OŚRODEK BADAWCZO-ROZWOJOWY INFORMATYKI

FIRMWARE

Europejski Program Badawczy Diebolda

44

Warszawa 1973

OŚRODEK BADAWCZO-ROZWOJOWY INFORMATYKI

FIRMWARE

Europejski Program Badawczy Diebolda

Wyłącznie do użytku na terenie PRL

44

Warszawa 1973

Tytuł oryginału: FIRMWARE
Document No E 73, May 1970

Tłumaczenie: S.Pawlak
Weryfikacja: A.B.Empacher
Redakcja: A.Idźkiewicz

Komitet Redakcyjny:

Mieczysław Gula, Franciszek Haratym, Andrzej Idźkiewicz, Janina Jerzykowska /sekretarz/, Jerzy Kisielnicki, Stanisław Nelken /zastępca przewodniczącego/,Krzysztof Skulski,Zdzisław Zapolski /przewodniczący/

Wydawca:
Działowy Ośrodek Informacji, Warszawa, ul.Marszałkowska 104/122

OBRI. Warszawa 1972 r. Nakład 850+161 egz.Objętość:ark.wvd. 2,5 ark.druk. 7. Format A4. Papier offsetowy, kl.III,80g 61x86/16.

Druk DOI/F. Zam. nr 82 /73 R-19 Cena zł 92,-

SPIS TREŚCI

STRESZCZENIE

Pomimo niedawnych sporów terminologicznych, pojęcie firmware [1/] jest tylko pozornie skomplikowane; przy rozpatrywaniu kompletnych systemów komputerowych - różnych klas wielkości - jak i licznych urządzeń peryferyjnych występują określone problemy połączenia sprzętu fizycznego /hardware/ z oprogramowaniem użytkowym /software/ w efektywnie funkcjonującą całość. Występuje przy tym następujące rozróżnienie: nazwa firmware odnosi się do informacji binarnej, utrwalonej jako mikroprogramy w pamięci stałej bloku sterowania, natomiast całokształt czynności związanych z określeniem tej informacji nazywa się mikroprogramowaniem.

Koncepcja wyposażenia bloku sterowania we własną pamięć sterującą nie jest bynajmniej nowa; to tylko względna powolność i wysoki koszt pamięci stałych ograniczał w komputerach wczesnych generacji rolę jej do minimum. W owych czasach konstruktorzy komputerów przy realizacji rozkazów maszynowych koncentrowali swą uwagę na projektowaniu układowym. Dopiero z chwilą pojawienia się w okresie trzeciej generacji szybszych i tańszych pamięci stałych, firmware stał się praktycznie realny i niektórzy producenci skorzystali z tej sposobności.

Postępowość firmware wyraża się zwiększeniem swobody projektowania logicznego. Daleko bowiem prościej jest wymienić mikroprogramy zawarte w pamięci stałej, chcąc zmienić charakterystykę logiczną sprzętu fizycznego, aniżeli przebudowywać obwody elektroniczne bloku sterującego. Jeszcze bardziej powinni być zainteresowani firmware'm użytkownicy, ponieważ drogą mikroprogramowania można poprawić wydajność zainstalowanego sprzętu oraz posiadanego oprogramowania użytkowego.

1/ Ze względu na brak jednoznacznego i zwięzłego polskiego odpowiednika dla angielskiego neologizmu firmware -którego zakres znaczeniowy omówiono na stronie 13 - termin ten pozostawiono w niniejszym przekładzie bez tłumaczenia /przyp.red./

Różnego rodzaju urządzenia peryferyjne, a zwłaszcza stacje zbierania i emisji danych, wykazują w swej konstrukcji wzrastający udział firmware'u. W dalszym rozwoju firmware obejmuje dyski, taśmy i bębny pamięciowe oraz synchronizatory urządzeń peryferyjnych. Już obecnie firma Standard Computer w swych komputerach rodziny IC-7000 wprowadziła mikroprogramowany procesor kanałów wejścia/wyjścia, który umożliwia współpracę dowolnych jednostek peryferyjnych - poczynając od kanałów danych i ich synchronizatorów, a skończywszy na super szybkich czytnikach kart oraz koncentratorach teledacyjnych. Technika mikroprogramowania bloku sterowania pozwala użytkownikowi swobodnie dobierać urządzenia peryferyjne do wymogów wejściowo/wyjściowych rozpatrywanego systemu zastosowań - o ile tylko obrany typ komputera spełnia wymogi stawiane jednostce centralnej.

Firmware pozwala rozszerzyć "inteligencję" wielu urządzeń peryferyjnych i małych komputerów, jakie anonsowano w ostatnich latach. Obecnie już kilku producentów oferuje małe komputery typu mikroprogramowanego, przystosowane do specyficznych prac w systemach transmisji danych. Można praktycznie przyjąć za regułę, że co bardziej typowe fragmenty oprogramowania użytkowego, prędzej czy później, zostaną zrealizowane na drodze firmware'owej. Nie oznacza to, że koszt pamięci sterującej jest zaniedbywany, gdyż sięga jeszcze około połowy kosztu mikroprogramowanego procesora; jednakże co najmniej jeden producent małych komputerów spodziewa się, że w niedługim czasie technika wielkoskalowych obwodów scalonych /LSI/ umożliwi radykalną redukcję kosztu firmware'u.

Dla ogółu użytkowników firmware jest pojęciem "mglistym" i "niesprawdzonym", zwłaszcza w odniesieniu do instalacji komputerowych w klasach wielkości średniej i dużej. Jednak tutaj właśnie należy oczekiwać daleko idących ulepszeń i związanej z tym obniżki kosztów przetwarzania. W przypadku pewnego użytkownika wprowadzenie właściwie przygotowanych mikroprogramów podniosło ogólną sprawność komputera IBM-360/50 o 25-30%.

Niestety, jest to na razie przykład dosyć odosobniony. Obecnie użytkownik musi z reguły borykać się sam ze wszystkimi kło-

potami mikroprogramowania i po prostu z konieczności godzi się na brak wymienności mikroprogramowania między różnymi syste - mami komputerowymi - co zwiększa koszty prac techniczno-eksploatacyjnych. Trudności tego rodzaju, podobnie jak brak języków wyższego rzędu do mikroprogramowania oraz brak procesorów do tłumaczenia jednych takich języków na drugie - wynika nie tyle ze względów technicznych, co ekonomicznych. I dopóki użytkownicy nie zaczną powszechniej żądać mikroprogramowa - nia swych problemów, dopóty przemysł będzie unikał opraco - wania takich języków, wymagających w fazie przygotowawczej znacznych nakładów pieniężnych i czasowych. Dlatego też aktualnie pionierami firmware'u są tylko nieliczni użytkowni - cy, którzy dysponują bogatym powtarzalnym oprogramowaniem specjalistycznym i mają przez to pewność, że wysiłek włożony w oprogramowanie firmware'owe nie pójdzie na marne - gdyż nie korzystają z komputerów uniwersalnych, które mogą w każdym tygodniu służyć coraz to innym zastosowaniom.

Generalnie rzecz biorąc, przyszłość firmware'u zapowiada się dosyć obiecująco. Przede wszystkim należy oczekiwać bliskiej współpracy pomiędzy użytkownikami i producentami małych komputerów, której powodem byłyby standardowe mikroprogramy o dostatecznie szerokim zastosowaniu; wzorem może tu być firma INTERDATA ze swym mikroprogramem BIM dla skoków multiplekso - wych. Użytkownicy i producenci większych komputerów będą raczej zainteresowani wymiennością firmware'u poprzez zastoso - wanie pamięci quasi-stałej, umożliwiającej zapisanie nowych mikroprogramów, gdyby zachodziła potrzeba takich zmian; w ten sposób objętość pamięci quasi-stałej można by zredukować do minimum, a wszystkie "zapasowe" mikroprogramy dałyby się prze - chowywać na dyskach lub w masowych pamięciach ferrytowych. Rozwiązaniem idealnym byłaby możliwość natychmiastowej adaptacji systemu komputerowego do efektywnej pracy w wyższych językach programowania, jak COBOL, FORTRAN, PL/1 i podobne. W miarę dalszego rozwoju i gromadzenia doświadczenia mikrooprogramowaniowego, producenci będą mogli oferować swe maszyny ze swobodnie dobieranym firmware'm - na wzór "superinstrukcji" ostatnio wy-

pracowanych w firmie Allen-Babcock, które mają szansę zaakcep - towania przez cały przemysł komputerowy.

Zwartość wielkoskalowych obwodów scalonych może doprowa - dzić w dalszej perspektywie do jeszcze szerszego rozwoju firm- ware'u. Ale również może doprowadzić do zaniku firmware'u.

W miarę bowiem tanienia produkcji układów wielkoskalowych mo- że dojść do sytuacji, w której przejście od specyfikacji logicz- nej do realizacji układowej może okazać się porównywalne w kosz- tach z przejściem od tejże specyfikacji do realizacji firmware'- owej. I wówczas zaczną znów dominować realizacje układowe, co znaczyłoby zamknięcie się linii rozwojowej w metodyce projekto- wej.

I. WPROWADZENIE

A. STWIERDZENIA PODSTAWOWE O ROLI FIRMWARE'U

Z punktu widzenia użytkownika, problemy firmware'owe należy rozpatrywać w trzech kategoriach sprzętu komputerowego (urządzenia peryferyjne, małe procesory typu specjalizowanego oraz większe procesory). Podział ten, niewątpliwie, odpowiada różnym stopniom złożoności systemowej; co do firmware'u, to obecnie wpływa on daleko bardziej na prostsze urządzenia peryferyjne i małe komputery aniżeli na wielkie systemy komputerowe. Ponieważ jednak zalety i ograniczenia firmware'u szczególnie uwypuklają się na przykładzie większych komputerów, od nich więc zaczniemy omówienie rozeznania wstępnego.

1. Odnośnie <u>większych komputerów</u> można stwierdzić, że:

a/ Firmware jest jeszcze zbyt nowy i za mało sprawdzony, aby można go było zalecać wszystkim użytkownikom.

b/ Możliwe jest dalsze zwiększanie wydajności mikroprogramowanych komputerów, jak to widać na przykładzie firmy Allen - Babcock. Jednakże użytkownik musi przygotować się na wysokie koszty - związane z opracowaniem specjalnych rozkazów, niemożliwością bezpośredniej wymiany mikroprogramowania wyspecjalizowanego między różnymi systemami komputerowymi oraz zwiększonym zakresem prac techniczno-eksploatacyjnych.

c/ Zastosowanie firmware'u w komputerze może przedłużyć okres jego efektywnej eksploatacji, a to poprzez wprowadzanie ulepszeń wynikających z dotychczasowego doświadczenia użytkowego. Jest więc rzeczą możliwą, że sprzęt komputerowy bazujący na tzw. architekturze trzeciej generacji - okaże się praktycznie efektywnym nawet w dłuższym okresie czasu.

d/ Wysoko wydajny i efektywny firmware użytkowy jest tak długo nierealizow dopóki nie zostaną wyjaśnione w sposób dokładny i pełny wszystkie problemy oprogramowaniowe dotyczące

rozpatrywanego zastosowania. Mikroprogramy specjalizowane można próbować tworzyć dopiero po wydzieleniu w oprogramowaniu użytkowym takich podprogramów, których realizacja na drodze firmware'owej wydaje się możliwa. Jednakże samo mikrozaprogramowanie tych podprogramów, których wybór wydawał się logiczny, wcale nie musi dać w wyniku zwiększenia wydajności systemowej. Sam użytkownik musi więc zdecydować, czy zakładane zwiększenie wydajności i efektywności systemu pokryje mu koszty czterech do sześciu tygodni, jakie są niezbędne do opracowania pojedynczego rozkazu firmware'owego.

e/ Obecnie większość producentów nie stwarza swym klientom warunków do projektowania własnego firmware'u. Ale nawet gdyby pojawiły się takie możliwości, to - poza co bardziej ambitnymi użytkownikami, ośrodkami usług telekomputerowych oraz placówkami uniwersyteckimi zainteresowanymi badaniami rozwojowymi - właściwie mało kto byłby w stanie z nich korzystać. Dopóki nie pojawią się języki wyższego rzędu do mikroprogramowania, użytkownik pogrążony w opracowywaniu własnych systemów będzie przeciwny ogólnemu trendowi - jakim jest ułatwianie łączności z komputerem.

Obecnie, gdy "mikroprogramista" dysponuje jedynie ograniczonymi narzędziami pracy, czynność mikroprogramowania jest nieciekawa i czasochłonna. Całkowity brak kompilatorów i możność stosowania jedynie kilku języków asemblacyjnych - oznacza, że mikroprogram opracowany na jakiś komputer wcale nie musi dać się przenieść na inny komputer, choćby nawet pochodzący od tego samego producenta. Dlatego też użytkownik zamierzający zmienić swój komputer musi przewidzieć całkowitą zmianę stosowanych dotychczas specjalnych rozkazów i programów.

Brak języków wyższego rzędu oraz brak procesorów do tłumaczenia jednych takich języków na drugie, wynika nie tyle ze względów technicznych, co przede wszystkim ekonomicznych. I dopóki użytkownicy nie zaczną powszechniej żądać mikroprogramowania swoich problemów, dopóty przemysł będzie unikał opracowania takich języków, wymagających w fazie przygotowawczej znacznych nakładów pieniężnych i czasowych.

f/ W okresie najbliższych pięciu lat wybór list rozkazowych, oferowanych użytkownikom poszczególnych komputerów, ulegnie zwiększeniu i będzie bardziej urozmaicony oraz kompleksowy. Bardziej złożone rozkazy, zwane superrozkazami,będą w większości postaci standardowej, a jako definiowalne w językach wyższych rzędów, staną się ogólnie dostępnymi.Ta dodatkowa elastyczność, oferowana użytkownikowi, będzie uzależniona od istnienia "zapisywalnych" pamięci w bloku sterowania. /Podobne superrozkazy będzie można realizować w komputerach konwencjonalnych, które nie posiadają firmware'u a więc w komputerach, w których nie występuje pamięć w bloku sterowania. Różnica polega na większym stopniu elastyczności związanej z mikroprogramowaniem/.
Z chwilą gdy użytkownicy zechcą posiadać możliwość mikroprogramowywania swoich systemów, będą zmuszeni rozwiązać we własnym zakresie problemy konserwacji technicznej, którymi zazwyczaj zajmują się bezpośrednio producenci. Najistotniejszym narzędziem będzie tutaj prawdopodobnie system diagnostyczny poziomu podstawowego, umożliwiający porównanie uruchamianego systemu z projektami oryginalnymi.

2. Małe komputery

Firmware może okazać się poręczny w odniesieniu do małych komputerów, przeznaczonych do specjalizowanych zastosowań.Opracowanie zastosowania będzie w dalszym ciągu względnie drogie,jeżeli uwzględni się wszystkie koszty - projektowania wstępnego, zdefiniowania i opracowania specjalnych rozkazów, uwzględnienia ich w systemie eksploatacyjnym, modyfikacji lub nowego okablowania pamięci stałej.
Jednakże w określonych sytuacjach, kiedy pożądana funkcja jest nie do zrealizowania ani na drodze programowej, jako zbyt czasochłonna, ani na drodze układowej, jako zbyt zawiła, rozwiązanie firmware'owe może okazać się jedynym ratunkiem.

