*Assembly Language Lecture Series:*

# x86-64 Shift Instructions

Sensei RL Uy, College of Computer Studies,
De La Salle University, Manila, Philippines

# Copyright Notice

# x86-64 Shift Instructions

1. **SHL / SAL**
   shift left/shift arithmetic left
2. **SHR**
   shift right
3. **SAR**
   shift arithmetic right
4. **SHLD**
   Double precision shift left
5. **SHRD**
   Double precision shift right

# x86-64 Shift Instructions: SHL/SAL

## SHL/SAL (shift left /shift arithmetic left)

**Syntax: SHL/SAL dst, count**

dst ← dst << count

*dst = r/m

*count = 1, CL or imm8

*count is masked to 5 bits (32-bit)

*count is masked to 6 bits (64-bit)

**Flags affected:**

*PF, SF, ZF

*CF  contains the value of the last bit shifted out; it is undefined if count >= to dst size (in bits)

*OF = 0 if MSB == CF(for 1-bit shift) else undefined

*AF – undefined

*all status flags no change: if count is 0

# x86-64 Shift Instructions: SHL/SAL

## SHL/SAL (shift left /shift arithmetic left)

**Syntax: SHL/SAL dst, count**

dst ← dst << count

*dst = r/m

*count = 1, CL or imm8

*count is masked to 5 bits (32-bit)

*count is masked to 6 bits (64-bit)

**Flags affected:**

*PF, SF, ZF

*CF  contains the value of the last bit shifted out; it is undefined if count >= to dst size (in bits)

*OF = 0 if MSB == CF(for 1-bit shift) else undefined

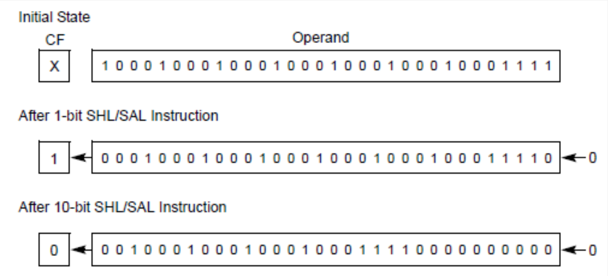
*AF – undefined

*all status flags no change: if count is 0

## Example:

```
section .text
MOV RAX, 0x0000_0000_0000_0001
SHL RAX, 63
```

1. **What will RAX contain after execution?**
2. **What will SF, ZF, PF, CF contain after execution?**



# x86-64 Shift Instructions: SHL/SAL

## SHL/SAL (shift left /shift arithmetic left)

**Syntax: SHL/SAL dst, count**

dst ← dst << count

*dst = r/m

*count = 1, CL or imm8

*count is masked to 5 bits (32-bit)

*count is masked to 6 bits (64-bit)

**Flags affected:**

*PF, SF, ZF

*CF  contains the value of the last bit shifted out; it is undefined if count >= to dst size (in bits)

*OF = 0 if MSB == CF(for 1-bit shift) else undefined

*AF – undefined

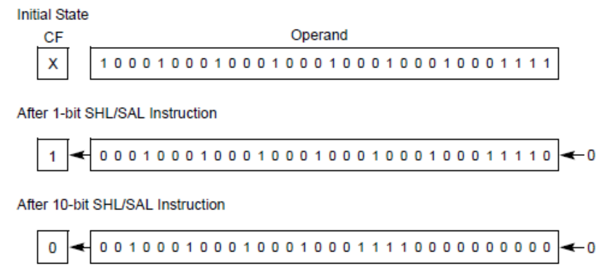
*all status flags no change: if count is 0

## Example:

```
section .text
MOV RAX, 0x0000_0000_0000_0001
SHL RAX, 63
```

1. What will RAX contain after execution?
2. What will SF, ZF, PF, CF contain after execution?

**RAX = 8000_0000_0000_0000**
**CF=0, PF=1, SF=1, ZF=0**



# x86-64 Shift Instructions: SHL/SAL

## SHL/SAL (shift left /shift arithmetic left)

**Syntax: SHL/SAL dst, count**

dst ← dst << count

*dst = r/m

*count = 1, CL or imm8

*count is masked to 5 bits (32-bit)

*count is masked to 6 bits (64-bit)

**Flags affected:**

*PF, SF, ZF

*CF contains the value of the last bit shifted out; it is undefined if count >= to dst size (in bits)

