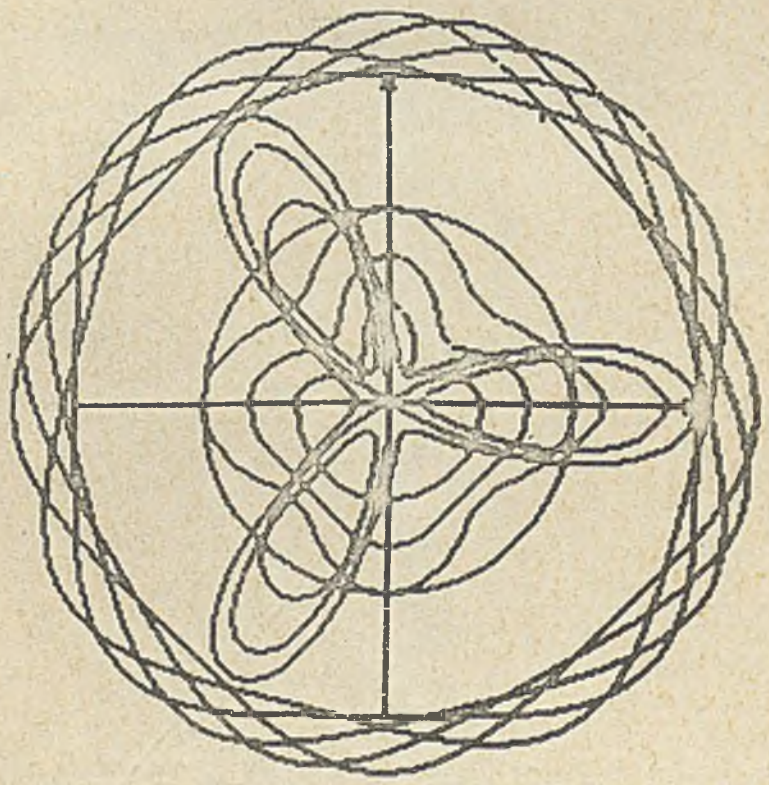
biuletyn informacyjny

1-2
'82

MERA IMM

NAUKI
I TECHNIKI
KOMPUTEROWE



Instytut Maszyn Matematycznych

Branżowy Ośrodek INTE

Podpis pod rysunkiem na okładce: Rodzina krzywych algebraicznych 4 i 6 stopnia wygenerowana za pomocą procedur języka graficznego PSG opracowanego w Pracowni Grafiki Komputerowej Instytutu Maszyn Matematycznych

Druk okładki IMM zam. 92/82 nakł. 500 egz.

Biuletyn Informacyjny NAUKI I TECHNIKI KOMPUTEROWE

Rok XX Nr 1-2 1982

Spis treści

Содержание

Contents

D W U M I E S I Ę C Z N I K

Wydaje:

I N S T Y T U T M A S Z Y N M A T E M A T Y C Z N Y C H

Branżowy Ośrodek Informacji Naukowej Technicznej i Ekonomicznej

Adres redakcji: ul.Krzywickiego 34, 02-078 Warszawa
tel.28-37-29 lub 21-84-41 w.244

dr inż. Stanisława BONKOWICZ-SITTAUER
Instytut Maszyn Matematycznych

Prowadzenie i rozpowszechnianie informacji o oprogramowaniu PIRIO

Uwagi wstępne

Opracowanie niniejsze omawia wyniki pewnego etapu pracy, którą wykonano w Instytucie Maszyn Matematycznych na rzecz Zakładów im. J. Krasickiego; całość dotyczy oprogramowania minikomputera MERA 400, a wymieniony etap obejmował wypracowanie i praktyczne sprawdzenie zasad i sposobów zapewnienia szerokiej informacji o oprogramowaniu tego minikomputera. Powstała w ten sposób koncepcja PIRIO/MERA 400, omówiona w części I niniejszego opracowania. Tworząc koncepcje PIRIO/MERA 400 wykorzystano wyniki analizy sposobów rozpowszechniania informacji o oprogramowaniu znanych firm komputerowych zagranicznych (IBM, ICL, PDP, WANG) i krajowych (RIAD, ODRA), jak również wyniki analizy obecnego stanu informacji o oprogramowaniu MERA 400. Przedmiotem zainteresowania były również opracowane w kraju koncepcje systemów informowania o oprogramowaniu komputerów, katalogi programów, itp. Uznano też, że skuteczność działania PIRIO wymaga zarówno wyraźnego sprecyzowania celów, jak i zaproponowania bardzo prostych sposobów jego realizacji.

Tak więc PIRIO/MERA 400 jest zbiorem zasad postępowania, których cel, środki i metody są kolejno omówione w tym opracowaniu. Zostanie tu opisany zakres informacji objętych PIRIO, sposoby ich pozyskiwania, forma przechowywania, wydawania i rozpowszechniania. Wymagało to zidentyfikowania rodzajów oprogramowania objętych PIRIO oraz przyjęcia właściwych dla potrzeb PIRIO - klasyfikacji rodzajów informacji o oprogramowaniu.

Podstawową przesłanką niniejszych rozważań było przekonanie, że niezbędne jest natychmiastowe podjęcie prac nad realizacją PIRIO/MERA 400 oraz, że prace te obecnie powinny być oparte wyłącznie na metodach tradycyjnych, wykorzystujących istniejące rezerwy organizacyjne. Natomiast w przyszłości, po wyczerpaniu tych rezerw, na podstawie doświadczeń zebranych w pierwszym okresie, okaże się zapewne konieczne takie usprawnienie działania PIRIO, które będzie możliwe jedynie metodami wspomaganymi komputerowo. Przewidując takie komputerowe wspomaganie procesu informacyjnego należy już obecnie prowadzić prace eksperymentalne i odpowiednie analizy, które w tradycyjnym sposobie realizacji PIRIO pozwolą uniknąć rozwiązań trudnych do skomputeryzowania. Zwłaszcza należy unikać takich rozwiązań, które przy wprowadzeniu wspomagania komputerowego wymagałyby zmian wyraźnie odczuwalnych przez odbiorców informacji. Znajdą się więc w

niniejszym opracowaniu również założenia dotyczące komputerowego wspomagania PIRIO/MERA400 wraz z wstępnym projektem odpowiedniego oprogramowania.

Opierając się na przedstawionej w części I niniejszego opracowania koncepcji PIRIO/MERA 400 w części II przedstawiono koncepcję PIRIO/IMM. Różnice między nimi wynikają z odmiennych celów oraz różnych warunków pozyskiwania informacji, sposobów jej przechowywania i rozpowszechniania. W obu jednak wypadkach jest to ta sama klasa procesów informacyjnych, tj. "prowadzenie i rozpowszechnianie informacji o oprogramowaniu".

Część I. PIRIO/MERA 400

1. Cele

Zasadniczym celem PIRIO/MERA 400 jest zaspokojenie zapotrzebowania na informację o oprogramowaniu tego minikomputera, zarówno producenta, jak i użytkowników.
Producent minikomputera MERA 400, jako jednocześnie generalny dostawca systemów sprzętowo-programistycznych, potrzebuje tej informacji do analizy kierunków zastosowań, a na ich podstawie do wytyczania kierunków rozwoju prac nad sprzętem i oprogramowaniem firmowym oraz prowadzenia długofalowej przemyślanej polityki koordynacji wszelkich prac nad oprogramowaniem maszyny MERA 400. Ponadto dobra informacja o działającym oprogramowaniu jest reklamą zarówno tego oprogramowania, jak i samego sprzętu. A w obecnej sytuacji ogólno-gospodarczej reklama taka bez wątpienia okaże się niezbędna.

Natomiast użytkownikom minikomputera MERA 400, tym, co sprzęt ten już posiadają, otrzymywanie systematycznej, aktualnej informacji o wszystkich klasach i rodzajach istniejącego i powstającego oprogramowania będzie umożliwiało optymalizację wykorzystywania tego sprzętu, podejmowania decyzji czy potrzebne oprogramowanie kupić, czy robić samemu. Oczywiście przy tych decyzjach istotne będzie aby kupno oprogramowania i jego instalacja były zawsze tańsze i szybsze od prac własnych. Ale jest to zagadnienie wykraczające poza temat tego opracowania. Równocześnie dzięki PIRIO szeroko rozpowszechniania informacja o oprogramowaniu użytkowym, będzie reklamą tego oprogramowania, szansą na jego wielokrotną sprzedaż, a więc odpowiednie zrefundowanie poniesionych kosztów. Wreszcie PIRIO/MERA 400 użytkownikom kupującym sprzęt pozwoli na dokładne określenie zakresu stosowania minikomputera MERA 400, jego przydatności i podjęcie decyzji o nabyciu sprzętu dla własnych potrzeb.

Tak więc sprawność działania PIRIO/MERA 400 leży w interesie zarówno producentów sprzętu jak i jego użytkowników.

2. Zakres informacji

W PIRIO/MERA 400 zawarte będą dane o wszystkich rodzajach oprogramowania działającego na tym minikomputerze, o twórcach tego oprogramowania i jego użytkownikach.

Podejmując pracę nad koncepcją PIRIO przede wszystkim należało dokonać identyfikacji rodzajów oprogramowania. Przyjęto więc po pierwsze klasyfikację poziomów oprogramowania, a mianowicie:

- oprogramowanie podstawowe obejmujące systemy operacyjne i procesory systemowe (np. translatory)
- biblioteki: ogólne, podstawowe i pomocnicze oraz biblioteki związane z poszczególnymi językami (np. FORTRAN, BASIC, COBOL, itp.) z rozróżnieniem ich merytorycznego zakresu (np. matematyczna, statystyczna, graficzna, itp.)
- oprogramowanie użytkowe wg klasyfikacji przedmiotowej, gdzie występują zarówno pojedyncze programy, jak i ich pakiety lub całe problemowo ukierunkowane systemy.

Przyjęto też rozróżnienie oprogramowania ze względu na charakter źródła, gdzie oprogramowanie powstało i sposób rozpowszechniania tego oprogramowania, a mianowicie:

a) oprogramowanie powstało u producenta sprzętu
b) w innej formie, ale na zamówienie producenta
c) opracowane bez zamówienia producenta sprzętu, ale przejęte przez niego do rozpowszechniania
d) wytworzone poza producentem i nie przejęte przez niego.

W odniesieniu do klas a) i b) oprogramowanie to może być przez producenta sprzętu włączone do standardu sprzętowo-programowego lub sprzedawane poza standardem. Zawsze jednak dostawca tego oprogramowania, tzn. producent sprzętu ponosi pełną odpowiedzialność za jego poprawność, jakość i niezawodność i gwarantuje serwis dostarczanego oprogramowania.
W odniesieniu do klasy c) producent sprzętu staje się też dostawcą przejętego oprogramowania (jest firmą rozpowszechniającą) i tym samym przejmuje na siebie pewną odpowiedzialność za jego jakość, przy czym stopień i zakres tej odpowiedzialności powinien być regulowany w każdej konkretnej sytuacji odpowiednią umową. Należy się też spodziewać, że na ogół umowy te będą przewidywały obciążanie obowiązkami serwisowymi raczej firmy autorskiej. Wkraczamy tu w zagadnienia prawne związane z handlem oprogramowaniem, w zasadzie dotychczas nie rozstrzygnięte i nie uporządkowane. Brak jest nie tylko przepisów szczegółowych ale i stosownych precedensów.
Wreszcie w odniesieniu do klasy d) należy stwierdzić, że producent sprzętu i animator PIRIO/MERA 400 nie ponosi żadnej odpowiedzialności ani za jakość samego oprogramowania ani nawet za wiarygodność informacji o tym oprogramowaniu. Jest on w tym wypadku tylko rozpowszechniającym tę informację. Natomiast pełną odpowiedzialność i obowiązki serwisowe ponosi firma autorska, sama lub wspólnie z firmą rozpowszechniającą.

Wymienione klasy a) - d) mogą odnosić się do wszystkich uprzednio wymienionych poziomów oprogramowania.

Również w każdym oprogramowaniu objętym działaniem PIRIO rozróżnia się trzy zakresy informacyjne:

- karta informacyjna
- instrukcja użytkownika
- dokumentacja robocza

Karta informacyjna danego oprogramowania zawiera informacje dla potencjalnych użytkowników, tj.takich,którzy dopiero zamierzają nabyć dane oprogramowanie wybierając najodpowiedniejsze dla swoich potrzeb. Karta informacyjna zawiera więc podstawowe wiadomości o zakresie i funkcjach danego oprogramowania, o sposobie jego eksploatacji oraz o możliwościach nabycia.Jest to informacja bardzo skondensowana,zawiera jedynie zasadnicze elementy.Wzór karty informacyjnej pokazuje rys.1.

PIRIO/MERA 400	KARTA INFORMACYJNA
1 Symbol katalogowy	2 Formalna (handlowa, symboliczna) nazwa programu, pakietu, biblioteki, systemu
3 Poziom oprogramowania	
4 Przeznaczenie	5 Klasyfikacja problemowa
	6 Data opracowania
7 System operacyjny	8 Język, translator
	9 Programy współpracujące
10 Szczegółowa konfiguracja	11 Dokumentacja użytkowa (nazwa, symbole)
12 Firma autorska	14 Nazwiska autorów
	15 Adres firmy autorskiej
13 Symbol KARTY ADRESATA	16 Telefon:
17 Firma rozpowszechniająca	19 Adres firmy rozpowszechniającej
18 Symbol KARTY ADRESATA	20 Telefon
	21 Telex
22 Inne informacje niezbędne dla zamawiającego	
23 Opracował informację ; nazwisko	25 Przygotował informację do rozpowszechniania Nazwisko
24 Data	26 Data
ZAKŁADY WYTWÓRCZE PRZYRZĄDÓW POMIAROWYCH I SYSTEMÓW MINIKOMPUTEROWYCH im. J. Krasickiego 02-231 WARSZAWA ul. Łopuszańska 117/123	

Rys.1. Wzór karty informacyjnej PIRIO/MERA 400

Instrukcja użytkowania, zwana też dokumentacją użytkową, zawiera informacje dla faktycznego użytkownika. A więc, przede wszystkim dokładne informacje o merytorycznym zakresie działania programu i o sposobach, metodach oraz algorytmach zastosowanych w programie. Dla programów użytkowych owe merytoryczne funkcje powinny być podawane w formie (w formaliźmie) przyjętej w danej dziedzinie zastosowań. Informacje te muszą być tak podane aby użytkownik mógł ocenić, w jakim stopniu dane oprogramowanie odpowiada jego wymaganiom merytorycznym.

Ponadto dokumentacja użytkowa zawiera wszelkie szczegóły dotyczące sposobu korzystania z danego oprogramowania. A więc: sposób jego wywołania, przygotowania i wprowadzenia danych wraz z ewentualnymi formularzami danych, czy charakterystykę innego sposobu ich przygotowywania, zasady interpretacji wyników i komunikatów, a zwłaszcza komunikatów o błędach. Konieczne też jest dokładniejsze niż w karcie informacyjnej omówienie warunków sprzętowo-programistycznych niezbędnych do działania danego oprogramowania.
W odniesieniu do programów eksploatowanych w trybie dialogowym (konwersacyjnym interaktywnym) w dokumentacji użytkowej musi być podany szczegółowo opis wszystkich ścieżek dialogu, ich interpretacja merytoryczna i uzasadnienie stosowania.

Dokumentacja robocza, zwana też techniczną lub szczegółową, zawiera bardzo dokładne i kompletne wiadomości o budowie i strukturze danego oprogramowania, o rozwiązaniach wszystkich fragmentów, schematy, listingi, pełne wykazy zmiennych, wydruki z przebiegów testowych, itp. Wszystko to musi być opracowane z taką dokładnością, aby konserwacja danego oprogramowania mógł wykonywać każdy kompetentny pracownik, a nie jedynie autor danego oprogramowania.

Tak więc dokumentacja robocza musi zawierać: dokładne przedstawienie struktury oprogramowania, tj. podział na części, poziomy, moduły, sposoby komunikacji między modułami danego poziomu i ewentualnie sposoby komunikacji między poziomami, dokładnie zilustrowane schematami i rysunkami. Muszą być również dokładnie przedstawione algorytmy działania programu odpowiadające metodom opisanym w instrukcji użytkowania, z powołaniem się na wymienione tam wzory i sposoby postępowania oraz z wskazaniem jak wzory te zostały dostosowane do potrzeb programowania, należy np. podać symbole użyte w programie oraz ich odpowiedniki w zapisie tradycyjnym. Najlepszą formę dla tych ostatnich są słowniki symboli zestawione w dużych programach (pakietach, systemach) osobno dla poszczególnych modułów elementarnych* . Przykład słownika symboli pokazano na rys. 2.

Są dwie zasadnicze przyczyny tak dużego skondensowania tej informacji. Z jednej strony przemawiają za tym możliwości percepcyjne odbiorców, a z drugiej względy organizacyjne animatorów. Należy bowiem uznać za pewnik, że uzasadnienie całego przedsięwzięcia, jakim jest PIRIO/MERA 400 zależy od osiągnięcia pewnego progu szybkości obiegu informacji. Otóż owo bardzo drastyczne ograniczenie danych zawartych w karcie informacyjnej ma na celu zapewnienie jak największej sprawności gromadzenia, przechowywania, przetwarzania i rozpowszechniania tychże danych.

---

* Przyjmuje się pojęcie modułu jako wyróżniającej się części oprogramowania; modułem elementarnym nazywa się najmniejszą formalną część oprogramowania nie wołającą innych, a modułami złożonymi - konstrukcje programowe o różnie rozbudowanej strukturze.

Nazwa modułu, typ, znaczenie, zakres, inne					
Lp.	Symbol	Typ	Rodzaj	Charakter	Znaczenie
.		int. real. dubl. logical "	zm. prosta tablica nazwa wspólnego obszaru, nazwa procedury,	dane, wyniki, parametr, formalny, zmienna lokalna, zmienna robocza, jeżeli zmienna ze wspólnego obszaru to jego nazwa	fizyczne lub organizacyjne znaczenie danego symbolu z powołaniem się na wzory. Gdy zmienna ze wspólnego obszaru to z jakiego i jakim zmiennym z innych modułów odpowiada. Gdy robocza i ma parę funkcji (znaczeń w danym module) to podać wszystkie

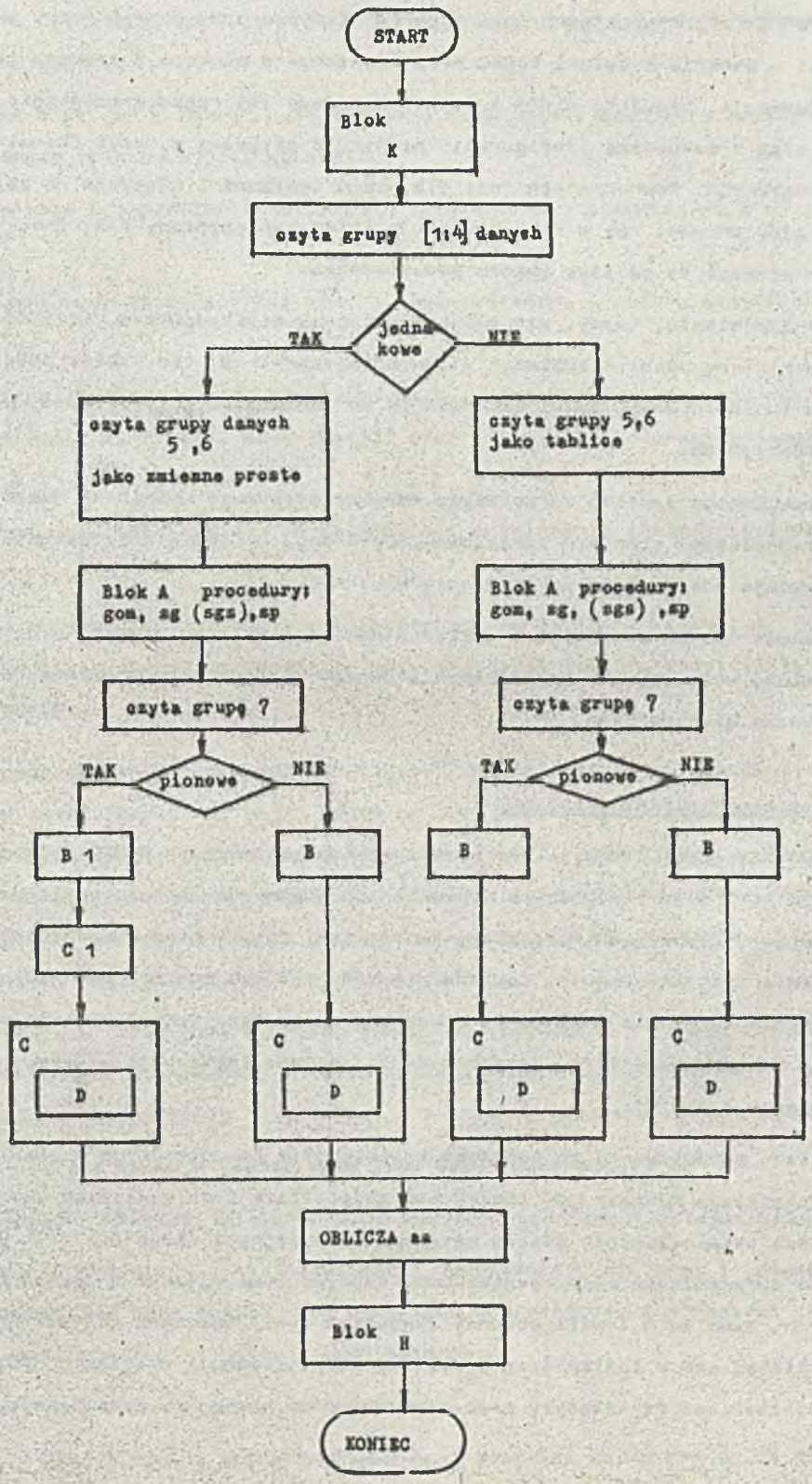
Rys. 2. Przykład słownika symboli

Bardzo ważnym elementem dokumentacji roboczej są schematy blokowe, zarówno ogólne, całego danego oprogramowania (rys. 3) jak i odpowiednio coraz bardziej szczegółowe schematy podprogramów - modułów złożonych i elementarnych (rys. 4 i 5). W skład dokumentacji roboczej ponadto wchodzą wydruki tekstu programu i przebiegów kontrolnych. Do każdego takiego przebiegu kontrolnego musi być dołączony zestaw zastosowanych danych testowych. Zwłaszcza ważne są tu przebiegi pokazujące sposób reagowania programu na błędy.

Należy też podać tu charakterystyki czasów wykonania danych obliczeń oraz szacunkowe czasy przygotowania danych. Przy programach realizowanych z użyciem dialogu ważne są czasy reakcji, itp.

Pożądane jest również aby w dokumentacji roboczej znalazły się uwagi i wnioski autorów dotyczące danego oprogramowania, sugestie co do ewentualnych rozszerzeń czy modyfikacji, itp.

Należy również zalecać stosowania generalnej zasady, że w dokumentacji użytkowej nie można powoływać się na dokumentację roboczą. Natomiast odwrotnie jest to ze wszech miar wskazane, a to w celu uzyskania jak najmniejszej liczby powtórzeń.



