Operation, Operation Codes, and Status Register

MNE-MONIC	OPERATION	#	a	al	d	A	i	(d),y	[d],y	(d,x)	d,x	d,y	a,x	al,x	a,y	r	rl	(a)	(d)	[d]	(a,x)	s	d,s	(d,s),y	xyc	PROCESSOR STATUS CODE 7 6 5 4 3 2 1 0	MNE-MONIC
																										N V M X D I Z C E = 0	
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	N V 1 B D I Z C E = 1	
ADC	A + M + C → A	69	6D	6F	65			71	77	61	75		7D	7F	79				72	67			63	73		N V . . . . Z C	ADC
AND	A∧M → A	29	2D	2F	25			31	37	21	35		3D	3F	39				32	27			23	33		N . . . . . Z .	AND
ASL	C ← [15/7 … 0] ← 0		0E		06	0A					16		1E													N . . . . . Z C	ASL
BCC	BRANCH IF C = 0															90										. . . .	BCC
BCS	BRANCH IF C = 1															B0										. . . .	BCS
BEQ	BRANCH IF Z = 1															F0										. . . .	BEQ
BIT	A∧M (NOTE 1)	89	2C		24						34		3C													M₇ M₆ . . . . Z .	BIT
BMI	BRANCH IF N = 1															30										. . . .	BMI
BNE	BRANCH IF Z = 0															D0										. . . .	BNE
BPL	BRANCH IF N = 0															10										. . . .	BPL
BRA	BRANCH ALWAYS															80										. . . .	• BRA
BRK	BREAK (NOTE 2)																									. . . • 0 I . .	BRK
BRL	BRANCH LONG ALWAYS																82					00				. . . .	★ BRL
BVC	BRANCH IF V = 0															50										. . . .	BVC
BVS	BRANCH IF V = 1															70										. . . .	BVS
CLC	0 → C						18																			 0	CLC
CLD	0 → D						D8																			. . . . 0 . . .	CLD
CLI	0 → I						58																			. . . . . 0 . .	CLI
CLV	0 → V						B8																			. 0	CLV
CMP	A-M	C9	CD	CF	C5			D1	D7	C1	D5		DD	DF	D9				D2	C7			C3	D3		N . . . . . Z C	CMP
COP	CO-PROCESSOR																					02				. . . . 0 I . .	★ COP
CPX	X-M	E0	EC		E4																					N . . . . . Z C	CPX
CPY	Y-M	C0	CC		C4																					N . . . . . Z C	CPY
DEC	DECREMENT		CE		C6	3A					D6		DE													N . . . . . Z .	DEC
DEX	X - 1 → X						CA																			N . . . . . Z .	DEX
DEY	Y - 1 → Y						88																			N . . . . . Z .	DEY
EOR	AVM → A	49	4D	4F	45			51	57	41	55		5D	5F	59				52	47			43	53		N . . . . . Z .	EOR
INC	INCREMENT		EE		E6	1A					F6		FE													N . . . . . Z .	INC
INX	X + 1 → X						E8																			N . . . . . Z .	INX
INY	Y + 1 → Y						C8																			N . . . . . Z .	INY
JML	JUMP LONG TO NEW LOC.																	DC								. . . .	★ JML
JMP	JUMP TO NEW LOC.		4C	5C														6C			7C					. . . .	JMP
JSL	JUMP LONG TO SUB.			22																						. . . .	JSL
JSR	JUMP TO SUB.		20																		FC					. . . .	JSR
LDA	M → A	A9	AD	AF	A5			B1	B7	A1	B5		BD	BF	B9				B2	A7		A3	B3			N . . . . . Z .	LDA
LDX	M → X	A2	AE		A6							B6			BE											N . . . . . Z .	LDX
LDY	M → Y	A0	AC		A4						B4		BC													N . . . . . Z .	LDY
LSR	0 → [15/7 … 0] → C		4E		46	4A					56		5E													0 . . . . . Z C	LSR
MVN	M → M BACKWARD																								54	. . . .	★ MVN
MVP	M → M FORWARD																								44	. . . .	★ MVP
NOP	NO OPERATION						EA																			. . . .	NOP
ORA	AVM → A	09	0D	0F	05			11	17	01	15		1D	1F	19				12	07			03	13		N . . . . . Z .	ORA
PEA	Mpc + 1, Mpc + 2 → Ms - 1, Ms S - 2 → S																					F4				. . . .	★ PEA
PEI	M(d), M(d + 1) → Ms - 1, Ms S - 2 → S																					D4				. . . .	★ PEI
PER	Mpc + rl, Mpc + rl + 1 → Ms - 1, Ms S - 2 → S																					62				. . . .	★ PER
PHA	A → Ms, S - 1 → S																					48				. . . .	PHA
PHB	DBR → Ms, S - 1 → S																					8B				. . . .	PHB
PHD	D → Ms, Ms - 1, S - 2 → S																					0B				. . . .	★ PHD
PHK	PBR → Ms, S - 1 → S																					4B				. . . .	★ PHK
PHP	P → Ms, S - 1 → S																					08				. . . .	PHP
PHX	X → Ms, S - 1 → S																					DA				. . . .	• PHX
PHY	Y → Ms, S - 1 → S																					5A				. . . .	• PHY
PLA	S + 1 → S, Ms → A																					68				N . . . . . Z .	PLA
PLB	S + 1 → S, Ms → DBR																					AB				N . . . . . Z .	★ PLB
PLD	S + 2 → S, Ms - 1, Ms → D																					2B				N . . . . . Z .	★ PLD
PLP	S + 1 → S, Ms → P																					28				N V M X D I Z C	PLP
PLX	S + 1 → S, Ms → X																					FA				N . . . . . Z .	PLX
PLY	S + 1 → S, Ms → Y																					7A				N . . . . . Z .	PLY
REP	M∧P → P	C2																								N V M X D I Z C	★ REP
ROL	[15/7 … 0] ← C		2E		26	2A					36		3E													N . . . . . Z C	ROL
ROR	C → [15/7 … 0]		6E		66	6A					76		7E													N . . . . . Z C	ROR
RTI	RTRN FROM INT.																					40				N V M X D I Z C	RTI
RTL	RTRN FROM SUB. LONG																					6B				. . . .	★ RTL
RTS	RTRN SUBROUTINE																					60				. . . .	RTS
SBC	A - M - C̄ → A	E9	ED	EF	E5			F1	F7	E1	F5		FD	FF	F9				F2	E7			E3	F3		N V . . . . Z C	SBC
SEC	1 → C						38																			 1	SEC
SED	1 → D						F8																			. . . . 1 . . .	SED
SEI	1 → I						78																			. . . . . 1 . .	SEI
SEP	MVP → P	E2																								N V M X D I Z C	★ SEP
STA	A → M		8D	8F	85			91	97	81	95		9D	9F	99				92	87			83	93		. . . .	STA
STP	STOP (1 → φ2)						DB																			. . . .	• STP
STX	X → M		8E		86							96														. . . .	STX
STY	Y → M		8C		84						94															. . . .	STY
STZ	00 → M		9C		64						74		9E													. . . .	• STZ
TAX	A → X						AA																			N . . . . . Z .	TAX
TAY	A → Y						A8																			N . . . . . Z .	TAY
TCD	C → D						5B																			N . . . . . Z .	★ TCD
TCS	C → S						1B																			. . . .	★ TCS
TDC	D → C						7B																			N . . . . . Z .	★ TDC
TRB			1C		14																					. . . . . Z .	• TRB
TSB	AVM → M		0C		04																					. . . . . Z .	• TSB
TSC	S → C						3B																			N . . . . . Z .	★ TSC
TSX	S → X						BA																			N . . . . . Z .	TSX
TXA	X → A						8A																			N . . . . . Z .	TXA
TXS	X → S						9A																			. . . .	TXS
TXY	X → Y						9B																			N . . . . . Z .	★ TXY
TYA	Y → A						98																			N . . . . . Z .	TYA
TYX	Y → X						BB																			N . . . . . Z .	★ TYX
WAI	0 → RDY						CB																			. . . .	• WAI
WDM	NO OPERATION (RESERVED)						42																			. . . .	★ WDM
XBA	B ⟷ A						EB																			N . . . . . Z .	★ XBA
XCE	C ⟷ E						FB																			. . . . E	★ XCE

