ZESZYTY NAUKOWE POLITECHNIKI ŚLĄSKIEJ 1987

Seria: INFORMATYKA z. 8 Nr kol.882

Halina NOWAK
Piotr STRÓŻYNA

MIKROPROGRAMOWANIE SEGMENTOWYCH MIKROPROCESORÓW INTEL 3000 NA PRZYKŁADZIE STEROWNIKA TWARDYCH DYSKÓW

Streszczenie. W artykule przestawiono krótką charakterystykę rodziny mikroprocesorów segmentowych INTEL 3000 ze szczególnym uwzględnieniem jednostki sterowania mikroprogramem. Po naszkicowaniu koncepcji wykorzystania rodziny 3000 do konstrukcji sterownika twardych dysków przedstawiono strukturę mikroprogramu i format mikroinstrukcji. Szczegółowo omówiono problemy adresowania mikroinstrukcji. Podane zostały także pewne praktyczne uwagi i wskazówki projektowe. W końcowej części pracy opisano narzędzia tworzenia i uruchamiania mikroprogramu sterownika twardych dysków.

1. Rodzina mikroprocesorowa INTEL 3000

1.1. Ogólna charakterystyka rodziny

Seria INTEL 3000, oznaczana dalej w tekście przez I3000 (i jej odpowiedniki produkcji ZSRR i CSRS), stanowi bipolarną rodzinę mikroprocesorów segmentowych (ang. bit-slice) [1,2]. W skład rodziny wchodzą następujące elementy:

1) Jednostka sterowania mikroprogramem I3001 - jej zadaniem jest kontrola realizacji mikroprogramu, tj. wyznaczanie adresów kolejnych mikroinstrukcji oraz sterowanie flagami.
2) Element przetwarzający I3002 - wykonujący operacje arytmetyczne, logiczne, przesunięcia i przesyły na 2-bitowych słowach. Element posiada 3 magistrale wejściowe, dwie wyjściowe, wejścia i wyjścia przeniesień. Wyposażony jest w 12 rejestrów roboczych. Łącząc układ I3002 można wykonywać operacje na słowach dowolnej długości.
3) Sumator z jednoczesnym przeniesieniem I3003 - 8-bitowy.
4) Jednostka sterowania przerwaniami I3214 - przyjmuje zgłoszenia na ośmiu poziomach priorytetowych.
5) Buforowany rejestr 8-bitowy I3212.
6) Dwukierunkowe 4-bitowe bufory I3216 i 3226.

Ponadto firma INTEL oferuje dość bogaty zestaw pamięci ROM (PROM i EPROM) i RAM.

Wszystkie elementy rodziny można w łatwy sposób łączyć ze sobą. Ich budowa i sposób działania dają bardzo duże możliwości konstrukcji urządzeń cyfrowych przy wykorzystaniu tej serii.

Do zalet układów budowanych w oparciu o elementy rodziny I3000 zaliczyć należy:

- łatwość projektowania i uruchamiania,
- regularną strukturę,
- zmniejszenie ilości wykorzystanych elementów (o ok. 60-80% w porównaniu z tradycyjnymi układami TTL SSI i MSI),
- zmniejszenie pobieranej mocy,
- stosunkowo niskie koszty,
- dużą szybkość działania,
- dużą niezawodność.

Istotną cechą urządzeń zbudowanych w oparciu o mikroprocesory serii I3000 jest konieczność ich mikroprogramowania. Zarówno ostateczny format mikroinstrukcji, jak i sam mikroprogram musi określić projektant urządzenia. W ten sposób osiąga się bardzo dużą elastyczność, jednakże tworzenie mikroprogramu przy braku odpowiednich narzędzi wspomagających może być bardzo żmudnym i czasochłonnym procesem.

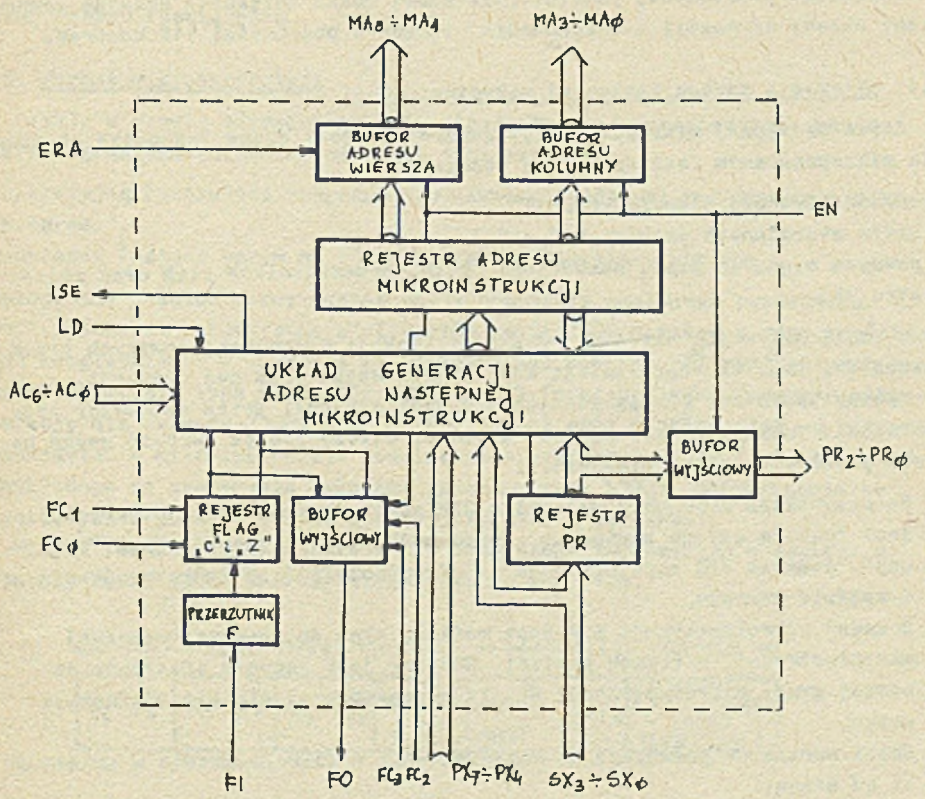
1.2. Jednostka sterowania mikroprogramem

Generalnym zadaniem jednostki sterowania mikroprogramem (sekwentera) jest kontrola przebiegu realizacji mikroprogramu poprzez wypracowywanie adresów następnych mikroinstrukcji.

