Introduction

This is a list of the opcodes used in the 65816 processor. It is used by the Super Nintendo System (SNES) console.

For the registers, the following convention will be used:

Convention	Description
A	the low byte of the accumulator, always 8-bit
B	the high byte of the accumulator, always 8-bit
C	the low and high bytes of the accumulator, always 16-bit
X	the X register, 8-bit or 16-bit
Y	the Y register, 8-bit or 16-bit
P	the processor register, always 8-bit

For the opcode arguments, the following convention will be used:

Argument	Example	Relative to
addr	LDA $1234	memory address, short, 16-bit
const	LDA #$80	constant, 8-bit or 16-bit
dp	LDA $10	direct page, relative, 8-bit
long	LDA $C21234	memory address, long, 24-bit
near	BRA $1234	branch, relative, 8-bit
sr	LDA $01,S	stack, relative, 8-bit

LDA, LDX, LDY

Reads a value from the memory and stores it in the specified register. LDA will store in the A or C register, LDX will store in the X register and LDY will store in the Y register.

flags
n highest bit of value loaded
z set if data loaded is zero

Opcode	Syntax	Bytes	Notes
A1	LDA (dp,X)	2	
A3	LDA sr,S	2	
A5	LDA dp	2	
A7	LDA [dp]	2	
A9	LDA const	2	One extra byte if in 16-bit mode
AD	LDA addr	3	
AF	LDA long	4	
B1	LDA (dp),Y	2	
B2	LDA (dp)	2	
B3	LDA (sr,S),Y	2	
B5	LDA dp,X	2	

Opcode	Syntax	Bytes	Notes
B7	LDA [dp],Y	2	
B9	LDA addr,Y	3	
BD	LDA addr,X	3	
BF	LDA long,X	4	

Opcode	Syntax	Bytes	Notes
A2	LDX const	2	One extra byte if in 16-bit mode
A6	LDX dp	2	
AE	LDX addr	3	
B6	LDX dp,Y	2	
BE	LDX addr,Y	3	

Opcode	Syntax	Bytes	Notes
A0	LDY const	2	One extra byte if in 16-bit mode
A4	LDY dp	2	
AC	LDY addr	3	
B4	LDY dp,X	2	
BC	LDY addr,X	3	

STA, STX, STY

Reads a value from the specified register and stores it in the memory. STA will read the A or C register, STX will read the X register and STY will read the Y register. All opcodes will store the retrieved value in the specified address.

The STZ opcode will always store the value of zero in the designed memory address. Unlike the STA, STX and STY opcodes, the STZ opcode doesn't affect flags. It also doesn't affect the A or C register.

Opcode	Syntax	Bytes	Notes
81	STA (dp,X)	2	
83	STA sr,S	2	
85	STA dp	2	
87	STA [dp]	2	
8D	STA addr	3	
8F	STA long	4	
91	STA (dp),Y	2	
92	STA (dp)	2	
93	STA (sr,S),Y	2	
95	STA dp,X	2	
97	STA [dp],Y	2	
99	STA addr,Y	3	
9D	STA addr,X	3	
9F	STA long,X	4	

Opcode	Syntax	Bytes	Notes
86	STX dp	2	
8E	STX addr	3	
96	STX dp,Y	2	

Opcode	Syntax	Bytes	Notes
84	STY dp	2	
8C	STY addr	3	
94	STY dp,X	2	

Opcode	Syntax	Bytes	Notes
64	STZ dp	2	
74	STZ dp,X	2	
9C	STZ addr	3	
9E	STZ addr,X	3	

TRB, TSB

Test all bits in the A or C register. For all bits set in the A or C register, the correspondent bits will be set or unset in the designed memory address. TSB will set the correspondent bits and TRB will reset the correspondent bits. All clear bits in the A or C register will be ignored and their correspondent bits in the designed address will not be altered.

Opcode	Syntax	Bytes	Notes
14	TRB dp	2	
1C	TRB addr	3	

Opcode	Syntax	Bytes	Notes
04	TSB dp	2	
0C	TSB addr	3	

ADC, SBC

Add or subtract the value in the A or C register with the value in the designed memory address. ADC will add the value and SBC will subtract the value.

