ZESZYTY NAUKOWE POLITECHNIKI ŚLĄSKIEJ 1983

Seria: INFORMATYKA z. 5 Nr kol. 759

Adam GACEK
Stanisław KOZIELSKI
Piotr STRÓŻYNA

MODELOWANIE LISTY ROZKAZÓW DLA POTRZEB KOMUNIKACJI Z OTOCZENIEM PRZY WYKORZYSTANIU KOMPUTERA DYDAKTYCZNEGO EW

Streszczenie. W pracy przedstawiono analizę doboru rozkazów dla przyjętej struktury systemu przerwań i systemu wejścia-wyjścia dydaktycznego komputera EW. Rozpatrzono różne możliwości organizacji cyklu rozkazowego z obsługą przerwania oraz różne sposoby realizacji wymiany informacji. Dla badanych wariantów funkcjonalnych podano ich realizację mikroprogramową.

1. Wstęp

Projektowanie listy rozkazów dla komputera mikroprogramowanego stanowi w swej istocie zadanie optymalizacyjne. Chodzi przy tym zarówno o określenie minimalnego zbioru rozkazów umożliwiających zrealizowanie w komputerze pożądanych operacji, jak i o ustalenie optymalnego sposobu realizacji poszczególnych rozkazów. W komputerze mikroprogramowanym istnieje zwykle możliwość zrealizowania większości rozkazów kilkoma sposobami w zależności od przyporządkowanych im sekwencji mikroprogramowych ze zbioru mikrorozkazów niedostępnych dla programisty.

Sekwencje te określają użytkową listę rozkazów komputera. Z kolei wiadomo, że w systemach operacyjnych pewne sekwencje rozkazów użytkowych mogę tworzyć tzw. makrorozkazy realizujące bardziej złożone operacje. Można przyjąć, że w komputerze mikroprogramowanym występują trzy poziomy programowania, a mianowicie:

- poziom mikrorozkazowy,
- poziom rozkazów użytkowych,
- poziom makrorozkazowy.

Ustalenie odpowiedniego podziału funkcji rozkazowych pomiędzy te trzy poziomy programowanie stanowi istotne zadanie w procesie projektowania listy rozkazów. Rozwiązanie tego zadania decyduje o elastyczności programowania komputera, czyli jego podatności do prostej i łatwej modyfikacji oprogramowania użytkowego, a także określa przydatność komputera do wybranych zastosowań. Zauważmy, że w myśl powyższych określeń najwyższą elastyczność programowania posiada komputer, którego lista rozkazów użytkowych odpowiada liście mikrorozkazów. Jednak taki komputer jest w is-

tocie mało użyteczny, gdyż wymaga zaprogramowania przez programistę każdej elementarnej operacji, tj. np. pobrania argumentu z pamięci operacyjnej, zwiększenia stanu licznika rozkazów o jeden itp. przez co jego programowanie jest bardzo złożone, a każdy program użytkowy jest bardzo rozbudowany. Przeciwnie, ekstremalnie najmniejszą elastyczność programowania posiada komputer specjalizowany realizujący zadanie według tzw. programu stałego. W tym przypadku modyfikacja programu wymaga wymiany całego programu stałego. Komputer taki jest jednak w swej klasie zastosowań bardzo użyteczny, gdyż raz zaprogramowany nie wymaga już żadnego nadzoru ze strony programisty.

Uwzględniając powyższe uwagi można stwierdzić, że zagadnienie projektowania listy rozkazów dla komputera o określonym przeznaczeniu sprowadza się w rezultacie do ustalenia optymalnego podziału funkcji realizowanych w komputerze w sposób układowy i programowy. Znaczną część problemów związanych z projektowaniem listy rozkazów można rozwiązać, stosując metody symulacyjne obejmujące modelowanie programowe i układowe. Przy zastosowaniu symulacji programowej projektowana lista rozkazów jest modelowana w komputerze o bardziej rozbudowanej liście rozkazów i dużych możliwościach programowych. W ten sposób jednak możliwe jest stosunkowo proste zamodelowanie listy rozkazów w zakresie funkcjonalnym, natomiast znacznie utrudnione jest zamodelowanie sposobu realizacji poszczególnych rozkazów na poziomie mikrorozkazowym.

Symulacja układowa polega na utworzeniu układowego modelu całego komputera, dla którego projektowana jest lista rozkazów lub wydzielonych jego bloków funkcjonalnych. Ten rodzaj symulacji można zrealizować za pomocą swobodnie mikroprogramowanego komputera modelującego o rozbudowanej i uniwersalnej strukturze logicznej. W tym przypadku w wyniku badań symulacyjnych fragment struktury komputera modelującego staje się wzorcem strukturalnym komputera, dla którego projektowana jest lista rozkazów.

Praca niniejsza dotyczy zastosowania symulacji układowej do projektowania listy rozkazów. W tym przypadku jako komputer modelujący został zastosowany komputer dydaktyczny EW opracowany w IICR specjalnie dla celów laboratoryjnego modelowania list rozkazowych. Sposób wykorzystania komputera modelującego w procesie projektowania listy rozkazów przedstawiono w pracy na przykładzie rozkazów komunikacji komputera z otoczeniem.

Komputer EW jest komputerem swobodnie mikroprogramowanym, ale ze względu na swoje dydaktyczne zastosowanie posiada uproszczoną strukturę [3], która nie umożliwia zamodelowania całych list rozkazowych, a jedynie pewne grupy rozkazowe. Jest to jednak zupełnie wystarczające, by w celach dydaktycznych zobrazować główne problemy związane z projektowaniem listy rozkazów.

2. System przerwań komputera dydaktycznego EW

Prezentowany system przerwań jest systemem czteropoziomowym. Jego strukturę przedstawia rys. 1.

Zgłoszenia przerwań (z klawiatury funkcyjnej oraz systemu wejścia-wyjścia) przyjmowane są do rejestru zgłoszeń. Wyjścia z tego rejestru mogą być maskowane zależnie od zawartości rejestru maski. Przy zgłoszeniach na kilku poziomach wybór najstarszego poziomu następuje w układzie hierarchii. Zgłoszenie wybrane powoduje generację sygnału przerwania INT, będącego sygnałem stanu wykorzystanym w układzie sterowania. Rozpoznanie przerwania i przejścia do obsługi przerwania w cyklu rozkazowym potwierdzane jest sygnałem "eni" wpisującym przyjęte do obsługi zgłoszenie do rejestru przerwań. Zapalenie odpowiedniego przerzutnika w tym rejestrze przerywa przy tym - w układzie samoblokady - możliwość dalszej generacji sygnału INT z danego poziomu. W rejestrze przerwań mogą się znaleźć równocześnie zgłoszenia z kilku poziomów, pod warunkiem, że zgłoszenia te nadchodziły w kolejności wzrostu ich priorytetów. Taka zawartość rejestru przerwań odpowiada sytuacji, kiedy została rozpoczęta, a potem zawieszona, realizacja kilku programów obsługi przerwań (dla odpowiednich poziomów), przy czym program aktualnie realizowany jest programem obsługi najstarszego spośród zgłoszonych poziomów.

