Mikrokontrolery? To takie proste...

Część 3
Mikrokontroler 8051 – opis układu

Kontynuujemy opis układów wewnętrznych mikrokontrolera 8051. W tym odcinku wyjaśnimy pojęcie i znaczenie „stosu" i jednostki arytmetyczno-logicznej

Przy okazji wszystkim „niecierpliwym" lub „wątpiącym" w swoje możliwości dotyczące programowania 8051 jeszcze raz przypominam, że w tej części cyklu nie opisujemy szczegółowo wszystkich komponentów procesora, jedynie w sposób przystępny staramy się wspólnie zrozumieć fakt istnienia tych elementów oraz ich znaczenie dla całego procesora. Autor robi to celowo, tak abyś mógł drogi Czytelniku „oswoić się" z naszym bohaterem. Wszystkie omówione elementy architektury 8051 będziemy sukcesywnie „przywoływać" z pamięci i sprawdzać ich działanie w praktyce podczas „nauki programowania 8051". Wtedy to zdobyta i „oswojona" wiedza okaże się kluczem do sukcesu którym będzie z pewnością pierwszy napisany przez Ciebie program na 8051.

Na początek krótkie przypomnienie: w poprzednim odcinku zajmowaliśmy się wewnętrzną pamięcią programu mikroprocesora oraz wewnętrzną pamięcią danych. W jej obrębie pobieżnie omówiliśmy istnienie „rejestrów specjalnych" – SFR oraz poznaliśmy funkcję licznika rozkazów PC. Szczegółowy opis SFR oraz „sprzętowe" działanie licznika rozkazów opiszemy przy okazji omawiania cyklu rozkazowego procesora oraz wstępu do asemblera. Na tym etapie jednak naszym celem będzie poznanie pozostałych bloków funkcjonalnych 8051-ki.

Wewnętrzna pamięć danych w 8052

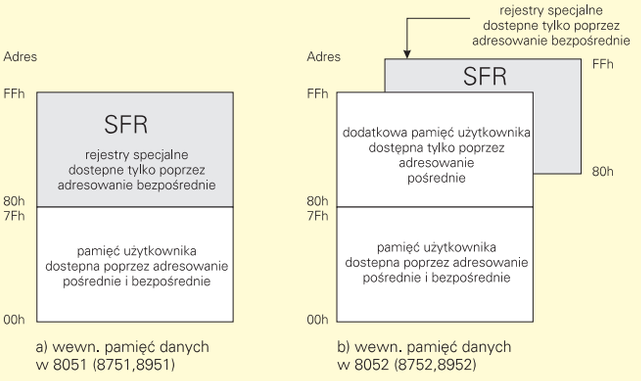
Wbrew pozorom tytuł tego akapitu nie wykracza poza ramy naszego minikursu na temat 8051. Jak już wiesz (z części I) mikrokontroler 8052 to rozszerzona wersja '51–ki, posiadająca:

– dodatkowy 16–bitowy uniwersalny układ licznikowy
– dodatkowe 128 bajtów wewnętrznej pamięci danych RAM.

I właśnie tym drugim elementem zajmiemy się teraz. Na **rysunku 1a** przedstawiłem znaną Ci już strukturę wewnętrznej pamięci danych w 8051. Jak mówiliśmy poprzednio obszar o adresach: 00h...7Fh (128 bajtów) zajmuje pamięć danych użytkownika, którą można adresować w sposób pośredni (poprzez rejestry wskaźnikowe R0 i R1) lub bezpośredni odwołując się niejako „wprost" do danej komórki pamięci. Pozostałe 128 bajtów pamięci o adresach 80h...FFh to obszar omawianych w skrócie „rejestrów specjalnych – SFR".

Obok na **rysunku 1b** pokazałem strukturę wewnętrznej pamięci danych w mikrokontrolerze 8052. Jak widać w przestrzeni tej istnieje dodatkowa, niejako zdublowana pamięć danych o adresach 80h...FFh. Przeznaczenie jej jest takie jak pamięci użytkownika o adresach 00h...7Fh, czyli w całości (128 bajtów) można ja wykorzystać do swoich potrzeb umieszczając w niej te wyniki obliczeń oraz pozostałe zmienne programowe, które po prostu „nie mieszczą" się w obrębie pamięci danych użytkownika o adresach 00h...7Fh. Ze względu jednak że ta dodatkowa, nazywana często „nakładkową", pamięć danych pokrywa się adresowo (adresy 80h...FFh) z obszarem rejestrów specjalnych SFR, należało ją w jakiś sposób rozróżnić, tak aby np. podczas odczytu lub zapisu którejś z komórek tej pamięci nie zmodyfikować przypadkiem któregoś z rejestrów specjalnych SFR.

