Macro Library Resource Study

Peter Camilleri
Version 1.0.0
January 18, 2020

This spreadsheet contains the results on a study of the 13 macro files in the CX16 repository.

For each macro, the affects of addressing modes, constants, and data values modify how the macros generate assembly code.

For each condition/path, this is expressed in terms of clock cycles (θ) and bytes (B) used.

adj_16 - zero page

	A		**B**		**C**		**D**		**E**		**F**	
.macro _adj_var_16 var,step	θ	B	θ	B	θ	B	θ	B	θ	B	θ	B
.ifconst step												
.if step												
clc			2	1	2	1	2	1	2	1		
.endif												
.if .lobyte(step)												
lda var					3	2	3	2	3	2		
adc #.lobyte(step)					2	2	2	2	2	2		
sta var					3	2	3	2	3	2		
.endif												
.if .hibyte(step)												
lda var+1			3	2					3	2		
adc #.hibyte(step)			2	2					2	2		
sta var+1			3	2					3	2		
.elseif .lobyte(step)												
bcc no_carry					3	2	2	2				
inc var+1					0	2	4	2				
.endif												
.else												
clc											2	1
lda var											3	2
adc #.lobyte(step)											2	2
sta var											3	2
lda var+1											3	2
adc #.hibyte(step)											2	2
sta var+1											3	2
.endif												
no_carry:												
.endmacro												
	0	**0**	**10**	**7**	**13**	**11**	**16**	**11**	**18**	**13**	**18**	**13**

A step is const 0
B step is const $xx00
C step is const $00xx, no carry
D step is const $00xx, with carry
E step is const
F step is not const

adj_16 – absolute

	A		B		C		D		E		F	
.macro _adj_var_16 var,step	θ	B	θ	B	θ	B	θ	B	θ	B	θ	B
.ifconst step												
.if step												
clc			2	1	2	1	2	1	2	1		
.endif												
.if .lobyte(step)												
lda var					4	3	4	3	4	3		
adc #.lobyte(step)					2	2	2	2	2	2		
sta var					4	3	4	3	4	3		
.endif												
.if .hibyte(step)												
lda var+1			4	3					4	3		
adc #.hibyte(step)			2	2					2	2		
sta var+1			4	3					4	3		
.elseif .lobyte(step)												
bcc no_carry					3	2	2	2				
inc var+1					0	3	6	3				
.endif												
.else												
clc											2	1
lda var											4	3
adc #.lobyte(step)											2	2
sta var											4	3
lda var+1											4	3
adc #.hibyte(step)											2	2
sta var+1											4	3
.endif												
no_carry:												
.endmacro												
	0	**0**	**12**	**9**	**15**	**14**	**20**	**14**	**22**	**17**	**22**	**17**

A step is const 0
B step is const $xx00
C step is const $00xx, no carry
D step is const $00xx, with carry
E step is const
F step is not const

adj_16 - zero page,x

	A		**B**		**C**		**D**		**E**		**F**	
.macro _adj_var_16 var,step	θ	B	θ	B	θ	B	θ	B	θ	B	θ	B
.ifconst step												
.if step												
clc			2	1	2	1	2	1	2	1		
.endif												
.if .lobyte(step)												
lda var,x					4	2	4	2	4	2		
adc #.lobyte(step)					2	2	2	2	2	2		
sta var,x					4	2	4	2	4	2		
.endif												
.if .hibyte(step)												
lda var+1,x			4	2					3	2		
adc #.hibyte(step)			2	2					2	2		
sta var+1,x			4	2					3	2		
.elseif .lobyte(step)												
bcc no_carry					3	2	2	2				
inc var+1,x					0	2	6	2				
.endif												
.else												
clc											2	1
lda var,x											4	2
adc #.lobyte(step)											2	2
sta var,x											4	2
lda var+1,x											4	2
adc #.hibyte(step)											2	2
sta var+1,x											4	2
.endif												
no_carry:												
.endmacro												
	0	**0**	**12**	**7**	**15**	**11**	**20**	**11**	**20**	**13**	**22**	**13**

A step is const 0
B step is const $xx00
C step is const $00xx, no carry
D step is const $00xx, with carry
E step is const
F step is not const

adj_16 - absolute,x (or y) – no page cross

	A		**B**		**C**		**D**		**E**		**F**	
.macro _adj_var_16 var,step	θ	B	θ	B	θ	B	θ	B	θ	B	θ	B
.ifconst step												
.if step												
clc			2	1	2	1	2	1	2	1		
.endif												
.if .lobyte(step)												
lda var,x					4	3	4	3	4	3		
adc #.lobyte(step)					2	2	2	2	2	2		
sta var,x					4	3	4	3	4	3		
.endif												
.if .hibyte(step)												
lda var+1,x			4	3					4	3		
adc #.hibyte(step)			2	2					2	2		
sta var+1,x			4	3					4	3		
.elseif .lobyte(step)												
bcc no_carry					3	2	2	2				
inc var+1,x					0	3	6	3				
.endif												
.else												
clc											2	1
lda var,x											4	3
adc #.lobyte(step)											2	2
sta var,x											4	3
lda var+1,x											4	3
adc #.hibyte(step)											2	2
sta var+1,x											4	3
.endif												
no_carry:												
.endmacro												
	0	**0**	**12**	**9**	**15**	**14**	**20**	**14**	**22**	**17**	**22**	**17**

A step is const 0
B step is const $xx00
C step is const $00xx, no carry
D step is const $00xx, with carry
E step is const
F step is not const

adj_16 - absolute,x (or y) with page cross

	A		**B**		**C**		**D**		**E**		**F**	
.macro _adj_var_16 var,step	θ	B	θ	B	θ	B	θ	B	θ	B	θ	B
.ifconst step												
.if step												
clc			2	1	2	1	2	1	2	1		
.endif												
.if .lobyte(step)												
lda var,x					5	3	5	3	5	3		
adc #.lobyte(step)					2	2	2	2	2	2		
sta var,x					5	3	5	3	5	3		
.endif												
.if .hibyte(step)												
lda var+1,x			5	3					5	3		
adc #.hibyte(step)			2	2					2	2		
sta var+1,x			5	3					5	3		
.elseif .lobyte(step)												
bcc no_carry					3	2	2	2				
inc var+1,x					0	3	7	3				
.endif												
.else												
clc											2	1
lda var,x											5	3
adc #.lobyte(step)											2	2
sta var,x											5	3
lda var+1,x											5	3
adc #.hibyte(step)											2	2
sta var+1,x											5	3
.endif												
no_carry:												
.endmacro												
	0	**0**	**14**	**9**	**17**	**14**	**23**	**14**	**26**	**17**	**26**	**17**

