Logical (Boolean) Instructions

- The 68000 supports four types of Boolean Instructions

Instruction	Operation	Mnemonic	X N Z V C
AND	to data register to memory immediate	**AND ea,Dn** **AND Dn,ea** **ANDI #k,ea**	×✓✓0 0
EOR	to data register Immediate	**EOR ea,Dn** **EORI #k,ea**	×✓✓0 0
NOT		**NOT ea**	×✓✓0 0
OR	to data register to memory Immediate	**OR ea,Dn** **OR Dn,ea** **ORI #k,ea**	×✓✓0 0

Logical And

- Consider the following C code

```
char a = 0xFF;
char b = 0x7E;
a = a & b;
```

Truth Table

A	B	A and B
0	0	0
0	1	0
1	0	0
1	1	1

- Assume that **a**(D1), **b**(D0)

```
and.b d0,d1
```

Logical OR

- Consider the following C code

```
char a = 0x80;
char b = 0x7E;
a = a | b;
```

Truth Table

A	B	A or B
0	0	0
0	1	1
1	0	1
1	1	1

- Assume that **a**(D1), **b**(D0)

```
or.b d0,d1
```

Logical EOR

- Consider the following C code

```
char a = 0xFF;
char b = 0x36;
a = a ^ b;
```

Truth Table

A	B	A xor B
0	0	0
0	1	1
1	0	1
1	1	0

- Assume that **a**(D1), **b**(D0)

```
eor.b d0,d1
```

Logical NOT

- Consider the following C code

```
char a = 0x7E;
a = ~a;
```

Truth Table

A	not A
0	1
1	0

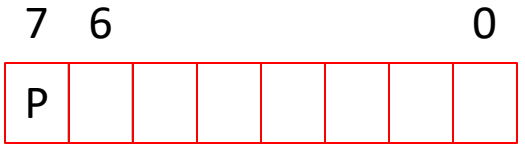
- Assume that **a**(D1)

```
not.b d1
```

Example

- Write a program to set the parity bit included with an ASCII character (byte) to zero, then convert the ASCII character to lowercase
  - Assume the byte is contained in D0

ASCII CODE	
Lower Case	Upper Case
a = 1100001	A = 1000001
b = 1100010	B = 1000010
c = 1100011	C = 1000011
…	…



Shift Instructions

- There are six instructions that shift an operand one or places to the left or to the right

Logical Operation	C Operation	Signed Data	Unsigned Data
shift left	<<	**ASL**	**LSL**
shift right	>>	**ASR**	**LSR**
circular shift left	N/A	**ROL**	
circular shift right	N/A	**ROR**	

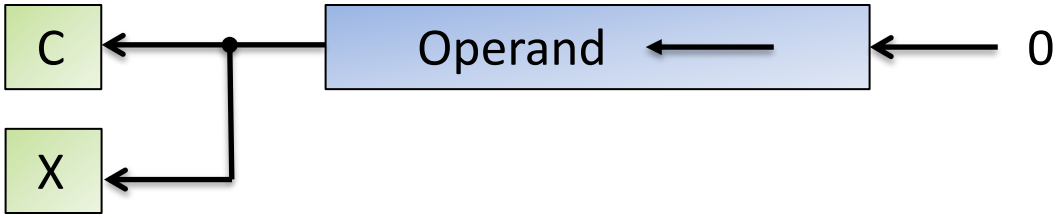
Expressing Shift Instructions in Assembly Language

- The shift count may be expressed in one of three ways

Shift Count	Example	Description
Immediate	`ASL.W #3,D0`	Allows a shift of 1-8 places
Register	`ASL.L D0,D1`	Allows a shift of 1-32 places
Value of 1	`ASL (A0)`	Shifts the WORD 1 place