Otóż konstruktorzy kontrolera 8052 rozwiązali ten problem w prosty sposób umożliwiając dostęp do tej dodatkowej „nakładkowej" części pamięci , jedynie za pośrednictwem adresowania pośredniego. Toteż jeżeli np. w przyszłości, drogi



Rys. 1. Pamięć w procesorach 8051 oraz 8052

Czytelniku, odwołasz się (zaadresujesz) do dowolnej komórki wewnętrznej pamięci danych w sposób bezpośredni a wskazywanym przez Ciebie adresem komórki będzie np. F0h (240 dziesiętnie) to z pewnością „dobierzesz" się do .... rejestru specjalnego o symbolu B (patrz tabela 1 w poprzednim odcinku), natomiast jeżeli wykonasz to samo tym razem jednak adresując komórkę w sposób pośredni poprzez rejestry wskaźnikowe R0 lub R1, to dokonasz zapisu (lub odczytu) komórki położonej w obszarze nakładkowym – czyli części dodatkowej pamięci danych.

Oczywiście jeżeli wykonasz tą ostatnią operację programując kostkę 8051, to w efekcie zaadresowania pośredniego komórki o adresie z zakresu 80h...FFh trafisz przysłowiową „kulą w płot", czyli nie uzyskasz oczekiwanego efektu , bo po prostu w tym obszarze 8051 adresowanym pośrednio po prostu nie ma nic. Warto o tym pamiętać, bowiem zgodnie z zasadą kompatybilności w dół, program napisany na procesor o mniejszych możliwościach (np. 8051) z pewnością będzie pracował poprawnie na 8052, ale nie odwrotnie. To samo dotyczy każdego członka rodziny MCS–51.

Stos i wskaźnik stosu

Z pojęciem „stosu" miałeś okazję, drogi Czytelniku, spotkać się w artykule omawiającym ogólne założenia dotyczące mikroprocesorów, pisaliśmy o tym w EdW. Jeżeli nie do końca rozumiesz istotę stosu postaram się Ci ją jeszcze raz przedstawić. Otóż najprościej można stos określić jako bardzo prostą w działaniu strukturę przechowującą bajty. Pod pojęciem „przechowania" rozumiemy oczywiście operacje zapisu z następnie odczytu dowolnej zmiennej lub rejestru SFR.

Wiesz już że w przypadku takich operacji tylko z udziałem np. wewnętrznej pamięci danych użytkownika, aby dokonać zapisu (odczytu) musisz daną komórkę pamięci najpierw zaadresować – czyli po prostu podać jej fizyczny adres.

W przypadku korzystania ze stosu adresowanie jest niekonieczne. Przy takim sposobie obsługi konieczne jest jednak zachowanie odpowiedniej kolejności w zapisie i odczycie tak aby nasze cenne dane nie „pomieszały się".

Otóż taki uporządkowany sposób przechowywania danych charakteryzuje właśnie stos. **Rysunek 2** wyjaśnia fizyczną budowę stosu. Jak widać wszystkie dane (bajty) przy zapisie odkładane są „na stos" jedna na drugą. Na wierzchołku stosu znajduje się zawsze ostatnio odłożona dana (w naszym przykładzie oznaczona jako X), toteż aby „dobrać się" do danej leżącej pod nią (Y) należy najpierw „zdjąć" ze stosu daną X, a potem dopiero odczytać Y. Można to porównać do stosu talerzy ustawionych jeden nad drugim. Odkładamy talerze na stos i zdejmujemy ze stosu. Nie możemy wyjąć talerza „z głębi stosu" – dostajemy się do niego dopiero po zdjęciu wszystkich stojących na nim.

