„Mikroprocesor” nie zawsze znaczy to samo

W drugiej, ostatniej części artykułu autor naświetla kilka kolejnych zagadnień, z których na co dzień rzadko zdajemy sobie sprawę. Tytułowa teza jest rzeczywiście uzasadniona...

CISC czy RISC?

W miarę rozwoju techniki komputerowej dała się zauważyć tendencja do wzbogacania list rozkazów procesorów i mikroprocesorów w coraz większy asortyment złożonych, skomplikowanych instrukcji, takich jak na przykład mnożenie i dzielenie. Złożone instrukcje wymagają odpowiednio skomplikowanego układu sterowania jednostki centralnej - realizowanego z konieczności jako mikroprogramowany - i muszą być w takiej sytuacji wykonywane w ciągu kilku, kilkunastu czy nawet kilkudziesięciu taktów sygnału zegarowego. Typowym przykładem jest rdzeń ’51, w którym wykonanie pojedynczej instrukcji wymaga 12 taktów zegara lub nawet więcej lub też HC11, o jeszcze bogatszej liście rozkazów. Takie procesory nazywane są CISC – Complex Instruction Set Computer.

Analiza tysięcy programów komputerowych wykazała, że w praktyce te złożone instrukcje wykorzystywane są stosunkowo rzadko, dominują w ogromnej większości przypadków instrukcje najprostsze. Pojawił się wobec tego pomysł, by ze złożonych instrukcji w ogóle zrezygnować (bo zawsze da się je zrealizować za pomocą ciągu prostych rozkazów), pozostawiając tylko taki zestaw instrukcji, by układ sterowania jednostki centralnej dał się wykonać jako kombinacyjny, a nie mikroprogramowany. W rezultacie pojedynczy rozkaz może zostać wykonany w ciągu jednego taktu zegara, zdecydowanie przyspieszając pracę

procesora. Z drugiej strony, braki w asortymencie dostępnych rozkazów spowodowane koniecznością uproszczenia układu sterowania powodują, że niektóre operacje muszą być realizowane programowo jako szereg instrukcji, a nie tylko jedna, co z kolei czas wykonania programu wydłuża. Okazało się, że zazwyczaj maszyny zaprojektowane według nowej koncepcji pracują wyraźnie szybciej. Takie procesory nazywane zostały RISC – Reduced Intruction Set Computer, a więc komputerami o zredukowanej liście rozkazów. Nazwa jest myląca, bo wyróżnikiem procesorów RISC jest szybkość, a uboga lista rozkazów nie jest cechą pierwszoplanową, lecz wynikiem ograniczeń konstrukcyjnych. Powoduje to, że czasami producenci prymitywnych mikrokontrolerów nazywają swoje konstrukcje RISC-ami, co jest oczywistym nadużyciem. Duża szybkość działania powinna wynikać ze struktury (architektury), a nie ze stosowania wyższych częstotliwości taktowania, dlatego procesory RISC muszą spełniać jeszcze inne wymagania. Z punktu widzenia użytkownika nie jest ważny czas

wykonania pojedynczej instrukcji, istotny jest czas realizacji zadanego algorytmu obliczeń. Wynika stąd, że lista rozkazów powinna być dobrana właśnie pod kątem efektywności, a wynikowy program powinien zawierać jak najmniej instrukcji niezwiązanych bezpośrednio z realizacją algorytmu, a więc przede wszystkim instrukcji transferu danych. Ponadto wykonanie rozkazu nie powinno niszczyć żadnego z argumentów, dlatego „rasowy” RISC jest zawsze trzyadresowy. Ponieważ obowiązuje przy tym rygorystyczna zasada, że pojedynczy rozkaz wykonywany jest w dokładnie jednym takcie zegara, nie może on być kompletowany z kilku słów pamięciowych – musi być wobec tego krótki, co oczywiście implikuje niewielką długość pól adresowych. Ponadto w jednym takcie zegara nie można z pamięci zewnętrznej pobrać 2 argumentów i odesłać do niej wyniku. Z tych powodów takie procesory wykonują operacje zawsze tylko na zawartości rejestrów wewnętrznych a nie komórkach pamięci. Zawierają większą liczbę takich rejestrów (16, 32 lub nawet więcej), a wśród rozkazów operujących na pamięci dopuszczalne są tylko rozkazy transferu danych do i z rejestrów. Pobranie w wykonanie takiego rozkazu w jednym takcie zegara jest możliwe tylko w architekturze harwardzkiej - i stanowi to kolejny wyróżnik procesorów RISC. Oczywiście szerokość słowa pamięci programu musi być na tyle duża, by mieścił się w nim kompletny rozkaz (jeden rozkaz – jedna komórka pamięci). Procesor pracuje “na zakładkę” - w czasie wykonywania jednego rozkazu równocześnie pobierany jest rozkaz następny. Technika taka nazywana jest z angielskiego pipeliningiem, na co nie ma - jak dotychczas - dobrego polskiego określenia.

