The 6502/65C02/65C816 Instruction Set Decoded

Introduction

Though the 6502 instruction set has a number of quirks and irregularities, large portions of it can be broken up into regular patterns. An understanding of these patterns can be beneficial to authors of assemblers or disassemblers for 6502 code--for example, the Apple II ROM uses the information described below to greatly reduce the size of the instruction tables used by the built-in machine language disassembler.

Note that the discussion below assumes a knowledge of 6502 programming. If you're looking for a tutorial or general programming reference for the 6502, I recommend starting at 6502.org. There are also some useful documents at Western Design Center.

Instruction Chart

Shown below are the instructions of the 6502, 65C02, and 65C816 processors. GREEN UPPERCASE indicates instructions found on all processors; Yellow Mixed Case indicates instructions introduced on the 65C02, and not documented indicates instructions found only on the 65C816. The bit manipulation instructions found only on the Rockwell and WDC versions of the 65C02 are not included in the table, nor are the "undocumented" instructions of the original 6502. (However, after noting the search engine strings commonly used to locate this page, I have added discussions of these points below.)

	x0	x1	x2	x3	x4	x5	x6	x7	x8	x9	xA	xB	xC	xD	xE	xF
0x	BRK s	ORA (d,X)	COP s	ORA d,S	TSB d	ORA d	ASL d	ORA [d]	PHP	ORA #	ASL A	PHD	TSB a	ORA a	ASL a	ORA al
1x	BPL r	ORA (d),Y	ORA (d)	ORA (d,S),Y	TRB d	ORA d,X	ASL d,X	ORA [d],Y	CLC	ORA a,Y	INC A	TCS	TRB a	ORA a,X	ASL a,X	ORA al,X
2x	JSR a	AND (d,X)	JSL al	AND d,S	BIT d	AND d	ROL d	AND [d]	PLP	AND #	ROL A	PLD	BIT a	AND a	ROL a	AND al
3x	BMI r	AND (d),Y	AND (d)	AND (d,S),Y	BIT d,X	AND d,X	ROL d,X	AND [d],Y	SEC	AND a,Y	DEC A	TSC	BIT a,X	AND a,X	ROL a,X	AND al,X
4x	RTI	EOR (d,X)	WDM	EOR d,S	MVP	EOR d	LSR d	EOR [d]	PHA	EOR #	LSR A	PHK	JMP a	EOR a	LSR a	EOR al
5x	BVC r	EOR (d),Y	EOR (d)	EOR (d,S),Y	MVN	EOR d,X	LSR d,X	EOR [d],Y	CLI	EOR a,Y	PHY	TCD	JML al	EOR a,X	LSR a,X	EOR al,X
6x	RTS	ADC (d,X)	PER rl	ADC d,S	STZ d	ADC d	ROR d	ADC [d]	PLA	ADC #	ROR A	RTL	JMP (a)	ADC a	ROR a	ADC al
7x	BVS r	ADC (d),Y	ADC (d)	ADC (d,S),Y	STZ d,X	ADC d,X	ROR d,X	ADC [d],Y	SEI	ADC a,Y	PLY	TDC	JMP (a,X)	ADC a,X	ROR a,X	ADC al,X
8x	BRA r	STA (d,X)	BRL rl	STA d,S	STY d	STA d	STX d	STA [d]	DEY	BIT #	TXA	PHB	STY a	STA a	STX a	STA al
9x	BCC r	STA (d),Y	STA (d)	STA (d,S),Y	STY d,X	STA d,X	STX d,Y	STA [d],Y	TYA	STA a,Y	TXS	TXY	STZ a	STA a,X	STZ a,X	STA al,X
Ax	LDY #	LDA (d,X)	LDX #	LDA d,S	LDY d	LDA d	LDX d	LDA [d]	TAY	LDA #	TAX	PLB	LDY a	LDA a	LDX a	LDA al
Bx	BCS r	LDA (d),Y	LDA (d)	LDA (d,S),Y	LDY d,X	LDA d,X	LDX d,Y	LDA [d],Y	CLV	LDA a,Y	TSX	TYX	LDY a,X	LDA a,X	LDX a,Y	LDA al,X
Cx	CPY #	CMP (d,X)	REP #	CMP d,S	CPY d	CMP d	DEC d	CMP [d]	INY	CMP #	DEX	WAI	CPY a	CMP a	DEC a	CMP al
Dx	BNE r	CMP (d),Y	CMP (d)	CMP (d,S),Y	PEI d	CMP d,X	DEC d,X	CMP [d],Y	CLD	CMP a,Y	PHX	STP	JML [a]	CMP a,X	DEC a,X	CMP al,X
Ex	CPX #	SBC (d,X)	SEP #	SBC d,S	CPX d	SBC d	INC d	SBC [d]	INX	SBC #	NOP	XBA	CPX a	SBC a	INC a	SBC al
Fx	BEQ r	SBC (d),Y	SBC (d)	SBC (d,S),Y	PEA s	SBC d,X	INC d,X	SBC [d],Y	SED	SBC a,Y	PLX	XCE	JSR (a,X)	SBC a,X	INC a,X	SBC al,X

