ZETO ŁÓDŹ CSI

# V SZKOŁA MIKROPROCESOROWA

## Łódź, 5—7 marca 1984 r.

*MATERIAŁY SEMINARYJNE*

***Do użytku służbowego***

ZAKŁAD ELEKTRONICZNEJ TECHNIKI OBLICZENIOWEJ
PRZEDSIĘBIORSTWO PAŃSTWOWE

CENTRUM SZKOLENIA INFORMATYCZNEGO

ul. Cz. Hutora 69
tel. 36-47-70

90-558 Łódź
telex 88 52 08

ZETO ŁÓDŹ

COMPUTING CENTRE
90-146 Lodz, Narutowicza Street 136
POLAND
УЧРЕЖДЕНИЕ ПО ЭЛЕКТРОННОЙ
ВЫЧИСЛИТЕЛЬНОЙ ТЕХНИКЕ
90—146 Лодзь, ул. Нарутовича 136
ПНР

Tlx.: 88 43 14 Telephone: 829 – 13
dir. 820 – 71

***Organizatorzy:***

***mgr Zygmunt BARTKOWSKI***
***— kierownik CSI***

***mgr inż. Jerzy DAŃDA***
***— sekretarz naukowy seminarium***

***mgr inż. Stanisław RYBARCZYK***
***— sekretarz organizacyjny seminarium***

***mgr Jan ŻURAW***
***— organizator wystawy***

Zbigniew Barański
Sławomir Pilarski

# UKŁAD PAMIĘCI OPERACYJNEJ, DOSTĘPNEJ DLA DWÓCH SYSTEMÓW MIKROPROCESOROWYCH TYPU MCS 80/85

W systemach mikrokomputerowych zawierających więcej niż jeden mikroprocesor problem przekazywania informacji pomiędzy mikroprocesorami może być rozwiązany np. przez zastosowanie pamięci operacyjnej dostępnej dla dwóch /lub więcej/ mikroprocesorów. Przedstawiony tutaj układ został opracowany tak by można go było wykonać w oparciu o układy scalone produkcji krajowej. Jest on przeznaczony do współpracy z dwoma mikrokomputerami typu MCS 80/85. Schemat blokowy tego układu jest przedstawiony na rys.1.

Układ składa się z pamięci operacyjnej, dwukierunkowych trójstanowych buforów szyn danych, trójstanowych buforów szyn adresowych, dekoderów adresu i układu sterującego. Pamięć operacyjna znajduje się w przestrzeni adresowej obu mikrokomputerów. Dekodery adresu określają położenie wspólnej pamięci w przestrzeniach adresowych obu mikrokomputerów. Układ sterujący na podstawie sygnałów z dekoderów adresu oraz sygnałów MEMR i MEMW obu mikrokomputerów decyduje, który z mikrokomputerów ma dostęp do pamięci i otwiera odpowiednie bufory. W przypadku równoczesnego żądania dostępu do pamięci wspólnej jeden z mikrokomputerów musi zaczekać i dostaje sygnał READY=L. Czekający mikrokomputer otrzyma dostęp do pamięci po zakończeniu dostępu drugiego mikrokomputera. Opisane powyżej rozwiązanie zostało zastosowane w układzie umożliwiającym współpracę dwóch systemów mikrokomputerowych typu MERA200 symulujących współdziałanie centrali telefonicznej z otoczenia. Jeden z mikrokomputerów MERA-200 - SYMULATOR symuluje otoczeniem zewnętrzne centrali a drugi MERA 200-SBC jest układem sterującym centralą telefoniczną. Szczegółowy schemat wspólnej pamięci operacyjnej dla dwóch mikrokomputerów typu MERA 200 jest przedstawiony na rys. 2.

Wspólną pamięć operacyjną o pojemności 384 bajty stanowią układy U7 i U8. Dekoder adresu tej pamięci i dla MERA 200 SBC zbudowany jest na układach U1, U16A, U23c. Przez przełączenie zwory łączącej wejście E1 układu U1 z jednym z sygnałów STR2 / - STR7/ oraz wejście E2 z sygnałem A12/ lub A12 istnieje możliwość zmiany adresu tej pamięci. Dla MERA 200 SYMULATOR adres pamięci wspólnej jest wybierany w taki sam sposób a dekoder tego adresu zbudowany jest na układach U30, U37b, U38b.

Sygnały wyjściowe z dekoderów adresu sterują przerzutniki U6b i U32a, które zmieniają stan gdy odpowiedni mikrokomputer żąda dostępu do wspólnej pamięci.

Układy U2 i U3 oraz U10 i U11 pracują jako trójstanowe, dwukierunkowe bufory szyn danych obu mikrokomputerów.

Uakłady U4 i U5 oraz U12 i U13 pracują jako trójstanowe bufory szyny adresowej.

Aby uniknąć konfliktu w przypadku gdyby oba mikrokomputery próbowały korzystać ze wspólnej pamięci jednocześnie, układ sterujący przydziela wspólną pamięć jednemu albo drugiemu mikrokomputerowi. MERA 200 SYMULATOR ma wyższy priorytet dostępu do tej pamięci niż MERA 200 SBC. Układ sterujący zbudowany jest na elementach: U9, U14, U16b, c, d, U17, U18a,c,d, U19, U20, U21.

Elementy U24a i U40a przesyłają sygnały blokujące fragmenty pamięci odpowiedniego mikrokomputera o adresach takich, jakie są przyporządkowane pamięci wspólnej.

Ponieważ w mikrokomputerze MERA 200 nie ma bezpośredniego dostępu do wejścia READY mikroprocesora 8085, konieczne jest wprowadzenie zmiany na płytkach mikroprocesorów. Sygnał podawany na wejście READY 8085 musi być iloczynem dwóch sygnałów: sygnału READY generowanego przez układ 74121 należący do mikrokomputeru i sygnału READY generowanego w układzie sterującym. W MERA 200 SBC sygnał ten jest tworzony przez elementy U17a i U23b a w MERA 200 SYMULATOR przez U37a.

W czasie gdy żaden z mikrokomputerów nie korzysta z pamięci wspólnej stany logiczne w układzie sterowania są następujące: na wyjściu Q przerzutnika U6b jest stan H, na wyjściu Q /przerzutnika U14a stan L. Na wyjściu Q przerzutnika U15a jest L a na

Q/ - stan H. Na wyjściu Q/ przerzutnika U32a jest L, na wyjściu Q U14b jest H a na Q/ - L. Stan L z wyjścia Q/ układu U14b wymusza stan H na wyjściu układu U20c i na wyjściach układów U22a,b,c. Na wejściach CS/ układów U10, U11, U12 i DS1/ układu U13 jest stan H i wyjścia tych układów są w stanie wysokiej impedancji Tak samo jest od strony MERA 200 SBC: stan L z wyjścia Q/ układu U14a wymusza stan H na wyjściu bramki U18d i na wejściach CS/ i DS/ układów U2, U3, U4, U5, których wyjścia są również w stanie wysokiej impedancji. Na wyjściach bramek U17b i U17c jest stan H i na wejściach READY obu mikroprocesorów jest H.

Gdy jeden z mikrokomputerów np. MERA 200 SBC zażąda dostępu do pamięci wspólnej tzn. na wyjściu 04 układu U1 będzie stan L, na wyjściu Q układu U6b pojawi się L, wyjście Q/ przerzutnika U14a zostanie ustawione w stan H, na wejściu D i wyjściu Q układu U15a będzie H. Na wyjściu Q/ U15a będzie stan L, na wyjściu bramki U16b będzie H i gdy sygnał MEMR/ lub MEMW/ będzie L to na wejściach CS/ układów U2, U3, U4 oraz DS1/ układu U5 będzie L. Adres z szyny adresowej MERA 200 SBC zostanie podany na wejścia adresowe układów pamięci U7 i U8 a o kierunku transmisji przez bufory szyny danych U2 i U3 będzie decydował poziom sygnału na linii MEMR/ w zależności od tego czy będzie się odbywało czytanie z, czy wpisywanie do pamięci wspólnej. Gdy w tym czasie MERA 200 SYMULATOR zgłosi żądanie dostępu, tzn. na wyjściu Q układu U14b pojawi się L, na Q/ stan H, na wszystkich wejściach bramki U17c będzie stan H, na jej wyjściu L i na wejście READY mikroprocesora 8085 MERA 200 SYMULATOR zostanie podany stan L - mikroprocesor będzie w stanie WAIT do momentu aż mikroprocesor w MERA 200 SBC zakończy cykl maszynowy MEMR lub MEMW i przejdzie do wykonywania następnego cyklu, w którym na szynę adresową zostanie wystawiony adres nie należący do obszaru adresowego pamięci wspólnej i na wyjściu 04 układu U1 pojawi się stan H.

Następnie po kolejnych cyklach zegara CLK /końcówka /37/ SBC/ na wejście READY mikroprocesora 8085 MERA 200 SYMULATOR zostanie podany stan H, zostaną otworzone bufory szyn danych i adresowe U10, U11, U12 i U13 oraz U25 i U26 i MERA 200 SYMULATOR przejmie kontrolę nad wejściami adresowymi i danymi układów pamięci U7 i U8.

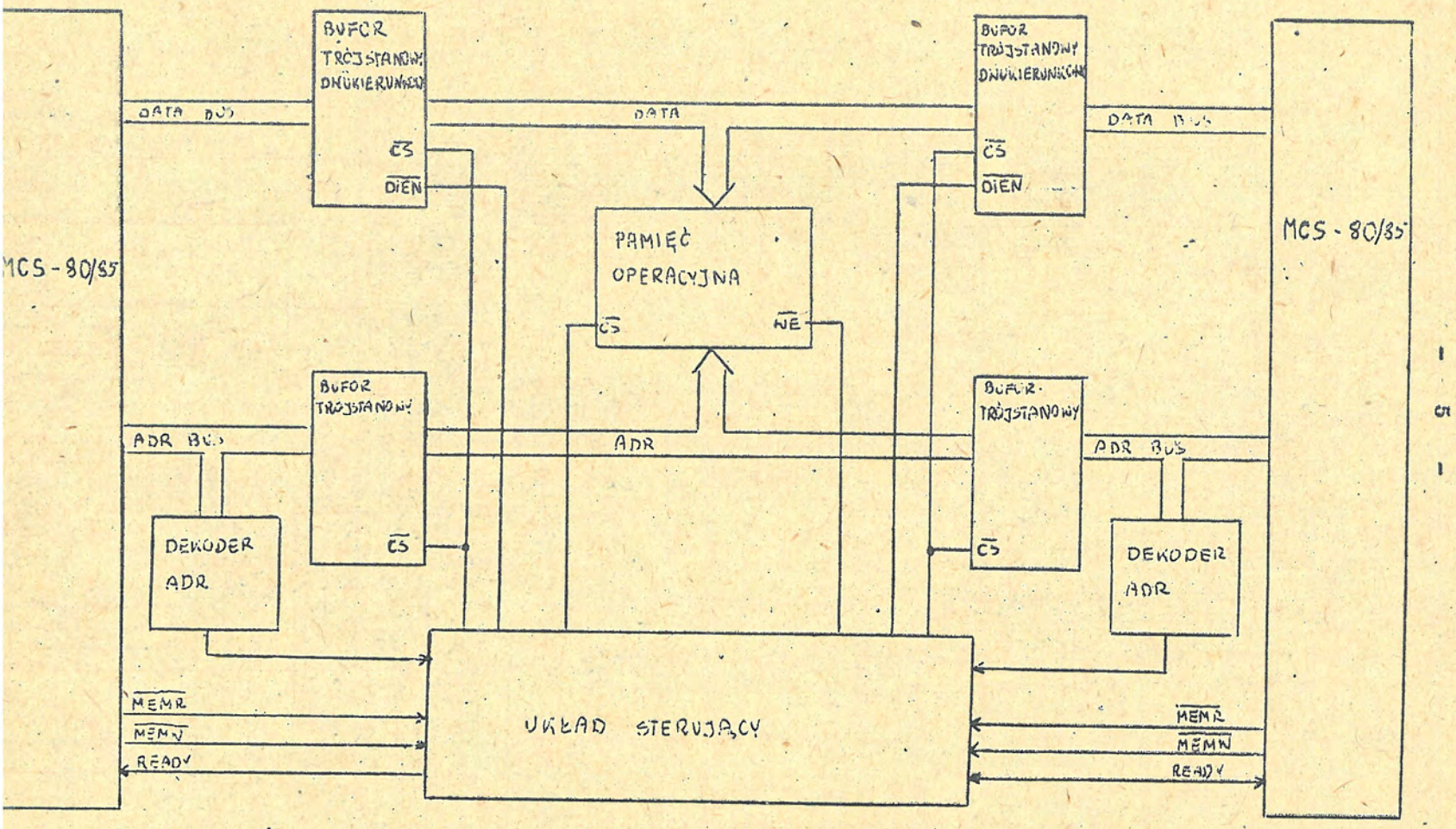
W przypadku gdy MERA 200 SBC nie korzysta z pamięci wspólnej a MERA 200 SYMULATOR zgłosi żądanie dostępu, wówczas operacja dostępu odbywa się w opisany już sposób, tak jak dla MERA 200 SBC.

Po zdekodowaniu jednego z adresów pamięci wspólnej przez którykolwiek z dekoderów adresu, gdy na wyjściu Q układu U6b lub U32a pojawi się stan L, poprzez układy odpowiednio U24a lub U40a wymuszany jest stan L na linii sterującej MEMR-W /Końcówka 78/ odpowiedniego mikrokomputera. W ten sposób, gdy mikrokomputer korzysta z pamięci wspólnej, jest blokowana jego pamięć wewnętrzna o adresach takich, jakie są przyporządkowane pamięci wspólnej. Opisana powyżej współpraca dwóch systemów mikrokomputerowych mogłaby zostać zrealizowana w inny sposób. Mikrokomputery mogłyby wymieniać dane między sobą np. przez układy WE/WY lub przy użyciu kanału DMA.

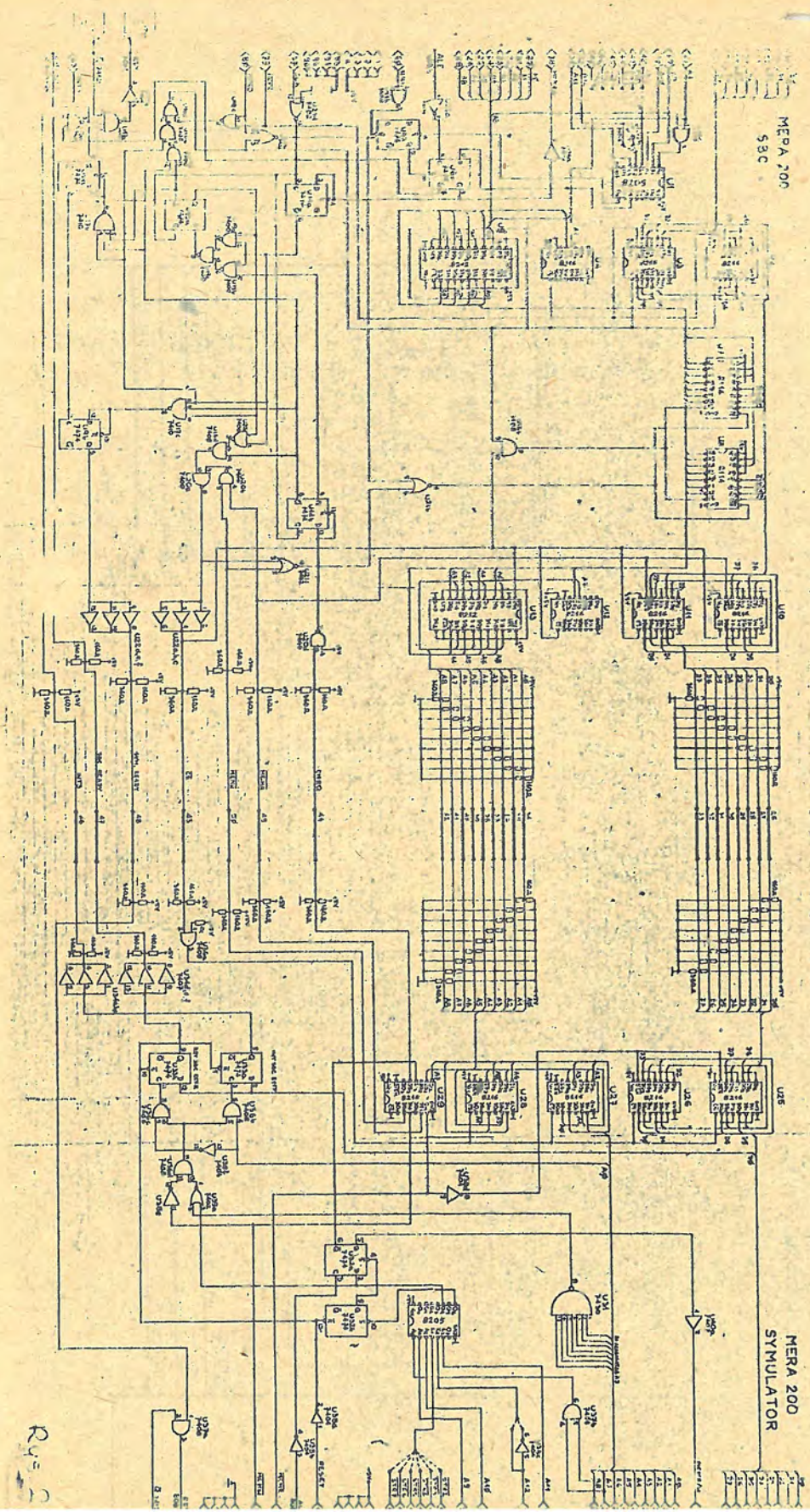
Jednak przy wykorzystaniu układów WE/WY, znaczna część czasu pracy każdego z mikrokomputerów byłaby zajęta na samą operację wysyłania danych na port WY lub ich odczytywania z portu WE a następnie umieszczenia ich pod odpowiednim adresem w pamięci. Adres ten musiałby być przesyłany łącznie z danymi. Byłoby to szczególnie niekorzystne ze względu na straty czasu, gdyby zaszła potrzeba częstego przesyłania dużych bloków danych.

Natomiast przy użyciu kanału DMA, przed każdą operacją przesłania danych, byłoby konieczne zaprogramowanie kontrolera DMA, a ponadto na czas trwania transmisji danych oba mikrokomputery musiałyby zostać wprowadzone w stan HOLD.

Wymiana danych między mikrokomputerami za pośrednictwem wspólnego obszaru pamięci operacyjnej umożliwia każdemu z mikrokomputerów dostęp do dowolnego bajtu danych ze wspólnego obszaru pamięci, a ponadto straty czasu na wymianę danych zostały ograniczone do minimum - każdy z mikrokomputerów będzie w stanie WAIT co najwyżej przez czas trwania jednego cyklu maszynowego drugiego mikrokomputera i to tylko wtedy, gdy oba mikrokomputery będą próbowały korzystać ze wspólnej pamięci w tym samym momencie.


MCS - 80/85
BUFOR TRÓJSTANOWY DWUKIERUNKOWY
DATA BUS
DATA
CS
DIEN
PAMIĘĆ OPERACYJNA
WE
BUFOR TRÓJSTANOWY DWUKIERUNKOWY
DATA BUS
CS
DIEN
MCS - 80/85
BUFOR TRÓJSTANOWY
ADR BUS
ADR
CS
DEKODER ADR
BUFOR TRÓJSTANOWY
CS
ADR BUS
DEKODER ADR
UKŁAD STERUJĄCY
MEMR
MEMW
READY
MEMR
MEMW
READY

Rys 1.


MERA 200
SYMULATOR

doc.dr inż.Jerzy FEDOROWSKI
Zakład Nawigacji Technicznej
Instytutu Nawigacji Morskiej
Wyższej Szkoły Morskiej
w Szczecinie
ul.Wały Chrobrego 1

# ALGORYTM GENERACJI PERSPEKTYWICZNEGO OBRAZU STATKU I OBRAZU RADAROWEGO DLA SYSTEMU MIKROKOMPUTEROWEGO

## 1. Wprowadzenie

W szkolnictwie morskim, w procesie dydaktycznym, istotną rolę odgrywają symulatory systemów i urządzeń, których instalacja i eksploatacja w warunkach lądowych jest niecelowa ze względu na wysoki koszt. Szczególne znaczenie ma symulacja procesów nawigacyjnych, których w warunkach lądowych w ogóle nie można zrealizować. Do tych procesów należą czynności związane z określaniem bieżącej pozycji statku będącego w ruchu, za pomocą urządzeń radionawigacyjnych, manewrowanie statkiem w ograniczonych akwenach, np. na torach wodnych, podczas wchodzenia i wychodzenia z portu oraz w celu zapobiegania zderzeniom. Takie manewrowanie w warunkach naturalnych często odbywa się podczas ograniczonej widzialności, co wymaga korzystania z radaru.

Szkolenie w tym zakresie na statku jest bardzo kosztowne ponadto w rzeczywistych warunkach nie można stwarzać niebezpiecznych sytuacji, które są bardzo pożądane ze względów szkoleniowych.

Z wymienionych powodów rozpowszechniają się na świecie różnego rodzaju, bardzo rozbudowane i kosztowne, symulatory nawigacyjne oparte na dużych komputerach. Ostatnio daje się zauważyć tendencję do stosowania również w tej dziedzinie systemów mikrokomputerowych / 6 /.

Użytkowane w Wyższej Szkole Morskiej w Szczecinie, już od dziesięciu lat, własnej budowy symulator sonarowy i symulator radarowy brytyjskiej firmy Redifon oraz ostatnio zakupiony symulator siłowni okrętowej norweskiej firmy Norcontrol, potwierdzają w całej rozciągłości ich przydatność jak i wysokie koszty zakupu łącznie przekraczające milion dolarów.

Symulacja procesów nawigacyjnych, jak wykazały doświadczenia, może mieć również znaczenie badawcze.

W tej sytuacji podejmowanie prób opracowywania i budowy urządzeń symulacyjnych, nawet o znacznie mniejszych możliwościach niż oferują znaczące firmy, wypełnić może lukę pomiędzy ich brakiem a dużą inwestycją. Takie działanie ma również duże znaczenie poznawcze i może mieć znaczenie gospodarcze.

W Zakładzie Nawigacji Technicznej Instytutu Nawigacji Morskiej WSM w Szczecinie opracowano już koncepcję i algorytmy symulacji radionamierzania / 2 /, radionawigacyjnego systemu Decca / 3 / oraz cyfrowej symulacji ruchu statku / 4 /, / 5 / oparte na systemie mikrokomputerowym.

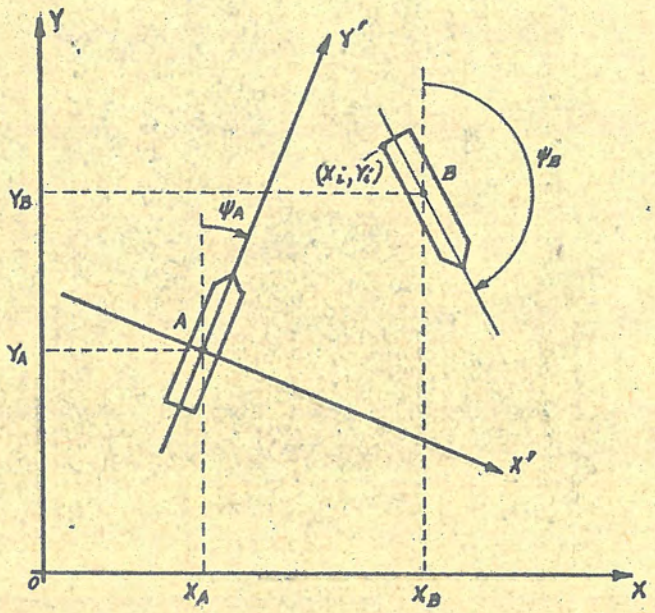
Jednym z bardzo ważnych a zarazem najtrudniejszym problemem symulacji manewrowania jest cyfrowa generacja perspektywicznego obrazu otoczenia widzianego przez okna mostka statku lub odpowiadającego danej sytuacji obrazu radarowego.

Przedmiotem niniejszego referatu jest algorytm cyfrowej generacji perspektywicznego obrazu poruszającego się statku, który może stwarzać zagrożenie kolizyjne oraz odpowiadającego takiej sytuacji obrazu radarowego.

## 2. Idea tworzenia obrazu

Tworzenie obrazu oparte jest na założeniu, że dwa statki, A statek obserwatora nazywany dalej własnym i B statek obserwowany nazywany dalej obcym, poruszają się w układzie współrzędnych prostokątnych X, Y. Ich chwilowe pozycje określone są odpowiednio za pomocą ich współrzędnych $X_A$, $Y_A$, $X_B$, $Y_B$ a kursy jako kąty $\psi_A$, $\psi_B$, liczone od dodatniego zwrotu

osi Y w kierunku zgodnym z kierunkiem ruchu strzałek zegara, rys.1.




Rys.1. Zasada transformacji współrzędnych

Stosując odpowiednie przekształcenia współrzędnych, to znaczy przesunięcia i obroty, zapisane za pomocą stosowanych macierzy transformacji w przestrzeni dwuwymiarowej / 1 /, drogą kolejnych transformacji uzyskuje się współrzędne wszystkich charakterystycznych punktów konturu statku obserwowanego B w ruchomym układzie współrzędnych, związanym ze statkiem obserwatora A.

Macierz transformacji współrzędnych statku B ma postać:

$$T_B = \begin{vmatrix} \cos\psi_B & -\sin\psi_B & 0 \\ \sin\psi_B & \cos\psi_B & 0 \\ -X_B\cos\psi_B - Y_B\sin\psi_B + X_B & X_B\sin\psi_B - Y_B\cos\psi_B + Y_B & 1 \end{vmatrix}$$

Macierz transformacji współrzędnych statku A ma postać:

$$T_A = \begin{vmatrix} \cos\psi_A & \sin\psi_A & 0 \\ -\sin\psi_A & \cos\psi_A & 0 \\ -X_A\cos\psi_A + Y_A\sin\psi_A + X_A & -X_A\sin\psi_A - Y_A\cos\psi_A + Y_A & 1 \end{vmatrix}$$

Macierzą łącznej transformacji jest iloczyn transformacji przy zachowanej odpowiedniej kolejności czynników:

$$T = T_B \cdot T_A$$

Współrzędne jednorodne punktów charakterystycznych konturu $x_i$, $y_i$ statku B, w ruchomym układzie współrzędnych X', Y' z uwzględnieniem przesunięcia przyjmują postać:

$$\begin{vmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot \\ x_i & y_i & 1 \\ \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot \\ x_n & y_n & 1 \end{vmatrix} \cdot T =$$

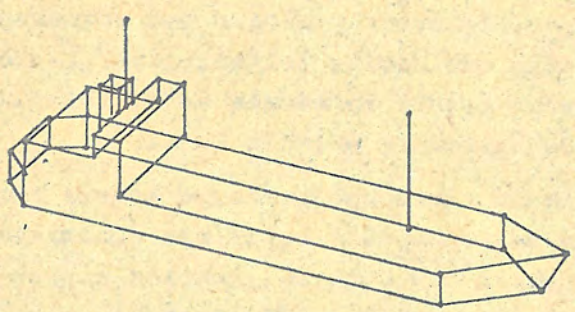
$$= \begin{vmatrix} B_{X1}\cos\Delta\psi + B_{Y1}\sin\Delta\psi - \Delta X\cos\psi_A + \Delta Y\sin\psi_A \\ \cdot \\ \cdot \\ B_{Xi}\cos\Delta\psi + B_{Yi}\sin\Delta\psi - \Delta X\cos\psi_A + \Delta Y\sin\psi_A \\ \cdot \\ \cdot \\ B_{Xn}\cos\Delta\psi + B_{Yn}\sin\Delta\psi - \Delta X\cos\psi_A + \Delta Y\sin\psi_A \end{vmatrix}$$

$$\begin{vmatrix} -B_{X1}\sin\Delta\psi + B_{Y1}\cos\Delta\psi - \Delta X\sin\psi_A - \Delta Y\cos\psi_A & 1 \\ \vdots & \vdots \\ -B_{Xi}\sin\Delta\psi + B_{Yi}\cos\Delta\psi - \Delta X\sin\psi_A - \Delta Y\cos\psi_A & 1 \\ \vdots & \vdots \\ -B_{Xn}\sin\Delta\psi + B_{Yn}\cos\Delta\psi - \Delta X\sin\psi_A - \Delta Y\cos\psi_A & 1 \end{vmatrix}$$

gdzie:

$$B_{xi} = X_i - X_B$$
$$B_{Yi} = Y_i - Y_B$$
$$\Delta\psi = \psi_B - \psi_A$$
$$\Delta X = X_A - X_B$$
$$\Delta Y = Y_A - Y_B$$

Rozpatrywany kontur obserwowanego statku B leży w płaszczyźnie lustra wody. Budując na nim trójwymiarową bryłę, utworzoną przez zbiór wysokości charakterystycznych punktów otrzymuje się figurę przestrzenną pokazaną na rys.2.



