Низькорівневе програмування та програмування мікроконтролерів

З використанням мови програмування **Assembler**

Бітові операції

класрум **im3ozqq4**

Boolean Operations in i386 Assembly

Introduction to bitwise logic at machine level.
Core tools:

AND, OR, XOR, NOT,
SHR, SHL, SAR, SAL,
ROR, ROL, RCR, RCL

Binary System Refresher

Reminder: each register is a sequence of bits, each bit being 0 or 1.

42							
0	0	1	0	1	0	1	0
2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
128	64	32	16	8	4	2	1

Truth Tables Overview

Truth tables show how input bits map to outputs for each operation.

not, !

a	!a
true	false
false	true

or, a || b

a	b	a||b
false	false	false
false	true	true
true	false	true
true	true	true

and, a && b

a	b	a&&b
false	false	false
false	true	false
true	false	false
true	true	true

xor, a ^ b

a	b	a^b
false	false	false
false	true	true
true	false	true
true	true	false

Why Bitwise Operations Matter

Applications: masking, flags, toggling, bit inversion in real systems.

i386 Boolean Instructions Overview

Instructions:
AND,
OR, Two-operand
XOR

NOT. one-operand

“Normal language” Boolean Operations

```
let a = b | c      OR
let a = b & c      AND
let a = b ^ c      XOR
let a = !b          NOT
```

“mapping” to Assembly

let a = b | c	**OR**	let b = b | c
let a = b & c	**AND**	let b = b & c
let a = b ^ c	**XOR**	let b = b ^ c
let a = !b	**NOT**	let b = !b

```
op  dst, src
```

```
dst = dst ⊕ src
```

Operands in i386

Allowed AND, OR, XOR:

- reg, reg
- reg, mem
- mem, reg
- reg, imm

Allowed NOT

- mem
- reg

```
and ax, bx
and ax, [esi]
and [edi], bx
and ax, 7
```

```
not ax
not [ebx]
```

Instruction Encoding Principles

Пам’ять не може бути src & dst

Не працює

NOT – Концепція

Унарний оператор. Інверсія

```
mov ax, 0xF0
not ax
```

```
rax : 0xf0
```

```
rax : 0xff0f
```

NOT – реальний кейс

Підготовка маски

```
mov bx, 0b1111
not bx
```

```
rbx : 0x0f
```

```
rbx : 0xfff0
```

AND – Концепція

Виконання логічної операції AND для кожного біта в регістрі

```
mov ax, 0xF1
mov bx, 0xC5
and ax, bx
```

```
rax : 0xf1
rbx : 0xc5
```

```
rax : 0xc1
```

AND – реальний приклад

```
mov ax, 0x01010101
mov bx, 0x00001111
and ax, bx
```

```
rax : 0x0101
```

OR – Концепція

Виконання логічної операції OR для кожного біта в регістрі

```
mov eax, 0x01010101
mov ebx, 0x00001111
or eax, ebx
```

```
rax : 0x01011111
```

OR – Реальний приклад

```
mov ax, 0xF000
mov bx, 0x0F00
mov cx, 0x00F0
mov dx, 0x000F
or ax, bx
or ax, cx
or ax, dx
```

```
rax : 0xff00
```

```
rax : 0xfff0
```

```
rax : 0xffff
```

XOR – Концепція

Виконання логічної операції XOR для кожного біта в регістрі

```
mov ax, 0xFFFF
mov bx, 0x0011
xor ax, bx
```

```
rax : 0xffee
```

XOR – “Зворотня”

Виконавши “однакову операцію” 2 рази ми повертаємося
До наших початкових даних

```
mov ax, 0xFFFF
mov bx, 0x0011
xor ax, bx
xor ax, bx
```

```
rax : 0xffee
```

```
rax : 0xffff
```

XOR – обнулення регістру

Технічно можна написати

```
mov ax, 0
```

Але інколи, щоб підкреслити семантику, не значення 0,
а видалення даних

```
xor ax, ax
```