A step is const 0
B step is const $xx00
C step is const $00xx, no carry
D step is const $00xx, with carry
E step is const
F step is not const

adj_16 - zero page indirect

	A		B		C		D	
.macro _adj_var_16 var,step	θ	B	θ	B	θ	B	θ	B
.ifconst step								
.if step								
clc			2	1	2	1		
.endif								
.if .lobyte(step)								
lda (var)					5	2		
adc #.lobyte(step)					2	2		
sta (var)					5	2		
.endif								
.if step								
ldy #1			2	2	2	2		
lda (var),y			5	2	5	2		
adc #.hibyte(step)			2	2	2	2		
sta (var),y			5	2	5	2		
.endif								
.else								
clc							2	1
lda (var)							5	2
adc #.lobyte(step)							2	2
sta (var)							5	2
ldy #1							2	2
lda (var),y							5	2
adc #.hibyte(step)							2	2
sta (va),y							5	2
.endif								
no_carry:								
.endmacro								
	0	**0**	**16**	**9**	**28**	**15**	**28**	**15**

A step is const 0
B step is const $xx00
C step is const
D step is not const

adj_16 - zero page indirect,y

	A		B		C		D	
.macro _adj_var_16 var,step	θ	B	θ	B	θ	B	θ	B
.ifconst step								
.if step								
clc			2	1	2	1		
.endif								
.if .lobyte(step)								
lda (var),y					5	2		
adc #.lobyte(step)					2	2		
sta (var),y					5	2		
.endif								
.if step								
iny			2	1	2	1		
lda (var),y			5	2	5	2		
adc #.hibyte(step)			2	2	2	2		
sta (var),y			5	2	5	2		
dey			2	1	2	1		
.endif								
.else								
clc							2	1
lda (var)							5	2
adc #.lobyte(step)							2	2
sta (var)							5	2
iny							2	1
lda (var),y							5	2
adc #.hibyte(step)							2	2
sta (va),y							5	2
dey							2	1
.endif								
no_carry:								
.endmacro								
	0	0	18	9	30	15	30	15

A step is const 0
B step is const $xx00
C step is const
D step is not const

cmp_16 - zero page

	A		B		C		D		E		F	
.macro _cmp_var_16 var, value	θ	B	θ	B	θ	B	θ	B	θ	B	θ	B
sec	2	1	2	1	2	1	2	1	2	1	2	1
lda var	3	2	3	2	3	2	3	2	3	2	3	2
.if (!.const(value))||.lobyte(value)												
sbc #.lobyte(value)							2	2	2	2	2	2
.endif												
bne not_equal	2	2	3	2	3	2	2	2	3	2	2	2
lda var+1	3	2	0	2	0	2	3	2	0	2	0	2
sbc #.hibyte(value)	2	2	0	2	0	2	2	2	0	2	0	2
bra all_done	3	2	0	2	0	2	3	2	0	2	0	2
not_equal:												
lda var+1	0	2	3	2	3	2	0	2	3	2	3	2
sbc #.hibyte(value)	0	2	2	2	2	2	0	2	2	2	2	2
bne all_done	0	2	3	2	2	2	0	2	3	2	2	2
inc	0	1	0	1	2	1	0	1	0	1	2	1
all_done:												
.endmacro												
	15	**18**	**16**	**18**	**17**	**18**	**17**	**20**	**18**	**20**	**18**	**20**

cmp_16 – absolute

	A		B		C		D		E		F	
.macro _cmp_var_16 var, value	θ	B	θ	B	θ	B	θ	B	θ	B	θ	B
sec	2	1	2	1	2	1	2	1	2	1	2	1
lda var	4	3	4	3	4	3	4	3	4	3	4	3
.if (!.const(value))||.lobyte(value)												
sbc #.lobyte(value)							2	2	2	2	2	2
.endif												
bne not_equal	2	2	3	2	3	2	2	2	3	2	2	2
lda var+1	4	3	0	3	0	3	4	3	0	3	0	3
sbc #.hibyte(value)	2	2	0	2	0	2	2	2	0	2	0	2
bra all_done	3	2	0	2	0	2	3	2	0	2	0	2
not_equal:												
lda var+1	0	3	4	3	4	3	0	3	4	3	4	3
sbc #.hibyte(value)	0	2	2	2	2	2	0	2	2	2	2	2
bne all_done	0	2	3	2	2	2	0	2	3	2	2	2
inc	0	1	0	1	2	1	0	1	0	1	2	1
all_done:												
.endmacro												
	17	**21**	**18**	**21**	**19**	**21**	**19**	**23**	**20**	**23**	**20**	**23**

A value is const $xx00, low bytes equal
B value is const $xx00, low bytes not equal, high not zero
C value is const $xx00, low bytes not equal, high zero
D value is not const or not $xx00, low bytes equal
E value is not const or not $xx00, low bytes not equal, high not zero
F value is not const or not $xx00, low bytes not equal, high zero

cmp_16 - zero page,x

	A		**B**		**C**		**D**		**E**		**F**	
.macro _cmp_var_16 var, value	θ	B	θ	B	θ	B	θ	B	θ	B	θ	B
sec	2	1	2	1	2	1	2	1	2	1	2	1
lda var,x	4	2	4	2	4	2	4	2	4	2	4	2
.if (!.const(value))||.lobyte(value)												
sbc #.lobyte(value)							2	2	2	2	2	2
.endif												
bne not_equal	2	2	3	2	3	2	2	2	3	2	2	2
lda var+1,x	4	2	0	2	0	2	4	2	0	2	0	2
sbc #.hibyte(value)	2	2	0	2	0	2	2	2	0	2	0	2
bra all_done	3	2	0	2	0	2	3	2	0	2	0	2
not_equal:												
lda var+1,x	0	2	4	2	4	2	0	2	4	2	4	2
sbc #.hibyte(value)	0	2	2	2	2	2	0	2	2	2	2	2
bne all_done	0	2	3	2	2	2	0	2	3	2	2	2
inc	0	1	0	1	2	1	0	1	0	1	2	1
all_done:												
.endmacro												
	17	**18**	**18**	**18**	**19**	**18**	**19**	**20**	**20**	**20**	**20**	**20**

cmp_16 – absolute,x (or y)

	A		**B**		**C**		**D**		**E**		**F**	
.macro _cmp_var_16 var, value	θ	B	θ	B	θ	B	θ	B	θ	B	θ	B
sec	2	1	2	1	2	1	2	1	2	1	2	1
lda var,x	4	3	4	3	4	3	4	3	4	3	4	3
.if (!.const(value))||.lobyte(value)												
sbc #.lobyte(value)							2	2	2	2	2	2
.endif												
bne not_equal	2	2	3	2	3	2	2	2	3	2	2	2
lda var+1,x	4	3	0	3	0	3	4	3	0	3	0	3
sbc #.hibyte(value)	2	2	0	2	0	2	2	2	0	2	0	2
bra all_done	3	2	0	2	0	2	3	2	0	2	0	2
not_equal:												
lda var+1,x	0	3	4	3	4	3	0	3	4	3	4	3
sbc #.hibyte(value)	0	2	2	2	2	2	0	2	2	2	2	2
bne all_done	0	2	3	2	2	2	0	2	3	2	2	2
inc	0	1	0	1	2	1	0	1	0	1	2	1
all_done:												
.endmacro												
	17	**21**	**18**	**21**	**19**	**21**	**19**	**23**	**20**	**23**	**20**	**23**

Add two clock cycles if a page boundary is crossed.

