



COMPUTER

ORIC

CODING

Ok, I've ripped this html page on the web. I hope that anybody will annoy me for that but I think it's a really great summary of what you can learn of 6502 compatible chips. I hope that mister Tabke will be ok !

By the way, anyone want to do an Oric with a 65816 CPU inside ? Could be cool, since this processor can emulate 6502 code.

At the end, you have a complete 6502/65C02/65816 opcode list, with compatibility notes and timings.

Have fun reading this !

---

## A 6502 Programmer's Introduction to the 65816

by Brett Tabke

After programming in 6502 language for over a decade, I was getting a bit BORED. One can only code the same routines with the same opcodes so many times before the nausea of repetition becomes overpowering. When I heard the news that CMD was building a cartridge based on a 20 MHz 65816 I was overjoyed. For years I've heard those with 65816 bases systems brag about its capabilities. To us old 6502 programmers, the opportunity to program the fabled 65816 is a new lease on life.

The 65816 is an 8-/16-bit register selectable upgrade to the 6502 series processor. With 24 bit addressing of up to 16 Megabytes of RAM, the powerful 65816 is a logical upgrade that leaves 6502 programmers feeling right at home. It is amazing how fast one can adapt to the new processor. It sounds funny to say it, but the only difficulty I have had learning the 65816 is that there are so many options and choices to complete the same task, that it is hard to decide which method is best.

To get started programming the 65816, I would recommend purchasing the book, "Programming the 65816" from The Western Design Center, manufacturer of the 65816. While it is a bit pricey, the sheer quality and content of the 600 page book is worth the money. Rarely, if ever, has there been a CPU manual as thorough and detailed as the Western Design book. If you know 6502 assembly, then Programming the 65816 is probably the only 65816 book you will ever need.

### Getting a Feel for the Modes

The 65816 may be operated in Native mode or 6502 Emulation mode. Emulation mode is a 100% 6502 compatible mode where the whole processor looks and feels like a vintage 6502. Native mode offers 8- or 16-bit user registers and full access to 24-bit addressing.

While in emulation mode, not only are all the 6502 opcodes present in their virgin form, but the new 65816 instructions are also available for usage. In fact, the first lesson to learn about programming the 65816 is that emulation mode is much more

powerful than a stock 6502. The only true difference between emulation mode and our venerable C64's 6510 processor is that unimplemented opcodes will not produce the results expected on the former. Since all 256 of the potential opcodes are now implemented on the 65816, older C64 software that uses previously unimplemented opcodes will produce erratic results.

To select between emulation and native modes, a new phantom hidden emulation bit (E) was added to the status register. Shown in programming models hanging on top of the Carry bit, the emulation bit is only accessible by one instruction. The new instruction (XCE) exchanges the status of the Carry bit and Emulation bit. To move to emulation mode, set the carry and issue an XCE instruction. To move to native mode, clear the carry and issue the XCE instruction.

### My, How Your Index Registers Have Grown!

While in native mode there are two new directly accessible bits present in the status register. The 65816 implements new hardware interrupt vectors which include a new hardware BRK vector in ROM; therefore, the old BRK bit of the status register is no longer needed. The BRK bit is replaced with the X bit to select either 8- or 16-bit index registers. The former empty bit 5 is now filled with the M bit to specify the accumulator and memory access as 8- or 16-bit.

Two new instructions are used to clear or set bits within the status register. The SEP instruction sets bits, and REP clears bits. SEP and REP use a one byte immediate addressing mode operand to specify which bits are to be set or cleared. For example, to set the X bit for 8 bit user registers:

```
SEP #%00010000 ; set bit 4 for 8-bit index
                ; registers.
```

Or to clear bit 4:

```
REP #%00010000 ; clear bit 4 for 16-bit index
                ; registers.
```

When in 8 bit mode, the index registers perform their function in standard 6502 form. When status bit X is set to 0, both the X and Y index registers become 16 bits wide. With a 16-bit index register you can now reach out to a full 64K with the various indexed addressing modes. An absolute load to an index register in 16-bit mode will retrieve 2 bytes of memory-the one at the effective address and the one at the effective address plus one. Simple things like INX or DEY work on a full 16 bit, which means you no longer have to specify a memory location for various counters, and loops based on index counters can now be coded in a more efficient manner.

The formerly empty status register bit 5 is now referred to as bit M. M is used to specify an 8- or 16-bit wide accumulator and memory accesses. When in 8 bit mode,

(M=1), the high order 8 bits are still accessible by exchanging the low and high bytes with a XBA instruction-it is like having two accumulators! However; when set for full 16-bit wide accumulator, all math and accumulator oriented logical instructions operate on all 16 bits! If you add up the clock cycles and bytes required to perform a standard two byte addition, you can start to see the true power of 16-bit registers.

## More Register Improvements

Zero Page has now been renamed to Direct Page-corporate thinking, go figure. A new processor register D was added to allow Direct Page to be moved anywhere within the first 64K of memory. The direct page register is 16 bits wide, so you can now specify the start of direct page at any byte. Several old instructions now include direct page addressing as well. To move direct page, just push the new value onto the stack (16 bits) and then PLD to pull it into the direct page register. You may also transfer the value from the 16-bit accumulator to the direct page register with the TCD instruction. Direct page may also be moved while in emulation mode.

While in native mode, the stack pointer is a full 16 bits wide, which means the stack is no longer limited to just 256 bytes. It can be moved anywhere within the first 64K of memory (although while in emulation mode, the stack is located at page one). There are several new addressing modes that can use the stack pointer as a quasi-index register to access memory. Numerous new push and pull instructions allow you to manipulate the stack. A few of the more useful stack instructions useful to programmers, are the new instructions to push & pull index registers with PHX/PHY and PLX/PLY.

