6502 - tsiraM Instruction set

Description	Op Code	Mnemonic	Assembly	Binary (in Hex)
Load the accumulator with a constant	A9	LDA	LDA #$07	A9 07
Load the accumulator from memory	AD	LDA	LDA $0010	AD 10 00
Store the accumulator in memory	8D	STA	STA $0010	8D 10 00
Load the accumulator from X register	8A	TXA	TXA	8A
Load the accumulator from Y register	98	TYA	TYA	98
Add with carry:Adds contents of an address to the accumulator and keeps the result in the accumulator	6D	ADC	ADC $0010	6D 10 00
Load the X register with a constant	A2	LDX	LDX #$01	A2 01
Load the X register from memory	AE	LDX	LDX $0010	AE 10 00
Load the X register from the accumulator	AA	TAX	TAX	AA
Load the Y register with a constant	A0	LDY	LDY #$04	A0 04
Load the Y register from memory	AC	LDY	LDY $0010	AC 10 00
Load the Y register from the accumulator	A8	TAY	TAY	A8
No Operation	EA	NOP	NOP	EA
Break	00	BRK	BRK	00
Compare a byte in memory to the X reg. Sets the Z (zero) flag if equal	EC	CPX	CPX $0010	EC 10 00
Branch *n* bytes if Z flag = 0	D0	BNE	BNE $EF	D0 EF
Increment the value of a byte	EE	INC	INC $0021	EE 21 00
System Calls (This is not part of original 6502 instruction set)				
If there is a #$01 in the X register Print the integer in the Y register	FF	SYS	SYS	FF
*If there is a #$02 in the X register Print the 0x00 terminated string stored at address in the Y register	FF	SYS	SYS	FF
If there is a #$03 in the X register Print the 0x00 terminated string from the address in the operand	FF	SYS	SYS $0010	FF 10 00

6502 - tsiraM Instruction set: Example Programs
https://tsiram6502.abiggeek.com/

Use https://tsiram6502.abiggeek.com/ to run them. **Only binaries (instructions) can be run in the online emulator, you cannot paste the assembler examples, they are just for reference**.

For more information on system calls take a look at this full System call API (linux on x86): http://blog.rchapman.org/posts/Linux_System_Call_Table_for_x86_64/ (this is to contrast with our small system call API with 2 calls (print integer and print string).

Program example 1: 2+2 (just for reference)

```
#6502 Assembler
LDA #$02
STA $0010
ADC $0010
LDX #$01
TAY
SYS             ← system call! (X was loaded with 1, Y with the value in the Accumulator)
BRK

#2+2 6502 opcodes (what you load into the emulator to run)
0xA9, 0x02, 0x8D, 0x10, 0x00, 0x6D, 0x10, 0x00, 0xA2, 0x01, 0xA8, 0xFF, 0x00
```

#Hello World

```
# 6502 Assembler (just for reference)
LDA #$03
SYS $0006
BRK
DATA 0x48, 0x65, 0x6C, 0x6C, 0x6F, 0x20, 0x57, 0x6F, 0x72, 0x6C, 0x64, 0x21, 0x0A,
0x00

# 6502 Instructions
0xA2, 0x03, 0xFF, 0x06, 0x00, 0x00, 0x48, 0x65, 0x6C, 0x6C, 0x6F, 0x20, 0x57, 0x6F,
0x72, 0x6C, 0x64, 0x21, 0x0A, 0x00
```

0006

#basic loop

```
#6502 Assembler for reference
00 LDA #$05         A9
02 STA $0040        8D
05 LDA #$01         A9
```

```
07 STA $0041        8D
0A TAY              A8   ← loop back
0B LDX #$01         A2
0D SYS              FF
0E ADC $0041        6D
11 TAX
12 CPX $0040        EC
15 BNE #$F3         D0
17 BRK                    00
```

6502 Instructions
0xA9, 0x05, 0x8D, 0x40, 0x00, 0xA9, 0x01, 0x8D, 0x41, 0x00, 0xA8, 0xA2, 0x01, 0xFF, 0x6D, 0x41, 0x00, 0xAA, 0xEC, 0x40, 0x00, 0xD0, 0xF3, 0x00

11110011
00001101

To find BNE jump value: PC will be at 17 (15 for BNE, 16 for relative offset, 17 for advance when done)
We want to go to 0A
 17   => 23 (decimal)
-0A   => 10 (decimal)
------   -----
           13 = C   2's comp! 0000 1101 => 11110010+1 11110011 = F 3

#powers program! Note: this program will run until my virtual 6502 force closes it (after somewhere around 1 minute). Since this is not a multitasking system, programs have the CPU until they give it up. Since this program will continually create powers of 2 forever, it only stops because I have written a routine to limit the amount of cycles for any given program run.

6502 Assembler for reference
LDA #$00
STA $0040
LDA #$01
ADC $0040 ← loop back
STA $0040
TAY
LDX #$01
SYS
BNE $F4
HLT

#6502 Instructions

0xA9, 0x00, 0x8D, 0x40, 0x00, 0xA9, 0x01, 0x6D, 0x40, 0x00, 0x8D, 0x40, 0x00, 0xA8, 0xA2, 0x01, 0xFF, 0xD0, 0xF4, 0x00

Addition / Subtraction / Overflow / End-around carry checks

// addition normal (as high as you can go)
A9, 3F, 8D, 10, 00, A9, 40, 6D, 10, 00, A8, A2, 01, FF, 00

// subtraction direction 1: 5 - 3 = 2
A9, 05, 8D, 10, 00, A9, FD, 6D, 10, 00, A8, A2, 01, FF, 00

// subtraction direction 2: -3 + 5 = 2
A9, FD, 8D, 10, 00, A9, 05, 6D, 10, 00, A8, A2, 01, FF, 00

// neg result subtraction dir 1: 2 - 5 = -3 (FD)
A9, 02, 8D, 10, 00, A9, FB, 6D, 10, 00, A8, A2, 01, FF, 00

// neg result subtraction dir 2: -5 + 2 = -3 (FD)
A9, FB, 8D, 10, 00, A9, 02, 6D, 10, 00, A8, A2, 01, FF, 00

// positive overflow should result in 40 and carryFlag
A9, 60, 8D, 10, 00, A9, 60, 6D, 10, 00, A8, A2, 01, FF, 00

// 2 negatives in range should result in DD
A9, F0, 8D, 10, 00, A9, ED, 6D, 10, 00, A8, A2, 01, FF, 00

// negative overflow should result in 35 and carryFlag
A9, 95, 8D, 10, 00, A9, A0, 6D, 10, 00, A8, A2, 01, FF, 00