Mikrokontrolery RISC zwykle nie mają zaimplementowanego stosu umieszczonego w pamięci (z przyczyn jak wyżej), zamiast niego występuje w jednostce centralnej specjalny rejestr służący do przechowywania adresów powrotnych w instrukcjach skoku ze śladem (CALL). Czasami jest spotykany stos sprzętowy, wykorzystujący nie pamięć, ale kilka przeznaczonych do tego dodatkowych rejestrów, umieszczonych bezpośrednio w jednostce centralnej. W razie konieczności stos użytkownika może być realizowany drogą programową.

Dążenie do maksymalizacji prędkości pracy procesorów RISC i ich specyficzne cechy skutkują pewnymi komplikacjami w opracowywaniu oprogramowania. Niewielki asortyment rozkazów, brak możliwości bezpośredniego operowania na komórkach pamięci, brak stosu, konieczność programowej realizacji wielu instrukcji dostępnych w procesorach CISC powoduje, że programowanie staje się trudniejsze. Tym niemniej inne zalety zaowocowały wyraźnie ostatnio zauważalną migracją nowych mikrokontrolerów w stronę struktury RISC, czasem przy pewnej rezygnacji z niektórych bardziej rygorystycznych założeń (trzyadresowość). Można tu wymienić AVR’y, CoolRisc firmy Xemics i - może nieco paradoksalnie - MSP430, pomimo jego architektury von Neumanna i dopuszczalności operowania na komórkach pamięci. Jest to dobry przykład wyjścia poza pewne ograniczenia struktury RISC ale z zachowaniem jej zalet. Jeżeli operandy umieszczone są w rejestrach wewnętrznych, to pojedynczy rozkaz wykonywany jest w jednym takcie zegara. Programista ma jednak także możliwość operowania na danych zawartych w komórkach pamięci, płacąc za to udogodnienie wydłużeniem czasu realizacji. Oczywiście żaden mikrokontroler “czystym”