Rys.2. Kształt i punkty charakterystyczne statku

Ograniczona liczebność zbioru punktów charakterystycznych wynika z dążenia do kompromisu pomiędzy podobieństwem bryły powstałej ze zbioru punktów łączonych odcinkami pros-

tych, do statku a liczbą operacji wykonywanych podczas każdej transformacji.

W przedstawionym tu algorytmie w celu zmniejszenia liczby operacji przyjęto taki kształt obserwowanego statku by możliwie jak największa liczba punktów charakterystycznych miała wspólne współrzędne w płaszczyźnie X, Y.

Po wykonaniu odpowiednich transformacji, punkty charakterystyczne bryły statku przenoszone są na pionową płaszczyznę prostopadłą do wzdłużnej płaszczyzny symetrii własnego statku, reprezentowaną przez ekran monitora. To przenoszenie wykonywane jest w taki sposób, że na ekranie monitora powstaje rzut centralny bryły statku, przy czym środkiem rzutu jest miejsce umieszczenia obserwatora.

Wymiar poziomy ekranu monitora, który reprezentuje okno mostka statku i odległość obserwatora od ekranu określają rozwartość kątową sektora obserwacji.

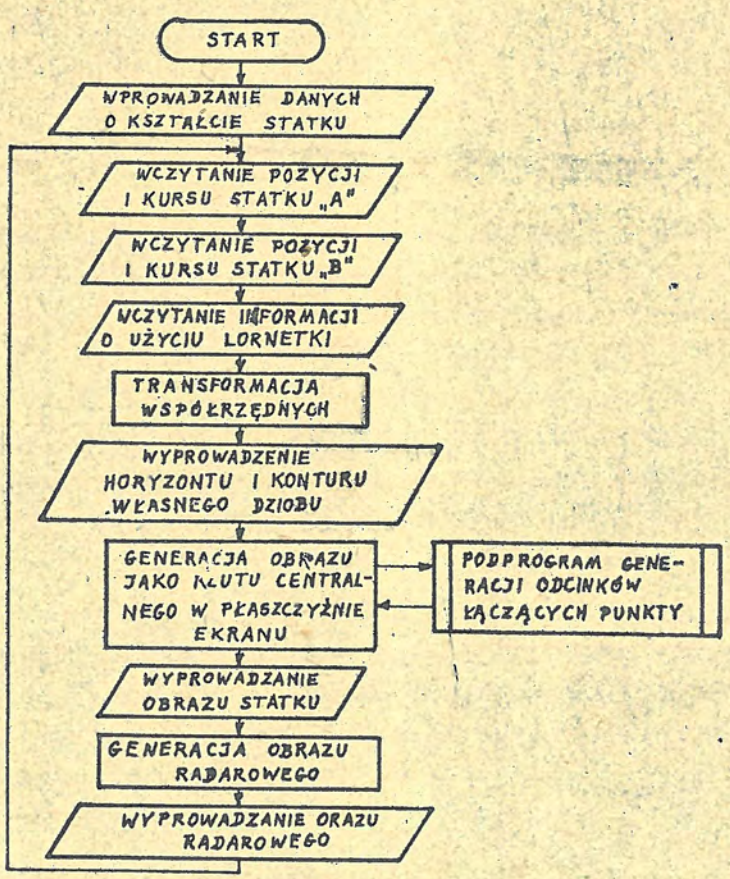
Z uwagi na możliwość praktycznej realizacji opisanego sposobu generacji obrazu, oprócz założenia że symulacja ma być wykonywana przez system mikroprocesorowy oparty na mikroprocesorze 8080A i musi być realizowana w czasie rzeczywistym, założono dodatkowo, że obraz powinien być odtwarzany metodą rastrową na typowym odbiorniku telewizyjnym. Zastosowanie grafoskopu konturowego byłoby wprawdzie korzystniejsze ale pociągnęłoby za sobą znacznie większe koszty.

Znormalizowana szerokość pasma, w kanale obrazu, typowych odbiorników telewizyjnych ogranicza rozdzielność obrazu, wzdłuż linii do około 500 punktów. Podczas eksperymentów stosowano podział obrazu na 256 x 192 elementy /pixel'e/. Takie ograniczenie rozdzielności obrazu powoduje, że obraz nawet dużego statku z odległości rzędu 1.5 Mm widoczny jest na ekranie w postaci jednego elementu, co leży znacznie poniżej zdolności rozdzielczej wzroku.

Mając powyższe na uwadze oraz to że w praktyce w celu lepszej obserwacji odległych obiektów stosuje się lornetkę, w algorytmie wprowadzono możliwość symulacji takiej obserwacji.

Wprowadzając informację o tym, że obraz odległego statku ma być widziany przez lornetkę o określonym powiększeniu, obraz statku na ekranie zostaje stosownie powiększony z uwzględnieniem typowego w tej sytuacji spłaszczenia perspektywy.

Istotnym uzupełnieniem algorytmu tworzenia perspektywicznego obrazu stało się odtwarzanie również obrazu radarowego. Wykorzystano w tym celu wynik transformacji współrzędnych, bez znaczącej rozbudowy algorytmu.
Schemat blokowy opisanego algorytmu przedstawia rys.3.




Rys.3. Schemat blokowy algorytmu

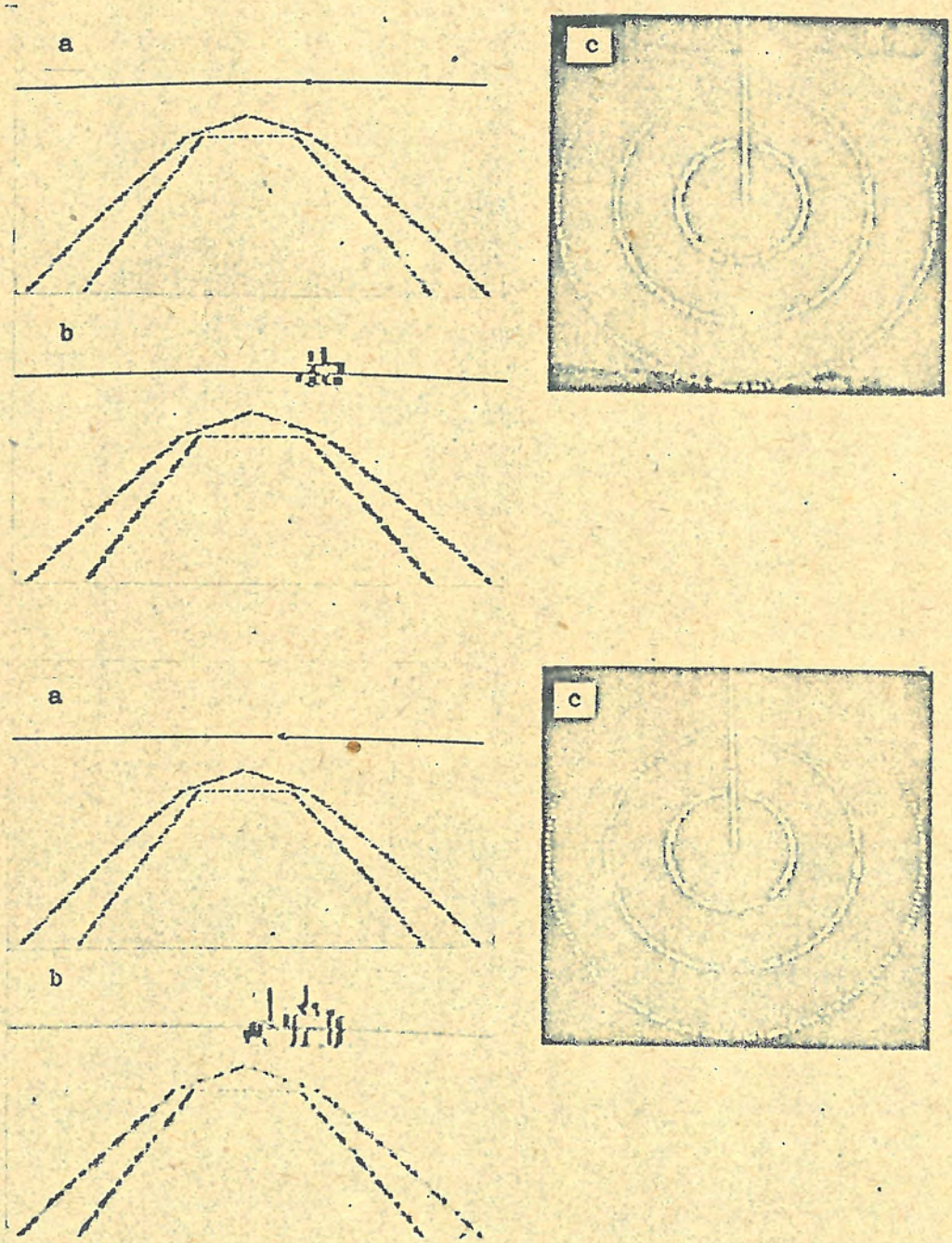
## 3. Zakończenie i wnioski

Z założenia, że algorytm ma być realizowany przez system oparty na mikroprocesorze 8080A w czasie rzeczywistym, wynikła konieczność wnikliwej analizy możliwości minimalizacji wykonywanych operacji. Szczególną uwagę zwrócono na ograniczenie stosowania funkcji trygonometrycznych. Okazało się, że muszą one występować jedynie w operacjach transformacji współrzędnych i odtwarzanie kręgów odległości w obrazie radarowym.

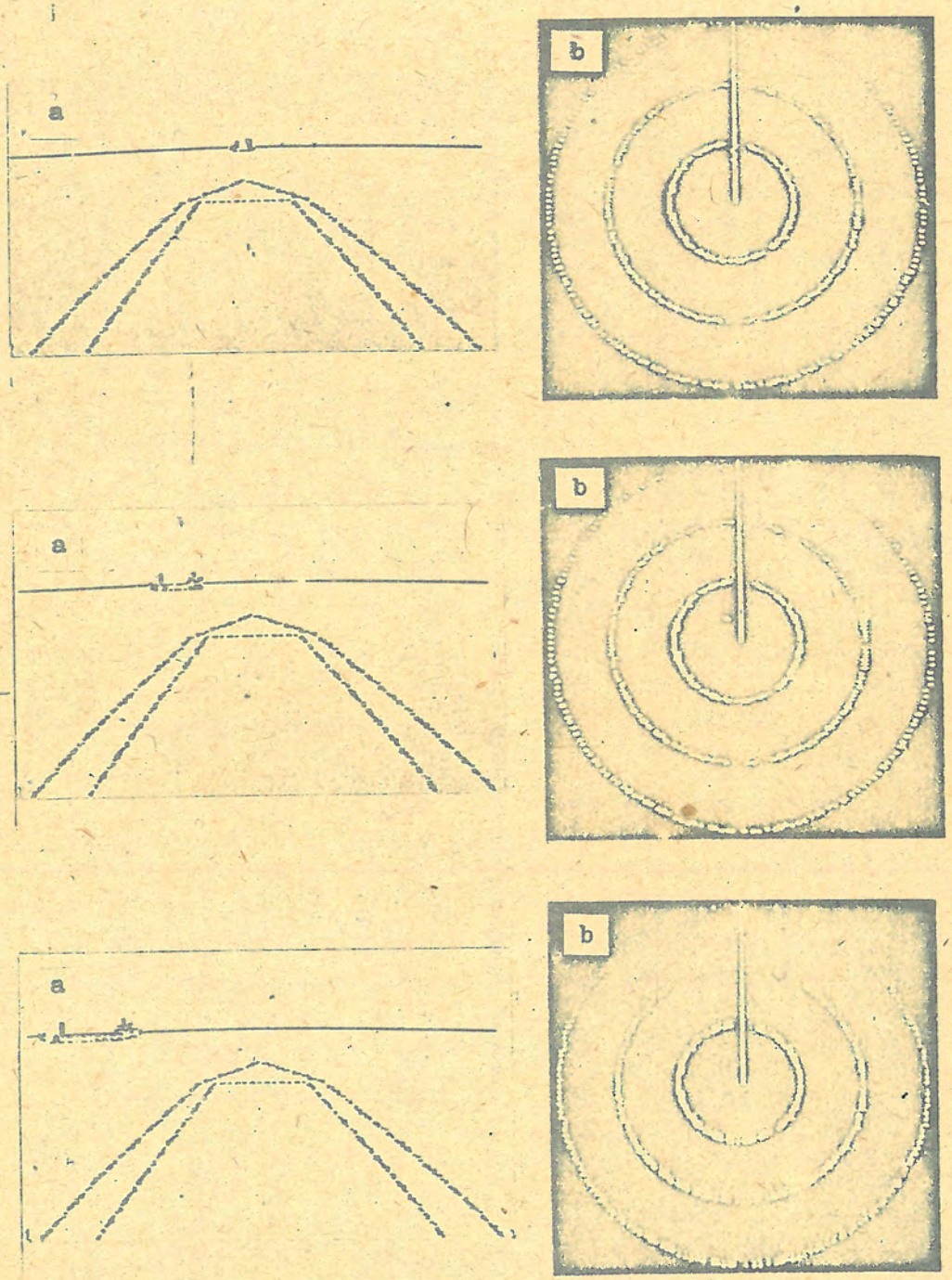
Z analizy błędów oraz uwzględnienia ograniczonej rozdzielności obrazu wynikła możliwość zastosowania interpolacyjnej metody wyznaczania wartości funkcji sin i cos, stablicowanych w przedziale od $0^{o}$ do $90^{o}$ co $10^{o}$. Dla wytwarzania kręgów odległości, w obrazie radarowym, zastosowano równania parametryczne co pozwoliło na całkowite wyeliminowanie w tym miejscu funkcji trygonometrycznych.

Fragmenty programu sprawdzono i uruchomiono na systemie SDK-85 firmy Intel / 7 /. Ze względu na dalsze doskonalenie koncepcji nie wykonano układu umożliwiającego wyprowadzenie informacji z tego systemu w postaci odpowiedniej dla typowego odbiornika telewizyjnego. W tej sytuacji opisany algorytm w całości sprawdzony został również na mikrokomputerze ZX81 z pamięcią 16K i przy pomocy uproszczonego programu grafiki. Rezultat tych prób przedstawiają zdjęcia wykonane z ekranu pokazane na rys.4, 5 i 6.

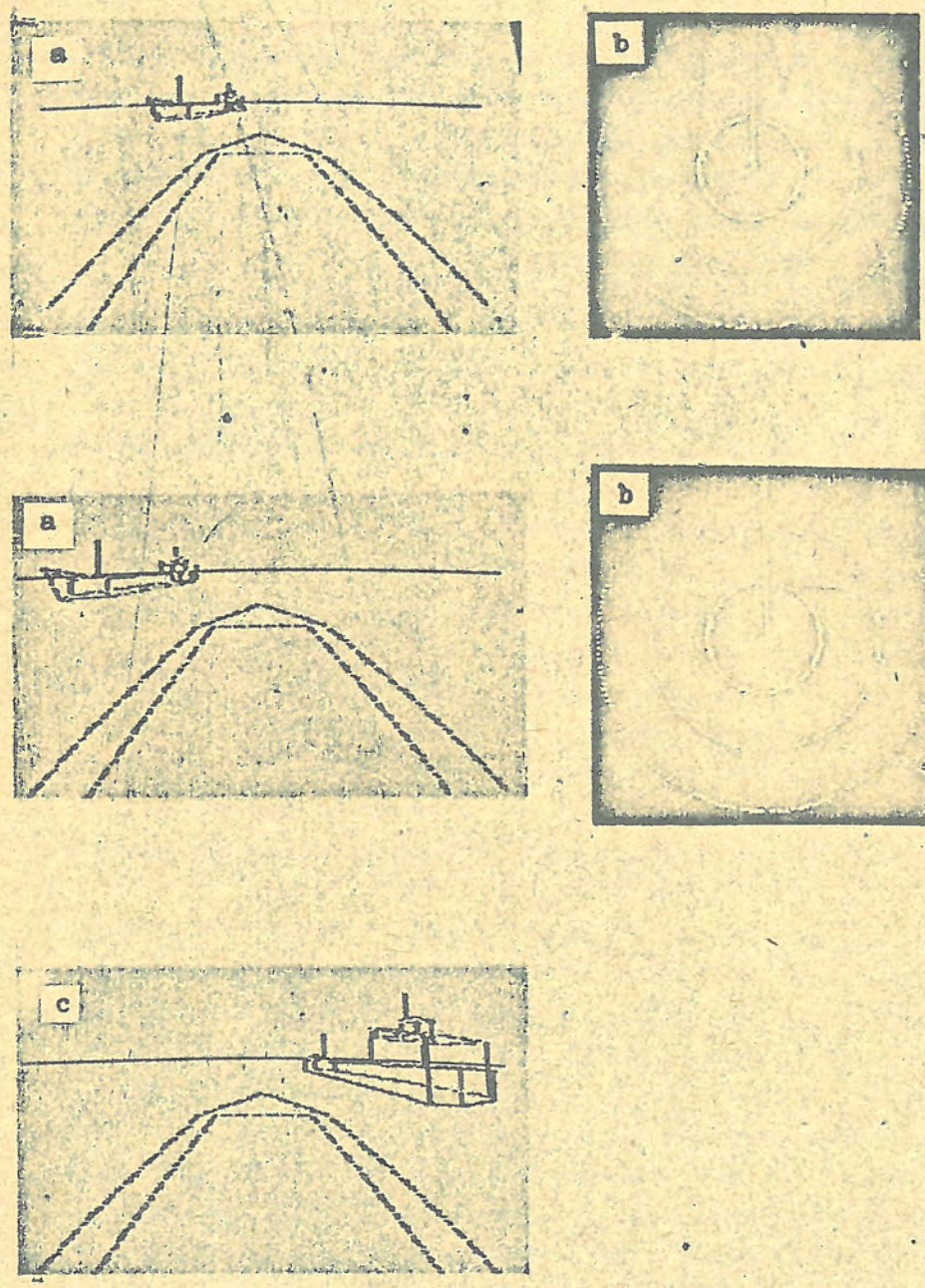
W końcowej fazie prac eksperymentalnych stwierdzono celowość takiej modyfikacji algorytmu by możliwa była wzajemna obserwacja, to znaczy aby równocześnie był symulowany obraz statku A widziany ze statku B. Stwierdzono również, że celowe jest zwiększenie sektora obserwacji do $180^{o}$, z podziałem tego sektora na pięć obrazów odpowiadających pięciu oknom mostka.




Rys.4 Widok zbliżającego się statku w kolejnych położeniach a/bez lornetki b/z lornetką c/na ekranie radarowym na zakresie obserwacji 1.5 Mm




Rys.5 a/widok zbliżającego się statku w kolejnych położeniach
b/na ekranie radarowym na zakresie obserwacji 1.5 M m




Rys.6 a/widok bliskiego statku b/na ekranie radarowym na zakresie obserwacji 1.5 Mm c/widok statku od strony rufy

W wyniku wprowadzenia do opisywanego algorytmu odtwarzania obrazu radarowego okazało się, że otwiera to możliwości przeprowadzenia na drodze symulacyjnej szeregu badań z zakresu interpretacji obrazu radarowego a w szczególności zobrazowań stosowanych w radarowych urządzeniach antykolozyjnych.

Literatura

1. Borsuk K.: Geometria analityczna wielowymiarowa. Warszawa, PWN 1976.
2. Fedorowski J.: Mikrokomputerowy system symulacji sygnałów radionamierzania, III Krajowa Konferencja Naukowo-Techniczna Zastosowanie Komputerów w Przemyśle, Szczecin 22-23 września 1983r., Tom 2, str.180.
3. Fedorowski J.: Symulacja systemu radionawigacyjnego z zastosowaniem systemu mikrokomputerowego. 13. Internationale Tagung "Automatisiete Entscheidungsvorbereitung für den Schiffsoffizier - Grundlage für hohe Effektivität im Schiffsführungsprozeβ", 9 und 10 November 1983 in Rostock-Warnemünde Deutsche Demokratische Republik.
4. Fedorowski J.: Symulacja ruchu statku oparta na systemie mikrokomputerowym. 13. Internationale Tagung "Automatisiete Entscheidungsvorbereitung für den Schiffsoffzier - Grundlage für hohe Effektivität im Schiffsführungsprozeβ", 9 und 10 November 1983 in Rostock-Warnemünde Deutsche Demokratische Republik.
5. Fedorowski J.: Komputerowa generacja obrazu, Zeszyty Naukowe Wyższej Szkoły Morskiej w Szczecinie, 1984 /w druku/.
6. Biuletyn: SIMRAD Echo SIMULATOR EDITION For maritime circles all over the world Nr 32, February 1981.
7. SDK-85 System Design Kit User's Manual Intel Comporation 1978.

Sławomir Pilarski
Instytut Informatyki
Politechniki Warszawskiej
00-650 Warszawa
ul. Nowowiejska 15/19

Piotr Zapendowski
Instytut Technologii Elektronowej
NPCP "UNITRA-CEMI"
02-668 Warszawa
Al. Lotników 32/46

# EMULATOR SPZE 48/41A
# WŁASCIWOSCI FUNKCJONALNE I UŻYTKOWE

## 1. WSTEP

Koncepcja tego "in-circuit real-time" emulatora powstała na przełomie roku 1982 i 1983. Służy on do uruchamiania systemów mikroprocesorowych zawierających mikrokomputery jednostrukturowe 8048, 8035 lub 8041A produkowane przez firmę Intel. Rodzaj pracy wybierany jest konwersacyjnie po włączeniu zasilania. System zapewnia łatwy dostęp do pamięci, rejestrów i buforów oraz informacji dotyczącej stanu procesora. Komunikacja użytkownika z systemem odbywa się przez klawiaturę funkcyjną i osiem siedmiosegmentowych wyświetlaczy. Ponadto możliwa jest transmisja danych z komputera posiadającego standardowe wyjście do perforatora taśmy papierowej.

## 2. MOŻLIWOSCI SYSTEMU

Najistotniejsze cechy emulatora dla trzech rodzajów pracy zostaną przedstawione w punktach.

Emulacja układu 8048

- emulacja do 4K bajtów programu

- możliwość pożyczenia systemowi użytkownika 256 bajtów RAM-u
- możliwość pracy z zewnętrzną pamięcią danych znajdującą się w systemie użytkownika

Emulacja układu 8035

- możliwość pożyczenia systemowi użytkownika 4k bajtów pamięci programu
- możliwości pracy z pamięcią programu znajdującą się w systemie użytkownika
- możliwość pożyczenia systemowi użytkownika 256 bajtów RAM-u
- możliwość pracy z zewnętrzną pamięcią danych znajdującą się w systemie użytkownika

Emulacja układu 8041A

- emulacja do 2k bajtów programu
- możliwość symulowania działania procesora nadrzędnego

Ważną wspólną cechą trzech rodzajów pracy jest możliwość uruchamiania programów o ośmiu poziomach zagnieżdżenia.

Istnieją następujące mechanizmy przerwania emulacji:

- zatrzymanie wykonywania programu przy odwołaniu do określonej komórki pamięci programu
- zatrzymanie wykonywania programu przy n-tym odwołaniu do określonej komórki pamięci programu /n 4000h/
- zatrzymanie po wykonaniu n rozkazów /n 4000h/
- przerwanie wykonywania programu klawiszem STOP EMULATION
- przerwanie wykonywania programu klawiszem RESET z doprowadzeniem systemu do stanu początkowego

## 3. MONITOR

Monitor emulatora zajmuje 4k bajty programu napisanego w języku assemblera mikroprocesora 8048.

Bezpośrednio dostępnymi funkcjami monitora są:

- MOV - przesuwanie obszarów pamięci programu i pamięci danych
- FIL - wypełnianie bajtów pamięci programu lub danych żądaną wartością
- RD - czytanie z czytnika taśmy dziurkowanej do pamięci programu lub danych /format taśmy - INTEL-Hex
- GO - start emulacji od podanego adresu z odpowiednim stanem procesora
- PM - przeglądanie i modyfikacja zawartości pamięci programu
- DM - przeglądanie i modyfikacja zawartości pamięci danych
- REG - przeglądanie i modyfikacja zawartości rejestrów R0-R7 wewnętrznej pamięci danych /dotyczy obu banków/
- MRG - przeglądanie i modyfikacja zawartości 15 rejestrów specjalnego przeznaczenia /1- i 2-bajtowych/
- CAL - kalkulator czterodziałaniowy /+,-,x,:/ dokonujący obliczeń przy podstawie dziesiętnej lub szesnastkowej dla liczb całkowitych. Można również dokonywać konwersji liczb dziesiętnych na szesnastkowe i odwrotnie.

Funkcjami sterującymi wprowadzanie i wyświetlaniem danych oraz realizację innych funkcji są:

- RET - użyta w momencie zakończenia poprawnej sekwencji poleceń powoduje ich wykonanie
- NXT - powoduje zwiększenie adresu o 1 lub przejście do następnego rejestru lub bitu w przypadku funkcji PM, DM, REG, MRG, separuje adresy i daną w przypadku funkcji MOV i FIL
- PRV - powoduje zmniejszenie adresu o 1 lub przejście do poprzedniego rejestru lub bitu w przypadku funkcji PM, DM, REG, MRG
- CLR - powoduje usunięcie ostatnio wprowadzonej wartości liczbowej przed zaakceptowaniem jej jednym z klawiszy RET, NXT, lub PRV

Po włączeniu zasilania użytkownik musi wybrać typ emulowanego mikroprocesora i zadeklarować /jeśli to możliwe/ z jaką pamięcią ma współpracować system użytkownika. Deklaracja ta dotyczy oczywi-

ście tylko pamięci programu i zewnętrznej pamięci danych. Każda z tych pamięci może znajdować się w uruchamianym systemie bądź może być wypożyczona z zasobów emulatora.

