„Mikroprocesor” nie zawsze znaczy to samo

Zaczniemy od banału: technika mikroprocesorowa opanowała już niemal wszystkie dziedziny życia. Banał drugi: zdecydowana większość elektroników ma na swoim koncie choćby najprostsze opracowania mikroprocesorowe. Banał trzeci: większość z nich (nas) zna tylko jedną lub co najwyżej dwie ich rodziny, sądząc przy tym często, że ich budowa jest zgodna z powszechnie obowiązującymi zwyczajami i stanowi niepodważalny standard.
Na koniec wstępu nie-banał: różnorodność dostępnych na rynku rozwiązań i pomysłowość ich konstruktorów są doprawdy imponujące, co powoduje, że współczesny konstruktor nie powinien pozwolić sobie na marazm i tkwienie w ciepłej „dolince“ przyzwyczajeń. Postaramy się nieco w tym pomóc.

Von Neumann vs Harvard

W historycznie najstarszych protoplastach współczesnych komputerów, opartych głównie na technologii przekaźnikowej, stosowana była najbardziej oczywista koncepcja architektoniczna, w której przetwarzane dane oraz ciąg rozkazów - tworzący program - były rozdzielone i przechowywane na odrębnych nośnikach informacji. Przykładowo w najbardziej dojrzałej konstrukcji Niemca Konrada Zusego (**fot. 1**) V4 (*Versuchsmodell 4*) z 1945 roku program był umieszczony na perforowanej taśmie filmowej (ze względu na jej większą trwałość mechaniczną niż taśmy papierowej) i z niej kolejno były wczytywane i wykonywane instrukcje. Miejscem przechowywania przetwarzanych danych były 1024 rejestry przekaźniko-

Fot. 1. Konrad Zuse przy jednym ze swoich komputerów

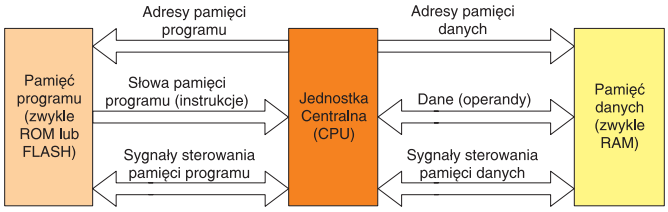
we o długości 32 bitów każdy. Maszyna Zusego po wielu wojennych i powojennych perypetiach już jako Z4 (**fot. 2**) trafiła do Szwajcarii, gdzie ją z powodzeniem eksploatowano aż do lat sześćdziesiątych.

Rozwiązania konstrukcyjne zastosowanego przez Zusego były efektywne tylko w powolnych urządzeniach przekaźnikowych, w których czas realizacji rozkazu był porównywalny, czy też nawet dłuższy, od czasu wczytywania go z mechanicznego nośnika. W najbardziej znanym i pierwszym w USA komputerze przekaźnikowym MARK-I, zbudowanym na Uniwersytecie Harvarda i oddanym do eksploatacji w 1944 roku, czas wykonania dodawania 2 liczb wynosił jedną trzecią sekundy, a dzielenia aż 10 sekund. Mimo tego MARK-I był przez wiele lat bardzo intensywnie wykorzystywany, głównie w amerykańskim programie atomowym, pracując często po 24 godziny na dobę. Rozwiązania konstrukcyjne stosowane w maszynach przekaźnikowych były pierwowzorem współczesnego modelu architektonicznego, w którym pamięci programu i danych są rozdzielone fizycznie (czyli mają oddzielne magistrale i/lub sygnały sterujące) oraz logicznie (obsługiwane są różnymi instrukcjami). O tego typu komputerach czy mikrokontrolerach mówi się, że mają architekturę harwardzką, wszakże tylko wtedy, gdy z obu pamięciami możliwa jest równoczesna komunikacja. Koncepcja taka jest obecnie często spotykana, chociaż, poza niewątpliwymi zaletami, wykazuje także pewne wady.