6502 Instructions

Most instructions that explicitly reference memory locations have bit patterns of the form **aaabbbcc**. The **aaa** and **cc** bits determine the opcode, and the **bbb** bits determine the addressing mode.

Instructions with **cc = 01** are the most regular, and are therefore considered first. The **aaa** bits determine the opcode as follows:

aaa opcode
000 ORA
001 AND
010 EOR
011 ADC
100 STA
101 LDA
110 CMP
111 SBC

And the addressing mode (**bbb**) is:

bbb addressing mode
000 (zero page,X)
001 zero page
010 #immediate
011 absolute
100 (zero page),Y
101 zero page,X
110 absolute,Y
111 absolute,X

Putting it all together:

	ORA	AND	EOR	ADC	STA	LDA	CMP	SBC
(zp,X)	01	21	41	61	81	A1	C1	E1
zp	05	25	45	65	85	A5	C5	E5
#	09	29	49	69		A9	C9	E9
abs	0D	2D	4D	6D	8D	AD	CD	ED
(zp),Y	11	31	51	71	91	B1	D1	F1
zp,X	15	35	55	75	95	B5	D5	F5
abs,Y	19	39	59	79	99	B9	D9	F9
abs,X	1D	3D	5D	7D	9D	BD	DD	FD

The only irregularity is the absence of the nonsensical immediate STA instruction.

Next we consider the **cc = 10** instructions. These have a completely different set of opcodes:

aaa opcode
000 ASL
001 ROL
010 LSR
011 ROR
100 STX
101 LDX
110 DEC
111 INC

The addressing modes are similar to the **01** case, but not quite the same:

bbb addressing mode
000 #immediate
001 zero page
010 accumulator
011 absolute
101 zero page,X
111 absolute,X

Note that **bbb = 100** and **110** are missing. Also, with STX and LDX, "zero page,X" addressing becomes "zero page,Y", and with LDX, "absolute,X" becomes "absolute,Y".

These fit together like this:

	ASL	ROL	LSR	ROR	STX	LDX	DEC	INC
#						A2		
zp	06	26	46	66	86	A6	C6	E6
A	0A	2A	4A	6A				
abs	0E	2E	4E	6E	8E	AE	CE	EE
zp,X/zp,Y	16	36	56	76	96	B6	D6	F6
abs,X/abs,Y	1E	3E	5E	7E		BE	DE	FE

Most of the gaps in this table are easy to understand. Immediate mode makes no sense for any instruction other than LDX, and accumulator mode for DEC and INC didn't appear until the 65C02. The slots that "STX A" and "LDX A" would occupy are taken by TXA and TAX respectively, which is exactly what one would expect. The only inexplicable gap is the absence of a "STX abs,Y" instruction.

Next, the **cc = 00** instructions. Again, the opcodes are different:

aaa opcode
001 BIT
010 JMP
011 JMP (abs)
100 STY
101 LDY
110 CPY
111 CPX

It's debatable whether the JMP instructions belong in this list. *I've included them because they do seem to fit, provided one considers the indirect JMP a separate opcode rather than a different addressing mode of the absolute JMP.*

The addressing modes are the same as the **10** case, except that accumulator mode is missing.

bbb addressing mode
000 #immediate
001 zero page
011 absolute
101 zero page,X
111 absolute,X

And here's how they fit together:

	BIT	JMP	JMP()	STY	LDY	CPY	CPX
#					A0	C0	E0
zp	24			84	A4	C4	E4
abs	2C	4C	6C	8C	AC	CC	EC
zp,X				94	B4		
abs,X				BC			

Some of the gaps in this table are understandable (e.g. the lack of an immediate mode for JMP, JMP(), and STY), but others are not (e.g. the absence of "zp,X" for CPY and CPX, and the absence of "abs,X" for STY, CPY, and CPX). Note that if accumulator mode (**bbb = 010**) were available, "LDY A" would be A8, which falls in the slot occupied by TAY, but the pattern breaks down elsewhere--TYA is 98, rather than B8, which we would expect it to be if it corresponded to the nonexistant "STY A".

