AmigaFlight Shift and Rotate Instructions

Andrew Duffy Morris

COLLABORATORS			
	TITLE : AmigaFlight Shift and Rotate Instructions		
ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY	Andrew Duffy Morris	July 1, 2022	

REVISION HISTORY			
NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

AmigaFlight Shift and Rotate Instructions

1.1 AmigaFlight® Help: Shift and Rotate Instructions

```
                Shift and Rotate Instructions
==============================
```

Shift operations in both directions are provided by the arithmetic instructions ASR and ASL and logical shift instructions LSR and LSL. The rotate instructions (with and without extend) available are ROXR, ROXL, ROR, and ROL. All shift and rotate operations can be performed in either registers or memory. Register shifts and rotates support all operand sizes and allow a shift count specified in the instruction of one or eight bits, or 0 to 63 bits specified in a data register. Memory shifts and rotates are for word operands only and allow single-bit shifts or rotates.

```
  Arithmetic Shifts
  -----------------

                ASL
                  Arithmetic Shift Left

                ASR
                  Arithmetic Shift Right

  Logical Shifts
  --------------

                LSL
                  Logical Shift Left

                LSR
                  Logical Shift Right

  Rotates
  -------

                ROL
                  Rotate Left (without Extend)

                ROR
```

```
                  Rotate Right (without Extend)

  Extended Rotates
  ----------------

                ROXL
                  Rotate Left with Extend

                ROXR
                  Rotate Right with Extend
```

1.2 AmigaFlight® Help: Arithmetic Shift Left

```
ASL Arithmetic Shift Left
=============================

  Arithmetically shift the destination operand left N bits. The
  explicit or implied source operand determines N, the number of
  bits to be shifted. An arithmetic shift with an implied shift
  count, shifts memory destination location one bit only to the
  left.

  Destn shifted by <count> -> Destn
  C <- OPERAND <- 0
  X <-


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

  Dx contains shift count
  Immediate shift data may be 1 - 8
  Memory can be shifted only 1 bit


Addressing Modes (for ASL memory)
---------------------------------
  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 except if Memory Shift when only Word Allowed


Status Flags
------------
  N  Set if most significant bit of result is set, else cleared
  Z  Set if result is zero
  V  Set if  the  most significant bit is changed at any time during
     shift operation, else clear
  C  Set according to last bit shifted out of operand, cleared for a
     shift count of zero
  X  Set according to last bit shifted out of operand, unaffected by
     a shift count of zero


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

  # = no. of instruction bytes
  p = no. of instruction clock periods
  n = shift count
```

1.3 AmigaFlight® Help: Arithmetic Shift Right

```
ASR Arithmetic Shift Right
==========================

  Arithmetically shift the destination  operand  right N  bits.  The
  explicit  or  implied source  operand  determines N, the number of
  bits to be shifted. An  arithmetic  shift  with  an implied  shift
  count, shifts  memory  destination location  one bit  only to  the
  right.


  Destn shifted by <count> -> Destn
  sign bit -> OPERAND -> C
                      -> X


Assembler Syntax
----------------
```

```
  ASR{.[B/W/L]} Dx,Dy
  ASR{.[B/W/L]} #<data>,Dy
  ASR{.W}   <ea>

  Dx contains shift count
  Immediate shift data may be 1 - 8
  Memory can be shifted only 1 bit


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 except if Memory Shift when only Word Allowed


Status Flags
------------
  N  Set if most significant bit of result is set, else cleared
  Z  Set if result is zero
  V  Set if the most significant  bit  is changed at any time during
     shift operation, else clear
  C  Set according to last bit shifted out of operand, cleared for a
     shift count of zero
  X  Set according to last bit shifted out of operand, unaffected by
     a shift count of zero


Instruction Size and Cycles to Execute
--------------------------------------
  Size...   Byte/Word Long
      #    p    #   p
  Dx,Dy   2   6+2n  2  8+2n
  #<data>,Dy  2   6+2n  2  8+2n
  (An)    2   12
  (An)+   2   12
  -(An)   2   14
  d16(An)   4   16
  d8(An,Ri) 4   18
  Abs short 4   16
  Abs long  6   20
```

```
  # = no. of instruction bytes
  p = no. of instruction clock periods
  n = shift count
```

