AmigaFlight Flow Control Instructions

Andrew Duffy Morris

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

REVISION HISTORY			
NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

AmigaFlight Flow Control Instructions

1.1 AmigaFlight® Help: Flow Control Instructions

```
Flow Control Instructions
=========================

Flow Control operations are accomplished using a series of conditional and
unconditional branch  instructions  and  return instructions, included  in
these instructions are the conditional setting instructions.


  Unconditional jump and branch instructions
  ------------------------------------------
  BRA <label>       Branch Always
  JMP <ea>          Jump

  Conditional branch instructions
  -------------------------------
  BCC <label>       Branch if Carry Clear
  BCS <label>       Branch if Carry Set
  BEQ <label>       Branch if Equal
  BGE <label>       Branch if Greater or Equal
  BGT <label>       Branch if Greater
  BHI <label>       Branch if High
  BLE <label>       Branch if Less or Equal
  BLS <label>       Branch if Low or Same
  BLT <label>       Branch if Less
  BMI <label>       Branch if Minus
  BNE <label>       Branch if Not Equal
  BPL <label>       Branch if Plus
  BVS <label>       Branch if Overflow
  BVC <label>       Branch if No Overflow

  Test condition, decrement and branch instructions
  -------------------------------------------------
  DBT <label>       No operation (condition always true)
  DBF <label>       Decr. and Branch Always unless Count = -1
  DBHI <label>      Decr. and Branch until High or Count = -1
  DBLS <label>      Decr. and Branch until Low or Same or Count = -1
  DBCC <label>      Decr. and Branch until Carry Clear or Count = -1
  DBCS <label>      Decr. and Branch until Carry Set or Count = -1
```

```
  DBNE <label>      Decr. and Branch until Not Equal or Count = -1
  DBEQ <label>      Decr. and Branch until Equal or Count = -1
  DBVC <label>      Decr. and Branch until No Overflow or Count = -1
  DBVS <label>      Decr. and Branch until Overflow or Count = -1
  DBPL <label>      Decr. and Branch until Plus or Count = -1
  DBMI <label>      Decr. and Branch until Minus or Count = -1
  DBGE <label>      Decr. and Branch until Greater or Equal or Count = -1
  DBLT <label>      Decr. and Branch until Less or Count = -1
  DBGT <label>      Decr. and Branch until Greater or Count = -1
  DBLE <label>      Decr. and Branch until Less or Equal or Count = -1
  DBRA <label>      Decr. and Branch Always unless Count = -1

  Conditional setting instructions
  --------------------------------
  SCC <ea>          Set if Carry Clear
  SCS <ea>          Set if Carry Set
  SEQ <ea>          Set if Equal
  SF <ea>           Set Never
  SGE <ea>          Set if Greater or Equal
  SGT <ea>          Set if Greater
  SHI <ea>          Set if High
  SLE <ea>          Set if Less or Equal
  SLS <ea>          Set if Lower or Same
  SLT <ea>          Set if Less
  SMI <ea>          Set if Minus
  SNE <ea>          Set if Not Equal
  SPL <ea>          Set if Plus
  ST <ea>           Set Always
  SVC <ea>          Set if No Overflow
  SVS <ea>          Set if Overflow

  Subroutine call instructions
  ----------------------------
  BSR <label>       Branch to Subroutine
  JSR <ea>          Jump to Subroutine

  Return instructions
  -------------------
  RTE               Return from Exception (Privileged)
  RTR               Return and Restore Condition Codes
  RTS               Return from Subroutine
```

1.2 AmigaFlight® Help: Branch if Carry Clear

```
BCC Branch if Carry Clear
=============================

  Continue program execution at  the specified label,  if the 'Carry
  Clear' condition is met. The .S version of this instruction forces
  an  8-bit  displacement  to  be  generated.  This  means  that the
  relative offset of  the  label must be in the range of -128 to 127
  bytes in distance from the current program counter. The .L version
  of this instruction forces an 16-bit displacement to be generated.
  This means  that  the relative offset of  the label must be in the
  range  of -32768 to  32767  bytes  in distance  from  the  current
```

program counter. The current program counter is defined to be the current instruction location plus two. If the BCC instruction is used, the assembler automatically decides which of the two displacements is most appropriate, and generates that instruction. This is sometimes known as automatic branch shortening.

Branch if C = 0

Assembler Syntax

BCC{.[S/L]} <label>

Data Size

Byte, Word

Sataus Flags

N Not affected
Z Not affected
V Not affected
C Not affected
X Not affected

Instruction Size and Cycles to Execute

Size...	Byte		Word	
	#	p	#	p
Branch Taken	2	10	4	10
Branch Not Taken	2	8	4	12

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

1.3 AmigaFlight® Help: Branch if Carry Set

BCS Branch if Carry Set
==========

Continue program execution at the specified label, if the 'Carry Set' condition is met. The .S version of this instruction forces an 8-bit displacement to be generated. This means that the relative offset of the label must be in the range of -128 to 127 bytes in distance from the current program counter. The .L version of this instruction forces the an 16-bit displacement to be generated. This means that the relative offset of the label must be in the range of -32768 to 32767 bytes in distance from the current program counter. The current program counter is defined to be the current instruction location plus two. If the BCS instruction is used, the assembler automatically decides which of

```
  the two displacements is most appropriate, and generates that
  instruction. This is sometimes known as automatic branch
  shortening.

  Branch if C = 1


Assembler Syntax
----------------
  BCS{.[S/L]} <label>


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


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



Instruction Size and Cycles to Execute
--------------------------------------
  Size...      Byte   Word
         #   p  #   p
  Branch Taken     2  10  4  10
  Branch Not Taken   2   8  4  12

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

1.4 AmigaFlight® Help: Branch if Greater or Equal

```
BGE Branch if Greater or Equal
==============================

  Continue program execution at the specified label, if the 'Greater
  or Equal' condition is met. The .S version of this instruction
  forces an 8-bit displacement to be generated. This means that the
  relative offset of the label must be in the range of -128 to 127
  bytes in distance from the current program counter. The .L version
  of this instruction forces the an 16-bit displacement to be
  generated. This means that the relative offset of the label must
  be in the range of -32768 to 32767 bytes in distance from the
  current program counter. The current program counter is defined to
  be the current instruction location plus two. If the BGE
  instruction is used, the assembler automatically decides which of
  the two displacements is most appropriate, and generates that
  instruction. This is sometimes known as automatic branch
```

```
  shortening.

  Branch if N.V+N'.V' = 1

  where . = Boolean AND
        + = Boolean OR
        ' = Complement


Assembler Syntax
----------------
  BGE{.[S/L]} <label>


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


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



Instruction Size and Cycles to Execute
--------------------------------------
  Size...      Byte   Word
         #   p  #   p
  Branch Taken     2  10  4  10
  Branch Not Taken   2   8  4  12

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

1.5 AmigaFlight® Help: Branch if Greater

```
BGT Branch if Greater
=========================

  Continue program execution at the specified label, if the 'Greater
  than' condition  is met. The .S version of this instruction forces
  an  8-bit  displacement  to  be  generated.  This  means  that the
  relative offset of the label must be in  the range  of -128 to 127
  bytes in distance from the current program counter. The .L version
  of this instruction forces an 16-bit displacement to be generated.
  This means  that  the relative offset of  the label must be in the
  range  of  -32768  to  32767  bytes in distance  from  the current
  program counter. The current program counter is defined to be  the
  current  instruction location  plus two. If the BGT instruction is
  used,  the  assembler  automatically  decides  which  of  the  two
```

```
  displacements is most appropriate, and generates that instruction.
  This is sometimes known as automatic branch shortening.

  Branch if N.V.Z'+N'.V'.Z' = 1

  where . = Boolean AND
        + = Boolean OR
        ' = Complement


Assembler Syntax
----------------
  BGT{.[S/L]} <label>


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


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


Instruction Size and Cycles to Execute
--------------------------------------
  Size...      Byte   Word
         #   p  #   p
  Branch Taken     2  10  4  10
  Branch Not Taken   2   8  4  12

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

1.6 AmigaFlight® Help: Branch if High