LITERATURA

1/ INTEL MCS-48 Family of single-chip microcomputers
USER's manual

2/ A.Pluta, Z.Poznański, J.Szynka
EMU - 48 - MAŁY SYSTEM URUCHOMIENIOWY DLA RODZINY MIKROKOMPUTERÓW JEDNOUKŁADOWYCH MCY 7848/35
Mikroprocesorowa Szkoła Zimowa 1983

Mgr Jerzy Grabowski
Sr[illegible]iskowe Centrum Obliczeniowe
CYFRONET
Instytut Energii Atomowej
05-400 Swierk-Otwock

Adres Prywatny:
ul. W.Czumy [illegible] m 21
01-355 Warszawa

# PL-80: STRUKTURALNY ASEMBLER DLA MIKROPROCESOROW I-8080. DEFINICJA I REALIZACJA.

## 1.0. ASSEMBLERY ST[illegible]URALNE

- Pojęcie assemblera strukturalnego
- Reguły nakładane na definicje i realizacje
- Własności
- Zastosowania
- Zalety i wady użycia assemblera strukturalnego
- Niekt[illegible] istniejące assemblery strukturalne

## 2.0. DE[illegible]NICJA PL-80

- Struktura blokowa
- Nagłówek programu
- Definicje stałych: stałe liczbowe, znakowe i adresowe
- Deklaracje zmiennych w pamięci: proste i tablicowe
- Rejestry lokalne procedury
- Deklaracja procedury
- Przechowywanie rejestrów przy wywoływaniu i powrocie: możliwość kontroli
- Procedury funkcyjne: wykorzystanie rejestrów do przekazywania wartości funkcji

## 3.0. PROCEDURY

- Parametry formalne procedury wołane przez wartość, adres zmiennej, adre[illegible]rocedury i pobierane "ręcznie"
- Wyk[illegible]tanie stosu na parametry aktu[illegible]e procedury: możliwość rekursji

4.0. CZESC INSTRUKCJI

4.1. WYRAŻENIA OBLICZANE

- Wyrażenia obliczane jedno- i dwubajtowe
- Parametry wyrażeń obliczanych
- Operatory wyrażeń jako odwzorowanie rozkazów arytmetycznych i logicznych procesora
- Beznawiasowy zapis wyrażeń obliczonych: przykłady zapisu i znaczenia

4.2. WYWOŁANIA PROCEDUR

4.3. INSTRUKCJE STRUKTURALNE

- Ogólne reguły zapisu instrukcji strukturalnych i ich zagnieżdżenia
- Lokalność rejestrów przy wchodzeniu do i wychodzeniu z treści instrukcji strukturalnej
- Instrukcja wyboru /CASE/
- Warunki logiczne: relacje i obliczane wyrażenia w nich stosowane, testy bitów stanu procesora, odstępstwa od reguły pełnej kontroli i świadomości programisty tworzonego przy kompilacji kodu
- Instrukcja warunkowa /IF/
- Instrukcja pętli /LOOP/ i instrukcje przerywania pętli /EXIT/ oraz nawrotu /NEXT/
- Instrukcja pętli "powtarzaj, o ile ..." /WHILE/
- Instrukcja pętli "powtarzaj, aż ..." /REPEAT/
- Uwagi o kontroli przez programistę oszczędności kodu generowanego przez kompilator

4.4. INSTRUKCJE DODATKOWE

- Instrukcja powrotu z procedury
- Instrukcje operacji na bitach stanu
- Instrukcje kontroli systemu przerwań

5.0. UWAGI REALIZACYJNE

- Przykłady tworzonego kodu dla obliczanych wyrażeń, odwołań do zmiennych prostych i parametrów formalnych procedur, instrukcji strukturalnych
- Wywoływanie procedur - parametrów formalnych
- Ogólne problemy realizacji: wybór języka i systemu do realizacji.

Dr inż. Andrzej Hildebrandt
Instytut Łączności
04-894 Warszawa, ul. Szachowa 1
zam.
( ) 594 Warszawa ul. Bruna 26 m 68

# C H I L L - JEZYK PROGRAMOWANIA SYSTEMOW TELEKOMUNIKACYJNYCH; POSTEPY PRAC NAD KRAJOWA IMPLEMENTACJA JEZYKA

## Wprowadzenie

CHILL /CCITT High Level Language/ jest języki programowania wysokiego poziomu opracowanym przez Międzynarodowy Komitet Doradczy do Spraw Telegrafii i Telefonii /CCITT/. Jego definicja znajduje się w Zaleceniu Z200 1 umieszczonym w tzw. Żółtej Księdze CCITT - wydawnictwie zawierającym komplet opracowanych przez tę organizację zaleceń. Język ten, tworzony z zamysłem stosowania go sterowanych programowo urządzeniach telekomutacyjnych, okazał się jednak na tyle ogólny, że może być stosowany do tworzenia dowolnych systemów pracujących w czasie rzeczywistym.

W chwili obecnej istnieje na świecie kilkanaście implementacji CHILLa, a kilka dużych, znajdujących się w handlu systemów komutacyjnych jest oprogramowanych w tym języku.
CHILL doczekał się dwóch, bardziej sformalizowanych do opisu zawartego w Z.200, opisów języka. /2/ /3/. Tłumaczenie broszury /Introduction to CHILL" zawierającej podstawowe informacje o programowaniu w tym języku dostępne jest w Instytucie Łączności /4/. Na przedniej, IV Szkole Mikroprocesorowej przedstawione były mechanizmy współzależności w języku CHILL /5/.

Pod względem własności użytkowych, CHILL zbliżony jest do Ady- istnieje szereg opracowań porównujących te języki.

## 0 - CHILL - początkowy etap wdrażania CHILLa

Prowadzone od tego czasu w Instytucie Łączności prace zaplanowane zostały w ten sposób, aby najpierw został wdrożony

podzbiór pełnego języka CHILL /otrzymał on nazwę O-CHILL/, a następnie, w zależności od potrzeb i rozwoju sytuacji dalsze jego rozszerzenie lub też CHILL kompletny. Spodziewamy się, że O-CHILL wraz z kompilatorem spełnią następujące zadania:

- umożliwią wkrótce rozpoczęcie szkolenia potencjalnych użytkowników CHILLa,
- umożliwią /w ograniczonym stopniu/ tworzenie użytecznych programów w CHILLu,
- staną się narzędziami do opracowywania kolejnych wersji kompilatorów wykonywanych metodą sznurowania,
- powiększą doświadczenie zespołu opracowującego kompilatory, dzięki czamu możliwe będzie uniknięcie niektórych błędów przy opracowywaniu kolejnych ich wersji.

O-CHILL jest językiem sekwencyjnym /pozbawionym środków programowania współbieżnego/, poziomem zaś odpowiada językowi PASCAL. Jego opis znajduje się w opracowaniu wewnętrznym IŁ /6/.

Już w trakcie prac nad O-CHILLem CCITT przyjął następującą, dość tolerancyjną, definicję kompilatora CHILLa:

"Kompilator jest kompilatorem CHILLa wtedy i tylko wtedy, gdy wszystkie programy akceptowane przez ten kompilator są zgodne z definicją CHILLa zawartą w Zaleceniu Z.200. Każdy kompilator CHILLa musi być wyposażony w wykaz tych elementów, które nie są akceptowane przez kompilator, pomimo ich zgodności z Z.200".

Dzięki temu, dość niespodziewanie, nasz kompilator O-CHILLa uzyskał także miano kompilatora CHILLa.

## Sposób w tworzeniu kompilatora

Na przyjęty przez nas sposób wytworzenia kompilatora znaczny wpływ miały będące do naszej dyspozycji komputery i kompilatory języków wysokiego pozio.

Koncepcja wytworzenia kompilatora opracowana przez dr inż. Ewę Ochmańską, a zawarta w dwóch opracowaniach wewnętrznych /7/ i /8/ zilustrowana jest na rys.1. Zgodnie z tą koncepcją

proces wytwarzania użytecznego w praktyce kompilatora składa się z trzech następujących faz.

Faza 1. W fazie tej wykorzystuje się komputer R32 i istniejący kompilator języka PASCAL. Napisany przez nas w PASCALu kompilator krzyżowy zostaje przetranslowany przez kompilator PASCALa i zainstalowany w maszynie R32 /kompilator na rys.1/. Może on tłumaczyć programy napisane w języku O-CHILL na język pośredni - kod maszynyumownej VM. Następnie nasz kompilator pierwotny /postać źródłowa/ zostaje przepisany ręcznie z PASCALa na O-CHILL, co nie stanowi większego problemu, jeśli pamiętamy, że O-CHILL był tak zaprojektowany, aby był możliwie zgodny z PASCALem. Otrzymany z przepisania kompilator /postać źródłowa/ zostaje przetranslowany przez kompilator , w wyniku czego otrzymujemy kompilator . Kompilator jest zatem programem zapisanym w kodzie maszyny umownej VM i po zainstalowaniu go na maszynie umownej VM mógłby tłumaczyć programy napisane w języku O-CHILL na język pośredni VM. Zainstalowanie go na maszynie VM nie jest jednak wykonalne z prostego powodu, że maszyna taka nie istnieje. Otrzymany w fazie 1 półprodukt doprowadza się zatem do postaci nadającej się do przekopiowania go z maszyny R32 do maszyny SM3.

Faza 2. Faza ta wykonywana jest na maszynie SM3 i polega na przystosowaniu przekopiowanego z maszyny R32 kompilatora do pracy na maszynie SM3. Używa się do tego celu tylko standardowy makroasembler MACRO-11 oraz konsolidator /oznaczone M+K na rys.2/. Jest to możliwe dzięki dobraniu odpowiedniej postaci języka pośredniego VM - czytelnej dla makroasemblera wyposażonego w odpowiednio przygotowaną bibliotekę makrodefinicji. Otrzymany w ten sposób kompilator zostaje zainstalowany na maszynie SM3. Jego jedyną wadą jest to, że produkuje programy wynikowe w mało użytecznym /chwilowo/ kodzie maszyny umownej VM.

Faza 3. W fazie tej następuje wytworzenie środków, które przystosowałyby kod VM produkowany przez kompilator do wykonywa-

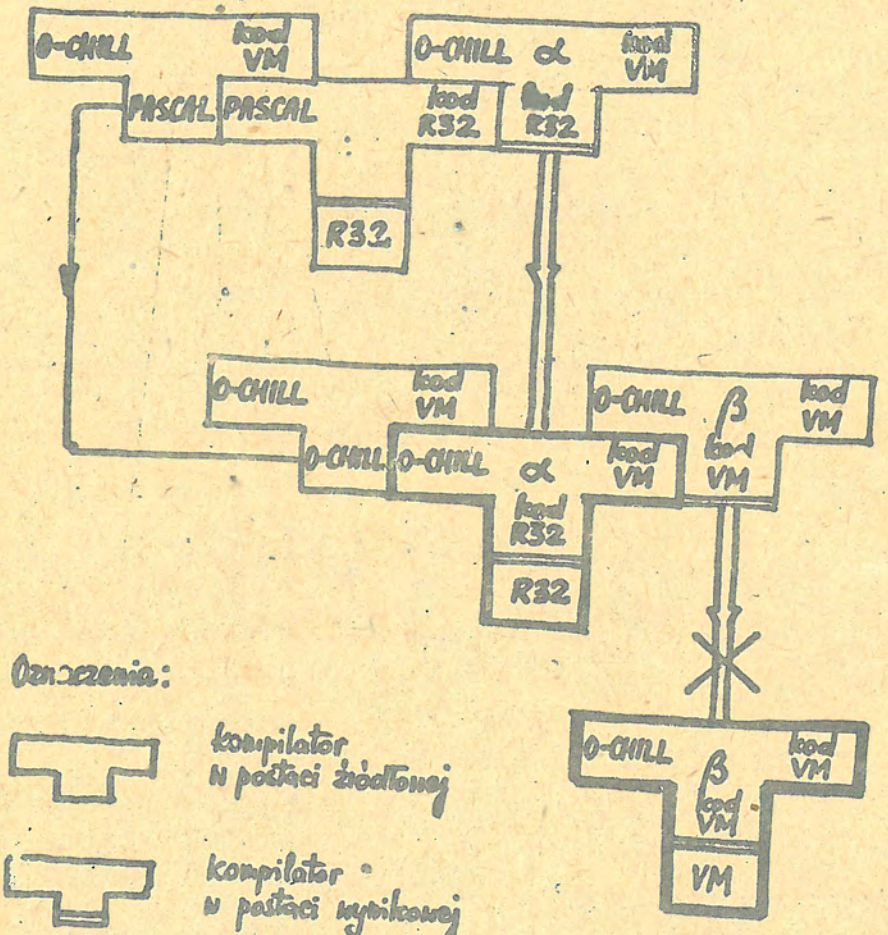
nia go na maszynie docelowej. Jeśli maszyną docelową jest znów SM3 to stosuje się procedurę zbliżoną do tej, która była stosowana w fazie 2 /przetworzenie przy użyciu makroassemblera i konsolidatora/. Jeśli maszyną docelową ma być mikroprocesor 8080 może to być wyspecjalizowany translator /oznaczony T na rys. 3a/. Inne rozwiązanie polega na umieszczeniu na maszynie docelowej emulatora maszyny umownej VM /rys. 3b/.

Przewiduje się, że pierwsza wersja otrzymanego w powyższy sposób kompilatora będzie uruchomiona w bieżącym roku. Kolejnym zamierzeniem jest opracowanie języka i kompilatora zawierającego mechanizmy współbieżności.

Zakończenie BIBLIOGRAFIA

Autor pragnie wyrazić w tym miejscu podziękowanie tym wszystkim osobom, które przyczyniają się do kontynuowania prac nad CHILLem. Wśród nich znajduje się Dyrekcja Zakładów TELKOM-ZWUT, które częściowo finansują nasze prace. W szczególności gorące podziękowania należą się drowi Andrzejowi Blinkiewiczowi ze ZWUTu za jego popracie dla CHILLa we wszelkich nadarzających się po temu okazjach.

1. CCITT Yellow Book. CCITT High Level Language
2. P.Haff, D.Bjoerner: CHILL Formal Definition. Dansk Datamatic Center, Lyngby, Denmark, 1981
3. P.Branquart, G.Louis, P.Wodon: An Analytical Description of CHILL, the CCITT High Level Language, Springer-Verlag, 1982
4. Wprowadzenie do języka CHILL. Wydawnictwo wewnętrzne Instytutu Łączności, Warszawa 1983
5. A.Hildebrandt, M.Porada: Mechanizmy współbieżności w języku CHILL. IV Szkoła Mikroprocesorowa, Łódź 1983
6. Kompilator O-CHILLa - projekt techniczny. Opracowanie wewnętrzne Instytutu Łączności, Warszawa 1983
7. E.Ochmańska: Projekt wdrożenia w minikomputerze SM3 języka programowania CHILL przeznaczonego dla central telekomunikacyjnych. Opracowanie wewnętrzne IŁ, Warszawa 1979
8. E.Ochmańska: Opracowanie kompilatora języka CHILL. Sprecyzowanie zadania. Opracowanie wewnętrzne Instytutu Łączności Warszawa 1982




Oznaczenia:

kompilator w postaci źródłowej

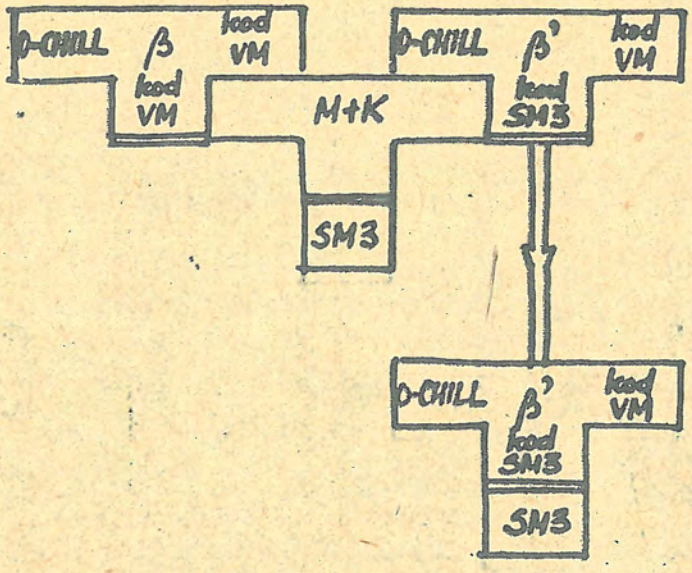
kompilator w postaci wynikowej

kompilator zainstalowany na maszynie A

przepisywanie kompilatora

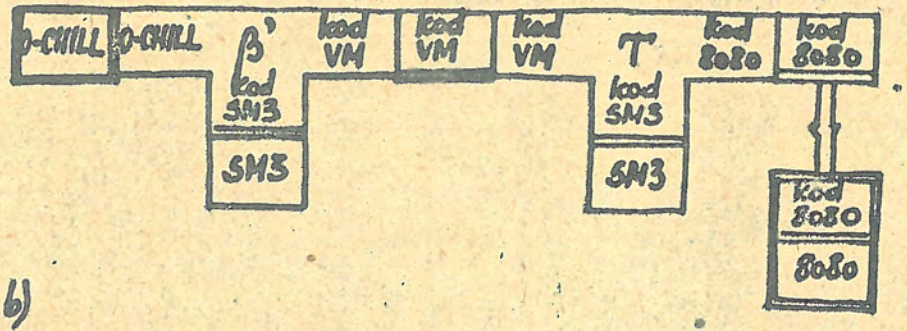
inicjalizowanie kompilatora

Rys. 1. Koncepcja wytworzenia kompilatora. Faza 1

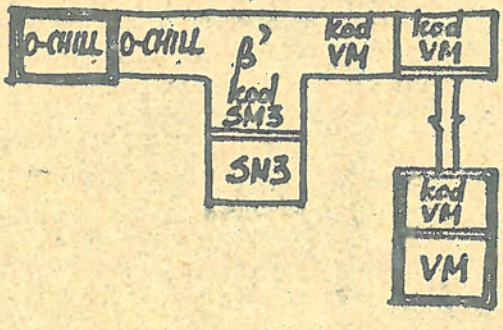



Rys. 2. Koncepcja wytworzenia kompilatora. Faza 2

a)




b)




Rys. 3. Użytkowanie kompilatora przy użyciu środków wytworzonych w fazie 3

Mgr inż. Roman Jeliński
Instytut Łączności O/Gdańsk
Zakład Telegrafii

# JĘZYK PROGRAMOWANIA CSDL DLA ELEKTRONICZNYCH CENTRAL TELEKOMUNIKACYJNYCH ZE STEROWANIEM ROZPROSZONYM

/streszczenie/

## Informacje wstępne

Język CSDL opracowano w celu wytworzenia oprogramowania systemu elektronicznych central telegraficzno-teleinformatycznych /ECTT/ akonstruowanych w Instytucie Łączności Oddział Gdańsk. Język uwzględnia podstawowe wymagania stawiane przed oprogramowaniem systemów telekomunikacyjnych, a mianowicie:

- duża współbieżność oprogramowania;
- silne uwarunkowanie czasowe;
- wysoka niezawodność systemu;
- daleko posunięta modularność /funkcjonalna oraz technologiczna/;
- przenoszalność oprogramowania.

Język CSDL spełnia również zalecenia notacji SDL rekomendowanej przez komitet CCITT. Przy projektowaniu języka wykorzystano idee zawarte w następujących językach programowania: język C, EDISON, CHILL oraz SDL. Obecna wersja języka jest rozwinięciem koncepcji opracowanej wcześniej w oparciu o makrorozwinięcia asemblera minikomputera PDP-11.

## Charakterystyka języka CSDL

Przy definiowaniu języka zakładano istnienie dedykowanego systemu operacyjnego ECTT.OS tworzącego wirtualną maszynę komutacyjną. Maszyna ta jest zorientowana na wykonanie i zarządzanie dużą ilością procesów sterujących pracą centrali ECTT.

Komunikacja między procesami oparta jest o koncepcję komunikatów. Oprócz powszechnie znanych typów wartości, w języku CSDL wprowadzono następujące typy specjalne:

- standardowy typ strukturalny PORT;
- typ wymuszeniowy STIM;
- typ opisu stanu procesu STATE;
- typ odsyłacz POINT.

W języku istnieje możliwość definiowania procesów oraz procedur. Wywoływanie procedur odbywa się na zasadach ogólnie przyjętych. Kreowanie, iniojowanie i zaniechanie procesów wykonują standardowe procedury języka. W języku zawarto również proste mechanizmy przekazywania i sprawdzania informacji diagnostycznych.
W wyrażeniach mogą występować operatory specjalne /operator na adresach, odsyłaczach, stanach procesu itp./ odnoszące się do specjalnych obiektów języka CSDL.
Istnieje również możliwość tworzenia sieci komunikacyjnych pomiędzy procesami za pomocą tzw. kanałów zarządzanych przez system operacyjny.
Struktury danych można deklarować z rezerwacją lokacji na stosie w pamięci dynamicznej oraz w przestrzeni programu /w pamięci statycznej/.
W procedurach oraz procesach można deklarować dane lokalne "widzialne" tylko w ich wnętrzu. Procesy są obiektami dynamicznymi - pojawiającymi się i znikającymi w zależności od potrzeb. Procesy deklaruje się jako obiekty zamknięte, posiadające swoje dane oraz swój program przebiegu. Proces posiada również obiekty "widzialne" w jego toczeniu tzw. porty procesu. Porty przeznaczone są do komunikacji procesu z jego otoczeniami /z innymi procesami/. Wysyłanie i przyjmowanie komunikatów z portów /do portów/ realizują standardowe procedury języka CSDL.
Struktura systemu w języku CSDL jest strukturą płaską. System dzieli się na moduły określające zakres widzialności obiektów. Moduł może zawierać deklaracje obiektów lokalnych /"widzialnych" tylko wewnątrz modułu/ oraz deklaracje obiektów eksportowanych/

"widzialnych" wewnątrz modułu oraz w jego bezpośrednim otoczeniu/. Moduł może również importować obiekty zadeklarowane w innych modułach. W modułach może wystąpić deklaracja: typu, zmiennej, procedury oraz procesu.
Grupy modułów tworzą tzw. bloki funkcjonalne /segmenty systemu/. Bloki funkcjonalne dysponują wszystkimi zasobami wymaganymi przez obiekty w modułach /dane, procesy, procedury, kanały, porty, kolejki procesów itp/. Bloki funkcjonalne kreowane są w czasie montażu systemu /konsolidacji/. W trakcie działania systemu komunikacja pomiędzy blokami realizuje się za pomocą przesyłania tzw. zleceń i odpowiedzi. Blok funkcjonalny, po przyjęciu zlecenia, funkcjonuje własnym "życiem" zgodnie z opisem sterowania zawartym wewnątrz bloku. Bloki funkcjonalne dobrze nadają się do "rozpraszania" sterowania i zasobów w systemie.

## Implementacje języka CSDL.

Jak wspomniano na wstępie, translator języka zakłada istnienie dedykowanego systemu operacyjnego ECTT.OS. System operacyjny realizuje podstawowe funkcje zarządzania procesami i zasobami. Translator produkuje postać wynikową będącą ciągiem modułów assemblera procesora docelowego /obecnie minikomputera SM-3 i SM-4/. W najbliższej przyszłości planuje się implementację translatora dla mikroprocesorów INTEL-8080 oraz Z80.

Mgr inż. Tadeusz S.Kozłowski
Przemysłowy Instytut Elektroniki NPCP "UNITRA-CEMI"
00-241 Warszawa ul. Długa 44/50
zam:
02-904 Warszawa, ul.Bernardyńska 11 m 43

# SYSTEM STEROWANIA REAKTORAMI DO PROCESOW DYFUZJI I EPITAKSJI.

## 1. Wprowadzenie

Cieplno-chemiczne procesy technologiczne w produkcji struktur półprzewodnikowych, takie jak dyfuzja czy epitaksja, wymagają precyzyjnego sterowania szeregiem czynników technologicznych /przepływ gazów, temperatura/ oraz zmniejszenia do minimum udziału operatora w celu przede wszystkim uniknięcia pomyłek /szczególnie istotne w epitaksji/ i wyeliminowania czynności normalnych mających wpływ na wyniki procesu /np. wprowadzanie wsadu z płytkami krzemowymi - dyfuzja/. Opracowany mikroprocesorowy system sterowania spełnia te wymagania dając dodatkowo niezbędne możliwości rozwoju urządzeń i technologii tych procesów.

## 2. Podstawowe funkcje realizowane przez system sterowania

### A. Regulacja temperatury

W aktualnej wersji systemu regulacja temperatury wykorzystywana jest przez system sterowania reaktorem do dyfuzji. Reaktor ten /w uproszczeniu/ jest w postaci poziomej rury kwarcowej umieszczonej wewnątrz cewki grzejnej. Cewka grzejna składa się z trzech niezależnych sekcji. Sterowanie mocą odbywa się dla każdej sekcji niezależnie poprzez układ sterujący kątem odcięcia tyrystora. System umożliwia regulację temperatury na dwóch poziomach /~700°C i~1200°C/ z dokładnością lepszą niż ± 0,5°C, wg algorytmu PD i PID, niezależnie dla każdej sekcji, jak również, w systemie MASTER-SLAVE, gdzie MASTER stanowi sekcja środkowa /w stanach

awaryjnych przewidziane jest również sterowanie temperaturą w otwartej pętli/. Celem regulacji temperatury jest uzyskanie w rurze kwarcowej tzw. "sfery płaskiej" o długości ok. 1 metra i różnicy temperatur nie większej niż $\pm$ 0,5$^{o}$C oraz rozgrzewu i studzenia wg programowanej charakterystyki prostoliniowej.

B. Sterowanie urządzeniem załadowczo-wyładowczym /ładowarką/

System umożliwia sterowanie urządzeniem załadowczo-wyładowczym /tylko dla procesów dyfuzji/ w zakresie ruchu w przód, w tył oraz ruchu oscylacyjnego. Parametry tego ruchu /poza oscylacyjnym/ tzn. prędkość i położenie są programowane.

C. Sterowanie rozrządem gazów i analogowymi regulatorami przepływu

System umożliwia otwieranie i zamykanie elektromagnetycznych zakresów odcinających oraz kontrolę ich stanu. Dla procesów dyfuzyjnych zrealizowana jest wersja ze sterowaniem maksymalnie 8 zaworami, a dla procesu epitaksji - maksymalnie 32 zaworami. Ponadto system przesyła informację o zadanym przepływie gazów do analogicznych regulatorów przepływu i pobiera z nich informację o rzeczywistym przepływie. Ilość regulatorów przepływu waha się w zależności od typu procesu od 3 do 8. System sterowania reaktorem do epitaksji umożliwia ponadto programowane narastanie przepływu wg jednej z dwóch stałych charakterystyk.