XOR – реальні кейси

Дуже багато використовується в
- криптографії
- шифруванні

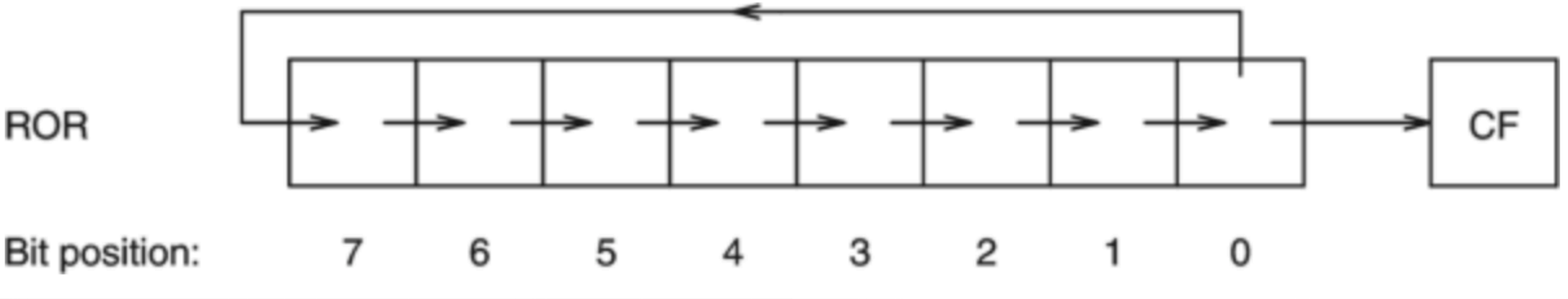
Bit Rotation

Процесор x86 має 8 команд зсуву регістрів

ROL, ROR
RCL, RCR
SHL, SAL
SHR, SAR

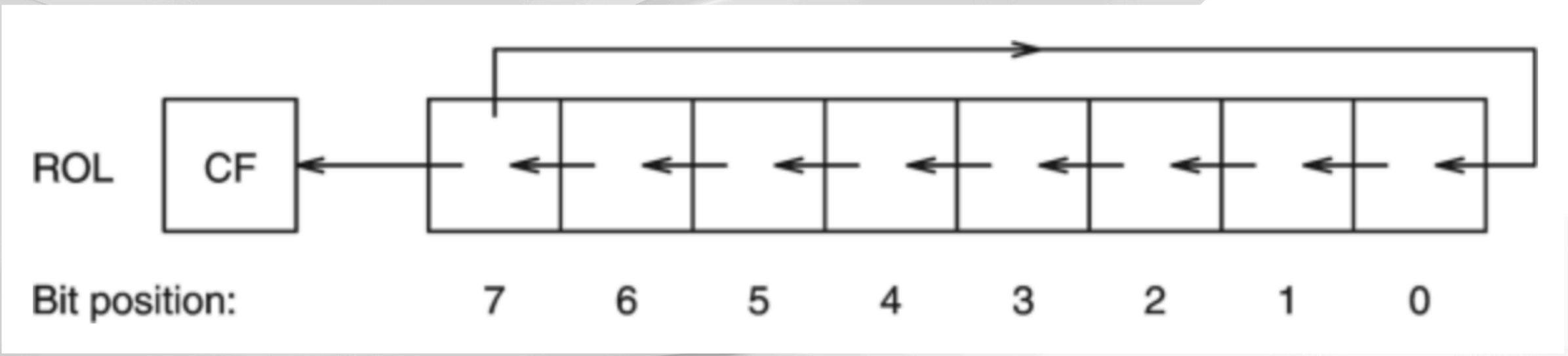
ROR

Циклічний зсув вправо, LSB → CF



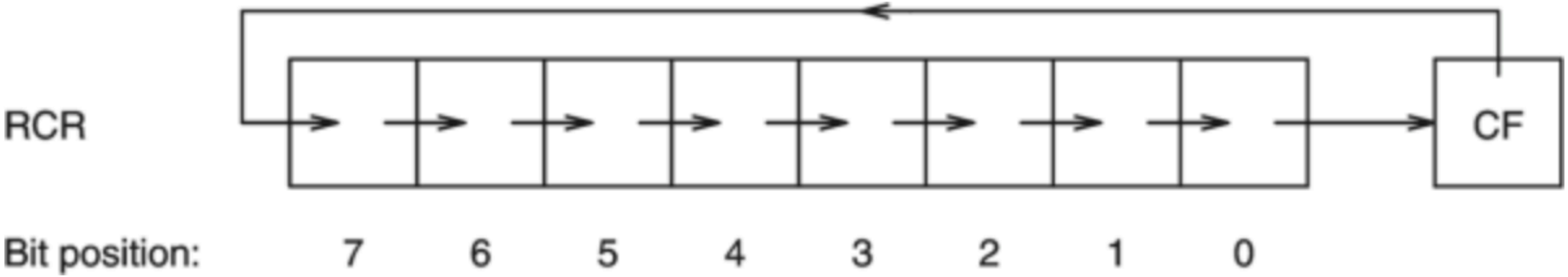
ROL

Циклічний зсув вліво, MSB → CF



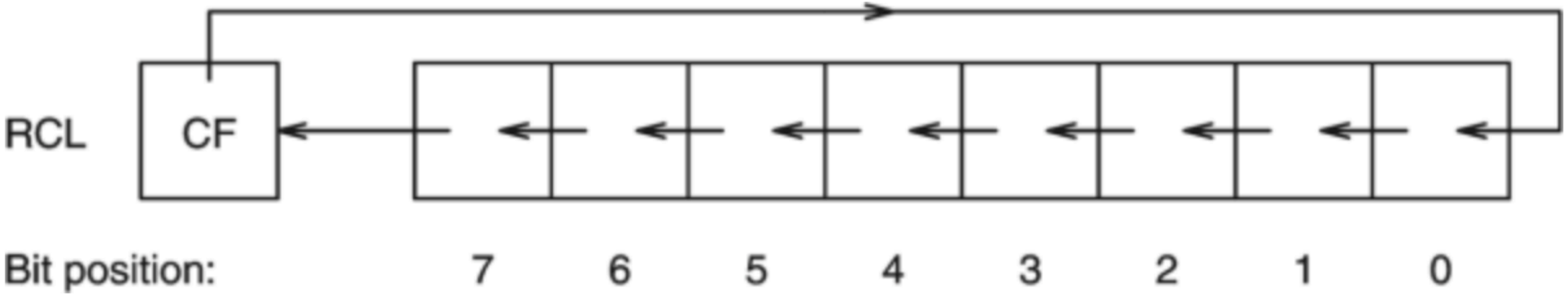
RCR

Зсув вправо через флаг **CF**



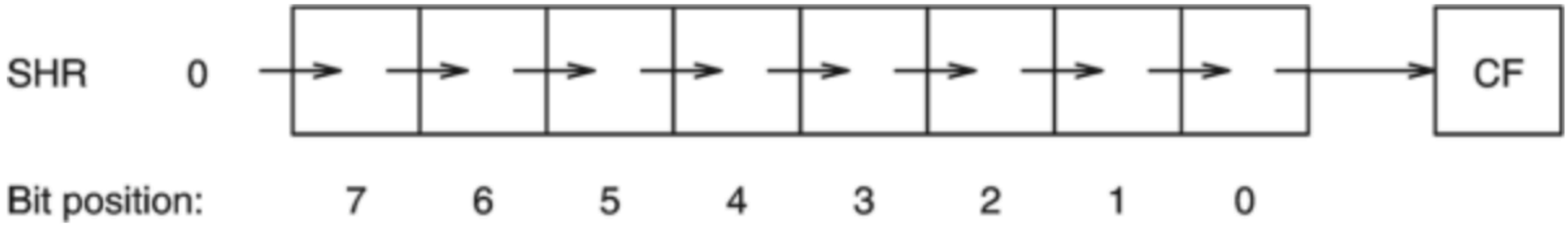
RCL

Зсув вліво через флаг **CF**



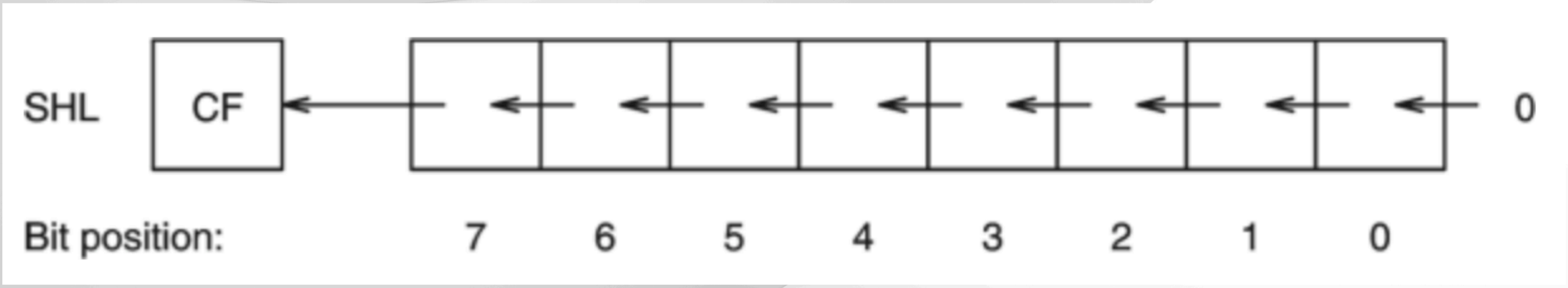
SHR

Зсув вправо
MSB = 0, LSB → CF



SHL

Зсув вправо
