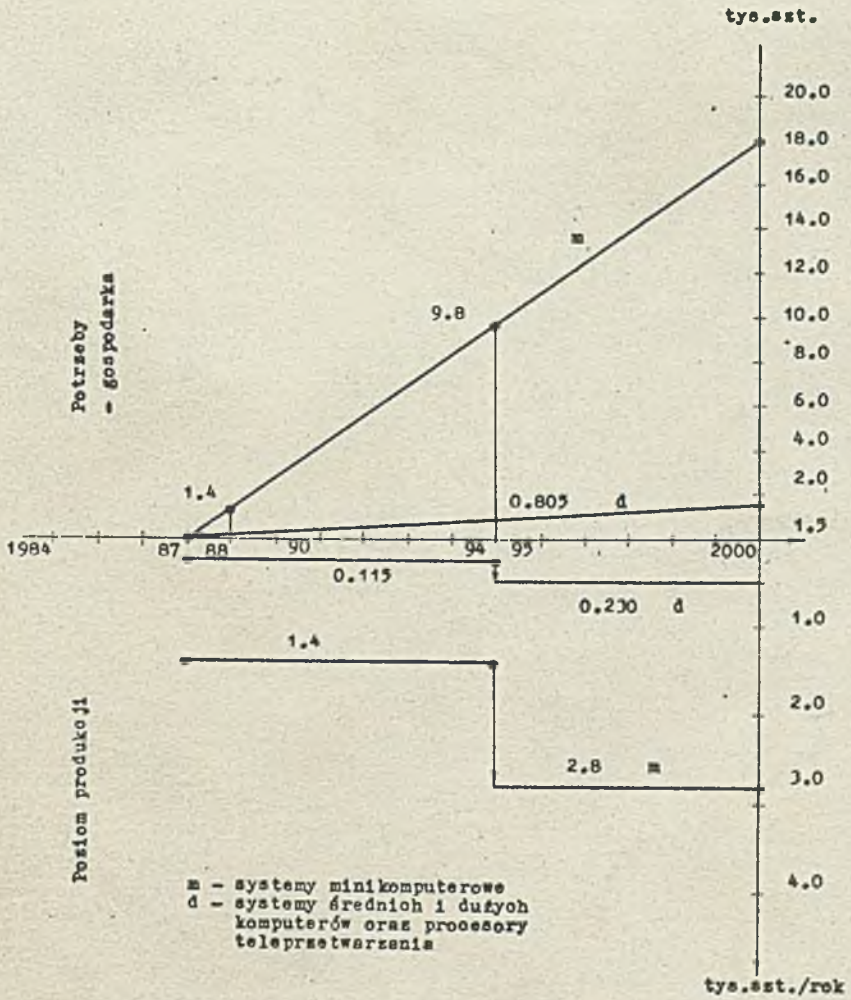
techniki komputerowe

4 '85

IMM

BIULETYN INFORMACYJNY



INSTYTUT MASZYN MATEMATYCZNYCH
BRANŻOWY OŚRODEK INTE

U W A G A !

Komitet Redakcyjny biuletynu informacyjnego "TECHNIKI KOMPUTEROWE" zawiadamia Czytelników zainteresowanych publikowaniem swoich opracowań na łamach "TK", że zmieniły się stawki honorariów autorskich za prace ukazujące się w wydawnictwach IMM. Obecnie wynagrodzenie uzależnione jest od rodzaju opracowania:

1/ za prace naukowo_techniczne oryginalne - stawka za stronę obliczeniową wynosi 300-400 zł

2/ za prace kompilacyjne - 200-300 zł za stronę obl.

3/ za sprawozdania, oferty, abstrakty, notki informacyjne i in. - 150-200 zł za stronę obl.

Kwalifikować artykuły i ustalać wysokość stawki będzie Komitet Redakcyjny na wniosek recenzenta.

Rysunek na okładce: Potrzeby krajowe i poziom produkcji systemów minikomputerowych oraz systemów średnich i dużych komputerów /Zob. art. Bonkowicz-Sittauer S. i Mocała J. - Metoda prognozowania krajowego zapotrzebowania na sprzęt informatyczny.../

TECHNIKI KOMPUTEROWE

Rok XXIII Nr 4 1985

Spis treści

str.

D W U M I E S I Ę C Z N I K

Wydaje:

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

Branżowy Ośrodek Informacji Naukowej Technicznej i Ekonomicznej

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

kand. nauk doc. Genadij W. ZELENKO
Moskowskij institut maszinostrojenija
Uczebno-metodiczeskij otdieł

Środki sprzętowe i programowe nauczania na wydziałach podwyższania kwalifikacji w dziedzinie techniki mikroprocesorowej

Na początku października br. Instytut Maszyn Matematycznych odwiedzili przedstawiciele radzieckiej organizacji naukowo-technicznej przebywający w Polsce na zaproszenie NOT w celu wymiany doświadczeń w zakresie komputerowego wspomagania nauczania oraz nauczania elementów informatyki. Publikowany artykuł jest referatem wygłoszonym na seminarium w IMM przez jednego z gości - doc. G.W.Zelenko.

Powszechne wdrożenie mikroprocesorowych układów wielkiej skali integracji (LSI) jako bazy elementowej aparatury radioelektronicznej wymaga przygotowania wielu specjalistów, z reguły nie mających doświadczenia w projektowaniu i eksploatacji cyfrowej techniki obliczeniowej. Powstaje przy tym wiele zadań o charakterze naukowo-metodycznym oraz organizacyjno-technicznym.

Należy zaznaczyć, że poziom wiedzy podstawowej u słuchaczy wydziału podwyższania kwalifikacji jest bardzo różny. Można wydzielić następujące grupy słuchaczy:

- obeznani z techniką cyfrową, wykonaną na tradycyjnych elementach cyfrowych,
- programiści użytkowi, pracujący z wykorzystaniem języków wyższego poziomu,
- programiści systemowi i specjaliści programujący w językach ukierunkowanych maszynowo,
- specjaliści, nie obeznani ani z oprogramowaniem ani z projektowaniem aparatury cyfrowej.

Zadaniem wykładowców jest danie słuchaczom zakresu wiedzy pozwalającej im na samodzielne projektowanie mikroprocesorowej aparatury radioelektronicznej, stosownie do ich działalności podstawowej. Należy jednak zaznaczyć, że część słuchaczy chce zaznajomić się z podstawami mikroprocesorowej techniki tylko w celu jej poprawnej eksploatacji i możliwości opracowania mikroprocesorowej aparatury radioelektronicznej (MAR).

Przy przygotowywaniu specjalistów w dziedzinie projektowania MAR opracowania naukowo-metodyczne i środki techniczno-programowe nauczania powinny być nastawione na szkolenie w zakresie:

- podstaw techniki pracy na schematach blokowych,
- podstaw logiczno-arytmetycznych EMC,
- budowy systemów mikroprocesorowych,
- programowania w językach maszynowo-zależnych np. assembler,
- programowania w językach wysokiego rzędu,
- projektowania systemowego MAR,

Wiedza teoretyczna z wykładów powinna być utrwalana i pogłębiana na zajęciach praktycznych, laboratoryjnych i w pracach kontrolnych. Do ich prowadzenia należy mieć odpowiednią bazę naukowo-laboratoryjną. Jako jej podstawę można wykorzystywać minikomputer z oprogramowaniem pozwalającym pisać i uruchamiać programy dla MAR i /lub specjalistycznych urządzeń laboratoryjnych. Przy wykorzystaniu minikomputera np. SM-1402 z systemem operacyjnym, pracującym w trybie podziału czasu, można jednocześnie uczyć do 15 słuchaczy każdemu słuchaczowi indywidualnie udostępnia się monitor ekranowy . Z reguły słuchacze mają możliwość napisania i uruchomienia tylko fragmentów programów docelowych MAR, nie związanych z bezpośrednią organizacją wejścia-wyjścia, co w dużym stopniu komplikuje nauczanie MAR.

Wykorzystanie minikomputera klasy SM, wraz z wieloma monitorami ekranowymi, okazuje się bardzo uciążliwe i stanowi przeszkodę nie do pokonania dla wielu użytkowników. Zastosowanie specjalistycznych urządzeń laboratoryjnych w celu zaznajomienia się z układami mikroprocesorowymi LSI jest szczególnie wskazane nie tylko ze względów techniczno-ekonomicznych ale i w sensie zbliżenia uczącego się do zastosowania układów mikroprocesorowych w technice. Przeanalizujemy dalej organizację takiego urządzenia laboratoryjnego, jego oprogramowanie oraz zadania rozwiązywane przy nauczaniu. Urządzenie to jest specjalistycznym jednopakietowym mikrokomputerem na bazie mikroprocesora КР580ИК80А . Na pakiecie EMC znajdują się:. klawiatura alfanumeryczna, moduł procesorowy (mikroukłady КР580ИК80А) z zegarem taktującym oraz kontrolerem systemowym , PAO o pojemności 16-64 Kbajt, podstawka do podłączenia układów pamięci stałej z programami obsługującymi i wieloma programowanymi mikroprocesorowymi układami LSI do organizacji wejścia-wyjścia. Do tych ostatnich odnoszą się: kontroler przerwań, kontroler bezpośredniego dostępu do pamięci, uniwersalny synchroniczno-asynchroniczny nadajnik-odbiornik, programowany interfejs równoległy, kontroler monitora rastrowego na ekranie telewizyjnym. Oprócz tego w mikro - EMC znajduje się element zadający czas, tj. TIMER LSI. Jako urządzenia zewnętrzne dla mikrokomputera wykorzystywane są: użytkowy magnetofon kasetowy oraz telewizor.

Wyżej omówiony komplet środków technicznych mikrokomputera przeznaczony do nauki pozwala poznać specyfikę praktycznie wszystkich peryferyjnych mikroprocesorowych układów LSI serii K580. W tym celu słuchacze wykonują prace laboratoryjne; w pierwszej - słuchacze otrzymują instrukcję do pracy z mikrokomputerem, zaznajamiają się też z możliwościami programu-monitora [I]. W tym celu do przystawki do podłączania pamięci stałych wstawia się mikroukład z zapisanym programem - monitora, który pozwala pisać i uruchamiać programy w PAO mikrokomputera na poziomie kodów maszynowych. Na przykładzie prostych programów słuchacze poznają system rozkazów mikroprocesora i sposoby adresacji.

W drugiej pracy laboratoryjnej słuchacze uczą się metod podłączania do mikrokomputera dodatkowego monitora standardowego przez peryferyjny programowany interfejs równoległy. Oprócz tego, chcąc poznać właściwości pracy LSI interfejsu w procesie pracy słuchacze łączą dwa mikrokomputery i otrzymują oceny ilościowe szybkości wymiany informacji.

Trzecia praca laboratoryjna ma na celu nauczenie właściwości pracy interfejsu szeregowego. Słuchacze łączą dwa mikrokomputery czterema przewodami, a następnie piszą i uruchamiają drajwery programowe dla różnych trybów pracy nadajników - odbiorników, otrzymując ilościowe oceny efektywności pracy w różnych trybach.

W czwartej pracy laboratoryjnej poznawane są właściwości kontrolera bezpośredniego dostępu i organizacji wejścia-wyjścia z wykorzystaniem systemu przerwań. W tym celu słuchacze piszą dla drajwerów odpowiednie programy sterujące wymianą między mikrokomputerem a kanałami szeregowym i równoległym na tle pracy programu głównego. Otrzymane w pracy parametry ilościowe są porównywane z wynikami drugiej i trzeciej pracy laboratoryjnej, w której wymiana odbywała się z wykorzystaniem metod obiegu programowego. .

Zadaniem piątej pracy laboratoryjnej jest zapoznanie słuchaczy z metodą wymiany informacji za pomocą bezpośredniego dostępu do pamięci. Na tle pracy programu głównego zapisuje się do pamięci metodą bezpośredniego dostępu informacje, otrzymywane z buforu nadajnika-odbiornika. W ten sposób odbywa się ilościowa ocena parametrów procesu wymiany.

W szóstej pracy laboratoryjnej słuchaczom demonstruje się pracę kontrolera monitora ekranowego; zmieniają oni programowo format obrazu na ekranie telewizora, a następnie obliczają parametry regulacji kontrolera i piszą odpowiedni drajwer.

Na podstawie tego wykazu prac można wyciągnąć wniosek, że ten bardzo prosty jednopakietowy szkoleniowy mikrokomputer odpowiada naukowo-metodycznym i techniczno-ekonomicznym wymaganiom nauczania oprogramowania w kodach maszynowych i specyfice mikroprocesorowych układów dużej skali integracji. Ładowanie translatora z języka wysokiego poziomu z magnetofonu do pamięci mikrokomputera pozwala przeprowadzać zajęcia praktyczne z oprogramowania.

Opisany mikrokomputer rozbudowany jest przez wprowadzenie do zestawu elektronicznego quasi dysku, czyli dodatkowej PAO, imitującej pamięć na dyskach magnetycznych [2] . Przed rozpoczęciem pracy - do dodatkowej PAO z magnetofonu kasetowego wpisuje się "zawartość" dysku, a po ukończeniu ta zawartość ponownie zapisuje się na taśmę magnetyczną. Wykorzystanie quasi dysku pokazuje słuchaczom funkcjonowanie systemu operacyjnego SP/M i programów funkcjonujących pod jego kontrolą.

Zestaw programów, które mogą być wykorzystane, ograniczone jest pojemnością dodatkowej PAO. Jednak wykorzystanie dodatkowej PAO o pojemności ponad 128 kbajt jest niezbyt celowe ponieważ wydłuża się czas wymiany z magnetofonem kasetowym. Ten czas określa szybkość wymiany z magnetofonem, która zazwyczaj nie przekracza wielkości 2400 bitów/s. Przy pojemności 64kbajt dodatkowej pamięci operacyjnej PAO na quasi dysku można jednocześnie przechowywać program Makroassemblera, Redaktor i Debugger. Zestaw tych programów systemowych pozwala słuchaczom w warunkach komfortowych przeprowadzać uruchomienie szkoleniowych programów docelowych MAR.

Opisane środki techniczne i programowe w dogodnych warunkach zapewniają wysoką wydajność nauczania słuchaczy.

Literatura

[1] Zelenko G.W., Panow W.W., Popow S.M.: Radioljubitel'ju o mikroprocessorach i mikro-EVM. Radio 1982 nr 9-12; 1983 nr 2-4, 6-12

[2] Zelenko G.W., Panow W.W., Popow S.N.: Elektronnyj "kvazidisk" dja personal'noj EVM. Mikroprocessornyje sredstva i sistemy 1984 nr 4

TECHNIKI KOMPUTEROWE 4/85

dr Jan WALASEK

Instytut Podstaw Informatyki PAN

Wspomaganie Ady w procesorze R-47

Wstęp

Publikacja ta powstała w ramach współpracy między IMM (Warszawa) i IKSAiP (Wrocław). W 1983 r. w czasie gdy jeden z zespołów IMM opracowywał projekt kompilatora Ady dla komputera R-32, z IKSAiP nadeszła propozycja opracowania również założeń sprzętowego wspomagania programów adowskich w projektowanym komputerze R-47. Po ustaleniu możliwości, jakimi będzie dysponował procesor R-47 i zebraniu doświadczeń z opracowywania generatora kodu dla wspomnianego wyżej kompilatora zdecydowano, że wspomaganie będzie polegało na rozszerzeniu repertuaru instrukcji procesora R-47 poza planowany wcześniej zestaw (równoważny IBM 370). Dodatkowe instrukcje będą wspomagały dostęp do danych, strukturalne instrukcje sterujące, obsługę wyjątków oraz aktywację i dezaktywację podprogramów.

Wyjątek w Adzie jest uogólnieniem maszynowego wyjątku (nadmiaru, podmiaru, dzielenia przez zero i podobnych). W Adzie pojęcie wyjątku obejmuje znacznie szerszą klasę zdarzeń, bo jest nim np. nadawanie zmiennej wartości spoza jej zakresu, indeksowanie tablicy wartością spoza zakresu indeksów, a ponadto programista może sam definiować wyjątki, np.: przepełnienie kolejki, wyczerpanie puli jakichś zasobów i in. Obsługa wyjątków polega na zgłaszaniu zaistnienia zdarzenia wyjątkowego i przekazaniu sterowania do podprogramu, który opracowuje ten wyjątek.

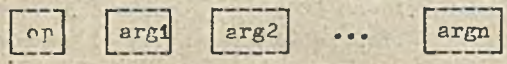
W językach wysokiego poziomu, a zwłaszcza w Adzie, z wywołaniem podprogramu i powrotem z niego związane są czynności daleko bardziej skomplikowane niż tylko zapamiętanie i odtworzenie adresu powrotu. Czynności te nazywamy odpowiednio aktywacją i dezaktywacją podprogramu, a informacje zestawiane przy aktywacji tworzą tzw. rekord aktywacji podprogramu.

Powyższe mechanizmy językowe wzięto pod uwagę przy opracowywaniu wspomagania sprzętowego omówionego w tej pracy.

Wspomaganie adowskich mechanizmów współbieżności odłożono na później, ze względu na trudności w pogodzeniu tych mechanizmów z systemami operacyjnymi przewidzianymi dla R-47 i brak doświadczeń w realizacji mechanizmów tego rodzaju.

Ogólna postać instrukcji wspomagającej

Instrukcje wspomagające, ogólnie rzecz biorąc, organizują przepływ danych pomiędzy obiektami procesora: pamięcią, rejestrami i licznikiem instrukcji. Instrukcja wspomagająca rozpoczyna się od bajtu, którego wartość jest różna od wartości kodów instrukcji podstawowych R-47. Po tym bajcie następuje właściwa instrukcja wspomagająca. Instrukcja n-argumentowa o kodzie operacji op i argumentach arg1, arg2, ... argn ma postać:



n1	n2	..	nk

Podinstrukcja przeszukuje listę wskazywaną przez pdinstr2 porównując jej elementy z wartością dostarczoną przez pdinstr1. Jako wynik, podinstrukcja dostarcza numer znalezionej wartości lub k + 1, gdy takiej wartości nie ma na liście. Elementy listy powinny być ustawione w porządku niemalejącym. Lista zawiera pojedyncze wartości przedziału [n1, nk] oraz podprzedziały tego przedziału. Podprzedziały są zapisywane następująco:

nd	ng	ng

gdzie nd i ng są odpowiednio granicą dolną i górną podprzedziału. Porównywanie jest operacją logiczną lub arytmetyczną zgodnie z rodzajem wartości dostarczonej przez pdinstr1. Relacja porządkująca listę oraz długości elementów muszą być zgodne z rodzajem i długością wartości dostarczonej przez pdinstr1.

