General Information
Author: Andrew John Jacobs
Published: 1999

This file contains a number of useful 6502 algorithms for number, string and memory operations. The code is written in the form of macros rather than subroutines to make them more flexible.

The routines in the library assume that 16 and 32 bit numbers are represented in little endian order, that is the least significant byte in the lowest memory location, so that they can be applied to addresses as well as pure numbers.

The string routines assume that they are working with null terminated 'C' style strings.

The main routines sacrifice code size for speed and are coded without any iteration. Compact versions which use iteration are provided for some algorithms and have the same name as the original routine with a 'C' suffix (eg. _XFR32 => _XFR32C).

Some of the macros use 65SC02 instructions for speed or to reduce the amount code generated if the assembler will accept them.

Where possible the macros detect optimizable cases and generate more efficient code.

Bugs & Enhancments:

If you find a bug I missed or have a new routine you would like to submit to the library then mail me at:

Andrew@obelisk.demon.co.uk

fount at http://www.obelisk.demon.co.uk/6502/algorithms.html

```
      NOLIST
;------------------------------------------------
; 6502 Standard Macro Library
;------------------------------------------------
; Copyright (C),1999 Andrew John Jacobs.
; All rights reserved.
;------------------------------------------------

;------------------------------------------------
; Revision History:
;
; 16-Aug-99 AJJ   Initial version.
;
; 14-Nov-01 AJJ Finally got around to filling in
;      some missing comments.
;
;------------------------------------------------

; Notes:
;
; This file contains a number of useful 6502
; algorithms for number, string and memory
; operations. The code is written in the form of
; macros rather than subroutines to make them
; more flexible.
;
; The routines in the library assume that 16 and
```

```
; 32 bit numbers are represented in little endian
; order, that is the least significant byte in
; the lowest memory location, so that they can
; be applied to addresses as well as pure
; numbers.
;
; The string routines assume that they are
; working with null terminated 'C' style strings.
;
; The main routines sacrifice code size for speed
; and are coded without any iteration. Compact
; versions which use iteration are provided for
; some algorithms and have the same name as the
; original routine with a 'C' suffix (eg. _XFR32
; => _XFR32C).
;
; Some of the macros use 65SC02 instructions for
; speed or to reduce the amount code generated if
; the assembler will accept them.
;
; Where possible the macros detect optimizable
; cases and generate more efficient code.
;
; Bugs & Enhancments:
;
; If you find a bug I missed or have a new
; routine you would like to submit to the library
; then mail me at:
;
; Andrew@obelisk.demon.co.uk

      PAGE
;------------------------------------------------
; Basic Operations
;------------------------------------------------

; Clear 2 bytes of memory at any location by
; setting it to zero. If 65SC02 instructions are
; available then STZ is used.
;
; On exit: A = ??, X & Y are unchanged.

      IF __65SC02__
_CLR16       MACRO MEM
       STZ MEM+0
       STZ MEM+1
       ENDM
      ELSE
_CLR16       MACRO MEM
       LDA #0
       STA MEM+0
       STA MEM+1
       ENDM
      ENDIF

; Clear 4 bytes of memory at any location by
; setting it to zero. If 65SC02 instructions are
; available then STZ is used.
;
```

```
; On exit: A = ??, X & Y are unchanged.

      IF __65SC02__
_CLR32       MACRO MEM
       STZ MEM+0
       STZ MEM+1
       STZ MEM+2
       STZ MEM+3
       ENDM
      ELSE
_CLR32       MACRO MEM
       LDA #0
       STA MEM+0
       STA MEM+1
       STA MEM+2
       STA MEM+3
       ENDM
      ENDIF

; Clear 4 bytes of memory at any location by
; setting it to zero iteratively. If 65SC02
; instructions are available then STZ is used.
;
; On exit: A = ??, X = $FF, Y is unchanged.

      IF __65SC02__
_CLR32C       MACRO MEM
       LDX #3
_LOOP\?       STZ MEM,X
       DEX
       BPL _LOOP\?
       ENDM
      ELSE
_CLR32C       MACRO MEM
       LDA #0
       LDX #3
_LOOP\?       STA MEM,X
       DEX
       BPL _LOOP\?
       ENDM
      ENDIF

; Transfer 2 bytes of memory from one location to
; another using the accumulator. The order in
; which the bytes are moved depends on the
; relative positions of SRC and DST. If SRC and
; DST are the same then no code is generated.