„Po co jednak jest ten „stos", czy nie jest to tylko niepotrzebna komplikacja .....", z pewnością wielu z Was w tej chwili zadaje sobie to pytanie. Otóż jak się okaże później a szczególnie podczas nauki programowania, struktura ta spełnia niezmiernie ważna rolę podczas wykonywania programu przez mikroprocesor. Na tym etapie powinieneś wiedzieć tylko dwie podstawowe rzeczy: stos służy do przechowywania zmiennych lub rejestrów SFR i druga sprawa: dostęp do nich odbywa się w sposób uporządkowany – w odpowiedniej kolejności, jak opisałem wcześniej.

„No tak ale gdzie jest ten stos ?....", już odpowiadam. W przypadku procesorów rodziny MCS–51 stos umieszczony jest... uwaga !, w wewnętrznej pamięci danych użytkownika, czyli w obszarze o adresach 00h...7Fh.

Jak wynika z rysunku 2 ilość tej pamięci zajętej przez stos będzie się zmieniać i zależeć od tego ile bajtów odłożyliśmy na ten stos.

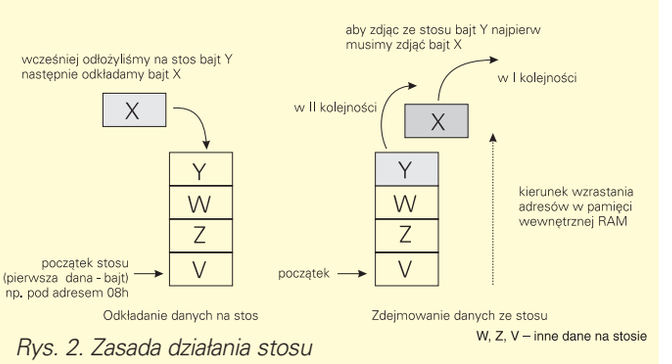
Aby ściśle określić miejsce położenia stosu, w architekturze '51–ki znajduje się tzw. licznik stosu a fachowo mówiąc „wskaźnik stosu". Fizycznie jest on po prostu 8–bitowym rejestrem w obszarze SFR, położonym pod adresem 81h (patrz tabela 1 w poprzedniej części). W mnemonice (nazewnictwie) procesorów MCS–51 posiada on symbol SP z angielskiego „stack pointer" – wskaźnik stosu.

Jego zadaniem jest automatyczne wskazywanie miejsca aktualnego wierzchołka stosu. Tak więc w przypadku odłożenia bajtu na stos, wskaźnik SP jest automatycznie (bez ingerencji programisty) zwiększany o 1, w przypadku zdjęcia danej ze stosu jest on zmniejszany. Sytuację te wyjaśnia **rysunek 3**.

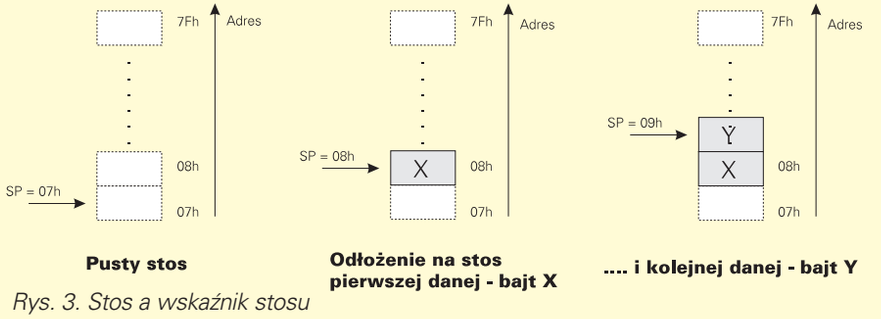
Podsumujmy więc: stos jest hierarchiczną strukturą do przechowywania danych (bajtów) z obszaru wewnętrznej pamięci RAM procesora (włączając SFR) a położenie jego wierzchołka jednoznacznie określa jego wskaźnik – SP. Przy korzystaniu ze stosu obowiązuje zasada, „ile bajtów odłożyłeś na stos, tyle potem musisz zdjąć", tak aby struktura stosu nie została zakłócona. W praktyce ma to szczególne znaczenie, bowiem stos wykorzystywany jest nie tylko poprzez świadome działanie użytkownika lecz także przechowywane są na nim ważne dla działania całego mikrokontrolera adresy powrotów z podprocedur oraz procedur obsługi przerwań, czyli innymi słowy mówiąc, aktualne zawartości 16–bitowego licznika rozkazów PC.