Rys. 3. Schemat całości programu

Uznano, że systematycznym zbieraniem informacji należy objąć źródła powstawania oprogramowania typu "a", "b" i "c", traktując źródła typu "d" jedynie przy okazji (zob. pkt 2). Kierując się sformułowanymi w punkcie 1 celami PIRIO oraz omówionym w punkcie 2 przeznaczeniem poszczególnych poziomów informacji, przyjęto niżej wymienione zasady jej rozpowszechniania. Każdy nabywca systemu MERA 400, wraz z zakupioną konfiguracją otrzymuje aktualny w danym okresie podstąwowy zbiór KART INFORMACYJNYCH. Równocześnie jest dla niego tworzona i włączona do zbioru adresatów KARTA ADRESATA. Każdy adresat raz w roku otrzymuje zbiór suplementowy KART INFORMACYJNYCH, które to karty powinien włączać do swojego zbioru podstawowego.

Na specjalne zamówienie, każdy, nie tylko użytkownik minikomputera, może otrzymać odpłatnie w dowolnym czasie zarówno wykaz aktualnego stanu oprogramowania, tzn. zbiór podstawowy wraz z aktualnym zbiorem suplementowym, jak i odpowiednio do zamówienia przygotowane, problemowe zestawienie kart informacyjnych.

Wraz z zakupem danego systemu sprzętowego nabywca otrzymuje kompletną dokumentację użytkową oprogramowania stanowiącego standard sprzętowo-programowy. Podobnie przy zakupie dodatkowego oprogramowania otrzymuje odpowiednią dokumentację użytkową.

Natomiast dokumentacja robocza jest przechowywana w firmie autorskiej oraz w firmie rozpowszechniającej i służy wyłącznie do konserwowania danego oprogramowania. Dokumentacja ta nigdy nie jest przekazywana użytkownikom.

4. Komputerowe wspomaganie PIRIO/MERA 400

Przy dostatecznie dużej liczbie kart katalogowych realizowanie PIRIO wyłącznie w tradycyjnej formie stwarza zbyt duże trudności w wyszukiwaniu odpowiedniej informacji oraz we wprowadzaniu wszelkiego rodzaju uzupełnień, poprawek, zmian, itp. Należy wtedy zastosować komputerową formę przechowywania tych informacji. Zakłada się przy tym, że komputerowe wspomaganie dla PIRIO dotyczyć będzie wyłącznie poziomu kart katalogowych, uznając, że brak jest uzasadnienia ekonomicznego dla komputerowej formy przechowywania zarówno instrukcji użytkowania, jak i tym bardziej dokumentacji roboczej.

Tak więc należy przewidywać, że potrzeba zastosowania komputerowego wspomagania działalności PIRIO/MERA 400 istnieje wówczas gdy liczebność zbioru kart informacyjnych będzie tak duża, że wszystkie czynności eksploatowania zbioru metodami tradycyjnymi staną się zbyt uciążliwe. Innymi słowy celowość wprowadzenia wspomagania komputerowego musi wynikać z przesłanek organizacyjno - ekonomicznych. Mimo że w chwili obecnej (kwiecień 1981) celowość ta jest niemożliwa do oszacowania, to licząc się z możliwością przyszłej komputeryzacji działania PIRIO/MERA 400, należy przemyśleć zasadnicze jej aspekty oraz przeprowadzić niezbędne eskperymenty programistyczne.

Zakłada się więc, że realizacja komputerowego wspomagania PIRIO/MERA 400 wymagać będzie utworzenia i wprowadzenia w komputerowej postaci zbiorów stosowanych przy wersji tradycyjnej, tj. zbioru podstawowego i suplementowego, z tym że ten ostatni zawierać będzie tylko nowe karty

informacyjne. Ponadto w wersji komputerowej przewiduje się prowadzenie osobnego zbioru modyfikacji treści zbioru "podstawowego", jak również przewiduje się prowadzenie odrębnych zbiorów źródeł i zbiorów adresatów.

Przewiduje się więc, że w zakres komputerowego wspomagania działania PIRIO/MERA 400, oprócz założenia wymienionych zbiorów będzie wchodziło:

- bieżące uzupełnianie i poprawianie wszystkich tych zbiorów, oczywiście z wyjątkiem zbioru podstawowego
- okresowe, doroczne wydruki zawartości zbioru suplementowego i zbioru modyfikacji oraz zbioru adresatów
- okresowe, doroczne łączenie zbioru podstawowego z suplementowym, wprowadzanie do zbioru podstawowego modyfikacji wg "zbioru modyfikacji" oraz kasowanie (zerowanie) zawartości zbioru suplementowego i zbioru modyfikacji
- na żądanie wydruk zawartości zbioru podstawowego i ewentualnie zbiorów suplementowych i zbioru modyfikacji
- wyszukiwanie ze zbioru podstawowego i suplementowego, wg żądanego kryterium, odpowiedniego zestawu kart informacyjnych lub wprowadzenie modyfikacji ze "zbioru modyfikacji" i wygenerowania wybranego zestawu.

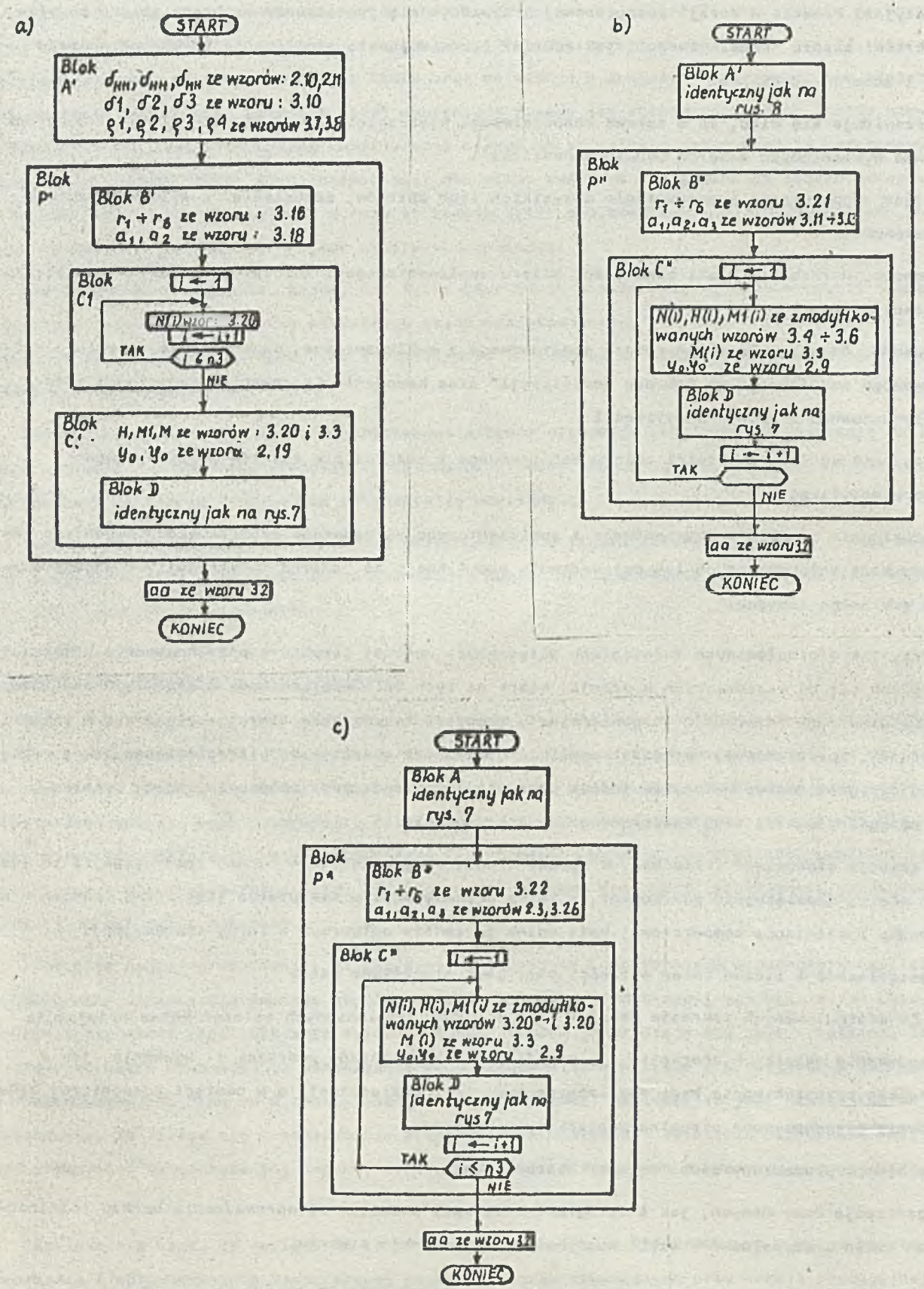
Przy tak sformułowanych założeniach dotyczących ogólnej struktury przechowywania informacji odnoszących się do zasadniczych operacji, które na tych informacjach będą musiały być realizowane, można uznać, że informacje zorganizowane w zbiorach tworzą bazę danych - ulokowaną w pamięci pomocniczej, np. dyskowej. Natomiast wszelkie wymienione operacje będą zrealizowane przez odpowiednie oprogramowanie. Należy tu jednak poczynić pewne dodatkowe założenia mające istotny wpływ na ogólny zakres tego oprogramowania, a mianowicie:

- informacje stanowiące treść bazy w sposób trwały są gromadzone w formie tradycyjnej i to jest ich stały, obowiązujący pierwowzór, a forma zapamiętania w komputerze jest formą roboczą; w wypadku zniszczenia komputerowej bazy można ją zawsze odtworzyć z formy tradycyjnej,
- częstotliwość i liczba zmian w treści bazy jest stosunkowo mała.

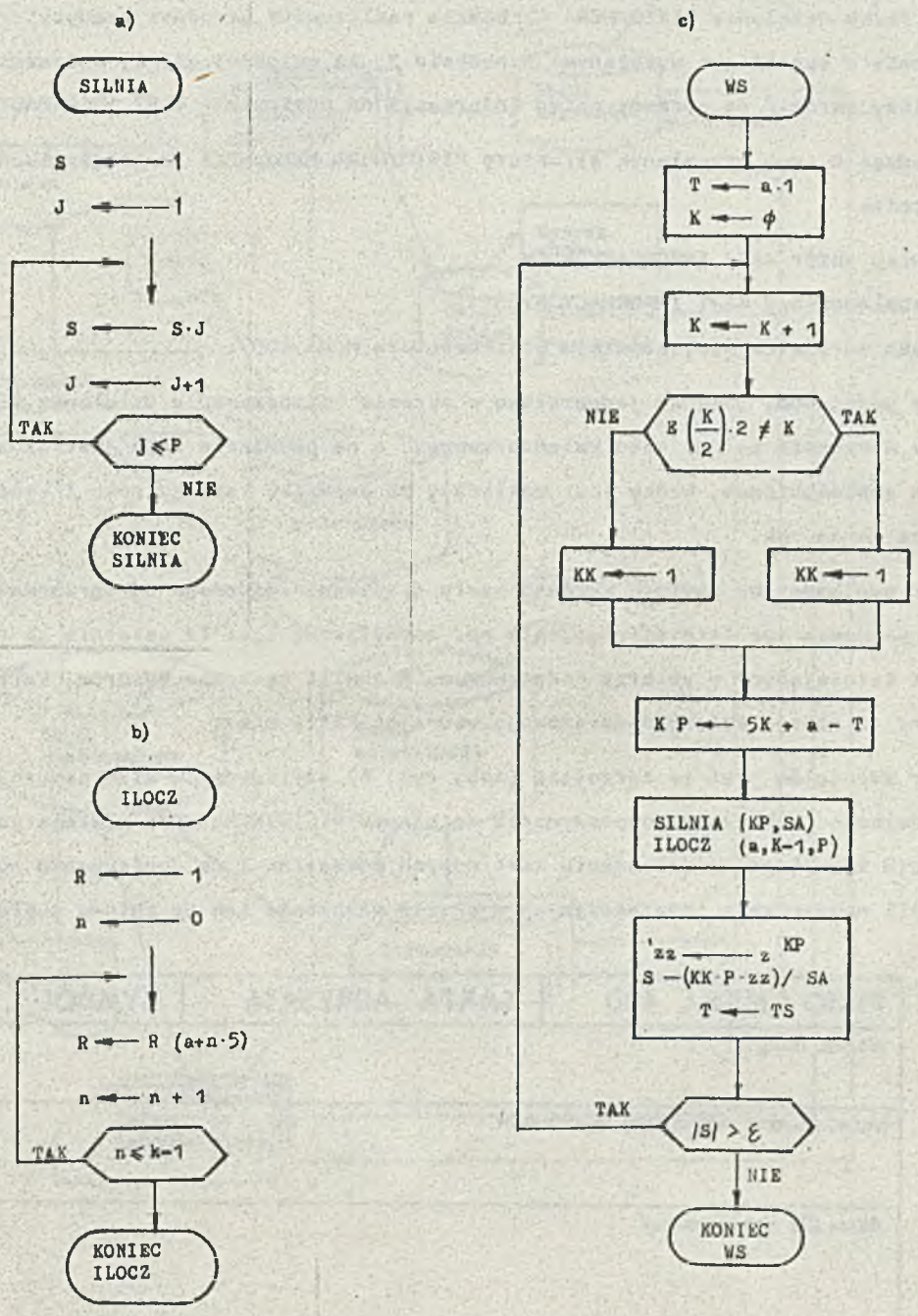
Ze sformułowanych zakresów działań oraz w wyniku wymienionych założeń można przyjąć, że

- dokonywanie wszelkich operacji, tj. modyfikacji, uzupełnień, poprawek i usuwania, jak i wszelkie przeszukiwania będą realizowane w pamięci operacyjnej, a w pamięci pomocniczej będą jedynie przechowywane aktualne postaci tych zbiorów,
- nie będzie prowadzony żaden rejestr historii bazy,
- organizacja bazy danych, jak i oprogramowania musi pozwalać na wprowadzenie bardzo różnorodnych zmian i uzupełnień - czyli musi być to organizacja otwarta.

Zasadniczą strukturę komputerowo wspomaganego PIRIO/MERA 400 ilustruje rys. 7.



Rys. 4. Przykłady schematów dużych modułów złożonych



Rys. 5. Przykłady schematów

a) b) - modułów elementarnych

c) - prostego modułu złożonego

3. Struktura i działania

Zgodnie z tym co już zasygnalizowano w uwagach wstępnych niniejszego opracowania uznano, że w obecnej fazie działania PIRIO/MERA 400 będzie realizowane metodami tradycyjnymi. Przyjęto również, zgodnie z sugestiami wyrażonymi w punkcie 2, że najwięcej uwagi, zwłaszcza we wstępnym okresie, należy zwrócić na sprawny obieg informacji na poziomie - KART INFORMACYJNYCH.

Wychodząc z tych przesłanek strukturę PIRIO/MERA 400 oparto na istnieniu następujących zbiorów - kartotek:

- podstawowy zbiór KART INFORMACYJNYCH
- zbiór suplementowy KART INFORMACYJNYCH
- kartoteka adresatów (użytkowników minikomputera MERA 400).

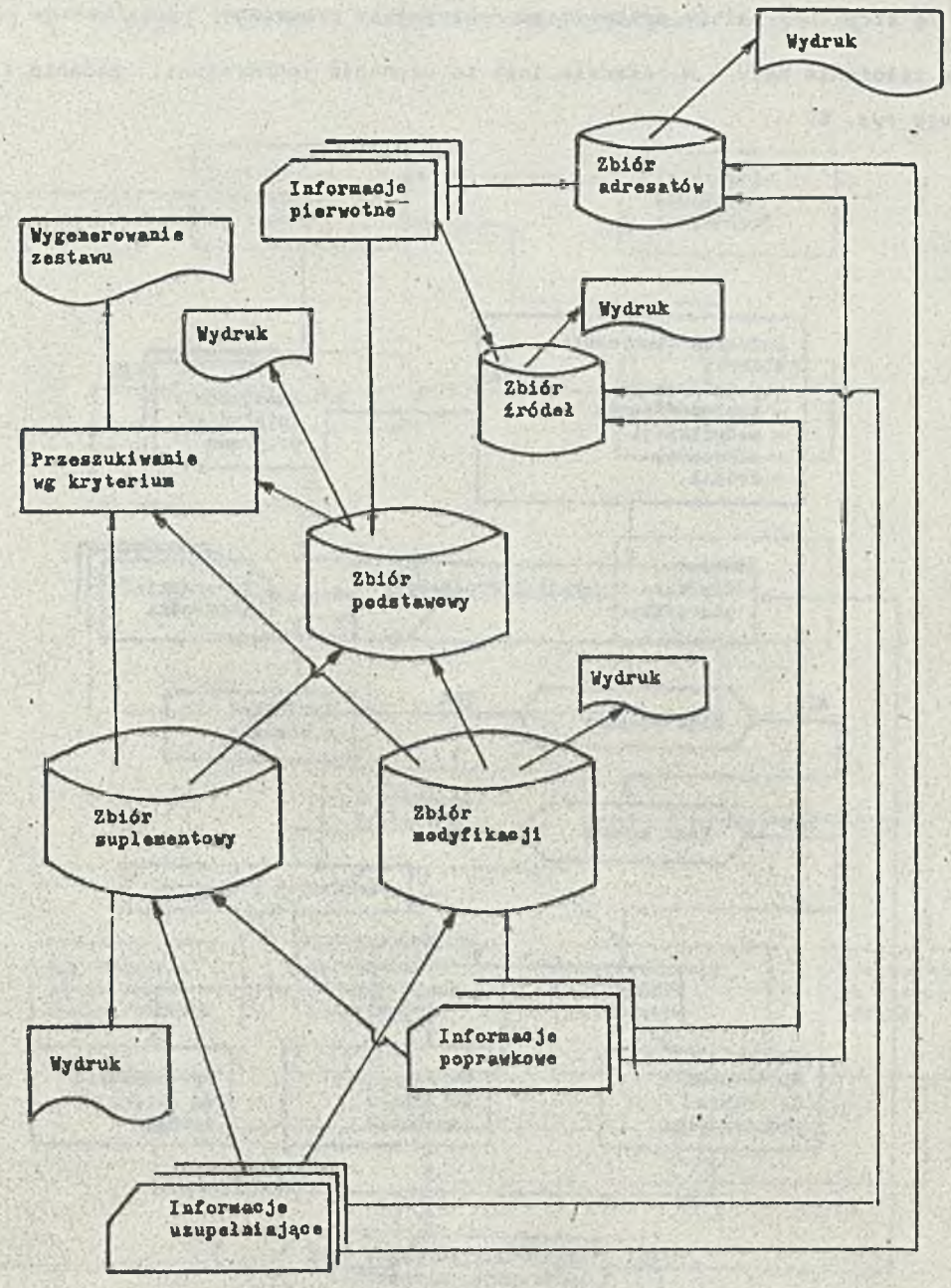
Zbiór podstawowy powstał jednorazowo w okresie rozpoczynania działania PIRIO. Pozostaje on niezmienny w okresie całego roku kalendarzowego, a na przełomie roku jest uzupełniany (łączony) ze zbiorem suplementowym, który jest zakładany na początku każdego roku i systematycznie uzupełniany przez tenże rok.

Zbiór suplementowy zawiera zarówno karty dla zupełnie nowego oprogramowania, jak i karty dla oprogramowania już istniejącego, ale np. zmodyfikowanego. Te ostatnie są formalnie "duplikatami" kart istniejących w zbiorze podstawowym. W chwili łączenia zbiorów, karty te wprowadza się jako jedyne do nowego zbioru podstawowego usuwając karty stare.

Zbiór adresatów jest to kartoteka (zob. rys. 6) użytkowników minikomputera MERA 400, utworzona jednorazowo w okresie rozpoczynania działania PIRIO/MERA 400 i systematycznie uaktualniana. Aktualizacja ta polega na dołączaniu kart nowych adresatów i na dopisywaniu na istniejących kartach symboli nowych kart informacyjnych w chwili włączania ich do zbioru suplementowego.