;
; On exit: A = ??, X & Y are unchanged.

_XFR16      MACRO SRC,DST
      IF SRC != DST
       IF SRC > DST
        LDA SRC+0
        STA DST+0
        LDA SRC+1
        STA DST+1
       ELSE
        LDA SRC+1
```

```
        STA DST+1
        LDA SRC+0
        STA DST+0
       ENDIF
      ENDIF
      ENDM

; Transfer 4 bytes of memory from one location to
; another using the accumulator. The order in
; which the bytes are moved depends on the
; relative positions of SRC and DST. If SRC and
; DST are the same then no code is generated.
;
; On exit: A = ??, X & Y are unchanged.

_XFR32      MACRO SRC,DST
      IF SRC != DST
       IF SRC > DST
        LDA SRC+0
        STA DST+0
        LDA SRC+1
        STA DST+1
        LDA SRC+2
        STA DST+2
        LDA SRC+3
        STA DST+3
       ELSE
        LDA SRC+3
        STA DST+3
        LDA SRC+2
        STA DST+2
        LDA SRC+1
        STA DST+1
        LDA SRC+0
        STA DST+0
       ENDIF
      ENDIF
      ENDM

; Transfer 4 bytes of memory from one location to
; another iteratively using the accumulator. The
; transfer may fail if SRC and DST overlap. If
; SRC and DST are the same then no code is
; generated.
;
; On exit: A = ??, X = $FF, Y is unchanged.

_XFR32C      MACRO SRC,DST
      IF SRC != DST
       LDX #3
_LOOP\?       LDA SRC,X
       STA DST,X
       DEX
       BPL _LOOP\?
      ENDIF
      ENDM

; Set the value of a 16 bit location DST with
; the given constant value.
```

```
;
; On exit: A = ??, X & Y unchanged.

_SET16I      MACRO NUM,DST
      IF NUM != 0
       LDA #LO NUM
       STA DST+0
       LDA #HI NUM
       STA DST+1
      ELSE
       _CLR16 DST
      ENDIF

      PAGE
;-----------------------------------------------
; Logical Operations
;-----------------------------------------------

; Calculate the logical NOT of the 16 bit value
; at location VLA and stores it in location RES.
;
; On exit: A = ??, X & Y are unchanged.

_NOT16      MACRO VLA,RES
      LDA VLA+0
      EOR #$FF
      STA RES+0
      LDA VLA+1
      EOR #$FF
      STA RES+1
      ENDM

; Calculate the logical NOT of the 32 bit value
; at location VLA and stores it in location RES.
;
; On exit: A = ??, X & Y are unchanged.

_NOT32      MACRO VLA,RES
      LDA VLA+0
      EOR #$FF
      STA RES+0
      LDA VLA+1
      EOR #$FF
      STA RES+1
      LDA VLA+2
      EOR #$FF
      STA RES+2
      LDA VLA+3
      EOR #$FF
      STA RES+3
      ENDM

; Calculate the logical NOT of the 32 bit value
; at location VLA iteratively and stores it in
; location RES.
;
; On exit: A = ??, X = $FF, Y are unchanged.

_NOT32C      MACRO VLA,RES
```

```
      LDX #3
_LOOP\?      LDA VLA,X
      EOR #$FF
      STA RES,X
      DEX
      BPL _LOOP\?
      ENDM

; Calculate the logical OR of the two 16 bit
; values at locations VLA and VLB. The result is
; stored in location RES. If VLA and VLB are the
; same the macro expands to a _XFR16.
;
; On exit: A = ??, X & Y are unchanged.

_ORA16      MACRO VLA,VLB,RES
      IF VLA != VLB
       LDA VLA+0
       ORA VLB+0
       STA RES+0
       LDA VLA+1
       ORA VLB+1
       STA RES+1
      ELSE
       _XFR16 VLA,RES
      ENDIF
      ENDM

; Calculate the logical OR of a 16 value at
; location VLA with a constant value and
; store the result at location RES.
;
; On exit: A = ??, X & Y are unchanged.