Zerowanie przerwań powinno nastąpić przy takim działaniu systemu, po zakończeniu programu obsługi przerwania. Sygnał zarujący "rint" spowoduje wtedy wyzerowania rejestru zgłoszeń i rejestru przerwań na pozycji wybranej przez układ hierarchii zgłoszeń przyjętych do obsługi, a więc odpowiadającej poziomowi, którego programowa obsługa została właśnie zakończona.

Praca z aktywnym systemem przerwań o przedstawionej strukturze wymaga uwzględnienia w trakcie projektowania listy rozkazów następujących aspektów, wpływających na technikę tworzenia oprogramowania podstawowego komputera:

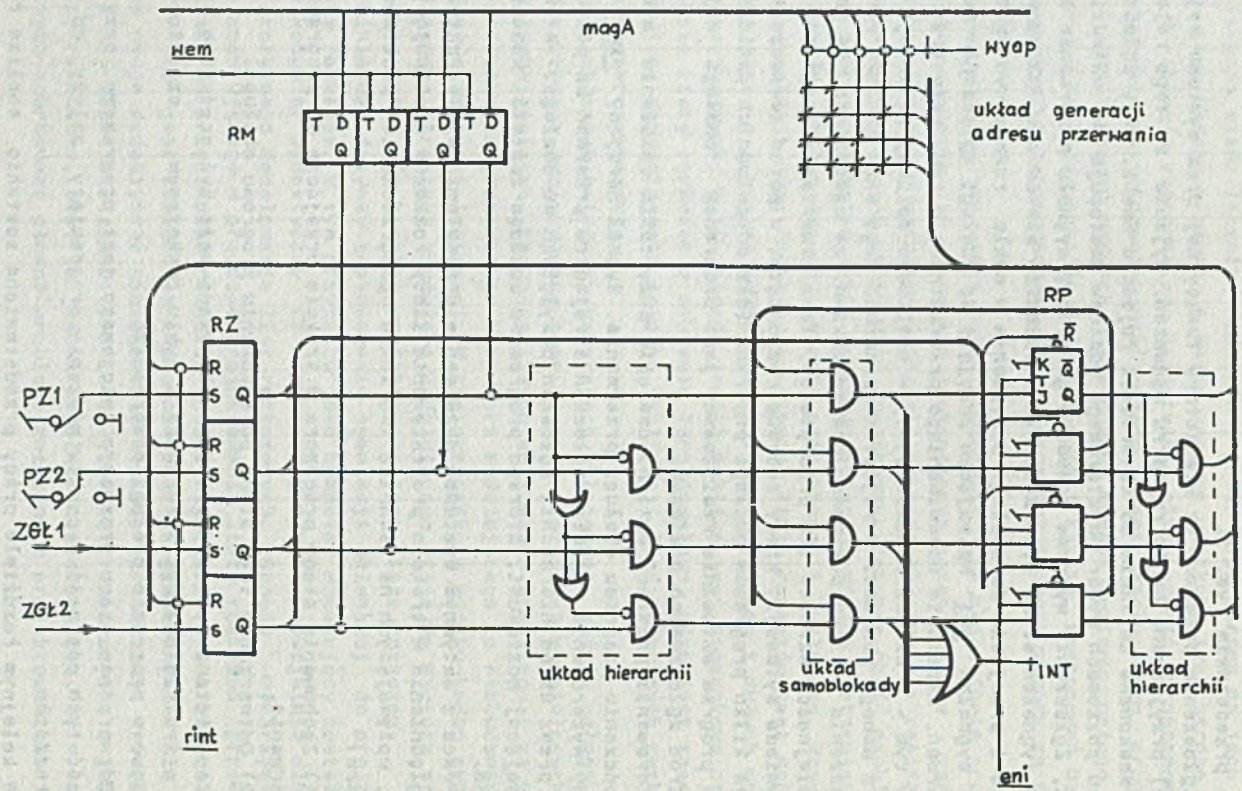
1) Zachowanie stanu procesora w trakcie przejścia do programu obsługi przerwania.

2) Odtwarzanie tego stanu po wykonaniu programu obsługi.

Zapamiętanie stanu procesora wykonywane może być częściowo automatycznie w trakcie obsługi przerwania w cyklu rozkazowym, częściowo zaś programowo w programie obsługi przerwania.

Odtwarzanie stanu procesora inicjowane jest programowo, przy czym złożoność tych operacji z punktu widzenia programisty zależy od dostępnych mu rozkazów.

W kolejnym rozdziale pracy przedstawiona zostanie analiza różnych wariantów chowania i odtwarzania stanu procesora w komputerze EW.



Rys. 1. Struktura systemu przerwań

3. Chowanie i odtwarzanie etanu procesora w przypadku przerwania; organizacja cyklu rozkazowego z uwzględnieniem obsługi przerwania

Jak wspomniano poprzednio, układ sterujący komputera informowany jest o wystąpieniu zgłoszenia w systemie przerwań za pośrednictwem sygnału INT. Podejmowana w takim przypadku obsługa przerwania w cyklu rozkazowym musi zapewnić przejście do realizacji odpowiedniego programu obsługi z ewentualnym zapamiętaniem etanu procesora w chwili przerwania. Stan procesora (po zakończeniu realizacji kolejnego rozkazu) określony jest zawartością licznika rozkazów L, rejestru akumulatora AK oraz rejestru roboczego X.

Rozważymy 3 typowe dla różnych komputerów warianty obsługi przerwania w cyklu rozkazowym.

a) Przejście do programu obsługi przerwania następuje bez jakiegokolwiek chowania rejestrów procesora.
b) W cyklu obsługi przerwania chowana jest zawartość licznika rozkazów, natomiast zawartości pozostałych rejestrów chowane są na początku programu obsługi przerwania.
c) W cyklu obsługi przerwania chowane są automatycznie zawartości wszystkich rejestrów określających stan procesora.

Zauważmy, że kolejne warianty pozwalają uprościć program obsługi przerwania, wymagają natomiast większego czasu na realizację, a także liczby zajmowanych komórek pamięci (czasami niepotrzebnie).