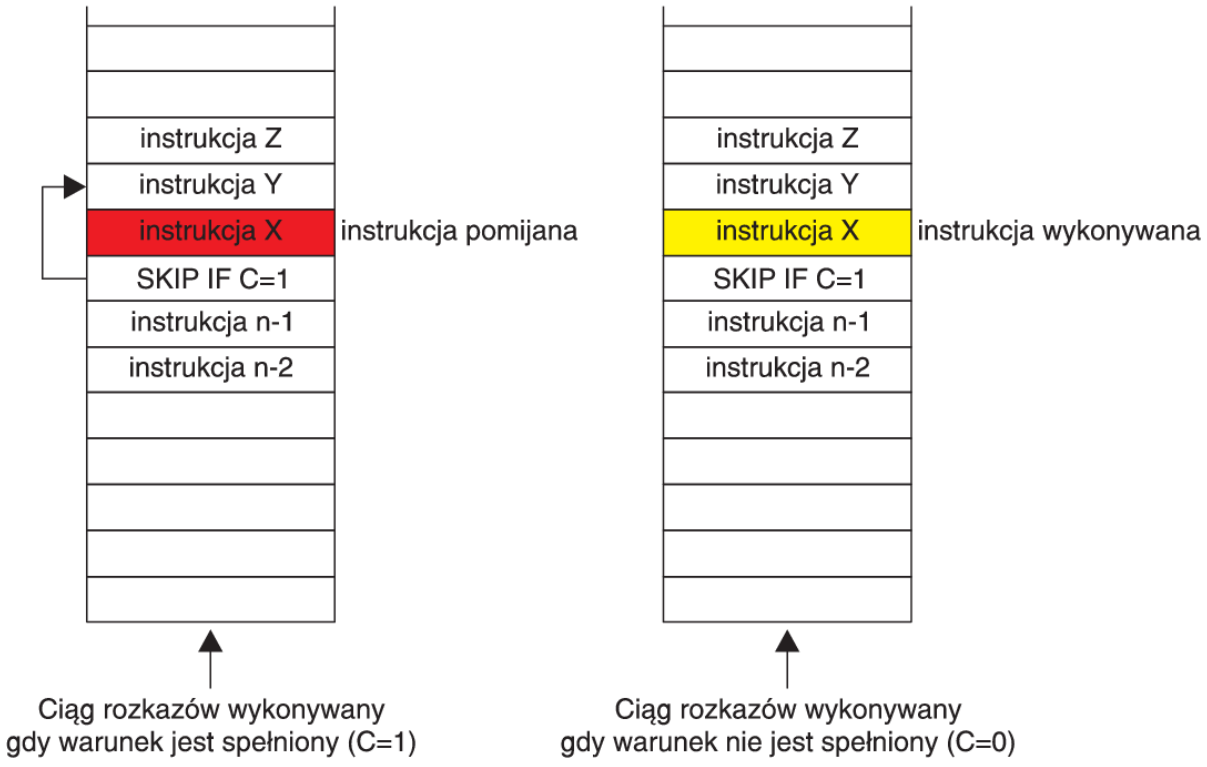
RISC-iem być nie może, chociażby ze względu na konieczność obsługi wbudowanych układów peryferyjnych, ale najważniejszy postulat, aby jedna instrukcja wykonywana była w jednym takcie zegara, bywa z powodzeniem realizowany.

JUMP IF... czy SKIP IF...?

W niemal każdym użytecznym programie potrzebne są instrukcje skoków warunkowych (rozgałęzień), uzależniające dalsze wykonywanie programu od wyniku wcześniejszych operacji. Z punktu widzenia wygody programowania i zwartości programu, im większy jest asortyment takich skoków – tym lepiej. Przykładem rdzenia mikrokontrolera bardzo bogato pod tym względem wyposażonego może być AVR. Operacja skoku warunkowego jest w zasadzie dwuargumentowa - jednym argumentem jest warunek skoku (np. stan znacznika przepełnienia, parzystości, przeniesienia połówkowego, wyzerowanie rejestru itp.), drugim – docelowy adres skoku. Oczywiście występowanie części adresowej powoduje, że w trybie adresowania bezpośredniego rozkaz staje się długi. Dla jego skrócenia stosowane są zatem zwykle inne tryby adresowania, a więc pośrednie poprzez rejestr lub – przede wszystkim – względne. Całkowite usunięcie pola adresowego stało się możliwe po wprowadzeniu instrukcji typu SKIP IF zamiast JUMP IF, w mikrokontrolerach po raz pierwszy chyba w PIC’ach. Ta bardzo wygodna także dla programisty instrukcja może być interpretowana jako warunkowy skok w przód o jedną pozycję i skutkuje po prostu ominięciem następnego rozkazu jeżeli warunek jest spełniony. W przeciwnym przypadku rozkaz ten jest wykonywany. Zamiast warunkowego skoku mamy zatem warunkowe wykonanie pojedynczego rozkazu (**rysunek 11**). Rozkaz ten może być całkiem dowolny, w szczególności może to być instrukcja skoku bezwarunkowego, ale także np. ustawienie pojedynczego bitu czy też wywołanie procedury (CALL). Zwiększa to istotnie elastyczność programowanie, pociąga za sobą wyraźne zmniejszenie liczby etykiet w programie i podnosi jego przejrzystość.