Podobnie jak to miało miejsce w dużych komputerach również i tu będą rozwijane różnego rodzaju superrozkazy, oferowane do wy-

boru użytkownikom. Niektóre z nich o wysoce specjalistycznym zastosowaniu, będą wykorzystywać nawet całkowitą pojemność mniejszych komputerów. Można więc powiedzieć, że firmware będzie niezastąpionym narzędziem wprowadzania dodatkowych funkcji systemowych. Jednakże z chwilą zrealizowania najważniejszych problemów dotyczących superrozkazów, będzie je także można wbudowywać układowo w kolejnych modelach produkowanego komputera, mając na celu uzyskanie jak największej wydajności.

3. Urządzenia peryferyjne

Podstawowe zastosowanie znajdzie prawdopodobnie firmware w urządzeniach peryferyjnych - szczególnie tam, gdzie wydajność sprzętu ogranicza sam operator. W tych ostatnich przypadkach pamięci stałe, magazynujące mikroprogramy, mogą być odpowiednio wolniejsze, a co za tym idzie, nie będą tak kosztowne.

Opracowywanie mikroprogramów dla takich urządzeń jest znacznie łatwiejsze niż dla dużych systemów komputerowych, jednakże producenci będą zapewne rownolegle rozwijać specjalizowany firmware, oferowany użytkownikom jako możliwość dodatkowa.

Będą również rozwijane synchronizatory urządzeń peryferyjnych z nieco bardziej złożonymi mikroprogramami, zróżnicowane w swych możliwościach, stosownie do sprzętu z jakim będą współpracować. Ideą przewodnią jest oferowanie użytkownikowi szerokiej swobody przy wyborze sprzętu odpowiadającego wymogom wejściowo/wyjściowym rozpatrywanego zastosowania. Niektóre z takich jednostek są dostępne już w chwili obecnej. Firmware na tym poziomie pozwoli użytkownikowi modyfikować własności sprzętu stosowanie do swych potrzeb problemowych. Jednakże obserwowany trend wprowadzania funkcji przetwarzaniowych do wszystkich jednostek systemu komputerowego spowoduje, że co bardziej "inteligentne" urządzenia peryferyjne przejmą część dotychczasowego obciążenia jednostek centralnych.

B. DEFINICJE ROBOCZE FIRMWARE'U

1. Genealogia pojęcia firmware

Słowo firmware anonsował Ascher Opler w zeszycie styczniowym czasopisma "DATAMATION" z roku 1967:

"Używam tego terminu na oznaczenie mikroprogramów utrwalonych wewnątrz komputera w pamięci bloku sterowania, która wyznacza strukturę logiczną tego komputera pod kątem określonego zastosowania, np. emulowania innego komputera".

Słowo to było od samego początku niewłaściwie interpretowane. W tym samym numerze "DATAMATION" w artykule omawiającym problemy wejściowo/wyjściowe inny autor stwierdził:

"Być może niektóre z wymienionych problemów zostaną rozwiązane przez to, co Ascher Opler w swym artykule nazwał firmware. Należy się więc zastanowić w jak szerokim zakresie będzie można wkomponować software do specjalizowanego hardware'u".

Rzecz w tym, że Opler w ogóle nie poruszał problemu "wkomponowania software'u do hardware'u". Wręcz przeciwnie, stwierdził, że "wszystkie problemy dotyczące dwustyku hardware/software nie znikną, lecz zostaną zastąpione bardziej złożonym problemem trójstyku hardware/firmware/software".

"Firmware" w niniejszym opracowaniu odnosi się do mikroprogramów utrwalonych wewnątrz komputera w pamięci bloku sterowania. Natomiast całokształt czynności związanych z określeniem jakiego rodzaju informacja ma być zawarta w tej pamięci sterującej objęto nazwą "mikroprogramowanie". Informacja ta występuje w formie binarnego mikroprogramu. Opler mówił wprawdzie i o dodatkowym etapie, związanym z tworzeniem specjalizowanych bloków logicznych, ale już sama obecność informacji logicznej wewnątrz bloku sterowania sprawę tę w zasadzie załatwia, przynajmniej od strony pojęciowej.

Kiedy mikroprogramy pokrywają się z pojęciem firmware'u ? Niektóry uważają za firmware mikrooprogramowanie zaprojektowa-

nego systemu zastosowań - to znaczy traktują go jako czynnik fakultatywny przy realizacji systemu. Inni utożsamiają firmware z etapem projektowania i wytwarzania komputera, a nie tylko z narzędziem służącym użytkownikowi. W niniejszym opracowaniu nie rozróżniono tych problemów, ale skoncentrowano się raczej na zagadnieniach i możliwościach wykorzystania firmware'u jako narzędzia użytkownika.

Wracając do Cplera, to przedstawia on trójstyk hardware/firmware/software w aspekcie możliwości, jakie stoją do dyspozycji konstruktora komputerów. Firmware jest tylko jedną z tych możliwości i reprezentuje zaledwie metodę lub technikę, za pomocą której może być zrealizowana określona funkcja. Na obecnym etapie podejścia projektowego jako pierwsze pojawia się pytanie: jaka ma być zasadnicza funkcja rozpatrywanego systemu? Sposób realizacji tej funkcji jest pozostawiony konstruktorowi - może on oprzeć się na tradycyjnym kompromisie między hardware'm i software'm lub pominąć software i ograniczyć się całkowicie do rozpatrywania hardware'u i firmware'u dla określonego zastosowania.

Koncepcje firmware'owe można rozpatrywać z trzech głównych punktów widzenia. Mogą one stanowić rozwiązanie kompromisowe, umożliwiające zarówno konstruktorowi jak i użytkownikowi, zastosowanie różnych poziomów przystosowania. Mogą być rozpatrywane jako zwykłe rozszerzenie dotychczasowej wiedzy o pamięciach hierarchicznych. I wreszcie mogą być rozpatrywane jako następny krok w ewolucji techniki projektowania komputerów.

2. Współpowiązanie hardware'u, software'u i firmware'u

Aby ułatwić zrozumienie zależności i powiązań pomiędzy hardware'm, firmware'm i software'm - rozpatrzmy następujący schemat:

HARDWARE	FIRMWARE	SOFTWARE
Elastyczność ——→ +	- ←—— Elastyczność	
Szybkość ——→ -	+ ←—— Szybkość	
Specjalizacja ——→ -	+ ←—— Specjalizacja	
←—— Oprogramowanie systemowe ——→		
←—— PRODUCENT ——→	←—— UŻYTKOWNIK ——→	

Lewa strona schematu dotyczy głównych kompromisów, jakie należy rozstrzygnąć przy produkcji i badaniach rozwojowych; strona prawa wskazuje na te kompromisy, które może rozstrzygać użytkownik. Przechodząc z lewej strony do środka schematu obserwujemy wzrost elastyczności, zmniejszenie szybkości oraz zmniejszenie tego, co może być nazwane przeznaczeniem specjalistycznym. Przeciwny kierunek zmian zaobserwujemy przechodząc od strony prawej do środka.

Z uwagi na zwiększającą się rolę podejścia integracyjnego w projektowaniu systemów oprogramowanie systemowe rozciąga się aż po obręb hardware.

Wydaje się, że coraz więcej producentów sprzętu komputerowego pozwala grupom oprogramowującym systemy wpływać na kierunek prac konstrukcyjnych, a zatem na określenie w jaki sposób mają być realizowane poszczególne operacje maszynowe. Konstruktor ma do dyspozycji trzy środki realizacyjne - hardware, software i firmware - z wynikającymi stąd wszystkimi kompromisami. Funkcje, co do których jest pewien, że są niezmienne, będzie on realizować na drodze hardware'owej; natomiast te funkcje, których niezmienności jest on już mniej pewny, będzie realizować na drodze firmware'owej. W ostateczności rozwiązanie firmware'owe będzie można później zastąpić hardware'owym, aby uzyskać maksymalną wydajność, lecz taka zamiana może mieć miejsce tylko w takim przypadku, gdy konstruktor nabrał już absolut-

nej pewności, że rozpatrywana funkcja nie zostanie już zmieniona. Analogiczne rozwiązania stosują się do zastępowania określonych programów software'owych przez firmware'owe.

Hardware, software i firmware razem współdziałają przy realizowaniu założonej funkcji w systemie. Odbywa się to według schematu: rozkaz użytkowy jest rozczytany z pamięci głównej komputera do pamięci w bloku sterowania; w następstwie tego zostaje wygenerowany ciąg specjalnych mikroinstrukcji, rządzących wykonaniem elementarnych operacji ładowania i rozładowania rejestrów oraz podawania impulsów sterujących do jednostki centralnej, pamięci i kanałów wejścia/wyjścia - w jednym cyklu zegarowym.
Po zakończeniu wykonywania tych mikroinstrukcji cykl jest powtarzany w odniesieniu do następnego rozkazu użytkowego.

II. ROLA FIRMWARE'U Z PUNKTU WIDZENIA PRODUCENTA

A. OGÓLNA ORGANIZACJA KOMPUTERÓW

W celu zrozumienia istoty mikrooprogramowania należy rozpatrywać konstrukcję komputera w kilku aspektach. Komputer można więc traktować przede wszystkim jako układ rejestrów, obwodów logicznych i generatorów sygnałów sterujących oraz połączeń wewnętrznych między nimi.
Rejestry stanowią pamięć statyczną; można w nich wydzielić takie, które są adresowane oraz takie, które są połączone z kanałami wejścia/wyjścia i pamięcią główną. Dane przesyłane są pomiędzy nimi poprzez połączenia wewnętrzne, które równocześnie przenoszą sygnały sterujące kasowania i ładowania rejestrów, generowania stałych itp. Obwody logiczne wykonują dodawania, przesunięcia, operacje bulowskie, ustalają pierwszeństwa, generują i sprawdzają bity kontrolne oraz przy użyciu szablonów logicznych mogą zakrywać część danego słowa lub też łączyć różne części słów ze sobą. Generatory sygnałów sterujących składają się z

zegarów, sekwenterów i dekoderów.

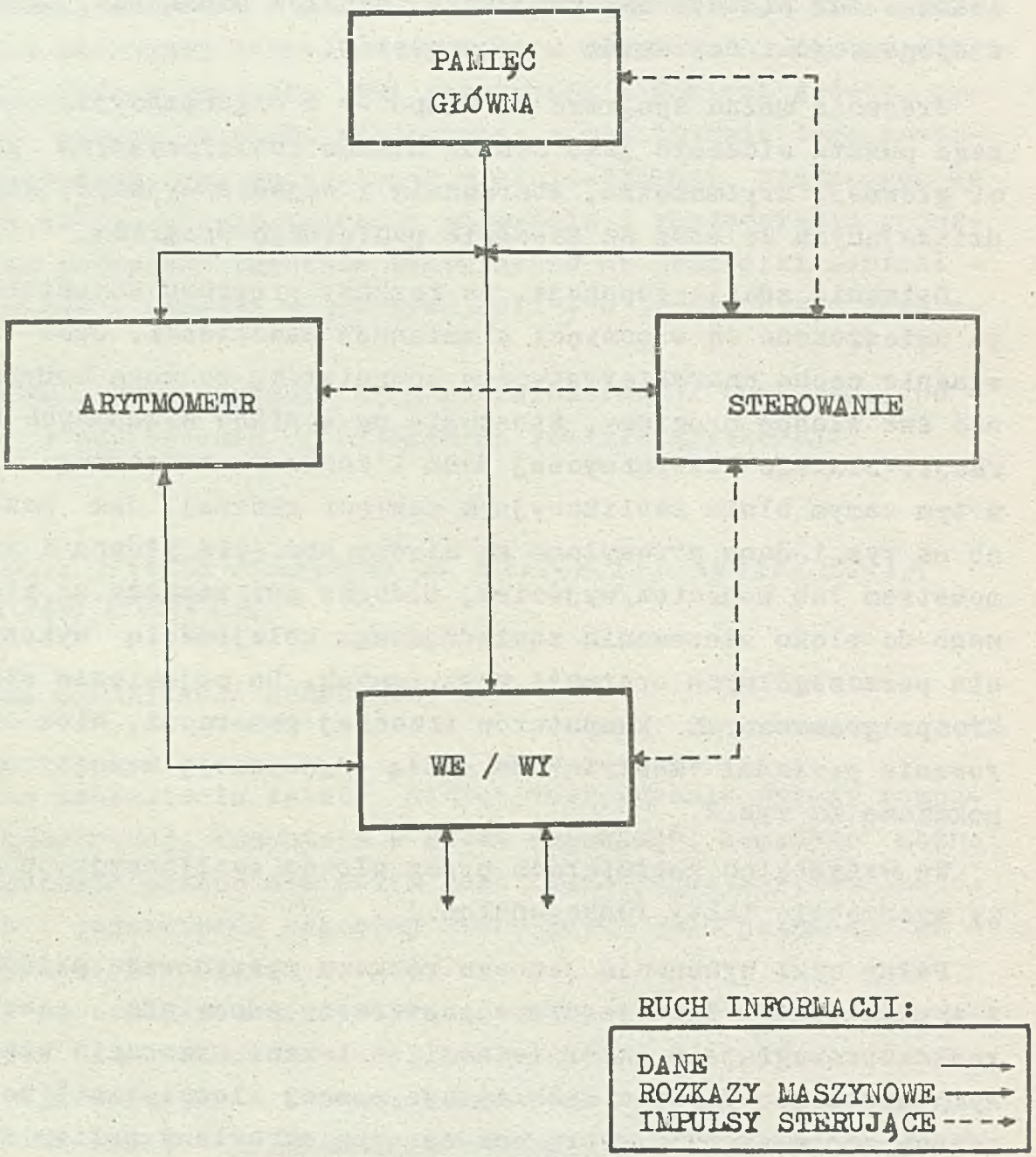
Komputer można również rozpatrywać jako skomplikowaną sieć logiczną - składającą się z bramek, które zawiadują przepuszczaniem lub blokadą napływających sygnałów binarnych, zgodnie z dyspozycjami dekoderów i sekwenterów.

Wreszcie można spojrzeć na komputer z organizacyjno-ogólnego punktu widzenia jako zestaw bloków realizacyjnych pamięci głównej, arytmometru, sterowania i wejścia/wyjścia, współdziałających ze sobą na zasadzie pamiętnego programu.