W 1945 roku zupełnie nowatorską strukturę komputera zaproponował genialny matematyk amerykański pochodzenia węgierskiego, John von Neumann. W maszynie von Neumanna zarówno program, jak i dane umieszczone są w jednej tylko, wspólnej pamięci. Te dwa rodzaje informacji stają się wobec tego fizycznie nierozróżnialne, a ich identyfikacja następuje jedynie poprzez miejsce, w którym we wspólnej pamięci są umieszczone, a więc przez adres. W początkowym okresie idea von Neumanna była dość trudna realizacyjnie, wymagając znacznej pojemności pamięci, nieco później jednak dzięki swoim niewątpliwym zaletom opanowała na wiele lat technikę komputerową. Tym niemniej od kilku lat daje się zaobserwować dość wyraźne odejście od klasycznej wersji koncepcji von Neumanna oraz zwrot w kierunku architektury harwardzkiej.

Jakie są wobec tego najważniejsze cechy każdego z rozwiązań?

Fot. 2. Komputer Z4



Rys. 3. Schemat blokowy komputera o architekturze harwardzkiej

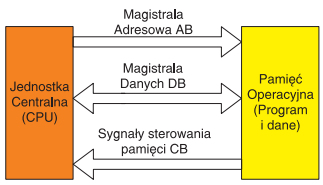
Klasyczne architektury komputerów

Komputery, w których pamięci programu i danych są rozdzielone fizycznie (czyli mają oddzielne magistrale i/lub sygnały sterujące) oraz logicznie (obsługiwane są różnymi instrukcjami) są nazywane komputerami o architekturze harvardzkiej.

Z kolei architektura von Neumanna charakteryzuje się tym, że zarówno program, jak i dane umieszczone są w jednej, wspólnej pamięci.

Von Neumann

- Program może być umieszczony (i uruchomiony) w dowolnym obszarze pamięci, także z założenia przeznaczonym dla danych. Pozwala to na dynamiczną („w biegu") wymianę fragmentów programu podczas normalnej pracy komputera, łatwe jest też uruchamianie i testowanie nowych programów. Nie ma ograniczenia na objętość każdego z bloków pamięci, istnieje tylko ograniczenie globalne wynikające z całkowitej pojemności pamięci. W tej samej pamięci można albo umieścić większy program, operujący na niewielkiej liczbie danych, albo też mniejszy program i dużo danych.
- Stałe programowe (np. tablica z generatorem znaków alfanumerycznych, definiującym kształt znaków na wyświetlaczu) mogą być bez żadnych problemów umieszczane i odczytywane (a także czasami modyfikowane) z pamięci programu przy wykorzystaniu standardowych rozkazów.
- W najbardziej klasycznej formie rejestry obsługujące urządzenia wejścia/wyjścia mogą być traktowane jak normalne komórki pamięci, z pełną dostępnością wszystkich rozkazów i trybów adresowania jednostki centralnej.
- Budowa systemu, z pojedynczymi magistralami



Rys. 4. Schemat blokowy komputera o architekturze von Neumanna

obsługującymi obie pamięci, staje się zdecydowanie prostsza, a układy scalone mają mniej wyprowadzeń.
- Wykonanie pojedynczego rozkazu wymaga zwykle wykonania kilku operacji na pamięci.

Harvard

- Długości słowa pamięci programu i danych mogą być i zazwyczaj bywają różne - np. dane są 8-bitowe, a słowo pamięci programu 16-bitowe.
- Istnieje potencjalna możliwość równoczesnej komunikacji jednostki centralnej z obu pamięciami.

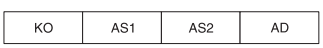
To właśnie ta ostatnia zaleta nabiera ostatnio decydującego znaczenia. Jeżeli konstrukcja jednostki centralnej pozwala na odczyt nowego rozkazu z pamięci programu jeszcze podczas transferu operandów niezbędnego dla realizacji poprzedniej instrukcji, wykonanie programu jest zdecydowanie szybsze, szczególnie, gdy kompletny rozkaz mieści się w pojedynczym słowie pamięci. Wada związana z niemożnością przechowywania stałych i tablic w pamięci programu bywa eliminowana przez wprowadzenie specjalnych rozkazów przenoszenia danych, operujących na pamięci programu.

W praktyce często spotykane są architektury mieszane, niebędące żadną z poprzednich w stanie „czystym". Typowym przykładem jest mikrokontroler 8051.