Mikroprogram zapisany jest zwykle w pamięci stałej adresowanej bezpośrednio przez sekwenter. Rysunek 1.1. przedstawia schemat blokowy elementu I3001.

Znaczenie poszczególnych jego końcówek jest następujące:

MA_8-MA_4 - wyjście adresu wiersza pamięci mikroprogramów,
MA_3-MA_0 - " " kolumny " "
PR_2-PR_0 - " rejestru wewnętrznego PR,
PX_7-PX_4, SX_3-SX_0 - wejścia informacyjne,
FC_3-FC_0 - wejścia sterowania flagami,
FI - wejście flagi,
FO - wyjście "
LD - wejście wstępnego ładowania adresu mikroinstrukcji,
EN - " odblokowujące wyjścia adresu, flag i rejestru PR,
ERA - " " " adresu wiersza,
AC_6-A[illegible] - wejścia określające funkcję wyznaczającą adres następnej mikroinstrukcji (skok),
ISE - wyjście strobu przerwań (aktywne przy realizacji skoku JZR 15),



Rys. 1.1. Schemat blokowy sekwentera I3001
Fig. 1.1. A structure of I3001 sequencer

Opis elementu I3001 w niniejszej pracy ograniczono do minimum niezbędnego do zrozumienia całości, natomiast szczegółowe omówienie działania sekwentera można znaleźć w [1].

1.3. Organizacja pamięci mikroprogramów

Organizację pamięci mikroprogramów narzuca format słowa adresowego generowanego przez element I3001. Adres (9-bitowy) składa się z dwóch części: 5-bitowego adresu wiersza (rzędu) - bity MA_8-MA_4 i 4-bitowego adresu kolumny - bity MA_3-MA_0. Przy opisie funkcji wyznaczających adres następnej mikroinstrukcji (skoków) operuje się odrębnie adresem wiersza oraz kolumny. Zatem w wersji podstawowej pamięć mikroprogramów należy traktować jako matrycę składającą się z 32 wierszy i 16 kolumn. Każdy adres komórki takiej pamięci musi zawierać dwie składowe.

Pojemność podstawowej pamięci (512 słów) można zwiększyć dodając zewnętrzne układy adresacji i dekodowania bloków o pojemności 512 komórek.

1.4. Generacja adresu następnej mikroinstrukcji

Adres następnej mikroinstrukcji generowany jest przez jednostkę sterowania mikroprogramem (sekwenter) na podstawie:

- adresu bieżącej mikroinstrukcji,
- bitów sterujących skokami (AC_6-AC_0),
- pewnych sygnałów stanu sekwentera (stanu przerzutników flag oraz rejestru PR).

Stan linii AC_6 - AC_0 określa jedną z 11 funkcji skoku realizowanych przez sekwenter. Bity te reprezentują kod funkcji skoku oraz jej argument.

Zasady tworzenia adresu następnej mikroinstrukcji przez sekwenter przy realizacji funkcji skoków są dość złożone, należy jednak zwrócić uwagę na pewne prawidłowości, a mianowicie:

1. We wszystkich skokach z wyjątkiem JZR adres następnej mikroinstrukcji jest funkcją adresu bieżącego, zatem możliwy jest jedynie skok "względny". Jedynie JZR zapewnia skok do mikroinstrukcji o żądanym adresie w rzędzie zerowym.
2. Z danej mikroinstrukcji nie jest możliwy skok do dowolnie wybranej mikroinstrukcji w ramach pamięci. Możliwe jest jedynie przejście do pewnej grupy mikroinstrukcji ściśle sprecyzowanej dla każdej funkcji skoku.
3. Skoki warunkowe pozwalają na rozgałęzienia w mikroprogramie w zależności od stanu:

- linii wejściowych sekwentera
- przerzutników wewnętrznych układu

4. Ładowanie rejestru PR odbywa się podczas wykonywania skoku JPX.
5. W czasie realizacji skoku JCE odblokowywane są wyjścia przerzutników PR_2-PR_0. Wyjścia te mogą być wykorzystane dowolnie przez konstruktora.

Adresy mikroinstrukcji wypracowywane są przez sekwenter zgodnie z funkcją skoku określoną przez wejścia AC_6 - AC_0, pod warunkiem, że wejście ładowania LD jest nieaktywne. W przeciwnym przypadku do rejestru adresu następnej mikroinstrukcji sekwentera wpisywany jest stan linii SX i PX. Poza tym warto zauważyć, że bufory wyjściowe adresów wiersza i kolumny można zablokować (sygnałami ERA i EN) i wymusić równocześnie inny adres, zmieniając w ten sposób sekwencję realizacji mikroprogramu.

Opisane powyżej mechanizmy tworzenia adresu następnej mikroinstrukcji są dość skomplikowane i dowodzą, że adresacja poszczególnych mikroinstrukcji w przypadku wykorzystania sekwentera I3001 jest niełatwa. Biorąc pod uwagę, że poszczególne wejścia elementu mogą być zarówno sterowane bezpośrednio przez mikroprogram, jak też wymuszane przez dodatkowe układy

logiczne, zastosowanie sekwentera I3001 stwarza projektantowi duże możliwości wyboru rozwiązania.

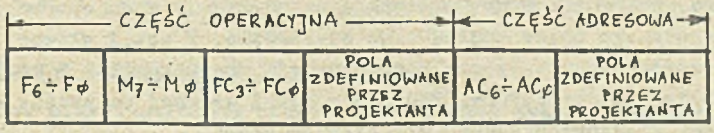
1.5. Format mikroinstrukcji

Zwykle w słowie mikroinstrukcji, niezależnie od przeznaczenia układu mikroprogramiwanego, można wyróżnić dwie części:

- operacyjną (sterującą bezpośrednio układami będącymi pod kontrolą mikroprogramu),
- adresową (mającą wpływ na adres następnej mikroinstrukcji).