Podinstrukcja przeszukiwania została zaprojektowana do realizacji adowskiej instrukcji CASE, do realizacji adowskiej instrukcji CASE, do realizacji dostępu do tablic, których indeksy są typu wyliczeniowego ze specyfikowaną reprezentacją oraz do przeszukiwania listy obsługi wyjątków.

Podinstrukcja kontroli zakresu

Format ogólny:

op	pdinstr1	pdinstr2	pdinstr3

Argumenty podinstrukcji określają kolejno: wartość kontrolowaną, adres listy dopuszczalnych wartości i numer wyjątku. Jeżeli kontrolowana wartość jest poprawna, pozostawia się ją na stosie, w przeciwnym razie następuje zgłoszenie wyjątku. Zgłoszenie polega na spowodowaniu przerwania i załadowaniu numeru wyjątku do rejestru przeznaczonego do tego celu.

Instrukcje

Instrukcje sterujące

Format ogólny instrukcji skoku warunkowego:

op	pdinstr1	pdinstr2

Podinstrukcja pdinstr1 dostarcza adresu pamięciowego, a pdinstr2 - wartości logicznej. Instrukcja ta powoduje przejście do miejsca określonego przez pdinstr, jeżeli wartość dostarczona przez pdinstr2 jest prawdą. W przeciwnym razie następuje przejście do następnej instrukcji.

Format ogólny instrukcji pętli:

op	pdinstr1	pdinstr2	pdinstr3

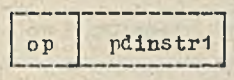
Są dwie instrukcje tego rodzaju odpowiednio dla adowskich instrukcji FOR ... LOOP oraz dla FOR ... REVERSE ... LOOP. Podinstrukcje określają kolejno: zmienną sterującą, wartość końcową i początek pętli. Instrukcja porównuje zmienną sterującą z wartością końcową i jeżeli są równe,

Kod podinstrukcji pdinstr określa rodzaj udostępnianej zmiennej i skąd zmienna pochodzi: czy z rejestru stało- , zmiennopozycyjnego, z rejestru warunku, licznika instrukcji czy z pamięci, n1 n2 określają granice bitowe zmiennej, a pdinstr jest podinstrukcją dostarczającą numer rejestru lub adresu pamięci. Wartość zmiennej jest przesyłana na stos wraz z informacją o rodzaju i długości.

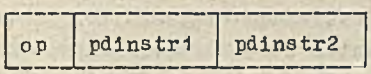
Podinstrukcje działań arytmetycznych, logicznych i relacji

Format ogólny:

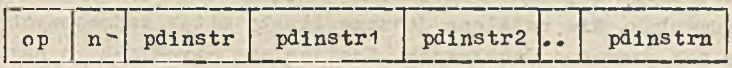
1)



2)



3)

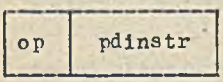


Format 1) odnosi się do działań jednoargumentowych, format 2) do działań dwuargumentowych, a format 3) do działań wieloargumentowych.
Podinstrukcje pdinstr1, pdinstr2, ..., pdinstrn dostarczają wartości argumentów. Rodzaj operacji jest określany przez kod op. Wynik operacji zastępuje na stosie wartości argumentów.

Ta rodzina podinstrukcji obejmuje podstawowe adowskie operacje arytmetyczne włącznie z potęgowaniem, logiczne - włącznie z AND THEN i OR ELSE oraz relacje arytmetyczne i logiczne. Ponieważ na stosie odnotowuje się rodzaj wyniku, nie ma potrzeby określania oddzielnych kodów dla operacji stałopozycyjnych, zmiennopozycyjnych itd. Argumenty operacji mogą mieć różne długości. Odpowiednie reguły określają zasady propagacji bitów i długość wynikową dla poszczególnych operacji.

Podinstrukcja odniesienia do podinstrukcji oddalonej

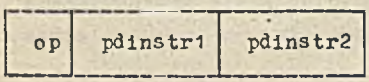
Format ogólny:



Podinstrukcja ta jest uogólnieniem operacji pośredniego adresowania. Argument pdinstr określa adres, pod którym znajduje się inna podinstrukcja, nazywana oddaloną. Podinstrukcja odniesienia działa tak, jak gdyby na jej miejsce wstawiono podinstrukcję, na którą wskazuje. Podinstrukcja oddalona może znajdować się w tej samej instrukcji, ale nie może to być podinstrukcja nadrzędna w stosunku do wskazującej.

Podinstrukcja przeszukiwania

Format ogólny:



Argument określa stałą, adres pamięci, numer rejestru, stan rejestru itd. Argument może podawać ww informacje bądź bezpośrednio, bądź może być podinstrukcją, która dostarcza tych informacji. Procesor musi najpierw wykonać podinstrukcje zagnieżdżone i dopiero otrzymawszy od podinstrukcji odpowiednie wartości może wykonać właściwą instrukcję. Podinstrukcje mają format ogólny taki jak instrukcje, tzn. składają się z części operacyjnej i ewentualnie następujących po niej argumentów. Z kolei argumenty podinstrukcji mogą być znów podinstrukcjami itd. Pojedyncza instrukcja wspomagająca może więc tworzyć dość skomplikowaną strukturę. Do przedstawiania instrukcji będziemy stosowali zapis nawiasowy, np.:

MOVE (A, PLUS (B, C))

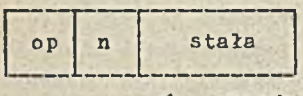
Instrukcja powyższa przesyła do obiektu A sumę obiektów B i C. Kodem operacji tej instrukcji jest MOVE, a argumentami: A oraz PLUS (B, C). Drugim argumentem jest podinstrukcja o kodzie operacji PLUS i argumentach B i C. Nawiasy wprowadzamy dla zwiększenia czytelności. Nie są one konieczne, gdyż kody operacji instrukcji i podinstrukcji określają jednoznacznie liczbę argumentów i dopuszczalną postać argumentów. Nie ustalono jeszcze liczby bitów zajmowanych przez kody operacji, ani nie określono ich wartości liczbowych. Zostaną one określone po odpowiednich badaniach statystycznych. Ponieważ instrukcje wspomagające będą zestawiane nie przez assembler, lecz przez kompilator Ady, nie jest konieczne, aby kody operacji i argumenty zajmowały całkowitą liczbę bajtów.

Instrukcje wspomagające mają dostęp do pamięci, rejestrów, licznika instrukcji i do wewnętrznego stosu procesora. Stos służy do zapamiętywania wartości dostarczanych przez podinstrukcje. Przesłanie do stosu powoduje umieszczenie przesyłanej informacji na wierzchołku stosu, a wykorzystanie informacji umieszczonej na stosie powoduje jej automatyczne usunięcie stamtąd. Tylko instrukcje mogą zmieniać zawartość pamięci, rejestrów i licznika instrukcji. Podinstrukcje mają dostęp do zawartości pamięci, rejestrów i licznika instrukcji, a mogą zmieniać tylko zawartość stosu procesora. Po wykonaniu instrukcji stos jest zawsze pusty. Poniżej prezentujemy szkicowe rodzinę instrukcji i podinstrukcji.

Podinstrukcje

Podinstrukcje udostępniania stałych

Format ogólny:

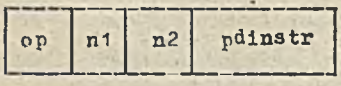


<--n bitów--> n = 1, 2, ..., 127

Kod operacji op określa jakiego rodzaju jest udostępniona stała: logiczna, arytmetyczna stałopozycyjna, czy arytmetyczna zmiennopozycyjna. Wartość stałej jest przesyłana na stos wraz z informacją o długości i rodzaju.

Podinstrukcje udostępniania zmiennych

Format ogólny:



n1, n2 = 0, 1, ..., 127

przechodzi do następnej instrukcji, w przeciwnym razie zwiększa lub zmniejsza zawartość zmiennej sterującej o jeden i przechodzi na początek pętli.

Instrukcje przesłań

Format ogólny:

1)

op	N1	n2	pdinstr1	pdinstr2

2)

op	pdinstr	pdinstr1	pdinstr2

Instrukcja formatu 1) przesyła wartość dostarczoną przez pdinstr2 do pamięci lub rejestru określonego przez pdinstr1 i granice bitowe n1 i n2. Instrukcja formatu 2) przesyła zawartość pamięci spod adresu określonego przez pdinstr2 pod adres określony przez pdinstr1 w rozmiarze określonym przez pdinstr.

Instrukcje obsługi podprogramów

Format ogólny instrukcji wywołania podprogramu:

op	pdinstr	n	pdinstr1	pdinstr2	...	pdinstrn

Format ogólny instrukcji powrotu z podprogramu i dezaktywacji podprogramu:

op

Instrukcja wywołania wywołuje podprogram znajdujący się pod adresem określonym przez pdinstr i przekazuje mu argumenty określone przez pdinstr1, pdinstr2, ..., pdinstrn. Wywołaniu podprogramu towarzyszy utworzenie specjalnego rekordu aktywacji podprogramu, przydzielenie pamięci dla podprogramu oraz dowiązanie listy wyjątków opracowywanych przez ten podprogram. Dane potrzebne do tworzenia rekordu aktywacji, przydzielania pamięci i dowiązywania listy obsługi wyjątków są zapisane na początku podprogramu.

Instrukcja powrotu przekazuje sterowanie za instrukcję wołającą usuwając jednocześnie rekord aktywacji bieżącego podprogramu i zwalniając przydzieloną pamięć. Przewiduje się również instrukcję usuwającą rekord aktywacji bez powrotu do podprogramu wołającego. Instrukcja taka ułatwi realizację procedur obsługi wyjątków. Instrukcje podprogramów zostaną bardziej precyzyjnie określone po ustaleniu zasad gospodarki pamięcią w podprogramach adowskich. Być może zasadniczą sprawą okaże się zwiększenie efektywności systemowych makroinstrukcji do gospodarki pamięcią. Przyjęte rozwiązania w tym zakresie będą miały istotny wpływ na postać rekordu aktywacji.

Instrukcje obsługi bloków

Obsługa adowskich instrukcji blokowych wymaga podobnych czynności jak obsługa podprogramów: tworzenia i skreślania rekordu aktywacji bloku, przydzielania i zwalniania pamięci oraz dołączania i odłączania listy wyjątków. Czynności te są realizowane przez instrukcje aktywacji i dezaktywacji bloku działające podobnie jak instrukcje obsługi podprogramów, lecz bez przekazywania argumentów i sterowania.

Rozszerzenie repertuaru instrukcji i podinstrukcji

Prezentowany repertuar instrukcji i podinstrukcji jest otwarty: przewiduje się możliwość jego rozszerzania. W pierwszej kolejności konieczne będzie uwzględnienie działań na standardowych rodzajach danych, tj. liczbach całkowitych półsłowowych i jednosłowowych, liczbach zmiennopozycyjnych w pojedynczej i podwójnej precyzji itd. Albowiem naszkicowany repertuar pozwala np. każdą adowską instrukcję przypisania bez odwołań funkcyjnych przedstawić jako pojedynczą instrukcję wspomagającą nawet przy najbardziej złożonej specyfikacji reprezentacji i dostępie do danych, ale w prostych przypadkach daje kod stosunkowo długi. Na przykład dostęp do jednosłowowej liczby całkowitej, której adres jest określony przez rejestr bazowy R i przesunięcie D, będzie wyglądał następująco:

ST_INT (0, 31, PLUS (REG_INT (B, 31, CONST_INT (4, R), CONST_INT (12, D)))

gdzie znaczenie użytych operacji jest następujące:

ST_INT - całkowita z pamięci,
REG_INT - całkowita z rejestru ogólnego,
CONST_INT - stała całkowita,
PLUS - dodawanie

Konieczne więc będzie uzupełnienie repertuaru podinstrukcji o podinstrukcje do dostępu i działań na standardowych danych. Ogólnie będzie to wyglądało tak, że często występująca superpozycja, np.

OP1 (C1 ., C2, X1, OP2 (X3)),

gdzie C1, C2 są ustalonymi stałymi, a X1, X2 - dowolnymi dopuszczalnymi stałymi lub podinstrukcjami, będzie równoważna pojedynczej podinstrukcji

OP (X1, X2),

gdzie OP jest kodem operacji nowej podinstrukcji.

W ten sposób rozszerzając konsekwentnie listę podinstrukcji możemy uzyskać efektywne narzędzia dostępu do skomplikowanych struktur danych: tablic z dynamicznymi zakresami, rekordów z dynamicznymi składowymi i superpozycji takich struktur, np. dostęp do elementu A (I, J) z tablicy

A: array (L1 .. U1, L2 .. U2) of REAL:

będzie realizowany podinstrukcją

ST_FLOAT (ARRAY_ELEMENT(pA, PI, pJ))

gdzie pA jest podinstrukcją udostępniającą deskryptor tablicy A, pI, pJ są podinstrukcjami udostępniającymi odpowiednio I i J.

Innym, przewidywanym kierunkiem rozszerzeń jest wprowadzanie nowych rodzajów danych, np. wektorów. Ponieważ rodzaj danej jest notowany na stosie, więc wprowadzenie podinstrukcji dostępu do wektora pozwoli równocześnie wprowadzić operacje na wektorach (np. dodawanie, mnożenie) bez wprowadzania specjalnych kodów dla tych operacji, gdyż operacja jest określana nie tylko przez kod operacji, ale również przez atrybuty argumentów operacji. Oprócz podinstrukcji, można wprowadzać nowe instrukcje, będące superpozycjami opisanych wyżej instrukcji i podinstrukcji.

Wnioski i podsumowanie

Ada jest językiem wybitnie ukierunkowanym na dane, prowadzącym w programowaniu do struktur z wielopoziomowym dostępem do danych. Złożony dostęp do danych powstaje również niejawnie wskutek rozłącznej kompilacji programów adowskich. Przedstawiona koncepcja stara się uwzględnić te wymagania Ady, ale bierze również pod uwagę współczesne możliwości sprzętowe. Realizacja przedstawionych instrukcji będzie niewątpliwie kosztowna, niemniej jednak mają one kilka ewidentnych zalet. Proponowane instrukcje eliminują jawne, oddzielne instrukcje ładowania na stos, stanowiące według Barreta i in. [2] około 40% instrukcji programu kodowanego w klasycznych maszynach stosowych, takich jak np. [2,3] , gdyż w naszych instrukcjach ładowania na stos są wewnętrznymi operacjami instrukcji. Następnie format instrukcji pozwala stosunkowo łatwo określić wewnątrz instrukcji niezależne obliczenia cząstkowe, możliwe więc będzie skonstruowanie procesora, w którym te niezależne obliczenia będą realizowane współbieżne.

Przewidywane sukcesywne rozszerzenia repertuaru instrukcji i podinstrukcji będą stosunkowo łatwe do wykorzystania. Wystarczy, żeby generator kodu zamiast określonych drzew lub poddrzew wstawiał nową, pojedynczą instrukcję lub podinstrukcję.

Mimo, że punktem wyjścia do przedstawionych udogodnień była Ada, nadają się one do pełnego wykorzystania do innych języków programowania: Fortranu, PL/1 czy Pascala. Otwartość repertuaru instrukcji z możliwością dodawania nowych rodzajów danych pozwala niewątpliwie tę użyteczność jeszcze zwiększyć.

Literatura

[1] Reference Manual for the Ada Programming Language. ANSI/MIL-STD 1815 A. United States Departament of Defence, January 1983.

[2] Barret R., Bryant C., Grau A.: An intermediate language to define dynamic semantics. Computer Languages 1984 nr 3/4, s. 149-159.

[3] Ibsen L.: A portable virtual machine for Ada. Software-Practice and Experience 1984 t. 14, s. 17-29.

dr inż. Stanisława BONKOWICZ-SITTAUER
mgr inż. Jerzy MOCAŁA
Instytut Maszyn Matematycznych

Metoda prognozowania krajowego zapotrzebowania na sprzęt informatyczny i program produkcji do roku 2000

W opracowaniu przedstawiono metodę prognozowania zapotrzebowania na środki techniki obliczeniowej; metoda oparta jest na analizie zastosowań.

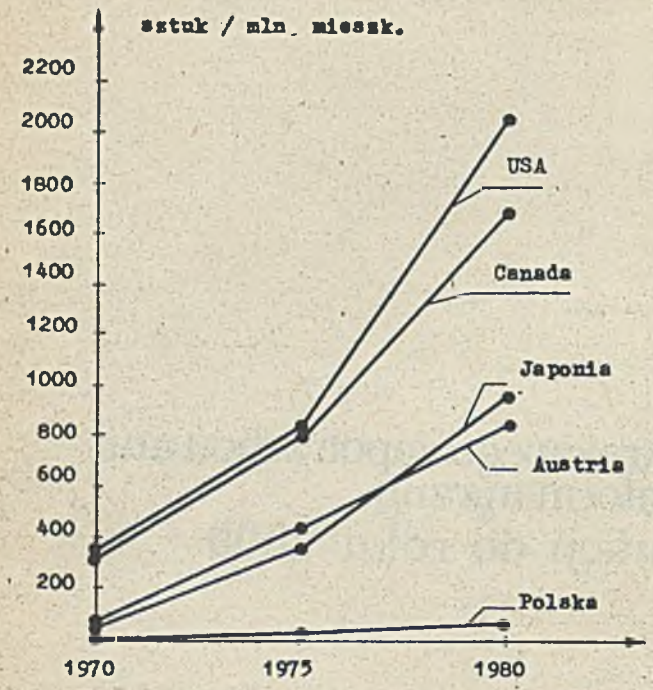
Pod pojęciem zapotrzebowania rozumie się tu liczbę systemów komputerowych, które powinny być zainstalowane w danej dziedzinie dla jej sprawnego działania.