Maszyny jedno-, dwu- oraz trzyadresowe i tryby adresowania

Każdy rozkaz komputera musi zawierać kompletny zestaw informacji pozwalający na jego poprawną interpretację i wykonanie. Informacje te dzieli się na dwie zasadnicze części: część operacyjną i część adresową. Jak sama nazwa wskazuje, część operacyjna określa rodzaj operacji, jaką ma dany rozkaz wykonać (np. dodanie dwóch argumentów albo przesłanie zawartości jednej z komórek pamięci do innej). Liczba stosowanych w praktyce operacji nie jest zbyt duża, rzędu kilkudziesięciu, i bez problemów da się zakodować na niewielkiej liczbie bitów. Pojedynczy bajt, czyli uporządkowanych 8 bitów, wystarcza zazwyczaj z zapasem. Rozkazy działają jednak na jakichś argumentach i w wyniku ich realizacji powstają wyniki (argumenty i wyniki noszą nazwę operandów). Wieloargumentowe operacje zawsze można sprowadzić do ciągu operacji co najwyżej dwuargumentowych, dlatego, ze względu na dążność do maksymalnej prostoty przy zachowaniu dostatecznej uniwersalności, niezmiernie rzadko (ale jednak) spotykane są instrukcje operujące na liczbie argumentów większej od dwóch. Tak więc w najbardziej ogólnym przypadku rozkaz powinien zawierać kod operacji KO (**rys. 5**) oraz adresy wszystkich operandów, a zatem adres pierwszego argumentu AS1, adres drugiego argumentu AS2 oraz adres komórki pamięci, do której należy wysłać wynik AD.

Takie maszyny nazywane są trzyadresowymi. Nawiasem mówiąc, istnieje jeszcze bardziej ogólna struktura rozkazu, zawierająca kolejne pole adresowe: adres komórki pamięci programu, z której ma zostać pobrany i wykonany następny rozkaz. Koncepcja taka należy na razie do egzotycznych. Powszechnie są stosowane raczej znacznie prostsze maszyny sekwencyjne, a więc takie, w których na-



Rys. 5. Format rozkazu trzyadresowego i jego pola informacyjne

stępny w kolejności realizacji rozkaz pobierany jest po prostu z następnej komórki pamięci programu - dodatkowy adres staje się wtedy zbędny (wyjątek stanowią tylko specjalne rozkazy skoków). Warto jednak zauważyć, że w systemach wieloprocesorowych odejście od sekwencyjności pozwala na wyraźne poprawienie efektywności realizacji całego programu, jednakże kosztem zdecydowanego powiększenia trudności w tworzeniu oprogramowania. Maszyny trzyadresowe pozwalają na uzyskanie zwartego programu, w szczególności niemal zbędne staje się stosowanie instrukcji przenoszenia danych. Wykazują jednak istotną wadę - część adresowa staje się bardzo długa, rozkaz zajmuje w pamięci dużo miejsca, a jego skompletowanie i wykonanie pochłania sporo czasu. Wyjaśnić to może przykład prostego hipotetycznego mikroprocesora trzyadresowego, o 8-bitowym słowie pamięciowym i 16-bitowym adresie. Pojedynczy rozkaz dwuargumentowy musi składać się z 7 bajtów (1 bajt to kod operacji, a 6 bajtów to 3 adresy), jego pobranie wymaga zatem 7 dostępów do pamięci. Do jego wykonania jest jeszcze konieczny odczyt 2 argumentów oraz zapis wyniku do pamięci, czyli dodatkowe 3 cykle komunikacji z pamięcią - razem aż 10. Z tego powodu maszyny 3-adresowe spotykane są bardzo rzadko wśród prostych mikroprocesorów, stanowią natomiast naturalne rozwiązane w dużych maszynach, w których cały, długi rozkaz mieści się w jednym lub co najwyżej 2 słowach pamięciowych.

Aby skrócić część adresową, można zmniejszyć liczbę pól adresowych, ograniczyć w jakiś sposób ich długość albo też zastosować oba rozwiązania równocześnie. Najprościej usunąć trzeci adres (wyniku). W rezultacie wynik jest transferowany do domyślnego, z góry określonego miejsca. Najczęściej jest wpisywany w miejsce jednego z argumentów powodując tym samym jego zniszczenie. Zwyczajowo jest to pierwszy z argumentów, ale spotykane są też rozwiązania, w których argument zastępowany wynikiem można wybierać. W takiej sytuacji poza kodem operacji i dwoma adresami argumentów w rozkazie musi pojawić się ta dodatkowa informacja, nazywana modyfikatorem adresu. Ogólnie rzecz ujmując, modyfikator adresu determinuje sposób interpretacji przez jednostkę centralną części adresowej rozkazu. Maszyny dwuadresowe pod względem funkcjonalnym są niemal równoważne trzyadresowym, przy wyraźnie krótszej części adresowej. Jest to obecnie najczęściej stosowane rozwiązanie w mikroprocesorach i mikrokontrolerach średniej, ale jednak nie najwyższej klasy.