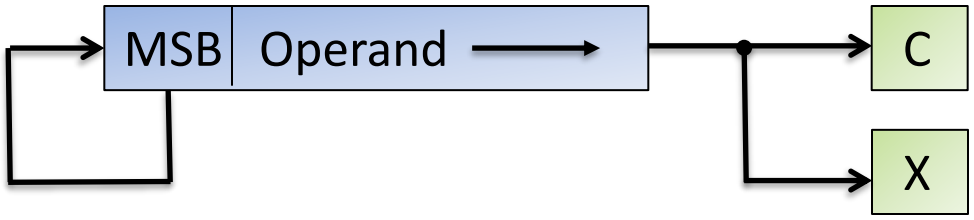
Arithmetic Shift Instructions

ASL



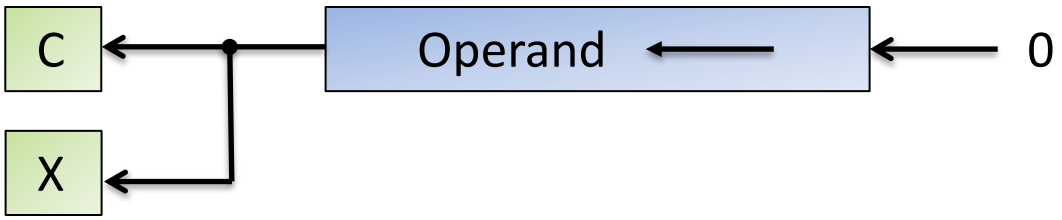
V-bit in CCR is affected

ASR



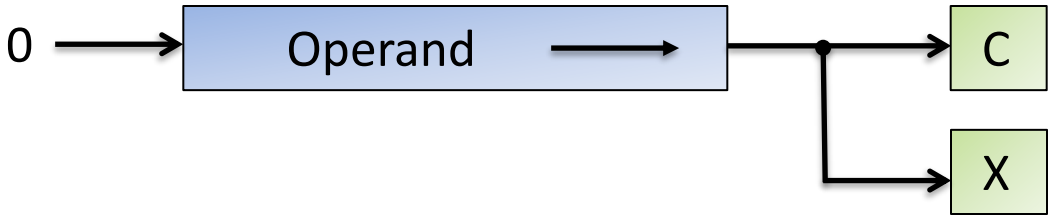
Logical Shift Instructions

LSL

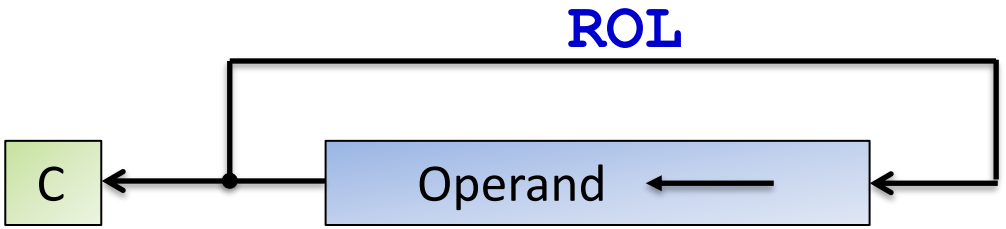


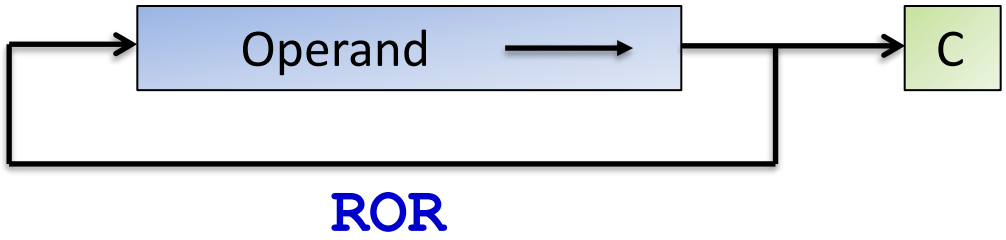
V-bit in CCR is NOT affected

LSR



Circular Shift Instructions

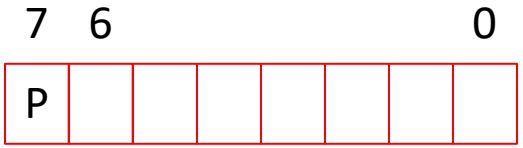




Example

- Write a program to set the parity bit (msb) of a byte. Assume that even parity is being used.