Stosownie do przedstawionych wariantów obsługi przerwania rozważymy różne sposoby wyjścia z programu obsługi przerwania. Obok skoku bezwarunkowego zastosować tu można rozkaz skoku według śladu wraz z dodatkową możliwością odtwarzania rejestrów arytmetycznych.

Wszystkie te trzy przedstawione warianty obsługi przerwania mogą zostać łatwo zamodelowane w maszynie cyfrowej EW.

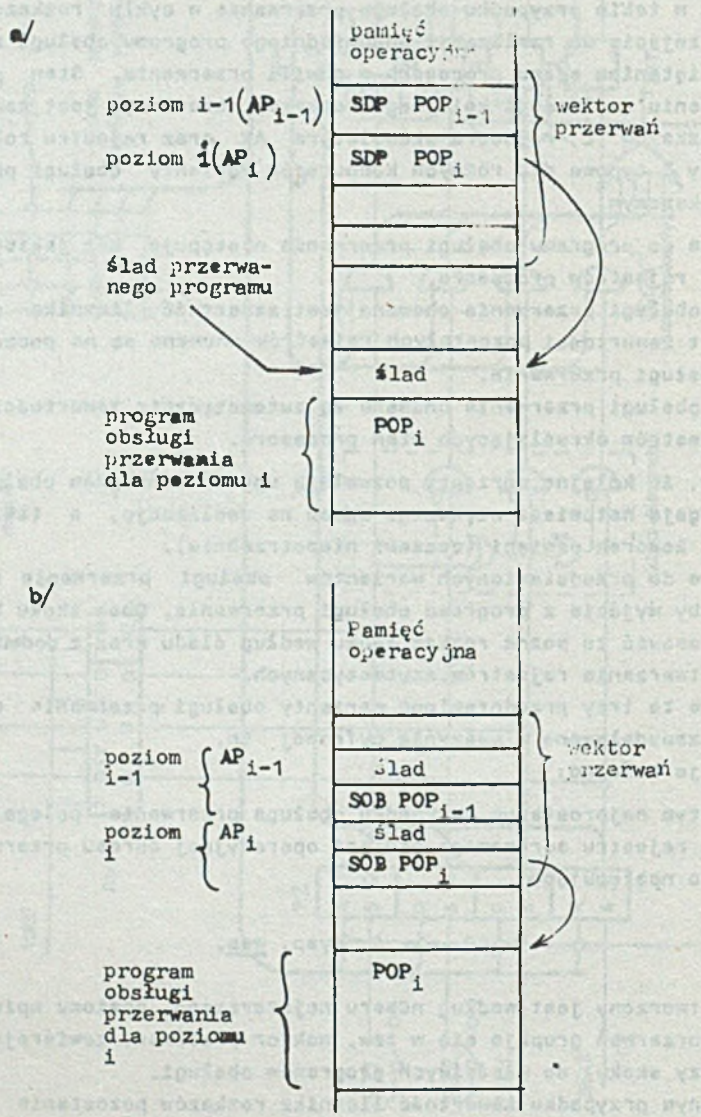
Omówimy je kolejno:

ad a) W tym najprostszym przypadku obsługa przerwania polega na wprowadzeniu do rejestru adresowego pamięci operacyjnej adresu przerwania AP. Zapiszemy to następująco:

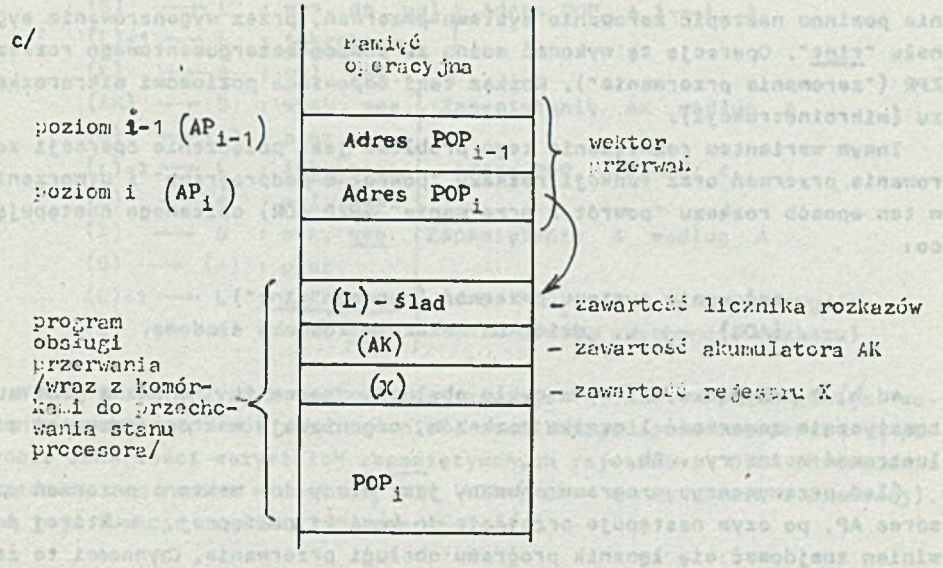
AP ⟵ A: wyap, wea.

Adres AP tworzony jest według numeru najstarszego poziomu wpisanego do RP. Adresy przerwań grupuje się w tzw. wektor przerwań, zawierający łączniki (rozkazy skoku) do właściwych programów obsługi.

W omawianym przypadku zawartość licznika rozkazów pozostanie (w trakcie obsługi przerwania w cyklu rozkazowym) nienaruszona. Programista może spowodować zapamiętanie zawartości licznika (śladu przerywanego programu), umieszczając w komórce łącznika rozkaz skoku do podprogramu (rys. 2a).



Rys. 2a,b. Organizacja wektora przerwań dla różnych przypadków obsługi przerwania w cyklu rozkazowym



Rys. 2c. Organizacja wektora przerwań dla różnych przypadków obsługi przerwania w cyklu rozkazowym

Przyjmiemy, że rozkaz skoku do podprogramu (SDP ADR) spowoduje zapamiętanie śladu w pierwszej komórce podprogramu, co zapiszemy następująco:

$$(L) + 1 \rightarrow (ADR)$$

$$ADR + 1 \rightarrow L$$

gdzie:

ADR - argument rozkazu (zawartość części adresowej rejestru rozkazów ADR = (AD)),

L - licznik rozkazów.

Zapamiętanie, a potem odtworzenie pozostałych rejestrów arytmetycznych zapewnić musi programista w programie obsługi przerwania. Oczywiście dla konkretnego programu obsługi przerwań czynności te (jak również chowanie śladu) mogę się okazać niepotrzebne, co pozwoli zaoszczędzić czas i pamięć komputera.