Usunięcie kolejnego pola adresowego z rozkazu prowadzi do maszyny jednoadresowej, w której w rozkazie wskazać można tylko jeden z argumentów operacji. Zarówno drugi argument jak i wynik muszą być adresowane w sposób domyślny (implikowany). W tym celu wprowadza się specjalny, wyróżniony rejestr związany z jednostką arytmetyczno-logiczną, zwany akumulatorem, który zawiera drugi argument operacji i do którego wpisywany jest wynik po jej wykonaniu. Długość rozkazu ulega radykalnemu skróceniu, ale w programie poza rozkazami związanymi bezpośrednio z realizacją algorytmu obliczeń pojawiają się bardzo liczne rozkazy transferu operandów pomiędzy akumulatorem i pamięcią. Powoduje to wzrost stopnia komplikacji programu, powiększa jego rozmiar i wydłuża czas wykonania, a intensywnie wykorzystywany akumulator tworzy wąskie „gardło". Maszyny jednoadresowe mają jednak przy tym niezaprzeczalną zaletę - są bardzo proste, a przez to i tanie. Pewną, dość wyraźną poprawę funkcjonalności można uzyskać, jeżeli istnieje możliwość wyboru umieszczania wyniku albo w akumulatorze, albo w miejscu adresowanego argumentu. Takie mikroprocesory nazywane bywają półtoraadresowymi. Przykładem mogą być tu mikrokontrolery PIC. Z kolei dość typowym przykładem mikrokontrolera jednoadresowego z nieznacznym odchyleniem w stronę półtoraadresowości jest '51.

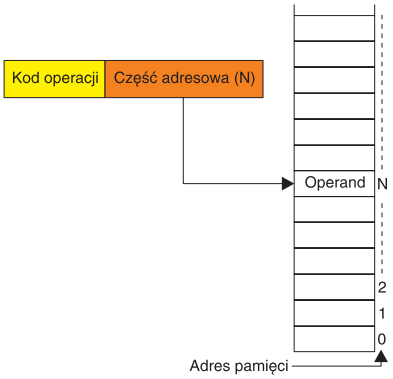
Inną możliwość ograniczenia długości części adresowej rozkazu stwarza skracanie pól adresowych, przy zachowaniu oczywistej zasady jednoznacznego wskazywania położenia operandu w pamięci. Osiąga się to poprzez stosowanie różnych trybów adresowania.

Najbardziej oczywistym trybem adresowania jest adresowanie bezpośrednie (**rys. 6**), w którym pole adresowe rozkazu zawiera po prostu adres operandu. Jedną z wad tego trybu, poza długością pola adresowego, jest fakt, że adres ten jest stały i nie można go w programie zmieniać (nie może być zatem wyliczany). Jeżeli - przykładowo - należy zwiększyć o jeden zawartość 100 kolejnych komórek pamięci, to w takim trybie adresowania należy w programie 100-krotnie umieścić instrukcję inkrementa-

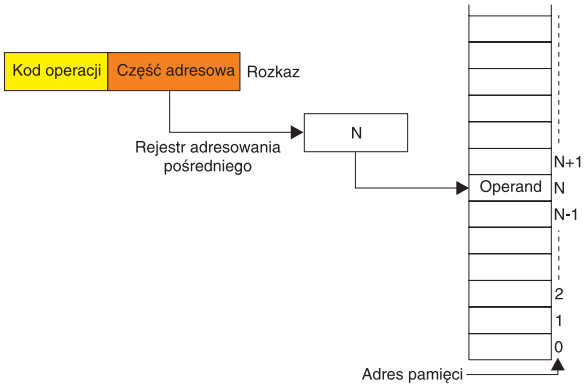
Symetria i ortogonalność jednostki centralnej

Przez symetrię rozumiane jest równouprawnienie wszystkich rejestrów jednostki centralnej i ich dostępność dla wszystkich funkcji.

Ortogonalność to dostępność wszystkich zaimplementowanych w danej jednostce centralnej trybów adresowania we wszystkich instrukcjach i dla wszystkich operandów.