No tak ale przecież stos składa się z 8–bitowych komórek pamięci, a licznik rozkazów (programu PC) jest 16–bitowy. W takim przypadku procesor na stos odkłada najpierw młodszy bajt rejestru PC, a następnie starszy bajt, wskaźnik stosu SP zostaje wiec zwiększony automatycznie o 2. Tak więc w prosty sposób można przechowywać inne rejestry podwójne np. wskaźnik adresu zewnętrznej pamięci – DPTR (tabela.1), składający się z dwóch 8–bitowych rejestrów DPH (adres 83h) oraz DPL (adres 82h).

W przypadku rejestru DPTR jak i innych SFR przechowywanie na stosie odbywa się „na żądanie" użytkownika – w potrzebnym dla niego momencie. O tym jak



Rys. 2. Zasada działania stosu



Rys. 3. Stos a wskaźnik stosu

w praktyce i dlaczego przechowuje się rejestry na stosie dowiesz się drogi Czytelniku przy okazji nauki programowania 8051.

Na **rysunku 4** przedstawiono dwie sytuacje w których używany jest stos. Pierwsza dotyczy przechowania rejestru licznika rozkazów (programu) podczas obsługi procedury po nadejściu przerwania, druga obrazuje jak świadomie można wykorzystać stos do przekazywania danych pomiędzy rejestrami. W praktyce ten ostatni przypadek jest niezmiernie rzadko wykorzystywany, lecz w tym przykładzie chodzi nam o zrozumienie samego sposobu działania struktury stosu.

Na koniec dwie pozostałe ważne informacje dotyczące stosu. Otóż po włączeniu zasilania procesora (lub jego resecie oczywiście) wskaźnik stosu SP przyjmuje domyślnie wartość 07h – czyli po prostu 7, wskazując tym samym że wierzchołek stosu – adres umieszczenia następnej danej – po odłożeniu jej na stos położony będzie w wewnętrznej pamięci danych pod adresem 08h (07h +1 zgodnie z opisaną wcześniej zasadą).

Jeżeli więc odłożymy jakiś bajt na stos, najpierw licznik SP zostanie automatycznie zwiększony o 1 (wskazując teraz 08h) , a następnie do komórki pamięci o tym adresie 08h, zostanie wpisany ten bajt. Przy zdjęciu ze stosu kolejność będzie odwrotna, najpierw zdjęty zostanie nasz bajt, a następnie zmniejszony zostanie wskaźnik SP o 1.

Wskaźnik stosu SP tak jak każdy rejestr SFR może być dowolnie modyfikowany przez programistę poprzez zapisanie w nim dowolnej 8–bitowej wartości (0...255)

W praktyce jednak sytuacja taka występuje tylko wtedy, jeżeli chcemy zmienić położenie stosu (czyli go przesunąć) na początku wykonywania programu. Operacja ta z oczywistych względów ma sens jeżeli stos w danej chwili jest „pusty" , w przeciwnym razie przy lekkomyślnej modyfikacji wskaźnika SP wszystkie dane odłożone wcześniej na stos staną się niedostępne (przynajmniej z punktu widzenia działania samego stosu).

I tak jeżeli np. zechcesz drogi Czytelniku wykorzystywać wewnętrzną pamięć danych o adresach 08h...20h dla swoich potrzeb (a nie na stos) , musisz na początku swego programu zmodyfikować wskaźnik SP wpisując do niego wartość np. 20h, co jest jednoznaczne z powiedzeniem mikroprocesorowi: „... uważaj mikroprocesorze, twój stos będzie rozpoczynał się od adresu 21h, (a nie od 08h), adresy 08h...20h są przeznaczone dla moich potrzeb... ."