A value is const $xx00, low bytes equal
B value is const $xx00, low bytes not equal, high not zero
C value is const $xx00, low bytes not equal, high zero
D value is not const or not $xx00, low bytes equal
E value is not const or not $xx00, low bytes not equal, high not zero
F value is not const or not $xx00, low bytes not equal, high zero

cmp_16 - zero page indirect

	A		B		C		D		E		F	
.macro _cmp_var_16 var, value	θ	B	θ	B	θ	B	θ	B	θ	B	θ	B
sec	2	1	2	1	2	1	2	1	2	1	2	1
ldy #1	2	2	2	2	2	2	2	2	2	2	2	2
lda (var)	5	2	5	2	5	2	5	2	5	2	5	2
.if (!.const(value))||.lobyte(value)												
sbc #.lobyte(value)							2	2	2	2	2	2
.endif												
bne not_equal	2	2	3	2	3	2	2	2	3	2	2	2
lda (var),y	5	2	0	2	0	2	5	2	0	2	0	2
sbc #.hibyte(value)	2	2	0	2	0	2	2	2	0	2	0	2
bra all_done	3	2	0	2	0	2	3	2	0	2	0	2
not_equal:												
lda (var),y	0	2	5	2	5	2	0	2	5	2	5	2
sbc #.hibyte(value)	0	2	2	2	2	2	0	2	2	2	2	2
bne all_done	0	2	3	2	2	2	0	2	3	2	2	2
inc	0	1	0	1	2	1	0	1	0	1	2	1
all_done:												
.endmacro												
	21	**20**	**22**	**20**	**23**	**20**	**23**	**22**	**24**	**22**	**24**	**22**

cmp_16 - zero page indirect,y

	A		B		C		D		E		F	
.macro _cmp_var_16 var, value	θ	B	θ	B	θ	B	θ	B	θ	B	θ	B
sec	2	1	2	1	2	1	2	1	2	1	2	1
lda (var)	5	2	5	2	5	2	5	2	5	2	5	2
.if (!.const(value))||.lobyte(value)												
sbc #.lobyte(value)							2	2	2	2	2	2
.endif												
bne not_equal	2	2	3	2	3	2	2	2	3	2	2	2
iny	2	2	0	2	0	2	2	2	0	2	0	2
lda (var),y	5	2	0	2	0	2	5	2	0	2	0	2
dey	2	2	0	2	0	2	2	2	0	2	0	2
sbc #.hibyte(value)	2	2	0	2	0	2	2	2	0	2	0	2
bra all_done	3	2	0	2	0	2	3	2	0	2	0	2
not_equal:												
iny	0	2	2	2	2	2	0	2	2	2	2	2
lda (var),y	0	2	5	2	5	2	0	2	5	2	5	2
dey	0	2	2	2	2	2	0	2	2	2	2	2
sbc #.hibyte(value)	0	2	2	2	2	2	0	2	2	2	2	2
bne all_done	0	2	3	2	2	2	0	2	3	2	2	2
inc	0	1	0	1	2	1	0	1	0	1	2	1
all_done:												
.endmacro												
	23	**26**	**24**	**26**	**25**	**26**	**25**	**28**	**26**	**28**	**26**	**28**

A value is const $xx00, low bytes equal
B value is const $xx00, low bytes not equal, high not zero
C value is const $xx00, low bytes not equal, high zero
D value is not const or not $xx00, low bytes equal
E value is not const or not $xx00, low bytes not equal, high not zero
F value is not const or not $xx00, low bytes not equal, high zero

dec_16 - zero page

	A		B	
.macro _dec_var_16 var	θ	B	θ	B
lda var	3	2	3	2
bne no_wrap	2	2	3	2
dec var+1	0	2	5	2
no_wrap:				
dec var	5	2	5	2
.endmacro				
	10	**8**	**16**	**8**

dec_16 – absolute

	A		B	
.macro _dec_var_16 var	θ	B	θ	B
lda var	3	3	3	3
bne no_wrap	2	2	3	2
dec var+1	0	3	6	3
no_wrap:				
dec var	6	3	6	3
.endmacro				
	11	**11**	**18**	**11**

dec_16 - zero page,x

	A		B	
.macro _dec_vax_16 var	θ	B	θ	B
lda var,x	4	2	4	2
bne no_wrap	2	2	3	2
dec var+1,x	0	2	6	2
no_wrap:				
dec var,x	6	2	6	2
.endmacro				
	12	**8**	**19**	**8**

dec_16 – absolute,x

	A		B		C		D	
.macro _dec_vax_16 var	θ	B	θ	B	θ	B	θ	B
lda var,x	4	2	4	2	5	2	5	2
bne no_wrap	3	2	2	2	2	2	3	2
dec var+1,x	0	2	6	2	0	2	7	2
no_wrap:								
dec var,x	6	2	6	2	7	2	7	2
.endmacro								
	13	**8**	**18**	**8**	**14**	**8**	**22**	**8**

A decrement with no wrap
B decrement with wrap
C decrement with no wrap, crosses page
D decrement with wrap, crosses page

dec_16 – absolute,y

	A		**B**		**C**		**D**	
.macro _dec_vax_16 var	θ	B	θ	B	θ	B	θ	B
lda var,y	4	2	4	2	5	2	5	2
bne no_wrap	3	2	2	2	3	2	2	2
lda var+1,y	0	2	4	2	0	2	5	2
dec	0	1	0	1	0	1	2	1
sta var+1,y	0	2	4	2	0	2	5	2
lda var,y	0	2	4	2	0	2	5	2
no_wrap:								
dec	2	1	2	1	2	1	2	1
sta var,y	4	2	4	2	5	2	5	2
.endmacro								
	13	**14**	**24**	**14**	**15**	**14**	**31**	**14**

dec_16 – zero page indirect

	A		**B**	
.macro _dec_zpp_16 var	θ	B	θ	B
lda (var)	5	2	5	2
bne no_wrap	3	2	2	2
ldy #$1	0	2	2	2
lda (var),y	0	2	5	2
dec	0	1	2	1
sta (var),y	0	2	5	2
lda (var)	0	2	5	2
no_wrap:				
dec	2	1	2	1
sta (var)	4	2	5	2
.endmacro				
	14	**16**	**33**	**16**

dec_16 – zero page indirect,y

	A		**B**	
.macro _dec_zpp_16 var	θ	B	θ	B
lda (var),y	5	2	5	2
bne no_wrap	3	2	2	2
iny	0	1	2	1
lda (var),y	0	2	5	2
dec	0	1	2	1
sta (var),y	0	2	5	2
dey				
lda (var)	0	2	5	2
no_wrap:				
dec	2	1	2	1
sta (var),y	5	2	5	2
.endmacro				
	15	**15**	**33**	**15**