Rys. 6. Adresowanie bezpośrednie

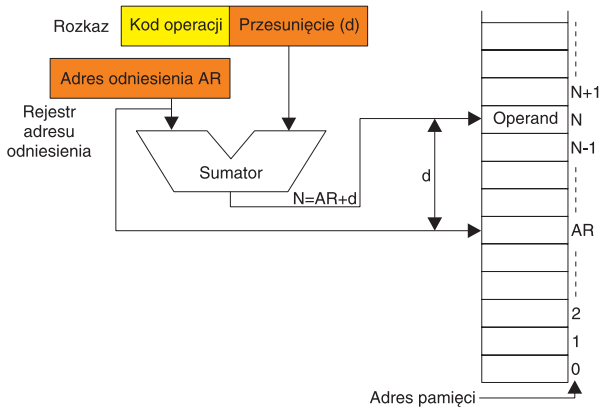


Rys. 7. Adresowanie pośrednie

cji, za każdym razem operującą na innym adresie. Aż się prosi, aby pojedynczą instrukcję powtarzać w pętli 100 razy, zmieniając przy tym wraz z kolejnymi obiegami pętli adres argumentu. Czas wykonania takiego fragmentu programu co prawda się nie zmniejszy (a nawet wyraźnie powiększy), ale program będzie zawierał tylko kilka instrukcji zamiast stu.

Wyliczanie adresu operandu umożliwia adresowanie pośrednie (**rys. 7**), w którym pole adresowe rozkazu nie zawiera bezpośrednio adresu operandu, a jest jedynie pośrednim wskaźnikiem miejsca, w którym tego adresu należy poszukiwać (jest to więc jakby adres adresu). Miejscem tym może być pamięć (rzadziej, raczej w rozwiązaniach wyższej klasy) albo wyróżniony rejestr jednostki centralnej. W tym ostatnim, dominującym w prostych mikroprocesorach przypadku, uzyskuje się przy okazji radykalne zmniejszenie długości pola adresowego, bo rejestrów służących do adresowania pośredniego jest zazwyczaj niewiele i do ich rozróżnienia wystarcza kilka bitów. W mikrokontrolerach '51 do adresowania pośredniego 8-bitowego służą wyłącznie rejestry R0 i R1 (pole adresowe skraca się zatem do jednego bitu), a do adresowania 16-bitowego przeznaczono tylko jeden rejestr DPTR (pole adresowe w ogóle znika, ale pozostaje oczywiście modyfikator adresu!). W PIC-ach do adresowania pośredniego przeznaczono tylko jeden rejestr-fantom. W AVR-ach są to trzy pary rejestrów X, Y, Z. W MSP430 każdy z rejestrów może być wskaźnikiem adresowym. Z punktu widzenia programisty im więcej istnieje rejestrów do adresowania pośredniego, tym lepiej. Z punktu widzenia konstruktora mikroprocesora jest dokładnie odwrotnie.

Adresowanie pośrednie znalazło bardzo szerokie zastosowanie w technice komputerowej i ma liczne odmiany. Może też być łączone z innymi trybami adresowania, w szczególności względnym. Zaletą - poza możliwością wyliczania adresu - jest krótkie pole adresowe rozkazu. Jeżeli rejestry adresowe mają wystarczającą długość, to możliwe jest zaadresowanie dowolnej komórki pamięci. Wadą jest to, że do rejestru należy



Rys. 8. Adresowanie względne

wcześniej wpisać docelowy adres, a więc często potrzebna jest do tego dodatkowa instrukcja, a czasem nawet dwie. Najbardziej efektywne staje się adresowanie pośrednie przy wielokrotnie powtarzanym dostępie do kolejnych komórek pamięci, np. przy pracy z danymi mającymi postać tablic, szczególnie wtedy, gdy istnieje możliwość automatycznej inkrementacji bądź dekrementacji rejestru adresowego (co jest technicznie stosunkowo proste). Ilustruje to prosty program dla mikrokontrolera '52, wstępnie zerujący wszystkie komórki pamięci:

```
        mov R0,#255
        clr A
zeruj:  mov @R0,A
        djnz R0, zeruj
```

Obsługa stosu też jest przykładem specyficznego rodzaju adresowania pośredniego, w którym rejestrem adresowym jest wskaźnik stosu, a każdy zapis czy też odczyt związany jest z jego automatyczną inkrementacją bądź dekrementacją.

Istnieją liczne odmiany adresowania pośredniego, które nie będą tu bliżej omawiane - indeksowe, bazowe, indeksowo-bazowe, stronicowe, cykliczne (cyrkulacyjne) i inne.