```
BHI Branch if High
==================

  Continue program execution at the  specified label,  if the 'High'
  condition  is  met. The .S version of  this  instruction forces an
  8-bit displacement to be generated. This means  that  the relative
  offset of the label must be  in the range  of -128 to 127 bytes in
  distance  from the current program counter. The .L version of this
  instruction  forces an  16-bit  displacement to be generated. This
  means that the  relative  offset of the label must be in the range
  of -32768 to 32767  bytes in  distance  from  the  current program
  counter. The current  program counter is defined to be the current
  instruction location plus two. If the BHI instruction is used, the
  assembler automatically decides which of the  two displacements is
```

most appropriate, and generates that instruction. This is sometimes known as automatic branch shortening.

Branch if C'.Z' = 1

where . = Boolean AND
' = Complement

Assembler Syntax

BHI{.[S/L]} <label>

Data Size

Byte, Word

Status Flags

N Not affected
Z Not affected
V Not affected
C Not affected
X Not affected

Instruction Size and Cycles to Execute

Size...	Byte		Word	
	#	p	#	p
Branch Taken	2	10	4	10
Branch Not Taken	2	8	4	12

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

1.7 AmigaFlight® Help: Branch if Less or Equal

BLE Branch if Less or Equal
==========

Continue program execution at the specified label, if the 'Less or Equal' condition is met. The .S version of this instruction forces an 8-bit displacement to be generated. This means that the relative offset of the label must be in the range of -128 to 127 bytes in distance from the current program counter. The .L version of this instruction forces an 16-bit displacement to be generated. This means that the relative offset of the label must be in the range of -32768 to 32767 bytes in distance from the current program counter. The current program counter is defined to be the current instruction location plus two. If the BLE instruction is used, the assembler automatically decides which of the two displacements is most appropriate, and generates that

```
  instruction.   This   is   sometimes   known   as   automatic   branch
  shortening.

  Branch if Z+N.V'+N'.V = 1

  where . = Boolean AND
        + = Boolean OR
        ' = Complement


Assembler Syntax
----------------
  BLE{.[S/L]} <label>


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


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


Instruction Size and Cycles to Execute
--------------------------------------
  Size...      Byte   Word
         #   p  #   p
  Branch Taken     2  10  4  10
  Branch Not Taken   2   8  4  12

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

1.8 AmigaFlight® Help: Branch if Low or Same

```
BLS Branch if Low or Same
=============================

  Continue program execution at  the specified label, if the 'Low or
  Same'  condition is met. The .S version of this instruction forces
  an  8-bit  displacement  to  be  generated.  This  means  that the
  relative offset of the label must  be in the range  of -128 to 127
  bytes in distance from the current program counter. The .L version
  of this instruction forces an 16-bit displacement to be generated.
  This means  that  the relative offset of the  label must be in the
  range  of  -32768  to  32767 bytes  in  distance  from the current
  program counter. The current program counter is defined to be  the
  current  instruction location plus two. If the BLS  instruction is
  used,  the  assembler  automatically  decides  which  of  the  two
```

```
  displacements is most appropriate, and generates that instruction.
  This is sometimes known as automatic branch shortening.

  Branch if C + Z = 1

  where + = Boolean OR


Assembler Syntax
----------------
  BLS{.[S/L]} <label>


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


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


Instruction Size and Cycles to Execute
--------------------------------------
  Size...      Byte   Word
         #   p  #   p
  Branch Taken     2  10  4  10
  Branch Not Taken   2   8  4  12

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

1.9 AmigaFlight® Help: Branch if Less

```
BLT Branch if Less
======================

  Continue program execution at  the specified label,  if the 'Less'
  condition  is  met. The .S  version of  this instruction forces an
  8-bit  displacement to be generated. This  means that the relative
  offset of  the label  must be in the range of -128 to 127 bytes in
  distance from the  current program counter. The .L version of this
  instruction  forces  an 16-bit displacement to be  generated. This
  means  that the  relative offset of the label must be in the range
  of  -32768 to 32767  bytes in  distance  from the  current program
  counter. The current program counter  is defined to be the current
  instruction location plus two. If the BLT instruction is used, the
  assembler automatically decides which  of the two displacements is
  most   appropriate,   and   generates   that   instruction.   This   is
  sometimes known as automatic branch shortening.
```

```
  Branch if N.V'+N'.V = 1

  where . = Boolean AND
        + = Boolean OR
        ' = Complement


Assembler Syntax
----------------
  BLT{.[S/L]} <label>


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


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


Instruction Size and Cycles to Execute
--------------------------------------
  Size...      Byte   Word
         #   p  #   p
  Branch Taken     2  10  4  10
  Branch Not Taken   2   8  4  12

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

1.10 AmigaFlight® Help: Branch if Minus

```
BMI Branch if Minus
=======================

  Continue program execution at  the specified label, if the 'Minus'
  condition  is  met. The .S  version of  this instruction forces an
  8-bit  displacement to be generated. This means  that the relative
  offset of the label must be  in the range  of -128 to 127 bytes in
  distance from the current program counter. The .L version  of this
  instruction  forces  an 16-bit  displacement to be generated. This
  means  that the relative offset of the  label must be in the range
  of -32768 to 32767  bytes in  distance  from the  current program
  counter. The current program counter is defined to be the  current
  instruction location plus two. If the BMI instruction is used, the
  assembler automatically decides which  of  the  two  displacements
  is  most  appropriate,  and  generates that  instruction. This  is
  sometimes known as automatic branch shortening.
```

```
  Branch if N = 1


Assembler Syntax
----------------
  BMI{.[S/L]} <label>


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


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


Instruction Size and Cycles to Execute
--------------------------------------
  Size...      Byte   Word
         #   p  #   p
  Branch Taken     2  10  4  10
  Branch Not Taken   2   8  4  12

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

1.11 AmigaFlight® Help: Branch if Not Equal

```
BNE Branch if Not Equal
=======================

  Continue  program  execution  at the  specified label, if the 'Not
  Equal' condition  is  met.  The .S  version  of  this  instruction
  forces an  8-bit displacement to be generated. This means that the
  relative offset of the  label must be in the range  of -128 to 127
  bytes in distance from the current program counter. The .L version
  of this instruction forces an 16-bit displacement to be generated.
  This  means  that the  relative offset of the label must be in the
  range  of  -32768  to  32767  bytes  in  distance from the current
  program counter. The current program counter  is defined to be the
  current  instruction location  plus two. If the BNE instruction is
  used,  the  assembler  automatically  decides  which  of  the  two
  displacements is most appropriate, and generates that instruction.
  This is sometimes known as automatic branch shortening.

