BIULETYN TECHNICZNO-INFORMACYJNY

DOM

PL ISSN 0239-6645
Nr ind. 35309

1987
2(296)

BIULETYN TECHNICZNO-INFORMACYJNY

SPIS TREŚCI

WYDAWCA: Zrzeszenie Producentów Środków Informatyki, Automatyki i Aparatury Pomiarowej „MERA"

KOLEGIUM REDAKCYJNE: mgr A. Chróścielewska, dr inż. J. Dyczkowski (redaktor naczelny), mgr J. Kutrowska (sekretarz redakcji)
RADA PROGRAMOWA: inż. J. Bartak, inż. D. Łochocki, mgr S. Majchrzak mgr inż. A. Musielak, inż. H. Oleksy, mgr inż. H. Pilko, dr inż. B. Piwowar, dr hab. inż. K. Urbaniec
Opracowanie: Redakcja Biuletynu Techniczno-Informacyjnego „Mera" przy Ośrodku Badawczo-Wdrożeniowym „Mercomp" ul. Poezji 19, 04-994 Warszawa tel. 12-90-11 w. 17-54
Druk: Przedsiębiorstwo Automatyki Przemysłowej „Mera-Pnefal", ul. Poezji 19, 04-994 Warszawa. Zam. 92/87. Nakład 1560 egz.
Warunki prenumeraty: 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 roczną w cenie 3900 zł należy zamawiać do 25 listopada na rok następny, półroczną do 10 czerwca na II półrocze (1950 zł).

mgr inż. ADAM CAŁY
dr inż. ZBIGNIEW CZECH
mgr inż. MAREK KONOPKA
Instytut Informatyki
Politechnika Śląska

KOMPILATOR JĘZYKA "ADA" DLA MIKROKOMPUTERA COMPAN-8

Inicjatorem i sponsorem projektu języka Ada, ocenianego przez niektórych informatyków [5] jako najambitniejszy projekt języka programowania kiedykolwiek podjęty, jest Departament Obrony Stanów Zjednoczonych. Mimo że język z założenia miał być stosowany głównie do programowania tzw. komputerowych systemów wbudowanych /ang. embedded computer systems/ niestety, przede wszystkim w różnego rodzaju jednostkach broni/, to zainteresowanie nim przejawiają również środowiska przemysłowe, pragnące zastosować go w systemach sterowania w czasie rzeczywistym oraz środowiska związane z telekomunikacją, mające na uwadze oprogramowanie systemów teletransmisji.

Istnieje pogląd, że Ada będąca podsumowaniem wyników badań teoretycznych oraz doświadczeń praktycznych, uzyskanych w dziedzinie programowania komputerów w latach siedemdziesiątych, będzie podstawowym językiem w systemach profesjonalnych w końcu lat 80 i 90 bieżącego stulecia. Stanowisko takie uzasadniają nowoczesne cechy języka, takie jak:

- Bogate typy danych, zaczerpnięte głównie z języka Pascal, pozwalające budować czytelne programy dla problemów z różnych dziedzin zastosowań: dużą czytelność uzyskuje się przez stosowanie pojęć /typów danych/ właściwych danej dziedzinie zastosowań.

- Ścisła kontrola typów danych /ang. strong typing/, zwiększająca niezawodność programów. Uzyskanie wysokiej niezawodności programów, tj. małego prawdopodobieństwa pozostawienia w programach niewykrytych błędów, było jednym z głównych celów stawianych przed językiem Ada. Cel ten został osiągnięty przez zdefiniowanie możliwie prostych i naturalnych konstrukcji językowych, zmniejszających ryzyko błędnego ich użycia jak również przez wprowadzenie mechanizmów, umożliwiających dogłębną kontrolę poprawności programu w czasie jego kompilacji: jednym z takich mechanizmów jest właśnie ścisła kontrola danych.
- Możliwości przeciążania /ang. overloading/ predefiniowanych operatorów Ady i podprogramów, jak też bardziej elastyczne niż w Pascalu reguły widzialności /ang. visibility rules/, zwiększające swobodę programisty przy konstruowaniu programów.
- Pakiety /ang. packages/, jednostki rodzajowe /ang. generics/ oraz możliwości oddzielnej kompilacji /ang. separate compilation/ ułatwiają lub wręcz umożliwiają budowanie dużych programów składanych z modułów przez liczne, często rozproszone geograficznie, zespoły programistów. Jak tego dowiodły badania w dziedzinie inżynierii programowania, efektywna modularyzacja jest niezbędna w przypadku, gdy mamy do czynienia z programami o dużej złożoności /np. o rozmiarach 50.000 czy 100.000 wierszy/ oraz cyklu życia, w którym powinny być one pielęgnowane /z reguły nie przez ich autorów/, wynoszącym 10-15 lat. Wiadomo, że opracowanie efektywnych i elastycznych metod modularyzacji dałoby realne perspektywy stworzenia w przyszłości "przemysłu" produkującego moduły programowe /komponenty/, które kupowane następnie "z półek" sklepów z oprogramowaniem służyłyby do budowania systemów, odpowiadających naszym celom: takie pojęcia Ady jak pakiet, jednostka rodzajowa, czy oddzielna kompilacja są z pewnością osiągnięciem przybliżającym ten cel.

- Programowanie współbieżne poprzez stosowanie zadań /ang. task/ wykonywanych równocześnie z innymi zadaniami, z możliwością ich wzajemnej komunikacji i synchronizacji. Środki programowe w tym zakresie są niezbędne dla współczesnych systemów cechujących się coraz większym rozproszeniem jego podsystemów /opartych na mikroprocesorach/ i coraz większą współbieżnością działania.
- Mechanizm obsługi wyjątków /ang. exception handling/, pozwalający definiować w czasie tworzenia programu reakcje na błędy, które zostaną wykryte podczas jego wykonywania. Obsługa wyjątków daje realne możliwości konstruowania systemów programowych, które uniknęłyby upadku dzięki natychmiastowemu wychwytywaniu pojawiających się błędów zarówno o charakterze urządzeniowym, jak też programowym.

Systemy takie, potrafiące naprawiać zaistniałe błędy lub w sposób kontrolowany ograniczać spełniane przez nie funkcje, określane są jako systemy tolerujące błędy /ang. faulttolerant software systems/.

- Możliwości "niskiego poziomu" /ang. low-level features/, pozwalające programiście precyzować szczegóły reprezentowania danych w komputerze docelowym /ang. target computer/; można wpływać również na adresację obiektów, jak również wlaczać do programu napisanego w języku Ada fragmenty napisane w asemblerze komputera docelowego.

Język Ada, podobnie jak wszystkie języki programowania ma również swoje słabe strony. Krytykowana jest np. duża ilość słów zastrzeżonych w języku, stanowiąca ok. 10% podstawowego słownika angielskiego, tzw. Basic English. Uważa się, że przeciążanie operatorów jest sprzeczne z wymaganiami ścisłej kontroli typów. Wskazuje się na wiele możliwości usprawnienia mechanizmów współbieżności zawartych w języku. Dobitnie wyraził swoją ocenę Ady C. A. R. Hoare w wykładzie nagrodzonym przez Towarzystwo ACM nagroda Turinga, stwierdzając: "Nie można dopuścić, aby ten język w obecnej postaci był używany w zastosowaniach, których cechą krytyczna jest niezawodność" [4].

Celem niniejszego artykułu jest przedstawienie podstawowych cech kompilatora języka Ada dla mikrokomputera ComPAN-8. Poniżej omówiony zostanie podzbiór języka Ada, który może być kompilowany za pomoca kompilatora. Dwa ostatnie rozdziały poświęcono przedstawieniu charakterystyki kompilatora oraz sposobu jego użytkowania.

Język Ada dla mikrokomputera ComPAN-8

W niniejszym rozdziale przedstawiony zostanie język Ada dla mikrokomputera ComPAN-8. Język ten jest podzbiorem standardu Ady zdefiniowanym w pracy C. A. R. Hoare [4].

W artykule opisane zostaną tylko te elementy języka, które są akceptowane przez dostępny kompilator. Tak więc, jeśli jakaś konstrukcja języka nie będzie tu omówiona, należy przyjąć, że nie można jej w tej wersji Ady stosować.

Jednostki leksykalne

W języku Ada wyróżnia się sześć typów jednostek leksykalnych: komentarz, identyfikator /który może być słowem zastrzeżonym/, literał numeryczny, literał znakowy, literał napisowy i ogranicznik.

Identyfikator w języku Ada, to ciag liter, cyfr i znaków podkreśleń. Pierwszym znakiem identyfikatora musi być litera, a ostatnim - litera lub cyfra. W wersji języka Ada na mikrokomputerze ComPAN-8, pierwszych dziesięć znaków jest znaczących.

W standardzie języka Ada wyróżnia się dwa rodzaje literałów numerycznych: literały dziesiętne i literały z podstawa. Ten drugi rodzaj literałów nie został zaimplementowany w wersji języka Ada na mikrokomputerze ComPAN-8.

Literał znakowy, to znak graficzny ujęty w znaki apostrofu. Znak graficzny to znak "widoczny" /np. litery, cyfry, znaki specjalne/.

Literał napisowy, to ciag znaków graficznych ujęty w znaki cudzysłowu.

Specjalną konstrukcją języka Ada jest pragma. Służy ona do przekazywania kompilatorowi informacji, dotyczacych tłumaczenia programu.

W wersji języka Ada na mikrokomputerze ComPAN-8 można stosować następujące pragmy:

LIST/ON/ -- Drukuj komunikaty o błędach na urządzeniu LST:
LIST/OFF/ -- Drukuj komunikaty o błędach na urządzeniu CON:
OPTIMIZE/TIME/ -- Optymalizuj czas działania programu.
OPTIMIZE/SPACE/ -- Optymalizuj rozmiar programu.
PRINT/SOURCE/ -- Drukuj kod źródłowy.
PRINT/A_CODE/ -- Drukuj kod pośredni.
PRINT/CODE/ -- Drukuj kod wynikowy.
SYSTEM/I8080/ -- Wybór procesora docelowego.
SYSTEM/I8085/ -- Wybór procesora docelowego.
SYSTEM/Z80/ -- Wybór procesora docelowego
-- /Dwie ostatnie pragmy nie
-- zostały zaimplementowane/.

Pragmy mogą wystąpić na początku jednostki kompilacji.

Deklaracje

W języku Ada zdefiniowanych jest kilka rodzajów rzeczy /ang. entities/, które mogą być deklarowane jawnie lub niejawnie za pomocą deklaracji. Należą do nich:

- literał numeryczny,
- obiekt,
- składowa rekordu,
- parametr instrukcji iteracji,
- typ,
- podtyp,
- podprogram /procedura lub funkcja/,
- parametr formalny podprogramu,
- nazwany blok lub instrukcja iteracji,
- operacja /w szczególności atrybut lub literał wyliczeniowy/.

Istnieje kilka rodzajów deklaracji. Wyróżnia się wśród nich deklaracje podstawową, za pomocą której deklaruje się: obiekt, typ, podtyp i podprogram. Pewne rodzaje deklaracji zawsze występują /jawnie/, jako części deklaracji podstawowych. Są to deklaracje składowej rekordu, parametru formalnego podprogramu i literału wyliczeniowego. Specyfikacja parametru instrukcji iteracji jest formą deklaracji, która występuje tylko w pewnych rodzajach instrukcji iteracji. Pozostałe formy deklaracji są niejawne: nazwa bloku i nazwa instrukcji ite-

racji są deklarowane niejawnie. W ten sam sposób są również deklarowane pewne operacje.

Typ charakteryzowany jest przez zbiór wartości oraz zbiór operacji, które mogą być wykonywane na tych wartościach. Wyróżnia się dwie klasy typów: skalarne i złożone. Do typów skalarnych zaliczamy: typy całkowite, typy rzeczywiste oraz typy definiowane przez wyliczenie ich wartości. Wartości typów skalarnych nie zawierają składowych /komponentów/. Typ tablicowy i rekordowy są typami złożonymi. Wartość typu złożonego składa się z wartości składowych /ang. component values/.
Na zbiór możliwych wartości danego typu może być nałożony warunek zwany zawężeniem /uwzględniany jest również przypadek, gdy zawężenie nie nakłada ograniczeń/. Należy dodać, że wartość spełnia zawężenie, jeżeli spełnia odpowiadający mu warunek.

Przez podtyp rozumie się typ wraz z zawężeniem. Mówimy, że wartość należy do /zbioru wartości/ podtypu danego typu, jeśli należy do /zbioru wartości/ tego typu i spełnia zawężenie. Typ ten określany jest mianem typu bazowego podtypu.

Do typów skalarnych należą: typy wyliczeniowe, typy całkowite oraz typy rzeczywiste. Typy wyliczeniowe i typy całkowite są typami dyskretnymi. Dla każdej wartości X typu lub podtypu dyskretnego T określony jest atrybut T'POS/X/, wyznaczający numer pozycji wartości X będący liczbą całkowitą. Atrybut T'POS/X/ jest funkcją. Typy całkowite i typy rzeczywiste są typami numerycznymi. Wartości wszystkich typów skalarnych są uporządkowane, można więc dla nich stosować operatory relacji.

W deklaracji typu wyliczeniowego, identyfikatory oznaczające literały wyliczeniowe muszą być różne. Literałem wyliczeniowym nie może być literał znakowy /w standardzie języka Ada jest to dopuszczalne/. Numerem pozycji pierwszego literału wyliczeniowego na liście deklaracji jest zero. Numer pozycji każdego innego literału wyliczeniowego jest o jeden większy od jego poprzednika na liście.

W przypadku, gdy ten sam identyfikator literału wyliczeniowego występuje w więcej niż jednej deklaracji typu wyliczeniowego mówimy, że odpowiednie literały są przeciążone /ang. oberloaded/. W każdym miejscu programu, w którym występuje przeciążony literał, jego typ musi być określony na podstawie kontekstu.

Dla każdego typu lub podtypu wyliczeniowego T zdefiniowane zostały następujące atrybuty:
T'FIRST Wyznacza pierwszą wartość typu lub podtypu T; wartość ta jest tego samego typu co T.
T'LAST Wyznacza ostatnią wartość typu lub podtypu T; wartość ta jest tego samego typu co T.
T'VAL/N/ Wyznacza wartość typu lub podtypu T o numerze pozycji N. Wartość N powinna być całkowita i leżeć w zakresie T'POS/T'BASE'FIRST/... T'POS/T'BASE'LAST/.
T'SUCC/X/ Wyznacza wartość, której numer pozycji jest o jeden większy od numeru pozycji wartości X. Wartość X musi być różna od T'BASE'LAST.
T'PRED/X/ Wyznacza wartość, której numer pozycji jest o jeden mniejszy od numeru pozycji wartości X. Wartość X musi być różna od T'BASE'FIRST.

Do zbioru wartości predefiniowanego typu znakowego CHARACTER należy 128 wartości znakowych kodu ASCII. Każdy z 95 znaków tego kodu, mający swój odpowiednik graficzny, może być oznaczony odpowiednim literałem znakowym. Przy korzystaniu z pozostałych znaków kodu ASCII /np. ze znaków sterujących/ pomocny jest atrybut CHARACTER'VAL/N/, gdzie N jest numerem znaku /od 0 do 127/. Wartość typu znakowego zajmuje 1 bajt w pamięci mikrokomputera.

Istnieje predefiniowany typ wyliczeniowy o nazwie BOOLEAN. Zawiera on dwa literały wyliczeniowe: FALSE i TRUE, uporządkowane zgodnie z relacją FALSE < TRUE. Typem boolowskim jest typ BOOLRAN lub dowolny typ pochodny od typu boolowskiego. Wartość typu boolowskiego zajmuje 1 bajt w pamięci mikrokomputera.

Istnieje predefiniowany typ całkowity INTEGER. Do zbioru wartości tego typu należą liczby całkowite z zakresu -32768 .. 32767. Wartość typu całkowitego zajmuje 2 bajty w pamięci mikrokomputera. Możliwe jest definiowanie nowych typów całkowitych, pochodzących od typu INTEGER. Służy do tego deklaracja mająca następującą postać:

```
type T is new INTEGER range L .. P;
```

lub jej skrócona forma:

```
type T is range L .. P;
```

Ostatnia deklaracja jest równoważna następującym deklaracjom:

```
type integer_type is new INTEGER;
subtype T is integer_type range integer_type/L/.
                                integer_type/P/;
```

gdzie integer_type jest typem anonimowym.

Granice L i P muszą być wyrażeniami statycznymi typu całkowitego. Dla każdego typu lub podtypu całkowitego T zdefiniowany jest atrybut T'POS/X/. Istnieje predefiniowany podtyp całkowity NATURAL mający definicję:

```
subtype NATURAL is INTEGER range 0 .. 32767
```

Typy rzeczywiste obejmują tylko typy zmiennopozycyjne /ang. floating point types/. Nie

istnieją typy stałopozycyjne /ang. fixed point types/. występujące w standardzie języka Ada.

Istnieje predefiniowany typ zmiennopozycyjny FLOAT. Do zbioru wartości tego typu należą liczby rzeczywiste z zakresu - 0.3402818E39 .. 0.3402818E39. Maksymalna liczba cyfr znaczących liczb rzeczywistych wynosi 7. Wartość typu rzeczywistego, zmiennopozycyjnego zajmuje 4 bajty w pamięci mikrokomputera. Możliwe jest definiowanie nowych typów zmiennopozycyjnych, pochodzących od typu FLOAT. Definicje nowych typów nie mogą jednak nakładać ograniczeń na zakres wartości oraz liczbę cyfr znaczących.

Do typów złożonych należą typy tablicowe i typy rekordowe. Istnieje predefiniowany typ tablicowy STRING. Obiekt typu tablicowego jest obiektem złożonym, składającym się ze składowych /komponentów/ tego samego podtypu. Nazwa składowej obiektu typu tablicowego zawiera jedną lub więcej wartości indeksowych typu dyskretnego.

Definicja niezawężonej tablicy /ang. unconstrained array definition/ definiuje typ tablicowy. Dla każdego obiektu tego typu liczba indeksów, typ i pozycja każdego indeksu, oraz podtyp komponentów są takie, jak w definicji typu: wartości dolnych i górnych granic każdego indeksu należą do zbioru wartości podtypu indeksu. Podtyp indeksu na danej pozycji jest określony wskaźnikiem typu, występującym na tej pozycji w definicji typu tablicowego. Separator <>, nazywany skrzynką /ang. box/, oznacza niezdefiniowany zakres wartości indeksów.

Deklarując obiekty za pomocą deklaracji typu z definicją, niezawężonej tablicy, należy użyć zawężeń indeksów /ang. index constraints/, wyznaczających zakresy możliwych wartości dla każdego indeksu.

Dla każdego obiektu A typu tablicowego zdefiniowane są następujące atrybuty:

A'FIRST Wyznacza dolną granicę zakresu wartości pierwszego indeksu.
A'FIRST/N/ Wyznacza dolną granicę zakresu wartości N - tego indeksu.
A'LAST Wyznacza górną granicę zakresu wartości pierwszego indeksu.
A'LAST/N/ Wyznacza górną granicę zakresu wartości N - tego indeksu.
A'LENGTH Wyznacza liczbę wartości zakresu pierwszego indeksu.
A'LENGTH/N/ Wyznacza liczbę wartości zakresu N - tego indeksu.

Wartościami predefiniowanego typu STRING są jednowymiarowe tablice, których składowe są typu CHARACTER. Składowe indeksowane są wartościami typu INTEGER:

Obiekt typu rekordowego jest obiektem złożonym, składającym się z nazwanych składowych /komponentów/. Deklaracja typu rekordowego zawiera deklaracje poszczególnych składowych. Identyfikatory wszystkich składowych w deklaracji typu rekordowego muszą być różne. Deklaracja składowej z kilkoma identyfikatorami jest równoważna sekwencji deklaracji z poszczególnymi, kolejnymi identyfikatorami.

U w a g a : Deklaracja typu rekordowego nie może zawierać specyfikacji dyskryminanty oraz wariantów, co jest dopuszczalne w standardzie Ady.

Deklaracje zawarte w programach napisanych w języku Ada dla mikrokomputera ComPAN-8 powinny występować w następującej kolejności:
1. Deklaracje typów, podtypów i obiektów /w dowolnym porządku/.
2. Specyfikacje adresów obiektów.
3. Deklaracje podprogramów.

W języku Ada zdefiniowano sześć klas operatorów. Poniżej podano je według rosnącego priorytetu.
operator_logiczny ::= and | or | xor
operator_relacyjny ::= = | /= | < | <= | > | >=
operator_binarny :: = + | - | &
operator_unarny :: = + | -
operator_multiplikatywny :: = * | / | mod | rem
operator_o_najwyższym priorytecie

::= ** | abs | not

W języku Ada można dokonywać jawnej konwersji typu.
konwersja_typu ::= nazwa_typu/wyrażenie/

W języku Ada dla mikrokomputera ComPAN-8 dopuszczalne są konwersje z typu całkowitego na rzeczywisty, zmiennopozycyjny oraz z rzeczywistego, zmiennopozycyjnego na całkowity. W przypadku przeciążenia literałów w typach wyliczeniowych, jeśli nie można jednoznacznie określić typu literału, należy podać jego typ za pomocą wyrażenia kwalifikowanego.

Instrukcje

Instrukcja może być prosta lub złożona. Instrukcja prosta nie zawiera w swym wnętrzu innych instrukcji. Instrukcja złożona może zawierać w sobie inne instrukcje proste lub złożone.
instrukcja ::= instrukcja_prosta | instrukcja_złożona
instrukcja_prosta ::= instrukcja_pusta
| instrukcja_przypisana
| instrukcja_wyjścia
| instrukcja_wywołania_procedury
| instrukcja_powrotu

```
instrukcja złożona ::= instrukcja_warunkowa
                     | instrukcja_wyboru
                     | instrukcja_iteracji
                     | instrukcja_bloku
instrukcja_pusta ::= null;
```

Instrukcje nie mogą być poprzedzone etykietą, gdyż w przedstawionej wersji kompilatora nie zaimplementowano instrukcji goto. Wykonanie instrukcji pustej nie powoduje realizacji żadnych operacji. Instrukcje wykonywane są zawsze w kolejności ich zapisania do chwili przeniesienia sterowania do innego miejsca.