W przypadku zapamiętania śladu powrót z programu obsługi przerwania może nastąpić za pomocą rozkazu powrotu z podprogramu lub skoku bezwarunkowego adresowanego pośrednio (przez komórkę śladową).

Funkcję realizowaną przez rozkaz powrotu z podprogramu (PWR ADR) określamy następująco:

$$(ADR) \rightarrow L.$$

Jak już wspomniano poprzednio na zakończenie programu obsługi przerwania powinno nastąpić zerowanie systemu przerwań, przez wygenerowanie sygnału "rint". Operację tę wykonać można za pomocą bezargumentowego rozkazu ZPR ("zerowanie przerwania"). Rozkaz taki odpowiada poziomowi mikrorozkazu (mikroinstrukcji).

Innym wariantem rozwiązania tego problemu jest połączenie operacji zerowania przerwań oraz funkcji rozkazu "powrót z podprogramu" i utworzenie w ten sposób rozkazu "powrót z przerwania" (PZP ADR) opisanego następująco:

. zerowanie systemu przerwań (sygnał "rint"),
. (ADR) ⟶ L, gdzie ADR wskazuje komórkę śladową.

ad b) W przypadku, kiedy w cyklu obsługi przerwania chowana jest automatycznie zawartość licznika rozkazów, organizację wektora przerwań zilustrować można rys. 2b.

Ślad przerywanego programu chowany jest wtedy do wektora przerwań pod adres AP, po czym następuje przejście do komórki następnej, w której powinien znajdować się łącznik programu obsługi przerwania. Czynności te zapiszemy w następujący sposób:

```
AP⟶A      : wyap, wea
(L)⟶S     : wyl, as, wea,
(S)⟶(A)   : pisz,
AP⟶L      : wyap, wel.
(L) + 1⟶L : inkrement
```

Zapamiętanie i odtworzenie pozostałych rejestrów określających stan procesora zapewnić musi programista w programie obsługi przerwania. Powrót z przerwania wykonać można identycznie jak w przypadku poprzednio omówionym.

ad c) W tym wariancie założymy, że w cyklu obsługi przerwania automatycznie zapamiętywane jest zawartość wszystkich rejestrów określających stan procesora. Jedno z możliwych rozwiązań takiej operacji przedstawiono na rys. 2c.

Wektor przerwań zawiera w takim przypadku jedynie adresy programów obsługi. Do zarezerwowanych początkowych komórek tych programów chowana jest zawartość licznika rozkazów, akumulatora i rejestru X.

Operacje te zapiszemy następująco:

```
AP   ⟶ A : wyap, wea,      ┐
((A))⟶ S : czytaj           ├ Adres POP_i ⟶ A
(S)  ⟶ A : wya, as, wea,    ┘
(L)  ⟶ S : wyl, as, wes,    ┐ Zapamiętanie śladu według A
(S)  ⟶ (A) : pisz           ┘
```

```
AP    ──→ A  : wyap, wea,
((A)) ──→ S  : czytaj
(S)   ──→ L  : wye, ea, wal   } Adres POP_i + 1 ──→ L, A
(L)+1 ──→ L  : inkrement
(L)   ──→ A  : wyl, wea,
(AK)  ──→ S  : wyak, wea   } Zapamiętanie AK według A
(S)   ──→ (A) : piez
(L)+1 ──→ L  : inkrement   } Adres POP_i + 2 ──→ L, A
(L)   ──→ A  : wyl, wea
(X)   ──→ S  : wyx, wea    } Zapamiętanie X według A
(S)   ──→ (A) : piez
(L)+1 ──→ L  : inkrement   } Adres POP_i + 3 ──→ L, A (przygoto-
(L)   ──→ A  : wyl, wea    } wanie adresu następnego rozkazu)
```

W omawianym przypadku celowe jest zaprojektowanie rozszerzonego wariantu rozkazu "powrót z przerwania", umożliwiającego odtworzenie przy powrocie zawartości wszystkich zapamiętywanych rejestrów. Dla odróżnienia zapiszemy ten rozkaz w postaci POW ADR (ADR - adres komórki śladowej), zaś jego funkcje zdefiniujemy następująco:

- zerowanie systemu przerwań (sygnał "rint"),
- (ADR) ──→ L (odtworzenie śladu),
- (ADR+1) ──→ AK (odtworzenie zawartości akumulatora),
- (ADR+2) ──→ X (odtworzenie zawartości rejestru X).

Dla prawidłowej obsługi przerwania konieczne jest w niektórych przypadkach maskowanie przerwań. Operację taką realizujemy za pomocą rozkazu MAS arg, którego część adresowa zawiera bity maski, przesyłane do rejestru maski:

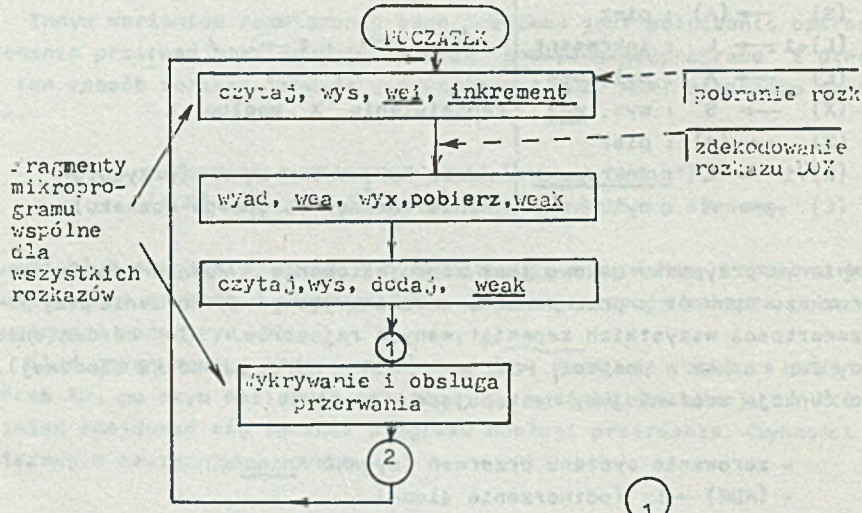
(AD) ──→ RM : wyad, wem

Włączenie obsługi przerwania do cyklu rozkazowego

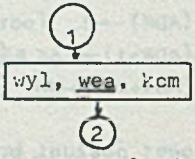
Przedstawione poprzednio warianty obsługi przerwania należy włączyć do całego cyklu rozkazowego. Problem ten rozważymy na przykładzie rozkazu DOX ADR (dodawania zawartości wskazanej komórki pamięci do rejestru X). Organizację przesłań międzyrejestrowych dla tego rozkazu przedstawiono w pracy [3].