  Branch if Z = 0
```

```
Assembler Syntax
----------------
  BNE{.[S/L]} <label>


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


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


Instruction Size and Cycles to Execute
--------------------------------------
  Size...      Byte   Word
         #   p  #   p
  Branch Taken     2  10  4  10
  Branch Not Taken   2   8  4  12

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

1.12 AmigaFlight® Help: Branch if Equal

```
BEQ Branch if Equal
=======================

  Continue program execution at  the specified label, if the 'Equal'
  condition  is  met. The .S  version of this instruction  forces an
  8-bit displacement to be generated. This  means  that the relative
  offset of the label must be  in the range  of -128 to 127 bytes in
  distance from the  current program counter. The .L version of this
  instruction  forces  an 16-bit  displacement to be generated. This
  means that the  relative offset of  the label must be in the range
  of -32768 to 32767 bytes in   distance  from the  current  program
  counter. The current program counter is defined to be the  current
  instruction location plus two. If the BEQ instruction is used, the
  assembler automatically decides  which  of  the  two displacements
  is  most  appropriate,  and  generates that  instruction. This  is
  sometimes known as automatic branch shortening.

  Branch if Z = 1


Assembler Syntax
----------------
  BEQ{.[S/L]} <label>
```

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


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


Instruction Size and Cycles to Execute
--------------------------------------
  Size...      Byte   Word
         #   p  #   p
  Branch Taken     2  10  4  10
  Branch Not Taken   2   8  4  12

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

1.13 AmigaFlight® Help: Branch if Plus

```
BPL Branch if Plus
======================

  Continue program execution at  the  specified label, if the 'Plus'
  condition  is met. The .S  version of  this  instruction forces an
  8-bit  displacement to be generated. This means that  the relative
  offset of the label must be in the range of -128  to 127 bytes  in
  distance from the current program counter. The .L version  of this
  instruction  forces  an 16-bit  displacement to be generated. This
  means that the  relative offset of the label must be  in the range
  of -32768 to 32767 bytes  in  distance  from  the  current program
  counter. The current program counter is defined to be the  current
  instruction location plus two. If the BPL instruction is used, the
  assembler automatically decides which  of  the  two  displacements
  is  most  appropriate,  and  generates  that  instruction. This is
  sometimes known as automatic branch shortening.

  Branch if N = 0


Assembler Syntax
----------------
  BPL{.[S/L]} <label>


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

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


Instruction Size and Cycles to Execute
--------------------------------------
  Size...      Byte   Word
         #   p  #   p
  Branch Taken     2  10  4  10
  Branch Not Taken   2   8  4  12

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

1.14 AmigaFlight® Help: Branch if Overflow

```
BVS Branch if Overflow
==========================

  Continue program execution at the specified label, if the
  'Overflow' condition is met. The .S version of this instruction
  forces an 8-bit displacement to be generated. This means that the
  relative offset of the label must be in the range of -128 to 127
  bytes in distance from the current program counter. The .L version
  of this instruction forces an 16-bit displacement to be generated.
  This means that the relative offset of the label must be in the
  range of -32768 to 32767 bytes in distance from the current
  program counter. The current program counter is defined to be the
  current instruction location plus two. If the BVS instruction is
  used, the assembler automatically decides which of the two
  displacements is most appropriate, and generates that instruction.
  This is sometimes known as automatic branch shortening.

  Branch if V = 1


Assembler Syntax
----------------
  BVS{.[S/L]} <label>


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


Status Flags
------------
```

```
  N  Not affected
  Z  Not affected
  V  Not affected
  C  Not affected
  X  Not affected


Instruction Size and Cycles to Execute
--------------------------------------
  Size...      Byte   Word
         #   p  #   p
  Branch Taken     2  10  4  10
  Branch Not Taken   2   8  4  12

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

1.15 AmigaFlight® Help: Branch if No Overflow

```
BVC Branch if No Overflow
=============================

  Continue program execution at  the  specified  label, if  the  'No
  Overflow' condition  is  met. The .S  version of  this instruction
  forces an 8-bit displacement to be generated. This means  that the
  relative offset of the label must be  in the range  of -128 to 127
  bytes in distance from the current program counter. The .L version
  of this instruction forces an 16-bit displacement to be generated.
  This means  that  the relative offset of  the label must be in the
  range  of -32768 to 32767  bytes  in  distance  from  the  current
  program counter. The current program counter is defined to be  the
  current instruction location plus two. If  the BVC  instruction is
  used,  the  assembler  automatically  decides  which  of  the  two
  displacements is most appropriate, and generates that instruction.
  This is sometimes known as automatic branch shortening.

  Branch if V = 0


Assembler Syntax
----------------
  BVC{.[S/L]} <label>


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


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

```
  X  Not affected


Instruction Size and Cycles to Execute
--------------------------------------
  Size...      Byte   Word
         #   p  #   p
  Branch Taken     2  10  4  10
  Branch Not Taken   2   8  4  12

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

1.16 AmigaFlight® Help: Branch Always

```
BRA Branch Always
=================

  Continue program execution at the specified label. The .S  version
  of this instruction  forces an 8-bit displacement to be generated.
  This means  that  the relative offset of  the label must be in the
  range of -128 to 127 bytes  in distance from the  current  program
  counter.  The .L  version  of this  instruction  forces  an 16-bit
  displacement to be generated. This means that  the relative offset
  of  the  label  must  be in the range  of -32768 to 32767 bytes in
  distance  from  the current  program  counter. The current program
  counter is  defined to  be the  current  instruction location plus
  two. If the BRA instruction is used, the assembler  automatically
  decides which  of  the two displacements is most appropriate,  and
  generates  that instruction. This  is sometimes known as automatic
  branch shortening.


Assembler Syntax
----------------
  BRA{.[S/L]} <label>


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


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


Instruction Size and Cycles to Execute
--------------------------------------
  Size...   Byte    Word
```

```
      #   p   #   p
      2  10     4  10

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

1.17 AmigaFlight® Help: Branch to Subroutine

```
BSR Branch to Subroutine
============================

  The long word address of the instruction immediately following
  this instruction is pushed on the stack, and program execution
  continues at the specified label. The .S version of this
  instruction forces an 8-bit displacement to be generated. This
  means that the relative offset of the label must be in the range
  of -128 to 127 bytes in distance from the current program counter.
  The .L version of this instruction forces an 16-bit displacement
  to be generated. This means that the relative offset of the label
  must be in the range of -32768 to 32767 bytes in distance from the
  current program counter. The current program counter is defined to
  be the current instruction location plus two. If the BSR
  instruction is used, the assembler automatically decides which of
  the two displacements is most appropriate, and generates that
  instruction. This is sometimes known as automatic branch
  shortening.


Assembler Syntax
----------------
  BSR{.[S/L]} <label>


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


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


Instruction Size and Cycles to Execute
--------------------------------------
  Size    Byte    Word
      #   p   #   p
      2  18   4  18

  # = no. of instruction bytes
```

```
  p = no. of instruction clock periods
```

1.18 AmigaFlight® Help: No operation (condition always true)

```
DBT No operation (condition always true)
==========================================

  If  the  specified condition is false,  decrement  the destination
  data register, and then compare the destination  register with -1.
  If the data  register doesn't equal -1, continue processing at the
  specified label. If either of the  conditions fail, then  continue
  instruction execution with  the next instruction. This instruction
  uses a 16-bit displacement as a  label offset. This means that the
  relative offset of the label  must be  in  the  range of -32768 to
  32767 bytes in distance from the current program counter.
  This instruction provides a primitive looping construct similar to
  the REPEAT UNTIL looping  construct of Pascal/ADA/Basic/C etc. The
  DBcc  instruction may be thought  of as a REPEAT loop UNTIL either
  the  condition  becomes  true,  or the loop  counter goes below 0.
  This, of  course, is  assuming that  the destination data register
  was  initially set to a positive value. (This instruction uses the
  bottom  16  bits  of  the  destination  data  register  for a loop
  counter, 0 to 65535.)