Kolejnym, ważnym trybem jest adresowanie względne (relatywne) - **rys. 8**. W tym trybie pole adresowe nie zawiera adresu operandu, ale jego przesunięcie (*displacement*, d) względem jakiegoś adresu bazowego. Innymi słowy, w trybie takim w polu adresowym nie ma informacji, że operand jest zawarty w komórce pamięci o przykładowym adresie 2345, jest natomiast informacja, że operand jest umieszczony w komórce o adresie o 123 większym od adresu odniesienia. Adresem odniesienia bywa zawartość specjalnych rejestrów (np. indeksowego) albo aktualny stan licznika rozkazów. Ten ostatni przypadek tworzy adresowanie względne bieżące, stosowane przede wszystkim, ale bynajmniej nie wyłącznie, w instrukcjach skoków. Przesunięcie jest traktowane zazwyczaj jako liczba ze znakiem, może być więc dodatnie albo ujemne. Zakres adresowania względnego jest zwykle ograniczony, co pozwala na skrócenie części adresowej kosztem pewnego zmniejszenia elastyczności programowania. W mikrokontrolerach '51 są możliwe skoki względne w zakresie -128...+127 w stosunku do aktualnego stanu licznika rozkazów (wskazującego zawsze następną do wykonania instrukcję). W AVR-ach jest to tylko -64...+63 (ale dotyczy słów 16-bitowych), a z kolei w MSP430 aż -512...+511. Jedną z zalet adresowania względnego bieżącego jest fakt, że jego konsekwentne stosowanie pozwala na uzyskanie wynikowych programów posiadających cechę przesuwalności (relokowalnych). Program taki może zostać umieszczony i uruchomiony w dowolnym obszarze pamięci, a nie tylko w jednym, z góry określonym miejscu. Ułatwia to m.in. gospodarowanie wieloma programami we wspólnej pamięci i przełączaniem pomiędzy nimi realizowanych aktualnie zadań.

Nie można pominąć także adresowania natychmiastowego (**rys. 9**), w którym pole adresowe rozkazu zawiera nie adres argumentu czy też wskaźnik adresowy, ale sam argument - stałą programową. Oczywiście taki tryb adresowania ma sens tylko dla argumentów, a nie dla wyniku (bo oznaczałoby to, że wynik jest znany przed wykonaniem instrukcji). Typowym przykładem może być rozkaz:

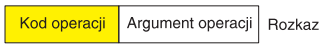
```
anl P3,#80h
```

dla '51.

Z trybami adresowania wiążą się dwa ważne pojęcia: symetrii i ortogonalności. Przez symetrię rozumiane jest równouprawnienie wszystkich rejestrów jednostki centralnej i ich dostępność dla wszystkich funkcji, w szczególności jako wskaźników adresowania pośredniego albo np. rejestrów indeksowych. Symetria jest cechą bardzo przydatną, ale stosunkowo rzadko spotykaną. O mikrokontrolerach '51 można by było mówić, że mają symetryczny rdzeń, gdyby adresowanie pośrednie mogło się odbywać nie tylko poprzez rejestry R0 i R1, ale także przez wszystkie pozostałe (R2...R7). W wielu mikroprocesorach poszczególne rejestry pełnią tylko określone zadania i nie mogą być wykorzystywane do innych celów. Przykładem ewolucji w kierunku symetrii mogą być mikroprocesory rodziny '86. W 8086 istniały wyraźnie wyróżnione, odrębne rejestry dla adresowania bazowego, indeksowego itp., w następcach (386, 486, Pentium) to ograniczenie stopniowo zanikało. Symetryczne są mikroprocesory rodziny 68000. Również w mikrokontrolerze MSP430 niemal każdy z 16 rejestrów wewnętrznych może przechowywać dane, pełnić funkcję wskaźnika adresowego albo też rejestru indeksowego.

Ortogonalność z kolei to dostępność wszystkich zaimplementowanych w danej jednostce centralnej trybów adresowania we wszystkich instrukcjach i dla wszystkich operandów. Jest to cecha bardzo rzadka, chociaż jej obecność prowadzi do bardzo interesujących rozwiązań. Z popularnych mikrokontrolerów największy stopień ortogonalności wykazuje MSP430, w którym dostępny jest przykładowo tak egzotyczny rozkaz, jak skok ze śladem (CALL) w trybie adresowania pośredniego z postinkrementacją. Zazwyczaj poszczególnym instrukcjom i ich operandom przyporządkowane są jednak tylko pewne wybrane tryby adresowania.