Wadą takiego rozwiązania jest to, że zawsze potrzebna jest ta dodatkowa instrukcja nawet, jeżeli następuje po niej zwykły rozkaz skoku. Ponadto asemblery nie wyróżniają zwykle tych warunkowo wykonywanych instrukcji i w trakcie analizy programu łatwo to przeoczyć. W zasadzie mikroprocesor powinien mieć zaimplementowane w liście rozkazów obie wersje realizacji warunkowych rozgałęzień programu, jest to jednak przypadek rzadki. Wyróżnia się tu AVR, w którym występują skoki warunkowane stanem dowolnego znacznika w rejestrze statusu, a ponadto instrukcje SKIP uzależnione od stanu dowolnego bitu w rejestrze lub w jednym z pierwszych 32 rejestrów wejścia/wyjścia. Niestety, z powodów dla autora całkowicie niezrozumiałych, rejestr statusu zawierający znaczniki, ma adres 63 i dla niego nie można stosować

Konstruktorzy, szczególnie amatorzy, stosują najczęściej elementy najlepiej im znane. Wiele osób podchodzi przy tym często do swojego wybrańca w sposób nadzwyczaj emocjonalny, uważając go za niekwestionowanego światowego lidera. Raz na kilka lat wskazany byłby jednak rodzaj rozwodu, a przynajmniej separacji ze swoim dotychczasowym ukochanym i jeżeli nie stała zmiana swojego elektronicznego partnera, to przynajmniej jakiś „skok w bok” i chociażby czasowe posmakowanie czegoś innego.



Rysunek 11. Ilustracja działania rozkazu SKIP IF (dla przykładowego warunku C=1)

instrukcji SKIP IF. Bez tego mankamentu możliwe byłoby zamienne wykorzystywanie obu dostępnych instrukcji dla realizacji rozgałęzień programu, warunkowanych stanem dowolnego z wielu dostępnych w rejestrze statusu znaczników.

Specyfika mikrokontrolerów

System mikroprocesorowy powstaje przez połączenie jednostki centralnej (CPU, czyli “rdzenia” mikroprocesora), pamięci operacyjnej i różnego rodzaju urządzeń peryferyjnych. Jeżeli wszystkie te elementy zostają zintegrowane w jednym układzie scalonym, tworzą mikrokomputer jednoukładowy (embedded microcomputer), mogący samodzielnie, bez dodatkowych komponentów, pełnić funkcję systemu mikroprocesorowego. Traci się wówczas co prawda zaletę uniwersalności, bo na asortyment i parametry wbudowanych w układzie scalonym podzespołów użytkownik nie ma żadnego wpływu i nie ma możliwości ich modyfikacji ani rozbudowy, ale ogromnie zyskuje na prostocie. Ze względu na te ograniczenia mikrokomputery jednoukładowe wykorzystywane są masowo przede wszystkim do realizacji prostych, dedykowanych układów sterowania (kuchenka mikrofalowa, pralka) i stąd ich częściej obecnie spotykana nazwa mikrokontrolerów. Trzeba tu jednak wyraźnie podkreślić, że w systemach uniwersalnych, szczególnie konfigurowalnych przez użytkownika, mikrokontrolery nie mogą efektywnie zastąpić mikroprocesorów. Umieszczenie wszystkich elementów systemu w jednym układzie scalonym pociąga za sobą pewne konsekwencje, mające istotny wpływ na rozwiązania architektoniczne.

Współczesne mikrokontrolery mają wbudowaną reprogramowalną pamięć programu typu Flash, zazwyczaj programowalną w systemie (ISP), bez konieczności umieszczania układu w podstawce programatora.

Pojawia się wówczas w sposób naturalny względna łatwość implementacji architektury harwardzkiej, bądź też eliminacji niektórych wad von Neumanna. Na przykład w 80C166 i pochodnych z 16-bitowej wewnętrznej pamięci programu odczytywane mogą być równocześnie od razu dwa słowa, formując 32-bitowy rozkaz.