Notes:
1. Bit immediate N and V flags not affected. When M = 0, M_{15} → N and M_{14} → V.
2. Break Bit (B) in Status register indicates hardware or software break.
3. ★ = New W65C816/802 Instructions
   • = New W65C02 Instructions
   Blank = NMOS 6502

+ Add
- Subtract
∧ AND
V OR
⊻ Exclusive OR

Detailed Instruction Operation

	ADDRESS MODE		CYCLE	VP	ML	VDA	VPA	ADDRESS BUS	DATA BUS	R/W
1.	Immediate #		1.	1	1	1	1	PBR,PC	Op Code	1
	(LDY,CPY,CPX,LDX,ORA,		2.	1	1	0	1	PBR,PC+1	IDL	1
	AND,EOR,ADC,BIT,LDA,	(1)(8)	2a.	1	1	0	1	PBR,PC+2	IDH	1
	CMP,SBC REP,SEP)									
	(14 Op Codes)									
	(2 and 3 bytes)									
	(2 and 3 cycles)									
2a.	Absolute a		1.	1	1	1	1	PBR,PC	Op Code	1
	(BIT,STY,STZ,LDY,		2.	1	1	0	1	PBR,PC+1	AAL	1
	CPY,CPX,STX,LDX,		3.	1	1	0	1	PBR,PC+2	AAH	1
	ORA,AND,EOR,ADC,		4.	1	1	1	0	DBR,AA	Data Low	1/0
	STA,LDA,CMP,SBC)	(1)	4a.	1	1	1	0	DBR,AA+1	Data High	1/0
	(18 Op Codes)									
	(3 bytes)									
	(4 and 5 cycles)									
2b.	Absolute (R-M-W) a		1.	1	1	1	1	PBR,PC	Op Code	1
			2.	1	1	0	1	PBR,PC+1	AAL	1
	(ASL,ROL,LSR,ROR		3.	1	1	0	1	PBR,PC+2	AAH	1
	DEC,INC,TSB,TRB)		4.	1	0	1	0	DBR,AA	Data Low	1
	(6 Op Codes)	(1)	4a.	1	0	1	0	DBR,AA+1	Data High	1
	(3 bytes)	(3)	5.	1	0	0	0	DBR,AA+1	IO	1
	(6 and 8 cycles)	(1)	6a.	1	0	1	0	DBR,AA+1	Data High	0
			6.	1	0	1	0	DBR,AA	Data Low	0
2c.	Absolute (JUMP) a		1.	1	1	1	1	PBR,PC	Op Code	1
	(JMP)(4C)		2.	1	1	0	1	PBR,PC+1	NEW PCL	1
	(1 Op Code)		3.	1	1	0	1	PBR,PC+2	NEW PCH	1
	(3 bytes)		1.	1	1	1	1	PBR, NEW PC	Op Code	1
	(3 cycles)									
2d.	Absolute (Jump to		1.	1	1	1	1	PBR,PC	Op Code	1
	subroutine) a		2.	1	1	0	1	PBR,PC+1	NEW PCL	1
	(JSR)		3.	1	1	0	1	PBR,PC+2	NEW PCH	1
	(1 Op Code)		4.	1	1	0	0	PBR,PC+2	IO	1
	(3 bytes)		5.	1	1	1	0	0,S	PCH	0
	(6 cycles)		6.	1	1	1	0	0,S-1	PCL	0
	(different order from N6502)		1.	1	1	1	1	PBR,NEW PC	Next Op Code	1
*3a.	Absolute Long al		1.	1	1	1	1	PBR,PC	Op Code	1
	(ORA,AND,EOR,ADC		2.	1	1	0	1	PBR,PC+1	AAL	1
	STA,LDA,CMP,SBC)		3.	1	1	0	1	PBR,PC+2	AAH	1
	(8 Op Codes)		4.	1	1	0	1	PBR,PC+3	AAB	1
	(4 bytes)		5.	1	1	1	0	AAB,AA	Data Low	1/0
	(5 and 6 cycles)	(1)	5a.	1	1	1	0	AAB,AA+1	Data High	1/0
*3b.	Absolute Long (JUMP) al		1.	1	1	1	1	PBR,PC	Op Code	1
	(JMP)		2.	1	1	0	1	PBR,PC+1	NEW PCL	1
	(1 Op Code)		3.	1	1	0	1	PBR,PC+2	NEW PCH	1
	(4 bytes)		4.	1	1	0	1	PBR,PC+3	NEW BR	1
	(4 cycles)		1.	1	1	1	1	NEW PBR,PC	Op Code	1
*3c.	Absolute Long (Jump to		1.	1	1	1	1	PBR,PC	Op Code	1
	Subroutine Long) al		2.	1	1	0	1	PBR,PC+1	NEW PCL	1
	(JSL)		3.	1	1	0	1	PBR,PC+2	NEW PCH	1
	(1 Op Code)		4.	1	1	1	0	0,S	PBR	0
	(4 bytes)		5.	1	1	0	0	0,S	IO	1
	(7 cycles)		6.	1	1	0	1	PBR,PC+3	NEW PBR	1
			7.	1	1	1	0	0,S-1	PCH	0
			8.	1	1	1	0	0,S-2	PCL	0
			1.	1	1	1	1	NEW PBR,PC	Next Op Code	1
4a.	Direct d		1.	1	1	1	1	PBR,PC	Op Code	1
	(BIT,STZ,STY,LDY,		2.	1	1	0	1	PBR,PC+1	DO	1
	CPY,CPX,STX,LDX,	(2)	2a.	1	1	0	0	PBR,PC+1	IO	1
	ORA,AND,EOR,ADC,		3.	1	1	1	0	0,D+DO	Data Low	1/0
	STA,LDA,CMP,SBC)	(1)	3a.	1	1	1	0	0,D+DO+1	Data High	1/0
	(18 Op Codes)									
	(2 bytes)									
	(3,4 and 5 cycles)									
4b.	Direct (R-M-W) d		1.	1	1	1	1	PBR,PC	Op Code	1
	(ASL,ROL,LSR,ROR		2.	1	1	0	1	PBR,PC+1	DO	1
	DEC,INC,TSB,TRB)	(2)	2a.	1	1	0	0	PBR,PC+1	IO	1
	(6 Op Codes)		3.	1	0	1	0	0,D+DO	Data Low	1
	(2 bytes)	(1)	3a.	1	0	1	0	0,D+DO+1	Data High	1
	(5,6,7 and 8 cycles)	(3)	4.	1	0	0	0	0,D+DO+1	IO	1
		(1)	5a.	1	0	1	0	0,D+DO+1	Data High	0
			5.	1	0	1	0	0,D+DO	Data Low	0
5.	Accumulator A		1.	1	1	1	1	PBR,PC	Op Code	1
	(ASL,INC,ROL,DEC,LSR,ROR)		2.	1	1	0	0	PBR,PC+1	IO	1
	(6 Op Codes)									
	(1 byte)									
	(2 cycles)									
6a.	Implied i		1.	1	1	1	1	PBR,PC	Op Code	1
	(DEY, INY, INX, DEX, NOP,		2.	1	1	0	0	PBR,PC+1	IO	1
	XCE, TYA, TAY,TXA, TXS,									
	TAX,TSX,TCS,TSC,TCD,									
	TDC,TXY,TYX,CLC,SEC,									
	CLI,SEI,CLV,CLD,SED)									
	(25 Op Codes)									
	(1 byte)									
	(2 cycles)									
*6b.	Implied i		1.	1	1	1	1	PBR,PC	Op Code	1
	(XBA)		2.	1	1	0	0	PBR,PC+1	IO	1
	(1 Op Code)		3.	1	1	0	0	PBR,PC+1	IO	1
	(1 byte)									
	(3 cycles)									