Zmianę sterowania powoduja instrukcje wyjścia, powrotu i wywołania procedury.

Można definiować wiele podprogramów o tej samej nazwie. W trakcie kompilacji rozstrzyga się o wyborze wywołania odpowiedniego podprogramu na podstawie kontekstu wywołania. Możliwość taka nazywa się przeciążeniem podprogramu. Obowiązują następujące reguły i ograniczenia, dotyczące przeciążania podprogramów. Nie wolno definiować podprogramów o nazwach nie będących identyfikatorami. Nie można więc przeciążać predefiniowanych operatorów. Nazwy wszystkich podprogramów powinny różnić się od nazw obiektów; jeśli tak nie jest, to podprogramy nie są widoczne i nie można ich wykonać.

Przeciążane podprogramy powinny różnić się między sobą rodzajem /procedura lub funkcja/ albo listą parametrów. Jeśli definiujemy funkcje i procedury o tej samej nazwie, to definicje funkcji powinny poprzedzać definicje procedur, gdyż w przeciwnym razie funkcje nie będą dostępne. W przypadku zdefiniowania kilku podprogramów tego samego rodzaju, o takich samych listach parametrów, widoczny będzie tylko pierwszy podprogram. Nie można przeciążyć funkcji wyłącznie pod względem typu zwracanej przez nią wartości.

Jednostką kompilacji jest pojedynczy podprogram /procedura lub funkcja/ bez parametrów formalnych. Struktura jednostki kompilacji jest taka sama jak struktura każdego podprogramu.

Wejście-wyjście w języku Ada

W języku Ada dla mikrokomputera ComPAN-8 możliwe jest wykonywanie tekstowych operacji wejścia-wyjścia /ang. TEXT_IO/ oraz operacji wejścia-wyjścia na sekwencyjnych plikach znakowych /ang. SEQUENTIAL_IO/. Istnieją trzy procedury, służące do realizacji tekstowych operacji wejścia-wyjścia o nazwach: NEW_LINE, PUT, GET.

Procedury wejścia GET oraz wyjścia PUT są jednoparametrowe i przeznaczone, odpowiednio, do wprowadzania i wyprowadzania tekstów. Parametry aktualne w wywołaniach procedur mogą być wyrażeniami typu

- całkowitego,
- rzeczywistego, zmiennopozycyjnego,
- boolowskiego,
- znakowego,
- STRING.

Do przeprowadzania operacji na sekwencyjnych plikach znakowych przeznaczone są procedury i funkcje o nazwach: OPEN, IS_OPEN, CLOSE, CREATE, DELETE, READ, WRITE, i NAME. Ponadto istnieje funkcja END_OF_FILE, która jest niezbędna do przetwarzania plików dowolnego typu, np. z rozszerzeniem .COM. Wszystkie wymienione procedury i funkcje realizują swe działanie za pośrednictwem systemu BDOS /ang. Basic Disk Operating System/.

W programach napisanych w języku Ada dla mikrokomputera ComPAN-8 możliwe jest wywoływanie procedur systemowych, przeznaczonych do obsługi urządzeń zewnętrznych i jednostek pamięci dyskowej. Procedury te wchodzą w skład systemu operacyjnego CP/M. Szczegółowy opis procedur zawiera opracowanie: "Dokumentacja użytkowa dyskowego systemu operacyjnego CP/M 2.2". /9, pkt. 2/.

Wywołanie procedury ma postać:

BDOS/nr/

lub

BDOS/nr, p/

gdzie parametr:

nr - określa numer procedury, jaką należy wykonać, a p - stanowi daną wejściową niezbędną do wykonania procedury.

Charakterystyka kompilatora

Kompilator języka Ada dla mikrokomputera ComPAN-8 jest kompilatorem jednoprzejściowym, tzn. po zainicjowaniu jego pracy, w jednym przejściu uzyskuje się program wynikowy w języku wewnętrznym mikrokomputera. Kompilacja programu przebiega w czterech fazach. Fazy te, inicjowane automatycznie /tj. bez ingerencji użytkownika/, realizowane sa przez moduły kompilatora o funkcjach:

Faza	Nazwa modułu	Funkcja
I	ADA.COM	Analiza leksykalna
II	ADA2.COM	Analiza syntaktyczno-semantyczna
III	ADA3.COM	Optymalizacja
IV	ADA4.COM	Generacja kodu

Każdy z modułów w wyniku swego działania tworzy plik zawierający pewną formę pośrednią programu źródłowego, z której korzysta następny moduł. Nazwy plików pośrednich, tworzonych w czasie kompilacji, składają się z pierwszej części nazwy programu źródłowego oraz rozszerzenia właściwego danemu modułowi, np., gdy kompilowany jest program źródłowy o nazwie PROG.ADA, to generowane

są pliki pośrednie o następujących nazwach:

Nazwa modułu	Plik wejściowy	Plik tworzony
ADA.COM	PROG.ADA	PROG.TOK
ADA2.COM	PROG.TOK	PROG.INT
ADA3.COM	PROG.INT	PROG.OPT
ADA4.COM	PROG.OPT	PROG.COM

Po dokonaniu kompilacji wszystkie pliki pośrednie /tj. z rozszerzeniami TOK /ang. tokens/, INT /ang. intermediate form/ i OPT /ang. optimized// zostają automatycznie skasowane. W trakcie kompilacji generowany jest raport, zawierający informacje o realizacji poszczególnych faz. Oto przykład raportu generowanego w czasie kompilacji poprawnego programu:

```
Ada Phase I /Lexical Analysis/

Ada Phase II /Syntactical/Semantic Analysis/
00000 Syntax error/s/
Ada Phase III /Optimization/
47 optimizations performed
Ada Phase IV /Code Generation/
01230 bytes of machine code generated
compilation complete.
```

W trakcie analizy leksykalnej kompilator wyprowadza znak ' +' po przeanalizowaniu 16 zdań programu źródłowego. A oto przykład raportu, jaki generowany jest w czasie kompilacji błędnego programu:

```
Ada Phase I /Lexical Analysis/
Error in line 9: Single character or single
                 quote expected
Error in line 9: Invalid symbol

Ada Phase II /Syntactical/Semantic Analysis/
Error # 00027 in line 00005 - Błędny identyfikator typu lub podtypu
Error # 00068 in line 00005 - Błędna deklaracja
Error # 00009 in line 00008 - Wystąpił nieoczekiwany symbol
Error # 00080 in line 00009 - Nierozpoznana instrukcja
Error # 00043 in line 00010 - Niezgodne typy
Error # 00017 in line 00011 - Niedopasowany identyfikator po 00007
identyfikator po
' end '                                   00007
Syntax error/s/

Ada Phase III not loaded due to Phase II errors
Fatal error - compilation aborted.
```

Jak widać, w raporcie tym sygnalizowane jest wykrycie w programie źródłowym zarówno błędów leksykalnych, jak i syntaktycznych. /Listy błędów wykrywanych przez kompilator podane są w Dodatkach B i C./

Komunikaty o błędach syntaktycznych mają podaną wyżej postać tylko wtedy, gdy do kompilatora dołączony jest opcjonalny plik o nazwie ERRORS.TXT, zawierający opisy błędów. Jeżeli plik ten nie występuje, to raport ma postać skróconą:

```
Ada Phase I /Lexical Analysis/
Error in line 9: Single character or single
quote expected
Error in line 9: Invalid symbol

Ada Phase II /Syntactical/Semantic Analysis/
Error # 00027 in line 00005
Error # 00068 in line 00005
Error # 00009 in line 00008
Error # 00009 in line 00009
Error # 00080 in line 00009
Error # 00043 in line 00010
Error # 00017 in line 00011
00007 Syntax error/s/

Ada Phase III not loaded due to Phase II errors
Fatal error - compilation aborted.
```

W podanym wyżej przykładzie, ze względu na występujące błędy, kompilacja została przerwana po wykonaniu Fazy II. Należy zaznaczyć, że w pewnych przypadkach kompilacja przerywana jest natychmiast, tj. bez dalszej analizy tekstu źródłowego. Np. po wystąpieniu błędu w pierwszym wierszu programu, powodującym nierozpoznanie rodzaju jednostki kompilacji, praca kompilatora jest przerywana.

Użytkowanie kompilatora

Aby skorzystać z kompilatora języka Ada dla mikrokomputera ComPAN-8 należy dysponować modułami kompilatora, zgromadzonymi na pojedynczej dyskietce:

Nazwa modułu	Objętość	
ADA.COM	5k	
ADA2.COM	19k	
ADA3.COM	19k	
ADA4.COM	10k	
KAPSE	6k	
ERRORS.TXT	9k	/moduł opcjonalny/

Funkcje modułów ADA.COM - ADA4.COM oraz ERRORS.TXT zostały omówione w poprzednim rozdziale. Moduł KAPSE /ang. Kernel Ada Programming Support Environment/ dołączony jest do każdego programu wynikowego, generowanego przez kompilator i składa się z procedur niezbędnych do wykonywania tego programu /moduł zawiera m.in. procedury działań arytmetycznych, wejścia-wyjścia itp./.

Obecnie omówione zostaną czynności, jakie należy wykonać dla skompilowania i wykonania przykładowego programu o nazwie PRZYKŁAD.ADA. Program ten dołączony jest na dyskietce z kompilatorem. Tekst programu przedstawiony jest w Dodatku A.

Po wprowadzeniu do pamięci operacyjnej mikrokomputera systemu operacyjnego CP/M dy-

skietkę z kompilatorem Ady i programem przykładowym umieszczamy w mechaniźmie A stacji dysków elastycznych, a następnie dokonujemy kompilacji programu przy pomocy zlecenia:

A>ADA PRZYKŁAD.ADA

/tekst podkreślony wyprowadzany jest przez mikrokomputer/. W wyniku kompilacji, na tej samej dyskietce otrzymujemy program przykładowy o nazwie PRZYKŁAD.COM. Aby wykonać ten program należy wydać zlecenie:

A>PRZYKŁAD

Oczywiście, możliwe jest również inne rozmieszczenie kompilatora Ady oraz tłumaczonego programu w mechanizmach stacji dysków. Oto kilka zleceń przykładowych dla innych rozmieszczeń:

A>ADA B:PR1.ADA

/Kompilator Ady znajduje się w mechaniźmie A, a program źródłowy PR1.ADA na dyskietce w mechaniźmie B./

A>B:ADA B:PROG.ADA

/Kompilator Ady i program źródłowy PROG.ADA umieszczono w mechaniźmie B./

A>B:ADA KALK.ADA

/Kompilator Ady znajduje się w mechaniźmie B, a program źródłowy KALK.ADA w mechaniźmie A./

U w a g i :

1. Podając w zleceniu kompilacji nazwę programu, można nie specyfikować jego rozszerzenia o postaci .ADA. Kompilator automatycznie wyszuka program o podanej nazwie i tym rozszerzeniu. Ostatnie przykładowe zlecenie może więc mieć również postać:

A>B:ADA KALK

2. Nazwy programów dla kompilatora Ady mogą mieć dowolne rozszerzenia. Kompilując te programy należy podać ich pełne nazwy, tj. łącznie z rozszerzeniem, np.:

A>ADA PROG.MOJ

Opracowanie niniejsze prezentuje kompilator języka Ada dla mikrokomputera ComPAN-8. Kompilator akceptuje podzbiór języka, który nie obejmuje takich cech jak: /a/ pakiety, /b/ jednostki rodzajowe, /c/ współbieżne wykonywanie zadań i /d/ obsługę wyjątków. Mimo tych braków, kompilator umożliwia budowanie stosunkowo złożonych programów użytkowych, pozwala na zapoznanie się ze szczegółami składniowymi języka Ada i jest prosty w obsłudze.

L i t e r a t u r a :

[1] A. Cały, Z Czech, M. Konopka: Kompilator języka Ada dla mikrokomputera ComPAN, Raport Instytutu Informatyki, Politechnika Śląska, Gliwice, 1986, s. 90.

[2] Z. Czech, M. Konopka: Projekt weryfikatora poprawności syntaktycznej tekstów w języku Ada, Raporty Instytutu Informatyki, Politechnika Śląska, Gliwice, 1984 i 1985, s. 72 i s. 28.

[3] A.N. Habermann: Ada for Experienced Programmers, Addison-Wesley, 1983.

[4] C.A.R. Hoare, : The Emperor's old clothes, Comm. ACM 24, 2 /1981/, 75-83.

[5] H.F. Ledgard, A. Singer: Scaling down Ada, or towards a standard subset, Comm. ACM 25, 2 /1982/, 121-125.

[6] E.W. Olsen, S.B. Whitehill: Ada for Programmers, Prentice - Hall, 1983.

[7] I.C. Pyle: Ada, WNT, Warszawa, 1986.

[8] Reference Manual for the Ada Programming Language, ANSI/MIL - STD - 1815A - 1983,

[9] Dokumentacja użytkowa dyskowego systemu operacyjnego CP/M 2.2, Tom X.

dr inż. TADEUSZ CZACHÓRSKI
mgr inż. MAREK KOWALÓWKA
mgr inż. ZDZISŁAW SZCZERBIŃSKI
mgr inż. ANDRZEJ WILK
Zakład Systemów Automatyki
Kompleksowej PAN
Gliwice

"AMOK" - ANALIZATOR MODELI KOLEJKOWYCH

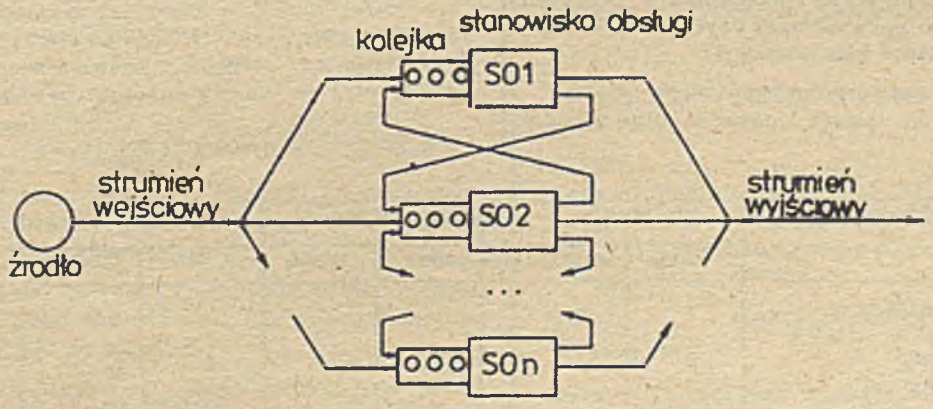
AMOK jest opracowanym w Zakładzie Systemów Automatyki Kompleksowej PAN w ramach CPBR 8.6 pakietem programowym, umożliwiającym praktyczne stosowanie modeli teorii obsługi masowej. Pakiet ten jest przeznaczony do pracy z mikrokomputerem ComPAN w wersji 16-bitowej lub każdym innym mikrokomputerem kompatybilnym z IBM PC/XT. Głównym zastosowaniem pakietu jest modelowanie i ocena efektywności systemów komputerowych, może być również wykorzystywany w innych tradycyjnych zastosowaniach teorii masowej obsługi, gdzie modelowane obiekty, np. linie produkcyjne czy magazyny, przedstawione są jako sieć kolejek i stanowisk obsługi.

Sieć stanowisk obsługi, pomiędzy którymi szukają kontaktu klienci, to model służący do opisu bardzo różnorodnych obiektów. Do tradycyjnych zastosowań należy m. in.: modelowanie pracy linii produkcyjnych, stanowisk przeładunkowych, magazynów, transportu, ruchu ulicznego, a także modelowanie systemów komputerowych i sieci komputerowych.

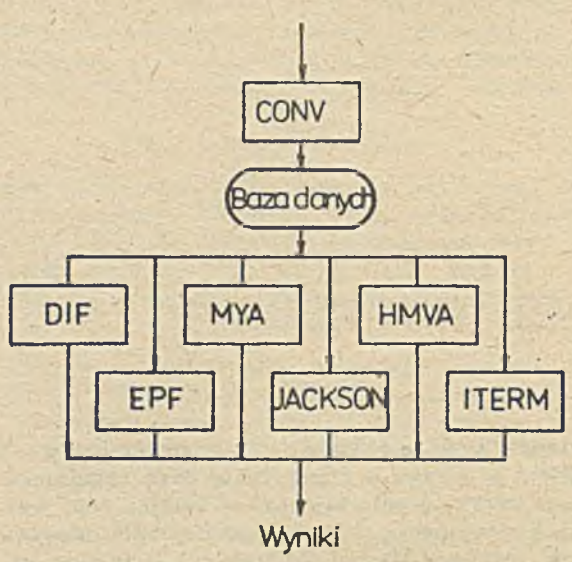
Ogólne zadanie, które rozwiązuje AMOK jest następujące:
Dana jest sieć stanowisk obsługi /rys. 1./. Liczba stanowisk i topologia sieci są dowolne. Momenty nadejścia klientów i czas ich obsługi są wielkościami losowymi. Każde stanowisko scharakteryzowane jest rozkładem prawdopodobieństwa czasu obsługi /czasu, przez który klient przebywa w stanowisku/ oraz regulaminem szeregowania klientów w kolejce /np. według kolejności zgłoszeń, w kolejności odwrotnej, wszyscy klienci obsługiwani są jednocześnie/. Opis sieci uzupełniają: rozkład odstępów czasu między klientami w strumieniu wejściowym oraz prawdopodobieństwa przejść klientów między stanowiskami.

Sieć może być otwarta - istnieje strumień wejściowy klientów nadchodzących do sieci i strumień wyjściowy klientów opuszczających sieć - lub zamknięta - w sieci krążą stale ci sami klienci. Klienci mogą dzielić się na klasy, różniące się czasem obsługi, priorytetem i drogą w sieci. Należy obliczyć miary pracy stanowisk sieci - wykorzystanie poszczególnych stanowisk /część czasu, przez który stanowisko pracuje/, średnią długość kolejek i średni czas czekania w nich, przepustowość /liczbę obsłużonych w jednostce czasu klientów/ poszczególnych stanowisk oraz średni czas przejścia przez całą sieć.

Obecnie nie ma jednej ogólnej metody rozwiązania przedstawionego powyżej problemu. Przyjęty sposób postępowania zależy od szczegółowych założeń, dotyczących typu rozkładów czasu obsługi, regulaminów kolejek, a także od rozmiaru sieci, liczby stanowisk, liczby klientów i ich klas. Wynika stąd potrzeba wie-



Rys. 1. Ogólna postać sieci stanowisk obsługi



Rys. 2. Struktura AMOKu

lu algorytmów obliczeniowych. Modele w postaci sieci stanowisk obsługi, mimo że proste pojęciowo, wymagają dość złożonego aparatu matematycznego i żmudnych czasem obliczeń. Dla ich praktycznego zastosowania jest więc konieczne użycie maszyny cyfrowej i przygotowanie odpowiedniego oprogramowania zorganizowanego w postaci pakietów obliczeniowych. Wśród istniejących w świecie tego typu pakietów najbardziej znane to: opracowany w IBM pakiet RESQ [3], pakiet BEST [1] autorstwa BGS Systems oraz francuski QNAP [2] /opracowany w INRIA przy współpracy Honeywell-Bull/. Na podstawie wiedzy autorów, AMOK jest pierwszym tego typu pakietem w Polsce, a także pierwszym w krajach RWPG.

Strukturę AMOKu ilustruje rys. 2. Moduł wejściowy CONV umożliwia użytkownikowi formułowanie zadań w specjalnie przygotowanym w tym celu języku opisu modeli nazwanym QNDEL oraz wybór metody rozwiązania i postaci wyników. Na podstawie opisu modelu tworzony jest zbiór danych wejściowych dla modułów obliczeniowych pakietu /JACKSON, MVA, itd./. Moduły te uwzględniają wszystkie podstawowe, o sprawdzonej użyteczności metody analizy sieci kolejek.

Tabela 1 zawiera główne cechy sieci rozpatrywanych przez poszczególne moduły pakietu. Tabela 2 podaje maksymalne rozmiary sieci /maksymalną liczbę stanowisk obsługi, liczbę klas, dopuszczalną liczbę klientów w przypadku sieci zamkniętej/ analizowanych przez wersję AMOKu, wygenerowaną przez autorów na mikrokomputerze IBM PC/XT z 640 KB pamięci operacyjnej. Następne rozdziały artykułu zawierają: semantykę języka opisu sieci kolejek, uwagi na temat metod leżących u podstaw modułów obliczeniowych AMOKu, przykłady wykorzystania AMOKu.

Tabela 2

Moduł	Liczba źródeł	Liczba stacji	Liczba klas	Liczba klientów
JACKSON	12	12	8	-
DIF	12	12	4	-
MVA	-	30	10	∞ [1]
HMVA	-	30	10	∞ [1]
EPF	-	5	1	10
ITERM	-	10	1	50

[1] brak ograniczeń ze strony algorytmu i ograniczeń pamięci.

Język opisu modelu

Opis sieci stanowisk obsługi w języku QNDEL [12] składa się z opisu poszczególnych elementów sieci: źródeł /w przypadku sieci otwartej/, stanowisk obsługi i ich kolejek oraz elementów biernych, przyznawanych klientom, a później przez nich zwalnianych. Prawdopodobieństwa przejść pomiędzy stanowiskami, określające drogę klientów w sieci zalicza się do parametrów elementów sieci.

Poniżej przedstawiona jest semantyka języka QNDEL.

/opis sieci/ ::= /opis elementu sieci/
...
/opis elementu sieci/

gdzie:

/opis elementu sieci/ ::= / /typ elementu//
/identyfikator/
/parametr 1/
...
/parametr n/

/typ elementu/ ::=SOURCE|STATION|RESOURCE
/identyfikator/ ::=NAME= nazwa elementu
/parametr/ ::= /nazwa parametru/ =/ definicja parametru/.

