Sprawozdanie Lab 1

Wykonanie: Wojciech Bulek i Michał Bęben

1. **Zadanie na 3.0**

Obraz zawierający tekst, zrzut ekranu, oprogramowanie, Strona internetowa

Opis wygenerowany automatycznie

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:

Obraz zawierający tekst, Czcionka, oprogramowanie, Strona internetowa

Opis wygenerowany automatycznie

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**:

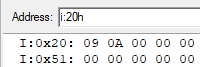
Obraz zawierający tekst, zrzut ekranu, oprogramowanie, Czcionka

Opis wygenerowany automatycznie

W ten sposób mogliśmy zapisać „dużą” liczbę w naszej pamięci rozdzielając ją na sąsiadujące bajty pamięci na naszym procesorze, tak jak jest to widoczne na załączonym obrazie.

Rzeczywistą częścią zadania jest sprawienie abyśmy mogli dodawać i odejmować liczby szesnastobitowe. Na wyższych zdjęciach było już to widoczne że dojdą dwie nowe wartości, **09 i 0A** które należy dodać do tego co mamy. Z związku z tym przeniesiemy to na górę wykonania programu.

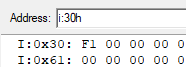
Dzięki rozdzieleniu liczby 1000 możemy dodawać po kolei części zadania. Widzimy to w akumulatorze.



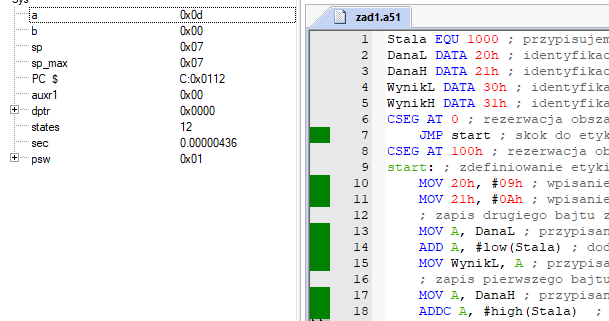
Obraz zawierający tekst, zrzut ekranu, oprogramowanie, numer

Opis wygenerowany automatycznie

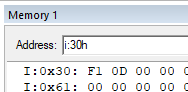
Przyjmuje on wartość **F1h** co jest wynikiem dodania E8 do 09. Tą wartość przenosimy do odpowiednich adresów wyniku tak jak to zostało opisane już wyżej.



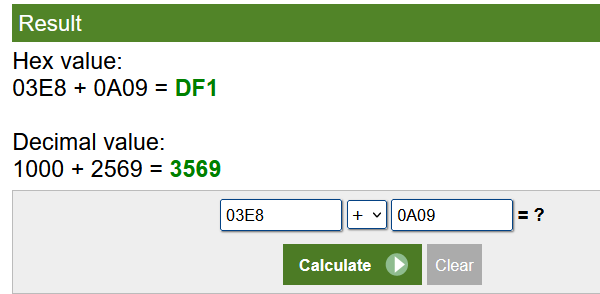
Analogicznie postępujemy z dodawaniem pozostałych wartości, najpierw przenosimy wartość **0A** do akumulatora, a następnie dodajemy do niego **03**. Akumulator pokaże wartość **0D**. **Funkcja ADDC różni się od zwykłego ADD flagą przeniesienia**, ale dla podanego przykładu nie jest ona ważna.



Po wszystkim ukaże nam się owoc naszych trudów:



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



**Podsumowanie:**

1. 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.
2. Odwoływanie się do komórek pamięci wykonywane jest przez albo bezpośredni symbol w systemie szesnastkowym, albo wcześniej ustaloną stałą.
3. 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.