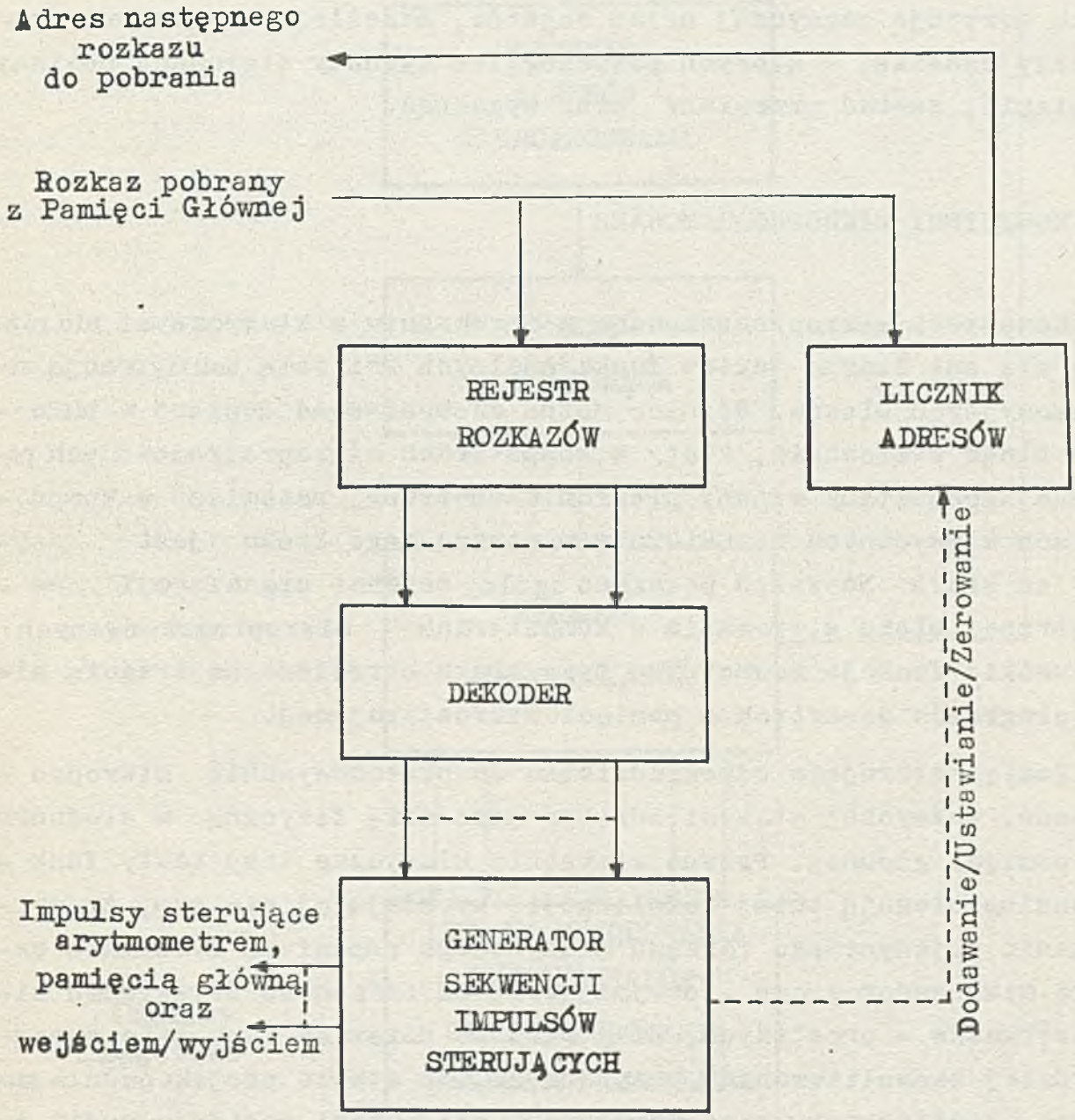
Ostatnie zdanie suponuje, że rozkazy programu komputerowego umieszczane są w pamięci o zmiennej zawartości. Jest to właśnie cechą charakterystyczną komputerów, że mogą modyfikować swe własne programy, stosownie do wyników wykonanych operacji. Dlatego też zazwyczaj dane i rozkazy przechowuje się w tym samym bloku realizacyjnym pamięci głównej. Jak pokazano na rys.1 dane przesyłane są między pamięcią główną i arytmometrem lub wejściem/wyjściem, podczas gdy rozkazy są kierowane do bloku sterowania zawiadującego kolejnością wykonywania poszczególnych operacji maszynowych. Do pojawienia się mikrooprogramowanych komputerów trzeciej generacji, blok sterowania posiadał "zaszytą" na stałą organizację wewnętrzną,jak pokazano na rys.2.

We wszystkich komputerach prócz bloków realizacyjnych należy wyodrębnić takty funkcjonalne.

Pełny cykl wykonania jednego rozkazu maszynowego składa się z trzech taktów funkcjonalnych: pierwszy odpowiada pobraniu rozkazu, drugi jego interpretacji, a trzeci wykonania właściwych operacji. Każdy z nich wymaga pewnej ilości chwil zegarowych,zanim sygnały sterujące osiągną określony poziom natężenia. Poziom ten musi być utrzymany dopóty, dopóki odnośny wynik częściowy nie pojawi się na wyjściu rozpatrywanego układu albo zostanie przesłany do odpowiednich rejestrów.
Kiedy wyniki częściowe zostały już wykorzystane, odnośne sygnały sterujące ulegają "wygaszeniu" - co wymaga znowu pewnego czasu, po którym dopiero zostaje wygenerowana nowa grupa sy -



Rys.1. OGÓLNA ORGANIZACJA KOMPUTERÓW



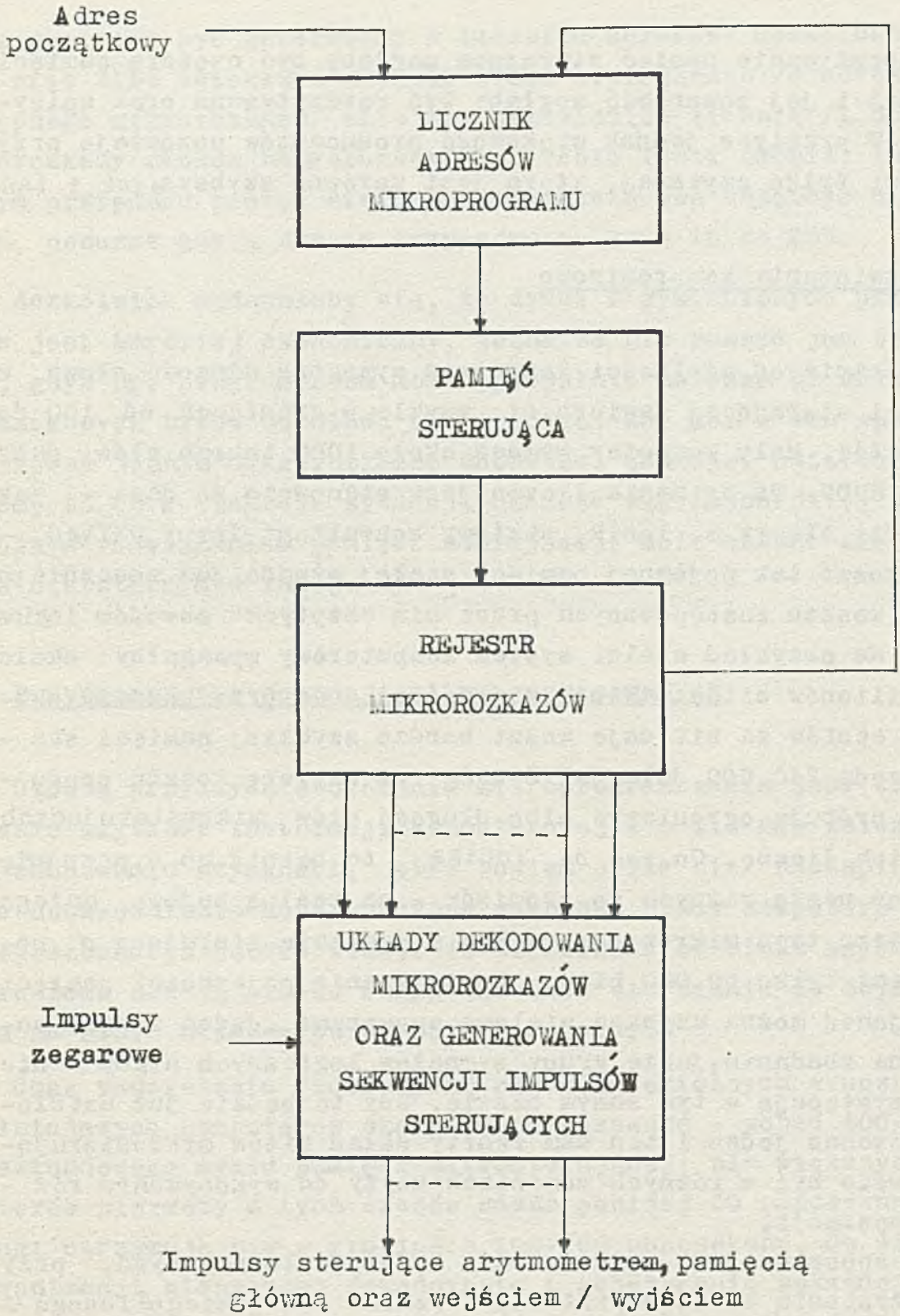
Rys.2. KLASYCZNA ORGANIZACJA BLOKU STEROWANIA

gnałów. Całością procesu generowania sekwencji sygnałów sterujących dyryguje zazwyczaj układ zegarów, określając precyzyjnie przedziały czasowe, w których poszczególne sygnały sterujące powinny wystąpić, zostać przesłane oraz wygasnąć.

B. KOMPUTERY MIKROPROGRAMOWANE

Komputery mikroprogramowane w porównaniu z klasycznymi nie różnią się ani liczbą taktów funkcjonalnych ani samą konfiguracją realizacyjnych bloków. Różnicę można zaobserwować dopiero w budowie bloku sterowania, który w komputerach mikroprogramowanych posiada zapamiętany własny program wewnętrzny, natomiast w komputerach klasycznych struktura wewnętrzna tego bloku jest „szyta" na stałe. Na rys.3 pokazano ogólny schemat organizacji wewnętrznej bloku sterowania w komputerach mikroprogramowanych: wszystkie funkcje zewnętrzne tego bloku określone są treścią mikroprogramów zawartych w pamięci mikrosterującej.

Pamięć sterująca odpowiedzialna za przechowywanie mikroprogramów, zazwyczaj stanowi odrębną jednostkę fizyczną w stosunku do pamięci głównej. Przede wszystkim klasyczne trzy takty funkcjonalne ulegają tutaj komplikacji, wyrażającej się tym, że wykonaniu pojedynczego rozkazu maszynowego odpowiada wykonanie całego mikropodprogramu - obejmującego od trzech do dziesięciu mikrorozkazów w prostszych, aż gdzieś do dziesięć razy tyle w najbardziej skomplikowanych przypadkach. Na etapie projektowania maszyny, konstruktor, programista lub projektant systemów musi określić układ bitów wewnątrz każdego słowa pamięci sterującej oraz kolejność, z jaką poszczególne słowa mikrosterujące - dyrygujące przesyłaniem danych oraz ustawieniem czy też zerowaniem rejestrów - mają być rozczytywane.
Pojemność pamięci sterującej zależy od typu maszyny. I tak na przykład w komputerach rodziny Honeywell-4200 pamięć sterująca obejmuje 2000 słów - 120-bitowych, a w komputerach IBM-360/50 obejmuje 2816 słów 90-bitowych.



Rys.3. ORGANIZACJA MIKROPROGRAMOWANEGO BLOKU STEROWANIA

Teoretycznie pamięć sterująca mogłaby być częścią pamięci głównej i jej zawartość mogłaby być rozczytywana oraz wpisywana. W praktyce jednak większość producentów pozostaje przy pamięci tylko czytanej, która jest zarówno szybsza, jak i tańsza.

1. Rozwiązania kompromisowe

Zależnie od wielkości komputera wymagana długość słowa w pamięci sterującej zawiera się zwykle w granicach od 100 do 300 bitów. Mały komputer wymaga około 1000 takich słów; duży około 8000. Ta ostatnia liczba jest stanowczo za duża - jak twierdzi Albert S. Tonik, etatowy konsultant firmy UNIVAC, - gdyż koszt tak pojemnej pamięci stałej wypada już znacznie powyżej kosztu zastępowanych przez nią "szytych" obwodów logicznych. Na przykład wielki system komputerowy wymagałby około 2,4 milionów bitów pamięci sterującej - co przy koszcie około 10 centów za bit daje koszt bardzo szybkiej pamięci stałej rzędu 240.000 dolarów. Z uwagi na barierę kosztu producenci próbują ograniczyć albo długość słów mikrosterujących albo ich liczbę. Co zaś do Tonika , to ocenił on - przy wzięciu pod uwagę różnych kompromisów - za realną budowę małego komputera typu mikroprogramowego z pamięcią sterującą o pojemności tylko 50.000 bitów. Zmniejszenie pojemności pamięci sterującej można uzyskać wieloma sposobami. Jeden z nich polega na zbadaniu, jakie grupy sygnałów logicznych nigdy nie współwystępują w tym samym czasie. Gdy to będzie już ustalone, wówczas jeden i ten sam zwarty układ bitów mikrosterujących może być w różnych momentach użyty do wykonywania różnych operacji.
Innym sposobem jest zastosowanie kodów wieloznacznych, przy wykorzystaniu do interpretacji licznika lub jakiegoś innego układu logicznego biorącego udział w aktualnie wykonywanej czynności.

Drugi kompromis dotyczy pytania: czy adres każdej następnej mikroinstrukcji ma być zawarty w słowie mikroprogramu, czy

też powinien być generowany w liczniku adresów? Konstruktor może więc albo zwiększyć długość słowa sterującego /o adres następnego mikrorozkazu/ albo też przewidzieć licznik /i dodać mikrorozkazy skoków bezwarunkowych/. Tenże Tonik ocenił, iż w pierwszym przypadku pamięć sterująca powiększa swą objętość o 15 do 35%, podczas gdy w drugim przypadku tylko o 10 do 25%.

Aczkolwiek wydawałoby się, że drugi z wymienionych przypadków jest bardziej ekonomiczny, jednakże nie zawsze jest to prawdą, gdyż np. programistom może specjalnie zależeć na dołączeniu dodatkowych bitów do słowa sterującego, aby móc w ten sposób realizować jednym mikrorozkazem dodatkowe funkcje. Ostatecznie bowiem, to co w jakiejś sytuacji okazuje się najbardziej ekonomicznym rozwiązaniem pamięci sterującej, może okazać się zupełnie nieistotnym w innej.

2. Przyspieszanie operacji mikroprogramowanych

Główną przyczyną stosowania mikroprogramowania jest uelastycznienie użytkowe instalacji komputerowej - o ile nie zależy nam na zachowaniu szybkości. Otóż bowiem o ile nie nastąpią dalsze udoskonalenia konstrukcyjne szybkość pracy komputera mikroprogramowanego będzie właściwie niezależna od bloku arytmometru, ogranicza zaś ją przede wszystkim blok sterowania ze swym zaszytym na stałe układem połączeń logicznych.

Czas zadziałania "zaszytych" obwodów logicznych wynosi dla wolniejszych komputerów około 100 nanosekund - wobec 400-500 nanosekundowego cyklu pamięci mikrosterującej; dla większych komputerów pierwszy z tych czasów spada poniżej 50 nanosekund, ale drugi utrzymuje się w granicach 100-200 nanosekund. Co więcej, w organizacji klasycznej dekodowanie i generowanie sekwencji impulsów sterujących trwa także krócej aniżeli w organizacji mikroprogramowanej.

