# AmigaFlight Integer Arithmetic Instructions

Andrew Duffy Morris

| COLLABORATORS | | | |
|---|---|---|---|
| | *TITLE* :<br><br>AmigaFlight Integer Arithmetic Instructions | | |
| *ACTION* | *NAME* | *DATE* | *SIGNATURE* |
| WRITTEN BY | Andrew Duffy Morris | July 20, 2024 | |

| REVISION HISTORY | | | |
|---|---|---|---|
| NUMBER | DATE | DESCRIPTION | NAME |
| | | | |

# Contents

# Chapter 1

# AmigaFlight Integer Arithmetic Instructions

## 1.1 AmigaFlight® Help: Integer Arithmetic Instructions

```
Integer Arithmetic Instructions
===============================

The arithmetic operations include the four basic operations of add,
subtract, multiply, and divide, as well as arithmetic compare, clear, and
negate.  The add and subtract operations are available for both address and
data operations, with data operations accepting all operand sizes.  Address
operations are limited to legal address size operands (16 or 32 bits).
Data, address, and memory compare operations are also available.  The clear
and negate instructions may be used on all size of data operands.


  Add Instructions
  ----------------
  ADD                 Add Binary
  ADDA <ea>,An        Add Address
  ADDI #d,<ea>        Add Immediate
  ADDQ #d,<ea>        Add Quick
  ADDX                Add Extended

  Clear
  -----
  CLR <ea>            Clear an Operand

  Compare Instructions
  --------------------
  CMP <ea>,Dn         Compare
  CMPA <ea>,An        Compare Address
  CMPI #d,<ea>        Compare Immediate
  CMPM (Ay)+,(Ax)+    Compare Memory

  Divide Instructions
  -------------------
  DIVS <ea>,Dn        Signed Divide
  DIVU <ea>,Dn        Unsigned Divide

  Sign Extend
  -----------
  EXT Dn            Sign Extend

  Multiply Instructions
  ---------------------
  MULS <ea>,Dn      Signed Multiply
  MULU <ea>,Dn      Unsigned Multiply

  Negate Instructions
  -------------------
  NEG <ea>          Negate
  NEGX <ea>         Negate with Extend

  Subtract Instructions
  ---------------------
  SUB               Subtract Binary
  SUBA <ea>,An      Subtract Address
  SUBI #d,<ea>      Subtract Immediate
  SUBQ #d,<ea>      Subtract Quick
  SUBX              Subtract with Extend

  Testing
  -------
  TAS               Test and Set an Operand
  TST               Test an Operand
```

## 1.2 AmigaFlight® Help: Add Binary

```
ADD Add Binary
==================

  Add the source operand to the  destination  operand  using  binary
  arithmetic. Store the result in the destination operand.

  Destn + Source  -> Destn


Assembler Syntax
----------------
  ADD{.[B/W/L]} <ea>,Dn
  ADD{.[B/W/L]} Dn,<ea>
  ADD{.[W/L]} <ea>,An
  ADD{.[B/W/L]} #<data>,<ea>


Addressing Modes
----------------
  Mode             Source  Destination

  Data Register Direct      * *
  Address Register Direct     - -
  Address Register Indirect   - *
  Postincrement Register Indirect   - *
  Predecrement Register Indirect    - *
  Register Indirect with Offset   - *
  Register Indirect with Index    - *
```

| | | |
|---|---|---|
| Absolute Short | - | * |
| Absolute Long | - | * |
| P.C. Relative with Offset | - | - |
| P.C. Relative with Index | - | - |
| Immediate | - | - |

| Mode | Source | Destination |
|---|---|---|
| Data Register Direct | * | * |
| Address Register Direct | * | - |
| Address Register Indirect | * | - |
| Postincrement Register Indirect | * | - |
| Predecrement Register Indirect | * | - |
| Register Indirect with Offset | * | - |
| Register Indirect with Index | * | - |
| Absolute Short | * | - |
| Absolute Long | * | - |
| P.C. Relative with Offset | * | - |
| P.C. Relative with Index | * | - |
| Immediate | * | - |

## Data Size
---------

Byte, Word or Long

## Status Flags
------------

| | |
|---|---|
| N | Set if negative |
| Z | Set if zero |
| V | Set if overflow |
| C | Set if carry is generated |
| X | Set same as carry |

## Instruction Size and Cycles to Execute
--------------------------------------

| BYTE/WORD | <ea>,Dn | | Dn,<ea> | |
|---|---|---|---|---|
| <ea> | # | p | # | P |
| Dn | 2 | 4 | | |
| An (word) | 2 | 4 | | |
| (An) | 2 | 8 | 2 | 12 |
| (An)+ | 2 | 8 | 2 | 12 |
| -(An) | 2 | 10 | 2 | 14 |
| d16(An) | 4 | 12 | 4 | 16 |
| d8(An,Ri) | 4 | 14 | 4 | 18 |
| Abs short | 4 | 12 | 4 | 16 |
| Abs long | 6 | 16 | 6 | 20 |
| d16(PC) | 4 | 12 | | |
| d8(PC,Ri) | 4 | 14 | | |
| Immediate | 4 | 8 | | |

# = no. of program bytes
p = no. of instruction clock periods

```
  LONG    <ea>,Dn   Dn,<ea>
  <ea>    #     p   #    P
  Dn    2     8
  An  (word)  2     8
  (An)    2    14   2   20
  (An)+   2    14   2   20
  -(An)   2    16   2   22
  d16(An)   4    18   4   24
  d8(An,Ri) 4    20   4   26
  Abs short 4    18   4   24
  Abs long  6    22   6   28
  d16(PC)   4    18
  d8(PC,Ri) 4    20
  Immediate 4    14

  # = no. of program bytes
  p = no. of instruction clock periods
```