*OF = 0 if MSB == CF(for 1-bit shift) else undefined

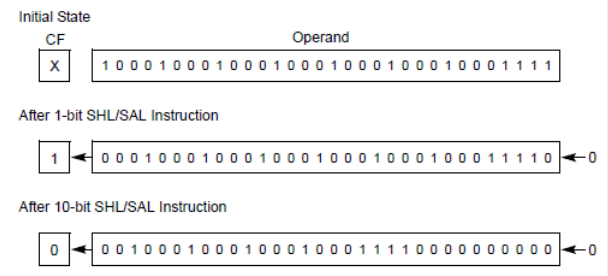
*AF – undefined

*all status flags no change: if count is 0

## Example:

```
section .text
MOV EAX, 0x0000_0001
SHL EAX, 31
```

1. **What will EAX contain after execution?**
2. **What will SF, ZF, PF, CF contain after execution?**



# x86-64 Shift Instructions: SHL/SAL

## SHL/SAL (shift left /shift arithmetic left)

**Syntax: SHL/SAL dst, count**

dst ← dst << count

*dst = r/m

*count = 1, CL or imm8

*count is masked to 5 bits (32-bit)

*count is masked to 6 bits (64-bit)

**Flags affected:**

*PF, SF, ZF

*CF contains the value of the last bit shifted out; it is undefined if count >= to dst size (in bits)

*OF = 0 if MSB == CF(for 1-bit shift) else undefined

*AF – undefined

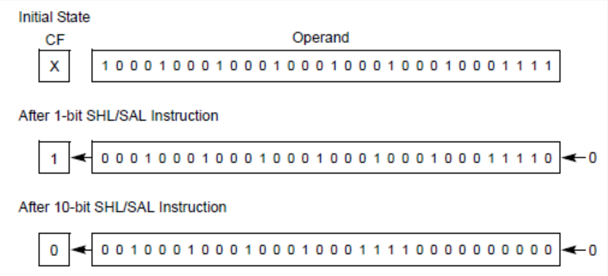
*all status flags no change: if count is 0

## Example:

```
section .text
MOV EAX, 0x0000_0001
SHL EAX, 31
```

1. What will EAX contain after execution?
2. What will SF, ZF, PF, CF contain after execution?

**EAX = 8000_0000**
**CF=0, PF=1, SF=1, ZF=0**



Initial State

CF: X

Operand: 1 0 0 0 1 0 0 0 1 0 0 0 1 0 0 0 1 0 0 0 1 0 0 0 1 0 0 0 1 1 1 1

After 1-bit SHL/SAL Instruction

1 ← 0 0 0 1 0 0 0 1 0 0 0 1 0 0 0 1 0 0 0 1 0 0 0 1 0 0 0 1 1 1 1 0 ← 0

After 10-bit SHL/SAL Instruction

0 ← 0 0 1 0 0 0 1 0 0 0 1 0 0 0 1 0 0 0 1 1 1 1 0 0 0 0 0 0 0 0 0 0 ← 0

# x86-64 Shift Instructions: SHR

## SHR (shift right)

**Syntax: SHR dst, count**
dst ← dst >> count
*dst = r/m
*count = 1, CL or imm8
*count is masked to 5 bits (32-bit)
*count is masked to 6 bits (64-bit)

**Flags affected:**
*PF, SF, ZF
*CF contains the value of the last bit shifted out; it is undefined if count >= to dst size (in bits)
*OF = MSB of the original operand (for 1-bit shift) else undefined
*AF – undefined
*all status flags no change: if count is 0

# x86-64 Shift Instructions: SHR

## SHR (shift right)

**Syntax: SHR dst, count**

dst ← dst >> count

*dst = r/m

*count = 1, CL or imm8

*count is masked to 5 bits (32-bit)

*count is masked to 6 bits (64-bit)

**Flags affected:**

*PF, SF, ZF

*CF contains the value of the last bit shifted out; it is undefined if count >= to dst size (in bits)