Mikroprogramowany komputer można przyspieszyć na kilka sposobów. Jeden z tych sposobów polega na powrocie do realizacji u-

kładowej pewnych częściej wykonywanych czynności, takich jak wywoływanie słowa z pamięci głównej lub generowanie stałych. Przy tym szybkość wykonania operacji mikroprogramowej można dodatkowo zwiększyć drogą tworzenia tzw. "zakładek" czasowych - tj. poprzez "nakładanie się"[2/] - cyklu pamięci mikrosterującej na cykl pamięci głównej. Inny sposób zaproponował pewien producent, który nie mogąc znaleźć dostatecznie szybkiego pojedynczego bloku pamięci sterującej zastosował dwa takie bloki rozczytywane na przemian-co daje taki sam efekt, jakby cykl takiej pamięci stał się dwa razy krótszy. Następnym sposobem przyspieszenia pracy mikroprogramowanego komputera jest wydłużenie słowa sterującego, przez co ulega zwiększeniu liczba operacji wykonywanych jednocześnie-chociaż zastosowanie licznika pozwala uzyskać niemal idealnie zwartą pamięć sterującą. Krótko więc mówiąc problemy zwiększania szybkości i zmniejszania pojemności pamięci sterującej zmuszają do rozwiązań kompromisowych. O ile nie przeważa ani kwestia opłacalności ani wydajności, można przyjąć, że stosowanie pamięci sterującej w komputerach klasy średniej lub dużej jest wygodne. Innymi słowy, koszt realizacji układowej bloku sterowania w komputerach byłby wówczas porównywalny z realizacją mikroprogramową.

Powody zdecydowania się na któryś z wymienionych sposobów mogą być bardzo różne. Istnieją więc opinie, że koszt mikroprogramowanego komputera jest mniejszy. Ale dla bardzo szybkich i dużych komputerów pierwszorzędną kwestią jest właśnie szybkość i dlatego w nich zastosowanie mikroprogramowania jest prawie z góry wykluczone.

Można dowodzić, że z chwilą rozpowszechnienia szybkich pamięci sterujących, mikroprogramowanie można by stosować i w wielkich komputerach. Ale operacje mikroprogramowane są raczej szeregowe niż równoległe i przez to wolniejsze. Ostatecznie trzeba pójść na kompromis i pewne funkcje realizować na drodze układowej, a inne, których szybkość nie jest tak krytyczna, realizować poprzez mikroprogramy.

2/ W oryginale "overlapping".

C. KOMPILATORY I SYSTEMY OPERACYJNE

Możliwość zastosowania firmware'u w kompilatorach i systemach operacyjnych budzi zrozumiałe zainteresowanie producentów sprzętu komputerowego. Wyraźną zaletą komputerów mikroprogramowanych jest fakt, iż faza projektowania daje się rozciągnąć właściwie na cały okres ich użytkowania, dzięki czemu wprowadzanie ulepszeń wynikających z doświadczenia użytkowego staje się względnie łatwe. Stanowi to zachętę do podnoszenia wydajności procesorów języków i systemów operacyjnych. Ścisłe określenie ulepszeń, jakie będą możliwe w tym zakresie jest dosyć trudne, ponieważ projektowanie firmware'u nigdy właściwie się nie kończy. Ale istnieją pewne komputery, na przykład HONEYWELL-8200, w których "superinstrukcja" spełniają różnorodne funkcje.

Komputer ten normalnie pracuje w reżymie emulującym kod rozkazowy maszyny HONEYWELL-800. Jeden z programów pamięci stałej pobiera oryginalne rozkazy wejścia/wyjścia - wykonywalne bezpośrednio tylko w odpowiedniej instalacji komputerowej typu HONEYWELL-800 - przegląda zaprogramowane tabliczki interpretacyjne i dokonuje stosownej konwersji numerów urządzeń, numerów kanałów oraz formatów rozkazowych do postaci akceptowalnej przez bardziej rozbudowaną instalację typu HONEYWELL-8200. Cały taki program zajmuje ponad 100 słów /mikrorozkazów/ pamięci stałej.

Dzięki superrozkazom w systemie HONEYWELL-8200 możliwa jest np. rozmowa dwu procesorów o różnych organizacjach - słownej i znakowej - poprzez wspólny zestaw urządzeń peryferyjnych. Gdy na przykład procesor znakowy chce zainicjować rozkaz wejścia / wyjścia na urządzeniach peryferyjnych akurat w tym momencie przydzielonych procesorowi słowowemu - synchronizator wejścia/ wyjścia otrzymawszy takie zgłoszenie, oczywiście je odrzuci, przesyłając stosowny sygnał do pamięci stałej, która powoduje wpisanie tego rozkazu i do obszaru pamięci zarezerwowanego na zlecenia dla procesora słowowego; następnie procesor słowowy skoro tylko będzie gotów pobiera ten rozkaz i wykonuje.

Odwrotną procedurę obserwujemy w przypadku, gdy procesor słowowy chce wykonać operacje wydawnicze na urządzeniach przydzielonych procesorowi znakowemu. Wówczas sekwencja mikrorozkazów zawartych w pamięci stałej procesora znakowego powoduje zawieszenie realizowanego przez ten procesor programu; po utrwaleniu stanu wewnętrznego tego procesora w pamięci notatnikowej, następuje pobranie zleconego rozkazu wydawniczego;po wykonaniu tego rozkazu dokonuje się przywrócenie oryginalnego stanu procesora znakowego. Realizacja takiej procedury wymaga 30-50 słów pamięci stałej.

Podobnie można poprawić wydajność kompilatora drogą analizowania aktualnych wydań kompilatorów i określania oszczędności czasowych wynikających z dodania specjalnych rozkazów. Badania przeprowadzone w tym zakresie wykazały,że samo wprowadzenie takich rozkazów zwiększa szybkość przetwarzania o około 10-15%; gdyby jednak takie specjalne instrukcje można było użyć w czasie układania kompilatora, wtedy cały dotychczasowy obraz ulega zmianie i można zaobserwować wzrost szybkości sięgający 30-40%.

Stosując specjalne funkcje w procesorach języków uzyskujemy dalsze korzyści. Stosowana w pracy konwersacyjnej technika interpretacyjna instrukcji FORTRAN'owskich absorbuje znaczną część czasu maszynowego. Można tutaj uzyskać znaczne oszczędności tego czasu, jeżeli translację najczęściej występujących fraz kodowych zrealizujemy na drodze firmware'owej.

Niewątpliwie nic nie stoi na przeszkodzie, aby dla każdego języka opracować specjalne funkcje lub rozkazy, utrwalone w osobnych bankach pamięci stałej. W ten sposób jeden i ten sam komputer można by szybko przełączać na pracę w języku FORTRAN, COBOL, PL/1 lub jakimkolwiek innym - w zależności wyłącznie od tego, który pakiet stałej pamięci sterującej został użyty. W miejsce takich kilku wymiennych pakietów pamięci stałych - co jest luksusem wręcz nie do pomyślenia dla większości użytkowników - producent może dostarczyć jeden, ale za to taki, którego zawartość mogłaby być łatwo wymieniana /quasi - stała pa-

pamięć sterująca/.

Do takiego rodzaju pamięci sterującej stosują się określenia: "wolny zapis - szybki odczyt" oraz "głównie odczyt".Aby była ona efektywna, pamięć ta musi pracować pod kontrolą programu głównego, przy czym szybkości rozczytania/wypisania powinny być zgodne z szybkościami układów logicznych. Występujące tutaj problemy techniczne są rozwiązywalne, ale można od razu zadać istotne pytanie: ile języków może dostarczyć producent? Jest rzeczą nieprawdopodobną, aby producent rozwinął na drodze firmware'owej więcej niż dwa lub trzy procesory, a nawet i może to być nie wskazane skoro takie pakiety firmware'owe trzeba by oddzielnie opracowywać dla każdego modelu komputerów w obrębie danej rodziny. Najogólniej rzecz bowiem biorąc, na poziomie mikrokodowym nie istnieje jakakolwiek zamienność.

D. ZNACZENIE MIKROPROGRAMOWANIA DLA PRZEMYSŁU KOMPUTEROWEGO

Z punktu widzenia producenta, mikroprogramowanie ma szereg zalet, a to z następujących m.in. względów:

- zastąpienie przez jednolitą konstrukcyjnie pamięć sterującą bardzo złożonego bloku sterowania w realizacji układowej prowadzi do generalnego uproszczenia oraz uporządkowania prac projektowo-konstrukcyjnych,
- łatwa adaptacja do technik symulacyjnych,
- maszyna mikroprogramowana jest łatwiejsza do zrozumienia i wygodniejsza w obsłudze,
- wykrywanie błędów można oprzeć na prostszych podstawach,
- możliwości diagnostyczne ulegają zwiększeniu,
- oprogramowanie, bez względu na różnorodność zastosowań staje się wysoko zunifikowanym produktem,
- pracochłonność wytwarzania sprzętu komputerowego ulega redukcji.

Generalnie rzecz biorąc, mikroprogramowanie pozwala producentowi złagodzić wiele "knotów" powstałych na etapie projektowania. Łatwość modyfikowania listy rozkazowej pozwala łagod-

niej traktować niezręczne założenia projektowe. Podobnie też zmiany organizacji logicznej na etapie bardziej zaawansowanego projektowania - co w praktyce jest chyba nieuniknione - przestają być zmorą odraczającą pierwsze uruchomienia na czas nieokreślony.
Po dodaniu nowych funkcji specjalnych - ułatwiających programowanie - można przystąpić niemal natychmiast do praktycznego sprawdzania nowo ustalonej koncepcji logicznej. Wreszcie, procedury mikrosterujące można symulować i sprawdzać na innym komputerze i to na długo przed ostatecznym zatwierdzeniem dokumentacji prototypowej.

Pamięć sterująca może również - w pewnym zakresie - dokonywać kontroli pracy komputera za pomocą procedur diagnostycznych. Każdy bit w słowie sterującym reprezentuje ściśle określoną funkcję, której wynik jest daleko bardziej wartościowym sprawdzianem aniżeli rezultaty klasycznych testów programowych. Kody wielobitowe spełniają podobną rolę w sprawdzaniu prawidłowości działania złożonych sieci logicznych, zawierających wiele bramek i przerzutników.

Ostatnio pojawiło się żądanie zwolenników idei firmware'u, aby producenci nie kryli się przed swymi klientami z oszczędnościami na pracach projektowych uzyskiwanymi dzięki mikroprogramowaniu. Okres realizacji nowego komputera - od koncepcji do gotowej maszyny - trwa średnio 4 lata. Ale w przypadku komputera HONEYWELL-4200, opartego na mikroprogramowaniu, okres ten twał tylko 2 lata; należałoby przeto oczekiwać, że wynikające stąd oszczędności odbijają się niższą ceną rynkową i zwiększonym napływem zamówień.

E. DLACZEGO PRZEMYSŁ KOMPUTEROWY UNIKA FIRMWARE'U?

Firmware niewątpliwie wpływa na zmniejszenie kosztu realizacji układowej, który jest głównym czynnikiem określającym opłacalność produkcji komputerów trzeciej generacji. Obecny

rozwój mikroprogramowania i obwodów scalonych doprowadził do bezprecedensowego wzrostu produkcji oraz ograniczania wymagań dotyczących kontroli technicznej, jak i dostawy systemów komputerowych.

Technika sterowania oparta na pamięci stałej -względnie jakiejkolwiek pamięci quasi - stałej otwiera przed mikroprogramowaniem olbrzymie możliwości. Jeżeli użytkownik, dysponuje doborem małych komputerów typu mikroprogramowanego, albo systemu "zastosowaniowo-zorientowanego", jak oferowane przez firmę Standard Computer - to może on rzeczywiście zmieniać pamięć sterującą posiadanego procesora, stosownie do potrzeby. Chociaż przyjęta w teorii, idea dopuszczenia użytkownika do tworzenia mikroprogramów nie wywołała entuzjazmu producentów. Są oni raczej zdecydowanie niechętni oferowaniu użytkownikom swobody w tym zakresie. Jak dotychczas, producenci średnich i dużych maszyn - może z wyjątkiem IBM-360/25 - ociągali się nawet z wyrażeniem zgody na wprowadzenie drobnych zmian lub uzupełnień w pamięci sterującej. I mają ku temu całkiem ważkie powody.

Główną obiekcją jest niemożność służenia wysokosprawną pomocą techniczną. Zmiana mikrokodów wymagałaby objęcia szerokim programem szkolenia personelu serwisowego, każda bowiem zmiana odpowiada tak jakby wyprodukowaniu nowego typu maszyny.Unieważniałyby one także większość wbudowanych procedur diagnostycznych i tak już komplikujących się w miarę projektowania coraz bardziej złożonych maszyn. Ogólnie rzecz biorąc,producenci sprzętu komputerowego kierują się polityką tworzenia wąskospecjalizowanego personelu serwisowego - a wobec znacznych trudności kadrowych są zdecydowanie przeciwni zwiększeniu wymagań szkoleniowych.

Producent ociąga się z wydaniem użytkownikowi zezwolenia na zmiany w pamięci mikrosterującej komputera także i z innych powodów. Kiedy źle działa procesor trudno jest dociec gdzie leży błąd: w programie użytkownika, w programie producenta, w firmwarze użytkownika,czy też w firmwarze producenta? Odpo -

wiedź na to pytanie jest znacznie trudniejsza w przypadku dużych komputerów niż małych, stosunkowo prostszych pod względem konstrukcyjnym. Co więcej, stworzenie użytkownikowi możliwości samodzielnego odpowiadania na takie pytania wiązałoby się z koniecznością udostniępnienia mu szczegółów dokumentacji stanowiących tajemnicę przemysłową producenta.

Niezależnie od powyższego, użytkownik realizując swoje mikrooprogramowanie niewątpliwie mógłby napotkać na poważne trudności w wykonywaniu tego w sposób efektywny. Narzędzia obecnie dostępne mikroprogramiście są równie nieporęczne, jak to miało miejsce ze zwykłym programowaniem w zaraniu rozwoju komputerów. Dr Yaohan Chu, profesor informatyki na Uniwersytecie Marylandzkim powiedział: "Jak przygotowanie programów dla pierwszych komputerów, tak i przygotowanie mikroprogramów jest obarczone nieuniknionymi błędami. Znaczenie tablic składających się z jedynek i zer, jakie należy sporządzać, jest trudne do rozszyfrowania[3/].

Możliwości stosowania języków asemblacyjnych przy mikroprogramowaniu są ograniczone, a odpowiednich kompilatorów jeszcze nie opracowano, podobnie jak systemów uruchamiania mikropro - gramów.

George Hoff, kierownik wydziału projektowania procesorów w firmie Honeywell wręcz uważa, że nie ma tu lepszej metody niż praktyczna reguła "prób i błędów".

Co więcej można dobrać taki mikrokod, aby realizujący go komputer spełniał dokładnie życzenia użytkownika - ale z ryzykiem popełnienia błędów w układach zabezpieczeń logicznych, które całkowicie podważają procedury ochronne systemu.