									RDY		
• 6c.	Wait For Interrupt										
	(WAI)		1.	1	1	1	1	1	PBR,PC	Op Code	1
	(1 Op Code)	(9)	2.	1	1	0	0	1	PBR,PC+1	IO	1
	(1 byte)		3.	1	1	0	0	0	PBR,PC+1	IO	1
	(3 cycles)	IRQ,NMI	1.	1	1	1	1	1	PBR,PC+1	IRQ(BRK)	1
• 6d.	Stop-The-Clock										
	(STP)		1.	1	1	1	1	1	PBR,PC	Op Code	1
	(1 Op Code)		2.	1	1	0	0	1	PBR,PC+1	IO	1
	(1 byte)	RES=1	3.	1	1	0	0	1	PBR,PC+1	IO	1
	(3 cycles)	RES=0	1c.	1	1	0	0	1	PBR,PC+1	RES(BRK)	1
		RES=0	1b.	1	1	0	0	1	PBR,PC+1	RES(BRK)	1
		RES=1	1a.	1	1	0	0	1	PBR,PC+1	RES(BRK)	1
	See 21a Stack		1.	1	1	1	1	1	PBR,PC+1	BEGIN	1

	ADDRESS MODE		CYCLE	VP	ML	VDA	VPA	ADDRESS BUS	DATA BUS	R/W
7.	Direct Indirect Indexed (d),y		1.	1	1	1	1	PBR,PC	Op Code	1
	(ORA,AND,EOR,ADC,		2.	1	1	0	1	PBR,PC+1	DO	1
	STA,LDA,CMP,SBC)	(2)	2a.	1	1	0	0	PBR,PC+1	IO	1
	(8 Op Codes)		3.	1	1	1	0	0,D+DO	AAL	1
	(2 bytes)		4.	1	1	1	0	0,D+DO+1	AAH	1
	(5,6,7 and 8 cycles)	(4)	4a.	1	1	0	0	DBR,AAH,AAL+Y	YL IO	1
			5.	1	1	1	0	DBR,AA+Y	Data Low	1/0
		(1)	5a.	1	1	1	0	DBR,AA+Y+1	Data High	1/0
8.	Direct Indirect		1.	1	1	1	1	PBR,PC	Op Code	1
	Indexed Long (d),y		2.	1	1	0	1	PBR,PC+1	DO	1
	(ORA,AND,EOR,ADC,	(2)	2a.	1	1	0	0	PBR,PC+1	IO	1
	STA,LDA,CMP,SBC)		3.	1	1	1	0	0,D+DO	AAL	1
	(8 Op Codes)		4.	1	1	1	0	0,D+DO+1	AAH	1
	(2 bytes)		5.	1	1	1	0	0,D+DO+2	AAB	1
	(6,7 and 8 cycles)		6.	1	1	1	0	AAB,AA+Y	Data Low	1/0
		(1)	6a.	1	1	1	0	AAB,AA+Y+1	Data High	1/0
9.	Direct Indexed Indirect (d,x)		1.	1	1	1	1	PBR,PC	Op Code	1
	(ORA,AND,EOR,ADC,		2.	1	1	0	1	PBR,PC+1	DO	1
	STA,LDA,CMP,SBC)	(2)	2a.	1	1	0	0	PBR,PC+1	IO	1
	(8 Op Codes)		3.	1	1	0	0	PBR,PC+1	IO	1
	(2 bytes)		4.	1	1	1	0	0,D+DO+X	AAL	1
	(6,7 and 8 cycles)		5.	1	1	1	0	0,D+DO+X+1	AAH	1
			6.	1	1	1	0	DBR,AA	Data Low	1/0
		(1)	6a.	1	1	1	0	DBR,AA+1	Data High	1/0
10a.	Direct,X d,x		1.	1	1	1	1	PBR,PC	Op Code	1
	(BIT,STZ,STY,LDY,		2.	1	1	0	1	PBR,PC+1	DO	1
	ORA,AND,EOR,ADC,	(2)	2a.	1	1	0	0	PBR,PC+1	IO	1
	STA,LDA,CMP,SBC)		3.	1	1	0	0	PBR,PC+1	IO	1
	(11 Op Codes)		4.	1	1	1	0	0,D+DO+X	Data Low	1/0
	(2 bytes)	(1)	4a.	1	1	1	0	0,D+DO+X+1	Data High	1/0
	(4,5 and 6 cycles)									
10b.	Direct,X(R-M-W) d,x		1.	1	1	1	1	PBR,PC	Op Code	1
	(ASL,ROL,LSR,ROR,		2.	1	1	0	1	PBR,PC+1	DO	1
	DEC,INC)	(2)	2a.	1	1	0	0	PBR,PC+1	IO	1
	(6 Op Codes)		3.	1	1	0	0	PBR,PC+1	IO	1
	(2 bytes)		4.	1	0	1	0	0,D+DO+X	Data Low	1
	(6,7,8 and 9 cycles)	(1)	4a.	1	0	1	0	0,D+DO+X+1	Data High	1
		(3)	5.	1	0	0	0	0,D+DO+X+1	IO	1
		(1)	6a.	1	0	1	0	0,D+DO+X+1	Data High	0
			6.	1	0	1	0	0,D+DO+X	Data Low	0
11.	Direct,Y d,y		1.	1	1	1	1	PBR,PC	Op Code	1
	(STX,LDX)		2.	1	1	0	1	PBR,PC+1	DO	1
	(2 Op Codes)	(2)	2a.	1	1	0	0	PBR,PC+1	IO	1
	(2 bytes)		3.	1	1	0	0	PBR,PC+1	IO	1
	(4,5 and 6 cycles)		4.	1	1	1	0	0,D+DO+Y	Data Low	1/0
		(1)	4a.	1	1	1	0	0,D+DO+Y+1	Data High	1/0