Two other new processors registers are the Program Bank Register (PBR) and the Data Bank Register (DBR). The Program Bank Register can be thought of as extending the program counter out to 24 bits. Although you can JSR and JMP to routines located in other RAM banks, individual routines on the 65816 still must run within a single bank of 64K-there's no automatic rollover from one bank of RAM to the next when executing successive instructions. In this sense, it may help to think of the 65816 processor as a marriage of Commodore's C128 Memory Management Unit (MMU) and an 'enhanced' 6502-a very similar concept.

The Data Bank Register is used to reach out to any address within the 16 megabyte address space of the 65816. When any of the addressing modes that specify a 16-bit address are used, the Data Bank byte is appended to the instruction address. This allows access to all 16 megabytes without having to resort to 24-bit addressing instruction, and helps enable code that can operate from any bank.

## New Addressing Modes

There are new addressing modes on the 65816. Several new instructions are designed to help relocatable code that can execute at any address. The use of relocatable code on the 6502 was extremely limited. With 16 megabytes of address space, writing relocatable code increases the overall utility of the program. To write relocatable code, several new instructions use Program Counter Relative Long addressing. This allows relative branching within a 64K bank of RAM. There's also Stack Relative addressing,

and a push instruction to place the program counter onto the stack, so that a code fragment can pull it back off and can instantly know its execution address.

Another new feature are two Block Move instructions, one for forward MVP and one for backward MVN. Simply load the 16-bit X register with the starting address, the Y index register with the ending address, the accumulator with the number of bytes to move, and issue the MVP or MVN instructions. MVN is for move negative, and MVP is for move positive, so that your moves don't overwrite themselves. Block Moves use two operand bytes: one for the source bank of 64K and one for the destination bank. Memory is moved at the rate of seven clock cycles per byte.

Several new addressing modes are used to access the full address space. A 65816 assembler would decode "long" addressing given this input:

```
LDA $0445F2 ; load byte from $45F2 of RAM
           ; bank 4
LDA $03412F,x; load byte from $412F of bank 3
           ; plus x.
```

Quite a few instructions have been given new addressing modes. How many times have you wanted to do this:

```
LDA ($12) ; load indirect without an
           ; offset.
```

Or how about a table of routine addresses:

```
JSR ($1234,x) ; jump to a subroutine via
               ; indexed indirect addressing!
```

Other fun new instructions:

TXY,TYX	Transfer directly between index registers
BRA	Branch always regardless of status bits
TSB	Test and set any bit of a byte
TRB	Test and reset (clear) any bit of a byte
INC A/DEC A	Increment or decrement the accumulator directly
STZ	Store a zero to any byte

## Summing Up

As you can see, the 65816 opens up a whole new world of programming-it feels like a new lease on life. Of course, it's going to take some time to learn the new processor. But while the 20 MHz speed is a nice perk, I believe that the real power of CMD's new peripheral is indeed the engine under its hood: the 65816-*asuper* CPU!

## Native Mode Options

While in Native Mode, the m flag controls the size of Accumulator A and most Memory Operations, while the x flag controls the size of the X and Y Index Registers. This provides 4 different configuration possibilities, as started below. The REP and SEP instructions are used in combination to swith configurations.

m	x	A/M	X/Y	Instructions
0	0	15-bit	16-bit	REP #30
0	1	16-bit	8-bit	REP #20 SEP #10
1	0	8-bit	16-bit	REP #10 SEP #20
1	1	8-bit	8-bit	SEP #30

It is important to note that the m flag will control the size of all operations deahg with memory except in operations involving the X and Y Index Registers (CPX, CPY, LDX, LDY, STX and STY) when the x flag controls the size.

## Emulation Notes

While in Emulation Mode, Accumulator A is forced to 8-bit mode. You can, however, access the upper 8 bits with instructions that specify Accumulator B, and all 16 bits at once with instructions that specify Accumulator C. The X and Y Index Registers are also forced to 8-bit mode, with no means available to access the upper 8 bits. To further assist in compability, the Stack is forced to Page 1 of Bank 0. The Direct page Register (D) is fully functional in this mode, allowing direct page to be placed anywhere in Bank 0. Likewise, the Program Bank Register (PBR) and Data Bank Register (DBR) are also fully functional. While it would seem that these latter items would allow programs to operate from any bank in Emulation mode, there are some caveats; interrups will force the program bank to zero without saving the PBR first, and RTI won't attempt to restore the bank. Therefore, Native mode would be recommended to execute programs in other banks.

### Guide to 6502/65C02/65816 Instructions

[|a|](#) [|b|](#) [|c|](#) [|d|](#) [|e|](#) [|i|](#) [|j|](#) [|l|](#) [|m|](#) [|n|](#) [|o|](#) [|p|](#) [|r|](#) [|s|](#) [|t|](#) [|w|](#) [|x|](#)

Assembler Example	HEX	Addressing Mode	02	C02	816	Bytes	Cycles
<b>ADC</b> <i>Add With Carry</i> [Flags affected: n,v,z,c]							
ADC (dp,X)	61	DP Indexed Indirect,X	x	x	x	2	6 <sup>1,2,4</sup>
ADC sr,S	63	Stack Relative			x	2	4 <sup>1,4</sup>
ADC dp	65	Direct Page	x	x	x	2	3 <sup>1,2,4</sup>
ADC [dp]	67	DP Indirect Long			x	2	6 <sup>1,2,4</sup>
ADC #const	69	Immediate	x	x	x	2 <sup>17</sup>	2 <sup>1,4</sup>
ADC addr	6D	Absolute	x	x	x	3	4 <sup>1,4</sup>
ADC long	6F	Absolute Long			x	4	5 <sup>1,4</sup>
ADC (dp),Y	71	DP Indirect Indexed, Y	x	x	x	2	5 <sup>1,2,3,4</sup>
ADC (dp)	72	DP Indirect		x	x	2	5 <sup>1,2,4</sup>
ADC (sr,S),Y	73	SR Indirect Indexed,Y			x	2	7 <sup>1,4</sup>
ADC dp,X	75	DP Indexed,X	x	x	x	2	4 <sup>1,2,4</sup>
ADC [dp],Y	77	DP Indirect Long Indexed, Y			x	2	6 <sup>1,2,4</sup>
ADC addr,Y	79	Absolute Indexed,Y	x	x	x	3	4 <sup>1,3,4</sup>
ADC addr,X	7D	Absolute Indexed,X	x	x	x	3	4 <sup>1,3,4</sup>
ADC long,X	7F	Absolute Long Indexed,X			x	4	5 <sup>1,4</sup>
<b>AND</b> <i>AND Accumulator With Memory</i> [Flags affected: n,z]							