## 1.3 AmigaFlight® Help: Add Address

```
ADDA  Add Address
===================

  Add the source operand to the  destination  operand  using  binary
  arithmetic.  Store  the  result  in  the destination operand. This
  opcode  is  a  subset  of  the  ADD  opcode, and requires that the
  destination be an address register.

  Destn + Source  -> Destn


Assembler Syntax
----------------
  ADDA{.[W/L]}  <ea>,An


Addressing Modes
----------------
  Mode             Source  Destination

  Data Register Direct      * -
  Address Register Direct     * *
  Address Register Indirect   * -
  Postincrement Register Indirect   * -
  Predecrement Register Indirect    * -
  Register Indirect with Offset   * -
  Register Indirect with Index    * -
  Absolute Short        * -
  Absolute Long       * -
  P.C. Relative with Offset   * -
  P.C. Relative with Index    * -
  Immediate       * -


Data Size
```

```
---------
  Word or Long


Status Flags
------------
  N  Not affected
  Z  Not affected
  V  Not affected
  C  Not affected
  X  Not affected


Instruction Size and Cycles to Execute
--------------------------------------
      Word    Long
  <ea>    #    p    #    p
  Dn    2    8    2    8
  An    2    8    2    8
  (An)    2   12    2   14
  (An)+   2   12    2   14
  -(An)   2   14    2   16
  d16(An)   4   16    4   18
  d8(An,Ri) 4   18    4   20
  Abs short 4   16    4   18
  Abs long  6   20    6   22
  d16(PC)   4   16    4   18
  d8(PC,Ri) 4   18    4   20
  Immediate 4   12    6   14

  # = no. of program bytes
  p = no. of instruction clock periods
```

## 1.4 AmigaFlight® Help: Add Immediate

```
ADDI  Add Immediate
=====================

  Add the source operand  to the destination  operand  using  binary
  arithmetic.  Store  the result in  the  destination  operand. This
  opcode is a subset of the ADD opcode, and requires that the source
  be an immediate value.

  Destn + Immediate Data -> Destn


Assembler Syntax
----------------
  ADDI{.[B/W/L]}  #<data>,<ea>


Addressing Modes
----------------
  Mode             Source  Destination
  Data Register Direct      - *
  Address Register Direct     - -
  Address Register Indirect   - *
  Postincrement Register Indirect   - *
  Predecrement Register Indirect    - *
  Register Indirect with Offset   - *
  Register Indirect with Index    - *
  Absolute Short        - *
  Absolute Long       - *
  P.C. Relative with Offset   - -
  P.C. Relative with Index    - -
  Immediate       * -


Data Size
---------
  Byte, Word or Long


Status Flags
------------
  N  Set if negative
  Z  Set if zero
  V  Set if overflow
  C  Set if carry is generated
  X  Set same as carry


Instruction Size and Cycles to Execute
--------------------------------------
      Byte/Word Long
  <ea>    #    p    #    p
  Dn    4    8    6   16
  (An)    4   16    6   28
  (An)+   4   16    6   28
  -(An)   4   18    6   30
  d16(An)   6   20    8   32
  d8(An,Ri) 6   22    8   34
  Abs short 6   20    8   32
  Abs long  8   24    10   36

  # = no. of program bytes
  p = no. of instruction clock periods
```

## 1.5 AmigaFlight® Help: Add Quick

```
ADDQ  Add Quick
=================

  Add the source operand to the  destination  operand  using  binary
  arithmetic.  Store  the  result in  the  destination operand. This
  opcode requires that the source be an immediate  value between one
  and eight.

  Destn + Immediate Data -> Destn where the immediate data may range
  from 1 to 8


Assembler Syntax
----------------
  ADDQ{.[B/W/L]}  #<data>,<ea>


Addressing Modes
----------------
  Mode             Source  Destination

  Data Register Direct      - *
  Address Register Direct     - *
  Address Register Indirect   - *
  Postincrement Register Indirect   - *
  Predecrement Register Indirect    - *
  Register Indirect with Offset   - *
  Register Indirect with Index    - *
  Absolute Short        - *
  Absolute Long       - *
  P.C. Relative with Offset   - -
  P.C. Relative with Index    - -
  Immediate       * -


Data Size
---------
  Byte, Word or Long


Status Flags
------------
  N  Set if negative
  Z  Set if zero
  V  Set if overflow
  C  Set if carry is generated
  X  Set same as carry

  The  Condition  Codes  are  not  affected  if  <ea>  is an address
  register


Instruction Size and Cycles to Execute
--------------------------------------
      Byte/Word Long
  <ea>    #    p    #    p
  Dn    2    4    2    8
  An (not byte) 2    4    2    8
  (An)    2   12    2   20
  (An)+   2   12    2   20
  -(An)   2   14    2   22
  d16(An)   4   16    4   24
  d8(An,Ri) 4   18    4   26
  Abs short 4   16    4   24
  Abs long  6   20    6   28
  # = no. of program bytes
  p = no. of instruction clock periods
```

