MOS Technology 6502 & ACME Cross-Assembler

Tento článek se zabývá jedním z nejvlivnějších mikroprocesorů historie, **MOS Technology 6502**, a jeho moderním programováním pomocí nástroje **ACME Cross-Assembler**. Tato kombinace představuje most mezi zlatou érou 8bitových počítačů a současným vývojem retro softwaru (tzv. homebrew a demoscene).

1. MOS Technology 6502

MOS Technology 6502 je 8bitový mikroprocesor navržený týmem Chucka Peddla v roce 1975. Ve své době způsobil tržní šok svou cenou **25 USD**, zatímco konkurenční Intel 8080 nebo Motorola 6800 stály téměř desetkrát tolik. Tato nízká cena umožnila vznik osobních počítačů pro masy.

Tento čip (nebo jeho varianty) poháněl systémy jako **Apple I & II**, **Commodore 64**, **Atari 2600**, **NES** (Nintendo) a **BBC Micro**.

1.1 Detailní architektura

Procesor 6502 je navržen jako **akumulátorově orientovaný procesor** s velmi malým počtem registrů. To zjednodušuje hardware čipu (méně tranzistorů), ale klade vyšší nároky na programátora a efektivní práci s pamětí.

A. Vnitřní sběrnice a ALU

Srdcem procesoru je **ALU (Arithmetic Logic Unit)**, která provádí sčítání, odčítání, logické operace (AND, ORA, EOR) a posuny bitů. ALU je propojena s Registry a vnitřní datovou sběrnicí.

- Procesor je **8bitový**, což znamená, že ALU zpracovává data po 8 bitech.
- Adresní sběrnice je **16bitová**, což umožňuje adresovat **64 KB** paměti ($0000 - $FFFF).
- Architektura je **Little Endian** – nejméně významný bajt (LSB) je uložen na nižší adrese.

B. Pipeline (Zřetězené zpracování)

Ačkoliv je 6502 často považován za jednoduchý procesor, využívá primitivní formu **pipeliningu**. Zatímco se jedna instrukce vykonává, procesor již načítá (Fetch) další instrukci z paměti. Díky tomu je 6502 taktovaný na 1 MHz výkonnostně srovnatelný s procesorem Intel 8080 nebo Zilog Z80 taktovaným na 2–4 MHz.

C. Programátorský model (Registry)

Registr	Zkratka	Velikost	Funkce
Akumulátor	A	8-bit	Primární registr pro veškerou matematiku a logiku.
Index X	X	8-bit	Počítadlo smyček, ofset pro adresování paměti.
Index Y	Y	8-bit	Podobný jako X, klíčový pro „Indirect Indexed“ adresování.
Stack Pointer	SP	8-bit	Ukazuje na vrchol zásobníku. Zásobník je „hard-wired“ na stránku $01 ($0100-$01FF).
Program Counter	PC	16-bit	Ukazatel na aktuálně vykonávanou instrukci.
Status Register	P / SR	8-bit	Obsahuje příznaky (Flags) výsledků operací.

D. Status Register (Flags)

Tento registr je klíčový pro větvení programu (instrukce `BNE`, `BEQ`, `BCC` atd.). Každý bit má svůj význam:

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
N	**V**	-	**B**	**D**	**I**	**Z**	**C**
Negative	Overflow	(Unused)	Break	Decimal	Interrupt	Zero	Carry

- **N (Negative):** Nastaven, pokud je bit 7 výsledku 1 (záporné číslo v doplňkovém kódu).
- **V (Overflow):** Přetečení při operacích se znaménkem.
- **D (Decimal):** Pokud je 1, procesor počítá v BCD kódu (Binary Coded Decimal) místo binárního.
- **I (Interrupt Disable):** Pokud je 1, maskovatelná přerušení (IRQ) jsou ignorována.
- **Z (Zero):** Nastaven na 1, pokud je výsledek operace 0.
- **C (Carry):** Přenos do vyššího řádu (při sčítání) nebo výpůjčka (při odčítání).

E. Systém přerušení (Interrupts)

Procesor reaguje na vnější události pomocí přerušení. Adresy obslužných rutin (vektory) jsou uloženy na samém konci paměti:

- **$FFFA - $FFFB:** NMI (Non-Maskable Interrupt) – nelze zakázat softwarem.
- **$FFFC - $FFFD:** RESET – adresa, kam procesor skočí po zapnutí.
- **$FFFE - $FFFF:** IRQ (Interrupt Request) / BRK – běžná přerušení.

