Microcomputers I – CE 320

Mohammad Ghamari, Ph.D.
Electrical and Computer Engineering
Kettering University

Lecture 13 : Special Bit Instructions

Announcement

- Lecture 12 is uploaded on the blackboard.
- Homework Exercise is uploaded on the blackboard.

Today’s Topics

- Learn bit-set and bit-clear instructions
- Branch on bit instructions

Bit Test & Manipulate Instruction

Mnemonic	Function	Operation
BCLR <opr>[2], msk8	Clear bits in memory	$M \leftarrow (M) \bullet (\overline{mm})$
BITA <opr>[1]	Bit test A	$(A) \bullet (M)$
BITB <opr>[1]	Bit test B	$(B) \bullet (M)$
BSET <opr>[2], msk8	Set bits in memory	$M \leftarrow (M) + (mm)$

Bit Manipulate Instructions

- There are special instructions to **set** or **clear** bits in a **memory byte**.
  - **BSET** *addr, mask*
    - Set bits in a memory location to 1 or do not affect them
  - **BCLR** *addr, mask*
    - Clear bits in a memory location to 0 or do not affect them
- Note that …
  - They can only be used on data in **memory**.
  - Use AND and OR instructions for setting/clearing bits on **registers**.
  - Therefore, two operands are needed.
    - 1st : the address
    - 2nd : immediate mask value
  - In the mask byte,
    - 1 means to affect the bit
    - 0 means preserve the bit
- Examples
  - BSET 0,X, $81 ; $81 = %10000001
  - BCLR 0,Y, $33 ; $33 = %00110011

Sets the most significant and least significant bits of the memory location pointed to by index register X.

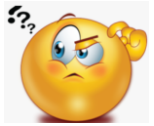
Clears the bits five, four, one, and zero of the memory location pointed to by index register Y.

Bit Test Instructions

- The BIT instruction is used to **set the CCR bits** to prepare for a branch instruction.
- It performs an **AND** operation affecting the CCR but discarding the result.

  - **BITA** ((A) · (M))
    - Bit test A
    - Perform an AND operation on the register A and a given mask, set the CCR bits, and discard the result.
  - **BITB** ((B) · (M))
    - Bit test B
    - Perform an AND operation on the register B and a given mask, set the CCR bits, and discard the result.
- Examples
  - **BITA** **#$44** ; $44 = %01000100
    - Tests the bit 6 and 2 of register A.
    - Updates Z and N bits of CCR accordingly.

BIT operation relates to the AND operation the same way that compares (CMP) relate to subtraction instructions (SUB).



Bit Condition Branch Instructions

- There are two instructions that **perform conditional branches based on the values of selected bits in memory**.

- These are **very useful when interacting with I/O hardware**, as we'll see later in the course.

- **BRCLR**, **BRSET**
  - Perform bitwise logical **AND** on the contents of the specified memory location and the mask supplied with the instruction.
  - **BRCLR**: branch if $(M) \cdot (mm) = 0$ [if selected bit(s) clear]
  - **BRSET**: branch if $(\overline{M}) \cdot (mm) = 0$ [if selected bit(s) set]

Bit Condition Branch Instructions

- **BRCLR** opr, msk, rel

- **BRSET** opr, msk, rel

opr: specifies the memory location to be checked and can be specified using direct, extended, and all indexed addressing modes.

msk: is an 8-bit mask that specifies the bits of the memory location to be checked. The bits to be checked correspond to those bit positions that are ones in the mask.

rel: is the branch offset and is specified in 8-bit relative mode.

Bit Condition Branch Instructions

Example1:

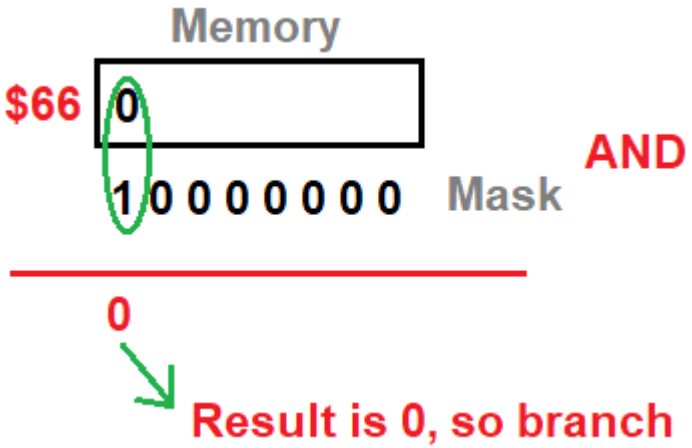
```
here  brclr   $66,$80,here
      ldd     $70
```

- **This instruction tells the HCS12 to perform bitwise logical AND on the contents of the specified memory location ($66) and the mask ($80) supplied with the instruction, then branch if the result is zero.**

- **The HCS12 will continue to execute the first instruction if the most significant bit of the memory location at $66 is 0. Otherwise, the next instruction will be executed.**

AND

INPUT		OUTPUT
A	B	
0	0	0
1	0	0
0	1	0
1	1	1



Mask: $80 **1**0000000 (binary)

Bit Condition Branch Instructions

Example 2:

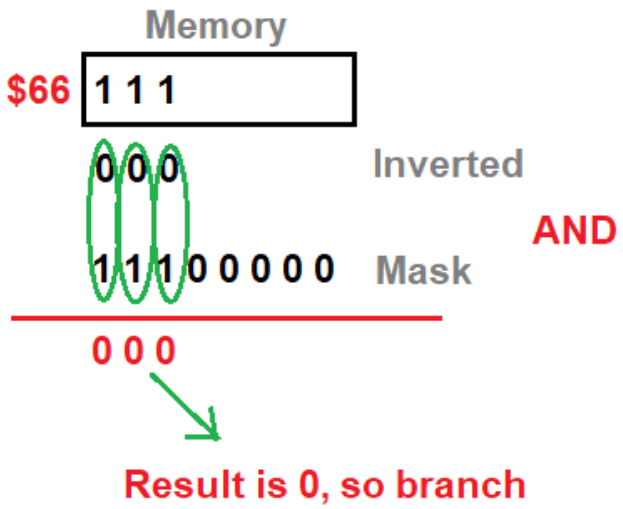
```
loop    inc     count
        …
        brset   $66,$e0,loop
        …
```

- **This instruction tells the HCS12 to perform the logical AND of the contents of the specified memory location ($66) inverted and the mask supplied with the instruction, then branch if the result is zero.**

- **The branch will be taken if the most significant three bits of the memory location at $66 are all ones.**

AND

INPUT		OUTPUT
A	B	
0	0	0
1	0	0
0	1	0
1	1	1



Mask: $e0 11100000 (binary)

Example 1

Two door sport car

- How to turn on the **cabin light** without affecting other bits?

	7	6	5	4	3	2	1	0
$0000	GBOXD	LEFTD	RGHTD	TRNKD	-	GBOXL	**CBNL**	TRNKL

Example 1

Two door sport car

- How to <u>turn on</u> the **cabin light** without affecting other bits?

Ans1:

Turn ON → Use **OR** with a proper mask byte

```
ldd     $00
oraa    #%00000010
staa    $00
```

	7	6	5	4	3	2	1	0
$0000	GBOXD	LEFTD	RGHTD	TRNKD	-	GBOXL	**CBNL**	TRNKL

Example 1

Two door sport car

- How to turn on the **cabin light** without affecting other bits?

Ans2: To **set** bits in a **memory byte,** we use **BSET** instruction

BSET *addr, mask*

(M) + (mm) ⇒ M

Set Bit(s) in Memory

BSET **$00, %00000010** **; $02**

	7	6	5	4	3	2	1	0
$0000	GBOXD	LEFTD	RGHTD	TRNKD	-	GBOXL	**CBNL**	TRNKL

Example 2

Two door sport car

- How to turn off the **glove box light** and **trunk light** without affecting other bits?

	7	6	5	4	3	2	1	0
$0000	GBOXD	LEFTD	RGHTD	TRNKD	-	**GBOXL**	CBNL	**TRNKL**

Example 2

Two door sport car

- How to <u>turn off</u> the **glove box light** and **trunk light** without affecting other bits?

- **Ans1:** Turn OFF → Use **AND** with a proper mask byte

```
ldaa    $00
anda    #%11111010
staa    $00
```

	7	6	5	4	3	2	1	0
$0000	GBOXD	LEFTD	RGHTD	TRNKD	-	**GBOXL**	CBNL	**TRNKL**

Example 2

Two door sport car

- How to <u>turn off</u> the **glove box light** and **trunk light** without affecting other bits?

Ans2: To **clear** bits in a **memory byte,** we use **BCLR** instruction

BCLR *addr, mask*

$(M) . (\overline{mm}) \Rightarrow M$

Clear Bit(s) in Memory

BCLR **$00, %00000101** **; $05**

	7	6	5	4	3	2	1	0
$0000	GBOXD	LEFTD	RGHTD	TRNKD	-	**GBOXL**	CBNL	**TRNKL**

Example 3

Two door sport car

- How to <u>turn on</u> the **cabin light** if either door is open (=the bit is set)?

	7	6	5	4	3	2	1	0
$0000	GBOXD	**LEFTD**	**RGHTD**	TRNKD	-	GBOXL	**CBNL**	TRNKL

Example 3

Two door sport car

- How to <u>turn on</u> the **cabin light** if either door is open (=the bit is set)?

```
            LDAA    $00
            BITA    #%01100000      ; #$60
            BNE     CBNLON
            BRA     SKIP
CBNLON:     BSET    $00, %00000010  ; $02
SKIP:
```

	7	6	5	4	3	2	1	0
$0000	GBOXD	**LEFTD**	**RGHTD**	TRNKD	-	GBOXL	**CBNL**	TRNKL

Example 4

Two door sport car

- How to turn off the cabin light if both doors are **closed**?

	7	6	5	4	3	2	1	0
$0000	GBOXD	**LEFTD**	**RGHTD**	TRNKD	-	GBOXL	**CBNL**	TRNKL

Example 4

Two door sport car

- How to turn off the cabin light if both doors are **closed**?

```
            BRCLR   $00, %01100000, CBNLOFF
            BRA     SKIP
CBNLOFF:    BCLR    $00, %00000010
SKIP:
```

	7	6	5	4	3	2	1	0
$0000	GBOXD	**LEFTD**	**RGHTD**	TRNKD	-	GBOXL	**CBNL**	TRNKL

Example 5

Two door sport car

- How to turn on the cabin light if both doors are **open**?

	7	6	5	4	3	2	1	0
$0000	GBOXD	**LEFTD**	**RGHTD**	TRNKD	-	GBOXL	**CBNL**	TRNKL

Example 5

Two door sport car

- How to <u>turn on</u> the cabin light if both doors are **open**?

```
            BRSET   $00,%01100000,CBNLON
            BRA     SKIP
CBNLON:     BSET    $00, %00000010          ; $02
SKIP:
```

	7	6	5	4	3	2	1	0
$0000	GBOXD	**LEFTD**	**RGHTD**	TRNKD	-	GBOXL	**CBNL**	TRNKL

Questions?

Wrap-up

What we've learned

- Bit set/clear instructions

- Bit condition branch instructions

What to Come

- Stack
- Subroutines