A decrement with no wrap
B decrement with wrap
C decrement with no wrap, crosses page
D decrement with wrap, crosses page

eql_16 - zero page

	A		B		C		D		E		F		G		H		I	
.macro _eql_var_16 var, value	θ	B	θ	B	θ	B	θ	B	θ	B	θ	B	θ	B	θ	B	θ	B
.ifconst value																		
.if value																		
lda var			3	2	3	2	3	2	3	2	3	2	3	2				
.if .lobyte(value)																		
cmp #.lobyte(value)							2	2	2	2	2	2	2	2				
.endif																		
bne not_equal			2	2	3	2	2	2	3	2	2	2	3	2				
lda var+1			3	2	0	2	3	2	0	2	3	2	0	2				
.if .hibyte(value)																		
cmp #.hibyte(value)			2	2	0	2					2	2	0	2				
.endif																		
.else																		
lda var	3	2																
ora var+1	3	2																
.endif																		
.else																		
lda var															3	2	2	2
cmp #.lobyte(value)															2	2	2	2
bne not_equal															2	2	3	2
lda var+1															3	2	0	2
cmp #.hibyte(value)															2	2	0	2
not_equal:																		
.endmacro																		
	6	**4**	**10**	**8**	**6**	**8**	**10**	**8**	**8**	**8**	**12**	**10**	**8**	**10**	**12**	**10**	**7**	**10**

A value is const 0
B value is const xx00, low bytes equal
C value is const xx00, low bytes not equal
D value is const 00xx, low bytes equal
E value is const 00xx, low bytes not equal
F value is const, low bytes equal
G value is const, low bytes not equal
H value is not constant, low bytes equal
I value is not constant, low bytes not equal

eql_16 – absolute

	A		B		C		D		E		F		G		H		I	
.macro _eql_var_16 var, value	θ	B	θ	B	θ	B	θ	B	θ	B	θ	B	θ	B	θ	B	θ	B
.ifconst value																		
.if value																		
lda var			4	3	4	3	4	3	4	3	4	3	4	3				
.if .lobyte(value)																		
cmp #.lobyte(value)							2	2	2	2	2	2	2	2				
.endif																		
bne not_equal			2	2	3	2	2	2	3	2	2	2	3	2				
lda var+1			4	3	0	3	4	3	0	3	4	3	0	3				
.if .hibyte(value)																		
cmp #.hibyte(value)			2	2	0	2					2	2	0	2				
.endif																		
.else																		
lda var	4	3																
ora var+1	4	3																
.endif																		
.else																		
lda var															4	3	4	3
cmp #.lobyte(value)															2	2	2	2
bne not_equal															2	2	3	2
lda var+1															4	3	0	3
cmp #.hibyte(value)															2	2	0	2
not_equal:																		
.endmacro																		
	8	**6**	**12**	**10**	**7**	**10**	**12**	**10**	**9**	**10**	**14**	**12**	**9**	**12**	**14**	**12**	**9**	**12**

A value is const 0
B value is const xx00, low bytes equal
C value is const xx00, low bytes not equal
D value is const 00xx, low bytes equal
E value is const 00xx, low bytes not equal
F value is const, low bytes equal
G value is const, low bytes not equal
H value is not constant, low bytes equal
I value is not constant, low bytes not equal

eql_16 – zero page,x

	A		**B**		**C**		**D**		**E**		**F**		**G**		**H**		**I**	
.macro _eql_var_16 var, value	θ	B	θ	B	θ	B	θ	B	θ	B	θ	B	θ	B	θ	B	θ	B
.ifconst value																		
.if value																		
lda var,x			4	2	4	2	4	2	4	2	4	2	4	2				
.if .lobyte(value)																		
cmp #.lobyte(value)							2	2	2	2	2	2	2	2				
.endif																		
bne not_equal			2	2	3	2	2	2	3	2	2	2	3	2				
lda var+1,x			4	2	0	2	4	2	0	2	4	2	0	2				
.if .hibyte(value)																		
cmp #.hibyte(value)			2	2	0	2					2	2	0	2				
.endif																		
.else																		
lda var,x	4	2																
ora var+1,x	4	2																
.endif																		
.else																		
lda var,x															4	2	4	2
cmp #.lobyte(value)															2	2	2	2
bne not_equal															2	2	3	2
lda var+1,x															4	2	0	2
cmp #.hibyte(value)															2	2	0	2
not_equal:																		
.endmacro																		
	8	**4**	**12**	**8**	**7**	**8**	**12**	**8**	**9**	**8**	**14**	**10**	**9**	**10**	**14**	**10**	**9**	**10**

A value is const 0
B value is const xx00, low bytes equal
C value is const xx00, low bytes not equal
D value is const 00xx, low bytes equal
E value is const 00xx, low bytes not equal
F value is const, low bytes equal
G value is const, low bytes not equal
H value is not constant, low bytes equal
I value is not constant, low bytes not equal

eql_16 – absolute,x/y no page cross

	A		**B**		**C**		**D**		**E**		**F**		**G**		**H**		**I**	
.macro _eql_vax_16 var, value	θ	B	θ	B	θ	B	θ	B	θ	B	θ	B	θ	B	θ	B	θ	B
.ifconst value																		
.if value																		
lda var,x			4	3	4	3	4	3	4	3	4	3	4	3				
.if .lobyte(value)																		
cmp #.lobyte(value)							2	2	2	2	2	2	2	2				
.endif																		
bne not_equal			2	2	3	2	2	2	3	2	2	2	3	2				
lda var+1,x			4	3	0	3	4	3	0	3	4	3	0	3				
.if .hibyte(value)																		
cmp #.hibyte(value)			2	2	0	2					2	2	0	2				
.endif																		
.else																		
lda var,x	4	3																
ora var+1,x	4	3																
.endif																		
.else																		
lda var,x															4	3	4	3
cmp #.lobyte(value)															2	2	2	2
bne not_equal															2	2	3	2
lda var+1,x															4	3	0	3
cmp #.hibyte(value)															2	2	0	2
not_equal:																		
.endmacro																		
	8	**6**	**12**	**10**	**7**	**10**	**12**	**10**	**9**	**10**	**14**	**12**	**9**	**12**	**14**	**12**	**9**	**12**

