Podstawy Techniki Mikroprocesorowej
LABORATORIUM 1

Wartości liczbowe:
W Asemblerze wartości liczbowe można wyrazić w formacie:

- *dziesiętnym* - 1234/1234d
- *ósemkowym* - 123o
- *szesnastkowym* - 123h, przy czym w wypadku rozpoczynania się od litery (A-F) należy dopisać na początku liczby zero (0)
- *dwójkowym* - 0111b

Znaki:
Znaki są interpretowane jako jedno lub dwubajtowe znaki ASCII. W każdym wypadku należy wpisać interesujący nas znak/znaki w apostrofy.

Ciągi znaków:
Można także deklarować ciągi znaków umieszczając je między apostrofami.

Definiowanie stałych:
Aby zdefiniować stałą należy użyć dyrektywy EQU.

```
1 <nazwa stalej> EQU <wartosc>
```

Definiowanie zmiennych:
Aby zdefiniować zmienną należy użyć dyrektywy SET.

```
1 <nazwa stalej> SET <wartosc>
```

Definiowanie "wskaźników":
W Asemblerze można deklarować stałe wskazujące na dane obszary w pamięci. Są to:

- *BIT* - adres obszaru w IRAM przechowującego bity
- *CODE* - adres w pamięci kodowej
- *DATA* - adres w pamięci IRAM adresowanej bezpośrednio

- *IDATA* - adres w pamięci IRAM adresowanej pośrednio
- *XDATA* - adres w pamięci XRAM

Akumulator (Rejestr A):
Akumulator jest 8-bitowym rejestrem przeznaczonym do krótkoterminowego zapisywania wartości. Do wartości zawartej w rejestrze można dostać się z pomocą jego symbolu *A* . Do adresu akumulatora w pamięci SFR można dostać się poprzez symbol *ACC* .

```
1 CLR A ; A - symbol akumulatora
2 PUSH ACC ; ACC - symbol adresu akumulatora
```

Można także uzyskać bezpośredni dostęp do adresów poszczególnych bitów akulumatora poprzez operator '.'.

```
1 SETB ACC.2 ; ustawienie bitu 2 akumulatora w stan wysoki
```

Rejestr B:
Rejestr B jest 8-bitowym rejestrem pomocniczym. Znajduje on swoje główne zastosowanie w operacjach mnożenia i dzielenia gidze nie może być zastąpiony żadnym innym rejestrem. Nie opłaca się go z tego powodu używać poza tymi przypadkami. Jego symbolem jest litera B. Tak jak akumulator możliwe jest adresowanie jego poszczególnych bitów za pomocą operatora kropki.

```
1 SETB B.4 ; B - symbol rejestru B
```

Rejestry R0-R7:
Uniwersalne rejestry znajdujące się domyślnie na pierwszych 8 bajtach wewnętrznej pamięci RAM (można przenieść je jednak do innego baku w pamięci). Każdy z tych rejestrów można przywołać poprzez symbol *Rn*, gdzie *n* jest numerem rejestru.

```
1 MOV R2, #1
```

Rejestrów *R*0 i *R*1 można użyć w celu przechowywania adresów do innych komórek w pamięci wewnętrznej.

Dodawanie:
Operację dodawania zawsze wykonuje się za pośrednictwem akumulatora. To zawsze w akumulatorze znajduje się pierwszy składnik dodawania i to właśnie tam zapisuje się uzyskana suma. Mamy dwa warianty dodawania. Pierwszym

jest dodawanie bez przeniesienia ADD. Drugim dodawanie z przeniesieniem ADDC, które uwzględnia w w dodawaniu przeniesienie z ostatnio wykonanego działania.

```
ADD A, <rejestr> ; dodanie z rejestru
ADD A, <adres z pamieci wewnetrznej> ; dodanie z pamieci
ADD A, @<rejestr 0/1> ; dodanie z adresu zapisanego w
   rejestrze
ADD A, #<wartosc> ; dodanie stalej

ADDC A, <rejestr> ; A + rejestr + pozyczka z porzedniego
   dodawania
```

Odejmowanie:
Operację odejmowania wykonuje się również za pośrednictwem akumulatora. To zawsze w akumulatorze znajduje się odjemna, a następnie obliczona różnica. W wypadku odejmowania zawsze uwzględniana jest pożyczka. Z tego powodu, kiedy chcemy wykonać proste odejmowanie wartości 8-bitowych należy wyzerować flagę pożyczki C.

```
SUBB A, <rejestr>
SUBB A, <adres pamieci wewnetrznej>
SUBB A, @<adres zapisany w rejestrach 0/1>
SUBB A, #<wartosc>
```

Mnożenie:
Operację mnożenia wykonuje się za pośrednictwem akumulatora oraz rejestru B. De facto sprowadza się ono do wymnożenia zawartości tych rejestrów a następnie zapisania 16-bitowego wyniku w formie $B|A$ (w B znajdzie się starszy bajt wyniku, w akumulatorze młodszy). W wypadku wystąpienia przepełnienia aktywuje się flaga CY.

```
MUL AB ; mnozenie
```