Assembler Syntax
----------------
  DBT Dn,<label>


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


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


Instruction Size and Cycles to Execute
--------------------------------------
  Condition Counter   Branch    #   p
  true    na    No    4  12

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

1.19 AmigaFlight® Help: Decrement and Branch Always unless Count = -1

```
DBF Decrement and Branch Always unless Count = -1
=================================================

  If the specified condition is false, decrement the destination
  data register, and then compare the destination register with -1.
  If the data register doesn't equal -1, continue processing at the
  specified label. If either of the conditions fail, then continue
  instruction execution with the next instruction. This instruction
  uses a 16-bit displacement as a label offset. This means that the
  relative offset of the label must be in the range of -32768 to
  32767 bytes in distance from the current program counter.
  This instruction provides a primitive looping construct similar to
  the REPEAT UNTIL looping construct of Pascal/ADA/Basic/C etc. The
  DBcc instruction may be thought of as a REPEAT loop UNTIL either
  the condition becomes true, or the loop counter goes below 0.
  This, of course, is assuming that the destination data register
  was initially set to a positive value. (This instruction uses the
  bottom 16 bits of the destination data register for a loop
  counter, 0 to 65535.)

  Decrement data register Dn (low order word) and Branch if result
  not -1


Assembler Syntax
----------------
  DBF Dn,<label>


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


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


Instruction Size and Cycles to Execute
--------------------------------------
  Condition Counter   Branch    #   p
  false   <>-1    Yes   4  10
  false   = -1    No    4  14

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

1.20 AmigaFlight® Help: Decrement and Branch until High or Count = -1

```
DBHI  Decrement and Branch until High or Count = -1
==================================================
```

If the specified condition is false, decrement the destination data register, and then compare the destination register with -1. If the data register doesn't equal -1, continue processing at the specified label. If either of the conditions fail, then continue instruction execution with the next instruction. This instruction uses a 16-bit displacement as a label offset. This means that the relative offset of the label must be in the range of -32768 to 32767 bytes in distance from the current program counter.
This instruction provides a primitive looping construct similar to the REPEAT UNTIL looping construct of Pascal/ADA/Basic/C etc. The DBcc instruction may be thought of as a REPEAT loop UNTIL either the condition becomes true, or the loop counter goes below 0. This, of course, is assuming that the destination data register was initially set to a positive value. (This instruction uses the bottom 16 bits of the destination data register for a loop counter, 0 to 65535.)

If C'.Z' = 0 then Decrement data register Dn (low order word) and Branch if result not -1

where . = Boolean AND
      ' = Complement

Assembler Syntax

```
DBHI  Dn,<label>
```

Data Size

Word

Status Flags

N	Not affected
Z	Not affected
V	Not affected
C	Not affected
X	Not affected

Instruction Size and Cycles to Execute

Condition	Counter	Branch	#	p
false	<>-1	Yes	4	10
true	na	No	4	12
false	= -1	No	4	14

= no. of instruction bytes

p = no. of instruction clock periods

1.21 AmigaFlight® Help: Decrement and Branch until Low or Same or Count = -1

DBLS Decrement and Branch until Low or Same or Count = -1

If the specified condition is false, decrement the destination data register, and then compare the destination register with -1. If the data register doesn't equal -1, continue processing at the specified label. If either of the conditions fail, then continue instruction execution with the next instruction. This instruction uses a 16-bit displacement as a label offset. This means that the relative offset of the label must be in the range of -32768 to 32767 bytes in distance from the current program counter.
This instruction provides a primitive looping construct similar to the REPEAT UNTIL looping construct of Pascal/ADA/Basic/C etc. The DBcc instruction may be thought of as a REPEAT loop UNTIL either the condition becomes true, or the loop counter goes below 0. This, of course, is assuming that the destination data register was initially set to a positive value. (This instruction uses the bottom 16 bits of the destination data register for a loop counter, 0 to 65535.)

If C+Z = 0 then Decrement data register Dn (low order word) and Branch if result not -1

where + = Boolean OR

Assembler Syntax

DBLS Dn,<label>

Data Size

Word

Status Flags

N Not affected
Z Not affected
V Not affected
C Not affected
X Not affected

Instruction Size and Cycles to Execute

Condition	Counter	Branch	#	p
false	<>-1	Yes	4	10
true	na	No	4	12
false	= -1	No	4	14

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

1.22 AmigaFlight® Help: Decrement and Branch until Carry Clear or Count = -1

```
DBCC  Decrement and Branch until Carry Clear or Count = -1
==========================================================

  If  the  specified condition is false,  decrement  the destination
  data register, and then compare the destination  register with -1.
  If the data  register doesn't equal -1, continue processing at the
  specified label. If either of the  conditions fail, then  continue
  instruction execution with  the next instruction. This instruction
  uses a 16-bit displacement as a  label offset. This means that the
  relative offset of the label  must be  in  the  range of -32768 to
  32767 bytes in distance from the current program counter.
  This instruction provides a primitive looping construct similar to
  the REPEAT UNTIL looping  construct of Pascal/ADA/Basic/C etc. The
  DBcc  instruction may be thought  of as a REPEAT loop UNTIL either
  the  condition  becomes  true,  or the loop  counter goes below 0.
  This, of  course, is  assuming that  the destination data register
  was  initially set to a positive value. (This instruction uses the
  bottom  16  bits  of  the  destination  data  register  for a loop
  counter, 0 to 65535.)

  If C = 1 then Decrement data register Dn (low order word) and
  Branch if result not -1


Assembler Syntax
----------------
  DBCC  Dn,<label>


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


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


Instruction Size and Cycles to Execute
--------------------------------------
  Condition Counter   Branch    #   p
  false   <>-1    Yes   4  10
  true    na    No    4  12
  false   = -1    No    4  14
```

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

1.23 AmigaFlight® Help: Decrement and Branch until Carry Set or Count = -1

```
DBCS  Decrement and Branch until Carry Set or Count = -1
=======================================================

  If  the  specified condition is false,  decrement  the destination
  data register, and then compare the destination  register with -1.
  If the data  register doesn't equal -1, continue processing at the
  specified label. If either of the  conditions fail, then  continue
  instruction execution with  the next instruction. This instruction
  uses a 16-bit displacement as a  label offset. This means that the
  relative offset of the label  must be  in  the  range of -32768 to
  32767 bytes in distance from the current program counter.
  This instruction provides a primitive looping construct similar to
  the REPEAT UNTIL looping  construct of Pascal/ADA/Basic/C etc. The
  DBcc  instruction may be thought  of as a REPEAT loop UNTIL either
  the  condition  becomes  true,  or the loop  counter goes below 0.
  This, of  course, is  assuming that  the destination data register
  was  initially set to a positive value. (This instruction uses the
  bottom  16  bits  of  the  destination  data  register  for a loop
  counter, 0 to 65535.)

  If C = 0 then Decrement data register Dn (low order word) and
  Branch if result not -1


Assembler Syntax
----------------
  DBCS  Dn,<label>


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


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


Instruction Size and Cycles to Execute
--------------------------------------
  Condition Counter   Branch    #   p
  false   <>-1    Yes   4  10
  true    na    No    4  12
  false   = -1    No    4  14
```

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

1.24 AmigaFlight® Help: Decrement and Branch until Not Equal or Count = -1

```
DBNE  Decrement and Branch until Not Equal or Count = -1
=======================================================

  If  the  specified condition is false,  decrement  the destination
  data register, and then compare the destination  register with -1.
  If the data  register doesn't equal -1, continue processing at the
  specified label. If either of the  conditions fail, then  continue
  instruction execution with  the next instruction. This instruction
  uses a 16-bit displacement as a  label offset. This means that the
  relative offset of the label  must be  in  the  range of -32768 to
  32767 bytes in distance from the current program counter.
  This instruction provides a primitive looping construct similar to
  the REPEAT UNTIL looping  construct of Pascal/ADA/Basic/C etc. The
  DBcc  instruction may be thought  of as a REPEAT loop UNTIL either
  the  condition  becomes  true,  or the loop  counter goes below 0.
  This, of  course, is  assuming that  the destination data register
  was  initially set to a positive value. (This instruction uses the
  bottom  16  bits  of  the  destination  data  register  for a loop
  counter, 0 to 65535.)