_ORA16I      MACRO VLA,NUM,RES
       LDA VLA+0
       ORA #LO NUM
       STA RES+0
       LDA VLA+1
       ORA #HI NUM
       STA RES+1
      ENDM

; Calculate the logical OR of the two 32 bit
; values at locations VLA and VLB. The result is
; stored in location RES. If VLA and VLB are the
; same the macro expands to a _XFR32.
;
; On exit: A = ??, X & Y are unchanged.

_ORA32      MACRO VLA,VLB,RES
      IF VLA != VLB
       LDA VLA+0
       ORA VLB+0
       STA RES+0
       LDA VLA+1
       ORA VLB+1
       STA RES+1
       LDA VLA+2
```

```
       ORA VLB+2
       STA RES+2
       LDA VLA+3
       ORA VLB+3
       STA RES+3
      ELSE
       _XFR32 VLA,RES
      ENDIF
      ENDM

; Calculate the logical OR of the two 32 bit
; values at locations VLA and VLB iteratively.
; The result is stored in location RES. If VLA
; and VLB are the same the macro expands to a
; _XFR32C.
;
; On exit: A = ??, X = $FF, Y is unchanged.

_ORA32C      MACRO VLA,VLB,RES
      IF VLA != VLB
       LDX #3
_LOOP\?       LDA VLA,X
       ORA VLB,X
       STA RES,X
       DEX
       BPL _LOOP\?
      ELSE
       _XFR32C VLA,RES
      ENDIF
      ENDM

; Calculate the logical AND of the two 16 bit
; values at locations VLA and VLB. The result is
; stored in location RES. If VLA and VLB are the
; same the macro expands to a _XFR16.
;
; On exit: A = ??, X & Y are unchanged.

_AND16      MACRO VLA,VLB,RES
      IF VLA != VLB
       LDA VLA+0
       AND VLB+0
       STA RES+0
       LDA VLA+1
       AND VLB+1
       STA RES+1
      ELSE
       _XFR16 VLA,RES
      ENDIF
      ENDM

; Calculate the logical AND of a 16 value at
; location VLA with a constant value and
; store the result at location RES.
;
; On exit: A = ??, X & Y are unchanged.

_AND16I      MACRO VLA,NUM,RES
       LDA VLA+0
```

```
       AND #LO NUM
       STA RES+0
       LDA VLA+1
       AND #HI NUM
       STA RES+1
      ENDM

; Calculate the logical AND of the two 32 bit
; values at locations VLA and VLB. The result is
; stored in location RES. If VLA and VLB are the
; same the macro expands to a _XFR32.
;
; On exit: A = ??, X & Y are unchanged.

_AND32      MACRO VLA,VLB,RES
      IF VLA != VLB
       LDA VLA+0
       AND VLB+0
       STA RES+0
       LDA VLA+1
       AND VLB+1
       STA RES+1
       LDA VLA+2
       AND VLB+2
       STA RES+2
       LDA VLA+3
       AND VLB+3
       STA RES+3
      ELSE
       _XFR32 VLA,RES
      ENDIF
      ENDM

; Calculate the logical AND of the two 32 bit
; values at locations VLA and VLB iteratively.
; The result is stored in location RES. If VLA
; and VLB are the same the macro expands to a
; _XFR32C.
;
; On exit: A = ??, X = $FF, Y is unchanged.

_AND32C      MACRO VLA,VLB,RES
      IF VLA != VLB
       LDX #3
_LOOP\?       LDA VLA,X
       AND VLB,X
       STA RES,X
       DEX
       BPL _LOOP\?
      ELSE
       _XFR32C VLA,RES
      ENDIF
      ENDM

; Calculate the exclusive OR of the two 16 bit
; values at locations VLA and VLB. The result is
; stored in location RES. If VLA and VLB are the
; same the macro expands to a _CLR16.
;
```

```
; On exit: A = ??, X & Y are unchanged.

_EOR16      MACRO VLA,VLB,RES
      IF VLA != VLB
       LDA VLA+0
       EOR VLB+0
       STA RES+0
       LDA VLA+1
       EOR VLB+1
       STA RES+1
      ELSE
       _CLR16 RES
      ENDIF
      ENDM

; Calculate the exclusive OR of a 16 value at
; location VLA with a constant value and
; store the result at location RES.
;
; On exit: A = ??, X & Y are unchanged.