## 1.6 AmigaFlight® Help: Add Extended

```
ADDX  Add Extended
====================

  Add  the  source operand  and  the extend bit  to  the destination
  operand  using  binary  arithmetic.  Store  the  result   in   the
  destination operand.

  Destn + Source + X -> Destn


Assembler Syntax
----------------
  ADDX{.[B/W/L]}  Dy,Dx
  ADDX{.[B/W/L]}  -(Ay),-(Ax)


Data Size
---------
  Byte, Word or Long


Status Flags
------------
  N  Set if result is zero
  Z  Cleared if result non-zero, else unchanged
  V  Set if overflow
  C  Set if carry is generated
  X  Set same as carry


Instruction Size and Cycles to Execute
--------------------------------------
      Byte/Word Long
      #    p    #    p
  Dy,Dx    2    4    2    8
  -(Ay),-(Ax) 2   18    2   30

  # = no. of program bytes
  p = no. of instruction clock periods
```

## 1.7 AmigaFlight® Help: Clear an Operand

```
CLR Clear an Operand
========================

  The specified destination address is cleared to 0.
  0 -> Destination


Assembler Syntax
----------------
  CLR{.[B/W/L]} <ea>

  <ea> - data alterable


Addressing Modes
----------------
  Mode             Source  Destination

  Data Register Direct      - *
  Address Register Direct     - -
  Address Register Indirect   - *
  Postincrement Register Indirect   - *
  Predecrement Register Indirect    - *
  Register Indirect with Offset   - *
  Register Indirect with Index    - *
  Absolute Short        - *
  Absolute Long       - *
  P.C. Relative with Offset   - -
  P.C. Relative with Index    - -
  Immediate       - -


Data Size
---------
  Byte, Word, Long


Status Flags
------------
  N  Always cleared
  Z  Always set
  V  Always cleared
  C  Always cleared
  X  Not affected


Instruction Size and Cycles to Execute
--------------------------------------
  Size...   Byte/Word Long
  <ea>    #    p    #    p
  Dn    2    4    2    6
  (An)    2   12    2   20
  (An)+   2   12    2   20
  -(An)   2   14    2   22
  d16(An)   4   16    4   24
  d8(An,Ri) 4   18    4   26
  Abs short 4   16    4   24
  Abs long  6   20    6   28

  # = no. of program bytes
  p = no. of instruction clock periods
```

## 1.8 AmigaFlight® Help: Compare

```
CMP Compare
===============

  Subtract the  source  operand from the destination operand and set
  the condition codes accordingly. This instruction does  not modify
  the destination address.

  Destn - Source


Assembler Syntax
----------------
  CMP{.[B/W/L]} <ea>,Dn
  CMP{.[W/L]} <ea>,An
  CMP{.[B/W/L]} #<data>,<ea>
  CMP{.[B/W/L]} (Ay)+,(Ax)+

  <ea> - all modes


Addressing Modes
----------------
  Mode             Source  Destination

  Data Register Direct      * *
  Address Register Direct     * -
  Address Register Indirect   * -
  Postincrement Register Indirect   * -
  Predecrement Register Indirect    * -
  Register Indirect with Offset   * -
  Register Indirect with Index    * -
  Absolute Short        * -
  Absolute Long       * -
  P.C. Relative with Offset   * -
  P.C. Relative with Index    * -
  Immediate       * -


Data Size
---------
  Byte, Word or Long


Status Flags
------------
  N  Set if negative
  Z  Set if zero
  V  Set if overflow
  C  Set if borrow, else clear
  X  Not affected
```

```
Instruction Size and Cycles to Execute
--------------------------------------
  Size...   Byte/Word Long
  <ea>    #    p    #    p
  Dn    2    4    2    6
  An (not byte) 2    4    2    6
  (An)    2    8    2   14
  (An)+   2    8    2   14
  -(An)   2   10    2   16
  d16(An)   4   12    4   18
  d8(An,Ri) 4   14    4   20
  Abs short 4   12    4   18
  Abs long  6   16    6   22
  d16(PC)   4   12    4   18
  d8(PC,Ri) 4   14    4   20
  Immediate 4    8    6   14

  # = no. of program bytes
  p = no. of instruction clock periods
```

## 1.9 AmigaFlight® Help: Compare Address

```
CMPA  Compare Address
=======================

  Subtract  the  source operand from the destination operand and set
  the condition codes accordingly. This instruction does  not modify
  the destination address. This opcode requires that the destination
  be an address register.

  Destn - Source


Assembler Syntax
----------------
  CMPA{.[W/L]}  <ea>,An

  <ea> - all modes


Addressing Modes
----------------
  Mode             Source  Destination

  Data Register Direct      * *
  Address Register Direct     * -
  Address Register Indirect   * -
  Postincrement Register Indirect   * -
  Predecrement Register Indirect    * -
  Register Indirect with Offset   * -
  Register Indirect with Index    * -
  Absolute Short        * -
  Absolute Long       * -
  P.C. Relative with Offset   * -
  P.C. Relative with Index    * -
  Immediate       * -


Data Size
---------
  Word or Long


Status Flags
------------
  N  Set if negative
  Z  Set if zero
  V  Set if overflow
  C  Set if borrow, else cleared
  X  Not affected


Instruction Size and Cycles to Execute
--------------------------------------
  Size...   Word    Long
  <ea>    #    p    #    p
  Dn    2    6    2    6
  An    2    6    2    6
  (An)    2   10    2   14
  (An)+   2   10    2   14
  -(An)   2   12    2   16
  d16(An)   4   14    4   18
  d8(An,Ri) 4   16    4   20
  Abs short 4   14    4   18
  Abs long  6   18    6   22
  d16(PC)   4   14    4   18
  d8(PC,Ri) 4   16    4   20
  Immediate 4   10    6   14

  # = no. of program bytes
  p = no. of instruction clock periods
```