Ze względu na fakt, że taki sposób mnożenia jest relatywnie wolny to przy mnożeniu przez 2 używa się o wiele wydajniejszego przesunięcia bitowego akumulatora w lewo. Ma ono dwa warianty.

```
RLC A ; A = 2A, w CY 9 bit wyniku
```

Dzielenie:
Operacje dzielenie wykonuje się za pośrednictwem akumulatora i rejestru B. Dzieli się w nim zawartość akumulatora przez zawartość rejestru. Iloraz zapisuje się w rejestrze A, reszta z dzielenia w rejestrze B. Flaga przeniesienia

przy dzieleniu ustawi się na stan wysoki, gdy wynik dzielenie będzie mniejszy od 1.

```
DIV AB ; dzielenie
```

Podobnie jak w wypadku mnożenia, polecenie DIV jest dosyć powolne. Dlatego przy dzieleniu przez 2 lepiej jest użyć przesunięcia bitowego akumulatora w prawą stronę.

```
RRC A ; A = A/2, w CY znajduje sie reszta z dzielenia
```

Rozkaz skoku CJNE:
Rozkaz *Compare and Jump if Not Equal* przeskakuje do wskazanego miejsca w programie w wypadku, gdy zadane mu wartości nie są równe. Może być on użyty do porównania dwóch wartości. Ustawia on flagę CY na stan wysoki, gdy wartość pierwsza jest mniejsza od drugiej. W przeciwnym wypadku flaga CY jest ustawiana na stan niski.

```
CJNE <wartosc 1>, <wartosc 2>, <nazwa etykiety>
; CY = 0 <-> wartosc 1 >= wartosc 2
; CY = 1 <-> wartosc 1 <  wartosc 2
; Niewykonanie skoku w wypadku rownosci wartosci

CJNE A, <RAM>, <etykieta>
CJNE Rn, #<wartosc>, <etykieta>
CJNE @<R0/1>, #<wartosc>, <etykieta>
```

Dyrektywa CSEG AT:
Dykretywa informująca kompilator od którego miejsca w pamięci kodowej ma rozpocząć się następna sekcja kodu. Można użyć ją do rozpoczęcia kodu programu, aby wskazać, że jego kod programu rozpoczyna się w komórce zero pamięci kodowej.

```
CSEG AT <adres w pamieci kodowej>
```

Segmenty:
Segmenty to alokowane przez programistę bloki pamięci. Segmenty dzielimy ze względu na przynależność do pamięci (kodowa, wewnętrzne itd). Możemy je "ręcznie" ustawić w danym miejscu pamięci (segment absolutny) za pomocą dyrektyw:

- *BSEG AT* - segment w pamięci bitowej
- *CSEG AT* - segment w pamięci kodowej

- *DSEG AT* - segment w pamięci wewnętrznej (adresowanej bezpośrednio)
- *ISEG AT* - segment w pamięci wewnętrznej (adresowanej pośrednio)
- *XSEG AT* - segment w pamięci zewnętrznej

Istnieje jednak także możliwość deklaracji segmentów relokowalnych, których umiejscowieniem w pamięci zajmuje się konsolidator. Aby zlecić utworzenie takiego segmentu należy użyć dyrektyw:

- *nazwa SEGMENT BIT* - segment w pamięci bitowej
- *nazwa SEGMENT CODE* - segment w pamięci kodowej
- *nazwa SEGMENT DATA* - segment w pamięci wewnętrznej (adresowanej bezpośrednio)
- *nazwa SEGMENT IDATA* - segment w pamięci wewnętrznej (adresowanej pośrednio)
- *nazwa SEGMENT XDATA* - segment w pamięci zewnętrznej

Aby użyć segmentów relokowalnych należy je najpierw wybrać za pomocą dyrektywy RSEG lub CSEG (w przypadku dyrektyw kodowych).

Dyrektywa DB/DW:
Dyrektywy służące do zainicjowania obszaru w pamięci kodowej określonymi wartościami. Możemy zainicjować jeden bajt za pomocą dyrektywy DB lub dwa bajty za pomocą dyrektywy DW.

```
1 <etykieta>:  DB  <wartosc 1-bajtowa>
2 <etykieta>:  DW  <wartosc 2-bajtowa>
```

Dyrektywa DBIT/DS:
Dyrektywy służące do rezerwacji pewnej liczby bajtów pamięci. Dyrektywa DS służy do inicjalizacji w pamięci IRAM miejsca na zmienną o danej nazwie. Można użyć również dyrektywy DBIT, aby zainicjalizować miejsce w pamięci bitowej.

```
1 <etykieta>: DS <wielkosc w bajtach>
2 <etykieta>: DBIT <wielkosc w bitach>
```

Dyrektywa NAME:
Dyrektywa NAME służy do określenie nazwy modułu obiektowego, który powstaje po kompilacji programu.