Tabela 1

Moduł	typ sieci		liczba klas		typ stanowisk	
	otwarta	zamknięta	jedna	wiele	BCMP	G/G/1
JACKSON	+			+	+	
MVA, HMVA		+		+	+	
EPF		+	+		+	+
ITERM		+	+		+	+
DIF	+			+		+

Parametry, które zależą od klasy klientów, np. czas obsługi i prawdopodobieństwa przejść, są podawane dla każdej klasy oddzielnie. W tym przypadku
/parametr/::=/nazwa parametru///pierwsza lista klas//=/definicja parametru/:

...

/nazwa parametru///k-ta lista klas//=/definicja parametru/:

gdzie:
/lista klas/::= nazwa klasy ,.... nazwa klasy.

Przyjęto następujące nazwy parametrów definiujących elementy sieci kolejek:
/nazwa parametru/::=MAX|SIZE|SCHEDULE|SERVICE|TRANSITION

gdzie:
MAX definiuje ograniczenie maksymalnej długości kolejki.

SIZE definiuje: dla stanowiska - liczbę kanałów obsługi, dla zasobu - wymiar zasobu.

SCHEDULE definiuje regulamin szeregowania klientów.

SERVICE definiuje: dla źródła - rozkład odstępów czasu pomiędzy chwilami generacji kolejnych klientów, dla stanowiska obsługi, rozkład czasu obsługi, dla zasobu - ilość pobieranego zasobu.

TRANSITION definiuje prawdopodobieństwa wyboru następnego stanowiska dla klientów opuszczających dany element /definiuje prawdopodobieństwa przejść pomiędzy stanowiskami.

Parametry MAX i SIZE przyjmują wartości całkowite i są opcjonalne. Jeżeli są pominięte, przyjmują standardowe wartości, wynikające z przyjętego w danym elemencie regulaminu szeregowania.

Parametr SCHEDULE podaje nazwę regulaminu szeregowania z występującą po niej, ujętą w nawiasy okrągłe, listą szeregowanych w tym elemencie sieci klas klientów. Lista ta jest niezbędna, w przypadku konieczności ustalenia hierarchii klas, np. przy zastosowaniu priorytetowego regulaminu szeregowania: w pozostałych przypadkach jest opcjonalna.
/definicja SCHEDULE/::=/nazwa regulaminu szeregowania/ //lista klas//
gdzie:
/nazwa regulaminu szeregowania/::=FCFS|LCFS|LCFS-PR|IS|PS|PRIOR|PRIOR-PS.

FCFS /first-come-first-served/ oznacza naturalny regulamin kolejki, tj. zgodny z kolejnością nadejścia klientów.

LCFS /last come-first-served/ oznacza regulamin obsługi w kolejności odwrotnej do kolejności nadejścia klientów.

LCFS-PR /last-come-first-served-preempted-resume/ oznacza regulamin LCFS z przerwaniem aktualnie wykonywanej obsługi w momencie nadejścia nowego klienta.

IS /infinite server/ oznacza stanowisko z nieskończoną liczbą równoległych kanałów obsługi.

PS /processor sharing/ oznacza obsługę z podziałem czasu procesora - wszyscy obecni w stanowisku klienci są obsługiwani jednocześnie, czas ich obsługi rośnie proporcjonalnie do liczby obsługiwanych klientów.

PRIOR oznacza kolejkę priorytetową bez przerwań - wybór klienta o najwyższym priorytecie i jego obsługa następują każdorazowo po zakończeniu obsługi poprzedniego klienta.

PRIOR-PR oznaczą regulamin priorytetowy z przerwaniami: obsługa klienta jest zawieszona w momencie nadejścia klienta o wyższym priorytecie.

Jeżeli parametr SCHEDULE nie występuje, przyjmuje się, że regulamin jest typu FCFS.

Ogólna definicję parametru SERVICE stanowi zestaw oddzielonych od siebie znakiem równości definicji prostych, którymi mogą być definicje rozkładu losowego odcinków czasu lub definicje pobrania /zwrotu/ zasobu przez klienta.
/definicja SERVICE/::=/prosta definicja SERVICE/

...

=/prosta definicja SERVICE/

gdzie:
/prosta definicja SERVICE/::=/definicja rozkładu losowego/|
/definicja pobrania zasobu/|
/definicja zwrotu zasobu/
/definicja rozkładu losowego/::=
/nazwa rozkładu/ /parametr rozkładu,..... parametr rozkładu/
/nazwa rozkładu/::=EXP|DET|UNI|COX|ERL|G|HIP
/definicja pobrania zasobu/::=RESERVE /liczba porcji/
/definicja zwrotu zasobu/::=RELEASE/nazwa zasobu /liczba porcji/.....,nazwa zasobu /liczba porcji//
uwzględniono następujące rozkłady:

EXP/m_1/ - rozkład wykładniczy o średniej m_1.

DET/m_1/ - rozkład stałej o wartości m_1.

UNI/r_1, r_2/ - jednostajny pomiędzy wartościami r_1 i r_2.

COX/p_1, m_1,...., p_n, m_n/ - Coxa z n wykładniczymi fazami: po fazie i, i=1, 2,...., n-1 następuje faza i+1 z prawdopodobieństwem p_{i+1} lub też faza i jest z prawdopodobieństwem $1-p_{i+1}$ fazą ostatnią.

ERL/n, m_1/ - rozkład Erlanga n-tego rzędu o średniej m_1.

G/m_1, C/ - rozkład ogólny o średniej m_1 i współczynniku zmienności C, C=var/m_1^2.

HIP/$p_1, m_1, \ldots, p_n, m_n$/ - hiperwykładniczy o n równoległych wykładniczych fazach, tylko jedna z nich, faza i, i = 1,...,n jest wybierana z prawdopodobieństwem p_i.

Prawdopodobieństwo przejść określa się następująco:

/definicja TRANSITION/::=/przesłanie proste/ /prawdopodobieństwo przesłania//przesłanie proste/,

/prawdopodobieństwo przesłania//przesłanie proste/
gdzie:
/przesłanie proste/::=nazwa elementu sieci: nazwa klasy klientów /nazwa elementu sieci/ OUT.

Występujaca powyżej nazwa elementu sieci jest nazwą elementu STATION lub RESOURCE nadaną mu przez użytkownika. Jeżeli podczas przesłania klient zmienia klasę, do której należy, to fakt ten zaznacza się w przesłaniu prostym,podając za znakiem dwukropka nazwę nadawanej klasy. Symbol OUT pozwala w sposób jawny zadeklarować, że klienci opuszczają sieć.

Klasa modeli opisywanych przez język AMOKu jest szersza od możliwości rozwiazywania problemów przez zawarte obecnie w tym pakiecie moduły analityczne: moduły te nie analizują synchronizacji klientów w sieci ani nie uwzględniają obecności elementów biernych.

Moduły obliczeniowe

Modelowanie w AMOKu polega na obliczeniu wskaźników efektywności pracy /średniej liczby klientów w każdym ze stanowisk obsługi, średniego czasu czekania w kolejce, przepustowości stanowisk itp./ charakteryzujacych stan ustalony /stan równowagi stochastycznej sieci stanowisk obsługi. Metody analityczne dostępne obecnie w AMOKu są metodami dokładnymi lub przybliżonymi. Po ustaleniu założeń dotyczących postaci strumieni wejściowych, rozkładów czasów obsługi, regulaminów kolejek metody dokładne prowadzą do ścisłych matematycznie rozwiazań; metody przybliżone upraszczają założenia lub też dokonuja przybliżeń w trakcie obliczeń.

Metody dokładne zawarte są w modułach JACKSON i MVA, są one związane z najogólniejszym modelem analitycznym, zwanym od nazwisk autorów modelem BCMP [4]. Model ten zakłada wiele klas klientów, sieci mieszane /otwarte dla pewnych klas klientów, zamknięte dla pozostałych/ oraz cztery typy stanowisk obsługi:

- stanowiska z naturalnym /FCFS/ regulaminem kolejki i wykładniczym, jednakowym dla wszystkich klas klientów rozkładem czasów obsługi,
- stanowiska z obsługą z podziałem procesora /PS/, gdzie moc obliczeniowa stanowiska jest dzielona pomiędzy wszystkich obecnych w nim klientów: rozkład czasów obsługi jest typu Coxa różnym dla każdej klasy klientów,
- stanowiska IS posiadajace nieskończona liczbę kanałów obsługi /lub przynajmniej nie mniejszą od liczby możliwych klientów/ /IS/, w których czas czekania na obsługę nie występuje, a rozkład czasu obsługi jest rozkładem typu Coxa, niezależnym od liczby klientów obecnych w stanowisku i charakterystycznym dla każdej klasy klientów,
- stanowiska z regulaminem szeregowania LCFS-PR i rozkładem czasów obsługi takim samym jak w dwu poprzednich rodzajach stanowisk.

Zaletą rozkładów Coxa, charakteryzujacych czas obsługi w trzech z czterech typów stanowisk omawianego modelu, jest to, iż przy ich pomocy można przybliżyć z dowolną dokładnością każdy występujacy w praktyce rozkład prawdopodobieństwa. Obliczenia numeryczne związane z modelem BCMP nie są jednak proste [10]. Największą trudność stanowi obliczenie stałej normalizacyjnej, która służy do skalowania prawdopodobieństw stanów: liczba stanów szybko wzrasta wraz ze wzrostem liczby stanowisk i liczby klientów w systemie. Przykładowo: zamknięta sieć dziesięciu stanowisk, w której krąży dziesięciu klientów może znajdować się w ponad 92.000 stanów. W celu ominięcia tej trudności zostały stworzone efektywne algorytmy obliczeniowe. Moduł JACKSON w AMOKu rozwiązuje otwartą sieć BCMP z czasami obsługi niezależnymi od liczby klientów w kolejce, w tym przypadku nie zachodzi konieczność obliczania stałej normalizacyjnej. Moduł MVA rozwiązuje zamkniętą sieć BCMP przy użyciu algorytmu wartości średnich [5] - który, przez zastosowanie procedury rekursywnej rozwiązującej sieć z liczbą klientów 1,2,... aż do N i unika jawnego obliczania stałej normalizacyjnej. Moduł HMVA realizuje wersję przybliżoną algorytmu MVA i jest wykorzystywany dla dużych sieci BCMP ze znaczną liczbą klientów [6].

Pozostałe moduły są realizacją metod przybliżonych, które pozwalają na analizę dużych i bardziej ogólnych sieci /np. sieci niezgodne z założeniami sieci typu BCMP, przede wszystkim sieci zawierające stanowiska z regulaminem obsługi FCFS i ogólnym rozkładem czasu obsługi, różnym dla różnych klas klientów/ przez zredukowanie ilości obliczeń do poziomu akceptowalnego przy wykorzystywaniu mikrokomputera:

- EPF - rozszerzona forma iloczynowa, metoda aproksymująca rozkład p/n/ liczby klientów w stanowisku sieci przez p/n/ dla pojedynczego stanowiska M/G/1/N [8],
- DIF - aproksymacja dyfuzyjna, gdzie proces N/t/, przedstawiający liczbę klientów obecnych w stanowisku w chwili t, jest zastępowany procesem dyfuzji: prawdopodobieństwo n klientów obecnych w stanowisku p/n/ jest aproksymowa-

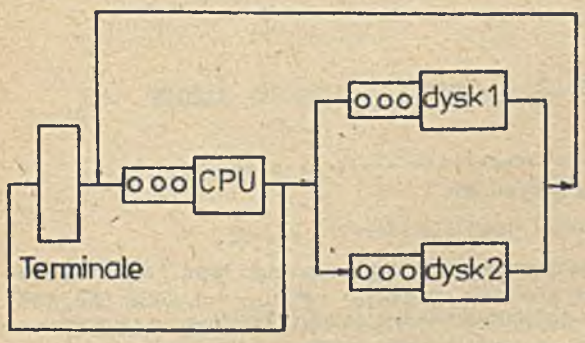
ne przez f/n/, funkcję gęstości prawdopodobieństwa procesu dyfuzji, otrzymywaną jako rozwiązanie równania dyfuzji z odpowiednio dobranymi współczynnikami i warunkami brzegowymi, [7].

● ITERM - metoda iteracyjna oparta o zastosowanie twierdzenia Norton' a do sieci stanowisk obsługi. Każde stanowisko widzi resztę sieci jako źródło klientów, które generuje strumień Poisson' a z parametrem zależnym od liczby klientów obecnych w rozpatrywanym stanowisku [9].

Przykłady wkorzystania pakietu

Przykład 1. Model systemu komputerowego

Rozpatrzmy pracę wielodostępnego systemu transakcyjnego. System składa się z jednostki centralnej, dwu dysków i zbioru terminali. Przetwarzane są trzy typy transakcji. Transakcje pierwszych dwu typów generowane są przez użytkowników, którzy przechowują swoje dane na /odpowiednio/ dysku 1 i dysku 2. Trzeci typ transakcji składa się z jednej transakcji wykonującej prace administracyjne i statystyczne w odniesieniu do baz danych na obu dyskach. Transakcje pierwszego typu wymagają średnio 10 dostępów do danych na dysku 1 w czasie jednego wykonania transakcje drugiego typu odwołują się do danych na dysku 2, średnio 4 razy w trakcie jednego przetwarzania. Transakcja administratora systemu odwołuje się średnio 13 razy do dysku 1 i 16 razy do dysku 2.



Rys. 3. Analizowany model systemu transakcyjnego

Model systemu ilustruje rys. 3. Terminale modelowane są jako jedno stanowisko opóźniające /stanowisko o nieskończonej liczbie kanałów obsługi/, jednostka centralna to stanowisko obsługujące w trybie podziału czasu, z kolei dyski modelowane są przez jednokanałowe stanowiska FCFS - obsługiwany jest tylko jeden klient, a kolejność obsługi jest zgodna z kolejnością nadchodzenia do stanowiska. Liczba klientów krążących w sieci jest równa liczbie aktywnych transakcji, a ta z kolei jest równa liczbie aktywnych terminali. W modelu rozpatrywane są trzy klasy klientów odpowiadające trzem typom transakcji. Prawdopodobieństwa przejść obliczane są na podstawie średniej, względnej liczby wizyt na stanowiskach, np. dla pierwszej klasy klientów prawdopodobieństwo przejść pomiędzy CPU i dyskiem 1 jest równe $\frac{10}{11} \approx 0.909$, pomiędzy CPU i terminalami $\frac{1}{11} \approx 0.091$. Zakładamy, że czas namysłu w terminalu może być aproksymowany przez rozkład wykładniczy dla pierwszego typu klientów i przez rozkład Erlanga odpowiednio 3 i 5 stopnia dla drugiego i trzeciego typu transakcji. Odpowiednio czasy namysłu wynoszą: 20 s, 30 s, 120 s.

Czasy obsługi w CPU /czasy nieprzerwanego przetwarzania transakcji/ mają rozkład wykładniczy /średnia = 0.2 s/ dla pierwszego typu transakcji, Cox' a /2 fazy, średnia = 0.14 s / dla drugiego typu i Erlang' a /2 stopnia, średnia = 1.5 s / dla transakcji administratora systemu. W systemie aktywnych jest 20 terminali. Połowa z nich generuje transakcje pierwszego, a połowa drugiego typu /$N_1 = N_2 = 10$/. Czasy obsługi w dyskach są niezależne od klasy transakcji i mają rozkład wykładniczy o średniej 0.12 s w dysku 1 i 0.18 s w dysku 2.

Problemy do rozwiązania

● Jakie są parametry wydajnościowe systemu dla podanych powyżej danych?

● Jaki jest wpływ transakcji administratora na wydajność systemu?

● Załóżmy, że:
/i/ liczba aktywnych terminali generujących transakcje pierwszego typu zostanie zmniejszona /$N_1 = 5$, $N_2 = 10$/,
/ii/ liczba aktywnych terminali pozostanie bez zmian, /$N_1 = 10$, $N_2 = 10$/, ale jednostka centralna CPU zostanie zastąpiona przez szybszą - wszystkie czasy zostaną dwukrotnie skrócone.
Jaki wpływ będą miały te zmiany na czas reakcji systemu?

Rozwiązanie:

Opis badanego modelu w języku QNDEL przedstawiony jest na rys. 4. Załóżmy, że opis modelu znajduje się w pliku EX. Zlecenie

CONV EX, CON, EX, BAS

wywołuje program konwertora CONV, który analizuje opis i tworzy plik wyjściowy EX, BAS zawierający dane o modelu. "CON" w zleceniu oznacza, że raport konwersji ma być prezentowany na ekranie monitora. Poczynione dla modelu założenia są zgodne z założeniami sieci typu BCMP, więc można wykorzystać do rozwiązania moduł MVA. Zlecenie

SOLUTION MVA, EX, BAS, CON, EX, RES, END

uruchamia moduł MVA i powoduje przesłanie wyników /rysunki 5-7/ do pliku EX, RES.

Wyniki pokazują, iż system jest przeładowany, wąskim zaś gardłem jest jednostka central-

```
*DESCRIPTION*

    /STATION/  NAME = TERMINALS
               SCHEDULE = IS
               SERVICE(:DBASE1) = EXP[20]
          ! TRANSACTIONS ACCESSING DATA BASE ON DISK 1 !
               SERVICE(:DBASE2) = ERL[3,30]
          ! TRANSACTIONS ACCESSING DATA BASE ON DISK 2 !
               SERVICE(:ADMIN)  = ERL[5,120]
          ! ADMINISTRATIVE TRANSACTION !
               TRANSIT = CPU
    /STATION/  NAME = CPU
               SCHEDULE = PS
               SERVICE(:DBASE1) = EXP[0.2]
               SERVICE(:DBASE2) = COX[1,0.1,0.5,0.08]
               SERVICE(:ADMIN)  = ERL[2,1.5]
               TRANSIT(:DBASE1) = [0.909] DISK1,[0.091] TERMINALS
               TRANSIT(:DBASE2) = [0.8] DISK2,[0.2] TERMINALS
               TRANSIT(:ADMIN)  = [0.433] DISK1,
                                  [0.533] DISK2,
                                  [0.034] TERMINALS
    /STATION/  NAME = DISK1
               SERVICE = EXP[0.12]
               TRANSIT = CPU
    /STATION/  NAME = DISK2
               SERVICE = EXP[0.18]
               TRANSIT = AS(DISK1)

*END*
```

Rys. 4. Opis modelu z rysunku 3

na CPU. Współczynnik wykorzystania CPU wynosi $\varrho_{CPU} = 0.9666$, co oznacza, że jednostka centralna jest zajęta przez 96,66% czasu. Trzy typy transakcji wykorzystują, odpowiednio, 63,97%, 19,48% i 12,84% czasu CPU. Dyski są znacznie mniej obciążone: $\varrho_{dysk1} = 0.3533$, $\varrho_{dysk2} = 0.2123$. Czas reakcji systemu /znajdujemy go w kolumnie MEAN CYCLE TIME dla terminali/ jest długi: $R_1 = 14.35$ s dla transakcji pierwszego typu i $R_2 = 5.37$ s dla drugiego typu. Przepustowość systemu, tj. liczba transakcji obsłużonych w ciągu sekundy /równa przepustowości terminali/ wynosi $Thr_1 = 0.29$, $Thr_2 = 0.28$ dla pierwszego i drugiego typu transakcji.

Po wyeliminowaniu transakcji administratora systemu /przeliczamy model z tylko dwoma typami transakcji/ wykorzystanie CPU wyniesie $\varrho_{CPU} = 0.9070$ a czasy reakcji $R_1 = 11.25$ s i $R_2 = 4.21$ s. Ograniczenie liczby transakcji pierwszego typu /$N_1 = 5$/ nieco poprawi sytuację: $\varrho_{CPU} = 0.8038$ /80,38% = 39,80% + 21,17% + 19,41%/, czas reakcji spadnie do $R_1 = 7,61$ s, $R_2 = 3.07$ s, ale rzeczywistą poprawę da przyspieszenie CPU. W tym przypadku $\varrho_{CPU} = 0.6830$, $R_1 = 4.75$ s, $R_2 = 1.82$ s. Przepustowość wzrośnie do $Thr_1 = 0.40$ i $Thr_2 = 0.31$ transakcji/s.

Przykład 2. Czas czekania na remont maszyn

W zakładzie przemysłowym pracuje 20 maszyn, ulegających w sposób losowy awariom. Rozkład czasu między awariami można przybliżyć rozkładem wykładniczym o średniej 150 dni. Awarie są dwojakiego rodzaju. Awaria pierwszego typu występuje z prawdopodobieństwem 0.3 i jej naprawa w warsztacie nr 1 trwa średnio 10 dni. Awarie drugiego typu występują z prawdopodobieństwem 0.7, a ich naprawa w warsztacie nr 2 wymaga średnio 5 dni pracy.

```
**********      MEAN-VALUE ANALYSIS      **********
************************************************************
                NUMBER OF STATIONS :  4
                NUMBER OF CLASSES  :  3
                CLASS :          POPULATION SIZE :
                 DBASE1                10
                 DBASE2                10
                 ADMIN                  1
************************************************************
                STATION :  TERMINALS

   TYPE OF SERVICE : INFINITE SERVER
   NUMBER OF CLASSES VISITING STATION : 3
   CLASSES VISITING STATION :  DBASE1, DBASE2, ADMIN

          MEAN       MEAN       MEAN       MEAN     THROUGH-   UTILIZA-

          SERVICE    QUEUE      RESPONSE   CYCLE      PUT       TION
CLASS     TIME       LENGTH     TIME       TIME
-
DBASE1   20.0000    5.8220     20.0000    14.3522    .2911      --
-
DBASE2   30.0000    8.5052     30.0000     5.2725    .2835      --
-
ADMIN    20.0000     .3424    120.0000   230.4574    .0028      --
-

GLOBAL   25.4036   14.6696     25.4036    10.9623    .5774      --
-
************************************************************
```

Rys. 5. Wyniki obliczeń /kontynuacja na rysunkach 6 i 7/

Oba rozkłady można przybliżyć rozkładem wykładniczym. Jakie jest prawdopodobieństwo, że uszkodzona maszyna będzie czekała na naprawę i ile wynosi średni czas czekania?