AND ( <i>dp</i> , X)	21	DP Indexed Indirect,X	x	x	x	2	6 <sup>1,2</sup>
AND <i>sr</i> ,S	23	Stack Relative			x	2	4 <sup>1</sup>
AND <i>dp</i>	25	Direct Page	x	x	x	2	3 <sup>1,2</sup>
AND [ <i>dp</i> ]	27	DP Indirect Long			x	2	6 <sup>1,2</sup>
AND # <i>const</i>	29	Immediate	x	x	x	2 <sup>17</sup>	2 <sup>1</sup>
AND <i>addr</i>	2D	Absolute	x	x	x	3	4 <sup>1</sup>
AND <i>long</i>	2F	Absolute Long			x	4	5 <sup>1</sup>
AND ( <i>dp</i> ),Y	31	DP Indirect Indexed, Y	x	x	x	2	5 <sup>1,2,3</sup>
AND ( <i>dp</i> )	32	DP Indirect		x	x	2	5 <sup>1,2</sup>
AND ( <i>sr</i> ,S),Y	33	SR Indirect Indexed,Y			x	2	7 <sup>1</sup>
AND <i>dp</i> ,X	35	DP Indexed,X	x	x	x	2	4 <sup>1,2</sup>
AND [ <i>dp</i> ],Y	37	DP Indirect Long Indexed, Y			x	2	6 <sup>1,2</sup>
AND <i>addr</i> ,Y	39	Absolute Indexed,Y	x	x	x	3	4 <sup>1,3</sup>
AND <i>addr</i> ,X	3D	Absolute Indexed,X	x	x	x	3	4 <sup>1,3</sup>
AND <i>long</i> ,X	3F	Absolute Long Indexed,X			x	4	5 <sup>1</sup>
<b>ASL</b> <i>Accumulator or Memory Shift Left</i> [Flags affected: n,z,c]							
ASL <i>dp</i>	06	Direct Page	x	x	x	2	5 <sup>2,5</sup>
ASL A	0A	Accumulator	x	x	x	1	2
ASL <i>addr</i>	0E	Absolute	x	x	x	3	6 <sup>5</sup>
ASL <i>dp</i> ,X	16	DP Indexed,X	x	x	x	2	6 <sup>2,5</sup>
ASL <i>addr</i> ,X	1E	Absolute Indexed,X	x	x	x	3	7 <sup>5,6</sup>
<b>BCC</b> <i>Branch if Carry Clear</i> [Flags affected: none][Alias: BLT]							
BCC <i>nearlabel</i>	90	Program Counter Relative	x	x	x	2	2 <sup>7,8</sup>
<b>BCS</b> <i>Branch if Carry Set</i> [Flags affected: none][Alias: BGE]							
BCS <i>nearlabel</i>	B0	Program Counter Relative	x	x	x	2	2 <sup>7,8</sup>
<b>BEQ</b> <i>Branch if Equal</i> [Flags affected: none]							
BEQ <i>nearlabel</i>	F0	Program Counter Relative	x	x	x	2	2 <sup>7,8</sup>
<b>BIT</b> <i>Test Bits</i> [Flags affected: z (immediate mode) n,v,z (non-immediate modes)]							
BIT <i>dp</i>	24	Direct Page	x	x	x	2	3 <sup>1,2</sup>
BIT <i>addr</i>	2C	Absolute	x	x	x	3	4 <sup>1</sup>
BIT <i>dp</i> ,X	34	DP Indexed,X		x	x	2	4 <sup>1,2</sup>
BIT <i>addr</i> ,X	3C	Absolute Indexed,X		x	x	3	4 <sup>1,3</sup>
BIT # <i>const</i>	89	Immediate		x	x	2 <sup>17</sup>	2 <sup>1</sup>
<b>BMI</b> <i>Branch if Minus</i> [Flags affected: none]							
BMI <i>nearlabel</i>	30	Program Counter Relative	x	x	x	2	2 <sup>7,8</sup>
<b>BNE</b> <i>Branch if Not Equal</i> [Flags affected: none]							

