AmigaFlight Logical Instructions

Andrew Duffy Morris

COLLABORATORS

	TITLE : AmigaFlight Logical Instructions		
ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY	Andrew Duffy Morris	July 20, 2024	

REVISION HISTORY

NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

AmigaFlight Logical Instructions

1.1 AmigaFlight® Help: Logical Instructions

```
Logical Instructions
====================

Logical operation instructions AND, OR, EOR, and NOT are available  for all
sizes of integer data operands.  A similar set  of  immediate  instructions
(ANDI, ORI,  and EORI) provide these logical operations  with all sizes  of
immediate data.


  Logical AND
  -----------
  AND                Logical AND
  ANDI              AND Immediate

  Exclusive OR
  ------------
  EOR Dn,<ea>        Exclusive OR Logical
  EORI               Exclusive OR Immediate

  Logical Complement
  ------------------
  NOT <ea>           Logical Complement

  Inclusive OR
  ------------
  OR                 Inclusive OR Logical
  ORI                Inclusive OR Immediate
```

1.2 AmigaFlight® Help: Logical AND

```
AND Logical AND
===================

  Logically AND the source operand to the destination operand. Store
  the result in the destination operand.
  Destn 'and' Source -> Destn


Assembler Syntax
----------------
  AND{.[B/W/L]} <ea>,Dn
  AND{.[B/W/L]} Dn,<ea>
  AND{.[B/W/L]} #<data>,<ea>
  AND{.B}   #<data>,CCR
  AND{.W}   #<data>,SR

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


Instruction Size and Cycles to Execute
--------------------------------------
  BYTE/WORD <ea>,Dn   Dn,<ea>
  <ea>    #     p   #    P
  Dn    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    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: AND Immediate

```
ANDI  AND Immediate
=====================

  Logically AND the source operand to the destination operand. Store
  the result in the destination operand. This  opcode is a subset of
  the AND opcode,  and   requires  that  the source  be an immediate
  value.

  Destn 'and' Immediate Data -> Destn


Assembler Syntax
----------------
  ANDI{.[B/W/L]}  #<data>,<ea>
  ANDI{.B}  #<data>,CCR
  ANDI{.W}  #<data>,SR    (Privileged Instruction)

  <ea> - data alterable or SR


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

  unless <ea> = SR when  the condition codes  are affected according
  to the operation


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
  SR    4   20

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

1.4 AmigaFlight® Help: Exclusive OR Logical

```
EOR Exclusive OR Logical
============================

  Exclusive OR the source  operand to the destination operand. Store
  the result in the destination operand.

  Destn 'eor' Source -> Destn


Assembler Syntax
----------------
  EOR{.[B/W/L]} Dn,<ea>
  EOR{.[B/W/L]} #<data>,<ea>
  EOR{.B}   #<data>,CCR
  EOR{.W}   #<data>,SR

  <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 most significant bit of is set, else cleared
  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   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
```

1.5 AmigaFlight® Help: Exlusive OR Immediate

```
EORI  Exlusive OR Immediate
=============================

  Exclusive OR the source  operand to the destination operand. Store
  the result in the destination operand. This  opcode is a subset of
  the EOR  opcode, and  requires  that  the  source  be an immediate
  value.

  Destn 'eor' Immediate Data -> Destn


Assembler Syntax
----------------
  EORI{.[B/W/L]}  #<data>,<ea>
  EORI{.B}  #<data>,CCR
  EORI{.W}  #<data>,SR    (Privileged Instruction)

  <ea> - data alterable or SR

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

  unless <ea> = SR when the condition codes are are affected
  according to the operation


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
  SR    4   20

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

1.6 AmigaFlight® Help: Logical Complement

```
NOT Logical Complement
==========================

  The ones complement of the destination  operand is  calculated and
  placed back in the destination location.

  Ones complement Destn -> Destn


Assembler Syntax
----------------
  NOT{.[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    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
```

1.7 AmigaFlight® Help: Inclusive OR Logical

```
OR  Inclusive OR Logical
============================

  Inclusive OR the source  operand to the destination operand. Store
  the result in the destination operand.

  Destn 'or' Source -> Destn
```

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

  Source <ea> - data
  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 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
--------------------------------------
  BYTE/WORD <ea>,Dn   Dn,<ea>
  <ea>     #     p   #    P
  Dn    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    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
```

1.8 AmigaFlight® Help: Inclusive OR Immediate

```
ORI Inclusive OR Immediate
==============================

  Inclusive OR the source  operand to the destination operand. Store
  the  result in the destination operand. This opcode is a subset of
  the OR opcode and requires that the source be an immediate value.

  Destn 'or' Immediate Data -> Destn


Assembler Syntax
```

```
----------------
  ORI{.[B/W/L]} #<data>,<ea>
  ORI{.B}   #<data>,CCR
  ORI{.W}   #<data>,SR    (Privileged Instruction)

  <ea> - data alterable or SR


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 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
--------------------------------------
      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
  SR    4   20

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