Analogiczny podział można zastosować w przypadku scalonego sekwentera I3001. Jednakże wykorzystanie elementów serii I300 narzuca z góry konieczność uwzględnienia w formacie mikroinstrukcji pewnych pól wynikających z zasady działania tych elementów. Oprócz pól "stałych" zawierających mikrooperację dla układów I3001 i I3002 projektant może w zależności od potrzeb przewidzieć w mikroinstrukcji dowolne pola dodatkowe. Pola te mogą służyć przykładowo do sterowania układami spoza rodziny I3000. Rozszerzenia te mogą dotyczyć zarówno części operacyjnej, jak i adresowej.

Ogólny format mikroinstrukcji dla urządzenia zbudowanego w oparciu o elementy serii I3000 przedstawiony jest na rys. 1.2.



Rys. 1.2. Ogólny format mikroinstrukcji dla mikroprocesorów I3000
Fig. 1.2. A general format of a microinstruction for I3000 microprocessors

Oznaczenia:

F_6-F_0 -bity sterujące funkcją przetwarzania elementu I3002,
M_7-M_0 -tzw. maska dla elementu I3002,
FC_3-FC_0 -bity sterujące flagami elementu I3001,
AC_6-AC_0 - " " skokami " "

2. Mikroprogram sterownika twardych dysków

2.1. Koncepcja zastosowania rodziny INTEL 3000 w sterowniku twardych dysków

Koncepcja wykorzystania elementów serii I3000 do konstrukcji sterownika twardych dysków dla mikrokomputera ComPAN-8 została szczegółowo przedstawiona w [3]. Elementy te zostały wykorzystane w następujący sposób:

a) 4 elementy przetwarzające I3002 tworzą 8-bitową jednostkę przetwarzania (procesor segmentowy), której zadaniem jest:
   - pośredniczenie w przesyłach danych między układami peryferyjnymi,
   - przetwarzanie informacji,
   - przechowywanie ważniejszych danych w pamięci notatnikowej (rejestrach roboczych),
   - adresacja pamięci buforowej.

b) Sekwenter I3001 wraz z elementami współpracującymi i pamięcią mikroprogramu tworzą mikroprogramowany układ sterowania całym urządzeniem.

Wykorzystano ponadto buforowane rejestry I3212. Wszelkie układy pośredniczące w transmisji między jednostką dyskową a sterownikiem, układy interfejsu z mikrokomputerem ComPAN-8 oraz pamięć buforowa stanowią w sterowniku grupę układów peryferyjnych, adresowanych mikroprogramowo.

2.2. Struktura mikroprogramu

Sterownik twardych dysków od strony maszyny ComPAN widziany jest jako urządzenie wykonujące pewne elementarne operacje dyskowe (SEEK, SELECT, RESTORE itd. [3]) inicjowane przez oprogramowanie mikrokomputera. Każda z tych operacji elementarnych realizowana jest przez odrębną procedurę mikroprogramową. Wybór procedury następuje na podstawie kodu operacji elementarnej wysłanego przez ComPAN-a do sterownika. W pętli głównej mikroprogramu następuje wprowadzenie kodu operacji do sekwentera i rozgałęzienie do odpowiedniej procedury. Ponadto w pętli głównej ustawiane są wstępne wartości niektórych danych dla mikroprogramu.

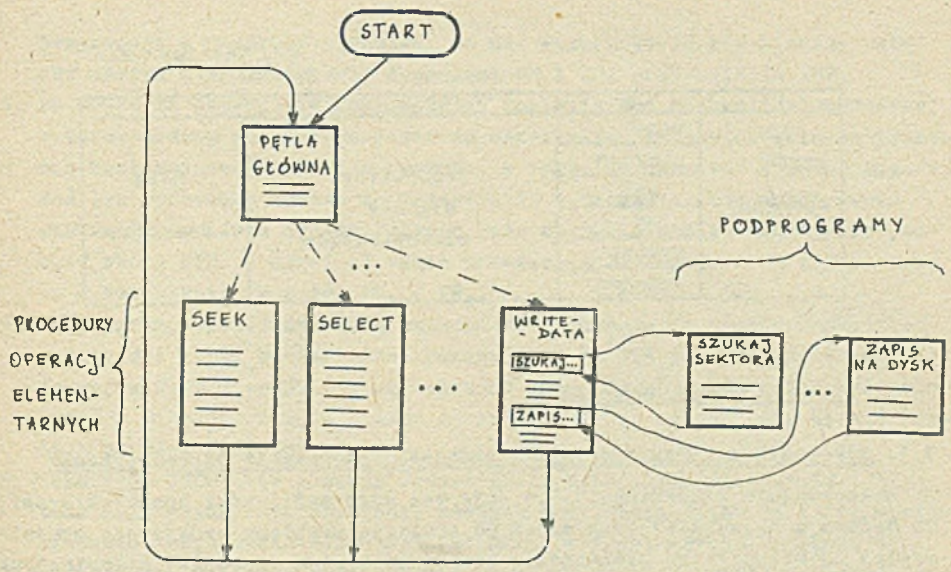
W czasie realizacji procedur operacji dyskowych powtarzane są pewne sekwencje mikroinstrukcji, np. szukanie sektora dla operacji zapisu i odczytu danych lub pola adresowego. Przyjęto, że powtarzające się fragmenty zostaną zorganizowane w podprogramy wywoływane przez procedury.

W rezultacie wyróżniono następujące mikropodprogramy:

- SZUKAJ SEKTORA,
- USTAW PARAMETRY BLOKU DANYCH,
- SPRAWDŹ POLE ADRESOWE,
- ZAPIS NA DYSK,
- ODCZYT Z DYSKU.

Rysunek 2.1. przedstawia ogólną strukturę mikroprogramu sterownika.

Wyróżniona została mikroinstrukcja startowa, od której zaczyna się realizacja mikroprogramu po inicjacji pracy sterownika.



Rys. 2.1. Ogólna struktura mikroprogramu sterownika twardych dysków
Fig. 2.1. A general structure of the hard disc controller microprogram

2.3. Format mikroinstrukcji sterownika

Słowo mikroinstrukcji sterownika twardych dysków składa się z 40 bitów. Oprócz pól stałych, wynikających z zastosowania rodziny I3000, mikroinstrukcja zawiera pola dodatkowe. Z reguły stosowano mikroprogramowanie poziome (pola kodowane), z wyjątkiem kilku bitów będących bezpośrednio sygnałami mikrosterującymi.