Algorithm
1. Set parity bit to 0 2. Count number of 1s in bit positions 0 through 6 3. If the number of 1s is odd, set parity bit to 1

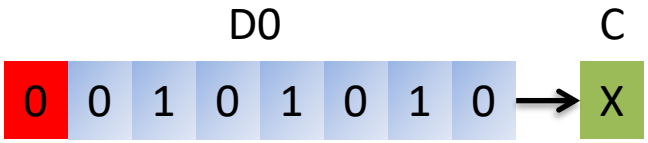


Example – Assembler Code

```
        move.b      ascii,d0      ;original ascii character
        andi.b      #$7f,d0       ;clear parity bit
        clr.l       d1            ;clear parity counter
        move.l      #7,d2         ;loop counter
* Count number of bits equal to 1
loop    ror.b       #1,d0         ;rotate lsb into carry flag
        bcc         zero          ;bit equal to 1?
        addq        #1,d1         ;increment parity counter
zero    subq        #1,d2         ;repeat until all 7
        bne         loop          ;bits are tested
* Set parity bit if number of bits is odd
        ror.b       #1,d0         ;restore d0
        lsr.b       #1,d1         ;move lsb into carry
        bcc         exit          ;exit on even number
        ori.b       #$80,d0       ;set parity bit
exit    ...
ascii   dc.b        %00101010     ;original byte
```

Example – Assembler Code

```
* Count number of bits equal to 1
loop    ror.b   #1,d0
        bcc     zero
        addq    #1,d1
zero    subq    #1,d2
        bne     loop
```

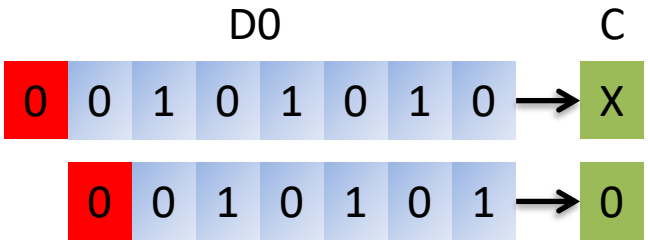


# of 1s found

loop counter

Example – Assembler Code

```
* Count number of bytes equal to 1
loop    ror.b   #1,d0
        bcc     zero
        addq    #1,d1
zero    subq    #1,d2
        bne     loop
```



d1 XXXXXX00

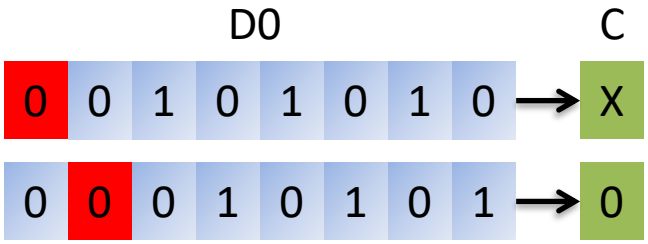
# of 1s found

d2 XXXXXX07

loop counter

Example – Assembler Code

```
* Count number of bytes equal to 1
loop    ror.b   #1,d0
        bcc     zero
        addq    #1,d1
zero    subq    #1,d2
        bne     loop
```



d1 XXXXXX00

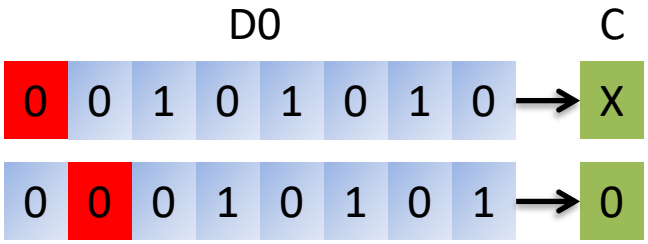
# of 1s found

d2 XXXXXX07

loop counter

Example – Assembler Code

```
* Count number of bytes equal to 1
loop    ror.b   #1,d0
        bcc     zero
        addq    #1,d1
zero    subq    #1,d2
        bne     loop
```



d1 XXXXXX00

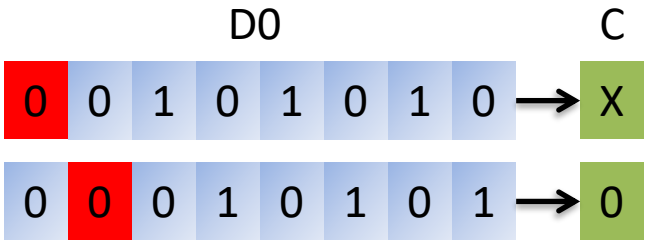
# of 1s found

d2 XXXXXX07

loop counter

Example – Assembler Code

```
* Count number of bytes equal to 1
loop    ror.b   #1,d0
        bcc     zero
        addq    #1,d1
zero    subq    #1,d2
        bne     loop
```



