ASSEMBLER <http://www.6502.org/tutorials/6502opcodes.html>

Table

Description automatically generated

**Big vs Little endian**

• 08CE • 0000 1000 1100 1110

• Big endian zapis: • 08CE

• Little endian zapis: • CE08

***Big endian*** je način zapisa podatka u memoriji tako da je na nižoj adresi viši bajt memorijske riječi – normalan zapis

***Little endian*** je način zapisa podatka u memoriji tako da je na nižoj adresi niži bajt memorijske riječi naopačke, jednostavnije za računanje -kod malih ili starijih procesora, mikrokontroleri

**6502**

• Stari procesor • Nastao 1975 • Koristi 5V napajanje • 0-0.8V – logička 0

• 2-5V – logičko 1 • 8 bita • 16 bit adresna sabirnica (65536 lokacija)

**Registri • A, X, Y, P, S, PC**

• A jedini za aritmetiku

• X, Y mogu se koristiti kao indeksi za memoriju

**Zastavice**

P registar sadrži procesorske zastavice. Svaki bit u ovom registru ima jedinstvenu svrhu, osim bita označenog 1. 1 bit je neiskorišten i može se zanemariti. Svaki od preostalih bitova u ovom registru naziva se zastavicom i označava određeni uvjet koji se dogodio ili predstavlja postavku konfiguracije

• Bit 1 se ne koristi

• N – negativna vrijednost (ako je operacija postavila bit 7 rezultata)

• V – Overflow - ako operacija izađe iz raspona -128 do 127 – broj je pozitivan ali je krivo zapisan, staje u 8 bitova ali nije onaj broj koji smo zapisali jer je rezultat negativna vrijednost 127+15=-14

• B – Break – izvršena je operacija BRK. Služi razlikovanju softverskog i hardverskog prekida

• D – Decimal mode – označava da procesor koristi BCD zapis brojeva

• I – Interrupt disable – procesor neće obrađivati prekide (osim NMI)

• Z – Zero – Rezultat operacije je 0 - za grananje

• C – Carry – operacija je uzrokovala carry

**Stack pointer**

• Registar S • Odnosi se na memoriju $100 - $1FF • Govorimo o 256 bajtova

• Postavlja se na vrh • Svako spremanje na vrh smanjuje S • Ofset je fiksnih $100

• LIFO • Korisnik mora paziti

**Pravila pisanja**

• 24 znači 24 decimalno

• $24 znači da koristimo heksadecimalni zapis (36 decimalno)

• #24 znači izravna vrijednost 24

• #$24 znači izravno 36 u decimalnom zapisu

• LDA – Load A with value

• ADC – ADD using carry

• SBC – Subtract using Carry

• SEC – Set carry directly

• CLC – Clear carry directly

**Carry**

• Mora biti podešen prije operacije

• ADC će ga koristiti i dodati ukupnoj vrijednosti

• SBC će ga koristiti kako bi ga dodao prije oduzimanja

**Branching**

• Grananje je jedan od osnovnih načina kontrole rada

• JMP – skače na neku lokaciju

• Direktno se upisuje u JMP

• BCC i BCS – skaču ako je Carry Clear ili Set

• BNE BEQ ako je Z clear ili set

• BPL BMI Ako je N clear ili set

• BVC BVS ako je V clear ili set

• Koristi se relativno adresiranje -128 do 127

|  |  |
| --- | --- |
| Sta, x | 5 |
| Inx, dex  Tax, txa  Tay, tya  Dey iny | 2 |
| lda | 2 |
| bne | 3 |
| pha | 3 |
| pla | 4 |
| Stx , y  Sty, x | 4 |
| loop | 256 |