Format mikroinstrukcji sterownika przedstawia tablica 2.1.

Tablica 2.1

Format mikroinstrukcji sterownika

	OZNACZENIE	FUNKCJA	
część operacyjna	M_7-M_0	maska dla elementu I3002	pola stałe
	F_6-F_0	funkcja przetwarzania I3002	
	FC_3-FC_2	sterowanie flagami wej. sekwentera I3001	
	FC_1-FC_0	sterowanie flagami wyj. sekwentera I3001	
część adresowa	AC_6-AC_0	funkcja skoku dla I3001	
	SP_2-SP_0	sterowanie multiplekserami warunku	pola dodatkowe
część operacyjna	PA_3-PA_0	adres układu peryferyjnego	
	WR/RD	zapis do/odczyt z układu peryferyjnego	

od. tablicy 2.1

1	2	3	4
część operacyjna	BLOK DAT/CRC WG RG STROBE RESTORE	jednobitowe sygnały mikrostrujące	pola dodatkowe

3. Sterowanie mikroprogramem

3.1. Założenia i ograniczenia dla jednostki sterowania mikroprogramem

Zastosowanie sekwentera I3001 daje tak duże możliwości konstruktorowi, że jednym z najważniejszych zadań po analizie problemu wydaje się ograniczenie do kilku tylko mechanizmów tworzenia adresu następnej mikroinstrukcji i rezygnacja z pozostałych. Pozwoli to na uzyskanie stosunkowo przejrzystej struktury urządzeniowej i uproszczenie syntezy mikroprogramu.

W projekcie sterownika twardych dysków przyjęto następujące założenia i zasady sterowania mikroprogramem:

1. W momencie inicjacji pracy sterownika należy zapewnić wpis adresu mikroinstrukcji startowej do rejestru adresowego mikroinstrukcji elementu I3001. Cel ten można osiągnąć jedynie poprzez wpis do sekwentera stanu linii SX i PX sygnałem LD. Sygnał LD będzie aktywny w czasie zerowania mikrokomputera ComPAN, przy równoczesnym podaniu adresu 0-0 na wejścia PX i SX.
2. W celu umożliwienia testowania wyniku operacji logicznych i arytmetycznych w jednostkach przetwarzania I3002 połączono wyjście flagi F0 sekwentera z wejściem przeniesienia CI procesorów segmentowych I3002, a wyjście przeniesienia CO z wejściem flagi FI elementu I3001. Przewiduje się wykorzystanie obu flag sekwentera (Z i C).
3. Testowanie sygnałów stanu sterownika (RRD, RRC, TRC, TRD, DATA, SECTOR, ATTEN) będzie odbywało się mikroprogramowo przez modyfikację adresu podczas wykonywania skoku JPX. Sygnały stanu wprowadzone zostaną poprzez multipleksery na wejścia PX sterownika. Dla zwiększenia obszaru dostępnego skokami JPX na jedno z testowanych wejść wprowadzi się wyjście F0 sterowane mikroprogramowo.
   Zrezygnowano z testowania sygnałów stanu przez przerwania, gdyż ich implementacja byłaby bardziej złożona sprzętowo i mikroprogramowo (np. problem pamiętania śladu i powrotu z przerwania), a przyjęte rozwiązanie mikroprogramowe jest dostatecznie szybkie dla zapewnienia poprawnej pracy sterownika.

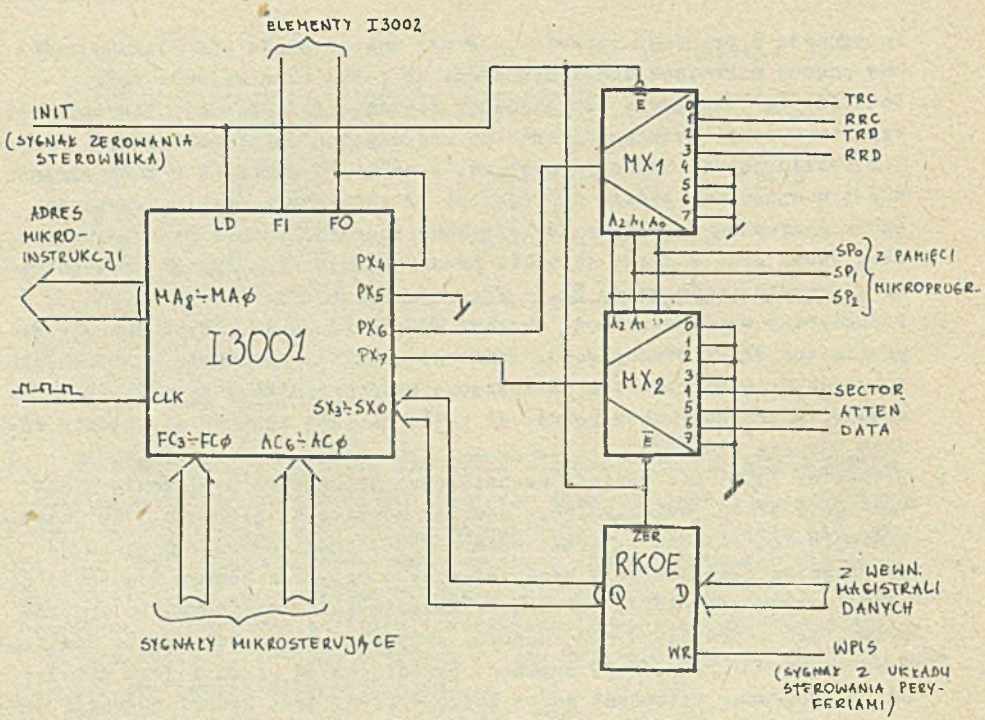
Rezygnacja z przerwań sprawiła, że nie wykorzystuje się blokady buforów adresu mikroinstrukcji sygnałami EN i ERA oraz wyjścia ISE.

