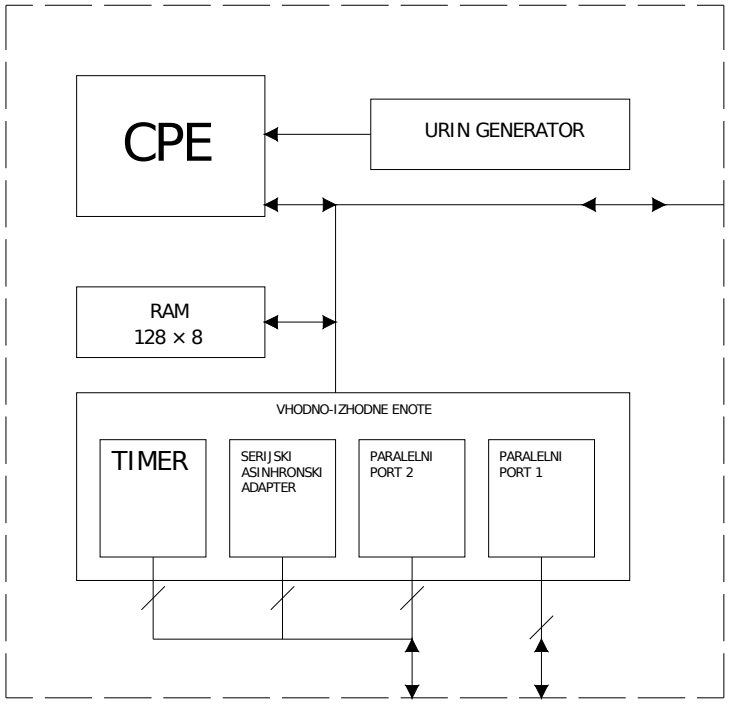
MC 6803

ZGRADBA MC 6803

MC 6803 je integrirano vezje s 40 nožicami, narejeno v NMOS-tehnologiji in poleg Motorole ga izdeluje še nekaj drugih proizvajalcev (Hitachi, SGS-Thomson,...). MC 6803 sodi v Motorolino družino mikrokontrolerjev M6801. deluje pri napajalni napetosti +5 V, vsi njegovi signali, preko katerih komunicira z okolico, pa uporabljajo TTL-kompatibilne signale. MC 6803 vsebuje dokaj zmogljive vhodno-izhodne enote in je uporaben v različnih aplikacijah, odvisno od dela, ki ga želimo opravljati. Sam MC vsebuje naslednje enote:

- CPE - izboljšana verzija MC 6800 z novimi ukazi
- Ura
- 128 bajtov RAM-a
- večnamenski timer - uporablja se za generiranje raznih pravokotnih izhodnih signalov, merjenje časa, proženje periodičnih prekinitev
- serijski asinhronski komunikacijski adapter - omogoča serijsko komunikacijo z zunajinimi napravami
- paralelni vmesnik port 1 - 8-bitni vmesnik, ki mu programsko dolčimo ali je določena linija vhodna ali izhodna
- paralelni vmesnik port 2 – 8-bitni vmesnik. Izhodne linije tega vmesnika si delijo 5 priključnih nožic mikrokontrolerja s signali serijskega vmesnika in timerja. Programsko lahko določimo, katera enota bo imela dostop do teh nožic.



SEZNAM UKAZOV V MC 6803

Operacija	Mnemonik	opis operacije
naloži v akumulator A	LDAA	s to operacijo lahko naložimo podatek ali vsebino pom. lok.
naloži v akumulator B	LDAB	
naloži v akumulator D	LDD	Akumulator D je 16-biten (A+B)
shrani vsebino A	STAA	shranjujemo lahko le v pomnilnik
shrani vsebino B	STAB	
shrani vsebino D	STD	upoštevati je treba predznačenje
seštej A+B=A	ABA	
seštej B in X: B+X=X	ABX	
prištej akumulatorju A	ADDA	prištejemo lahko podatek ali pomn. vseb.
prištej akumulatorju B	ADDB	
prištej akumulatorju D	ADDD	
odštej A-B=A	SBA	
odštej od A	SUBA	
odštej od B	SUBB	
odštej od D	SUBD	
zmnoži A*B=D	MUL	16-bitni rezultat
vpiši 0 v pom. lokacijo	CLR	s tem lahko čistimo pom. lokacijo
vpiši 0 v A	CLRA	
vpiši 0 v B	CLRB	
zmanjšaj za 1 pom. lok.	DEC	
zmanjšaj za 1 A	DECA	
zmanjšaj za 1 B	DECB	
zmanjšaj za 1 X	DEX	
povečaj za 1 pom. lok.	INC	
povečaj za 1 A	INCA	
povečaj za 1 B	INCB	
povečaj za 1 X	INX	
primerjaj vsebino A z B	CBA	
primerjaj z vsebino A	CMPA	
primerjaj z vsebino B	CMPB	
deli vsebino A z 2	LRSA	
deli vsebino B z 2	LRSB	
vejitev pri pogoju: C=0	BCC	C=Carry bit-1 prenos na višje mesto
vejitev pri pogoju: C=1	BCS	
vejitev pri pogoju: Z=0	BNE	
vejitev pri pogoju: Z=1	BEQ	Z = Zero bit - 1, ko je rezultat 0
vejitev pri pogoju: N=1	BMI	N = negativno
vejitev pri pogoju: N=0	BPL	
brezpogojna vejitev	BRA	

