

Logic & Bit-Manipulation Instructions



Chapter 7

Sections 7.4, 7.9, 7.12, 7.13

Bit-manipulation instructions

- Logic operations on bytes
 - and, or, complement, exclusive-or
- Shift & rotate bits of a byte
 - logical & arithmetic shift, circular rotate
- Manipulate individual bits of a byte
 - set, clear, test

Logic Instructions

- Bit-wise boolean logic operation between a register r (A, B, CCR) and a memory or immediate operand.
 - Operations on bytes only
 - Result saved in the register
- Four logic operations
 - AND (AND r), OR (ORAR), XOR (EOR r)
 - "Complement" (COM r , COM mem)
- Flags
 - N & Z set, 0 \rightarrow V, C undefined for AND, OR, XOR
 - N, Z, V, C all set when CCR is involved.

Logic Instructions

TABLE 7-13 Logic Instructions

Function	Opcode	Symbolic Operation	Addressing Modes						Condition Codes			
			I M M	D I R	E X T	I D X	I D R	I N H	N	Z	V	C
AND A with Memory	ANDA	A AND (M) → A	x	x	x	x	x		↕	↕	0	—
AND B with Memory	ANDB	B AND (M) → B	x	x	x	x	x		↕	↕	0	—
AND CCR with Memory	ANDCC	CCR AND #data → CCR	x						↓	↓	↓	↓
Exclusive OR A with Memory	EORA	A EOR (M) → A	x	x	x	x	x		↕	↕	0	—
Exclusive OR B with Memory	EORB	B EOR (M) → B	x	x	x	x	x		↕	↕	0	—
Inclusive OR A with Memory	ORAA	A OR (M) → A	x	x	x	x	x		↕	↕	0	—
Inclusive OR B with Memory	ORAB	B OR (M) → B	x	x	x	x	x		↕	↕	0	—
Inclusive OR CCR with Constant	ORCC	CCR OR #data → CCR	x						↑	↑	↑	↑
One's-Complement Memory	COM	(M)* → (M)			x	x	x		↕	↕	0	1
One's Complement A	COMA	A* → A						x	↕	↕	0	1
One's Complement B	COMB	B* → B						x	↕	↕	0	1

Example

If memory location \$0010 contains \$B3 and A contains \$64, what is the result of the following instructions:

ANDA \$10, ANDA # \$10, ORAA \$10, ORAA # \$10, EORA \$10, COMA, COM \$10

Solution:

ANDA \$10	A	0 1 1 0 0 1 0 0
	(0010)	<u>1 0 1 1 0 0 1 1</u>
		0 0 1 0 0 0 0 0

ANDA # \$10	A	0 1 1 0 0 1 0 0	Immediate addressing
	\$10	<u>0 0 0 1 0 0 0 0</u>	
		0 0 0 0 0 0 0 0	

ORAA \$10	A	0 1 1 0 0 1 0 0
	(0010)	<u>1 0 1 1 0 0 1 1</u>
		1 1 1 1 0 1 1 1

Example (cont)

ORAA	#\$10	A	0 1 1 0 0 1 0 0	Immediate addressing
		\$10	0 0 0 1 0 0 0 0	
			<u>0 1 1 1 0 1 0 0</u>	
EORA	\$10	A	0 1 1 0 0 1 0 0	
		(0010)	<u>1 0 1 1 0 0 1 1</u>	
			1 1 0 1 0 1 1 1	
COMA		A	<u>0 1 1 0 0 1 0 0</u>	
			1 0 0 1 1 0 1 1	
COM	\$10	(0010)	<u>1 0 1 1 0 0 1 1</u>	
		(0010)	0 1 0 0 1 1 0 0	

