

Shift and Rotate Instructions



Chapter 7
Section 7.10

Shift and Rotate Instructions

□ Shift

■ Logical Shift (LSL, LSL r , LSR, LSR r)

- Shift a register r (A,B,D) or a memory operand to the left or right *one bit*.

■ Arithmetic Shift (ASL, ASL r , ASR, ASR r)

- Left: multiplying by 2
- Right: dividing by 2 (MSB is replicated)

■ N, Z, V, C

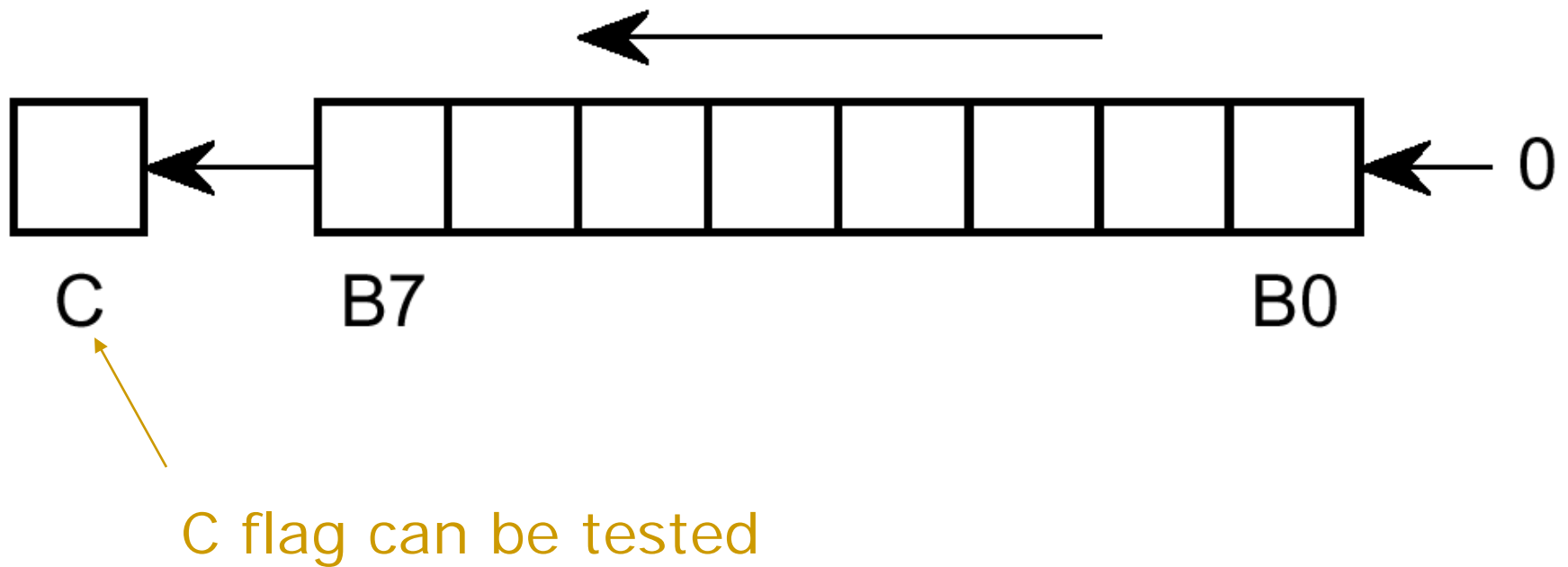
□ Rotate (ROL, ROR, ROL r , ROR r)

- Rotate a register r (A,B) or a memory operand to the left or right *through carry (C) one bit*.

- N, Z, V, C

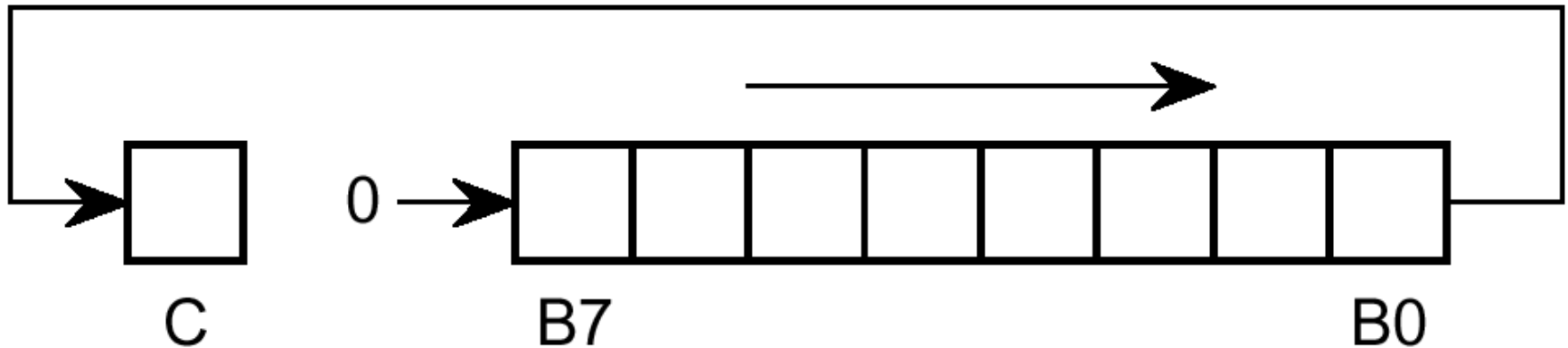
Logical shift left (LSL) instructions.