PROBLEM SEŠTEVANJA ŠTEVIL V SIMULATORJU MC 6803

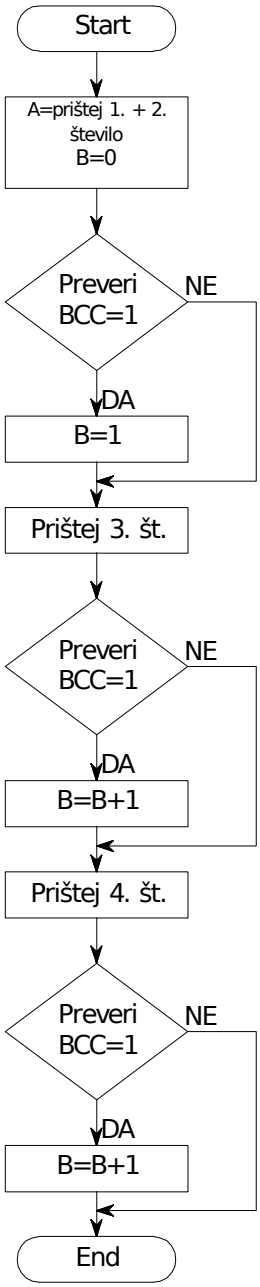
Osnovna računska operacija s katero smo se soočili, poteka v programskem jeziku assembler na zelo preprost način. V akumulator A in B naložimo dve števili in ju potem seštejemo z ukazom ADD. Če pri tem presežemo 8-bitno dolžino, program ne deluje več pravilno

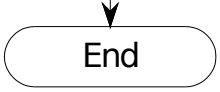
```
org $fffe ; reset vektor, ki kaže na zacetek programa
```

```
        fdb start
        org $2000
        sm $1000 55 55 ; del pomnilnika s programsko kodo
start   ldaa #0
        ldab #0
        ldaa $1000
        adda $1001 ; seštejem  prvi dve števili
        staa $1006
        stab $1005
        end
```

SEŠTEVANJE ŠTIRIH ŠTEVIL

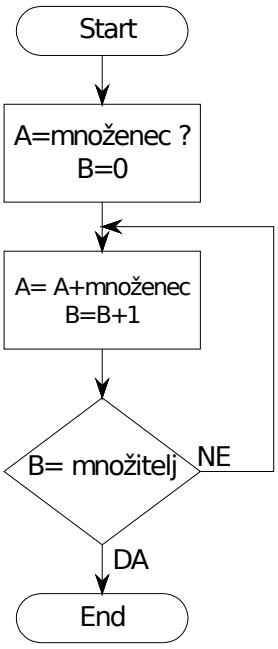
```
org $fffe ; reset vektor, ki kaže na začetek programa
        fdb start
        org $2000
```





MNOŽENJE ŠTEVIL

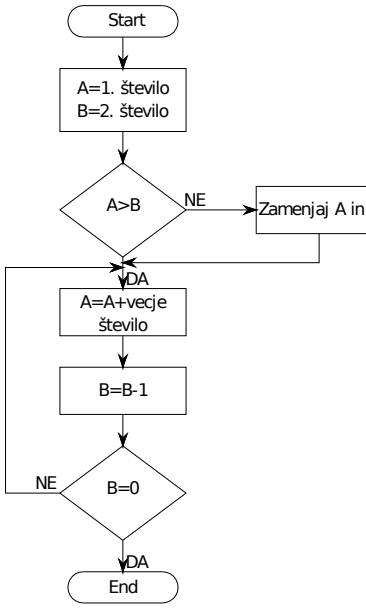
Naslednji zadan problem je bil množenje števil. Množimo lahko na več načinov, najpreprostejši je, da uporabimo ukaz MUL.




```
        org $fffe
        fdb zacni
        org $2000;del
pomnilnika s programsko kodo
zacni   sm $1000 25 50;na
1000 postavi 25, na 1001 pa 50
        ldaa $1000;v vsebino
akumulatorja a se naloži
pomnilniški naslov
        ldab $1001
        mul;zmnoži
        std $1005;akumulator a
in b skupaj
        end

        org $fffe     ; reset
vektor, ki kaže na začetek
programa
```

```
     fdb start
       org $1000
       sm $0101 3 5
start  ldaa $0101       ; a=1.število
       ldab $0102
       sba
       bcc zam
nazaj  ldaa #0
       ldab $0101      ; B=3
pon    adda $0102       ; A=A+5
       subb #1        ; B=B-1
       bne pon   ; ponovi, če B ni enak 0
       staa $0100
       bra konec    ; shrani, ko je B=0
zam ldaa $0101
      staa $0105
      ldaa $0102
      staa $0101
      ldaa $0105
      staa $0102
      bra nazaj
konec      end
```