D. Obsługa pulpitu operatora

Pulpit operatora zawiera klawiaturę /od 45 do 60 klawiszy/, diody świecące oraz zespół wyświetlaczy cyfrowych i alfanumerycznych. System umożliwia operatorowi z klawiatury m.in. zaprogramowanie procesu technologicznego, odczyt wprowadzonych danych, zapisanie programu technologicznego w zewnętrznej pamięci EPROM, odczyt parametrów procesu bieżącego, zainicjowanie wybranego procesu technologicznego w reżimie automatycznym, prowadzenie procesu w reżimie ręcznym, odczyt informacji o awarii i szereg operacji pomocniczych.

E. Przechowywanie programów technologicznych w zewnętrznej wymiennej pamięci EPROM.

Panel mikrokomputera /dyfuzja/ lub pulpit operatora /epitaksja/ wyposażone są w dostępne z zewnątrz gniazdo /podstawka TEXTOOL/ do pamięci EPROM. System umożliwia zaprogramowanie i odczyt tej zewnętrznej pamięci dzięki czemu użytkownik ma możliwość założenia "biblioteki" programów technologicznych w postaci zbioru układów EPROM typu 2708 lub 2716. W pamięci EPROM typu 2716 można umieścić co najmniej 2 programy technologiczne, a maksymalnie 9.

F. Automatyczny restart procesu technologicznego po powrocie napięć zasilających.

Właściwość tę zapewnia system sterowania poprzez wykorzystanie buforowanej pamięci RAM /zasilanej z akumulatora/ oraz pakietu wczesnej detekcji zaniku napięcia sieci. W buforowanej pamięci RAM /2 Kbajty/ rezyduje m.in., program technologiczny aktualnie wykonywany. Wielkość i typ pamięci RAM oraz typ akumulatora wyznaczają maksymalny czas przechowywania danych bez zniekształceń. Aktualnie system umożliwia przechowywanie w buforze danych nie dłużej niż 15 minut. W momencie podjęcia produkcji układów pamięci CMOS czas ten będzie można znacznie zwiększyć. Specyfika procesu epitaksji krzemu ogranicza wykorzystanie automatycznego restartu po powrocie napięć zasilających tylko do przypadku obsługi chwilowych zaników napięcia /poniżej 1 sekundy/

G. Wydruk historii procesu technologicznego na drukarkę DZM-180

Funkcję tę realizuje tylko system sterowania reaktorem do epitaksji krzemu. Wydruki robione są na zakończenie pojedyńczego etapu procesu i dają informację o wszelkich odchyłkach parametrów poza dopuszczalne granice o stanach awaryjnych, o czasie trwania etapu i inne.

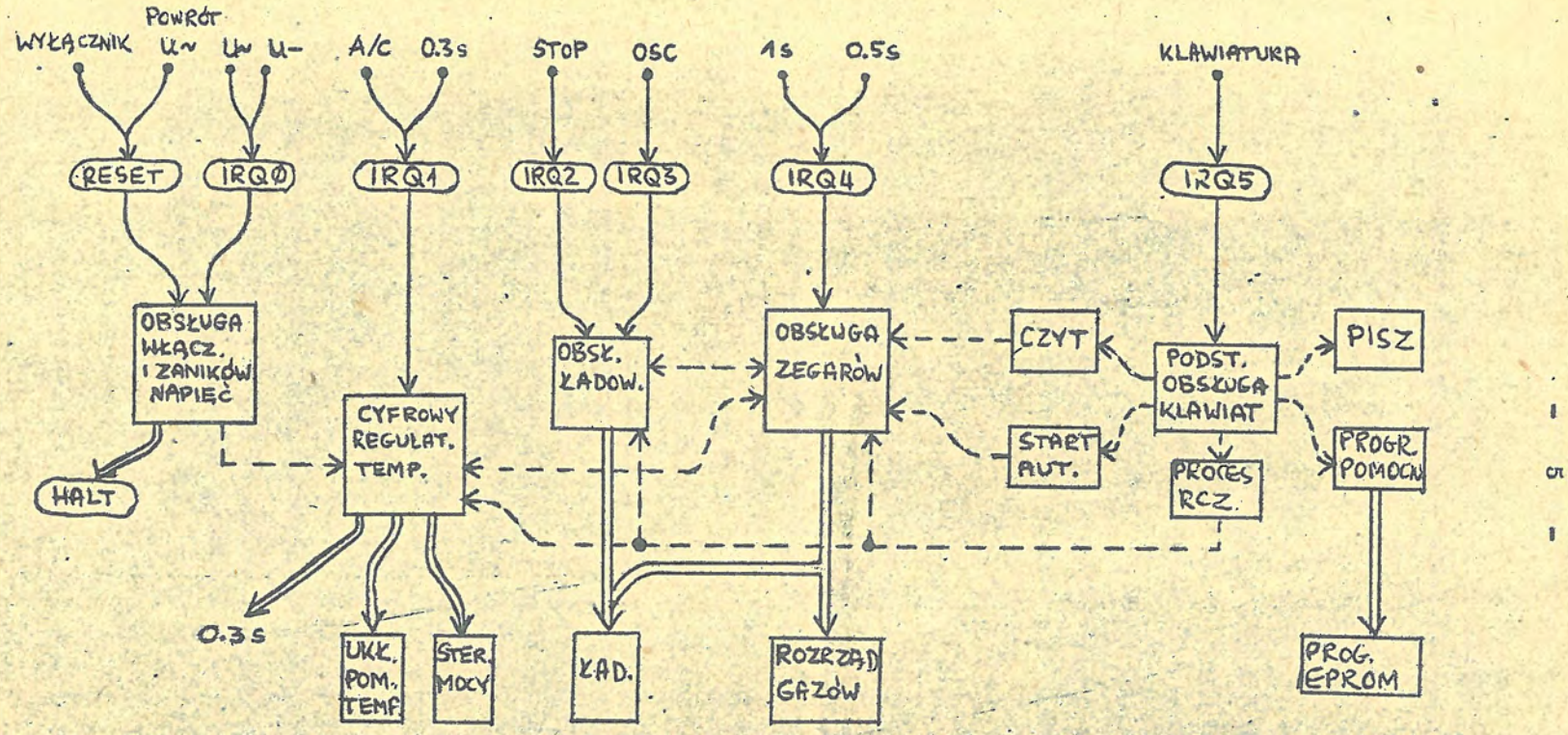
## 3. Architektura systemu mikrokomputerowego

Omawiany system sterowania zrealizowany jest w oparciu o mikroprocesor INTEL 8080A. W swojej podstawowej konfiguracji wykorzystuje on system pakietowy MSA-80 /pakiet CPU, RAM, EPROM, przetworniki A/C, C/A, pakiet programatora/. Ponadto opracowano szereg pakietów specjalizowanych nie wchodzących w skład tego systemu. Są to m.in. pakiety sterowania zaworami, pakiet zegara, pakiety obsługi zaników napięć i restartu, pakiet sterowania mocą grzejną, pakiet sterowania ładowarką, pakiet 16-bitowego przetwornika A/C i inne.
System sterowania reaktorem do dyfuzji będzie przystosowany do współpracy z systemem centralnego monitorowania, rejestracji i sterowania wieloma pracami do dyfuzji.

## 4. Oprogramowanie systemu sterowania

Bazuje ono na systemie przerwań, który zrealizowany jest w oparciu o kontroler 8259. Kontroler ten zaprogramowany jest na 8 poziomów przerwań, gdzie poziom IRQ0 ma najwyższy priorytet. Rysunek 1 pokazuje organizację oprogramowania systemu sterowania reaktorem do dyfuzji.
W oprogramowaniu systemu wyróżniamy 8 podstawowych modułów i 2 opcjonalne, zależne od typu procesu. Są to:

- moduł obsługi włączenia wyłącznikiem sieciowym i obsługi przerwań od znaków napięć stałych i napięcia sieci;
- moduł podstawowej obsługi klawiatury;
- moduł obsługi przerwań zegarowych oraz startu/restartu procesu technologicznego w reżimie automatycznym;
- moduł obsługi przerwań od urządzenia załadowczo-wyładowczego;
- moduł pisania programów technologicznych;
- moduł czytania programów technologicznych i parametrów bieżącego procesu;
- moduł obsługi reżimu pracy ręcznej;
- moduł programów pomocniczych realizujących m.in. funkcje:




Rys. 1. Organizacja oprogramowania systemu sterowania reaktorem do dyfuzji

- wprowadzania programu technologicznego do pamięci stałej EPROM
- manipulowania zaworami w rozrządzie gazów
- kontroli diod pulpitu
- wprowadzania i czytania parametrów modyfikowalnych regulatora temperatury /opcjonalnie/

- moduł cyfrowego regulatora temperatury /opcjonalny/
- moduł wydruku historii procesu technologicznego na drukarkę DZM-180

Oprogramowanie systemu sterowania reaktorem do dyfuzji zajmuje ok. 24 Kbajtów, system sterowania reaktorem do epitaksji krzemu ok. 20 Kbajtów. W obu systemach wykorzystuje się pamięć RAM o wielkości 6 Kbajtów, w tym 2 Kbajty pamięci buforowanej.

5. Organizacja mechanizmu sygnalizacji błędów i awarii

W systemie wyróżnia się 3 typy błędów:

1. błąd operatora nie mający wpływu na stan urządzenia lub procesu technologicznego powstający przeważnie na etapie programowania procesu technologicznego, próby czytania programu procesu nie istniejącego itp.
2. błąd typu "niedopuszczalna operacja" powstający m.in. w momencie próby startu procesu przy niespełnionych warunkach początkowych, próby niewłaściwej ingerencji w proces itp.
3. błąd typu "awaria" wykrywany przez system, powstały w wyniku niesprawności urządzenia technologicznego.

Błąd 1 sygnalizowany jest zapaleniem się odpowiedniej diody na pulpicie operatora.
Błąd 2 sygnalizowany jest zapaleniem się odpowiedniej diody i sygnałem dźwiękowym.
Błąd 3 sygnalizowany jest zapaleniem się diody o odpowiednim znaczeniu dla typu awarii i sygnałem dźwiękowym. Błędy te /ich szczególna specyfikacja/ zapamiętywane są w buforze awarii. Operator może odczytać tę specyfikację wywołując z klawiatury odpowiedni program.

Bibliografia

1. Dokumentacja eksploatacyjna systemu MSA-80

Doc. dr inż. Jacek Kamler
Centr.Ośr.Elektr.Sprzętu Powsz.Użytku
Warszawa ul. Ratuszowa 11 tel. 199001 w. 744
zam.
00/367 Warszawa ul. Kopernika 8/18 m 78
tel. 266015

# TELETEXT I MIKROKOMPUTERY
# NOWE SRODKI MASOWEJ POPULARYZACJI INFORMATYKI

## 1. Rys.hisotryczny

Teletext jest to system wyświetlania na ekranie odniornika telewizyjnego dodatkowych informacji alfanumerycznych i graficznych przesyłanych w postaci sygnału cyfrowego dodanego do istniejącego sygnału telewizyjnego.

System ten powstał w Anglii we wczesnych latach siedemdziesiątych. Pierwszy z patentów zgłoszony był 5-go lutego 1972r. przez P.Rainigera, J.D.B.Millara i F.G.Parkera z BBC. Nosił on tytuł "The Transmission of Alphanumeric Data by Television". W następnych latach w wyniku prac prowadzonych w laboratorium BBC i telewizji niezależnej IBA stworzono i uzgodniono wspólny system, którego parametry opublikowano w styczniu 1976r. jako "Specifications of Standards for Broadcast Teletext Signals".

W latach siedemdziesiątych opracowano również we Francji podobny system o nazwie ANTIOPE różniący się na tyle od Teletextu, że późniejsze wysiłki, mające na celu ujednolicenie obu systemów, spełzły na niczym.

Obecnie istnieje więc na świecie kilka milionów odbiorników dostosowanych do systemu Teletext, ale równocześnie w szeregu krajów podjęto decyzję o wyborze systemu ANTIOPE lub systemów z nim związanych.

W Polsce w Warszawie nadawany jest od kilku lat próbny sygnał w systemie Teletext przestawiający kilka stron kontrolnych.

2. Działanie systemu Teletext

Do przesyłania sygnału cyfrowego wykorzystano w systemie Teletext wolne linie, czyli okresy odchylania poziomego, występujące w sygnale telewizyjnym podczas impulsu wygaszania odchylania pionowego. Wstępnie ustalono, że będą to linie nr 17 i 18 oraz 330 i 331. W czasie trwania tych linii jest przesyłany sygnał cyfrowy w kodzie NRZ z szybkością przesyłania 9,9375 Mb/s. Informacje teletextowe są wyświetlane na ekranie w postaci 24 wierszy po 40 znaków w każdym.
Są to znaki alfanumeryczne albo uproszczone znaki graficzne. Każdy znak jest określony przez jeden bajt składający się z siedmiu bitów informacyjnych i bitu parzystości.

Sygnał przesyłany w czasie jednej linii odpowiada jednemu wierszowie na ekranie. W okresie linii przesyła się 45 bajtów. Pierwsze 5 bajtów każdego wiersza zawierają kolejno: bajt 1 i 2- sekwencję rozbiegową, 3 - kod rozpoznawczy, 4 i 5 - numer magazynu i numer wiersza. Pierwszy wiersz każdej strony jest wierszem tytułowym i zawiera numer strony. Bajty z numerami magazynów, wierszy i stron są dodatkowo zabezpieczone czterema bitami nadmiarowymi w kodzie Hamminga. Umożliwia to korekcję pojedyńczych błędów i odrzucenie informacji przy błędach wielokrotnych.

Wiersze są nadawane kolejno co 20 ms /jeżeli wykorzystywana jest tylko jedna linia/. Do przesłania jednej strony tekstu potrzeba więc 20 ms x 24 = 480 ms, lub 240 ms przy wykorzystywaniu dwóch linii. Magazyn zawierający 100 stron tekstu byłby więc przesyłany przez 24 sekundy i tyle trzebaby, w najbardziej niekorzystnym przypadku czekać na pojawienie się wybranej strony. Jest to długie oczekiwanie i stanowi to podstawowy mankament systemu Teletext.
Radykalnym rozwiązaniem byłoby wykorzystanie wszystkich linii sygnału telewizyjnego. Możnaby wówczas przesyłać 600 stron na sekundę. Niezbędny do tego jest wolny kanał telewizyjny.

Odbioru sygnału Teletextu dokonuje się za pomocą dekodera połączonego z układem odbiornika telewizyjnego. Uproszczony schemat blokowy takiego odbiornika przedstawiono na rys. 2.

Sygnał do dekodera teletextu jest pobierany z detektora wizji. Sygnał wyjściowy dekodera jest doprowadzany do układu interface. Układ ten działa jako przełącznik analogowego sygnału wizyjnego i sygnału wytwarzanego w dekoderze Teletextu. Działanie przełącznika jest na tyle szybkie, że możliwe jest umieszczanie napisów teletextowych na obrazie telewizyjnym np. w celu tłumaczenia nia obcojęzycznego dialogu lub dostarczania informacji dla słabosłyszących

3. Działanie dekodera Teletextu

Schemat podstawowych działań wykonywanych w dekoderze przedstawiono na rys. 2. Z wejściowego sygnału wizyjnego jest wydzielany sygnał cyfrowy. Sygnał ten jest następnie przekształcony na 8-mio bitową postać równoległą. W sygnale tym następuje sprawdzanie parzystości i kodu Hamminga, a następnie ewentualna korekcja pojedyńczych błędów. Przesyłane numery stron są sprawdzane z numerem strony wybranym przez widza do wyświetlania. W przypadku uzyskania zgodności numerów strona zostaje wpisana do pamięci. Treść pamięci jest wyświetlana na ekranie po przetworzeniu w generatorze na sygnały znaków i zamianie na postać szeregową. Sygnał zegara jest wytwarzany w dekoderze i synchronizowany z odbieranym sygnałem cyfrowym. Wszystkie przebiegi czasowe potrzebne do synchronizacji odbiornika, obsługi pamięci i wyświetlania są otrzymywane przez podział częstotliwości 6 MHz wytwarzanej przez wewnętrzny generator i synchronizowanej z odbieranym sygnałem synchronizacji odbiornika.

4. Rozwój konstrukcji dekodera Teletextu.

Stosowane dotychczas dekodery Teletextu zawierały wyspecjalizowane układy scalone. Przykładem może być dekoder z układami serii SAA 5000 firmy Philips. Dekoder taki zawiera 10 układów scalonych. Zaopatrzony jest w pamięć jednej strony /1 kbajt/ umożliwia wyświetlanie 96 znaków alfanumerycznych i grafiki. Pole znaku stanowi matrycę 7x5 punktów/ 14x10 dzięki systemowi zaokrąglania /dla znaków alfanumerycznych i matrycę 2x3 elementy dla grafiki. Znaki są wyświetlane w jednym rodzaju alfabetu

/np. angielskim lub rosyjskim/. Dekoder ten pojawił się na rynku w r. 1978 i jest dostosowany do Teletextu tzw. pierwszego poziomu. W kolejnych latach wprowadzono szereg udoskonaleń systemu czyli kolejnych coraz wyższych poziomów. Trzymano się przy tym zasady kompatybilności w dół co oznacza, że sygnał Teletextu o wyższym poziomie wytwarza prawidłowy obraz w odbiorniku z dekoderem niższego poziomu. Wprowadzono w tym celu np. niewidoczne na ekranie wiersze /Ghost rows/, których treść stanowią rozkazy umożliwiające dekoderom wyższych poziomów spełnianie bardziej złożonych funkcji. Na ekranie odbiornika w miejscu takiego niewidocznegowiersza nie ma tekstu, a odbiornik z dekoderem niższego poziomu nie wykonuje tych dodatkowych czynności.

W r. 1983 w firmie Philips zakończono opracowanie nowej generacji układów do dekodera, programowanych za pomocą mikrokomputera. Dzięki zastosowaniu mikrokomputera powstaje szereg nowych możliwości, a mianowicie:

- sterowanie w jednolity sposób licznymi już układami cyfrowymi stosowanymi w odbiorniku telewizyjnym. Ilustruje to np. rys.1, na którym pokazano wykorzystanie jednego mikrokomputera do wybierania stacji telewizyjnych, regulacji funkcji analogowych odbiornika /jasność konstrast itd./, oraz do sterowania dekoderem Teletextu
- dostosowanie w prostszy sposób właściwości odbiornika do wymagań różnych klientów /np. zaprogramowane kanały telewizyjne, inne alfabety teletextowe itp./ poprzez zmiany w programie, a nie przez bardziej kosztowne zmiany układowe.
- rozbudowę systemu informatycznego, którego centralną część stanowić może odbiornik telewizyjny ze sterowaniem mikrokomputerowym i jednolitym systemem transmisji danych.

5. Konstrukcja dekodera Teletextu ze sterowaniem mikrokomputerowym.

Uproszczony schemat dekodera drugiej generacji pokazano na rys. 3. Zawiera on dwa wyspecjalizowane układy. Pierwszy z nich to procesor wizyjny typu SAA5230. Układ ten służy do wydzielania

sygnału cyfrowego, regeneracji zegara i synchronizacji sygnału 6 MHz. Drugi układ nazwany EURO CCT /Computer Controlled Teletext/ typu SAA 5240 spełnia pozostałe funkcje przedstawione na rys. 2, a ponadto zawiera bardzo rozbudowany programowany system wybierania danych. W porównaniu z poprzednim modelem dekodera uproszczono bardzo znacznie interface z pamięcią umożliwiając jednocześnie dołączenie pamięci zawierającej do 8 stron telewizyjnych.

Podobnie jak i poprzednio dekoder jest sterowany sygnałem wizyjnym, a sygnał wyjściowy jest kierowany do interface. Dekoder jest połączony z mikrokomputerem liniami SDA /dane/ i SCL /zegar/. Zadaniem mikrokomputera jest między innymi programowanie rejestrów układu EURO CCT. W układzie tym znajduje się dziesięć rejestrów z możliwością zapisu. Treść ich określa stan pracy dekodera a więc odbieranie sygnału w jednej linii lub we wszystkich liniach specjalnie przeznaczonego do tego celu kanału telewizyjnego, rodzaj synchronizacji, rodzaj międzyliniowości, rodzaj transmisji /7 bitów + bit parzystości czy 8 bitów/ itp. Jedenasty rejestr zawiera jeden bajt danych z dowolnie wybranego miejsca pamięci stron. Dane w tym rejestrze mogą być zarówno czytane jak i zapisywane poprzez szynę transmisji między szynę transmisji między mikrokomputerem a dekoderem Teletextu.

6. Mikrokomputer

Mikrokomputer stosowany w sprzęcie masowym jakim jest odbiornik telewizyjny musi być tani i prosty. Firmą Philips wykorzystuje do tego celu jeden z mikrokomputerów rodziny MAB 8400. Stanowią one uproszczoną odmianę mikrokomputera 8048 umieszczoną w obudowie 28-mio nóżkowej. Najważniejszą ich cechą, w tym zastosowaniu, jest możliwość bezpośredniej współpracy z wielonadajnikowym systemem dwukierunkowej szeregowej transmisji. Mikrokomputer jest w tym celu wyposażony w wyspecjalizowane wyjście/wejście szeregowe dla zegara, zaś dla danych wykorzystuje się jedną z linii portów. Mikrokomputery tej rodziny zawierają pamięć RAM /64 lub 128 bajtów/, pamięć ROM /1-4 kbajtów/

oraz 20 linii wej./wyj. Liczba instrukcji wynosi 80 i stanowi podgrupę listy instrukcji mikrokomputera 8084.

7. System transmisji

Przesyłanie danych między poszczególnymi elementami systemu, którego jednym z członów jest opisany poprzednio dekoder Teletextu jest dokonywane za pomocą dwuprzewodowej, dwukierunkowej szyny wykorzystywanej w systemie pracy z wieloma nadajnikami nadrzędnymi /master transmitter/. W systemie firmy Philips, nazywanym $I^2C$ /Inter IC bus/, jedna z linii służy do przesyłania danych, druga zaś zegara. Organizacja szyny umożliwia współpracę wielu nadrzędnych nadajników, odbiorników /slave receiver/, oraz podporządkowanych nadajników /slave transmitter/. Funkcja "wired AND" jaką spełnia linia danych umożliwia przeprowadzenie arbitrażu przy równoczesnym nadawaniu. Pierwszeństwo osiąga nadajnik transmitujący bajt o najniższej wartości binarnej. Linia zegara, również spełniająca "wired AND" umożliwia osiągnięcie zgodności częstotliwości i fazy zegarów wszystkich urządzeń dołączonych do szyny. Częstotliwości zegara mogą wynosić od 0 do 100 KHz. Dane są przesyłane w postaci 8-mio bitowych bajtów, z potwierdzeniem odbioru po każdym bajcie.

8. Dalsze możliwości rozbudowy domowego systemu wizyjno-informatycznego.

Dzięki zastosowaniu dwukierunkowej szyny o możliwościach pracy z wieloma nadrzędnymi nadajnikami system, którego centralną część stanowi odbiornik telewizyjny, może być nieomal dowolnie rozszerzany. Można go wykorzystywać do zdalnego sterowania takimi urządzeniami i systemami jak magnetowid czy telewizja kablowa, poprzez wykorzystanie odpowiednio dostosowanego układu zdalnego sterowania. Potrzebne są do tego układy mogące współpracować z opisanym poprzednio systemem transmisji. Przykładem takim może być opracowany przez firmę Philips układ sterujący strojeniem i funkcjami analogowymi odbiornika /system z rys.1./ o oznaczeniu SAB 3035.

Prócz podanych wyżej możliwości sterowania przyjęcie jednolitego systemu transmisji umożliwia współpracę różnych systemów przesyłania danych. Jednym z nich jest opisany wyżej system Teletext. Zamiast przesyłania kolejnych stron do wyświetlania na ekranie można przesyłać dane lub programy /telesoftware/ do komputera domowego lub do gier telewizyjnych. Wykorzystując pamięć 4 łączonych stron można przesłać program do 4 kbajtów. Dekoder może być wtedy programowo przestawiony na odbiór bajtów 8-mio bitowych /a nie 7 + parzystość/. Programy te mogą być przez szynę danych wyprowadzane na zewnątrz z pamięci dekodera i wprowadzane do domowego mikrokomputera lub zapisywane na drukarce czy też taśmie magnetofonowej.

Dalszą możliwością jest współpraca z systemem Viewdata /lub jego odmianami/ tj. z systemem przesyłania poprzez łącze telefoniczne danych wytwarzających alfanumeryczne lub graficzne obrazy na ekranie odbiornika telewizyjnego. Ilość takich stron, pochodzących z pamięci dużej maszyny, jest teoretycznie nieograniczona, a czas dostępu do nich wynosi pojedyńcze sekundy.

9. Podsumowanie.

Pierwotnym przeznaczeniem systemu Teletext było dostarczanie aktualnych, często potrzebnych w życiu codziennym informacji takich jak pogoda, godzina, aktualności, sport czy wyniki gier liczbowych. W takim systemie liczba stron nie może być zbyt duża jeśli czas dostępu ma się zawierać w rozsądnych granicach. Poważną konkurencję stanowi tu system telefoniczny /Viewdata/ z jego wielką liczbą stron i krótkim czasem dostępu. Pojawia się przeto pytanie czy próby wykorzystania Teletextu jako systemu informatycznego w sensie przesyłania telesoftware itp. są słuszne. Trudno w tej chwili na to odpowiedzieć. Opisywany jednak wyżej sposób transmisji oprogramowania wzbudza jednak wątpliwośc

Stosunkowo długie programy byłyby przesyłane bez żadnego zabezpieczenia nadmiarowego. A należy pamiętać, że w kanałach telewizyjnych zakłócenia są rzeczą powszechną. Z drugiej strony jednak w polskich warunkach, przy ogólnie znanej sytuacji z telefonami, możliwość przesyłania oprogramowania i danych różnego rodzaju przez łatwo dostępny kanał telewizyjny jest warta poważnego potraktowania.

Bibliografia

- Broadcast Teletext Specification. Sept. 1976 Wydane przez BBC IBA, BREMA
- Amos S.W. : A History of CEEFAX . March 1978 BBC Engineering
- The Teletext Decoder. Philips Technical information 050
- Computer controlled teletext . Philips Technical publication 112
- Introduction to digital systems. Serial data buses. Philips Data handbook . Integrated circuits. Part 3 Sept 1982
- Bilip A.: System Teletext . RadioElektronik 1983r. Nr.9
- Janczewska E.: Gazeta telewizyjna. Przegląd Telekomunikacyjny 1982r. Nr.4.

Podpisy rysunków

Rys.1 Odbiornik telewizyjny z dekoderem Teletextu

Rys.2 Schemat funkcjonalny dekodera Teletextu

Rys.3 Uproszczony schemat dekodera Teletextu sterowanego za pomocą mikrokomputera.

Dr Artur Krępski
Instytut Podstaw Informatyki
Polska Akademia Nauk
00-901 Warszawa PKiN
skr.poczt.22

# "PASCAL MT+" - DIALEKT JEZYKA PASCAL DLA MIKROKOMPUTEROW"

**Streszczenie**

Omówiono rozszerzenia języka Pascal MT+ w porównaniu ze standardem Pascala, system SPP /zawierający m.in. specjalny edytor oraz kompilator Pascal MT+/, a także doświadczenia z wykorzystania tych narzędzi na mikrokomputerze Z-80 /marki ALTOS/.