A value is const 0
B value is const xx00, low bytes equal
C value is const xx00, low bytes not equal
D value is const 00xx, low bytes equal
E value is const 00xx, low bytes not equal
F value is const, low bytes equal
G value is const, low bytes not equal
H value is not constant, low bytes equal
I value is not constant, low bytes not equal

eql_16 – absolute,x/y with page cross

	A		**B**		**C**		**D**		**E**		**F**		**G**		**H**		**I**	
.macro _eql_vax_16 var, value	θ	B	θ	B	θ	B	θ	B	θ	B	θ	B	θ	B	θ	B	θ	B
.ifconst value																		
.if value																		
lda var,x			5	3	5	3	5	3	5	3	5	3	5	3				
.if .lobyte(value)																		
cmp #.lobyte(value)							2	2	2	2	2	2	2	2				
.endif																		
bne not_equal			2	2	3	2	2	2	3	2	2	2	3	2				
lda var+1,x			5	3	0	3	5	3	0	3	5	3	0	3				
.if .hibyte(value)																		
cmp #.hibyte(value)			2	2	0	2					2	2	0	2				
.endif																		
.else																		
lda var,x	5	3																
ora var+1,x	5	3																
.endif																		
.else																		
lda var,x															5	3	5	3
cmp #.lobyte(value)															2	2	2	2
bne not_equal															2	2	3	2
lda var+1,x															5	3	0	3
cmp #.hibyte(value)															2	2	0	2
not_equal:																		
.endmacro																		
	10	**6**	**14**	**10**	**8**	**10**	**14**	**10**	**10**	**10**	**16**	**12**	**10**	**12**	**16**	**12**	**10**	**12**

A value is const 0
B value is const xx00, low bytes equal
C value is const xx00, low bytes not equal
D value is const 00xx, low bytes equal
E value is const 00xx, low bytes not equal
F value is const, low bytes equal
G value is const, low bytes not equal
H value is not constant, low bytes equal
I value is not constant, low bytes not equal

eql_16 – zero page indirect

	A		B		C		D		E		F		G		H		I	
.macro _eql_zpp_16 var, value	θ	B	θ	B	θ	B	θ	B	θ	B	θ	B	θ	B	θ	B	θ	B
.ifconst value																		
.if value																		
lda (var)			5	2	5	2	5	2	5	2	5	2	5	2				
.if .lobyte(value)																		
cmp #.lobyte(value)							2	2	2	2	2	2	2	2				
.endif																		
bne not_equal			2	2	3	2	2	2	3	2	2	2	3	2				
ldy #1			2	2	0	2	2	2	0	2	2	2	0	2				
lda (var),y			5	2	0	2	5	2	0	2	5	2	0	2				
.if .hibyte(value)																		
cmp #.hibyte(value)			2	2	0	2					2	2	0	2				
.endif																		
.else																		
lda (var)	5	2																
ldy #1	2	2																
ora (var),y	5	2																
.endif																		
.else																		
lda (var)															5	2	5	2
cmp #.hibyte(value)															2	2	2	2
bne not_equal															2	2	3	2
ldy #1															2	2	0	2
lda (var),y															5	2	0	2
cmp #.lobyte(value)															2	2	0	2
not_equal:																		
.endmacro																		
	12	**6**	**16**	**10**	**8**	**10**	**16**	**10**	**10**	**10**	**18**	**12**	**10**	**12**	**18**	**12**	**10**	**12**

A value is const 0
B value is const xx00, low bytes equal
C value is const xx00, low bytes not equal
D value is const 00xx, low bytes equal
E value is const 00xx, low bytes not equal
F value is const, low bytes equal
G value is const, low bytes not equal
H value is not constant, low bytes equal
I value is not constant, low bytes not equal

eql_16 – zero page indirect,y

	A		B		C		F		G		H		I	
.macro _eql_zpy_16 var, value	θ	B	θ	B	θ	B	θ	B	θ	B	θ	B	θ	B
.ifconst value														
.if value														
lda (var),y			5	2	5	2	5	2	5	2				
.if .lobyte(value)														
cmp #.lobyte(value)							2	2	2	2				
.endif														
bne not_equal			2	2	3	2	2	2	3	2				
iny			2	1	0	1	2	1	0	1				
ldy (var),y			5	2	0	2	5	2	0	2				
dey			2	1	0	1	2	1	0	1				
cmp #.hibyte(value)			2	2	0	2	2	2	0	2				
.else														
iny	2	2												
lda (var).y	5	2												
dey	2	2												
ora (var),y	5	2												
.endif														
.else														
iny											2	1	2	1
lda (var),y											5	2	5	2
dey											2	1	2	1
cmp #.hibyte(value)											2	2	2	2
bne not_equal											2	2	3	2
lda (var),y											5	2	0	2
cmp #.lobyte(value)											2	2	0	2
not_equal:														
.endmacro														
	14	**8**	**18**	**10**	**8**	**10**	**20**	**12**	**10**	**12**	**20**	**12**	**14**	**12**

A value is const 0
B value is const xx00, low bytes equal
C value is const xx00, low bytes not equal
F value is const, low bytes equal
G value is const, low bytes not equal
H value is not constant, low bytes equal
I value is not constant, low bytes not equal

gte_16 - zero page

	A		B	
.macro _gte_var_16 var, value	θ	B	θ	B
sec	2	1	2	1
.if (!.const(value))||.lobyte(value)				
lda var			3	2
sbc #.lobyte(value)			2	2
.endif				
lda var+1	3	2	3	2
sbc #.hibyte(value)	2	2	2	2
.endmacro				
	7	**5**	**12**	**9**

gte_16 – absolute

	A		B	
.macro _gte_var_16 var, value	θ	B	θ	B
sec	2	1	2	1
.if (!.const(value))||.lobyte(value)				
lda var			4	3
sbc #.lobyte(value)			2	2
.endif				
lda var+1	4	2	4	3
sbc #.hibyte(value)	2	2	2	2
.endmacro				
	8	**5**	**14**	**11**

gte_16 - zero page,x

	A		B	
.macro _gte_vax_16 var, value	θ	B	θ	B
sec	2	1	2	1
.if (!.const(value))||.lobyte(value)				
lda var,x			4	2
sbc #.lobyte(value)			2	2
.endif				
lda var+1,x	4	2	4	2
sbc #.hibyte(value)	2	2	2	2
.endmacro				
	8	**5**	**14**	**9**

gte_16 – absolute,x/y

	A		B		C		D	
.macro _gte_vax_16 var, value	θ	B	θ	B	θ	B	θ	B
sec	2	1	2	1	2	1	2	1
.if (!.const(value))||.lobyte(value)								
lda var,x			4	3			5	3
sbc #.lobyte(value)			2	2			2	2
.endif								
lda var+1,x	4	3	4	3	5	3	5	3
sbc #.hibyte(value)	2	2	2	2	2	2	2	2
.endmacro								
	8	**6**	**14**	**11**	**9**	**6**	**16**	**11**