When the carry flag is set, a value of one will be added in the addition or subtraction operation.

The operation can overflow or underflow. It is the coder responsibility to check these cases and adjust the resulting value. A common method is to check the carry flag for overflow or underflow and cap the value to a prefixed setting.

flags
n set if highest bit of result was set
v set if signed overflow?
z set if result is zero
c set if overflow or underflow

Opcode	Syntax	Bytes	Notes
61	ADC (dp,X)	2	
63	ADC sr,S	2	
65	ADC dp	2	
67	ADC [dp]	2	
69	ADC const	2	One extra byte if in 16-bit mode

Opcode	Syntax	Bytes	Notes
6D	ADC addr	3	
6F	ADC long	4	
71	ADC (dp),Y	2	
72	ADC (dp)	2	
73	ADC (sr,S),Y	2	
75	ADC dp,X	2	
77	ADC [dp],Y	2	
79	ADC addr,Y	3	
7D	ADC addr,X	3	
7F	ADC long,X	4	

Opcode	Syntax	Bytes	Notes
E1	SBC (dp,X)	2	
E3	SBC sr,S	2	
E5	SBC dp	2	
E7	SBC [dp]	2	
E9	SBC const	2	One extra byte if in 16-bit mode
ED	SBC addr	3	
EF	SBC long	4	
F1	SBC (dp),Y	2	
F2	SBC (dp)	2	
F3	SBC (sr,S),Y	2	
F5	SBC dp,X	2	
F7	SBC [dp],Y	2	
F9	SBC addr,Y	3	
FD	SBC addr,X	3	
FF	SBC long,X	4	

INC, INX, INY, DEC, DEX, DEY

Increment or decrement the specified register by one. It is equivalent to the ADC or SBC opcodes with the value of one without the carry flag set.

INC will increment the A or C register. INX will increment the X register. INY will increment the Y register. DEC will decrement the A or C register. DEX will decrement the X register. DEY will decrement the Y register. The operation can underflow or overflow.

INC and DEC can also increment or decrement a value in memory instead of the A or C register.

flags
n set if most significant bit of result is set
z set if result is zero

Opcode	Syntax	Bytes	Notes
1A	INC	1	increment the A or C register
C8	INY	1	increment the Y register
E6	INC dp	2	

Opcode	Syntax	Bytes	Notes
E8	INX	1	increment the X register
EE	INC addr	3	
F6	INC dp,X	2	
FE	INC addr,X	3	

Opcode	Syntax	Bytes	Notes
3A	DEC	1	decrement the A or C register
88	DEY	1	decrement register Y
C6	DEC dp	2	
CA	DEX	1	decrement register X
CE	DEC addr	3	
D6	DEC dp,X	2	
DE	DEC addr,X	3	

AND, ORA, EOR

A binary operation is done between the register A or C and the memory. The AND opcode will do a binary AND, the ORA opcode will do a binary OR and the EOR opcode will do a binary XOR. The result will be stored in the A or C register.

flags
n set if highest bit is set
z set if result is zero

Opcode	Syntax	Bytes	Notes
21	AND (dp,X)	2	
23	AND sr,S	2	
25	AND dp	2	
27	AND [dp]	2	
29	AND const	2	One extra bit if in 16-bit mode
2D	AND addr	3	
2F	AND long	4	
31	AND (dp),Y	2	
32	AND (dp)	2	
33	AND (sr,S),Y	2	
35	AND dp,X	2	
37	AND [dp],Y	2	
39	AND addr,Y	3	
3D	AND addr,X	3	
3F	AND long,X	4	

Opcode	Syntax	Bytes	Notes
01	ORA (dp,X)	2	
03	ORA sr,S	2	
05	ORA dp	2	
07	ORA [dp]	2	
09	ORA const	2	One extra byte if in 16-bit mode
0D	ORA addr	3	