Zapis zadania w języku AMOKu przedstawia rys. 8. Obliczone prawdopodobieństwo, że warsztat nr 1 pracuje /kolumna UTILIZATION/ wynosi 0.374, a prawdopodobieństwo, że drugi warsztat pracuje wynosi 0.436. Średni czas czekania na naprawę to czas odpowiedzi /kolumna MEAN RESPONSE TIME/ pomniejszony o średni czas obsługi, a więc na naprawę w pierwszym warsztacie trzeba czekać średnio 15.3 - 10 = 5.3 dni, a w drugim warsztacie 8.4 - 5 = 3.4 dni.

Przykład 3. Liczba łóżek niezbędnych w izbie porodowej

Jak świadczą zebrane dane statystyczne, do izby porodowej w miejscowości X zgłaszają się średnio 3 kobiety dziennie, a czas ich pobytu w izbie wynosi średnio 6 godzin. Odstępy między zgłoszeniami kobiet mają w przybliżeniu rozkład wykładniczy, a czas ich pobytu w izbie można aproksymować rozkładem Erlanga drugiego rzędu. Należy określić niezbędną liczbę łóżek w izbie, tak by prawdopodobieństwo iż wszystkie będą zajęte i następną położnicę będzie trzeba przewieźć do sąsiedniej miejscowości było mniejsze od 0.1.

Rys. 9 przedstawia zapis problemu w języku AMOKu. Stanowisko POŁOŻNICE jest tu źródłem zgłoszeń, a stanowisko ŁÓŻKA - badanym obiektem. Zakładając, że wszystkie zgłoszenia mogą być przyjęte /stanowisko ŁÓŻKA jest typu INFINIT SERVER/ obliczamy prawdopodobieństwo sytuacji, że w stano-

STATION : CPU

TYPE OF SERVICE : PROCESSOR SHARING
NUMBER OF CLASSES VISITING STATION : 3
CLASSES VISITING STATION : DBASE1, DBASE2, ADMIN

CLASS	MEAN SERVICE TIME	MEAN QUEUE LENGTH	MEAN RESPONSE TIME	MEAN CYCLE TIME	THROUGH-PUT	UTILIZA-TION
DBASE1	.2000	3.6611	1.1444	1.9815	3.1989	.6397
DBASE2	.1400	1.2431	.8769	6.1775	1.4175	.1984
ADMIN	1.5000	.6401	7.4785	4.2033	.0856	.1284
GLOBAL	.2055	5.5444	1.1791	3.2869	4.7020	.9666

STATION : DISK1

TYPE OF SERVICE : FCFS
NUMBER OF CLASSES VISITING STATION : 2
CLASSES VISITING STATION : DBASE1, ADMIN

CLASS	MEAN SERVICE TIME	MEAN QUEUE LENGTH	MEAN RESPONSE TIME	MEAN CYCLE TIME	THROUGH-PUT	UTILIZA-TION
DBASE1	.1200	.5168	.1777	3.2612	2.9078	.3489
ADMIN	.1200	.0070	.1905	26.7675	.0370	.0044
GLOBAL	.1200	.5238	.1779	3.5573	2.9449	.3533

Rys. 6. Wyniki obliczeń /kontynuacja z rysunku 5/

wisku jest 0, 1, ..., 30 klientów. Suma prawdopodobieństw P/0/+P/1/+P/2/+P/3/+P/4/ jest większa od 0,999, a więc w izbie niezbędne są co najmniej 4 łóżka.

AMOK nie jest przedsięwzięciem zamkniętym. Do końca 1987 r. będzie można definiować model wejściowy, posługując się manipulatorem kulowym /myszką/, wybierając strukturę i parametry modelu z wyświetlanego na ekranie menu, a wyniki będzie można otrzymywać również w postaci graficznej. Przygotowywane są następne moduły obliczeniowe przeznaczone do analitycznego modelowania sieci, zawierających zasoby pasywne oraz do modelowania zjawisk synchronizacji pracy klientów. Planuje się także dodanie modułu symulacyjnego oraz rozszerzenie możliwości języka QNDEL.

Oprócz opracowanej dla ZUK MERA-ELZAB instrukcji użytkowania pakietu ukaże się w 1987 r. wydanie skryptowe rozszerzonego podręcznika użytkowania AMOKu [11].

Zakładamy, iż AMOK powinien stać się systemem ekspertowym przeznaczonym do wspomagania pracy analityka systemów komputerowych lub innych systemów opisywanych w języku modeli masowej obsługi. Stanowi on również narzędzie dydaktyczne, które jest pomocne przy zaznajamianiu się studentów z kolejkowymi modelami

STATION : DISK2

TYPE OF SERVICE : FCFS
NUMBER OF CLASSES VISITING STATION : 2
CLASSES VISITING STATION : DBASE2, ADMIN

CLASS	MEAN SERVICE TIME	MEAN QUEUE LENGTH	MEAN RESPONSE TIME	MEAN CYCLE TIME	THROUGH-PUT	UTILIZA-TION
DBASE2	.1800	.2517	.2219	8.5961	1.1340	.2041
ADMIN	.1800	.0103	.2265	21.6771	.0456	.0082
GLOBAL	.1800	.2620	.2221	9.1024	1.1796	.2123

Rys. 7. Wyniki obliczeń /kontynuacja z rysunku 5/

systemów komputerowych i problemami numerycznymi, pojawiającymi się podczas rozwiązywania takich modeli.

Pierwotna wersja AMOKu, przygotowana dla systemu ODRA-1305 rozpowszechniana jest przez Zakład Systemów Automatyki Kompleksowej PAN w Gliwicach. Opisana w tym artykule wersja przeznaczona dla ComPANa-16 i innych mikrokomputerów kompatybilnych z IBM PC/XT/AT/ rozpowszechniana jest przez ZSAK PAN oraz przez Zakłady Urządzeń Komputerowych MERA-ELZAB w Zabrzu.

Literatura:

[1] BEST/1 User's Guide, BGS Systems, Inc., BE 78-020-1, Lincoln, MA, July 1978.

[2] D. Potier, New User's Introduction to QNAP2, Rapport Technique No 40, INRIA, Roquencourt 1984.

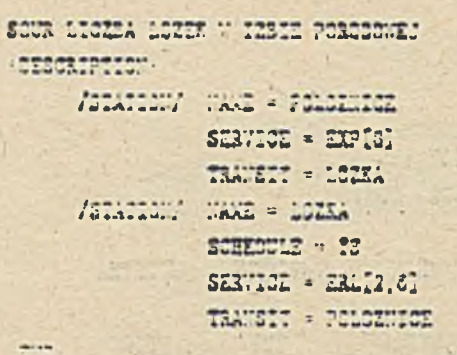
[3] C.H. Sauer, E.A. Mac Nair, J.F. Kurose, The Research Queueing Package Version 2: Introduction and Examples, IBM Research Report RA-138, Yorktown Heights, New York 1983.

[4] F. Baskett, M. Chandy, R. Muntz, J. Palacios, Open, Closed and Mixed Networks of Queues-with Different Classes of Customers, J. ACM, vol. 22, no. 2, 1975.

[5] M. Reiser, S.S. Lavenberg, Mean Value


```
SCUR CZAS CZEKANIA NA REMONT
*DESCRIPTION*
      /STATION/   NAME = MASZYNY
                  SCHEDULE = IS
                  SERVICE = EXP[150]
                  TRANSIT = [0.3] WARSZTAT-1, [0.7] WARSZTAT-2
      /STATION/   NAME = WARSZTAT-1
                  SERVICE = EXP[10]
                  TRANSIT = MASZYNY
      /STATION/   NAME = WARSZTAT-2
                  SERVICE = EXP[5]
                  TRANSIT = MASZYNY
*END*
```

Rys. 8. Opis modelu zakładu remontu maszyn



Rys. 9. Opis modelu izby porodowej

Analysis of Closed Multichain Queueing Networks, J. ACM, vol. 27, no. 2, 1980.
[6] K. M. Chandy, D. Neuse, Linearizer: A Heuristic Algo- rithm for Queueing Network Models of Computing Systems, Comm. ACM, vol. 25, no. 2, 1982.
[7] E. Gelenbe, G. Pujolle, The Behaviour of a Single Queue in a General Queueing Network, Acta Informatica, Fasc. 7, 1976.
[8] A. W. Shum, Queueing Models for Computer Systems with General Service Time Distribution, Ph. D. Thesis, Harvard univ., Cambridge, Mass. 1976.
[9] R. Marie, An Approximate Analytical Method for General Queueing Networks, IEEE Trans. on Software Eng., vol. SE-5, no. 5, 1979.
[10] J. P. Buzen, Computational Algorithms for Closed Queueing Networks with Exponential Servers, Comm. ACM, vol. 16, no. 9, 1973.
[11] T. Czachórski, M. Kowalówka, Z. Szczerbiński, A. Wilk; Analiza pracy systemów komputerowych przy wykorzystaniu modeli statystycznych. Pakiet programowy AMOK, Politechnika Śląska, Gliwice /w druku/.
[12] A. Wilk, QNDEL - Język opisu sieci kolejek, Podstawy Sterowania, vol. 15, pp. 283-296, 1985.

mgr inż. JANUSZ MIERZWA
dr inż. JÓZEF IZYDORCZYK
mgr inż. ANDRZEJ KONOPACKI
mgr inż. ANNA MAŁESKA
dr inż. ANDRZEJ WARTAK
doc. dr inż. ADAM WOLISZ
Zakład Systemów Automatyki
Kompleksowej PAN
Gliwice

KONCENTRATOR — STACJA USŁUGOWA "SEZAM"

Koncepcja koncentratora

Aktualny stan rozwoju środków informatyki charakteryzuje się tym, że coraz częściej na niedużym obszarze /pojedynczy budynek biurowy, kompleks zabudowań uczelni/ instaluje się wiele systemów komputerowych - najczęściej mikrokomputerów - których zastosowania wymagają wzajemnej wymiany informacji i korzystania ze wspólnych danych. Ekonomicznie uzasadnione jest również wspólne wykorzystanie niewielkiej stosunkowo ilości drogich urządzeń peryferyjnych /pamięci o dużej pojemności, drukarki o dużej wydajności i wysokiej jakości wydruków, stoły kreślarskie/ przez liczne, względnie tanie mikrokomputery, wspomagające bezpośrednio poszczególne stanowiska pracy.

Ze względu na bardzo dużą różnorodność wymogów dotyczących odległości między mikrokomputerami, wymaganej przepustowości połączeń, oraz rodzajów przesyłanej informacji /dane, głos, obrazy/ powstała tendencja do stosowania dla poszczególnych klas zastosowań możliwie tanich rozwiązań, która spowodowała opracowanie licznych, różnorodnych produktów.

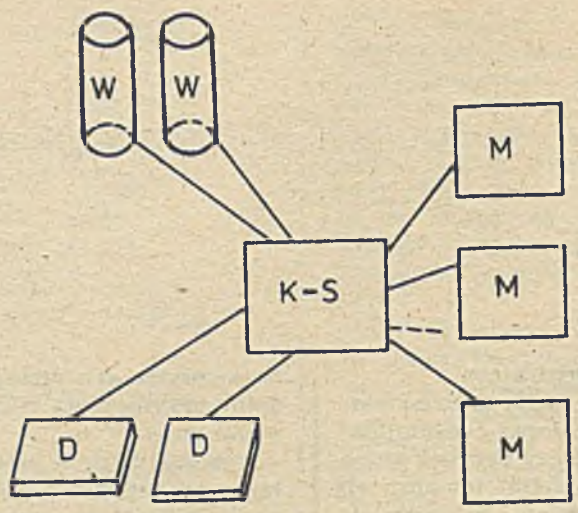
Koncentrator SEZAM jest specjalizowanym zestawem sprzętowo-programowym, umożliwiającym tworzenie struktur wielokomputerowych z heterogenicznych mikrokomputerów. Podstawowa konfiguracja przedstawiona na rys. 1 zakłada komunikację mikrokomputerów z koncentratorem poprzez łącza szeregowe o standardzie elektrycznym i mechanicznym, zgodnym ze standardem V24, ale ze zmodyfikowanym protokołem i podwyższoną, sięgającą do 150 kbit/s szybkością transmisji. Ponieważ wyjścia tego typu dostępne są w praktycznie wszystkich mikrokomputerach rozwiązanie takie nie wymaga żadnych adaptacji czy rozszerzeń sprzętowych w łączonych mikrokomputerach.

Przewidziano również możliwość tworzenia bardziej rozbudowanych przestrzennie struktur, łączących większą ilość mikrokomputerów oraz urządzeń zewnętrznych, co ilustruje rys. 2.

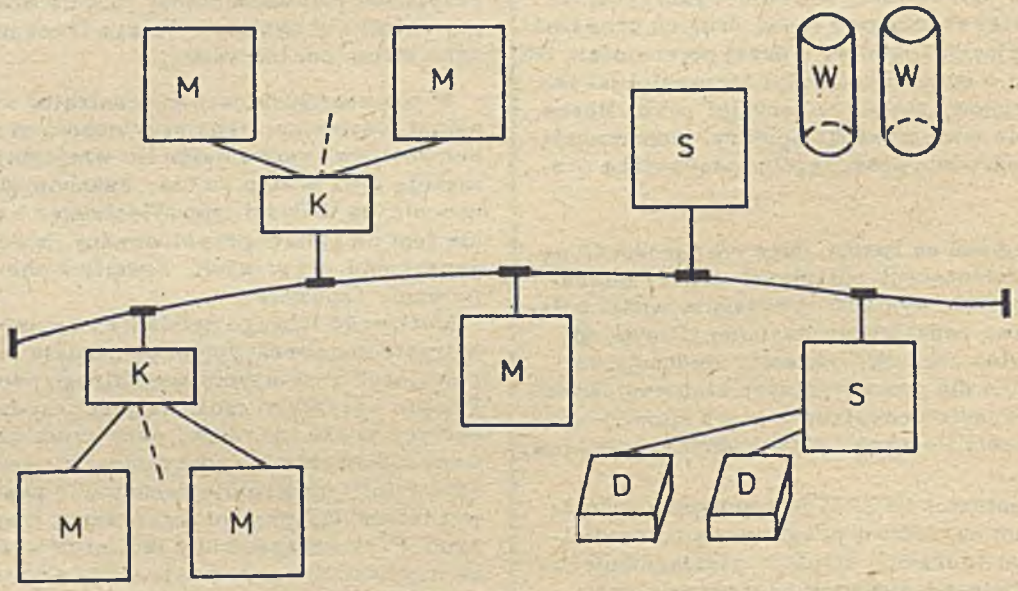
Komunikacja poprzez sieć lokalną wymaga wyposażenia w sterownik sieci /opcja aktualnie opracowywana/ i dlatego też przewiduje się dwa tryby przyłączania mikrokomputerów do magistrali sieci:
- tryb pośredni, w którym użytkownik, poprzez łącze szeregowe, połączony jest z koncentratorem wyposażonym we wzmiankowany sterownik,
- tryb bezpośredni, w którym mikrokomputer dysponuje indywidualnym sterownikiem; ten przypadek jest uzasadniony jedynie wówczas, gdy dołączany system wymaga transmisji dużych strumieni informacji.

W obu konfiguracjach koncentrator świadczy usługi jedynie na rzecz przyłączonych mikrokomputerów, zapewniając im wzajemną komunikację oraz dostęp do tzw. zasobów globalnych /obecnie są to dyski typu Winchester i drukarki/, nie jest natomiast przystosowany do realizacji programów użytkowych. Zrealizowane oprogramowanie zapewnia:
- Możliwość łatwego ustalania i modyfikacji w trybie konwersacyjnym przydziału zasobów globalnych przy użyciu specjalnego programu. Przede wszystkim możliwe jest przydzielanie partycji dysku twardego, przy czym zastosowane mechanizmy synchronizacji i kontroli uprawnień umożliwiają zachowanie prawidłowej informacji przy przetwarzaniu rozproszonym. Przy korzystaniu z drukarek wprowadzone mechanizmy gwarantują spójność wydruku.
- Możliwość korzystania z przydzielonych zasobów globalnych wg zasad obowiązujących w systemie operacyjnym mikrokomputera dla zasobów lokalnych; użycie zasobów globalnych nie wymaga więc dokonywania żadnych zmian w programach użytkowych.
- Możliwość tworzenia specjalnego oprogramowania ukierunkowanego na realizacje zadań w strukturze wielokomputerowej. Opracowano zbiór procedur, których dołączenie do oprogramowania użytkowego umożliwia przesył komunikatów między procesami obliczeniowymi, biegnącymi w różnych mikrokomputerach, a także programowe modyfikowanie przydziału zasobów globalnych.

Dostęp do usług świadczonych przez koncentrator możliwy jest przy dotrzymaniu przez dołączany mikrokomputer zbioru zdefiniowanych protokołów komunikacyjnych, ustalających warunki prawidłowego współdziałania z koncentra-



a/ Konfiguracja podstawowa



b/ Konfiguracja z siecią lokalną

Oznaczenia: D - drukarka
W - dysk twardy
K - koncentrator
S - stacja usługowa
M - mikrokomputer
K-S - koncentrator - stacja usługowa

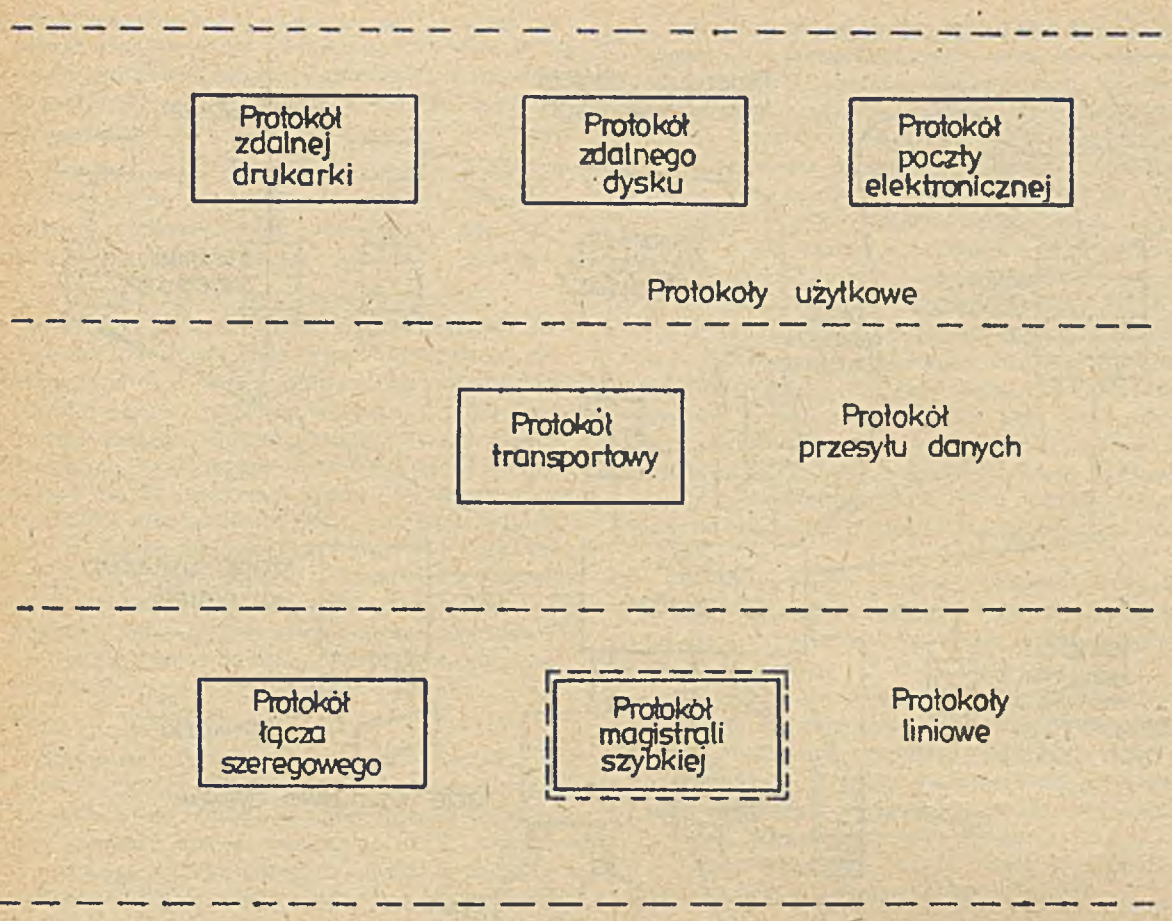
Rys. 1. Konfiguracje wielokomputerowe

torem, przede wszystkim formaty wymienianych jednostek informacji, reakcje na określone zdarzenia oraz niezbędne warunki czasowe.

Hierarchię protokołów przedstawiono na rys. 3. /linią przerywaną oznaczono protokoły, będące w fazie opracowywania/. Odzwierciedla ona fakt, iż pewne protokoły dla realizacji swoich funkcji wykorzystują usługi niższych w hierarchii protokołów.

Należy podkreślić, że protokoły liniowe wykorzystywane są wyłącznie dla potrzeb warstw oprogramowania komunikacyjnego, nie zapewniają bowiem dostępu do żadnej z usług koncentratora, a ich usługi nie są wprost wykorzystywane przez oprogramowanie użytkowe.

W dalszej części niniejszego artykułu przedstawione zostaną kolejno: zasady transmisji na łączach szeregowych, usługi udostępniane przez koncentrator oraz struktura oprogramowania zarówno koncentratora jak i dołączonych do niego mikrokomputerów.