Wnioski dotyczą oceny, prognoz rozwoju i potrzeby tego typu narzędzi dla sprzętu mikrokomputerowego.

## 1. Wstęp

System SPP /ang. Speed Programming Package/ jest zestawem kilku bardzo prostych, a jednocześnie wygodnych i wydajnych narzędzi wspomagających pracę użytkownika języka Pascal MT+.

Kompilator języka Pascal MT+ oprócz tego, że akceptuje pełny standard języka Pascal, dostarcza takie bogactwo różnego rodzaju rozszerzeń i narzędzi, że dostępne są wszelkie zasoby mikrokomputera i systemu operacyjnego /CP/M/ - nie ma więc potrzeby używania języka assemblera.

System SPP razem z kompilatorem Pascal MT+ stanowi bardzo efektywne narzędzie do programowania szerokiej klasy problemów na mikrokomputerach o stosunkowo małych wymaganiach co do ich konfiguracji. Dostępny jest pod systemami operacyjnymi CP/M i MP/M dla komputerów zbudowanych na bazie mikroprocesorów Intel-8080 i Z-80 wyposażonych w pamięć operacyjną od 52 Kbajtów /do 64 Kb/, klawiaturę i monitor ekranowy oraz przynajmniej jeden przewijak dyskietek /miękkich dysków/.

Dokumentacja w postaci krótkiego /70 stron/ opisu systemu SPP dla użytkownika /2/ oraz podręcznika Pascala MT+ /200 stron - 3 / stanowi rzadki przykład solidności - zawiera informacje w pełni wystarczające do instalowania /a nawet modyfikowania/, użytkowania i uczenia /liczne przykłady!/ systemu SPP.

Stosunkowo wysoki koszt systemu SPP /łącznie z kompilatorem Pascal MT+ - ok. 500 $ USA/ spowodowany jest dobrą jakością, dużym zapotrzebowaniem i brakiem liczącego się konkurenta.

System rozpowszechnia firma Digital Research /Kalifornia, USA/: liczbę instalacji ocenia się w tysiące. Można przewidywać /por. np. 4 / dalsze intensywne rozpowszechnianie się systemu SPP z kompilatorem Pascal MT+ i duży wzrost ilości oprogramowania użytkowego /aplikacyjnego/ wytwarzanego z jego pomocą.

## 2. System SPP

### 2.1. Charakterystyka użytkowa

Użytkownik steruje działaniem systemu SPP i poszczególnymi narzędziami wybierając jedną z kilku możliwości wyszczególnionych na ekranie monitora w postaci tzw. menu - rys. 1 /ang. soreen oriented menu-driven/.

```
                Speed Programming Package

Options:        E/ dit
                R/ eformat
                S/ yntax check
                V/ariable check
                X/ eg
                D/ ir
                L/ ink
                F/ ast compile
                Q/ uit

Command?
```

Rys. 1. Menu systemu SPP

System SPP dostarcza następujących narzędzi /w nawiasach podano ich jednoliterowe kody/: edytor /E/, program formatujący teksty pascalowe /B/, program zastawiający listę identyfikatorów w programie pascalowym uporządkowaną alfabetycznie /V/, analizator składniowy /S/ oraz kompilator /F/ języka Pascal MT+.

Dodatkowo umożliwia się korzystanie z następujących komend systemu operacyjnego bez konieczności opuszczania systemu SPP: wyświetlanie na ekranie monitora indeksu plików na zadanej dyskietce /D/, połączenie /L/ i wykonanie /X/ programu w Pascalu.

Wreszcie komenda Q służy do opuszczania systemu SPP i powrotu do systemu operacyjnego.

Zasadą działania systemu SPP jest operowanie od razu na całości tekstu pascalowego; na początku działania systemu tekst ten jest wczytywany z pliku źródłowego do specjalnego bufora w pamięci operacyjnej, a na końcu - jest wpisywany na ten sam plik /stara wersja tekstu jest przy tym kopiowana automatycznie na dodatkowy plik/.

## 2.2. Edytor

Edytor systemu SPP służy do konwersacyjnego operowania na tekście źródłowym /znajdującym się w całości w buforze w pamięci operacyjnej/ za pomocą komend uaktywnianych kluczem kontrolnym /CONTRL/ i klawiszem klawiatury z jednoliterowym kodem.

Ekran monitora służy jako "okienko" przesuwające się po tekście źródłowym - wszelkie zmiany na ekranie są automatycznie wprowadzane do tekstu w buforze.

Edytor w jednym z trzech trybów: wstawiania tekstu, zamieniania i usuwania; może operować całymi liniami, słowami /ciągami znaków oddzielonych spacjami/ oraz pojedynczymi znakami.

Dostępne są komendy do poruszania się kursorem po ekranie /we wszystkich kierunkach, na początek i koniec ekranu/ i po całym tekście /liniami i stronami będącymi wielokrotnościami ekranu, na początek i koniec tekstu, do linii o zadanym numerze/.

Specjalna komenda /z własnym menu/ umożliwia wyświetlanie na ekranie monitora zawartości dowolnego pliku i jej watawienie w dane miejsce w operowanym tekście, zaznaczenie początkowej i końcowej linii bloku tekstu źródłowego i wypisanie go na zadany plik oraz dostarcza informacji o stanie wykorzystania zasobów systemu SPP /np. zajętość bufora, długość tekstu mierzoną w liniach itd./.

Dodatkowe komendy służą do wyszukiwania i zamieniania zadanych napisów - ciągów znaków /automatycznie w całym tekście, zadaną liczbę razy i pojedynczo z każdorazowym potwierdzeniem przez użytkownika/, adjustację tekstu, wstawianie pustych linii, kopiowanie z miejsca na miejsce całych fragmentów tekstu źródłowego.

## 2.3. Charakterystyka funkcjonalna

System SPP jest zakodowany w języku Pascal w postaci jednego programu podzielonego na część rezydentną /stale przechowywaną w pamięci operacyjnej/ oraz 9 nakładek /ładowanych do pamięci w przypadku odwołania się do nich/ realizujących poszczególne narzędzia; część rezydenta zajmuje 6 Kb, nakładki razem /w tym głównie kompilator Pascal MT+/ - 103 Kb.


Rys. 2. Schemat podziału pamięci dla systemu SPP

Długość bufora na tekst źródłowy zależy od konfiguracji sprzętu i systemu operacyjnego; wynosi od 7 Kbajtów - znaków /np. dla systemu KP/M/ do 19 Kb /system CP/M w minimalnej konfiguracji i 64 Kb pamięci operacyjnej/.

Uważam, że stosunkowo silne ograniczenia na długość tekstu źródłowego przetwarzanego przez system SPP stanowią jego zaletę - wymuszają na użytkowniku podział tekstu programu na rozsądnie małe segmenty, co może skłaniać również do logicznej modularyzacji programu.

Z drugiej strony edytowanie tekstu w całości znajdującego się w pamięci operacyjnej /przy stosunkowo wolnej pracy urządzeń wejścia/wyjścia/ jest bardzo efektywne. Odpowiedź systemu na większość komend /nawet tych związanych z wyszukiwaniem ciągów znaków/ jest szybka, niemal natychmiastowa.

## 3. Kompilator Pascal MT+

### 3.1. Akceptowany język - rozszerzania standardu Pascala

Kompilator Pascal MT+ akceptuje pełny język Pascal zgodny ze standardem ISO 1 ; nieistotne odstępstwa są następujące: niedozwolona instrukcja skoku w procedurach rekurencyjnych, zignorowanie błędu w instrukcji wyboru, gdy wartość wyrażenia wyboru nie odpowiada żadnej z etykiet wyboru oraz prawidłowa wartość funkcji eof dla pliku nietekstowego tylko wówczas, gdy ostatni element pliku jest jednocześnie na końcu sektora dyskietki.

Pascal MT+ rozszerzono o wiele konstrukcji dodatkowych /nie zawsze jednak zgodnie nie tylko z duchem Pascala, ale i regułami semantycznymi standardu/. Celem tych rozszerzeń jest albo udostępnienie zasobów mikrokomputera i systemu operacyjnego /co w praktyce pozwala unikać użytkowania języka assemblera/, albo też dążenie do dostarczenia konstrukcji umożliwiających efektywniejsze kodowanie /niż można by to zrobić stosując tylko standard języka/.

Tutaj zestawiamy najważniejsze rozszerzenie /niektórych użyto w przykładzie §4/, bardziej zwracając uwagę na rodzaj użytego mechanizmu językowego niż na jego opis. Są one następujące:

1/ wprowadzono pojęcie modułu /bl[illegible] ujęty w nawiasy module i modend/ i umożliwiono ich oddzielną kompilację; wszystkie obiekty globalne modułu są automatycznie eksportowane, a w celu importu - ich typ należy poprzedzić słowem external, np.

```
var i, j, k: external integer;
external procedure p/x: real/;
```

2/ jest możliwe opisanie struktury nakładkowej programu /dodatkowa konwencja zadawania grupy nakładkowej procedury w nawiasach kwadratowych przy deklaracji procedury - por. §4/ oraz dynamiczne ładowanie procedury w czasie wykonania programu /ang. chaining/;

3/ wprowadzono dodatkową instrukcję exit - wyjście z danej procedury/funkcji oraz procedurę hlt - wyjście do systemu operacyjnego;

4/ wprowadzono dodatkowe typy standardowe - bajt i słowo /oznaczane odpowiednio przez byte oraz word/, a także standardową funkcję konwersji wrd /z wartości dowolnego typu porządkowego na typ słowo/;

5/ wprowadzono dodatkowy typ standardowy string /np. var s: string 50 wraz z całym zestawem standardowym procedur i funkcji;

6/ wprowadzono dodatkowe funkcje i procedury - bez żadnych gwarancji bezpiecznego używania służące operacjom na bitach, bajtach i wektorach bajtów /znaków/; odpowiadają one zwykle pojedynczym rozkazom maszynowym /są więc bardzo efektywne/;

7/ wprowadzono dodatkowe procedury i funkcje operujące na plikach; udostępniające bezpośrednio w Pascalu pliki o dostępie swobodnym, odpowiadające operacjom na zbiorach danych w systemie operacyjnym /m.in. otwieranie, zamykanie, badanie wyników operacji na pliku/, wczytywanie i zapisywanie wartości w formacie szesnastkowym, odpowiadające operacjom kanałowym;

wprowadzono procedury i funkcje /m.in. addr, sizeof/ udostępniające atrybuty obiektów programu pascalowego;

9/ umożliwiono zapisywanie stałych w programie w notacji ósemkowej i szesnastkowej;

10/ wprowadzono operacje logiczne na całych słowach /a nie tylko na wartościach logicznych/;

11/ rozszerzono składnię instrukcji wyboru - po słowie else można zadać instrukcję, którą należy wykonać zawsze wtedy, gdy wartość wyrażenia wyboru nie odpowiada żadnej z wartości etykiet wyboru.

## 3.2. Dodatkowe narzędzia programowania

Kompilator Pascal MT+ ma wbudowanych szereg narzędzi, których celem jest ułatwienie pracy przy tworzeniu programów. Są one następujące:

1/ parametry kompilacji - zadawane w instrukcjach stanowiących rozszerzenie komentarza, np. /*SE+,X+ */; służą do sterowania procesem kompilacji np. w celu zwiększenia efektywności kodu wynikowego, włączenia sprawdzań generowanych przez kompilator, warunkowej kompilacji itp.

2/ parametry kompilatora - wyszczególniane w komendzie systemu operacyjnego uaktywniającej kompilator /mtplus/ np. MTPLUS A:TEST.SRC SB PX A; służą opisaniu zasobów systemu w czasie działania kompilatora;

3/ program łączący - łączy do 32 plików wynikowych /w tym także modułów wybieranych z bibliotek/ tworząc jeden moduł ładowalny; umożliwia zadawanie struktury nakładkowej zarówno kodu programu, jak i segmentów danych;

4/ dekoder programów wynikowych na postać asembleropodobną - dodatkowo kompilator Pascal MT+ pozwala dokonywać wstawek w języku asemblera /przygotowano do tego specjalną, prostą składnię rozpoznawaną przez kompilator/ za pomocą dodatkowej instrukcji inline;

5/ bibliotekarz - program służy dwóm celom:

- przekształcaniu sekwencyjnych plików wynikowych w bibliotekę modułów ładowalnych /jak np. PASLIB/,
- przekształcaniu wynikowych plików tworzonych przez kompilator Pascal MT+ na format akceptowany przez inne programy łączące /np. L80 czy Link80/;

6/ program do odpluskwiania - w postaci oddzielonej nakładki kompilatora Pascal MT+; umożliwia wyświetlanie wartości zmiennych i wyświetlanie zawartości tablicy symboli /nazw procedur i nazw zmiennych/ wykonywanie programu pojedynczymi instrukcjami połączone z ich trasowaniem /w postaci numeru linii liczonych osobno w każdym module/, wykonywanie programu od początku lub zadanego punktu przerwania /można również kasować zdefiniowane uprzednio punkty przerwania wykonania programu/, modyfikowanie zawartości pamięci oraz instruowanie użytkownika; korzystanie z tego programu wymaga kompilacji programu pascalowego z własnym parametrem D kompilacji.

## 3.3. Charakterystyka funkcjonalna

Kompilator Pascal MT+ jest dwuprzebiegowy /I przebieg - analiza składniowa, II przebieg - generowanie kodu/, składa się z części rezydentnej i sześciu nakładek - razem 52 Kbajty. Wymaga 36 Kb na kod kompilatora w pamięci operacyjnej oraz przynajmniej 7 Kb na tablicę symboli /maks. 17 Kb/, a ponadto pliku roboczego na kod pośredni kompilowanego programu /około 1/3 rozmiaru pliku źródłowego/.

W sumie, zaleca się przynajmniej 48 Kb pamięci operacyjnej /oprócz pamięci na system operacyjny/ na kompilowanie średnich programów pascalowych.

Program łączący /nie jest nakładkowany/ wymaga 11 Kb pamięci operacyjnej na swój kod, a całą resztę dostępnej pamięci zużywa na tablicę symboli i kod łączonych programów.

Faza łączenia wymaga dostępu do systemowych plików zawierających moduły wynikowe dołączane do programu wynegerowanego przez kompilator. Koniecznym takim plikiem jest biblioteka PASLIB

:/zawiera 16,8 Kb kodu i 2,3 Kb danych/, z której wybiera się tylko te moduły, do których wystąpiły odwołania z programu wynikowego /zwykle od 2 do 6 Kb/. Inne pliki są dołączane w całości /nie są bibliotekami/ w zależności od rodzaju u: tych rozszerzeń języka Pascal MT+.

Program wynikowy składa się z modułów generowanych oddzielnie dla każdej procedury, funkcji i programu głównego w porz.dku zgodnym z kolejnością ich części instrukcyjnych. Ograniczenie na kod modułu dla procedury czy funkcji wynosi 2 Kb.

Szybkość kompilacji wynosi średnio 2000 b/min - przykładowo kompilacja zestawu modułów źródłowych ok. 1 Mb wynosiła ok. 8 godz. pod systemem CP/M.

Zawartość kodu jest dość dobra np. dla programu z §4 wynosi 887 bajtów /625+116+146/ oraz 204 bajty /200+2+2/ danych.

Schemat podziału pamięci w czasie wykonywania programów pascalowych /rys. 3/ może być elastycznie modyfikowany przez użytkownika za pomocą odpowiednich parametrów kompilacji i programu łączącego.


Rys. 3. Podział pamięci w czasie wykonania programu

Skład, którego początek wskazuje standardowa zmienna SYSTEM występuje tylko w przypadku użycia zmiennych wskaźnikowych; podobnie stos na segmenty danych procedur wywoływanych rekurencyjnie /jego koniec wskazuje zmienna SFP/ występuje tylko wtedy, gdy włączony jest parametr kompilacji S /dozwolone użycie procedur i funkcji rekurencyjnych/.

## 4. Przykład

Celem zilustrowania sposobu działania w systemie Pascal MT+ w praktyce i zaprezentowania jego możliwości językowych, w niniejszym rozdziale podano przykładowy program LICZ.

Celem programu LICZ jest wczytanie ze standardowego pliku wejściowego /klawiatura/ jednolitego kodu operacji /W,Z,S/ i jej wykonanie na pliku tesktowym LICZ.TST znajdującym się na dyskietce na przewijaku B. Operacje te oznaczają: liczenie wierszy /W/, liczenie znaków /Z/ oraz zatrzymanie programu /S/. Wszystkie informacje program wpisuje na standardowy plik wyjściowy /monitor ekranowy/.

Cały program /w celach dydaktycznych/ został podzielony na trzy moduły, oddzielnie kompilowane: część rezydentną - program główny /LICZ/ i dwie nakładki /LICZ_LW, LICZ_LZ/. /Zmienna plikowa f /typu text/ jest zasobem dzielonym między modułami - zadeklarowana jest globalnie w module LICZ i jest importowana przez obie nakładki, tj. moduły LICZ_LW i LICZ_LZ.

Niżej podajemy teksty źródłowe poszczególnych modułów.

```
module LICZ_LW;
  var f: external text;          /* obiekt importowany */
procedure Lw;
  /* licz wiersze na pliku tekstowym f */
  var l: integer;                /* licznik wierszy */
  begin
    l:= 0;
    While not eof /f/ do
    begin
      l:=l+1; readln/f/.
    end;
     writeln/' liczba wierszy na pliku źródłowym: ',l/
    end;
modend.
```

Tabl.1. Moduł źródłowy LICZ_LW

```
program LICZ f,input,output/;
/* licz wiersze i znaki na pliku tekstowym f,        */
/* któremu jest przypisany zbiór danych B:LICZ,TST */
   var
     f: text;                  /* obiekty     */
     z: char;                  /* globalne    */
   external 1 procedure lw; /* zewnętrzna procedura w nakładce
                               Nr 1*/
   external 2 procedure lz; /* zewnętrzna procedura w nakładce
                               Nr 2*/
   begin
     writeln /' są dostępne operacje na pliku tekstowym P:LICZ.TST:'/
     writeln /'w - liczenie wierszy.'/;
     writeln /'z - liczenie znaków.'/;
     writeln /'s - stop.'/;
     assing/f,'B:LICZTST'/;
     repeat
       reset/f/;
       if ipresult   255 then
          begin
            /* otwarcie pliku B:LICZ.TST bez błędów */
            write /'Wprowadź znak: W, Z, S - '/; writeln;
            case z of
                 W   W   lw;
                 Z   z   lz;
            else
              if not /z in C S  s J / then
                 writeln /'Niedozwolony znak'/
              end /*case*/
          end
       else writeln /'Błąd przy otwarciu pliku B:LICZ.TST'/
     until z  inC S  s J;
   writeln /'koniec programu'/
end.
```

Tabl.2. Moduł źródłowy LICZ

```
module LICZ lz;
  var f: external text;          /* obiekt importowany */
  procedure lz:
  /* licz znaki na pliku tekstowym f */
    var l: integer;              /* licznik znaków */
  begin
    l:= 0;
    while not sof/f/ do
    begin
      while not eoln/f/ do
      begin
        l:= L+1; get/f/
      end
      readln/f/
    end;
    writeln /'Liczba znaków na pliku źródłowym: ',l/
end;
modend.
```

Tabl. 3. Moduł źródłowy LICZ_LZ

## 5. Wnioski

1/ System SPP jest wygodnym, prostym i efektywnym narzędziem pracy programisty /szczególnie przydatnym do przygotowania, zredagowania i kompilowania tekstów w języku Pascal/;

a/ jeśli zgodzić się z założeniami systemu SPP, to posiada on niewiele wad - przypadkowe /zupełnie nieczytelne/ skojarzenie kluczy kontrolnych z komendami edytora oraz przykre przerywanie pracy edytora /powrót do systemu operacyjnego/ przy próbie zapisania edytowanego tekstu bez uprzedniego zezwolenia na zapis na danej dyskietce,

b/ zwraca uwagę ogromna prostota i zwartość edytora w systemie SPP / o rząd zmniejsza w porównaniu np. z edytorem WordStar/. w pełni wystarczająca do edytowania tekstów programów pascalowych: stanowi to kolejną egzemplifikację znanej dewizy

"małe /i wyspecjalizowane/ jest piękne" spopularyzowanej w dziedzinie programowania przez autorów systemu UNIX,

c/ załamanie się pracy systemu SPP / i strata zawartości buforów/ zdarzają się raczej rzadko.

2/ Kompilator Pascal MT+ akceptuje pełny język Pascal zgodny ze standardem ISO; prócz tego dostarcza bogactwo różnorodnych rozszerzeń i dodatkowych narzędzi.

Nie wszystkie rozszerzenia są zgodne z duchem, a nawet składnią języka Pascal. Pozwala to z jednej strony na korzystanie z wszystkich zasobów mikrokomputera i systemu operacyjnego bezpośrednio w języku Pascal MT+ /i czyni zbędnym programowanie w języku assemblera/, ale z drugiej -ceną za to jest gwałtowny wzrost niebezpieczeństwa języka.

Rzeczywiście zasadne są częste zastrzeżenia nawet w podręczniku Pascala MT+, że jest to język tylko dla doświadczonych programistów - jeśli chce się tworzyć programy nie tylko efektywne, ale i niezawodne.

3/ można oczekiwać dalszego wzrostu popularności systemu SPP i kompilatora Pascal MT+, a nawet zaryzykuję stwierdzenie - tendencji do wzorowania się przez inne firmy mikrokomputerowe.

4/ Obecne oprogramowanie systemowe /systemy operacyjne CP/M i MP/M, Pascal MT+/ są wystarczające do wykonywania programów, ale nie do pracy nad ich pełnym przygotowaniem /projektowaniem i uruchamianiem/; szczególną uwagę zwraca niebezpieczeństwo użycia większości mechanizmów - rozszerzeń języka.

5/ Pożądany byłby, moim zdaniem, system o prostocie i efektywności podobnej do systemu SPP, wspomagający programistę na etapie projektowania oraz testowania /uruchamiania programów.

BIBLIOGRAFIA

1 Addymann A.M.: Specyfication for the computer programming language Pascal. Sacond draft proposal, Document ISO/TC 97/SC 5 N595, 1981

2 SpeedProgramming Package - User's Guide Rel. 5.2, Digital Research, Calif rnia USA, 1982

3 Pascal MT+ - User's Guide Rel. 5, MT MicroSystems Inc., New York, Lifeboat Associates, 1981

4 J.Pournelle, Tje Debate goes on - A Discussion of the future of microcomputer languages, August 1983, str. 312-328, Byte

---

ADA — na RIAD (IMM)

KMIKOWSKI - dostęp do SPD
il. pakietów w Pascalu maleje

Pascal - j. do uczenia

Mgr inż. Stanisław Nielicki
Środowiskowe Centrum Obliczeniowe
CYFRONET
Instytut Energii Atomowej
05-400 Swierk-Otwock

# SYSTEM WSPOMAGAJĄCY OPROGRAMOWANIEM MIKROKOMPUTERA INTEL-8080 ZREALIZOWANY NA MINIKOMPUTERZE PDP11/45

Chociaż na całym świecie następuje dynamiczny rozwój systemów mikrokomputerowych rozszerzonych o coraz doskonalszy sprzęt i oprogramowanie, to nadal są bardzo użyteczne systemy wspomagające tworzenie programów dla mikroprocesorów zrealizowane na innych komputerach.

Minikomputer PDP11/45 wyposażony w dyski, taśmy magnetyczne i pamięci kasetowe pracując pod systemem operacyjnym RSX11-M umożliwia wielu użytkownikom jednocześnie szybkie i wygodne redagowanie oraz przechowywanie zbiorów tekstowych system wspomagający oprogramowanie mikrokomputera pozwala na kompilację, rozbudowane testowanie i ładowanie programów wykonywanych przez mikroprocesor.

Opisany poniżej system powstawał stopniowo wykorzystując potrzeby i doświadczenia użytkowników minikomputera PDP11/45 w SCO-CYFRONET w Swierku.

System wspomagający przygotowanie programów dla mikroprocesora INTEL-8080 składa się z następujących programów:

1. Assembler relokatywny - R80
2. Konsolidator - L80
3. Symulator procesora - S80
4. Asembler strukturalny - P80

## Asembler i konsolidator

Asembler relokatywny umożliwia generowanie przesuwalnego kodu pośredniego, który po przetwarzeniu przez konsolidator ma postać standardowego kodu wynikowego zdefiniowanego przez firmę INTEL. Pozwala to na budowanie programów z niezależnie tłumaczonych modułów oraz używanie bibliotek podprogramów. Zbiór dyrektyw asemblera jest wzorowany na dyrektywach asemblera MACRO-11 /asembler PDP11/.

## Symulator procesora

Program symulatora umożliwia wykonywanie instrukcji programów przeznaczonych dla mikroprocesora oraz pozwala na pełną kontrolę przebiegu symulacji. Wydruki symulatora, ślady zmian zawartości rejestrów i pamięci, pułapki etc. pozwalają na uruchamianie złożonych programów nawet przez niezaawansowanych programistów. Symulator nie odtwarza przebiegów czasowych rzeczywistego mikroprocesora może jednak zliczać cykle symulowanego programu i informować o rzeczywistych czasach realizacji na oryginalnym sprzęcie.

## Asembler strukturalny

Asembler strukturalny zawiera w sobie chechy języka wysokiego poziomu /programowanie strukturalne/ oraz klasycznego asemblera m.in. bezpośredni dostęp do rejestrów procesora.

mgr inż. Krzysztof Plusczok
Zakład Systemów Automatyki
Kompleksowej PAN
44-100 Gliwice, ul. Bałtycka 5
tel: 31-08-11 w. 197
zam.:
41-902 Bytom pl.Kościuszki 12/19

# R T D S - 8

## Wstęp

Wprowadzenie do konstrukcji systemów cyfrowych układów scalonych dużej skali integracji, a głównie mikroprocesorów i współpracujących z nimi układów programowanych otwarło nie tylko nowe możliwość, ale i stworzyło nowe problemy. Między innymi pojawiły się problemy z uruchamianiem i testowaniem systemów aplikacyjnych zawierających tak złożone układy. "Klasyczne" narzędzia i metody okazały się mało efektywne w przypadku stosowania układów dużej skali integracji, w których brak np. bezpośredniego dostępu do stanów elementów wewnętrznych układu /np. do stanu rejestrów mikroprocesora/. Efektywne projektowanie i uruchamianie układów i systemów mikroprocesorowych wymaga stosowania specjalizowanych środków sprzętowych i programowych ułatwiających procesy projektowania i uruchamiania. Środki te nazywamy krótko systemami wspomagającymi. Przed środkami wspomagającymi konstrukcję systemów mikrokomputerowych stawia się zadania:

- projektowania, uruchamiania i testowanie części urządzeniowej systemu,
- projektowania, uruchamiania, testowania i weryfikacji części programowej,
- integracji urządz[illegible]owo-programowej uruchamianego systemu.

Stopień spełnienia tych wymagań może stanowić kryterium klasyfikacji systemów wspomagających. Według tego kryterium wśród systemów wspomagających wyróżniamy trzy podstawowe klasy:

- proste uniwersalne zestawy mikrokomputerowe, np. zestawy typu KIT pozwalające m.in. na budowanie w oparciu o nie systemów aplikacyjnych,
- systemy dla rozwoju oprogramowania i prowadzenia prac modelowych w konfiguracji systemu wspomagającego,
- systemy wspomagające konstrukcję oprogramowania i integrację urządzeniowo-programową systemu uruchamianego w oparciu o emulację układową.