  If Z = 1 then Decrement data register Dn (low order word) and
  Branch if result not -1


Assembler Syntax
----------------
  DBNE  Dn,<label>


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


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


Instruction Size and Cycles to Execute
--------------------------------------
  Condition Counter   Branch    #   p
  false   <>-1    Yes   4  10
  true    na    No    4  12
  false   = -1    No    4  14
```

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

1.25 AmigaFlight® Help: Decrement and Branch until Equal or Count = -1

```
DBEQ  Decrement and Branch until Equal or Count = -1
=====================================================

  If  the  specified condition is false,  decrement  the destination
  data register, and then compare the destination  register with -1.
  If the data  register doesn't equal -1, continue processing at the
  specified label. If either of the  conditions fail, then  continue
  instruction execution with  the next instruction. This instruction
  uses a 16-bit displacement as a  label offset. This means that the
  relative offset of the label  must be  in  the  range of -32768 to
  32767 bytes in distance from the current program counter.
  This instruction provides a primitive looping construct similar to
  the REPEAT UNTIL looping  construct of Pascal/ADA/Basic/C etc. The
  DBcc  instruction may be thought  of as a REPEAT loop UNTIL either
  the  condition  becomes  true,  or the loop  counter goes below 0.
  This, of  course, is  assuming that  the destination data register
  was  initially set to a positive value. (This instruction uses the
  bottom  16  bits  of  the  destination  data  register  for a loop
  counter, 0 to 65535.)

  If Z = 0 then Decrement data register Dn (low order word) and
  Branch if result not -1


Assembler Syntax
----------------
  DBEQ  Dn,<label>


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


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


Instruction Size and Cycles to Execute
--------------------------------------
  Condition Counter   Branch    #   p
  false   <>-1    Yes   4  10
  true    na    No    4  12
  false   = -1    No    4  14
```

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

1.26 AmigaFlight® Help: Decrement and Branch until No Overflow or Count = -1

```
DBVC  Decrement and Branch until No Overflow or Count = -1
==========================================================

  If  the  specified condition is false,  decrement  the destination
  data register, and then compare the destination  register with -1.
  If the data  register doesn't equal -1, continue processing at the
  specified label. If either of the  conditions fail, then  continue
  instruction execution with  the next instruction. This instruction
  uses a 16-bit displacement as a  label offset. This means that the
  relative offset of the label  must be  in  the  range of -32768 to
  32767 bytes in distance from the current program counter.
  This instruction provides a primitive looping construct similar to
  the REPEAT UNTIL looping  construct of Pascal/ADA/Basic/C etc. The
  DBcc  instruction may be thought  of as a REPEAT loop UNTIL either
  the  condition  becomes  true,  or the loop  counter goes below 0.
  This, of  course, is  assuming that  the destination data register
  was  initially set to a positive value. (This instruction uses the
  bottom  16  bits  of  the  destination  data  register  for a loop
  counter, 0 to 65535.)

  If V = 1 then Decrement data register Dn (low order word) and
  Branch if result not -1


Assembler Syntax
----------------
  DBVC  Dn,<label>


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


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


Instruction Size and Cycles to Execute
--------------------------------------
  Condition Counter   Branch    #   p
  false   <>-1    Yes   4  10
  true    na    No    4  12
  false   = -1    No    4  14
```

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

1.27 AmigaFlight® Help: Decrement and Branch until Overflow or Count = -1

```
DBVS  Decrement and Branch until Overflow or Count = -1
=======================================================

  If  the  specified condition is false,  decrement  the destination
  data register, and then compare the destination  register with -1.
  If the data  register doesn't equal -1, continue processing at the
  specified label. If either of the  conditions fail, then  continue
  instruction execution with  the next instruction. This instruction
  uses a 16-bit displacement as a  label offset. This means that the
  relative offset of the label  must be  in  the  range of -32768 to
  32767 bytes in distance from the current program counter.
  This instruction provides a primitive looping construct similar to
  the REPEAT UNTIL looping  construct of Pascal/ADA/Basic/C etc. The
  DBcc  instruction may be thought  of as a REPEAT loop UNTIL either
  the  condition  becomes  true,  or the loop  counter goes below 0.
  This, of  course, is  assuming that  the destination data register
  was  initially set to a positive value. (This instruction uses the
  bottom  16  bits  of  the  destination  data  register  for a loop
  counter, 0 to 65535.)

  If V = 0 then Decrement data register Dn (low order word) and
  Branch if result not -1


Assembler Syntax
----------------
  DBVS  Dn,<label>


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


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


Instruction Size and Cycles to Execute
--------------------------------------
  Condition Counter   Branch    #   p
  false   <>-1    Yes   4  10
  true    na    No    4  12
  false   = -1    No    4  14
```

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

1.28 AmigaFlight® Help: Decrement and Branch until Plus or Count = -1

```
DBPL  Decrement and Branch until Plus or Count = -1
==================================================

  If  the  specified condition is false,  decrement  the destination
  data register, and then compare the destination  register with -1.
  If the data  register doesn't equal -1, continue processing at the
  specified label. If either of the  conditions fail, then  continue
  instruction execution with  the next instruction. This instruction
  uses a 16-bit displacement as a  label offset. This means that the
  relative offset of the label  must be  in  the  range of -32768 to
  32767 bytes in distance from the current program counter.
  This instruction provides a primitive looping construct similar to
  the REPEAT UNTIL looping  construct of Pascal/ADA/Basic/C etc. The
  DBcc  instruction may be thought  of as a REPEAT loop UNTIL either
  the  condition  becomes  true,  or the loop  counter goes below 0.
  This, of  course, is  assuming that  the destination data register
  was  initially set to a positive value. (This instruction uses the
  bottom  16  bits  of  the  destination  data  register  for a loop
  counter, 0 to 65535.)

  If N = 1 then Decrement data register Dn (low order word) and
  Branch if result not -1

  where . = Boolean AND
        + = Boolean OR
        ' = Complement


Assembler Syntax
----------------
  DBPL  Dn,<label>


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


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


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

```
  Condition Counter   Branch    #   p
  false   <>-1    Yes   4  10
  true    na    No    4  12
  false   = -1    No    4  14

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

1.29 AmigaFlight® Help: Decrement and Branch until Minus or Count = -1

```
DBMI  Decrement and Branch until Minus or Count = -1
=====================================================

  If  the  specified condition is false,  decrement  the destination
  data register, and then compare the destination  register with -1.
  If the data  register doesn't equal -1, continue processing at the
  specified label. If either of the  conditions fail, then  continue
  instruction execution with  the next instruction. This instruction
  uses a 16-bit displacement as a  label offset. This means that the
  relative offset of the label  must be  in  the  range of -32768 to
  32767 bytes in distance from the current program counter.
  This instruction provides a primitive looping construct similar to
  the REPEAT UNTIL looping  construct of Pascal/ADA/Basic/C etc. The
  DBcc  instruction may be thought  of as a REPEAT loop UNTIL either
  the  condition  becomes  true,  or the loop  counter goes below 0.
  This, of  course, is  assuming that  the destination data register
  was  initially set to a positive value. (This instruction uses the
  bottom  16  bits  of  the  destination  data  register  for a loop
  counter, 0 to 65535.)