Zanika wyraźne rozróżnienie bloku rejestrów wewnętrznych i wewnętrznej pamięci danych, czasami zupełnie znika także blok rejestrów (jak w ‘196) . W mikroprocesorach uniwersalnych rejestry wewnętrzne są zdecydowane uprzywilejowane w stosunku do komórek pamięci. W ‘51 sytuacja staje się wręcz odwrotna: istnieje zaledwie kilka rozkazów, które mogą operować tylko na rejestrach R0...R7 (właściwie tylko jeden – *CJNE Rn,#data,rel*, natomiast bardzo wiele rozkazów działa tylko na pamięci, ale nie na rejestrach (operacje logiczne, bitowe, obsługa stosu). W szczególności nie istnieje rozkaz przenoszenia danych pomiędzy rejestrami (*mov R3,R2*), co z całą pewnością jest unikalną cechą tego mikrokontrolera. Rozkaz przesłania zawartości jednej komórki pamięci do innej w trybie adresowania bezpośredniego jest za to jak najbardziej dostępny (*mov 56,65*).

Cenną zaletą wielu mikrokontrolerów jest rozwinięta obsługa operacji bitowych (zerowanie, ustawianie i negowanie pojedynczych bitów, skoki warunkowane stanem bitu). Ma to bardzo istotne znaczenie przy realizacji układów sterowania. Nieco mniej istotny jest asortyment operacji arytmetycznych, chociaż ostatnio ulega to zmianie, wraz z rosnącym rozpowszechnieniem systemów pomiarowych i kontrolno-pomiarowych, w których w naturalny sposób pojawia się zapotrzebowanie na arytmetyczną obróbkę wyników (skalowanie, linearyzacja, konwersja jednostek itp.).

Rzuca się w oczy ogromne zróżnicowanie mikrokontrolerów pod względem wielkości, możliwości, ceny, obudów – od 8 do grubo ponad 100 wyprowadzeń. Pojawiły się mikrokontrolery bardzo proste i tanie (PIC, Cx051, ATtiny). Być może podstawowym parametrem zaczyna stawać się pobór mocy i związana z tym możliwość długookresowego zasilania bateryjnego. Wiąże się z tym także rozszerzanie zakresu dopuszczalnej częstotliwości zegara nie tylko w górę, ale także w dół (np. 4 kHz). Elastyczność w doborze chwilowej wartości częstotliwości taktowania pozwala na optymalny wybór pomiędzy szybkością i poborem mocy (Dallas, Xemics, TI).

Nie można także pominąć wyraźnie ostatnio obserwowalnej ewolucji mikrokontrolerów w kierunku programowalnych systemów na chipie (FPSLIC, PSoC, Triscend).

Mikroprocesory specjalne

Asortyment produkowanych obecnie mikroprocesorów i mikrokontrolerów jest przeogromny. Przykładowo, pod koniec 2002 roku produkowano na świecie kilkadziesiąt rodzin samych tylko mikroprocesorów 32-bitowych, przy czym każda rodzina składa się z licznych egzemplarzy o zróżnicowanych parametrach. Co prawda wiele z mikroprocesorów występuje jedynie w postaci “wirtualnej”, jako komponenty IP (Intelectual Property) przeznaczone do docelowego wbudowania w inne układy, istniejąc tylko jako zweryfikowane opisy w jednym ze stosowanych w tym celu języków opisu sprzętu (VHDL, Verilog).

Oczywiście najbardziej rozbudowane mikroprocesory, o dużych możliwościach obliczeniowych i przeznaczone do pracy wieloprogramowej, są zwykle najdroższe. Równocześnie istnieje duże zapotrzebowanie na układy prostsze, ale też i bardziej atrakcyjne eko-

nomicznie – stąd tak duże zróżnicowanie oferty rynkowej, obejmującej zarówno elementy za 2 jak i za 2000 zł. Niektóre zastosowania wymagają szczególnych cech mikroprocesorów. Inne wymagania mogą być stawiane układom w systemach radarowych, inne w samochodach, inne w telefonach komórkowych, a jeszcze inne w systemach medycznych. Znajduje to swoje odzwierciedlenie w rozwiązaniach architektonicznych, czego szczególnie wyrazistym przykładem mogą być procesory sygnałowe (DSP – Digital Signal Processing). Łączą one przy tym bardzo dużą moc obliczeniową ze stosunkowo umiarkowaną ceną, co stało się możliwe dzięki ogromnemu zapotrzebowaniu i naprawdę masowej produkcji.