_EOR16I      MACRO VLA,NUM,RES
       LDA VLA+0
       EOR #LO NUM
       STA RES+0
       LDA VLA+1
       EOR #HI NUM
       STA RES+1
      ENDM

; Calculate the exclusive OR of the two 32 bit
; values at locations VLA and VLB. The result is
; stored in location RES. If VLA and VLB are the
; same the macro expands to a _CLR32.
;
; On exit: A = ??, X & Y are unchanged.

_EOR32      MACRO VLA,VLB,RES
      IF VLA != VLB
       LDA VLA+0
       EOR VLB+0
       STA RES+0
       LDA VLA+1
       EOR VLB+1
       STA RES+1
       LDA VLA+2
       EOR VLB+2
       STA RES+2
       LDA VLA+3
       EOR VLB+3
       STA RES+3
      ELSE
       _CLR32 RES
      ENDIF
      ENDM

; Calculate the exclusive OR of the two 32 bit
; values at locations VLA and VLB iteratively.
; The result is stored in location RES. If VLA
```

```
; and VLB are the same the macro expands to a
; _XFR32C.
;
; On exit: A = ??, X = $FF, Y is unchanged.

_EOR32C      MACRO VLA,VLB,RES
      IF VLA != VLB
       LDX #3
_LOOP\?       LDA VLA,X
       EOR VLB,X
       STA RES,X
       DEX
       BPL _LOOP\?
      ELSE
       _CLR32C RES
      ENDIF
      ENDM

      PAGE
;------------------------------------------------
; Shift Operations
;------------------------------------------------

; Perform an arithmetic shift left on the 16 bit
; number at location VLA and store the result at
; location RES. If VLA and RES are the same then
; the operation is applied directly to the memory
; otherwise it is done in the accumulator.
;
; On exit: A = ??, X & Y are unchanged.

_ASL16      MACRO VLA,RES
      IF VLA != RES
       LDA VLA+0
       ASL A
       STA RES+0
       LDA VLA+1
       ROL A
       STA RES+1
      ELSE
       ASL VLA+0
       ROL VLA+1
      ENDIF
      ENDM

; Perform an arithmetic shift left on the 32 bit
; number at location VLA and store the result at
; location RES. If VLA and RES are the same then
; the operation is applied directly to the memory
; otherwise it is done in the accumulator.
;
; On exit: A = ??, X & Y are unchanged.

_ASL32      MACRO VLA,RES
      IF VLA != RES
       LDA VLA+0
       ASL A
       STA RES+0
       LDA VLA+1
```

```
        ROL A
        STA RES+1
        LDA VLA+2
        ROL A
        STA RES+2
        LDA VLA+3
        ROL A
        STA RES+3
       ELSE
        ASL VLA+0
        ROL VLA+1
        ROL VLA+2
        ROL VLA+3
       ENDIF
       ENDM

; Perform a left rotation on the 16 bit number at
; location VLA and store the result at location
; RES. If VLA and RES are the same then the
; operation is applied directly to the memory,
; otherwise it is done in the accumulator.
;
; On exit: A = ??, X & Y are unchanged.

_ROL16      MACRO VLA,RES
       IF VLA != RES
        LDA VLA+0
        ROL A
        STA RES+0
        LDA VLA+1
        ROL A
        STA RES+1
       ELSE
        ROL VLA+0
        ROL VLA+1
       ENDIF
       ENDM

; Perform a left rotation on the 32 bit number at
; location VLA and store the result at location
; RES. If VLA and RES are the same then the
; operation is applied directly to the memory,
; otherwise it is done in the accumulator.
;
; On exit: A = ??, X & Y are unchanged.

_ROL32      MACRO VLA,RES
       IF VLA != RES
        LDA VLA+0
        ROL A
        STA RES+0
        LDA VLA+1
        ROL A
        STA RES+1
        LDA VLA+2
        ROL A
        STA RES+2
        LDA VLA+3
        ROL A
```

```
       STA RES+3
      ELSE
       ROL VLA+0
       ROL VLA+1
       ROL VLA+2
       ROL VLA+3
      ENDIF
      ENDM

; Perform an logical shift right on the 16 bit
; number at location VLA and store the result at
; location RES. If VLA and RES are the same then
; the operation is applied directly to the memory
; otherwise it is done in the accumulator.
;
; On exit: A = ??, X & Y are unchanged.