Example - Masking Operations

Assume the A accumulator has a packed BCD number to be converted to ASCII and printed (using a subroutine called PRINT). Write a small segment of code using logic instructions to do this.

```
0000          1  LS_MASK: EQU          %00001111 ; Least sig nibble mask
              2  ;
0000 B701     3          tfr          a,b          ; Save the BCD number in B
              4  ; Need to print the most significant nibble first
0002 44       5          lsra          ; Shift 4 bits to right
0003 44       6          lsra
0004 44       7          lsra
0005 44       8          lsra
0006 8A30     9          oraa         #$30        ; Convert to ASCII
0008 160014   10         jsr          PRINT      ; Go print it
000B B710    11         tfr          b,a          ; Get the original back
000D 840F    12         anda         #LS_MASK   ; Set most sig bits to 0
000F 8A30    13         oraa         #$30        ; Convert to ASCII
0011 160014   14         jsr          PRINT      ; Print it
              15        ;          - - -
              16        ; Dummy subroutine
0014 3D      17        PRINT:      rts
```

Bit Test and Manipulation

Instructions

□ BITr

- Perform AND operation between a register r (A,B) and a memory operand, and the result is not saved.

□ BSET

- Set bits in a memory operand by performing OR operation between the operand and a "mask".
- The bits to be set is indicated by setting the corresponding bits in the mask.

□ BCLR

- Clear bits in a memory operand by performing AND operation between the operand and a "mask".
- The bits to be cleared is indicated by setting the corresponding bits in the mask.

