MOS 6502 / MOS 6510

65xx CPU Family

Registers

Instruction Set

Addressing Modes

The 65xx-Family:

Type	Features, Comments
6502	NMOS, 16 bit address bus, 8 bit data bus (Commodore VIC-20)
6502A	accelerated version of 6502 (BBC Micro Model A+B / Atari 1200 XL)
6502B	accelerated version of 6502 (Atari 400)
6502C	accelerated version of 6502, CMOS (Atari 65 XE / 130 XE)
65C02	16 bit vsn, additional instructions and address modes (Apple IIe)
6503, 6505, 6506	12 bit address bus <4 KiB>
6504	13 bit address bus <8 KiB>
6507	13 bit address bus <8 KiB>, no interrupts
6509	20 bit address bus <1 MiB> by bank switching (Commodore PET 700)
6510	as 6502 with additional 6 bit I/O-port (Commodore 64)
6511	integrated micro controller with I/O-port, serial interface, and RAM (Rockwell)
65F11	as 6511, integrated FORTH interpreter
7501	as 6502, HMOS (Commodore 16 / Plus 4)
8500	as 6510, CMOS (Commodore 64G)
8502	as 6510 with switchable 2 MHz option, 7 bit I/O-port (Commdore 128)
65816 (65C816)	16 bit registers and ALU, 24 bit address bus <16 MiB>, up to 24 MHz (Western Design Centre)
65802 (65C802)	as 65816, pin compatible to 6502, 64 KiB address bus, up to 16 MHz

6502 Registers

PC	....	program counter	(16 bit)
AC	....	accumulator	(8 bit)
X	....	X register	(8 bit)
Y	....	Y register	(8 bit)
SR	....	Status Register <NV-BDIZC>	(8 bit)
(P)		(or Processor Register)	

Status Registers (or Processor Register)

N	....	Negative
V	....	Overflow
-	....	ignored
B	....	Break
D	....	Decimal (use BCD for arithmetic's)
I	....	Interrupt (IRQ disable)
Z	....	Zero
C	....	Carry

Processor Stack:
LIFO, top down, 8 bit range,
0x0100 - 0x01FF

Data Transfer Instructions

LDA : Load Accumulator with Memory
LDX : Load X Register with Memory
LDY : Load Y Register with Memory
TAX : Transfer Acc. into X Register
TAY : Transfer Acc. Into Y Register

STA : Store Accumulator in Memory
STX : Store X Register in Memory
STY : Store Y Register in Memory
TXA : Transfer X Register into Acc.
TYA : Transfer Y Register into Acc.

Arithmetic Operation Instructions

ADC : Add Memory with Accumulator With Carry

SBC : Subtract Memory From Accumulator With Borrow

Logical Operation Instructions

AND : AND Memory with Accumulator
EOR : EXCLUSIVE-OR Memory with Accumulator

ORA : OR Memory With Accumulator

Shift and Modify Instructions

DEC : Decrement Memory by one
DEX : Decrement X Register by one
DEY : Decrement Y Register by one
ASL : Shift Left One Bit
ROL : Rotate Left One Bit

INC : Increment Memory by one
INX : Increment X Register by one
INY : Increment Y Register by one
LSR : Shift Right One Bit
ROR : Rotate Right One Bit

Test Instructions

CMP : Compare Memory and Accumulator
BIT : Test Bits in Memory with Accumulator

CPX : Compare Memory with X Register
CPY : Compare Memory with Y Register

Branch Instructions

BCC : Branch on Carry Clear
BEQ : Branch on Result Zero
BMI : Branch on Result Minus
BVC : Branch on Overflow Clear

BCS : Branch on Carry Set
BNE : Branch on Result Not Zero
BPL : Branch on Result Plus
BVS : Branch on Overflow Set

Modify Processor Status Register Instructions

CLC : Clear Carry Flag
CLD : Clear Decimal Mode
CLI : Clear Interrupt Flag
CLV : Clear Overflow Flag

SEC : Set Carry Flag
SED : Set Decimal Mode
SEI : Set Interrupt Flag

Jump Instructions

JMP : Jump to New Location
JSR : Jump to Subroutine
BRK : Jump to Interrupt Routine

RTS : Return from Subroutine
RTI : Return from Interrupt Routine

Stack Operation Instructions

PHA	:	Push Accumulator on Stack	PLA	:	Pull Accumulator from Stack
PHP	:	Push P (SR) Register on Stack	PLP	:	Pull P (SR) Register from Stack
TXS	:	Transfer X Register to Stack Pointer	TSX	:	Trans. Stack Pointer to X Register

Do Nothing Instructions

NOP : No Operation

A	Accumulator	OPC A	operand is AC
Abs	Absolute	OPC $HHLL	operand is address $HHLL
abs,X	absolute, X-indexed	OPC $HHLL,X	operand is address incremented by X with carry
abs,Y	absolute, Y-indexed	OPC $HHLL,Y	operand is address incremented by Y with carry
#	Immediate	OPC #$BB	operand is byte (BB)
impl	Implied	OPC	operand implied
ind	Indirect	OPC ($HHLL)	operand is effective address; effective address is value of address
X,ind	X-indexed, indirect	OPC ($BB,X)	operand is effective zeropage address; effective address is byte (BB) incremented by X without carry
ind,Y	indirect, Y-indexed	OPC ($LL),Y	operand is effective address incremented by Y with carry; effective address is word at zeropage address
rel	relative	OPC $BB	branch target is PC + offset (BB), bit 7 signifies negative offset
zpg	zeropage	OPC $LL	operand is of address; address hibyte = zero ($00xx)
zpg,X	zeropage, X-indexed	OPC $LL,X	operand is address incremented by X; address hibyte = zero ($00xx); no page transition
zpg,Y	zeropage, Y-indexed	OPC $LL,Y	operand is address incremented by Y; address hibyte = zero ($00xx); no page transition