## 1.10 AmigaFlight® Help: Compare Immediate

```
CMPI  Compare Immediate
=========================

  Subtract the  source  operand from the destination operand and set
  the condition codes accordingly. This instruction does  not modify
  the  destination  address.  This  opcode  is  a  subset of the CMP
  opcode, and requires that the source be an immediate value.

  Destn - Immediate Data


Assembler Syntax
----------------
  CMPI{.[B/W/L]}  #<data>,<ea>
  <ea> - data alterable


Addressing Modes
----------------
  Mode             Source  Destination

  Data Register Direct      - *
  Address Register Direct     - -
  Address Register Indirect   - *
  Postincrement Register Indirect   - *
  Predecrement Register Indirect    - *
  Register Indirect with Offset   - *
  Register Indirect with Index    - *
  Absolute Short        - *
  Absolute Long       - *
  P.C. Relative with Offset   - -
  P.C. Relative with Index    - -
  Immediate       * -


Data Size
---------
  Byte, Word or Long


Status Flags
------------
  N  Set if negative
  Z  Set if zero
  V  Set if overflow
  C  Set if borrow, else cleared
  X  Not affected


Instruction Size and Cycles to Execute
--------------------------------------
  Size...   Byte/Word Long
  <ea>    #    p    #    p
  Dn    4    8    6   14
  (An)    4   12    6   20
  (An)+   4   12    6   20
  -(An)   4   14    6   22
  d16(An)   6   16    8   24
  d8(An,Ri) 6   18    8   26
  Abs short 6   16    8   24
  Abs long  8   20    10  28

  # = no. of program bytes
  p = no. of instruction clock periods
```

## 1.11 AmigaFlight® Help: Compare Memory

```
CMPM  Compare Memory
=====================
  Subtract the  source operand  from the destination operand and set
  the condition codes accordingly. This instruction does  not modify
  the  destination  address.  This  opcode  is  a  subset of the CMP
  opcode, and requires that the source and destination  operands are
  both indirect with post increment mode.

  Destn - Source


Assembler Syntax
----------------
  CMPM{.[B/W/L]}  (Ay)+,(Ax)+


Addressing Modes
----------------
  Mode             Source  Destination

  Data Register Direct      - -
  Address Register Direct     - -
  Address Register Indirect   - -
  Postincrement Register Indirect   - *
  Predecrement Register Indirect    - -
  Register Indirect with Offset   - -
  Register Indirect with Index    - -
  Absolute Short        - -
  Absolute Long       - -
  P.C. Relative with Offset   - -
  P.C. Relative with Index    - -
  Immediate       - -


Data Size
---------
  Byte, Word, Long


Status Flags
------------
  N  Set if negative
  Z  Set if zero
  V  Set if overflow
  C  Set if borrow, else cleared
  X  Not affected


Instruction Size and Cycles to Execute
--------------------------------------
      # p
  Byte/Word 2 12
  Long    2 20

  # = no. of program bytes
  p = no. of instruction clock periods
```

## 1.12 AmigaFlight® Help: Signed Divide

```
DIVS  Signed Divide
=====================

  Divide the source operand by the destination operand using a
  signed divide. Store the result in the destination operand. The
  destination operand is expected to be a 32-bit value, and the
  source operand is expected to be a 16-bit value. The 16-bit
  quotient is placed in the lower 16 bits of the destination
  operand. The 16-bit remainder is placed in the upper 16 bits of
  the destination operand. Dividing by zero will cause a processor
  trap. If overflow is set, the operands remain unaffected.

  Destn/Source -> Destn


Assembler Syntax
----------------
  DIVS{.W}  <ea>,Dn

  <ea> - data


Addressing Modes
----------------
  Mode             Source  Destination

  Data Register Direct      * *
  Address Register Direct     - -
  Address Register Indirect   * -
  Postincrement Register Indirect   * -
  Predecrement Register Indirect    * -
  Register Indirect with Offset   * -
  Register Indirect with Index    * -
  Absolute Short        * -
  Absolute Long       * -
  P.C. Relative with Offset   * -
  P.C. Relative with Index    * -
  Immediate       * -


Data Size
---------
  Word


Status Flags
------------
  N  Set if quotient -ve, else cleared but undefined if overflow
  Z  Set if quotient = 0, else cleared but undefined if overflow
  V  Set if overflow
  C  Always cleared
  X  Not affected
```

```
Instruction Size and Cycles to Execute
--------------------------------------
  <ea>    # p
  Dn    2 <158
  (An)    2 <162
  (An)+   2 <162
  -(An)   2 <164
  d16(An)   4 <166
  d8(An,Ri) 4 <168
  Abs short 4 <166
  Abs long  6 <170
  d16(PC)   4 <166
  d8(PC,Ri) 4 <168
  Immediate 4 <162

  # = no. of program bytes
  p = no. of instruction clock periods
```