Zastosowanie emulacji układowej, której idea polega na symulowaniu mikroprocesora systemu uruchamianego przez układ zawierający analogiczny mikroprocesor otoczony logiką umożliwiającą jego sterowanie i monitorowanie, pozwala na uruchamianie systemu mikroprocesorowego w:

- docelowej konfiguracji,
- rzeczywistych warunkach pracy,
- bez konieczności wprowadzania do struktury uruchamianej specjalnych środków potrzebnych jedynie w procesie uruchamiania.

Prezentowany systemu RTDS-8 /Real Time Development System for 8-bit microprocesors należy do trzeciej klasy systemów wspomagających według klasyfikacji wymienionej powyżej.

## Konfiguracja systemu RTDS-8

System RTDS-8 składa się z dwóch podstawowych bloków funkcjonalnych:

- mikrokomputera bazowego, który wraz z odpowiednim oprogramowaniem stanowi system wspomagający konstrukcję oprogramowania,
- uniwersalnego emulatora mikroprocesorów 8-bitowych.

Mikrokomputer bazowy to dwupakietowy /pakiet procesora i pakiet we/wy,/uniwersalny system mikrokomputerowy wyposażony w procesor 8085 lub 8080 i zawierający w wersji podstawowej 16 kB pamięci RAM oraz 8 kB pamięci EPROM.

Pakiet we/wy pozwala na współpracę RTDS-8 z:

- monitorem ekranowym TTY-CRT stanowiącym konsolę systemową /MERA-7952/,
- jednostką pamięci na dyskach elastycznych - pamięć masowa systemu /PLx45D/,
- drukarką mozaikową /DZM-180/,
- stacja taśmy papierowej /SPTP-3/,
- programatorem pamięci EPROM typu 2746,
- systemem komputerowym ze skromnym oprogramowaniem rozwojowym /poprzez łącze szeregowe/.

Oprogramowanie podstawowe systemu stanowi:

- MONITOR systemowy umieszczony w pamięci typu ROM,
- DYSKOWY SYSTEM OPERACYJNY z biblioteką programów usługowych.

MONITOR systemowy posiada standardowe dla tego typu programu możliwości rozszerzone o oprogramowanie sterujące pamięcią dyskową oraz o zlecenia: testowania PAO, ładowania systemu operacyjnego, formatowania dyskietek, programatora pamięci EPROM 2716, ekspansji PAO o pamięć emulowaną.
DYSKOWY SYSTEM OPERACYJNY jest jednoprogramowym systemem kompatybilnym z CP/M wersja 1.4. W podstawowej konfiguracji RTDS-8 może pracować CP/M w wersji 16 kB, oraz przy pracy bez emulatora - systemy do 32 kB. Ekspansję pamięci psiąga się poprzez dołączenie, jako rozszerzenia PAO, modułu pamięci emulowanej z bloku emulatora /16 kB/.
Biblioteka programów usługowych systemu zawiera:

- edytor tekstu /ED/,
- assembler dla mikroprocesorów 8080/8085 /ASM/,
- debugger /DDT/,
- systemowe programy organizacyjne /STAT, PIP, LOAD, DUMP, COPY, MOVCPM, SYSGEN, SUBMIT/,

oraz

- interpreter języka BASIC /TBASIC/,
- interpreter języka FORTH /FORTH/,

- programy sterujące emulacją,
- programy pomocnicze i testy.

Uniwersalny emulator układowy mikroprocesorów 8-bitowych, to układ oparty na oryginalnej metodzie forsowanego wykonywania instrukcji. Dla modułu mikrokomputera bazowego pakiety modułu emulatora stanowią układy peryferyjne o określonej strukturze portów we/wy i pamięci danych.

Uniwersalność emulatora systemu RTDS-8 wynika z programowego sterowania przebiegiem emulacji na poziomie cykli maszynowych oraz na skonfigurowaniu elementów emulatora wokół wyodrębnionej wewnętrznej magistrali mającej cechy szerokiej klasy magistral mikroprocesorowych. W wyniku tego emulator systemu RTDS-8 posiada stałą strukturę urządzeniową, a emulacja żądanego mikroprocesora wymaga przyłączenia właściwej sondy emulacyjnej i zainicjowania odpowiedniego programu sterującego. Przygotowano sondy emulacyjne dla mikroprocesorów 8085 i 8080.
Podstawowymi modułami emulatora są: sonda emulowanego mikroprocesora i adapter.

Moduł sondy jest układem zależnym od typu emulowanego mikroprocesora. Moduł ten emuluje sygnały mikrprocesora wyprowadzając je bezpośrednio do podstawki mikroprocesora w systemie uruchamianym.

Moduł adaptera realizuje przejście sygnałów z sondy na magistralę wewnętrzną /wymienioną wyżej/ o organizacji umożliwiającej współpracę z pozostałymi modułami emulatora wymienionymi poniżej.

Moduł emulacji pamięci zawierający 16 kB pamięci RAM zorganizowanej w 4 bloki po 4 kB. Bloki te mogą być dołączone do systemu uruchamianego po zadekladowaniu adresu początkowego i trybu dostępu do bloku /ROM, RAM/. Moduł komparatora stanów zawiera dwa analogiczne układy rozpoznające żądany stan magistrali adresowej w określonym cyklu maszynowym. Układy te pozwalają na realizację punktów zatrzymań programu wykonywanego w czasie rzeczywistym w trybie emulacji oraz na warunkowe określanie trybu śladowania przebiegu programu.

Moduł śladowania pozwala w czasie rzeczywistym na warunkowe zapamiętanie stanu 1024 cykli maszynowych. Deklaracja warunków śladowania i startu śladowania pozwala na selektywne śladowanie cykli.

Oprogramowanie systemu pozwala na prowadzenie pełnego procesu konstrukcji oprogramowania dla budowanego systemu w ramach RTDS-8. Edytor i assembler pozwalają na przygotowanie tekstu programu, kompilację i uzyskanie w pamięci dyskowej pliku z wynikową postacią programu. Użycie EMULATORA tzn. programu sterującego modułami uniwersalnego emulatora mikroprocesorów 8 bitowych pozwala na dialogowe prowadzenie procesu uruchamiania w konfiguracji tworzonego systemu. Zlecenia EMULATORA można podzielić funkcjonalnie na trzy podstawowe grupy:

- zlecenia pozwalające na przygotowanie odpowiedniej konfiguracji
- zlecenia pozwalające na przygotowania odpowiedniej konfiguracji pamięci emulowanej i załadowanie programu,
- zlecenia pozwalające na obserwowanie i modyfikowanie stanu zasobów na kontrolowany przebieg programu oraz obsługi śladowania przebiegu programu,
- zlecenia pozwalające na składowanie zawartości pamięci.

Szersze omówienie możliwości i konfiguracji systemu RTDS-8 można znaleźć w broszurze informacyjnej pt. "RTDS- Charakterystyka systemu", wydanej przez producenta oraz w czasopiśmie "Informatyka" 1/83.

Uwagi końcowe

System RTDS-8 opracowany w Zakładzie Systemów Automatyki Kompleksowej PAN /44-100 Gliwice, ul. Bałtycka 5 tel. 31-08-11/ został wdrożony do produkcji w Zakładach Urządzeń Komputerowych "MERA-ELZAB" 44-800 Zabrze, ul. Kruczkowskiego 19, tel. 72-20-21/. Plan rozwoju systemu RTDS-8 przewidują przygotowanie sond emulacyjnych dla mikroprocesorów typu 8048 i Z-80 przygotowanie wersji z pamięcią operacyjną 64kB, zastąpienie monitora ekranowego TTY-CRT wyświetlaczem typu VIDEO-RAM. W dziedzinie rozwoju

oprogramowania przewiduje się zaimplementowanie:

systemu opreacyjnego ISIS-II z relokowalnym makroassemblerem i kompilatorem języka PL/M oraz systemu operacyjnego CP/M 2.2 m.in. z kompilatorem języka PASCAL oraz zaimplementowanie nowych wersji interpreterów języka BASIC. Przewiduje się również opracowanie zestawu procedur pozwalających na dołączenie RTDS-8 do systemu ODRA-1305 jako inteligentnego terminala.

Mgr inż. Jerzy Szyller
Uniwersytet Warszawski
Centrum Informatyczne

# REX-80 - KONCEPCJA SYSTEMU OPERACYJNEGO DLA ZESTAWU WIELOPROCESOROWEGO M-140

## 1. Konfiguracja zestawu M-140

Rysunek 1 przedstawia schematyczną konfigurację zestawu M-140. W konfiguracji zestawu zostały schematycznie uwidocznione te elementy, których istnienie ma szczególne znaczenie dla koncepcji systemu operacyjnego REX/80.
Zestaw składa się z następujących elementów:

- procesora obliczeniowego
- od kilku do kilkunastu kontrolerów wyposażonych w pamięci lokalne
- pamięci wspólnej
- interfejsów międzyprocesorowych
- bloku obsługi przerwań międzyprocesorowych
- rejestrów ochrony pamięci dla procesora obliczeniowego
- szyny I-41 typu MULTIBUS

Procesor obliczeniowy w zestawie M-140 spełnia rolę elementu obliczeniowego oraz elementu koordynującego dostęp kontrolerów do wspólnych struktur danych. Funkcja koordynacji polega przede wszystkim na blokowaniu dostępu pozostałych kontrolerów do wszystkich wspólnych struktur danych systemowych /kolejek/ w chwili gdy na tych strukturach działa jeden kontroler. Innymi słowy koordynacja polega na umożliwieniu wyłącznego dostępu do wspólnych danych jednemu kontrolerowi. Wyłączny dostęp do struktur danych zorganizowanych w postaci kolejek pozwala na przesyłanie komunikatów między zadaniami wykonywanymi przez różne kontrolery.
Kontrolery spełniają rolę lokalnych sterowników dołączonych do nich urządzeń wejścia/wyjścia na zasadzie wykonywania zadań obsługi tych urządzeń. Ponad to kontrolery mają możliwość wykonywania innych związanych z nimi zadań. Kontrolerzy mają

dostęp do przydzielonego im obszaru pamięci, zwanej pamięcią lokalną /do 8 kilobajtów/ oraz do pamięci wspólnej w systemie dla wszystkich kontrolerów i procesora obliczeniowego /obszar od 8 do 64 kilobajtów/.
Kod programów wykonywanych przez kontrolery i procesor obliczeniowy mieści się w ich pamięci lokalnej lub w pamięci wspólnej. Koordynacja działania kontrolerów odbywa się za pośrednictwem interfejsów międzyprocesorowych, bloku przerwań międzyprocesorowych oraz procesora obliczeniowego. Za pomocą interfejsu i bloku przerwań międzyprocesorowych wszystkie aktywne elementy systemu /tj. kontrolery i procesor obliczeniowy/ mogą przesyłać między sobą informacje mające charakter globalny. Procesor obliczeniowy jako element regulujący przepływ informacji między aktywnymi elementami systemu na możliwość powiadamiania kontrolerów o napływającej dla nich informacji.

Rejestry ochrony pamięci są związane jedynie z procesorem obliczeniowym i umożliwiają zapis do obszaru pamięci, który nie jest wykorzystywany przez zadanie bieżąco wykonywane przez ten procesor.

Wszystkie kontrolery fizycznie komunikują się z pamięcią wspólną i procesorem obliczeniowym / i na odwrót/ za pomocą szyny I - 41. Szyna I - 41 ma postać zbliżoną do standardowej szyny MULTIBUS i składa się z trzech części: szyny adresowej, szyny danych i sterowania.

## 2. Przeznaczenie zestawu M-140

Zestaw M-140 ze względu na swoją wieloprocesorową architekturę może służyć do budowy oryginalnych systemów dla różnych zastosowań. Wyróżniamy dwie podstawowe klasy zastosowań: przetwarzanie danych oraz sterowanie obiektami w czasie rzeczywistym. W klasie przetwarzania danych zestaw M-140 z odpowiednim oprogramowaniem systemowym powinien dobrze spełniać rolę wielostanowiskowego lokalnego systemu do przygotowywania i wstępnego przetwarzania danych, oraz rolę zdalnej stacji terminalowej współpracującej

z komputerem nadrzędnym. Dla efektywnej realizacji wymienionych zastosowań jest jednak niezbędne istnienie w zestawie dysku kasetowego i kontrolera jednostki multipleksera umożliwiającej połączenie z maszynami Jednolitego Systemu. Podstawowym parametrem charakteryzującym systemy tego typu jest czas odpowiedzi systemu na zapytanie kilku użytkowników, który powinien się wahać w granicach od dziesiątych części sekundy do kilku sekund.

W klasie sterowania zestw M-140 powinien umożliwić sterowanie oraz zbieranie informacji napływających z obiektów różnych typów jak testery, przyrządy pomiarowe lub niewielkie obiekty przemysłowe czy laboratoryjne. Istnienie interfejsu pomiarowego IEC 488 oraz możliwość podłączenia do szyny I-41 specjalizowanych intelgentnych /zawierających własny procesor/ jednostek sterujących stwarza dobrą podstawę dla tworzenia systemów tego typu. Podstawową cechą systemów sterowania i zbierania danych w czasie rzeczywistym jest możliwie jak najszybsza reakcja systemu na zdarzenia pomiędzy którymi odstęp czasowy nie przekracza pojedyńczych milisekund, a czasem nawet dziesiątek czy setek mikrosekund.

Interesuje nas konstrukcja przede wszystkim takiego systemu operacyjnego, który umożliwiłby właściwe wykorzystanie architektury zestawu M-140 dla budowy systemów w drugiej klasie zastosowań. Równocześnie będziemy próbowali stworzyć zręby systemu operacyjnego w ten sposób aby możliwe było konstruowanie wielostanowiskowych systemów przetwarzania.

## 3. Struktura systemu REX/80.

System operacyjny REX/80 /Realtime Executive/ będzie posiadał strukturę modularną, tak aby umożliwiał tworzenie systemów dla określonych konfiguracji zestawu M-140 i dla określonych zastosowań użytkowników.

Rysunek 2 obrazuje schematyczną strukturę systemu REX/80.

Podstawowym modułem systemu jest pestka, która realizuje mechanizmy synchronizacji zadań i aktywnych elementów systemu oraz mechanizmy działania na komunikatach. W niniejszym opracowaniu opiszemy koncepcję systemu REX/80 i podamy rozwiązanie szczegółów modułu pestki systemu REX/80. Pestka systemu REX/80 stanowi pomost między sprzętem zestawu M-140 a oprogramowaniem systemu REX/80, stanowi rozszerzenie możliwości sprzętowych zestawu M-140. Pestka systemu REX/80 będzie jednym stałym modułem o niewielkich rozmiarach, tak aby otaczając ją wybranym zbiorem zadań można było tworzyć system dla określonego zastosowania.

Zadania otaczające pestkę będziemy dzielić na zadania systemowe i zadania użytkowe. Specyficzną rolę wśród zadań systemowych będą pełnić zadania wejścia/wyjścia. Przewidujemy, że zadania te będą rezydować w pamięciach lokalnych każdego z procesorów lokalnych, spełniając rolę pośrednika pomiędzy zadaniami użytkowymi a otoczeniem. Istotną częścią tych zadań będzie obsługa przerwań związanych z naturą urządzeń wejścia/wyjścia współpracujących z procesorami lokalnymi.

Zadanie /a/ zarządzania zbiorami będą umożliwiały standaryzację informacji przesyłanych między zadaniami użytkowymi w systemie a światem zewnętrznym. Postać informacji docierającej od zadań użytkowych do zadania zarządzania zbiorami zawsze powinna być jednolita, natomiast będzie ona różnicowana w zależności od charakterystyki urządzenia wejścia/wyjścia na poziomie zadania obsługi odpowiedniego urządzenia wejścia/wyjścia.

Zadanie czasomierza będzie umożliwiało odliczanie upływającego czasu i synchronizowanie działania zadań użytkowych z określoną wartością upływającego czasu.

Zadanie inicjujące, którego postać nie została jeszcze dokładnie określona powinno umożliwiać start systemu REX/80, a więc jego ładowanie z dysku elastycznego, taśmy papierowej lub innego nośnika, uruchamianie oraz w pewnych ramach rekonfigurację sprzętową i programową systemu.

Zadanie śledzące będzie pozwalało nadzorować uruchomienie jednego zadania użytkowego w systemie z dowolnego stanowiska monitorowego.

Pozostałe zadania w systemie będą zadaniami użytkowymi, które będą wykonywane w pierwszym planie i wtedy będą podlegały zwykłym regułom szeregowania, opisanym dalej oraz będą mogły być zadaniami drugoplanowymi, wykonywanymi tylko wtedy gdy procesor związany z takim zadaniem nie będzie wykazywał żadnego zadania pierwszoplanowego.

## 4. Podstawowe pojęcia związane z zasadą działania pestki systemu REX/80.

Jak powiedzieliśmy, pestka systemu jest tym modułem systemu, który umożliwia synchronizację zadań i procesorów w systemie.

Oryginalność zestawu w porównaniu z innymi spotykanymi dotąd mikrokomputerami polega na jego wieloprocesorowości. Fakt ten powoduje konieczność synchronizowania nie tylko wielu zadań, jak w systemach wielozadaniowych z jednym procesorem, ale równoczesnego działania wielu procesorów. Jednemu procesorowi w zestawie nadawana jest cechy nadrzędności /procesor obliczeniowy/, która powoduje, że właśnie poprzez ten procesor organizowany jest dostęp pozostałych podrzędnych procesorów /kontrolerów/ do wspólnych struktur danych. Komunikacja między procesorem nadrzędnym i podrzędnym /i na odwrót/ odbywa się poprzez interfejs międzyprocesorowy i blok obsługi przerwań międzyprocesorowych. Każdy następny procesor podrzędny dodawany w zestawie zaopatrywany jest w interfejs międzyprocesorowy a jego obecność jest sygnalizowana procesorowi nadrzędnemu doprowadzeniem odpowiedniej linii do kontrolera przerwań międzyprocesorowych.

Z każdym procesorem związana jest pamięć lokalna, do której nie mają dostępu inne procesory. W pamięci lokalnej każdego z procesorów będzie się mieścić część systemu operacyjnego, w tym jego pestka. W pamięci lokalnej kontrolerów będzie się mieścić obsługa sprzętu związanego z tym procesorem w postaci podprogramów obsługi przerwań i zadań obsługi urządzeń/drajwerów/.
W pamięci lokalnej będą mogły się mieścić również inne zadania bliżej związane z konfiguracją procesora. W pamięci ogólnie dostępnej przez wszystkie procesory tzw. pamięci wspólnej będą się mieścić pozostałe zadania.

Zadaniem będziemy nazywali taką samodzielną jednostkę programową, która będzie posiadała odrębny opis, obszar stosu i danych oraz taką, której działanie można zsynchronizować z innymi zadaniami według określonych reguł.

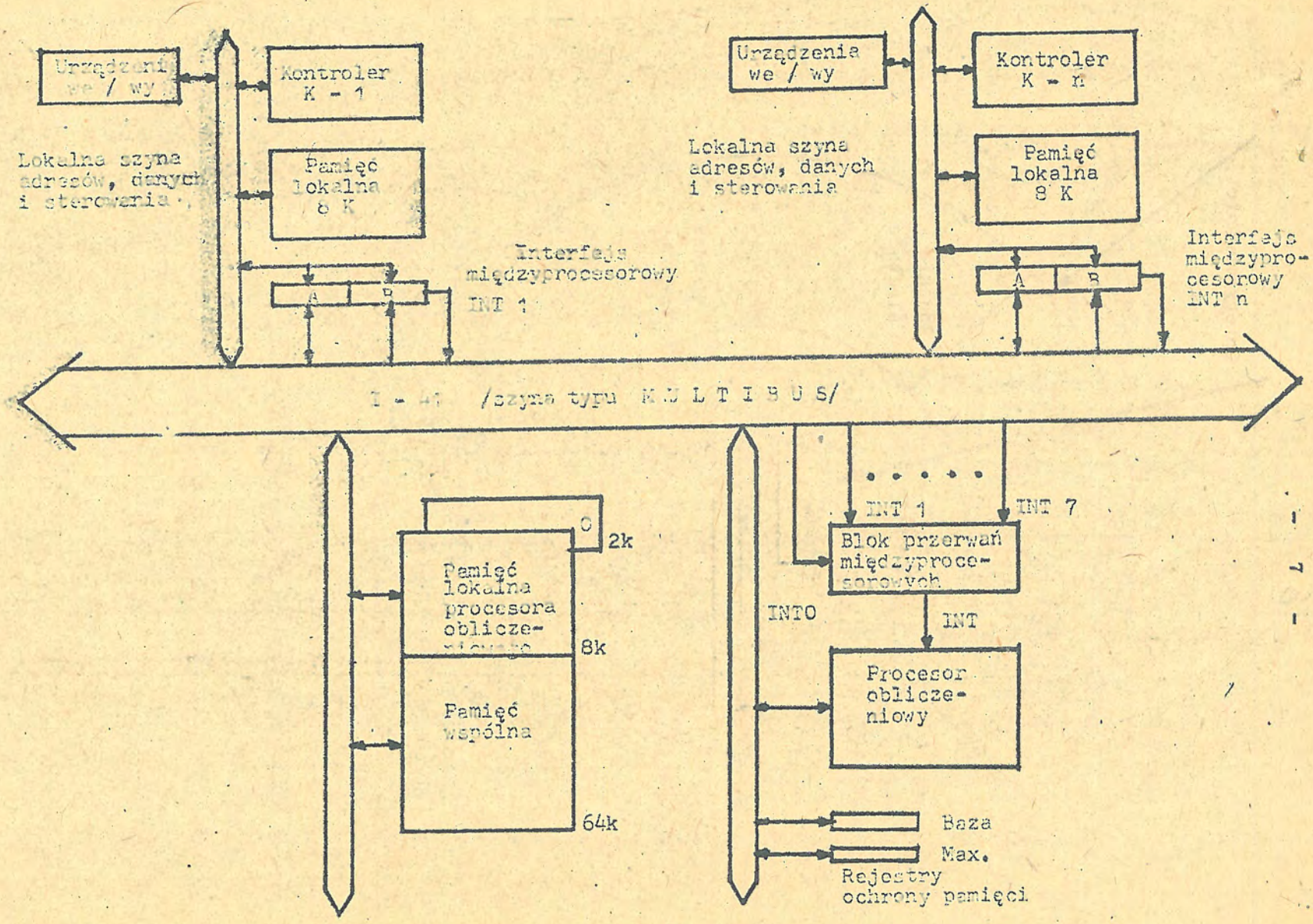
Podczas konfigurowania oprogramowania systemu REX/80 wybrano zadania wiąże się z określonym procesorem /w tym również z nadrzędnym/ w ten sposób, że mogą one być wykonywane jedynie przez ten procesor. Każde zadanie posiada kilka atrybutów, z których bardzo istotnym jest priorytet. Na podstawie priorytetu, system wybiera, które z zadań związanych z danym procesorem będzie bieżąco wykonywane.

Zadania w systemie dzielą się na pierwszoplanowe i tła. Z każdym procesorem można związać wiele zadań pierwszoplanowych i tła. Zadanie tła jest wykonywane przez procesor tylko w przypadku, gdy żadne z zadań pierwszoplanowych nie może być wykonywane.

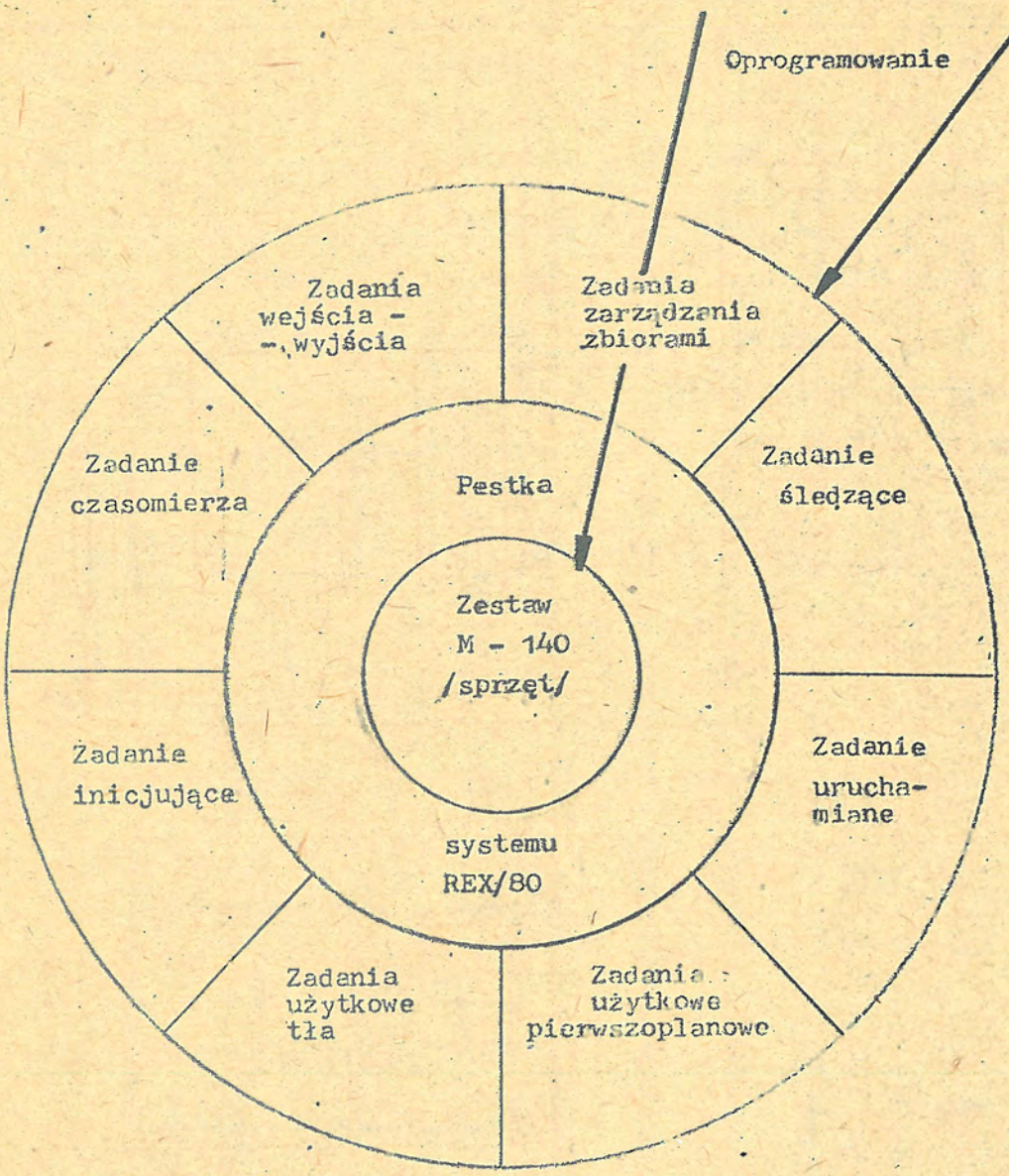
Podstawowym czynnikiem wpływającym na kolejność wykonywania zadań pierwszoplanowych w systemie jest występowanie zdarzeń znaczących. Pod pojęciem zdarzenia będziemy rozumieli ustawienie określonego warunku bitowego w rejestrze warunków zadania. Ustawienie warunku może być powodowane pojawieniem się przerwania lub wystąpieniem określonej operacji ustawienia warunku w bieżące wykonywanym zadaniu przez dany procesor.