Opcode	Syntax	Bytes	Notes
0F	ORA long	4	
11	ORA (dp),Y	2	
12	ORA (dp)	2	
13	ORA (sr,S),Y	2	
15	ORA dp,X	2	
17	ORA [dp],Y	2	
19	ORA addr,Y	3	
1D	ORA addr,X	3	
1F	ORA long,X	4	

Opcode	Syntax	Bytes	Notes
41	EOR (dp,X)	2	
43	EOR sr,S	2	
45	EOR dp	2	
47	EOR [dp]	2	
49	EOR const	2	One extra byte if in 16-bit mode
4D	EOR addr	3	
4F	EOR long	4	
51	EOR (dp),Y	2	
52	EOR (dp)	2	
53	EOR (sr,S),Y	2	
55	EOR dp,X	2	
57	EOR [dp],Y	2	
59	EOR addr,Y	3	
5D	EOR addr,X	3	
5F	EOR long,X	4	

BIT

Does a binary AND between the A or C register and the memory. Unlike the AND opcode, the binary AND operation doesn't alter the A or C register.

flags
n Takes value of most significant bit of memory data
v Takes value from bit 6 or 14 of memory data
z Set if logical AND of mem and acc is zero.

Opcode	Syntax	Bytes	Notes
24	BIT dp	2	
2C	BIT addr	3	
34	BIT dp,X	2	
3C	BIT addr,X	3	
89	BIT const	2	Add one extra bit if in 16-bit mode

CMP, CPX, CPY

Compare the designed register with the memory and sets flags based on the comparison. CMP will compare the A or C register, CPX will compare the X register and CPY will compare the Y register. Generally, a conditional branch opcode, based on a flag status, is used after these opcodes.

flags
n set if most significant bit of result is set
z set if result is zero
c set if register is equal or greater than memory

Opcode	**Syntax**	**Bytes**	**Notes**
C1	CMP (dp,X)	2	
C3	CMP sr,S	2	
C5	CMP dp	2	
C7	CMP [dp]	2	
C9	CMP const	2	One extra byte if in 16-bit mode
CD	CMP addr	3	
CF	CMP long	4	
D1	CMP (dp),Y	2	
D2	CMP (dp)	2	
D3	CMP (sr,S),Y	2	
D5	CMP dp,X	2	
D7	CMP [dp],Y	2	
D9	CMP addr,Y	3	
DD	CMP addr,X	3	
DF	CMP long,X	4	

Opcode	**Syntax**	**Bytes**	**Notes**
E0	CPX const	2	One extra byte if in 16-bit mode
E4	CPX dp	2	
EC	CPX addr	3	

Opcode	**Syntax**	**Bytes**	**Notes**
C0	CPY const	2	One extra byte if in 16-bit mode
C4	CPY dp	2	
CC	CPY addr	3	

LSR, ASL

Shift bits in the designed register or memory address left or right. The LSR opcode will shift the bits right and the ASL opcode will shift the bits left.

LSR is equivalent to divide the number by two and ASL is equivalent to multiply the number by two. The operation can underflow or overflow. The bit shifted out will become the new carry. The bit shifted in is zero.

flags: LSR
n cleared

flags: LSR
z set if result is zero
c bit zero becomes new carry

flags: ASL
n set if most significant bit of result is set
z set if result is zero
c set if highest bit is moved into carry

Opcode	**Syntax**	**Bytes**	**Notes**
46	LSR dp	2	
4A	LSR	1	shift A or C register
4E	LSR addr	3	
56	LSR dp,X	2	
5E	LSR addr,X	3	

Opcode	**Syntax**	**Bytes**	**Notes**
06	ASL dp	2	
0A	ASL	1	the operand is the register A
0E	ASL addr	3	
16	ASL dp,X	2	
1E	ASL addr,X	3	

ROL, ROR

Shift bits in the designed register or memory address left or right. The ROL opcode will shift the bits left and the ROR opcode will shift the bits right.