Procesory sygnałowe przeznaczone są – jak samo ich określenie na to wskazuje – do cyfrowej obróbki sygnałów “z natury” analogowych, odpowiadających chociażby dźwiękom, w tym muzyce, czy też obrazom. Okazuje się, że większość operacji niezbędnych do przetwarzania tego typu sygnałów, takich jak kodowanie, dekodowanie, kompresja, filtracja, eliminacja zakłóceń, daje się zazwyczaj sprowadzić do ciągu wielokrotnie i cyklicznie powtarzanych jednostkowych instrukcji typu MAC (Multiply And Accumulate). Instrukcja taka działa na trzech operandach i powoduje wymnożenie dwóch argumentów (zwykle jest to jakaś zmienna i odpowiadający jej współczynnik, czyli stała programowa), a następnie dodanie wyniku mnożenia do zawartości specjalnego rejestru – zwanego akumulatorem – zawierającego już rezultat poprzednich tego samego typu operacji. Nie ma on rzecz jasna, poza nazwą, niczego wspólnego ze standardowym akumulatorem maszyn jednoadresowych. Takie zadanie może być oczywiście zrealizowane w każdym mikroprocesorze, czy też mikrokontrolerze, nawet w 8051, ale pojawia się tu krytyczny problem z czasem realizacji. Procesory sygnałowe stosowane są w aplikacjach wymagających jak największej szybkości, narzuconej po prostu strumieniem cyfrowych danych wejściowych, będących reprezentantem stosunkowo szybko zmieniających się wielkości analogowych. Dlatego też w procesorach DSP jednostka MAC jest zawsze realizowana sprzętowo, tak, aby cały rozkaz mógł być wykonany w jednym takcie zegara. Ponieważ jednocześnie, z uwagi na wymaganą dokładność przetwarzania, procesory takie operują na danych 16- lub 32-bitowych, stanowi to poważne wyzwanie konstrukcyjne, tym bardziej, że akumulator powinien mieć długość odpowiednio większą, aby wielokrotnie powtarzane sumowanie wyników mnożenia nie spowodowały przepełnienia. Co prawda ewentualne wystąpienie przepełnienia można kontrolować programowo, i w razie potrzeby korygować wynik, ale wymaga to dodatkowych instrukcji i czasu poświęcanego na ich realizację. W prostych, 16-bitowych, stałoprzecinkowych procesorach sygnałowych akumulator ma zazwyczaj długość 40 bitów. Osobom, które otarły się o projektowanie układów cyfrowych, można zaproponować przymiarkę do opracowania układu o 72 wejściach i 40 wyjściach, w którym dopuszczalne jest wystąpienie każdego z ponad 4722366483000000000000 wektorów wejściowych. Możliwość wystąpienia przepełnienia w trakcie powtarzanych wielokrotnie operacji jest zresztą zmorą programistów DSP i stanowi jedną z zasadniczych przyczyn ewolucji w kierunku jeszcze bardziej skomplikowanych 32-bitowych procesorów zmiennoprzecinkowych, w których ten problem praktycznie nie występuje.

Poza jednostkami MAC (czasem nawet kilkoma, często mogącymi pracować równocześnie) w procesorach sygnałowych stosowanych jest też wiele innych, niestandardowych rozwiązań, w tym specjalne tryby adresowania i nietypowe formaty danych. To, że układy takie, mimo stosunkowo bardzo dużego stopnia komplikacji, pozostają relatywnie tanie, zawdzięczamy masowej produkcji – w zasadzie w każdym telefonie komórkowym musi znajdować się element pełniący taką funkcję, chociaż obecnie jest już zazwyczaj tylko częścią większego, wyspecjalizowanego układu scalonego.