Metoda nie uwzględnia ograniczeń ekonomicznych, technicznych i organizacyjnych. Czy takie podejście jest słuszne? Zdaniem autorów tak, ze względu na to, że daje pogląd o liczbie potencjalnych "miejsc pracy dla komputerów", natomiast wspomniane ograniczenia mogą uniemożliwić lub ograniczyć wprowadzanie takiej liczby systemów. Metoda szczególnie nadaje się zatem do wykorzystania przy planowaniu rozwoju: uzyskuje się wstępną ocenę, jakie powinno być nasycenie kraju systemami komputerowymi.

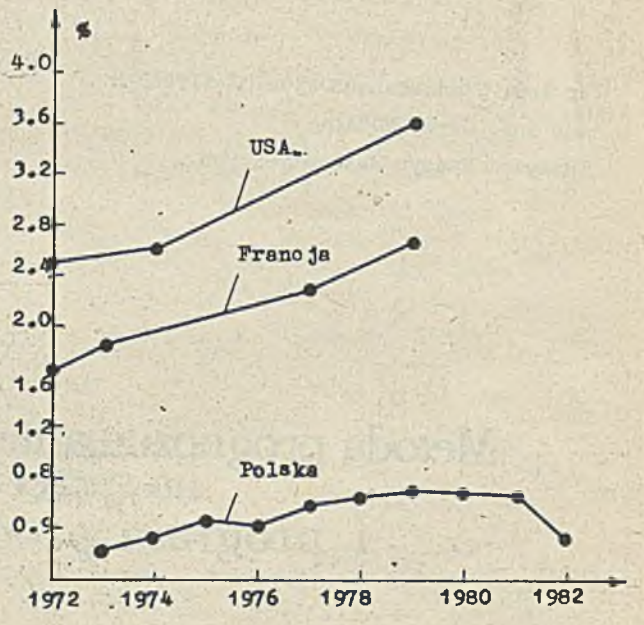
Od paru lat obserwuje się opóźnienie Polski w stosowaniu komputerów już nie tylko w porównaniu z państwami kapitalistycznymi, ale i socjalistycznymi.

Porównanie niektórych wskaźników /rys.1 i 2/ będących miarą roli informatyki w społeczeństwie nie daje żadnych racjonalnych podstaw do podejmowania decyzji o rozwoju produkcji sprzętu komputerowego. Przede wszystkim konieczna jest więc ocena, jakie powinno być nasycenie kraju systemami komputerowymi. Dopiero wtedy można określić wielkość potrzebnej produkcji, a wreszcie ocenić, czy jest ona możliwa przy istniejących ograniczeniach.

Po prezentacji metody i określeniu krajowego zapotrzebowania na sprzęt komputerowy w opracowaniu przedstawiono konieczny dla zaspokojenia tych potrzeb poziom produkcji sprzętu komputerowego. Następnie porównano wynik przeprowadzonej prognozy z wynikami osiągniętymi innymi metodami. Dokonano także oceny miejsca Polski wśród innych krajów socjalistycznych przy założeniu rozwoju zastosowań informatyki w kraju według ilości systemów wynikających z proponowanej metody prognozowania.



Rys.1.Liczba komputerów na 1 mln mieszkańców



Rys.2. Udział wydatków na informatykę w dochodzie narodowym

Prognoza zapotrzebowania na środki techniki obliczeniowej - metoda

W proponowanej przez autorów metodzie prognozowania przyjęto następujące założenia:

- założenie 1 - prognoza potrzeb nie uwzględnia eksportu
- założenie 2 - prognoza nie obejmuje systemów komputerowych w układach sterowania i regulacji
- założenie 3 - prognoza dotyczy trzech rodzajów systemów:
  a/ systemy mikrokomputerowe /u/
  b/ systemy minikomputerowe /m/
  c/ systemy średnie i duże /d/
- założenie 4 - w prognozie nie uwzględnia się w sposób bezpośredni ograniczeń ekonomicznych, technicznych i organizacyjnych.

Metoda prognozy - opis ogólny

Metoda oparta jest na analizie zastosowań w wybranych dziedzinach użytkowania systemów cyfrowych. Dla tych dziedzin określa się ilość i jakość systemów komputerowych, a dla pozostałych obszarów prognozuje się potrzebne ilości przez porównanie z obszarami wybranymi.

W prognozie można wydzielić następujące etapy:

- etap Ø - ustalenie przedziału czasowego dla prognozy
- etap 1 - podział interesujących nas dziedzin zastosowań na rozłączne podobszary
- etap 2 - wybór podobszarów dla prognozy ilościowej i jakościowej dla końca przyjętego przedziału czasowego
- etap 3 - prognoza liczby potrzebnych systemów dla wybranych w etapie 2 podobszarów zastosowań w odniesieniu do przyjętego końca przedziału czasowego
- etap 4 - oszacowanie ilościowe i jakościowe dla pozostałych podobszarów, uwzględniające wyniki prognozy dla podobszarów wybranych w etapie 2

- etap 5 - ustalenie struktury systemów i obliczenie ilości sprzętu peryferyjnego dla końca przyjętego przedziału czasowego
- etap 6 - przyjęcie modelu /rozkładu potrzeb/ zapotrzebowania na sprzęt informatyczny w całym prognozowanym okresie i ustalenie poziomu produkcji zaspokajającej to zapotrzebowanie

Metoda prognozy - przyjęte założenia

Dla przeprowadzenia prognozy wg metody opisanej w poprzednim punkcie, przyjęto następujące rozwiązania szczegółowe:

- etap Ø - przedział czasowy - lata 1988-2000
- etap 1 - zgodnie z założeniem 1 rozważa się tylko zapotrzebowanie krajowe; interesujące nas dziedziny zastosowań zostały podzielone na:
  a/ zastosowanie komputerów w szkolnictwie podstawowym i średnim
  b/ zastosowanie komputerów w gospodarce narodowej /w podziale na przemysł, budownictwo, rolnictwo, leśnictwo, itd./
- etap 2 - dla prognozy ilościowej i jakościowej zapotrzebowania na sprzęt komputerowy w roku 2000 wybrano:
  - potrzeby szkolnictwa podstawowego i średniego
  - potrzeby przemysłu
- etap 3 - prognozy dla wybranych w etapie 2 dziedzin dokonuje się dla szkolnictwa podstawowego i średniego wg wstępnych opracowań dotyczących komputerowego wspomagania nauczanie, a dla przemysłu - wg ustaleń ekspertów na systemy dla różnych zastosowań /tab.1/; przyjęto przy tym, że liczba przedsiębiorstw w roku 2000 będzie taka sama, jak w 1984 r.
- etap 4 - oszacowanie ilościowe i jakościowe potrzeb w roku 2000 dla pozostałych działów gospodarki narodowej dokonuje się wg wzoru:

$$K_i^r = K_w^r \cdot W_i^r \cdot \frac{L_i}{L_w} \qquad /1/$$

gdzie: r - rodzaj systemu /u, m, d - zał.3/

K_i^r - liczba systemów klasy r w i-tym dziale gospodarki

K_w^r - liczba systemów klasy r w wybranym dziale /przemysł/

L_w - liczba zatrudnionych w wybranym dziale /przemysł/

L_i - liczba zatrudnionych w i-tym dziale gospodarki

W_i^r - współczynnik wagowy uwzględniający podobieństwo działu i-tego do działu wybranego /współczynnikiem tym można pośrednio uwzględniać wpływ ograniczeń/

- etap 5 - w roku 2000 przewiduje się poniższe struktury systemów komputerowych rodzaju u, m, d /ustalenia ekspertów/.

A. Typowy zestaw mikrokomputerowy:

a/ monitor ekranowy semigraficzny - 1
b/ pamięć dyskietkowa - 1
c/ drukarka - 1
d/ klawiatura - 1

Przyjęte odchylenia:
10% - bez drukarek
15% - 2 jednostki pamięci dyskietkowej
15% - 1 jednostka pamięci kasetowej
40% - 1 jednostka dysków twardych /np. Winchester/
40% - pamięć back-up /np. Streamer - potokowa/
10% - 1 ploter
10% - monitory graficzne
5% - digitizery
20% - klawiatury z literami alfabetu polskiego
20% - drukarki z literami alfabetu polskiego
0.5% - 1 jednostka pamięci magnetycznej szpulowej

B. Typowy zestaw minikomputerowy:

a/ monitor ekranowy - 2
b/ klawiatura - 2
c/ pamięć dyskietkowa - 2
d/ dyski twarde - 2
e/ pamięć magnetyczna szpulowa - 1
f/ pamięć magnetyczna kasetowa - 1
g/ drukarka - 1

Przyjęte odchylenia:
25% - 4 jednostki pamięci dyskietkowej
10% - 4 jednostki dysków twardych
10% - plotery
10% - digitizery

C. Typowy zestaw średnich i dużych systemów komputerowych:

a/ lokalne monitory ekranowe - 8
b/ zdalne monitory ekranowe - 8
c/ klawiatura - 16
d/ terminale inteligentne ujęte w grupie mikrokomputery - 16
e/ procesor teleprzetwarzania - 4
f/ dyski twarde - 8
g/ jednostki pamięci taśmowej szpulowej - 6
h/ drukarki wierszowe - 2
i/ drukarki trwałej kopii - 4

Przyjęte odchylenia:
10% - 1 drukarka wierszowa
10% - 4 jednostki dysków twardych
25% - 16 sztuk monitorów ekranowych zdalnych
25% - 8 sztuk drukarek trwałej kopii

● etap 6 - zakłada się, że zapotrzebowanie ilościowe na sprzęt komputerowy będzie rosło liniowo od wartości 0 w 1988 r. do wartości ustalonej przez prognozę w roku 2000.

Prognoza zapotrzebowania na środki techniki obliczeniowej do roku 2000 - obliczania

Całkowite zapotrzebowanie /K/ w roku 2000 na systemy komputerowe ustalonego rodzaju wyraża się wzorem:

$$K^r = K_g^r + K_n^r \qquad /2/$$

gdzie: K_g^r - liczba systemów komputerowych rodzaju r /u, m, d/ w gospodarce narodowej

K_n^r - liczba systemów komputerowych do nauczania /szkoły podstawowe i średnie/

Prognoza dotycząca liczby systemów komputerowych dla gospodarki narodowej

Prognozowania liczby systemów ustalonego rodzaju dla gospodarki narodowej dokonuje się /zgodnie z punktem "Prognoza zapotrzebowania na środki techniki obliczeniowej - metoda"/ na podstawie wzoru:

$$K_g^r = K_w^r + \sum_i K_w^r \cdot w_i^r \cdot \frac{L_i}{L_w} \qquad /3/$$

gdzie: K_g^r - liczba systemów rodzaju r /u, m, d/ dla całej gospodarki

K_w^r - liczba systemów rodzaju /u, m, d/ ustalonych bezpośrednio dla wybranego działu gospodarki

L_w - liczba zatrudnionych w wybranym dziale gospodarki

w_i^r - współczynnik wagowy /porównaj wzór 1/

r -- systemy u, m, d.

W obliczeniach porównawczych /$K_w^r \cdot w_i^r \cdot \frac{L_i}{L_w}$/ uwzględniono stosunek liczby zatrudnionych w danym i-tym dziale, do zatrudnionych w dziale wybranym - w.

Bezpośredniego obliczenia przewidywanego zapotrzebowania na systemy komputerowe dokonano dla przemysłu. Na podstawie Rocznika Statystycznego uzyskano liczbę przedsiębiorstw w podziale na pięć klas: bardzo małe, małe, średnie, duże i bardzo duże /tab.1/. Wytypowano następujące rodzaje zastosowań:

a/ zarządzanie,
b/ finanse i księgowość
c/ biurotyka
d/ laboratoria, kontrola jakości
e/ techniczne przygotowanie produkcji /TPP/
f/ komputerowe wspomaganie projektowania KWP /CAD/.

Następnie, przy pomocy ekspertów, ustalono ile poszczególnych rodzajów systemów powinno przypadać na dane zastosowanie w przedsiębiorstwie danej klasy. Wartości te przedstawiono w tab.1.

Dla pozostałych działów gospodarki liczbę systemów oblicza się wg wzoru /1/. Przyjęte dane przedstawia tab.2. W tabeli tej przytoczono zaczerpnięte z Rocznika Statystycznego z 1984 r. dane odnoczące się do zatrudnienia w 1983 r. Na marginesie należy zaznaczyć, że w różnych tabelach w tym roczniku /tab.5/18/ i tab. 2/85// podane są różne liczby zatrudnionych. Ponadto z analizy innych danych zaczerpniętych z tegoż Rocznika Statystycznego wynika, że zatrudnienie w gospodarce uspołecznionej zmniejsza się. Przyjęto więc, że tendencja ta będzie się utrzymywała również do roku 2000. Za podstawę przyjęto liczby pośrednie z obu wymienionych tabel.

Tab.1. Rodzaj i liczba systemów komputerowych w zastosowaniach danego typu dla przemysłu w roku 2000 /ustalenia ekspertów/

Lp.	Rodzaj placówki gospodarczej ze względu na zatrudnienie	Liczba	Systemy komputerowe w dziedzinach						Liczba systemów komputerowych
			zarządzanie	finanse i księgowość	biurotyka	laboratoria	TPP	KWP /CAD/	
1	bardzo małe - 100 osób	32 024	1 u			1 u			64 048 u
2	małe 101-500 osób	7 133	1 u		1 u	1 u	1 u		28 532 u
3	średnie 501 - 1000 osób	1 767	1 u	1 u	1 u	1 u	1 u 1 m	5 u	17 670 u 1 767 m
4	duże 1001 - 2000 osób	1 056		4 u 1 m	3 u	1 u 1 m	2 u 1 m	6 u	16 866 u 4 168 m
5	bardzo duże 2001 -	779		5 u 1 m	4 u	2 u 1 m 1 d	3 u 1 m	6 u 1 m	15 580 u 3 316 m 779 d
	Ogółem	42 759							142 696 u 9 251 m 779 d

u - system mikrokomputerowy
m - system minikomputerowy
d - średni i duży system komputerowy

TPP - technologiczne przygotowanie produkcji
KWP /CAD/ - komputerowo wspomagane projektowanie

/Wobec braku danych dotyczących liczby przedsiębiorstw w roku 2000, przyjęto dane wg Rocznika Statystycznego 1984 - tab. 39/318/ i 42/322//

Tab.3 zawiera ostateczną prognozę zapotrzebowania na przyjęte trzy rodzaje systemów. Należy zauważyć, że występująca w tej tablicy pozycja: "Oświata i wychowanie" dotyczy liczby systemów potrzebnych przy wspomaganiu innej pozadydaktycznej działalności.

Do dalszych rozważań przyjęto ostatecznie, że potrzeby dotyczące systemów w roku 2000 będą wynosiły:

a/ systemy mikrokomputerowe - 300 tys. sztuk

b/ systemy minikomputerowe - 18 tys. sztuk

c/ średnie i duże systemy komputerowe - 1,5 tys. sztuk.

Prognozowanie zapotrzebowania na sprzęt peryferyjny dokonano na podstawie przyjętego typowego zestawu na system danego rodzaju /u, m, d/ oraz na podstawie dodatkowych założeń, ujmujących w ilu procentach wszystkich systemów danego rodzaju będą występowały określone odchylenia od konfiguracji typowych /pkt. "Metoda prognozy - przyjęte założenia"/.