_LSR16      MACRO VLA,RES
      IF VLA != RES
       LDA VLA+1
       LSR A
       STA RES+1
       LDA VLA+0
       ROR A
       STA RES+0
      ELSE
       LSR VLA+1
       ROR VLA+0
      ENDIF
      ENDM

; Perform an logical shift right on the 32 bit
; number at location VLA and store the result at
; location RES. If VLA and RES are the same then
; the operation is applied directly to the memory
; otherwise it is done in the accumulator.
;
; On exit: A = ??, X & Y are unchanged.

_LSR32      MACRO VLA,RES
      IF VLA != RES
       LDA VLA+3
       LSR A
       STA RES+3
       LDA VLA+2
       ROR A
       STA RES+2
       LDA VLA+1
       ROR A
       STA RES+1
       LDA VLA+0
       ROR A
       STA RES+0
      ELSE
       LSR VLA+3
       ROR VLA+2
       ROR VLA+1
       ROR VLA+0
      ENDIF
```

```
      ENDM

; Perform a right rotation on the 16 bit number
; at location VLA and store the result at
; location RES. If VLA and RES are the same then
; the operation is applied directly to the memory
; otherwise it is done in the accumulator.
;
; On exit: A = ??, X & Y are unchanged.

_ROR16      MACRO VLA,RES
      IF VLA != RES
       LDA VLA+1
       ROR A
       STA RES+1
       LDA VLA+0
       ROR A
       STA RES+0
      ELSE
       ROR VLA+1
       ROR VLA+0
      ENDIF
      ENDM

; Perform a right rotation on the 32 bit number
; at location VLA and store the result at
; location RES. If VLA and RES are the same then
; the operation is applied directly to the memory
; otherwise it is done in the accumulator.
;
; On exit: A = ??, X & Y are unchanged.

_ROR32      MACRO VLA,RES
      IF VLA != RES
       LDA VLA+3
       ROR A
       STA RES+3
       LDA VLA+2
       ROR A
       STA RES+2
       LDA VLA+1
       ROR A
       STA RES+1
       LDA VLA+0
       ROR A
       STA RES+0
      ELSE
       ROR VLA+3
       ROR VLA+2
       ROR VLA+1
       ROR VLA+0
      ENDIF
      ENDM

      PAGE
;------------------------------------------------
; Arithmetic Operations
;------------------------------------------------
```

```
; Increment the 16 bit value at location MEM
; by one.
;
; On exit: A, X & Y are unchanged.

_INC16      MACRO MEM
      INC MEM+0
      BNE _DONE\?
      INC MEM+1
_DONE\?      EQU *
      ENDM

; Increment the 32 bit value at location MEM
; by one.
;
; On exit: A, X & Y are unchanged.

_INC32      MACRO MEM
      INC MEM+0
      BNE _DONE\?
      INC MEM+1
      BNE _DONE\?
      INC MEM+2
      BNE _DONE\?
      INC MEM+3
_DONE\?      EQU *
      ENDM

; Decrement the 16 bit value at location MEM
; by one.
;
; On exit: A = ??, X & Y are unchanged.

_DEC16      MACRO MEM
      LDA MEM+0
      BNE _DONE\?
      DEC MEM+1
_DONE\?      DEC MEM+0
      ENDM

; Decrement the 32 bit value at location MEM
; by one.
;
; On exit: A = ??, X & Y are unchanged.

_DEC32      MACRO MEM
      LDA MEM+0
      BNE _DEC0\?
      LDA MEM+1
      BNE _DEC1\?
      LDA MEM+2
      BNE _DEC2\?
      DEC MEM+3
_DEC2\?      DEC MEM+2
_DEC1\?      DEC MEM+1
_DEC0\?      DEC MEM+0
      ENDM

; Add two 16 bit numbers together and store the
```

```
; result in another memory location. RES may be
; the same as either VLA or VLB.
;
; On exit: A = ??, X & Y are unchanged.

_ADD16      MACRO VLA,VLB,RES
      IF VLA != VLB
        CLC
        LDA VLA+0
       ADC VLB+0
       STA RES+0
       LDA VLA+1
       ADC VLB+1
       STA RES+1
      ELSE
       _ASL16 VLA,RES
      ENDIF
      ENDM

; Add two 32 bit numbers together and store the
; result in another memory location. RES may be
; the same as either VLA or VLB.
;
; On exit: A = ??, X & Y are unchanged.

