Computer Organization and Assembly Language - Lab
Fall 2020

Faisal Khan: faisal.khan@nu.edu.pk

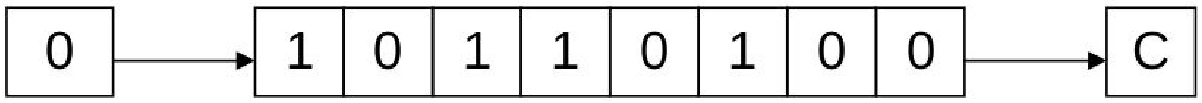
Bit Manipulation

Shifting and Rotations

Shift Logical Right (SHR)

In Shift Logical Right
- bits are moved one step to the right
- '0' is appended to the left most bit
- Bit value from the right most bit goes to the Carry Flag

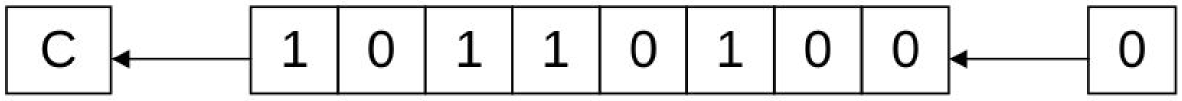
In Right Shifting, division by 2 takes place for unsigned integers.



Shift Logical Left (SHL) / Shift Arithmetic Left (SAL)

In Shift Logical Left
- Bits are moved one step to the left
- '0' is appended to the right most bit
- Bit value from left most bit goes to the Carry Flag

In Left Shifting, multiplication by 2 takes place.

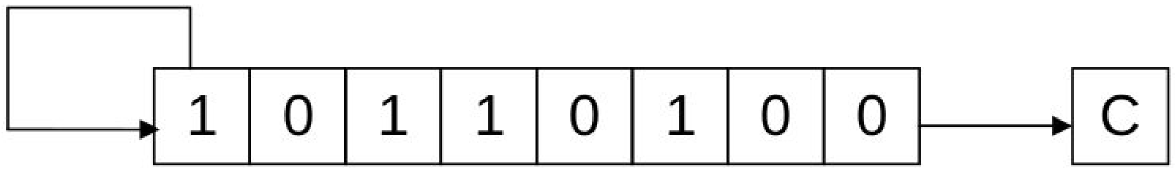


Shift Arithmetic Right (SAR)

In some cases, we may have a signed integer which will have a '1' in the most significant bit (left most bit). Shifting all bits will change the sign on the integer which is not right. So, in Shift Arithmetic Right

- All bits are moved one step to the right
- Former most significant bit's value is copied to new most significant bit
- Bit value from right most bit goes to the Carry Flag

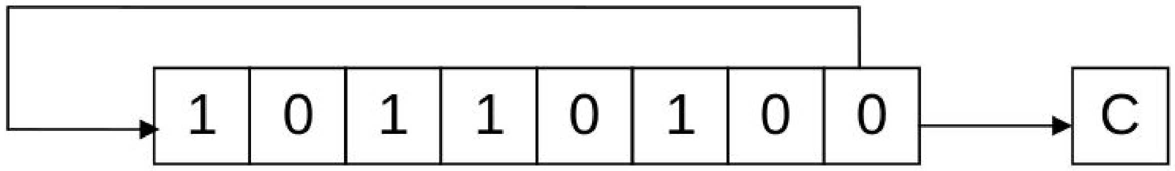
In Arithmetic Right Shifting, division by 2 takes place for signed integers.



Rotate Right (ROR)

In Rotate Right

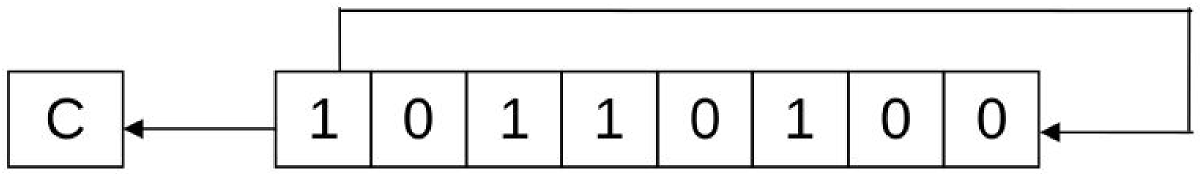
- Every bit moves one place to the right
- Bit that is dropped from the right most is inserted back at the left most bit.
- Same dropped bit is copied to Carry Flag as well



Rotate Left (ROL)