12a.	Absolute,X a,x		1.	1	1	1	1	PBR,PC	Op Code	1
	(BIT,LDY,STZ,		2.	1	1	0	1	PBR,PC+1	AAL	1
	ORA,AND,EOR,ADC,		3.	1	1	0	1	PBR,PC+2	AAH	1
	STA,LDA,CMP,SBC)	(4)	3a.	1	1	0	0	DBR,AAH,AAL+X	XL IO	1
	(11 Op Codes)		4.	1	1	1	0	DBR,AA+X	Data Low	1/0
	(3 bytes)	(1)	4a.	1	1	1	0	DBR,AA+X+1	Data High	1/0
	(4,5 and 6 cycles)									
12b.	Absolute,X(R-M-W) a,x		1.	1	1	1	1	PBR,PC	Op Code	1
	(ASL,ROL,LSR,ROR,		2.	1	1	0	1	PBR,PC+1	AAL	1
	DEC,INC)		3.	1	1	0	1	PBR,PC+2	AAH	1
	(6 Op Codes)		4.	1	1	0	0	DBR,AAH,AAL+X	XL IO	1
	(3 bytes)		5.	1	0	1	0	DBR,AA+X	Data Low	1
	(7 and 9 cycles)	(1)	5a.	1	0	1	0	DBR,AA+X+1	Data High	1
		(3)	6.	1	0	0	0	DBR,AA+X+1	IO	1
		(1)	7a.	1	0	1	0	DBR,AA+X+1	Data High	0
			7.	1	0	1	0	DBR,AA+X	Data Low	0
*13.	Absolute Long,X al,x		1.	1	1	1	1	PBR,PC	Op Code	1
	(ORA,AND,EOR,ADC,		2.	1	1	0	1	PBR,PC+1	AAL	1
	STA,LDA,CMP,SBC)		3.	1	1	0	1	PBR,PC+2	AAH	1
	(8 Op Codes)		4.	1	1	0	1	PBR,PC+3	AAB	1
	(4 bytes)		5.	1	1	1	0	AAB,AA+X	Data Low	1/0
	(5 and 6 cycles)	(1)	5a.	1	1	1	0	AAB,AA+X+1	Data High	1/0
14.	Absolute,Y a,y		1.	1	1	1	1	PBR,PC	Op Code	1
	(LDX,ORA,AND,EOR,ADC,		2.	1	1	0	1	PBR,PC+1	AAL	1
	STA,LDA,CMP,SBC)		3.	1	1	0	1	PBR,PC+2	AAH	1
	(9 Op Codes)	(4)	3a.	1	1	0	0	DBR,AAH,AAL+Y	YL IO	1
	(3 bytes)		4.	1	1	1	0	DBR,AA+Y	Data Low	1/0
	(4,5 and 6 cycles)	(1)	4a.	1	1	1	0	DBR,AA+Y+1	Data High	1/0
15.	Relative r		1.	1	1	1	1	PBR,PC	Op Code	1
	(BPL,BMI,BVC,BVS,BCC,		2.	1	1	0	1	PBR,PC+1	Offset	1
	BCS,BNE,BEQ,BRA)	(5)	2a.	1	1	0	0	PBR,PC+1	IO	1
	(9 Op Codes)	(6)	2b.	1	1	0	0	PBR,PC+1	IO	1
	(2 bytes)		1.	1	1	1	1	PBR,PC+Offset	Op Code	1
	(2,3 and 4 cycles)									
*16.	Relative Long rl		1.	1	1	1	1	PBR,PC	Op Code	1
	(BRL)		2.	1	1	0	1	PBR,PC+1	Offset Low	1
	(1 Op Code)		3.	1	1	0	1	PBR,PC+2	Offset High	1
	(3 bytes)		4.	1	1	0	0	PBR,PC+2	IO	1
	(4 cycles)		1.	1	1	1	1	PBR,PC+Offset	Op Code	1
17a.	Absolute Indirect (a)		1.	1	1	1	1	PBR,PC	Op Code	1
	(JMP)		2.	1	1	0	1	PBR,PC+1	AAL	1
	(1 Op Code)		3.	1	1	0	1	PBR,PC+2	AAH	1
	(3 bytes)		4.	1	1	1	0	0,AA	NEW PCL	1
	(5 cycles)		5.	1	1	1	0	0,AA+1	NEW PCH	1
			1.	1	1	1	1	PBR,NEW PC	Op Code	1
*17b.	Absolute Indirect (a)		1.	1	1	1	1	PBR,PC	Op Code	1
			2.	1	1	0	1	PBR,PC+1	AAL	1
	(JML)		3.	1	1	0	1	PBR,PC+2	AAH	1
	(1 Op Code)		4.	1	1	1	0	0,AA	NEW PCL	1
	(3 bytes)		5.	1	1	1	0	0,AA+1	NEW PCH	1
	(6 cycles)		6.	1	1	1	0	0,AA+2	NEW PBR	1
			1.	1	1	1	1	NEW PBR,PC	Op Code	1
• 18.	Direct Indirect (d)		1.	1	1	1	1	PBR,PC	Op Code	1
	(ORA,AND,EOR,ADC,		2.	1	1	0	1	PBR,PC+1	DO	1
	STA,LDA,CMP,SBC)	(2)	2a.	1	1	0	0	PBR,PC+1	IO	1
	(8 Op Codes)		3.	1	1	1	0	0,D+DO	AAL	1
	(2 bytes)		4.	1	1	1	0	0,D+DO+1	AAH	1
	(5,6 and 7 cycles)		5.	1	1	1	0	DBR,AA	Data Low	1/
		(1)	5a.	1	1	1	0	DBR,AA+1	Data Low	