  If N = 0 then Decrement data register Dn (low order word) and
  Branch if result not -1


Assembler Syntax
----------------
  DBMI  Dn,<label>


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


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


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

```
  Condition Counter   Branch    #   p
  false   <>-1    Yes   4  10
  true    na    No    4  12
  false   = -1    No    4  14

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

1.30 AmigaFlight® Help: Decrement and Branch until Greater or Equal or Count = -1

```
DBGE  Decrement and Branch until Greater or Equal or Count = -1
================================================================
```

If the specified condition is false, decrement the destination data register, and then compare the destination register with -1. If the data register doesn't equal -1, continue processing at the specified label. If either of the conditions fail, then continue instruction execution with the next instruction. This instruction uses a 16-bit displacement as a label offset. This means that the relative offset of the label must be in the range of -32768 to 32767 bytes in distance from the current program counter.
This instruction provides a primitive looping construct similar to the REPEAT UNTIL looping construct of Pascal/ADA/Basic/C etc. The DBcc instruction may be thought of as a REPEAT loop UNTIL either the condition becomes true, or the loop counter goes below 0. This, of course, is assuming that the destination data register was initially set to a positive value. (This instruction uses the bottom 16 bits of the destination data register for a loop counter, 0 to 65535.)

If N.V+N'.V' = 0 then Decrement data register Dn (low order word) and Branch if result not -1

where . = Boolean AND
+ = Boolean OR
' = Complement

```
Assembler Syntax
----------------
  DBGE  Dn,<label>


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


Status Flags
------------
  N  Not affected
  Z  Not affected
  V  Not affected
```

```
  C  Not affected
  X  Not affected


Instruction Size and Cycles to Execute
--------------------------------------
  Condition Counter   Branch    #   p
  false   <>-1    Yes   4  10
  true    na    No    4  12
  false   = -1    No    4  14


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

1.31 AmigaFlight® Help: Decrement and Branch until Less or Count = -1

```
DBLT  Decrement and Branch until Less or Count = -1
===================================================

  If  the  specified condition is false,  decrement  the destination
  data register, and then compare the destination  register with -1.
  If the data  register doesn't equal -1, continue processing at the
  specified label. If either of the  conditions fail, then  continue
  instruction execution with  the next instruction. This instruction
  uses a 16-bit displacement as a  label offset. This means that the
  relative offset of the label  must be  in  the  range of -32768 to
  32767 bytes in distance from the current program counter.
  This instruction provides a primitive looping construct similar to
  the REPEAT UNTIL looping  construct of Pascal/ADA/Basic/C etc. The
  DBcc  instruction may be thought  of as a REPEAT loop UNTIL either
  the  condition  becomes  true,  or the loop  counter goes below 0.
  This, of  course, is  assuming that  the destination data register
  was  initially set to a positive value. (This instruction uses the
  bottom  16  bits  of  the  destination  data  register  for a loop
  counter, 0 to 65535.)

  If N.V'+N'.V = 0 then Decrement data register Dn (low order word)
  and Branch if result not -1

  where . = Boolean AND
        + = Boolean OR
        ' = Complement


Assembler Syntax
----------------
  DBLT  Dn,<label>


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

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


Instruction Size and Cycles to Execute
--------------------------------------
  Condition Counter   Branch    #   p
  false   <>-1    Yes   4  10
  true    na    No    4  12
  false   = -1    No    4  14

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

1.32 AmigaFlight® Help: Decrement and Branch until Greater or Count = -1

```
DBGT  Decrement and Branch until Greater or Count = -1
======================================================

  If  the  specified condition is false,  decrement  the destination
  data register, and then compare the destination  register with -1.
  If the data  register doesn't equal -1, continue processing at the
  specified label. If either of the  conditions fail, then  continue
  instruction execution with  the next instruction. This instruction
  uses a 16-bit displacement as a  label offset. This means that the
  relative offset of the label  must be  in  the  range of -32768 to
  32767 bytes in distance from the current program counter.
  This instruction provides a primitive looping construct similar to
  the REPEAT UNTIL looping  construct of Pascal/ADA/Basic/C etc. The
  DBcc  instruction may be thought  of as a REPEAT loop UNTIL either
  the  condition  becomes  true,  or the loop  counter goes below 0.
  This, of  course, is  assuming that  the destination data register
  was  initially set to a positive value. (This instruction uses the
  bottom  16  bits  of  the  destination  data  register  for a loop
  counter, 0 to 65535.)

  If N.V.Z'+N'.V'.Z' = 0 then Decrement data register Dn (low order
  word) and Branch if result not -1

  where . = Boolean AND
        + = Boolean OR
        ' = Complement


Assembler Syntax
----------------
  DBGT  Dn,<label>


Data Size
```

```
---------
  Word


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


Instruction Size and Cycles to Execute
--------------------------------------
  Condition Counter   Branch    #   p
  false   <>-1    Yes   4  10
  true    na    No    4  12
  false   = -1    No    4  14

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

1.33 AmigaFlight® Help: Decrement and Branch until Less or Equal or Count = -1

```
DBLE  Decrement and Branch until Less or Equal or Count = -1
============================================================

  If  the  specified condition is false,  decrement  the destination
  data register, and then compare the destination  register with -1.
  If the data  register doesn't equal -1, continue processing at the
  specified label. If either of the  conditions fail, then  continue
  instruction execution with  the next instruction. This instruction
  uses a 16-bit displacement as a  label offset. This means that the
  relative offset of the label  must be  in  the  range of -32768 to
  32767 bytes in distance from the current program counter.
  This instruction provides a primitive looping construct similar to
  the REPEAT UNTIL looping  construct of Pascal/ADA/Basic/C etc. The
  DBcc  instruction may be thought  of as a REPEAT loop UNTIL either
  the  condition  becomes  true,  or the loop  counter goes below 0.
  This, of  course, is  assuming that  the destination data register
  was  initially set to a positive value. (This instruction uses the
  bottom  16  bits  of  the  destination  data  register  for a loop
  counter, 0 to 65535.)

  If Z+N.V'+N'.V = 0 then Decrement data register Dn (low order
  word) and Branch if result not -1

  where . = Boolean AND
        + = Boolean OR
        ' = Complement


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

```
  DBLE  Dn,<label>


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


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


Instruction Size and Cycles to Execute
--------------------------------------
  Condition Counter   Branch    #   p
  false   <>-1    Yes   4  10
  true    na    No    4  12
  false   = -1    No    4  14

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

1.34 AmigaFlight® Help: Decrement and Branch Always unless Count = -1

```
DBRA  Decrement and Branch Always unless Count = -1 (Same as DBF)
================================================================

  If  the  specified condition is false,  decrement  the destination
  data register, and then compare the destination  register with -1.
  If the data  register doesn't equal -1, continue processing at the
  specified label. If either of the  conditions fail, then  continue
  instruction execution with  the next instruction. This instruction
  uses a 16-bit displacement as a  label offset. This means that the
  relative offset of the label  must be  in  the  range of -32768 to
  32767 bytes in distance from the current program counter.
  This instruction provides a primitive looping construct similar to
  the REPEAT UNTIL looping  construct of Pascal/ADA/Basic/C etc. The
  DBcc  instruction may be thought  of as a REPEAT loop UNTIL either
  the  condition  becomes  true,  or the loop  counter goes below 0.
  This, of  course, is  assuming that  the destination data register
  was  initially set to a positive value. (This instruction uses the
  bottom  16  bits  of  the  destination  data  register  for a loop
  counter, 0 to 65535.)

  Decrement data register Dn (low order word) and Branch if result
  not -1


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

```
  DBRA  Dn,<label>


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


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


Instruction Size and Cycles to Execute
--------------------------------------
  Condition Counter   Branch    #   p
  false   <>-1    Yes   4  10
  false   = -1    No    4  14

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

1.35 AmigaFlight® Help: Jump

```
JUMP  Jump
============

  Continue program execution at  the new  address  specified  by the
  instruction.

  Destination -> PC


Assembler Syntax
----------------
  JMP <ea>

  <ea> - control


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
---------
  Unsized


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


Instruction Size and Cycles to Execute
--------------------------------------
  <ea>    # p
  (An)    2 8
  d16(An)   4 10
  d8(An,Ri) 4 14
  Abs short 4 10
  Abs long  6 12
  d16(PC)   4 10
  d8(PC,Ri) 4 14

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

1.36 AmigaFlight® Help: Jump to Subroutine

```
JSR Jump to Subroutine
==========================

  Push   the  long-word  address  of  the  instruction   immediately
  following the JSR instruction onto the  stack, and  then  continue
  program execution at the new adress specified by the instruction.