PIRIO / MERA 400	KARTA ADRESATA	SYMBOL
Nazwa firmy		
Nazwa komórki posiadającej minikomputer		
Adres dla korespondencji		
Nazwiska kompetentnych osób, telefony		
Symbole KART INFORMACYJNYCH tej firmy		

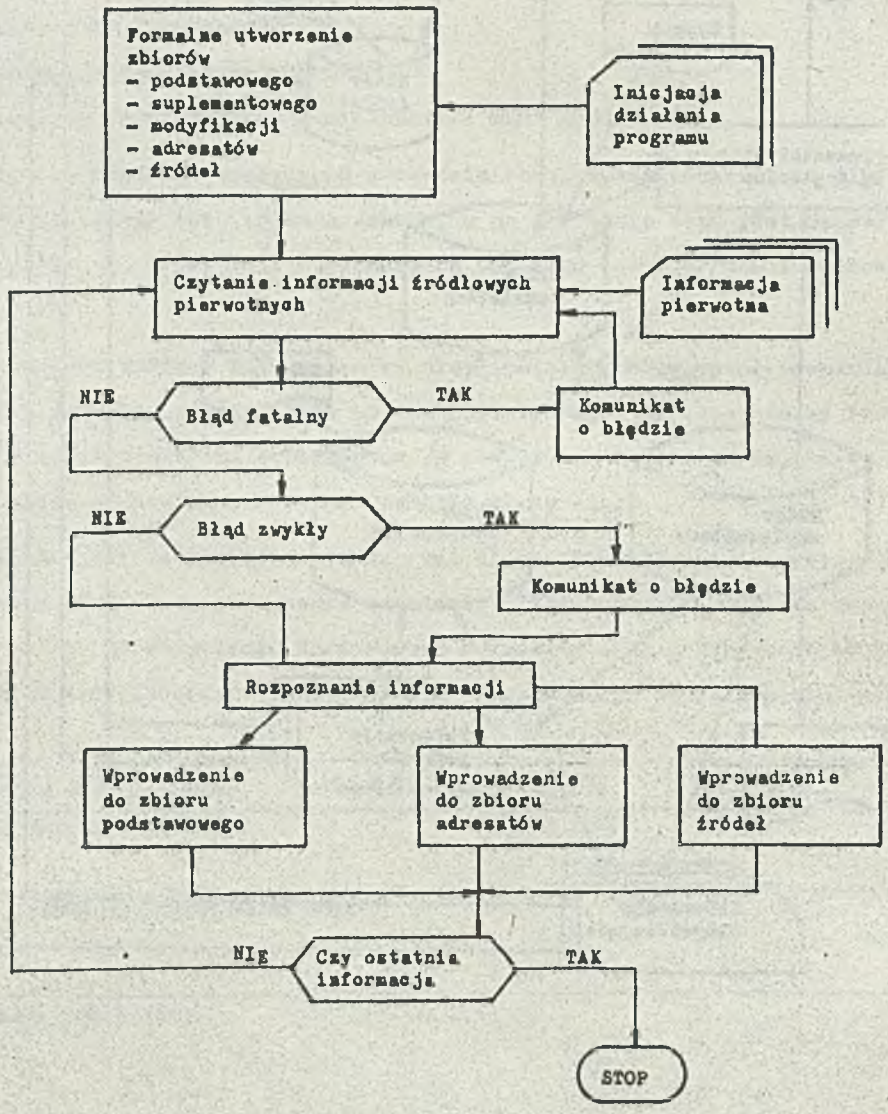
Rys.6. Karta adresata PIRIO/MERA 400



Rys. 7. Ogólna struktura komputerowego wspomagania PIRIO/MERA 400

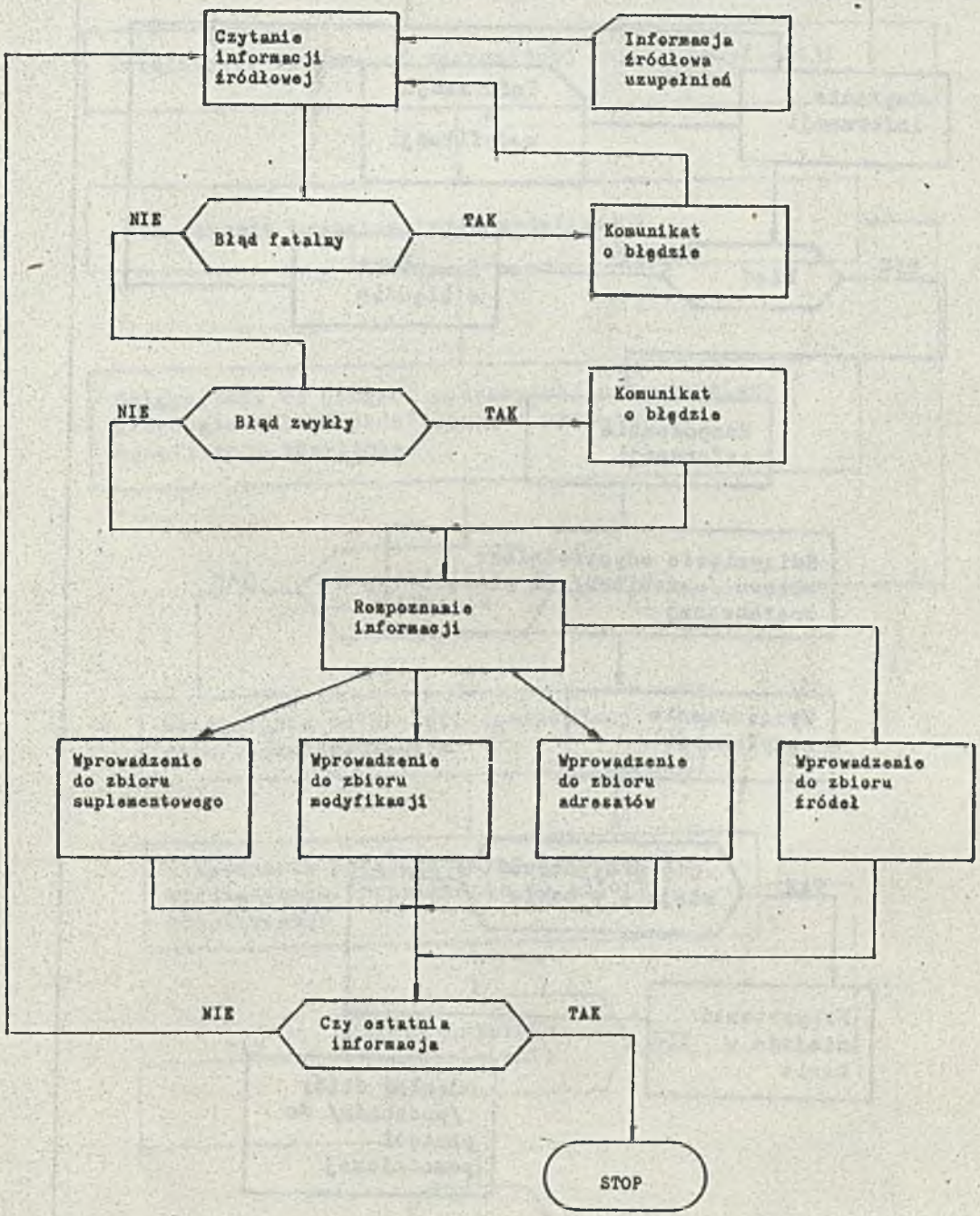
Zgodnie z tą strukturą należy przewidzieć następujące programy:

a) program założenia bazy - w zasadzie jest to czynność jednorazowa; zadania tego programu ilustruje rys. 8.



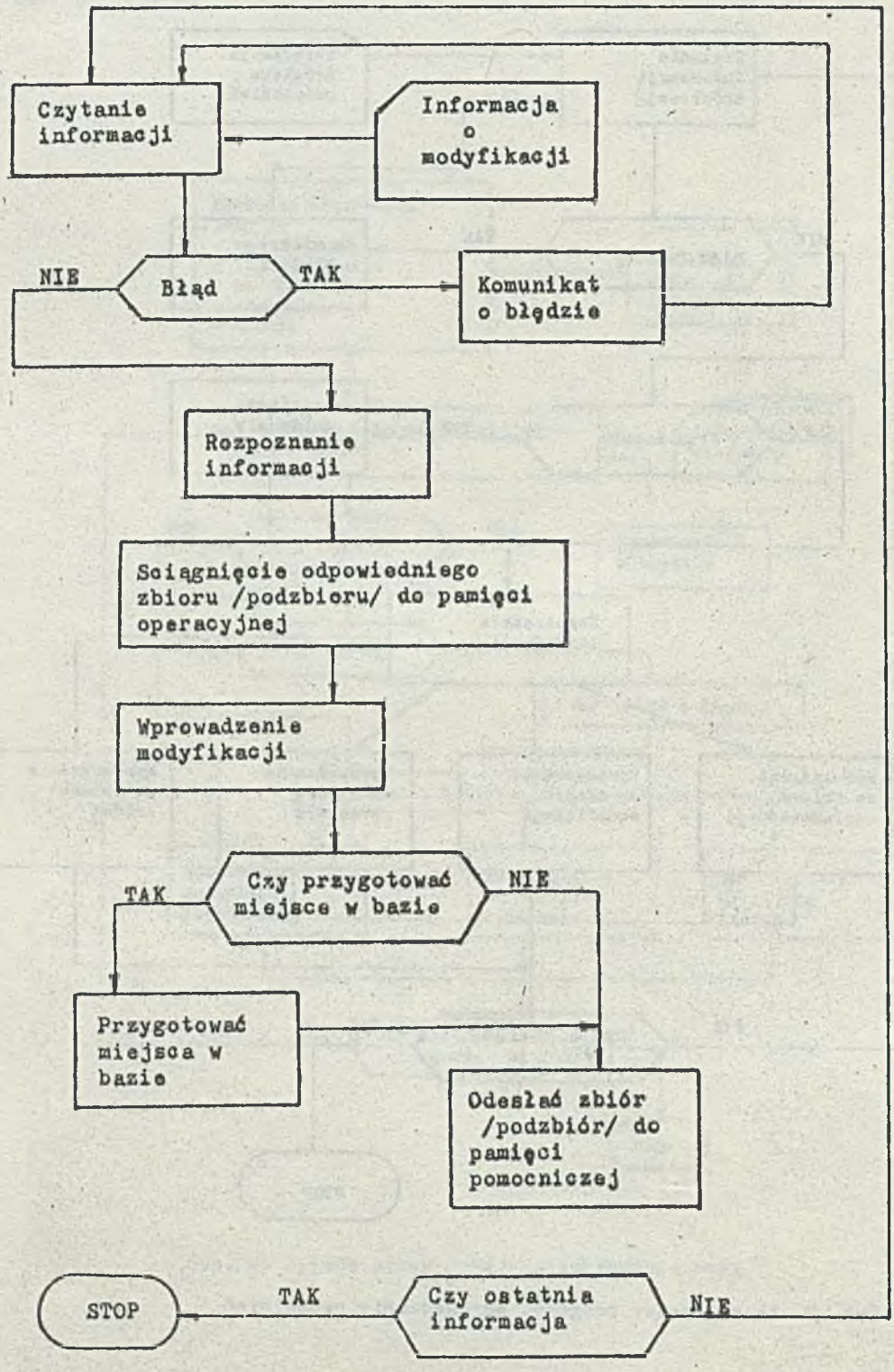
Rys. 8. Zasada pracy programu zakładania bazy

b) program wprowadzenia informacji uzupełniających - pokazano na rys. 9,



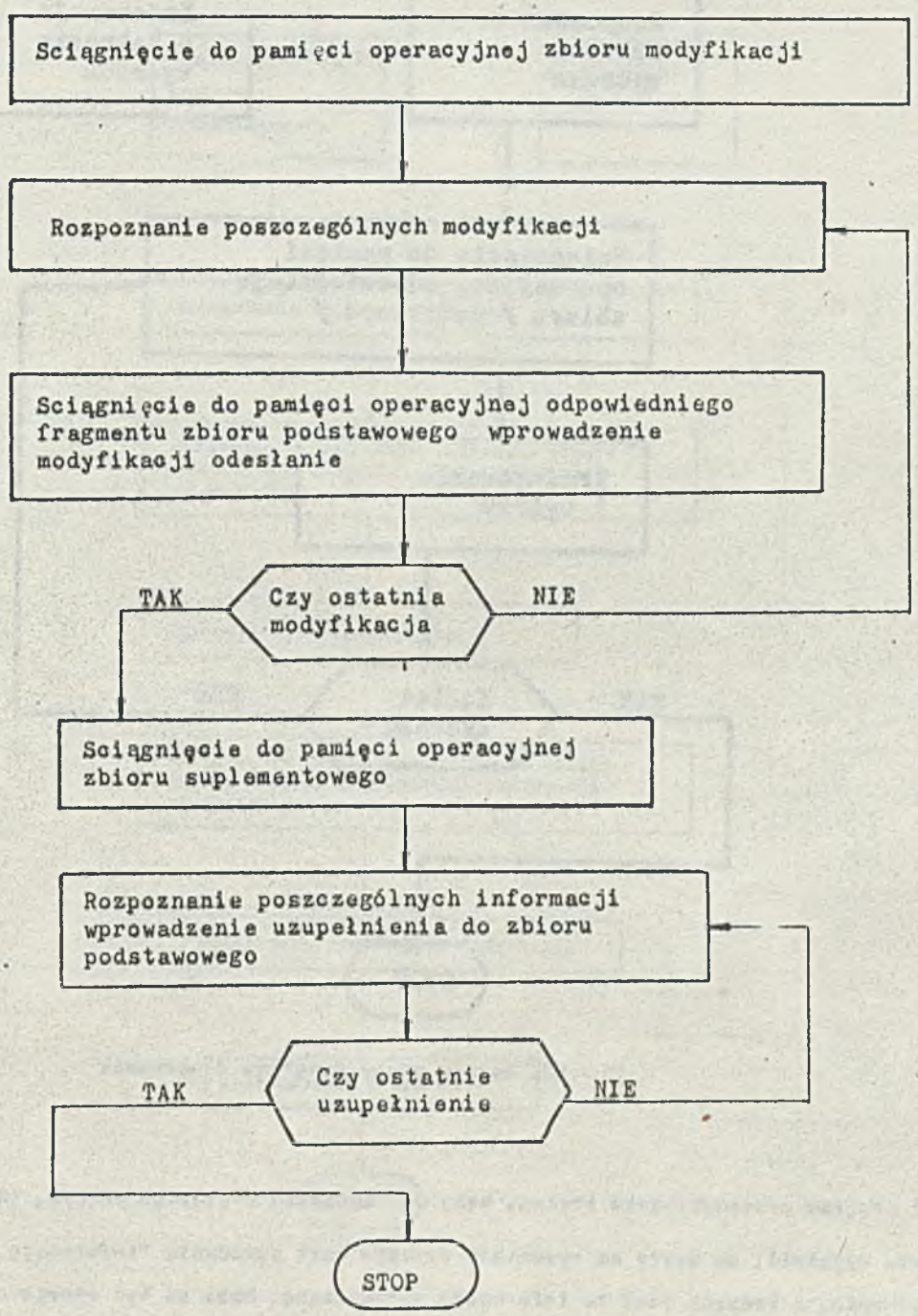
Rys. 9. Zasada pracy programu wprowadzania uzupełnień

c) program wprowadzenia poprawek na rys. 10,



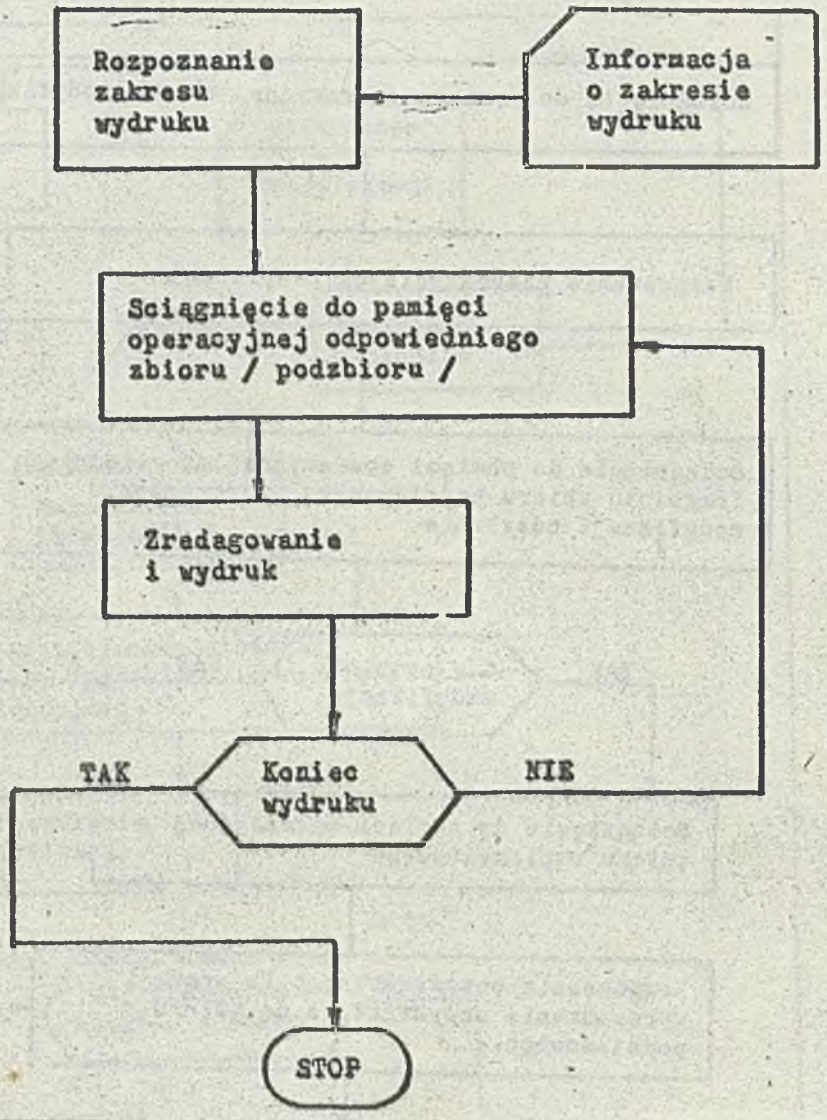
Rys. 10. Zasada pracy programu wprowadzania modyfikacji (poprawek)

d) program łączenia zbiorów na rys. 11,



Rys.11. Zasada pracy programu łączenia zbiorów

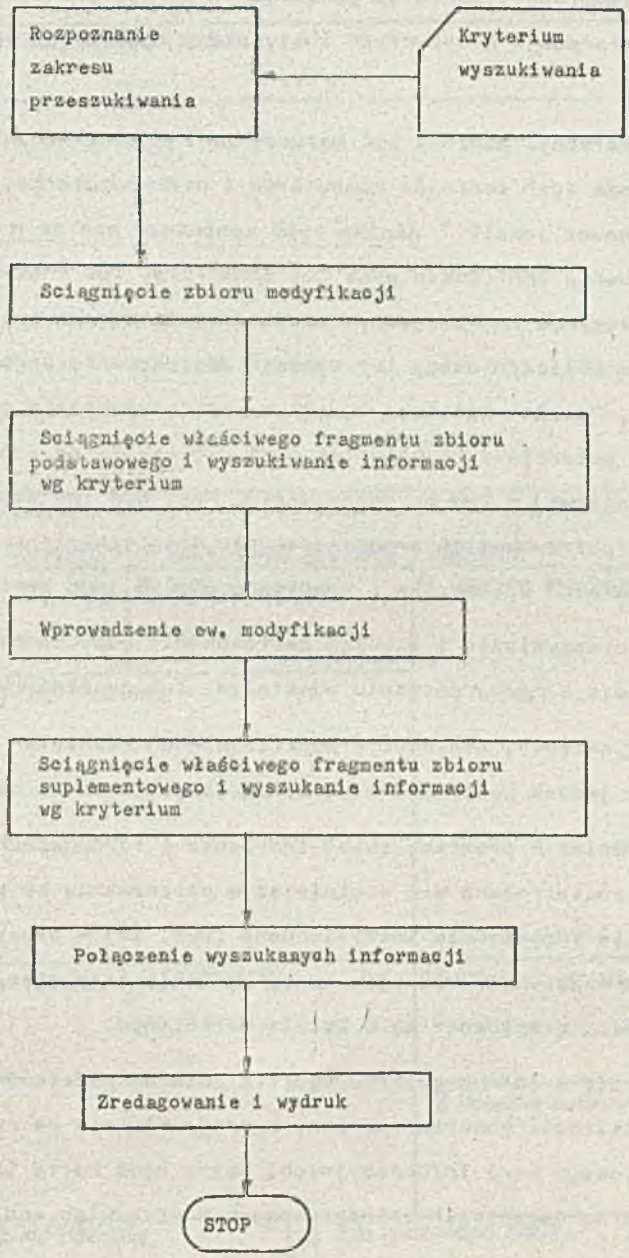
a) program wydruku treści bazy lub poszczególnych jej zbiorów na rys. 12.



Rys.12. Zasady pracy programu drukowania

f) program przeszukiwania treści, bazy wg żądanego kryterium na rys. 13.

Należy wyjaśnić, że użyte na rysunkach symbole kart oznaczają "informację wejściową" a nie urządzenia, z których jest ta informacja wprowadzona. Może to być równie dobrze czytnik taśmy perforowanej przy dużej liczbie danych, jak i monitor ekranowy; tak zostało to przyjęte w programie eksperymentalnym. Opis i analizę prac eksperymentalnych zawiera osobne opracowanie.



Rys. 13. Zasada pracy programu przeszukiwania bazy wg założonego kryterium

Część II. PIRIO/IMM

W obecnej sytuacji (sierpień 1981) przewidując, że istnienie Instytutu w dużej mierze będzie zależeć od samodzielnego pozyskiwania klientów i zleceniodawców na powstające w Instytucie wytwory, staje się uzasadnione szybkie zorganizowanie bardzo prostego w realizacji, a dzięki temu sprawnego, obiegu informacji o wytworach Instytutu w ogóle, a o oprogramowaniu w szczególności.

Adresatami tej informacji powinni być wszyscy potencjalni nabywcy tegoż oprogramowania. A więc wszyscy posiadacze tych rodzajów komputerów i minikomputerów, na które istnieje w Instytucie oprogramowanie, nawet jeżeli danego typu komputera nie ma w IMM (np. ODRA). Adresatami informacji o oprogramowaniu IMM-owskim mogą być inne firmy lub instytucje zainteresowane np. danym problemem merytorycznym rozwiązywanym przez dane oprogramowanie, itp. Mogą się wtedy stać klientami Ośrodka Obliczeniowego lub zamówić dostosowanie oprogramowania do swoich możliwości sprzętowych, itp. Działalność taka,będąca w istocie poważnie potraktowaną reklamą, wraz z rozpoznaniem potrzeb potencjalnych klientów musi być oparta na odpowiednim zasobie wiedzy o istniejącym (lub powstającym) w IMM oprogramowaniu. Musi być też zapewniona możliwość natychmiastowego dotarcia do potrzebnej informacji, co nie jest spełnione w sytuacji, gdy trzeba dopiero odszukać kompetentnego pracownika i odwoływać się do jego pamięci.

Celowe jest więc przemyślenie i szybkie zastosowanie odpowiednich zasad prowadzenia i rozpowszechniania informacji o oprogramowaniu powstałym lub powstającym w Instytucie - PIRIO/IMM.

Jak wspomniano we wstępie, propozycje poszczególnych rozwiązań oparte na doświadczeniach PIRIO/MERA 400,zostały jednak dostosowane do wymienionych celów i warunków Instytutu.

Nie wdając się również w problemy zasad tworzenia i rozpowszechniania dokumentacji użytkowej czy roboczej, skoncentrowano się w niniejszym opracowaniu na problemach poziomu KART INFORMACYJNYCH. Karty te odpowiednio zmodyfikowane (rys. 14) w stosunku do pierwowzoru (rys. 1), wypełnione dla wszystkich rodzajów oprogramowania istniejących i powstających lub powstałych w przeszłości, gromadzone są w Dziale Marketingu.

Karty te grupuje się w jednym zbiorze będącym zbiorem podstawowym. Natomiast, ze względu na niezbyt dużą częstotliwość powstawania nowych programów nie ma powodów do utrzymywania drugiego zbioru suplementowego kart informacyjnych. Każda nowa karta lub karta modyfikująca starą jest od ręki rozsyłana do wszystkich zainteresowanych oraz włączana do zbioru.

W celu umożliwienia szybkiego określania kręgu zainteresowanych danym oprogramowaniem potencjalnych jego odbiorców, prowadzony jest w Dziale Marketingu zbiór adresatów (rys. 15). Treść KARTY ADRESATA odbiega w tym wypadku znacznie od pierwowzoru (zob. rys. 6). Wynika to z faktu, że w PIRIO/MERA 400 adresat był na ogół również autorem jakiegoś oprogramowania objętego PIRIO, ponadto nie klasyfikowano adresatów pod kątem zainteresowań. Natomiast w KARCIE ADRESATA - PIRIO/IMM owo potencjalne zainteresowanie adresata musi być uwzględnione. Jak z tego co powiedziano wynika - PIRIO/IMM ma wspomagać Dział Marketingu zarówno w prowadzeniu reklamy

PIRIO / IMM	KARTA INFORMACYJNA
1 Symbol katalogowy	2 Poziom oprogramowania
3 Nazwa programu, pakietu, biblioteki, systemu	
4 Przeznaczenie	
5 Klasyfikacja problemowa	6 Data opracowania
PROGRAMISTYCZNE UWARUNKOWANIA DZIAŁANIA PROGRAMU	
7 System operacyjny	8 Język, translator
9 Programy współpracujące	
10 Szczegółowa konfiguracja sprzętowa	
13 Wykaz użytkowników (symbole KART ADRESATA)	11 Dokumentacja użytkowa
	12 Nazwiska autorów
14 Nazwisko opracowującego informację	15 Data opracowania KARTY
INSTYTUT MASZYN MATEMATYCZNYCH 02-078 Warszawa, ul. Krzymickiego 34	telefony: 21 84 41 do 49; 29 92 71 TLx: 81 35 17

Rys. 14. Karta informacyjna PIRIO/IMM

czynnej, a więc w rozpowszechnianiu informacji z własnej inicjatywy, jak i w reklamie biernej, a więc w odpowiadaniu na pytania. Wydaje się, że zaproponowane tu zasady PIRIO/IMM okażą się skuteczne dla obu tych celów.