Figure 7-7



Logical shift right (LSR) instructions.

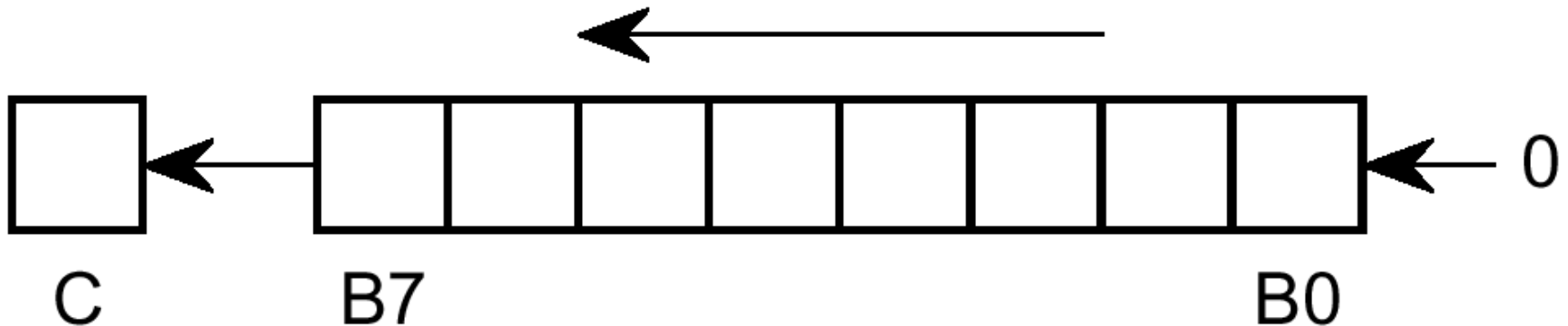
Figure 7-8



Arithmetic shift left (ASL) instructions.

Figure 7-5

Exactly the same as LSL.
Use for multiplication by 2.