Tym niemniej, jeżeli potrzeby mikroprogramowania większych komputerów okażą się dostatecznie duże, niektóre z wymienionych problemów mogłyby być przynajmniej częściowo rozwiązane. I

3/ Yaohan CHU /wrzesień 1968/ "Języki wyższego rzędu do opisu mikroprogramowanych komputerów", /A Higher- Order Language for Describing Microprogrammed Computers/ University of Maryland, Computer Science Center, Technical Report 68 -78 /praca wykonana na zlecenie NASA/.

tak na przykład producenci już w najbliższej przyszłości mogliby opracować podstawowe systemy diagnostyczne, które sprawdzałyby maszynę w stosunku do pierwotnie zaprojektowanego stanu, oraz podjąć się odpowiedzialności za konserwację w tym zakresie. Gdyby mikroprogramy umieszczone przez użytkownika w pamięci stałej nie chciały dać się uruchomić - musiałby on sam wykryć przyczyny tych trudności i wprowadzić stosowne zmiany. Prowadziłoby to do sytuacji kłopotliwych dla obydwu stron: producenci znaleźliby się w sytuacji podobnej do poddostawców[4], a znowu tylko bardzo zamożni i doświadczeni użytkownicy ośmieliliby się podjąć tak rozszerzoną odpowiedzialność.

Innym rozwiązaniem dla użytkownika jest zażądanie, aby producent dokonał zmian w specjalnych rozkazach lub podprogramach na zasadzie "płatnego zlecenia". To mogłoby zapobiec naruszeniu praw do własności przemysłowej. Ale w istocie rzeczy sprowadzałoby się to do zaprojektowania specjalnego systemu; wysokie koszty takich prac mogą być czynnikiem istotnie odstraszającym, gdyż trudno byłoby znaleźć innych użytkowników chcących w nich partycypować, z uwagi na zapotrzebowanie na podobne usługi.

Ale nawet gdy pojedynczy użytkownik zgadza się ponieść wszystkie koszty związane z opracowaniem nowego systemu, jest rzeczą wręcz nieprawdopodobną, aby producent chciał wiązać swoje środki na rozwiązanie niepowtarzalnego problemu. Producent widzi większe zyski w rozwijaniu systemów zastosowań, ulepszaniu kompilatorów i tworzeniu bardziej wydajnych systemów operacyjnych, które byłyby atrakcyjne dla wszystkich użytkowników. Zatem rozwój niepowtarzalnego firmware'u nie będzie możliwy bez zasadniczych ulepszeń metodologicznych w technice produkcji.

Z chwilą pojawienia się takowych zapewne powstaną wyspecjalizowane "przedsiębiorstwa firmware'owe", zajmujące się opracowywaniem mikroprogramowanych procesorów języków. Opłacalność podejmowania szerszych wysiłków w tym kierunku wydaje się dosyć wątpliwa w chwili obecnej. Koszty rozwojowe będą tak długo niewspółmiernie wysokie w przemyśle komputerowym, póki każdy nowy model będzie wymagał oddzielnych pakietów firmware'owych. Prze-

[4] W oryginale "OEM supplier"

ba też w tym miejscu przyznać, że przedsiębiorstwa firmware'owe nie będą wręcz mogły tak głęboko zaangażować się w konserwację mikroprogramowania, tak jak to będą w stanie uczynić produ - cenci. W dalszej perspektywie wpływ tych przedsiębiorstw na budowę procesorów języków zapewne można będzie porównać z obserwowalnym współcześnie wpływem "przedsiębiorstw software'owych" na budowę pakietów programów użytkowych.

III. ROLA FIRMWARE'U Z PUNKTU WIDZENIA UŻYTKOWNIKA

A. TRZY POZIOMY KOMPLIKACJI W MIKROPROGRAMOWANIU

Rozpatrzenie różnych zastosowań firmware'u z użytkowego punktu widzenia wymaga uwzględnienia stosownych kryteriów klasyfi - kacyjnych. Chyba najbardziej logiczną jest klasyfikacja według poziomu złożoności, który - ogólnie rzecz biorąc - może być mierzony pojemnością pamięci sterującej. Użytkownik jest oczywiście zawsze najbardziej zainteresowany rozwiązaniami najmniej skomplikowanymi - i zwykle takie najpierw pojawiają się na rynku.

1. Poziom pierwszy

Najniższy poziom zastosowania firmware'u dotyczy sterowania urządzeń peryferyjnych. Na przykład SYSTEM-21 firmy Viatron obejmuje stałoprogramowany procesor, w którego bloku sterowania występuje mikroprogramowana pamięć stała. Użytkownicy pracu - jący na takich urządzeniach nie potrzebują zapoznawać się z problemami mikroprogramowania, powinni oni tylko w pełni zdawać sobie sprawę ze wszystkich różnorodnych funkcji, jakie może realizować pamięć stała.

Zastosowanie firmware'u w urządzeniach peryferyjnych pociąga za sobą dwie poważne konsekwencje. Po pierwsze wytwórca uzysku-

je możliwość zmieniania własności tych urządzeń stosownie do wymagań klientów; po drugie - oprogramowanie przestaje być wreszcie odpowiedzialne za realizację operacji wejścia/wyjścia. W rezultacie obserwujemy wzrost "inteligencji" urządzeń peryferyjnych.

W przeciwieństwie do reszty rozwiązań układowych, których ceny systematycznie maleją, koszt urządzeń peryferyjnych z dodatkowymi przystawkami firmware'owymi powinien według wszelkiego prawdopodobieństwa pozostać bez zmiany. Jednakże urządzenia peryferyjne oparte na firmwarze będą realizować więcej funkcji niż dotychczas, co oznacza, że ich eksploatacja stanie się bardziej opłacalna.

2. Poziom drugi

Następnym poziomem zastosowań firmware'u jest oprogramowanie specjalizowane małych komputerów. Niektórzy, lecz na pewno nie wszyscy, producenci opracowują listy rozkazowe dla specjalizowanych zastosowań; jednocześnie jest rzeczą niesłychanie rzadko spotykaną, aby poszczególni użytkownicy sami opracowali swoje własne mikroprogramy. Nadmienić jednak należy, że dotychczasowe doświadczenie eksploatacyjne nad komputerami mikroprogramowanymi jest raczej niewielkie, w dodatku zaś odnosi się głównie do zastosowań wąsko specjalizowanych procesorów.

Działalność handlowa producentów małych komputerów właściwie ukierunkowana jest na dostawców systemów kompletowanych, firmujących cudze wyroby, a nie na indywidualnych użytkowników. Aby producent małych komputerów uzyskał zdrowe podstawy finansowe, jest sprawą niemal niezbędną, aby zapewnił sobie duży portfel zamówień. Nic więc dziwnego w tym, że w większości przypadków małe mikroprogramowane komputery wchodzą najpierw w skład większych systemów, zanim właściwi użytkownicy wejdą z nimi w bliższy kontakt. W rezultacie więc decydujący głos w modyfikowaniu mikroprogramów dla małych komputerów będzie miał projektant większych systemów.

Małe mikroprogramowane komputery wykorzystują zwykle niewymazywalne pamięci stałe, oparte na rdzeniach ferrytowych lub diodach. Wszelkie przeto zmiany w konfiguracji bitów mikroprogramowych wymagają określonej fizycznej ingerencji, a więc przewinięcia rdzeni lub wymiany odpowiednich diod.

Chodzi więc o to, że w praktyce po prostu nie może być mowy o wprowadzaniu jakichkolwiek zmian w raz ustalonym firmware, zaprojektowanym pod kątem określonego zastosowania. Firma Burroughs wprowadziła w modelu TC-500 dysk jako pamięć sterującą, której zawartość może zostać zmieniona przez inżyniera serwisu technicznego; wydaje się jednak, że takie zlecenia nie będą liczne.

3. Poziom trzeci

Najwyższy poziom złożoności dotyczy mikroprogramowego uelastyczniania list rozkazowych w większych systemach komputerowych. W chwili obecnej użytkownik zazwyczaj nie jest w stanie opracować swych własnych zastosowaniowo zorientowanych mikroprogramów i rzeczywiście może nie być nawet świadomy tego, że to właśnie mikroprogramy określają funkcje wykonywane przez komputer.

Tym niemniej jeden producent, a mianowicie firma Standard Computer oferuje użytkownikowi możliwość opracowania swych własnych mikroprogramów zastosowaniowo zorientowanych. Na tym poziomie złożoności firma Standard jest odosobniona ze swoją ofertą. Co do firmy IBM, to model 360/25 posiada quasi-stałą pamięć mikrosterującą, lecz dotychczasowa polityka tej firmy nie dopuszcza możliwości wprowadzania w niej jakichkolwiek zmian przez użytkownika.
Wszelkie zmiany mogą być dokonywane tylko przez firmę, zgodnie z jej polityką rozwojową.

Rodzina IC komputerów firmy Standard charakteryzuje się pracą wielojęzyczną i szerokimi możliwościami adaptacyjnymi. Można tutaj zacytować oficjalne stanowisko tej firmy:

"Jeśli sobie życzy, użytkownik może sam modyfikować strukturę logiczną tego komputera i uzyskiwać zoptymalizowane systemy

dla różnorodnych problemów, zapewniając sobie radykalny wzrost efektywności zastosowań".

Każdy z modeli rodziny IC zawiera tzw. "wewnętrzny komputer". Firmowy prospekt tak go charakteryzuje: "Wewnętrzny komputer zawiera między innymi pamięć sterującą, układ nadzorczy[5], wbudowany na stałe translator, rejestr minirozkazów z dekoderami, miniarytmometr, sygnalizatory i rejestry wyjściowe.
Pamięć sterująca jest używana głównie do przechowywania superrozkazów, przechowywania danych i stałych używanych przez te superrozkazy oraz do chwilowego przechowywania danych wejściowo/wyjściowych przesyłanych pomiędzy pamięcią główną i kanałami zewnętrznymi".[6]

Koncepcja firmy Standard jest unikalna pod tym względem, że bazuje na quasi-stałej pamięci sterującej; inne komputery mikroprogramowane - z wyjątkiem modułu 360/25 firmy IBM - używają tylko pamięci stałej. Użytkownik może poza tym adaptować system IC stosownie do swoich wymagań, poprzez załadowanie pakietu mikropro - gramów typu MINIFLOWS. Dla użytkowników, którzy życzyliby sobie rozszerzenia pakietu MINIFLOWS firma przewiduje dwa udogodnienia: dodatkową przystawkę wejścia/wyjścia, która korzysta z około 60% zawartości pamięci sterującej oraz pakiet TARGET rozkazów reali - zujących dynamiczną zamianę zawartości fragmentów pamięci steru - jącej zawartością odpowiednich miejsc pamięci głównej.

B. PRZEPROGRAMOWYWANIE I PROBLEMY EMULACYJNE

Emulacja odnosi się do pojawiających się przejściowo potrzeb przeprogramowywania. Jest już rzeczą praktycznie udowodnioną, że emulacja ułatwia użytkownikowi rozwiązywanie wielu problemów o - programowaniowych, jakie pojawiają się przy wymianie sprzętu kom - puterowego II generacji na sprzęt trzeciej generacji. Można spo - dziewać się, że przeprogramowanie stanie się znacznie mniej istotnym problemem dla użytkowników z chwilą pojawienia się komputerów

5/ W oryginale "scheduler"
6/ IC-700 System Manual, wydawnictwo firmy Standard Computer Corp. /Form 7002-1/

czwartej generacji - choćby dlatego, że do tego czasu techniki emulacyjne zostaną niewątpliwie ulepszone. Również należy wziąć pod uwagę, że rozpowszechnienie języków wyższego rzędu może całkowicie wyeliminować potrzeby emulacyjne. Wymagałoby to jednakże nie tylko pojawienia się szeroko dostępnych procesorów języków, lecz także wprowadzenia wręcz zakazu stosowania języków asemblacyjnych w nowych instalacjach komputerowych do przetwarzania danych.

Jednak w dalszym ciągu pozostaje pewne duże "ale": emulowanie będzie stosowane jeszcze przez dłuższy okres czasu, ponieważ w dalszym ciągu nadal eksploatuje się wiele programów drugiej i trzeciej generacji. Dla wszystkich tych, którzy tego potrzebują, emulator może być uważany za symulator interpretacyjny, którego wysoką wydajność zapewni mikroprogramowo zrealizowana pętla interpretacyjna - pobranie rozkazu, zrealizowanie dostępu do zawartości wyspecyfikowanego adresu oraz wykonywanie wskazanej operacji. W każdym problemie emulacyjnym występuje swoisty konglomerat problemów firmware'owych, hardware'owych i software'owych - zależnie od kompromisów, które były osiągalne wtedy gdy projektowano dany emulator.

Jeden z komputerów firmy Standard, a mianowicie model IC-400 jest w istocie niczym innym jak uniwersalnym emulatorem. Został on bowiem specjalnie zaprojektowany z myślą o wykonywaniu mikroprogramów - umieszczonych w pamięci sterującej - emulujących działanie takich maszyn firmy IBM jak modele 1401, 1130, 7040/7044 i 7090/7094, tak aby programy ułożone dla każdej z tych maszyn mogły być użytkowane bez żadnych modyfikacji. Program, który ma być emulowany, trzeba najpierw umieścić w pamięci głównej; następnie - kiedy program ma być realizowany - kolejno jego rozkazy są pobierane z tej pamięci głównej i wszystkie wskazane operacje indeksowania i adresowania pośredniego są wykonywane. Potem następuje dekodowanie i wykonanie tego rozkazu przez mikroprogram, który realizuje wszystkie niezbędne operacje składowe. Powyższy cykl jest powtarzany dla każdego rozkazu zawartego w programie emulowanym.

Inni producenci oferują na bazie firmware'u emulatory dla różnego typu komputerów, a specjalnie dla maszyn starszych typów. Rodzina SPECTRA firmy RCA może emulować działanie modeli RCA-301 i RCA-501. Komputer IBM-360/85 może emulować działanie maszyn typu 7040/7044 i 7090/7094, a komputer IBM 360 / 30 działanie wszystkich maszyn rodziny IBM-1400. Właściwie możliwa jest emulacja komputera. W istocie technika emulowania uległa tak szerokiemu rozpowszechnianiu, że prawdopodobnie aż 75% użytkowników maszyn trzeciej generacji IBM stosuje ją w tej lub innej postaci, przeznaczając około 75% swojego czasu maszynowego na prace emulacyjne.

Doświadczenia firmy American Tabulating dowodzą, że czas ten może być dobrze wykorzystany. Firma ta swego czasu zbudowała bibliotekę podstawowych programów dla maszyn IBM II generacji typów 1402, 7044 i 7094; bibliotekę tę firma postanowiła wykorzystać w sprzęcie komputerowym III generacji. Istniały dwie możliwości rozwiązania tego problemu. Jak podaje generalny dyrektor firmy American Tabulating, Jim Corcoran, pierwszą z nich było wydzierżawienie komputera IBM-360/65 za około 60 tys. dolarów miesięcznie. Drugą możliwością - którą zaakceptowano i zrealizowano - było wydzierżawienie komputera IC-4000 za 15 tys. dolarów miesięcznie, zdolnego do emulowania wszystkich trzech maszyn IBM drugiej generacji. Taka decyzja oznaczała preferowanie oszczędności przed szybkością, gdyż komputer IC-4000 realizuje programy IBM 7094 około dwa razy wolniej, a programy IBM-1401 w czasie zbliżonym do tego, w jakim były realizowane przedtem na komputerach oryginalnych. Te różnice wydajności nie usprawiedliwiałyby widocznie zwiększenia wydatków firmy o 45 000 dolarów miesięcznie.