_ADD32      MACRO VLA,VLB,RES
      IF VLA != VLB
       CLC
       LDA VLA+0
       ADC VLB+0
       STA RES+0
       LDA VLA+1
       ADC VLB+1
       STA RES+1
       LDA VLA+2
       ADC VLB+2
       STA RES+2
       LDA VLA+3
       ADC VLB+3
       STA RES+3
      ELSE
       _ASL32 VLA,RES
      ENDIF
      ENDM

; Subtract two 16 bit numbers and store the
; result in another memory location. RES may be
; the same as VLA or VLB.
;
; On exit: A = ??, X & Y are unchanged.

_SUB16      MACRO VLA,VLB,RES
      SEC
      LDA VLA+0
      SBC VLB+0
      STA RES+0
      LDA VLA+1
      SBC VLB+1
      STA RES+1
```

```
      ENDM

; Subtract two 32 bit numbers and store the
; result in another memory location. RES may be
; the same as VLA or VLB.
;
; On exit: A = ??, X & Y are unchanged.

_SUB32      MACRO VLA,VLB,RES
      SEC
      LDA VLA+0
      SBC VLB+0
      STA RES+0
      LDA VLA+1
      SBC VLB+1
      STA RES+1
      LDA VLA+2
      SBC VLB+2
      STA RES+2
      LDA VLA+3
      SBC VLB+3
      STA RES+3
      ENDM

; Negate the signed 16 bit number at location VLA
; and stored the result at location RES. RES may
; be the same as VLA.
;
; On exit: A = ??, X & Y are unchanged.

_NEG16      MACRO VLA,RES
      SEC
      LDA #0
      SBC VLA+0
      STA RES+0
      LDA #0
      SBC VLA+1
      STA RES+1
      ENDM

; Negate the signed 32 bit number at location VLA
; and stored the result at location RES. RES may
; be the same as VLA.
;
; On exit: A = ??, X & Y are unchanged.

_NEG32      MACRO VLA,RES
      SEC
      LDA #0
      SBC VLA+0
      STA RES+0
      LDA #0
      SBC VLA+1
      STA RES+1
      LDA #0
      SBC VLA+2
      STA RES+2
      LDA #0
      SBC VLA+3
```

```
      STA RES+3
      ENDM

; Calculates the absolute value of signed 16 bit
; number at location VLA and stores it in the RES
; location. Less code is generated if VLA and RES
; are the same location. If 65SC02 instructions
; are available a BRA is used to shorten the
; generated code.
;
; On exit: A = ??, X & Y are unchanged.

_ABS16      MACRO VLA,RES
      BIT VLA+0
      IF VLA != RES
       BPL _MOVE\?
       _NEG16 VLA,RES
       IF __65SC02__
        BRA _DONE\?
       ELSE
        JMP _DONE\?
       ENDIF
_MOVE\?       EQU *
       _XFR16 VLA,RES
      ELSE
       BPL _DONE\?
       _NEG16 VLA,RES
      ENDIF
_DONE\?      EQU *
      ENDM

; Calculates the absolute value of signed 32 bit
; number at location VLA and stores it in the RES
; location. Less code is generated if VLA and RES
; are the same location. If 65SC02 instructions
; are available a BRA is used to shorten the
; generated code.
;
; On exit: A = ??, X & Y are unchanged.

_ABS32      MACRO VLA,RES
      BIT VLA+0
      IF VLA != RES
       BPL _MOVE\?
       _NEG32 VLA,RES
       IF __65SC02__
        BRA _DONE\?
       ELSE
        JMP _DONE\?
       ENDIF
_MOVE\?       EQU *
       _XFR32 VLA,RES
      ELSE
       BPL _DONE\?
       _NEG32 VLA,RES
      ENDIF
_DONE\?      EQU *
      ENDM
```

```
; Calculate the 16 bit product of two 16 bit
; unsigned numbers. Any overflow during the
; calculation is lost. The number at location
; VLA is destroyed.
;
; On exit: A = ??, X = $FF, Y is unchanged.

_MUL16      MACRO VLA,VLB,RES
      _CLR16 RES
      LDX #16
_LOOP\?      EQU *
      _ASL16 RES,RES
      _ASL16 VLA,VLA
      BCC _NEXT\?
      _ADD16 VLB,RES,RES
_NEXT\?      DEX
      BPL _LOOP\?