Tab.2. Przyjęte dane do prognozowania zapotrzebowania na komputery w gospodarce uspołecznionej /w roku 2000/

Lp.	Dział gospodarki	Zatrudnienie w gospodarce uspołecznionej w tys.			$\frac{L_i}{L_w}$	Przyjęte wagi xx/ w_i dla klas komputerów		
		1983 tab. 5/18/	1983 tab. 2/85/	2000 przewidywania		u	m	d
1	2	3	4	5	6	7	8	9
1	Przemysł	4 442	4 605.0	4 500	1	1	1	1
2	Budownictwo	1 088	1 098.3	1 200	0.267	0.8	0.8	0.6
3	Rolnictwo	854	1 032.0	900	0.200	0.6	0.5	0.6
4	Leśnictwo	150	157.8	100	0.222	0.5	0.4	0.4
5	Transport	882	881.9	900	0.200	0.7	0.6	0.7
6	Łączność	163	163.2	100	0.022	0.8	0.6	0.6
7	Handel wewnętrzny	1 084	1 229.9	900	0.200	0.5	0.6	0.5
8	Handel zagraniczny	30	29.6	30	0.007	0.7	0.6	0.5
9	Gospodarka komunalna	334	333.9	300	0.067	0.6	0.4	0.4
10	Gospodarka mieszkaniowa	202	202.2	200	0.044	0.4	0.4	0.4
11	Nauka i rozwój techniki	112	112.0	100	0.022	0.9	0.6	0.7
12	Oświata i wychowanie	856	861.5	800	0.187	0.9	0.4	0.5
13	Kultura i sztuka	82	82.9	80	0.019	0.3	0.4	0.4
14	Ochrona zdrowia i opieka społeczna	680	679.6	500	0.111	0.9	0.5	0.5
15	Kultura fizyczna, turystyka i wypoczynek	91	95.6	90	0.020	0.5	0.3	0.3
16	Administracja państwowa i wymiar sprawiedliwości	189	228.9	200	0.044	0.8	0.6	0.7
17	Finanse i ubezpieczenia	121	153.6	100	0.022	0.9	0.7	0.6
	Ogółem	11 568	12 147.7	11 000				

x/ Rocznik Statystyczny - 1984 r.

xx/ ustalenia ekspertów

Tab.3. Prognozowanie zapotrzebowania gospodarki uspołecznionej na systemy komputerowe różnych rodzajów w roku 2000

Lp.	Dział gospodarki	Liczba systemów komputerowych w sztukach		
		mikrokomputery	minikomputery	duże i średnie komputery
1	Przemysł	142 696	9 251	779
2	Budownictwo	30 080	1 976	125
3	Rolnictwo	17 124	925	93
4	Leśnictwo	1 570	81	7
5	Transport	19 977	1 110	109
6	Łączność	3 198	120	10
7	Handel wewnętrzny	14 270	2 010	78
8	Handel zagraniczny	700	399	3
9	Gospodarka komunalna	4 880	248	21
10	Gospodarka mieszkaniowa	2 511	163	14
11	Nauka i rozwój techniki	2 825	122	12
12	Oświata i wychowanie	24 016	692	73
13	Kultura i sztuka	813	70	6
14	Ochrona zdrowia i opieka społeczna	14 255	513	43
15	Kultura fizyczna, turystyka i wypoczynek	1 427	55	5
16	Administracja państwowa i wymiar sprawiedliwości	5 023	244	24
17	Finanse i ubezpieczenia	2 825	142	10
	Ogółem	288 190	17 761	1 412

Tab.4. Prognoza zapotrzebowania na sprzęt peryferyjny dla systemów mikrokomputerowych w roku 2000 /gospodarka uspołeczniona/

Lp.	Klasa sprzętu	Rodzaj sprzętu	Liczba tys.szt.	Razem tys.szt.
1	Drukarki	litery alfabetu polskiego	54.0	270.0
		litery alfabetu łacińskiego	216.0	
2	Klawiatury	litery alfabetu polskiego	60.0	300.0
		litery alfabetu łacińskiego	240.0	
3	Monitory	semigraficzne	270.0	300.0
		graficzne	30.0	
4	Pamięć dyskietkowa			345.0
5	Dyski twarde	Winchester		120.0
6	Pamięć back-up	Streamer		120.0
7	Plotery			30.0
8	Pamięć magnetyczna kasetowa			45.0
9	Pamięć magnetyczna szpulowa			1.5
10	Digitizery			15.0

Tab.5. Prognoza zapotrzebowania na sprzęt peryferyjny dla systemów minikomputerowych w roku 2000 /gospodarka uspołeczniona/

Lp.	Klasa sprzętu	Liczba /tys.sztuk/
1	Drukarki	18.0
2	Klawiatury	36.0
3	Monitory	36.0
4	Pamięć dyskietkowa	45.0
5	Dyski twarde	39.6
6	Taśmy magnetyczne szpulowe	18.0
7	Plotery	1.8
8	Pamięć magnetyczna kasetowa	18.0
9	Digitizery	1.8

Tab.6. Prognoza zapotrzebowania na sprzęt peryferyjny dla średnich i dużych systemów komputerowych w roku 2000 /gospodarka uspołeczniona/

Lp.	Klasa sprzętu	Liczba /tys.szt./
1	Procesor teleprzetwarzania	1.5
2	Lokalne monitory ekranowe	12.0
3	Zdalne monitory ekranowe	15.375
4	Klawiatury	27.375
5	Drukarki wierszowe	2.85
6	Drukarki trwałej kopii	7.5
7	Dyski twarde	11.85
8	Jednostki pamięci taśmowej szpulowej	9.0

Prognoza liczby systemów komputerowych w procesie nauczania

Program oświatowy dla szkół podstawowych i średnich wymaga specjalnego mikrokomputera. Przyjęto dwie typowe konfiguracje dla szkół:

- konfiguracja I: a/ monitor semigraficzny
  b/ klawiatura - litery alfabetu polskiego
  c/ pamięć zewnętrzna - magnetofon
- konfiguracja II: a/ monitor graficzny
  b/ klawiatura - litery alfabetu języka polskiego
  c/ pamięć zewnętrzna - pamięć dyskietkowa
  d/ ploter - A3
  e/ wejście graficzne - prosty digitizer
  f/ drukarka z literami alfabetu języka polskiego

Podstawą oszacowania liczby mikrokomputerów szkolnych było przyjęcie dwóch strategii wprowadzania sprzętu do szkół.

Strategia I

Mikrokomputer jest traktowany jako "wyposażenie osobiste ucznia". Założono, że na 1 mikrokomputer, w zależności od rodzaju szkoły, będzie przypadała pewna liczba uczniów /por.tab.8/.

Strategia II

W każdej szkole tworzone jest laboratorium komputerowe o przyjętej liczbie mikrokomputerów.

Prognozy dokonano na podstawie danych przedstawionych w tab.7. Ponieważ szkoły dla pracujących działają na zapleczu szkół dla niepracujących, to do obliczeń użyto danych o szkolnictwie dla niepracujących.

Tab.7. Liczba szkół i uczniów w r. 1983/84

Dane zaczerpnięte z tab.1/659/ i 3/661/ - Rocznika Statystycznego 1984

Lp.	Rodzaje szkół	Liczba szkół 83/84		Liczba uczniów 83/84	
		ogółem	dla niepracujących	ogółem	dla niepracujących
1	Podstawowe	15 981	15 768	4632 258	4 616 701
2	Niepełne średnie /zawodowe/	3 401	3 308	722 814	717 409
3	Licea ogólnokształcące	1 144	887	372 833	326 426
4	Średnie techniczne	5 087	3 305	639 622	493 630

Obliczenia przeprowadzono dla trzech wariantów:

Wariant I - strategia I - szkoły podstawowe - 10 uczniów/1 mikrokomputer
- licea ogólnokształcące - 5 uczniów/1 mikrokomputer
- niepełne średnie - 10 uczniów/1 mikrokomputer
- średnie techniczne - 3 uczniów/1 mikrokomputer

Wariant 2 - strategia II - liczba mikrokomputerów w szkolnym laboratorium - 30 szt. /liczba została zaproponowana w trakcie dyskusji specjalistów z Zakładów Zrzeszenia w dniu 85.01.17/

Wariant 3 - strategia II - liczba mikrokomputerów w laboratorium - 10 sztuk /przyjęto za propozycjami Zrzeszenia/

Wyniki obliczeń trzech wariantów przedstawiono w tab.8.

Tab.8. Prognozowana liczba systemów mikrokomputerowych dla szkolnictwa

Lp.	Rodzaje szkół	Liczba systemów /tys.szt./			Wskaźnik: $\frac{\text{liczba uczniów}}{\text{mikrokomputer}}$		
		Wariant 1	Wariant 2	Wariant 3	Wariant 1	Wariant 2	Wariant 3
1	Podstawowe	461.7	473.0	157.7	10	10	30
2	Licea ogólnokształcące	65.8	26.6	8.9	5	12	37
3	Niepełne średnie /zawodowe/	71.7	99.2	33.1	10	7	21
4	Średnie techniczne	164.5	99.2	33.1	3	5	15
	Ogółem	763.7	698.0	232.8	8	9	26

Ponieważ wariant 1 i 2 dają podobne wyniki liczbowe, do dalszych rozważań /obliczenia sprzętu peryferyjnego/ przyjęto, że potrzeby szkolnictwa /K_{II}/ w roku 2000 kształtują się na poziomie 760 tys. mikrokomputerów.

W tab.9 przedstawiono potrzeby odnoszące się do różnych przyjętych konfiguracji w zależności od rodzaju szkoły, a w tab.10 wynikające z tego potrzebne liczby sprzętu peryferyjnego w roku 2000.

Tab.9. Prognozowane zapotrzebowanie na mikrokomputery do nauczania /wg konfiguracji/ w roku 2000

Lp.	Rodzaj szkoły	Konfiguracja /tys.szt./	
		I	II
1	Podstawowe	460.0	-
2	Licea ogólnokształcące	32.5	32.5
3	Niepełne średnie /zawodowe/	71.0	-
4	Średnie techniczne	-	164.0
	Ogółem	563.5	196.5

Tab.10. Sprzęt peryferyjny dla mikrokomputerów szkolnych w roku 2000

Lp.	Klasa sprzętu	Liczba /tys.szt./
1	Drukarki - litery alfabetu jęz. polskiego	196.5
2	Klawiatury - litery alfabetu jęz. polskiego	760.0
3	Monitory - semigraficzne - graficzne	563.5 196.5
4	Pamięć dyskietkowa	196.5
5	Plotery	196.5
6	Magnetofon kasetowy	563.5
7	Digitizery	196.5

Program produkcji na lata 1988-2000

W tab.11 zestawiono prognozowane zapotrzebowanie na sprzęt peryferyjny systemów mikrokomputerowych dla gospodarki /g/ i nauczania /n/, a w tab.12 pokazano oszacowane ilości sprzętu peryferyjnego we wszystkich przyjętych trzech rodzajach systemów komputerowych wg uogólnionych klas sprzętu. Chcąc odpowiedzieć na pytanie: "Jak wielka powinna być produkcja, aby zapewnić zaspokojenie potrzeb w okresie 1988-2000?" przyjęto poniższe założenia.

Założenia:

- Okres amortyzacji sprzętu wynosi 7 lat
- Potrzeby rosną liniowo w rozpatrywanym okresie czasu
- Produkcja zaspokajająca prognozowane potrzeby rozpoczyna się 1 stycznia 1988 roku

Wynikający z tych założeń wymagany poziom produkcji przedstawiono na rys. 3 ÷ 11. Rys. 3 ÷ 10 odnoszą się do systemów mikrokomputerowych, a rysunek 11 do systemów dużych i średnich komputerów przy czym na rys.3 /górna część wykresu/ przeprowadzono analizę różnych wariantów rozkładu zapotrzebowania na systemy mikrokomputerowe w latach 1988-2000.

Tab.11. Zbiorcze zestawienie prognozowanych potrzeb na sprzęt peryferyjny dla mikrokomputerów w roku 2000

Lp.	Klasa sprzętu	Rodzaj sprzętu	Zastosowanie	Sprzęt /w tys.szt./	Ogółem /tys.szt./	
Ø	Mikrokomputer		g	300.0	1060.0	
			n	760.0		
1	Drukarki	litery alfabetu polskiego	g	54.0	250.5	466.5
			n	196.5		
		litery alfabetu łacińskiego	g	216.0	216.0	
			n	-		
2	Klawiatury	litery alfabetu polskiego	g	60.0	820.0	1060.0
			n	760.0		
		litery alfabetu łacińskiego	g	240.0	240.0	
			n	-		
3	Monitory	semigraficzne	g	270.0	833.5	1060.0
			n	563.5		
		graficzne	g	30.0	226.5	
			n	196.5		
4	Pamięć dyskietkowa		g	345.0	541.5	
			n	196.5		
5	Dyski twarde	Winchester	g	120.0	120.0	
			n	-		
6	Pamięć back-up	Streamer	g	120.0	120.0	
			n	-		
7	Pamięć magnetyczna kasetowa		g	45.0	45.0	
			n	-		
8	Magnetofon kasetowy		g	-	563.5	
			n	563.5		
9	Pamięć magnetyczna szpulowa		g	1.5	1.5	
			n	-		
10	Plotery		g	30.0	226.5	
			n	196.5		
11	Digitizery		g	15.0	211.5	
			n	196.5		

g - gospodarka uspołeczniona

n - nauczanie

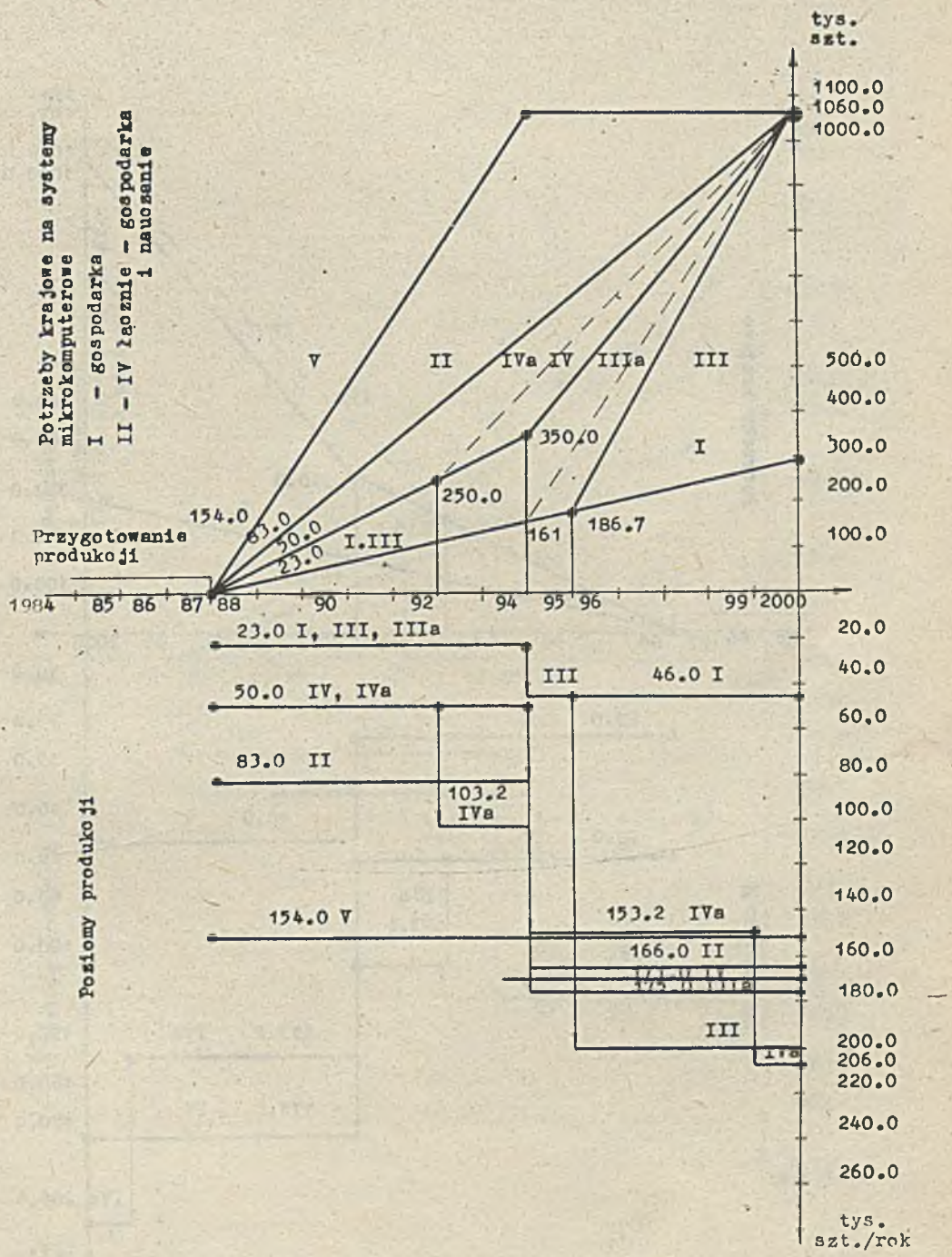
Tab.12. Zapotrzebowanie na środki techniki obliczeniowej w roku 2000 /łącznie z programem komputerowego wspomagania nauczania/

Lp.	Klasa sprzętu	Sprzęt w tys.szt.		
		u	m	d
1	Drukarki	466.5	18.0	2.85
2	Drukarki trwałej kopii	-	-	7.5
3	Klawiatury	1060.0	36.0	27.375
4	Monitory	1060.0	36.0	27.375
5	Pamięć dyskietkowa	541.5	45.0	-
6	Dyski Winchester	120.0	-	-
7	Dyski twarde	-	39.6	11.85
8	Pamięć Streamer	120.0	-	-
9	Pamięć magnetyczna kasetowa	45.0	18.0	-
10	Magnetofon kasetowy	563.5	-	-
11	Pamięć magnetyczna szpulowa	1.5	18.0	9.0
12	Plotery	226.5	1.8	-
14	Procesory teleprzetwarzania	-	-	1.5

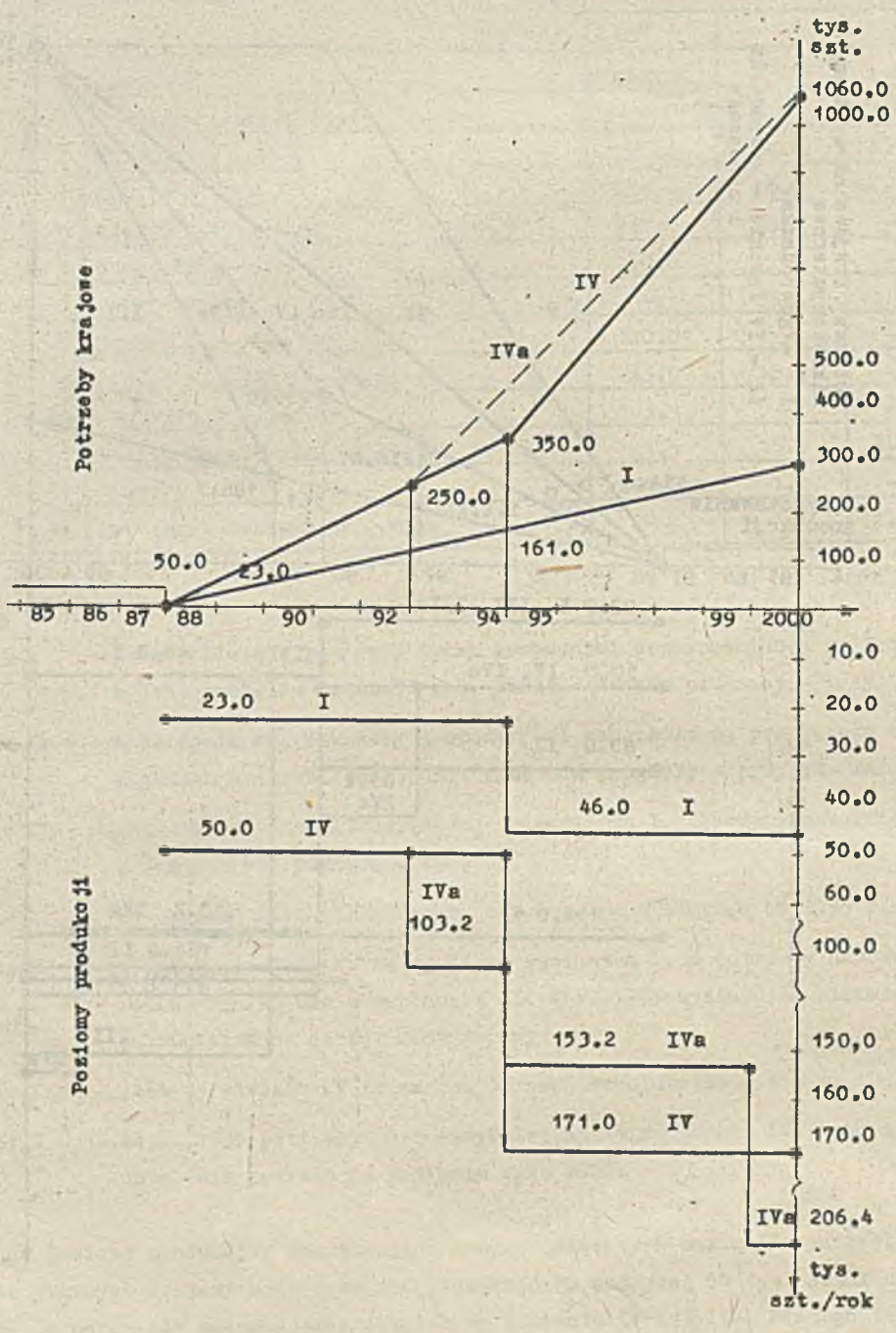
- Wariant I - uwzględnia się potrzeby tylko gospodarki uspołecznionej wg przyjętego modelu rozkładu potrzeb /zob. punkt - Metoda prognozy - przyjęte założenia/
- Wariant II - uwzględnia się potrzeby gospodarki i nauczania wg przyjętego modelu rozkładu potrzeb /zob. punkt - Metoda prognozy - przyjęte założenia/
- Wariant III - potrzeby gospodarki narodowej wg wariantu I, potrzeby szkolnictwa pojawiają się dopiero w 1996 r.
- Wariant IIIa - to wariant III, ale potrzeby szkolnictwa występują od 1995 r.
- Wariant IV - potrzeby gospodarki są zgodne z wariantem I, a potrzeby nauczania do 1994 r. rosną wolniej niż w wariancie II. W r. 1995 występuje zwiększenie potrzeb szkolnictwa na sprzęt komputerowy
- Wariant IVa - jest to wariant IV ze zmianą potrzeb szkolnictwa już w r. 1993
- Wariant V - do r. 1995 potrzeby rosną szybciej niż w wariancie II, a później występuje ustalenie potrzeb na poziomie roku 2000.