4. Do wejść PX sekwentera doprowadzony zostanie kod operacji elementarnej wykorzystywany w mikroprogramie do rozgałęzień do poszczególnych procedur realizujących operacje dyskowe, a także do powrotów z podprogramów. Kod ten wpisywany będzie do rejestru PR sekwentera przy wykonywaniu skoku JPX, a adres będący funkcją kodu operacji będzie generowany podczas skoku JPR. Z uwagi na takie przeznaczenie rejestru PR zrezygnowano z wykorzystania wyjść PR_2 - PR_0.
5. Podprogramy wywoływane będą skokami JZR, gdyż są to jedyne funkcje zapewniające skoki bezwzględne. Ponieważ skoki JZR przekazują sterowanie do rzędu 0, w wierszu tym umieszczone będą początki wszystkich podprogramów. Mikroprogramowy powrót do pętli głównej odbywać się będzie również poprzez skok JZR.
6. Sekwenter I3001 nie posiada wewnętrznego mechanizmu pamiętania i odtwarzania śladu. Wykorzystanie pewnych możliwości elementu I3001 wyeliminowało konieczność wykorzystania układów dodatkowych. Po skoku do podprogramu ślad nie jest zapamiętywany w ogóle, a powrót następuje do mikroinstrukcji, której adres jest funkcją zarówno wykonanego podprogramu, jak i procedury wywołującej. Przyjęto, że powrót będzie realizowany skokiem JPR, którego argument określa adres rzędu i jest zależny od podprogramu, natomiast adres kolumny równy jest kodowi operacji dyskowej, w której następuje wywołanie. Kod ten przed wywołaniem podprogramu wpisywany będzie do rejestru PR sekwentera.
W przypadku konieczności powtórnego wywołania tego samego podprogramu w procedurze wywołującej nastąpi wpis do rejestru PR ciągu bitów różnego od kodu operacji dyskowej, ale zapewniającego właściwy powrót.

3.2. Połączenie sekwentera z układami współpracującymi

Rysunek 3.1. prezentuje rozwiązanie urządzeniowe połączenia sekwentera I3001 z układami współpracującymi. Rozwiązanie to jest sprzętową realizacją założeń omówionych w poprzednim punkcie.

Kod operacji elementarnej wpisywany jest do rejestru RKOE w pętli głównej mikroprogramu. Wyjścia RKOE podawane są bezpośrednio na wejścia SX sekwentera. Na wejścia PX_7 i PX_6 podawane są sygnały stanu sterownika poprzez multipleksery warunku MX1 i MX2 sterowane w mikroprogramie polem SP_2 - SP_0. Na wejście sekwentera LD, na wejścia blokujące multiplekserów i na wejście zerujące RKOE doprowadzono sygnał zerowania INIT. W czasie, gdy jest on aktywny, następuje przejście do mikroinstrukcji startowej.



Rys. 3.1. Połączenie sekwentera I3001 z elementami współpracującymi
Fig. 3.1. I3001 sequencer connection with co-operative devices

3.3. Adresowanie mikroinstrukcji

Przy syntezie mikroprogramu dla sterownika twardych dysków nie dysponowano mikroasemblerem dla INTELA 3000. Asembler taki nie został również przez projektantów sterownika napisany ze względu na ograniczenia czasowe przy realizacji tematu. Zostało jedynie stworzone proste oprogramowanie pomocnicze opisane w p. 4.2. Cały ciężar rozmieszczenia mikroprogramu w pamięci spadł zatem na jego autorów.

Przydzielanie adresów mikroinstrukcjom w pamięci stałej mimo mnogości skoków sekwentera, jest bardzo skomplikowane. Stosunkowo duży zestaw funkcji skoków pozornie stwarza szerokie możliwości adresowania. Jednakże szczegółowa analiza poszczególnych funkcji ujawnia szereg ograniczeń, które projektant musi pokonać. Wynikają one głównie z faktu, że z danej mikroinstrukcji można skoczyć jedynie pod pewne adresy ściśle uzależnione od wybranej funkcji skoku. Nie zawsze zatem istnieje skok, który pozwoliłby na przejście między dwiema dowolnie wybranymi komórkami pamięci mikroprogramów. Trudności te rosną wraz ze wzrostem obszaru pamięci zajmowanego przez mikroprogram. Cały mikroprogram sterownika twardych dysków składa się z ok. 380 mikroinstrukcji (ok. 74% obszaru pamięci), więc rozmieszczenie tego mikroprogramu w pamięci nie było zadaniem trywialnym.

Przy rozmieszczaniu mikroprogramu należy wziąć pod uwagę, że poszczególne funkcje skoku przekazują sterowanie do określonych obszarów, np. do wierszy lub kolumn o ściśle sprecyzowanych adresach. Przykładowo skoki warunkowe testujące stan flag (JFL, JCF, JZF) powodują przejście do jednej z kolumn o numerach: 2, 3, 10 lub 11. W mikroprogramie sterownika twardych dysków wystąpiło bardzo dużo skoków warunkowych. Konieczne było więc zarezerwowanie na wstępie komórek, do których przechodzi się w wyniku skoków warunkowych, aby komórek tych nie zająć przez przypadkowe mikroinstukcje, uniemożliwiając tym samym realizację wszystkich skoków warunkowych.

Wprowadzono zatem zasadę dedykowania pewnych pól pamięci określonym skokom, tj. rezerwowania obszarów dla potrzeb przekazania sterowania przy zastosowaniu określonej funkcji skoku. W pamięci mikroprogramów wyróżniono następujące pola dydykowane:

- kolumny 2, 3, 10 i 11 dla skoków JFL, JCF, JZF,
- " 0, 1, 4, 5, 8, 9 dla skoków JPX, testujących sygnały stanu sterownika,
- wiersz 0 jako wiersz początków podprogramów,
- " 3 " " " procedur realizujących operacje elementarne (wiersz ten został w zasadzie wybrany dowolnie).