Obsługa przerwania wykonywana jest zazwyczaj po zakończeniu wszystkich operacji związanych z danym rozkazem i stwierdzeniu aktywnego stanu sygnału INT. Czynnością rozpoczynającą obsługę przerwania musi być zapisanie przerwania sygnałem "oni" do rejestru przerwań RP.

Organizację cyklu rozkazowego z uwzględnieniem różnych wariantów obsługi przerwania przedstawiono na rys. 3.

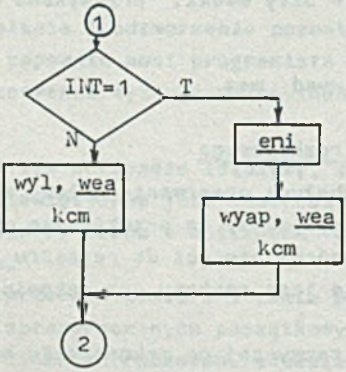


1° Postać rozkazu nieprzerywalnego:

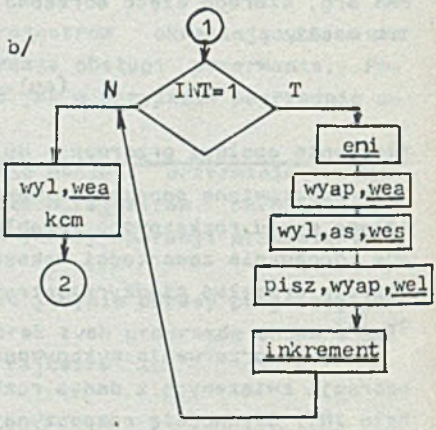


2° Trzy warianty obsługi przerwania dla przerywalnej postaci rozkazów:

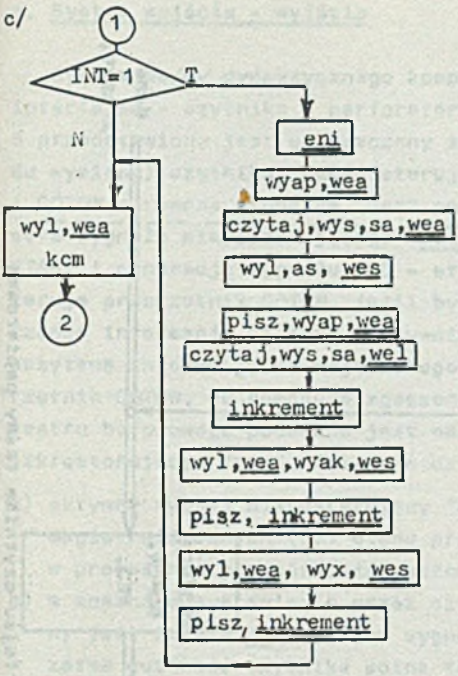
a/



b/



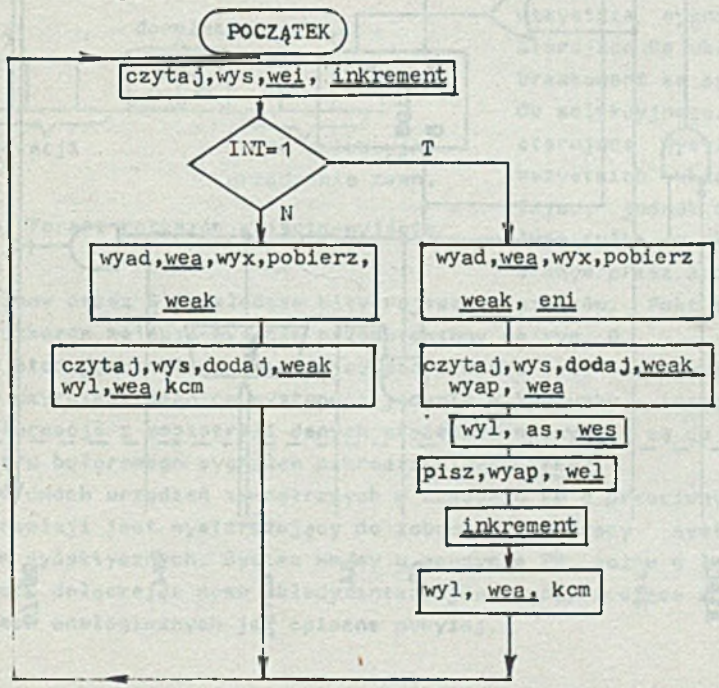
Rys. 3a,b. Organizacja cyklu rozkazowego z obsługą przerwania