MSB → CF, LSB = 0



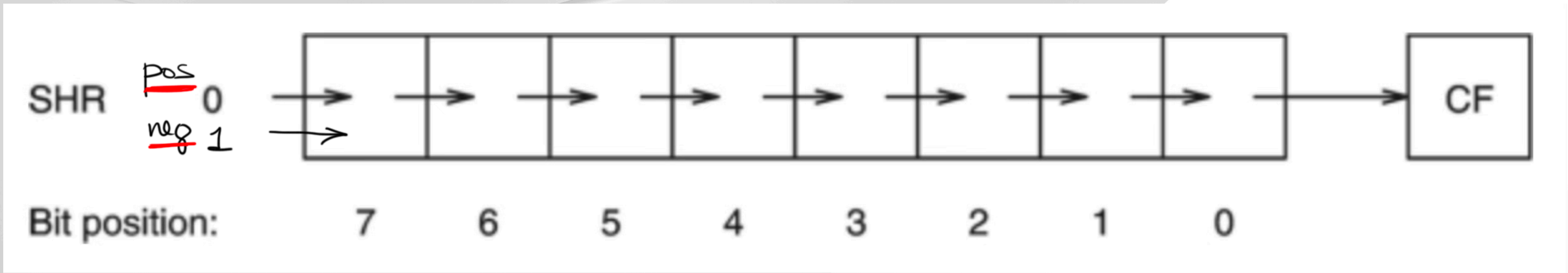
Представлення від’ємних чисел

```
  -3 11111101
  -2 11111110
  -1 11111111
   0 00000000
   1 00000001
   2 00000010
   3 00000011
   4 00000100
 ...........
 124 01111100
 125 01111101
 126 01111110
 127 01111111
-128 10000000
-127 10000001
-126 10000010
-125 10000011
```

https://en.wikipedia.org/wiki/Two's_complement

SAR

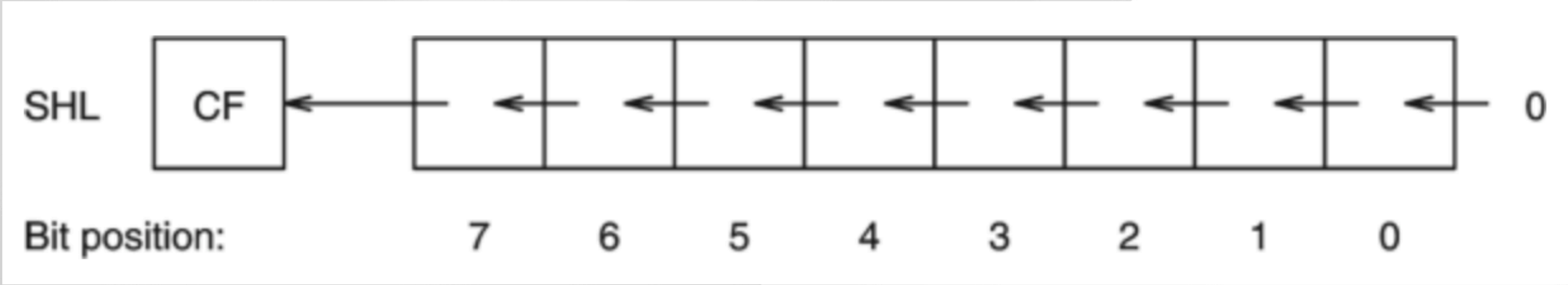
Арифметичний зсув вправо



Для від’ємних чисел MSB = 1, LSB → CF
Для **не**від’ємних чисел MSB = 0, LSB → CF

SAL = SHL

Арифметичний зсув вліво — це те саме що звичайний
Оскільки знак не впливає на цю операцію



MSB → CF , LSB = 0

Розмір даних має значення

```
mov eax, 0x12345678
```

```
shr eax, 1
```

```
rax : 0x091a2b3c
```

```
shr ax, 1
```

```
rax : 0x12342b3c
```

```
shr ah, 1
```

```
rax : 0x12342b78
```

```
shr al, 1
```

```
rax : 0x1234563c
```

Розмір даних має значення

```
mov eax, 0x12345678
```

Абсолютно різні команди

```
shr eax, 1
```

```
rax : 0x091a2b3c
```

```
shr ax, 1
```

```
rax : 0x12342b3c
```

```
shr ah, 1
```

```
rax : 0x12342b78
```

```
shr al, 1
```

```
rax : 0x1234563c
```

Combining Boolean Operations

Chaining AND, OR, XOR, NOT for complex transformations.