C. ADAPTACJA DO PROBLEMU

1. Doświadczenia projektanta systemów

Pewne przedsiębiorstwo projektujące duże systemy zdecydowa-

ło się na rewizję listy mikrorozkazów małego komputera, który miał pracować w systemie wielodostępu. Powodem tej rewizji było dążenie do zwiększenia wydajności rzeczonego komputera, a przy okazji także i ułatwień w programowaniu. I tak np. następująca sekwencja mikrooperacyjna: ustaw licznik adresów, dodaj 1 do licznika, zapamiętaj nową zawartość licznika - w starej liście wymagała 3 mikrorozkazów 32 -bitowych, a w nowej daje się załatwić już tylko 1 mikrorozkazem 32-bitowym. Oszczędza się więc w ten sposób 64 bity pamięciowe na każdorazowym wykonaniu tego mikrorozkazu, nie licząc skrócenia czasu układania oraz uruchamiania programów. Ponadto projektant systemu dodał kilka specjalnych udogodnień, takich jak zmienny przecinek z 20-cyfrową mantysą, podczas gdy komputer w stanie dostarczonym przez producenta mógł pracować w zmiennym przecinku tylko z mantysami sześciocyfrowymi.

Sumarycznym wynikiem wspomnianej rewizji mikrokodu było około dwukrotne zwiększenie szybkości oraz zmniejszenie pojemności pamięci stałej. Jednakże autorzy tej rewizji stwierdzili, że uzyskana przez nich poprawa sprawności maszyny mogła i powinna być dokonana przez producenta komputera. Gdyby pierwotna konstrukcja była dostatecznie przemyślana, projektant systemu nie musiałby wykonywać żmudnych prac związanych z mikroprogramowaniem.

Na zaprojektowanie specjalnych mikrokodów projektant zużył około czterech roboczo-miesięcy. Sam problem był dosyć trudny a czas jego rozwiązania okazał się dosyć długi, pomimo zatrudnienia do mikrokodowania programisty systemowego doświadczonego w programowaniu w języku maszyny obeznanego z hardwarem i posiadającego nawet dyplom inżyniera elektryka. W przeciwieństwie do tego, co twierdził producent małych mikroprogramowanych komputerów, projektant systemu uważa, że osoba wykonująca mikroprogramowanie musi coś wiedzieć o budowie logicznej komputera; ponadto, zdaniem tego projektanta: "hardware ma pewne cechy przemilczane w instrukcjach producenta, które należy znać; mikrokodowanie jest niezwykle precyzyjne i bardzo trudne do sprawdzenia". Producent w omawianym przykładzie dostar-

cza symulator do sprawdzania mikrokodów, lecz zdaniem projek - tanta systemu chociaż symulator ten da się uruchomić, jest trudny w użyciu.

Wielodostępny system, którego częścią jest omawiany mikro - programowany komputer, znajduje się w dalszym ciągu w stadium rozwoju - przeto niemożliwe jest określenie jego osiągów i zdolności przerobowej.
A ponieważ funkcje realizowane przez mikroprogramowany kompu - ter można również wykonywać na takiej maszynie, której użyt - kownik nie może zmieniać - trudno więc stwierdzić jasno, co o- statecznie da się uzyskać dzięki możności zmiany mikropro - gramów.

Projektant systemu sądzi , że całkowity koszt zakupu komputera oraz rewizji jego mikroprogramowania był mniejszy niż procesora rozważanego jako rozwiązanie wariantowe. Jednak nie prowadzono szczegółowych analiz, ponieważ część prac mikroprogramowych wykonało podporządkowane przedsiębiorstwo, zaś czas pracy komputera UNIVAC-1108, za pomocą którego przeprowadzono "montaż" poszczególnych sekwencji mikrokodowych a następnie niezbędne symulacje nie był rejestrowany. Tak więc wydaje się, że w tym konkretnym przypadku mogło okazać się taniej a na pewno szybciej - gdyby projektant systemu zdecydował się od razu na zakup komputera z taką charakterystyką, jakiej potrzebował.

2. Zastosowania specjalizowane mikroprogramowanych komputerów

Przykłady elastycznych mikroprogramów, które mogą być zmieniane przez użytkownika lub dodawane do systemu przez produ - centa, są w dalszym ciągu rzadkością; z jednym lub dwoma wy - jątkami dotyczą one małych komputerów typu specjalizowanego. Najbardziej tutaj wyróżniają się niewątpliwie procesory telekomunikacyjne, lecz istnieje również dostrzegalny popyt na zasto - sowania do sterowania procesów, automatycznego zbierania i rejestracji danych, z kontroli i prób przetwarzania sygnałów radarowych oraz analiz sejsmograficznych.

W szeregu aktualnie opracowywanych projektów małe komputery znalazły już zastosowanie jako procesory peryferyjne dla większych systemów. Ich podstawową funkcją jest odciążenie centralnego procesora od sterowania transmisją danych. Małe komputery znalazły szczególnie zastosowanie w dwóch dziedzinach, które oczekuje niewątpliwie szybki rozwój:

. koncentratory teledacyjne i synchronizatory jednostek końcowych,
. sprzęgi teledacyjne [7] dla większych maszyn.

Koncentratory typu programowanego mogą sprawnie multipleksować przesyłanie danych z kilku linii o małej szybkości na jedną lub więcej linii o średniej szybkości - co obniża koszty transmisji. Dawniej koncentratory były budowane wyłącznie jako konstrukcje czysto hardware'owe; pojawienie się nowych zadań prowadziło wówczas do konieczności zakupu nowego typu koncentratora lub wykonania kosztownych zmian układowych. Współczesny mały komputer z odpowiednim programem może wykonywać równie dobrze funkcje koncentratora; przegrupowywanie danych przed transmisją po liniach średniej szybkości, dokonywanie konwersji kodów, redakcja danych itd. W roli sprzęgów teledacyjnych, małe komputery mogą próbkować linie transmisyjne celem sprawdzenia przepływu danych, przetwarzać dane próbkowe oraz zestawiać napływające znaki kodowe w grupy wymagane przez jednostkę centralną.

Firmware może odegrać główną rolę w zastosowaniu do tych celów - przykładowo, takich małych komputerów jak PDP-8 lub DDP-416 z zestawem pamiętanych programów standardowych czy też procesorów firmy Interdata z wymiennym mikroprogramowaniem. Te ostatnie mogą służyć jako przykład udanego, jak dotąd, zastosowania małych komputerów - na Waszyngtońskim Uniwersytecie Stanowym. Pierwotnym zamiarem było zastąpienie lub uzupełnienie małym komputerem posiadanego przez Uniwersytet synchronizatora peryferyjnego typu IBM-2702, pod kątem następujących czterech istot -

---

[7] / W oryginale "communications interfaces".

nych założeń[8]:

- mały komputer - o modyfikowalnym oprogramowaniu - rozszerza elastyczność sterowania odległymi urządzeniami końcowymi;
- ceny małych komputerów zmalały do takiego poziomu, że oparte na nich systemy mogą kosztować taniej niż synchronizator IBM-2702;
- wobec wymagań IBM łączenia każdej dzierżawionej linii transmisji z synchronizatorem 2702 i adapterem 3233 oraz przystawkami 103A i 103F, ..., należy spodziewać się znacznych oszczędności poprzez obniżenie kosztów jednostkowych albo przez bezpośrednie podłączenie linii transmisji, też użycie tańszego typu danych na dzierżawionych łączach prywatnych;
- zrealizowanie w jednym małym komputerze peryferyjnym tylu zadań teledacyjnych, jak tylko to jest możliwe - pozwala oczekiwać radykalnej redukcji liczby i złożoności zmian w systemie na dużym komputerze.

W systemie tym zastosowano mały komputer MODEL-3 firmy Interdata, uzupełniony multipleksorem łączącym 32 teledatory małej szybkości z kanałem peryferyjnym MODEL-u 3 oraz przystawką do kanału multipleksującego IBM-2870. Otóż, aby ułatwić manipulowanie danymi multipleksora, firma Interdata przewidziała specjalny rozkaz BIM skoku multipleksowego[9] zawarty w pamięci stałej, który jest wykonywany w takt przerw zegarowych co każdą 1/7 czasu jednobitowego.

Rozkaz BIM służy do upakowania, rozpakowania, buforowania, formowania i rozdzielania danych przesyłanych szeregowo do lub z urządzeń końcowych.

8/ BURNER, MILLION, RICHARD&SOBOLEWSKI /1969/ "Zastosowanie małego komputera jako synchronizatora urządzeń końcowych w wielkim systemie komputerowym" /The Use of a Small Computer as a Terminal Controller for a Large Computing System/ AFIPS Conference Proceedings, Vol.34 /Spring Joint Computer Conference/.

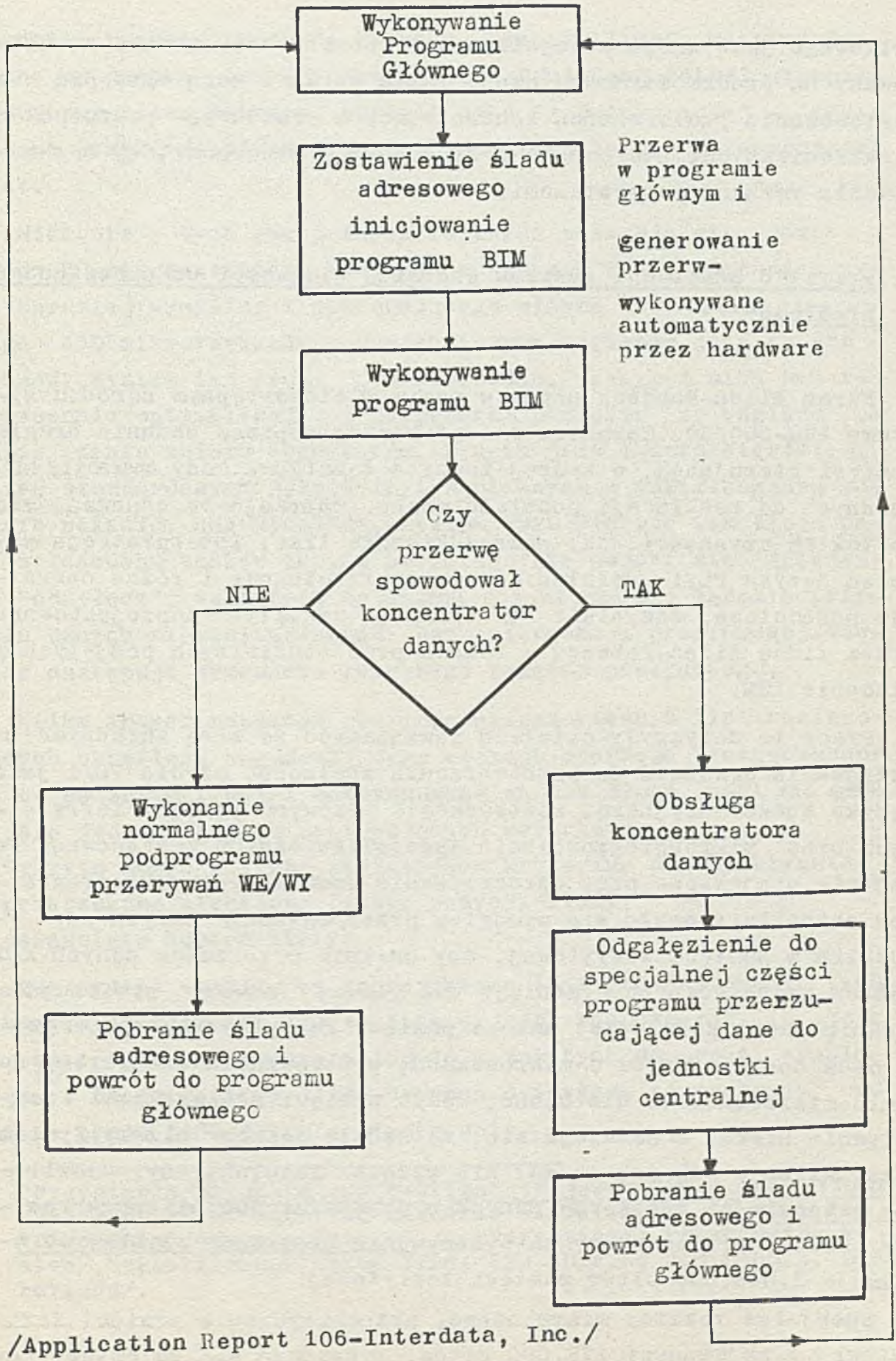
9/ W oryginale "Branch-If-Multiplex"

Po odebraniu ciągu bitów składających się na pełny znak alfanumeryczny zostaje on zapisany w ustalonym obszarze pamięci ferrytowej, odpowiadającym urządzeniu końcowemu,z którego został rozczytany. Podobną sytuację mamy przy wysyłaniu danych: treść jednoznakowa ustalonego obszaru zostaje rozczytana i przesłana do odpowiadającego urządzenia końcowego /por.schemat blokowy na na rys.4 z przykładowo użytym rozkazem BIM/. Trudno jest określić faktyczne koszty zaprojektowania rozkazu BIM, ponieważ było to wynikiem połączonych wysiłków firmy Interdata i Uniwersytetu; firma Interdata podaje tylko, że rozkaz ten będzie można wbudować do systemu teledacyjnego o łącznej cenie 20 tys. dolarów.

Projektanci systemu twierdzą, że zastosowanie małego komputera pozwoliło zrealizować cztery wyjściowe założenia. Obecnie Uniwersytet dodaje do systemu MODEL-4 firmy Interdata, umożliwiający pełne przetwarzanie danych; dane będą grupowane przez multipleksor pod kontrolą pamięci sterującej MODELU-3, który został przystosowany do współpracy z 64 urządzeniami końcowymi. W rezultacie wolny czas MODELU-3 - około 50% obecnie wykorzystywany do wykonywania innych zadań, zostanie zajęty przez rozszerzony rozkaz BIM. Jak powiedział przedstawiciel Uniwersytetu dr O.W.Richard, zamiast MODELU-3 można by również użyć innego małego komputera o stałym programie, ale całą jego moc absorbowałoby od początku wykonywanie funkcji realizowanych przez rozkaz BIM.

Uniwersytet stwierdził również, że synchronizator peryferyjny pozwala uprościć budowę i działanie systemu operacyjnego. To prawda, że trudno jest właściwie określić jaką w istocie rolę spełnia firmware w tych zastosowaniach - niemniej zapewnia on systemom nieuzyskiwalną innymi sposobami elastyczność, której przykładem jest rozkaz BIM i możliwość podwojenia liczby obsługiwanych linii transmisyjnych oraz pozwoli w przyszłości na dalsze modyfikacje.

Jeżeli w ogóle jakiś wniosek może być z tego wyciągnięty, to chyba taki, że jeśli jakiś znaczny fragment oprogramowania u-



/Application Report 106-Interdata, Inc./

Rys.4.SCHEMAT BLOKOWY PROGRAMU SKOKU MULTIPLEKSOWEGO "BIM" FIRMY INTERDATA

żytkowego jest wąsko specjalizowany, powinien on zostać zrealizowany na drodze firmware'owej. Takie warunki mogą wystąpić w zastosowaniu do urządzeń kontrolujących produkcję podzespołów elektronicznych, sterowania przyrządami i procesami, czy też sterowania różnymi podsystemami.