Porównując poziomy produkcji, gwarantujące zaspokojenie tych wariantów potrzeb, dochodzi się do wniosku, że przemysł krajowy może rozwinąć produkcję co najwyżej 50 tys. sztuk systemów mikrokomputerowych, co odpowiada zaspokojeniu potrzeb wg wariantu IV lub IVa. Dlatego też do ustalenia poziomu produkcji sprzętu peryferyjnego dla mikrokomputerów rozpatruje się wariant IV i IVa oraz wariant I jako przypadek skrajny, zapewniający jedynie zaspokojenie potrzeb gospodarki.

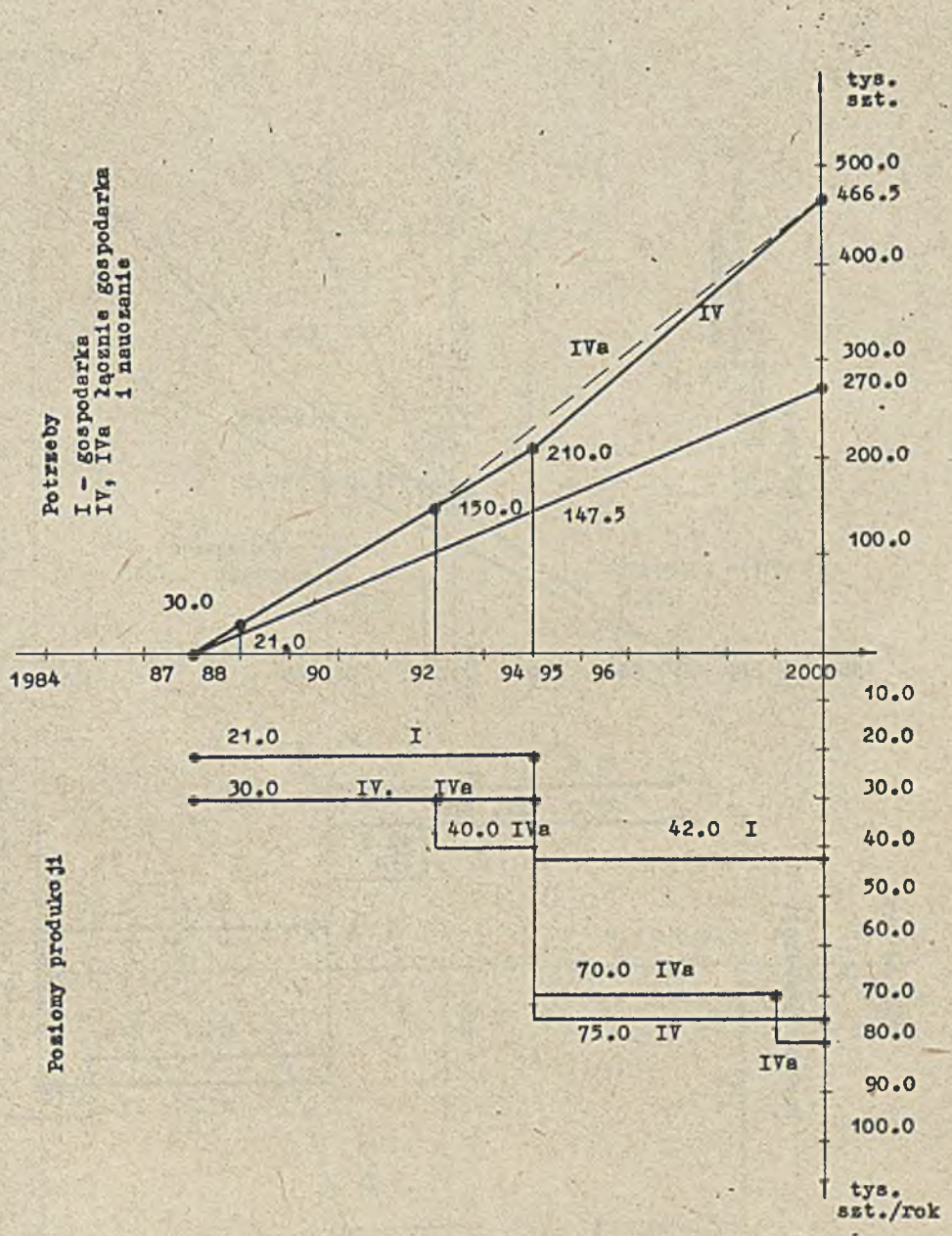
Podobne rozważania można przeprowadzić w odniesieniu do systemów minikomputerowych i systemów średnich i dużych komputerów.



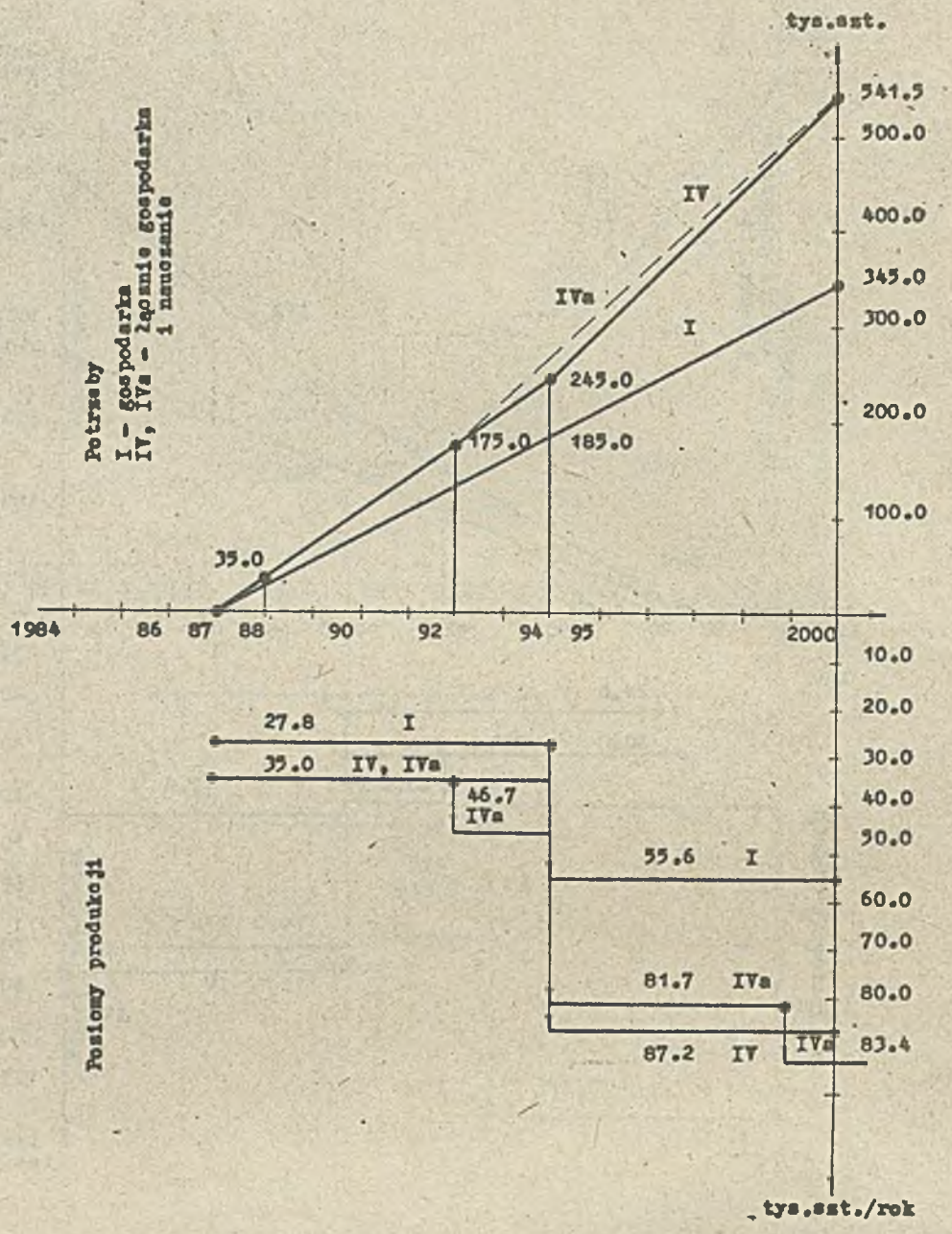
Rys.3. Warianty potrzeb krajowych i poziomy produkcji systemów mikrokomputerowych



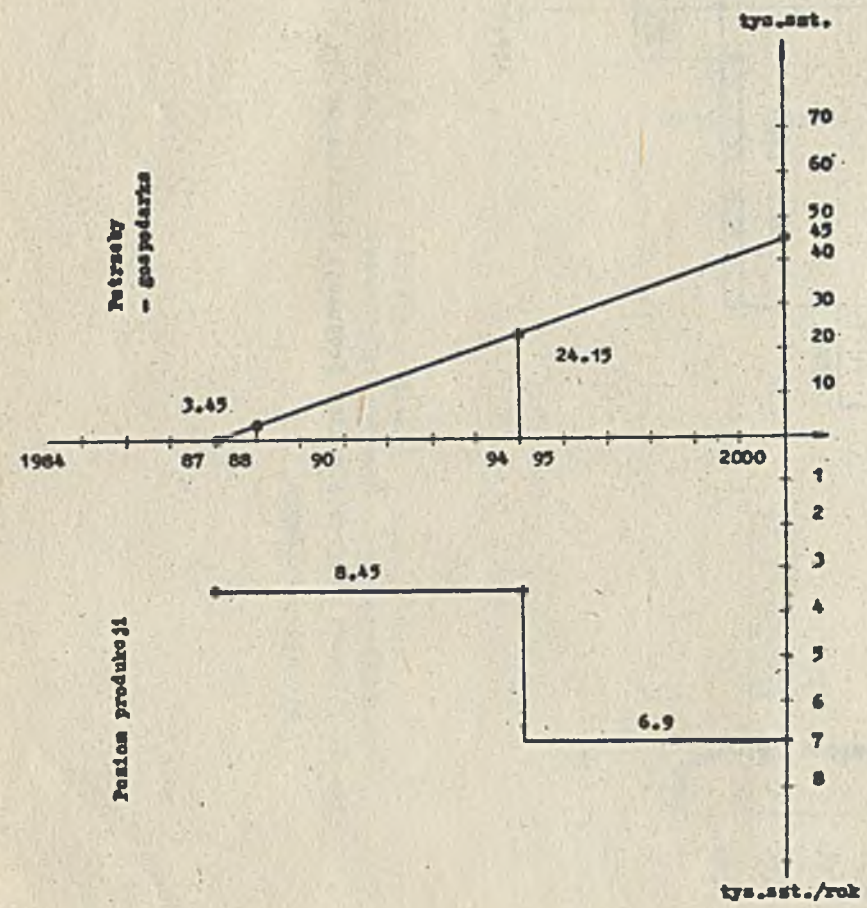
Rys. 4. Potrzeby krajowe i poziom produkcji klawiatur i monitorów dla systemów mikrokomputerowych



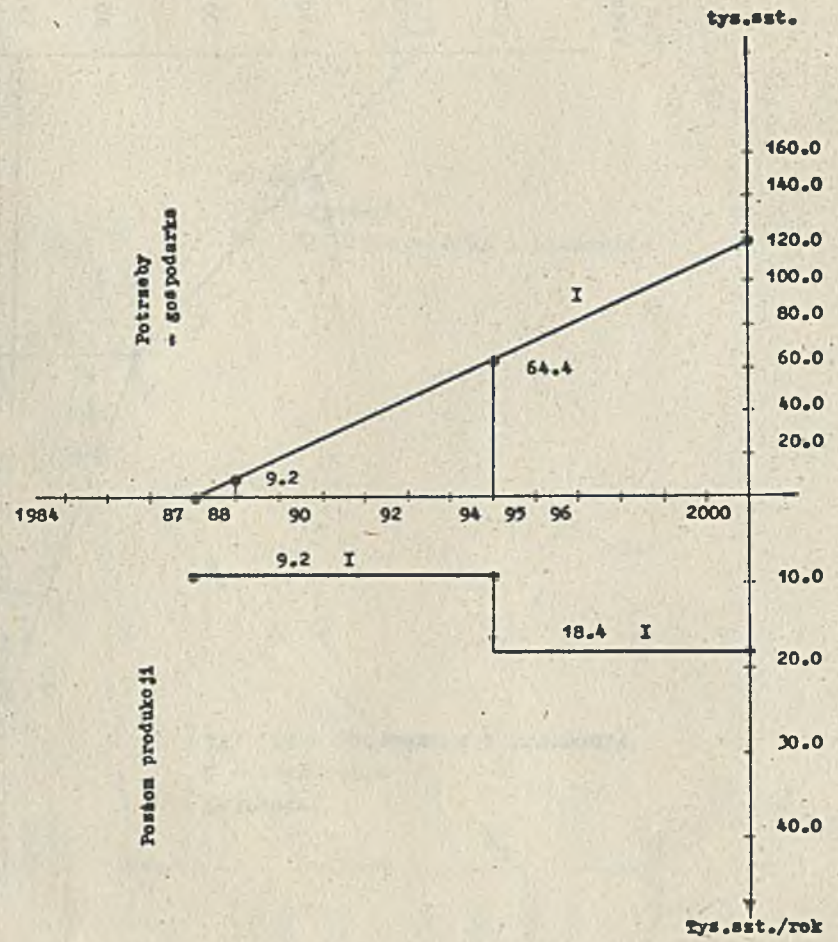
Rys. 5. Potrzeby krajowe i poziomy produkcji drukarek dla systemów mikrokomputerowych



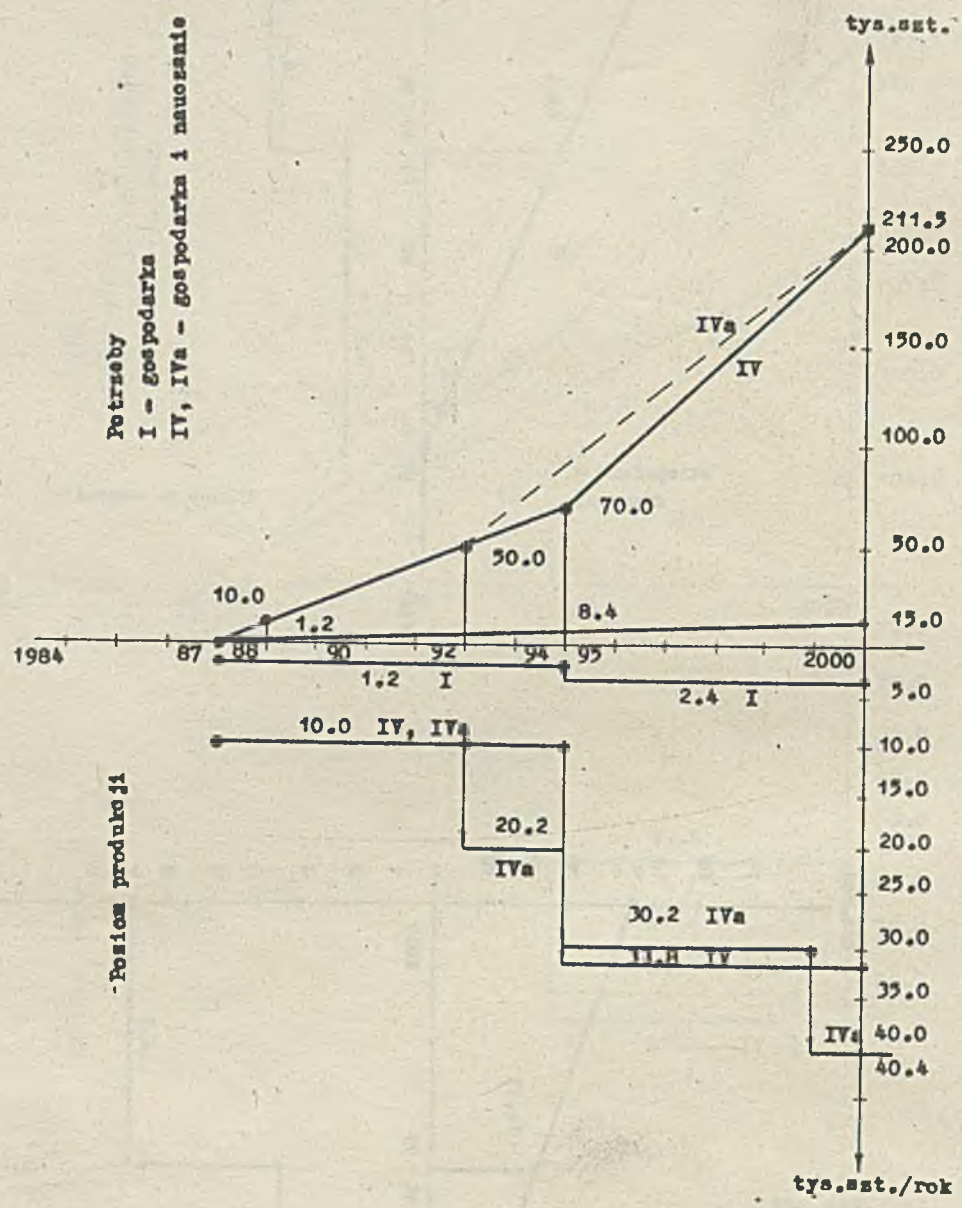
Rys. 6. Potrzeby krajowe i poziom produkcji pamięci dyskietkowych dla systemów mikrokomputerowych