## 1.13 AmigaFlight® Help: Unsigned Divide

```
DIVU  Unsigned Divide
=======================

  Divide  the  source  operand  by  the  destination operand using a
  unsigned divide. Store the result in the destination  operand. The
  destination  operand  is  expected to be a  32-bit value, and  the
  source  operand  is expected  to  be  a 16-bit  value.  The 16-bit
  quotient  is  placed  in  the  lower  16  bits of  the destination
  operand. The 16-bit remainder is  placed in  the upper 16  bits of
  the destination operand. Dividing by  zero will cause  a processor
  trap. If overflow is set, the operands remain unaffected.

  Destn/Source -> Destn


Assembler Syntax
----------------
  DIVU{.W}  <ea>,Dn

  <ea> - data


Addressing Modes
----------------
  Mode             Source  Destination

  Data Register Direct      * *
  Address Register Direct     - -
  Address Register Indirect   * -
  Postincrement Register Indirect   * -
  Predecrement Register Indirect    * -
  Register Indirect with Offset   * -
  Register Indirect with Index    * -
  Absolute Short        * -
  Absolute Long       * -
  P.C. Relative with Offset   * -
  P.C. Relative with Index    * -
  Immediate       * -


Data Size
---------
  Word


Status Flags
------------
  N  Set if most significant bit of quotient is set, else cleared
     but undefined if overflow
  Z  Set if quotient = 0, else cleared but undefined if overflow
  V  Set if overflow
  C  Always cleared
  X  Not affected


Instruction Size and Cycles to Execute
--------------------------------------
  <ea>    # p
  Dn    2 <140
  (An)    2 <144
  (An)+   2 <144
  -(An)   2 <146
  d16(An)   4 <148
  d8(An,Ri) 4 <150
  Abs short 4 <148
  Abs long  6 <152
  d16(PC)   4 <148
  d8(PC,Ri) 4 <150
  Immediate 4 <144

  # = no. of program bytes
  p = no. of instruction clock periods
```

## 1.14 AmigaFlight® Help: Sign Extend

```
EXT Sign Extend
===================

  Extend the sign bit of a register from an 8-bit value  to a 16-bit
  value, EXT.W,  or  from  a 16-bit value to 32-bit value, EXT.L. If
  the  instruction EXT.W is used,  then  bit  7  is copied into bits
  8-15. If the instruction EXT.L is used, bit 15 is copied into bits
  16-31.

  Sign Extended Destn -> Destn


Assembler Syntax
----------------
  EXT{.[W/L]} Dn
```

```
Addressing Modes
----------------
  Mode             Source  Destination

  Data Register Direct      - -
  Address Register Direct     - -
  Address Register Indirect   - -
  Postincrement Register Indirect   - -
  Predecrement Register Indirect    - -
  Register Indirect with Offset   - -
  Register Indirect with Index    - -
  Absolute Short        - -
  Absolute Long       - -
  P.C. Relative with Offset   - -
  P.C. Relative with Index    - -
  Immediate       - -


Data Size
---------
  Word, Long


Status Flags
------------
  N  Set if negative
  Z  Set if zero
  V  Always cleared
  C  Always cleared
  X  Not affected

Instruction Size and Cycles to Execute
--------------------------------------
      # p
      2 4

  # = no. of program bytes
  p = no. of instruction clock periods
```

## 1.15 AmigaFlight® Help: Signed Multiply

```
MULS  Signed Multiply
=======================

  Multiply the source operand and the destination operand generating
  a signed value. Store the result in the destination  operand. Both
  operands are expected to be  16-bit  values, and  the  destination
  operand receives a 32-bit result.

  Destn x Source -> Destn


Assembler Syntax
```

```
----------------
  MULS{.W}  <ea>,Dn

  <ea> - data


Addressing Modes
----------------
  Mode             Source  Destination

  Data Register Direct      * *
  Address Register Direct     - -
  Address Register Indirect   * -
  Postincrement Register Indirect   * -
  Predecrement Register Indirect    * -
  Register Indirect with Offset   * -
  Register Indirect with Index    * -
  Absolute Short        * -
  Absolute Long       * -
  P.C. Relative with Offset   * -
  P.C. Relative with Index    * -
  Immediate       * -


Data Size
---------
  Word


Status Flags
------------
  N  Set if negative
  Z  Set if zero
  V  Always cleared
  C  Always cleared
  X  Not affected


Instruction Size and Cycles to Execute
--------------------------------------
  <ea>    # p
  Dn    2 <70
  (An)    2 <74
  (An)+   2 <74
  -(An)   2 <76
  d16(An)   4 <78
  d8(An,Ri) 4 <80
  Abs short 4 <78
  Abs long  6 <82
  d16(PC)   4 <78
  d8(PC,Ri) 4 <80
  Immediate 4 <74

  # = no. of program bytes
  p = no. of instruction clock periods
```

## 1.16 AmigaFlight® Help: Unsigned Multiply

```
MULU  Unsigned Multiply
=========================

  Multiply the source operand and the destination operand generating
  an unsigned value. Store the result in  the  destination  operand.
  Both   operands  are   expected  to  be  16-bit  values,  and  the
  destination operand receives a 32-bit result.

  Destn x Source -> Destn


Assembler Syntax
----------------
  MULU{.W}  <ea>,Dn

  <ea> - data


Addressing Modes
----------------
  Mode             Source  Destination

  Data Register Direct      * *
  Address Register Direct     - -
  Address Register Indirect   * -
  Postincrement Register Indirect   * -
  Predecrement Register Indirect    * -
  Register Indirect with Offset   * -
  Register Indirect with Index    * -
  Absolute Short        * -
  Absolute Long       * -
  P.C. Relative with Offset   * -
  P.C. Relative with Index    * -
  Immediate       * -


Data Size
---------
  Word


Status Flags
------------
  N  Set if most significant bit of result is set, else cleared
  Z  Set if zero
  V  Always cleared
  C  Always cleared
  X  Not affected


Instruction Size and Cycles to Execute
--------------------------------------
  <ea>    # p
  Dn    2 <70
  (An)     2 <74
  (An)+    2 <74
  -(An)    2 <76
  d16(An)   4 <78
  d8(An,Ri) 4 <80
  Abs short 4 <78
  Abs long  6 <82
  d16(PC)   4 <78
  d8(PC,Ri) 4 <80
  Immediate 4 <74

  # = no. of program bytes
  p = no. of instruction clock periods
```

