„Mikroprocesor” nie zawsze znaczy to samo

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

CISC czy RISC?

W miarę rozwoju techniki komputerowej obserwowana jest tendencja do wzbogacania list instrukcji procesorów i mikroprocesorów o coraz bardziej złożone, takie jak na przykład mnożenia i dzielenia. Złożone instrukcje wymagają odpowiednio skomplikowanego układu sterowania jednostki centralnej - realizowanego z konieczności jako mikroprogramowany - i muszą być wykonywane w ciągu kilku, kilkunastu czy nawet kilkudziesięciu taktów sygnału zegarowego. Typowym przykładem jest rdzeń mikrokontrolera '51, w którym wykonanie pojedynczej instrukcji wymaga nawet 12 lub więcej taktów zegara albo HC11, o jeszcze bogatszej liście instrukcji. Takie procesory nazywane są CISC - *Complex Instruction Set Computer*.

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

Z drugiej strony, ograniczenie asortymentu dostępnych instrukcji, spowodowane koniecznością uproszczenia układu sterowania, powoduje, że niektóre operacje muszą być realizowane programowo w wielu instrukcjach, a nie tylko w jednej, co z kolei wydłuża czas wykonania programu. Okazało się bowiem, że maszyny zaprojektowane według nowej koncepcji zazwyczaj pracują wyraźnie szybciej. Takie procesory nazwane zostały RISC - *Reduced Intruction Set Computer*, a więc komputerami o zredukowanej liście instrukcji. Nazwa jest myląca, bo wyróżnikiem procesorów RISC jest ich szybkość, a uboga lista instrukcji nie jest cechą pierwszoplanową, lecz wynikiem ograniczeń konstrukcyjnych. Powoduje to, że czasami producenci prymitywnych mikrokontrolerów nazywają je RISC-ami, co jest oczywistym nadużyciem. Duża szybkość działania powinna wynikać ze struktury (architektury), a nie z zastosowania wyższych częstotliwości sygnału taktowania. Procesory RISC muszą spełniać jeszcze inne wymagania.

Z punktu widzenia użytkownika nie jest ważny czas wykonania pojedynczej instrukcji, lecz istotny jest czas realizacji zadanego algorytmu obliczeń. Wynika stąd, że lista instrukcji 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 instrukcji nie powinno kasować żadnego z argumentów, dlatego „rasowy“ RISC jest zawsze trzyadresowy. Ponieważ obowiązuje przy tym rygorystycznie zasada, że pojedynczy rozkaz wykonywany jest dokładnie w 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 komórek 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 i wykonanie takiego rozkazu w jednym takcie zegara jest możliwe tylko w architekturze harwardzkiej - i stanowi to kolejny wyróżnik procesorów RISC. Oczywiście długość 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*, co tłumaczone jest jako *przetwarzanie potokowe*.

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 zauważalną ostatnio migracją rozwiązań 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 umieszczane 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 operacji.

Oczywiście żaden mikrokontroler „czystym" RISC-em 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.

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 rozwód, a przynajmniej separacja ze swoim dotychczasowym ukochanym.

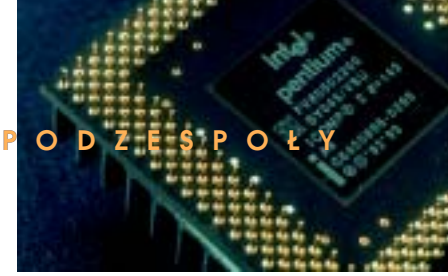
JUMP IF... czy SKIP IF...?

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 wyposażonego pod tym względem 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.), a drugim docelowy adres skoku. Występowanie części adresowej powoduje, że w trybie adresowania bezpośredniego rozkaz staje się długi. Do 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 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 (**rys. 11**). Rozkaz ten może być dowolny. W szczególności może to być instrukcja skoku bezwarunkowego, ale także np. ustawienie pojedynczego bitu czy też wywołanie procedury (CALL). Zwiększa to istotnie elastyczność programowania, pociąga za sobą wyraźne zmniejszenie liczby etykiet w programie i podnosi jego przejrzystość.

Wadą takiego rozwiązania jest to, że zawsze potrzebna jest dodatkowa instrukcja, nawet jeżeli następuje po niej 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 instrukcji obie wersje warunkowych rozgałęzień programu, ale jest to jednak przypadek rzadki. Wyróżnia się tu AVR, w którym występują skoki warunkowane stanem dowolnego znacznika w rejestrze statusu, a także instrukcje SKIP uzależnione od stanu dowolnego bitu w rejestrze lub w jednym z pierwszych 32 rejestrów wejścia/wyjścia. Niestety rejestr statusu zawierający znaczniki ma adres 63 i dla niego nie można stosować instrukcji SKIP IF. Bez tego mankamentu możliwe byłoby zamienne wykorzystywanie obu dostępnych instrukcji dla realizowania 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 rodzaj i parametry wbudowanych w układ scalony bloków funkcjonalnych użytkownik nie ma żadnego wpływu i nie ma możliwości ich modyfikacji ani rozbudowy, ale system 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 część-



za 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, 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 jego architektura wynika z jakiegoś kompromisu. 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 interesujących rozwiązaniach, 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 układ nieco archaiczny. 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 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 CoolRisca).

Najważniejsze cechy jednostki centralnej określające jej walory użytkowe 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, sprzętowy multiplikator,
- parametry i sposób obsługi.

Inne:
- dostępność, 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 parametrów. 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". Chociażby chwilowe posmakowanie czegoś innego.

Maciej Nowiński