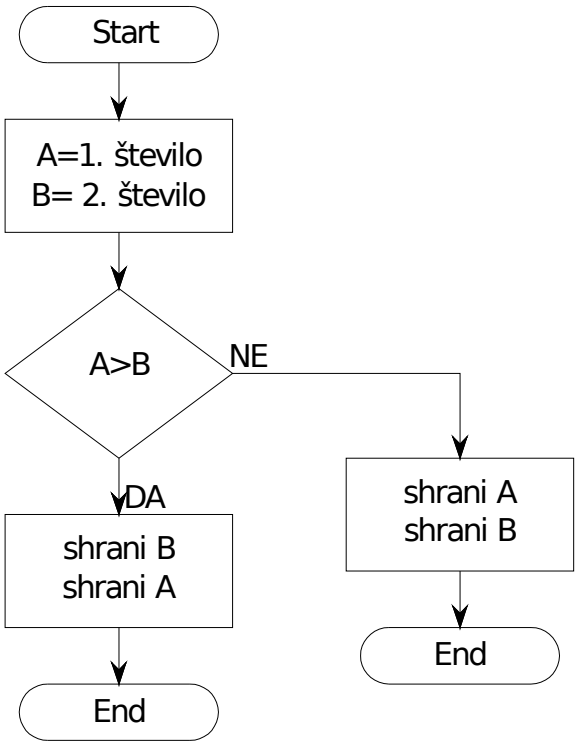


UREJANJE ŠTEVIL PO VELIKOSTI

DVE ŠTEVILI

Urejanje števil v programskem jeziku assembler je ena od težjih zadanih nalog. Pri tem moramo primerjati dve števili. Pri štirih pa se malce zaplete..

```
       org $fffe
       fdb start
       org $2000
start  ldaa #6
       staa $300A;1.število
       ldab #5
       stab $300B;2.število
       cmpa $300B;primerjaj
       bmi  ponovi;vejitev, če je N:=1, če
je 0, preskoci
       stab $4000
       staa $4001
       bra nap ;brezpogojna vejitev,
preskok, deklariramo kazalec
ponovi
       staa $4000
       stab $4001
nap    end
```



RAZVRŠČANJE 4 ŠTEVIL PO VELIKOSTI

```
        org $fffe    ; reset vektor, ki kaže na začetek programa
        fdb start
        org $2000
        sm $300a 6 8 7 2
start   ldab #0
ponovi  ldaa $300a
        cmpa $300b
        bcs nadbc
        staa $2222
        ldaa $300b
        staa $300a
        ldaa $2222
        staa $300b
nadbc   ldaa $300b
        cmpa $300c
        bcs nada
        staa $2222
        ldaa $300c
        staa $300b
```

```
        ldaa $2222
        staa $300c
nada    ldaa $300c
        cmpa $300d
        bcs nad3
        staa $2222
        ldaa $300d
        staa $300c
        ldaa $2222
        staa $300d
nad3    addb #1
        cmpb #3
        bne ponovi
        end
```

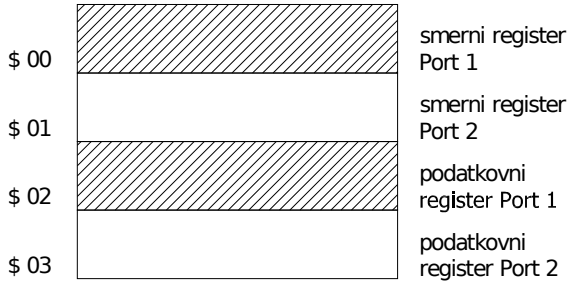
IGRANJE S VMESNIKOM PORT 1

```
      org $fffe
      fdb start
      org $2000
start ldaa #$F8
      staa $00
      ldaa #$00
      staa $01

      ldaa #%00000000
      staa $02
      ldaa #%00000000
      staa $03

      ldaa $03
      bita #%00000001
      ldaa #%00001000
      staa $02

      end
```



INDEKSNO NASLAVLJANJE V PRAKSI

Indeksno naslavljanje lahko uporabimo za seštevanje, množenje podatkov v tabelah. V našem primeru smo uporabili tabelo odvisnosti poti in časa od hitrosti. Podan je bil čas in hitrost, izračunati pa je bilo potrebno pot.

```
        org $fffe
        fdb start
        org $1000
        sm $4000 4 5 6 9 2 4 6 3
start   ldx #$4000
        ldaa #4
        staa $5555
ponovi  ldaa 0,x
        ldab 4,x
        mul
        stab 8,x
        inx
        dec $5555
        bne ponovi
        end
```

V	T	S
4	2	8
5	4	20
8	6	48
9	3	27