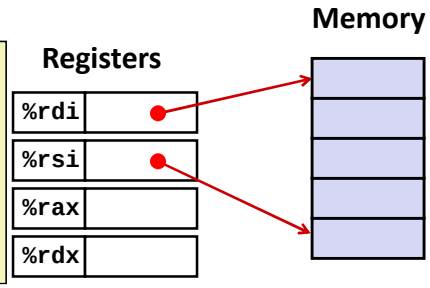
Understanding Swap()

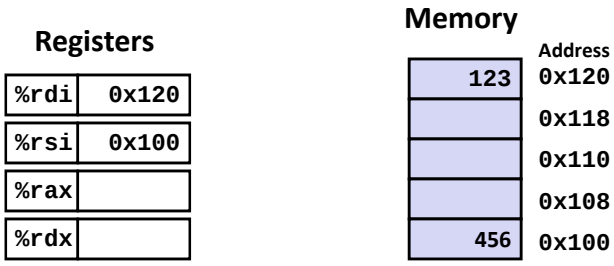
```
void swap
   (long *xp, long *yp)
{
  long t0 = *xp;
  long t1 = *yp;
  *xp = t1;
  *yp = t0;
}
```



Register	Value
%rdi	xp
%rsi	yp
%rax	t0
%rdx	t1

```
swap:
   movq    (%rdi), %rax  # t0 = *xp
   movq    (%rsi), %rdx  # t1 = *yp
   movq    %rdx, (%rdi)  # *xp = t1
   movq    %rax, (%rsi)  # *yp = t0
   ret
```


Understanding Swap()




```
swap:
   movq    (%rdi), %rax  # t0 = *xp
   movq    (%rsi), %rdx  # t1 = *yp
   movq    %rdx, (%rdi)  # *xp = t1
   movq    %rax, (%rsi)  # *yp = t0
   ret
```

Complete Memory Addressing Modes

- **Remember, the addresses used for accessing memory in `mov` (and other) instructions can be computed in several different ways**
- **Most General Form:**

  D(Rb,Ri,S) **Mem[Reg[Rb] + S*Reg[Ri] + D]**

  - D: Constant "displacement" value represented in 1, 2, or 4 bytes
  - Rb: Base register: Any of the 16 integer registers
  - Ri: Index register: Any, except for `%rsp`
  - S: Scale: 1, 2, 4, or 8 (*why these numbers?*)
- **Special Cases: can use any combination of D, Rb, Ri and S**

  (Rb,Ri) **Mem[Reg[Rb]+Reg[Ri]]** **(S=1, D=0)**

  D(Rb,Ri) **Mem[Reg[Rb]+Reg[Ri]+D]** **(S=1)**

  (Rb,Ri,S) **Mem[Reg[Rb]+S*Reg[Ri]]** **(D=0)**

Address Computation Examples

`%rdx`	`0xf000`
`%rcx`	`0x0100`

(Rb,Ri) **Mem[Reg[Rb]+Reg[Ri]]**
D(,Ri,S) **Mem[S*Reg[Ri]+D]**
(Rb,Ri,S) **Mem[Reg[Rb]+S*Reg[Ri]]**
D(Rb) **Mem[Reg[Rb] +D]**

Expression	Address Computation	Address
`0x8(%rdx)`		
`(%rdx,%rcx)`		
`(%rdx,%rcx,4)`		
`0x80(,%rdx,2)`		