Detailed Instruction Operation (continued)

	ADDRESS MODE		CYCLE	VP	ML	VDA	VPA	ADDRESS BUS	DATA BUS	R/W
*19	Direct Indirect Long [d]		1.	1	1	1	1	PBR,PC	Op Code	1
	(ORA,AND,EOR,ADC		2.	1	1	0	1	PBR,PC+1	DO	1
	STA,LDA,CMP,SBC)	(2)	2a.	1	1	0	0	PBR,PC+1	IO	1
	(8 Op Codes)		3.	1	1	1	0	0,D+DO	AAL	1
	(2 bytes)		4.	1	1	1	0	0,D+DO+1	AAH	1
	(6,7 and 8 cycles)		5.	1	1	1	0	0,D+DO+2	AAB	1
			6.	1	1	1	0	AAB,AA	Data Low	1/0
		(1)	6a.	1	1	1	0	AAB,AA+1	Data High	1/0
20a.	Absolute Indexed Indirect (a,x)		1.	1	1	1	1	PBR,PC	Op Code	1
	(JMP)		2.	1	1	0	1	PBR,PC+1	AAL	1
	(1 Op Code)		3.	1	1	0	1	PBR,PC+2	AAH	1
	(3 bytes)		4.	1	1	0	0	PBR,PC+2	IO	1
	(6 cycles)		5.	1	1	0	1	PBR,AA+X	NEW PCL	1
			6.	1	1	0	1	PBR,AA+X+1	NEW PCH	1
			1.	1	1	1	1	PBR, NEW PC	Op Code	1
*20b	Absolute Indexed Indirect		1.	1	1	1	1	PBR,PC	Op Code	1
	(Jump to Subroutine Indexed		2.	1	1	0	1	PBR,PC+1	AAL	1
	Indirect) (a,x)		3.	1	1	1	0	0,S	PCH	0
	(JSR)		4.	1	1	1	0	0,S-1	PCL	0
	(1 Op Code)		5.	1	1	0	1	PBR,PC+2	AAH	1
	(3 bytes)		6.	1	1	0	0	PBR,PC+2	IO	1
	(8 cycles)		7.	1	1	0	1	PBR,AA+X	NEW PCL	1
			8.	1	1	0	1	PBR,AA+X+1	NEW PCH	1
			1.	1	1	1	1	PBR,NEW PC	Next Op Code	1
21a.	Stack (Hardware		1.	1	1	1	1	PBR,PC	IO	1
	Interrupts) s	(3)	2.	1	1	0	0	PBR,PC	IO	1
	(IRQ,NMI,ABORT,RES)	(7)	3.	1	1	1	0	0,S	PBR	0
	(4 hardware interrupts)		4.	1	1	1	0	0,S-1	PCH	0
	(0 bytes)		5.	1	1	1	0	0,S-2	PCL	0
	(7 and 8 cycles)		6.	1	1	1	0	0,S-3	P	0
			7.	0	1	1	0	0,VA	AAVL	1
			8.	0	1	1	0	0,VA+1	AAVH	1
			1.	1	1	1	1	0,AAV	Next Op Code	1
21b.	Stack (Software		1.	1	1	1	1	PBR,PC	Op Code	1
	Interrupts) s	(3)	2.	1	1	0	1	PBR,PC+1	Signature	1
	(BRK,COP)	(7)	3.	1	1	1	0	0,S	PBR	0
	(2 Op Codes)		4.	1	1	1	0	0,S-1	PCH	0
	(2 bytes)		5.	1	1	1	0	0,S-2	PCL	0
	(7 and 8 cycles)		6.	1	1	1	0	0,S-3 (COP Latches)	P	0
			7.	0	1	1	0	0,VA	AAVL	1
			8.	0	1	1	0	0,VA+1	AAVH	1
			1.	1	1	1	1	0,AAV	Next Op Code	1
21c.	Stack (Return from		1.	1	1	1	1	PBR,PC	Op Code	1
	Interrupt) s		2.	1	1	0	0	PBR,PC+1	IO	1
	(RTI)	(3)	3.	1	1	0	0	PBR,PC+1	IO	1
	(1 Op Code)		4.	1	1	1	0	0,S+1	P	1
	(1 byte)		5.	1	1	1	0	0,S+2	PCL	1
	(6 and 7 cycles)		6.	1	1	1	0	0,S+3	PCH	1
	(different order from N6502)	(7)	7.	1	1	1	0	0,S+4	PBR	1
			1.	1	1	1	1	PBR,PC	New Op Code	1
21d.	Stack (Return from		1.	1	1	1	1	PBR,PC	Op Code	1
	Subroutine) s		2.	1	1	0	0	PBR,PC+1	IO	1
	(RTS)		3.	1	1	0	0	PBR,PC+1	IO	1
	(1 Op Code)		4.	1	1	1	0	0,S+1	PCL	1
	(1 byte)		5.	1	1	1	0	0,S+2	PCH	1
	(6 cycles)		6.	1	1	0	0	0,S+2	IO	1
			1.	1	1	1	1	PBR,PC	Op Code	1
*21e.	Stack (Return from		1.	1	1	1	1	PBR,PC	Op Code	1
	Subroutine Long) s		2.	1	1	0	0	PBR,PC+1	IO	1
	(RTL)		3.	1	1	0	0	PBR,PC+1	IO	1
	(1 Op Code)		4.	1	1	1	0	0,S+1	NEW PCL	1
	(1 byte)		5.	1	1	1	0	0,S+2	NEW PCH	1
	(6 cycles)		6.	1	1	1	0	0,S+3	NEW PBR	1
			1.	1	1	1	1	NEW PBR,PC	Next Op Code	1
21f.	Stack (Push) s		1.	1	1	1	1	PBR,PC	Op Code	1
	(PHP,PHA,PHY,PHX,		2.	1	1	0	0	PBR,PC+1	IO	1
	PHD,PHK,PHB)	(1)	3a.	1	1	1	0	0,S	Register High	1
	(7 Op Codes)		3.	1	1	1	0	0,S-1	Register Low	1
	(1 byte)									
	(3 and 4 cycles)									
21g.	Stack (Pull) s		1.	1	1	1	1	PBR,PC	Op Code	1
	(PLP,PLA,PLY,PLX,PLD,PLB)		2.	1	1	0	0	PBR,PC+1	IO	1
	(Different than N6502)		3.	1	1	0	0	PBR,PC+1	IO	1
	(6 Op Codes)		4.	1	1	1	0	0,S+1	Register Low	1
	(1 byte)	(1)	4a.	1	1	1	0	0,S+2	Register High	1
	(4 and 5 cycles)									
*21h.	Stack (Push Effective		1.	1	1	1	1	PBR,PC	Op Code	1
	Indirect Address) s		2.	1	1	0	1	PBR,PC+1	DO	1
	(PEI)	(2)	2a.	1	1	0	0	PBR,PC+1	IO	1
	(1 Op Code)		3.	1	1	1	0	0,D+DO	AAL	1
	(2 bytes)		4.	1	1	1	0	0,D+DO+1	AAH	1
	(6 and 7 cycles)		5.	1	1	1	0	0,S	AAH	0
			6.	1	1	1	0	0,S-1	AAL	0
*21i.	Stack (Push Effective		1.	1	1	1	1	PBR,PC	Op Code	1
	Absolute Address) s		2.	1	1	0	1	PBR,PC+1	AAL	1
	(PEA)		3.	1	1	0	1	PBR,PC+2	AAH	1
	(1 Op Code)		4.	1	1	1	0	0,S	AAH	0
	(3 bytes)		5.	1	1	1	0	0,S-1	AAL	0
	(5 cycles)									
*21j.	Stack (Push Effective		1.	1	1	1	1	PBR,PC	Op Code	1
	Program Counter Relative		2.	1	1	0	1	PBR,PC+1	Offset Low	1
	Address) s		3.	1	1	0	1	PBR,PC+2	Offset High	1
	(PER)		4.	1	1	0	0	PBR,PC+2	IO	1
	(1 Op Code)		5.	1	1	1	0	0,S	PCH+OFF+ CARRY	0
	(3 bytes)									
	(6 cycles)		6.	1	1	1	0	0,S-1	PCL+OFFSET	0
*22.	Stack Relative d,s		1.	1	1	1	1	PBR,PC	Op Code	1
	(ORA,AND,EOR,ADL,		2.	1	1	0	1	PBR,PC+1	SO	1
	STA,LDA,CMP,SDC)		3.	1	1	0	0	PBR,PC+1	IO	1
	(8 Op Codes)		4.	1	1	1	0	0,S+SO	Data Low	1/0
	(2 bytes)	(1)	4a.	1	1	1	0	0,S+SO+1	Data High	1/0
	(4 and 5 cycles)									