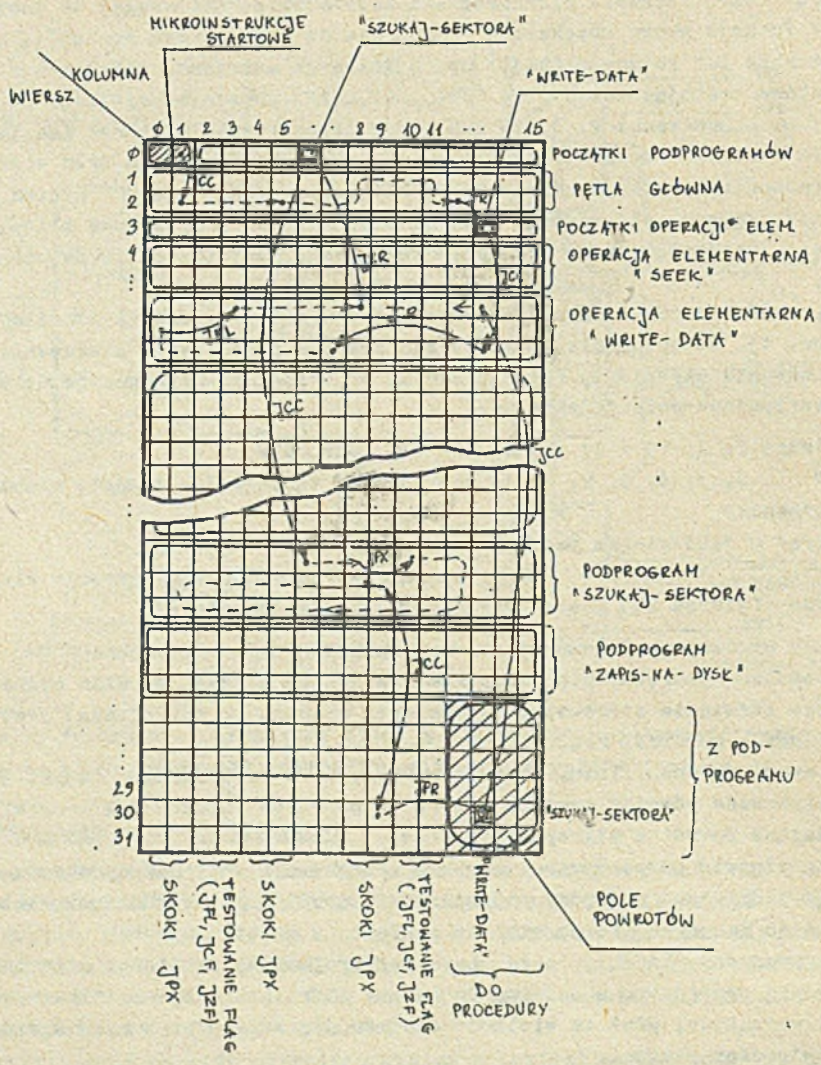
Ponadto wyróżniono 2 komórki startowe mikroprogramu (o adresach 0-0 i 0-1). Konieczność wyodrębnienia dwóch komórek wynika z tego, że stan sygnału FO podczas zerowania sterownika (wymuszenia sygnału LD sekwentera) jest nieokreślony - porównaj rys. 3.1.

Z uwagi na specyficzny mechanizm powrotów z podprogramów (punkt 3.1.6) zarezerwowane również zostało tzw. pole powrotów (blok 5 x 4 komórki) zapewniające powrót z mikropodprogramów.

Aby ułatwić adresowanie i uzyskać przejrzystą mapę mikroprogramu całą pamięć podzielono pomiędzy podprogramy i procedury operacji dyskowych przydzielając im pewną liczbę wierszy pamięci (z wyjątkiem dwóch wierszy zarezerwowanych). Liczba wierszy była proporcjonalna do liczby mikroinstrukcji danej procedury, a ściślej do liczby mikroinstrukcji zawierających skoki warunkowe, gdyż ta wielkość okazała się krytyczna z punktu widzenia zajętości pamięci.

Podział pamięci mikroprogramów na pola dedykowane przedstawia schematycznie rys. 3.2.

Po tak dokonanym podziale pamięci przystąpiono do przydziału adresów poszczególnym mikroinstrukcjom. Tym mikroinstrukcjom, do których przechodzi się skokami specjalnymi, przydzielono adresy zarezerowowane uprzednio. Natomiast w innych przypadkach wykorzystano skoki JCC i JCR oraz pozostałe wolne adresy z obszaru dedykowanego danej procedurze. Niemożliwe było ścisłe przestrzeganie zajmowania wyłącznie obszarów dedykowanych. Szczególnie poprawki wprowadzone na etapie uruchamiania mikroprogramu przyczyniły się do pewnych zmian w pierwotnie założonej strukturze.



Rys. 3.2. Mapa pamięci mikroprogramu sterownika twardych dysków
Fig. 3.2. A map of microprogram memory for the hard disc controller

Na zakończenie należy zaznaczyć, że z powodu trudności w bezpośrednim przejściu do żądanego adresu do mikroprogramu wprowadzono puste mikroinstrukcje, których zadaniem jest jedynie przekazanie sterowania (skoki).

Na rysunku 3.2 zilustrowano przykładowe skoki w pamięci mikroprogramów przy realizacji jednej z procedur.

4. Narzędzia wspomagające mikroprogramowanie

4.1. Języki mikroprogramowania

Od czasu wprowadzenia przez Wilkesa pojęcia mikroprogramowania powstało wiele narzędzi ułatwiających tworzenie i uruchamianie mikroprogramów. Przede wszystkim należą do nich języki mikroprogramowania i ich translatory. Głównymi przesłankami wprowadzenia tych języków były:

- chęć uwolnienia mikroprogramisty od konieczności pamiętania kodów binarnych i uwzględniania szczegółów konkretnej architektury urządzenia mikroprogramowanego,
- dążenie do minimalizacji liczby błędów w mikroprogramie,
- skrócenie czasu syntezy i uruchamiania mikroprogramu.

Podobnie jak wśród języków programowania, wyróżnić można języki mikroprogramowania na poziomie przesyłów międzyrejestrowych (mikroasemblery), języki wyższego rzędu i metajęzyki, pozwalające na zadeklarowanie struktury urządzenia, formatu mikroinstrukcji itp. Do najbardziej rozpowszechnionych języków mikroprogramowania wyższego rzędu należą: SIMPL, MPL, EMPL, S^x, MPGL, MPGS, Pumpkin, CHAMIL i in. [7, 8] . Języki te są na ogół maszynowo niezależne, pozwalają na pisanie strukturalnych mikroprogramów i operowanie na danych symbolicznych. Struktura sterująca tych języków oparta jest na strukturach języków: ALGOL, PASCAL, PL/I.