Rys. 2. Hierarchiczna struktura protokołów komunikacyjnych

Program obsługi łącza szeregowego

Program obsługi łącza szeregowego realizuje przesył bezpołączeniowy z natychmiastowym potwierdzeniem poprawności transmisji. Wymiana informacji po łączach szeregowych bazuje na sprzężeniu mikrokomputerów z koncentratorem według elektrycznego i mechanicznego standardu V24 z wykorzystaniem linii /z punktu widzenia koncentratora/: 102, 103, 104, 105, 106, 107, 108.2, 114. Zastosowany protokół łącza szeregowego umożliwia inicjowanie wymiany informacji, tj. przesłanie pojedynczego pakietu zarówno przez koncentrator jak i mikrokomputer w dowolnym momencie czasu. Przesył pakietu realizowany jest w 5 fazach:

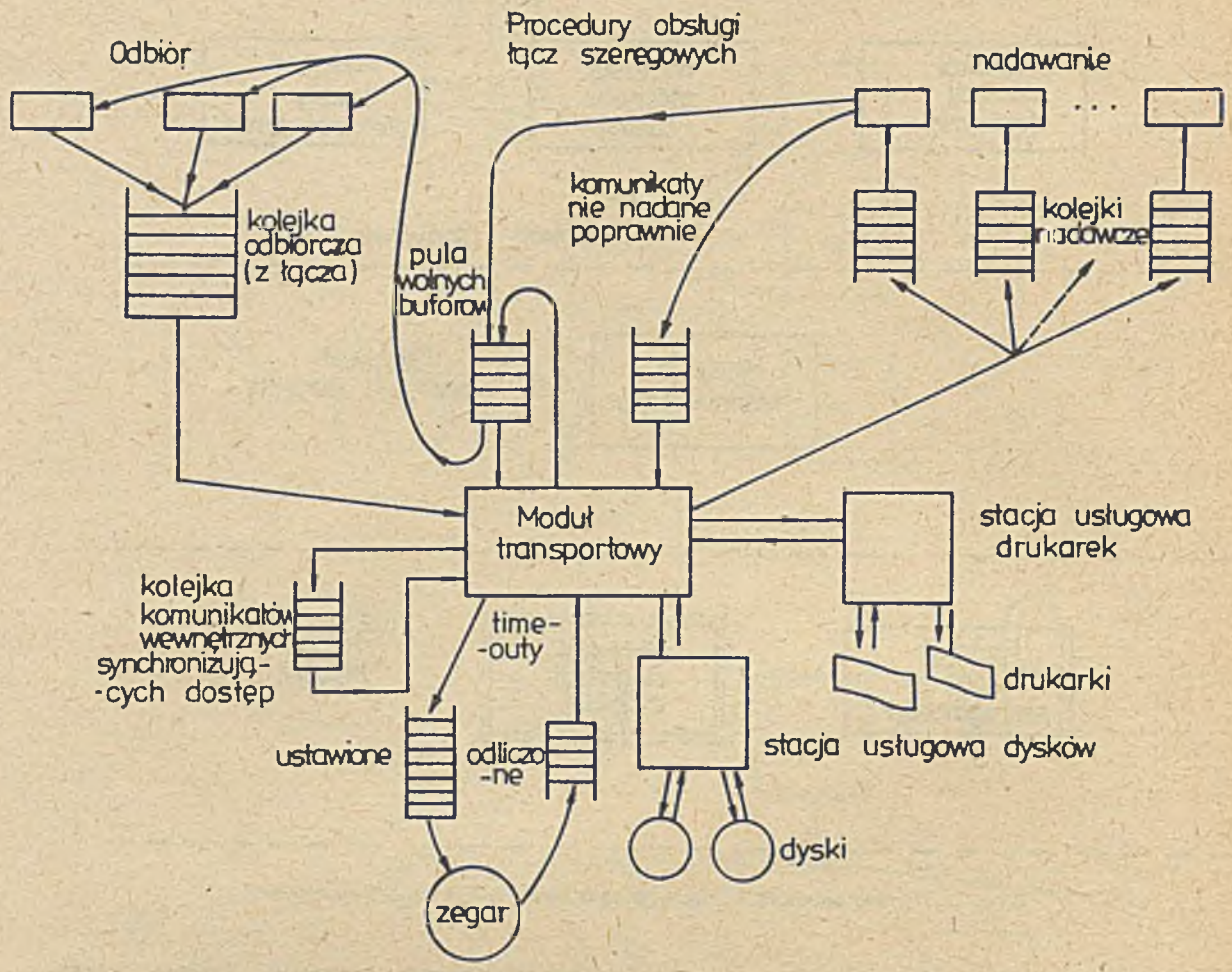
1. Faza rezerwacji łącza. Łącze wykorzystywane jest w trybie "half duplex". rezerwacji dokonuje się poprzez ustawienie na 1 linii: 108.2 przez stronę inicjującą nadawanie /o ile linia nie jest zajęta - w identyczny sposób - przez partnera/, oraz sprawdzenie, po krótkiej zwłoce, czy nie nastąpiła równoczesna rezerwacja przez partnera. W przypadku równoczesnej rezerwacji obaj partnerzy zwalniają łącze i ponawiają próbę rezerwacji po ustalonej, różnej dla koncentratora i mikrokomputerów, zwłoce. Sygnał rezerwacji podtrzymywany jest przez cały czas aktywności partnera nadającego i stanowi warunek odbioru znaków z linii danych.

2. Faza inicjowania transmisji. Partner inicjujący nadawanie wysyła - w trybie asynchronicznym - pojedynczy bajt o ustalonej zawartości i przechodzi do oczekiwania na zgodę na nadawanie, tj. odbiór pojedynczego bajtu /możliwe jest też otrzymanie odmowy/. Nadmiernie długie oczekiwanie powoduje sygnalizację niesprawności partnera /lub łącza/ i zwolnienie łącza.

3. Faza transmisji. Nadawanie pakietu przebiega w trybie asynchronicznym lub synchronicznym /zależnie od wersji programu obsługi łącza/, być może z szybkością różną od szybkości używanej w fazie inicjacji. Pakiet zawiera informacje o swej długości /w bajtach/ oraz dwubajtową sumę kontrolną. Nadmiernie długie oczekiwanie na nadanie lub odbiór kolejnego bajtu są sygnalizowane i powodują zwolnienie łącza.

4. Faza potwierdzenia. Po zakończeniu transmisji partner nadający oczekuje na potwierdzenie zakończenia odbioru, co sygnalizuje pojedynczy bajt, wysyłany w trybie asynchronicznym z prędkością taką jak w fazie inicjowania; zawartość bajtu świadczy o poprawnym /tj. bez błędu sumy kontrolnej/ lub błędnym odbiorze. Nadmiernie długie oczekiwanie na potwierdzenie powoduje sygnalizację niesprawności i zwolnienie łącza.

5. Faza zwolnienia łącza - po poprawnej trans-



Rys. 3. Ogólny schemat przepływu komunikatów w koncentratorze SEZAM

misji lub zbyt dużej liczbie ponawianych niepoprawnych transmisji.

Wyodrębnienie szybkości i trybu przesyłu w fazie transmisji wynika z dążenia do wykorzystania maksymalnych parametrów układów transmisji szeregowej dostępnych w poszczególnych mikrokomputerach. Np. popularny element I8251 pracuje poprawnie do 20 kbit/s w trybie asynchronicznym, ale aż do 64 kbit/s w trybie synchronicznym i takie parametry były wykorzystane w poszczególnych fazach. Element SIO rodziny Zilog używany był natomiast we wszystkich fazach w trybie pracy asynchronicznej z szybkością 153 kbit/s.

Biorąc pod uwagę sygnalizowaną przez użytkowników potrzebę zmniejszenia szybkości na niektórych łączach na rzecz wydłużenia kabla przyjęto, iż szybkość transmisji będzie zadawana indywidualnie dla każdego łącza /w zasadzie poprzez sygnał zegarowy z mikrokomputera/ i będzie identyczna dla torów nadawania i odbioru.

Usługi transportowe koncentratora

Adresy transportowe przydzielane są poszczególnym procesom obliczeniowym, rezydującym w koncentratorach /np. realizującym usługi dostępu do dysku i drukarki/ lub procesom rezydującym w poszczególnych mikrokomputerach. Adresacja ta jest globalna, tzn. w ramach konfiguracji obejmującej sieć lokalną adresy są niepowtarzalne, a usługi dostępu do dysku i drukarki świadczone są pod wyróżnionymi adresami.

Usługi transportowe polegają na marszrutowaniu komunikatów między adresami: źródłowym i przeznaczenia. Przesył komunikatów między koncentratorem a mikrokomputerami odbywa się poprzez wykorzystanie usług odpowiedniego protokołu liniowego. Jeśli protokół liniowy stwierdzi brak warunków dla prawidłowego nadania komunikatu z koncentratora /do mikrokomputera lub w sieć lokalną/ sygnalizuje wówczas ten stan transportowi, a ten generuje specjalny komunikat /error indication/ do procesu źródłowego, o ile proces rezyduje w mikrokomputerze podłączonym do tego koncentratora w którym wykryto niesprawność łącza. Przesył komunikatów między adresami przeznaczenia nie jest przez protokół transportowy potwierdzany. Brak również wstępnego uzgad-

niania warunków i zasad wymiany informacji. Usługi transportowe świadczone są więc w trybie bezpołączeniowym i bezpotwierdzeniowym sygnalizowany jest jedynie brak możliwości nadania komunikatu z koncentratora źródłowego.

Ewentualne potwierdzenie odbioru komunikatów należy do protokołu wymiany informacji między użytkownikami /procesami/ oraz do modułów zdalnego dysku i zdalnej drukarki.

Dostęp do zdalnego dysku

Usługi tego typu polegają na udostępnianiu fragmentów dysku twardego, dołączonego do koncentratora. Korzystanie z fragmentu dysku odbywa się na zasadach wyłączności lub dzielonego dostępu. Rezydujący w koncentratorze program, realizujący dostęp do dysku, nosi nazwę stacji usługowej dysku. Dysk twardy jest w trakcie wstępnej operacji formatowania dzielony na wiele fragmentów /partycji/, traktowanych jako odrębne dyski wirtualne /logiczne/. Operacja formatowania wykonywana jest przez odrębny, specjalizowany program i polega m. in. na zapisaniu danych o strukturze dysku twardego na jednej z zarezerwowanych ścieżek tego dysku.

Stacja usługowa nie jest świadoma jakiejkolwiek struktury plików czy kartotek zakładanych na dyskach wirtualnych /zlecenia dostępu muszą podawać wprost adresy ścieżek i sektorów/ i zarządza korzystaniem z dysku wyłącznie z dokładnością do dysku wirtualnego. Dyski wirtualne identyfikowane są za pomocą numeru /od 0 do N<64/ lub przez 5-znakową nazwę, rozpoczynającą się literą. Dla każdego dysku wirtualnego istotne są następujące informacje:
- Warunki organizowania dostępu - wyróżniono dwie kategorie dysków wirtualnych:
  - PUBLIC - przydzielanych do odczytu jednocześnie wielu użytkownikom, a do zapisu w danej chwili tylko jednemu,
  - PRIVATE - przydzielanych do zapisu i do odczytu tylko jednemu użytkownikowi w danej chwili.
- Sposób ochrony dysku przed nielegalnym dostępem - zastosowano hasło ochrony, które służy do zidentyfikowania uprawnionego użytkownika /przydział dysku kategorii PUBLIC wyłącznie do odczytu nie wymaga znajomości hasła/.
- Rodzaj systemu operacyjnego dla jakiego dysk wirtualny został wstępnie przygotowany. Wprowadzenie tego parametru wynika stąd, że w różnych systemach stosowane są różne znaczniki pustej kartoteki i różne sposoby podziału dysku na ścieżki systemowe, obszar kartoteki i obszar danych. Znajomość tego parametru jest niezbędna dla dostosowania się użytkownika do wymagań, narzuconych przez określony system operacyjny. Rozwiązanie takie pozwala np. użytkownikowi pracującemu aktualnie pod systemem CP/M korzystać z dysku wirtualnego, przeznaczonego dla DOS'a, przez zastosowanie odpowiednich procedur konwersji formatów.
- Parametry fizyczne /np. pojemność dysku/, których znajomość jest niezbędna użytkownikowi dla dobrania odpowiedniej systemowej tablicy opisu dysku.

Na oprogramowanie stacji usługowej dysku składają się dwa odrębne moduły programowe, obsługujące dwie grupy zleceń:
- zlecenia konfiguracji służące do uzyskania uprawnień do korzystania z dysku zdalnego lub zmiany parametrów tego dysku:
- zlecenia dostępu /zapis, odczyt/ z kontrolą uprawnień do wykonania tych funkcji.

Każdy z tych modułów odbiera żądania realizacji usług i wysyła odpowiedzi, korzystając z innego adresu transportowego. Listę usług związanych z dostępem do dysku przedstawiono w tabelach.

Dostęp do zdalnej drukarki

Usługi dotyczące udostępniania zdalnej drukarki umożliwiają użytkownikom uzyskanie dostępu w trybie bezpośrednim /na zasadzie wyłączności/ lub w trybie zbuforowanym do drukarek sprzężonych ze wskazanym koncentra-

Zestawienie usług konfiguracji — Tabela 1

Zlecenie	Parametry	Uwagi
Przydziel dysk CONNECT	- nazwa lub numer dysku wirtualnego - tryb korzystania /R/W lub R/O/ hasło ochrony	- Dyski PRIVATE zawsze R/W - Dyski PUBLIC - hasło nieistotne przy trybie R/O
Odłącz dysk DISCONNECT	- numer dysku wirtualnego	
Zmień nazwę NEWNAME	- numer dysku wirtualnego nowa nazwa	zmiana nazwy następuje tylko wtedy, gdy dysk jest przydzielony w trybie R/W
Zmień hasło NEWPASSWORD	- numer dysku wirtualnego - nowe hasło	j. w.
HELP	Odpowiedzią na to zlecenie jest lista parametrów kolejnych dysków wirtualnych	

Tabela 2

Zestawienie usług dostępu

Zlecenie	Parametry	Uwagi
Odczyt sektora READ	- numer dysku wirtualnego - numer ścieżki dysku wirtualnego /od 0/ - numer sektora na ścieżce /od 0/	- sektor długości 512 bajtów - sprawdzenie uprawnień przed wykonaniem operacji /czy dysk przydzielony i używany we właściwym trybie/
Zapis sektora WRITE	- parametry j. w.	

torem, a ściślej do zarzadzajacego tym dostępem procesu nazywanego stacją usługową drukarki. Korzystanie ze zdalnej drukarki w trybie bezpośrednim wymaga uprzedniego jej zarezerwowania. Rezerwacja jest uwzględniana o ile drukarka jest jeszcze wolna, w przeciwnym wypadku jest odrzucana /zlecenia rezerwacji nie są kolejkowane/.

Stacja usługowa drukarek działająca w trybie bezpośrednim kieruje bezpośrednio do wydruku kolejne łańcuchy znaków przesłane z mikrokomputera przy użyciu procedur transportowych, nie ingerując w zawartość przekazanego łańcucha znaków /w aktualnej wersji długość łańcucha nie może przekroczyć 528 znaków/. Kolejny łańcuch znaków może być przesłany przez zdalnego użytkownika dopiero po otrzymaniu potwierdzenia obsłużenia poprzedniego łańcucha. Po zakończeniu całości wydruku użytkownik powinien zwolnić rezerwację drukarki, aby udostępnić ją innym użytkownikom.

W ramach protokołu zdalnej drukarki sygnalizowany jest użytkownikom fakt zarezerwowania drukarki, jej niesprawność oraz błędy formalne w formatach zleceń. Korzystanie ze zdalnej drukarki w trybie zbuforowanym umożliwia jej równoczesne przydzielenie kilku użytkownikom. Kolejne, pochodzące od nich łańcuchy znaków zapisywane są w odpowiednio utworzonych na dysku twardym plikach. Dopisywanie kolejnych fragmentów tekstu do pliku wykonywane jest do chwili otrzymania wyróżnionego znacznika końca tekstu. Plik zostaje wówczas uznany za gotowy do wydruku i zamknięty. Stacja usługowa, działając w trybie z buforowaniem, drukuje te pliki tekstowe w kolejności ich zamykania.

Sprzęt i oprogramowanie koncentratora

Koncentrator SEZAM w części urządzeniowej został opracowany na bazie modułowego mikrokomputera Mister Z-80. Rozwiązanie takie umożliwiło wykorzystanie znajdujących się w produkcji sprawdzonych elementów sprzętu, oraz zapewnia możliwość konfigurowania koncentratora /np. liczby łącz szeregowych, liczby drukarek/ zgodnie z indywidualnymi wymaganiami użytkowników. Opracowany został, na etapie modelu, również drugi wariant, w którym minimalna konfiguracja, zapewniająca udostępnianie dysków i dwóch drukarek czterem mikrokomputerom ma postać systemu, zrealizowanego na pojedynczej karcie formatu Stańsaab, z możliwością rozbudowy o dalsze karty, zwiększające liczbę użytkowników oraz zapewniające dołączenie dalszych zasobów i adaptera sieci lokalnej. Mimo znacznie niższych kosztów wykonania konieczność uruchomienia produkcji nowego wyrobu, jego badań, serwisu itp. przesądziła o zaniechaniu rozwoju tego wariantu.

Obecnie koncentrator zrealizowano w oparciu o wersje jednoprocesorowe, jednak architektura systemu Mister Z-80, umożliwiająca realizacje systemów wieloprocesorowych, stwarza możliwość podniesienia wydajności SEZAMu poprzez rozłożenie jego funkcji na 2 - 3 procesory. Struktura programowa, której zarys przedstawiono niżej, sprzyja również ewentualnej modyfikacji.

Na oprogramowanie koncentratora składa się zbiór współbieżnie wykonywanych modułów programowych sprzężonych i synchronizujących swój bieg poprzez system kolejek. Do modułów tych należy zaliczyć: procedury nadawania i odbioru pakietów przez łącza szeregowe /łącze sieci lokalnej, moduły realizujące funkcje stacji usługowej dysków oraz usługowej drukarek, moduł obsługi zegara, a także odgrywający rolę specjalną, moduł transportowy. Ponieważ podstawowym kryterium leżącym u podstaw projektu oprogramowania koncentratora było dążenie do minimalizacji narzutów na realizację usług /a specjalnie dostępu do zasobów/ zrezygnowano, po dłuższej analizie, z użycia dla organizacji pracy koncentratora, któregoś ze standardowych systemów operacyjnych, na rzecz rozwiązania specjalizowania.

Opracowany zbiór procedur umożliwia tworzenie i operacje na listach połączonych /kolejkach/, natomiast aktywizacja poszczególnych modułów następuje przez przerwanie zewnętrzne /od łącza/ lub z inicjatywy modułu transportowego /ogólny schemat przepływu informacji w koncentratorze przedstawia rys. 3/, który określa każdorazowo drogę przepływu informacji przez koncentrator.

Należy podkreślić, że dla uproszczenia gospodarki pamięcią koncentratora przyjęto jej podział na bufory o stałej wielkości, równej maksymalnej długości przekazywanej informacji, a przyjęcie - poprzez łącze - żądania wykonania usług koncentratora stanowi równocześnie rezerwację bufora, w którym zapamiętywane będą wszystkie informacje w trakcie wykonywania usługi /np. odczytany przez stację dyskową sektor/. Ograniczenia na maksymalną liczbę buforów, przyznawanych wybranym torom przepływu informacji stanowią efektywne narzędzie sterowania przepływem.

Zrealizowane oprogramowanie zapewnia równoległość pracy urządzeń dołączonych do koncentratora, np. pozycjonowania głowicy dysku i obsługę łącz szeregowych.

Oprogramowanie mikrokomputerów dołączonych do koncentratora

Korzystanie z usług oferowanych przez koncentratory wymaga stosowania odpowiedniego oprogramowania sieciowego dołączonych do nich mikrokomputerów. Oprogramowanie sieciowe mikrokomputerów składa się z odpowiednio zmodyfikowanego systemu operacyjnego oraz procedur i programów użytkowych. Modyfikacja systemu operacyjnego polega na dołączeniu dodatkowych procedur obsługi /driver'ów/ łącza szeregowego, dysków oraz drukarek.

Zadaniem procedury obsługi łącza szeregowego jest realizacja protokołu liniowego /przesył bloków danych między mikrokomputerem a koncentratorem/, natomiast procedury obsługi zdalnych dysków i drukarek buforują związane z odczytem/zapisem sektorów na dysku lub zapisem znaków na drukarce a następnie, korzystając z danych o aktualnym przydziale urządzeń zdalnych /zawierających m. in. adresy stacji usługowych/, wysyłają przez łącza szeregowe zlecenia realizacji odpowiednich usług dostępu do zasobów zdalnych.

W skład procedur użytkowych wchodzą procedury transportowe, procedury przydziału zasobów zdalnych oraz procedury poczty elektronicznej. Zadaniem procedur transportowych jest realizacja protokołu transportowego, umożliwiającego przesył bloków danych między procesami zdalnymi. Procedury przydziału zasobów zdalnych umożliwiają m. in. : przyłączanie i odłączanie drukarek, dysków zdalnych oraz sprawdzanie aktualnej konfiguracji urządzeń zdalnych i stanu przydzielonych zasobów. Procedury te korzystają z danych zawartych w procedurach obsługi urządzeń zdalnych oraz, poprzez procedury transportowe, z usług konfiguracji świadczonych przez stacje usługowe. W przypadku zmiany przydziału zasobów zdalnych modyfikowane są dane zawarte w procedurach obsługi urządzeń zdalnych. Procedury poczty elektronicznej pozwalają wyświetlić dowolny tekst na wydzielonym polu ekranu zdalnego mikrokomputera. Dedykowany program użytkowy umożliwia uzyskanie w trybie konwersacyjnym informacji o aktualnym przydziale zasobów oraz dokonywanie zmian w tym przydziale.

Przyjęcie powyższej struktury oprogramowania umożliwia niezmodyfikowanym programom użytkowym korzystanie z uprzednio przydzielonych urządzeń zdalnych, natomiast programy z dołączonymi procedurami sieciowymi mogą dodatkowo dynamicznie zmieniać przydział zasobów, a także korzystać ze środków komunikacji międzyprocesowej. Oprogramowanie sieciowe zrealizowane zostało dla mikrokomputerów pracujących pod systemem operacyjnym CP/M i jest opracowywane dla systemów MP/P i PC-DOS.