No instructions have the form **aaabbb11**.

The conditional branch instructions all have the form **xxy10000**. The flag indicated by **xx** is compared with **y**, and the branch is taken if they are equal.

xx	flag
00	negative
01	overflow
10	carry
11	zero

This gives the following branches:

BPL	BMI	BVC	BVS	BCC	BCS	BNE	BEQ
10	30	50	70	90	B0	D0	F0

The remaining instructions are probably best considered simply by listing them. Here are the interrupt and subroutine instructions:

BRK	JSR abs	RTI	RTS
00	20	40	60

(JSR is the only absolute-addressing instruction that doesn't fit the **aaabbbcc** pattern.)

Other single-byte instructions:

PHP	PLP	PHA	PLA	DEY	TAY	INY	INX
08	28	48	68	88	A8	C8	E8

CLC	SEC	CLI	SEI	TYA	CLV	CLD	SED
18	38	58	78	98	B8	D8	F8

TXA	TXS	TAX	TSX	DEX	NOP
8A	9A	AA	BA	CA	EA

"Undocumented" 6502 Instructions

The above-described instructions (the ones shown in GREEN UPPERCASE in the table at the top of this page) are the only ones documented in any manufacturer's official data sheets. The question often arises, "What do all those other leftover bytes do if you try to execute them as instructions?"

In general the behavior of instructions other than those listed above cannot be described exactly, as they tend to be somewhat unstable, and do not always behave the same way on chips made by different manufacturers, and some instructions don't even behave the same way twice on the same chip. Those looking for a precise listing of "undocumented" instruction behaviors will have to look elsewhere, and should beware that the behaviors described on other web pages may be specific to 6502s made by a particular (often unspecified) manufacturer.

However, there are some facts that seem to be common across all 6502s. The most interesting case is the **cc = 11** instructions: these execute the adjacent **cc = 01** and **cc = 10** instructions *simultaneously*. For example, **AF** executes **AD** ("LDA absolute") and **AE** ("LDX absolute") at the same time, putting the same value in both the accumulator and the X register.

In some cases the **01** and **10** instructions are incompatible. For example, **8F** executes **8D** ("STA absolute") and **8E** ("STX absolute") at the same time. So which register actually gets written to memory? Usually some mixture of the two, in a manner that varies depending on who made the 6502, when it was made, the phase of the moon, and other unpredictable variables.

The behavior of the **11** instructions is especially problematic in those cases where the adjacent **01** or **10** instruction is also undocumented. Sometimes you can get a partial idea of what happens by looking at what the missing **01** or **10** instruction would be if that opcode/addressing mode combination weren't missing. **Xxxx1011** instructions are also problematic--some of these seem to mix not only the adjacent **01** and **10** instructions, but also the immediate mode of the corresponding **10** instruction.

Most of the missing **00**, **01**, and **10** instructions seem to behave like NOPs, but using the addressing mode indicated by the **bbb** bits. But apparently this isn't always reliable--there are reports of some of these instructions occasionally locking up the processor.

Instructions of the form **xxxx0010** usually lock up the processor, so that a reset is required to recover. The instructions **82**, **C2**, and **E2** (corresponding to the nonexistant immediate mode of STX, DEC, and INC) may sometimes behave as two-byte NOPs, but don't count on it.

65C02 Instructions

The new instructions of the 65C02 are much less logical than those listed above. The designers of the 65C02 apparently chose to continue leaving the **cc = 11** instructions empty, and this didn't leave much space for new instructions. Some instructions landed in logical places, but others had to be assigned wherever there was room, whether it made sense or not.

The new zero-page indirect addressing mode fills the previously-unused **bbb = 100** slot of the **cc = 10** instructions, but the opcodes are those of the **cc = 01** instructions.

