Podprogramy – JSR, RTS

To, k čemu u 8080 sloužily instrukce CALL a RET, zajišťují u 6502 instrukce JSR a RTS (Jump to Subroutine / Return from Subroutine). JSR používá pouze absolutní adresní mód, tj. za instrukcí jsou nižší a vyšší bajty cílové adresy. JSR uloží na zásobník vyšší a nižší bajt návratové adresy, a pak do PC nahraje přečtenou adresu. Čímž se vlastně provede skok na nějakou adresu (jako u JMP), s tím rozdílem, že na zásobníku je adresa, kam se má program vrátit (ve skutečnosti je o 1 menší, s čímž počítá instrukce RTS).

K návratu slouží instrukce RTS. Ta přečte ze zásobníku dva bajty, z nich složí adresu, přičte 1 (viz výše) a na ni skočí. Pokud podprogram zanechal zásobník v takovém stavu, v jakém ho našel, tak se skočí na instrukci, následující za příslušnou instrukcí JSR.

Rozdíl mezi RTS a RTI je v tom, že RTI načítá ze zásobníku i uloženou hodnotu příznakového registru P a k návratové adrese nepřičítá 1 (instrukce BRK i přerušení ukládají pravou návratovou adresu).

Úhrnem lze o instrukční sadě procesoru 6502 v souvislosti se skoky říct, že je hodně omezená. Podmíněné skoky pouze relativní, skoky do podprogramu a návraty pouze nepodmíněné a s absolutní adresou, jen nepodmíněný skok lze adresovat i nepřímou adresou (ale nelze relativně).

9.11 Aritmetika 6502

V této kapitole konečně donutíme procesor 6502 něco spočítat.

Když jsem v minulých kapitolách naznačoval, že to s ortogonalitou instrukční sady procesoru 6502 není, ani přes velké množství adresních módů, moc slavné, tak věřte, že jsem si to nejhorší šetřil až na závěr.

INC, DEC

Nejjednodušší aritmetické instrukce jsou inkrement a dekrement, tedy přičtení jedničky a odečtení jedničky. 6502 má k tomu účelu instrukce INC a DEC. Tyto instrukce zvýší (INC) nebo sníží (DEC) obsah paměťové buňky o 1. Můžete je použít s následujícími adresními módy:

- abs: INC 1234h – zvýší obsah buňky na adrese 1234h o 1
- zp: INC 12h – zvýší obsah buňky na adrese 0012h o 1
- abx: INC 1234h,X – zvýší obsah buňky na adrese (1234h + X) o 1
- zpx: INC 12h,X – zvýší obsah buňky v nulté stránce paměti na adrese (12h+X) o 1 (nezapomeňte, že nultá stránka má vždycky horní byte adresy rovný 0, pokud tedy bude v X hodnota FFh, nebude se pracovat s adresou 0111h, ale 0011h!)

Totéž pro instrukci DEC.

INX, INY, DEX, DEY

Obdoba instrukcí INC, DEC, ale místo obsahu paměti se pracuje s registry X (INX, DEX) a Y (INY, DEY).

Instrukce inkrementu a dekrementu nastavují podle výsledku operace příznaky N a Z.

Možná jste si všimli, že jsem v seznamu neuvedl instrukce, které inkrementují/dekrementují obsah akumulátoru A. Neuvedl jsem je, protože je procesor 6502 nemá.

ADC, SBC

Sčítání a odčítání 6502 samozřejmě obsahuje. ADC (Addition with Carry) přičte parametr a hodnotu příznaku C k registru A a výsledek ponechá v registru A. SBC (Subtraction with Carry) odečte od obsahu registru A parametr a negaci příznaku C a výsledek uloží do registru A.

Co z toho vyplývá? Zaprvé: 6502 vždycky uvažuje stav příznaku C (přenos). Neexistuje instrukce pro sčítání nebo odčítání, která by jeho stav ignorovala. Pokud chceme „jen" sčítat dvě čísla, je potřeba předtím nastavit C na nulu instrukcí CLC, jinak může být výsledek o 1 vyšší. Pokud výsledek sčítání přeteče 255, bude C=1, jinak zůstane nulový.

U instrukce pro odčítání platí přesný opak – pokud chceme zanedbat přenos, musíme příznak nastavit na 1 (instrukcí SEC). Pokud výsledek při odčítání podteče nulu (výsledkem je záporné číslo), bude příznak C roven 0, jinak 1.

A tak se může stát, pokud neošetříte příznaky správně, že dvojice instrukcí ADC #1, SBC #1 ve skutečnosti dělají věci nečekané. Viz následující kód – sledujte instrukce a výsledek v registru A:

```
0000 A9 08           LDA #8
0002 E9 01           SBC #1
0004 69 01           ADC #1
0006 38              SEC
0007 E9 01           SBC #1
0009 69 01           ADC #1
000B A9 00           LDA #0
000D E9 01           SBC #1
000F 69 01           ADC #1
```

Adresní módy těchto instrukcí jsou stejné jako např. u instrukce LDA, tedy:

- imm – přímý operand: ADC #1 přičte 1
- abs, zp – přímo zadaná adresa, buď plná, nebo v zero page
- abx, aby – absolutní adresa, zvýšená o obsah registru X či Y

- zpx – adresa v zero page, indexovaná přes registr X
- izx, izy – nepřímo adresovaný operand

Porovnání - CMP

Instrukce CMP porovná hodnotu v registru A s operandem. Vnitřně funguje tak, že od hodnoty v registru A odečte hodnotu operandu, podle výsledku nastaví příznaky N, Z a C a výsledek zahodí.

Mohou nastat tři situace, které si ukážeme v následující tabulce:

SITUACE	N	Z	C
A = operand	0	1	1
A > operand	0	0	1
A < operand	1	0	0

(Hodnoty uvažujeme jako čísla bez znaménka)

Instrukce CMP nabízí stejné adresační možnosti jako instrukce ADC či SBC.

```
0000 A9 08           LDA #8
0002 C9 08           CMP #8
0004 C9 07           CMP #7
0006 C9 09           CMP #9
```

CPX, CPY

Podobně jako existují obdoby instrukcí INC a DEC pro práci s registry X a Y, tak i CMP má obdoby CPX a CPY. Liší se od CMP tím, že neporovnávají operand s hodnotou registru A, ale s registrem X, resp. Y. CPX a CPY mají jen tři adresní módy: přímý operand (imm), absolutní adresa (abs) nebo adresa v nulté stránce (zp).

9.12 Logické a bitové operace 6502

Blížíme se ke konci, zbývá doprobrat už jen pár instrukcí, konkrétně logické operace a manipulace s bity.

AND, ORA, EOR

Trojice instrukcí pro základní bitové operace – and, or, xor (exclusive or) se u procesoru 6502 jmenují AND, ORA a EOR. Provedou danou logickou operaci s obsahem registru A, výsledek uloží do A a nastaví příznaky N a Z.