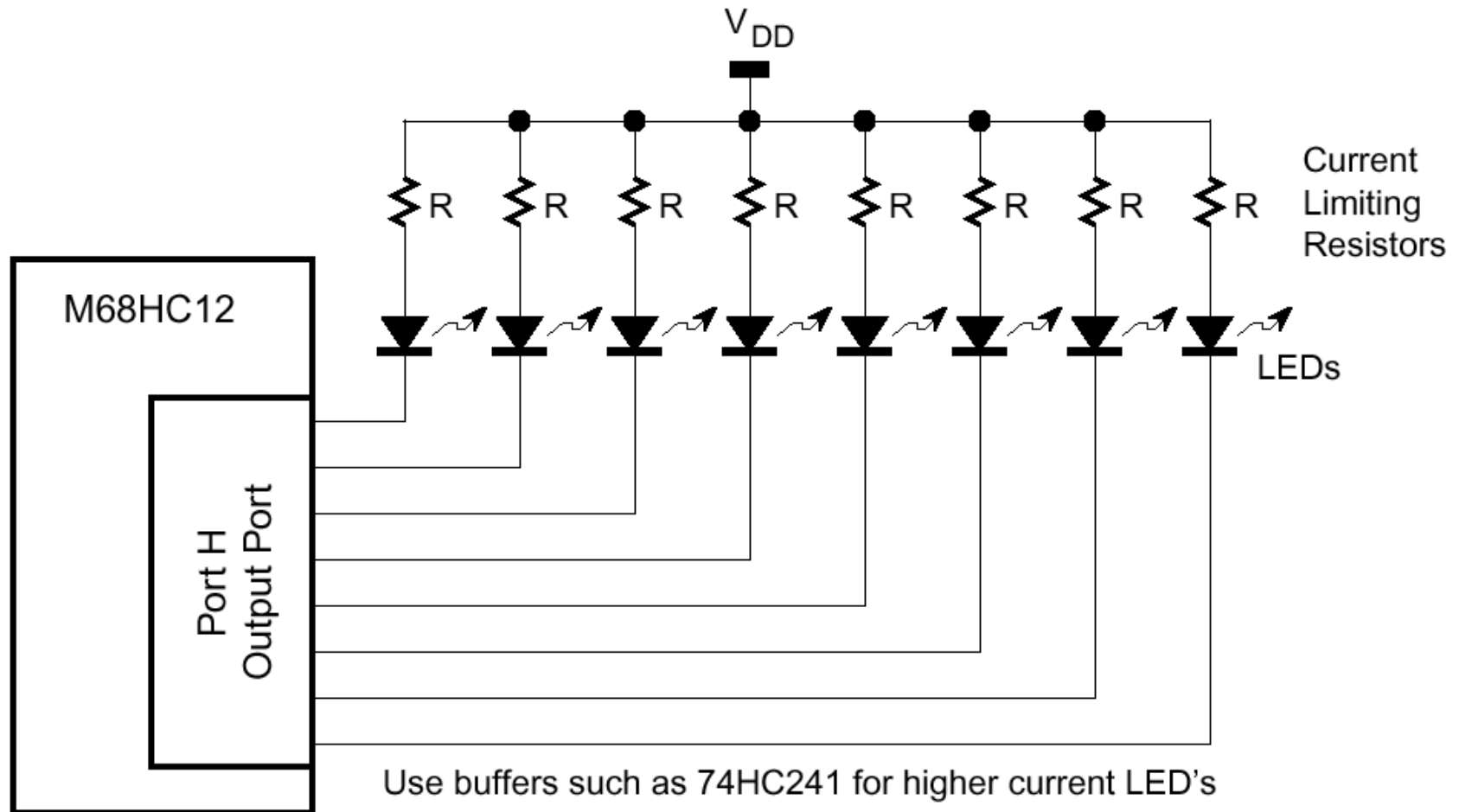
□ N, Z, 0 \rightarrow V

Logic and Shift/Rotate Instructions

□ Bit Test and Manipulation

Mnemonic	Function	Operation
BCLR	Clear Bits in Memory	$(M) \bullet (\overline{mm}) \Rightarrow M$
BITA	Bit Test A	$(A) \bullet (M)$
BITB	Bit Test B	$(B) \bullet (M)$
BSET	Set Bits In Memory	$(M) + (mm) \Rightarrow M$

BCLR and BSET used for turning on and off LEDs (Figure 7-4)



LED Program using BCLR and BSET

Example 7-20

Use bset/bclr instructions to successively turn on LEDs zero to seven connected to “Port H” on the 68HC12.

Solution:

```
1 ; Equates for all the bits
0000      2 BIT0:      EQU      %00000001
0000      3 BIT1:      EQU      %00000010
0000      4 BIT2:      EQU      %00000100
0000      5 BIT3:      EQU      %00001000
0000      6 BIT4:      EQU      %00010000
0000      7 BIT5:      EQU      %00100000
0000      8 BIT6:      EQU      %01000000
0000      9 BIT7:      EQU      %10000000
0000     10 ALL:       EQU      %11111111
0000     11 REGS:      EQU      $0000          ; Start of the I/O regs
0000     12 PORTH:     EQU      REGS+$24      ; Offset for Port H
0000     13 DDRH:      EQU      REGS+$25      ; Data dir register
0000     14 ;          - - -
```

(Continue on next slide)

LED Program using BCLR and BSET

(Example 7-20 cont)

```
0000 4C25FF      15          bset    DDRH,ALL      ; Make all bits outputs
0003 4C24FF      16  LOOP:    bset    PORTH,ALL    ; Turn out all LEDs
0006 4D2401      17          bclr    PORTH,BIT0   ; Turn on bit 0
0009 4D2402      18          bclr    PORTH,BIT1   ; Turn on bit 1
000C 4D2404      19          bclr    PORTH,BIT2   ; Turn on bit 2
000F 4D2408      20          bclr    PORTH,BIT3   ; Turn on bit 3
0012 4D2410      21          bclr    PORTH,BIT4   ; Turn on bit 4
0015 4D2420      22          bclr    PORTH,BIT5   ; Turn on bit 5
0018 4D2440      23          bclr    PORTH,BIT6   ; Turn on bit 6
001B 4D2480      24          bclr    PORTH,BIT7   ; Turn on bit 7
001E 20E3        25          bra     LOOP        ; Do it forever
```

Example 7-21

What do you expect to see on the LEDs as the program in Example 4-24 runs?

Solution:

We expect to see the LEDs go out and then come on one at a time, starting from the right, until all are on and then to repeat for 8 times.

What do we actually see?

All lights will appear to be on because the BCLR and BSET instructions take only $0.5 \mu\text{s}$ to execute, much too fast for our eyes to respond. If you traced the program one step at a time, you would see the expected behavior.

Bit test and branch

- BRCLR/BRSET branch if selected bits of a memory byte are 0/1

Example: branch to label if bit 2 of memory byte M is 0 (clear)

```
brclr M,%00000100,label
```

↑
Memory
byte to test

Mask

↑
Branch address

Example: "stall" until bit 3 of memory byte M is 0

Here: brset M,%00000100,Here