Classic AmigaOS Programming

An introduction

E. Th. van den Oosterkamp

ISBN: 9781690195153

Web site: www.edsa.uk/amiga

Author: edwin@edsa.uk

Corrections

Page 25, typo

The comment for the 3rd MOVEA example was “post-increment A5 by 2”, where A5 should be A0.

Page 38, 39, 40, 41, flag affected by the bit instructions

The text and comments with the examples speak of the C flag, while it is actually the Z flag that is used by the instructions.

The following pages in this document are the corrected pages of the book, where the corrections are marked in purple.

```
Operation:   Source → Destination
Syntax:      MOVE <ea>,<ea>
Sizes:       Byte, Word and Long.

CCR:         X N Z V C
             - * * 0 0

Source:      Dn An (xx).W (xx).L #imm (An) (An)+ -(An) d(An) d(An.Xn) d(PC) d(PC.Xn)
Destination: Dn An (xx).W (xx).L #imm (An) (An)+ -(An) d(An) d(An.Xn) d(PC) d(PC.Xn)

Examples:    MOVE.B   #$FF,D0         ; Move the 8 bit value $FF into register D0
             MOVE.L   label(PC),D1    ; Move the long at label(PC) to register D1
             MOVE.W   D0,(A0)+        ; Move the word in D0 to the address in A0 and
                                      ; post-increment A0 by 2.
```

Please note that the MOVE instruction does not allow for moving data to an address register. For this purpose the MOVEA instruction is to be used. A number of assemblers will automatically substitute MOVE with MOVEA when the destination of the MOVE is an address register. Please note that MOVEA does not change the CCR, where MOVE does - this could lead to a bug where the programmer uses MOVE with an address register and then expects the zero flag to be set if the pointer happens to be NULL - which will not happen since the assembler will silently replace the MOVE with MOVEA, causing the zero flag to never be changed.

MOVEA (Move address)

Move the data in the source to an address register. If the data is word sized then it will be sign-extended to make it long sized. The MOVEA instruction will not modify the flags of the CCR.

```
Operation:   Source → Destination
Syntax:      MOVEA <ea>,An
Sizes:       Word and Long.

CCR:         X N Z V C
             - - - - -

Source:      Dn An (xx).W (xx).L #imm (An) (An)+ -(An) d(An) d(An.Xn) d(PC) d(PC.Xn)
Destination: Dn An (xx).W (xx).L #imm (An) (An)+ -(An) d(An) d(An.Xn) d(PC) d(PC.Xn)

Examples:    MOVEA.L  4.W,A6          ; Move the long stored at address $4 into A6.
             MOVEA.L  #4,A0           ; Move the long value $00000004 to register A0.
             MOVEA.W  (A0)+,A5        ; Move the word at the address in A0 to A5 and
                                      ; post-increment A0 by 2.
```

MOVEM (Move multiple registers)

Move multiple registers in one operation. Either the source is a list of registers or the destination is. Data can be moved word or long sized and when the operation moves words into registers then they will be sign-extended in the registers. This sign-extending will happen for data registers as

The rotation works by shifting all bits into one direction while the bit that shifts out on one end is shifted back in at the other end. This bit is also used to update the C flag in the CCR. For example, the ROR instruction will shift all bits to the right, which means that the lowest significant bit is shifted out. This bit is used to update the C flag of the CCR and is also used for the new value of the most significant bit.

```
Operation:   Destination = Destination rotated by <source>
Syntax:      ROL Dn,Dn
             ROR Dn,Dn
             ROL #Imm,Dn
             ROR #Imm,Dn
             ROL <ea>
             ROR <ea>
Sizes:       Byte, Word and Long.

CCR:         X N Z V C
             - * * 0 *

Destination: Dn An (xx).W (xx).L #imm (An) (An)+ -(An) d(An) d(An.Xn) d(PC) d(PC.Xn)

Examples:    ROL.L    #4,D0           ; Rotate all the bits in D0 4 times to the left.
             ROR      (A0)            ; Rotate the word pointed at by A0 to the right once.
```

NOT (Logical complement)

Perform a logical complement on the value in the destination and store the result in the destination. The result will also be used to update the CCR.

```
Operation:   Destination = !Destination
Syntax:      NOT <ea>
Sizes:       Byte, Word and Long.

CCR:         X N Z V C
             - * * 0 0

Destination: Dn An (xx).W (xx).L #imm (An) (An)+ -(An) d(An) d(An.Xn) d(PC) d(PC.Xn)

Examples:    NOT.B    D0        ; Complement the bottom 8 bits in D0.
             NOT      (A0)      ; Complement the word pointed at by the address in A0.
```

Bit instructions

BSET (Bit test and set)

Test the current status of a specified bit in the destination and reflect the bit's status with the Z flag of the CCR, then set the specified bit in the destination to 1. In other words, this instruction sets the specified bit and uses the Z flag of the CCR to show the bit's previous status.

If the destination is a register then the bit number in the source is taken as modulo 32. If the destination is a memory location then the operation will be byte sized and the bit number in the source will be modulo 8. In both cases bit zero is the least significant bit of the destination. The source is either a data register or an immediate literal.

```
Operation:   Z-flag = Current bit status.
             new bit status = 1.
Syntax:      BSET Dn,<ea>
             BSET #Imm,<ea>
Sizes:       Byte and Long.

CCR:         X N Z V C
             - - * - -

Source:      Dn An (xx).W (xx).L #imm (An) (An)+ -(An) d(An) d(An.Xn) d(PC) d(PC.Xn)
Destination: Dn An (xx).W (xx).L #imm (An) (An)+ -(An) d(An) d(An.Xn) d(PC) d(PC.Xn)

Examples:    BSET    #31,D1         ; Set the most significant bit of D1 and update
                                    ; the Z flag to the bit’s previous value.
```

BCLR (Bit test and clear)

Test the current status of a specified bit in the destination and reflect the bit's status with the Z flag of the CCR, then set the specified bit in the destination to 0. In other words, this instruction clears the specified bit and uses the Z flag of the CCR to show the bit's previous status.