Kryteria oceny mikrokontrolerów

W prostych aplikacjach daje się zauważyć przygniatająca przewaga ilościowa mikrokontrolerów nad mikroprocesorami uniwersalnymi, szczególnie w konstrukcjach amatorskich. Niestety nie istnieje mikrokontroler doskonały, bo z przyczyn wskazanych wyżej każda konstrukcja jest jakimś kompromisem. Dobór mikrokontrolera do konkretnego zastosowania powinien być zatem poprzedzony oceną przydatności różnych układów. W Polsce (i nie tylko) panuje pewna monokultura klonów 8051, zresztą mikrokontrolera o wielu naprawdę ciekawych rozwiązaniach i zaprojektowanego przed niemal 25 laty chyba optymalnie, biorąc pod uwagę występujące ograniczenia technologiczne i ekonomiczne. Tym niemniej jednoadresowość, brak symetrii i niektóre inne mankamenty powodują, że obecnie jest to już jednak rozwiązanie nieco archaiczne. Jest stosowany głównie “siłą rozpędu” i przyzwyczajeń projektantów, mimo powszechnej dostępności nowszych i w jakimś stopniu lepszych układów. Niektóre jego cechy (chociażby bogaty asortyment operacji bitowych) są jednakże nadal nie do pobicia i mogą budzić zazdrość użytkowników chociażby znacznie nowocześniejszych AVR’ów. Ocena jakości i przydatności mikrokontrolerów musi być zatem wielopłaszczyznowa i uwzględniać wiele cech i parametrów, najistotniejszych w konkretnej aplikacji. Bogaty asortyment i jakość układów peryferyjnych kompensować może mankamenty jednostki centralnej (vide: PIC’e), a zbyt mała pamięć programu uniemożliwić zastosowanie elementu z innych względów idealnego (chociażby CoolRisc’a).

Najważniejsze cechy jednostki centralnej to:

- wieloadresowość,
- liczba rejestrów wewnętrznych,
- tryby adresowania,
- lista rozkazów, w tym operacje bitowe,
- liczba znaczników i asortyment skoków warunkowych,
- symetria i ortogonalność,
- długość słowa danych i programu,
- obsługa przerwań,
- szybkość i elastyczność doboru częstotliwości taktowania.

Dla pamięci:

- wielkość ROM i RAM, przestrzenie adresowe,
- rodzaj pamięci programu, sposób programowania,
- dodatkowa pamięć EEPROM dla danych.

Układy peryferyjne:

- dostępny asortyment - porty, układy licznikowo-czasowe (timery z ewentualnymi opcjami capture i compare), CRC, I2C, U(S)ART, CAN, USB, DAC, PWM, ADC, sprzętowa mnożarka,
- parametry i sposób obsługi.

Inne:

- osiągalność handlowa, popularność,
- przyzwyczajenia projektanta, cena itp.,
- napięcie zasilania i pobór mocy,

- rodzaj obudowy,
- dostępność i cena narzędzi uruchomieniowych.

Jak widać dobór mikrokontrolera dla konkretnej aplikacji nie zawsze jest prosty i często wymaga przeanalizowania wielu aspektów zagadnienia. W praktyce konstruktorzy, szczególnie amatorzy, stosują najczęściej elementy najlepiej im znane, co zresztą nie jest niczym nagannym i pozwala na szybsze opracowanie projektu. Wiele osób podchodzi przy tym często do swojego wybrańca w sposób nadzwyczaj emocjonalny, uważając go za niekwestionowanego światowego lidera. Raz na kilka lat wskazany byłby jednak rodzaj rozwodu, a przynajmniej separacji ze swoim dotychczasowym ukochanym, i jeżeli nie stała zmiana swojego elektronicznego partnera, to przynajmniej jakiś “skok w bok” i chociażby czasowe posmakowanie czegoś innego.

Maciej Nowiński

—

Artykuł ukazał się w Elektronice Praktycznej nr 1/2004