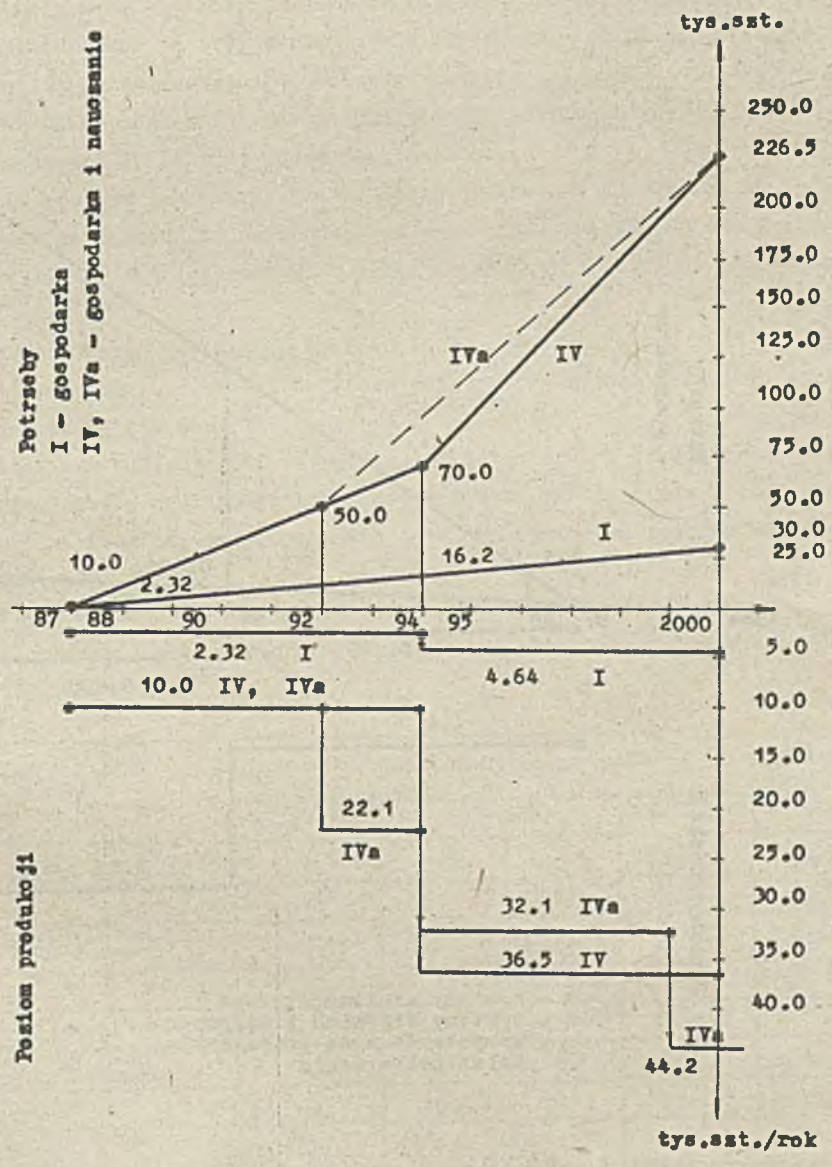
Rys. 7. Potrzeby krajowe i poziom produkcji pamięci kasetowych dla systemów mikrokomputerowych



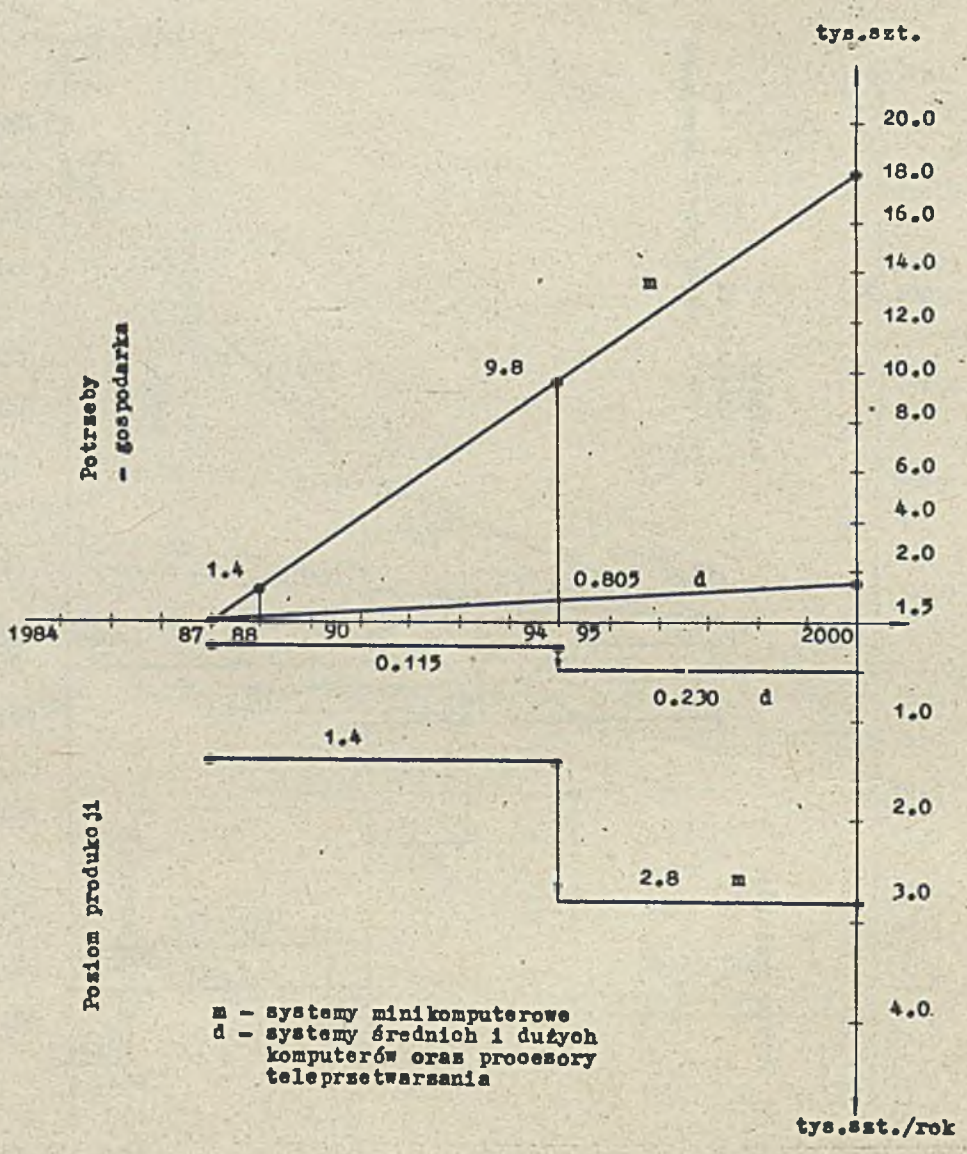
Rys. 8. Potrzeby krajowe i poziom produkcji pamięci, zarówno typu Winchester, jak i back-up dla systemów mikrokomputerowych



Rys. 9. Potrzeby krajowe i poziom produkcji digitizerów dla systemów mikrokomputerowych



Rys. 10. Potrzeby krajowe i poziomy produkcji ploterów dla systemów mikrokomputerów



Rys. 11. Potrzeby krajowe i poziom produkcji systemów minikomputerowych oraz systemów średnich i dużych komputerów

Porównanie

Prognoza SKI

W opracowaniu SKI - "Prognoza zapotrzebowania na sprzęt informatyczny w Polsce na lata 1984 - 2000" liczbową prognozę zapotrzebowania wykonano metodą prognozowania strukturalnego. Punktem wyjścia jest tu założenie dotyczące wielkości nakładów. Istotne informacje zawarte w tym opracowaniu przedstawiono w tab.13. Pokazano tu trzy warianty wielkości nakładów na zastosowania informatyki.

Wariant A zawiera dla całego horyzontu czasowego prognozy, tj. lata 1984-2000 wskaźnik udziału nakładów inwestycyjnych na zastosowania informatyki do nakładów inwestycyjnych ogółem równy 0.27%. Jest to najgorszy z możliwych wariantów, jakie mogą zaistnieć.

Wariant B uwypukla problem rekonstrukcji technicznej, chociaż opiera się na tych samych założeniach procesu rozwojowego gospodarki narodowej, co wariant A. Zwiększono wskaźnik udziału nakładów inwestycyjnych dla zastosowań informatyki w nakładach ogółem z 0.27% do 0.54% w pięciolatce 1996-2000, tj. dwukrotnie.

Wariant C jest powtórzeniem wariantu B do roku 1990, a poczynając od 1991 r. udział nakładów inwestycyjnych na zastosowania informatyki do inwestycji ogółem wynosi 4-krotnie więcej niż w wariancie A i dwukrotnie więcej niż w wariancie B.

Wszystkie przedstawione warianty SKI są wariantami ostrożnymi i przedstawiają nie tyle aspiracje /potrzeby/, co przewidywane możliwości gospodarcze kraju. Warianty A i C stanowią kres dolny i kres górny pola prognoz. Prognozy znajdujące się poza tym obszarem, zdaniem SKI, mają znikome prawdopodobieństwo realizacji.

Tab.13. Zestawienie prognozowanych nakładów i zainstalowanych systemów w latach 1984 - 2000

Wariant	Klasa systemu	Liczba zainstalowanych systemów i poniesione nakłady				Ogółem 1984--2000	Stan posiadania w 2000 r.
		84-85	86-90	91-95	96-2000		
A	Komputery duże i średnie	15	40	45	50	150	95
	Minikomputery	140	390	450	520	1500	970
	Mikrokomputery	1200	3400	3900	4500	13000	8400
	Nakłady w mld zł	3.0	16.4	19.0	22.1	63.4	-
	% udział w nakładach ogółem	0.27	0.27	0.27	0.27	0.27	-
B	Komputery duże i średnie	15	55	70	100	240	170
	Minikomputery	140	460	720	1060	2380	1780
	Mikrokomputery	1200	4000	6300	9100	20600	15400
	Nakłady w mld zł	3.0	19.7	30.4	44.2	100.0	-
	% udział w nakładach ogółem	0.27	0.324	0.378	0.540	0.373	-
C	Komputery duże i średnie	15	55	100	210	380	310
	Minikomputery	140	460	1000	2100	3700	3100
	Mikrokomputery	1200	4000	8800	18200	32200	27000
	Nakłady w mld zł	3.0	19.7	42.5	88.4	156.3	-
	% udział w nakładach ogółem	0.27	0.324	0.529	1.080	0.745	-

/wg cen 1982 r./

Wskaźniki Centrum Koordynacyjnego MK ETO

Centrum Koordynacyjne zaproponowało rozwój przemysłu komputerowego w takim tempie, aby w roku 2000 osiągnąć poniższe wartości wskaźników - liczba komputerów/100 tys. zatrudnionych w gospodarce narodowej:

- systemy mikrokomputerowe - 1400
- systemy minikomputerowe - 175
- średnie i duże komputery - 30.

Ocena wyników

W tab.14 zestawiono wyniki prognozowania potrzeb metodą autorów /bez programu dla nauczania/ i wyniki prognozy SKI oraz liczby systemów wynikające z zalecanych przez Centrum MK ETO wartości wskaźników: liczba systemów komputerowych/100 tys. zatrudnionych. Wyniki metody autorów i metody SKI różnią się o rząd wielkości. Zamieszczono też obliczone dla tych wyników wskaźniki /"liczba ko komputerów/100 tys. zatrudnionych". Dla silniejszego podkreślenia omawianych różnic warto zwrócić uwagę, że średnia roczna produkcja dla przewidywanych przez SKI wariantów A, B, C nakładów przedstawia się jak w tab.15.

Z danych zestawionych w tab.14 widać, że prognozy potrzeb wg autorów i wg Centrum Koordynacyjnego są zbliżone.

Tab.14. Porównanie wyników prognoz

Lp.	Wyszczególnienie	Liczba systemów komputerowych w roku 2000 /szt./			Wskaźnik: liczba systemów kom- x/ puterowych na 100 tys. zatrudn. w roku 2000		
		d	m	u	d	m	u
1	Wariant A - SKI	95	970	8400	0.475	4.85	42
2	Wariant B - SKI	170	1780	15400	0.85	8.9	177
3	Wariant C - SKI	310	3100	27000	1.55	15.5	135
4	Prognoza potrzeb	1412	17761	188190	7.06	89.8	1441
5	wg Centrum Koordynacyjnego MK ETO	6000	35000	280000	30	175	1400

x/ do wyliczeń przyjęto 20 mln zatrudnionych w gospodarce

W tab.16 przedstawiono wartości wskaźnika - liczba komputerów/100 tys. zatrudnionych obliczonego na podstawie wyników prognozy autorów z wartościami średnimi dla krajów: Bułgaria, Węgry, NRD, Kuba, Polska, ZSRR, Czechosłowacja, zakładanymi w prognozach tych krajów. Jak widać, wartości wynikające z prognozy autorów są poniżej przedstawionej średniej.

Tab.15. Srednia roczna produkcja dla zaspokojenia potrzeb wynikających z prognoz SKI

Lp.	Klasa systemu	Średnia roczna produkcja /szt./		
		A	B	C
1	Systemy duże	10	16	25
2	Minikomputery	100	158	246
3	Mikrokomputery	866	1373	1800

Tab.16. Wartości wskaźnika: liczba komputerów/100 tys. zatrudnionych w gospodarce narodowej

Wyszczególnienie	Wartość wskaźnika			Klasa systemów
	1985	1990	2000	
Średnia wartość wskaźnika x/	130	337	1772	mikrokomputery
Polska	23	356	1441	
Średnia wartość wskaźnika x/	37	72	213	mini, duże i średnie komputery
Polska	16	43	96	

x/ Średnia wartość wyliczona z prognozowanych wartości wskaźników dla krajów członków MK ds TO /Bułgaria, Węgry, NRD, Kuba, Polska, ZSRR, Czechosłowacja/

Podsumowanie

Przedstawiona przez autorów metoda prognozowania krajowego zapotrzebowania na systemy komputerowe jest pewną propozycją metodyczną. Jej istota polega na analizie różnych dziedzin zastosowań. Konkretne wartości przyjmowane przez autorów należy traktować jako pewne przybliżenie, które może być w kolejnych iteracjach uszczegółowione i uściślone. Podobne podejście spotyka się wprawdzie w literaturze [3], ale prezentowane rozważanie jest pomysłem oryginalnym.

Przyjęte przez autorów wartości wyjściowe mogą podlegać dyskusji, niemniej duża zbieżność wyników z propozycjami Centrum Koordynacyjnego czyni propozycje autorów wiarygodnymi.

Na tym tle przewidywania sformułowane przez SKI muszą budzić poważny niepokój.

Przyjęcie przez inne kraje socjalistyczne tempa wzrostu komputeryzacji zaproponowanego przez Centrum Koordynacyjne, przy pozostaniu Polski na poziomie proponowanym przez SKI, spowoduje jeszcze większe niż obecnie opóźnienie naszej gospodarki.

Literatura

[1] Notatka CKI ds TO na temat wyboru tempa i propozycji produkcji komputerowej do roku 2000

[2] Prognoza zapotrzebowania na sprzęt infbrmatyczny w Polsce na lata 1984-2000. Sekretariat Komitetu Informatyki

[3] J.Kowalczyk: O określeniu zapotrzebowania Gospodarki Narodowej Krajów Socjalistycznych na Środki Techniki Obliczeniowej. Biuletyn "MERA" 1983 nr 2

[4] Rocznik Statystyczny GUS - 1984 r.

TECHNIKI KOMPUTEROWE 4/85

mgr Andrzej SZEWCZUK
Instytut Maszyn Matematycznych

Quasi-równoległość i symulacja w Pascalu

Poniższe opracowanie oparte jest na pracy magisterskiej mgr Andrzeja Szewczuka "Quasi-równoległość i symulacja w Pascalu - język Simpas i jego implementacja w systemie OS/JS" (napisanej pod kierunkiem dr Pawła Gburzyńskiego w Zaocznym Studium Matematyki Uniwersytetu Warszawskiego), za którą Autor otrzymał wyróżnienie w I Ogólnopolskim Konkursie Polskiego Towarzystwa Informatycznego na najlepsze prace dyplomowe z dziedziny informatyki. Konkurs był zainicjowany i zorganizowany przez Koło PTI we Wrocławiu w 1984 r.
Komitet Redakcyjny gratuluje Autorowi i Promotorowi.