  PC -> SP@- : Destination -> PC


Assembler Syntax
----------------
  JSR <ea>

  <ea> - control


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
---------
  Unsized


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


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

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

1.37 AmigaFlight® Help: Return and Restore Condition Codes

```
RTR Return and Restore Condition Codes
======================================

  Load the condition code and a new  program counter from the stack.
  Proceed with execution at the new program counter address.

  SP@+ -> CC : SP@+ -> PC
```

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


Data Size
---------
  Unsized


Status Flags
------------
  Set according to word on stack


Instruction Size and Cycles to Execute
--------------------------------------
       # p
  Unsized   2 20

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

1.38 AmigaFlight® Help: Return from Subroutine

```
RTS Return from Subroutine
==========================

  Load a new program counter from the top of the stack, and  proceed
  with execution at this new address.

  SP@+ -> PC


Assembler Syntax
----------------
  RTS


Data Size
---------
  Unsized


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


Instruction Size and Cycles to Execute
```

```
--------------------------------------
      # p
  Unsized   2 16

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

1.39 AmigaFlight® Help: Set if Carry Clear

```
SCC Set if Carry Clear
==========================

  Set  the specified  byte  address to  0xFF  if  the 'Carry  Clear'
  condition is met, or to 0x00 if the condition is not met.

  If C = 0 then 1's -> destn else 0's -> destn


Assembler Syntax
----------------
  SCC <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  Not affected
  Z  Not affected
  V  Not affected
  C  Not affected
  X  Not affected
```

```
Instruction Size and Cycles to Execute
--------------------------------------
      True    False
  <ea>    #     p   #     p
  Dn    2     6   2     4
  (An)    2    13   2    13
  (An)+   2    13   2    13
  -(An)   2    15   2    15
  d16(An)   4    17   4    17
  d8(An,Ri) 4    19   4    19
  Abs short 4    17   4    17
  Abs long  6    21   6    21

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

1.40 AmigaFlight® Help: Set if Carry Set

```
SCS Set if Carry Set
========================

  Set  the  specified  byte  address  to  0xFF  if  the 'Carry  Set'
  condition is met, or to 0x00 if the condition is not met.

  If C = 1 then 1's -> destn else 0's -> destn


Assembler Syntax
----------------
  SCS <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  Not affected
  Z  Not affected
  V  Not affected
  C  Not affected
  X  Not affected


Instruction Size and Cycles to Execute
--------------------------------------
      True     False
  <ea>    #      p   #     p
  Dn    2      6   2     4
  (An)    2    13   2    13
  (An)+   2    13   2    13
  -(An)   2    15   2    15
  d16(An)   4    17   4    17
  d8(An,Ri) 4    19   4    19
  Abs short 4    17   4    17
  Abs long  6    21   6    21

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

1.41 AmigaFlight® Help: Set if Equal

```
SEQ Set if Equal
====================

  Set the specified byte address to 0xFF if the 'Equal' condition is
  met, or to 0x00 if the condition is not met.

  If Z = 1 then 1's -> destn else 0's -> destn


Assembler Syntax
----------------
  SEQ <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  Not affected
  Z  Not affected
  V  Not affected
  C  Not affected
  X  Not affected


Instruction Size and Cycles to Execute
--------------------------------------
      True        False
  <ea>    #     p     #     p
  Dn    2     6     2     4
  (An)    2    13     2    13
  (An)+   2    13     2    13
  -(An)   2    15     2    15
  d16(An)   4    17     4    17
  d8(An,Ri) 4    19     4    19
  Abs short 4    17     4    17
  Abs long  6    21     6    21

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

1.42 AmigaFlight® Help: Set Never

```
SF  Set Never
=================

  Set  the  specified  byte  address  to  0xFF  if  the 'Set  Never'
  condition is met, or to 0x00 if the condition is not met.

  0's -> destn always


Assembler Syntax
----------------
  SF  <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  Not affected
  Z  Not affected
  V  Not affected
  C  Not affected
  X  Not affected


Instruction Size and Cycles to Execute
--------------------------------------
  <ea>    # p
  Dn    2 4
  (An)    2 13
  (An)+   2 13
  -(An)   2 15
  d16(An)   4 17
  d8(An,Ri) 4 19
  Abs short 4 17
  Abs long  6 21

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

1.43 AmigaFlight® Help: Set if Greater or Equal

```
SGE Set if Greater of Equal
===========================
```

Set the specified byte address to 0xFF if the 'Greater or Equal' condition is met, or to 0x00 if the condition is not met.

If N.V+N'.V' = 1 then 1's -> destn else 0's -> destn
where . = Boolean AND
+ = Boolean OR
' = Complement

Assembler Syntax

SGE <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	Not affected
Z	Not affected
V	Not affected
C	Not affected
X	Not affected

Instruction Size and Cycles to Execute

	True		False	
<ea>	#	p	#	p
Dn	2	6	2	4
(An)	2	13	2	13
(An)+	2	13	2	13
-(An)	2	15	2	15
d16(An)	4	17	4	17

```
  d8(An,Ri) 4    19   4    19
  Abs short 4    17   4    17
  Abs long  6    21   6    21

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

1.44 AmigaFlight® Help: Set if Greater

```
SGT Set if Greater
======================

  Set the specified byte  address to 0xFF if the 'Greater' condition
  is met, or to 0x00 if the condition is not met.

  If N.V.Z'+N'.V'.Z' = 1 then 1's -> destn else 0's -> destn
  where . = Boolean AND
        + = Boolean OR
        ' = Complement


Assembler Syntax
----------------
  SGT <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  Not affected
  Z  Not affected
```

```
  V  Not affected
  C  Not affected
  X  Not affected


Instruction Size and Cycles to Execute
--------------------------------------
      True    False
  <ea>    #     p   #     p
  Dn    2     6   2     4
  (An)    2    13   2    13
  (An)+   2    13   2    13
  -(An)   2    15   2    15
  d16(An)   4    17   4    17
  d8(An,Ri) 4    19   4    19
  Abs short 4    17   4    17
  Abs long  6    21   6    21

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

1.45 AmigaFlight® Help: Set if High

```
SHI Set if High
===============

  Set the specified byte  address to 0xFF if the 'High' condition is
  met, or to 0x00 if the condition is not met.