Pojawienie się zdarzenia znaczącego w środowisku danego kontrolera lub PO może spowodować natychmiastową zmianę kolejności wykonywania zadań, ale jedynie w obrębie kolejki zadań związanych z tym kontrolerem lub procesorem obliczeniowym. W ten sposób zakładamy, że wszystkie zadania, które mają ze sobą lub z otoczeniem współdziałać maksymalnie szybko umieszczone są w jednej kolejce związanej z danym procesorem.

Niezależnie od /lokalnej/ współpracy w ramach tej samej kolejki zadania mają możliwość /globalnego/ kooperowania z zadaniami, umieszczanymi w kolejkach związanymi z innymi procesorami, za pomocą mechanizmu przesyłania komunikatów. Jednakże, ingerencja zadania z innej kolejki nie będzie powodować natychmiastowej zmiany kolejności wykonywania zadań.




Rysunek 1. Konfiguracja zestawu M-140 - elementy sprzętowe systemu REX/80.




Rysunek 2. Struktura systemu REX/80.

mgr inż. Wojciech Trojnar
Z-d Systemów Informatyki Kompleksowej
Gliwice, ul.Bałtycka 5

# FORTH - JEGO MODEL I ZASTOSOWANIA

## 1. Wstęp

FORTH został opracowany przez Charles'a Moore'a podczas jego prac programowych w takich dziedzinach jak astronomia chromatografia i systemy biurowe. Moore stwierdził, że rozwój oprogramowania w takich językach, jak ALGOL czy FORTRAN zabiera mu zbyt wiele czasu w stosunku do tego czasu jaki uważałby za niezbędny. Aby poprawić swoją sprawność w programowaniu zbudował własne środki rozwoju programów, które obecnie noszą nazwę FORTH.

FORTH powstał w okresie kilkunastu lat praktyki programowej. Obecnie łączy on w sobie język wyższego rzędu, język asemblera, system operacyjny z edytorem i zbiór procedur uruchamiania programów. Całość tworzy jednolite środowisko komputerowe, w którym użytkownik operuje jego elementami według jednolitych reguł syntaktycznych. Język FORTH posiada cechy strukturalnego języka wyższego rzędu. Postać języka jest kompromisem między elegancją języka opisu algorytmów takiego jak PASCAL, a prostotą i efektywnością jego implementacji oraz wygodą przy operowaniu przez użytkownika na elementach języka. Stąd wyrażenia arytmetyczno-logiczne są zapisane w odwrotnej notacji polskiej. Implementacja języka wykorzystuje technikę interpretacji, kompilacji i języka pośredniego. Umożliwia programowanie na "dowolnym poziomie" oddalenia obiektów języka od języka maszynowego. Użytkownik może pisać programy w języku FORTH-asemblera lub w języku obiektów utworzonych według pewnych reguł syntaktycznych. Obiekty języka nazywamy "słowami".

FORTH składa się ze słów i programów interpretacji tych słów. Każdemu słowu odpowiada nazwa i przyporządkowany jej program. Użytkownik może wywołać wykonanie słowa lub zdefiniować

nowe słowo z już istniejących w systemie słów. W pierwszym przypadku mówimy o interpretacji, a w drugim o kompilacji słowa. Połączenie techniki interpretacji i kompilacji stwarza dużą elastyczność przy posługiwaniu się komputerem w trybie interaktywnym. Programowanie polega na definiowaniu kolejnych słów, aż do momentu kiedy program przyporządkowany kolejno zdefiniowanemu słowu jest pożądanym programem. Aby wykonać program wystarczy na klawiaturze wypisać nazwę słowa lub wywołać przy pomocy odpowiednich słów interpretację danego słowa z dysku. Czas wykonywania programu w języku FORTH dla pewnych typów mikroprocesorów jest o 20-75% większy niż czas wykonywania analogicznego programu napisanego w języku asemblera [1]. Użytkownik może elastycznie dobierać czas wykonywania programu, ponieważ FORTH jest językiem "dowolnego rzędu". Krytyczne słowa mogą być zdefiniowane w języku asemblera. Pamięć operacyjna potrzebna do przechowywania programu w języku FORTH jest na ogół mniejsza od pamięci potrzebnej dla analogicznego programu asemblerowego, ponieważ program wynikowy jest w postaci kodu nieznanego /thereaded code/.

Zbiór słów nazywamy słownikiem. Repertuar słów w słowniku pozwala budować strukturalne programy, przy pomocy wyspecjalizowanych słów-kompilatorów oraz programować własne kompilatory przy pomocy słów-metakompilatorów. Tworzonym słowom użytkownik może nadawać własne nazwy. W ten sposób tekst programu może być bliski terminologii używanej w danej dziedzinie zastosowań. Język FORTH może stanowić bazę dla projektowania języków zorientowanych problemowo.

FORTH jest rozpowszechniony w postaci programu źródłowego napisanego w języku asemblera. Część tego programu /około 800 bajtów pamięci w przypadku mikroprocesora INTEL-8080/ jest napisana w języku asemblera tego mikroprocesora. Pozostała część /około 7Kb/ jest napisana dyrektywami asemblera /DW, DB w przypadku asemblera ASM80/. Czas przeniesienia programu FORTH na inny mikroprocesor wynosi 1-2 osobomiesięcy.

Uprzednio podany podział na język, system operacyjny itd, ma tylko przyporządkujące znaczenie. Wynika on z dotychczasowych sposobów implementacji systemów komputerowych. Tradycyjnie kompilatory języków wyższego rzędu oraz systemy operacyjne są rozróżnialnymi częściami oprogramowania. FORTH nie wprowadza takiego podziału, dlatego mówiąc o języku lub systemie operacyjnym odwołujemy się do znaczeń tych terminów jakie występują w tradycyjnych systemach komputerowych.

System operacyjny FORTH'a zarządza terminalem, dyskiem i drukarką. Pamięć dyskowa wraz z buforami dyskowymi w pamięci operacyjnej tworzą pamięć wirtualną. Programy operacji na urządzeniach WE/WY umożliwiają wczytanie znaku lub linii z klawiatury oraz wprowadzenie tekstu na ekran. Do przygotowania programu i danych służy Edytor ekranowy. FORTH jest zbiorem słów. Aby zapewnić przenośność oprogramowania dla różnych implementacji zdefiniowano standardowy zbiór słów. Obecnie istnieją dwa takie standardy: fig-FORTH i FORTH-79. Pierwszy z nich został opracowany przez FORTH Interes Group. Grupa ta rozpowszechnia podręcznik "fis-FORTH Installation Manual" oraz tekst programu w asemblerze dla dziewięciu mikroprocesorów. Standard FORTH-79 jest opracowany przez FORTJ Satndard Team.

W chwili obecnej FORTH jest zaimplementowany w większości systemów mikrokomputerowych. Znalazł on zastosowanie w wielu publikacjach takich jak: sterowanie procesami technologicznymi, medycyna, systemu zabrania danych itp. Z praktyki autora wynika, że FORTH jest doskonałym narzędziem do pisania programów testujących aplikacyjne systemy mikrokomputerowe.
Wynika to z łatwej modyfikacji programów napisanych w tym języku.

Czas przygotowania testu oraz czas modyfikacji testu jest najkrótszym w systemie FORTH spośród znanych autorowi systemów programowania.

## 2. Maszyna-FORTH

W poprzednim rozdziale przedstawiono system FORTH jako zbiór słów i reguł wyboru tych słów do wykonania. Słowa i reguły tworzą środowisko, w którym są wykonywane programy. Środowisko to dalej nazywać będziemy "maszyną-FORTH". Funkcjonowanie Procesora wirtualnego, jakim jest maszyna-FORTH wygodnie jest przedstawić w podobny sposób jak przedstawia się funkcjonowanie procesorów maszyn cyfrowych. Schemat procesora przedstawia rysunek 1. Elementy wirtualne połączone są magistralą, którą przepływa informacje między nimi. Większość przedstawionych elementów posiada swoje odpowiedniki w typowym procesorze mikroprogramowanej maszyny cyfrowej. Program w języku FORTH jest ciągiem znaków ASCII /nazwy słów, literały łańcuchy/ oddzielonych spacjami lub znakiem kończącym linię. Program rezyduje w pamięci dyskowej lub "pamięci" operatora korzystającego z terminala FORTH'a. W danej chwili tekst jest pobierany z terminala lub z dysku. Po inicjacji systemu maszyna FORTH oczekuje na linię tekstu z terminala. Operator wprowadza linie tekstu, którą kończy znakiem CR. Wprowadzony tekst jest programem poddawanym interpretacji. Linia tekstu składa się z łańcuchów tekstu oddzielonych spacjami. Na rysunku 1 przedstawiono klawiaturę, dysk, bufor terminala i bufory dyskowe jako "pamięć programów". Istnieje analogia między pamięcią FORTH-maszyny a pamięcią procesora maszyny mikroprogramowanej. Łańcuch tekstu, który w szczególności jest nazwą słowa, ma być zinterpretowany i wykonany przez procesor wirtualny. W tym celu przesyła się go do oddzielnego bufora zwanego WORD. Przypuśćmy, że w buforze terminala znajduje się tekst:

```
VLIST    FORTH
↑
```

Wskaźnik bufora pokazuje literę V. Po przesłaniu łańcucha tekstu z bufora terminala do bufora WORD wskaźnik bufora przeniesie się na koniec słowa VLIST:

VLIST FORTH
↑

W buforze WORD będzie znajdował się tekst VLIST. Łatwo zauważyć, że wyżej opisany proces jest podobny do tego, jaki wykonuje procesor maszyny cyfrowej przy pobraniu rozkazu z pamięci operacyjnej do rejestru rozkazu. W procesorze wirtualnym jeden z obiektów nazwaliśmy "pamięcią sterującą". W tej pamięci znajduje się słownik tzn. zbiór nazw słów wraz z programami, które są wykonywane po wywołaniu nazw słowa. Słowa w pamięci sterującej tworzą strukturę drzewiastą. Program zwany "interpretatorem zewnętrznym" pobiera z bufora WORD nazwę słowa, sprawdza czy istnieje w pamięci sterującej słowo o tej nazwie i następnie wywołuje wykonanie programu tego słowa. Przedstawiony proces jest odpowiednikiem cyklu pobrania instrukcji procesora maszyny cyfrowej. Wykonanie słowa VLIST powoduje zapis na ekranie nazw wszystkich słów znajdujących się w słowniku.

Po zakończeniu wykonania słowa, FORTH-maszyna pobiera kolejny łańcuch tekstu z buforem. Gdy programy wszystkich słów w linii zostaną wykonane, następnie wczytanie kolejnej linii tekstu z terminala. Proces ten jest wykonywany w nieskończonej pętli, tak jak wykonuje się cykl instrukcji procesora maszyny cyfrowej. Interpretator zewnętrzny może być w jednym z dwóch trybów pracy: "interpretacja" lub "kompilacja". Powyżej opisano pracę interpretatora w trybie "interpretacja". W trybie "kompilacja" interpretator kompiluje program nowego słowa wzbogacającego repertuar słów znajdujących się w słowniku. Dla wyjaśnienia tego określenia posłużymy się przykładem definicji słowa, nie tłumacząc semantyki kompilowanego programu. Niech tekst w buforze terminala ma postać:

: KWADRAT-PRZECIWPROSTOKATNEJ DUP * SWAP DUP * + ;

W trabie "interpretacja" jest wykonany program słowa ":" który wytworzył nową nazwę KWADRAT-PRZECIWPROSTOKATNEJ w słowniku

i zmienił stan interpretatora zewnętrznego na "kompilacja". Kolejne słowo nie jest wykonywane lecz kompilowane tzn. adres pośredni początku programu kompilowanego słowa jest ustawiony do pierwszego wolnego miejsca pamięci sterującej. Proces ten jest kontynuowany aż do napotkania nazwy słowa ";", które jest wykonywane w trybie "kompilacja". Program tego słowa zmienia stan interpretatora zewnętrznego na stan "interpretacja". Rezultatem powyższego procesu jest utworzenie nowego słowa w pamięci sterującej o nazwie KWADRAT-PRZECIPROSTOKATNEJ i programie składającym się z sekwencji programów o nazwach słów DUP * DWAP DUP * +.

Proces kompilacji można porównań z rozszerzeniem listy instrukcji maszyny mikroprogramowanej. Dokonuje się to przez zapisanie do pamięci sterującej tej maszyny, mikroprogramu, który jest wykonywany po przesyłaniu do rejestru rozkazu kodu zdefiniowanej instrukcji.

## 3. Zastosowania

Język FORTH posiada wszystkie cechy języka wyższego rzędu a co za tym idzie przy pomocy tego języka można zapisać wszystkie problemy jakie są możliwe do sformułowania w języku algorytmicznym. To co odróżnia go od innych języków to sposób implementacji. Pewne cechy FORTHa dają mu przewagę nad innymi językami. Są nimi:

- interaktywny rozwój programów,
- elastyczny dobór struktur programowych,
- możliwość doboru czasu wykonania programu i zajętości pamięci

Duże korzyści ze stosowania języka FORTH można osiągnąć przy oprogramowaniu aplikacji wymagających interaktywnej współpracy użytkownika sprzętu obliczeniowego z otoczeniem. Taką klasą zastosowań są programy testujące i programy zbierania danych. Dalej zostaną przedstawione 3 takie zastosowania.

3.1. Testowanie drukarki graficznej

Mikrokomputer z systemem FORTH został użyty do testowania oprogramowania drukarki graficznej zrealizowanej przez zastąpienie dotychczasowego kontrolera drukarki przez kontroler mikroprocesorowy. Program w języku FORTH przesyła na drukarkę sekwencje znaków alfanumerycznych i sterujących. Oprogramowanie testowe powstawało w czasie kolejnych etapów uruchamiania. Po każdej zmianie w programie kontrolera drukarki program testu sprawdzał czy poprzednio uruchomione oprogramowanie kontrolera jest poprawnie wykonywane. Fragmenty testu są zaprogramowane jako pojedyńcze słowa w języku FORTH. Po wykryciu błędu poprawki testowano przy pomocy tego słowa którego wykonanie powodowało błędną pracę drukarki. Czas przygotowania testu był niewielki w porównaniu do czasu spędzonego na rozwoju programu kontrolera drukarki.

3.1. Symulacja Jednostki Grupowej JSG-7801

System RTDS-8 może się komunikować z komputerem ODRA-1305 przez łącze szeregowe za pośrednictwem jednostki grupowej JSG-7801. W tym celu system RTDS-8 wyposażono w program komunikacyjny, który przesyła i odbiera dane zgodnie z protokołem jednostki grupowej. Łącze szeregowe od strony systemu było podłączone do modemu do którego w tym samym czasie było podłączonych kilka innych terminali. Uruchamianie programu komunikacyjnego w czasie pracy systemu mogło powodować zakłócenia w pracy na pozostałych terminalach. Aby zminimalizować czas wyłącznej pracy uruchomionego programu komunikacyjnego z systemem ODRA-1305 zaimplementowano na innym mikrokomputerze /również RTDS-8/ program w języku FORTH który symulował zachowanie się jednostki grupowej. Tekst programu zajmował 7 ekranów. Program symulatora był nieco mniej złożony niż program komunikacyjny ale czas uruchamiania programu symulatora jest nieporównywalnie mniejszy od czasu uruchamiania programu komunikacyjnego napisanego w języku PL/M-80.

3.3. Stacja zbierania danych STA.

W Instytucie Automatyki Politechniki Śląskiej zrealizowano na mikrokomputerze MERA-80 stację zbierania danych odbieranych ze stacji telemetrycznych. Mikrokomputer komunikuje się ze stacjami za pośrednictwem łącz komutowanych. Dane odebrane z obiektów zgromadzone są na dyskach, drukowane są na drukarce i zapisywane na monitorze. Programy komunikacji z urządzeniami, oraz programy zarządzające zbiorami dyskowymi napisano w języku asemblera. Komunikacja i synchronizacja tych programów dokonuje się przez wywołanie procedur monitora wielozadaniowego. Program rozpoznawania zleceń operatora napisano w języku FORTH. Program ten został dołączony do całości oprogramia jako jedno z zadań systémowych. Zlecenia systemowe są zaprogramowane jako słowa w języku FORTH. Operator stacji zbierania danych może tworzyć programy w postaci ciosów zleceń wyrażonych w postaci sekwencji słów w języku FORTH.

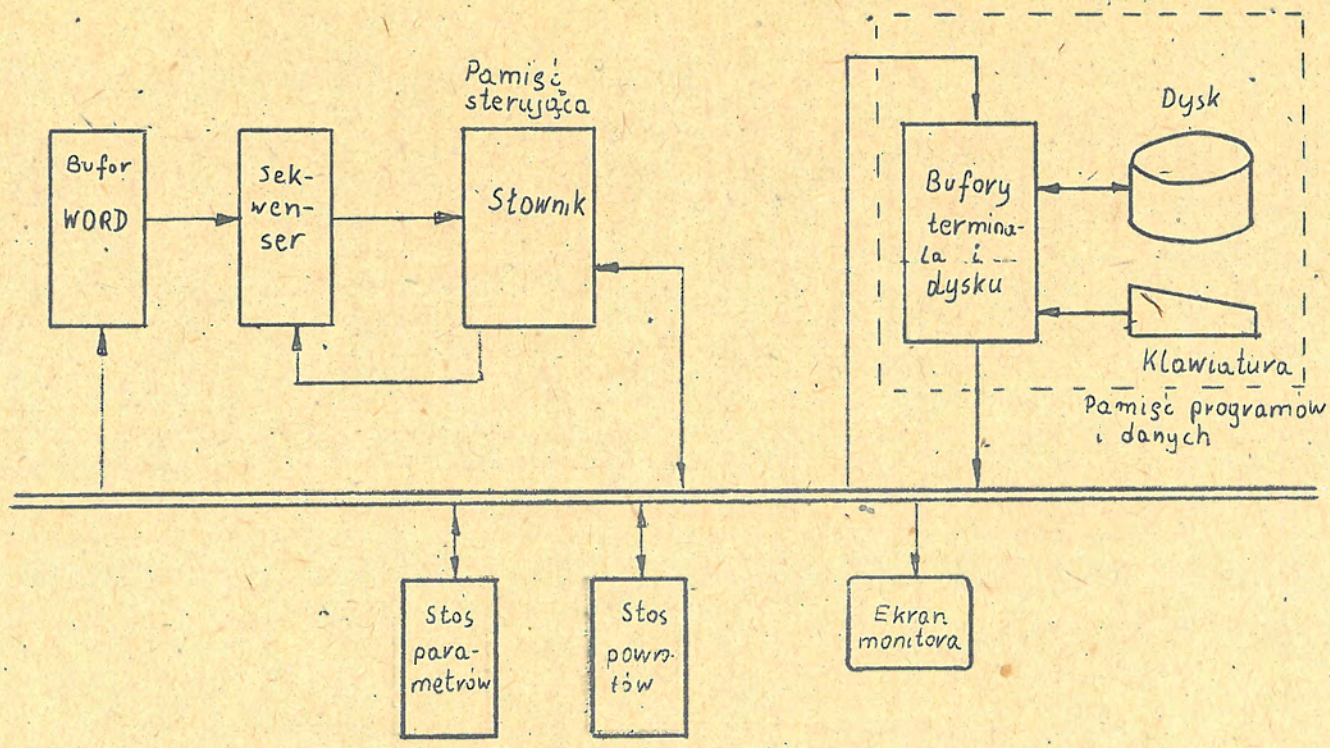
Wydaje się, że wykorzystanie interpretatora języka FORTH jest najszybszym sposobem napisania programu interpretacji zleceń.

4. Podsumowanie

FORTH przypuszczalnie będzie znajdował coraz więcej zastosowań w miarę poszerzania się kręgu ludzi znających ten język. Jednocześnie należy się spodziewać implementacji systemów w których zastosowano pewne techniki wywodzące się z FORTHa. Przykładem takiego systemu może być ECHONET 2

LITERATURA

[1] Leo Brodie - "Starting FORTH" Forth, Inc.
[2] C.Bradford Barber "ECHONET" Byte September/October Vol.8. No. 9/10




Rysunek 1. Maszyna-FORTH

FORTH – 1 osobotydzień z książki wersja źródłowa (interpr.)

Janusz Zalewski
Pazińskiego 5a m 17
04-643 Warszawa
/12-15-36/

# RAPORT DLA KLUBU RZYMSKIEGO "MIKROELEKTRONIKA I SPOŁECZEŃSTWO - NA DOBRE I ZŁE" - WPROWADZENIE

## 1. WPROWADZENIE

Klub Rzymski powstał w kwietniu 1968 roku, w rzymskiej Accademia dei Lincei, jako nieformalne stowarzyszenie kilkudziesięciu osób. Od 1973 roku jest zarejestowany w Genewie jako organizacja międzynarodowa i zrzesza około 100 członków z kilkudziesięciu krajów. Pierwszy raport Klubu Rzymskiego, "Granice wzrostu", D.H. Meadows i in., opublikowano w roku 1972 /wydanie polskie - 1973 r./, a drugi - "Ludzkość w punkcie zwrotnym", M. Mesarovic, E. Pestel - w 1974 /wydanie polskie - 1977 r./. Oprócz tego powstało wiele innych raportów, mniej znanych w Polsce, jak: "Uczyć się bez granic", "Cele ludzkości" czy "Sto stron dla przyszłości".

Najnowszy raport pt. "Mikroelektronika a społeczeństwo", wydany 2 lata temu, miał szeroki rezonans w Polsce. Współredaktorem tomu i jednym z autorów jest członek PAN, filozof-marksista, prof. Adam Schaff. Obszerne fragmenty raportu /niemal pięć rozdziałów/ opublikowano już w języku polskim /1-10/, a całość tłumaczenia została złożona w Wydawnictwie Książka i Wiedza. Ponieważ opublikowano obszerną recenzję raportu /11/, a także - szereg wywiadów z prof. Schaffem /12-13/, poniżej nie zreferowano szczegółowo treści, przedstawiając jedynie - oprócz ogólnej charakterystyki - kilka refleksji dotyczących wybranych wątków książki, bardzo bogatej w fakty, spostrzeżenia i opinie.

Rozwój mikroelektroniki, tak jak każdej nowej techniki, jest ambiwalentny. Wielu naukowców, już we wczesnych stadiach

jej rozwoju, wskazywało zarówno na występowanie bardzo korzystnych zjawisk jak i na ich negatywne następstwa. Jeszcze przed opub kowaniem raportu prof. Janusz Groszkowski mówił, np.: "Z tej dziedziny nauki i techniki, jaką jest radiotechnika i zrodzona z niej nowoczesna elektronika, wynikają niezliczone zastosowania, będące podstawą dzisiejszej cywilizacji. Wyliczanie samych ich nazw zajęłoby zbyt wiele czasu. Wspomnę tu tylko o telekomunikacji ze sterowaniem na odległość, która pozwala na obserwacje pierścieni Saturna odległych od nas o półtora miliarda kilometrów. Wspomnę tu o komputerach, rozwiązujących błyskawicznie najbardziej skomplikowane obliczenia, czy o aparaturze, umożliwiającej wgląd we wnętrze atomu. Są to urządzenia i zastosowania wspaniałe, jeśli się je zastosouje do celów poznawczych i dla dobra ludzkości. Ale, jak wiemy, nie zawsze tak jest, bo włączając na przykład odbiornik radiowy czy telewizor często nie słyszymy prawdy albo prawda bywa zagłuszana przez odpowiednie urządzenia elektroniczne. Istnieją też urządzenia oparte na zastosowaniu radiotechniki i elektroniki, za pomocą których można z odległości tysiąca kilometrów nakierowywać pociski śmiercionośne na określone obiekty" /J. Mikke, "Wizerunki ludzi myślących", str. 104-105, Wyd. RiT, Warszawa, 1982/.

Podstawowym przesłaniem raportu, owym "na dobre i złe", jest właśnie wskazanie na obosieczność nowej technologii, która równie dobrze może służyć rozkwitowi ludzkości, jak i spowodować jej zagładę. Ten temat rozwija szerzej we "Wprowadzeniu" Aleksander King.

## 2. TECHNOLOGIA I JEJ ZASTOSOWANIA

Kolejne dwa rozdziały, tworzące wyraźnie pierwszą część raportu, dotyczą technologii i jej zastosowań, i nie są specjalnie interesujące dla elektroników lub informatyków, którzy znają te zagadnienia z praktyki.

Warto jednakże zwrócić uwagę przynajmniej na dwa aspekty rozwoju mikroelektroniki, a właśnie jej dziecka - techniki komputerowej: gwałtowne rozprzestrzenianie się mocy obliczenio-

wej /wnikanie mikroprocesorów do niezliczonej liczby urządzeń/ i równie gwałtowny wzrost skupionej mocy obliczeniowej. Wyrazem pierwszej z tych tendencji, świadczącym jednocześnie o skali problemu, jest fakt, że w 1981 roku 6,5 mln nowych samochodów amerykańskich wyposażono w mikroprocesory. Druga tendencja przejawia się w rozwoju superkomputerów, gigantycznych maszyn, złożonych z wielu procesorów wykonujących równolegle miliony operacji.

Wsiąkanie mikroelektroniki w społeczeństwo następuje najskuteczniej dzięki rozpowszechnieniu komputerów osobistych. Można zaryzykować twierdzenie, że żadne inne urządzenie elektroniczne nie wywołało tak wielu skutków społecznych w tak krótkim czasie. Możliwości funkcjonalne tych niewielkich urządzeń są olbrzymie, a przykładów, które można wymienić jest zbyt dużo. Najszersze zastosowanie znajdują w nich nowości technologiczne. Przewiduje się, że już wkrótce pojawią się na rynku pierwsze komputery osobiste, oparte na mikroprecesorach 32-bitowych, choć w 1983 roku były w sprzedaży zaledwie 2 mikroprocesory tego rodzaju: zestaw NCR/32 i Bellmac-32. Spośród nowoczesnego wyposażenia komputerów osobistych zwraca się uwagę na tzw. "myszki" i "dyski półprzewodnikowe".

Myszka jest niewielkim aparacikiem przesuwanym po płaskim pulpicie i służącym do poruszenia kursora po ekranie monitora. Umożliwia to uzyskanie rozdzielczości równej pojedyńczemu elementowi obrazu /ang. pixel, picture element/, na co nie pozwala tradycyjna technika z użyciem pióra świetlnego, które obejmuje co najmniej kilka elementów jednocześnie. Myszka jest sprzężona z czujnikiem elektromechanicznym lub optoelektronicznym, a całe urządzenie z komputerem przez sprzęg V24.

Innym przejawem postępu jest półprzewodnikowa pamięć masowa odpowiadająca pojemnością pamięci dyskowej, tzw. "dysk półprzewodnikowy". Dostęp do płytki z pamięcią RAM 512 Kb następuje przez dwa porty - port adresowy i port danych - natomiast organizacja pamięci odzwierciedla budowę dysku z podziałem na ścieżki i sektory. Dzięki temu komputer komunikuje się z pamięcią RAM jak z pamięcią dyskową lecz w wielokrotnie krótszym czasie -

czas dostępu wynosi 100 ns zamiast kilkudziesięciu milisekund. Obecna konstrukcja umożliwia rozszerzenie pojemności "dysków" półprzewodnikowych do 4 MB.