*OF = MSB of the original operand (for 1-bit shift) else undefined

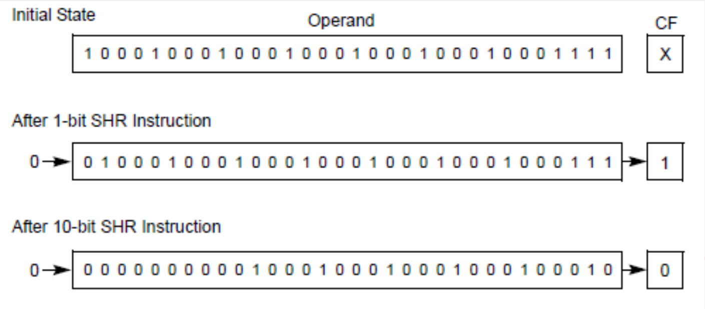
*AF – undefined

*all status flags no change: if count is 0

## Example:

```
section .text
MOV RAX, 0x8000_0000_0000_0000
SHR RAX, 63
```

1. **What will RAX contain after execution?**
2. **What will SF, ZF, PF, CF contain after execution?**



Initial State

Operand CF

1 0 0 0 1 0 0 0 1 0 0 0 1 0 0 0 1 0 0 0 1 0 0 0 1 0 0 0 1 1 1 1 X

After 1-bit SHR Instruction

0 → 0 1 0 0 0 1 0 0 0 1 0 0 0 1 0 0 0 1 0 0 0 1 0 0 0 1 0 0 0 1 1 1 → 1

After 10-bit SHR Instruction

0 → 0 0 0 0 0 0 0 0 0 0 1 0 0 0 1 0 0 0 1 0 0 0 1 0 0 0 1 0 0 0 1 0 → 0

# x86-64 Shift Instructions: SHR

## SHR (shift right)

**Syntax: SHR dst, count**

dst ← dst >> count

*dst = r/m

*count = 1, CL or imm8

*count is masked to 5 bits (32-bit)

*count is masked to 6 bits (64-bit)

**Flags affected:**

*PF, SF, ZF

*CF contains the value of the last bit shifted out; it is undefined if count >= to dst size (in bits)

*OF = MSB of the original operand (for 1-bit shift) else undefined

*AF – undefined

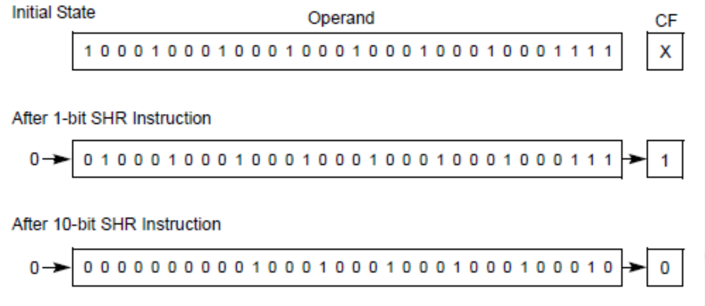
*all status flags no change: if count is 0

## Example:

```
section .text
MOV RAX, 0x8000_0000_0000_0000
SHR RAX, 63
```

1. What will RAX contain after execution?
2. What will SF, ZF, PF, CF contain after execution?

**RAX = 0000_0000_0000_0001**
**CF=0, PF=0, SF=0, ZF=0**



# x86-64 Shift Instructions: SHR

## SHR (shift right)

**Syntax: SHR dst, count**

dst ← dst >> count

*dst = r/m

*count = 1, CL or imm8

*count is masked to 5 bits (32-bit)

*count is masked to 6 bits (64-bit)

**Flags affected:**

*PF, SF, ZF

*CF contains the value of the last bit shifted out; it is undefined if count >= to dst size (in bits)

*OF = MSB of the original operand (for 1-bit shift) else undefined

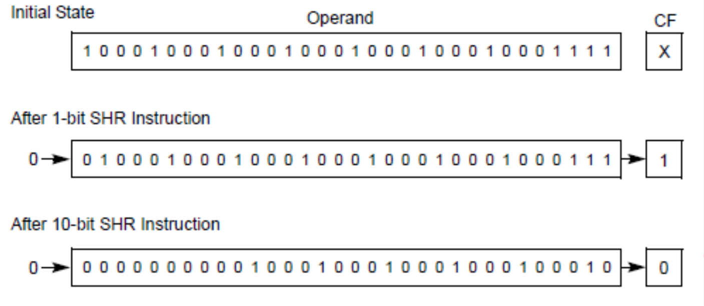
*AF – undefined

*all status flags no change: if count is 0

## Example:

```
section .text
MOV EAX, 0x8000_0000
SHR EAX, 31
```

1. **What will EAX contain after execution?**
2. **What will SF, ZF, PF, CF contain after execution?**