PIRIO / IMM	KARTA ADRESATA	Symbol
Nazwa firmy	*Profil problemowy firmy*	
Nazwa komórki zainteresowanej oprogramowaniem	*Profil problemowy*	
Adres dla korespondencji		
Nazwiska kompetentnych osób do rozmów; telefony		
Programy przekazane tej firmie: nazwy symboliczne i symbole KART INFORMACYJNYCH		

Rys. 15. Karta adresata PIRIO/IMM

Biuletyn Informacyjny NAUKI I TECHNIKI KOMPUTEROWE

mgr inż. Bożena PADZIK
mgr inż. Tadeusz WILCZEK
Instytut Maszyn Matematycznych

Ogólny opis systemu operacyjnego UNIX

Wstęp

W czasie wstępnych prac nad perspektywami rozwoju minikomputerów w Polsce, wynikła konieczność zaznajomienia się z istniejącymi systemami operacyjnymi mającymi cechy efektywnej przenoszalności i uniwersalności. Jednym z nich jest system operacyjny UNIX, w Polsce używany wyłącznie jako system operacyjny minikomputera PDP-11.

Rosnące zainteresowanie systemem UNIX skłoniło nas do opracowania tego tematu, którego celem jest przekazanie ogólnych wiadomości o tym systemie i przedstawienie ciekawszych, zastosowanych w nim rozwiązań.

UNIX jest wielodostępnym uniwersalnym systemem operacyjnym. W pierwotnej wersji powstał w firmie BELL LABORATORIES jako system operacyjny minikomputera PDP-7. Jego twórcą jest Ken Thompson. W latach siedemdziesiątych powstały 4 wersje tego systemu. System zaimplementowano dla maszyn PDP-9, -11/20, 11/34, -11/40, 11-45, -11/60, -11/70 oraz Interdata 8/32. Popularność i zastosowanie systemu stale rośnie. Kolejne wersje zaimplementowano na minikomputerach firm Zilog, Perkin-Elmer, SWTP, Onyx i Cromemco. Ostatnio powstała wersja dla 32-bitowego minikomputera Perkin-Elmer 3220.

System UNIX zyskał sobie miano standardowego systemu operacyjnego lat osiemdziesiątych. Wiadomości zawarte w tym opracowaniu dotyczą czwartej wersji systemu, zaimplementowanej na maszynach PDP-11/70 i Interdata 8/32. Głównymi elementami systemu są:

- system plików
- obsługa wejścia/wyjścia
- zarządzanie procesami
- zarządzanie pamięcią.

Opracowanie oparte jest na materiałach zamieszczonych w "The Bell System Technical Journal" pt. "UNIX TIME SHARING SYSTEM" 1 oraz na materiałach z seminarium, które było zorganizowane w Zespole Struktur Systemowych IMM [2].

System plików

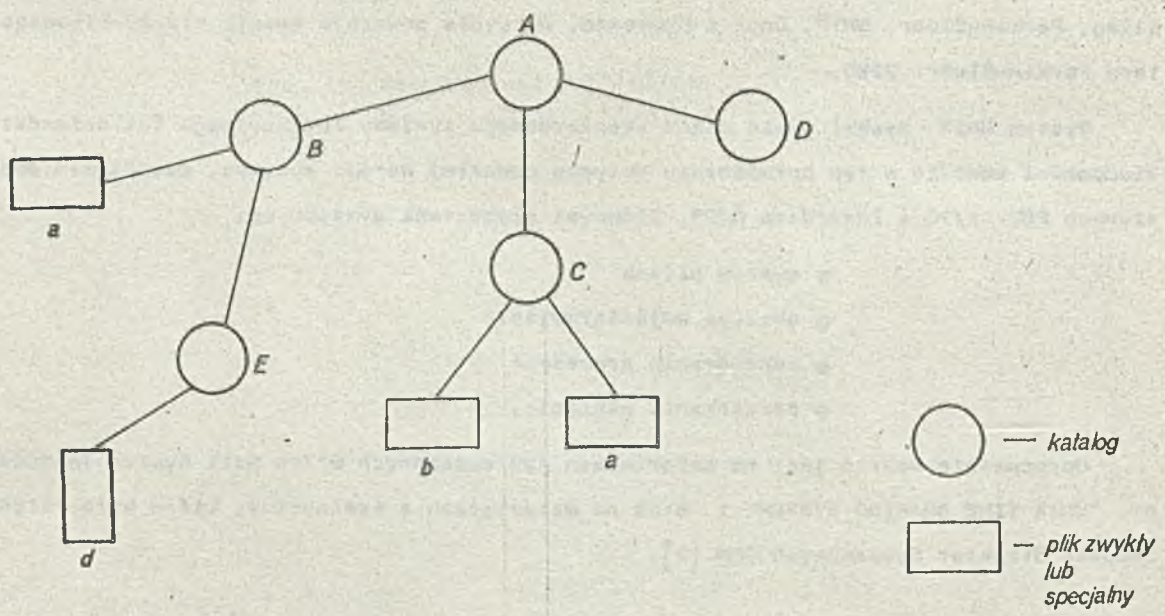
Jedną z ważniejszych cech systemu operacyjnego UNIX jest organizacja systemu plików. Plik jest to zbiór informacji tworzących pewną logiczną całość i zapisanych w logicznie spójnym miejscu. Pliki mogą być tworzone przez użytkownika lub przez system. Z punktu widzenia użytkownika istnieją trzy rodzaje plików:

- pliki zwykłe,
- pliki specjalne,
- katalogi.

Plik zwykły zawiera dowolne informacje zapisane przez użytkownika binarnie lub znakowo. Struktura pliku zwykłego jest nadzorowana przez program, który go wykorzystuje, a nie przez system.

Plik specjalny związany jest z urządzeniem wejścia lub wyjścia. Plik taki jest czytany i zapisywany jako normalny plik dyskowy, ale np. rezultatem wykonania funkcji "otwórz plik" jest bezpośrednia aktywacja odpowiedniego urządzenia. Wszystkie pliki specjalne skatalogowane są w katalogu o nazwie /dev. Traktowanie urządzeń wejścia/wyjścia i plików w sposób jednolity ma wiele zalet. Ich nazwy mają tę samą składnię i znaczenie, dzięki czemu mogą być używane wymiennie bez żadnych dodatkowych zmian w programie. Możliwe jest również stosowanie jednolitego mechanizmu ochrony.

Katalog zawiera informacje o grupie plików, tzn. ich nazwy i odsyłacze do ich deskryptorów. Każdy użytkownik posiada katalog swoich plików. Katalogi tworzą strukturę drzewiastą o nieograniczonej liczbie poziomów i liczbie plików w katalogu. Pliki zwykłe i specjalne mogą być wyłącznie liśćmi tego drzewa, węzłami są katalogi.



Rys. 1. Drzewiasta struktura katalogów

Plik jest identyfikowany przez nazwę łańcuchową. Jest to sekwencja nazw katalogów poprzedzających plik w strukturze drzewa zakończona nazwą własną pliku. Nazwy te oddzielane są znakiem "/". Jeżeli nazwa łańcuchowa rozpoczyna się znakiem "/", to sekwencja rozpoczyna się od korzenia drzewa katalogów, np. nazwa łańcuchowa /B/E/d identyfikuje w naszym wypadku plik o nazwie własnej d. Nazwa "/" identyfikuje katalog będący korzeniem drzewa. Jeśli brak jest na początku znaku "/", to sekwencja rozpoczyna się od katalogu danego użytkownika. Nazwa pusta identyfikuje ten katalog. Mogą istnieć różne pliki zwykłe o takich samych nazwach własnych np. plik /B/a i /C/a. Ten sam plik zwykły może być również skatalogowany w kilku katalogach pod różnymi nazwami, gdyż katalog odwzorowuje nazwę pliku na adres jego deskryptora (linking). Każdy katalog ma co najmniej dwa elementy:

- nazwę "." identyfikującą ten właśnie katalog,
- nazwę ".." identyfikującą "ojca" tego katalogu czyli katalog, w którym tenże jest utworzony.

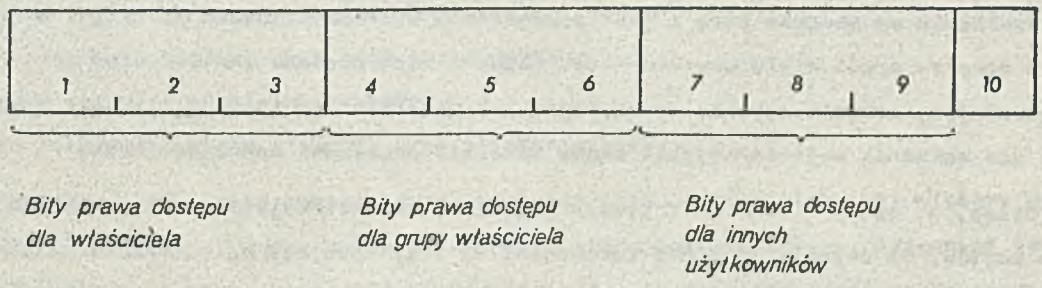
Pliki zwykłe i specjalne mogą być chronione przed zapisem, odczytem i wykonaniem a katalogi przed tworzeniem i usuwaniem plików, listowaniem nazw plików oraz przeszukiwaniem katalogu.

Użytkownicy pliku dzielą się na 3 kategorie:

- właściciel pliku,
- grupa właściciela czyli użytkownicy uprzywilejowani,
- inni użytkownicy.

Każdy użytkownik systemu posiada swój identyfikator. W deskryptorze pliku zapisywany jest identyfikator jego właściciela.

Z każdym plikiem związana jest informacja dziesięciobitowa znajdująca się w deskryptorze i określająca prawa dostępu dla każdej kategorii użytkowników oddzielnie (rys. 2).



Rys. 2. Bity ochrony pliku

Bit 10 ma znaczenie dla pliku, który zawiera program. Jeżeli bit 10 ma wartość "1", to na czas działania tego programu identyfikator użytkownika jest zamieniany na identyfikator właściciela pliku. Dzięki temu użytkownik uzyskuje prawo dostępu do plików do tej pory dla niego niedostępnych. Może z nich korzystać tylko w sposób ściśle określony przez program. Mechanizm ten chroni

pliki przed korzystaniem z nich w niewłaściwy sposób.

Korzystanie z plików i manipulowanie nimi możliwe jest przez funkcje i operacje. Poniżej przedstawiono niektóre z nich podając postać formalną w języku C, znaczenie oraz ewentualnie opis działania.

.filep = open (name, flag) - otwarcie pliku

Funkcja ta wyszukuje plik o nazwie łańcuchowej wymienionej na pozycji name, sprawdza czy tryb pracy podany na pozycji flag nie narusza praw dostępu i w zmiennej filep zwraca wartość odsyłacza do deskryptora tego pliku. Odsyłacz ten pozwala jednoznacznie zidentyfikować dany plik. Jeśli wystąpił błąd, to zwracana jest wartość -1.

.filep = creat (name, flag) - utworzenie pliku

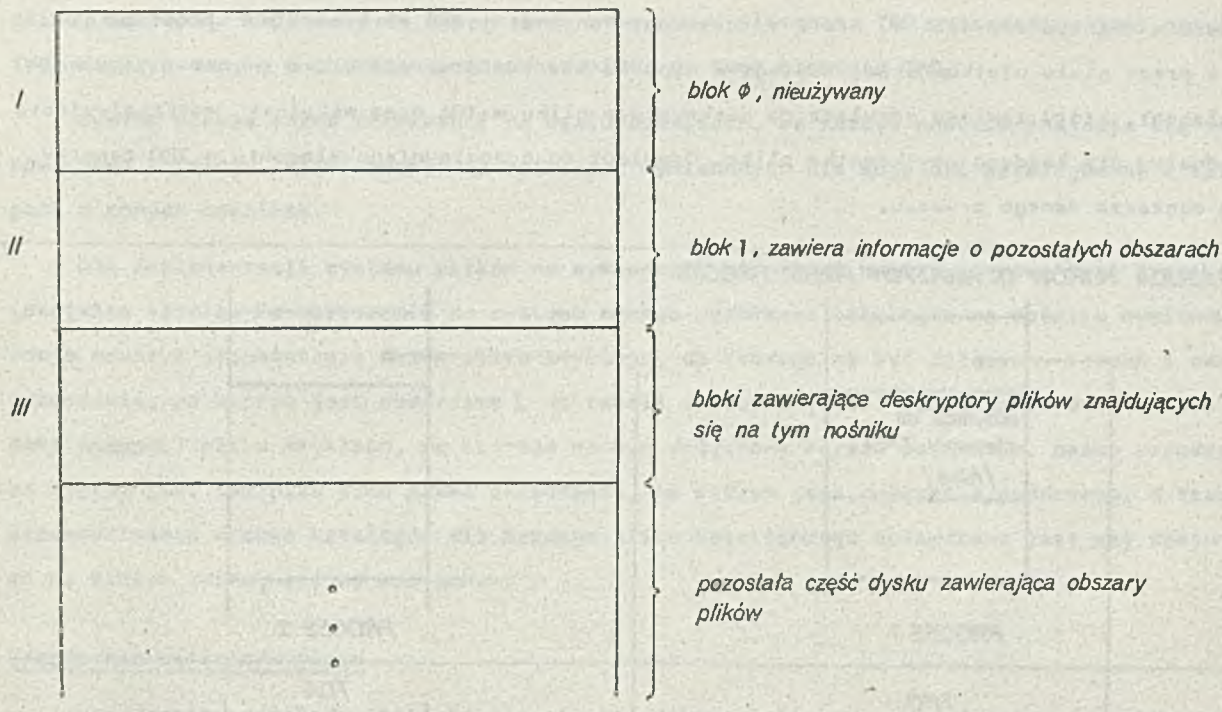
Ta funkcja tworzy nowy plik o nazwie łańcuchowej name, jeśli plik taki nie istniał, lub też zmniejsza długość pliku do zera, jeśli taki plik już istnieje. Zwraca odsyłacz filep do deskryptora tego pliku lub wartość -1 jeśli niemożliwe jest utworzenie żądanego pliku. Argument flag określa prawa dostępu dla 3 grup użytkowników. Na jego podstawie ustawiane są bity ochrony pliku.

- close (filep) - zamknięcie pliku wskazywanego przez odsyłacz filep
- link (pname, kname) - dołączenie pliku zwykłego o nazwie pname do katalogu o nazwie kname
- unlink (pname, kname) - odłączenie pliku pname od katalogu kname
- write (filep, buffer, n) - zapisanie do pliku wskazywanego przez odsyłacz filep n bajtów z tablicy buffer
- read (filep, buffer, n) - odczytanie z pliku wskazywanego przez filep n bajtów do tablicy buffer.

Plik po utworzeniu ma długość zero i jest powiększany w efekcie zapisu do niego. Może być jednocześnie otwarty przez wielu użytkowników. Każdy z nich posiada indywidualny wskaźnik wejścia/wyjścia wskazujący pozycję w pliku do czytania lub pisania. Dostęp do pliku jest w zasadzie sekwencyjny, ale wskaźnik wejścia/wyjścia można dowolnie przesuwać operacją lseek.

.lseek (filep, n, sp) - przesuń wskaźnik wejścia/wyjścia w pliku wskazanym przez filep o n bajtów. Sp określa sposób przesuwania. Są trzy sposoby: od początku, od bieżącej pozycji i od końca pliku. Istnieje jeszcze wiele innych funkcji i operacji dotyczących współpracy ze zbiorami, jak np. zmień prawo dostępu, utwórz katalog, usuń zbiór, zmień właściciela, itp., ale nie będziemy ich tu bliżej omawiać.

Pliki przechowywane są na dyskach. Dysk podzielony jest na 512 bajtowe bloki bezpośrednio adresowalne. System operacyjny dzieli obszar dyskowy na 4 części (rys. 3).



Rys. 3. Logiczny podział dysku

Katalogi są zbiorami 16 bajtowych pól, przy czym 14 bajtów stanowi nazwę pliku, a 2 indeks tablicy deskryptorów, zawierającej deskryptory wszystkich plików zgromadzonych na danym nośniku. Deskryptor pliku (64 bajtowa kostka) zawiera następujące informacje:

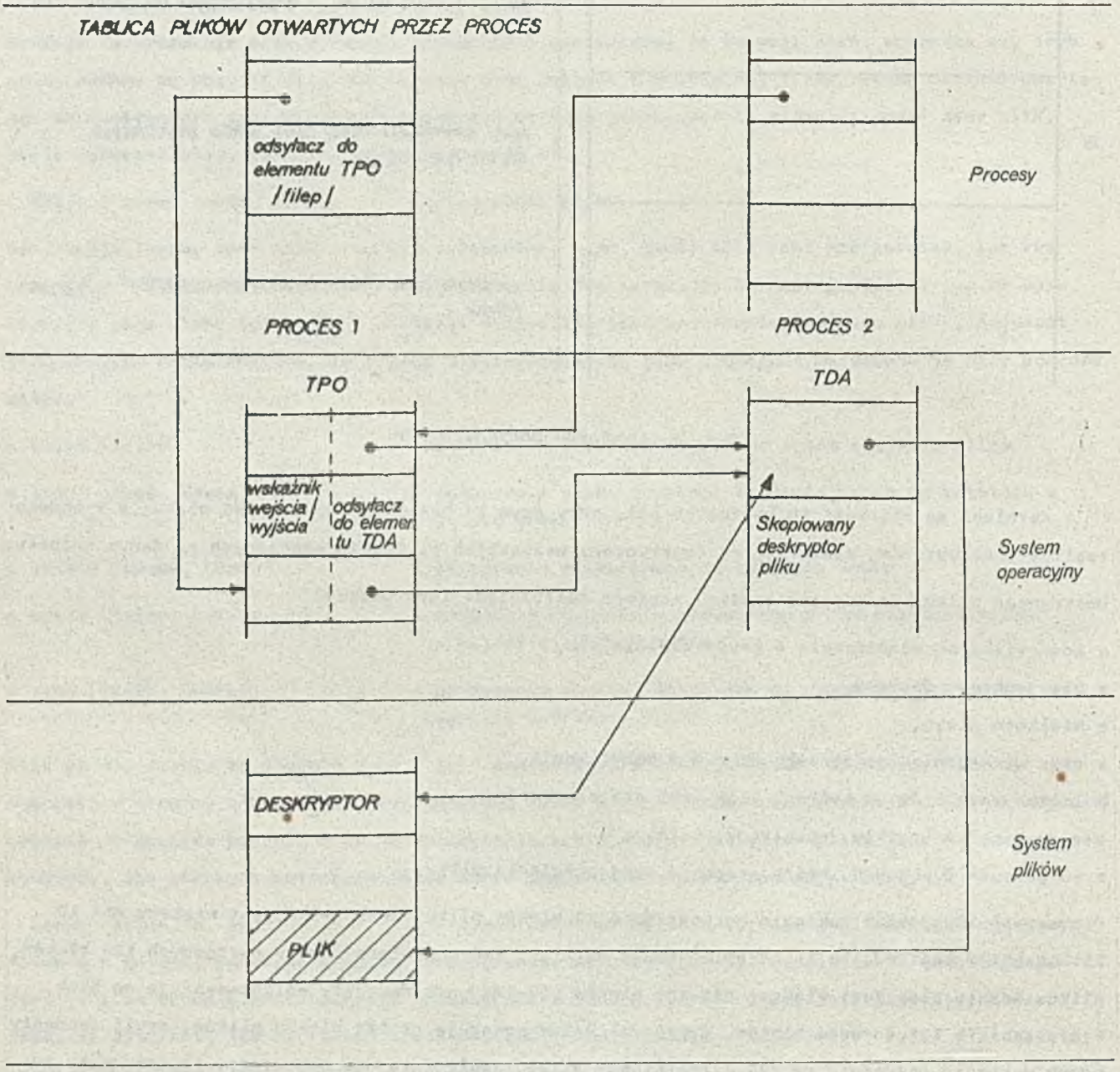
- identyfikator właściciela i grupy właściciela,
- bity ochrony dostępu,
- wielkość pliku,
- czas utworzenia, ostatniego użycia i modyfikacji,
- liczba katalogów, w których plik jest skatalogowany,
- rodzaj pliku (zwykły, specjalny, katalog),
- 13 adresów dyskowych, które stanowią mapę położenia pliku.

Pierwszych 10 adresów wskazuje bezpośrednio 10 bloków pliku. Jeśli plik jest większy niż 10 bloków (5120 bajtów), to 11 adres wskazuje na blok, który zawiera adresy następnych 128 bloków pliku. Jeżeli plik jest większy niż 138 bloków (10+128), to dwunasty adres wskazuje na blok, który zawiera 128 adresów bloków, z których każdy wskazuje na 128 bloków plików, czyli dwunasty adres wskazuje pośrednio na 128 x 128 bloków pliku. Jeśli i to nie wystarczy, to trzynasty adres początkuje pośredni potrójny adres następnych 128 x 128 x 128 bloków pliku. Maksymalna więc wielkość pliku wynosi:

$$10 + 128 + 128^2 + 128^3 \text{ bloków.}$$

W systemie znajdują się dwie tablice: tablica deskryptorów aktywnych (TDA) i tablica plików otwartych (TPO).

TDA zawiera deskryptory plików, które są używane. Ponieważ jeden plik może być jednocześnie otwarty przez wielu użytkowników, dlatego w wypadku każdego otwarcia pliku tworzony jest w TPO nowy element, który zawiera odsyłacz do deskryptora pliku w TDA oraz wskaźnik wejścia/wyjścia indywidualny dla każdego użytkownika pliku. Odsyłacz do odpowiedniego elementu w TPO zawarty jest w obszarze danego procesu.



Rys. 4. Sposób dostępu dwóch procesów do tego samego pliku

Dostęp do pliku polega właściwie na odnalezieniu jego deskryptora, np. implementacja funkcji "open" wygląda następująco: na podstawie nazwy łańcuchowej wyszukiwany jest deskryptor

pliku, następnie kopiowany do TOA, w nowo utworzonym elemencie TPO umieszczany jest odsyłacz do tego deskryptora, a w obszarze procesu odsyłacz do tego elementu TPO.

System plików można rozmieścić na wielu nośnikach. Na każdym nośniku znajduje się pełne poddrzewo katalogu. Pliki zwykłe na nośnikach wymiennnych nie mogą być katalogowane w katalogach z innych nośników.

Dla implementacji systemu plików na wymiennych nośnikach wykorzystywana jest przez system specjalna tabela. Po dołączeniu do systemu nowego poddrzewa katalogów na nośniku wymiennym (operacja mount z argumentami: nazwa pliku zwykłego, do którego ma być dołączony korzeń i nazwa urządzenia, na którym jest poddrzewo), do tabeli zapisywany jest odsyłacz do deskryptora (indeks pozycji) pliku zwykłego, do którego ma być dołączony korzeń poddrzewa, nazwa urządzenia, na którym jest ten plik oraz nazwa urządzenia, na którym jest dołączone poddrzewo. W czasie przeszukiwania drzewa katalogów dla każdego pliku katalogowego sprawdzane jest czy znajduje się on na stałym dysku, czy na wymiennym.

Urządzenia wejścia/wyjścia

Urządzenia wejścia/wyjścia dzielę się na znakowe i blokowe. Urządzenia te są identyfikowane przez nazwę typu urządzenia i numer w ramach typu. Dla urządzeń znakowych i blokowych istnieją w systemie tablice podporządkowujące nazwie urządzenia adres podprogramu jego obsługi.