      ENDM

; Calculate the 32 bit product of two 16 bit
; unsigned numbers. The number at location VLA
; is destroyed.
;
; On exit: A = ??, X = $FF, Y is unchanged.

_MUL16X      MACRO VLA,VLB,RES
      _CLR32 RES
      LDX #16
_LOOP\?      EQU *
      _ASL32 RES,RES
      _ASL16 VLA,VLA
      BCC _NEXT\?
      _ADD16 VLB,RES,RES
      BCC _NEXT\?
      _INC16 RES+2
_NEXT\?      EQU *
      DEX
      BPL _LOOP\?
      ENDM

; Calculate the 32 bit product of two 32 bit
; unsigned numbers. Any overflow during the
; calculation is lost. The number at location
; VLA is destroyed.
;
; On exit: A = ??, X = $FF, Y is unchanged.

_MUL32      MACRO VLA,VLB,RES
      _CLR32 RES
      LDX #32
_LOOP\?      EQU *
      _ASL32 RES,RES
      _ASL32 VLA,VLA
      BCC _NEXT\?
      _ADD32 VLB,RES,RES
_NEXT\?      EQU *
      DEX
      BPL _LOOP\?
      ENDM
```

```
; These two macros write the code necessary
; to multiply a 16 bit at location VLA by
; a 16 bit constant NUM and store the 16 bit
; result in location RES.
;
; On exit: A = ??, X & Y unchanged.

_MUL16I      MACRO VLA,NUM,RES
      IF NUM = 1
       _XFR16 VLA,RES
      ELSE
       _CLR16 RES
       __MUL16I VLA,NUM,RES
      ENDIF
      ENDM

__MUL16I   MACRO VLA,NUM,RES
       IF NUM & $FFFE
        __MUL16I VLA,(NUM/2),RES
        _ASL16 RES,RES
       ENDIF
       IF NUM & $0001
         _ADD16 VLA,RES,RES
       ENDIF
      ENDM

; Divide the 16 bit number at location VLA
; by the 16 bit number at location VLB
; leaving the 16 bit quotient at QUO and
; the 16 bit remainder in REM. The value in
; location VLA is destroyed.
;
; On exit: A = ??, X = $FF, Y is unchanged.

_DIV16      MACRO VLA,VLB,QUO,REM
      _CLR16 REM
      LDX #16
_LOOP\?      EQU *
      _ASL16 VLA,VLA
      _ROL16 REM,REM
      _SUB16 REM,VLB,REM
      BCS _NEXT\?
      _ADD16 REM,VLB,REM
_NEXT\?      EQU *
      _ROL16 QUO,QUO
      DEX
      BPL _LOOP\?
      ENDM

; Divide the 32 bit number at location VLA
; by the 16 bit number at location VLB
; leaving the 16 bit quotient at QUO and
; the 16 bit remainder in REM. The value in
; location VLA is destroyed.
;
; On exit: A = ??, X = $FF, Y is unchanged.

_DIV16X      MACRO VLA,VLB,QUO,REM
```

```
      _CLR16 REM
      LDX #32
_LOOP\?      EQU *
      _ASL32 VLA,VLA
      _ROL16 REM,REM
      _SUB16 REM,VLB,REM
      BCS _NEXT\?
      _ADD16 REM,VLB,REM
_NEXT\?      EQU *
      _ROL16 QUO,QUO
      DEX
      BPL _LOOP\?
      ENDM

; Divide the 32 bit number at location VLA
; by the 32 bit number at location VLB
; leaving the 32 bit quotient at QUO and
; the 32 bit remainder in REM. The value in
; location VLA is destroyed.
;
; On exit: A = ??, X = $FF, Y is unchanged.

_DIV32      MACRO VLA,VLB,QUO,REM
      _CLR32 REM
      LDX #32
_LOOP\?      EQU *
      _ASL32 VLA,VLA
      _ROL32 REM,REM
      _SUB32 REM,VLB,REM
      BCS _NEXT\?
      _ADD32 REM,VLB,REM
_NEXT\?      EQU *
      _ROL32 QUO,QUO
      DEX
      BPL _LOOP\?
      ENDM

      PAGE
;------------------------------------------------
; Comparative Operations
;------------------------------------------------

; Compares two 16 bit values in memory areas VLA
; and VLB. The comparison starts with the most
; significant bytes and returns as soon as a
; difference is detected.