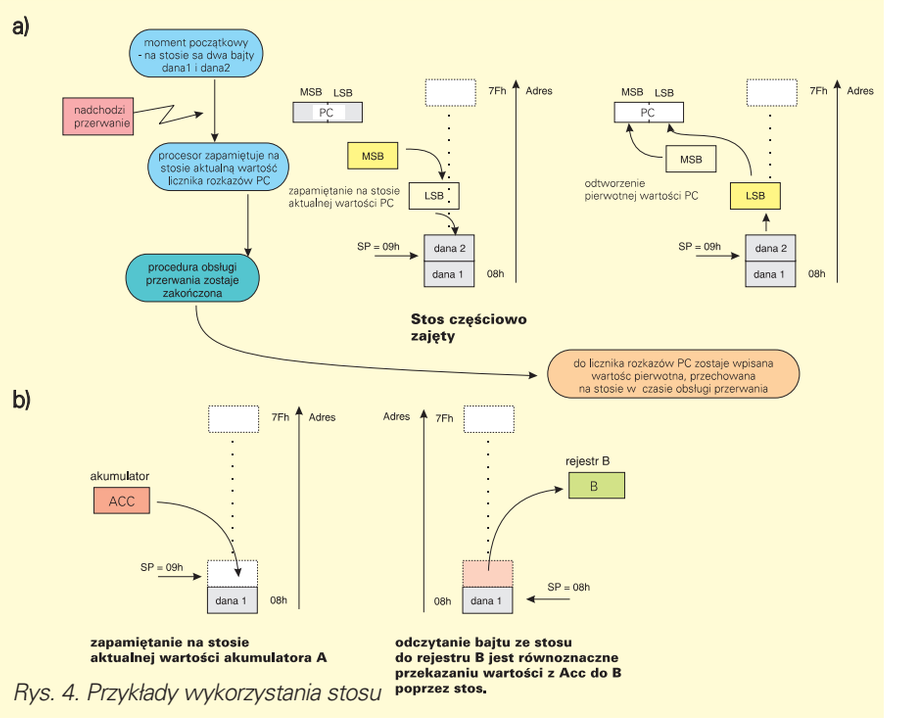
Na koniec jeszcze jedna istotna uwaga. Otóż jak widać ze sposobu działania stosu, poprzez nieumiejętne korzystanie z niego bardzo łatwo jest go „zamazać" lub mówiąc inaczej zniszczyć. Przykładem niech będzie sytuacja w której:

a) mikroprocesor wykonuje swój rutynowy program
b) nagle nadchodzi przerwanie z wejścia INT0
c) procesor przerywa działanie pętli głównej programu
d) następuje skok do wykonania procedury obsługi przerwania
e) zanim jednak to nastąpi procesor automatycznie zapisuje na stos aktualną zawartość licznika rozkazów PC (tak aby potem wiedzieć gdzie ma wrócić do pętli głównej programu)
f) po odłożeniu na stos licznika PC (2 bajty), procesor wykonuje procedurę obsługi przerwania
g) w tej procedurze użytkownik świadomie używa stosu do przechowywania tymczasowo pewnych wartości, zapisując na stos np. 3 bajty (te 3 bajty znajdą się w sąsiedztwie bezpośrednio „nad" bajtami z licznika rozkazów PC)
h) z powodu błędu w programie – popełnionego przez użytkownika – pod koniec procedury zostają zdjęte ze stosu dwa (a nie trzy) bajty, które przechowywały dane użytkownika
i) następuje zakończenie procedury obsługi przerwania, procesor zdejmuje ze wierzchołka stosu adres powrotu do pętli głównej programu (2 bajty, które wpisuje do licznika rozkazów PC) i tu następuje ..... „kompletny klops" – program prawdopodobnie „zawiesi się" lub po prostu „zwariuje".

Powód tego jest oczywisty, do licznika rozkazów PC nie zostały wpisane wcześniej przechowane 2 bajty będące pierwotną zawartością PC, a za to wpisany zostaje bajt pozostawiony przez użytkownika (zawierający najpewniej inną wartość liczbową) w procedurze obsługi przerwania, oraz starszy bajt licznika rozkazów PC.

W tej sytuacji procesor powróci w zupełnie inne miejsce programu, niż w te w którym się znajdował w momencie nadejścia przerwania, czego skutki dla działania procesora okazać się mogą opłakane.