1.4 AmigaFlight® Help: Logical Shift Left

```
LSL Logical Shift Left
==========================

  Logically  shift the destination operand left N bits. The explicit
  or implied source operand determines N, the number of bits  to  be
  shifted. A logical shift with  an implied  shift count, shifts the
  specified memory destination location one bit only.

  Destn shifted by <count> -> Destn

  C <- OPERAND <- 0
  X <-


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

  <ea> - memory alterable only

  Dx contains shift count
  Immediate shift data may be 1 - 8
  Memory can be shifted only 1 bit


Addressing Modes (for LSL memory)
---------------------------------
  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 except if Memory Shift when only Word
```

```
Status Flags
------------
  N  Set if most significant bit of result is set, else cleared
  Z  Set if zero
  V  Set if the most significant  bit is changed at any time  during
     shift operation, else clear
  C  Set according to last bit shifted out of operand, cleared for a
     shift count of zero
  X  Set according to last bit shifted out of operand, unaffected by
     a shift count of zero


Instruction Size and Cycles to Execute
--------------------------------------
  Size    Byte/Word Long

      # p # p
  Dx,Dy   2 6+2n  2 8+2n
  #<data>,Dy  2 6+2n  2 8+2n
  (An)    2 12
  (An)+   2 12
  -(An)   2 14
  d16(An)   4 16
  d8(An,Ri) 4 18
  Abs short 4 16
  Abs long  6 20

  # = no. of instruction bytes
  p = no. of instruction clock periods
  n = shift count
```

1.5 AmigaFlight® Help: Logical Shift Right

```
LSR Logical Shift Right
=======================

  Logically shift the destination operand right N bits. The explicit
  or implied source operand determines N, the number of bits  to  be
  shifted. A logical shift  with an implied  shift count, shifts the
  specified memory destination location one bit only.

  Destn shifted by <count> -> Destn

  0 -> OPERAND -> C
               -> X


Assembler Syntax
----------------
  LSR{.[B/W/L]} Dx,Dy
  LSR{.[B/W/L]} #<data>,Dy
  LSR{.W}   <ea>
```

```
  <ea> - memory alterable only

  Dx contains shift count
  Immediate shift data may be 1 - 8
  Memory can be shifted only 1 bit


Addressing Modes (for LSR memory)
---------------------------------
  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 except if Memory Shift when only Word


Status Flags
------------
  N  Set if negative
  Z  Set if zero
  V  Set if the most significant bit is changed  at any  time during
     shift operation, else clear
  C  Set according to last bit shifted out of operand, cleared for a
     shift count of zero
  X  Set according to last bit shifted out of operand, unaffected by
     a shift count of zero


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

  # = no. of instruction bytes
```

```
  p = no. of instruction clock periods
  n = shift count
```

1.6 AmigaFlight® Help: Rotate Left (without Extend)

```
ROL Rotate Left (without Extend)
================================

  Rotate the destination operand left N bits. The explicit or
  implied source operand determines N, the number of bits to be
  rotated. A rotate with an implied shift count rotates the
  specified memory destination location one bit only.

  Destn rotated by <count> -> Destn

  C <- OPERAND <- high order bit


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

  <ea> - memory alterable only

  Dx contains shift count
  Immediate shift data may be 1 - 8
  Memory can be shifted only 1 bit


Addressing Modes (ROL memory)
-----------------------------
  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 except if Memory Shift when only Word


Status Flags
```

```
------------
  N  Set if most significant bit of result is set, else cleared
  Z  Set if zero
  V  Always cleared
  C  Set according to last bit shifted out of operand, cleared for a
     shift count of zero
  X  Not affected


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

  # = no. of instruction bytes
  p = no. of instruction clock periods
  n = shift count
```