d1 XXXXXX00

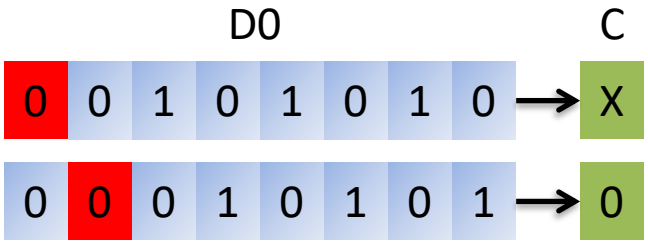
# of 1s found

d2 XXXXXX07

loop counter

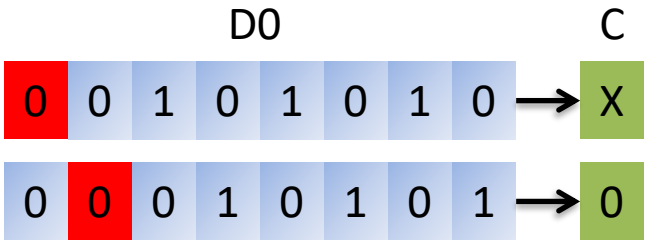
Example – Assembler Code

```
* Count number of bytes equal to 1
loop    ror.b   #1,d0
        bcc     zero
        addq    #1,d1
zero    subq    #1,d2
        bne     loop
```



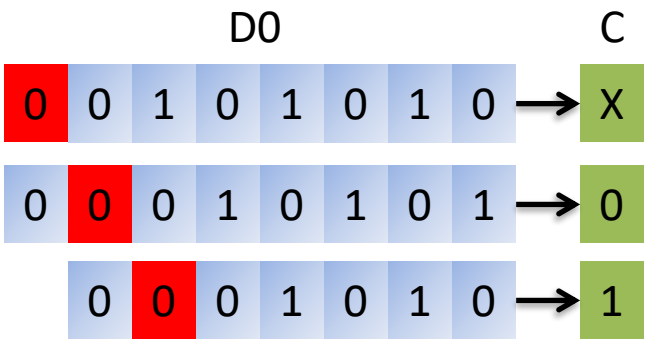
Example – Assembler Code

```
* Count number of bytes equal to 1
loop    ror.b   #1,d0
        bcc     zero
        addq    #1,d1
zero    subq    #1,d2
        bne     loop
```



Example – Assembler Code

```
* Count number of bytes equal to 1
loop    ror.b   #1,d0
        bcc     zero
        addq    #1,d1
zero    subq    #1,d2
        bne     loop
```



d1 XXXXXX00

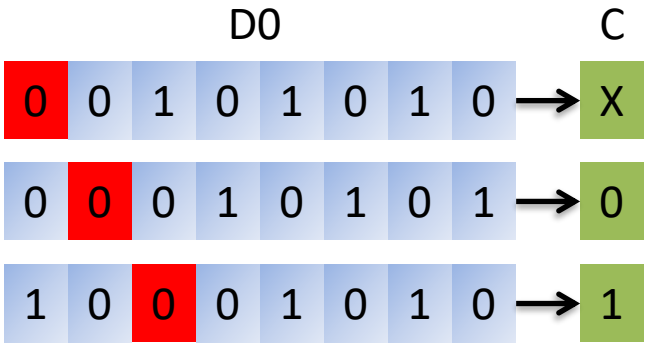
# of 1s found

d2 XXXXXX06

loop counter

Example – Assembler Code

```
* Count number of bytes equal to 1
loop    ror.b   #1,d0
        bcc     zero
        addq    #1,d1
zero    subq    #1,d2
        bne     loop
```



d1 XXXXXX00

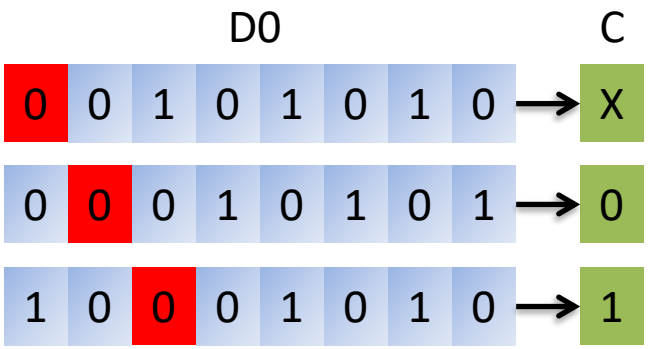
# of 1s found

d2 XXXXXX06

loop counter

Example – Assembler Code

```
* Count number of bytes equal to 1
loop    ror.b   #1,d0
        bcc     zero
        addq    #1,d1
zero    subq    #1,d2
        bne     loop
```