## 1.17 AmigaFlight® Help: Negate

```
NEG Negate
==============

  The destination operand is subtracted from zero, and the result is
  placed back in the destination location.

  0 - Destn -> Destn


Assembler Syntax
----------------
  NEG{.[B/W/L]} <ea>

  <ea> - data alterable


Addressing Modes
----------------
  Mode             Source  Destination

  Data Register Direct      - *
  Address Register Direct     - -
  Address Register Indirect   - *
  Postincrement Register Indirect   - *
  Predecrement Register Indirect    - *
  Register Indirect with Offset   - *
  Register Indirect with Index    - *
  Absolute Short        - *
  Absolute Long       - *
  P.C. Relative with Offset   - -
  P.C. Relative with Index    - -
  Immediate       - -


Data Size
---------
  Byte, Word, Long
```

```
Status Flags
------------
  N  Set if negative
  Z  Set if zero
  V  Set if overflow
  C  Set if borrow, else cleared
  X  Set same as carry bit


Instruction Size and Cycles to Execute
--------------------------------------
      Byte/Word Long
  <ea>    #    p    #    p
  Dn    2    4    2    6
  (An)    2   12    2   20
  (An)+   2   12    2   20
  -(An)   2   14    2   22
  d16(An)   4   16    2   24
  d8(An,Ri) 4   18    4   26
  Abs short 4   16    4   24
  Abs long  6   20    6   28

  # = no. of program bytes
  p = no. of instruction clock periods
```

## 1.18 AmigaFlight® Help: Negate with Extend

```
NEGX  Negate with Extend
==========================

  The destination  operand  and the extend bit  are  subtracted from
  zero, and the result is placed back in the destination location.

  0 - Destn - X -> Destn


Assembler Syntax
----------------
  NEGX{.[B/W/L]}  <ea>

  <ea> - data alterable


Addressing Modes
----------------
  Mode             Source  Destination

  Data Register Direct      - *
  Address Register Direct     - -
  Address Register Indirect   - *
  Postincrement Register Indirect   - *
  Predecrement Register Indirect    - *
  Register Indirect with Offset   - *
  Register Indirect with Index    - *
  Absolute Short        - *
  Absolute Long       - *
  P.C. Relative with Offset   - -
  P.C. Relative with Index    - -
  Immediate       - -


Data Size
---------
  Byte, Word, Long


Status Flags
------------
  N  Set if negative
  Z  Cleared if result non-zero, else unchanged
  V  Set if overflow
  C  Set if borrow, else cleared
  X  Set same as carry bit


Instruction Size and Cycles to Execute
--------------------------------------
      Byte/Word Long
  <ea>    #    p    #    p
  Dn    2    4    2    6
  (An)    2   12    2   20
  (An)+   2   12    2   20
  -(An)   2   14    2   22
  d16(An)   4   16    2   24
  d8(An,Ri) 4   18    4   26
  Abs short 4   16    4   24
  Abs long  6   20    6   28

  # = no. of program bytes
  p = no. of instruction clock periods
```

## 1.19 AmigaFlight® Help: Test an Operand

```
TST Test an Operand
======================

  Compare  the  specified  operand  to zero, and  set the  condition
  codes. The destination address is left unmodified.


Assembler Syntax
----------------
  TST{.[B/W/L]} <ea>

  <ea> - data alterable


Addressing Modes
----------------
  Mode             Source  Destination
  Data Register Direct      - *
  Address Register Direct     - -
  Address Register Indirect   - *
  Postincrement Register Indirect   - *
  Predecrement Register Indirect    - *
  Register Indirect with Offset   - *
  Register Indirect with Index    - *
  Absolute Short        - *
  Absolute Long       - *
  P.C. Relative with Offset   - -
  P.C. Relative with Index    - -
  Immediate       - -


Data Size
---------
  Byte, Word, Long


Status Flags
------------
  N  Set if negative
  Z  Set if zero
  V  Always cleared
  C  Always cleared
  X  Not affected


Instruction Size and Cycles to Execute
--------------------------------------
      Byte/Word Long
  <ea>    #    p    #    p
  Dn    2    4    2    4
  (An)    2    8    2   12
  (An)+   2    8    2   12
  -(An)   2   10    2   14
  d16(An)   4   12    2   16
  d8(An,Ri) 4   14    4   18
  Abs short 4   12    4   16
  Abs long  6   16    6   20

  # = no. of program bytes
  p = no. of instruction clock periods
```