1.7 AmigaFlight® Help: Rotate Right (without Extend)

```
ROR Rotate Right (without Extend)
=================================

  Rotate the destination operand right N bits. The explicit or
  implied source operand determines N, the number of bits to be
  rotated. A rotate with an implied shift count rotates the
  specified memory destination location one bit only.

  Destn rotated by <count> -> Destn

  high order bit -> OPERAND -> C


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

  <ea> - memory alterable only

  Dx contains shift count
  Immediate shift data may be 1 - 8
  Memory can be shifted only 1 bit
```

```
Addressing Modes (ROR memory)
-----------------------------
  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 except if Memory Shift when only Word


Status Flags
------------
  N  Set if most significant bit of result is set, else cleared
  Z  Set if zero
  V  Always cleared
  C  Set according to last bit shifted out of operand, cleared for a
     shift count of zero
  X  Not affected


Instruction Size and Cycles to Execute
--------------------------------------
  Size...   Byte/Wora Long
      #    p    #   p
  Dx,Dy   2   6+2n  2  8+2n
  #<data>,Dy  2   6+2n  2  8+2n
  (An)    2   12
  (An)+   2   12
  -(An)   2   14
  d16(An)   4   16
  d8(An,Ri) 4   18
  Abs short 4   16
  Abs long  6   20

  # = no. of instruction bytes
  p = no. of instruction clock periods
  n = shift count
```

1.8 AmigaFlight® Help: Rotate Left with Extend

```
ROXL  Rotate Left with Extend
=============================
```

Rotate the destination operand left N bits. The extend bit is included as part of the rotation. The explicit or implied source operand determines N, the number of bits to be rotated. A rotate with an implied shift count rotates the specified memory destination location one bit only.

Destn rotated by <count> -> Destn

C <- OPERAND <- high order bit
X <-

Assembler Syntax

```
ROXL{.[B/W/L]}  Dx,Dy
ROXL{.[B/W/L]}  #<data>,Dy
ROXL{.[B/W/L]}  <ea>
```

<ea> - memory alterable only

Dx contains shift count
Immediate shift data may be 1 - 8
Memory can be shifted only 1 bit

Addressing Modes (ROXL memory)

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 except if Memory Shift when only Word

Status Flags

N Set if most significant bit of result is set, else cleared
Z Set if zero
V Always cleared
C Set according to last bit shnfted out of operand, cleared for a shift count of zero
X Set according to last bit shifted out of operand, unaffected by

```
     a shift count of zero


Instruction Size and Cycles to Execute
--------------------------------------
  Size...   Byte/Word Long
      #    p    #   p
  Dx,Dy   2   6+2n  2  8+2n
  #<data>,Dy  2   6+2n  2  8+2n
  (An)    2   12
  (An)+   0   12
  -(An)   2   14
  d16(An)   4   16
  d8(An,Ri) 4   18
  Abs short 4   16
  Abs long  6   20

  # = no. of instruction bytes
  p = no. of instruction clock periods
  n = shift count
```

1.9 AmigaFlight® Help: Rotate Right with Extend

```
ROXR  Rotate Right with Extend
==============================

  Rotate the destination operand right  N  bits. The  extend  bit is
  included as part of the ritation. The explicit or  implied  source
  operand determines N, the  number of bits  to be rotated. A rotate
  with   an   implied   shift   iount   rotates   the   specified   memory
  destination lmcation one bit only.

  Destn rotated by <count> -> Destn

  C <- OPERAND <- high order bit
  X <-


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

  <ea> - memory alterable only

  Dx contains shift count
  Immediate shift data may be 1 - 8
  Memory can be shifted only 1 bit


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

```
  Data Register Direct      - -
  Address Register Direct     - -
  Address Register Indirect   - *
  Postincrement Register Indirect   - *
  Predecrement Register Indirect    - *
  Rcgister 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 except if Memory Shift when only Word

Status Flags

- N Set if most significant bit of result is set, else cleared
- Z Set if zero
- V Always cleared
- C Set according to last bit shifted out of operand, cleared for a shift count of zero
- X Set according to last bit shifted out of operand, unaffected by a shift count of zero

Instruction Size and Cycles to Execute

Size...	Byte/Word		Long	
	#	p	#	p
Dx,Dy	2	6+2n	2	8+2n
#<data>,Dy	2	6+2n	2	8+2n
(An)	2	12		
(An)+	2	12		
-(An)	2	14		
d16(An)	4	16		
d8(An,Ri)	4	18		
Abs short	4	16		
Abs long	6	20		

# = no. of instruction bytes
p = no. of instruction clock periods
n = shift count