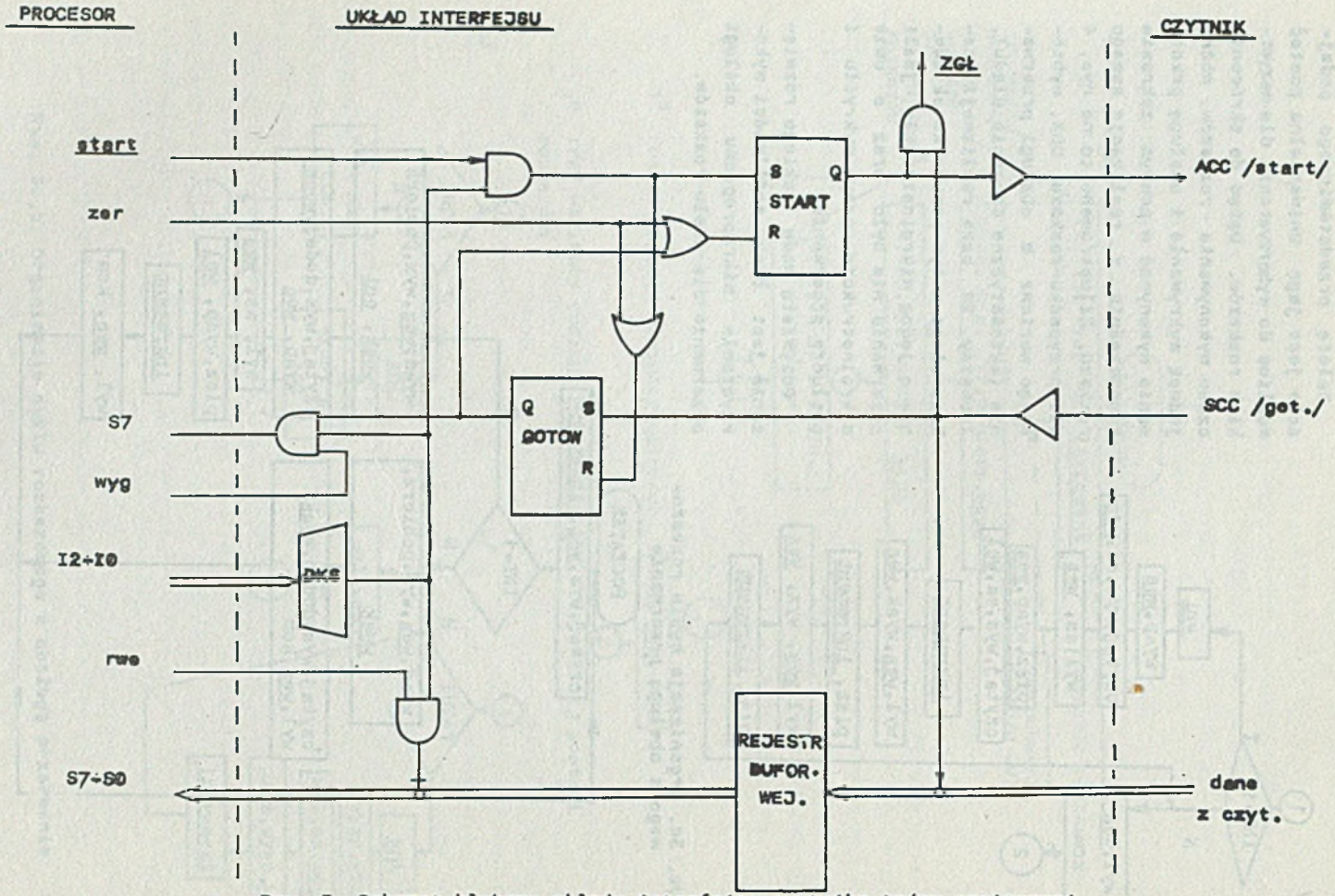
Rys. 3c. Organizacja cyklu rozkazowego z obsługą przerwania

Zaletę przedstawionego podejścia jest jego uniwersalna postać możliwa do wykorzystania dla wszystkich rozkazów. Dążąc do skrócenia czasu wykonywania rozkazów, można jednak wykrywanie i obsługę przerwania wykonywać w pewnym zakresie równocześnie z realizacją samego rozkazu. Zilustrowano to na rys. 4 dla przypadku rozkazu DOX, wybierając wariant b obsługi przerwania (automatyczne chowanie śladu). Zauważmy, że czas realizacji rozkazu ulega w tym przypadku skróceniu o jedną mikroinstrukcję, jeśli przerwania nie było oraz o dwie mikroinstrukcje przy wykryciu i obsłudze przerwania.

Oczywistą wadą takiego rozwiązania jest brak możliwości wykorzystania mikroprogramu obsługi przerwania dla wielu rozkazów.



Rys. 4. Równoległa realizacja rozkazu i obsługi przerwania

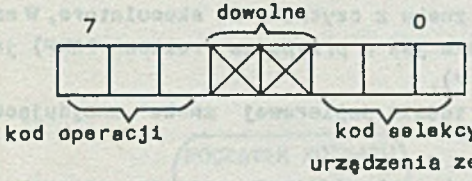


Rys. 5. Schemat blokowy układu interfejsu czytnika taśmy papierowej

4. System wejścia - wyjścia

System we/wy dydaktycznego komputera EW obejmuje jedynie dwa układy interfejsu - czytnika i perforatora (dziurkarki) taśmy papierowej. Na rys. 5 przedstawiony jest uproszczony schemat blokowy układu interfejsu (obwodu wymiany) czytnika. Część sterującą układu stanowią przerzutniki START i GOTOW zerowane sygnałem "zer" po włączeniu zasilania. Wysłanie z procesora sygnału mikrosterującego "start" powoduje zapalenie się przerzutnika START i generację sygnału ACC - startu dla czytnika. Ponadto sygnał "start" zeruje przerzutnik GOTOW, jeśli był on zapalony. Po wczytaniu jednego rzędka informacji czytnik uaktywnia sygnał gotowości SCC, który wpisuje wczytaną informację do wejściowego rejestru buforowego i ustawia przerzutnik GOTOW, co powoduje zgaszenie przerzutnika START. Informacja z rejestru buforowego podawana jest na magistralę danych procesora sygnałem mikrosterującym "rwe". Gotowość czytnika może być testowana dwojako:

a) aktywny sygnał mikrosterujący "wyg" powoduje podanie na najstarszy bit magistrali danych (S7) stanu przerzutnika GOTOW; stan bitu S7 może być w procesorze bezpośrednio testowany,
b) w momencie wystawienia przez czytnik sygnału gotowości (SCC) generowany jest impuls ZGŁ, będący sygnałem zgłoszenia dla układu przerwań; zatem gotowość czytnika można testować pośrednio przez przerwanie.



Rys. 6. Format rozkazów wejścia-wyjścia

Należy zauważyć, że wszystkie sygnały mikrosterujące dla układów we/wy bramkowane są sygnałem kodu selekcyjnego. Sygnały sterujące wysyłane są do wszystkich układów interfejsu, jednak efekt wywołują tylko w układzie wybranym przez kod selekcyjny określony przez 3 najmłodsze bity rejestru rozkazów. Fakt ten narzuca format rozkazów wejścia-wyjścia przedstawiony na rys. 6.

Część sterująca układu interfejsu perforatora jest identyczna jak w układzie czytnika. Różnice występują jedynie w kierunku transmisji danych. Informacje z magistrali danych procesora wpisywane są do wyjściowego rejestru buforowego sygnałem mikrosterującym "rwy".

Zestaw dwóch urządzeń zewnętrznych w maszynie EW o przeciwnych kierunkach transmisji jest wystarczający do zobrazowania pracy systemu we/wy dla celów dydaktycznych. System we/wy w maszynie EW można w łatwy sposób rozbudować, dołączając nowe układy interfejsu współpracujące z procesorem na zasadach analogicznych jak opisane powyżej.

5. Tryby wymiany informacji w komputerze EW

Biorąc pod uwagę możliwości sprzętowe komputera EW można wyróżnić trzy różne - z programowego punktu widzenia - możliwe sposoby realizacji wymiany informacji, które określimy jako: pętle mikroprogramowe (połączenie mikroprogramowe), pętle programowe (połączenie programowe), praca z przerwaniami. Kryterium wyodrębnienia tych trybów jest sposób testowania gotowości urządzenia zewnętrznego.