W przypadku systemu CP/M sieciowe oprogramowanie systemowe jest logicznie częścią procedur BIOS, a procedury obsługi łącza szeregowego mają ustalone odskoki i adresy danych. W tym przypadku opracowano także specjalny program konfiguratora przydziału zasobów zdalnych, który pozwala w trybie konwersacyjnym, realizować funkcje procedur użytkowych /przydzielanie/zwalnianie zasobów zdalnych, wyświetlanie stanu przydziału zasobów, wysyłanie komunikatów/ oraz dodatkowo, korzystać z wielodostępnej bazy danych.

Korzystać z usług koncentratora - stacji usługowej SEZAM mogą mikrokomputery różnych typów, pod warunkiem dokonania odpowiednich modyfikacji i uzupełnień ich oprogramowania. Zmiany takie i modyfikacje zostały opracowane dla systemu operacyjnego CP/M-80 i zaimplementowane na mikrokomputerach ComPAN, Mister Z-80 oraz ComPAN-P /implementacja na innych mikrokomputerach pracujących pod systemem CP/M-80 byłaby również prosta/. Opracowane rozwiązanie jest zadowalające również pod względem efektywnościowym i tak np. ładowanie programów ze zdalnego dysku twardego odbywa się z szybkością tylko o około 40% niższą niż w przypadku lokalnego dysku twardego.

Obecnie dobiegają końca prace nad dołączeniem do koncentratora SEZAM mikrokomputerów zgodnych z IBM PC XT/AT pracujących pod systemem DOS. Koncentrator SEZAM umożliwia transmisje po łączach szeregowych z szybkością do 150 kbit/s. Ograniczenia konstrukcyjne wyjść szeregowych w dołączanych mikrokomputerach mogą wymusić stosowanie w ich przypadku szybkości niższych np. dla mikrokomputerów zgodnych z IBM PC szybkość maksymalna wynosi 115 kbit/s, a dla mikrokomputerów wykorzystujących elementy I8251 szybkość maksymalna w trybie synchronicznym sięga tylko 64kbit/s.

Wykorzystanie pełnej szybkości 150 kbit/s możliwe jest w przypadku mikrokomputerów, używających układów transmisji szeregowej o wyższych parametrach, np. popularnych elementów SIO firmy Zilog.

Koncentrator - stacja usługowa SEZAM wdrażany jest do produkcji przez Zakłady Elektroniki Górniczej w Tychach.

mgr inż. JERZY GRZYWOCZ
dr inż. STANISŁAW KOZIELSKI
Instytut Informatyki
Politechnika Śląska

OPROGRAMOWANIE UMOŻLIWIAJĄCE WYMIANĘ ZBIORÓW DANYCH POMIĘDZY BAZAMI DANYCH MIKROKOMPUTERA COMPAN-8 I KOMPUTERA ODRA 1305

Artykuł prezentuje oprogramowanie opatrzone nazwą "System TRANS", umożliwiające wymianę danych między bazą danych dBASE II mikrokomputera ComPAN, a bazą danych DBMS komputera ODRA 1305. Systemy zarządzania obu baz są systemami typu relacyjnego. Wystarczającą ilustracja pojęcia relacji jest tabela, której kolumny identyfikowane są nazwami atrybutów, zaś wiersze /krotki/ utożsamiane są z rekordami.

Oba systemy umożliwiają manipulowanie danymi za pomocą języka opartego na algebrze relacji. Ważniejsze cechy obu baz omówione zostały w rozdziałach 2 i 3 niniejszego artykułu. Spośród najważniejszych celów integracji omawianych baz danych wymienimy następujące:

- Z punktu widzenia użytkownika systemu ComPAN: możliwość przechowywania plików danych bazy dBASE II w pamięciach zewnętrznych /dyski, taśmy/ systemu ODRA, zapewniających znacznie większe bezpieczeństwo przechowywania oraz większą pojemność zbiorów.
- Z punktu widzenia użytkownika systemu DBMS na ODRZE 1305: możliwość bardzo wygodnego wprowadzania i edycji danych przy wykorzystaniu różnorodnych mechanizmów /np. całoekranowego trybu pracy/ systemu dBASE II.

Uwagi te dotyczą celów najistotniejszych, bowiem obie bazy danych pozwalają na dowolną manipulację przechowywanymi danymi bez względu na źródło ich pochodzenia. Należy dodać, że integracja baz danych stwarza pewne ograniczenia w stosunku do autonomicznego wykorzystania obu baz. Zostaną one omówione w następnym rozdziale niniejszego artykułu.

Do eksploatacji systemu TRANS konieczne jest, poza podłączeniem mikrokomputera ComPAN 8 do systemu ODRA 1305, zainstalowanie na mikrokomputerze ComPAN oprogramowania, umożliwiającego wykorzystanie go jako końcówki ODRY. Podkreślić należy, że w prezentowanym systemie nie ustala się żadnej relacji nadrzędności łączonych baz danych, ani też kolejności wykonywania translacji. Niemniej przewidując bardziej prawdopodobny sposób wykorzystania systemu, ukierunkowano w niektórych punktach jego opis na taki wariant pracy, kiedy użytkownik dBASE II pragnie przechować swoje pliki danych w systemie ODRA. Należy wspomnieć, że do realizacji takiego celu może okazać się wystarczajace zwyczajne przechowanie przesłanych danych w pamięci zbiorów /PZS/ systemu GEORGE 3, bez angażowania /i instalowania/ bazy danych DBMS. Ten rodzaj pracy przedstawiony zostanie w rozdziale "Możliwości funkcjonalne oprogramowania łączącego obie bazy. Przebieg transmisji".

Cechy charakterystyczne systemów zarządzania bazami danych dBASE i DBMS

Cechy obu systemów można przedstawić porównując te systemy. Porównane zostaną te ich wersje, które zostały zintegrowane przez autorów artykułu, a więc dBASE II i DBMS ISSUE 1. Oba systemy można porównać ze względu na następujace parametry:

- ograniczenia na parametry relacji,
- możliwości operowania na danych,
- możliwości pracy interakcyjnej i wsadowej,
- możliwości współpracy z innymi programami
- ochronę danych.

Ograniczenia na parametry relacji

Ilość danych możliwych do umieszczenia w relacji limitowana jest rodzajem pamięci zewnętrznej, w której znajduje się relacja. System dBASE II na mikrokomputerze ComPAN 8 zapisuje relacje na dyskietkach o pojemności 120 Kbajtów, natomiast system DBMS korzysta ze zbiorów PZS systemu GECRGE 3, mogących pomieścić 245 K słów /24-bitowych/. W systemie DBMS deklarowana jest maksymalna liczba krotek w relacji, w systemie dBASE maksymalna liczba krotek jest stała i wynosi 65534. W systemie dBASE II relacje mogą zawierać 32 atrybuty, natomiast w systemie DBMS tylko 24. W obu systemach atrybuty mogą przybierać /w zależności od zadeklarowania/ wartości będące ciągami znaków, liczbami całkowitymi lub rzeczywistymi. W systemie dBASE atrybuty mogą przybierać również wartości logiczne. W systemie DBMS nie można jawnie zadeklarować typu logicznego.

Dziedziny dla atrybutów tekstowych i numerycznych są znacznie szersze w systemie

DBMS, z jednym wyjątkiem: zbiór znaków akceptowanych przez system dBASE zawiera duże i małe litery, natomiast system DBMS akceptuje tylko duże. System DBMS wymaga zdefiniowania pól kluczowych w relacji i nie dopuszcza do umieszczenia w relacji dwóch krotek o identycznym kluczu. W systemie dBASE II nie istnieje pojęcie pól kluczowych przypisanych do relacji. Stąd relacja w tym systemie może zawierać dwie krotki z równymi wartościami wszystkich pól.

Możliwości operowania na danych

Operacje na danych w relacyjnych bazach danych można podzielić na kilka klas:
- operacje na schemacie relacyjnej bazy danych obejmujące:
  - zakładanie relacji,
  - kopiowanie relacji,
  - usuwanie istniejących relacji,
  - przemieszczanie relacji.
- Operacje aktualizacji danych w relacjach obejmujące:
  - dopisywanie nowych rekordów,
  - modyfikację istniejących rekordów,
  - usuwanie rekordów z relacji.
- Operacje z zakresu algebry relacji; najważniejsze z nich są operacje:
  - selekcji,
  - projekcji,
  - łączenia.
- Operacje raportowania danych.

Biorąc pod uwagę wbudowane w systemy dBASE i DBMS mechanizmy operowania na danych, można uważać oba systemy w zakresie operacji relacyjnej bazy danych za równoważne. Możliwości aktualizacji danych są znacznie bogatsze w systemie dBASE. Równoważność obu systemów należy uznać również w klasie trzech podstawowych operacji z zakresu algebry relacji, chociaż system dBASE II nie realizuje pełnowartościowej operacji projekcji /realizujące projekcję zlecenie COPY nie usuwa z relacji wynikowej ewentualnych duplikatów, występujących w niej krotek/. W zakresie raportowania danych znacznie mocniejsze mechanizmy ma wbudowane system dBASE II.

Możliwości pracy interakcyjnej i wsadowej

Można zdefiniować trzy tryby współpracy z programem:
- tryb interakcyjny, w którym program realizuje pojedyncze zlecenia użytkownika, zgłaszając swoją gotowość do dialogu po wykonaniu każdego ze zleceń,
- tryb wsadowy wewnętrzny, w którym użytkownik po przejściu do dialogu z programem powoduje wykonanie przez program pewnej sekwencji zleceń /makrozlecenie/,
- tryb wsadowy zewnętrzny, w którym użytkownik powoduje wykonanie przez program pewnego zlecenia lub sekwencji zleceń, nie przechodząc do dialogu z programem.

Współpraca z systemem dBASE II możliwa jest we wszystkich trzech trybach, natomiast z systemem DBMS tylko z dwóch pierwszych.

Możliwości współpracy z innymi programami

Należy rozważyć cztery aspekty współpracy systemów dBASE i DBMS z innymi programami:
- możliwości uruchomienia systemu z poziomu innego programu,
  możliwości uruchomienia innych programów z poziomu systemu,
- możliwości przetwarzania w systemie danych wytworzonych przez inny program,
- możliwości przetwarzania przez inny program danych pochodzących z systemu /w szczególności relacji/.

System dBASE zapewnia wszystkie wymienione możliwości współpracy z innymi programami, natomiast w systemie DBMS nie jest możliwe uruchamianie innych programów z poziomu konwersacji z systemem.

Ochrona danych

Problem ochrony danych można rozważać w dwóch aspektach:
- ochrony danych przed niepowołanym dostępem,
- ochrony danych przed zniszczeniem na skutek zdarzeń losowych.

Dane zgromadzone w relacjach systemu DBMS chronione są przed niepowołanym dostępem. Relacje reprezentowane są bowiem przez zbiory dyskowe systemu GEORGE 3. Zbiory te figurują w kartotekach użytkowników, a dostęp do tych kartotek i zbiorów w nich opisanych chroniony jest przez hasło. System dBASE II nie dysponuje programowym zabezpieczeniem danych przed niepowołanym dostępem. Ochrona zbiorów danych przed zniszczeniem realizowana jest poprzez tworzenie ich kopii.

W systemie DBMS zbiory dyskowe, reprezentujące relacje podlegają okresowemu składowaniu na taśmę magnetyczną, na takich samych zasadach jak wszystkie inne zbiory systemu GEORGE 3. Trwałość taśmy magnetycznej zmniejsza do minimum prawdopodobieństwo bezpowrotnej utraty danych. Składowanie zbiorów na taśmę jest inicjowane przez obsługę techniczną systemu GEORGE 3, co zwalnia użytkownika z uciążliwego obowiązku samodzielnego tworzenia dodatkowych kopii danych. W systemie dBASE II dla mikrokomputera ComPAN 8 nie ma systemowych mechanizmów tworzenia rezerwowych kopii danych. Użytkownik musi tworzyć takie kopie samodzielnie na dodatkowych dyskietkach.

Ograniczenie w zakresie integracji baz danych dBASE II i DBMS /ISSUE 1/

Ograniczenia w zakresie integracji obu baz danych wynikają z różnych ograniczeń na parametry relacji w każdej z tych baz oraz z niemożliwości zakładania nowej relacji w trybie wsadowym w systemie dBASE II.

Przesył relacji z systemu dBASE II do systemu DBMS podlega następującym ograniczeniom:
- przesyłana relacja nie może mieć więcej niż 24 atrybuty,
- małe litery występujące jako wartości atrybutów tekstowych nie mogą mieć istotnego znaczenia /zostaną zamienione na duże/,
- nazwy atrybutów, liczące więcej niż 8 znaków i nie zawierające znaku dwukropka nie mogą zostać przeniesione do systemu DBMS w swojej pierwotnej postaci,
- wartości typu logicznego muszą ulec konwersji na postać tekstową,
- liczba przesyłanych krotek nie może spowodować przepełnienia relacji w systemie DBMS /liczba krotek w tej relacji nie może przekroczyć zadeklarowanej wartości maksymalnej/.

Ograniczenia nałożone na przesyłanie relacji z systemu DBMS do systemu dBASE II są następujące:
- relacja docelowa przesyłu w systemie dBASE II musi istnieć /nie można założyć relacji w systemie dBASE II w trybie wsadowym/,
- wartości atrybutów nie mogą przekraczać ograniczeń na wartości atrybutów w systemie dBASE /łańcuchy znaków nie dłuższe niż 254 znaki, liczby rzeczywiste nie większe niż 1.8E63/,
- przesyłane krotki nie mogą być dłuższe niż 1000 znaków.

Możliwości funkcjonalne oprogramowania łączącego obie bazy. Przebieg transmisji

Koncepcję integracji obu baz danych oparto na założeniu, że dzięki odpowiedniemu połączeniu oraz wykorzystaniu odpowiedniego oprogramowania mikrokomputer ComPAN może spełniać rolę konwersacyjnej końcówki komputera ODRA 1305. Użytkownik ma wtedy możliwość dowolnego wyboru środowiska programowego /system operacyjny CP/M wraz z bazą dBASE II lub system operacyjny GEORGE 3 wraz z bazą DBMS/ bez zmiany urządzenia, na którym pracuje.

Wymiana danych między relacją systemu DBMS i plikiem systemu dBASE uwarunkowana jest zgodnością struktur obu zbiorów. Struktury te są zgodne przy spełnieniu ograniczeń przedstawionych w poprzednim rozdziale niniejszego artykułu. Nie wszystkie spośród tych ograniczeń są przy tym krytyczne, tzn. takie, że ich przekroczenie uniemożliwia wymianę danych. Kontrola struktur zbiorów wymaga przesyłania wraz z danymi również opisu struktury. Przyjęto, że przesyłany opis będzie zgodny z postacią wydruku struktury pliku w systemie dBASE II, niezależnie od kierunku transmisji.

Różny sposób przechowywania danych w obu bazach powoduje konieczność konwersji danych do postaci pośredniej. Wszystkie dane reprezentowane są wtedy w postaci znakowej. Zbiór przesyłanych informacji składa się w związku z tym z dwóch części:
- opisu struktury,
- właściwych danych.

Każda wymiana danych między obiema bazami składa się z trzech etapów.
1. Utworzenie opisu struktury zbioru; wyprowadzenie danych z jednej bazy i ich konwersja do postaci pośredniej.
2. Przesył zbioru zawierającego opis struktury i dane,
3. Kontrola zgodności struktur zbiorów; konwersja danych z postaci pośredniej i wprowadzenie ich do drugiej bazy.

W przypadku przesyłu danych do systemu DBMS możliwe jest wykonanie dodatkowej operacji, a mianowicie automatyczne utworzenie potrzebnej relacji na podstawie przesłanego opisu struktury. Towarzyszące temu opisowi dane są wtedy wprowadzane do nowo utworzonej relacji. Oprogramowanie systemu TRANS realizuje wszystkie spośród wymienionych operacji, za wyjątkiem etapu /2/. Sam przesył zbioru wykonywany jest bowiem za pośrednictwem programu, umożliwiającego wykorzystanie mikrokomputera ComPAN jako końcówki ODRY 1305.

Możliwości wykorzystania systemu TRANS bez eksploatacji baz danych DBMS

Przyjęte założenia umożliwiają przechowywanie danych z bazy dBASE w zbiorach PZS systemu ODRA 1305 bez eksploatacji bazy danych DBMS. Wprowadzenie przechowywanych w ten sposób danych z powrotem do pliku bazy dBASE możliwe jest dzięki identycznej organizacji informacji w przesyłanym zbiorze dla obu kierunków transmisji. Dla tak ograniczonej funkcji nie jest wprawdzie konieczne przesyłanie opisu struktury pliku dBASE, pozostawiono jednak to rozwiązanie celem:
- ujednoliccenia obsługi systemu TRANS,
- umożliwienia kontroli i eliminacji pomyłek w przypadku przechowywania w PZS zawartości kilku plików dBASE.

Struktura oprogramowania umożliwiającego wymianę danych między obiema bazami

Oprogramowanie systemu TRANS składa się z czterech części, realizujących odrębne funkcje, stosownie do wyróżnionych etapów transmisji. Poszczególne składowe oprogramowania zrealizowane zostały jako odrębne programy, lub też jako zbiory zleceń /makrozlecenia/ interpretowane w systemach zarządzania obu baz lub na poziomie systemów operacyjnych obu komputerów. Elementy te scharakteryzujemy krótko wg wykonywanych przez nie funkcji.
/1/ Wyprowadzanie danych z bazy dBASE II. Na etapie tym tworzony jest plik, zawierający opis struktury pliku

relacji bazy danych dBASE oraz dane pochodzące z tej relacji. Postać danych i opis struktury omówiono w rozdziale poprzednim. Program, realizujący te funkcje ma postać pliku zleceń interpretowanych w bazie dBASE.

/2/ Wprowadzanie danych do bazy DBMS. Sposób realizacji tego etapu zależy od tego, czy w bazie danych DBMS istnieje relacja, do której mają być wprowadzone dane.
- Jeśli relacja taka istnieje, to jej struktura porównywana jest z opisem struktury pliku dBASE, umieszczonym w przesyłanym zbiorze. Wykryte niezgodności powodują wydruk ostrzeżeń dla użytkownika, bądź też zatrzymanie programu. Jeśli wprowadzanie jest możliwe to dane po konwersji umieszczane są we właściwej relacji, przy wykorzystaniu procedur bibliotecznych systemu DBMS. Wprowadzona krotka /rekord/ może przy tym zostać odrzucona jeśli:
- w relacji jest już krotka o tej samej wartości klucza,
- wartość wyrażenia kwalifikującego dla danej krotki jest negatywna,
- relacja uległa przepełnieniu.
- Jeśli relacja, do której mają być wprowadzone dane nie istnieje, to na podstawie opisu struktury pliku dBASE przesłanego wraz z danymi, tworzony jest zbiór zleceń /makroinstrukcja/ w języku QL bazy DBMS, deklarujących potrzebną relację. Wykonanie tej makroinstrukcji w systemie DBMS prowadzi następnie do automatycznego utworzenia wymaganej relacji. Proces wyprowadzania danych realizowany jest jak poprzednio. Program realizujący funkcje tego etapu wykonany został w języku Pascal. Jest to między innymi związane z postacią procedur bibliotecznych systemu DBMS, dostępnych tylko w tym języku.

/3/ Wyprowadzanie danych z bazy DBMS. Celem tego etapu jest utworzenie w PZS systemu ODRA 1305 zbioru zawierającego dane wskazanej relacji bazy DBMS oraz opis jej struktury. Postać danych i opisu struktury są takie same jak na etapie transmisji z bazy dBASE do DBMS. Do wyprowadzenia danych z relacji DBMS wykorzystuje się procedury biblioteczne tego systemu. Rozwiązanie to określiło konieczność wykonania omawianego programu w języku Pascal.

/4/ Wprowadzenie danych do bazy dBASE. Pierwszą operacją realizowaną na tym etapie jest kontrola zgodności struktury relacji, z której pochodzą dane, ze strukturą pliku dBASE, do którego dane mają zostać wprowadzone. Wykryte różnice są sygnalizowane, możliwe jest jednak zignorowanie tych ostrzeżeń. Po kontroli struktur i przekształceniu danych do postaci akceptowanej w systemie dBASE następuje wprowadzenie danych do wskazanego pliku. Operacje tego etapu realizowane są częściowo za pomocą programu wykonanego w języku Pascal, częściowo zaś za pośrednictwem pliku zleceń interpretowanych w systemie dBASE.

Elementy składowe programów, realizujące funkcje 1 i 4, a także 2 i 3 zostały scalone w odrębne całości dla wygody użytkownika.

Rola użytkownika w procesie transmisji

Jednym z założeń postawionych w trakcie projektowania systemu TRANS było zminimalizowanie liczby operacji, które samodzielnie musi wykonać użytkownik tego systemu. Stąd też jedyną całkowicie samodzielną akcją użytkownika jest początkowe zainicjowanie działania programu, bądź to przy pracy autonomicznej mikrokomputera ComPAN, bądź też w takim trybie pracy, kiedy ComPAN spełnia rolę końcówki komputera ODRA 1305. Dalsze działania użytkownika podzielić można na dwie kategorie.
- Odpowiedzi na pytania programów systemu TRANS. Pytania te dotyczą przede wszystkim nazw zbiorów /plików dBASE lub relacji DBMS/, na których mają być wykonywane wskazane operacje oraz w przypadku ComPAN-a - mechanizmów dyskowych, używanych do tworzenia zbiorów roboczych. Ponadto w przypadkach, kiedy w wyniku tworzenia nowych zbiorów mogą ulec zniszczeniu istniejące zbiory o identycznych nazwach, ostrzega się użytkownika i żąda jego zgody na kontynuację pracy programu.
- Inicjowanie kolejnych etapów transmisji. Rozpoczęcie przesyłu zbioru między dwoma komputerami oraz wprowadzanie danych do drugiej danych wymaga oddzielnych akcji użytkownika. Dla uproszczenia tych działań, po utworzeniu zbioru gotowego do transmisji, wyświetlany jest szczegółowy wykaz operacji, jakie winien wykonać użytkownik, aby dokonać przesyłu, a potem uruchomić program wprowadzania. Dalszy schemat postępowania jest zgodny z punktem /1/.