# x86-64 Shift Instructions: SHR

## SHR (shift right)

**Syntax: SHR dst, count**

dst ← dst >> count

*dst = r/m

*count = 1, CL or imm8

*count is masked to 5 bits (32-bit)

*count is masked to 6 bits (64-bit)

**Flags affected:**

*PF, SF, ZF

*CF contains the value of the last bit shifted out; it is undefined if count >= to dst size (in bits)

*OF = MSB of the original operand (for 1-bit shift) else undefined

*AF – undefined

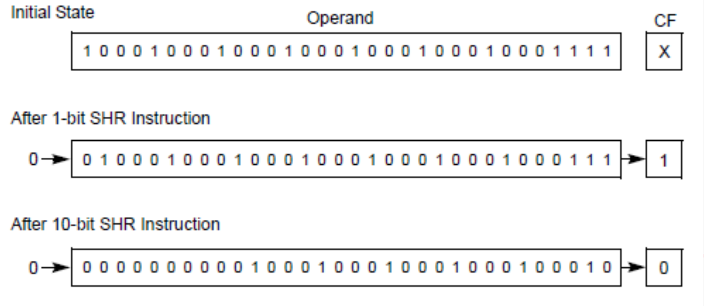
*all status flags no change: if count is 0

## Example:

```
section .text
MOV EAX, 0x8000_0000
SHR EAX, 31
```

1. What will EAX contain after execution?
2. What will SF, ZF, PF, CF contain after execution?

**EAX = 0000_0001**
**CF=0, PF=0, SF=0, ZF=0**



# x86-64 Shift Instructions: SAR

## SAR (shift arithmetic right)

**Syntax: SAR dst, count**

dst ← dst $>>_{\text{arithmetic}}$ count

*dst = r/m

*count = 1, CL or imm8

*count is masked to 5 bits (32-bit)

*count is masked to 6 bits (64-bit)

**Flags affected:**

*PF, SF, ZF

*CF contains the value of the last bit shifted out; it is undefined if count >= to dst size (in bits)

*OF = 0 (for 1-bit shift) else undefined

*AF – undefined

*all status flags no change: if count is 0

# x86-64 Shift Instructions: SAR

## SAR (shift arithmetic right)

**Syntax: SAR dst, count**

dst ← dst $>>_{\text{arithmetic}}$ count

*dst = r/m

*count = 1, CL or imm8

*count is masked to 5 bits (32-bit)

*count is masked to 6 bits (64-bit)

**Flags affected:**

*PF, SF, ZF

*CF contains the value of the last bit shifted out; it is undefined if count >= to dst size (in bits)

*OF = 0 (for 1-bit shift) else undefined

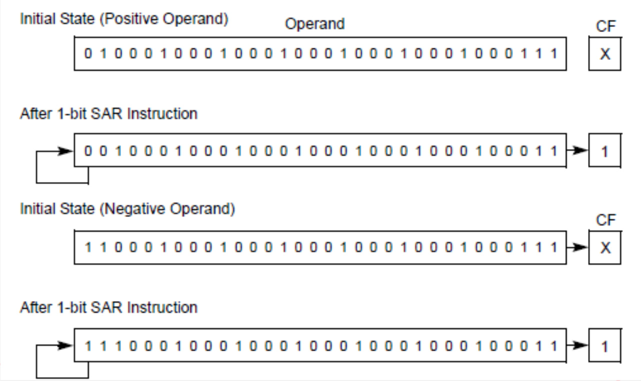
*AF – undefined

*all status flags no change: if count is 0

## Example:

```
section .text
MOV RAX, 0x8000_0000_0000_0000
SAR RAX, 63
```

1. **What will RAX contain after execution?**
2. **What will SF, ZF, PF, CF contain after execution?**



# x86-64 Shift Instructions: SAR

## SAR (shift arithmetic right)

**Syntax: SAR dst, count**

dst ← dst $>>_{\text{arithmetic}}$ count

*dst = r/m

*count = 1, CL or imm8

*count is masked to 5 bits (32-bit)

*count is masked to 6 bits (64-bit)

**Flags affected:**

*PF, SF, ZF

*CF contains the value of the last bit shifted out; it is undefined if count >= to dst size (in bits)