	ORA	AND	EOR	ADC	STA	LDA	CMP	SBC
(zp)	12	32	52	72	92	B2	D2	F2

"JMP (abs,X)" is right where it ought to be (**011 111 00**), if one continues to regard the indirect JMP as a separate opcode from the absolute JMP:

	JMP()
abs,X	7C

"BIT zp,X" and "BIT abs,X" ended up exactly where one would expect them to be, but "BIT #" had to be moved because its slot was already taken by JSR:

	BIT
#	89
zp,X	34
abs,X	3C

TSB ended up in a reasonable place (**000bbb00**):

	TSB
zp	04
abs	0C

But the above assignments exhaust the logical possibilities for opcodes that explicitly reference memory locations, so TRB and STZ had to be put wherever room could be found:

	TRB	STZ
zp	14	64
abs	1C	9C
zp,X		74
abs,X		9E

That leaves the relative branch instruction

BRA
80

and the single-byte instructions:

INC A	DEC A	PHY	PLY	PHX	PLX
1A	3A	5A	7A	DA	FA

Additional instructions found on some 65C02s

Actually, I lied when I said above that the designers of the 65C02 chose to leave the **cc = 11** instructions unused. On 65C02s made by Rockwell and by WDC, some of these instructions are used for additional bit setting, clearing, and testing instructions. These instructions are missing on 65C02s made by other manufacturers. (And since this page is part of a set of Apple II-related pages, I should point out that Apple never shipped any computers that used Rockwell or WDC 65C02s, so none of the instructions in this section are available on an unmodified Apple II.)

The bit set and clear instructions have the form **xyyy0111**, where **x** is 0 to clear a bit or 1 to set it, and **yyy** is which bit at the memory location to set or clear.

	RMB0	RMB1	RMB2	RMB3	RMB4	RMB5	RMB6	RMB7
zp	07	17	27	37	47	57	67	77

	SMB0	SMB1	SMB2	SMB3	SMB4	SMB5	SMB6	SMB7
zp	87	97	A7	B7	C7	D7	E7	F7

Similarly, the test-and-branch instructions are of the form **xyyy1111**, where **x** is 0 to test whether the bit is 0, or 1 to test whether it is 1, and **yyy** is which bit to test.

	BBR0	BBR1	BBR2	BBR3	BBR4	BBR5	BBR6	BBR7
zp,rel	0F	1F	2F	3F	4F	5F	6F	7F

	BBS0	BBS1	BBS2	BBS3	BBS4	BBS5	BBS6	BBS7
zp,rel	8F	9F	AF	BF	CF	DF	EF	FF

Additionally, the WDC version of the 65C02 includes the 65C816's STP and WAI instructions (see below).

"Undocumented" 65C02 Instructions

There aren't really any undocumented instructions on the 65C02--any instructions not listed above are documented as performing no operation.

However, these alternate NOPs are not created equal. Some have one- or two-byte operands (which they don't do anything with), and they take different amounts of time to execute.

Instruction Bytes	Cycles
xxxxxx10	2	2
xxxxxx11	1	1
0100/0100	2	3
xTx10100	2	4
01011100	3	8
11x11100	3	4

65C816 Instructions

The 65C816 uses the **cc = 11** instructions, but not for Rockwell bit-manipulation opcodes. Most of these are put to work supplying the new long addressing modes of the 65C816:

bbb addressing mode
000 offset,S
001 [direct page]
011 absolute long
100 (offset,S),Y
101 [direct page],Y
111 absolute long,X

These combine with the **01** opcodes:

	ORA	AND	EOR	ADC	STA	LDA	CMP	SBC
d,S	03	23	43	63	83	A3	C3	E3
[dp]	07	27	47	67	87	A7	C7	E7
al	0F	2F	4F	6F	8F	AF	CF	EF
(d,S),Y	13	33	53	73	93	B3	D3	F3
[dp],Y	17	37	57	77	97	B7	D7	F7
al,X	1F	3F	5F	7F	9F	BF	DF	FF

The missing **010** and **110** instructions are all single-byte instructions:

PHD	PLD	PHK	RTL	PHB	PLB	WAI	XBA
0B	2B	4B	6B	8B	AB	CB	EB

TCS	TSC	TCD	TDC	TXY	TYX	STP	XCE
1B	3B	5B	7B	9B	BB	DB	FB

The remaining instructions are a grab-bag assigned to the few remaining unused positions:

COP sig	02
JSL al	22
WDM	42
PER rl	62
BRL rl	82
REP #	C2
SEP #	E2
MVP sb,db	44
MVN sb,db	54
PEI dp	D4
PEA abs	F4
JMP al	5C
JML (abs)	DC
JSR (abs,X)	FC

Original: July 27, 2004
Modified: May 3, 2005