Przedstawiony system został zrealizowany i sprawdzony. Przeprowadzone próby praktyczne wykazały jego pełną użyteczność.

mgr inż. KRZYSZTOF BUCHAŁA
Zakład Systemów Automatyki
Kompleksowej PAN - Gliwice

RTDS-16 SYSTEM URUCHAMIANIA SYSTEMÓW MIKROPROCESOROWYCH

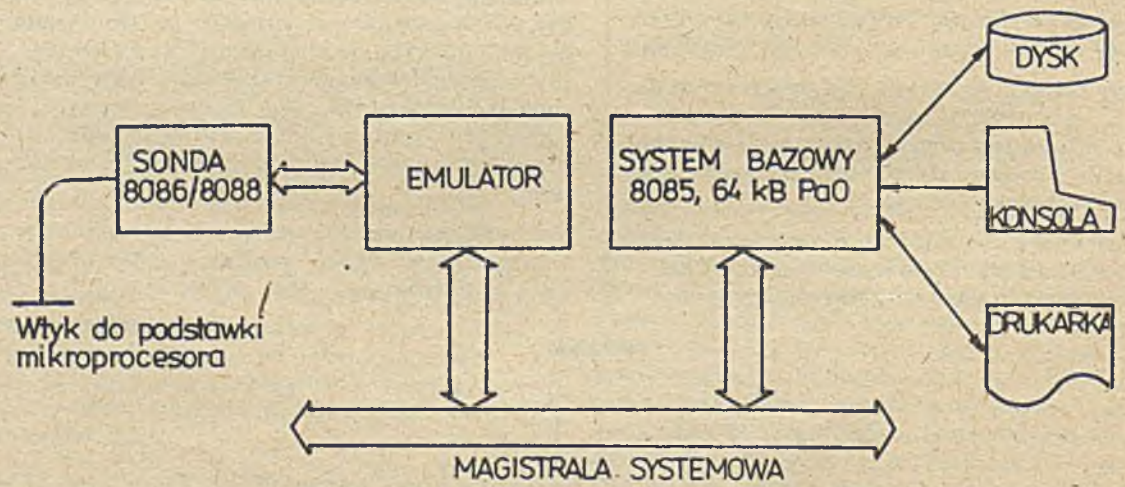
Emulatory układowe /ang. in-circuit emulators/ są dość szeroko stosowanym narzędziem uruchamiania i integracji sprzętowo-programowej systemów mikroprocesorowych. Działanie tych urządzeń opiera się na zastąpieniu mikroprocesora w układzie uruchamianym systemem zawierającym identyczny mikroprocesor, otoczony układami umożliwiającymi zarówno jego pracę w czasie rzeczywistym jak i dodatkowo śledzenie tej pracy oraz wpływanie na jej przebieg. Możliwość sterowania procesem systemu uruchamianego /prototypowego/, osiągnięta dzięki wprowadzonej tu możliwości dostępu do zasobów tego systemu oraz zmiany stanu procesora pracującego w środowisku takiego systemu, czyni emulatory układowe aktywnym narzędziem znakomicie wspomagającym projektowanie i uruchamianie systemów mikrokomputerowych. Ta cecha obok innych właściwości emulatorów, takich jak: emulacja pamięci prototypu, sprzętowa realizacja punktów zatrzymania programu, rejestracja przebiegu procesu systemu uruchamianego w czasie rzeczywistym, czy interaktywny lub wsadowy tryb pracy, w sposób istotny wyróżniają narzędzia te od innych środków uruchomieniowych, tak urządzeniowych /oscyloskopy, analizatory stanów logicznych, itp./ jak i programowych /debuggery, itp./.

Podstawowa charakterystyka systemu RTDS-16

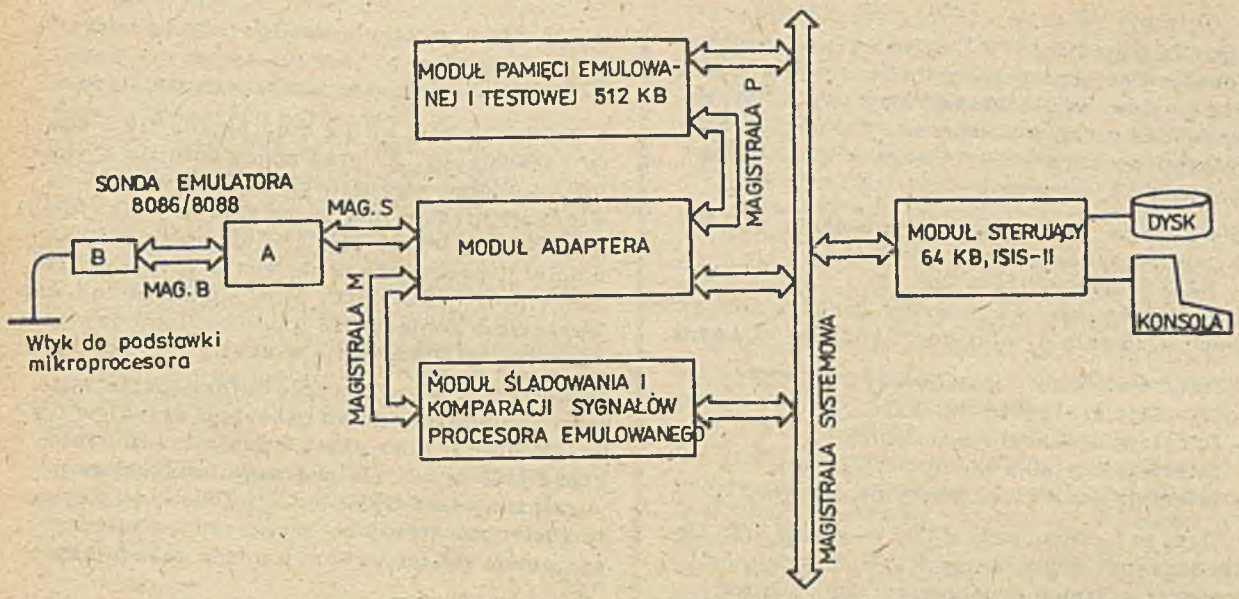
Opracowany w Zakładzie Systemów Automatyki Kompleksowej PAN w Gliwicach system RTDS-16 /Real Time Development System for 16-bit microprocessors/ służy do wspomagania fazy integracji sprzętowo-programowej systemów mikroprocesorowych 16-bitowych, opartych na procesorach Intel 8086 i 8088. Jest on przedstawicielem nowej generacji emulatorów mikroprocesorów 16-bitowych, stanowiącym kontynuację linii emulatorów mikroprocesorów 8-bitowych systemu RTDS-8 [1, 2].

Oprogramowanie narzędziowe RTDS-16, eksploatowane w środowisku systemu operacyjnego ISIS-II zainstalowanego na 8-bitowym systemie bazowym /ze stronicowaną pamięcią operacyjną/, wspomaga w procesie konstrukcji programów zadania edycji, kompilacji i łączenia modułów, tworzonych w języku asemblera oraz języku wysokiego poziomu PL/M. Proces uruchamiania programów i układów prototypowych jest wspomagany układowym emulatorem mikroprocesów 8086/8088. Podsystem emulatora RTDS-16 należy do systemu sterującego emulacją, którym jest procesor systemu bazowego, o oddzielnych magistralach: wewnętrznych emulatora i systemowej systemu sterującego. Emulator dołączany jest do systemu prototypowego za pośrednictwem sondy emulującej. Ogólną strukturę systemu RTDS-16 ilustruje rys. 1.

Emulację mikroprocesorów 8086/8088 zaimplementowano w oparciu o wydzieloną dwubramową tzw. pamięć testową, lokowaną w przestrzeni adresowej mikroprocesora emulowanego, z której wykonywany jest MONITOR - pro-



Rys. 1



Rys. 2

gram sterowania i monitorowania stanu prototypu, komunikujący się z programem sterowania emulacją, rezydującym w systemie bazowym. Przełączanie trybów pracy procesora sondy pomiędzy emulacją procesora w środowisku prototypu a trybem wykonywania programu MONITOR opiera się na selektywnej i dynamicznej emulacji wybranych wektorów przerwań oraz wektora RESET. Takie rozwiązanie pozwala na emulację mikroprocesora bez długotrwałego wstrzymywania jego pracy w cyklach WAIT /stosuje się tu tzw. technikę martwej pętli - tj. pętli wykonywanej z pamięci testowej/. Pamięć testowa stanowi niezależny moduł emulatora RTDS-16 o małej pojemności /8kB/ w porównaniu z zakresem adresowania mikroprocesorów 8086 i 8088 /1MB/. Lokacja tej pamięci w obszarze adresowym prototypu powinna być ustalana przez użytkownika w niekonfliktowym dla pamięci systemu uruchamianego obszarze. Pamięć ta jest dostępna dla procesora systemu sterującego /bazowego/ poprzez drugą bramę, jako wydzielony blok stronicowanej pamięci operacyjnej systemu RTDS-16.

Zastosowany w systemie RTDS-16 emulator układowy umożliwia emulację procesorów 8086 i 8088 w trybie MAXIMUM jak i MINIMUM dla częstotliwości zegara do 15 MHz, dopuszczając współpracę procesora emulowanego na magistrali lokalnej z koprocesorem numerycznym lub procesorem wejścia-wyjścia /bez możliwości monitorowania pracy współprocesora/. Emulator ten jest emulatorem uniwersalnym mikroprocesorów 8086 i 8088, tzn. nie posiada modułów urządzeniowych ani programowych specyficznych dla emulowanych mikroprocesorów. Typ mikroprocesora zainstalowanego w sondzie jest rozpoznawany programowo i implikuje automatyczne odpowiednie skonfigurowanie systemu RTDS-16. Istotną cecha emulatora RTDS-16 jest dynamiczna i selektywna monopolizacja zasobów systemu uruchamianego, wykorzystywanych przez emulator. Wielkość tych zasobów określona jest rodzajem realizowanej przez emulator funkcji. Emulator posiada dodatkową możliwość oddania wszystkich zasobów systemu uruchamianego do jego dyspozycji, co oznacza pracę prototypu zupełnie bez ograniczeń - w warunkach rzeczywistych.

Konfiguracja systemu

Strukturę systemu RTDS-16 przedstawiono na rys. 2. System składa się z następujących modułów:
/a/ modułu sterującego /zwanego też systemem bazowym/,
/b/ modułu adaptera,
/c/ modułu pamięci emulowanej i testowej,
/d/ modułu śladowania i komparacji stanów sygnałów magistrali procesora emulowanego /z sondą sygnałów zewnętrznych/,
/e/ uniwersalnej sondy mikroprocesorów 8086/8088, w postaci sondy zasadniczej "A" i sondy buforującej "B".

Moduł /a/ stanowi 8-bitowy mikrokomputer ogólnego przeznaczenia /oparty na procesorze 8085/ z zainstalowanym skrośnym oprogramowaniem narzędziowym procesora 8086. Moduły /b/ - /e/, tworzą podsystem emulacji procesorów 8086/8088. Wszystkie moduły RTDS-16 skonfigurowane są wokół magistrali systemowej z 8-bitowym przesyłem danych pomiędzy poszczególnymi modułami. Magistrala "S" doprowadza do adaptera sygnały mikroprocesora sondy oraz wyprowadza linie sterujące emulacją np. odcinaniem sondy od systemu prototypowego, forsowaniem przerwań,

itp. Na magistrali "P" odbywa się przesył informacji pomiędzy procesorem sondy a pamięcią emulowaną i testową. Magistrala "M" doprowadza m. in. buforowane linie magistrali mikroprocesora emulowanego do modułu śladowania i komparacji.

System bazowy wyposażony jest w:
- pamięć operacyjną 64 kB RAM,
- RAM - dysk 256/448 kB,
- jednostkę pamięci na dwóch dyskach elastycznych 5,25 cala o pojemności 360/720 kB każdy,
- interfejs konsoli systemowej TTY-CRT,
- interfejs szeregowy RS-232C,
- interfejs drukarki równoległej,
- interfejs czytnika taśmy papierowej,
- interfejs dziurkarki taśmy papierowej.

Jest to konfiguracja RTDS w wersji M3. Dotychczasowi użytkownicy RTDS-8 będą mogli tworzyć systemy uruchomieniowe 16-bitowe droga wymiany pakietów emulatora 8-bitowego na pakiety emulatora 16-bitowego.

Konfiguracja RTDS z dyskami elastycznymi 8-calowymi typu PLx45D wymaga wersji 3.4 systemu operacyjnego ISIS-II. Konfiguracja RTDS/M2 z dyskami 5,25-calowymi typu Robotron K5600.10 wymaga wersji 4.0 ISIS-II. W tym drugim przypadku, ze względu na fakt, że skrośne oprogramowanie narzędziowe mikroprocesorów 8086/8088 wymaga pojemności dysków większej niż pojemność jednostronnych dyskietek 5,25-calowych, niezbędne jest rozszerzenie pamięci o moduł RAM-dysku o pojemności 256 kB.

Oprogramowanie systemu bazowego RTDS-16 stanowią:

- monitor systemowy,
- dyskowy system operacyjny kompatybilny z ISIS-II, wersja 4.0, z biblioteką programów usługowych oraz z wspomnianym wcześniej skrośnym oprogramowaniem narzędziowym procesorów 8086/8088, zawierającym:
  - edytor EDIT,
  - skrośny asembler ASM86,
  - kompilator języka PL/M - PL/M86,
  - program łączący LINK86,
  - program lokujący LOC86,
  - program konwersji kodów CONV86,
  - program konwersji formatów CH86,
  - program zarządzania biblioteką LIB86,
- program sterowania emulacją mikroprocesorów 8086/8088, składający się z następujących modułów:
  - EM - podstawowy plik sterujący emulacją,
  - EM.MED - program MONITOR,
  - EM.CFG - program konfigurowania pamięci testowej i emulowanej,
  - EM.MSG - komunikaty o błędach,
  - EM.ASS - asembler,
  - EM.DAS - disasembler,
  - EM.CMP - program konfiguracji komparatorów,
  - EM.TRC - program postprocesora śladu.

Sonda emulująca składa się z sondy zasadniczej "A" oraz sondy buforów sygnałów kontrolno-sterujących "B". Sonda "A" zawiera wymiennie mikroprocesor 8086 lub 8088 pracujący w trybie MAXIMUM, co umożliwia emulację procesora prototypu w dowolnym reżimie pracy /MAX lub MIN/, jak również jest warunkiem koniecznym implementacji założonych funkcji śladowania w czasie rzeczywistym.

Sonda "A" zawiera ponadto bufory magistrali mikroprocesora, układ generacji sygnałów trybu MINIMUM oraz układ transformacji protokołu DMA. Sonda "B" buforuje i selekcjonuje w zależności od trybu MAX/MIN stroby i sygnały kontrolno-sterujące przekazywane między systemem prototypowym a sondą zasadnicza "A".

Pakiet adaptera zawiera:
- bufory magistrali mikroprocesora emulowanego,
- układ rozpoznawania operacji dostępu do pamięci testowej,
- układ selektywnej emulacji wektorów przerwań i wektora RESET,
- układ sterowania emulacją, obejmujący monitorowanie statusu sondy, blokowanie sondy od strony prototypu lub pamięci emulowanej, forsowanie przerwań i sygnału RESET, itp.,
- logikę śladowania kolejki wewnętrznej, pozwalającą w konsekwencji na śladowanie wykonywanych instrukcji i zatrzymywanie procesów systemu uruchamianego w momencie wykonania instrukcji uprzednio pobranej z magistrali,
- układ symulacji pamięci testowej, pozwalający na forsowanie instrukcji z pamięci operacyjnej systemu bazowego ze wstrzymywaniem procesora emulowanego w cyklach WAIT oraz umożliwiający wykonywanie programów prototypu z zatrzymywaniem procesora na kolejnych cyklach magistrali lub wyłącznie na cyklach dostępu do pamięci /praca procesora z pustą kolejką wewnętrzną/.

Moduł pamięci emulowanej i testowej zawiera 512 kB dwubramowej dynamicznej pamięci RAM, podzielonej logicznie na 64 bloki o organizacji 4K x 16b - w przypadku procesora 8086 lub 8K x 8b - w przypadku procesora 8088, alokowanej dowolnie w przestrzeni adresowej prototypu za pośrednictwem układu konfiguratora pamięci. Pamięć testowa jest jednym z bloków pamięci emulowanej. Bloki pamięci emulowanej mogą być deklarowane pod adresami k*2000H dla k $\in$ /0,1,2,..127/ i mogą mieć zadeklarowany jeden z trzech atrybutów dostępu:

RAM	- dostęp typu odczyt/zapis,
ROM	- dostęp wyłącznie dla odczytu,
GUARDED	- dostęp zabroniony.

Dla atrybutu GUARDED jakakolwiek próba zapisu lub odczytu do/z zadeklarowanego obsza-

ru pamięci jest wykrywana układowo i sygnalizowana operatorowi. Konfigurator pamięci emulowanej pozwala zakładać tzw. fold back poprzez zadeklarowanie tego samego bloku pamięci w różnych obszarach adresowych procesora emulowanego z dowolnie wybranym typem pamięci /atrybutem/. Prawo dostępu do pamięci emulowanej ze strony obydwu procesorów, tj. systemu prototypowego i systemu bazowego, jest rozstrzygane za pośrednictwem układu arbitra, który określa priorytet procesorów i przydziela im cykle pamięci. Dostęp do pamięci emulowanej i testowej od strony systemu sterującego RTDS-16 odbywa się poprzez "okno" /obejmujące 16Kx8 bitów/ w obszarze adresowym tego systemu.

Emulacja pamięci realizowana jest z wtrącaniem od 2 do 4 cykli WAIT procesora emulowanego /w zależności od zegara tego procesora/.

Moduł śladowania i komparacji zawiera dwa identyczne układy komparacji stanów magistrali oraz logikę śladowania tych stanów. Każdy komparator /zrealizowany na elementach pamięci RAM/ pozwala na realizację urzadzeniowych punktów zatrzymań programu wykonywanych w czasie rzeczywistym oraz na selektywne śladowanie przebiegu programów. Warunkiem zatrzymania lub śladowania może być dowolna kombinacja adresu, danej, rodzaju cyklu magistrali, rodzaju segmentu, czy sygnałów zewnętrznych definiowanych przez użytkownika. Pojedynczy komparator wykrywa relacje równości, nierówności, mniejszości, większości, przynależności do obszaru, jak również spełnienia warunku określonego maską bitów. Relacje te mogą być deklarowane zarówno na adresach jak i na danych procesora emulowanego.

W przypadku emulacji mikroprocesora 8086 dodatkowo można ustawiać warunki komparacji na danej 16-bitowej /wyłącznie dla przesyłów 16-bitowych w pojedynczym cyklu/ albo 8-bitowej. W tym ostatnim przypadku dana może być wykrywana, w zależności od deklaracji, na młodszej, starszej albo dowolnej połówce magistrali.

Układ śladowania stanów magistrali pozwala na bezwarunkowe lub selektywne zapamiętanie 2048 cykli magistrali podczas pracy procesora emulowanego w czasie rzeczywistym. Deklarowany jest warunek startu śladowania oraz ewentualny właściwy warunek śladowania. W przypadku śladowania selektywnego warunek może dotyczyć przesyłu informacji na magistrali, albo wykonania /a ściślej: pobrania z kolejki wewnętrznej do modułu wykonawczego mikroprocesora/ bajtu przesyłanego na magistrali w cyklu CODE ACCESS. W tym drugim przypadku śladowane są 128 sekwencje po 16 cykli magistrali każda. Pojedynczą sekwencję stanowia cyklicznie zapamiętywane cykle od pobrania bajtu instrukcji do jego wykonania /w sensie zdefiniowanym wyżej/. W związku z tym chcąc śladować tylko instrukcje wykonywane, należy tak inicjować komparatory, by wykrywały ostatnie bajty tych instrukcji. Śladowanie instrukcji wykonywanych realizowane jest niezależnie, czy instrukcja generuje cykl wykonawczy, czy nie. W obu przypadkach uaktywniany jest układ śladowania kolejki wewnętrznej /umieszczony na pakiecie adaptera. Pamięć śladowania stanowi bufor cykliczny, którego zapełnienie może powodować wstrzymanie procesu systemu uruchamianego.

Perspektywy rozwoju

Przedstawiona w niniejszym artykule konstrukcja systemu uruchamiania RTDS-16 jest wdrażana do produkcji w Zakładach Urządzeń Komputerowych MERA-ELZAB w Zabrzu. Z uwagi na ograniczone możliwości skrośnego oprogramowania narzędziowego mikroprocesorów 16-bitowych jakie posiada system operacyjny ISIS-II, opracowywana jest wersja emulatora typu "slave", komunikującego się łączem szeregowym z komputerem nadrzędnym, o bogatym oprogramowaniu narzędziowym, pracującym pod systemem PC-DOS /np. IBM, ComPAN, itp./. Potencjał funkcjonalny komputera sterującego pozwoli zwiększyć efektywność i wygodę wykorzystania tak kompleksowego narzędzia, jakim jest emulator.