A const value xx00
B not const value or not xx00
C const value xx00, page cross
D not const value or not xx00, page cross

gte_16 - zero page indirect

	A		B	
.macro _gte_zpp_16 var, value	θ	B	θ	B
sec	2	1	2	1
.if (!.const(value))||.lobyte(value)				
lda (var)			5	2
sbc #.lobyte(value)			2	2
.endif				
ldy #1	2	2	2	2
lda (var),y	5	2	5	2
sbc #.hibyte(value)	2	2	2	2
.endmacro				
	11	**7**	**18**	**11**

gte_16 - zero page indirect,y

	A		B	
.macro _gte_zpy_16 var, value	θ	B	θ	B
sec	2	1	2	1
.if (!.const(value))||.lobyte(value)				
lda (var),y			5	2
sbc #.lobyte(value)			2	2
.endif				
iny	2	1	2	1
lda (var),y	5	2	5	2
dey	2	1	2	1
sbc #.hibyte(value)	2	2	2	2
.endmacro				
	13	**7**	**20**	**11**

A const value xx00
B not const value or not xx00

inc_16 - zero page

	A		B	
.macro _inc_var_16 var	θ	B	θ	B
inc var	5	2	5	2
bne no_wrap	3	2	2	2
inc var+1	0	2	5	2
no_wrap:				
.endmacro				
	8	**6**	**12**	**6**

inc_16 – absolute

	A		B	
.macro _inc_var_16 var	θ	B	θ	B
inc var	6	3	6	3
bne no_wrap	3	2	2	2
inc var+1	0	3	6	3
no_wrap:				
	9	**8**	**14**	**8**

inc_16 - zero page,x

	A		B	
.macro _inc_vax_16 var	θ	B	θ	B
inc var,x	6	2	6	2
bne no_wrap	3	2	2	2
inc var+1,x	0	2	6	2
no_wrap:				
.endmacro				
	9	**6**	**14**	**6**

inc_16 – absolute,x

	A		B		C		D	
.macro _inc_vax_16 var	θ	B	θ	B	θ	B	θ	B
inc var,x	6	3	6	3	7	3	7	3
bne no_wrap	3	2	2	2	3	2	2	2
inc var+1,x	0	3	6	3	0	3	7	3
no_wrap:								
.endmacro								
	9	**8**	**14**	**8**	**10**	**8**	**16**	**8**

A decrement with no wrap
B decrement with wrap
C decrement with no wrap, crosses page
D decrement with wrap, crosses page

inc_16 – absolute,y

	A		B		C		D	
.macro _inc_vax_16 var	θ	B	θ	B	θ	B	θ	B
lda var,y	4	3	4	3	5	3	5	3
inc	2	1	2	1	2	1	2	1
sta var,y	4	3	4	3	5	3	5	3
bne no_wrap	3	2	2	2	3	2	2	2
lda var+1,y	0	3	4	3	5	3	5	3
inc	0	1	2	1	0	1	2	1
sta var+1,y	0	3	4	3	5	3	5	3
no_wrap:								
.endmacro								
	9	**13**	**18**	**13**	**20**	**13**	**21**	**13**

inc_16 – zero page indirect

	A		B	
.macro _inc_zpp_16 var	θ	B	θ	B
lda (var)	5	2	5	2
inc	2	1	2	1
sta (var)	5	2	5	2
bne no_wrap	3	2	2	2
ldy #1	0	2	2	2
lda (var),y	0	3	4	3
inc	0	1	2	1
sta (var),y	0	3	4	3
no_wrap:				
.endmacro				
	10	**14**	**21**	**14**

inc_16 – zero page indirect,y

	A		B	
.macro _inc_vax_16 var	θ	B	θ	B
lda (var),y	5	2	5	2
inc	2	1	2	1
sta (var),y	5	2	5	2
bne no_wrap	3	2	2	2
iny	0	1	2	1
lda (var),y	0	2	5	2
inc	0	1	2	1
sta (var),y	0	2	5	2
dey	0	1	2	1
no_wrap:				
.endmacro				
	10	**12**	**25**	**12**

A decrement with no wrap
B decrement with wrap
C decrement with no wrap, crosses page
D decrement with wrap, crosses page

lbra

	A	
.macro lbra target	θ	B
jmp target	3	3
.endmacro		
	3	**3**

lbcc, also lbcs, lbne, lbeq, lbpl, lbmi, lbvc, and lbvs

	A		B	
.macro lbcc target	θ	B	θ	B
bcs not_taken	2	2	3	2
jmp target	3	3	0	3
not_taken:				
.endmacro				
	5	**5**	**3**	**5**

A branch taken
B branch not taken

lbslt

	A		B		C		D	
.macro lbslt target	θ	B	θ	B	θ	B	θ	B
bvs sign_flipped	2	2	2	2	3	2	3	2
bpl not_taken	2	2	3	2	0	2	0	2
taken:								
jmp target	3	3	0	3	3	3	0	3
sign_flipped:								
bpl taken	0	2	0	2	3	2	2	2
not_taken:								
.endmacro								
	7	**9**	**5**	**9**	**9**	**9**	**5**	**9**

A branch taken
B branch not taken
C branch taken, sign flipped
D branch not taken, signed flipped

lbsle

	A		B		C		D		E	
.macro lbsle target	θ	B	θ	B	θ	B	θ	B	θ	B
beq taken	2	2	2	2	2	2	2	2	3	2
bvs sign_flipped	2	2	2	2	3	2	3	2	0	2
bpl not_taken	2	2	3	2	0	2	0	2	0	2
taken:										
jmp target	3	3	0	3	3	3	0	3	3	3
sign_flipped:										
bpl taken	0	2	0	2	3	2	2	2	0	2
not_taken:										
.endmacro										
	9	**11**	**7**	**11**	**11**	**11**	**7**	**11**	**6**	**11**

A branch taken
B branch not taken
C branch taken, sign flipped
D branch not taken, signed flipped
E branch taken, equal

lbsge

	A		B		C		D	
.macro lbsge target	θ	B	θ	B	θ	B	θ	B
bvs sign_flipped	2	2	2	2	3	2	3	2
bmi not_taken	2	2	3	2	0	2	0	2
taken:								
jmp target	3	3	0	3	3	3	0	3
sign_flipped:								
bmi taken	0	2	0	2	3	2	2	2
not_taken:								
.endmacro								
	7	**9**	**5**	**9**	**9**	**9**	**5**	**9**