Powszechnie stosuje się również asemblery mikroprogramów. Umożliwiają one symboliczne wprowadzanie zawartości pól kodowych i symboliczną adresację. Zwykle jeden wiersz programu źródłowego odpowiada jednej mikroinstrukcji.

Coraz częściej translatory języków mikroprogramowania wyposaża się w moduły symulacji oraz kontroli i diagnostyki błędów. Przyspiesza to znacznie proces testowania i uruchamiania mikroprogramów.

Interesujące przykłady języków mikroprogramowania można znaleźć w [4, 5, 6].

4.2. Wspomaganie mikroprogramisty sterownika twardych dysków.

4.2.1. Programy usługowe

Tworzenie mikroprogramu dla sterownika twardych dysków było ułatwione dzięki pakietowi programów pomocniczych, które realizowały następujące funkcje:

- symboliczne wprowadzanie mikroprogramu,
- sortowanie źródłowego zbioru mikroinstrukcji wg adresów,
- symboliczny wydruk treści mikroprogramu,
- wydruk mapy zajętości pamięci mikroprogramu,
- generacja mikroprogramu binarnego.

Programy te napisane zostały w języku CBASIC na systemie RTDS-8. Wszystkie programy zapewniają wygodny, konwersacyjny tryb pracy. Wykorzystanie ich uwalnia projektanta mikroprogramu od konieczności pamiętania binarnych kodów mikroinstrukcji, pozwala na znaczne zaoszczędzenie czasu i uniknięcie wielu błędów. Ułatwia także uzyskanie dokumentacji mikroprogramu. Dodatkową zaletą jest umieszczenie mikroprogramu zarówno w postaci źródłowej, jak i wynikowej w pamięci na dyskach elastycznych. Binarny kod wynikowy, generowany przez oprogramowanie usługowe, reprezentuje zawartość pamięci mikroprogramów i stanowi podstawę do zaprogramowania pamięci PROM.

4.2.2. Symboliczne wprowadzanie mikroinstrukcji

Na kilka słów uwagi zasługuje program symbolicznego wprowadzania mikroinstrukcji, tworzący zbiór źródłowy mikroprogramu. Program nie jest w pełnym tego słowa znaczeniu asemblerem, gdyż cały ciężar adresacji mikroinstrukcji musi wziąć na siebie projektant. Użytkownik wprowadza w odpowiedzi na zapytania programu kolejne mikroinstukcje, podając ich adres i zawartość każdego pola w słowie mikroinstrukcji- por. format mikroinstrukcji w tablicy 2.1. Pola FC, SP, PA określane są symbolicznie. Maska M_7-M_0

Tablica 4.1

Przykład mikroinstrukcji w postaci symbolicznej

Oznaczenie pola mikroinstrukcji	Przykładowa postać symboliczna
M	FD
F_6-F_0	6,1,R5
FC_3-FC_2	HCZ
FC_1-FC_0	FFO
AC	JFX,1
SP	DAT
PA	TRACK
WR/RD	0
BLOK	0
DAT/CRC	0
WG	1
RG	1
STROBE	1
RESTORE	1

podawana jest w formie liczby szesnastkowej, natomiast pola jednobitowe (BLOK, WR/RD, DAT/CRC, WG, RG, RESTORE, STROBE) definiuje się za pomocą cyfry binarnej (0 lub 1). W polu adresowym AC należy podać symboliczną nazwę skoku wraz z jego argumentem. Wyjątek stanowi pole F_6-F_0 określające przetwarzanie elementów I3002, które rozbite jest na 3 podpola zgodnie z opisem katalogowym [2]. W podpolach definiuje się liczbowe kody grupy F i R oraz nazwę jednego z rejestrów elementu.

W tablicy 4.1. zamieszczono przykładową mikroinstrukcję w postaci akceptowanej przez program wprowadzania symbolicznego.
Dodatkową funkcją opisanego programu jest sprawdzenie, czy komórka o podanym przez wprowadzającego adresie nie została już wcześniej zajęta.

4.3. Uruchamianie mikroprogramu

W trakcie uruchamiania mikroprogramu sterownika twardych dysków zastosowano następujące podstawowe środki:

- zastąpiono docelowe pamięci PROM szybkimi pamięciami EPROM, których zawartość mogła być wielokrotnie zmieniana,
- do sterownika dołączono układ pracy krokowej oraz zatrzymania na zadanym adresie mikroinstrukcji,
- całe słowo mikroinstrukcji wyprowadzono na dodatkową łączówkę, w której podłączono pulpit uruchomieniowy pozwalający na:
  a) obserwację stanu sygnałów mikrosterujących na diodach elektroluminescencyjnych,
  b) wymuszanie pewnych sygnałów (po wyjęciu pamięci mikroprogramów z podstawki).

Po uruchomieniu mikroprogramu przy zastosowaniu wyżej wyszczególnionych narzędzi odłączono od sterownika układ pracy krokowej, a mikroprogram "zaszyto" w pamięciach stałych PROM.

5. Podsumowanie

5.1. Propozycja rozszerzenia możliwości wspomagania mikroprogramisty

Oprogramowanie wspomagające pracę projektanta sterownika twardych dysków można udoskonalić uzyskując dalsze zmniejszenie pracochłonności i czasochłonności mikroprogramowania.

Pierwszą propozycją modyfikacji jest umożliwienie wprowadzania symbolicznego wszystkich pól mikroinstrukcji, a zwłaszcza pola sterowania przetwarzaniem procesora segmentowego I3002. W tym celu można posłużyć się katalogowymi nazwami symbolicznymi mikroinstrukcji tego elementu oraz symbolicznymi nazwami rejestrów wewnętrznych nadanymi przez projektanta. Dodatkowym ułatwieniem dla mikroprogramisty byłaby rezygnacja z konieczności wprowadzania pól w przypadku, gdy ich stan określa operację pustą ("no operation").

Następnym poważnym krokiem byłoby powierzenie oprogramowaniu wspomagającemu zadania automatycznej adresacji mikroinstrukcji. Jednakże z uwagi na wiele ograniczeń przy wyznaczaniu adresu następnej mikroinstrukcji (p. 1.4) nie byłoby to zadanie łatwe.