Dostępność operandów w wielu trybach adresowania, symetria i ortogonalność są bardzo istotnymi cechami jednostki centralnej, być może nawet ważniejszymi od bogactwa listy rozkazów i zasobów wewnętrznych. Bardzo popularny swego czasu mikroprocesor 6502 składał się głównie z „niczego“, a jego lista rozkazów zawierała bodajże niewiele ponad 50 instrukcji i to tych najprostszych. Pomimo tego konkurował z powodzeniem ze znacznie bardziej skomplikowanym Z80, przewyższając go właśnie asortymen-



Rys. 9. Adresowanie natychmiastowe

Rys. 10. Kodowanie rozkazów dwuargumentowych w mikrokontrolerze MSP430

tem dostępnych trybów adresowania.

Jako reprezentatywny przykład, na **rys. 10** przedstawiono sposób kodowania rozkazów dwuargumentowych w 16-bitowym mikrokontrolerze MSP430. Każdy rozkaz zawiera 4-bitowy kod operacji KO, co teoretycznie pozwala na zakodowanie 16 rozkazów. W rzeczywistości rozkazów dwuargumentowych jest tylko 12, a pozostałe kombinacje wykorzystywane są przez inne rozkazy (jednoargumentowe i skoki), w których odmienna jest interpretacja reszty słowa rozkazowego. Jest to bardzo często stosowana „sztuczka" konstrukcyjna. Jeden bit (B/W) zarezerwowano dla rozróżniania operacji ośmio- i szesnastobitowych - mikrokontroler ten pozwala na wykonywanie tych samych rozkazów nie tylko na dwubajtowych słowach, ale też na pojedynczych bajtach, z tym że oba argumenty (i wynik) muszą mieć tę samą długość, a rejestry wewnętrzne nie mogą być dzielone na dwie części. W przypadku ewentualnych konfliktów (np. dodanie 8-bitowego argumentu do zawartości jednego z rejestrów) przewidziane zostały odpowiednie, jednoznaczne reguły zachowań. Dwa 4-bitowe pola AS i AD, każde wskazujące jeden z 16 rejestrów uniwersalnych, stanowią części adresowe operandów. Interpretacja zawartości pierwszego z rejestrów zależy od związanego z nim dwubitowego modyfikatora adresu pierwszego argumentu MS, co może sugerować, że dostępne są dla niego 4 tryby adresowania (konkretnie: bezpośrednie rejestrowe, pośrednie, pośrednie z autoinkrementacją i indeksowe). W rzeczywistości trybów tych jest aż siedem (dochodzi jeszcze względne bieżące, bezpośrednie odniesione do dowolnej komórki pamięci oraz natychmiastowe), co wynika z oryginalnej koncepcji umieszczenia wśród równouprawnionych rejestrów uniwersalnych także licznika rozkazów. W ten sposób przykładowo adresowanie indeksowe „wchłania" niejako - odniesione do licznika rozkazów - adresowanie względne bieżące. Dla drugiego argumentu i równocześnie wyniku przewidziano już tylko jeden bit modyfikatora adresu MD, co powoduje, że może on (tzn. operand) być wskazywany tylko w dwóch trybach adresowania, tym niemniej i tak odpowiadających czterem w typowych mikroprocesorach. Jest to jedyne odstępstwo od ortogonalności, spowodowane po prostu ograniczoną długością słowa rozkazowego, którego wszystkich 16 bitów zostało już wykorzystanych. Sposób ułożenia tych elementów informacyjnych w rozkazie robi wrażenie nieco chaotycznego i przypadkowego, ale widocznie projektant miał tu jakieś swoje racje, a może po prostu fantazję.

Siedem rozkazów jednoargumentowych rozpoczyna się unikalną kombinacją 0001, po której występuje dalszy ciąg kodu konkretnej operacji, wyróżnik bajt/słowo, a następnie 4-bitowy adres rejestru i dwubitowy modyfikator adresu identyczne, jak w rozkazach dwuargumentowych.

Rozkazy skoków warunkowych rozpoczynają się kombinacją 001, uzupełnioną trzema bitami kodującymi konkretny warunek skoku i 9-bitowym adresem względnym ze znakiem.

Maciej Nowiński