;
; On exit: A = ??, X & Y are unchanged.

_CMP16      MACRO VLA,VLB
      LDA VLA+1
      CMP VLB+1
      BNE _DONE\?
      LDA VLA+0
      CMP VLB+0
_DONE\?      EQU *
      ENDM

; Compares two 32 bit values in memory areas VLA
```

```
; and VLB. The comparison starts with the most
; significant bytes and returns as soon as a
; difference is detected.
;
; On exit: A = ??, X & Y are unchanged.

_CMP32      MACRO VLA,VLB
      LDA VLA+3
      CMP VLB+3
      BNE _DONE\?
      LDA VLA+2
      CMP VLB+2
      BNE _DONE\?
      LDA VLA+1
      CMP VLB+1
      BNE _DONE\?
      LDA VLA+0
      CMP VLB+0
_DONE\?      EQU *
      ENDM

      PAGE
;------------------------------------------------
; Memory Operations
;------------------------------------------------

; Transfers a block of memory from one place to
; another by copying the bytes starting at the
; front of the block and going forward. SRC and
; DST are destroyed during the copy.
;
; On exit: A, X & Y = ??.

_MEMFWD      MACRO  SRC,DST,LEN
      LDY #0
      LDX LEN+1
      BEQ _FRAG\?
_PAGE\?      LDA (SRC),Y
      STA (DST),Y
      INY
      BNE _PAGE\?
      INC SRC+1
      INC DST+1
      DEX
      BNE _PAGE\?
_FRAG\?      CPY LEN+0
      BEQ _DONE\?
      LDA (SRC),Y
      STA (DST),Y
      INY
      BNE _FRAG\?
_DONE\?      EQU *
      ENDM

; Transfers a block of memory from one place to
; another by copying the bytes starting at the
; end of the block and going backwards.

_MEMREV      MACRO  SRC,DST,LEN
```

```
      NOP
      ENDM

; Tranfers a block of memory from one location to
; another. Depending on the relative positions of
; the blocks an appropriate transfer method is
; used.

_MEMCPY      MACRO  SRC,DST,LEN
      _CMP16 SRC,DST
      BCC _SAFE\?
      _MEMFWD SRC,DST,LEN
      IF __65SC02__
       BRA _DONE\?
      ELSE
       JMP _DONE\?
      ENDIF
_SAFE\?      _MEMREV SRC,DST,LEN
_DONE\?      EQU *
      ENDM

      PAGE
;------------------------------------------------
; String Operations
;------------------------------------------------

; Calculates length of a null terminated string
; by searching for its end. The address of the
; string in STR is destroyed during the search.
;
; On exit: A & Y = ??, X is unchanged.

_STRLEN      MACRO  STR,LEN
      LDY #0
      STY LEN+1
_LOOP\?      LDA (STR),Y
      BEQ _DONE\?
      INY
      BNE _LOOP\?
      INC LEN+1
      INC STR+1
      IF __65SC02__
       BRA _LOOP\?
      ELSE
       JMP _LOOP\?
      ENDIF
_DONE\?      STY LEN+0
      ENDM

; Copies a null terminated string from one memory
; location to another. The source and destination
; addresses are destroyed during the copy process.
;
; On exit: A & Y = ??, X is unchanged.

_STRCPY      MACRO SRC,DST
      LDY #0
_LOOP\?      LDA (SRC),Y
      STA (DST),Y
```

```
      BEQ _DONE\?
      INY
      BNE _LOOP\?
      INC SRC+1
      INC DST+1
      IF __65SC02__
       BRA _LOOP\?
      ELSE
       JMP _LOOP\?
      ENDIF
_DONE\?      EQU *
      ENDM

;

_STRCMP      MACRO VLA,VLB
      LDY #0
_LOOP\?      LDA (VLA),Y
      CMP (VLB),Y
      BNE _DONE\?
      LDA (VLA),Y
      BEQ _DONE\?
      INY
      BNE _LOOP\?
      INC VLA+1
      INC VLB+1
      IF __65SC02__
       BRA _LOOP\?
      ELSE
       JMP _LOOP\?
      ENDIF
_DONE\?      EQU *
      ENDM

;

_STRNCMP   MACRO VLA,VLB,LEN
      ENDM

      LIST
```