	ADDRESS MODE		CYCLE	VP	ML	VDA	VPA	ADDRESS BUS	DATA BUS	R/W
*23.	Stack Relative Indirect		1.	1	1	1	1	PBR,PC	Op Code	1
	Indexed (d,s),y		2.	1	1	0	1	PBR,PC+1	SO	1
	(ORA,AND,EOR,ADC,		3.	1	1	0	0	PBR+PC+1	IO	1
	STA,LDA,CMP,SDC)		4.	1	1	1	0	0,S+SO	AAL	1
	(8 Op Codes)		5.	1	1	1	0	0,S+SO+1	AAH	1
	(2 bytes)		6.	1	1	0	0	0,S+SO+1	IO	1
	(7 and 8 Cycles)		7.	1	1	0	0	DBR,AA+Y	Data Low	1/0
		(1)	7a.	1	1	1	0	DBR,AA+Y+1	Data High	1/0
*24a.	Block Move Positive		1.	1	1	1	1	PBR,PC	Op Code	1
	(forward) xyc		2.	1	1	0	1	PBR,PC+1	DBA	1
	(MVP)		3.	1	1	0	1	PBR,PC+2	SBA	1
	(1 Op Code)	N-2	4.	1	1	1	0	SBA,X	Source Data	1
	(3 bytes)	Byte	5.	1	1	1	0	DBA,Y	Dest. Data	0
	(7 cycles)	C=2	6.	1	1	0	0	DBA,Y	IO	1
	x = Source Address		7.	1	1	0	0	DBA,Y	IO	1
	y = Destination		1.	1	1	1	1	PBR,PC	Op Code	1
	c =Number of Bytes to Move -1		2.	1	1	0	1	PBR,PC+1	DBA	1
	x,y Decrement		3.	1	1	0	1	PBR,PC+2	SBA	1
	MVP is used when the	N-1	4.	1	1	1	0	SBA,X-1	Source Data	1
	destination start address	Byte	5.	1	1	1	0	DBA,Y-1	Dest. Data	0
	is higher (more positive)	C=1	6.	1	1	0	0	DBA,Y-1	IO	1
	than the source start address.		7.	1	1	0	0	DBA,Y-1	IO	1
	FFFFFF		1.	1	1	1	1	PBR,PC	Op Code	1
	Dest. Start	N Byte	2.	1	1	0	1	PBR,PC+1	DBA	1
	Dest. End	Last	3.	1	1	0	1	PBR,PC+2	SBA	1
	Source Start	C=0	4.	1	1	1	0	SBA,X-2	Source Data	1
	Source End		5.	1	1	1	0	DBA,Y-2	Dest. Data	0
	000000		6.	1	1	0	0	DBA,Y-2	IO	1
			7.	1	1	0	0	DBA,Y-2	IO	1
			1.	1	1	1	1	PBR,PC+3	Next Op Code	1
*24b.	Block Move Negative		1.	1	1	1	1	PBR,PC	Op Code	1
	(backward) xyc		2.	1	1	0	1	PBR,PC+1	DBA	1
	(MVN)		3.	1	1	0	1	PBR,PC+2	SBA	1
	(1 Op Code)	N-2	4.	1	1	1	0	SBA,X	Source Data	1
	(3 bytes)	Byte	5.	1	1	1	0	DBA,Y	Dest. Data	0
	(7 cycles)	C=2	6.	1	1	0	0	DBA,Y	IO	1
	x = Source Address		7.	1	1	0	0	DBA,Y	IO	1
	y = Destination		1.	1	1	1	1	PBR,PC	Op Code	1
	c = Number of Bytes to Move -1		2.	1	1	0	1	PBR,PC+1	DBA	1
	x,y Increment		3.	1	1	0	1	PBR,PC+2	SBA	1
		N-1	4.	1	1	1	0	SBA,X+1	Source Data	1
	FFFFFF	Byte	5.	1	1	1	0	DBA,Y+1	Dest. Data	0
	Source End	C=1	6.	1	1	0	0	DBA,Y+1	IO	1
	Dest.End		7.	1	1	0	0	DBA,Y+1	IO	1
	Source Start		1.	1	1	1	1	PBR,PC	Op Code	1
	Dest. Start		2.	1	1	0	1	PBR,PC+1	DBA	1
		N Byte	3.	1	1	0	1	PBR,PC+2	SBA	1
	000000	C=0	4.	1	1	1	0	SBA,X+2	Source Data	1
	MVN is used when the		5.	1	1	1	0	DBA,Y+2	Dest. Data	0
	destination start address		6.	1	1	0	0	DBA,Y+2	IO	1
	is lower (more negative)		7.	1	1	0	0	DBA,Y+2	IO	1
	than the source start		1.	1	1	1	1	PBR,PC+3	Next Op Code	1
	address.									

Notes:

(1) Add 1 byte (for immediate only) for M=0 or X=0 (i.e. 16 bit data), add 1 cycle for M=0 or X=0.

(2) Add 1 cycle for direct register low (DL) not equal 0.

(3) Special case for aborting instruction. This is the last cycle which may be aborted or the Status, PBR or DBR registers will be updated.

(4) Add 1 cycle for indexing across page boundaries, or write, or X=0. When X=1 or in the emulation mode, this cycle contains invalid addresses.

(5) Add 1 cycle if branch is taken.

(6) Add 1 cycle if branch is taken across page boundaries in 6502 emulation mode (E=1).

(7) Subtract 1 cycle for 6502 emulation mode (E=1).

(8) Add 1 cycle for REP,SEP.

(9) Wait at cycle 2 for 2 cycles after $\overline{NMI}$ or $\overline{IRQ}$ active input.

Abbreviations:

AAB Absolute Address Bank
AAH Absolute Address High
AAL Absolute Address Low
AAVH Absolute Address Vector High
AAVL Absolute Address Vector Low
C Accumulator
D Direct Register
DBA Destination Bank Address
DBR Data Bank Register
DO Direct Offset
IDH Immediate Data High
IDL Immediate Data Low
IO Internal Operation
P Status Register
PBR Program Bank Register
PC Program Counter
R-M-W Read-Modify-Write
S Stack Address
SBA Source Bank Address
SO Stack Offset
VA Vector Address
x,y Index Registers
★ = New W65C816/802 Addressing Modes
● = New W65C02 Addressing Modes
Blank = NMOS 6502 Addressing Modes

Recommended W65C816 and W65C802 Assembler Syntax Standards

Directives

Assembler directives are those parts of the assembly language source program which give directions to the assembler; this includes the definition of data area and constants within a program. This standard excludes any definitions of assembler directives.

Comments

An assembler should provide a way to use any line of the source program as a comment. The recommended way of doing this is to treat any blank line, or any line that starts with a semi-colon or an asterisk as a comment. Other special characters may be used as well.

The Source Line

Any line which causes the generation of a single W65C816 or W65C802 machine language instruction should be divided into four fields: a label field, the operation code, the operand, and the comment field.

The Label Field—The label field begins in column one of the line. A label must start with an alphabetic character, and may be followed by zero or more alphanumeric characters. An assembler may define an upper limit on the number of characters that can be in a label, so long as that upper limit is greater than or equal to six characters. An assembler may limit the alphabetic characters to upper-case characters if desired. If lower-case characters are allowed, they should be treated as identical to their upper-case equivalents. Other characters may be allowed in the label, so long as their use does not conflict with the coding of operand fields.

The Operation Code Field—The operation code shall consist of a three character sequence (mnemonic) from Table 3. It shall start no sooner than column 2 of the line, or one space after the label if a label is coded.

Many of the operation codes in Table 3 have duplicate mnemonics; when two or more machine language instructions have the same mnemonic, the assembler resolves the difference based on the operand.

If an assembler allows lower-case letters in labels, it must also allow lower-case letters in the mnemonic. When lower-case letters are used in the mnemonic, they shall be treated as equivalent to the upper-case counterpart. Thus, the mnemonics LDA, lda, and LdA must all be recognized, and are equivalent.

In addition to the mnemonics shown in Table 3, an assembler may provide the alternate mnemonics shown in Table 6.

Alternate Mnemonics

Standard	Alias
BCC	BLT
BCS	BGE
CMP A	CMA
DEC A	DEA
INC A	INA
JSL	JSR
JML	JMP
TCD	TAD
TCS	TAS
TDC	TDA
TSC	TSA
XBA	SWA

JSL should be recognized as equivalent to JSR when it is specified with a long absolute address. JML is equivalent to JMP with long addressing forced.

The Operand Field—The operand field may start no sooner than one space after the operation code field. The assembler must be capable of at least twenty-four bit address calculations. The assembler should be capable of specifying addresses as labels, integer constants, and hexadecimal constants. The assembler must allow addition and subtraction in the operand field. Labels shall be recognized by the fact that they start alphabetic characters. Decimal numbers shall be recognized as containing only the decimal digits 0...9. Hexadecimal constants shall be recognized by prefixing the constant with a "$" character, followed by zero or more of either the decimal digits or the hexadecimal digits "A"..."F". If lower-case letters are allowed in the label field, then they shall also be allowed as hexadecimal digits.