Dostęp do urządzeń blokowych jest buforowany. W systemie jest lista 10 - 70 buforów. Nagłówek bufora zawiera nazwę urządzenia. Bufory są opróżniane według strategii LRU (Least Recently Used). Program obsługujący dysk przechowuje kolejkę zleceń transmisji i sortuje je, aby zmniejszyć czasy transmisji.

Programy obsługujące urządzenia znakowe korzystają z kolejek znaków buforowanych we wspólnym obszarze. W wypadku urządzeń wyjściowych znaki są przesyłane z programu użytkownika do kolejki znaków, z której są pojedynczo pobierane przez program obsługi urządzenia.

Dla urządzeń wejściowych sprawa wygląda podobnie. W wypadku terminali są trzy kolejki: jedna wyjściowa, taka jak w innych urządzeniach znakowych wyjściowych i dwie wejściowe. Do jednej są ładowane znaki wprowadzane z klawiatury, po wprowadzeniu znaku końca linii cała kolejka jest kopiowana do drugiej, z której znaki są pobierane przez program obsługi terminala.

Pewne urządzenia nieblokowe (np. linie komunikacyjne lub szybkie drukarki) są za szybkie, aby kolejkować znaki, korzystają one z własnych lub pożyczanych od systemu buforów.

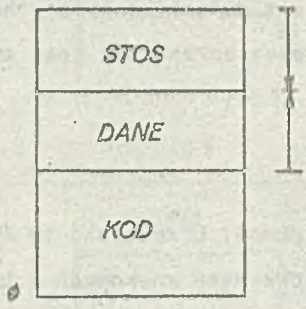
Procesy

Podstawą działania systemu operacyjnego UNIX są procesy. Proces jest to wykonywanie kodu w określonym środowisku (rejestry, dane, stos, otwarte pliki, itp.) i w konkretnym celu. Procesy są systemowe i użytkowe.

Programy użytkowe wykonywane są w ramach procesów użytkowych. Proces użytkowy może pracować w trybie użytkowym (wykonywanie kodu programu) i systemowym (realizacja ekstrakodów). W trybie

systemowym proces korzysta z obszarów danych niedostępnych dla procesów użytkowych np. deskryptory otwartych plików . Wirtualna pamięć procesu składa się z trzech obszarów:

- obszar kodu - rozpoczyna się od wirtualnego adresu ø, może być dzielony przez wiele procesów
- obszar danych - może rosnąć w górę od ustalonej bazy
- obszar stosu - może rosnąć w dół od limitu obszaru



Rys. 5. Wirtualna pamięć procesu

W systemie znajduje się tablica opisująca procesy. Każdy element tej tablicy przyporządkowany jest jednemu procesowi i zawiera m.in.takie informacje:

- nazwa procesu,
- lokalizacja segmentów kodu i danych w pamięci,
- priorytet, itp.

Ponieważ kod procesu może być dzielony przez kilka procesów,w systemie znajduje się tablica kodów. Każdy element tej tablicy zawiera informacje:

- nazwę,
- lokalizacja w pamięci operacyjnej i pamięci pomocniczej,
- liczba procesów wykorzystujących ten kod.

Procesy mogą być tworzone dynamicznie operacją fork:

id = fork( )

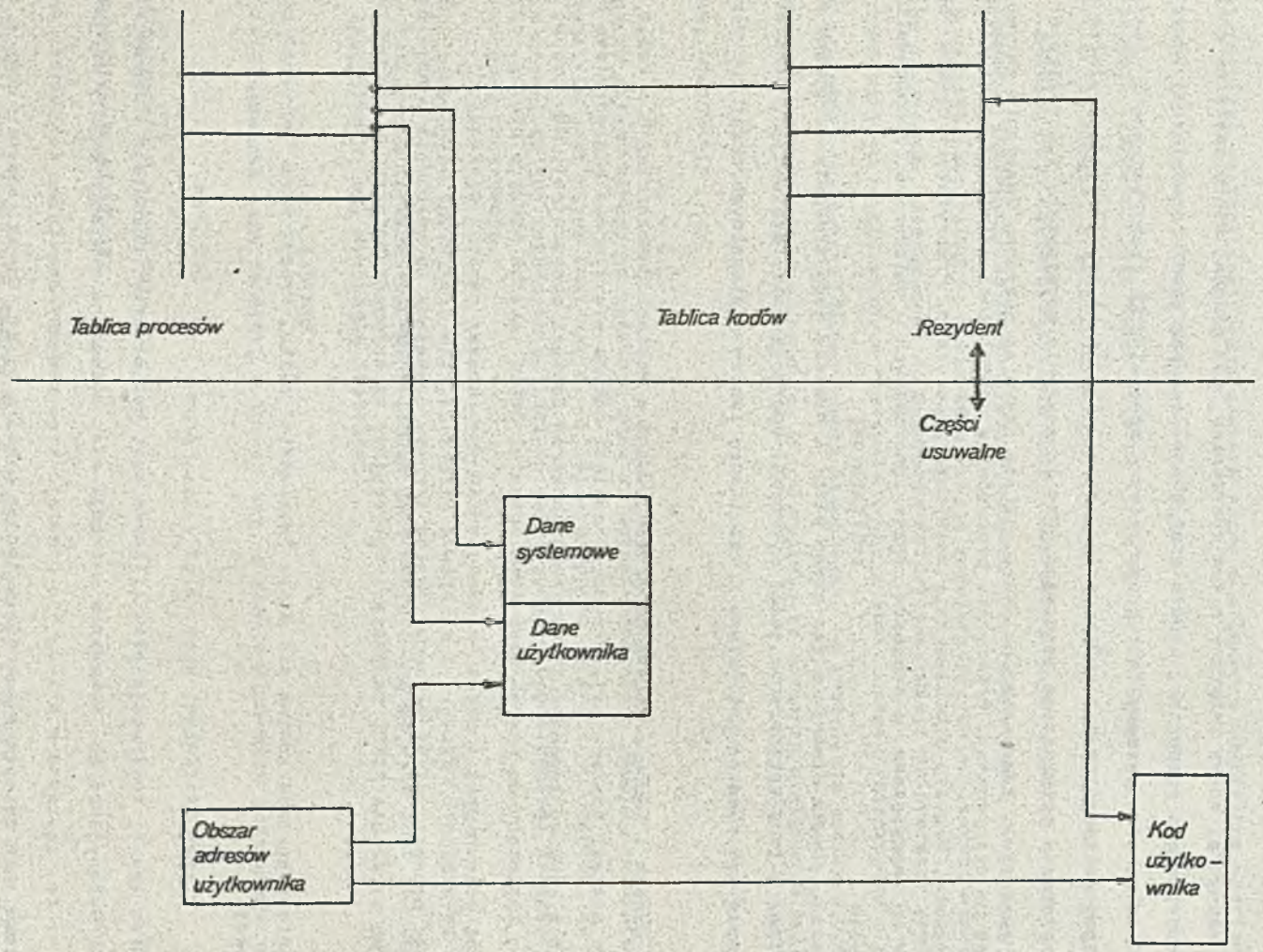
(operacja jest bezparametrowa lecz nawiasy są niezbędne). Nowo powstały proces jest kopią procesu tworzącego. Kopiowany jest kod, obszary danych, tablica plików otwartych, stos, rejestry i inne. Procesy te stają się zupełnie niezależne. Proces tworzący uzyskuje status "ojca", a utworzony - "potomka". Wartość zwracana id dla ojca równa się identyfikatorowi potomka, dla potomka równa ø.

Komunikacja między procesami mającymi tego samego przodka jest możliwa dzięki wykorzystywaniu mechanizmu zwanego pipe.

Operacja

filep = pipe( )

tworzy plik bez nazwy. Plik ten jest dzielony między wszystkie procesy potomne. Dostęp do pliku jest uzyskiwany operacjami read i write. Informacje z pliku mogą być odczytywane ściśle w kolej-



Rys. 6. Struktury danych sterowania procesem

ności zapisu.

Synchronizacja procesów odbywa się za pomocą zdarzeń. Proces zawiesza się w oczekiwaniu na zajście zdarzenia

id = wait (event)

Event jest to zazwyczaj adres związany z tym zdarzeniem, np. proces, który czeka na zakończenie się procesu będącego jego potomkiem, będzie miał event równy adresowi jego własnej pozycji w tabeli procesów. Adres ten dostępny jest przez zmienną specjalną $$ (zob. zmienne o specjalnym znaczeniu). Kiedy jakiś potomek zakończy pracę, zasygnalizuje to przesłaniem odsyłacza do elementu tablicy procesów opisującego jego ojca. Pod id zostanie podstawiony identyfikator procesu, który się zakończył. Zdarzenia, na które nikt nie czeka są gubione. Zdarzenie aktywuje wszystkie czekające na nie procesy. Priorytety procesów są dynamiczne, zależą od priorytetów zdarzeń, które je aktywowały. Procesy użytkowe mają najniższe priorytety. Priorytety procesów "okupujących" procesor są zmniejszane, a procesów często ignorowanych - zwiększane. Zakończenie procesu powoduje operacja exit(status)

Kończy ona proces, zamyka otwarte przez niego pliki i niszczy jego środowisko. Ojciec procesu jest zawiadamiany o jego zakończeniu jeśli wykonywał operację wait. Status jest przekazywany ojcu. Proces może być również zakończony w rezultacie innych nielegalnych akcji lub zleceń użytkownika.

Segmenty danych i kodów procesów są przechowywane w pamięci pomocniczej i w razie potrzeby sprowadzane do pamięci operacyjnej. Przydział pamięci odbywa się zgodnie z algorytmem FIRST-FIT. Obszary danych i kodu są ładowane do spójnych obszarów pamięci operacyjnej. Gdy proces rozrasta się, cały jest kopiowany do nowego większego obszaru. Jeśli takiego obszaru nie ma, odsyłany jest do pamięci pomocniczej i czeka na odpowiednio duży obszar pamięci operacyjnej.

Do pamięci pomocniczej są usuwane również procesy czekające na zdarzenia o najniższych priorytetach wg strategii FIFO. Do pamięci operacyjnej są sprowadzane procesy najdawniej usunięte.

Wykonywanie programów odbywa się przez "wykonanie" pliku. Proces może "wykonać" plik za pomocą instrukcji

exec (filename, arg1, arg2 ..... argn)

Przed wykonaniem takiej instrukcji plik filename musi być otwarty, pozostaje również otwarty po zakończeniu wykonywania go. Wykonanie operacji exec polega na wymianie kodu i obszaru danych procesu na te, zapisane w pliku. Nie powstaje więc nowy proces tylko istniejący zmienia kod i środowisko. Nie zmienione pozostają jednak otwarte pliki. Dawne segmenty kodu i danych procesu są stracone.

SHELL

W systemie UNIX interpreter języka opisu prac nie jest integralną częścią systemu. Jest on wykonywany jako jedna z prac. Istnieje możliwość dołączenia do systemu różnych interpreterów dających użytkownikowi różne języki opisu prac.

Standardowym interpreterem zapewniającym komunikację z systemem UNIX jest program SHELL. Czyta on instrukcje użytkownika i interpretuje je jako polecenia wykonania odpowiednich programów standardowych lub zdefiniowanych przez użytkownika. Dzięki temu zbiór instrukcji języka opisu prac jest nieograniczony.

W najprostszej postaci instrukcja zawiera nazwę i argumenty oddzielone spacjami:

$$\text{command } arg_1\ arg_2\ \ldots\ldots\ arg_n$$

Nazwa instrukcji (command) jest nazwą zbioru, który zawiera program do wykonania. SHELL poszukuje zbioru o tej nazwie, a jeżeli taki nie istnieje, zbioru o nazwie rozszerzonej o przedrostek (bin). Katalog /bin zawiera wszystkie zbiory potrzebne do wykonywania standardowych instrukcji. Argumenty instrukcji są parametrami wykonywach programów. SHELL dostarcza, poza instrukcjami w najprostszej postaci, wiele konstrukcji rozszerzających możliwości współdziałania użytkownika z systemem. Standardowo do wykonania każdej instrukcji interpreter powołuje nowy proces, czeka na jego zakończenie, a następnie drukuje znak zachęty informujący o gotowości przyjęcia następnej instrukcji. Można jednak spowodować, że instrukcja będzie wykonywana w tle, czyli bez oczekiwania na jej zakończenie. W tym wypadku za ostatnim argumentem należy umieścić znak "&".

Przykładowo instrukcja

```
ls -l &
```

listuje nazwy plików bieżącego katalogu wraz za datami ostatniego użycia, rozmiarem i innymi informacjami. ls jest nazwą instrukcji, -l jej argumentem, a znak "&" sprawi, że interpreter nie będzie czekał na jej zakończenie, lecz po zainicjowaniu odpowiedniego procesu będzie gotowy przyjąć następną instrukcję.

Proces wykonujący instrukcję może korzystać z wejścia i wyjścia. Standardowo są to pliki specjalne związane z terminalem. Można jednak jako wejście lub wyjście użyć innych plików zwykłych lub specjalnych, np. wykonanie instrukcji

```
ls -l >wyj
```

spowoduje wylistowanie nazw do pliku o nazwie wyj, a przy wykonywaniu instrukcji edytora

```
ed <wej
```

dane będą pobierane z pliku wej.

Proces wykonujący instrukcję

```
ed <wej >wyj
```

będzie pobierał dane z pliku o nazwie wej a wyniki wysyłał do pliku wyj. Poza tym możliwe jest podłączenie wyjścia jednej instrukcji do wejścia innej za pomocą konstrukcji

```
ls | ed
```

W tym wypadku listowane przez instrukcję ls nazwy plików będą stanowiły dane wejściowe dla instrukcji ed. Za pomocą tego mechanizmu (pipe) można połączyć kilka instrukcji

```
ls | grep old | wc
```

Wejściami instrukcji grep drukującej nazwy zbiorów z bieżącego katalogu, zawierające powien łańcuch (w tym wypadku old) i wc zliczającej liczbę znaków, słów i linii na wejściu będą wyjścia instrukcji poprzedzających, czyli odpowiednio ls i grep.

Po wykonaniu każdej instrukcji zwracany jest kod powrotu, który informuje czy została ona wykonana poprawnie, czy nastąpił błąd. Kod powrotu może być testowany.
SHELL dostarcza następujących konstrukcji sterujących:

a) <u>if</u> command-list1
<u>then</u> command-list2
<u>else</u> command-list3
<u>fi</u>

Konstrukcja ta działa podobnie jak instrukcja if then else w innych językach, np. PASCAL, ALGOL 68, itp.
Command-list jest to sekwencja instrukcji oddzielonych średnikiem lub umieszczanych w nowej linii. Najpierw wykonywane są instrukcje z command-list1, potem testowany jest kod powrotu ostatniej wykonanej instrukcji w tej sekwencji, a następnie, w zależności od wartości tego kodu, wykonywana jest sekwencja command-list2 lub command-list3. Można również tworzyć wyrażenia logiczne za pomocą operatorów && (ANDF) i || (ORF), argumentami są w tym wypadku kody powrotu danych instrukcji.
Np.

command1 && command2

Instrukcja command2 jest wykonywana tylko wtedy, gdy command1 zakończy się poprawnie.
Gdy

command1 || command2

instrukcja command2 jest wykonywana tylko wtedy, gdy command1 zakończy się błędnie. Człon else może być opuszczony. Cała konstrukcja if ... fi jest traktowana jako jedna instrukcja, tak samo pozostałe konstrukcje sterujące.

b) <u>while</u> command-list1
<u>do</u> command-list2
<u>done</u>

Działanie tej konstrukcji jest podobne do działania instrukcji while w języku PASCAL.

SHELL dostarcza również instrukcji break i continue z argumentami będącymi liczbami naturalnymi i mówiącymi ile zagłębień pętli opuścić (break), lub który poziom zacząć testować (continue).

c) <u>case</u> word <u>in</u>

wzorzec1) command-list1;;
wzorzec2) command-list2;;
⋮
<u>esac</u>

Word jest zmienną typu string, wzorzec jest stringiem zawierającym oprócz zwykłych znaków alfanumerycznych również następujące znaki specyfikujące

* - odpowiada dowolnemu stringowi (nawet zerowemu)

? - odpowiada pojedynczemu dowolnemu znakowi

[...] - odpowiada jednemu ze znaków zawartych wewnątrz

Jeśli w nawiasach znajdują się dwa znaki rozdzielone myślnikiem, to odpowiada to dowolnemu znakowi zawartemu leksykalnie między nimi.

Wzorzec można również wyrazić jako alternatywę

wzorzec1 | wzorzec2)

SHELL próbuje znaleźć zgodność między zmienną word i kolejnymi wzorcami, jeśli taka zgodność wystąpi wykonywana jest odpowiednia command-list i instrukcja jest kończona.

Przykład

```
case word in
        *.c ) command1;;
    [a-k] *) command2;;
    [m,n,o]  command3;;
     xx | yy) command4;;
     ? )      command5;;
     żaba )   command6;;
     *   )    command7;;
esac
```

Jeśli zmienna word będzie stringiem zakończonym znakami .c to wykonana zostanie instrukcja command1, jeśli stringiem rozpoczynającym się literą z przedziału a-k to command2, jeśli znakiem m,n lub o to command3, jeśli stringiem xx lub yy to command4, jeśli pojedynczym dowolnym znakiem to command5, jeśli stringiem "żaba", to command6, jeśli poprzednie wzorce nie będą odpowiadały to będzie wykonana instrukcja command7, gdyż wzorzec * pełni taką rolę jak etykieta default.

d) for word in w1, w2, w3 ..... w_n
do command-list
done

Pętla ta jest wykonywana n-razy, za każdym razem zmienna word przyjmuje kolejną wartość w1, w2 ... w_n.

SHELL może czytać i wykonywać instrukcje zapisane w pliku. Taki plik jest nazywany procedurą języka SHELL. "Wywołaniem" takiej procedury jest instrukcja

sh filename , arg1 arg2 ...

Argumenty arg1, arg2,... nie są argumentami instrukcji sh, lecz argumentami procedury czyli kolejnymi argumentami instrukcji występujących w pliku filename, a zaznaczonymi tam jako

$1, $2 ... . Np. jeśli plik zawiera instrukcję

```
who | grep $1
```

to wykonanie instrukcji

```
sh wg fred
```

odpowiada wykonaniu instrukcji

```
who | grep fred
```

Dodatkowymi argumentami instrukcji sh umożliwiającymi śledzenie wykonywania procedury są:

- -v - listuj wszystkie linie z pliku, tak jak są tam zapisane,
- -x - listuj instrukcje w kolejności, w jakiej są wykonywane,
- -n - nie wykonuj instrukcji,
- -e - zatrzymaj procedurę, gdy wystąpi błąd.

Argumenty te są umieszczane po nazwie instrukcji, np.

```
sh -v wg fred
```

Zmienne w SHELL są typu string. Argument procedury języka SHELL może być postaci name=value i wtedy name pełni rolę parametru formalnego przekazywanego przez wartość, a value jest jego wartością. Takie zmienne jak name są nazywane parametrami kluczowymi. Podstawienie wartości pod zmienną wewnątrz procedury wygląda podobnie:

```
user=fred
```

Generalnie notacja paramterów i zmiennych w procedurze wygląda następująco:

$ {name}

Jeżeli parametr nie ma nadanej wartości, to podstawiany jest string pusty.

Zapis

$ {d-.}

oznacza, że należy podstawić wartość zmiennej d lub, jeśli takiej nie ma, to "." . A w wypadku zapisu

$ {d- $1}

wartość parametru $

Zapis

$ {d? message}

oznacza, że jeśli d nie będzie miało nadanej wartości, to wysłany będzie komunikat message lub w wypadku zapisu $ {d?} komunikat standardowy.

Oto niektóre zmienne o specjalnym znaczeniu:

- $? kod powrotu ostatnio wykonanej instrukcji (string cyfr),
- $ # liczba parametrów pozycyjnych (string cyfr),
- $$ nr procesu wykonującego tę instrukcję,
- $! nr procesu ostatnio zainicjowanego w tle,
- $- obecne "flagi" SHELL'a

Wartości tych zmiennych mogą być odczytywane ale nie mogą być zmieniane.
Argumenty procedur będące nazwami plików mogą być generowane za pomocą znaków specyfikujących używanych w konstrukcji case. I tak np.

*.c	- oznacza wszystkie nazwy plików w bieżącym katalogu kończące się na .c,
[a-z]*	- oznacza wszystkie nazwy plików w bieżącym katalogu zaczynające się na jedną z liter od a do z,
user/srb/test/?	- oznacza wszystkie nazwy plików z katalogu user/srb/test będące pojedynczymi znakami
*	- oznacza wszystkie pliki z bieżącego katalogu, oprócz zaczynających się od znaku "." .

Każdy znak specjalny ?, * , $ ... traci swoje znaczenie jeśli zostanie poprzedzony znakiem "\".
Grupa znaków specjalnych traci znaczenie jeśli będzie ograniczona znakami ' ' .
Wszystkie znaki specjalne oprócz $, ' , " , \ tracą swoje znaczenie jeśli są ograniczone znakami " " .

W języku SHELL istnieje możliwość podstawiania instrukcji. Operacja ta polega na podstawieniu wartości wyjściowej instrukcji pod zmienną, np.

d=`pwd`

d - zmienna
pwd - instrukcja drukująca nazwę bieżącego katalogu.
Wykonanie tej operacji spowoduje podstawienie pod zmienną d nazwy bieżącego katalogu.

Zakończenie

Na zakończenie podsumujemy główne cechy użytkowe systemu UNIX:

- organizacja hierarchicznego systemu plików,
- możliwość inicjowania procesów asynchronicznych,
- duża liczba programów dostępnych pod systemem,
- wygodny i efektywny język do współpracy użytkownika z systemem,
- łatwość współpracy z urządzenia wejścia/wyjścia,
- wysoki stopień przenoszalności.

Ostatnia cecha w największym stopniu przyczynia się do wzrostu popularności systemu. Wynika ona przede wszystkim z cech języka, w którym system UNIX jest napisany, jest to język C. Został stworzony dla maszyny PDP-11 z systemem operacyjnym UNIX, ale nie jest związany z konkretną maszyną czy systemem. W tej chwili kompilatory tego języka działają na wielu maszynach. Uniwersalność tego języka i brak ograniczeń sprawiają, że jest on wygodny i efektywny dla bardzo wielu zastosowań. Niezwykle istotną cechą języka C jest możliwość zapisywania programów wysoce przenoszalnych; zasadniczo przeniesienie takiego programu między różnymi maszynami wymaga implementacji funkcji bibliotecznych.

System UNIX jest w około 90% napisany w języku C. Pod systemem UNIX dostępne są różne programy pomocnicze, jak np.

- edytor tekstowy,
- assembler,
- debugger symboliczny,

oraz kompilatory niektórych języków: C, FORTRAN 77, BASIC, SNOBOL, APL, ALGOL 68, PASCAL. Sukces systemu UNIX polega nie tyle na zastosowaniu w nim nowych rozwiązań, gdyż wiele z nich było już uprzednio stosowanych w innych systemach [4,5,6], lecz raczej na pełnym wykorzystaniu dobrych i twórczych pomysłów, a w szczególności na pokazaniu, że mogą one być kluczami do implementacji małych lecz efektywnych systemów operacyjnych.

