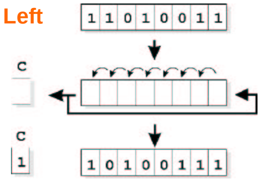
Rotate Instructions

Bits shifted out one side are shifted back in the other side, resulting in a rotation. **C** is set as before.

Rotate Left




```
1101 0011 -> rotate left by 1 -> 1010 0111
                                     C = 1
```


Examples

```
move.w   #4,d0
rol.w    d0,d2     * rotate by 4
rol.b    #2,d7 * rotate by 2
rol.l    d1        * rotate by 1
```

Rotate Right

```
1110 1001 -> rotate right by 3 -> 0011 1101
                                     C = 1
```


ROL and ROR

- Shift count = 0
  - -> **C** is cleared, operants unaffected.
- No shift count operant
  - -> Shift of **1** position.
- If destination is a memory location
  - -> Shift count must be **1** and operation size must be **word**.

Uses of Shift Instructions

- Simple Multiplication/Division
- Field Extraction

Multiply by 10

```
1234 -> 12340
```

-> Digits are shifted **left** 1 place

Multiply by 2

```
%01101 -> %11010
    13 -> 26
```

-> Digits are shifted **left** 1 place in the binary system

Multiply by 11 in the decimal system?

```
12 * 11     =   12 * 10^1
            +   12 * 10^0

            =   120
            +    12
            =   132
```

-> Identify powers of 10 in the multiplier

Multiply binary number by 10?

Method(a) $10 = 2^3 + 2^1$

2^3 -> Shift left 3 places

2^1 -> Shift left 1 place

Example: %110 * %1010

```
Shift 3  -> %110000
Shift 1  -> %001100
Add      -> %111100
```


Method (b)

Method(b) $10 = (2^2 + 2^0) * 2$

2^2 -> *Shift left 2 places*

2^0 -> *No shift*

*2 -> *Shift left 1 place*

Example: *%110 * %1010*

```
Shift 3  -> %011000
No shift -> %000110
Add      -> %011110
Shift 1  -> %111100
```


Program to multiply a in $2000 by 10

```
          org       $1000

 10 ->move.w $2000,d0     * get N
 40 ->lsl.w     #2,d0     * N << 2
 50 ->add.w     $2000,d0  * add N
100 ->lsl.w     #1,d0     * N << 1
      move.w    d0,$2000  * save result
      trap      #0        * end
```

Division

Division by powers of 2 is possible using right shift operation:

```
CASE 1: %1010 ÷ %10
     -> %1010 shift right by 1
     -> %0101

CASE 2: %1011 ÷ %100
     -> %1011 shift right by 2
     -> %0010
```


Division

```
CASE 2: %1011 ÷ %100
     -> %1011 shift right by 2
     -> %0010
```

In CASE 2 above, the **l**east **S**ignificant **B**it has been **Lost**.

11÷4 =2

i.e.: We need to determine the remainder somehow

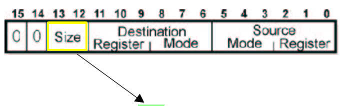
Field Extraction

Often required to **extract** an arbitrary sequence of bits from data.

Example:

Extract the size field of the move instruction

$3200 = move.w d0,d1



$3200 = %0011 0010 0000 0000

We can achieve this in 2 stages:

- **Clear** all bits except bits we're interested in. This is known as masking.
- **Shift** bits until they occupy the least significant position then read value.

```
1.Clear: %0011 0010 0000 0000
         %0011 0000 0000 0000 & <-mask
         %0011 0000 0000 0000
2.Shift: %0011 0000 0000 0000
         shift right 12 places
         %0000 0000 0000 0011
```

Bit Extraction Program

Write code to store in d0 the size field of an instruction located at $2000.

```
move.w $2000,d0      * get the instruction
and.w  #$3000,d0     * mask the bits
move.w #12,d1        * shift count 12
lsr.w  d1,d0         * shift >> 12
```

How would you change the program to change the size field of an instruction to be the size indicated by the d0 register?