*OF = 0 (for 1-bit shift) else undefined

*AF – undefined

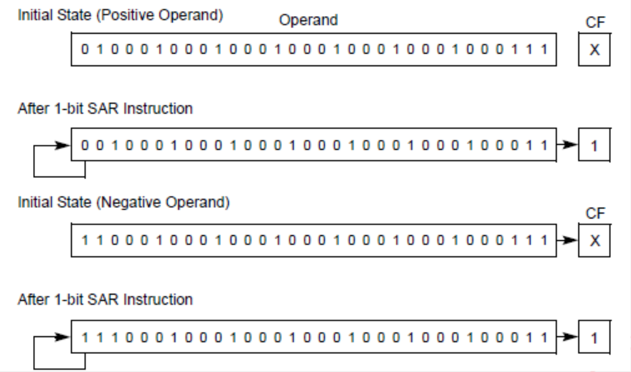
*all status flags no change: if count is 0

## Example:

```
section .text
MOV RAX, 0x8000_0000_0000_0000
SAR RAX, 63
```

1. What will RAX contain after execution?
2. What will SF, ZF, PF, CF contain after execution?

**RAX = FFFF_FFFF_FFFF_FFFF**
**CF=0, PF=1, SF=1, ZF=0**



Initial State (Positive Operand)

0 1 0 0 0 1 0 0 0 1 0 0 0 1 0 0 0 1 0 0 0 1 0 0 0 1 0 0 0 1 1 1

After 1-bit SAR Instruction

0 0 1 0 0 0 1 0 0 0 1 0 0 0 1 0 0 0 1 0 0 0 1 0 0 0 1 0 0 0 1 1

Initial State (Negative Operand)

1 1 0 0 0 1 0 0 0 1 0 0 0 1 0 0 0 1 0 0 0 1 0 0 0 1 0 0 0 1 1 1

After 1-bit SAR Instruction

1 1 1 0 0 0 1 0 0 0 1 0 0 0 1 0 0 0 1 0 0 0 1 0 0 0 1 0 0 0 1 1

# x86-64 Shift Instructions: SAR

## SAR (shift arithmetic right)

**Syntax: SAR dst, count**

dst ← dst $>>_{\text{arithmetic}}$ count

*dst = r/m

*count = 1, CL or imm8

*count is masked to 5 bits (32-bit)

*count is masked to 6 bits (64-bit)

**Flags affected:**

*PF, SF, ZF

*CF  contains the value of the last bit shifted out; it is undefined if count >= to dst size (in bits)

*OF = 0 (for 1-bit shift) else undefined

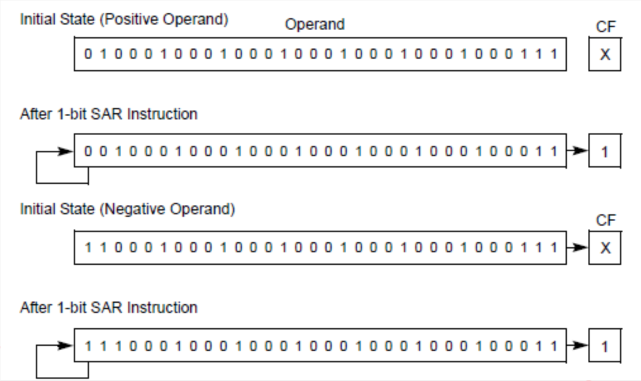
*AF – undefined

*all status flags no change: if count is 0

## Example:

```
section .text
MOV EAX, 0x8000_0000
SAR EAX, 31
```

1. **What will EAX contain after execution?**
2. **What will SF, ZF, PF, CF contain after execution?**



# x86-64 Shift Instructions: SAR

## SAR (shift arithmetic right)

**Syntax: SAR dst, count**

dst ← dst $>>_{arithmetic}$ count

*dst = r/m

*count = 1, CL or imm8

*count is masked to 5 bits (32-bit)

*count is masked to 6 bits (64-bit)

**Flags affected:**

*PF, SF, ZF

*CF contains the value of the last bit shifted out; it is undefined if count >= to dst size (in bits)

*OF = 0 (for 1-bit shift) else undefined

*AF – undefined

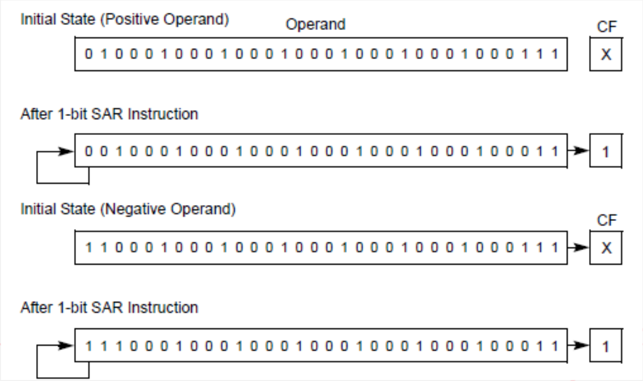
*all status flags no change: if count is 0