## 1.20 AmigaFlight® Help: Subtract Binary

```
SUB Subtract Binary
=======================

  Subtract  the  source operand  from the  destination operand using
  binary arithmetic. Store the result in the destination operand.

  Destn - Source  -> Destn
```

```
Assembler Syntax
----------------
  SUB{.[B/W/L]} <ea>,Dn
  SUB{.[B/W/L]} Dn,<ea>
  SUB{.[B/W/L]} <ea>,Ad
  SUB{.[B/W/L]} #<data>,<ea>

  Source <ea> - all modes
  Destn <ea>  - alterable memory


Addressing Modes
----------------
  Mode             Source  Destination

  Data Register Direct      * *
  Address Register Direct     * *
  Address Register Indirect   * -
  Postincrement Register Indirect   * -
  Predecrement Register Indirect    * -
  Register Indirect with Offset   * -
  Register Indirect with Index    * -
  Absolute Short        * -
  Absolute Long       * -
  P.C. Relative with Offset   * -
  P.C. Relative with Index    * -
  Immediate       * -


Addressing Modes
----------------
  Mode             Source  Destination

  Data Register Direct      * -
  Address Register Direct     - -
  Address Register Indirect   - *
  Postincrement Register Indirect   - *
  Predecrement Register Indirect    - *
  Register Indirect with Offset   - *
  Register Indirect with Index    - *
  Absolute Short        - *
  Absolute Long       - *
  P.C. Relative with Offset   - -
  P.C. Relative with Index    - -
  Immediate       - -


Data Size
---------
  Byte, Word or Long


Status Flags
------------
  N  Set if negative
  Z  Set if zero
  V  Set if overflow
  C  Set if borrow, else cleared
  X  Set same as carry
```

Instruction Size and Cycles to Execute
--------------------------------------

| BYTE/WORD | <ea>,Dn | | Dn,<ea> | |
|---|---|---|---|---|
| <ea> | # | p | # | P |
| Dn | 2 | 4 | | |
| An | 2 | 4 | | |
| (An) | 2 | 8 | 2 | 12 |
| (An)+ | 2 | 8 | 2 | 12 |
| -(An) | 2 | 10 | 2 | 14 |
| d16(An) | 4 | 12 | 4 | 16 |
| d8(An,Ri) | 4 | 14 | 4 | 18 |
| Abs short | 4 | 12 | 4 | 16 |
| Abs long | 6 | 16 | 6 | 20 |
| d16(PC) | 4 | 12 | | |
| d8(PC,Ri) | 4 | 14 | | |
| Immediate | 4 | 8 | | |

```
  # = no. of program bytes
  p = no. of instruction clock periods
```

| LONG | <ea>,Dn | | Dn,<ea> | |
|---|---|---|---|---|
| <ea> | # | p | # | P |
| Dn | 2 | 8 | | |
| An | 2 | 8 | | |
| (An) | 2 | 14 | 2 | 20 |
| (An)+ | 2 | 14 | 2 | 20 |
| -(An) | 2 | 16 | 2 | 22 |
| d16(An) | 4 | 18 | 4 | 24 |
| d8(An,Ri) | 4 | 20 | 4 | 26 |
| Abs short | 4 | 18 | 4 | 24 |
| Abs long | 6 | 22 | 6 | 28 |
| d16(PC) | 4 | 18 | | |
| d8(PC,Ri) | 4 | 20 | | |
| Immediate | 4 | 14 | | |

```
  # = no. of program bytes
  p = no. of instruction clock periods
```

## 1.21 AmigaFlight® Help: Subtract Address

```
SUBA  Subtract Address
========================

  Subtract  the  source operand from the destination  operand  using
  binary  arithmetic. Store the  result in the  destination operand.
  This opcode is a subset of the SUB opcode, and  requires  that the
  destination be an address register.

  Destn - Source  -> Destn
```

```
Assembler Syntax
----------------
  SUBA{.[W/L]}  <ea>,An

  <ea> - all modes


Addressing Modes
----------------
  Mode             Source  Destination

  Data Register Direct      * -
  Address Register Direct     * *
  Address Register Indirect   * -
  Postincrement Register Indirect   * -
  Predecrement Register Indirect    * -
  Register Indirect with Offset   * -
  Register Indirect with Index    * -
  Absolute Short        * -
  Absolute Long       * -
  P.C. Relative with Offset   * -
  P.C. Relative with Index    * -
  Immediate       * -


Data Size
---------
  Word or Long


Status Flags
------------
  N  Not affected
  Z  Not affected
  V  Not affected
  C  Not affected
  X  Not affected


Instruction Size and Cycles to Execute
--------------------------------------
      Byte/Word Long
  <ea>    #    p    #    p
  Dn    2    8    2    8
  An (not byte) 2    8    2    8
  (An)    2   12    2   14
  (An)+   2   12    2   14
  -(An)   4   14    4   16
  d16(An)   4   16    4   18
  d8(An,Ri) 4   18    4   20
  Abs short 4   16    4   18
  Abs long  6   20    6   22
  d16(PC)   4   16    4   18
  d8(PC,Ri) 4   18    4   20
  Immediate 4   12    6   14
  # = no. of program bytes
  p = no. of instruction clock periods
```

## 1.22 AmigaFlight® Help: Subtract Immediate

```
SUBI  Subtract Immediate
==========================

  Subtract  the  source  operand from the destination  operand using
  binary arithmetic. Store  the  result in  the destination operand.
  This opcode is a subset of the SUB opcode, and requires  that  the
  source be an immediate value.

  Destn - Immediate Data -> Destn


Assembler Syntax
----------------
  SUBI{.[B/W/L]}  #<data>,<ea>

  <ea> - data alterable


Addressing Modes
----------------
  Mode             Source  Destination

  Data Register Direct      - *
  Address Register Direct     - *
  Address Register Indirect   - *
  Postincrement Register Indirect   - *
  Predecrement Register Indirect    - *
  Register Indirect with Offset   - *
  Register Indirect with Index    - *
  Absolute Short        - *
  Absolute Long       - *
  P.C. Relative with Offset   - -
  P.C. Relative with Index    - -
  Immediate       * -


Data Size
---------
  Byte, Word or Long


Status Flags
------------
  N  Set if negative
  Z  Set if zero
  V  Set if overflow
  C  Set if borrow, else cleared
  X  Set same as carry
```

```
Instruction Size and Cycles to Execute
--------------------------------------
      Byte/Word Long
  <ea>    #    p    #    p
  Dn    4    8    6   16
  (An)    4   16    6   28
  (An)+   4   16    6   28
  -(An)   4   18    6   30
  d16(An)   6   20    8   32
  d8(An,Ri) 6   22    8   34
  Abs short 6   20    8   32
  Abs long  8   24    10  36

  # = no. of program bytes
  p = no. of instruction clock periods
```