d1 XXXXXX00

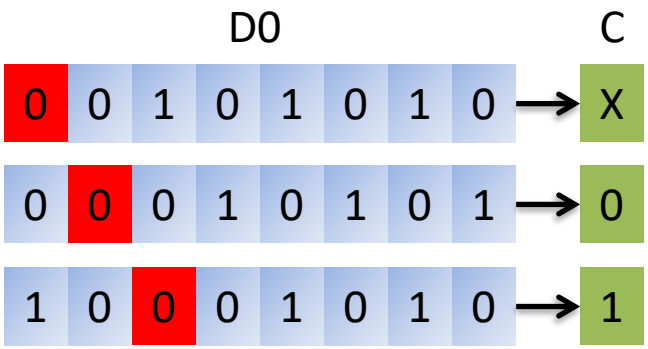
# of 1s found

d2 XXXXXX06

loop counter

Example – Assembler Code

```
* Count number of bytes equal to 1
loop    ror.b   #1,d0
        bcc     zero
        addq    #1,d1
zero    subq    #1,d2
        bne     loop
```



d1 XXXXXX01

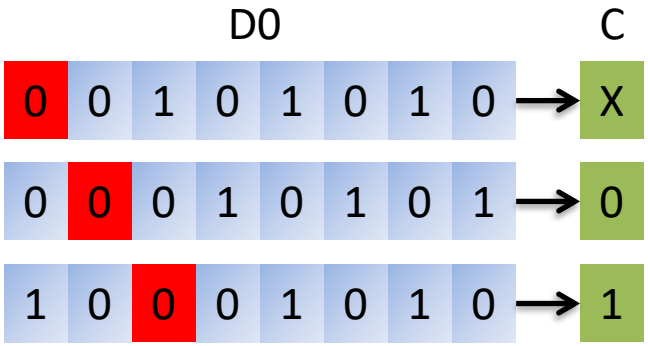
# of 1s found

d2 XXXXXX06

loop counter

Example – Assembler Code

```
* Count number of bytes equal to 1
loop    ror.b   #1,d0
        bcc     zero
        addq    #1,d1
zero    subq    #1,d2
        bne     loop
```



d1 XXXXXX01

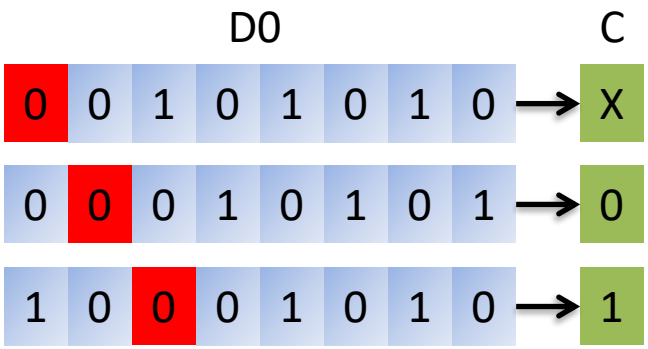
# of 1s found

d2 XXXXXX05

loop counter

Example – Assembler Code

```
* Count number of bytes equal to 1
loop    ror.b   #1,d0
        bcc     zero
        addq    #1,d1
zero    subq    #1,d2
        bne     loop
```