  If C'.Z' = 1 then 1's -> destn else 0's -> destn
  where . = Boolean AND
        ' = Complement


Assembler Syntax
----------------
  SHI <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  Not affected
  Z  Not affected
  V  Not affected
  C  Not affected
  X  Not affected


Instruction Size and Cycles to Execute
--------------------------------------
      True    False
  <ea>          #     p     #     p
  Dn            2     6     2     4
  (An)          2    13     2    13
  (An)+         2    13     2    13
  -(An)         2    15     2    15
  d16(An)       4    17     4    17
  d8(An,Ri)     4    19     4    19
  Abs short     4    17     4    17
  Abs long      6    21     6    21

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

1.46 AmigaFlight® Help: Set if Less or Equal

```
SLE Set if Less or Equal
============================

  Set the specified byte  address  to  0xFF  if  the 'Less or Equal'
  condition is met, or to 0x00 if the condition is not met.

  If Z+N.V'+N'.V = 1 then 1's -> destn else 0's -> destn
  where . = Boolean AND
        + = Boolean OR
        ' = Complement


Assembler Syntax
----------------
  SLE <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  Not affected
  Z  Not affected
  V  Not affected
  C  Not affected
  X  Not affected


Instruction Size and Cycles to Execute
--------------------------------------
       True    False
  <ea>    #     p   #     p
  Dn    2     6   2     4
  (An)    2    13   2    13
  (An)+   2    13   2    13
  -(An)   2    15   2    15
  d16(An)   4    17   4    17
  d8(An,Ri) 4    19   4    19
  Abs short 4    17   4    17
  Abs long  6    21   6    21

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

1.47 AmigaFlight® Help: Set if Lower or Same

```
SLS Set if Lower or Same
============================

  Set the specified byte  address  to  0xFF  if  the 'Lower or Same'
  condition is met, or to 0x00 if the condition is not met.
```

```
  If C+Z = 1 then 1's -> destn else 0's -> destn
  where + = Boolean OR
```

Assembler Syntax

```
  SLS <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	Not affected
Z	Not affected
V	Not affected
C	Not affected
X	Not affected

Instruction Size and Cycles to Execute

	True		False	
<ea>	#	p	#	p
Dn	2	6	2	4
(An)	2	13	2	13
(An)+	2	13	2	13
-(An)	2	15	2	15
d16(An)	4	17	4	17
d8(An,Ri)	4	19	4	19
Abs short	4	17	4	17
Abs long	6	21	6	21

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

1.48 AmigaFlight® Help: Set if Less

```
SLT Set if Less
===============

  Set the specified byte  address to 0xFF if the 'Less' condition is
  met, or to 0x00 if the condition is not met.

  If N.V'+N'.V = 1 then 1's -> destn else 0's -> destn
  where . = Boolean AND
        + = Boolean OR
        ' = Complement


Assembler Syntax
----------
  SLT <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  Not affected
  Z  Not affected
  V  Not affected
  C  Not affected
  X  Not affected
```

```
Instruction Size and Cycles to Execute
--------------------------------------
      True    False
  <ea>    #     p   #     p
  Dn    2     6   2     4
  (An)    2    13   2    13
  (An)+   2    13   2    13
  -(An)   2    15   2    15
  d16(An)   4    17   4    17
  d8(An,Ri) 4    19   4    19
  Abs short 4    17   4    17
  Abs long  6    21   6    21

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

1.49 AmigaFlight® Help: Set if Minus

```
SMI Set if Minus
====================

  Set the specified byte address to 0xFF if the 'Minus' condition is
  met, or to 0x00 if the condition is not met.

  If N = 1 then 1's -> destn else 0's -> destn


Assembler Syntax
----------------
  SMI <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  Not affected
  Z  Not affected
  V  Not affected
  C  Not affected
  X  Not affected


Instruction Size and Cycles to Execute
--------------------------------------
      True    False
  <ea>    #     p   #     p
  Dn    2     6   2     4
  (An)    2    13   2    13
  (An)+   2    13   2    13
  -(An)   2    15   2    15
  d16(An)   4    17   4    17
  d8(An,Ri) 4    19   4    19
  Abs short 4    17   4    17
  Abs long  6    21   6    21

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

1.50 AmigaFlight® Help: Set if Not Equal

```
SNE Set if Not Equal
========================

  Set  the  specified  byte  address  to  0xFF  if  the 'Not  Equal'
  condition is met, or to 0x00 if the condition is not met.

  If Z = 0 then 1's -> destn else 0's -> destn


Assembler Syntax
----------------
  SNE <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  Not affected
  Z  Not affected
  V  Not affected
  C  Not affected
  X  Not affected


Instruction Size and Cycles to Execute
--------------------------------------
      True    False
  <ea>    #     p   #     p
  Dn    2     6   2     4
  (An)    2    13   2    13
  (An)+   2    13   2    13
  -(An)   2    15   2    15
  d16(An)   4    17   4    17
  d8(An,Ri) 4    19   4    19
  Abs short 4    17   4    17
  Abs long  6    21   6    21

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

1.51 AmigaFlight® Help: Set if Plus

```
SPL Set if Plus
===============

  Set the specified byte  address to 0xFF if the 'Plus' condition is
  met, or to 0x00 if the condition is not met.

  If N = 0 then 1's -> destn else 0's -> destn


Assembler Syntax
----------------
  SPL <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  Not affected
  Z  Not affected
  V  Not affected
  C  Not affected
  X  Not affected


Instruction Size and Cycles to Execute
--------------------------------------
       True    False
  <ea>    #     p   #     p
  Dn    2     6   2     4
  (An)    2    13   2    13
  (An)+   2    13   2    13
  -(An)   2    15   2    15
  d16(An)   4    17   4    17
  d8(An,Ri) 4    19   4    19
  Abs short 4    17   4    17
  Abs long  6    21   6    21

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

1.52 AmigaFlight® Help: Set Always

```
ST  Set Always
==================
```

Set the specified byte address to 0xFF if the 'Always' condition is met, or to 0x00 if the condition is not met.

1's -> destn always

Assembler Syntax

ST <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	Not affected
Z	Not affected
V	Not affected
C	Not affected
X	Not affected

Instruction Size and Cycles to Execute

<ea>	#	p
Dn	2	6
(An)	2	13
(An)+	2	13
-(An)	2	15
d16(An)	4	17
d8(An,Ri)	4	19
Abs short	4	17
Abs long	6	21

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

1.53 AmigaFlight® Help: Set if No Overflow

```
SVC Set if No Overflow
==========================

  Set the specified  byte address  to  0xFF  if  the  'No  Overflow'
  condition is met, or to 0x00 if the condition is not met.

  If V = 0 then 1's -> destn else 0's -> destn


Assembler Syntax
----------------
  SVC <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  Not affected
  Z  Not affected
  V  Not affected
  C  Not affected
  X  Not affected


Instruction Size and Cycles to Execute
```

```
--------------------------------------
      True    False
  <ea>    #     p   #     p
  Dn    2     6   2     4
  (An)    2    13   2    13
  (An)+   2    13   2    13
  -(An)   2    15   2    15
  d16(An)   4    17   4    17
  d8(An,Ri) 4    19   4    19
  Abs short 4    17   4    17
  Abs long  6    21   6    21

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

1.54 AmigaFlight® Help: Set if Overflow

```
SVS Set if Overflow
=======================

  Set the specified byte address to 0xFF if the 'Overflow' condition
  is met, or to 0x00 if the condition is not met.

  If V = 1 then 1's -> destn else 0's -> destn


Assembler Syntax
----------------
  SVS <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  Not affected
  Z  Not affected
  V  Not affected
  C  Not affected
  X  Not affected


Instruction Size and Cycles to Execute
--------------------------------------
      True    False
  <ea>    #     p   #     p
  Dn    2     6   2     4
  (An)    2    13   2    13
  (An)+   2    13   2    13
  -(An)   2    15   2    15
  d16(An)   4    17   4    17
  d8(An,Ri) 4    19   4    19
  Abs short 4    17   4    17
  Abs long  6    21   6    21

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