5.1. Pętla mikroprogramowa

Testowanie gotowości realizowane jest w pętli mikroprogramu rozkazu oczekiwania na gotowość, który kończy się dopiero po jej nadejściu. Zatem testowanie to odbywa się w sposób niewidoczny dla programisty (poziomu wynikowego maszyny). Rozkaz testowania trwa tak długo, jak długo urządzenie zewnętrzne wykonuje operację wyprowadzania (wprowadzania) danej elementarnej (np. znaku).

W tym przypadku czynności związane z wymianą informacji (inicjacja pracy urządzenia zewnętrznego, przesłanie danej i testowanie gotowości) mogę być wykonane przez jeden rozkaz.

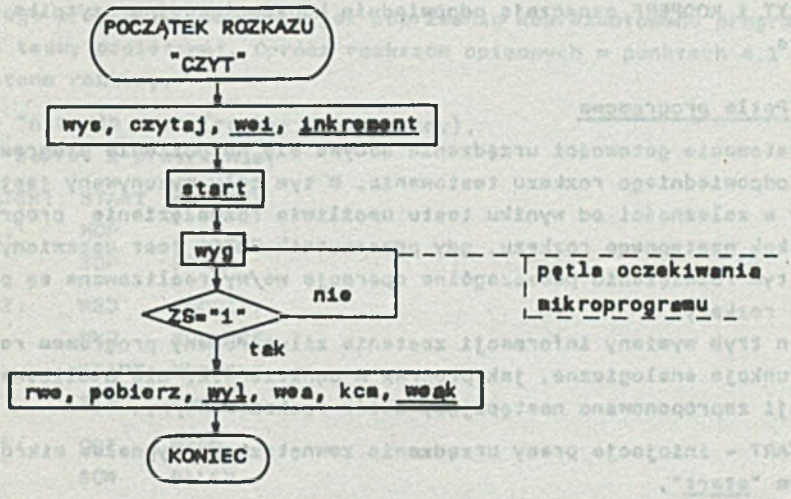
Jako przykład rozkazów wymiany informacji w trybie pętli mikroprogramowej zostaną zaprezentowane rozkazy CZYT i PERF w formie schematów blokowych mikroprogramów:

a) Rozkaz CZYT - wczytanie jednego znaku z czytnika do akumulatora. W części argumentowej rozkazu (podobnie jak w przypadku rozkazu PERF) jest kod selekcyjny urządzenia (rys. 7).
b) Rozkaz PERF - wyperforowanie na taśmie papierowej znaku znajdującego się w akumulatorze (rys. 8).

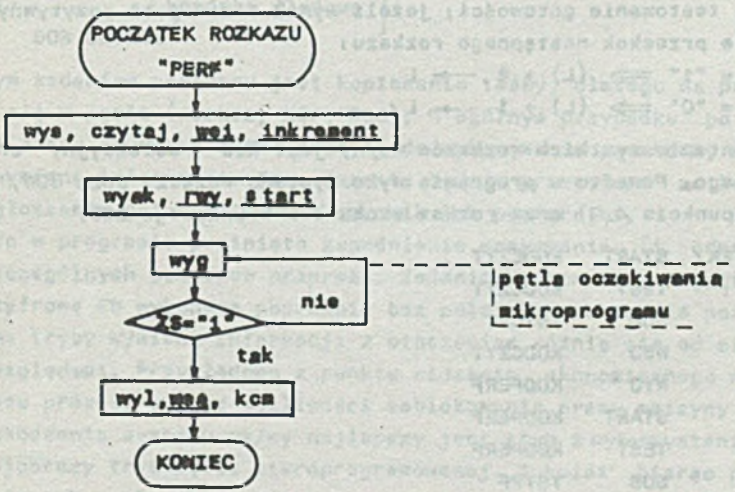
Przykładem zastosowania powyższych rozkazów jest program maszyny EW kopiujący na perforatorze taśmę papierową wprowadzaną z czytnika. Proces kopiowania kończy się po wyperforowaniu znaku zgodnego ze wzorcem umieszczonym w pamięci operacyjnej, w komórce o adresie symbolicznym WZÓR.

W programie skorzystano ponadto z rozkazów:

- ODE - odjęcie od zawartości AK zawartości komórki o adresie wskazanym przez argument rozkazu:
  (AK) - (ADR) $\longrightarrow$ AK
- SOW - skok warunkowy (przy zerowej zawartości akumulatora):
  (AK) = ϕ $\Longrightarrow$ ADR $\longrightarrow$ L, ADR $\longrightarrow$ A
  (AK) $\neq \phi$ $\Longrightarrow$ (L) + 1 $\longrightarrow$ L
- STOP - zatrzymanie pracy maszyny.



Rys. 7. Schemat blokowy mikroprogramu rozkazu CZYT



Rys. 8. Schemat blokowy mikroprogramu rozkazu PERF

```
POCZATEK:  CZYT    KODCZYT
           PERF    KODPERF
           ODE     WZÓR
           SOW     POCZATEK
           STOP
```

KODCZYT i KODPERF oznaczają odpowiednio kody selekcyjne czytnika i perforatora.

5.2. Pętla programowa

Testowanie gotowości urządzenia odbywa się na poziomie programu za pomocą odpowiedniego rozkazu testowania. W tym celu wykonywany jest rozkaz, który w zależności od wyniku testu umożliwia rozgałęzienie programu np. przeskok następnego rozkazu, gdy przerzutnik GOTOW jest ustawiony.

W tym rozwiązaniu poszczególne operacje we/wy realizowane są przez odrębne rozkazy.

Ten tryb wymiany informacji zostanie zilustrowany programem realizującym funkcje analogiczne, jak program w punkcie 4.2. Dla zrealizowania tych funkcji zaproponowano następujący zbiór rozkazów we/wy:

a) START - inicjacja pracy urządzenia zewnętrznego sygnałem mikrosterującym "start",
b) WEJ - pobranie zawartości rejestru buforowego układu interfejsu do akumulatora:
   (RWE) ⟶ AK,
c) WYJ - przesłanie zawartości akumulatora do wyjściowego rejestru buforowego:
   (AK) ⟶ RWY,
d) TEST - testowanie gotowości; jeżeli wynik testu jest pozytywny, to następuje przeskok następnego rozkazu:
   GOTOW = "1" ⟹ (L) + 2 ⟶ L
   GOTOW = "0" ⟹ (L) + 1 ⟶ L

Argumentem wszystkich rozkazów we/wy jest kod selekcyjny urządzenia zewnętrznego. Ponadto w programie wykorzystano rozkazy ODE, SOW, STOP (opisane w punkcie 4.2) oraz rozkaz skoku bezwarunkowego SOB.

```
POCZATEK:  START   KODCZYT
TSTCZT:    TEST    KODCZYT
           SOB     TSTCZT
           WEJ     KODCZYT
           WYJ     KODPERF
           START   KODPERF
TSTPF:     TEST    KODPERF
           SOB     TSTPF
           ODE     WZÓR
           SOW     POCZATEK
           STOP
```

5.3. Praca z przerwaniami

Rozkazy we/wy powodują jedynie inicjację pracy urządzeń peryferyjnych i przesył danych (między procesorem a układami interfejsu). Testowanie gotowości osiąga się przez przerwanie. Nie jest potrzebny rozkaz testowania gotowości.