## Example:

```
section .text
MOV EAX, 0x8000_0000
SAR EAX, 31
```

1. **What will EAX contain after execution?**
2. **What will SF, ZF, PF, CF contain after execution?**

**EAX = FFFF_FFFF**
**CF=0, PF=1, SF=1, ZF=0**



# x86-64 Shift Instructions: SHLD

## SHLD (Double precision shift left)

**Syntax: SHLD dst,src,count**

dst ← src << count

*dst = [r/m]_16_32_64

*src = r16_32_64

*count = 1, CL or imm8

*count is masked to 5 bits (32-bit)

*count is masked to 6 bits (64-bit)

**Flags affected:**

*PF, SF, ZF

*CF contains the value of the last bit shifted out

*OF = 1 if sign change occurred (for 1-bit shift) else undefined

*AF – undefined

*all status flags no change: if count is 0

*all status flags undefined: if count >= dst size (in bits)

# x86-64 Shift Instructions: SHLD

## SHLD (Double precision shift left)

**Syntax: SHLD dst,src,count**

dst ← src << count

*dst = [r/m]_16_32_64

*src = r16_32_64

*count = 1, CL or imm8

*count is masked to 5 bits (32-bit)

*count is masked to 6 bits (64-bit)

**Flags affected:**

*PF, SF, ZF

*CF  contains the value of the last bit shifted out

*OF = 1 if sign change occurred (for 1-bit shift) else undefined

*AF – undefined

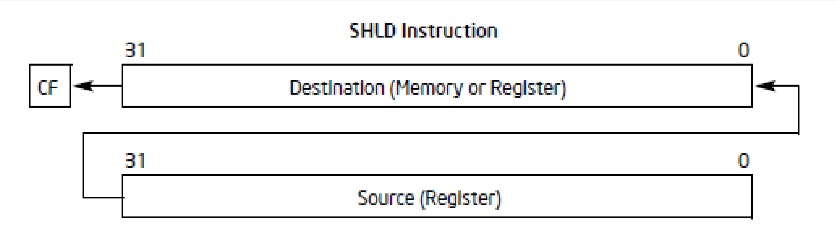
*all status flags no change: if count is 0

*all status flags undefined:
if count >= dst size (in bits)

## Example:

```
section .text
mov RAX,0x1234_5678_8765_4321
mov RBX,0xABCD_EFCD_DCFE_DCBA
SHLD RAX,RBX,32
```

1. **What will RAX contain after execution?**
2. **What will RBX contain after execution**
3. **What will SF, ZF, PF, CF contain after execution?**



# x86-64 Shift Instructions: SHLD

## SHLD (Double precision shift left)

**Syntax: SHLD dst,src,count**

dst ← src << count

*dst = [r/m]_16_32_64

*src = r16_32_64

*count = 1, CL or imm8

*count is masked to 5 bits (32-bit)

*count is masked to 6 bits (64-bit)

**Flags affected:**

*PF, SF, ZF

*CF contains the value of the last bit shifted out

*OF = 1 if sign change occurred (for 1-bit shift) else undefined

*AF – undefined

*all status flags no change: if count is 0

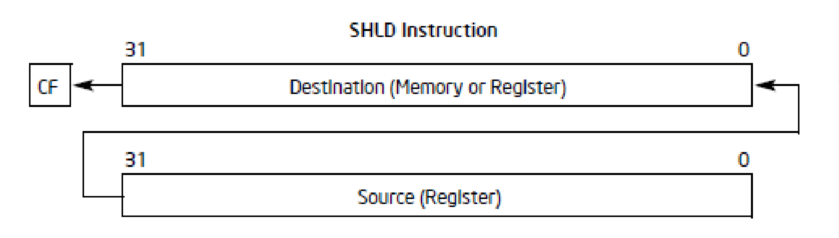
*all status flags undefined: if count >= dst size (in bits)

## Example:

```
section .text
mov RAX,0x1234_5678_8765_4321
mov RBX,0xABCD_EFCD_DCFE_DCBA
SHLD RAX,RBX,32
```

1. What will RAX contain after execution?
2. What will RBX contain after execution
3. What will SF, ZF, PF, CF contain after execution?