In Rotate Left

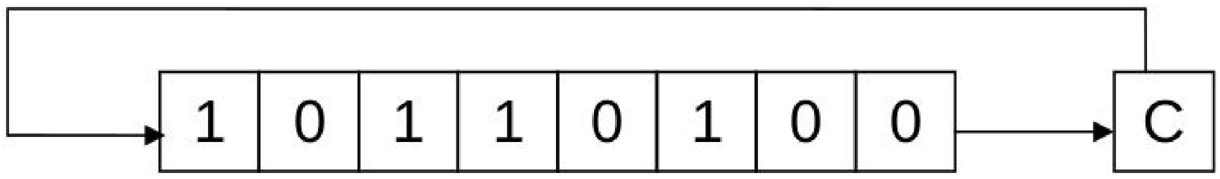
- Every bit moves one place to the left
- Bit that is dropped from the left most is inserted back at the right most bit
- Same dropped bit is copied in the Carry Flag as well



Rotate Through Carry Right (RCR)

In Rotate Through Carry Right

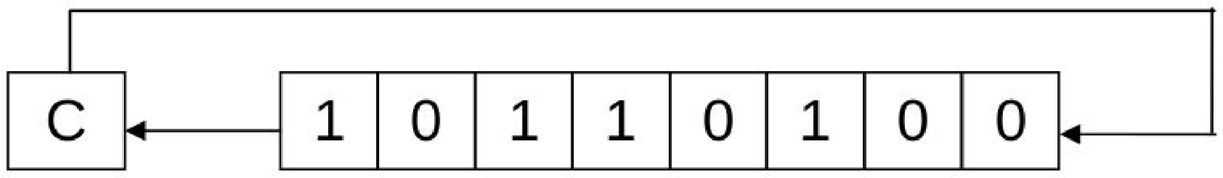
- Every bit moves one place to the right
- Value of Carry Flag is copied to Left most bit
- Dropped bit from rightmost is copied to Carry Flag



Rotate Through Carry Left (RCL)

In Rotate Through Carry Left

- Every bit moves one place to the left
- Value of Carry Flag is copied to Right most bit
- Dropped bit from leftmost is copied to Carry Flag



Bitwise Logical Operations

AND Operation:
Performs logical bitwise *and* of two operands and returns the result to the destination operand.

Example:

```
mov ax, 5 (00000101)
mov bx, 7 (00000111)
AND ax, bx

ax = 00000101
```

OR Operation:
Performs logical bitwise ‘inclusive or’ of the two operands and returns result to destination operand.

Example:

```
Mov ax, 5 (00000101)
Mov bx, 7 (00000111)
OR ax, bx

Ax = 00000111
```

XOR Operation:
Performs logical bitwise ‘exclusive or’ of two operands and returns result to destination operand. Due to it’s reversible property, it’s used in many cryptography algorithms, image processing and drawing operations.

Example:

```
Mov ax, 5 (00000101)
Mov bx, 7 (00000111)
XOR ax, bx

Ax = 00000010
```

NOT Operation:
Inverts the bits of operand.

Example:

```
Mov ax, 5 (00000101)
NOT ax
```

Ax = (11111010)

Masking Operations

Selective Bit Clearing:
Use AND operation you can clear specific bits according to your requirements with the help of a mask. Source operand has the mask. Mask contains ‘1’ on bits that you don’t want to change and ‘0’ on bits which you want to clear.

Example:

You have 15 (00001111) and you want to clear bit 0 and 2.

Mov ax, 00001111
AND ax, 11111010 (mask)

Ax = 00001010 (all bits are same except for 0 and 3 - they are cleared)

Selective Bit Setting:
Use OR operation to set selective bits with the help of a mask. Mask contains ‘0’ on bits that you don’t want to change and ‘1’ on bits which you want to set.

Example:
You have 00101101 and you want to set bit 1 and 4.

Mov ax, 00101101
OR ax, 00010010

Ax = 00111111

Selective Bit Inversion:
Use XOR operation to invert selective bits with the help of a mask. Mask contains ‘0’ on bits that you don’t want to change and ‘1’ on bits which you want to invert. XOR allows selective inversion whereas NOT inverts every bit.

Example:

You have 00101101 and you want to invert bit 1 and 2.

Mov ax, 00101101

XOR ax, 00000110

Ax = 00101011

Selective Bit Testing:
Using TEST instruction you can test any bit, whether it’s set or not, without any changes in destination operand. The same can be done with AND operation but it destroys the destination operand’s mask which might be needed in some cases. If you don’t want to change the destination then TEST is suitable. By checking flags you can see if a particular bit was set or cleared.

Example:

Mov ax, 00001110
TEST ax, 00000001

Result: Because we’re checking only 0 bit and it’s clear in the *ax* so ZF will be set.