Pamiętajmy zatem o stosie jako o ważnej strukturze w architekturze 8051, oraz o tym że tylko umiejętne iświadome, oczywiście, z niego korzysta-



Rys. 4. Przykłady wykorzystania stosu

nie przynosi często efekty w postaci znacznego przyśpieszenia działania programu oraz zmniejszenia jego rozmiarów. O tym jak w praktyce korzystać z dobrodziejstw stosu powiemy dowiemy się podczas nauki programowania.

Jednostka arytmetyczno-logiczna

Pod tym pojęciem kryje się jeden z elementów architektury 8051 odpowiedzialny za wykonywanie operacji arytmetyczno – logicznych. Blok ten nazywany w skrócie jako ALU, potrafi wykonywać operacje na liczbach (składnikach) 8–bitowych. Z matematyki wiemy że do wykonania najprostszego działania dwuskładnikowego potrzebne są: po pierwsze składniki, po drugie w wyniku działania powstaje wynik, który też należy gdzieś przechować (umieścić).

Do wprowadzenia (np. przez programistę) składników działania służą zarówno niektóre rejestry specjalne z grupy SFR jak i dowolna komórka wewnętrznej pamięci danych. Dla różnych działań występują jednak pewne ograniczenia w swobodzie umiejscawiania składników, lecz to temat na oddzielny artykuł.

Jednym z najważniejszych rejestrów z grupy SFR jest akumulator oznaczany dużą literą A (ang. „accumulator").

W tabeli 1 z poprzedniej części widać że akumulator umieszczony jest pod adresem E0h (224 dziesiętnie). Rejestr ten służy jednostce ALU za miejsce pobrania argumentu oraz umieszczenia wyniku większości operacji arytmetyczno logicznych.

Rejestr ten może być adresowany bitowo (podobnie jak bajty spod adresów 20h...2Fh – patrz poprzedni odcinek), dzięki czemu możliwe jest testowanie dowolnych jego bitów bez potrzeby wykonywania dodatkowych operacji logicznych. Muszę w tym miejscu zasygnalizować że dodatkowo rejestr A poza funkcjami związanymi z jednostką ALU służy do pobierania i umieszczania bajtów w zewnętrznej pamięci danych, dokładnie o tym powiemy w kolejnych odcinkach kursu.

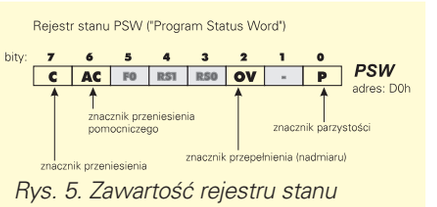
Przy przesyłaniu tego rejestru na stos (umieszczenie lub pobranie ze stosu) wykorzystuje się adresowanie bezpośrednie tego rejestru. Wtedy opisujemy go symbolem ACC (lub Acc). Dokładnie takie przypadki poznasz przy okazji programowania 8051.

Drugim po akumulatorze ważnym rejestrem współpracującym z ALU jest, także 8–bitowy, rejestr B. Służy on do umieszczenia jednego ze składników mnożenia lub dzielenia, a po wykonaniu jednej z tych operacji w rejestrze tym umieszczany jest

– w przypadku mnożenia starszy bajt 16–bitowego wyniku mnożenia dwóch liczb 8–bitowych
– w przypadku dzielenia: reszta z dzielenia dwóch liczb 8–bitowych.

Oczywiście zarówno rejestr B jak i akumulator A mogą być wykorzystywane dowolnie jako rejestry uniwersalne.

Trzecim ważnym rejestrem związanym z ALU jest „słowo stanu programu" nazywane w skrócie jako PSW (od ang. „program status word") – patrz **rysunek 5**. Z tabeli 1 możemy odczytać że rejestr ten wchodzi w skład SFR a jego adres to D0h (208 dziesiętnie). W skład tego rejestru wchodzi 8 bitów nazywanych znacznikami z których cztery informują o przebiegu wykonania operacji arytmetyczno – logicznych. I tak:



Rys. 5. Zawartość rejestru stanu

PSW.0 (bit 0) – oznaczany jako P, to znacznik parzystości, ustawiany automatycznie w każdym cyklu maszynowym wskazuje na to czy liczba jedynek (na poszczególnych) pozycjach bitowych w akumulatorze A jest parzysta (P=1) lczy nieparzysta (P=0).