|  |  |  |
| --- | --- | --- |
| A picture containing text, stationary, writing implement  Description automatically generated**Prugice**  LDX #$00  loop:  TXA  STA $200, X  INX  BNE loop  // sta $300, x za drugi dio ekrana itd. | | **Crno bijele prugice**  **– ispiši svakog drugog - neparni**  LDX #$FF  LDA #$01  loop:  STA $0200, X  INX  INX  BNE loop |
|  | |  |
| **Fill 512 bytes of memory from location $200**  **with value $1**    LDA #$01  LDX #$00  loop:  STA $200, X  STA $300, X  INX  BNE loop | | **Drugi način za 512**  LDA #$08  LDX #$00  start:  STA $200, X  STA $300, X  INX  BNE start |
| **Ispisati pola ekrana s jednom pola s drugom bojom**    LDX #$00 ; 2  loop:  LDA #$08 ;2  STA $200,X ;5  STA $300,X ;5  LDA #$02 ;2  STA $400,X ;5  STA $500,X ;5  INX ;2  BNE loop ;3 | | Ili ----  LDX #$00  LDA #$03  loop:  STA $200,X ;5  STA $300,X ;5  INX ;2  BNE loop ;3 -- 15 x 256  LDA #$02 ;2  loop2:  STA $400,X ;5  STA $500,X ;5  INX ;2  BNE loop2 ;3 -- 15 x 256 |
| **Svaki treći**  LDX #$00  loop:  LDA #$01  STA $200,X  INX  TXA  CMP #$0  BEQ end  INX  INX  JMP loop  end: | | **Svaki peti**  LDX #$00  loop:  LDA #$01  STA $200,X  INX  TXA  CMP #$0  BEQ end  INX  INX  inx  inx  JMP loop  end: |
| **Obojati 1 / 4 ekrana jednom bojom**  LDA #$01  LDX #$00 ; krećemo od 00 da pokrijemo rubni slučaj  loop:  STA $200, X  DEX  BNE loop  Trik – iscrtavamo prije nego smanjimo vrijednost  – staviti komentar -ispravnije je staviti INX umjesto dex | **Drugi način**  LDA #$08  LDX #$00  start:  STA $200, X  STA $300, X  INX  BNE start | |
| **Povukli smo crtu u memoriji**  LDA #$01  LDX #$05  start:  STA $200, X  DEX  BNE start | **Spremanje na stack**  LDA #$01  PHA  LDA #$02  PHA  LDA #$03  PHA  LDA #$FF  PLA  PLA  PLA | |
| **; Add four bytes together using absolute**  **addressing mode**  LDA $0203  CLC  ADC $0202  ADC $0201  ADC $0200 | Drugi način:  LDX #$03  CLC  LDA $0200, X  DEX  ADC $0200, X  DEX  ADC $0200, X  DEX  ADC $0200, X | |
| **Subrutina**  loop:  LDA #$01  STA $300, X  JSR druga  INX  BNE loop  JMP end  druga:  LDA #$5  STA $400,X  RTS  end: | **Subrutina i stack -isto samo stack**  LDA #$06  loop:  STA $300, X  JSR druga  INX  BNE loop  JMP end  druga:  PHA  LDA #$5  STA $400,X  PLA  RTS  end: | |

|  |  |
| --- | --- |
| **Pola ekrana jedna pola druga boja**  LDX #$00  loop:  LDA #$08  STA $200,X  STA $300,X  LDA #$02  STA $400,X  STA $500,X  INX     BNE loop | **Isto to samo sa subrutinom**  LDA #$08  loop:  STA $200, X  STA $300, X  JSR druga  INX  BNE loop  JMP end  druga:  PHA  LDA #$02  STA $400,X  STA $500,X  PLA  RTS  end: |
| **Subrutina četvrtine ekrana**  LDA #$06  LDX #$00  loop:  STA $200,X  JSR sub  INX  BNE loop  JMP end  sub:  TAY  LDA #$05  STA $300,X  LDA #$1  TYA  RTS  end: | **Subrutina četvrtine ekrana sa stackom**  LDA #$06  LDX #$00  loop:  STA $200,X  JSR sub  INX  BNE loop  JMP end  sub:  PHA  LDA #$05  STA $300,X  LDA #$1  PLA  RTS  end: |

-----------------------------------------------

LDX #$03

CLC

LDA $0200, X

DEX

ADC $0200, X

DEX

ADC $0200, X

DEX

ADC $0200, X

Looping it

LDX #$03

LDA $0200, X

DEX

CLC

ADD\_LOOP:

ADC $0200, X

DEX

BPL ADD\_LOOP

Indirect indexed addressing

LDA #$00

STA $10

LDA #$02

STA $11

LDY #$03

LDA ($10), Y

DEY

CLC

ADD\_LOOP:

ADC ($10), Y

DEY

BPL ADD\_LOOP

------------------------------------------------------

LDA #$01 - spremi vrijednost 1 u registar a, ako je #broj to je decimalni broj, ako je #$ heksadecimalni broj

LDA $200 – odi u lokaciju 200 i uzmi vrijednost i spremi u akumulator

LDA $200, X – odi u lokaciju 200 i na njezinu vrijednost dodaj vrijednost x

Ako stisnemo disasemble on će ih pretvoriti u neke heksadecimalne kodove – opcode

---------------------