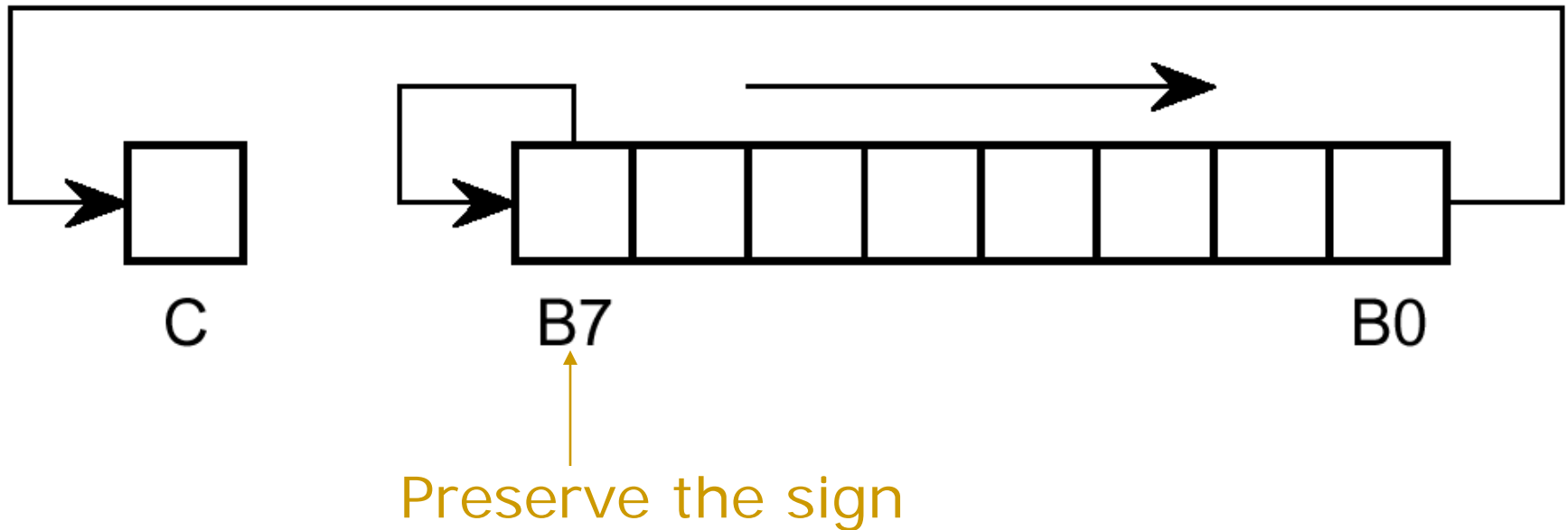


Watch out for "overflow": $V = N \text{ xor } C$

Arithmetic shift right (ASR) instructions.

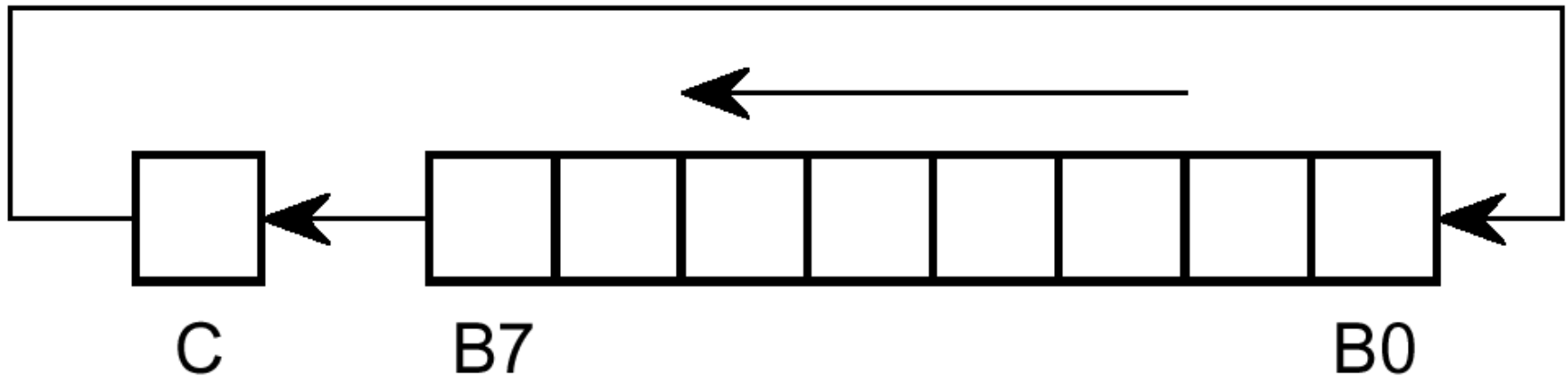
Figure 7-6

Use for division of a signed number by 2.



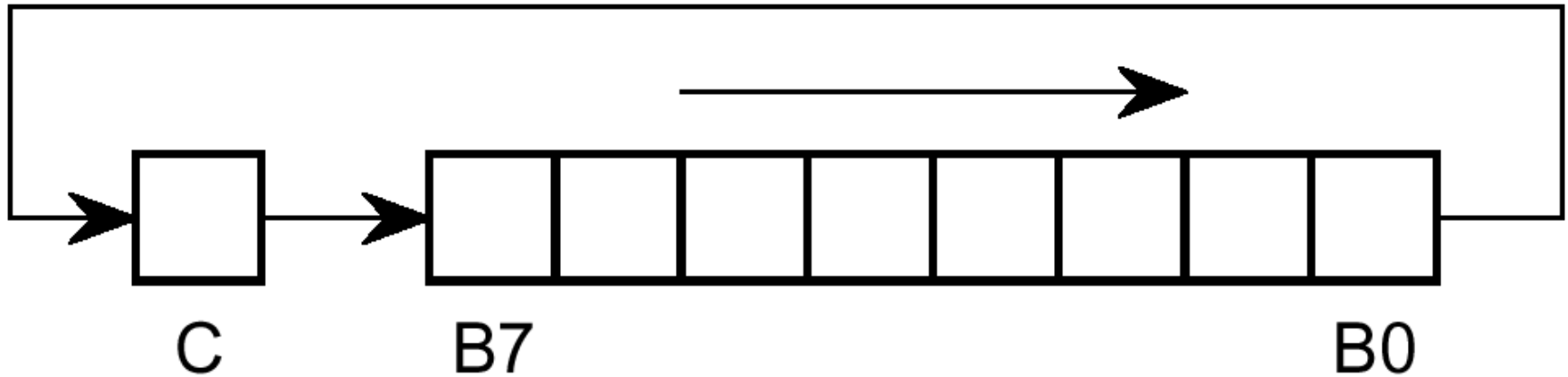
Rotate left (ROL) instructions.