BNE <i>nearlabel</i>	D0	Program Counter Relative	x	x	x	2	2 <sup>7,8</sup>
<b>BPL</b> <i>Branch if Plus</i> [Flags affected: none]							
BPL <i>nearlabel</i>	10	Program Counter Relative	x	x	x	2	2 <sup>7,8</sup>
<b>BRA</b> <i>Branch Always</i> [Flags affected: none]							
BRA <i>nearlabel</i>	80	Program Counter Relative		x	x	2	3 <sup>8</sup>
<b>BRK</b> <i>Break</i> [Flags affected: b,i (6502) b,d,i (65C02/65816 Emulation) d,i (65816 Native)]							
BRK	00	Stack/Interrupt	x	x	x	2 <sup>18</sup>	7 <sup>9</sup>
<b>BRL</b> <i>Branch Long Always</i> [Flags affected: none]							
BRL <i>label</i>	82	Program Counter Relative Long			x	3	4
<b>BVC</b> <i>Branch if Overflow Clear</i> [Flags affected: none]							
BVC <i>nearlabel</i>	50	Program Counter Relative	x	x	x	2	2 <sup>7,8</sup>
<b>BVS</b> <i>Branch if Overflow Set</i> [Flags affected: none]							
BVS <i>nearlabel</i>	70	Program Counter Relative	x	x	x	2	2 <sup>7,8</sup>
<b>CLC</b> <i>Clear Carry</i> [Flags affected: c]							
CLC	18	Implied	x	x	x	1	2
<b>CLD</b> <i>Clear Decimal Mode Flag</i> [Flags affected: d]							
CLD	D8	Implied	x	x	x	1	2
<b>CLI</b> <i>Clear Interrupt Disable Flag</i> [Flags affected: i]							
CLI	58	Implied	x	x	x	1	2
<b>CLV</b> <i>Clear Overflow Flag</i> [Flags affected: v]							
CLV	B8	Implied	x	x	x	1	2
<b>CMP</b> <i>Compare Accumulator With Memory</i> [Flags affected: n,z,c]							
CMP ( <i>dp</i> , X)	C1	DP Indexed Indirect, X	x	x	x	2	6 <sup>1,2</sup>
CMP <i>sr</i> , S	C3	Stack Relative			x	2	4 <sup>1</sup>
CMP <i>dp</i>	C5	Direct Page	x	x	x	2	3 <sup>1,2</sup>
CMP [ <i>dp</i> ]	C7	DP Indirect Long			x	2	6 <sup>1,2</sup>
CMP # <i>const</i>	C9	Immediate	x	x	x	2 <sup>17</sup>	2 <sup>1</sup>
CMP <i>addr</i>	CD	Absolute	x	x	x	3	4 <sup>1</sup>
CMP <i>long</i>	CF	Absolute Long			x	4	5 <sup>1</sup>
CMP ( <i>dp</i> ), Y	D1	DP Indirect Indexed, Y	x	x	x	2	5 <sup>1,2,3</sup>
CMP ( <i>dp</i> )	D2	DP Indirect		x	x	2	5 <sup>1,2</sup>
CMP ( <i>sr</i> , S), Y	D3	SR Indirect Indexed, Y			x	2	7 <sup>1</sup>
CMP <i>dp</i> , X	D5	DP Indexed, X	x	x	x	2	4 <sup>1,2</sup>
CMP [ <i>dp</i> ], Y	D7	DP Indirect Long Indexed, Y			x	2	6 <sup>1,2</sup>
CMP <i>addr</i> , Y	D9	Absolute Indexed, Y	x	x	x	3	4 <sup>1,3</sup>

CMP <i>addr</i> ,X	DD	Absolute Indexed,X	x	x	x	3	4 <sup>1,3</sup>
CMP <i>long</i> ,X	DF	Absolute Long Indexed,X			x	4	5 <sup>1</sup>
<b>COP</b> <i>Co-Processor Enable</i> [Flags affected: d,i]							
COP <i>const</i>	02	Stack/Interrupt			x	2 <sup>18</sup>	7 <sup>9</sup>
<b>CPX</b> <i>Compare Index Register X with Memory</i> [Flags affected: n,z,c]							
CPX <i>#const</i>	E0	Immediate	x	x	x	2 <sup>19</sup>	2 <sup>10</sup>
CPX <i>dp</i>	E4	Direct Page	x	x	x	2	3 <sup>2,10</sup>
CPX <i>addr</i>	EC	Absolute	x	x	x	3	4 <sup>10</sup>
<b>CPY</b> <i>Compare Index Register Y with Memory</i> [Flags affected: n,z,c]							
CPY <i>#const</i>	C0	Immediate	x	x	x	2 <sup>19</sup>	2 <sup>10</sup>
CPY <i>dp</i>	C4	Direct Page	x	x	x	2	3 <sup>2,10</sup>
CPY <i>addr</i>	CC	Absolute	x	x	x	3	4 <sup>10</sup>
<b>DEC</b> <i>Decrement</i> [Flags affected: n,z]							
DEC A	3A	Accumulator		x	x	1	2
DEC <i>dp</i>	C6	Direct Page	x	x	x	2	5 <sup>2,5</sup>
DEC <i>addr</i>	CE	Absolute	x	x	x	3	6 <sup>5</sup>
DEC <i>dp</i> ,X	D6	DP Indexed,X	x	x	x	2	6 <sup>2,5</sup>
DEC <i>addr</i> ,X	DE	Absolute Indexed,X	x	x	x	3	7 <sup>5,6</sup>
<b>DEX</b> <i>Decrement Index Register X</i> [Flags affected: n,z]							
DEX	CA	Implied	x	x	x	1	2
<b>DEY</b> <i>Decrement Index Register Y</i> [Flags affected: n,z]							
DEY	88	Implied	x	x	x	1	2
<b>EOR</b> <i>Exclusive-OR Accumulator with Memory</i> [Flags affected: n,z]							
EOR ( <i>dp</i> ,X)	41	DP Indexed Indirect,X	x	x	x	2	6 <sup>1,2</sup>
EOR <i>sr</i> ,S	43	Stack Relative			x	2	4 <sup>1</sup>
EOR <i>dp</i>	45	Direct Page	x	x	x	2	3 <sup>1,2</sup>
EOR [ <i>dp</i> ]	47	DP Indirect Long			x	2	6 <sup>1,2</sup>
EOR <i>#const</i>	49	Immediate	x	x	x	2 <sup>17</sup>	2 <sup>1</sup>
EOR <i>addr</i>	4D	Absolute	x	x	x	3	4 <sup>1</sup>
EOR <i>long</i>	4F	Absolute Long			x	4	5 <sup>1</sup>
EOR ( <i>dp</i> ),Y	51	DP Indirect Indexed, Y	x	x	x	2	5 <sup>1,2,3</sup>
EOR ( <i>dp</i> )	52	DP Indirect		x	x	2	5 <sup>1,2</sup>
EOR ( <i>sr</i> ,S),Y	53	SR Indirect Indexed,Y			x	2	7 <sup>1</sup>
EOR <i>dp</i> ,X	55	DP Indexed,X	x	x	x	2	4 <sup>1,2</sup>
EOR [ <i>dp</i> ],Y	57	DP Indirect Long Indexed, Y			x	2	6 <sup>1,2</sup>
EOR <i>addr</i> ,Y	59	Absolute Indexed,Y	x	x	x	3	4 <sup>1,3</sup>