3. Przykład adaptacji systemu średniej wielkości do określonego problemu

Firma Allen-Babcock używa w swoim wielodostępnym ośrodku systemu IBM-360/50. Komputer zmodyfikowano poprzez dodanie drugiej pamięci sterującej, w której zawarto dodatkowe kody operacji niezbędnych do realizacji podziału czasu. Operacje te odnoszą się do takich czynności jak: przeszukiwanie list, interpretacja wyrażeń języka PL/1, działania zmiennoprzecinkowe i różne operacje pomocnicze. Wszystkie te operacje zostały zaprojektowane przez firmę Allen-Babcock w ramach prac studialnych podjętych na zlecenie IBM.

Prace te dotyczyły czterech powiązanych ze sobą zagadnień: użytkowania urządzeń do przetwarzania zdalnego, użycia PL/1 jako języka konwersacyjnego, zastosowania masowych pamięci ferrytowych oraz mikrooprogramowania specjalistycznych zastosowań. Podejście projektowe przy opracowywaniu dodatkowych mikroprogramów charakteryzowało się wymogiem przechowywania danych użytkownika w pamięci ferrytowej, aby uniknąć przerzutów danych z/do bębnów magnetycznych. Istnieją dwa główne powody preferujące właśnie takie podejście: masowe pamięci ferrytowe charakteryzuje czas dostępu rzędu 6 mikrosekund, w porównaniu z czterema do ośmiu milisekundami dla bębna; koszt pamięci masowej jest relatywnie niski, a oczekuje się, że będzie jeszcze niższy. System firmy Allen - Babcock okazał się wysoce interaktywny; umożliwia bezpośredni dostęp do danych o objętości 600 milionów znaków; obszar przeznaczony na wykonywanie programów użytkowych obejmuje 2.048.000 bitów pamięci ferrytowej.

Specjalne rozkazy mikrokodowe, przechowywane w pamięci ferrytowej o pojemności 128.000 bitów, rozpadają się na cztery ka-

tegorie: rozkazy "gęste", rozkazy "częstotliwe", rozkazy "zależne od danych" oraz rozkazy, które Paul Desjardins, wiceprezes firmy Allen Babcock, określa jako "z dużą szansą" - takie jak na przykład oczywiście niezbędne rozkazy do przetwarzania list.

Niektóre z tych specjalnych rozkazów składają się nawet z 15 do 20 mikrorozkazów. Realizacja systemu EVAL, uważanego za najbardziej wymyślny i posiadającego własne kody operacyjne, wymaga 450 mikrorozkazów i użycia innych rozkazów jako podprogramów. System ten służy do wielu zadań. Jednym z nich jest realizowanie optymalizującej gospodarki pamięcią ze względu na zmniejszenie zbioru argumentów. Drugim jest interpretowanie wyrażeń standardowych języka PL/1 zapisanych w modyfikowanym zapisie polskim. Raz wywołany, system EVAL pracuje tak długo, póki w łańcuchu znaków zapisu polskiego nie pojawi się specjalny "odsyłacz", względnie do momentu pojawienia się żądania przesyłu danych nierealizowanego bez interwencji programowej. Wówczas następuje wykonanie kolejnego rozkazu maszynowego.

Kilka innych rozkazów służy do przeszukiwania list zawierających określony argument; tego rodzaju operacje rozpoczynane są od początku listy i kontynuowane są tak długo, póki nie spełni się jeden z trzech następujących warunków:

. wykryto pozycję listy spełniającą kryterium przeszukiwania,
. przeszukano określoną liczbę pozycji listy,
. osiągnięto koniec listy.

Desjardins ocenił, że zaprojektowanie i przetestowanie tych specjalnych rozkazów zajęło firmie jeden roboczo-rok i około 75 maszynogodzin komputera IBM-7094 zużytych na prace symulacyjne[10/]. Obecnie firma Allen-Babcock dzierżawi te programy od IBM za około 1200 dolarów miesięcznie.

10/ "Przystawka hardware'owa realizująca język RPQ w modelu 360/50" /Description of Model 50 Hardware ASSIST RPQ/. "System Programowania Konwersacyjnego" /Conversational Programming System/, opublikowana przez firmę IBM /Doc.No 360 D-03.4. 016, revised/.

Wymienione rozkazy są widocznie warte tych wydatków. Desjardins powiedział, że komputer IBM-360/65 z równoważnym software wcale nie pracuje szybciej niż zmodyfikowany komputer IBM-360/50; co więcej, niektóre z przebiegów maszynowych były dwa do trzech razy szybsze na modelu 50. Ogólne polepszenie wydajności ocenił on na 25 do 30%, nie licząc znacznych oszczędności na objętości programów użytkowych w pamięci ferrytowej.

Niestety, rozkazów tych nie można w prosty sposób przenosić na inne instalacje komputerowe, jak pakiety zastosowań. Zostały one zaprojektowane z myślą o zaspokojeniu doraźnych potrzeb firmy Allen-Babcock. Z tego powodu, chcąc takie rozkazy zastosować w innej instalacji, należałoby rozpocząć prace projektowe od początku. Pewną analogię może stanowić problem efektywnego obliczania pierwiastków kwadratowych. Teoretycznie najlepszym rozwiązaniem byłoby ułożenie i wbudowanie odpowiedniego mikroprogramu - ale w przypadku systemu wykorzystującego taki mikroprogram jedynie sporadycznie, byłoby to nieuzasadnione, jako stosunkowo kosztowne.

Śledzenie wydajności pracy maszyny jest pożyteczną metodą określania czy firmware może być zaadaptowany do określonego problemu. Śledząc wykonywanie programów czasochłonnych można wykryć, które z nich należy mikrozaprogramować, aby polepszyć wydajność systemu - przy założeniu, że użytkownik dysponuje kimś, kto byłby w stanie opracować pożądany firmware i że producent zapewni niezbędną pomoc przy wykonywaniu tej pracy.

4. Przykład adaptacji komputera wielodostępnego

Firma Call-A-Computer zastosowała język asemblacyjny MINIFLOW do zaprojektowania 85 specjalnych rozkazów dla użytkowanego przez nią komputera IC-7000 firmy Standard. Większość rozkazów zaprojektowano, aby poprawić wydajność asemblerów oraz kompilatorów. Firma Call-A-Computer prócz asemblacyjnego używa dodatkowo języków BASIC i FORTRAN.

Wykrycie podprogramów, które powtarzały się wielokrotnie w pracy komputera stanowiło punkt wyjścia do zaprojektowania większości z tych rozkazów. Rozkazy te różnią się złożonością. Kilka zbudowano tylko z dwóch mikrorozkazów, podczas gdy inne wymagały od 30 do 40 mikrorozkazów. Trudno jest określić dokładniej stopień złożoności tych instrukcji, ponieważ wiele z nich korzysta z podprogramów już mikrozaprogramowanych przez producenta. Zatem jest możliwe, że jeden rozkaz obejmuje 200 mikrorozkazów, podczas gdy drugi ma tylko 10, ponieważ dalszych dwieście występuje w podprogramach. W rzeczywistości jeden rozkaz może się wielokrotnie włączać i wyłączać z rozmaitych mikroprogramowanych podprogramów w trakcie swego wykonywania.

Przykładem specjalnego rozkazu dodanego przez firmę Call-A-Computer jest operacja, którą nazwano "przerzut binarny"; pozwala ona na przemieszczenie łańcucha złożonego z 1 do 36 bitów z obrębu jednego słowa na dowolne miejsce w innym słowie. Inny rozkaz "porównanie binarne" służy do porównania łańcucha zawartego w jednym słowie z identycznej długości łańcuchem zawartym w innym słowie; np. bity 3-9 jednego słowa porównywane są z bitami 14-20 drugiego słowa. Takie "przerzuty" i "porównania" wykorzystuje się do budowania rozkazów maszynowych przez asembler lub kompilator.

Firma Call-A-Computer zrealizowała również pełny zbiór mikrorozkazów typu mieszanego, które upraszczają podprogramy softwareowe. Rozpatrzmy np. mnożenie liczby stałoprzecinkowej przez zmiennoprzecinkową. Normalnie wynik moglibyśmy uzyskać dopiero po kilku rozkazach, najpierw konwertujących liczbę stałoprzecinkową do postaci zmiennoprzecinkowej, następnie mnożących obie liczby zmiennoprzecinkowe i ewentualnie jeszcze sprowadzających wynik do postaci stałoprzecinkowej, gdyby to było konieczne . Tymczasem teraz wystarczy do tego jeden rozkaz specjalny.

Jak podaje firma Call-A-Computer zaprojektowanie każdego z wymienionych 85 rozkazów specjalnych wymagało cztery do sześ-

ciu roboczotygodni pracy. Na koncepcję rozkazu poświęcono o - koło dwa roboczotygodnie, jeden roboczotydzień na pisanie mikroprogramów i czwarty tydzień na ich uruchomienie. Co zyskała firma Call-A-Computer? Nie można tego określić w sposób precyzyjny, a to z powodu jeszcze zbyt małego doświadczenia. Tym niemniej firma wyliczyła, że łączny czas operacji skrócił się średnio o około 20%, a w szczególnych przypadkach nawet o 50 do 60%.

IV. PODSUMOWANIE

A. WNIOSKI BEZPOŚREDNIE

1. Sterowanie urządzeń peryferyjnych

Takie jednostki, jak stacje zbierania i emisji danych oraz inne urządzenia peryferyjne będą budowane z coraz większym udziałem firmware'u. Jednostki te staną się jednocześnie bardziej równouprawnione, zgodnie z aktualnym trendem w rozwoju środków technicznych informatyki, wyrażającym się przydzielaniem zdolności przetwarzaniowej wszystkim jednostkom składowym systemu komputerowego.
Dla większości sprzętu tego rodzaju szybkość działania pamięci stałej nie odgrywa roli krytycznej. Na przykład dla kilku tysięcy urządzeń końcowych typu TC-700, zamówionych przez kilka największych banków angielskich w firmie Burroughs czas reakcji pamięci bloku sterowania jest po prostu nieistotny w stosunku do układów firmware'owych zainstalowanych w tych urządzeniach. Urządzenia typu TC-700 stanowią odmianę typu TC-500 i współpracują bezpośrednio z jednostką centralną z szybkością zależną od czynności kasjera bankowego.

W innych systemach przystosowanych do pracy operatorów, urządzenia zawierające wolne pamięci stałe mogą być rozwiązane nawet jeszcze ekonomiczniej. W różnych urządzeniach peryferyj-

nych, łącznie z dyskami, taśmami, bębnami i ekranopisami, sterowanie będzie w przyszłości z reguły realizowane drogą firm - ware'ową. Dodatkowe moduły pamięci stałej mogą również być przy - łączane peryferyjnie do jednostek centralnych w charakterze urządzeń sterujących /synchronizatorów/. Takie urządzenia peryferyjne będą w każdym razie zdolne do wykonywania bardziej złożonych czynności i będą dostosowywane prawdopodobnie od razu przez producenta, do wyspecjalizowanych potrzeb klientów.

Zwiastunem takich jednostek przyszłościowych jest synchroprocesor wejścia/wyjścia występujący w systemie komputerowym typu IC-7000 firmy Standard Computer. Może on obsługiwać do czterech kanałów zewnętrznych z odpowiednimi synchronizatorami, do dziesięciu jednostek taśm magnetycznych na każdy kanał, szybkie czytniki kart, drukarki wierszowe, do ośmiu jednostek pamięci z bezpośrednim dostępem oraz poprzez koncentrator komunikacyjny może obsługiwać zdalne urządzenia końcowe. Sercem tego procesora jest wymienny blok mikroprogramowanej pamięci sterującej, o pojemności 2048 słów 18-bitowych; w ten sposób użytkownik może adaptować synchro-procesor do swoich wymagań wejścia/wyjścia podobnie jak adaptowałby centralny procesor do swoich ogólnych wymagań systemowych.

Innym przykładem praktycznego zastosowania firmware'u jest SYSTEM-21 firmy Viatron[11], który będzie "obudowywać" mikroprocesor typu VIATRON-1101 z niewymiennymi mikroprogramami w pamięci stałej.
Pamięć modelu Viatron-1101 ma posiadać pojemność 512 słów 12-bitowych umożliwiającą wykonywanie do 512 rozkazów. Już w trakcie opracowywania niniejszej publikacji, firma Viatron zapowiedziała produkcję jednego typu takiego bloku firmware'owego- co prawda firma Viatron nie używa terminu firmware - który ma

---

11/ Firma Viatron podobno w 1971 roku znikła z rynku USA. Nie musi to jednak źle świadczyć o jej rozwiązaniach konstrukcyjnych /przyp.red./.

zawierać "wszystkie rozkazy odpowiadające funkcjom dziurkarki klawiaturowej, łącznie z rozróżnieniem dolnego i górnego poziomu klawiatury, ustawienia tabulatora, duplikacji i wprowadzania zer nieznaczących".

Wprawdzie użytkownik nie może sam definiować zawartości pamięci mikrosterującej, firma Viatron na dodatkowe zlecenie byłaby jednak w stanie opracować specjalne mikroprogramy.[12/]

W omawianych rozwiązaniach i innych aktualnie rozwijanych, wiele funkcji, które normalnie wykonywane są przez jednostkę centralną - realizowanych jest drogą firmware'ową i przez urządzenia peryferyjne. Takie zwiększanie "inteligencji" urządzeń peryferyjnych powiększy ogólną wydajność komputerów, gdyż odciąża jednostkę centralną od absorbujących manipulacji danymi.

2. Specjalizowanie małych komputerów

Można praktycznie przyjąć za regułę odnośnie zastosowania mikroprogramów w małym komputerze, że co bardziej specjalizowane fragmenty oprogramowania użytkowego powinny zostać zrealizowane na drodze firmware'owej. Należy jednak dobrze zastanowić się nad tym, co może być objęte firmware'm, ponieważ pamięci stałe są nadal stosunkowo drogie - reprezentują połowę kosztu małego procesora - a stosowanie firmware'u jest jeszcze nowością.

Wspomniana reguła ulegnie zapewne dezaktualizacji z chwilą masowego wprowadzenia na rynek wielkoskalowych obwodów scalonych /LSI/ i oczekiwanego obniżenia ich ceny. Niektóre firmy uważają, że moment ten nadejdzie wkrótce. Firma Viatron anonsowała podjęcie prac nad uniwersalnym 18-bitowym kompute - rem z pamięcią o pojemności od 4K do 32K, realizowanym w technice LSI. Jeżeli tylko obwody te będą działać tak, jak zapewnia Viatron, koszt firmware'u spadnie do ułamka tego, co znamy obecnie. A zatem prawdopodobnie i inne małe komputery będą

12/ Oryginalna nazwa bloku "Microprogram Record Data/Entry".

budowane w technice LSI, łącznie z pamięciami mikrosterującymi. Na życzenie użytkownika, producent będzie prawdopodobnie wprowadzał w nich określone zmiany adaptacyjne.