Dla tego przypadku, podobnie jak poprzednio zaprezentowano program kopiowania taśmy papierowej. Oprócz rozkazów opisanych w punktach 4.1 i 4.2 wykorzystano rozkazy:

NOP - "nie rób nic" (rozkaz przerywalny),
PZP - powrót z przerwania.

```
POCZATEK:  START  KODCZYT
PETLA:     NOP
           SOP    PETLA
OBSŁCZ:    WEJ    KODCZYT
           WYJ    KODPERF
           START  KODPERF
           PZP    APCZYT
OBSŁPF:    ODE    WZÓR
           SOW    DALEJ
           STOP
DALEJ:     START  KODCZYT
           PZP    APPERF
APPERF:           ; komórka śladowa   ┐
           SOB OBSŁPF                 ┘  WEKTOR PRZERWAŃ
APCZYT:           ; komórka śladowa   ┐
           SOB OBSŁCZ                 ┘
```

Jedynym zadaniem programu jest kopiowanie taśmy, dlatego na przerwanie oczekuje się w pętli (rozkazy NOP, SOB). W ogólnym przypadku po wstępnej inicjacji pracy czytnika można przejść do wykonywania procedury nie związanej z wymianę informacji. Procedura ta będzie przerywana za każdym razem po zgłoszeniu gotowości przez czytnik lub perforator.

Ponadto w programie pominięto zagadnienie maskowania (i odmaskowywania) poszczególnych poziomów przerwań. Zadanie kopiowania będzie przez maszynę cyfrową EW wykonane poprawnie bez potrzeby maskowania poziomów.

Opisane tryby wymiany informacji z otoczeniem różnią się od siebie pod różnymi względami. Przykładowo z punktu widzenia ekonomicznego wykorzystania czasu procesora oraz możliwości zablokowania pracy maszyny w przypadku uszkodzenia systemu we/wy najlepszy jest tryb z wykorzystaniem przerwań, a najgorszy tryb pętli mikroprogramowanej. Z kolei biorąc pod uwagę osiągnięcia maksymalnej prędkości transmisji danych, a także łatwość programowania (na poziomie wynikowym) tryb komunikacji przy wykorzystaniu przerwań jest najgorszy, zaś pętla mikroprogramowa najlepsza.

Prezentacja i analiza w czasie zajęć dydaktycznych wszystkich sposobów wymiany informacji możliwych do realizacji w komputerze EW jest bardzo korzystna. Jedynym sposobem wymiany informacji, którego nie można na laboratoryjnym zestawie EW zaprezentować jest transmisja z bezpośrednim dostępem do pamięci.

6. Zakończenie

Projektowanie listy rozkazów dla komputera mikroprogramowanego polega na dokonaniu syntezy rozkazów maszynowych o funkcjach wymaganych do zrealizowania zadań wynikających z przeznaczenia komputera. W procesie syntezy rozkazów maszynowych wyróżnić można następujące fazy:

- określenie zbioru operacji pożądanych do wykonania zadań założonych w komputerze,
- określenie funkcji rozkazów umożliwiających zrealizowanie wymaganych operacji,
- analiza funkcji rozkazów - podział rozkazów na operacje elementarne,
- przyporządkowanie operacjom elementarnym odpowiednich mikrorozkazów,
- określenie sekwencji mikrorozkazów.

Przedstawiony w pracy proces projektowania listy rozkazów dla potrzeb komunikacji komputera z otoczeniem, oparty na zasadzie modelowania układowego, posiada głównie aspekt dydaktyczny. Jednakże problemy, jakie się tu pojawiają, są zbieżne z tymi, jakie występują przy projektowaniu użytkowych list rozkazów. Własności układowe komputera dydaktycznego EW umożliwiają rozwiązywanie w czasie ćwiczeń laboratoryjnych zagadnień związanych z modelowaniem rozkazów i ich optymalizacją, a to pozwala na zobrazowanie całego procesu projektowania list rozkazów dla komputerów użytkowych.

LITERATURA

[1] Węgrzyn S., Konsek M.: Przesyły międzyrejestrowe w programowanych maszynach cyfrowych o adresowanych rejestrach pamięci. Podstawy Sterowania, 1975, t. 5, z. 4.

[2] Konsek M.: Organizacja cyklu rozkazowego jako regulamin obsługi zgłoszeń. Podstawy Sterowania, 1976, t. 6, z. 1.

[3] Gacek A., Kozielski S., Stróżyna P.: Organizacja komputera dydaktycznego z możliwością modelowania listy rozkazów. Zeszyty Naukowe Politechniki Śląskiej, Informatyka, zeszyt 5.

Recenzent: Prof. dr hab. inż. Andrzej Grzywak

Wpłynęło do Redakcji 15.09.1982 r.

МОДЕЛИРОВАНИЕ НАБОРА КОМАНД ДЛЯ СРЕДСТВ СВЯЗИ С ОКРУЖЕНИЕМ ПРИ ИСПОЛЬЗОВАНИИ ДИДАКТИЧЕСКОЙ ЭВМ ТИПА EW

Р е з ю м е

В статье представлено анализ отбора машинных команд для принятой структуры системы перерывов и системы ввода-вывода дидактической ЭВМ типа EW. Рассмотрено различные варианты организации цикла машинной команды с обслуживанием перерывов и различные методы реализации обмена информации. Представлено микропрограммную реализацию исследованных функциональных вариантов.

INSTRUCTION SET MODELLING FOR COMMUNICATION BETWEEN EW DIDACTICAL COMPUTER AND ITS ENVIRONMENT

S u m m a r y

Analysis of instruction choice for a given interrupt and input/output system structure is presented in the paper. Various possibilities of instruction cycle organization with interrupt handling and different ways of input/output performance in EW didactic computer are described. Functional variants discussed in the paper are illustrated by microprograms.