EOR <i>addr</i> ,X	5D	Absolute Indexed,X	x	x	x	3	4 <sup>1,3</sup>
EOR <i>long</i> ,X	5F	Absolute Long Indexed,X			x	4	5 <sup>1</sup>
<b>INC</b> <i>Increment</i> [Flags affected: n,z]							
INC A	1A	Accumulator		x	x	1	2
INC <i>dp</i>	E6	Direct Page	x	x	x	2	5 <sup>2,5</sup>
INC <i>addr</i>	EE	Absolute	x	x	x	3	6 <sup>5</sup>
INC <i>dp</i> ,X	F6	DP Indexed,X	x	x	x	2	6 <sup>2,5</sup>
INC <i>addr</i> ,X	FE	Absolute Indexed,X	x	x	x	3	7 <sup>5,6</sup>
<b>INX</b> <i>Increment Index Register X</i> [Flags affected: n,z]							
INX	E8	Implied	x	x	x	1	2
<b>INY</b> <i>Increment Index Register Y</i> [Flags affected: n,z]							
INY	C8	Implied	x	x	x	1	2
<b>JMP</b> <i>Jump</i> [Flags affected: none][Alias: JML for all Long addressing modes]							
JMP <i>addr</i>	4C	Absolute	x	x	x	3	3
JMP <i>long</i>	5C	Absolute Long			x	4	4
JMP ( <i>addr</i> )	6C	Absolute Indirect	x	x	x	3	5 <sup>11,12</sup>
JMP ( <i>addr</i> ,X)	7C	Absolute Indexed Indirect		x	x	3	6
JMP [ <i>addr</i> ]	DC	Absolute Indirect Long			x	3	6
<b>JSR</b> <i>Jump to Subroutine</i> [Flags affected: none][Alias: JSL for Absolute Long]							
JSR <i>addr</i>	20	Absolute	x	x	x	3	6
JSR <i>long</i>	22	Absolute Long			x	4	8
JSR ( <i>addr</i> ,X)	FC	Absolute Indexed Indirect			x	3	8
<b>LDA</b> <i>Load Accumulator from Memory</i> [Flags affected: n,z]							
LDA ( <i>dp</i> ,X)	A1	DP Indexed Indirect,X	x	x	x	2	6 <sup>1,2</sup>
LDA <i>sr</i> ,S	A3	Stack Relative			x	2	4 <sup>1</sup>
LDA <i>dp</i>	A5	Direct Page	x	x	x	2	3 <sup>1,2</sup>
LDA [ <i>dp</i> ]	A7	DP Indirect Long			x	2	6 <sup>1,2</sup>
LDA # <i>const</i>	A9	Immediate	x	x	x	2 <sup>17</sup>	2 <sup>1</sup>
LDA <i>addr</i>	AD	Absolute	x	x	x	3	4 <sup>1</sup>
LDA <i>long</i>	AF	Absolute Long			x	4	5 <sup>1</sup>
LDA ( <i>dp</i> ),Y	B1	DP Indirect Indexed, Y	x	x	x	2	5 <sup>1,2,3</sup>
LDA ( <i>dp</i> )	B2	DP Indirect		x	x	2	5 <sup>1,2</sup>
LDA ( <i>sr</i> ,S),Y	B3	SR Indirect Indexed,Y			x	2	7 <sup>1</sup>
LDA <i>dp</i> ,X	B5	DP Indexed,X	x	x	x	2	4 <sup>1,2</sup>
LDA [ <i>dp</i> ],Y	B7	DP Indirect Long Indexed, Y			x	2	6 <sup>1,2</sup>
LDA <i>addr</i> ,Y	B9	Absolute Indexed,Y	x	x	x	3	4 <sup>1,3</sup>
LDA <i>addr</i> ,X	BD	Absolute Indexed,X	x	x	x	3	4 <sup>1,3</sup>

LDA <i>long</i> ,X	BF	Absolute Long Indexed,X		x	4	5 <sup>1</sup>
<b>LDX</b> <i>Load Index Register X from Memory</i> [Flags affected: n,z]						
LDX <i>#const</i>	A2	Immediate	x	x	x	2 <sup>19</sup> 2 <sup>10</sup>
LDX <i>dp</i>	A6	Direct Page	x	x	x	2 3 <sup>2,10</sup>
LDX <i>addr</i>	AE	Absolute	x	x	x	3 4 <sup>10</sup>
LDX <i>dp</i> ,Y	B6	DP Indexed,Y	x	x	x	2 4 <sup>2,10</sup>
LDX <i>addr</i> ,Y	BE	Absolute Indexed,Y	x	x	x	3 4 <sup>3,10</sup>
<b>LDY</b> <i>Load Index Register Y from Memory</i> [Flags affected: n,z]						
LDY <i>#const</i>	A0	Immediate	x	x	x	2 <sup>19</sup> 2 <sup>10</sup>
LDY <i>dp</i>	A4	Direct Page	x	x	x	2 3 <sup>2,10</sup>
LDY <i>addr</i>	AC	Absolute	x	x	x	3 4 <sup>10</sup>
LDY <i>dp</i> ,X	B4	DP Indexed,X	x	x	x	2 4 <sup>2,10</sup>
LDY <i>addr</i> ,X	BC	Absolute Indexed,X	x	x	x	3 4 <sup>3,10</sup>
<b>LSR</b> <i>Logical Shift Memory or Accumulator Right</i> [Flags affected: n,z,c]						
LSR <i>dp</i>	46	Direct Page	x	x	x	2 5 <sup>2,5</sup>
LSR A	4A	Accumulator	x	x	x	1 2
LSR <i>addr</i>	4E	Absolute	x	x	x	3 6 <sup>5</sup>
LSR <i>dp</i> ,X	56	DP Indexed,X	x	x	x	2 6 <sup>2,5</sup>
LSR <i>addr</i> ,X	5E	Absolute Indexed,X	x	x	x	3 7 <sup>5,6</sup>
<b>MVN</b> <i>Block Move Negative</i> [Flags affected: none][Registers: X,Y,C]						
MVN <i>srcbk,destbk</i>	54	Block Move		x	3	1 <sup>3</sup>
<b>MVP</b> <i>Block Move Positive</i> [Flags affected: none][Registers: X,Y,C]						
MVN <i>srcbk,destbk</i>	44	Block Move		x	3	1 <sup>3</sup>
<b>NOP</b> <i>No Operation</i> [Flags affected: none]						
NOP	EA	Implied	x	x	x	1 2
<b>ORA</b> <i>OR Accumulator with Memory</i> [Flags affected: n,z]						
ORA ( <i>dp</i> ,X)	01	DP Indexed Indirect,X	x	x	x	2 6 <sup>1,2</sup>
ORA <i>sr</i> ,S	03	Stack Relative			x	2 4 <sup>1</sup>
ORA <i>dp</i>	05	Direct Page	x	x	x	2 3 <sup>1,2</sup>
ORA [ <i>dp</i> ]	07	DP Indirect Long			x	2 6 <sup>1,2</sup>
ORA <i>#const</i>	09	Immediate	x	x	x	2 <sup>17</sup> 2 <sup>1</sup>
ORA <i>addr</i>	0D	Absolute	x	x	x	3 4 <sup>1</sup>
ORA <i>long</i>	0F	Absolute Long			x	4 5 <sup>1</sup>
ORA ( <i>dp</i> ),Y	11	DP Indirect Indexed, Y	x	x	x	2 5 <sup>1,2,3</sup>
ORA ( <i>dp</i> )	12	DP Indirect		x	x	2 5 <sup>1,2</sup>