## 1.23 AmigaFlight® Help: Subtract Quick

```
SUBQ  Subtract Quick
======================

  Subtract  the  source  operand from the destination  operand using
  binary arithmetic. Store  the  result in the  destination operand.
  This opcode is a subset of the SUB  opcode, and requires that  the
  source be an immediate value in the range of 1-8.

  Destn - Immediate Data -> Destn where the immediate data may range
  from 1 to 8


Assembler Syntax
----------------
  SUBQ{.[B/W/L]}  #<data>,<ea>

  <ea> - alterable


Addressing Modes
----------------
  Mode             Source  Destination

  Data Register Direct      - *
  Address Register Direct     - *
  Address Register Indirect   - *
  Postincrement Register Indirect   - *
  Predecrement Register Indirect    - *
  Register Indirect with Offset   - *
  Register Indirect with Index    - *
  Absolute Short        - *
  Absolute Long       - *
  P.C. Relative with Offset   - -
  P.C. Relative with Index    - -
  Immediate       * -
```

```
Data Size
---------
  Byte, Word or Long


Status Flags
------------
  N  Set if negative
  Z  Set if zero
  V  Set if overflow
  C  Set if borrow generated
  X  Set same as carry

  The  Condition  Codes  are  not  affected  if  <ea>  is an address
  register


Instruction Size and Cycles to Execute
--------------------------------------
      Byte/Word Long
  <ea>    #    p    #    p
  Dn    2    4    2    8
  An (not byte) 2    4    2    8
  (An)     2   12    2   20
  (An)+    2   12    2   20
  -(An)    2   14    2   22
  d16(An)    4   16    4   24
  d8(An,Ri) 4   18    4   26
  Abs short 4   16    4   24
  Abs long  6   20    6   28

  # = no. of program bytes
  p = no. of instruction clock periods
```

### 1.24 AmigaFlight® Help: Subtract with Extend

```
SUBX  Subtract with Extend
============================

  Subtract  the   source  operand  and   the  extend  bit  from  the
  destination operand  using  binary arithmetic. Store the result in
  the destination operand.

  Destn - Source - X -> Destn


Assembler Syntax
----------------
  SUBX{.[B/W/L]}  Dy,Dx
  SUBX{.[B/W/L]}  -(Ay),-(Ax)


Addressing Modes
----------------
```

```
  Mode             Source  Destination

  Data Register Direct      - -
  Address Register Direct     - -
  Address Register Indirect   - -
  Postincrement Register Indirect   - -
  Predecrement Register Indirect    - -
  Register Indirect with Offset   - -
  Register Indirect with Index    - -
  Absolute Short        - -
  Absolute Long       - -
  P.C. Relative with Offset   - -
  P.C. Relative with Index    - -
  Immediate       - -


Data Size
---------
  Byte, Word or Long


Status Flags
------------
  N  Set if negative
  Z  Cleared if result non-zero, else unchanged
  V  Set if overflow
  C  Set if borrow, else cleared
  X  Set same as carry


Instruction Size and Cycles to Execute
--------------------------------------
      Byte/Word Long
      #     p   #    p
  Dy,Dx   2     4   2    8
  -(Ay),-(Ax) 2    18   2   30

  # = no. of program bytes
  p = no. of instruction clock periods
```

## 1.25 AmigaFlight® Help: Test and Set an Operand

```
TAS Test and Set an Operand
===============================

  Test the byte address specified in the destination, and set the N
  and Z condition codes appropriately. Set the high order bit of the
  operand. These operations are  performed  using  read-modify-write
  memory cycles and are  guaranteed  indivisible  operations.   This
  instruction  is  useful  for   synchronization   between  multiple
  processors.

  Set high order bit (bit 7) of operand
```

```
Assembler Syntax
----------------
  TAS{.B} <ea>

  <ea> - data alterable


Addressing Modes
----------------
  Mode             Source  Destination

  Data Register Direct      - *
  Address Register Direct     - -
  Address Register Indirect   - *
  Postincrement Register Indirect   - *
  Predecrement Register Indirect    - *
  Register Indirect with Offset   - *
  Register Indirect with Index    - *
  Absolute Short        - *
  Absolute Long       - *
  P.C. Relative with Offset   - -
  P.C. Relative with Index    - -
  Immediate       - -


Data Size
---------
  Byte


Status Flags
------------
  N  Set if most significant bit of operand is set
  Z  Set if zero
  V  Always cleared
  C  Always cleared
  X  Not affected


Instruction Size and Cycles to Execute
--------------------------------------
  <ea>    # p
  Dn    2 4
  (An)    2 14
  (An)+   2 14
  -(An)   2 16
  d16(An)   4 18
  d8(An,Ri) 4 20
  Abs short 4 18
  Abs long  6 22

  # = no. of program bytes
  p = no. of instruction clock periods
```