

Sprawozdanie Lab 1

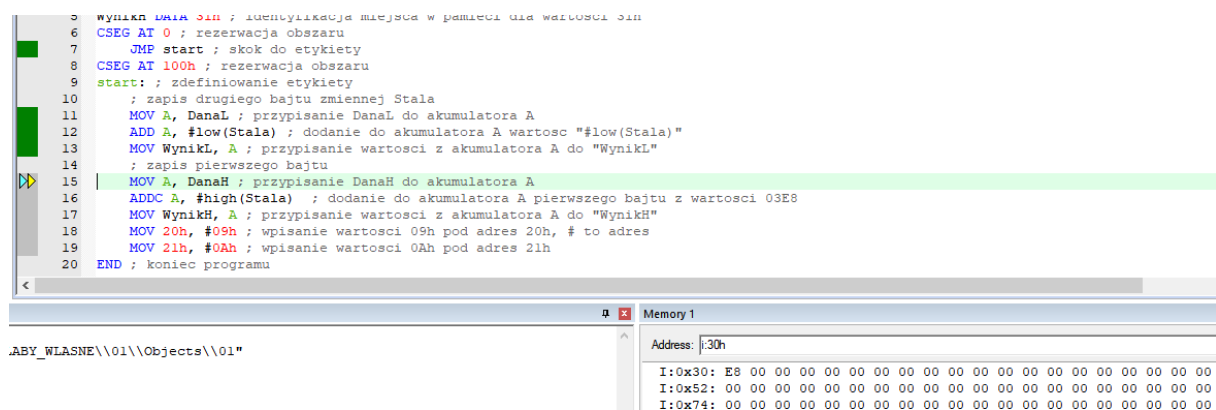
Wykonanie: Wojciech Bulek i Michał Bęben

1. Zadanie na 3.0

```
1 Stala EQU 1000 ; przypisujemy wartosc 1000 do zmiennej Stala, hex = 03E8
2 DanaL DATA 20h ; identyfikacja miejsca w pamieci dla wartosci 20h
3 DanaH DATA 21h ; identyfikacja miejsca w pamieci dla wartosci 21h
4 WynikL DATA 30h ; identyfikacja miejsca w pamieci dla wartosci 30h
5 WynikH DATA 31h ; identyfikacja miejsca w pamieci dla wartosci 31h
6 CSEG AT 0 ; rezerwacja obszaru
7 JMP start ; skok do etykiety
8 CSEG AT 100h ; rezerwacja obszaru
9 start: ; zdefiniowanie etykiety
10 ; zapis drugiego bajtu zmiennej Stala
11 MOV A, DanaL ; przypisanie DanaL do akumulatora A
12 ADD A, #low(Stala) ; dodanie do akumulatora A wartosc "#low(Stala)"
13 MOV WynikL, A ; przypisanie wartosci z akumulatora A do "WynikL"
14 ; zapis pierwszego bajtu
15 MOV A, DanaH ; przypisanie DanaH do akumulatora A
16 ADDC A, #high(Stala) ; dodanie do akumulatora A pierwszego bajtu z wartosci 03E8
17 MOV WynikH, A ; przypisanie wartosci z akumulatora A do "WynikH"
18 MOV 20h, #09h ; wpisanie wartosci 09h pod adres 20h, # to adres
19 MOV 21h, #0Ah ; wpisanie wartosci 0Ah pod adres 21h
20 END ; koniec programu
```

Skończony Program przedstawia się następująco, do każdej linii został dodany opis wskazujący na to co dokładnie robimy w danym momencie w programie. Sam proces i uruchomienie wyjaśnimy poniżej:

Pierwszym elementem działania programu jest zapis liczby 1000 w systemie dziesiętnym w pamięci procesora w systemie ósemkowym. 1000 w systemie ósemkowym to **03E8**, tak więc do pełnego zapisu takiej liczby będziemy potrzebowali dwóch komórek pamięci. Liczbę taką rozbijamy na dwie części, low i high, **03** i **E8**. W programie dzieje się to w tym miejscu:



```
5 WynikH DATA 31h ; identyfikacja miejsca w pamieci dla wartosci 31h
6 CSEG AT 0 ; rezerwacja obszaru
7 JMP start ; skok do etykiety
8 CSEG AT 100h ; rezerwacja obszaru
9 start: ; zdefiniowanie etykiety
10 ; zapis drugiego bajtu zmiennej Stala
11 MOV A, DanaL ; przypisanie DanaL do akumulatora A
12 ADD A, #low(Stala) ; dodanie do akumulatora A wartosc "#low(Stala)"
13 MOV WynikL, A ; przypisanie wartosci z akumulatora A do "WynikL"
14 ; zapis pierwszego bajtu
15 MOV A, DanaH ; przypisanie DanaH do akumulatora A
16 ADDC A, #high(Stala) ; dodanie do akumulatora A pierwszego bajtu z wartosci 03E8
17 MOV WynikH, A ; przypisanie wartosci z akumulatora A do "WynikH"
18 MOV 20h, #09h ; wpisanie wartosci 09h pod adres 20h, # to adres
19 MOV 21h, #0Ah ; wpisanie wartosci 0Ah pod adres 21h
20 END ; koniec programu
```

Memory 1

Address: 30h

I:0x30:	E8	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
I:0x52:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
I:0x74:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00

Widzimy że został zapisany drugi bajt do pamięci, **E8**, pod komórkę **30h**. **Zostało to zrobione z pomocą akumulatora do którego dodaliśmy z pomocą wbudowanej funkcji #low().** W drugiej punkcie dodajemy pozostałą część liczby, **03**, do komórki sąsiedniej **31h**:

Analogicznie postępujemy z dodawaniem pozostałych wartości, najpierw przenosimy wartość **0A** do akumulatora, a następnie dodajemy do niego **03**. Akumulator pokaże wartość **0D**. Funkcja **ADDC** różni się od zwykłego **ADD** flagą przeniesienia, ale dla podanego przykładu nie jest ona ważna.

a	0x0d
b	0x00
sp	0x07
sp_max	0x07
PC \$	C:0x0112
auxr1	0x00
dptr	0x0000
states	12
sec	0.00000436
psw	0x01

```

1  Stala EQU 1000 ; przypisujemy
2  DanaL DATA 20h ; identyfikacja
3  DanaH DATA 21h ; identyfikacja
4  WynikL DATA 30h ; identyfikacja
5  WynikH DATA 31h ; identyfikacja
6  CSEG AT 0 ; rezerwacja obszaru
7      JMP start ; skok do etykiety
8  CSEG AT 100h ; rezerwacja obszaru
9  start: ; zdefiniowanie etykiety
10     MOV 20h, #09h ; wpisanie
11     MOV 21h, #0Ah ; wpisanie
12     ; zapis drugiego bajtu z
13     MOV A, DanaL ; przypisanie
14     ADD A, #low(Stala) ; dodanie
15     MOV WynikL, A ; przypisanie
16     ; zapis pierwszego bajtu
17     MOV A, DanaH ; przypisanie
18     ADDC A, #high(Stala) ;

```

Po wszystkim ukaże nam się owoc naszych trudów:

Memory 1	
Address:	0:30h
I:0x30:	F1 0D 00 00 00 00
I:0x61:	00 00 00 00 00 00

Czyli, czytając od tyłu **0DF1h** co jest ostatecznym wynikiem. Jego poprawność można sprawdzić na jakimkolwiek dostępnym kalkulatorze:

Result

Hex value:

$$03E8 + 0A09 = \mathbf{DF1}$$

Decimal value:

$$1000 + 2569 = \mathbf{3569}$$

03E8

+ ▾

0A09

= ?

Calculate ▶

Clear

Podsumowanie:

- a) Działania na większych liczbach muszą brać pod uwagę środowisko w którym się znajdują. Komórki pamięci są ośmiobitowe i każda operacja przekraczająca maksymalną wartość FF musi być rozbita na części.
- b) Odwoływanie się do komórek pamięci wykonywane jest przez albo bezpośredni symbol w systemie szesnastkowym, albo wcześniej ustaloną stałą.
- c) ADD i ADDC są potrzebne do dodawania wartości. ADDC ma flagę przeniesienia, która wskazuje czy podczas dodawania przekroczono maksymalną wartość FF. Odejmowanie wykonywane jest z pomocą SUBB analogicznie do dodawania.