2. ACME Cross-Assembler

ACME (Advanced Cross Macro Assembler) je multiplatformní nástroj, který umožňuje psát kód pro procesory rodiny 6502 na moderních operačních systémech (Windows, Linux, macOS). Výstupem je binární soubor spustitelný na emulátoru nebo reálném historickém hardwaru.

Díky podpoře maker a flexibilní syntaxi je ACME standardem v **Commodore 64 demo scéně**.

2.1 Struktura projektu v ACME

ACME nevnucuje pevnou strukturu, ale využívá „pseudo-opkódy“ (preprocesorové direktivy) začínající vykřičníkem.

<code asm> ; Příklad definice konstant a paměti !to „output.prg“, cbm ; Cílový soubor * = $0801 ; Počáteční adresa (Origin)

; Definice maker pro čitelnost !macro SetBorderColor .color {

```
  lda #.color
  sta $d020
```

} </code >

2.2 Klíčové direktivy

- `**!source „soubor.asm“**`: Vloží jiný zdrojový kód (modularita projektu).
- `**!binary „data.bin“**`: Vloží surová data (např. sprity, charsety).
- `**!fill 100, $EA**`: Vyplní 100 bajtů hodnotou $EA (NOP).
- `**!if**` / `**!else**`: Podmíněný překlad kódu.
- `**!zone**`: Vytváří lokální jmenné prostory pro návěští (Labels), což umožňuje používat názvy jako `.loop` opakovaně v různých částech kódu.

3. Praktická ukázka: Raster Bar (C64)

Následující kód demonstruje sílu 6502 ve spojení s ACME. Vytvoříme efekt stabilního „raster baru“ synchronizovaného s vykreslováním obrazu.

<code asm> ;

; ACME Assembler Source: Stable Raster Interrupt ; Platforma: Commodore 64 ;

!to „raster.prg“, cbm

;— Basic Bootstrap ($0801) — * = $0801 !byte $0b, $08, $0a, $00, $9e, $32, $30, $36, $34, $00, $00, $00 ; To odpovídá „10 SYS 2064“

;— Hlavní program — * = $0810

Init:

```
  sei             ; Vypnout přerušení
  lda #$7f
  sta $dc0d       ; Vypnout CIA přerušení
  lda $dc0d       ; Potvrdit vypnutí
```

```
  lda #$01
  sta $d01a       ; Zapnout raster interrupt na VIC-II
```

```
  lda #$80        ; Číslo řádku, kde chceme přerušení (128)
  sta $d012
  lda $d011       ; Vymazání 9. bitu rasteru (pro řádky < 255)
  and #$7f
  sta $d011
```

```
  lda #<IRQ_Handler  ; Nastavení vektoru přerušení (Low byte)
  sta $0314
  lda #>IRQ_Handler  ; High byte
  sta $0315
```

```
  cli             ; Povolit přerušení
  jmp * ; Nekonečná smyčka (CPU čeká na přerušení)
```

;— Obsluha přerušení (Interrupt Routine) — IRQ_Handler:

```
  ; Uložit registry dělá C64 KERNAL automaticky, ale zde
  ; přepisujeme standardní rutinu, tak musíme potvrdit IRQ.
  lsr $d019       ; Potvrzení přerušení (Acknowledge)
```

```
  ; Samotný efekt
  inc $d020       ; Změna barvy rámečku
  ldx #$0f        ; Čekací smyčka pro viditelnost pruhu
```

.wait:

```
  dex
  bne .wait
  dec $d020       ; Vrátit barvu zpět
```

```
  ; Skok na standardní systémovou rutinu (pro klávesnici atd.)
  jmp $ea31
```

</code >

4. Zdroje obrázků a licence

Všechny použité obrázky pocházejí z **Wikimedia Commons** a jsou volně šiřitelné:

- **Obr. 1 (MOS 6502):** Autor: Konstantin Lanzet (CC BY-SA 3.0). Dostupný z: https://commons.wikimedia.org/wiki/File:MOS_6502AD_4586.jpg
- **Obr. 2 (Pinout):** Autor: Gona.eu (Public Domain / CC0). Dostupný z: https://commons.wikimedia.org/wiki/File:Pinout_of_MOS_6502.svg

---

Kategorie: Hardware, Programování, Assembler, Retro Computing