Wstęp

W ostatnim dziesięcioleciu Pascal stał się jednym z najpopularniejszych języków programowania; opierając się na nim zdefiniowano inne języki - także do programowania współbieżnego i symulacji.

Tematem opracowania jest propozycja kolejnego rozszerzenia Pascala - języka Simpas, wyposażonego w konstrukcje umożliwiające programowanie quasi - równoległe i symulację procesów współbieżnych. Wprowadzone mechanizmy zostały zaczerpnięte z języka Simula 67 [2] - stąd nazwa Simpas.

Quasi-równoległość nie jest współbieżnością fizyczną (jednoczesne wykonywanie instrukcji różnych fragmentów programu), a jedynie logiczną. W danej chwili może istnieć wiele jednostek równoległych programu - korutyn, z których każda wykonywana jest sekwencyjnie. Możliwe jest natomiast zawieszenie wykonania korutyny, a później - wznowienie (od miejsca zawieszenia). Jednak w danym momencie są wykonywane instrukcje co najwyżej jednej korutyny. Quasi-równoległość polega więc na przeplataniu się w czasie wykonania fragmentów (ciągów instrukcji) różnych korutyn. Na podstawie quasi-równoległości tworzy się następnie mechanizmy quasi-równoległej symulacji. Wprowadza się pojęcie czasu (symulowanego), sposoby jego modelowania oraz synchronizacji procesów.

Simpas powstał z Pascala przez dodanie nowych słów kluczowych oraz typów, zmiennych, procedur i funkcji standardowych (w Simpasie). Nowe słowa kluczowe są poprzedzane w tekście programu znacznikiem "%%" (podwójny znak procenta). Programy w Simpasie są wstępnie tłumaczone na Pascal przez preprocesor Simpasu, a następnie - na kod maszynowy - przez kompilator Pascala. Do programu wynikowego dołączone jest ostoczenie programowe Simpasu zapewniające wykonanie programu zgodnie z semantyką Simpasu.

Ponieważ preprocesor generuje także nowe identyfikatory, to konieczne jest zastosowanie pewnych umów i ograniczeń. Dla wyróżnienia generowanych identyfikatorów posłużę się jednym z rozszerzeń Pascala IMM*/dołączającym do zbioru liter znak '_' (podkreślenie, łącznik) - można by tego nie robić i zamiast łącznika zastosować np. ciąg liter "YYY". Zabrania się więc używania identyfikatorów zaczynających się od: "MAIN_", "REF_", "TYPE_", "XXX" i "_". Napisy nie mogą zawierać ciągu "%%".

Do opisu składni Simpasu użyję elementów notacji Backusa - Naura (BNF) z dodatkowymi umowami:

- sekwencja ujęta w nawiasy kwadratowe może wystąpić, ale nie musi,
- ciąg "...." oznacza fragment wynikający z kontekstu (zgodny ze składnią Pascala).

Standardowe symbole literowe "czystego" Pascala - np. "begin", "integer", "write" - będę pisał małymi literami, pozostałe - identyfikatory i nowe słowa kluczowe - dużymi. Identyfikatory "CLASS" i "PROCESS" są zarówno nowymi słowami kluczowymi, jak i nazwami nowych typów standardowych, ale znaczenie ich konkretnego użycia jednoznacznie wynika z kontekstu.

Implementacja Simpasu została częściowo wykonana dla Pascala IMM - uruchomiono otoczenie programowe, a testy były prowadzone przy zastosowaniu "ręcznego" tłumaczenia programów z Simpasu na Pascal.

Niniejsze opracowanie powstało na podstawie mojej pracy magisterskiej - [5]. Prezentowany tutaj język Simpas różni się nieco od wersji opisanej w mojej pracy. Wprowadzono zmiany notacyjne zwiększające wygodę programowania i czytelności tekstu programu oraz bardziej zbliżające Simpas do Simuli. Opis implementacji jest ogólniejszy i abstrahujący od konkretnego modelu komputera i kompilatora Pascala.

Język Simpas

Klasy

Korutynami w Simpasie są obiekty klas tworzone dynamicznie na podstawie deklaracji klasy obiektów, której składnia podobna jest do składni deklaracji procedury, a mianowicie:

```
%% CLASS <nazwa>(THIS:REF_<nazwa>...);
....
 [<deklaracje zmiennych lokalnych>]
 [%% ATR<deklaracje atrybutów> %% END]
....
%% BEGIN_CL
   <program klasy>
%% END_CL
```

gdzie:
- <nazwa> - unikalna nazwa klasy obiektów,

---

*/ Pascal IMM - wersja języka i kompilator (F i F1), które powstały w Instytucie Maszyn Matematycznych w Warszawie. Implementacja ta jest przeznaczona dla maszyn serii IBM 360/370 oraz Jednolitego Systemu pracujących w systemie operacyjnym OS [3], [4].

- REF_< nazwa > - typ (wskaźnikowy) referencji obiektów tej klasy (definicja tego typu jest generowana przez preprocesor),
- THIS - parametr wskazujący obiektu,
- nawiasy "BEGIN_CL" i "END_CL" pełnią podobną rolę, jak najbardziej zewnętrzne nawiasy "begin" i "end" w deklaracji procedury,
- <program klasy > - ciąg instrukcji.

Klasa może mieć więcej parametrów, jednak THIS musi być wyspecyfikowany jako pierwszy. Deklaracja klasy może zawierać także definicje stałych i typów oraz deklaracje procedur i funkcji - ich nazwy są lokalne dla treści klasy. Natomiast nazwy typów użytych w deklaracjach atrybutów muszą mieć zasięg globalny (definicja na poziomie programu głównego). Nawiasy instrukcji złożonej ("begin" i "end") procedury lokalnej powinny być poprzedzone znacznikiem "%%".

Przykład 1.

```
%% CLASS ABC (THIS:REP_ABC; Z: real);
var J: integer;
%% ATR A: real;
  B: array [1.. 100] of real;
%% END
function FFF (X: real): real; ...
   %% begin ... %% end;
%% BEGIN_CL....
   for J := 1 to 100 do B [J]:= FFF ( Z + A + J);
%% END_CL   ABC
```

gdzie:
- ABC - nazwa klasy,
- Z,J - zmienne lokalne,
- A,B - atrybuty,
- FFF - funkcja lokalna

Obiekty klas są identyfikowane za pomocą zmiennych referencyjnych. Deklaracje tych zmiennych mają normalną (pascalowską) formę.

Przykład 2.

```
X,Y: REF_ABC;
T: array [1.. 100] of REF_ABC;
```

Dla każdej zmiennej referencyjnej danej klasy można utworzyć oddzielny obiekt tej klasy. Obiekty powstają w wyniku wykonania sekwencji generującej. Niech <nk> oznacza nazwę klasy, a <ref > - zmienną referencyjną typu REF-< nk>, wówczas składnię sekwencji generującej można opisać następująco:

```
%% CREATE_CL <ref>;
[<inicjacja atrybutów>]
%% START <nk>(<ref>....);
```

Inicjacja atrybutów polega na nadaniu wartości początkowej atrybutom obiektu przy wykorzystaniu zewnętrznego dostępu do atrybutów (opisanych nieco dalej). Instrukcja "START" rozpoczyna wykonanie programu obiektu. Musi być ona zgodna z deklaracją klasy <nk>; tzn. polega to na zgodności parametrów formalnych (w deklaracji) i aktualnych (w instrukcji "START"); podobnie musi istnieć zgodność instrukcji wywołania procedury z jej deklaracją.

Przykład 2.1.c.
Utworzenie obiektu klasy ABC o referencji T [34] :

```
%% CREATE_CL  T  [34];
%% START ABC ( T [34] , 1.5 );
```

W wyniku wykonania sekwencji generującej powstaje nowy obiekt klasy, a jego referencja zostaje przypisana zmiennej <ref>. Obiekt klasy jest dynamicznym egzemplarzem klasy, ma on własny egzemplarz atrybutów i zmiennych lokalnych (zgodny z deklaracją klasy).

Z każdym obiektem związany jest jego program, którym jest program klasy działający na atrybutach i zmiennych lokalnych obiektu. Wykonanie programu obiektu może zostać zawieszone. Z drugiej strony - różne obiekty (być może tej samej klasy) są identyfikowane za pomocą różnych zmiennych referencyjnych (różne zmienne referencyjne mogą natomiast wskazywać na ten sam obiekt). Wobec tego uzasadnione jest pojęcie zewnętrznego dostępu do atrybutów obiektu. Ma ono formę odwołania się do elementu zmiennej dynamicznej, a mianowicie:

<ref>↑.<atrybut> gdzie <ref> - referencja obiektu

Przykład 3

Niech dane będą deklaracje z przykładu 2, wówczas:

```
X↑.A:=3.14;
U:=T[34]↑.B[52];
T[1]↑.A:=T[100]↑.A;
PPP(Y↑.B);
```

są syntaktycznie i semantycznie prawidłowymi, zewnętrznymi odwołaniami do atrybutów.

Zewnętrzne odwołanie do atrybutu obiektu danej klasy może wystąpić, zarówno wewnątrz, jak i na zewnątrz treści klasy. W pierwszym przypadku jest to zazwyczaj odwołanie do atrybutów innego obiektu (którego program jest zawieszony - ale niekoniecznie). Wewnątrz treści klasy do atrybutów obiektu, którego program jest wykonywany odwołujemy się tak, jak do zmiennych lokalnych - za pośrednictwem identyfikatora (patrz przykład 1). Parametr wskazujący THIS jest referencją obiektu, którego program się wykonuje.

Każdy obiekt dowolnej klasy ma dodatkowy, standardowy, generowany przez preprocesor atrybut _CL standardowego typu wskaźnikowego CLASS, który jest referencją programu tego obiektu. Jest on używany przez mechanizmy quasi - równoległości.

Systemy quasi - równoległe

Jednostką dynamiczną programu w Pascalu jest program główny lub wywołanie procedury (jednocześnie może istnieć wiele różnych jednostek dynamicznych tej samej - w sensie deklaracji - procedury - rekurencja). W Simpasie jednostką dynamiczną jest również obiekt klasy wraz ze swoim programem.

Jednostki dynamiczne programu w Simpasie mogą znajdować się w jednym z trzech stanów:

a) przyłączonym, b) samodzielnym c) zakończonym

Pascalowe jednostki dynamiczne zawsze znajdują się w stanie a), - dana jednostka jest przyłączona do tej jednostki, w której nastąpiło jej wywołanie.

W momencie generacji obiekt klasy znajduje się w stanie a). Po pierwszym wywołaniu w jego programie procedury DETACH przechodzi w stan b). Natomiast po wyczerpaniu się listy instrukcji (dojście do END_CL) przechodzi w stan c).

System quasi - równoległy (SQR) składa się z programu głównego systemu (składowa nadrzędna) i z samodzielnych obiektów klas.

Programem głównym SQR może być program główny w Simpasie lub procedura (nierekurencyjna), gdzie zamiast zewnętrznych nawiasów "begin" i "end" użyto "BEGIN_PAR" i "END_PAR". Znajduje się on zawsze w stanie samodzielnym. Zmienna standardowa MAIN_PGM (typu CLASS) jest jego referencją.

SQR powstaje w momencie rozpoczęcia wykonania programu głównego systemu (jest on wówczas jedyną składową SQR), a przestaje istnieć z chwilą zakończenia jego wykonania. W danym momencie wykonywane są instrukcje tylko jednej składowej - składowa aktywna (bieżąca). Każda składowa ma swoje sterowanie lokalne (SL), którym jest miejsce ostatnio wykonywanej instrukcji w programie składowej. W szczególności SL może wskazywać instrukcję wywołania procedury lub instrukcję "START" w sekwencji generującej. Po wykonaniu danej instrukcji SL przenosi się do następnej instrukcji - zgodnie z semantyką Pascala (pętla, skoki, instrukcje warunkowe,...). Sterowaniem systemu (SS) jest SL bieżącej składowej. Niech SP oznacza sterowanie procesora - miejsce wykonywanej instrukcji. SP może przebywać w programie składowej, procedurze z niej wywołanej lub w programie obiektu przyłączonego.

Podczas wykonywania instrukcji składowej X zachodzi równość SL(X) = SS = SP. Gdy w składowej X zostanie wykonana sekwencja generująca obiekt Y, to wówczas SL(X) i SS zatrzymują się, a SP przenosi się do programu obiektu (przyłączonego) Y. Jeśli w programie obiektu Y nie zostanie wywołana procedura standardowa DETACH, to po jego zakończeniu (przechodzi on w stan zakończony), a SP powraca do X - znowu zachodzi równość SL(X) = SS = SP. Jeśli natomiast DETACH zostanie wywołana (wywołanie powinno mieć formę: DETACH (_CL) , to Y staje się składową systemu, SL(Y) = SP, wykonanie Y zostaje zawieszone, a SP:= SS (SS =SL(X))- kontynuowane jest wykonanie składowej X.

Do przekazywania sterowania (SS i SP) między składowymi służy procedura standardowa RESUME z parametrem typu CLASS. Wywołanie RESUME(Y) w składowej X powoduje: zawieszenie wykonania składowej X (SL(X) zatrzymuje się), wznowienie wykonania składowej Y - SP:=SS:=SL(Y) - Y staje się składową aktywną.

Wywołanie DETACH w składowej jest równoważne wywołanie. RESUME(MAIN_PGM) - wznowienie składowej nadrzędnej. Wobec tego wywołanie DETACH w programie głównym SQR nie powoduje żadnej zmiany. Po zakończeniu wykonania programu obiektu samodzielnego przechodzi on w stan zakończony i wznawiana jest składowa nadrzędna - ukryta RESUME(MAIN_PGM).

Wykonanie sekwencji generującej lub wywołanie DETACH czy RESUME może nastąpić także w procedurze wywołanej ze składowej (jednak parametr aktualny DETACH musi być referencją wykonywanego programu obiektu). W takiej sytuacji SS jest ewentualnie modyfikowane po zakończeniu procedury (procedur) pośredniej (pośrednich).

Przykład 4

Należy rozwiązać następujący problem. Niech zbiór wejściowy (input) zawiera dokumenty danego formatu. Każdy dokument należy do pewnej grupy określonej numerem. Zadanie polega na przepisaniu dokumentów do zbioru wyjściowego dokonując równocześnie ich grupowania. Dla uproszczenia przyjmijmy, że liczba grup jest nie większa od 20 oraz, że jeśli liczba dokumentów w grupie przekracza 100, to można je wypisać w kilku porcjach.
Rozważmy program:

```
program GRUPDOK (input, output);
type DOKUMENT=...
var DOK: DOKUMENT;
    PB: array [1.. 20] of REF_PROCBUF;
    NRGR, ILPR, J: integer;
procedure CZYTAJDOK; ....
   {wczytuje dokument do zmiennej DOK i określa numer grupy - NRGR}
%% CLASS PROCBUF (THIS:REF_PROCBUF);
    var BUFOR: array [1..100] of DOKUMENT;
      ILDOK : integer;
    %% ATR NRPR: integer; %% END
    procedure PISZBUF;
      {wypisuje numer procesu (NRPR) oraz zawartość bufora
       - elementy tablicy BUFOR od 1 do ILDOK}
    %% BEGIN_CL
      ILDOK:=0;
      repeat
        ILDOK:=ILDOK + 1;
        BUFOR [ILDOK]:= DOK;
        if ILDOK = 100 then
        begin
            PISZBUF;
            ILDOK:=0
```

```
        end;
        DETACH (_CL)
    until eof;
    PISZBUF
  %% END-CL; {PROCBUF }
%% BEGIN_PAR
  ILPR:=0;
  repeat
    CZYTAJDOK;
    for J:= 1 to ILPR do
    if NRGR=PB [J]↑.NRPR  then
    begin
      RESUME (PB  [J] ↑._CL;

      goto 1
    end;
    ILPR:=ILPR+1;
    %% CREATE-CL      PB [ILPR];
    PB [ILPR]↑. NRPR:= NRGR;
    %% START PROCBUF (PB [ILPR]);
1: until eof;
  for J:= 1 to ILPR do
    RESUME (PB [J]↑._CL) ;
%% END_PAR.
```

Każdą grupę dokumentów przetwarza oddzielny proces buforowy - obiekt klasy PROCBUF. Składową nadrzędną SQR jest program główny GRUPDOK. Zmienna globalna ILPR oznacza bieżącą liczbę procesów buforowych, a w tablicy PB (od 1 do ILPR) zapisane są ich referencje. Program główny wczytuje cyklicznie dokumenty ze zbioru wejściowego. Sprawdza, czy istnieje proces buforowy przetwarzający dokumenty grupy, do której należy bieżący dokument. Jeśli tak, to wznawia ten proces. W przeciwnym razie tworzy nowy obiekt klasy PROCBUF dla nowej grupy dokumentów - inicjuje atrybut NRPR (numer procesu)na wartość bieżącej grupy dokumentów. Po napotkaniu końca zbioru wejściowego wznawiane są kolejno wszystkie procesy buforowe.

Obiekt klasy PROCBUF ma lokalną tablicę BUFOR, zmienną ILDOK oraz atrybut NRPR. Program obiektu cyklicznie pobiera dokumenty do bufora (jeżeli otrzyma sterowanie od składowej nadrzędnej). W wypadku zapełnienia się bufora wypisuje go. Po każdym kroku wywoływana jest procedura DETACH. Pierwsze (po utworzeniu obiektu) wywołanie powoduje "odłączenie" obiektu - sterowanie wraca do programu głównego w miejsce oznaczone etykietą "1". Kolejne wywołania powodują wznowienie programu głównego - po powrocie wykonuje się "goto 1". Po opuszczeniu pętli "repeat" wypisywana jest aktualna zawartość bufora i wykonanie programu obiektu kończy się; sterowanie wraca do programu głównego - wykonuje się kolejny krok drugiej pętli "for".