**RAX = 8765_4321_ABCD_EFCD**
**RBX = ABCD_EFCD_DCFE_DCBA**
**CF=0, PF=0, SF=1, ZF=0**



# x86-64 Shift Instructions: SHRD

## SHRD (Double precision shift right)

**Syntax: SHRD dst,src,count**

dst ← src >> count

*dst = [r/m]_16_32_64

*src = r16_32_64

*count = 1, CL or imm8

*count is masked to 5 bits (32-bit)

*count is masked to 6 bits (64-bit)

**Flags affected:**

*PF, SF, ZF

*CF contains the value of the last bit shifted out

*OF = 1 if sign change occurred (for 1-bit shift) else undefined

*AF – undefined

*all status flags no change: if count is 0

*all status flags undefined: if count >= dst size (in bits)

# x86-64 Shift Instructions: SHRD

## SHRD (Double precision shift right)

**Syntax: SHRD dst,src,count**

dst ← src << count

*dst = [r/m]_16_32_64

*src = r16_32_64

*count = 1, CL or imm8

*count is masked to 5 bits (32-bit)

*count is masked to 6 bits (64-bit)

**Flags affected:**

*PF, SF, ZF

*CF  contains the value of the last bit shifted out

*OF = 1 if sign change occurred (for 1-bit shift) else undefined

*AF – undefined

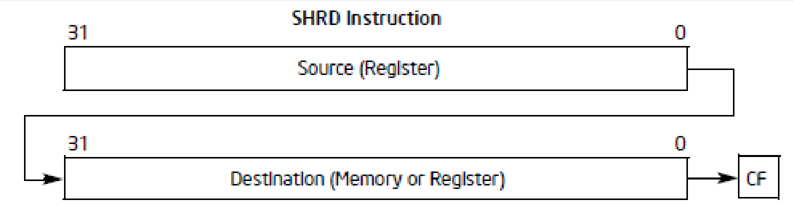
*all status flags no change: if count is 0

*all status flags undefined: if count >= dst size (in bits)

## Example:

```
section .text
mov RAX,0x1234_5678_8765_4321
mov RBX,0xABCD_EFCD_DCFE_DCBA
SHRD RAX,RBX,32
```

1. **What will RAX contain after execution?**
2. **What will RBX contain after execution**
3. **What will SF, ZF, PF, CF contain after execution?**



# x86-64 Shift Instructions: SHRD

## SHRD (Double precision shift right)

**Syntax: SHRD dst,src,count**

dst ← src >> count

*dst = [r/m]_16_32_64

*src = r16_32_64

*count = 1, CL or imm8

*count is masked to 5 bits (32-bit)

*count is masked to 6 bits (64-bit)

**Flags affected:**

*PF, SF, ZF

*CF contains the value of the last bit shifted out

*OF = 1 if sign change occurred (for 1-bit shift) else undefined

*AF – undefined

*all status flags no change: if count is 0

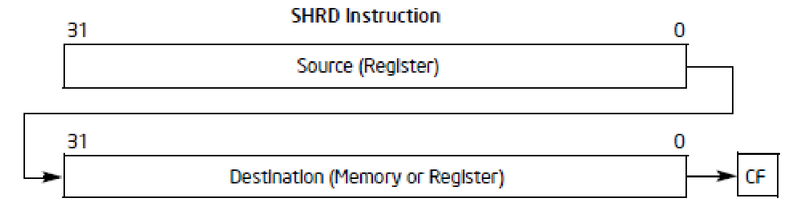
*all status flags undefined: if count >= dst size (in bits)

## Example:

```
section .text
mov RAX,0x1234_5678_8765_4321
mov RBX,0xABCD_EFCD_DCFE_DCBA
SHRD RAX,RBX,32
```

1. What will RAX contain after execution?
2. What will RBX contain after execution
3. What will SF, ZF, PF, CF contain after execution?

**RAX = DCFE_DCBA_1234_5678**
**RBX = ABCD_EFCD_DCFE_DCBA**
**CF=1, PF=1, SF=1, ZF=0**


SHRD Instruction
31
0
Source (Register)
31
0
Destination (Memory or Register)
CF