All constants, no matter what their format, shall provide at least enough precision to specify all values that can be represented by a twenty-four bit signed or unsigned integer represented in two's complement notation.

Table 8 shows the operand formats which shall be recognized by the assembler. The symbol **d** is a label or value which the assembler can recognize as being less than $100. The symbol **a** is a label or value which the assembler can recognize as greater the $FF but less than $10000; the symbol **al** is a label or value that the assembler can recognize as being greater than $FFFF. The symbol EXT is a label which cannot be located by the assembler at the time the instruction is assembled. Unless instructed otherwise, an assembler shall assume that EXT labels are two bytes long. The symbols **r** and **rl** are 8 and 16 bit signed displacements calculated by the assembler.

Note that the operand does not determine whether or not immediate addressing loads one or two bytes; this is determined by the setting of the status register. This forces the requirement for a directive or directives that tell the assembler to generate one or two bytes of space for immediate loads. The directives provided shall allow separate settings for the accumulator and index registers.

The assembler shall use the <, >, and ^ characters after the # character in immediate address to specify which byte or bytes will be selected from the value of the operand. Any calculations in the operand must be performed before the byte selection takes place. Table 7 defines the action taken by each operand by showing the effect of the operator on an address. The column that shows a two byte immediate value show the bytes in the order in which they appear in memory. The coding of the operand is for an assembler which uses 32 bit address calculations, showing the way that the address should be reduced to a 24 bit value.

Byte Selection Operator

Operand	One Byte Result	Two Byte Result
#$01020304	04	04 03
#<$01020304	04	04 03
#>$01020304	03	03 02
#^$01020304	02	02 01

In any location in an operand where an address, or expression resulting in an address, can be coded, the assembler shall recognize the prefix characters <, |, and >, which force one byte (direct page), two byte (absolute) or three byte (long absolute) addressing. In cases where the addressing mode is not forced, the assembler shall assume that the address is two bytes unless the assembler is able to determine the type of addressing required by context, in which case that addressing mode will be used. Addresses shall be truncated without error if an addressing mode is forced which does not require the entire value of the address. For example,

```
LDA  $0203      LDA  |$010203
```

are completely equivalent. If the addressing mode is not forced, and the type of addressing cannot be determined from context, the assembler shall assume that a two byte address is to be used. If an instruction does not have a short addressing mode (as in LDA, which has no direct page indexed by Y) and a short address is used in the operand, the assembler shall automatically extend the address by padding the most significant bytes with zeroes in order to extend the address to the length needed. As with immediate addressing, any expression evaluation shall take place before the address is selected; thus, the address selection character is only used once, before the address of expression.

The ! (exclamation point) character should be supported as an alternative to the | (vertical bar).

A long indirect address is indicated in the operand field of an instruction by surrounding the direct page address where the indirect address is found by square brackets; direct page addresses which contain sixteen-bit addresses are indicated by being surrounded by parentheses.

The operands of a block move instruction are specified as source bank, destination bank—the opposite order of the object bytes generated.

Comment Field—The comment field may start no sooner than one space after the operation code field or operand field depending on instruction type.

Address Mode Formats

Addressing Mode	Format
Immediate	#d
	#a
	#al
	#EXT
	#<d
	#<a
	#<al
	#<EXT
	#>d
	#>a
	#>al
	#>EXT
	#^d
	#^a
	#^al
	#^EXT
Absolute	!d
	!a
	a
	!al
	!EXT
	EXT
Absolute Long	>d
	>a
	>al
	al
	>EXT
Direct Page	d
	<d
	<a
	<al
	<EXT
Accumulator	A
Implied Addressing	(no operand)
Direct Indirect Indexed	(d),y
	(<d),y
	(<a),y
	(<al),y
	(<EXT),y
Direct Indirect Indexed Long	[d],y
	[<d],y
	[<a],y
	[<al],y
	[<EXT],y
Direct Indexed Indirect	(d,x)
	(<d,x)
	(<a,x)
	(<al,x)
	(<EXT,x)
Direct Indexed by X	d,x
	<d,x
	<a,x
	<al,x
	<EXT,x
Direct Indexed by Y	d,y
	<d,y
	<a,y
	<al,y
	<EXT,y
Absolute Indexed by X	d,x
	!d,x
	a,x
	!a,x
	!al,x
	!EXT,x
	EXT,x

Addressing Mode	Format	
Absolute Indexed by Y	!d,y	
	d,y	
	a,y	
	!a,y	
	!al,y	
	!EXT,y	
	EXT,y	
Absolute Long Indexed by X	>d,x	
	>a,x	
	>al,x	
	al,x	
	>EXT,x	
Program Counter Relative and Program Counter Relative Long	d	(the assembler calculates r and rl)
	a	
	al	
	EXT	
Absolute Indirect	(d)	
	(!d)	
	(a)	
	(!a)	
	(!al)	
	(EXT)	
Direct Indirect	(d)	
	(<a)	
	(<al)	
	(<EXT)	
Direct Indirect Long	[d]	
	[<a]	
	[<al]	
	[<EXT]	
Absolute Indexed	(d,x)	
	(!d,x)	
	(a,x)	
	(!a,x)	
	(!al,x)	
	(EXT,x)	
	(!EXT,x)	
Stack Addressing	(no operand)	
Stack Relative Indirect Indexed	(d,s),y	
	(<d,s),y	
	(<a,s),y	
	(<al,s),y	
	(<EXT,s),y	
Block Move	d,d	
	d,a	
	d,al	
	d,EXT	
	a,d	
	a,a	
	a,al	
	a,EXT	
	al,d	
	al,a	
	al,al	
	al,EXT	
	EXT,d	
	EXT,a	
	EXT,al	
	EXT,EXT	

Note: The alternate ! (exclamation point) is used in place of the | (vertical bar).

Addressing Mode Summary

Address Mode	Instruction Times In Memory Cycles		Memory Utilization In Number of Program Sequence Bytes	
	Original 8 Bit NMOS 6502	New W65C816	Original 8 Bit NMOS 6502	New W65C816
1. Immediate	2	2(3)	2	2(3)
2. Absolute	4(5)	4(3,5)	3	3
3. Absolute Long	—	5(3)	—	4
4. Direct	3(5)	3(3,4,5)	2	2
5. Accumulator	2	2	1	1
6. Implied	2	2	1	1
7. Direct Indirect Indexed (d),y	5(1)	5(1,3,4)	2	2
8. Direct Indirect Indexed Long [d], y	—	6(3,4)	—	2
9. Direct Indexed Indirect (d,x)	6	6(3,4)	2	2
10. Direct, X	4(5)	4(3,4,5)	2	2
11. Direct, Y	4	4(3,4)	2	2
12. Absolute, X	4(1,5)	4(1,3,5)	3	3
13. Absolute Long, X	—	5(3)	—	4
14. Absolute, Y	4(1)	4(1,3)	3	3
15. Relative	2(1,2)	2(2)	2	2
16. Relative Long	—	3(2)	—	3
17. Absolute Indirect (Jump)	5	5	3	3
18. Direct Indirect	—	5(3,4)	—	2
19. Direct Indirect Long	—	6(3,4)	—	2
20. Absolute Indexed Indirect (Jump)	—	6	—	3
21. Stack	3-7	3-8	1-3	1-4
22. Stack Relative	—	4(3)	—	2
23. Stack Relative Indirect Indexed	—	7(3)	—	2
24. Block Move X, Y, C (Source, Destination, Block Length)	—	7	—	3

NOTES:
1. Page boundary, add 1 cycle if page boundary is crossed when forming address.
2. Branch taken, add 1 cycle if branch is taken.
3. M = 0 or X = 0, 16 bit operation, add 1 cycle, add 1 byte for immediate.
4. Direct register low (DL) not equal zero, add 1 cycle.
5. Read-Modify-Write, add 2 cycles for M = 1, add 3 cycles for M = 0.