If the destination is a register then the bit number in the source is taken as modulo 32. If the destination is a memory location then the operation will be byte sized and the bit number in the source will be modulo 8. In both cases bit zero is the least significant bit of the destination. The source is either a data register or an immediate literal.

```
Operation:   Z-flag = Current bit status.
             new bit status = 0.
Syntax:      BCLR Dn,<ea>
             BCLR #Imm,<ea>
Sizes:       Byte and Long.

CCR:         X N Z V C
             - - * - -

Source:      Dn An (xx).W (xx).L #imm (An) (An)+ -(An) d(An) d(An.Xn) d(PC) d(PC.Xn)
Destination: Dn An (xx).W (xx).L #imm (An) (An)+ -(An) d(An) d(An.Xn) d(PC) d(PC.Xn)

Examples:    BCLR    #0,D1          ; Clear the least significant bit of D1 and
                                    ; update the Z flag to the bit’s previous value.
```

BCHG (Bit test and change)

Test the current status of a specified bit in the destination and reflect the bit's status with the Z flag of the CCR, then negate the specified bit in the destination. In other words, this instruction changes the specified bit to its opposite value and uses the Z flag of the CCR to show the bit's previous status.

If the destination is a register then the bit number in the source is taken as modulo 32. If the destination is a memory location then the operation will be byte sized and the bit number in the source will be modulo 8. In both cases bit zero is the least significant bit of the destination. The source is either a data register or an immediate literal.

```
Operation:   Z-flag = Current bit status.
             new bit status = ! current bit status.
Syntax:      BCHG Dn,<ea>
             BCHG #Imm,<ea>
Sizes:       Byte and Long.

CCR:         X N Z V C
             - - * - -

Source:      Dn An (xx).W (xx).L #imm (An) (An)+ -(An) d(An) d(An.Xn) d(PC) d(PC.Xn)
Destination: Dn An (xx).W (xx).L #imm (An) (An)+ -(An) d(An) d(An.Xn) d(PC) d(PC.Xn)

Examples:    BCHG    #2,D1         ; Change bit 2 of D1 and update the Z flag to
                                   ; the previous value of bit 2.
```

BTST (Bit test)

Test the current status of a specified bit in the destination and reflect the bit's status with the Z flag of the CCR. The destination will remain unchanged.

If the destination is a register then the bit number in the source is taken as modulo 32. If the destination is a memory location then the operation will be byte sized and the bit number in the source will be modulo 8. In both cases bit zero is the least significant bit of the destination. The source is either a data register or an immediate literal.

```
Operation:   Z-flag = Current bit status.
Syntax:      BTST Dn,<ea>
             BTST #Imm,<ea>
Sizes:       Byte and Long.

CCR:         X N Z V C
             - - * - -

Source:      Dn An (xx).W (xx).L #imm (An) (An)+ -(An) d(An) d(An.Xn) d(PC) d(PC.Xn)
Destination: Dn An (xx).W (xx).L #imm (An) (An)+ -(An) d(An) d(An.Xn) d(PC) d(PC.Xn)
```

```
Examples:    BTST     #31,D0          ; Test bit 31 of D0 and set the Z flag accordingly.
             BTST     #7,(A0)         ; Test bit 7 of the byte pointed to by A0 and
                                      ; set the Z flag accordingly.
```

Comparison instructions

CMP (Compare)

Compare the source and destination by subtracting the source from the destination and setting the CCR flags according to the result. The result is then discarded and the destination will therefore remain unchanged. The destination is always one of the data registers.

```
Operation:   Destination - Source
Syntax:      CMP <ea>,Dn
Sizes:       Byte, Word and Long.

CCR:         X N Z V C
             - * * * *

Source:      Dn An (xx).W (xx).L #imm (An) (An)+ -(An) d(An) d(An.Xn) d(PC) d(PC.Xn)
Destination: Dn An (xx).W (xx).L #imm (An) (An)+ -(An) d(An) d(An.Xn) d(PC) d(PC.Xn)

Examples:    CMP.L    #$431,D0        ; Compare 32 bits value $00000431 with D0.
             CMP.B    (A0),D2         ; Compare the byte pointed at by A0 with the
                                      ; bottom 8 bits of D2.
```

CMPA (Compare Address)

Compare the source and destination by subtracting the source from the destination and setting the CCR flags according to the result. The result is then discarded and the destination will therefore remain unchanged. The destination is always one of the address registers.

```
Operation:   Destination - Source
Syntax:      CMPA <ea>,An
Sizes:       Word and Long.

CCR:         X N Z V C
             - * * * *

Source:      Dn An (xx).W (xx).L #imm (An) (An)+ -(An) d(An) d(An.Xn) d(PC) d(PC.Xn)
Destination: Dn An (xx).W (xx).L #imm (An) (An)+ -(An) d(An) d(An.Xn) d(PC) d(PC.Xn)

Examples:    CMPA.L   #$431,A0        ; Compare 32 bits value $00000431 with A0.
             CMPA.W   (A0),A2         ; Compare the word pointed at by A0 with the
                                      ; bottom 16 bits of A2.
```