Wokół języka C
Piękno (?) języka C – wskaźniki

Kontynuujemy temat zmiennych i sprawy, która też jest omówiona w drugiej części artykułu o asemblerze (strona 34).

Wskaźniki

Podczas pisania programu zmienna ma tylko nazwę, natomiast w procesorze podczas pracy programu to zarezerwowany przez kompilator „odpowiedniej wielkości kawałek" pamięci RAM. Najbardziej interesuje nas oczywiście zawartość tego „kawałka pamięci", a zwykle mniej obchodzi nas, jaki adres, czyli który numer komórki RAM, kompilator przydzieli danej zmiennej (niemniej w programie możemy określić ten adres – numer komórki za pomocą tzw. *operatora pobrania adresu* oznaczanego **&**).

Często jednak warto się interesować nie tylko zawartością, ale i adresami, choćby w przypadku tablic (i innych typów „tablicopodobnych"). Tablice najczęściej są „obsługiwane" w ten sposób, że jedna po drugiej przetwarzane są informacje z kolejnych komórek tablicy. Aby to było wygodne, warto wykorzystać specyficzny typ zmiennej, zwany ***zmienną wskaźnikową*** lub krótko ***wskaźnikiem***. ***Wskaźnik jest to zmienna zawierająca adres*** (najczęściej adres tablicy albo adres „zwykłej" zmiennej).

To jest zasadniczo proste, ale początkujący mają duże kłopoty ze zrozumieniem sensu tego dość obszernego zagadnienia. W procesorach AVR wskaźniki znajdują zastosowanie praktycznie tylko w przypadku tablic, dlatego nie musisz wgłębiać się w szczegóły. Załóżmy więc, że będziemy wykorzystywać 12-elementową tablicę znakową (statyczną, czyli o określonej wielkości), którą zadeklarujemy, podając jej rozmiar (liczbę elementów) w nawiasach kwadratowych:

```
char jakas_tablica[12];
```

Kompilator zarezerwuje dla niej dwanaście komórek pamięci RAM. Możemy „dostać się" do dowolnego elementu tablicy (zaadresować), podając nazwę tablicy i numer elementu w tablicy. Poniższa instrukcja

```
zm_a = jakas_tablica[1];
```

skopiuje do zmiennej ***zm_a*** drugi element tablicy (tak, drugi, bo numeracja elementów zaczyna się od zera)

Ale gdy na przykład mamy odczytać lub zapisać wszystkie kolejne elementy tablicy, warto byłoby to jakoś uprościć. Znakomicie pomagają w tym ***wskaźniki***, które też ***są zmiennymi, zawierającymi adres początku tablicy***.

Język C wymaga deklarowania wszystkich zmiennych, także wskaźników. Możemy zadeklarować:

```
char *zm_wsk;
```

Znak gwiazdki * informuje kompilator, że nie chodzi nam o zmienną „zwykłą", tylko o *zmienną wskaźnikową* – o ***wskaźnik***. Każda zmienna, także wskaźnik, to zarezerwowane miejsce w pamięci RAM. Będzie to określone miejsce, w którym nie będziemy przechowywać danych, tylko adres, gdzie tych danych szukać. Wskaźnik jest więc „pośrednikiem" do „właściwych danych"

Tu uważni Czytelnicy drugiej części artykułu o asemblerze i o trybach adresowania procesorów AVR (strony 34, 35) dowiedzą się o ***adresowaniu pośrednim***, gdzie numer komórki do odczytu/zapisu nie jest zawarty w treści rozkazu, tylko w jednym z rejestrów nomen omen *wskaźnikowych* (X, Y lub Z).

W grupie asemblerowych rozkazów LD (odczyt) i ST (zapis) mamy wersje „z plusem" i „z minusem", które idealnie nadają się do przetworzenia zmiennej tablicowej, zawartej gdzieś w pamięci RAM. Można w uproszczeniu powiedzieć, że w przypadku procesorów AVR zmienna wskaźnikowa finalnie będzie jednym z 16-bitowych rejestrów X, Y, Z. Zawarta tam 16-bitowa liczba – adres wskaże, gdzie w pamięci RAM są „właściwe dane".

Jest to naprawdę proste, tylko wyjaśnijmy istotny szczegół: przy deklarowaniu „zwykłych" zmiennych podany typ pokazuje wielkość zmiennej, np. *char* to typ danych wykorzystujących 1 bajt (8 bitów). Natomiast wielkość (typ) wskaźnika nie jest dowolna, tylko ściśle określona. W małych 8-bitowych procesorach ATtiny i ATmega rejestry wskaźnikowe X, Y, Z są 16-bitowe, bo są to pary „najwyższych" rejestrów 8-bitowych. A więc mogą zaadresować pamięć RAM o wielkości do 2^{16}, czyli 65536 komórek (64kB), co z powodzeniem wystarcza we wszystkich tych procesorach, zawierających znacznie mniej RAM-u. Natomiast w „dużych komputerach" zmienne wskaźnikowe mają 32 bity lub nawet więcej, zależnie od ilości pamięci RAM i architektury. Nie ma więc sensu deklarować wielkości wskaźnika – jest ona sztywno wyznaczona przez sprzęt: przez rodzaj procesora i wielkość pamięci RAM.

Tak jak przy innych zmiennych, także *przy deklarowaniu wskaźników podajemy typ, ale nie typ danych wskaźnika, tylko typ „właściwych danych", na które ten wskaźnik będzie wskazywał*, by je prawidłowo obsługiwać. Przykład poniżej:

```
char *zm_wsk; //wskaźnik obsługuje zmienną typu char
int *inny_wsk; //wskaźnik obsługuje zmienną typu int
```

Wykonując operacje z udziałem wskaźników (adresów), wykorzystujemy dwa specjalne operatory (**&** i *). Jeden, oznaczony **&** (ampersand), to *operator pobrania adresu* (zmiennej, tablicy, struktury...). Drugi oznaczony gwiazdką * to operator *dereferencji*, *wyłuskania*, który *pozwala odwołać się do „prawdziwych danych"* wskazywanych przez dany wskaźnik. Jeśli na przykład mamy w programie zadeklarowane zmienne, wskaźnikową **ws** („na razie pustą") i dwie „zwykłe" typu *int*: **zm** zainicjowaną wartością 5 i „na razie pustą" zmienną **a**

```
int *ws, zm=5, a;
```

to instrukcja pobrania adresu z operatorem **&**

```
ws = &zm
```

wpisze do zmiennej wskaźnikowej **ws** *adres* zmiennej **zm**, natomiast instrukcja

```
a = *ws;
```

wpisze do zmiennej **a** *zawartość* zmiennej wskazywanej przez wskaźnik **ws**. We wskaźniku jest adres zmiennej **zm**, więc do zmiennej **a** wpisana zastanie *zawartość* zmiennej **zm**, czyli liczba 5.

Początkujący programiści AVR-ów nie muszą wnikać w zawiłości wskaźników – na razie niech tylko uwierzą na słowo i zapamiętają, że przy seryjnym przetwarzaniu danych (tablic) warto wykorzystywać adresowanie pośrednie i wskaźniki, ponieważ to znakomicie ułatwia tworzenie programów.

A my w następnym odcinku omówimy funkcje.

Piotr Górecki