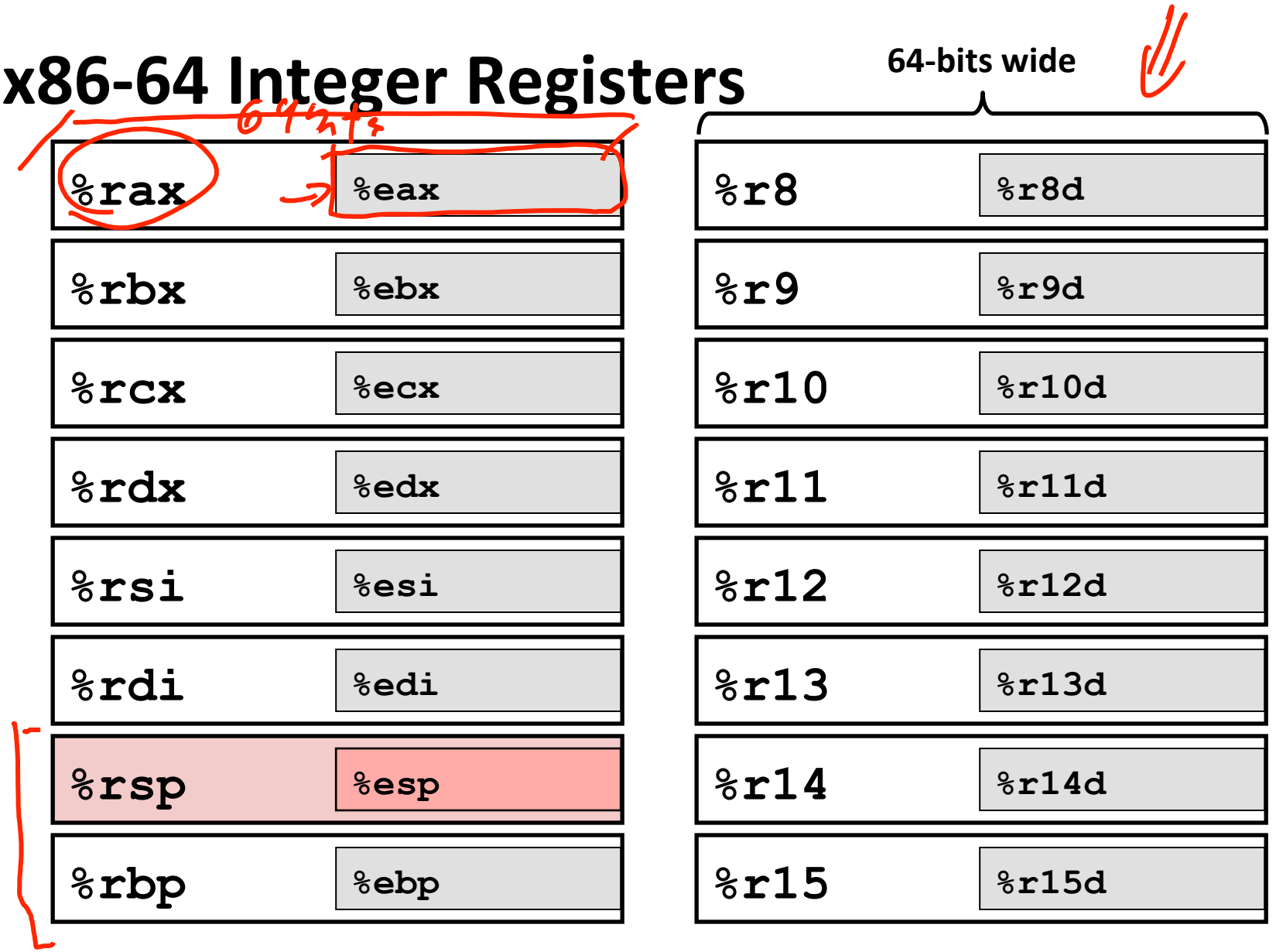
x86-64 Integer Registers



- Extend existing registers, and add 8 new ones; *all* accessible as 8, 16, 32, 64 bits.

32-bit vs. 64-bit operands

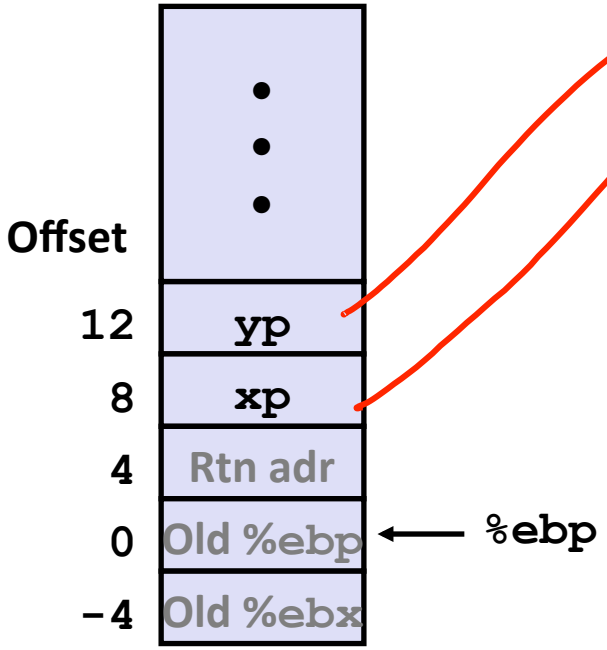
- **Long word `l` (4 Bytes) ↔ Quad word `q` (8 Bytes)**

- **New instruction forms:**
  - `movl` → `movq`
  - `addl` → `addq`
  - `sall` → `salq`
  - etc.

- **x86-64 can still use 32-bit instructions that generate 32-bit results**
  - Higher-order bits of destination register are just set to 0
  - Example: `addl`

Swap Ints in 32-bit Mode

```
void swap(int *xp, int *yp)
{
  int t0 = *xp;
  int t1 = *yp;
  *xp = t1;
  *yp = t0;
}
```




```
swap:
    pushl %ebp
    movl  %esp,%ebp
    pushl %ebx

    movl 12(%ebp),%ecx
    movl 8(%ebp),%edx
    movl (%ecx),%eax
    movl (%edx),%ebx
    movl %eax,(%edx)
    movl %ebx,(%ecx)

    movl -4(%ebp),%ebx
    movl %ebp,%esp
    popl %ebp
    ret
```

Setup

Body

Finish

Swap Ints in 64-bit Mode

```
void swap(int *xp, int *yp)
{
  int t0 = *xp;
  int t1 = *yp;
  *xp = t1;
  *yp = t0;
}
```

```
swap:
    movl    (%rdi), %edx
    movl    (%rsi), %eax
    movl    %eax, (%rdi)
    movl    %edx, (%rsi)
    retq
```

- **Arguments passed in registers (why useful?)**
  - First (**xp**) in **%rdi**, second (**yp**) in **%rsi**
  - 64-bit pointers
- **No stack operations required**
- **32-bit data**
  - Data held in registers **%eax** and **%edx**
  - **movl** operation (the **l** refers to data width, not address width)

Swap Long Ints in 64-bit Mode

```
void swap_l
   (long int *xp, long int *yp)
{
   long int t0 = *xp;
   long int t1 = *yp;
   *xp = t1;
   *yp = t0;
}
```

```
swap_l:
    movq    (%rdi), %rdx
    movq    (%rsi), %rax
    movq    %rax, (%rdi)
    movq    %rdx, (%rsi)
    retq
```

- **64-bit data**
  - Data held in registers **%rax** and **%rdx**
  - **movq** operation
  - "q" stands for quad-word