The bit shifted in is the old carry. The bit shifted out will be the new carry.

flags: ROL
n set if most significant bit of result is set
z set if result is zero
c the high bit (7 or 15) becomes the new carry

flags: ROR
n set if most significant bit of result is set
z set if result is zero
c low bit becomes the new carry

Opcode	**Syntax**	**Bytes**	**Notes**
26	ROL dp	2	
2A	ROL	1	affects register A or C
2E	ROL addr	3	
36	ROL dp,X	2	
3E	ROL addr,X	3	

Opcode	**Syntax**	**Bytes**	**Notes**
66	ROR dp	2	
6A	ROR	1	affects register A or C
6E	ROR addr	3	
76	ROR dp,X	2	

Opcode	Syntax	Bytes	Notes
7E	ROR addr,X	3	

BPL, BMI, BRA, BCC, BCS, BNE, BEQ, BVC, BVS, BRL

Change the execution flow based on a conditional branch. If the condition is meet, the execution flow will be changed for the designed address. Otherwise, it continues normally.

The majority of the conditional branches are based in flag status. The flags were setup by anterior opcodes.

flags:
none

Opcode	Syntax	Bytes	Notes
10	BPL near	2	BPL - branch if minus flag is clear, branch if plus
30	BMI near	2	BMI - branch if minus flag set, branch if minus
50	BVC near	2	BVC - branch if overflow clear
70	BVS near	2	BVS - branch if overflow set
80	BRA near	2	BRA - always branch
82	BRL addr	3	similar to BRA, but with longer range
90	BCC near	2	BCC - branch if carry flag clear
B0	BCS near	2	BCS - branch if carry flag set
D0	BNE near	2	BNE - branch if zero flag clear, branch if not equal
F0	BEQ near	2	BEQ - branch if zero flag set, branch if equal

JMP

Change the execution flow to the designed address.

flags:
none

Opcode	Syntax	Bytes	Notes
4C	JMP addr	3	
5C	JMP long	4	
6C	JMP (addr)	3	
7C	JMP (addr,X)	3	
DC	JMP [addr]	3	

JSR, JSL

Calls a sub routine which will be executed.

The JSR opcode will put two bytes in the stack, relative to the return address. It is expected that they will be retrieved by an RTS opcode in the sub routine.

The JSL opcode will put three bytes in the stack, relative to the return address. It is expected that they

will be retrieved by an RTL opcode in the sub routine.

Opcode	Syntax	Bytes	Notes
20	JSR addr	3	
22	JSL long	4	
FC	JSR (addr,X)	3	

RTS, RTL

RTS will pull two bytes from the stack and RTL will pull three bytes from the stack. They will be setup as the new control flow address.

It is expected that the bytes in the stack were pushed from a JSR opcode for the RTS opcode and from a JSL opcode for the RTL opcode.

flags:
none

Opcode	Syntax	Bytes	Notes
60	RTS	1	pull two bytes from the stack as the return address
6B	RTL	1	pull three bytes from the stack as the return address

CLC, CLD, CLI, CLV, SEC, SED, SEI

Set or reset flags.

Opcode	Syntax	Bytes	Notes
18	CLC	1	clear carry flag
38	SEC	1	set carry flag
58	CLI	1	clear interrupt flag
78	SEI	1	set interrupt flag
B8	CLV	1	clear overflow flag
D8	CLD	1	clear decimal flag
F8	SED	1	set decimal flag

TAX, TAY, TXA, TYA, TSX, TXS, TXY, TYX, TCD, TDC, TCS, TSC

Transfer bytes between the specified registers or special addresses.

If the setting of the registers have different settings of 8-bit or 16-bit, the copy of the high byte of the transfer can be truncated based on the opcode behavior.

flags: all except TCS
n set if most significant bit of transfer value is set
z set if transferred value is zero

Opcode	Syntax	Bytes	Notes
1B	TCS	1	from C to stack pointer

Opcode	Syntax	Bytes	Notes
3B	TSC	1	from stack pointer to C
5B	TCD	1	from C to direct page
7B	TDC	1	from direct page to C
8A	TXA	1	from X register to A or C
98	TYA	1	from Y register to A or C
9A	TXS	1	from X register to stack pointer
9B	TXY	1	from X register to Y register
A8	TAY	1	from A or C to Y register
AA	TAX	1	from A or C to X register
BA	TSX	1	from stack pointer to X register
BB	TYX	1	from Y register to X register

MVP, MVN

Move a block of bytes from a specified address to another address. The X register will be the start address to copy from. The Y register will be start address to copy to. The C register will be the number of bytes to copy minus one.