d1 XXXXXX01

# of 1s found

d2 XXXXXX05

loop counter

Example – Assembler Code

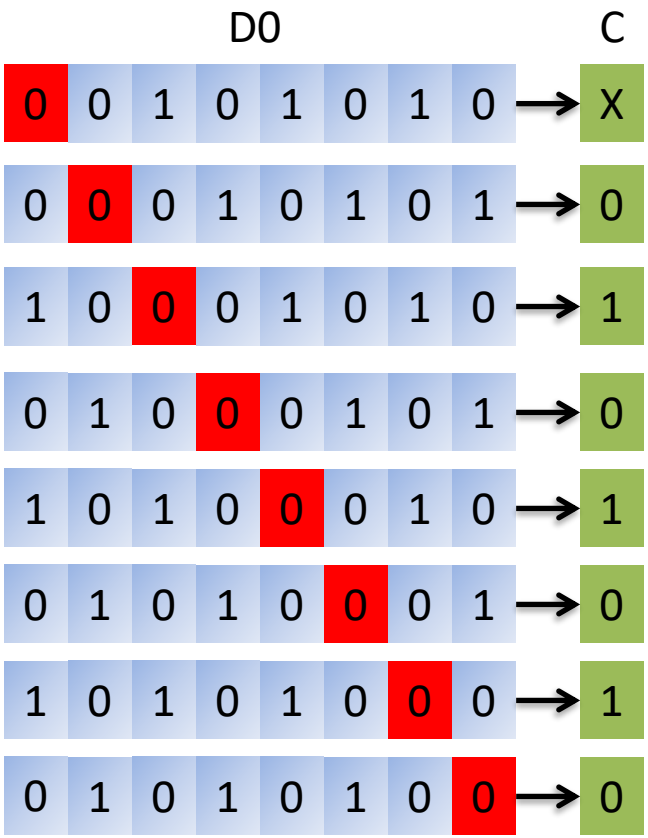
```
* Count number of bytes equal to 1
loop    ror.b   #1,d0
        bcc     zero
        addq    #1,d1
zero    subq    #1,d2
        bne     loop
```

d1 XXXXXX03

# of 1s found

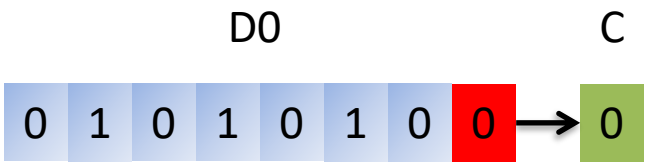
d2 XXXXXX00

loop counter



Example – Assembler Code

```
* Set parity bit if number of bits is odd
  ror.b       #1,d0
  lsr.b       #1,d1
  bcc         exit
  ori.b       #$80,d0
```

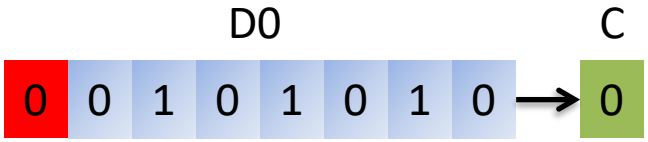


# of 1s found

loop counter

Example – Assembler Code

```
* Set parity bit if number of bits is odd
   ror.b       #1,d0
   lsr.b       #1,d1
   bcc         exit
   ori.b       #$80,d0
```

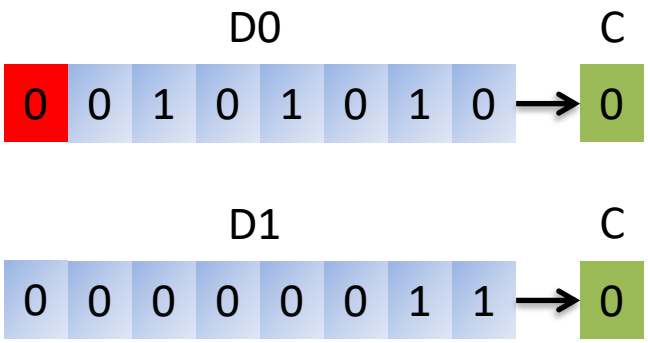


# of 1s found

loop counter

Example – Assembler Code

```
* Set parity bit if number of bits is odd
   ror.b       #1,d0
   lsr.b       #1,d1
   bcc         exit
   ori.b       #$80,d0
```