Ważnym aspektem rozwoju komputerów osobistych jest ich łączenie w bardziej złożone zestawy. Podstawową przyczyną tej tendencji jest konieczność wspólnego korzystania z bardziej kosztownych urządzeń, jak drukarki, pamięci dyskowe itp. Dzięki współdzieleniu zasobów komputery osobiste w sieci są znacznie tańsze od pojedyńczych. Jednakże, standardowe sprzęgi typu V24 nie mogą sprostać wszystkim wymaganiom stawianym takiej sieci, gdyż chodzi tu o połączenia między komputerami, a nie tylko - terminala z komputerem. Obserwuje się, więc, nową ważną tendencję związaną z rozwojem połączeń międzykomputerowych na niewielkie odległości - powstawanie technologii sieci lokalnych.

Jedną z ważniejszych jest, obchodząca niedawno 10-lecie, znormalizowana sieć Ethernet, której twórca Robert Metcalfe otrzymał w ubiegłym roku nagrodę czasopisma Electronics /No. 20, 6 October 1983/. Ethernet umożliwia przesyłanie informacji między komputerami w obrębie sieci o długości 2500 m, z szybkością 10 Mb/s. Już obecnie wiele firm produkuje standardowe mikroukłady sprzęgające komputery osobiste z siecią Ethernet. Sprzedaje się również oprogramowanie umożliwiające wykorzystanie sieci jako poczty elektronicznej, do automatycznego składu i druku tekstów itp. Przewiduje się, że wkrótce około setki różnych urządzeń powszechnego użytku zostanie wyposażonych w mikroprocesory umożliwiające połączenie ich w sieć domową, obejmującą telewizory, telefony, lodówki, pralki, suszarki, nawilżacze, oświetlenie, ogrzewanie, chłodzenie, zabezpieczenie mieszkań przed włamaniem itd.

## 3. KOMPUTERYZACJA A GOSPODARKA

Kolejne trzy rozdziały raportu są powiązane tematycznie i dotyczą wpływu mikroelektroniki na stanowisko pracy, na przedsiębiorstwo i na ekonomię jako całość.

O ile rozdziały dotyczące technologii są zbyt powierzchowne dla elektroników, to drugiej części raportu z pewnością wiele zarzutów postawią ekonomiści. Wydaje się, że za dużo jest tu spekulacji i przypuszczeń, a za mało obserwacji popartych rzetelnymi badaniami empirycznymi. Jedno jest pewne, że technika, która zwiększa możliwości przetwarzania danych ludzi i instytucji o kilka rzędów wielkości, musi mieć znaczący wpływ na ich działanie. Jednakże, ocena kierunków tych zmian a tym bardziej ich przewidywanie jest dość trudne.

W jednej z niewielu prac analizujących badania empiryczne w tej dziedzinie /R. Kling, Social Analyses of Computing, Computing Surveys, Vol. 12, No. 1, pp. 61-110, 1980/ zweryfikowano przypuszczalne zmiany, jakie wprowadza komputeryzacja instytucji, w odniesieniu do:

- procesu pracy,
- podejmowania decyzji,
- struktury władzy w instytucjach.

Szczegółowe badania empiryczne nie potwierdzają większości przypuszczeń co do wpływów komputeryzacji na pracę. Okazuje się, że wraz z zastosowaniem komputerów niekoniecznie musi zmieniać się charakter pracy /w szczególności dotyczy to tzw. białych kołnierzyków/. Pozornie oczywisty fakt, że komputery umożliwiają znaczne zwiększenie nadzoru nad personelem, również nie znajduje jednoznacznego potwierdzenia w wynikach badań. Spośród dość nieoczekiwanych spostrzeżeń warto zwrócić uwagę na możliwości zakłóceń rytmu pracy, a więc działalności instytucji, spowodowane kłopotami z eksploatacją komputerów. Na ogół, potwierdzają się przypuszczenia, że skuteczność użycia komputerów zależy w znacznym stopniu od świadomości i swobody ich wykorzystania przez pracowników.

W podejmowaniu decyzji komputery są użyteczne szczególnie w przeprowadzaniu typowych operacji, jak np. automatyzacja rozliczeń finansowych. Stwierdzono natomiast, że skuteczność informacyjnych systemów zarządzania jest dość problematyczna, gdyż w odróżnieniu od sterowania obiektami technicznymi należy uwzględnić tu czynnik ludzki, co prowadzi na ogół do istotnej

komplikacji modelu, podważającej skuteczność zarządzania. Mimo istnienia dość złożonych systemów informacyjnych dla potrzeb administracji miejskich w USA, stosunkowo rzadko wykorzystuje się je w planowaniu. Stwierdzono, że wykorzystanie tych systemów następuje często jedynie w celu udokumentowania określonych programów działania, a nie w celu obiektywnej analizy zjawisk.

W odniesieniu do struktury władzy w instytucjach, rozumianej jako posiadanie wpływu na istotne decyzje i na działalność instytucji, stwierdzono że grupy lub jednostki dysponujące dostępem do komputerów na ogół wzmacniają swoją pozycję. Dotyczy to zarówno przedsiębiorstw jak i organów władzy różnych szczebli od samorządowych i municypalnych do najwyższych w państwie.

Problematykę wpływu komputerów na przedsiębiorstwo i ekonomię omówiono dość dokładnie, choć tylko pod względem teoretycznym, w wydanej w Polsce lecz raczej niezauważonej książce laureata nagrody Nobla - Herberta Simona /Podejmowanie decyzji kierowniczych, PWE, Warszawa, 1982/.

## 4. OGOLNOSWIATOWE SKUTKI ELEKTRONIZACJI

Najpokaźniejszą część raportu stanowią cztery rozdziały dotyczące globalnych konsekwencji rozwoju mikroelektroniki - jej skutków w skali światowej. Zanalizowano w nich wpływ mikroelektroniki na rozwój Krajów Trzeciego Swiata, na zbrojenia, na porozumiewanie się i na politykę międzynarodową. We wszystkich tych dziedzinach istotną rolę odgrywa promocja mikroelektroniki, a raczej całej technologii informacyjnej, przez państwo.

Mianem technologii informacyjnej lub technologii informacji /ang. information technology/ można określić encyklopedycznie ogół świadomości o metodach, procesach, sposobach i środkach przetwarzania informacji. Termin ten powstał przez analogię do tradycyjnie rozumianej technologii jako ogół wiadomości o metodach, procesach, sposobach i środkach przeróbki i obróbki materiałów /np. technologia mechaniczna, technologia chemiczna, technologia elektronowa itp./.

Nie ulega wątpliwości, że rola administracji państwowej w subsydiowaniu technologii informacyjnej jest podstawowa dla jej prawidłowego rozwoju i leży w interesie państwa pragnącego utrzymać lub wzmocnić swoją pozycję na arenie międzynarodowej.

Zachodnioeuropejski program ESPRIT - European Strategic Programme for Research and Development in Information Technologies - wprowadzany przez EWG, powstał jako odpowiedź na coraz wyraźniej zaznaczającą się przewagę Japonii i USA w przemysłowych zastosowaniach nowych technologii, a szczególnie - mikroelektroniki. Ze względu na przenikanie elektroniki praktycznie do wszystkich gałęzi przemysłu, początkowo niegroźne uleganie wpływom w kilku wąskich dziedzinach, obecnie przeradza się w prostą zależność ekonomiczną. Stwierdzono, że w tych warunkach tożsamość Europy Zachodniej i jej polityczna niezależność ulegają zagrożeniu. Strategicznym celem programu ESPRIT jest "dorównanie czołówce światowej, jeśli nie prześcignięcie jej - pod względem technologicznym - w ciągu 10 lat".

Jednakże nakłady wymagane do efektywnych działań przekraczają możliwości pojedyńczych firm /a nawet większych korporacji/ lub poszczególnych członków wspólnoty. Szanse powodzenia mają jedynie przedsięwzięcia prowadzone na poziomie całej wspólnoty, oparte na łącznym długoterminowym planowaniu. Wyróżniono 5 obszarów tematycznych, w których rozpoczyna się prace:

- nowoczesna mikroelektronika, stanowiąca podstawę fizyczną wszelkich systemów informatycznych
- technologia oprogramowania, umożliwiająca sterowanie zachowaniem tych systemów
- nowoczesne metody przetwarzania informacji, umożliwiające optymalizację zachowania przez kombinację sprzętu i oprogramowania
- systemy biurowe,
- komputerowo zintegrowane wytwarzanie.

Na pierwsze 5 lat całego 10-letniego programu zamierza się przeznaczyć 750 mln ECU /European Counting Unit, nieco mniej niż dolar USA/. Fazę pilotową rozpoczęto w połowie roku ubiegłego, z budżetem 11,5 mln ECU.

W Wielkiej Brytanii rząd zaaprobował projekt tzw. komisji Alveya i zamierza wyasygnować w ciągu najbliższych 5 lat - 350 mln funtów na rozwój technologii informacyjnej. Około 50 mln przeznacza się dla instytucji akademickich, a 300 mln dla przemysłu /w praktyce, koszt badań w przemyśle będzie pokrywany tylko w połowie z pieniędzy państwowych, co oznacza obciążenie budżetu tylko 150 mln funtów/. Pieniądze będą pochodzić z Ministerstwa Handlu i Przemysłu, z Ministerstwa Obrony oraz z Ministerstwa Nauki i Szkolnictwa.

Instytucje działające w tym programie mogą być tylko brytyjskie i muszą posiadać niezbędne doświadczenie. Ponieważ program Alveya uważa się za komplementarny względem ESPRIT, w wyjątkowych wypadkach, przy zbieżności tematów i celów, dopuszcza się udział instytucji zagranicznych. W planie programu założono cztery główne kierunki badawcze:

- układy o bardzo dużym stopniu scalenia /VLSI/,
- inżynieria oprogramowania,
- systemy ekspertyzy i bazy wiedzy /ang. expert systems, knowledge-based systems/,
- współpraca człowieka z maszyną /ang. man-machine interface/.

Cztery kierunki programu są niezależne i będą koordynowane przez 8-osobowy Dyrektoriat z przewagą reprezentacji przemysłu. Administrowanie poszczególnymi częściami programu zostało powierzone już istniejącym instytucjom, pod nadzorem Dyrektoriatu.

Dla zapewnienia efektywności programu przewiduje się m.in.:

- prowadzenie projektów pilotowych, które mają wykazać, że koncentracja środków prowadzi do lepszych wyników i do osiągnięcia celów niedostępnych przy rozproszeniu nakładów,
- angażowanie możliwie mininalnej liczby instytucji w jeden projekt, gdyż w dużych grupach współpraca jest mniej skuteczna,
- zapewnienie dostatecznego przepływu informacji między współpracującymi instytucjami, m.in. przez organizację sympozjów, publikowanie biuletynów, raportów itp.

## 5. ZAJĘCIE ZAMIAST PRACY

Taki tytuł nosi ostatni rozdział raportu napisany przez Adama Schaffa, a jego konkluzja sprowadza się do stwierdzenia, że wskutek upowszechnienia automatyzacji praca przestanie zajmować człowiekowi większą część życia. Aby nie odczuwać frustracji spowodowanej nadmiarem czasu wolnego lub zgłagodzić ją, będziemy musieli znaleźć substytut pracy. Przewiduje się, że taką funkcję może pełnić ustawiczne kształcenie.

Należy przypuszczać, że ważną rolę w naszych codziennych zajęciach będzie odgrywać suma informatyka jako dziedzina wiedzy. Tematyka ściśle komputerowa wkroczyła nawet do beletrystyki a ostatnim przykładem jest powieść Tracy Kiddera "Dusza nowej maszyny" /The Soul of a New Machine/ - opis procesu konstruowania 32-bitowego minikomputera ECLIPSE firmy Data General. Opis jest pasjonujący dla specjalisty, gdyż ukazuje ogrom umysłowych i fizycznych zmagań i stopniowe pokonywanie oporu materii podczas przebiegu i realizacji całego przedsięwzięcia przez ludzi, którzy nie traktują swoich obowiązków jak pracy z otrzymywanym za nią wynagrodzeniem lecz właśnie - jak zajęcie. Może to być jednakże literatura frapująca także dla laika, gdyż na tle walki zespołu o kształt maszyny przedstawione są wszystkie podstawowe zasady konstrukcyjne nowoczesnego komputera. Jest to jeden ze sposobów przenikania mikroelektroniki do naszego życia.

O wpływie tej techniki na naszą codzienność mówią sami bohaterowie powieści, uczestniczący w dorocznej Krajowej Konferencji Komputerowej /National Computer Conference, NCC/: "... wyszliśmy z budynku targowego do kawiarni położonej w pewnej odległości od Koloseum. Siedząc tam i obserwując naturalny chaos nowojorskiej ulicy, zdałem sobie sprawę jak niezauważalna jest rewolucja informatyczna. Opuszczając ten swoisty jarmark, jakim jest NCC oczekujesz, że twój sposób patrzenia na świat zewnętrzny zmieni się, jednak w zasięgu wzroku nie ma niczego nadzwyczajnego. Nie ma cyborgów, pół-maszyn, pół-ludzi przemierzających ulice, nie ma armii bezrobotnych, dźwigających transparenty wyklinające komputer, nie ma obserwujących nas kamer telewizyjnych. /.../

Oczywiście komputery były wszędzie wokół nas - w piszczącej kasie kawiarnianej, w piecu mikrofalowym i w szafie grającej, sterowały ruchem ulicznym za pomocą świateł, były pod maskami trąbiących samochodów i w samolotach przelatujących nad naszymi głowami - lecz zauważalne zmiany wydawały się bardzo nieznaczne.

LITERATURA


1/ Barnaby F.: Mikroelektronika i wojna, Przegląd Techniczny, nr 19 /4032/, str. 26-29 /8.05.1983/

2/ Barnaby F.: Zastosowanie mikroelektroniki w urządzeniach wojskowych, Informatyka, nr 7-8, str. 5-7 /1983/

3/ Evans J.: Wpływ komputeryzacji na miejsce pracy, Informatyka, nr 5, str. 2-7, nr.6, str. 11-15 /1983/

4/ King A.: Następna rewolucja przemysłowa czy tylko nowa technologia? Przegląd Techniczny, nr 4 /4017/, str. 9-12 /1983/

5/ King A.: Mikroprocesor - klucz do Utopii? Przegląd Techniczny, nr 5 /4018/, str. 35-38 /1983/

6/ King A.: Mikroelektronika a suwerenność w skali świata, Przegląd Techniczny, nr 6 /4019/, str. 34-35 /1983/

7/ King A.: Mikroelektronika i przemysł a zróżnicowanie państw, Przegląd Techniczny, nr 7 /4020/, str. 39-40//1983/

8/ King A.: Na lepsze czy gorsze? Przegląd Techniczny, nr 8 /4021/, str. 39-40 /1983/

9/ King A.: Na dobre i złe? Polityka, nr 5 /1396/, str. 11 /1934/

10/ Schaff A.: Zajęcie kontra praca, Przegląd Techniczny, nr 9 /4022/, str. 38-41 "/1983/

11/ Zalewski J.: Nowa siła, Informatyka, nr 4, str. 34-36 /1983/

12/ Kastory B.: Rozwój ku przestarzałości, Życie Warszawy, nr 19, 24 stycznia 1983

13/ Rostocki M.: Rewolucja albo degradacja, Przegląd Techniczny, nr 28-29 /4012-4013/, str. 23-24 /1982/

Piotr Zapendowski

Instytut Technologii Elektronowej NPCP
Al.Lotników 32/46
02-668 Warszawa

# ZASADNIENIA KOMUNIKACJI Z OPERATOREM. MINIMALIZACJA INTERFEJSU MIKROKOMPUTER - OPERATOR.

## 1. Wstęp

Współczesne zastosowania technik obliczeniowych w wielu wypadkach wymagają konstruowania dużych, otwartych sieci komputerowych, złożonych z autonomicznych podsystemów. Każdy z takich podsystemów powinien dysponować następującymi typami układów pośredniczących, obsługującymi:

- połączenia z innymi podsystemami,
- urządzenia peryferyjne służące celom dokumentacji,
- połączenie z operatorem /klawiatura, wyświetlacze, monitor/.

W innych wypadkach ograniczenia w kosztach, zasilaniu lub wymiarach nakładają na konstruktora systemu konieczność rezygnacji z zastosowania monitora jako źródła informacji o realizacji przez system poleceń operatora.

W obu wymienionych wyżej sytuacjach istnieje potencjalna możliwość wyeliminowania jednego z głównych składników w kosztach, objętości i pobieranej mocy - monitora alfanumerycznego. Czy tego typu operacja będzie możliwa - o tym zadecyduje konstruktor przez zastosowanie lub nie, odpowiednich metod projektowania opracowywanego systemu.

## 2. Analiza parametrów eksploatacyjnych konkurencyjnych terminali.

Analizę przeprowadzono drogą porównania parametrów monitora ekranowego z grupą 12 wyświetlaczy numerycznych, która to ilość jak wynika z rozważań autora, jest wystarczająca dla zastąpienia funkcji monitora.

Powszechnie używany wyświetlacz numeryczny typuCQZP 12 jest siedmiosegmentowa kostka o wymiarach 5mm x 10mm x 19mm, o objętości ok. 1 cm sześć. Monitor ekranowy MERA-7910 ma wymiary 38cm x 36cm x 40cm i 4600 razy przewyższa całkowitą objętość dwunastu wyświetlaczy.

Ten sam monitor w roku 1982 kosztował 370 tys.zł. Zestaw dwunastu wyświetlaczy był wtedy osiemnaście razy tańszy.

Jeden segment wyświetlacza CQZP-12 pobiera moc rzędu 25 mW. 7 segmentów tego wyświetlacza pobiera moc nie większą niż 200 mW. Identyczną moc pobiera 12 wyświetlaczy pod warunkiem dynamicznego ich uruchamiania. Moc przeciętnie pobierana przez monitor MERA-7910 jest około 350 razy większa, konieczne jest zasilanie z sieci.

Możliwości tego monitora pozwalają na jednoczesne wyświetlanie 1920 znaków /24 linie x 80 znaków/, czyli 160 razy więcej niż grupa 12 wyświetlaczy.

Na ekranie monitora można wyświetlić znaki, złożone z dowolnej kombinacji punktów świetlnych, umieszczonych w matrycy 7 x 6 punktów. W praktyce dostępne są jednak tylko znaki objęte kodem ASCII, których jest 96 wliczając małe i duże litery. Każdy programista dobrze jednak wie, że w danym systemie komputerowym używa się albo małych, albo dużych liter, a jednych i drugich w wyjątkowych wypadkach.

Pojedynczy wyświetlacz dysponuje ilością 128 kombinacji zapalonych i zgaszonych segmentów. Wynika to z faktu, że prostota konstrukcji pozwala samodzielnie projektować postać znaków. Doświadczenia przeprowadzone przez autora dowiodły, że spośród alfabetu łacińskiego jedynie K, Q, Y i Z nie dadzą się wymodelować. Wszystkie cyfry mają oczywiście swoje odpowiedniki wśród kombinacji. Jak również nieliczne znaki specjalne. Dodatkowo niektóre "małe" litery można wyświetlać na dwa sposoby - "górny" i "dolny".

Praca przy monitorach ekranowych została przez Ministerstwo Przemysłu Maszynowego zaliczona do pierwszej kategorii zatrudnienia. O wyświetlaczach numerycznych rozporządzenie Ministra nie wspomina.

Podsumowując wyniki rozdziału można by stwierdzić, że instalując monitor ekranowy kosztem utraty zdrowia, 18-krotnego wzrostu ceny, 350-krotnego wzrostu pobieranej mocy i 4600-krotnego wzrostu objętości uzyskujemy możliwość wyświetlania 160 razy więcej znąków, niż przysługując się zestawem dwunastu wyświetlaczy cyfrowych. W wielu zastosowaniach profesjonalnych jest to zysk decydujący dla właściwego wykorzystania dostępnych zasobów.

## 3. Metody projektowania oprogramowania

Jak już powiedziano wyżej podstawową zaletą monitora ekranowego jest możliwość wyświetlania dużej ilości napisów informacyjnych, które przez dłuższy czas mogą pozostać widoczne dla operatora. Napisy te mają więc dwojaki charakter:

- komunikujący o rozpoczęciu lub zakończeniu realizacji określonego polecenia,
- przedstawiający zawartości określonych rejestrów pojedyńczych lub indeksowych, łącznie z nazwami tych rejestrów.

Przechowywanie pierwszego rodzaju wiadomości jest z reguły zbędne, gdyż operator musi mieć z góry zaplanowany tryb postępowania /innymi słowy musi umieć posługiwać się systemem, znać jego funkcje i możliwości/. Niezbędne jest więc tylko podanie komunikatu, która z czynności została zrealizowana i na jakim etapie system się aktualnie znajduje.

Trudność stanowi konieczność stosowania skrótów, starannego dobierania liter tak, aby całość była czytelna nawet dla operatora z minimalnym stażem /tzn. takiego, któremu raz przeczytano wyświetlony tekst/. Z eksperymentów, przeprowadzonych w gronie pracowników Zespołu /ok. 10 osób/ wynika, że teksty formowane w wyrazy lub ich skróty o długości nie mniejszej od trzech liter mogą nie być czytelne przy pierwszym czytaniu. Jednak już drugie czytanie wykonywane jest bezbłędnie. Trudności polegały przy tym raczej na opanowaniu dość specyficznego dla wyświetlaczy alfabetu niż na odczytywaniu całych wyrazów. Test polegał na dwukrotnym odczytaniu sześciu komunikatów złożonych z dwóch 3-literowych skrótów.

W celu pozostawienia projektantowi pewnej swobody autor zaleca jednak stosować skróty 4-literowe jako bardziej czytelne. Daje to możliwość od zapisania w całości wyrazu 4-literowego do ok. 500 różnych skrótów dla wyrazu 12-literowego. Pozwala również swobodnie unikać liter charakterystycznych dla polskiego alfabetu.

Innym zasadnieniem jest prezentacja zawartości określonych rejestrów. Taka prezentacja musi spełniać kryteria łatwości dostępu, czytelności i dostatecznego zakresu dla liczb umieszczanych w tych rejestrach. Łatwość dostępu uzyskać można przez wprowadzenie przyporządkowania funkcji lub rejestrów poszczególnym klawiszom. Wtedy dostęp uzyskamy po prostu przez wciśnięcie odpowiedniego klawisza, a w przypadku rejestrów indeksowych dodatkowo wybranie numeru indeksu. W takiej sytuacji nie istnieje nawet konieczność zapisywania przeglądanych wartości.

Przez czytelność autor rozumie czytelność oznaczeń rejestrów. To zasadnienie omówiono wyżej.

Zakresy liczb są określone zastosowaniami i tak dla potrzeb uruchamiania systemów mikroprocesorowych wystarczające są do czterech cyfr szesnastkowych dla indeksów i ta sama ilość dla zawartości rejestrów /4 znaki nazwy + 4 cyfry indeksu + 4 cyfry zawartości/. W sieciach istnieje konieczność przesyłania danych, będą zatem 4 znaki określające funkcje przesyłania, 4 znaki nazwy przesyłanego rejestru i 4 cyfry ewentualnego indeksu /przy założeniu indeksów nie większych niż czterocyfrowe/.

Ważną rolę w systemie spełnia funkcja sterująca automatycznym przejściem do następnego lub poprzedniego rejestru lub indeksu, istotnie oszczędzająca czas i ilość decyzji operatora, związanych z kolejnością wyboru klawiszy.

## 4. System operacyjny

Podstawową cechą systemu operacyjnego jest jego modularność. Musi on się składać z szeregu prostych, o możliwie najmniejszej ilości stopni wywołania, funkcji. Funkcje te będzie można przypisać poszczególnym klawiszom. Celem uniknięcia nadmiernego rozbu-

dowania klawiatury również klawiszom cyfrowym można przypisać podwójne znaczenie. Możliwość ta wynika z faktu, że wprowadzanie danych jest konieczne dopiero po wybraniu określonego polecenia.

Operator powinien spodziewać się określonych informacji w określonych miejscach, prowadzi to do podziału wyświetlaczy na grupy, z których każda będzie wypełniać określone zadania, np:

- wyświetlacze 1-4 mogą być przeznaczone dla wypisywania nazw funkcji lub rejestrów indeksowych,
- wyświetlacze 5-8 dla nazw rejestrów lub indeksów,
- wyświetlacze 9-12 dla zawartości rejestrów.

Minimalny zestaw czterech funkcji sterujących powinien zawierać funkcję syngalizującą zakończenie cyklu poleceń, dwie funkcje sterujące automatycznym zwiększeniem lub zmniejszeniem indeksu oraz separujących dane, wyświetlane na różnych grupach wyświetlaczy, oraz funkcje usuwającą ostatnio wprowadzoną wartość.

Wszystkie te zabiegi stają się zbędne lub niemożliwe do zrealizowania jeżeli zamiast wyświetlaczy użyjemy monitora alfanumerycznego.

## 5. Podsumowanie.

Jak wynika z analizy przeprowadzonej w częściach 2 i 3 zasadnicze różnice między konkurencyjnymi systemami sprowadzają się do problemu porozumienia się z operatorem /lingwistyka/, szybkości i dostępu do danych /modelowanie systemu operacyjnego według recepty z części 4 daje tu przewagę wyświetlaczom/ i sposobu magazynowania informacji /ekran jest magazynem dla 1920 znaków/.

Jako dowód skuteczności przedstawionych w tym artykule metod można przedstawić system mikroprocesorów zrealizowany wspólnie przez mgr inż. Sławomira Pilarskiego i autora niniejszego artykułu /patrz 1 /, gdzie ze względu na specyfikę problemu użyto ośmiu wyświetlaczy.

Na zakończenie trzeba dodać, że intencją autora nie była wydanie opinii o wyższości jednego z przeanalizowanych systemów nad drugim, lecz jedynie próbą skonfrontowania różnic i podobieństw.

LITERATURA:

1 Pilarski S., Zapendowski P.: Emulator SPZE-48/41A- właściwości funkcjonalne i użytkowe", Materiały Seminaryjne V Szkoły Mikroprocesorowej, Łódź 5-7.III.1984r., w przygotowaniu.

Dz. Uic.. MH ; PM

. Poz 9.. nr 7 z dnia 19 V 1983

Dział XIV PRACE RÓŻNE POZ 5 /5 operator mikroskopów i monitorów ekranowych

EO

CSI – *to kompleksowa, wielopoziomowa i spójna sieć modułów szkoleniowych*

CSI – *to szybkie dostosowanie tematyki szkolenia do bieżących potrzeb twórców i użytkowników informatyki*

CSI – *to integracja środowiska informatycznego*

CSI – *to ciągle rosnący poziom szkolenia*

CSI – *to wymiana doświadczeń z ośrodkami szkolenia informatycznego w kraju i poza granicą*

CSI – *to zaspakajanie ilościowych potrzeb w zakresie szkolenia informatycznego*

ZETO-Łódź z. 67, 200