The opcode arguments are the source bank and the destination bank, in this order.

MVP must be used if the Y register is greater than the X register. Otherwise, MVN must be used instead.

Opcode	Syntax	Bytes	Notes
44	MVP src,dest	3	
54	MVN src,dest	3	

PEA, PEI, PER

Push bytes in the stack, based on the opcodes arguments. They always work in 16-bit mode.

Opcode	Syntax	Bytes	Notes
62	PER label	3	
D4	PEI (dp)	2	
F4	PEA const	3	the argument is always 16-bit

PHA, PHP, PHX, PHY, PHB, PHD, PHK, PLA, PLP, PLX, PLY, PLB, PLD

Push or pull bytes from the stack. The setting of the register or special address as 8-bit or 16-bit will set the number of bytes pushed/pulled from the stack.

flags: PHA/PHP/PHX/PHY/PHB/PHD/PHK
none

flags: PLA/PLX/PLY/PLB/PLD
n set if most significant bit of result is set
z set if result is zero

Opcode	**Syntax**	**Bytes**	**Notes**
08	PHP	1	push flag register (8-bit)
0B	PHD	1	push direct page register (16-bit)
48	PHA	1	push A or C register
4B	PHK	1	push program bank register (8-bit)
5A	PHY	1	push Y register (8-bit or 16-bit)
8B	PHB	1	push data bank register (8-bit)
DA	PHX	1	push X register (8-bit or 16-bit)

Opcode	**Syntax**	**Bytes**	**Notes**
28	PLP	1	pull flag register (8-bit)(sets all flags)
2B	PLD	1	pull direct page register (16-bit)
68	PLA	1	pull A or C register
7A	PLY	1	pull Y register (8-bit or 16-bit)
AB	PLB	1	pull data bank register (8-bit)
FA	PLX	1	pull X register (8-bit or 16-bit)

NOP

Does absolutely nothing. It is mostly used to waste cycles. A few special registers or addresses can't be fetch until a minimum number of cycles has passed.

If the coder needs to erase a code segment, for better performance, a branch or jump is recommended instead of a long line of NOP opcodes.

Opcode	**Syntax**	**Bytes**	**Notes**
EA	NOP	1	

REP, SEP

Setups the processor flags with the opcode argument. For each bit set in the argument, the correspondent flag is set or reset. REP will reset the flags and SEP will set the flags.

bits	**description**
$01	carry flag
$02	zero flag
$04	irq flag?
$08	decimal mode
$10	X and Y register: 0 = 16-bit, 1 = 8-bit
$20	accumulator: 0 = 16-bit, 1 = 8-bit
$40	overflow flag
$80	negative flag

flags
all

Opcode	Syntax	Bytes	Notes
C2	REP const	2	const is always 8-bit
E2	SEP const	2	const is always 8-bit

XBA

Exchanges the A and B registers.

flags
n set if the most significant bit of the new value in the low order 8 bits (A) of the accumulator is set. (former bit 15)
z set if new value of the lower order 8 bit accumulator (A) is zero.

Opcode	Syntax	Bytes	Notes
EB	XBA	1	

XCE

Exchanges the carry flag and the emulation flag.

flags
e from previous carry flag
c from previous emulation flag
m native mode flag only. switching to native 65816 mode sets to one
x x is a native mode flag only
b brk is an emulation 6502 flag only. it is set to 1 to become the x flag in native mode

Opcode	Syntax	Bytes	Notes
FB	XCE	1	

Miscellaneous

Opcode	Syntax	Bytes	Notes
00	BRK	?	software break
02	COP const	?	coprocessor empowerment
40	RTI	1	return from interrupt
CB	WAI	1	wait for interrupt
DB	STP	1	stop processor