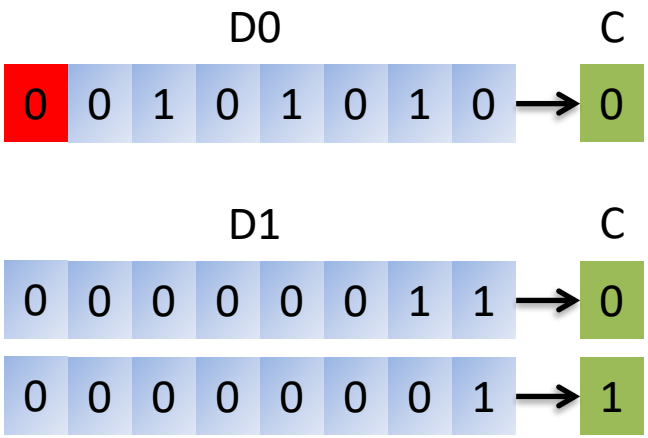


# of 1s found

loop counter

Example – Assembler Code

```
* Set parity bit if number of bits is odd
   ror.b        #1,d0
   lsr.b        #1,d1
   bcc          exit
   ori.b        #$80,d0
```



d1 XXXXXX01

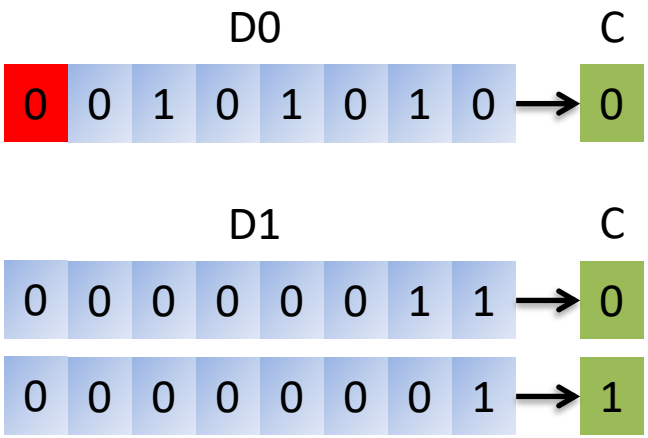
# of 1s found

d2 XXXXXX00

loop counter

Example – Assembler Code

```
* Set parity bit if number of bits is odd
   ror.b       #1,d0
   lsr.b       #1,d1
   bcc         exit
   ori.b       #$80,d0
```



d1 XXXXXX01

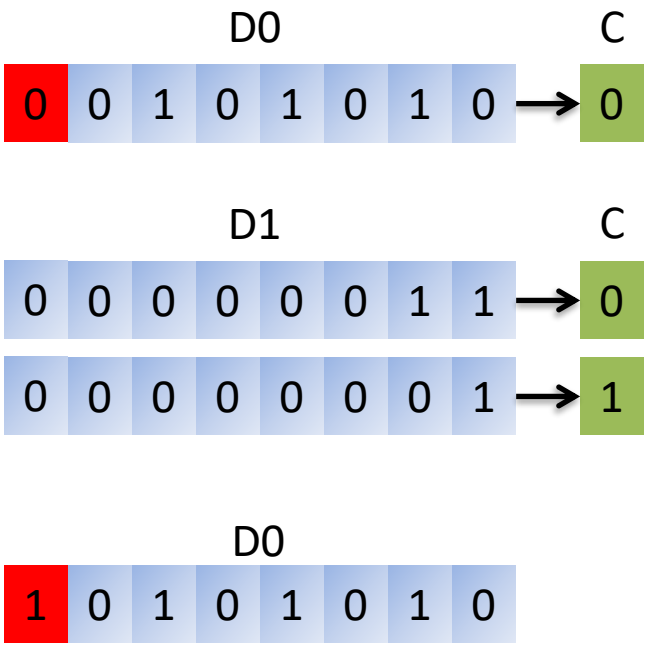
# of 1s found

d2 XXXXXX00

loop counter

Example – Assembler Code

```
* Set parity bit if number of bits is odd
   ror.b       #1,d0
   lsr.b       #1,d1
   bcc         exit
   ori.b       #$80,d0
```



d1 XXXXXX01

# of 1s found

d2 XXXXXX00

loop counter

Summary

- Boolean instructions manipulate data at the bit level
  - AND clears bit(s), OR sets bit(s), and EOR flips bit(s)
  - All support byte, word and longword operands
- Shift and rotate instructions
  - All support byte, word and longword operands
  - All shifts and rotates are allowed in both directions
  - Only data register and memory locations may be used as operands
    - Shift count limited to 1 bit when memory is the destination
  - Arithmetic shifts operate on 2's-complement values
    - Multiplication by power of 2 (left shift)
    - Division by power of 2 (right shift)
  - Logical shifts for everything else
  - Rotates preserve data unlike arithmetic/logical shifts