Pomimo ogólnego charakteru informacji zawartych w tym opracowaniu, mamy nadzieję, że pozwolę one w pewnym stopniu zaznajomić się ze znanym dziś na całym świecie standardowym systemem operacyjnym UNIX.

Literatura

[1] UNIX TIME-SHARING SYSTEM - zbiór artykułów The Bell System Technical Journal, 1978, T.57 nr 6

[2] UNIX - Wiadomości ogólne. Marchewka-Padzik B., Wilczek T.: maszynopis nie publikowany (materiały seminaryjne)

[3] Ramowa koncepcja multikomputera SOLID (Nr tematu 326) - opr. zbior. pod kierunkiem mgr inż. E.Jezierskiej-Ziemkiewicz (maszynopis)

[4] Deutsch L.P., Lampson B.W.: SDS 930 time-sharing system preliminary reference manual. Doc. 30.10.10, Project GENIE, U. of California at Berkeley 1965

[5] Feiertag R.J., Organick E.T.: The Multics input-output system. Proc.Third Symp. on Oper. Syst. Princ., Oct. 18-20, New York 1971 s.35-41

[6] Bobrow D.G. i in.: TENEX, a paged time sharing system for the PDP-10. Comm. ACM 1972 t.15 nr 3 s.135-143.

Biuletyn Informacyjny NAUKI I TECHNIKI KOMPUTEROWE

mgr inż. Joanna WYŁUPEK
Instytut Maszyn Matematycznych

E3S - nowa norma ESONE dla systemów modularnych
Podstawowe wymagania funkcjonalne i sprzętowe

Dokument normalizacyjny określany w skrócie E3S (= ESSS - ESONE Small System Standard) przygotowany został przez jeden z Zespołów Roboczych Komitetu ESONE (European Standards on Nuclear Electronics) i przedstawiony Komitetowi do zatwierdzenia. Norma zawiera zalecenia z zakresu realizacji sprzętowej i oprogramowania, przeznaczone dla systemów modularnych o różnym stopniu złożoności, poczynając od najprostszych systemów jednoprocesorowych a na rozbudowanych systemach wieloprocesorowych i wielozadaniowych kończąc.

System E3S ma szansę stać się standardem światowym, ale ma poważnych konkurentów do tego miana. Początkowo był to opracowany przez IEEE przy udziale ESONE system P896. Gdy zainteresowanie tym systemem zmalało, pojawił się nowy konkurent. Jest nim zaproponowany przez pięć zachodnich firm system VME.

Opisywany system E3S, zdefiniowany przez ESONE, organizację, która określiła wszystkie istniejące już i uznane normy CAMAC, nazywamy w Polsce systemem CAMAC S. Przedstawione dalej zalecenia oparte są na projekcie normy z września 1981 r. ("Proposal for a Small System Standard E3S Submitted by Small System Study Group to ESONE AGA Zurich 1981").

W opracowaniu dokonano skrótów oraz pominięto cały rozdział dokumentu dotyczący oprogramowania.

KONSTRUKCJE MECHANICZNE

Mechanika E3S jest zgodna z mechaniką eurokarty. Płytka E3S może być pojedynczą lub podwójną eurokartą:

wysokość 100 mm lub 233,35 mm
głębokość 220 mm
grubość 1.6 mm

Płytka może być wyposażona w płytę czołową. Szerokość tej płyty musi być wielokrotnością 4TE (1TE = 5,08 mm). Bloki E3S umieszczane są w kasetach o wysokości 3U lub 6U (1U = 44,45 mm) i szerokości 84TE, przy czym w kasecie o wysokości 6U mogą być umieszczane zarówno bloki o pojedynczej jak i podwójnej wysokości. Bloki połączone są z magistralą kasety złączami nakładanymi typu C. Skrajna lewa pozycja kasety, będąca stanowiskiem bloku arbitra wyposażona jest w złącze C96 (3x32 kontakty). Na pozostałych dwudziestu pozycjach używane są złącza C64. Bloki o podwójnej wysokości mogą być wyposażone w dodatkowe (górne) złącze DIN 41612 służące do dowolnych połączeń zewnętrznych.

STRUKTURA MAGISTRALI

Magistrala E3S zgodna jest z brytyjską normą BSI-EUROBUS i składa się z następujących linii:

HØ-H24	- 24 linie adresu/danych (24 bitowy adres, 16 bitowe dane na liniach HØ-H15)
AdMØ,AdM1	- 2 linie modyfikacji adresu
BytWk,BytAdØ,BytAd1	- 3 linie pracy bajtowej
CcBn,CcRes,CcFin	- 3 linie sterowania cyklem (początek, odpowiedź, koniec)
BusGr,BusAcq	- 2 linie dostępu do magistrali (przydział, przyjęcie)
It	- 1 linia przerwania
Rs,CcAbort,PF	- 3 linie sterowania kasety (zerowanie, kasowanie cyklu, zanik zasilania)
BusDeal	- 1 linia zmiany przydziału magistrali
Rq(n)	- 1 linia żądania przydziału łącząca indywidualnie każde z 20 stanowisk ze stanowiskiem arbitra
Rezi-Rez8	- 8 linii adresowych rezerwowych
	9 linii ØV
	3 linie +5V
Razem	60 linii.

Każda z tych linii ma przydzielony w E3S konkretny styk złącza C64 i C96. Wszystkie nadajniki magistrali są układami z otwartym kolektorem i wszystkie linie są aktywizowane stanem niskim (logika ujemna).

CYKLE MAGISTRALI

Informacje przesyłane są po magistrali adresów/danych asynchronicznie pod kontrolą trzech sygnałów przeplotu (ang. handshake): CcBn, CcRes, CcFin. Prawo zapoczątkowania i prowadzenia transmisji ma tylko ten blok, który uprzednio uzyskał przydział magistrali.

Po uzyskaniu przydziału, blok staje się aktualnym blokiem Master i może zainicjować jeden z trzech podstawowych cykli magistrali: cykl zapisu, cykl odczytu lub cykl wektorowy. Każdy cykl magistrali rozpoczyna się od zaadresowania bloku, do którego jest kierowany. Blok, który zdekodował swój adres staje się blokiem Slave.

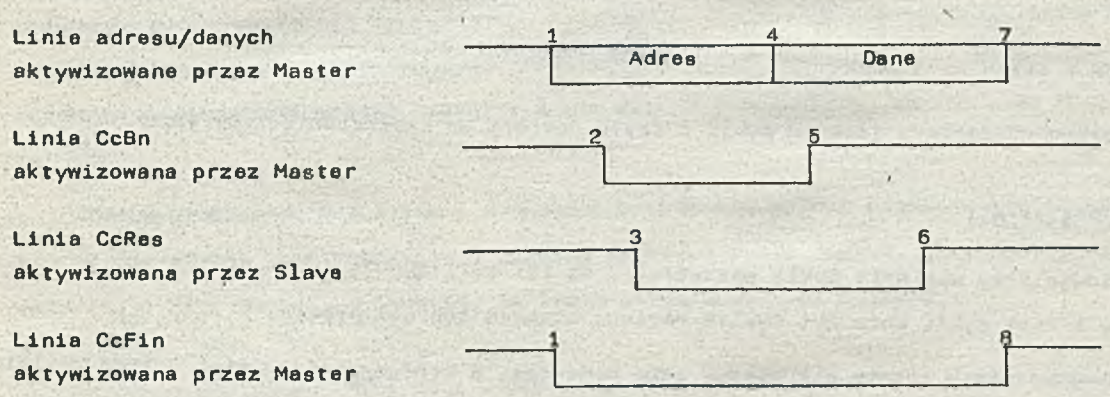
Linia CcBn aktywizowana jest tylko przez blok Master, linia CcRes tylko przez blok Slave, a linia CcFin zależnie od rodzaju cyklu przez blok Master, oba bloki lub nie jest aktywizowana wcale.

Aby zapobiec zablokowaniu magistrali, E3S przewiduje wprowadzenie mechanizmu time-out, który powoduje, że arbiter sygnałem CcAbort kasuje rozpoczęty cykl, jeśli trwa on zbyt długo.

Podstawowe cykle magistrali

- Cykl zapisu

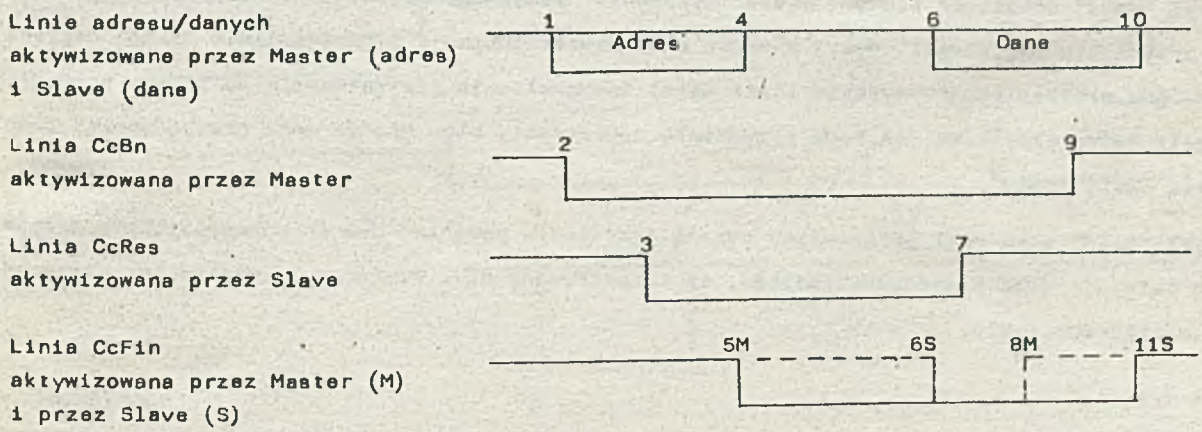
W cyklu zapisu blok Master przesyła dane do bloku Slave.



W cyklu zapisu blok Slave informowany jest o rodzaju cyklu już w fazie adresowej. Informację tę niesie linia CcFin w momencie pojawienia się sygnału na linii CcBn.

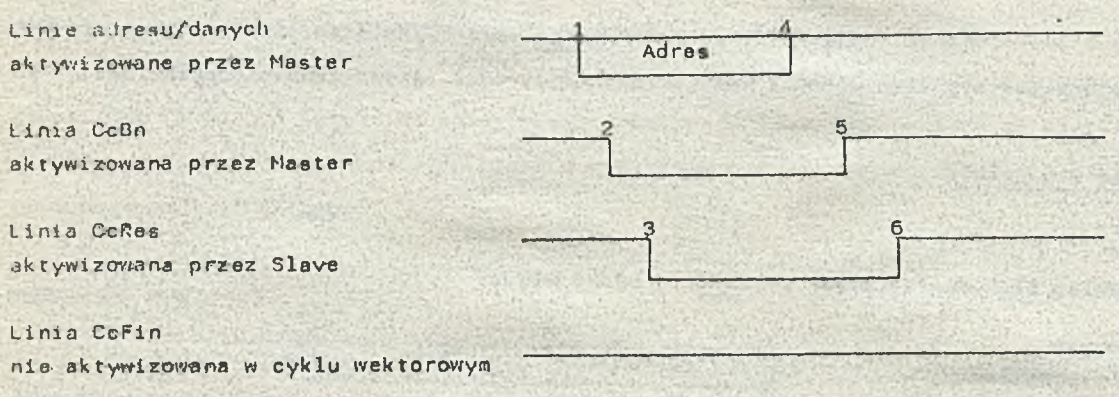
- Cykl odczytu

W cyklu odczytu blok Master czyta dane z bloku Slave.



- Cykl wektorowy

W tym cyklu nie są przesyłane dane, blok Master przesyła do bloku Slave tylko adres.



Cykl wektorowy odróżniany jest od cyklu odczytu dopiero po zakończeniu fazy adresowej.

Warianty cykli magistrali

E3S przewiduje trzy warianty cykli magistrali. Są to: cykl zwykły, cykl "hold", cykl "retain". Każdy z tych cykli może być cyklem zapisu, odczytu lub wektorowym.

Poszczególne warianty różnią się między sobą momentem, w którym bieżący blok Master zwalnia magistralę, umożliwiając arbitrowi kasety dokonanie nowego przydziału.

- Cykl zwykły

W cyklu zwykłym zmiana przydziału może być dokonywana podczas trwania cyklu, gdyż bieżący blok Master zwalnia linię przyjęcia przydziału (BusAcq) natychmiast po wysłaniu sygnału CcBn.

- Cykl "hold"

Cykl "hold" umożliwia blokowi Master wykonanie następnego cyklu bez konieczności żądania nowego przydziału magistrali. W tym wypadku blok Master zdejmuje sygnał z linii BusAcq dopiero wówczas, gdy arbiter stanem wysokim linii Rq(n) zasygnalizuje przygotowanie do zmiany przydziału. Jeżeli żaden inny blok nie żąda przydziału magistrali, blok Master może przeprowadzić dowolnie wiele cykli "hold".

Cykl "hold" jako cykl opóźniający zmianę przydziału powinien być stosowany tylko w przypadku, gdy istnieje duże prawdopodobieństwo, że rozpatrywany blok skorzysta z możliwości przeprowadzenia następnego cyklu.

- Cykl "retain"

W cyklu "retain" i bezpośrednio po nim (do momentu pojawienia się sygnału CcBn w następnym cyklu) linia BusAcq utrzymywana jest bezwarunkowo w stanie niskim. Stan niski linii BusAcq powstrzymuje arbitra od dokonania nowego przydziału.

Cykl "retain" umożliwia blokowi Master przeprowadzenie dwóch niepodzielnych cykli, a w szczególności operacji czytaj/modyfikuj/pisz.

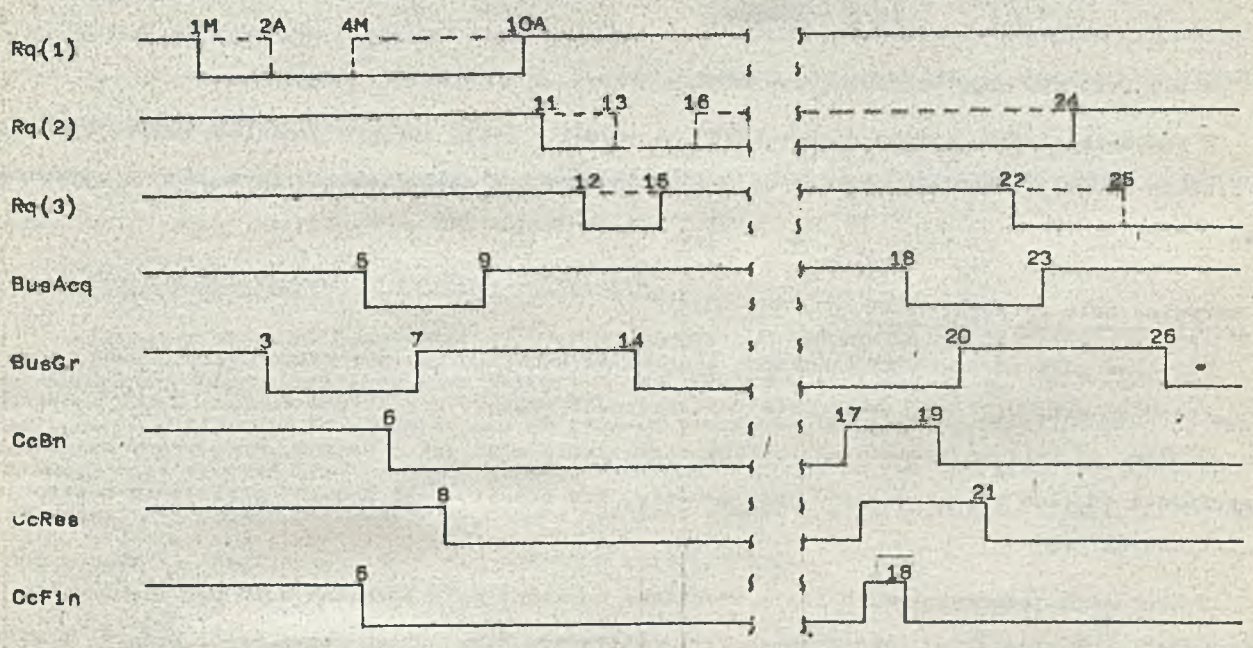
Aby zapobiec monopolizacji magistrali należy ograniczyć liczbę kolejnych cykli "retain" inicjowanych przez jeden blok.

PRZYDZIAŁ MAGISTRALI

W prostym systemie "singlemaster" jeden blok systemu ma stały dostęp do magistrali, będąc na stałe blokiem Master. W bardziej złożonych systemach za przydział magistrali odpowiedzialny jest blok arbitra.

E3S nie określa żadnych obowiązujących ani nawet zalecanych algorytmów arbitrażowych. Określony jest natomiast sposób, w jaki żądający przydziału i wybrany przez arbitra blok uzyskuje możliwość przeprowadzenia transmisji. Potencjalny Master żąda przydziału linią Rq(n). Arbiter przydzielając magistralę aktywizuje jednocześnie linię BusGr i Rq(n). Stan niski na linii BusAcq oznacza, że przydział został przyjęty i nie może być zmieniony, dopóki blok Master nie zwolni tej linii.

Poniższy diagram przedstawia mechanizm przydziału wolnej magistrali oraz magistrali, po której prowadzona jest transmisja z użyciem zwykłych cykli zapisu. (M oznacza, że linię Rq(n) aktywizuje blok Master, A oznacza, że linię aktywizuje blok arbitra).



ADRESOWANIE

E3S przewiduje adresowanie logiczne 24- i 16-bitowe z możliwością przyszłego rozszerzenia adresu do 32 bitów (8 linii adresowych rezerwowych). Wraz z adresem przesyłanym liniami adresu/ /danych na magistralę musi być podawana informacja o trybie adresowania. Informację tę niosą linie modyfikacji adresu AdMØ i AdM1 oraz linie pracy bajtowej BytWk i BytAdØ. Dla rozpoznania swojego adresu na magistrali, blok E3S musi uwzględniać także stan tych linii.

Całe pole adresowe podzielone jest w zależności od stanu linii BytWk i BytAØ na tzw. pole pseudoadresów PAS ((BytWk, BytAdØ) = (Ø,1)) i normalne pole adresowe NAS ((BytWk, BytAdØ) ≠ (Ø,1)). W PAS dane przesyłane są zawsze w postaci 16-bitowych słów. W NAS w zależności od stanu linii BytWk i BytAdØ przesyłane są 16-bitowe słowa lub 8 bitowe bajty. I tak:

BytWk	BytAdØ	
1	0	przesyłanie młodszego bajtu w NAS
1	1	przesyłanie starszego bajtu w NAS
0	0	przesyłanie 16-bitowego słowa w NAS
0	1	przesyłanie 16-bitowego słowa w PAS

Zarówno PAS jak i NAS może zawierać adresy 16- i 24-bitowe. Informacja o długości słowa adresowego przesyłana jest liniami AdMØ i AdM1:

AdM1	AdMØ	
0	0	zarezerwowane
0	1	16-bitowy adres
1	0	24-bitowy adres
1	1	zarezerwowane

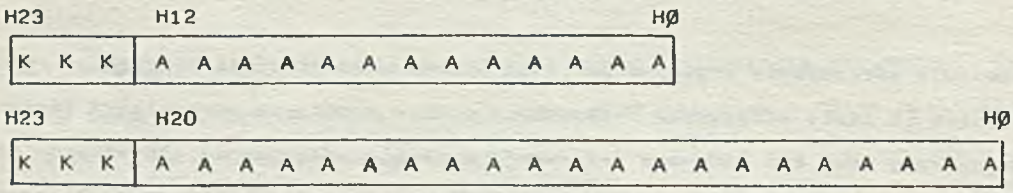
W systemie E3S mogą współpracować bloki adresowane słowem 16- i 24-bitowym.

Z powyższego opisu wynika na przykład, że jeżeli w cyklu odczytu prześlemy konkretny adres po liniach HØ-H24, to możemy uzyskać do 8 różnych danych w zależności od użytego trybu adresowania.

Normalne pole adresowe (NAS)

Normalne pole adresowe przeznaczone jest do adresowania pamięci zawierających programy i dane. Zaadresowany blok musi właściwie interpretować sygnały przesyłane liniami BytWk i BytAdØ i prawidłowo odpowiadać zarówno na rozkazy przesyłania słów jak i bajtów. Przy pracy bajtowej, niezależnie od tego czy przesyłany jest młodszy, czy starszy bajt danych, przesłanie następuje po liniach HØ-H7.

W systemach jednokasetowych 16- i 24-bitowe normalne pole adresowe może być używane bez ograniczeń. W systemach wielokasetowych trzy najstarsze bity adresu niosą numer kasety, a więc pamięć wewnątrz określonej kasety jest adresowana słowem 13- lub 21-bitowym.



Pole pseudoadresów (PAS)

W polu pseudoadresów mieszczą się adresy wszystkich urządzeń peryferyjnych i bloków wejścia/wyjścia (bloki te będą dalej nazywane także urządzeniami). W PAS zawarte są również adresy (wektory) przesyłane w cyklach wektorowych.

- Adresowanie urządzeń

W E3S możliwy jest swobodny lub chroniony dostęp do urządzeń adresowanych w PAS. Urządzenia z dostępem swobodnym są adresowane słowem 16-bitowym. 24-bitowy adres wykorzystują tylko urządzenia z dostępem chronionym.

Adresowanie urządzeń z dostępem swobodnym

16-bitowy adres przy dostępie swobodnym podzielony jest na dwa bajty. Starszy bajt jest identyfikatorem bloku, młodszy niesie adres rejestru wewnątrz bloku.

H15							H8	H7							HØ
K	K	K	B	B	B	B	B	R	R	R	R	R	R	R	R

Identyfikator bloku składa się z numeru kasety (K) i adresu bloku w kasecie (B). Przy przesyłaniu danych między blokami umieszczonymi w tej samej kasecie K=0 (również w systemach wielokasetowych). Niezerowa wartość K jest rozpoznawana przez blok tzw. łącznika (ang. linker) jako odwołanie międzykasetowe.

Adresy bloków B=Ø oraz B=28 i 31 są zarezerwowane. Bity R niosące adres rejestru wewnątrz bloku umożliwiają rozróżnienie 256 rejestrów.

Adresowanie urządzeń z dostępem chronionym

Ochrona dostępu do urządzenia odpowiada sytuacji, w której dostęp do danego kanału jest zarezerwowany przez określony blok Master. Przy chronionym dostępie używa się 24-bitowego słowa adresowego. Młodszych 16 bitów adresu ma to samo znaczenie, co przy dostępie swobodnym. Najstarszy bajt zawiera identyfikator bloku wysyłającego adres; w tym przypadku identyfikator nazywany jest sygnaturą. Jeżeli sygnatura nie jest zgodna z identyfikatorem bloku, który dokonał wcześniej rezerwacji, kanał nie zostanie zaadresowany.

H23							H16	H15							H8	H7							HØ
K	K	K	B	B	B	B	B	K	K	K	B	B	B	B	B	R	R	R	R	R	R	R	R
sygnatura								identyfikator								rejestr							

PRZYPORZĄDKOWANIE ADRESÓW W POLU ADRESOWYM BLOKU

Każdy blok adresowany w PAS musi mieć przyporządkowany identyfikator (H8-H15). Przyporządkowanie musi być dokonywane sprzętowo, w możliwie prosty sposób (np. przełącznikami). Adresy wewnątrz bloku liczone są od Ø do 255 (R=0÷255) względem początku pola adresowego bloku.