Różne pakiety firmware'u, takie jak pamięci stałe występujące jako część SYSTEMU-21, prawdopodobnie będzie można zamawiać fakultatywnie. Może to doprowadzić do standaryzacji wielu funkcji. Użytkownik będzie mógł wtedy wybrać któryś z dostępnych pakietów, nawet gdyby dokładnie nie odpowiadał jego wymaganiom, bo nie będzie chciał tracić czasu na opracowanie własnych rozkazów. Jest to prawdopodobnie bardziej opłacalne, płacić trzeba bowiem nie za elementy systemu, ale przede wszystkim ludziom, którzy decydują, jak te elementy ułożyć.

3. Polepszanie pracy większych komputerów

Firmware może w sposób istotny poprawić wydajność średnich i wielkich komputerów, ale tylko dla takich nielicznych użytkowników, którzy mogą poprzestać na wąsko specjalizowanych zastosowaniach. Przeszkody, jakie należy przezwyciężyć, będą znechęcać pozostałych. Obecnie użytkownik, który chce przystosować firmware do swoich potrzeb, drogą zaprojektowania swoich własnych mikroprogramów, reprezentowałby kierunek przeciwny trendowi ułatwiania komunikacji między człowiekiem a maszyną.

Narzędzia dostępne dla mikroprogramisty są także wyjątkowo toporne. Chociaż istnieje kilka języków asemblacyjnych dla producentów i jeden dla użytkowników, nikt dotąd jeszcze nie opracował języka wyższego rzędu do celów mikroprogramowania. Ale nawet gdyby języki tego rodzaju były opracowane to i tak upłynie jeszcze kilka lat zanim pojawią się procesory do ich przetwarzania. Na razie więc, jeżeli użytkownik zażąda zmiany własności posiadanego komputera - przez dodanie lub zmianę mikroprogramów - to w efekcie zwiąże się na stałe z posiadanym sprzętem. Jego programów nie będzie można realizować na innej maszynie, a co więcej - nie będzie też żadnej wymienności z innymi komputerami danej rodziny.

Jeżeli użytkownik zechce zmienić systemy, będzie musiał za - projektować mikroprogramy na nowo. Jest niewielu, przynajmniej na razie programistów zdolnych do efektywnego mikroprogramowa - nia. W rezultacie niemal cały trud mikroprogramowania spada o- becnie na projektantów hardware'u i programistów systemowych- i tak już przeciążonych swymi normalnymi zadaniami. Tymczasem do efektywnego mikrooprogramowania niezbędna jest znajomość opera- cji komputerowych aż do najmniejszych szczegółów konstrukcyjnych.

Użytkownik instalacji bogato oprogramowanej musi sam zadecy- dować, które z jego zastosowań mają być obudowane rozkazami "spe- cjalnie zamówionymi". Czy ma być projektowany firmware dla jed- nego z tych zastosowań, dla kilku z nich czy też dla wszystkich? Obecnie wysoki koszt nie pozwoliłby na tworzenie wielu ekstra - bloków pamięci stricte stałej. A znowu, gdy dynamicznie zmienne pamięci mikrosterujące zaczną być powszechnie stosowane -co u- możliwi swobodne przepisywanie firmware'u do lub z pamięci mi - krosterującej - pojawią się wtedy problemy utrwalania śladów mikroprogramowych, podobnie jak to ma miejsce obecnie w maszy - nach wieloprogramowych. A w takiej sytuacji zyski na zwiększo - nej sprawności operacyjnej byłyby - o ironio losu - tracone w rozdętym systemie operacyjnym. Wreszcie koszt zmian, wprowa - dzanych w mikrooprogramowaniu przez użytkownika, zwiększałby koszty oprogramowania użytkowego - gdyż najpierw trzeba opraco- wać koncepcyjnie superrozkazy, następnie je zaprojektować, a do- piero potem można oprogramować problem przy ich użyciu.

Ale te przeszkody nie odstraszą mniejszości, która będzie nie- ugięcie nalegać na wbudowanie pamięci quasi-stałej do systemów komputerowych. Szczególnie są w tym już zainteresowane liczne u- niwersytety, gdyż byłoby to wspaniałe narzędzie do eksperymen - towania i nauczania. Również użytkownicy przemysłowi stwierdzają- ąc, że większość czasu komputerowego zużywana jest na jedno tylko lub dwa zastosowania, mogą nabrać ochoty na pisanie mi - kroprogramów we własnym zakresie.

B. POSTULATY ROZWOJOWE

Najbliższa przyszłość firmware'u jest właściwie już wytyczona. Przede wszystkim więc firmware będzie stosowany w urządzeniach peryferyjnych, ułatwiając producentom realizowanie specyficznych wymagań ze strony klientów. Następnie już sam wzrost parku małych komputerów z mikroprogramowaniem typu wąsko specjalizowanego powodować będzie automatycznie dalszy rozwój firmware'u. Wiele z tych wąskich zastosowań będzie wymagać specjalnych procedur, na przykład w rodzaju rozkazu BIM firmy Interdata. Rozkazy tego typu po zaprojektowaniu i sprawdzeniu testowym mogą nadawać się się do wdrożenia w układach hardware'owych w ramach możliwych kompromisów - i zaoferowania do specjalizowanych zastosowań.

W przyszłości większe komputery będą wyposażone w quasi-stałe lub nawet dynamiczne pamięci sterujące. Quasi-stała pamięć sterująca - której jeden tylko blok jest potrzebny w procesorze jest bardziej opłacalna od stricte-stałej; dodatkowe mikroprogramy można przechowywać na dyskach lub w dużych pamięciach ferrytowych. Użytkownikowi prawdopodobnie stworzy się możliwość kombinowania pamięci quasi-stałych i stricte-stałych, co pozwoli na sprawniejsze realizowanie funkcji sterowania. W innym wariancie producent mógłby udostępniać dodatkowe bloki pamięci stałej, które użytkownik mógłby adaptować do swoich typowych problemów.

Zarówno klient jak i producent - z chwilą uświadomienia sobie elastyczności, jaką stwarza pamięć quasi-stała - muszą sobie postawić pytanie: jak z tej swobody należałoby w rozsądny sposób korzystać? Jednym z rozwiązań jest zaoferowanie użytkownikowi przez producenta tylko zestawu standardowych rozkazów oraz dostosowanego do tego zestawu systemu diagnostyki podstawowej; jeżeli użytkownik będzie chciał rozszerzyć lub zmodyfikować zestaw otrzymanych rozkazów - będzie wówczas musiał przyjąć pełną odpowiedzialność za całość mikroprogramowania. Najprawdopodobniej jednak wraz z upływem czasu producent zacznie oferować użytkownikowi pewne opcje w określaniu zestawu roz -

kazowego, ograniczając własne wymogi standaryzacyjne do istotnie niezbędnych z uwagi na parametry produkowanego hardware'u. Można sobie wyobrazić, że w stosowaniach opartych na jednym tylko języku będzie możliwa redukcja pamięci sterującej, a zaoszczędzoną pojemność będzie można wykorzystać do innych zadań.

Nie może to być dokonane już teraz. Podobno kompilator języka FORTRAN dla komputera IBM-7090 generując program przedmiotowy wykorzystywał tylko 19 spośród ponad 200 rozkazów stosowanych w tym komputerze, co oznacza absolutne niewykorzystanie pozostałych 90%[13]. Przy korzystaniu z pamięci quasi-stałej, można by było umieścić w pamięci sterującej tylko te potrzebne rozkazy.

W miarę zdobywania doświadczenia w mikroprogramowaniu, producent będzie mógł oferować szeroki wybór superrozkazów, podobnych do różnych rozkazów przeszukiwania list, zaprojektowanych przez firmę Allen-Babcock na zmodyfikowany komputer IBM-360/50. Przecież typowe funkcje, wymagające powtarzalnych decyzji logicznych, można mikrozaprogramować i oferować jako superrozkazy, które z czasem mogłyby stać się standardami przemysłowymi. Jednakże aby uzyskać szeroką akceptację superrozkazów trzeba je dostosować do języków wyższego rzędu; natomiast zakres zastosowań superrozkazów będzie wypadkową żądań użytkownika oraz polityki cen producenta. Zalety takiej akceptacji są bezsporne: superrozkazy nie tylko zwiększają wydajność programów użytkowych lecz także ułatwiają programowanie, a zatem obniżają związane z tym koszty. Co więcej, programista uzyska wolną rękę w selekcji i organizowaniu podprogramów.

Zatem użytkownik uzyska olbrzymią elastyczność doboru środków. Będzie w stanie zaadaptować cały komputer do specjalizowanego zastosowania, będzie w stanie dorzucić lub usunąć któryś rozkazów standardowych, będzie wreszcie miał możność zastosowania różnych typów superozkazów. Ponadto, niczym użytkow -

---

13/ FLYNN & MACLAREN /1967/ "Znów o mikroprogramowaniu". /Microprograming Revisited/. Proceedings of the ACM National Meeting.

nik wielojęzycznego komputera, będzie mógł elastycznie adaptować posiadaną maszynę do pracy w którymś z powszechnych języków, takich np. COBOL, FORTRAN, PL/1 itp. - dzięki wymiennym pakietom firmware'owym.

Wreszcie, będzie mógł zachować posiadaną bibliotekę programów, opierając się na emulatorach zakładowych do pamięci sterującej.

Tego rodzaju rozwój spotęguje możliwości komputerów, bez zasadniczych zmian w ich architekturze konstrukcyjnej. Jeszcze szersze prace prowadzone są w zakresie polepszania wydajności procesorów językowych. W końcowym wyniku funkcje kompilatorów zostaną wdrożone układowo /na poziomie hardware'u/ a firmware będzie tutaj odskocznią od software'u. Właściwie wydaje się, że już istnieje dążenie do bezpośredniego wykonywania języków wyższego rzędu - podejście, którego pionierem jest Burroughs - Corp.

Przykładem roli, jaką może spełniać firmware, jest obecnie realizowany komputer IC-9000 firmy Standard; posługuje się on podobno pamięcią sterującą, która realizuje dwie funkcje: tłumaczenie i wykonanie. Wbudowana "maszyna tłumacząca" ma interpretować program źródłowy, napisany w języku wyższego rzędu, w innym języku, który nadaje się do bezpośredniego wykonywania. W ten sposób program źródłowy byłby załatwiany w pojedynczym przebiegu maszynowym, dosłownie na poziomie systemu załadowczego, tj. z całkowitym pominięciem jakiegokolwiek kompilatora. Oczywiście, programy pisane w językach innych niż te, do których będzie przystosowana maszyna, trzeba będzie kompilować.

Uruchamianie programów będzie również możliwe bliżej poziomu języka źródłowego. Na przykład wystarczy kilka mikrorozkazów, ułożonych we właściwej kolejności, aby program DUMP zamiast w szesnastkowym - drukował w łatwo zrozumiałym zapisie dziesiątkowym.

C. DALSZA PRZYSZŁOŚĆ

Jak w skali długookresowej wpłynie na rozwój firmware'u po-

jawienie się systemów komputerowych zrealizowanych w technice układów wielkoskalowych /LSI/?

Firmware spełniał dotąd podwójną rolę: nie tylko w znacznym stopniu ułatwił produkcję komputerów, ale i umożliwił adaptację sprzętu uniwersalnego do wąsko specjalizowanych zastosowań. Wprowadzenie układów wielkoskalowych tak długo nie umniejszy roli firmware'u póki przemysł nie zacznie powszechnie stosować komputerów do sterowania produkcją obwodów o własnościach zaprojektowanych stosownie do wymagań klienta; a na razie pamięci stricte-stałe będą jednym z najłatwiejszych do wyprodukowania i testowania obwodów wielkoskalowych. Chociaż bowiem komputery są już wykorzystywane w projektowaniu obwodów wielkoskalowych o własnościach określanych przez zamawiającego - koszty takiej produkcji, łącznie z testowaniem, są jeszcze dość wysokie.

Ogólne skutki wpływu wielkoskalowych układów scalonych na firmware będą takie same jak na hardware: niższy koszt, mniejsze gabaryty, większa szybkość. W roku 1963 jeden mikroobwód mógł zawierać tylko od dwóch do trzech bramek. Obecnie opanowane technologie pozwalają na umieszczenie w tym samym gabarycie około 50 bramek logicznych nie licząc innych elementów elektronicznych; liczba elementów wielkoskalowo scalonych może w 40-końcówkowym pakiecie wahać się w granicach od 200-250 bramek aż gdzieś do 3000-5000 bramek. Pamięci quasi-stałe o pojemności 256 lub 512 bitów są już dosyć powszechne, a produkcja pamięci stricte-stałej o pojemności 2000-3000 bitów została już właściwie opanowana.

Niektórzy producenci komputerów zaczęli stosować "moduły hybrydowe" stanowiące mieszaninę dyskretnych elementów elektronicznych i średnioskalowych obwodów scalonych /MSI/ zawierających najwyżej około 100 bramek każdy. Na jednym 40-końcówkowym module krzemowym średnicy 8-13 cm można upakować od 300 do 500 obwodów średnioskalowych[14]. Zaprojektowanie pełnej konfiguracji obwodów średnio-skalowych w module hybrydowym, przy wykorzystaniu pomocy komputera trwa około tygodnia. W takiej

---

[14] W oryginale "flip-chips".

konfiguracji jeden obwód średnioskalowy może być np. ośmiobitowym przesuwnikiem, inny 256-bitową pamięcią stałą, itd., przy czym każdy z tych obwodów może być wyposażony w indywidualną bramkę sterującą, co upraszcza problem przełączania pamięci stałej lub innych elementów. Kombinując średnioskalowe obwody pamięci stałej można by budować superrozkazy, jakkolwiek praktycznie możliwe to będzie dopiero po tym, gdy rozwój teorii superrozkazów nawiąże do tej technologii.

Zastosowanie średnioskalowych układów scalonych będzie wyprzedzało szerszą akceptację układów wielkoskalowych /typu MOS lub bipolarnego/, torując w ten sposób drogę zastosowaniu wielkoskalowych pamięci scalonych w większych komputerach. Każda z tych pamięci mogłaby być mikroprogramowana i spełniać albo różne funkcje systemu operacyjnego albo też umożliwiać adaptację komputera do wymagań klienta. System komputerowy mógłby zresztą obejmować pewną liczbę mikroprocesorów wykonujących superrozkazy zrealizowane w technice wielkoskalowej, podczas gdy jednostka centralna wykonywałaby inne zadania.

Oparcie systemów komputerowych na wielkoskalowych układach scalonych będzie wymagało użycia licznych mikroprogramowanych pamięci stałych. Ale rozwiązanie ogółu kompromisów typu "firm - ware - czy hardware" będzie zależeć od konkretnych warunków ekonomicznych produkcji komputerów. W miarę postępu techniki projektowania komputero-wspomaganego być może, że dojdzie w końcu do takiej sytuacji, że przejście od założeń projektowych do realizacji hardware'owej będzie równie tanie jak do realizacji firmware'owej. A dojście do takiej sytuacji najprawdopodobniej oznaczałoby ostateczny koniec firmware'u: różnorodne superrozkazy byłyby wówczas realizowane układowo zaś adaptacja komputera do wymogów klienta następowałaby w drodze wymiany odpowiednich pakietów wielkoskalowych.

Cena zł 92.-

DRUK I ZNAK W-WA