Figure 7-9



Rotate right (ROR) instructions.

Figure 7-10



Logical Shift Instructions

Logical Shifts		
Mnemonic	Function	Operation
LSL LSLA LSLB	Logic Shift Left Memory Logic Shift Left A Logic Shift Left B	
LSLD	Logic Shift Left D	
LSR LSRA LSRB	Logic Shift Right Memory Logic Shift Right A Logic Shift Right B	
LSRD	Logic Shift Right D	

Arithmetic Shift & Rotate Instructions

Arithmetic Shifts		
Mnemonic	Function	Operation
ASL ASLA ASLB	Arithmetic Shift Left Memory Arithmetic Shift Left A Arithmetic Shift Left B	
ASLD	Arithmetic Shift Left D	
ASR ASRA ASRB	Arithmetic Shift Right Memory Arithmetic Shift Right A Arithmetic Shift Right B	
Rotates		
Mnemonic	Function	Operation
ROL ROLA ROLB	Rotate Left Memory Through Carry Rotate Left A Through Carry Rotate Left B Through Carry	
ROR RORA RORB	Rotate Right Memory Through Carry Rotate Right A Through Carry Rotate Right B Through Carry	

Example 7-23

Assume the A value is \$A9. What is the result of each of the following instructions? ASLA, ASRA, LSLA, LSRA, ROLA, RORA.

Solution:

The easiest way to look at these instructions is to show the values in binary. Before each instruction is executed, A contains %10101001. After each instruction, then, we find the following:

<u>Before</u>	<u>After</u>	<u>C</u>	<u>A Register</u>	<u>Comments</u>
10101001	ASLA	1	01010010	Zero shifted into bit-0.
10101001	ASRA	1	11010100	Sign bit is preserved.
10101001	LSLA	1	01010010	Same result as ASLA.
10101001	LSRA	1	01010100	Different than the ASRA.
10101001	ROLA	1	0101001C	Carry bit is rotated into bit-0.
10101001	RORA	1	C1010100	Carry bit is rotated into bit-7.

Example 7-22

Write M68HC12 code to multiply a 16-bit number in the D register by 10 using arithmetic left shift instructions instead of the *EMUL* instruction.

```
0000 7C0009      1      std      TEMP      ; Save in location TEMP
0003 59          2      asld           ; X2
0004 59          3      asld           ; X2 again = X4
0005 F30009      4      addd      TEMP      ; Add the original.  Now X5
0008 59          5      asld           ; X2 = X10
                                6
0009             7      TEMP:    DS      2      ; Temp storage
```

Example 7-24

Show how to load the ASCII code for the number 4 into accumulator B and then shift the least significant nibble into the most significant.

```
0000 C634      1   ldab    #'4'      ; ASCII code for 4
0002 58        2   lslb          ; Shift four bit positions
0003 58        3   lslb
0004 58        4   lslb
0005 58        5   lslb
```

What is in B after these instructions have been executed?

Solution:

B = \$40

Example

Use a rotate instruction in a loop to successively turn on LEDs zero to seven as in Example 7-2-.

Solution:

A zero must be shifted through the bits in a pattern like 11111110, 11111100, 11111000, . . . A program sequence to do this is:

Example (cont)

```
0000          1  COUNT:    EQU      8          ; Going to do 8 bits
0000          2  ALLBITS:  EQU      %11111111 ; Spec all bits
0000          3  FIRST:    EQU      %11111110 ; Turn on bit 0
0000          4  PORTH:    EQU      $24       ; Address of Port H
              5  ;          - - -
              6  ; Turn off all bits
              7  OUTER:
0000 4C24FF    8          bset      PORTH,ALLBITS
0003 86FE      9          ldaa     #FIRST   ; Initialize for bit-0
0005 C608     10         ldab     #COUNT
0007 5A24     11  LOOP:    staa     PORTH   ; Turn on a bit
0009 160014   12         jsr      Delay   ; Delay for a while
000C 10FE     13         clc          ; clear carry bit to rotate
              14 ;          ; into LSB
000E 45       15         rola     ; Shift the ACCA left
000F 0431F5   16         dbne    b,LOOP   ; Do it for 8 bits
0012 20EC     17         bra      OUTER  ; Do it forever
              18 ;          - - -
              19 ; Dummy subroutine
0014 3D       20  Delay:    rts
```