Literatura:

[1] K. Plusczok: "RTDS-8 - System wspomagający uruchamianie systemów mikroprocesorowych" INFORMATYKA nr 1/83

[2] M. Konsek: "Rozwój systemu RTDS-8", Biuletyn MERA nr 5, 6/84.

mgr inż. ANDRZEJ MENCEL
mgr inż. JACEK WOJCIECHOWSKI
Ośrodek Badawczy Elektrotechniki
i Automatyki Górniczej
Katowice

LOKALNA SIEĆ KOMPUTEROWA W KWK "LENIN" – WNIOSKI EKSPLOATACYJNE

W latach siedemdziesiątych i osiemdziesiątych zainstalowano w kopalniach węgla kamiennego znaczną ilość komputerowych systemów kontroli i sterowania. Systemy te zrealizowano na bazie minikomputera PRS-4, produkowanego przez ZEG Tychy. Każdy z komputerów realizując swój algorytm kontrolno-sterujący, rejestruje także dane pomiarowe z czujników. Ze względu na ograniczenia pamięciowe tych systemów /"okrojone" pamięci operacyjne, brak pamięci zewnętrznych/, okres przechowywania danych jest krótki, co utrudnia lub uniemożliwia prowadzenie analiz długoterminowych. W przypadku systemów nadzorujących bezpieczeństwo kopalni, niedopuszczalne jest zakłócanie realizacji ich funkcji poprzez rozbudowę konfiguracji o pamięci zewnętrzne i odpowiednie oprogramowanie przetwarzające. Rozszerzenie możliwości funkcjonalnych tych systemów jest możliwe poprzez ich integrację za pomocą Lokalnej Sieci Komputerowej LSK, która wzbogaca je w nowe funkcje, nie ingerując prawie w ich wewnętrzną strukturę. Elementy układowe i oprogramowanie sieciowe dla kopalnianych systemów czasu rzeczywistego zostały opracowane w CN-P EMAG, przy współpracy ZSAK PAN, w /ramach problemu węzłowego 06.4 w latach 1982--85/, natomiast pilotująca sieć obiektowa w minimalnej konfiguracji pracuje od wielu miesięcy w KWK LENIN w Mysłowicach. Etap badań obiektowych jest dla twórców sieci bardzo angażujący emocjonalnie, bowiem LSK objęła swoim działaniem jeden z najbardziej newralgicznych systemów kontrolnych kopalni, mianowicie - metanometrię.

Podstawowe parametry sieci LSK

Z punktu widzenia sprzętowego sieć LSK tworzą urządzenia systemów obiektowych dodatkowo wyposażone w kontrolery sieci lokalnej, połączone ze sobą kablem koncentrycznym poprzez moduły nadawczo-odbiorcze. Oprogramowanie sieciowe, stanowiące o organizacji logicznej LSK, jest rozłożone pomiędzy urządzenia tworzące sieć LSK. Oprogramowanie najniższego poziomu zlokalizowane jest w kontrolerze komunikacyjnym. Poziom wyższy oprogramowania znalazł się w minikomputerze PRS-4, pełniącym podwójną rolę: jednostki centralnej systemu obiektowego oraz hosta sieci.

W zależności od nadzorowanego procesu działa on pod kontrolą innego systemu operacyjnego. Jeden z nich to wieloprogramowy system czasu rzeczywistego, w którym zarządzanie procesami aktywizowane jest zdarzeniami zewnętrznymi, a przydział czasu procesora odbywa się według regulaminu cyklicznego. Drugi to również wieloprogramowy system czasu rzeczywistego z podziałem czasu. Czas procesora dzielony jest pomiędzy dwa procesy: proces pomiarowy i proces operatorski. Algorytm podziału czasu działa na dwu poziomach, najpierw rozdzielając czas pomiędzy procesy, a następnie pomiędzy zadania wykonywane w ramach procesu operatorskiego.

Projekt oprogramowania LSK bazował na koncepcji podziału logicznego zadań sieciowych na warstwy. Porównując powstałe warstwy z modelem sieciowym Open System Interconnection /ISO/, można zauważyć pewne różnice. W ramach warstwy LINIOWEJ pojawiła się dodatkowa podwarstwa DOSTĘPU DO ŁĄCZA, która przejęła większość funkcji warstwy liniowej, a mianowicie:
- oznaczanie flagami początku i końca ramki,
- liczenie i uzupełnianie ramki o sumę kontrolną.

Warstwa ta jednak przede wszystkim realizuje rozszerzony o priorytety algorytm CSMA/CD. Dokonuje ponadto konwersji z postaci równoległej na szeregową i odwrotnie, koduje ramkę kodem Manchester, generuje również sekwencje synchronizujące. Warstwa liniowa ograniczona została do zapewnienia buforowania pakietów wychodzących i nadchodzących z linii, rozpoznawania priorytetów i wysyłania pakietów zgodnie z nimi. Warstwa SIECI zniknęła. Jeśli pojawiłby się problem łączenia z innymi sieciami lokalnymi, będzie można do niej wrócić, ponieważ w strukturze ramki zostało to przewidziane. Obecnie, część funkcji warstwy sieciowej przejęła warstwa liniowa /mechanizm priorytetów/, a funkcje kontroli przepływu oraz dzielenia przesyłki na pakiety przejęła warstwa TRANSPORTOWA.

Warstwa transportowa umożliwia bezpołączeniowy i połączeniowy przesył danych. Po nawiązaniu połączenia warstwa transportowa zapewnia bezbłędną transmisję przesyłek, w poprawnej kolejności, bez utraty lub powielenia i w określonym czasie, ograniczonym time-outem. Warstwa ta realizuje następujące funkcje:
- multipleksowanie połączeń.

- segmentowanie przesyłek i badanie ich sekwencyjności,
- adresowanie,
- sterowanie przepływem przy pomocy kredytów w pakietach potwierdzeń,
- wykrywanie błędów
  - nieprawidłowy kolejny numer pakietu w ramach przesyłki,
  - przekłamany pakiet,
  - przepełniony bufor komunikatów,
  - przekroczenie time-outów
    - na operacje,
    - na odbiór pakietu,
    - na przydział bufora w kontrolerze,
- retransmisje pakietów w wypadku wykrycia błędów.

W miejscu warstwy SESJI i PREZENTACJI pojawiła się warstwa ZDALNEGO DOSTĘPU DO ZBIORÓW. Dzięki niej procesy związane z zastosowaniem, a umieszczone w warstwie UŻYTKOWNIKA, mogą za pomocą zleceń systemu operacyjnego, w którym pracują, odwoływać się do zdalnej bazy danych. Usługi, jakie ta warstwa świadczy, to:
- nawiązywanie i utrzymywanie połączenia z procesem bezpośredniej realizacji dostępu do zbiorów,
- rezerwowanie i zwalnianie rezerwacji dostępu do zbioru,
- zapis i odczyt zbiorów.

Najniższe warstwy funkcjonalne modelu OSI/ISO/ zrealizowane są jednak nie programowo, a układowo lub w logice mikroprogramowanej. Idąc od samego "dołu", mamy więc warstwę FIZYCZNĄ, która najpełniej kształtuje zewnętrzne charakterystyki sieci. W przypadku LSK są to następujące cechy:
- konfiguracja magistralowa,
- transmisja bitowo-szeregowa,
- funkcje specjalne: detekcja kolizji, wykrywanie zajętości łącza.

Przechodząc do szczegółów - łącze fizyczne charakteryzują:
- kabel koncentryczny WL50 - 0,96/2,95
- długość maksymalna nośnika 1 km,
- liczba stacji: teoretyczna 63, praktyczna ok. 20,
- przesył: w paśmie podstawowym,
- kodowanie: samosynchronizujące Manchester,
- szybkość transmisji: 0,5 Mbit/s,
- sygnał w linii bipolarny 1,5 V pp
- sprzężenie układu nadawczo-odbiorczego z linią bezpośrednie,
- detekcja kolizji - przez porównanie sygnału nadawanego z odbieranym,
- połączenie modułu nadawczo-odbiorczego z kontrolerem sieci: transformatorowe,
- zasilanie z kontrolera sieciowego, separowane przez przetwornicę DC/DC.

Warstwa LINIOWA zrealizowana jest w kontrolerze sieciowym. Funkcje tej warstwy realizowane są częściowo układowo, częściowo zaś programowo. Układowe funkcje przeważają w podwarstwie DOSTĘPU, która została specjalnie dopracowana pod kątem jej przydatności w sieciach czasu rzeczywistego. Zaimplementowana tu reguła dostępu posiada następujące cechy charakterystyczne:
- dostęp typu rywalizacyjno-rezerwacyjnego,
- respektowanie priorytetu wiadomości,
- gwarancja ograniczonego czasu oczekiwania dla wiadomości o najwyższym priorytecie,
- przybliżone równouprawnienie stacji,
- rezerwacja czasu łącza dla szybkiego potwierdzenia odbioru.

Sterowanie regułą dostępu realizuje sekwencer programowany INTEL 3001, współpracujący z pamięcią PROM 16x512 bit. Pozostałe funkcje warstwy LINIOWEJ realizuje specjalizowany mikrokomputer, oparty na procesorze 8080 z pamięcią 4 KB RAM i 4 KB EPROM

Pilotująca sieć LSK w KWK "LENIN"

Kopalnia ta pracuje w trudnych warunkach geologicznych o dużym zagrożeniu metanowym i wyposażona jest w system metanometryczny, na który składają się dwa rejestratory CMC-1 /oparte na PRS-4/, w pełni obsadzone czujnikami kontroli atmosfery kopalnianej. Dla realizacji pilotującej sieci LSK wytypowano jeden z rejestratorów CMC-1. Podstawowym zadaniem eksperymentu było zwiększenie zasobów fizycznych i rozszerzenie funkcji tej stacji poprzez podłączenie jej do LSK, w której jeden z węzłów sieci wyposażony jest w dużą pamięć zewnętrzną. Węzeł ten /również oparty na minikomputerze PRS-4/ obsługuje bazę danych, prowadzi przetwarzanie długoterminowe i posiada możliwość równoczesnej realizacji zadań podsystemu kontroli parametrów produkcji SKPP. Odległość pomiędzy węzłami drogą kablową wynosiła na początku eksperymentu 550 m.

Podstawowym zadaniem stacji CMC-1 jest kontrola stężenia metanu oraz parametrów procesu wentylacyjnego i odmetanowania w celu wykrycia zagrożeń. W wypadku przekroczenia wartości krytycznych centrala alarmuje obsługę oraz wyłącza zasilanie zagrożonych rejonów. Ograniczenie wielkości pamięci rejestratora powoduje, iż dane pomiarowe z poszczególnych czujników przechowywane są w systemie przez okres czterech minut. W związku z tym utrudnione jest prowadzenie działań profilaktycznych, ponieważ decyzje o ich prowadzeniu podejmuje się między innymi na podstawie śledzenia parametrów atmosfery kopalnianej przez dłuższy okres czasu.

Rozszerzenie funkcji CMC-1 oparte zostało o rozszerzoną bazę danych, która obejmuje zbiory przechowujące pomiary czujników za okres tygodnia. Zdalna /w stosunku do CMC-1/ baza danych wyposażona jest w pamięć dyskową. Na początku eksperymentu była to PD MERA 9425, obecnie - dysk Winchester firmy

AMEPROD AC-985. Tygodniową bazę danych metanometrycznych stanowią pomiary ze wszystkich czujników podłączonych do stacji CMC-1, nadsyłane do zdalnej bazy danych w postaci wektora pomiarowego co 4 minuty, to jest z częstością zgodną z cyklem pomiarowym stacji metanometrycznej. Dane gromadzone są chronologicznie. Po upływie tygodnia dane tracą ważność i zostają zastąpione nowymi. Oprogramowanie, znajdujące się w warstwie UŻYTKOWNIKA hosta-kontrolera bazy danych, pozwala analizować pomiary dowolnych czujników z dowolnej chwili czasowej w ciągu tygodnia i prezentować graficznie na ekranie monitora i drukarce w postaci diagramów. Oprogramowanie analityczne produkuje kilka typów raportów. Jednym z nich jest "Przegląd pomiarów czujników". Umożliwia on wydruk pomiarów czujników różnych typów, o różnych zakresach wartości mierzonych. Przewidziano dwa tryby wydruku:
a/ od chwili bieżącej,
b/ od żadanego momentu czasowego.

Tryb "a" podaje wyniki pomiarów za okres godziny, licząc wstecz od chwili bieżącej, natomiast tryb "b" podaje wyniki za jedną godzinę, licząc wprzód od wskazanego momentu czasowego. Raport pozwala wyświetlić dane pomiarowe na ekranie monitora w formie pseudograficznej, przy czym jednorazowo można przedstawić do 5 czujników.

Raport "Przegląd pomiarów ze zwiększoną dokładnością wydruku" umożliwia jednoczesne zobrazowanie na ekranie monitora dwu czuników skala graficznej prezentacji jest większa niż w raporcie poprzednim. Raport "Pomiar bieżący" umożliwia ciągłe śledzenie na ekranie monitora lub drukarce rozwoju sytuacji metanowej wybranego rejonu kopalni. Raport ten może być powołany w trybie normalnej i zwiększonej dokładności.

Oddzielny blok programów służy dla obsługi dodatkowego monitora inżyniera wentylacji. Poprzez ten monitor można również korzystać z dodatkowej bazy danych, która zawiera informacje o położeniu poszczególnych czujników w wyrobiskach.

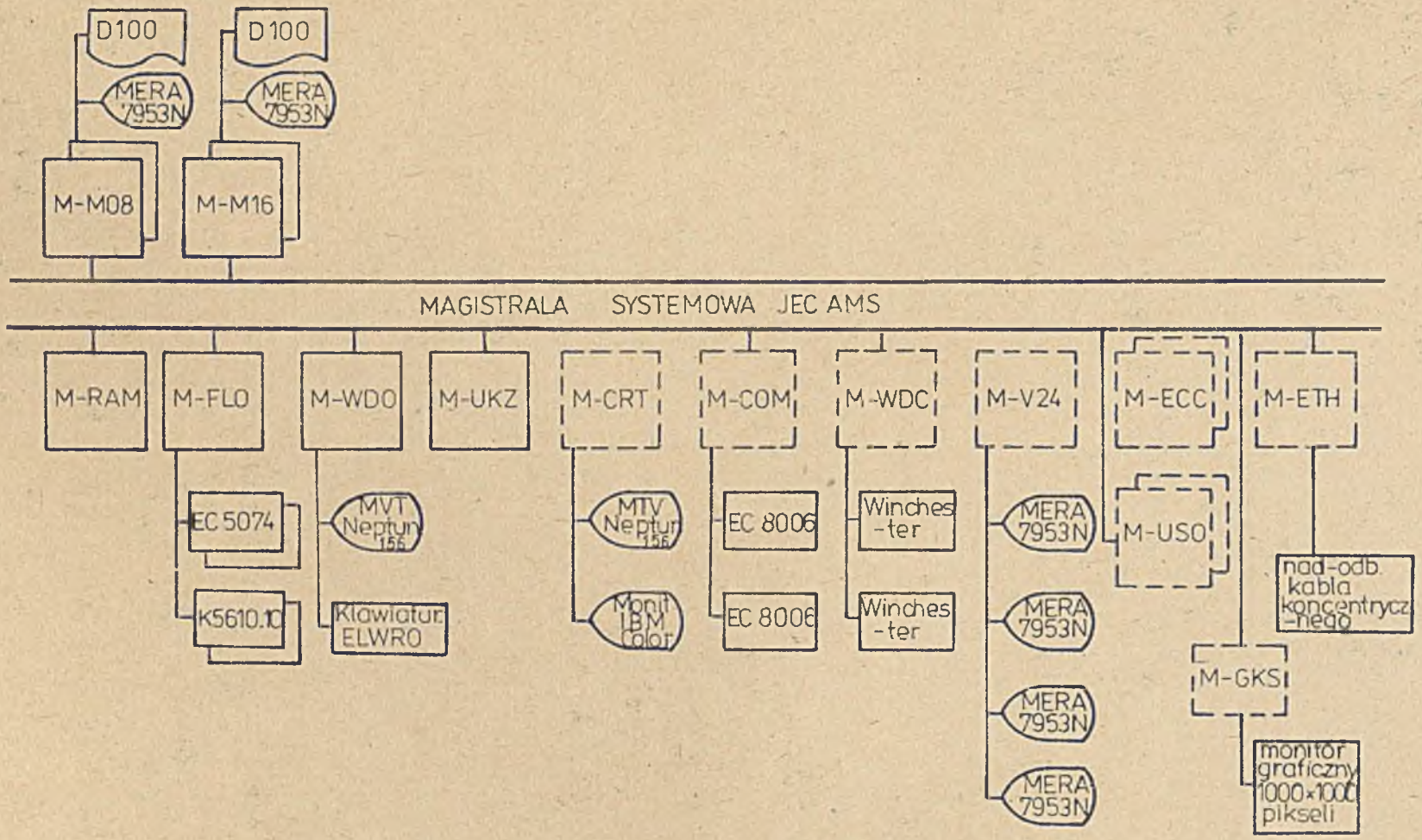
Wnioski eksploatacyjne

Wdrożenie nowego systemu na obiekcie poprzedza zwykle okres prób laboratoryjnych. Tak było i w przypadku sieci LSK, chociaż wartości tych prób nie można było przeceniać, gdyż zasymulowanie dynamiki obiektu w warunkach laboratoryjnych nie było możliwe. Przenoszenie sprzętu i oprogramowania sieciowego na pracujące systemy obiektowe odbywało się etapami tak, by w żadnej mierze nie obniżać poziomu bezpieczeństwa, gwarantowanego przez te systemy. W przypadku centrali metanometrycznej CMC-1 ułatwienie stwarzał fakt, iż dwa pracujące jednocześnie podsystemy dublują swa pracę i jest możliwe przenoszenie urządzeń obiektowych /czujniki i urządzenia wyłączające/ z jednej centrali na drugą.

Obciążenie systemów operacyjnych dodatkowymi zadaniami spowodowało, że systemy obiektowe przy pełnej obsadzie czujników pracują na granicy możliwości czasowych. Tak ekstremalne warunki spowodowały, iż okres wstępnego testowania sieci LSK trwał wiele miesięcy. W okresie tym wyeliminowano nie tylko błędy w nowo dołączanych programach, lecz również "pluskwy", które nagle ujawniły się w eksploatowanych dotąd pakietach użytkowych i systemach operacyjnych. Realizacja sieci wymagała zainstalowania dodatkowego komputera obsługującego dyskową bazę danych. Komputer ten zainstalowano początkowo w pomieszczeniu o bardzo złych parametrach klimatycznych, wysokim zapyleniu i poziomie wibracji. Te trudne warunki spowodowały, że po kilku miesiącach poważnej awarii uległa pamięć dyskowa MERA 9425, zastąpiona później przez dysk Winchester AC-985.

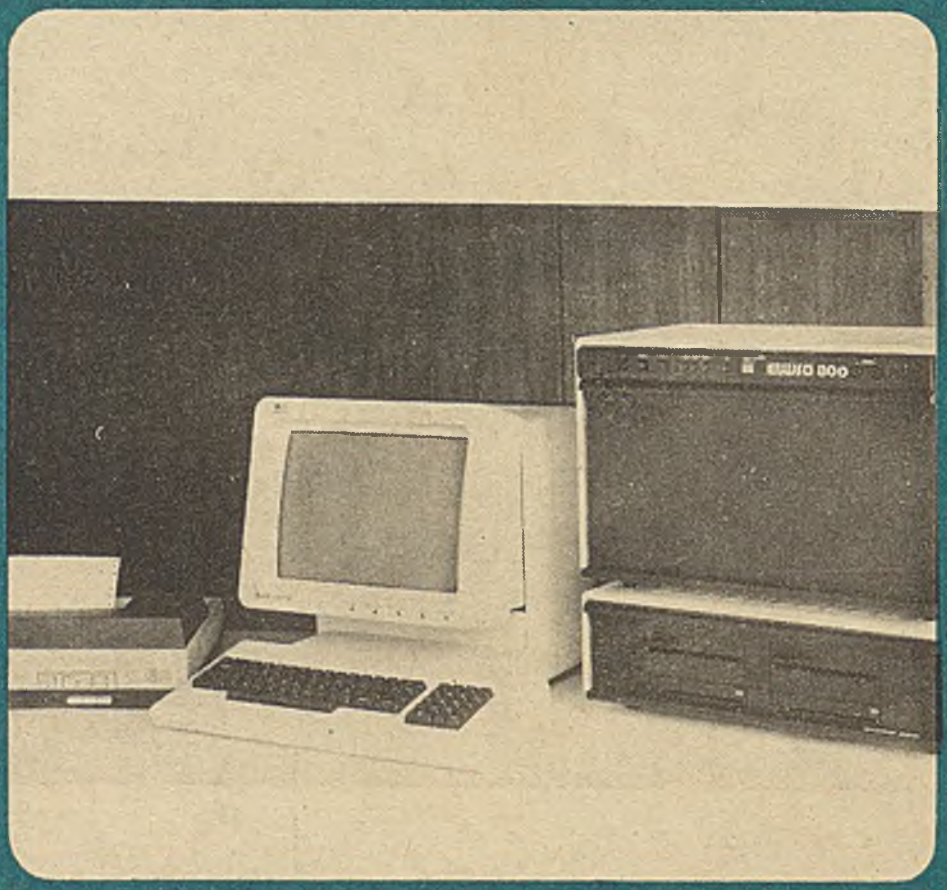
Kabel LSK, stanowiący połączenie CM C-1 z kontrolerem dyskowej bazy danych o długości 550 m, poprowadzono kanałem kablowym przeznaczonym dla sieci energetycznej. Poziom zakłóceń okazał się zbyt wysoki dla LSK, powodował bowiem okresowe przerwy w transmisji, zakończone rozłączeniem połączenia. Przełożenie kabla magistralnego w inne miejsce zdecydowanie poprawiło niezawodność transmisji, nie zmienia to jednak faktu, że moduł nadawczo-odbiorczy LSK w obecnej postaci odpowiada klasie urządzeń o przeznaczeniu biurowym.

Obecny stan wdrożenia sieci LSK w KWK "LENIN" można nazwać etapem rozwoju funkcji użytkowych. Po usunięciu wszystkich słabych punktów sprzętu i oprogramowania i przełożeniu nieufności obsługi, rozpoczął się okres "przyswajania" systemu przez służby metanometryczne kopalni. Można nawet zaryzykować opinię, że koncepcja sieciowej integracji kopalnianych systemów kontroli i sterowania zdobyła ważny przyczółek i to w najtrudniejszym sektorze zastosowań. Realizacja sieci LSK odpowiada też światowym trendom w dziedzinie automatyzacji procesów, gdyż potentaci w dziedzinie sieci lokalnych w systemach automatyki przemysłowej upatrują olbrzymie, trudne jeszcze do oszacowania, źródło zysków.



Zestawienie modułów systemu ELWRO 800

325 zł



MERA