PSW.2 (bit2) – oznaczany jako OV, to znacznik przepełnienia (nadmiaru), ustawiany w wyniku wykonania dodawania lub odejmowania, a przy operacji dzielenia ustawienie go wskazuje na dzielenie przez zero.

PSW.6 (bit 6) – oznaczany jako AC, to znacznik przeniesienia pomocniczego, do którego wpisywane jest przeniesienie lub pożyczka z bitu 3, wykorzystywany jest przy korekcji dziesiętnej liczb.

PSW.7 (bit 7) – znacznik przeniesienia oznaczany jako C, do którego następuje przeniesienie z najbardziej znaczącego bitu w wyniku wykonania operacji logicznych przesunięć liczb 8–bitowych lub w wypadku przekroczenia wyniku poza zakres liczb zapisanych w naturalnym kodzie dwójkowym (>255).

Pozostałe znaczniki nie mają związku z ALU, toteż ich omówieniem zajmiemy się przy innej okazji.

W praktyce najczęściej nie jest konieczne pamiętanie o wszystkich wymienionych znacznikach, no może poza znacznikiem C (PSW.7). Jak się okaże podczas nauki programowania, znaczniki te działają jak gdyby automatycznie, to znaczy istnieją instrukcje programowania 8051, które uwzględniają wspomniane znaczniki, toteż nie jest koniecznym badanie samego bitu słowa PSW, a jedynie wykonanie odpowiedniej instrukcji która uwzględni odpowiedni stan danego znacznika.

Znając pobieżnie główne 3 rejestry związane z ALU zapoznajmy się wstępnie z operacjami jakie można wykonywać przy jej pomocy na liczbach 8–bitowych, są to

a) operacje arytmetyczne

– dodawanie argumentów
– dodawanie z przeniesieniem
– odejmowanie z pożyczką

W tych trzech przypadkach pierwszy z argumentów operacji (składnik lub odjemna) umieszczana jest w akumulatorze, drugi składnik lub odjemnik umieszczony jest w wewnętrznej pamięci danych, lub jest argumentem bezpośrednim rozkazu. Wynik działania umieszczany jest w akumulatorze. Dodatkowo w słowie PSW ustawiane są odpowiednio znaczniki: przeniesienia C i nadmiaru OV, co jest sygnałem przekroczenia zakresu liczb 8–bitowych odpowiednio bez lub ze znakiem.

Pozostałe operacje arytmetyczne to:

– mnożenie dwóch 8–bitowych liczb bez znaku, gdzie jeden składnik wpisywany jest do akumulatora drugi do rejestru B, 16–bitowy wynik umieszczany jest w rejestrach B.A odpowiednio starszy bajt w B , młodszy w A;
– dzielenie dwóch liczb 8–bitowych, gdzie dzielna umieszczana jest w akumulatorze A, a dzielnik w B, 8–bitowy wynik dzielenia znajduje się po tej operacji w A, natomiast B przechowuje resztę z dzielenia.
– inkrementacja (zwiększanie o 1) lub dekrementacja (zmniejszenie o 1) akumulatora lub dowolnej komórki w wewnętrznej pamięci danych
– korekcja dziesiętna wyniku zapisanego w akumulatorze

b) operacje logiczne

– logiczna suma (OR)
– iloczyn logiczny (AND)
– różnica symetryczna (EXOR)
– negacja (NOT) zawartości akumulatora A
– przesuwanie cykliczne akumulatora w lewo lub prawo, z lub bez przeniesienia (znacznika C –> PSW.7).

Ze wszystkimi operacjami tak logicznymi jak i arytmetycznymi zapoznasz się drogi Czytelniku podczas omawiania poszczególnych instrukcji, na razie ważne jest abyś zapamiętał omówione tu podstawowe zagadnienia związane z ALU procesora 8051 i pochodnych.

Nie załamuj się, jeśli coś nie jest dla Ciebie do końca jasne. Spróbuj rozjaśnić obraz materiałem z poprzednich odcinków. Jeśli to nie pomoże, zrozumiesz te szczegóły przy omawianiu praktycznych przykładów.

Sławomir Surowiński

c.d. w EdW 8/97

Uwaga! W części 2 (EdW 6/97, s. 41) zamiast: „Pamięć mikroprocesowa" powinno być „Pamięć mikroprocesora".