Rejestry bloku

W polu adresowym bloku cztery do ośmiu najmłodszych adresów zarezerwowanych jest dla rejestrów odnoszących się do bloku jako całości (bez podziału na kanały).

- Rejestr echa (R=0)

Rejestr echa przeznaczony jest do testowania drogi przesyłania danych. Blok Master zapisuje dowolne słowo do rejestru echa, a następnie odczytuje je i porównuje z tym, co było zapisane.

- Rejestr etykiety (R=1)

Zawartość tego rejestru ustawiana jest przez użytkownika np. przełącznikami i służyć może np. do oznaczenia typu bloku. Zapisywanie najmłodszego bitu tego rejestru powoduje wyzerowanie bloku.

- Rejestry zarezerwowane (R=2 i R=3)

Adresy tych rejestrów zarezerwowane są dla ewentualnych przyszłych zastosowań.

- Rejestry globalnego stanu bloku (R=4 do R=7)

Rejestry te mogą, ale nie muszą występować w blokach E3S. Jeżeli występują, to służą do przechowywania szeroko pojętej informacji o stanie bloku (np. może to być zbiór stanów poszczególnych kanałów bloku).

Pole wektorów przerwań

Pole wektorów przerwań (jeśli istnieje) zaczyna się od adresu R=4 lub R=8, zależnie od tego czy blok wyposażony jest w rejestry stanu globalnego. Jeżeli wektory nie zajmują kolejnych adresów w bloku, to opuszczone adresy muszą pozostać niewykorzystane. Wektory przerwań nie są rejestrami. Dane zapisywane do wektora są gubione, a dane odczytywane nieokreślone. Zaadresowanie konkretnego wektora (przesłanie go w tzw. "cyklu wektorowym") powoduje wykonanie przez blok określonej funkcji. W polu adresowym bloku może leżeć wiele wektorów przerwań, a każdy z nich może być związany z inną funkcją.

Rejestry kanałów

Blok składa się zwykle z jednego lub kilku kanałów (w szczególnej sytuacji blok zawiera tylko opisane wyżej rejestry bloku i nie ma ani jednego kanału). Każdy kanał jest niezależną jednostkę funkcjonalną. Z punktu widzenia magistrali kanał przedstawia się jako zbiór adresowanych kolejno rejestrów. Kanały w bloku muszą zajmować pola adresowe tej samej wielkości, nawet jeżeli nie wszystkie kanały wykorzystują całe przypisane im pole.

Pola adresowe kanałów zaczynają się powyżej pola adresowego rejestrów bloku i ewentualnego pola wektorów przerwań.

Adresy w określonym kanale liczone są od Ø do N względem początku pola adresowego tego kanału. Adres Ø przyporządkowany jest zawsze rejestrowi sterowania/stanu (CSR), który jest jedynym rejestrem obligatoryjnym kanału.

Inne rejestry, jeśli występują, umieszczone są w polu adresowym kanału w następującej kolejności: rejestry danych, rejestr przerwań, rejestr sygnatury, inne rejestry dodatkowe (np. rejestr adresu pamięci, licznik słów, dodatkowy stan).

- Rejestr sterowania/stanu (CSR)

Rejestr CSR ma następujący format:

H15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	HØ
Rsv	Stan					RsC	Err	Rdy		IEn	Funkcja				GO
R/W	RO					WO	RO	RO		R/W	R/W				WO

R/W oznacza bit, który można zarówno odczytać jak i zapisać

RO oznacza bit, który można tylko odczytać

WO oznacza bit, który można tylko zapisać.

Poszczególne bity i pola rejestru mają następujące znaczenie i funkcje:

Rsv - bit rezerwacji kanału, ustawiany przez blok dokonujący rezerwacji. Stan tego bitu w żaden sposób nie oddziałuje na kanał.

Stan - wskaźnik zależnego od urządzenia stanu lub błędu. Jeśli potrzebnych jest więcej bitów stanu, mogą być one umieszczone w dodatkowym rejestrze. Konkretne wartości bitów stanu mogą powodować wysłanie przerwania.

RsC - zerowanie kanału. Zapisanie RsC=1 zeruje wszystkie pozostałe bity CSR i sprowadza kanał do określonego stanu. Po zakończeniu tej operacji bit RsC jest automatycznie zerowany.

Err - błąd; jest sumą wszystkich warunków błędu wskazywanych przez bity stanu urządzenia. W prostych kanałach może być jedynym wskaźnikiem błędu.

Rdy - gotowość; Rdy=1 oznacza, że kanał zakończył operację i/lub jest gotów do wykonania następnej. Pojawienie się gotowości może powodować wysłanie przerwania.

IEn - odblokowanie przerwań. IEn=Ø blokuje przerwania wysyłane przez kanał. Bit ten jest zerowany automatycznie gdy kanał generuje przerwanie.

Funkcja - określa, zależną od kanału funkcję, która ma być wykonana gdy bit GO zostanie ustawiony na 1. W trakcie wykonywania operacji urządzenie nie może zmieniać bitów funkcji

GO - zapisanie GO=1 powoduje wykonanie przez kanał funkcji określonej bitami H1-H4 i wyzerowanie bitu gotowości (Rdy). Jeżeli Rdy=Ø ustawienie GO=1 nie powoduje żadnej reakcji kanału. GO ma charakter impulsu wyzwalającego. Kanał może być wystarczającą prosty, aby sam zapis lub odczyt rejestru danych wyzwalał jego działanie. Bit GO nie musi być wtedy używany, ale jego pozycja w CSR jest zarezerwowana.

H6 i H10 - bity 6 i 10 w rejestrze stanu są zarezerwowane.

- Rejestry danych

Jeżeli kanał wyposażony jest w rejestry danych, to są one umieszczone pod kolejnymi adresami 1, 2, itd., w polu adresowym kanału.

Dane odczytane mogą być uznane za ważne tylko jeżeli Rdy=1. Zapis do rejestru danych jest możliwy również tylko przy Rdy=1. Próba zapisu gdy Rdy=0 powinna wywoływać sygnalizację błędu.

- Rejestr przerwań

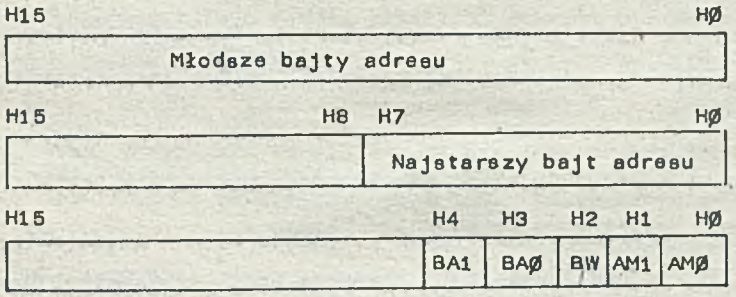
Rejestr przerwań występuje tylko w kanałach, które mogą wysyłać przerwanie. Zawartość rejestru przerwań może być ustawiana programowo, sprzętowo lub ręcznie (przełącznikami).

- Rejestr sygnatury

Rejestr sygnatury występuje tylko w blokach wyposażonych w możliwość ochrony dostępu do kanałów. Zapis do rejestru dokonywany jest sprzętowo podczas rezerwacji kanału. Nie ma żadnych ograniczeń dotyczących odczytu rejestru.

- Rejestr adresu

Dane zawarte w rejestrze adresu jednoznacznie wyznaczają adres na magistrali. Oznacza to, że oprócz 16- lub 24-bitowego słowa przesyłanego liniami adresowymi, musi się tam znaleźć pełna informacja o trybie adresacji (AdMØ, AdM1, BytAdØ, BytAd1, BytWk), a więc rejestr adresu może zajmować do 3 kolejnych adresów w polu adresowym kanału:



Jeżeli blok pracuje tylko w jednym trybie adresacji, rejestr adresu może być krótszy, a sygnały wysyłane liniami modyfikacji adresu i pracy bajtowej ustawiane sprzętowo.
Próba zapisu do rejestru adresu trybu adresacji, którego dany blok nie realizuje, powinna wywoływać sygnalizację błędu.

PRZERWANIA

Przerwania w systemie E3S przesyłane są za pomocą cykli wektorowych, w których następuje transmisja zawartości rejestru przerwań kanału przerywającego do pola wektorów przerwań procesora (zob. pkt "Pole wektorów przerwań"). Bity 8-15 przesyłanego wektora określają identyfikator bloku procesora, a bity Ø-7 numer przerwania zewnętrznego. Wysyłając przerwanie, kanał gasi bit IEn i jednocześnie może ustawiać bit gotowości i/lub bity stanu. Ustawianie bitów stanu staje się konieczne, gdy numer przerwania nie wyznacza jednoznacznie jego przyczyny, tzn. gdy kilka

kanałów wysyła to samo przerwanie lub gdy jeden kanał wysyła przerwanie z różnych przyczyn.

Blok, którego kanały wysyłają przerwania, musi mieć możliwość generacji cykli wektorowych, a więc współpraca takiego bloku z procesorem (również generującym cykle), bezwzględnie wymaga obecności w systemie arbitra dokonującego przydziału magistrali.

PRACA WIELOPROCESOROWA

E3S wprowadzając scentralizowany mechanizm przydziału magistrali umożliwia projektowanie i budowę systemów typu "multimaster", w których może ze sobą współpracować kilka procesorów, ale pojawiają się problemy dostępu do pamięci i urządzeń systemu.

Dostęp do pamięci

Bloki procesorów zawierają zwykle pamięci lokalne, ale system może być wyposażony w dodatkowe bloki pamięci, które są traktowane jako rozszerzenie pamięci lokalnych lub jako pamięć wspólna. W systemie musi być zapewniona możliwość dzielenia pamięci przez procesory o różnej strukturze adresowania. Przyjęto, że w przyszłości adresy pamięci będą zawsze odwzorowywane.

Odwzorowanie realizowane jest przez mniej lub bardziej skomplikowane uogólnienie rejestrów bazy i granicy, które dokonują automatycznej translacji adresów logicznych na adresy fizyczne. Odwzorowanie będzie przeprowadzane przez blok procesora, a więc adresy na magistrali będą zawsze adresami fizycznymi 16- lub 24-bitowymi, zależnie od typu pamięci.

Ochrona pamięci, wprowadzana automatycznie przez odwzorowywanie, powinna w maksymalnym stopniu być realizowana sprzętowo (całkowicie sprzętowa realizacja ochrony możliwa byłaby przy odwzorowywaniu dokonywanym w blokach pamięci, ale wprowadzenie takiego wymagania przez normę powodowałoby znaczny, a nieuzasadniony wzrost kosztów najprostszych systemów).

Dostęp do kanałów

W systemach wieloprocesorowych, blok procesora musi mieć możliwość rezerwowania kanałów na pewien okres czasu do wyłącznego swojego użytku. W zależności od wymaganego stopnia nienaruszalności takiej rezerwacji, ochrona dostępu do zarezerwowanego kanału może być zrealizowana programowo lub sprzętowo. E3S dopuszcza tutaj pewną dowolność definiując dwa "poziomy" rezerwacji kanałów:

rezerwację

rezerwację z ochroną dostępu.

- Rezerwacja

Dla sygnalizowania rezerwacji używane są bity Rsv w rejestrach CSR wszystkich kanałów. Procesor, który chce zarezerwować określony kanał, musi przejść przez trzy następujące fazy:

- faza 1: cykl "retain"

odczyt i zapamiętanie wartości CSR

A = (CSR);

- faza 2: cykl zwykły

  zapis do CSR zmiennej A z bitem 15 ustawionym na 1;

  jeżeli kanał był wolny, zostanie zarezerwowany, jeżeli był już zarezerwowany (bit 15 = 1), zapis nie spowoduje żadnej zmiany;

- faza 3: sprawdzenie czy rezerwacja powiodła się:

  jeżeli bit 15 zapamiętanego w fazie 1 CSR:

  = 1 rezerwacja nie powiodła się

  = 0 procesor dokonał rezerwacji.

Przedstawiony mechanizm rezerwacji może być realizowany zarówno przy adresowaniu 16- jak 24-bitowym.

- Ochrona dostępu

Przebieg rezerwacji z ochroną dostępu programowo jest identyczny z przedstawionym w punkcie "Rezerwacja".

Cała różnica polega na zachowaniu się sprzętu: w fazie 2, jeżeli rezerwacja powiodła się, identyfikator procesora wprowadzany jest automatycznie do rejestru sygnatury rezerwowanego kanału.

Dostęp do rezerwowanego w ten sposób kanału realizowany jest przy użyciu 24-bitowego adresu; sygnatura niesiona przez najstarszy bajt adresu porównywana jest w fazie adresowej cyklu z zawartością rejestru sygnatury. Jeżeli nie wystąpi zgodność, kanał nie zostanie zaadresowany. Wyjątek stanowi tu sytuacja, w której zawartość rejestru sygnatury jest zerowa, tzn. kanał jest wolny. Zerowanie bitu Rsv rejestru CSR automatycznie zeruje rejestr sygnatury. Bloki realizujące sprzętową ochronę dostępu muszą być wyposażone w przełącznik blokujący mechanizm ochrony i umożliwiający pracę ze zwykłą rezerwacją.

SYSTEMY WIELOKASETOWE

E3S przewiduje możliwość zestawiania dużych systemów wielokasetowych (do 7 kaset). Transmisja międzykasetowa realizowana jest przez blok łącznika, którego obecność niezbędna jest w każdej kasecie. Łączniki połączone są ze sobą magistralą międzykasetową. W trakcie transmisji większość linii tej magistrali (linie adresu/danych, linie modyfikacji adresu, pracy bajtowej, sterowanie cyklu i in.) może być traktowana jako bezpośrednie połączenie odpowiadających im linii dwóch współpracujących kaset. Dane przesyłane są cykl po cyklu jak wewnątrz kasety. Adresacja jest jednolita we wszystkich kasetach. Kasety są numerowane od 1 do 7 (K=1 do K=7). W obrębie jednej kasety każdy blok Master adresuje lokalne bloki używając K=Ø. Jeżeli Master chce skomunikować się z blokiem umieszczonym w innej kasecie, podaje w adresie numer tej kasety (K=1 do K=7). Taki adres nie może być zdekodowany przez żaden blok Slave w kasecie, ale zostaje rozpoznany przez łącznik. Załóżmy, że magistrala międzykasetowa jest wolna i łącznik może na nią podać otrzymany adres wraz z sygnałem CcBn. Każdy łącznik rozpoznaje od strony magistrali międzykasetowej numer swojej kasety. Zaadresowany w ten sposób łącznik kasety docelowej żąda przydziału

magistrali lokalnej, a gdy uzyska przydział (łącznik powinien mieć najwyższy priorytet w kasecie) podaje na nią adres bloku docelowego ze zmienionymi bitami K (K=Ø) i sygnał CcBn. W ten sposób magistrale danych dwóch kaset zostają połączone w jedną całość a cykl magistrali rozpoczęty.

Łącznik w kasecie źródłowej, zanim poda adres na magistralę międzykasetową musi uzyskać do niej dostęp. Dokładny opis magistrali międzykasetowej nie jest jeszcze gotowy, ale E3S zaleca, aby była ona wyposażona w oddzielną linię żądania, na której realizowana będzie suma zgłoszeń od wszystkich łączników i współpracującą z nią w konfiguracji "daisy chain" linię przydziału. Taka realizacja przydziału ma tę wadę, że wprowadza ustalone priorytety kaset (kaseta 1 ma najwyższy priorytet, kaseta 7 najniższy).

Niedogodność ta może być złagodzona przez umieszczenie często współpracujących par bloków Master/Slave w jednej kasecie i przez umieszczenie w kasetach o najwyższym priorytecie tych procesorów, które powinny mieć pierwszeństwo w dostępie do magistrali. Jeżeli kilka łączników z różnych kaset żąda jednocześnie dostępu do magistrali, sygnał przydziału uzyska tylko łącznik kasety o najwyższym priorytecie. Pozostałe łączniki muszą wysłać do arbitrów swoich kaset sygnał zmiany przydziału BusDeal. W odpowiedzi na ten sygnał arbiter skasuje rozpoczęty cykl (CcAbort) i dokona nowego przydziału.

W IMM w ramach tematu "Program opracowań urządzeń sterujących CAMAC S" prowadzone są prace nad rozpoznaniem przewidywanych obszarów zastosowań systemów z "inteligencją rozłożoną przestrzennie" celem sformułowania zgodnych z E3S lub VME (w zależności od tego, który z systemów zostanie uznany za standard światowy) wymagań na urządzenia sterujące systemu.

Osoby i zespoły, które zainteresowane są wprowadzeniem nowego systemu w Polsce, a w szczególności osoby z dużym doświadczeniem w dziedzinie zastosowań systemów modularnych, proszone są o kontakt z autorką opracowania.

Biuletyn Informacyjny NAUKI I TECHNIKI KOMPUTEROWE

mgr inż. Stanisław ZAGÓRNY
Instytut Maszyn Matematycznych

Porównanie rozwiązań źródeł zasilania przy zastosowaniu zasilaczy liniowych lub impulsowych (z przetwarzaniem)

Wstęp

Urządzenia elektroniczne, budowane na układach scalonych, zasilane są głównie ze źródła o napięciu 5V i dużym prądzie obciążenia (zwykle około kilkudziesięciu amperów). W opracowaniu przedstawiono możliwość realizacji tego źródła za pomocą nowoczesnego zasilacza impulsowego (z przetwarzaniem) bądź też przez zastosowanie zwielokrotnionej liczby zasilaczy liniowych.

Każdy z wymienionych wariantów pociąga za sobą określone następstwa związane z wymiarami i ciężarem, parametrami elektrycznymi oraz ekonomiką rozwiązania (wkład dewizowy związany z elementami z krajów kapitalistycznych w zasilaczu z przetwarzaniem bądź też elementy sprowadzane w ramach importu centralnego z krajów socjalistycznych w zasilaczu liniowym). Zostało to ujęte w tabeli 2 po wcześniejszym krótkim scharakteryzowaniu urządzeń elektronicznych i źródeł zasilania oraz po analizie celowości stosowania zasilaczy z przetwarzaniem.

Krótka charakterystyka urządzeń elektronicznych i źródeł zasilania

W dotychczasowych rozwiązaniach urządzeń elektronicznych najczęściej stosuje się zasilacze ze stabilizacją "szeregową" liniową, tj. zasilacze wykorzystujące podczas pracy obszar aktywny charakterystyk tranzystora. Są to układy, które ze względu na wysoki współczynnik stabilizacji bywają często stosowane. Jednak mała ich sprawność pociąga za sobą duże wymiary i znaczne ilości wydzielanego ciepła, co staje się powodem, że spełnienie wymagań na chłodzenie staje się technicznie trudne. Układy te na obecnym etapie swojego rozwoju stają się więc już klasycznym środkiem zasilania urządzeń elektronicznych, szczególnie o dużej liczbie zawartych w nich elementów (duże moce strat - duże prądy obciążeń).

Jakościowy rozwój urządzeń komputerowych, charakteryzujący się stosowaniem układów scalonych o coraz większym stopniu integracji, a zatem i coraz większą mocą przypadającą na jednostkę objętości układów elektroniki, stawia nowe jakościowo wymagania wobec urządzeń zasilających. Miniaturyzacja urządzeń pociąga bowiem za sobą konieczność miniaturyzacji urządzeń zasilających. Pozostawienie układów zasilania w tradycyjnych dotychczasowych rozwiązaniach powoduje wzrost objętości i ciężaru całego urządzenia, pobieranie większej ilości energii z sieci, wydzielanie większej ilości ciepła, itp.

Pewne nasycenie kraju sprzętem komputerowym i innym powoduje wzrost zapotrzebowania na energię elektryczną, które w określonych porach staje się dość krytyczne. Jednak przy takim stanie rzeczy prawie wcale nie zwraca się uwagi na sprawność urządzeń, co jest zjawiskiem niezbyt zrozumiałym. Przy tym na sprawność należy patrzeć w sposób możliwie szeroki i nie traktować jej jako parametru tylko ściśle energetycznego. Większa bowiem sprawność to nie tylko mniejsze zużycie energii, ale także:

- mniejsze ilości wydzielanego ciepła (a więc łatwiejsze rozwiązanie zagadnienia wentylacji i łatwiejsze warunki eksploatacyjne)
- mniejsze gabaryty urządzenia
- mniejszy ciężar
- mniejsze zużycie materiałów, a zatem i mniejsze zużycie energii przez obrabiarki na obróbkę podzespołów
- mniejsze powierzchnie magazynowe
- mniejsze powierzchnie zajmowane przez urządzenie
- łatwiejszy transport, itp.

Można powiedzieć, że współczesne urządzenia komputerowe nakładają na zasilacze wymagania, które można przedstawić następująco:

- duże prądy obciążenia źródeł (przeważnie dziesiątki amperów)
- stosunkowo niskie napięcia
- małe wymiary i ciężar
- wysoka sprawność i niezawodność.

Spełnienie tych wymagań powoduje konieczność zastosowania źródeł zasilania zrealizowanych w odpowiednich technikach.

Próby zwiększenia sprawności energetycznej spowodowały ukierunkowanie prac na nieciągłą czyli przerywaną stabilizację napięcia. Poszukiwanie rozwiązań bardziej ekonomicznych z jednoczesną redukcją wymiarów (których głównym reprezentantem był transformator sieciowy) doprowadziło do budowy stabilizowanych zasilaczy z przetwarzaniem częstotliwości. Są one w ostatnich latach stosowane coraz częściej - można nawet powiedzieć masowo - przeważnie dla mocy od kilkudziesięciu watów do kilku kilowatów.

Możliwość budowy tego typu zasilaczy nastąpiła w momencie pojawienia się na rynku półprzewodnikowym odpowiedniej bazy elementowej, głównie tranzystorów mocy o dopuszczalnych wysokich napięciach wstecznych, szybkich diod mocy i diod mocy Schottky'ego, kondensatorów o małej impedancji dla wyższych częstotliwości oraz rdzeni ferrytowych na transformatory wyjściowe o małych stratach dla wykorzystywanych częstotliwości przetwarzania (zwykle 20-50 kHz a nawet do 100 kHz). Transformator taki jest wielokrotnie mniejszy od transformatora sieciowego 50-hercowego pracującego w układzie zasilacza tradycyjnego.

Krótka analiza opłacalności, celowości budowy i zastosowania zasilaczy z przetwarzaniem

Porównując produkowane zasilacze z przetwarzaniem z zasilaczami ze stabilizacją szeregowo-liniową dostrzega się takie ich zalety:

- wysoka sprawność (65-75%),
- stosunkowo niska cena (o 10-30% tańsze),
- mały ciężar i gabaryty (ok. 50% mniejsze).

Dla zasilaczy średniej mocy korzyści wymiarowe przedstawiają się następująco:

- zmniejszenie objętości od ok. 100 cm^3/W (a nawet więcej) do 20 cm^3/W (a nawet mniej)
- zmniejszenie wymiarów liniowych około dwukrotnie
- zmniejszenie zajmowanej powierzchni około trzy do czterokrotnie.

Poniżej w tabeli 1 za artykułem "Power-supply choice looms large in sophisticated designs" (Electronics, October 14, 1976) przedstawiamy porównanie charakterystyk zasilaczy stabilizowanych zrealizowanych różnymi technikami.