A branch taken
B branch not taken
C branch taken, sign flipped
D branch not taken, signed flipped

lbsgt

	A		B		C		D		E	
.macro lbslt target	θ	B	θ	B	θ	B	θ	B	θ	B
beq not_taken	2	2	2	2	2	2	2	2	3	2
bvs sign_flipped	2	2	2	2	3	2	3	2	0	2
bmi not_taken	2	2	3	2	0	2	0	2	0	2
taken:										
jmp target	3	3	0	3	3	3	0	3	0	3
sign_flipped:										
bmi taken	0	2	0	2	3	2	2	2	0	2
not_taken:										
.endmacro										
	9	**11**	**7**	**11**	**11**	**11**	**7**	**11**	**3**	**11**

A branch taken
B branch not taken
C branch taken, sign flipped
D branch not taken, signed flipped
E branch not taken, equal

lbult

	A		B	
.macro lbult target	θ	B	θ	B
bcs not_taken	2	2	3	2
jmp target	3	3	0	3
not_taken:				
.endmacro				
	5	**5**	**3**	**5**

A branch taken
B branch not taken

lbule

	A		B		C	
.macro lbule target	θ	B	θ	B	θ	B
beq taken	2	2	2	2	3	2
bcs not_taken	2	2	3	2	0	2
taken:						
jmp target	3	3	0	3	3	3
not_taken:						
.endmacro						
	7	**7**	**5**	**7**	**6**	**7**

A branch taken
B branch not taken
C branch taken, equal

lbuge

	A		B	
.macro lbuge target	θ	B	θ	B
bcc not_taken	2	2	3	2
jmp target	3	3	0	3
not_taken:				
.endmacro				
	5	**5**	**3**	**5**

A branch taken
B branch not taken

lbugt

	A		B		C	
.macro lbult target	θ	B	θ	B	θ	B
bcc not_taken	2	2	2	2	3	2
beq not_taken	2	2	3	2	0	2
jmp target	3	3	0	3	0	3
not_taken:						
.endmacro						
	7	**7**	**5**	**7**	**3**	**7**

A branch taken
B branch not taken
C branch not taken, equal

bslt

	A		B		C		D	
.macro bslt target	θ	B	θ	B	θ	B	θ	B
bvs sign_flipped	2	2	2	2	3	2	3	2
bmi target	3	2	2	2	0	2	0	2
bra not_taken	0	2	3	2	0	2	0	2
sign_flipped:								
bpl target	0	2	0	2	3	2	2	2
not_taken:								
.endmacro								
	5	**8**	**7**	**8**	**6**	**8**	**5**	**8**

A branch taken
B branch not taken
C branch taken, sign flipped
D branch not taken, signed flipped

bsle

	A		B		C		D		E	
.macro bsle target	θ	B	θ	B	θ	B	θ	B	θ	B
beq target	2	2	2	2	2	2	2	2	3	2
bvs sign_flipped	2	2	2	2	3	2	3	2	0	2
bmi target	3	2	2	2	0	2	0	2	0	2
bra not_taken	0	2	3	2	0	2	0	2	0	2
sign_flipped:										
bpl target	0	2	0	2	3	2	2	2	0	2
not_taken:										
.endmacro										
	7	**10**	**9**	**10**	**8**	**10**	**7**	**10**	**3**	**10**

A branch taken
B branch not taken
C branch taken, sign flipped
D branch not taken, signed flipped
E branch taken, equal

bsge

	A		B		C		D	
.macro bsge target	θ	B	θ	B	θ	B	θ	B
bvs sign_flipped	2	2	2	2	3	2	3	2
bpl target	3	2	2	2	0	2	0	2
bra not_taken	0	2	3	2	0	2	0	2
sign_flipped:								
bmi target	0	2	0	2	3	2	2	2
not_taken:								
.endmacro								
	5	**8**	**7**	**8**	**6**	**8**	**5**	**8**

A branch taken
B branch not taken
C branch taken, sign flipped
D branch not taken, signed flipped

bsgt

	A		B		C		D		E	
.macro bslt target	θ	B	θ	B	θ	B	θ	B	θ	B
beq not_taken	2	2	2	2	2	2	2	2	3	2
bvs sign_flipped	2	2	2	2	3	2	3	2	0	2
bpl target	3	2	2	2	0	2	0	2	0	2
bra not_taken	0	2	3	2	0	2	0	2	0	2
sign_flipped:										
bmi target	0	2	0	2	3	2	2	2	0	2
not_taken:										
.endmacro										
	7	**10**	**9**	**10**	**8**	**10**	**7**	**10**	**3**	**10**

A branch taken
B branch not taken
C branch taken, sign flipped
D branch not taken, signed flipped
E branch not taken, equal

set_16 - zero page

	A		B		C		D		E	
.macro _set_var_16 var,value	θ	B	θ	B	θ	B	θ	B	θ	B
.ifconst value										
.if .lobyte(value)										
lda #.lobyte(value)			2	2			2	2		
sta var			3	2			3	2		
.else										
stz var	3	2			3	2				
.endif										
.if .hibyte(value)										
lda #.hibyte(value)					2	2	2	2		
sta var+1					3	2	3	2		
.else										
stz var+1	3	2	3	2						
.endif										
.else										
lda #.lobyte(value)									2	2
sta var									3	2
lda #.hibyte(value)									2	2
sta var+1									3	2
.endif										
.endmacro										
	6	**4**	**8**	**6**	**8**	**6**	**10**	**8**	**10**	**8**

set_16 – absolute

	A		B		C		D		E	
.macro _set_var_16 var,value	θ	B	θ	B	θ	B	θ	B	θ	B
.ifconst value										
.if .lobyte(value)										
lda #.lobyte(value)			2	2			2	2		
sta var			4	3			4	3		
.else										
stz var	4	3			4	3				
.endif										
.if .hibyte(value)										
lda #.hibyte(value)					2	2	2	2		
sta var+1					4	3	4	3		
.else										
stz var+1	4	3	4	3						
.endif										
.else										
lda #.lobyte(value)									2	2
sta var									4	3
lda #.hibyte(value)									2	2
sta var+1									4	3
.endif										
.endmacro										
	8	**6**	**10**	**8**	**10**	**8**	**12**	**10**	**12**	**10**