Po dokonaniu powyższych rozszerzeń otrzymalibyśmy klasyczny mikroasembler. Kolejnym etapem modyfikacji może być próba zuniwersalizowania takiego asemblera w celu uniezależnienia się od konkretnej struktury sprzętowej. W tym przypadku projektant deklaruje format mikroinstrukcji, określa nazwy symboliczne i przyporządkowane im wartości binarne. Definiuje zatem syntaktykę i semantykę języka mikroprogramowania. Program generujący mikroprogram binarny według reguł podanych przez użytkownika można określić mianem metaasemblera mikroprogramu.

Oprogramowanie wspomagające warto również wzbogacić o symulator pozwalający na śledzenie toku wykonywania mikroprogramu. Należy ponadto zauważyć, że bardzo dobrym narzędziem uruchamiania zarówno urządzenia, jak i sterującego nim mikroprogramu byłby układowy emulator INTELa 3000. Zagadnienia emulacji zostały szczegółowo przedstawione w [9].

5.2. Uwagi końcowe

Zastosowanie elementów mikroprocesorowych rodziny I3000 do budowy urządzeń cyfrowych niesie wiele korzyści (p. 1.1). Jednakże wykorzystanie tej rodziny ma również pewne wady. Jedną z nielicznych, choć podstawowych jest dość skomplikowane mikroprogramowanie, zwłaszcza przy braku odpowiednich narzędzi wspomagających. Szczególne trudności stwarza adresowanie poszczególnych mikroinstrukcji. Sekwenter I3001 nie przewiduje mechanizmu skoków do/powrotów z podprogramów. Aby to osiągnąć trzeba angażować dodatkowe układy logiczne i stosować dość wyszukane techniki. Poza tym wydaje się, że długość słowa elementu przetwarzania I3002 (2 bity) jest zbyt krótka dla typowych zastosowań - trzeba zatem wykorzystać wiele takich elementów. Być może powyższe czynniki wpłynęły na stopniowe wycofywanie się firm zachodnich z produkcji mikroprocesorów rodziny I3000 (względnie ich odpowiedników). Konstruktorzy wykorzystują powszechnie inne serie mikroprocesorów bipolarnych, np. AMD 2900. Rodzina ta, łącząc zalety segmentowych mikroprocesorów bipolarnych, posiada tę przewagę nad serią I3000, że adresowanie mikroinstrukcji jest znacznie prostsze (sekwencyjne). Przewidziano też możliwość wykorzystania mikroprogramów, które nawet mogą być zagnieżdżane. Ponadto element przetwarzający wykonuje operacje na słowach 4-bitowych.

Bardzo często ostateczny kształt wielu urządzeń powstałych w naszym kraju jest wynikiem kompromisu pomiędzy zamierzeniami projektanta a jego możliwościami (np. dostępną bazą elementową). Typowym przykładem takiej sytuacji jest wykorzystywanie do konstrukcji urządzeń cyfrowych układów rodziny I3000, dostępnej w Polsce. Wydaje się, że w najbliższym czasie elementy tej rodziny będą szeroko stosowane w technice cyfrowej. Dużym jednak optymizmem napawa fakt coraz większej dostępności w kraju innych

serii mikroprocesorów segmentowych, chociażby wspomnianej serii AMD 2900 produkowanej w krajach RWPG.

LITERATURA

[1] Rzymkowski K.: "Mikroprocesory" ROINTE Energetyki i Energii Atomowej, Warszawa 1979.

[2] Series 3000 Bipolar Microcomputer System. Katalog firmy INTEL, 1976.

[3] Nowak H., Stróżyna P.: Realizacja sterownika twardych dysków z wykorzystaniem mikroprocesorów segmentowych, Zeszyty Naukowe Politechniki Śląskiej, s. Informatyka, z. 8, Gliwice 1986.

[4] Firth N. R.: The Role of Software Tools in the Developement of the ECLIPSE MV/8000 Microcode, "Sigmicro Newsletter" vol. 11 nr 3 i 4, 1980.

[5] Ballieu G., Lewi J., Willems Y.D.: A Microprogramming Language at Register Transfer Level, "Microprocessing and Microprogramming" vol. 8 nr 3, 4, 5, 1981.

[6] Skordalakis E.: Towards a More Flexible Microlanguage for Bit-Sliced Microcomputers, "Microprocessing and Microprogramming" vol.7 nr 1, 1981.

[7] Sint M.: A Survey of High Level Languages for Microprogramming, "Sigmicro Newsletter" vol. 11 nr. 3, 4, 1980.

[8] Bushell R.G.: Higher Level Languages for Microprogramming, "Euromicro Journal" vol. Nr 2, 1978.

Recenzent: dr inż. Jerzy Jaworowski

Wpłynęło do Redakcji 14.06.1985

МИКРОПРОГРАММИРОВАНИЕ МИКРОПОЦЕССОРОВ INTEL 3000 НА ПРИМЕРЕ КОНТРОЛЛЕРА МАГНИТНЫХ ДИСКОВ

Р е з ю м е

В статье дана краткая характеристика микропорцессорного набора INTEL 3000 с обращением внимания на микропрограммное устройство управления После эскиза концепции использования набора I3000 для конструкции контроллера магнитных дисков, представлена структура микропрограммы и формат микрокоманд. Рассматриваются также проблемы адрессации микрокоманд. Даны некоторые практические соображения и проектные советы. В заключительной части работы описаны средства разработки и совершенствования микропрограммы для контроллера твёрдых дисков.

INTEL 3000 FAMILY OF BIT-SLICED MICROPROCESSORS MICROPROGRAMMING ON THE BASIS OF THE HARD DISC CONTROLLER

S u m m a r y

A brief characterization of INTEL 3000 bipolar bit-sliced microprocessor family was presented in the paper. Particular attention was paid to microprogram control unit. After having outlined the idea of making use of the 3000 family to construct a hard disc controller, a structure of microprogram and a format of microinstruction were described. Problems of microinstruction addressing were discussed in detail. Some practical advice of designing as well as remarks were stated then. Finally, development methods for the microprogram of the hard disc controller were shown.