Caveats and Application Information

Stack Addressing

When in the Native mode, the Stack may use memory locations 000000 to 00FFFF. The effective address of Stack, Stack Relative, and Stack Relative Indirect Indexed addressing modes will always be within this range. In the Emulation mode, the Stack address range is 000100 to 0001FF. The following opcodes and addressing modes will increment or decrement beyond this range when accessing two or three bytes:

JSL; JSR(a,x); PEA; PEI; PER; PHD; PLD; RTL; d,s; (d,s),y

Direct Addressing

The Direct Addressing modes are often used to access memory registers and pointers. The effective address generated by Direct; Direct,X and Direct,Y addressing modes will always be in the Native mode range 000000 to 00FFFF. When in the Emulation mode, the direct addressing range is 000000 to 0000FF, except for [Direct] and [Direct],Y addressing modes and the PEI instruction which will increment from 0000FE or 0000FF into the Stack area.

When in the Emulation mode and DH is not equal to zero, the direct addressing range is 00DH00 to 00DHFF, except for [Direct] and [Direct],Y addressing modes and the PEI instruction which will increment from 00DHFE or 00DHFF into the next higher page.

When in the Emulation mode and DL in not equal to zero, the direct addressing range is 000000 to 00FFFF.

Absolute Indexed Addressing (W65C816 Only)

The Absolute Indexed addressing modes are used to address data outside the direct addressing range. The W65C02 and W65C802 addressing range is 0000 to FFFF. Indexing from page FFXX may result in a 00YY data fetch when using the W65C02 or W65C802. In contrast, indexing from page ZZFFXX may result in ZZ+1,00YY when using the W65C816.

Future Microprocessors (i.e., W65C832)

Future WDC microprocessors will support all current W65C816 operating modes for both index and offset address generation.

$\overline{\text{ABORT}}$ Input (W65C816 Only)

$\overline{\text{ABORT}}$ should be held low for a period not to exceed one cycle. Also, if $\overline{\text{ABORT}}$ is held low during the Abort Interrupt sequence, the Abort Interrupt will be aborted. It is not recommended to abort the Abort Interrupt. The $\overline{\text{ABORT}}$ internal latch is cleared during the second cycle of the Abort Interrupt. Asserting the $\overline{\text{ABORT}}$ input after the following instruction cycles will cause registers to be modified:

- **Read-Modify-Write:** Processor status modified if $\overline{\text{ABORT}}$ is asserted after a modify cycle.
- **RTI:** Processor status will be modified if $\overline{\text{ABORT}}$ is asserted after cycle 3.
- **$\overline{\text{IRQ}}$, $\overline{\text{NMI}}$, $\overline{\text{ABORT}}$ BRK, COP:** When $\overline{\text{ABORT}}$ is asserted after cycle 2, PBR and DBR will become 00 (Emulation mode) or PBR will become 00 (Native mode).

The Abort Interrupt has been designed for virtual memory systems. For this reason, asynchronous $\overline{\text{ABORT}}$'s may cause undesirable results due to the above conditions.

VDA and VPA Valid Memory Address Output Signals (W65C816 Only)

When VDA or VPA are high and during all write cycles, the Address Bus is always valid. VDA and VPA should be used to qualify all memory cycles. Note that when VDA and VPA are both low, invalid addresses may be generated. The Page and Bank addresses could also be invalid. This will be due to low byte addition only. The cycle when only low byte addition occurs is an optional cycle for instructions which read memory when the Index Register consists of 8 bits. This optional cycle becomes a standard cycle for the Store instruction, all instructions using the 16-bit Index Register mode, and the Read-Modify-Write instruction when using 8- or 16-bit Index Register modes.

Apple II, IIe, IIc and II+ Disk Systems (W65C816 Only)

VDA and VPA should not be used to qualify addresses during disk operation on Apple systems. Consult your Apple representative for hardware/software configurations.

DB/BA Operation when RDY is Pulled Low (W65C816 Only)

When RDY is low, the Data Bus is held in the data transfer state (i.e., $\phi 2$ high). The Bank address external transparent latch should be latched when the $\phi 2$ clock or RDY is low.

M/X Output (W65C816 Only)

The M/X output reflects the value of the M and X bits of the processor Status Register. The REP, SEP and PLP instructions may change the state of the M and X bits. Note that the M/X output is invalid during the instruction cycle following REP, SEP and PLP instruction execution. This cycle is used as the opcode fetch cycle of the next instruction.

All Opcodes Function in All Modes of Operation

It should be noted that all opcodes function in all modes of operation. However, some instructions and addressing modes are intended for W65C816 24-bit addressing and are therefore less useful for the W65C802. The following is a list of instructions and addressing modes which are primarily intended for W65C816 use:

JSL; RTL; [d]; [d],y; JMP al; JML; al; al,x

The following Instructions may be used with the W65C802 even though a Bank Address is not multiplexed on the Data Bus:

PHK; PHB; PLB

The following instructions have "limited" use in the Emulation mode:

- The REP and SEP instructions cannot modify the M and X bits when in the Emulation mode. In this mode the M and X bits will always be high (logic 1).
- When in the Emulation mode, the MVP and MVN instructions use the X and Y Index Registers for the memory address. Also, the MVP and MVN instructions can only move data within the memory range 0000 (Source Bank) to 00FF (Destination Bank) for the W65C816, and 0000 to 00FF for the W65C802.

Indirect Jumps

The JMP (a) and JML (a) instructions use the direct Bank for indirect addressing, while JMP (a,x) and JSR (a,x) use the Program Bank for indirect address tables.

Switching Modes

When switching from the Native mode to the Emulation mode, the X and M bits of the Status Register are set high (logic 1), the high byte of the Stack is set to 01, and the high bytes of the X and Y Index Registers are set to 00. To save previous values, these bytes must always be stored before changing modes. Note that the low byte of the S, X and Y Registers and the low and high byte of the Accumulator (A and B) are not affected by a mode change.

How Hardware Interrupts, BRK, and COP Instructions Affect the Program Bank and the Data Bank Registers

When in the Native mode, the Program Bank register (PBR) is cleared to 00 when a hardware interrupt, BRK or COP is executed. In the Native mode, previous PBR contents is automatically saved on Stack.

In the Emulation mode, the PBR and DBR registers are cleared to 00 when a hardware interrupt, BRK or COP is executed. In this case, previous contents of the PBR are not automatically saved.

Note that a Return from Interrupt (RTI) should always be executed from the same "mode" which originally generated the interrupt.

Binary Mode

The Binary mode is set whenever a hardware or software interrupt is executed. The D flag within the Status Register is cleared to zero.

WAI Instruction

The WAI instruction pulls RDY low and places the processor in the WAI "low power" mode. $\overline{\text{NMI}}$, $\overline{\text{IRQ}}$ or RESET will terminate the WAI condition and transfer control to the interrupt handler routine. Note that an $\overline{\text{ABORT}}$ input will abort the WAI instruction, but will not restart the processor. When the Status Register I flag is set ($\overline{\text{IRQ}}$ disabled), the $\overline{\text{IRQ}}$ interrupt will cause the next instruction (following the WAI instruction) to be executed without going to the $\overline{\text{IRQ}}$ interrupt handler. This method results in the highest speed response to an $\overline{\text{IRQ}}$ input. When an interrupt