ORA ( <i>sr</i> , <i>S</i> ), <i>Y</i>	13	SR Indirect Indexed, <i>Y</i>			x	2	7 <sup>1</sup>
ORA <i>dp</i> , <i>X</i>	15	DP Indexed, <i>X</i>	x	x	x	2	4 <sup>1,2</sup>
ORA [ <i>dp</i> ], <i>Y</i>	17	DP Indirect Long Indexed, <i>Y</i>			x	2	6 <sup>1,2</sup>
ORA <i>addr</i> , <i>Y</i>	19	Absolute Indexed, <i>Y</i>	x	x	x	3	4 <sup>1,3</sup>
ORA <i>addr</i> , <i>X</i>	1D	Absolute Indexed, <i>X</i>	x	x	x	3	4 <sup>1,3</sup>
ORA <i>long</i> , <i>X</i>	1F	Absolute Long Indexed, <i>X</i>			x	4	5 <sup>1</sup>
<b>PEA</b> <i>Push Effective Absolute Address</i> [Flags affected: none]							
PEA <i>addr</i>	F4	Stack (Absolute)			x	3	5
<b>PEI</b> <i>Push Effective Indirect Address</i> [Flags affected: none]							
PEI ( <i>dp</i> )	D4	Stack (DP Indirect)			x	2	6 <sup>2</sup>
<b>PER</b> <i>Push Effective PC Relative Indirect Address</i> [Flags affected: none]							
PER <i>label</i>	62	Stack (PC Relative Long)			x	3	6
<b>PHA</b> <i>Push Accumulator</i> [Flags affected: none]							
PHA	48	Stack (Push)	x	x	x	1	3 <sup>1</sup>
<b>PHB</b> <i>Push Data Bank Register</i> [Flags affected: none]							
PHB	8B	Stack (Push)			x	1	3
<b>PHD</b> <i>Push Direct Page Register</i> [Flags affected: none]							
PHD	0B	Stack (Push)			x	1	4
<b>PHK</b> <i>Push Program Bank Register</i> [Flags affected: none]							
PHK	4B	Stack (Push)			x	1	3
<b>PHP</b> <i>Push Processor Status Register</i> [Flags affected: none]							
PHP	08	Stack (Push)	x	x	x	1	3
<b>PHX</b> <i>Push Index Register X</i> [Flags affected: none]							
PHX	DA	Stack (Push)		x	x	1	3 <sup>10</sup>
<b>PHY</b> <i>Push Index Register Y</i> [Flags affected: none]							
PHY	5A	Stack (Push)		x	x	1	3 <sup>10</sup>
<b>PLA</b> <i>Pull Accumulator</i> [Flags affected: n,z]							
PLA	68	Stack (Pull)	x	x	x	1	4 <sup>1</sup>
<b>PLB</b> <i>Pull Data Bank Register</i> [Flags affected: n,z]							
PLB	AB	Stack (Pull)			x	1	4
<b>PLD</b> <i>Pull Direct Page Register</i> [Flags affected: n,z]							
PLD	2B	Stack (Pull)			x	1	5
<b>PLP</b> <i>Pull Processor Status Register</i> [Flags affected: n,z]							
PLP	28	Stack (Pull)	x	x	x	1	4
<b>PLX</b> <i>Pull Index Register X</i> [Flags affected: n,z]							
PLX	FA	Stack (Pull)		x	x	1	4 <sup>10</sup>
<b>PLY</b> <i>Pull Index Register Y</i> [Flags affected: n,z]							