Tabela 1

Parametr \ Rodzaj stabilizacji	Impulsowe z przetwarzaniem	Liniowe
Koszt wytwarzania	średni do kosztownego	umiarkowany
Koszt eksploatacji	mały	wyższy
Wymiary	najmniejsze	największe
Ciężar	najmniejszy	największy
Sprawność	znakomita	niezła
Stabilizacja	bardzo dobra	znakomita
Szumy (zakłócenia)	wymagają filtracji	nie przedstawiają problemu
Średni czas między uszkodzeniami	60 tys. h	60 tys. h + 100 tys. h
Naprawa	dość złożona	łatwa
Wpływ zmian częstotliwości sieci	nieczułe	nieczułe

W warunkach polskich produkcja zasilaczy przedstawia się następująco:

- w ZDEMP Politechniki Śląskiej znajdują się w produkcji doświadczalnej następujące zasilacze impulsowe (z przetwarzaniem):

  SPS-1B. 5.40.SC 5V/40A przewidziany do wdrożenia w 1982 r. w Zakładach MERA-ZAP

  SPS-1B. 9.25.SC 9V/25A

  SPS-1B.12.20.SC 12V/20A

  SPS-1B.15.15.SC 15V/15A

SPS-1B.24.10.SC 24V/10A

SPS-1B.48.5 .SC 48V/5 A

- Zakłady MERA-ZAP produkują następujące zasilacze:

EZP-02.00 5V/20A impulsowy z przetwarzaniem

EZS 5V/7A
EZS 9V/3A
EZS 12V/4A
EZS 15V/2A
EZS 24V/4A
} liniowe (szeregowe)

Jak widać z powyższego zestawienia brak jest ścisłego odpowiednika rozpatrywanego umownego źródła zasilania w rozwiązaniu liniowym (jest to bowiem rozwiązanie nietechnologiczne).

W celu uzyskania wymaganego prądu 40A należałoby więc zastosować 6 zasilaczy typu EZS 5V/7A, ale takie rozwiązanie oprócz trudności natury ekonomiczno-technicznej pociąga za sobą konieczność podziału odbiorników (układów zasilanych) na pola obciążeń, gdyż wymienionych zasilaczy nie można łączyć równolegle.

Niżej, w tabeli 2 podano zestawienie parametrów oraz wyniki analiz zebrane dla zasilaczy liniowych i impulsowych (z przetwarzaniem). Do analiz wykorzystano głównie materiały z Zakładów MERA-ZAP.

Tabela 2

Rodzaj stabilizacji		Zasilacze impulsowe (z przetwarzaniem)			Zasilacze liniowe z regul.szereg.	
TYP ZASILACZA / DANE TECHNICZNE i INNE		SPS-1B 5.40.SC 5V/40A ZDEMP-Pol.Śl.	EZP 02.00 5V/20A MERA-ZAP	ZIP 5V/40A model 1976 IMM	EZS 5V/7A MERA-ZAP	5V/40A składany z 6 x EZS5V/7A MERA-ZAP
Napięcie wyjściowe	V	5	5	5	5	5
Prąd wyjściowy	A	40	20	40	7	42
Moc wyjściowa	W	200	100	200	35	210
Sprawność	%	75	≥70	67	30	30
Wymiary	mm	123x160x223	121x171x167	179x191x178	165x174x127	127x165x (6x174)
Objętość	dcm^3	4,28	3,46	6,08	3,65	21,9
Gęstość mocy	cm^3/W	21	34	30	104	104
Ciężar	N	34	-	44	42	252
Cena zbytu	tys.zł	17,115	20-25	-	10	60
Wkład dewizowy z KK	$	22*	22*	22*		

* jest to obecna wysokość wkładu dewizowego; w funkcji czasu ceny elementów elektronicznych mają zawsze tendencje spadkowe

Zakończenie

Wymagania stawiane urządzeniom zasilającym do nowoczesnych urządzeń elektronicznych (wysokie parametry elektryczne, mechaniczne i eksploatacyjne, większe moce wyjściowe) mogą spełnić, na obecnym etapie rozwoju, jedynie zasilacze z przetwarzaniem.

Jakkolwiek możliwości budowy i zastosowań zasilaczy z przetwarzaniem są w pewnym stopniu ograniczone dostępnością na rynku krajowym głównie takich elementów, jak tranzystory mocy (wysoko-napięciowe) i diody mocy Schottky'ego, to trzeba podkreślić, że sprawa ta dotyczy zarówno zasilaczy liniowych jak i zasilaczy z przetwarzaniem. Z tym zastrzeżeniem, że tranzystory do zasilaczy liniowych są sprowadzane w ramach importu centralnego i są produkowane przez kraje socjalistyczne, natomiast wysokonapięciowe tranzystory mocy i diody mocy Schottky'ego pochodzą z krajów kapitalistycznych, ale chyba nie ulega wątpliwości, że produkcja tych elementów musi być podjęta przez kraje socjalistyczne.

Z przedstawionej analizy (oraz dostępnej literatury) wynika, że koszty wytwarzania zasilaczy z przetwarzaniem są niższe od kosztów wytwarzania zasilaczy liniowych. Poza tym należy podkreślić, że dodatkowe cechy, którymi charakteryzują się zasilacze z przetwarzaniem, należy postawić niejednokrotnie na pierwszym miejscu. Można do nich przede wszystkim zaliczyć (porównując różne techniki projektowania zasilaczy):

- najmniejsze koszty eksploatacji
- najmniejsze wymiary
- najmniejszy ciężar
- największą sprawność.

Reasumując należy stwierdzić, że podjęcie produkcji i stosowania zasilaczy z przetwarzaniem - w urządzeniach elektronicznych, a szczególnie w urządzeniach komputerowych - w warunkach krajowych, to tylko sprawa czasu. Zasilacze te wytyczyły bowiem kierunek nieodwracalnych zmian w dziedzinie źródeł zasilania.

Sprawozdania z konferencji

III Krajowa konferencja naukowo-techniczna
"Zastosowanie mikroprocesorów w automatyce i pomiarach"
Warszawa, 23 września 1982 r.

Oddział Warszawski Elektroniki i Telekomunikacji oraz Sekcja Automatyki i Pomiarów SEP organizują co dwa lata konferencje na temat zastosowania mikroprocesorów w automatyce i pomiarach. Trzecia konferencja zgromadziła w Warszawie 250 uczestników. W sesji plenarnej wygłoszono 7 referatów, a 43 referaty i komunikaty przedstawiono na posiedzeniu sekcji plakatowej.

Szczególne zainteresowanie wzbudziły referaty: Z.Mówka (Chemoautomatyka - Toruń) "Sterowniki mikroprocesorowe w automatyzacji procesów technologii chemicznej", J.Jakubiec i L.Zieleźnik IMEiE Politechnika Śląska "Programowany tester układów numerycznego sterowania obrabiarek", K.Fręczek, W.Wojsznis, K.Szulc (ELWRO)"Sterownik mikroprocesorowy ELWRO-80", M.Słodczyk, A.Syryczyński (PIAP) ""Jednopłytkowy pakiet mikrokomputera M-800 dla obsługi 16-bitowych sterowników i koncentratorów danych systemu MIR-PROWAY", A.Stańczuk, W.Stańczuk "Pakietowy system mikroprocesorowy MSP-80" i inne.

Sesji plakatowej towarzyszyła wystawa niektórych funkcjonujących urządzeń (m.in. sterownika z UNIMA) i sprzedaż materiałów informacyjnych przez jednostki UNITRA.

Konferencja stanowiła tradycyjny już przegląd prac prowadzonych obecnie w kraju w zakresie zastosowań mikroprocesorów w automatyce i pomiarach.

Materiały z konferencji można zakupić w Oddziale Warszawskim Elektroniki i Telekmunikacji SEP, 00-043 Warszawa, ul. Czackiego 3/5. Zainteresowani udziałem w następnej konferencji mogą zgłaszać swoje adresy do Sekcji Automatyki i Pomiarów przy ZG SEP, adres j.w.

doc.dr inż. Henryk ORŁOWSKI

Seminarium na temat:
"Zastosowanie sterownika mikroprocesorowego MIKRO-80 w gospodarce narodowej"
Poznań, 27 października 1982 r.

Ośrodek Badawczo-Rozwojowy Systemów Automatyki w Poznaniu zorganizował seminarium w celu prezentacji sterownika MIKRO-80 i jego zastosowań. Sterownik ten został opracowany w OBR SA i jest produkowany przez Zakłady Systemów Automatyki w Poznaniu. Jest on oparty na mikroprocesorze 8080 i produkowany w całości z elementów z I obszaru płatniczego. Prace prowadzono od r.1978; obecnie w seryjnej produkcji znajdują się pakiety (Eurokarta 1,5 x 233,4 x 160): procesora, pamięć RAM, ROM, pakiet przerwań i komunikacji dwukasetowej, wejściowy i wyjściowy dla sygnałów dwustanowych, transmisji szeregowej i pakiet pomocniczy do przygotowywania, uruchamiania i testowania oprogramowania użytkowego. W drugiej kolejności opracowano dalszych dziewięć pakietów, które są obecnie w jednostkowej produkcji. Dotychczas wyprodukowano ponad 40 sterowników, produkcja stale się rozwija a zapotrzebowanie rośnie.

Na seminarium przedstawiono także wiele zastosowań już zrelizowanych (np.sterowanie procesem elektrolizy w Hucie Konin) i będących w trakcie zaawansowanej realizacji. Zastosowania te dotyczyły: hutnictwa, magazynów wysokiego składowania, ruchu ulicznego, siłowni okrętowych i układów klimatyzacyjnych.

Na seminarium postanowiono założyć klub użytkowników sterowników MIKRO-80.

Materiały z seminarium, dane techniczne sterownika i jego oprogramowanie, cennik pakietów - można otrzymać w OBR SA, 61-807 Poznań, ul.Czerwonej Armii 66/72.

doc. dr inż. Henryk ORŁOWSKI

OFERTA

sprzedaży systemu zarządzania bazą danych SAD oraz systemu konwersacyjnego wyszukiwania KWINTET

Charakterystyka systemów SAD i KWINTET

System SAD stanowi narzędzie programowe dostępne dla użytkowników w różnych językach programowania (m.in. COBOL, PL/1, Assembler) i jest przeznaczony do prowadzenia niewielkich (ok. 50-80 tys. rekordów) baz danych. Struktura baz charakteryzuje się przystosowaniem do wielokryteriowego wyszukiwania i możliwością dowolnego wiązania rekordów (wiązanie wykonuje użytkownik).

System umżliwia wykonywanie wszystkich typowych operacji na danych, ponadto jest on uzupełniony zestawem programów usługowych wspomagających czynności administratora bazy oraz niektóre czynności użytkownika (np. wyszukiwanie, listowanie bazy).

Cechę charakterystyczną odróżniającą SAD od klasycznych modeli relacyjnych i hierarchicznych można nazwać "samoopisywaniem"; do opisu struktury zapisów w bazie służą inne zapisy, również umieszczone w bazie. Umożliwia to użytkownikowi dynamiczne wprowadzanie nowych struktur do bazy danych.

SAD działa na maszynach Jednolitego Systemu pod nadzorem systemu operacyjnego OS/JS oraz IBM 360 i IBM 370/145 pod nadzorem systemów operacyjnych OS MFT, MVT i VS1; wykorzystuje metody dostępu BDAM, BSAM i QSAM. System ma niewielkie wymagania w stosunku do sprzętu, np. do wykonyania przeciętnego programu potrzebuje 130K pamięci operacyjnej.

KWINTET jest systemem konwersacyjnym, przeznaczonym dla nieprogramistów i służącym do wyszukiwania informacji z baz danych organizowanych przez SAD. KWINTET ma następujące właściwości:

- obsługuje równocześnie wielu użytkowników (uprzednio rejestrowanych w systemie), z których każdy może współpracować z inną bazą danych,
- do komunikacji z systemem służy język KWINTET, sformalizowany ale bliski językowi naturalnemu, niezależny od zawartości bazy; istnieje polska i angielska wersja języka KWINTET,
- zlecenia w języku KWINTET wprowadza się przez klawiaturę monitora ekranowego, a wyniki otrzymuje na ekranie lub drukarce; użytkownik może określić format i uporządkowanie wyników,
- możliwe jest wykonywanie prostych obliczeń na wartościach wybranych z bazy,
- zlecenia sformułowane w języku KWINTET mogą być przechowywane w bibliotece i wielokrotnie wykonywane,
- istnieje aparat ułatwiający korzystanie z systemu niewprawnemu użytkownikowi.

System KWINTET działa na maszynach Jednolitego Systemu pod nadzorem systemu operacyjnego OS/JS oraz na maszynach IBM/360 i IBM 370/145 pod nadzorem systemu OS MVT i VS1. Urządzeniami terminalnymi do komunikacji z systemem są monitory MERA 7910 lub IBM 3277 i drukarki EC 7186 lub IBM 3284 podłączone lokalnie lub zdalnie. Do obsługi urządzeń wykorzystano metodę dostępu BTAM. Współpraca z systemem wymaga minimum 200K pamięci operacyjnej.

Systemy SAD i KWINTET są eksploatowane obecnie w Instytucie Maszyn Matematycznych oraz w kilku innych ośrodkach w Polsce.

Dostarczana dokumentacja

Nabywcy dostarcza się dokumentację użytkową, która obejmuje:

- dla systemu SAD
  - ogólny opis systemu zarządzania bazą danych SAD
  - opis zastosowań
  - podręcznik programisty
  - podręcznik programisty systemowego
- dla systemu KWINTET
  - opis zastosowań
  - opis języka
  - podręcznik programisty systemowego
  - podręcznik operatora

Szkolenie użytkowników

W ramach podanej ceny dostawca zapewnia przeszkolenie:

- dla systemu SAD
  - 5 programistów nabywcy
- dla systemu KWINTET
  - 5 użytkowników systemu
  - 1 programisty systemowego
  - 1 operatora

Koszty pobytu tych osób w Warszawie pokrywa nabywca systemu.

Konserwacja

W ramach podanej ceny dostawca zapewnia roczną konserwację systemu, tj. usuwanie wykrytych błędów i odpowiednią korektę dokumentacji.

W celu lokalizacji błędów i instalacji poprawionego systemu nabywca udostępni nieodpłatnie swoją maszynę.

Ceny

Cena sprzedaży systemu, obejmująca wszystkie wymienione świadczenia wynosi:

- systemu SAD -
- systemu KWINTET -

Za każdy następny rok konserwacji systemu opłata wynosi:

- systemu SAD -
- systemu KWINTET -

OFERTA

Kompilator języka PASCAL DOC PB

Krótka charakterystyka

Język PASCAL został opracowany pod koniec lat sześćdziesiątych w Eidgenössische Technische Hochschule w Zurychu przez Niklausa Wirtha; podstawą opracowania był język ALGOL 60.

PASCAL jest uniwersalnym językiem algorytmicznym wysokiego poziomu o zagnieżdżonej strukturze proceduralnej. Zasadnicze różnice w stosunku do ALGOL-u 60 dotyczą strukturowania danych. Wprowadzenie w PASCAL-u różnorodnych typów - proste, wyliczeniowe, okrojone, tablice, rekordy, pliki, zbiory (w sensie teorii mnogości), wskaźnikowe - pozwala na rozwiązywanie problemów z zakresu przetwarzania danych i wprowadzenie nieregularnych struktur danych typu listy.

Zasady budowy i forma wyrażeń są analogiczne jak w języku ALGOL 60, natomiast zasady konstruowania innych elementów języka, zwłaszcza deklaracji pozwalają w sposób naturalny i wygodny korzystać z dodatkowych udogodnień języka.

Język PASCAL stanowi podstawę nowoczesnego programowania w systematyczny sposób przy wykorzystaniu małej liczby pojęć podstawowych. Jest to możliwe ponieważ:

- struktura i konstrukcja języka nakłaniają do dobrego stylu programowania
- klasa algorytmów dających się zapisać w języku w sposób naturalny jest bardzo szeroka
- opis języka jest prosty a zarazem precyzyjny.

Powyższe zalety spowodowały, że PASCAL pomyślany początkowo jako język dydaktyczny do nauki programowania, w stosunkowo szybkim czasie zdobył sobie dużą popularność jako język użytkowy i publikacyjny. Efektywne implementacje na bardzo wielu maszynach (m.in. CDC, IBM, JS EMC, SM EMC, UNIVAC, Honeywell) umożliwiają użytkownikom szeroką wymianę progamów między ośrodkami.

Język PASCAL DOC PB zachowuje wszystkie zalety języka standardowego (opracowanego przez Wirtha) i jednocześnie wyposażony jest w liczne rozszerzenia pozwalające na pełne wykorzystanie specyficznych możliwości systemu operacyjnego DOC PB (RSX-11M). Do najważniejszych rozszerzeń należą:

- możliwość konwersacyjnego działania programu
- możliwość korzystania z prawie wszystkich rodzajów plików używanych w DOC PB oraz RSX-11M, a w szczególności z plików o dostępie bezpośrednim
- możliwość korzystania z procedur zewnętrznych
- możliwość kompilowania procedur bez programu głównego, co pozwala na tworzenie prywatnych bibliotek użytkownika zawierających procedury i funkcje pisane w PASCAL-u
- możliwość łatwego łączenia procedur pisanych w PASCAL-u z procedurami pisanymi w makro-

assemblerze lub FORTRAN-ie

- możliwość tworzenia struktury nakładkowej programu, co pozwala na pisanie i wykonywanie dużych programów
- możliwość korzystania z dynamicznych tablic jako parametrów procedur
- możliwość korzystania z pełnego zbioru znaków reprezentowanych w systemie
- możliwość komunikacji z systemem za pomocą procedur standardowych
- możliwość śledzenia działania programu
- możliwość konwersacyjnego oczyszczania programu w czasie jego wykonywania za pomocą specjalnych procedur (DEBUG) dostarczanych razem z kompilatorem
- możliwość tworzenia profili dynamicznych programu, co ułatwia optymalizację
- możliwość dopasowania kodu wynikowego programu do sprzętowej charakterystyki jednostki arytmetycznej.

Kompilator PAS akceptujący język PASCAL DOC PB napisany jest w PASCAL-u z małymi fragmentami kodowanymi w makroassemblerze. W czasie instalowania kompilatora istnieje możliwość jego modyfikacji tak, aby był on maksymalnie dopasowany do sprzętowej charakterystyki zestawu, jak również do wymagań użytkowników pracujących na tym zestawie. Kompilator korzysta z dostarczanej razem z nim systemowej biblioteki dla programów napisanych w języku PASCAL DOC PB o nazwie PASLIB, która zawiera procedury i funkcje obsługi programu w czasie jego wykonywania.

Przedmiot dostawy

Przedmiotem dostawy jest nośnik zawierający postać dystrybucyjną kompilatora oraz dokumentacja. Dokumentacja kompilatora języka PASCAL DOC PB obejmuje następujące podręczniki:

- PASCAL DOC PB - Podręcznik instalacji
- PASCAL DOC PB - Opis języka
- PASCAL DOC PB - Podręcznik użytkownika.

Uwaga: Nośnik (kaseta dyskowa typu EC5269-01 lub taśma magnetyczna na małym krążku) powinien być dostarczany przez odbiorcę. Nagrywanie odbywać się będzie w Instytucie Maszyn Matematycznych na urządzeniach: CM5300.01 lub CM5400.

Wymagana konfiguracja EMC

Kompilator języka PASCAL DOC PB przystosowany jest do pracy w mapowanym systemie DOC PB lub RSX-11M. Do zainstalowania i działania kompilatora wymagana jest konfiguracja EMC (SM4 lub PDP11), na której działa wymieniony system operacyjny, zawierająca:

- co najmniej jedną jednostkę pamięci dyskowej
- partycję 32K pamięci.

Termin dostawy i proponowana cena sprzedaży

W ciągu 2 miesięcy od daty podpisania umowy. Proponowana cena 143 tys. zł.

OFERTA

Karty opisów patentowych

wg zarządzenia nr 20 Przewodniczącego KNiT z dn.20 maja 1971 r.

Branżowy Ośrodek Informacji Naukowej Technicznej i Ekonomicznej Instytutu Maszyn Matematycznych wydaje karty dokumentacyjne opisów patentowych w klasach MKP G06 i G11 Międzynarodowej Klasyfikacji Patentowej. W ich skład wchodzą następujące podklasy:

- G06 C Mechaniczne cyfrowe maszyny matematyczne
- G06 D Cyfrowe maszyny liczące przepływowo-ciśnieniowe
- G06 F Cyfrowe maszyny matematyczne, w których przynajmniej część przeprowadzonych obliczeń wykonywana jest elektrycznie; urządzenia do przekazywania danych cyfrowych
- G06 G Analogowe maszyny matematyczne
- G06 J Hybrydowe układy liczące
- G06 K Rozpoznawanie danych; przedstawienie danych; nośniki zapisu; manipulacja nośnikami zapisu
- G06 M Mechanizmy liczące; liczenie przedmiotów nie ujęte gdzie indziej
- G11 B Zapis informacji z wykorzystaniem ruchu względem występującego między nośnikiem zapisu i przetwornikiem
- G11 C Zapis informacji bez wykorzystania ruchu względnego występującego między nośnikiem zapisu a przetwornikiem
- G11 D Przesyłanie danych cyfrowych między pamięciami o ruchu względnym między nośnikiem zapisu a przetwornikiem oraz pamięciami bez ruchu względnego

Wszystkim zainteresowanym instytucjom i osobom prywatnym proponujemy karty opisów patentowych patentów udzielonych we Francji, RFN, USA i W.Brytanii; są one jednym z najszybszych nośników informacji w zakresie danej tematyki.

Koszt jednej karty formatu A6 na kartonie, druk dwustronny wynosi 20 zł + koszty przesyłki.

Zgłoszenia prosimy składać według wymienionych podklas MKP do dnia 15 stycznia 1983 r.

Instytut Maszyn Matematycznych
Branżowy Ośrodek Informacji Naukowej
Technicznej i Ekonomicznej
ul. Krzywickiego 34, 02-078 Warszawa

„Tłumacz-tekst" u/612:500 A4

Cena zł 280.-

WARUNKI PRENUMERATY

Prenumeratę na kraj przyjmują Oddziały RSW "Prasa-Książka-Ruch" oraz urzędy pocztowe i doręczyciele w terminie do dnia 25 listopada na rok następny.

Cena prenumeraty rocznej zł 840.

Jednostki gospodarki uspołecznionej, instytucje, organizacje i wszelkiego rodzaju zakłady pracy zamawiają prenumeratę w miejscowych Oddziałach RSW "Prasa-Książka-Ruch", w miejscowościach zaś, w których nie ma Oddziałów RSW - w urzędach pocztowych.

Czytelnicy indywidualni opłacają prenumeratę wyłącznie w urzędach pocztowych i u doręczycieli.

Prenumeratę ze zleceniem wysyłki za granicę przyjmuje RSW "Prasa-Książka-Ruch", Centrala Kolportażu Prasy i Wydawnictw, ul. Towarowa 28, 00-958 Warszawa, konto PKO Nr 1153-201045.

Prenumerata ze zleceniem wysyłki za granicę jest droższa od prenumeraty krajowej o 50% dla zleceniodawców indywidualnych i o 100% dla zlecających instytucji i zakładów pracy.

INDEKS-35362