A set const value 0000
B set const value 00xx
C set const value xx00
D set const value xxxx
E set non-const value

set_16 – zero page,x

	A		**B**		**C**		**D**		**E**	
.macro _set_vax_16 var,value	θ	B	θ	B	θ	B	θ	B	θ	B
.ifconst value										
.if .lobyte(value)										
lda #.lobyte(value)			2	2			2	2		
sta var,x			4	2			4	2		
.else										
stz var,x	4	2			4	2				
.endif										
.if .hibyte(value)										
lda #.hibyte(value)					2	2	2	2		
sta var+1,x					4	2	4	2		
.else										
stz var+1,x	4	2	4	2						
.endif										
.else										
lda #.lobyte(value)									2	2
sta var,x									4	2
lda #.hibyte(value)									2	2
sta var+1,x									4	2
.endif										
.endmacro										
	8	**4**	**10**	**6**	**10**	**6**	**12**	**8**	**12**	**8**

set_16 – absolute,x – no page cross

	A		**B**		**C**		**D**		**E**	
.macro _set_vax_16 var,value	θ	B	θ	B	θ	B	θ	B	θ	B
.ifconst value										
.if .lobyte(value)										
lda #.lobyte(value)			2	2			2	2		
sta var,x			4	3			4	3		
.else										
stz var,x	4	3			4	3				
.endif										
.if .hibyte(value)										
lda #.hibyte(value)					2	2	2	2		
sta var+1,x					4	3	4	3		
.else										
stz var+1,x	4	3	4	3						
.endif										
.else										
lda #.lobyte(value)									2	2
sta var,x									4	3
lda #.hibyte(value)									2	2
sta var+1,x									4	3
.endif										
.endmacro										
	8	**6**	**10**	**8**	**10**	**8**	**12**	**10**	**12**	**10**

A set const value 0000
B set const value 00xx
C set const value xx00
D set const value xxxx
E set non-const value

set_16 – absolute,x – with page cross

	A		**B**		**C**		**D**		**E**	
.macro _set_vax_16 var,value	θ	B	θ	B	θ	B	θ	B	θ	B
.ifconst value										
.if .lobyte(value)										
lda #.lobyte(value)			2	2			2	2		
sta var,x			5	3			5	3		
.else										
stz var,x	5	3			5	3				
.endif										
.if .hibyte(value)										
lda #.hibyte(value)					2	2	2	2		
sta var+1,x					5	3	5	3		
.else										
stz var+1,x	5	3	5	3						
.endif										
.else										
lda #.lobyte(value)									2	2
sta var,x									5	3
lda #.hibyte(value)									2	2
sta var+1,x									5	3
.endif										
.endmacro										
	10	**6**	**12**	**8**	**12**	**8**	**14**	**10**	**14**	**10**

A set const value 0000
B set const value 00xx
C set const value xx00
D set const value xxxx
E set non-const value

set_16 – absolute,y

	A		**B**	
.macro _set_vay_16 var,value	θ	B	θ	B
lda #.lobyte(value)	2	2	2	2
sta var,y	4	3	5	3
lda #.hibyte(value)	2	2	2	2
sta var+1,y	4	3	5	3
.endmacro				
	12	**10**	**14**	**10**

A no page cross
B with page cross

set_16 – zero page indirect

	A	
.macro _set_zpp_16 var,value	θ	B
lda #.lobyte(value)	2	2
sta (var)	4	2
ldy #1	2	2
lda #.hibyte(value)	2	2
sta (var),y	4	2
.endmacro		
	14	**10**

set_16 – zero page indirect,y

	A	
.macro _set_zpy_16 var,value	θ	B
lda #.lobyte(value)	2	2
sta (var),y	4	2
iny	2	1
lda #.hibyte(value)	2	2
sta (var),y	4	2
dey	2	2
.endmacro		
	16	**11**

tst_16 - zero page

	A		B		C	
.macro _tst_var_16 var	θ	B	θ	B	θ	B
lda var+1	3	2	3	2	3	2
bne all_done	3	2	2	2	2	2
ora var	0	2	3	2	3	2
bpl all_done	0	2	3	2	2	2
lsr	0	1	0	1	2	1
all_done:						
.endmacro						
	6	**9**	**11**	**9**	**12**	**9**

tst_16 – absolute

	A		B		C	
.macro _tst_var_16 var	θ	B	θ	B	θ	B
lda var+1	4	3	4	3	4	3
bne all_done	3	2	2	2	2	2
ora var	0	3	4	3	4	3
bpl all_done	0	2	3	2	2	2
lsr	0	1	0	1	2	1
all_done:						
.endmacro						
	7	**11**	**13**	**11**	**14**	**11**

tst_16 – zero page,x

	A		B		C	
.macro _tst_var_16 var	θ	B	θ	B	θ	B
lda var+1,x	4	2	4	2	4	2
bne all_done	3	2	2	2	2	2
ora var,x	0	2	4	2	4	2
bpl all_done	0	2	3	2	2	2
lsr	0	1	0	1	2	1
all_done:						
.endmacro						
	7	**9**	**13**	**9**	**14**	**9**

A high byte != 0
B high byte = 0, low byte < $80
C high byte = 0, low byte >= $80

tst_16 – absolute,x/y

	A		**B**		**C**		**D**		**E**		**F**	
.macro _tst_var_16 var	θ	B	θ	B	θ	B	θ	B	θ	B	θ	B
lda var+1,x	4	3	4	3	4	3	5	3	5	3	5	3
bne all_done	3	2	2	2	2	2	3	2	2	2	2	2
ora var,x	0	3	4	3	4	3	0	3	5	3	5	3
bpl all_done	0	2	3	2	2	2	0	2	2	2	3	2
lsr	0	1	0	1	2	1	0	1	0	1	2	1
all_done:												
.endmacro												
	7	**11**	**13**	**11**	**14**	**11**	**8**	**11**	**14**	**11**	**17**	**11**

A high byte != 0
B high byte = 0, low byte < $80
C high byte = 0, low byte >= $80
D high byte != 0
E high byte = 0, low byte < $80
F high byte = 0, low byte >= $80

bult

	A		B	
.macro bult target	θ	B	θ	B
bcc target	3	2	2	2
not_taken:				
.endmacro				
	3	**2**	**2**	**2**

A branch taken
B branch not taken

bule

	A		B		C	
.macro bult target	θ	B	θ	B	θ	B
beq target	2	2	2	2	3	2
bcc target	3	2	2	2	0	2
.endmacro						
	5	**4**	**4**	**4**	**3**	**4**

A branch taken
B branch not taken
C branch taken, equal

buge

	A		B	
.macro buge target	θ	B	θ	B
bcs target	3	2	2	2
.endmacro				
	3	**2**	**2**	**2**

A branch taken
B branch not taken

bugt

	A		B		C	
.macro bult target	θ	B	θ	B	θ	B
beq not_taken	2	2	2	2	3	2
bcs target	3	2	2	2	0	2
.endmacro						
	5	**4**	**4**	**4**	**3**	**4**

A branch taken
B branch not taken
C branch not taken, equal