PLY	7A	Stack (Pull)	x	x	1	4 <sup>10</sup>
<b>REP</b> <i>Reset Processor Status Bits</i> [Flags affected: all except b per operand]						
REP #const	C2	Immediate		x	2	3
<b>ROL</b> <i>Rotate Memory or Accumulator Left</i> [Flags affected: n,z,c]						
ROL dp	26	Direct Page	x	x	x	2 5 <sup>2,5</sup>
ROL A	2A	Accumulator	x	x	x	1 2
ROL addr	2E	Absolute	x	x	x	3 6 <sup>5</sup>
ROL dp,X	36	DP Indexed,X	x	x	x	2 6 <sup>2,5</sup>
ROL addr,X	3E	Absolute Indexed,X	x	x	x	3 7 <sup>5,6</sup>
<b>ROR</b> <i>Rotate Memory or Accumulator Right</i> [Flags affected: n,z,c]						
ROR dp	66	Direct Page	x	x	x	2 5 <sup>2,5</sup>
ROR A	6A	Accumulator	x	x	x	1 2
ROR addr	6E	Absolute	x	x	x	3 6 <sup>5</sup>
ROR dp,X	76	DP Indexed,X	x	x	x	2 6 <sup>2,5</sup>
ROR addr,X	7E	Absolute Indexed,X	x	x	x	3 7 <sup>5,6</sup>
<b>RTI</b> <i>Return from Interrupt</i> [Flags affected: all except b]						
RTI	40	Stack (RTI)	x	x	x	1 6 <sup>9</sup>
<b>RTL</b> <i>Return from Subroutine Long</i> [Flags affected: none]						
RTL	6B	Stack (RTL)		x	1	6
<b>RTS</b> <i>Return from Subroutine</i> [Flags affected: none]						
RTS	60	Stack (RTS)	x	x	x	1 6
<b>SBC</b> <i>Subtract with Borrow from Accumulator</i> [Flags affected: n,v,z,c]						
SBC (dp,X)	E1	DP Indexed Indirect,X	x	x	x	2 6 <sup>1,2,4</sup>
SBC sr,S	E3	Stack Relative			x	2 4 <sup>1,4</sup>
SBC dp	E5	Direct Page	x	x	x	2 3 <sup>1,2,4</sup>
SBC [dp]	E7	DP Indirect Long			x	2 6 <sup>1,2,4</sup>
SBC #const	E9	Immediate	x	x	x	2 <sup>17</sup> 2 <sup>1,4</sup>
SBC addr	ED	Absolute	x	x	x	3 4 <sup>1,4</sup>
SBC long	EF	Absolute Long			x	4 5 <sup>1,4</sup>
SBC (dp),Y	F1	DP Indirect Indexed, Y	x	x	x	2 5 <sup>1,2,3,4</sup>
SBC (dp)	F2	DP Indirect		x	x	2 5 <sup>1,2,4</sup>
SBC (sr,S),Y	F3	SR Indirect Indexed,Y			x	2 7 <sup>1,4</sup>
SBC dp,X	F5	DP Indexed,X	x	x	x	2 4 <sup>1,2,4</sup>
SBC [dp],Y	F7	DP Indirect Long Indexed, Y			x	2 6 <sup>1,2,4</sup>
SBC addr,Y	F9	Absolute Indexed,Y	x	x	x	3 4 <sup>1,3,4</sup>
SBC addr,X	FD	Absolute Indexed,X	x	x	x	3 4 <sup>1,3,4</sup>

SBC <i>long</i> ,X	FF	Absolute Long Indexed,X		x	4	5 <sup>1,4</sup>
<b>SEC</b> <i>Set Carry Flag</i> [Flags affected: c]						
SEC	38	Implied	x	x	x	1 2
<b>SED</b> <i>Set Decimal Flag</i> [Flags affected: d]						
SED	F8	Implied	x	x	x	1 2
<b>SEI</b> <i>Set Interrupt Disable Flag</i> [Flags affected: i]						
SEI	78	Implied	x	x	x	1 2
<b>SEP</b> <i>Set Processor Status Bits</i> [Flags affected: all except b per operand]						
SEP	E2	Immediate		x	2	3
<b>STA</b> <i>Store Accumulator to Memory</i> [Flags affected: none]						
STA ( <i>dp</i> ,X)	81	DP Indexed Indirect,X	x	x	x	2 6 <sup>1,2</sup>
STA <i>sr</i> ,S	83	Stack Relative			x	2 4 <sup>1</sup>
STA <i>dp</i>	85	Direct Page	x	x	x	2 3 <sup>1,2</sup>
STA [ <i>dp</i> ]	87	DP Indirect Long			x	2 6 <sup>1,2</sup>
STA <i>addr</i>	8D	Absolute	x	x	x	3 4 <sup>1</sup>
STA <i>long</i>	8F	Absolute Long			x	4 5 <sup>1</sup>
STA ( <i>dp</i> ),Y	91	DP Indirect Indexed, Y	x	x	x	2 6 <sup>1,2</sup>
STA ( <i>dp</i> )	92	DP Indirect		x	x	2 5 <sup>1,2</sup>
STA ( <i>sr</i> ,S),Y	93	SR Indirect Indexed,Y			x	2 7 <sup>1</sup>
STA <i>dp</i> X	95	DP Indexed,X	x	x	x	2 4 <sup>1,2</sup>
STA [ <i>dp</i> ],Y	97	DP Indirect Long Indexed, Y			x	2 6 <sup>1,2</sup>
STA <i>addr</i> ,Y	99	Absolute Indexed,Y	x	x	x	3 5 <sup>1</sup>
STA <i>addr</i> ,X	9D	Absolute Indexed,X	x	x	x	3 5 <sup>1</sup>
STA <i>long</i> ,X	9F	Absolute Long Indexed,X			x	4 5 <sup>1</sup>
<b>STP</b> <i>Stop Processor</i> [Flags affected: none]						
STP	DB	Implied		x	1	3 <sup>14</sup>
<b>STX</b> <i>Store Index Register X to Memory</i> [Flags affected: none]						
STX <i>dp</i>	86	Direct Page	x	x	x	2 3 <sup>2,10</sup>
STX <i>addr</i>	8E	Absolute	x	x	x	3 4 <sup>10</sup>
STX <i>dp</i> ,Y	96	DP Indexed,Y	x	x	x	2 4 <sup>2,10</sup>
<b>STY</b> <i>Store Index Register Y to Memory</i> [Flags affected: none]						
STY <i>dp</i>	84	Direct Page	x	x	x	2 3 <sup>2,10</sup>
STY <i>addr</i>	8C	Absolute	x	x	x	3 4 <sup>10</sup>
STY <i>dp</i> ,X	94	DP Indexed,X	x	x	x	2 4 <sup>2,10</sup>
<b>STZ</b> <i>Store Zero to Memory</i> [Flags affected: none]						
STZ <i>dp</i>	64	Direct Page		x	x	2 3 <sup>1,2</sup>
STZ <i>dp</i> ,X	74	DP Indexed,X		x	x	2 4 <sup>1,2</sup>

STZ <i>addr</i>	9C	Absolute	x	x	3	4 <sup>1</sup>
STZ <i>addr,X</i>	9E	Absolute Indexed,X	x	x	3	5 <sup>1</sup>
<b>TAX</b> <i>Transfer Accumulator to Index Register X</i> [Flags affected: n,z]						
TAX	AA	Implied	x	x	x	1 2
<b>TAY</b> <i>Transfer Accumulator to Index Register Y</i> [Flags affected: n,z]						
TAY	A8	Implied	x	x	x	1 2
<b>TCD</b> <i>Transfer 16-bit Accumulator to Direct Page Register</i> [Flags affected: n,z]						
TCD	5B	Implied		x	1	2
<b>TCS</b> <i>Transfer 16-bit Accumulator to Stack Pointer</i> [Flags affected: none]						
TCS	1B	Implied		x	1	2
<b>TDC</b> <i>Transfer Direct Page Register to 16-bit Accumulator</i> [Flags affected: n,z]						
TDC	7B	Implied		x	1	2
<b>TRB</b> <i>Test and Reset Memory Bits Against Accumulator</i> [Flags affected: z]						
TRB <i>dp</i>	14	Direct Page	x	x	2	5 <sup>2,5</sup>
TRB <i>addr</i>	1C	Absolute	x	x	3	6 <sup>3</sup>
<b>TSB</b> <i>Test and Set Memory Bits Against Accumulator</i> [Flags affected: z]						
TSB <i>dp</i>	04	Direct Page	x	x	2	5 <sup>2,5</sup>
TSB <i>addr</i>	0C	Absolute	x	x	3	6 <sup>5</sup>
<b>TSC</b> <i>Transfer Stack Pointer to 16-bit Accumulator</i> [Flags affected: n,z]						
TSC	3B	Implied		x	1	2
<b>TSX</b> <i>Transfer Stack Pointer to Index Register X</i> [Flags affected: n,z]						
TSX	BA	Implied	x	x	x	1 2
<b>TXA</b> <i>Transfer Index Register X to Accumulator</i> [Flags affected: n,z]						
TXA	8A	Implied	x	x	x	1 2
<b>TXS</b> <i>Transfer Index Register X to Stack Pointer</i> [Flags affected: none]						
TXS	9A	Implied	x	x	x	1 2
<b>TXY</b> <i>Transfer Index Register X to Index Register Y</i> [Flags affected: n,z]						
TXY	9B	Implied		x	1	2
<b>TYA</b> <i>Transfer Index Register Y to Accumulator</i> [Flags affected: n,z]						
TYA	98	Implied	x	x	x	1 2
<b>TYX</b> <i>Transfer Index Register Y to Index Register X</i> [Flags affected: n,z]						
TYX	BB	Implied		x	1	2
<b>WAI</b> <i>Wait for Interrupt</i> [Flags affected: none]						
WAI	CB	Implied		x	1	3 <sup>15</sup>
<b>WDM</b> <i>Reserved for Future Expansion</i> [Flags affected: none (subject to change)]						
WDM	42	n/a		x	2 <sup>16</sup>	n/a <sup>16</sup>
<b>XBA</b> <i>Exchange B and A 8-bit Accumulators</i> [Flags affected: n,z]						

XBA	EB	Implied	x	1	3
<b>XCE</b> <i>Exchange Carry and Emulation Flags</i> [Flags affected: m,b/x,c,e]					
XCE	FB	Implied	x	1	2

**NOTES**

- <sup>1</sup> Add 1 cycle if m=0 (16-bit memory/accumulator)
- <sup>2</sup> Add 1 cycle if low byte of Direct Page Register is non-zero
- <sup>3</sup> Add 1 cycle if adding index crosses a page boundary
- <sup>4</sup> Add 1 cycle if 65C02 and d=1 (65C02 in decimal mode)
- <sup>5</sup> Add 2 cycles if m=0 (16-bit memory/accumulator)
- <sup>6</sup> Subtract 1 cycle if 65C02 and no page boundary crossed
- <sup>7</sup> Add 1 cycle if branch is taken
- <sup>8</sup> Add 1 cycle if branch taken crosses page boundary on 6502, 65C02, or 65816's 6502 emulation mode (e=1)
- <sup>9</sup> Add 1 cycle for 65816 native mode (e=0)
- <sup>10</sup> Add 1 cycle if x=0 (16-bit index registers)
- <sup>11</sup> Add 1 cycle if 65C02
- <sup>12</sup> 6502: Yields incorrect results if low byte of operand is \$FF (i.e., operand is \$xxFF)
- <sup>13</sup> 7 cycles per byte moved
- <sup>14</sup> Uses 3 cycles to shut the processor down: additional cycles are required by reset to restart it
- <sup>15</sup> Uses 3 cycles to shut the processor down: additional cycles are required by interrupt to restart it
- <sup>16</sup> Byte and cycle counts subject to change in future processors which expand WDM into 2-byte opcode portions of instructions of varying lengths
- <sup>17</sup> Add 1 byte if m=0 (16-bit memory/accumulator)
- <sup>18</sup> Opcode is 1 byte, but program counter value pushed onto stack is incremented by 2 allowing for optional signature byte
- <sup>19</sup> Add 1 byte if x=0 (16-bit index registers)

Copyright © 1996 Creative Micro Designs, Inc.

Reprinted with permission from Commodore World magazine, Issue #16.