Dla porównania przedstawiam poniżej tekst analogicznego programu w Simuli:

```
begin
    ref (DOKUMENT) DOK;
    ref (PROCBUF) array PB [1:20];
    integer NRGR, ILPR, J;
    class DOKUMENT; ...
    comment struktura danych - klasa bezinstrukcyjna;
    procedure CZYTAJDOK; ...
    class PROCBUF (NRPR);
    integer NRPR;
    begin
      ref (DOKUMENT) array BUFOR [1: 100];
      integer ILDOK, J ;
```

```
      procedure PISZBUF; ...
      for J:=1 step 1 until 100 do
         BUFOR [J] :-new DOKUMENT;
      ILDOK:= 0;
REPEAT: ILDOK:= ILDOK + 1;
      .... comment w tym miejsou powinna wystąpić sekwencja instrukcji przypisana i przenosząca
           wartości atrybutów obiektu klasy DOKUMENT o referencji DOK do obiektu o referencji
           BUFOR[ILDOK]- odpowiednik pascalowej instrukcji  BUFOR [ILDOK]:= DOK;
      if ILDOK=100 then
      begin
        PISZBUF;
        ILDOK:= 0
      end;
      if ¬ ENDFILE the goto REPEAT;
      PISZBUF;
   end PROCBUF;
   DOK:-new DOKUMENT;
   ILPR:=0;
REPEAT: CZYTAJDOK;
        for J:=1 step 1 until ILPR do
        if NRGR=PB [J] . NRPR then
        begin
          resume (PB [J]) ;
          goto E1;
        end;
        ILPR:= ILPR+1;
        PB[ILPR]:- new PROCBUF (NRGR );
E1:     if ¬ ENDFILE  then goto REPEAT;
        for J:=1 step 1 until ILPR do
           resume (PB [J]);
        end
```

Ten bardzo prosty przykład nie ilustruje oczywiście wszystkich możliwości systemów quasi - równoległych. Co więcej, powyższe zadanie można łatwo zaprogramować w "czystym" Pascalu. Jednak w bardziej skomplikowanych sytuacjach technika systemów quasi - równoległych może okazać się wielce przydatna. Pozwala ona w sposób zwarty i logiczny(deklaracja klasy) opisać różnorakie procesy, których przebieg może dzielić się na etapy wykonywane na przemian z etapami innych procesów. Mechanizmy quasi - równoległości służą także do konstrukcji bardziej skomplikowanych mechanizmów symulacji quasi - równoległej.

Programowanie symulacji

Symulacja quasi - równoległa służy do modelowania procesów zachodzących w systemach rzeczywistych. Proces jest programem sekwencyjnym, a jego wykonanie dzieli się (dynamicznie) na zdarzenia. Zdarzenie to ciąg instrukcji, któremu odpowiada chwila czasowa - czas zdarzenia. Między zdarzeniami czas zmienia się skokowo. Jest to więc symulacja systemów z dyskretnym czasem.

Zdarzenia - a dokładniej ich czasy - są planowane dynamicznie i reprezentowane jako elementy listy zdarzeń SQS (sequential set) w postaci zawiadomień o zdarzeniach. Zawiadomienie jest parą (t, p), gdzie: t - czas zdarzenia, p - referencja procesu. Zawiadomienia są uporządkowane w SQS względem t w sposób niemalejący. Niech (t0, p0) oznacza pierwszy element listy SQS, wówczas t0 jest (bieżącym) czasemu, a p0 - referencją bieżącego procesu - są wykonywane jego instrukcje (bieżące zdarzenie). W danym momencie proces może mieć co najwyżej jedno zawiadomienie. Proces, który ma zawiadomienie nazywamy aktywnym, w przeciwnym wypadku - biernym. W szczególności więc proces bieżący jest również aktywny.

W Simpasie wprowadza się więc konstrukcje procesu, która jest rozszerzeniem konstrukcji klasy z tą różnicą, że:

- zamiast "CLASS" występuje "PROCESS",
- zamiast "BEGIN_CL" występuje "BEGIN_PR",
- zamiast "END_CL" występuje "END_PR"

Przykład 5

```
%% PROCESS ABC(THIS:REF_ABC);
...
%% BEGIN_PR
...
%% END_PR; {ABC}
```

Dla każdej deklaracji klasy procesów preprocesor Simpasu generuje typ referencji obiektów tej klasy procesów. Dostęp do atrybutów jest identyczny jak dla obiektów klasy. Każdy obiekt klasy procesów oprócz atrybutu _CL ma atrybut _PR standardowego (wskaźnikowego)typu PROCESS, który jest referencją procesu obiektu. Procesem obiektu jest program obiektu mający dodatkowe własności umożliwiające planowanie zdarzeń i tworzenie kolejek procesów.

Również obiekty klasy procesów powstają w wyniku wykonania sekwencji generującej, przy czym zamiast "CREATE_CL" występuje "CREATE_PR".

Przykład 6

```
%% CREATE_PR X;
%% START ABC (X),
```

gdzie X jest typu REF_ABC.

Wykonanie sekwencji generującej powoduje utworzenie nowego obiektu klasy procesów, nadania jego referencji zmiennej referencyjnej oraz natychmiastowe "odłączenie" programu obiektu - wywołanie procedury DETACH przed pierwszą instrukcją klasy procesów (sterowanie (SS i SP)wraca do następnej instrukcji po sekwencji generującej). Ponadto proces obiektu staje się bierny (nie ma zawiadomienia o zdarzeniu).

System symulacyjny (SS) jest nadsystemem SQR, a w jego skład wchodzą:

- proces główny systemu,
- procesy obiektów (aktywne lub bierne).

Programem głównym SS może być program główny w Simpasie lub procedura, gdzie zamiast zewnętrznej pary nawiasów: "begin" i "end" użyto "BEGIN_SIM" i "END_SIM". Zmienna standardowa MAIN_PROC typu PROCESS jest referencją procesu głównego.

SS powstaje w momencie rozpoczęcia wykonania procesu głównego. Sytuację tę można opisać symbolicznie w następujący sposób:

SS < SQS =<(0,MAIN_PROC)>, ts = 0, pb = MAIN_PROC >

gdzie:

ts - czas systemu,
pb - proces bieżący

Lista SQS nie jest dostępna dla użytkownika. Do planowania zdarzeń używa się specjalnych instrukcji i procedur standardowych Simpasu, odwołujących się m.in. do referencji procesów obiektów - wartości atrybutu _PR. Podczas symulacji użytkownik nie powinien używać procedur DETACH i RESUME.

Do aktywacji procesu (zmiana stanu z biernego na aktywny), czyli zaplanowania zdarzenia w procesie wykorzystuje się instrukcję "ACTIVATE". Najprostsza wersja tej instrukcji ma postać:

%% ACTIVATE<referencja procesu> %%

Jej wykonanie powoduje stworzenie zawiadomienia o zdarzeniu dla aktywowanego procesu z czasem zdarzenia równym czasowi systemu i wstawieniu go na początek listy SQS. Proces ten staje się bieżący (wznawia się wykonywanie jego instrukcji).

Wywołanie procedury PASSIVATE powoduje zawieszenie wykonywania instrukcji bieżącego procesu, usunięcie jego zawiadomienia z SQS - proces staje się bierny oraz wznowienie wykonania następnego procesu z SQS, który staje się procesem bieżącym, a czas jego zdarzenia staje się czasem systemu.

Natomiast wywołanie HOLD (t) (t typu real) powoduje zwiększenie czasu zdarzenia bieżącego procesu o max (0,t) i wstawienie go w odpowiednie miejsce listy SQS - za zdarzeniami o czasach nie przekraczających "nowego" czasu zdarzenia. Następnie wznawiany jest proces z początku SQS (zmienia się także czas systemu).

Przykład 7

Niech: X1, X2 - zmienne referencyjne obiektów klas/y/ procesów;
P1 = X1↑.PR, i=1,2 ; PG = MAIN_PROC;
(P1, PG typu PROCESS). Prześledźmy przykład przebiegu symulacji:

1) Rozpoczęcie wykonania procesu głównego;
SS=<SQS =<(0,PG)>, ts=0, pb=PG>

2) Wygenerowanie obiektów X1, X2,
SS - bez zmian

3) %% ACTIVATE P1 %% ;
SS =<SQS=<(0,P1)>, (0,PG)>, ts=0, pb = P1>

4) HOLD (3);
SS =<SQS =<(0,PG) , (3,P1)>, ts=0, pb = PG>

5) HOLD (1);
SS =<SQS =<(1,PG) , (3,P1)>, ts=1, pb = PG>

6) PASSIVATE;
SS =<SQS =<(3,P1)>, ts=3, pb=P1>

7) %% ACTIVATE P2 %%;
SS =<SQS =<(3,P2) , (3,P1)>,ts=3, pb=P2>

8) HOLD (- 1);
SS =<SQS =<(3,P1) , (3,P2)>,ts=3, pb=1>

9) Zakończenie wykonanie P1;
SS =<SQS =<(3,P2)>, ts=3, pb=P2>

10) %% ACTIVATE PG %%;
SS=< SQS =<(3,PG) , (3,P2)>, ts=3, pb=PG>

11) Zakończenie wykonania PG;
SS przestaje istnieć koniec symulacji.

Pozostałe wersje instrukcji aktywacji mają postać:

%% ACTIVATE p %% AT t %% [PRIOR] ;
%% ACTIVATE p %% DELAY t %% [PRIOR];
%% ACTIVATE p %% AFTER s %% ;
%% ACTIVATE p %% BEFORE s %%;

gdzie: p,s - referencje procesów, t-czas(wyrażenie typu real).
Instrukcja reaktywacji ma analogiczną składnię - zamiast "ACTIVATE" występuje "REACTIVATE".
Do planowania zdarzeń istnieją również procedury: CANCEL i WAIT; do badania stanu SS-funkcje: CURRENT, CURTIME (time), EVTIME, NEXTEV, IDLE, TERMINATED; do obsługi kolejek procesów - procedury i funkcji: NEWQUEUE (tworzy nową - pustą kolejkę), INTO, OUT, PRECEDE, FOLLOW, EMPTY,

CARDINAL, FIRST, LAST, PREDE (pred), SUCCE(suc) - w nawiasach podaję ewentualnie odmienne nazwy w Simuli. Dokłady opis wyżej wymienionych instrukcji, procedur (z wyjątkiem NEWQUEUE) i funkcji oraz wiele przykładów programów symulacyjnych znajdzie Czytelnik w podręczniku Simuli [2].

Podsumowanie

Simpas nie ma wszystkich udogodnień języka Simula. Nie jest możliwe tworzenie hierarchii klas przez ich prefiksowanie. Natomiast mechanizmy programowania quasi - równoległego i symulacji są bardzo podobne. Największą wadą Simpasu wydaje mi się to, że referencje: obiektu, jego programu i ewentualnie procesu są różnymi wartościami różnych typów. Jednak takie rozwiązanie spowodowane zostało specyfiką Pascala.

Implementacja

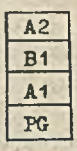
Metoda implementacji

Implementacja Simpasu polega na wykorzystaniu pewnych własności Pascala i zastosowaniu kompilatora tego języka. Programy w Simpasie są najpierw tłumaczone na Pascal przez preprocesor Simpasu który analizuje i modyfikuje fragmenty wyróżnione znacznikiem "%%". Konstrukcje niepascalowskie zostają zastąpione konstrukcjami "czystego"Pascala. Wykorzystuje się własności procedury, typów wskaźnikowych i zmiennych dynamicznych oraz instrukcji "with". Do tekstu programu wynikowego preprocesor wstawia także wywołania pewnych dodatkowych procedur należących do otoczenia programowego ("running - systemu") Simpasu. Otoczenie programowe Simpasu jest rozszerzeniem otoczenia Pascala m.in. również o procedury i funkcje standardowe Simpasu. Programy (podprogramy) otoczenia są dołączane do programu wynikowego (wyprodukowanego przez kompilator Pascala) w momencie kompletowania programu przeznaczonego do wykonania. Otoczenie programowe spełnia rolę łącznika między programem, a systemem operacyjnym komputera. Zapewnia ono wykonanie programu zgodnie z semantyką Simpasu.

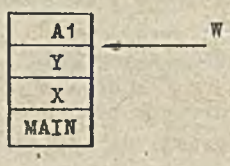
Konieczne jest więc, aby dana implementacja Pascala umożliwiała korzystanie z biblioteki procedur zewnętrznych, w tym wypadku - napisanych w Pascalu i języku maszynowym.

Dynamiczny przydział pamięci

Klasy i klasy procesów są implementowane w Simpasie jako procedury. Wykorzystuje się mechanizm dynamicznego przydziału pamięci dla procedur. Pascal dopuszcza bowiem rekursywne wywoływanie procedur. Konieczne jest więc dynamiczne przydzielanie pamięci roboczej dla każdego wywołania procedury, ponieważ jednocześnie może istnieć kilka wywołań tej samej (w sensie deklaracji) procedury (rekursja). Pamięć robocza jest przydzielana w postaci pola roboczego (work area) o określonej strukturze oraz długości - stałej dla danej procedury. Pole zawiera m.in. obszar danych (zmienne lokalne) i obszar przechowania, w którym zapamiętuje się stan danego wywołania w momencie wstrzymania jego wykonania. Jeśli bowiem w wywołaniu X procedury P wywołana zostanie procedura Q (być może P=Q), to w obszarze przechowania X zapamiętuje się adres powrotu do X, chwilowy stan rejestrów itp. Pole robocze jest tworzone w momencie wywołania procedury, a likwidowane po zakończeniu wykonania. Tworzenie pól roboczych przebiega - zgodnie z semantyką Pascala - według strategii LIFO (last in - first out). Z tego powodu są one zorganizowane logicznie w stos, który często jest także stosem fizycznym. Ilustruje to rysunek 1, gdzie: PG - pole robocze programu głównego, Ai, Bi - pola robocze wywołań procedur A i B; sytuacja odpowiada wywołaniu A w programie głównym, B w A i A w B (rekursja).



Rys.1



Rys. 2

Jednak metoda ta nie może zostać automatycznie zastosowana w quasi - równoległości i symulacji, ponieważ przydzielanie i likwidacja pól roboczych dla programów obiektów klas i procesów nie przebiega według strategii LIFO. Jeśli na przykład przyjmiemy, że A1, B1 są obiektami klas A i B, to rysunek 1 przedstawia sytuację, gdzie wygenerowano kolejno obiekty A1, B1, A2. Ale może się zdarzyć, że program obiektu A1 zakończy się przed programami obiektów B1 i A2 - pole A1 powinno ulec likwidacji. Wtedy jednak pola B1 i A2 mogą stać się niedostępne, lub ulec zniszczeniu przez wywołanie innej procedury w PG.

Wynika stąd, że dla potrzeb quasi-równoległości trzeba zmodyfikować metodę przydziału pamięci. Konieczne jest odmienne traktowanie procedur oraz klas i procesów. Można na przykład - pozostając przy stosie fizycznym - stworzyć podstos pól roboczych programów obiektów, a procedurom przydzielać pola dopiero powyżej tego podstosu - rysunek 2: MAIN - pole programu głównego SQR, X,Y - pola programów obiektów, W - wierzchołek podstosu pól programów obiektów, A1 - pole procedury A.

Po przydzieleniu pola programu obiektu wierzchołek podstosu jest przesuwany na koniec tego pola - podstos rośnie. Wywołanie procedury nie powoduje przesunięcia W. W tym miejscu celowe wydaje się przyjęcie ograniczenia, że sekwencja generująca może być wykonana tylko w programie składowej SQR. Zapobiegnie to powstawaniu w podstosie niepotrzebnych "dziur" po polach procedur pośrednich (między programem składowej, a programem tworzonego obiektu).

Po zakończeniu programu obiektu jego pole może być zwolnione. Nie wiąże się to jednak z modyfikacją W, chyba że zwalniane pole znajduje się na szczycie podstosu - oszczędność pamięci. Gospodarkę pamięcią można jeszcze udoskonalić wykorzystując na pola programów nowo tworzonych obiektów również "dziury" w podstosie po programach zakończonych. Jest to możliwe i wskazane ponieważ wszystkie pola programów obiektów tej samej klasy są równej długości (na ogół tworzy się wiele obiektów tej samej klasy).

Reasumując: łatwość implementacji Simpasu przy danej implementacji Pascala zależy od możliwości zmodyfikowania metody przydziału pamięci dla procedur.

W Pascalu IBM pole robocze procedury jest tworzone i likwidowane odpowiednio przez inicjator i terminator procedury należące do otoczenia programowego Pascala. Kod produkowany przez kompilator jest tej postaci, że wywołanie procedury (skok ze śladem do początku kodu procedury) dzieli się na:

● wywołanie inicjatora, ● wykonanie instrukcji procedury, ● wywołanie terminatora.

Początek pola roboczego stanowi obszar przechowania zawartości rejestrów. Zgodnie z konwencją IBM obszary przechowania są powiązane w listę dwukierunkową - czyli pola robocze także. Jest to więc szczególny rodzaj stosu logicznego. Pamięć jest pobierana od systemu operacyjnego większymi (niż jedno pole) porcjami za pomocą makroinstrukcji programu nadzorczego (supervisor) GETMAIN (zwalniana przez FREEMAIN). Wobec tego stos pól procedur może się składać z kilku kawałków fizycznych.

W Simpasie metoda ta została zmodyfikowana przez wymianę inicjatora i terminatora procedury. Przy wywołaniu procedury lub tworzeniu obiektu klasy pobiera się porcję pamięci równą polu procedury lub programu obiektu. Sposób ten jest bardziej czasochłonny ale zapewnia lepszą gospodarkę pamięcią i pozwala na wygodne manipulowanie stosem pól roboczych przez zastosowanie klasycznych operacji na listach.

Klasy

Preprocesor Simpasu usuwa z treści klasy deklaracje atrybutów i wykorzystuje je do wygenerowania definicji typu obiektów danej klasy:

```
TYPE _<nk>= record
                _CL : CLASS;
              < deklaracje atrybutów >
```

```
                end;
```

gdzie <nk>- nazwa klasy. Generowana jest również definicja:

```
REF_<nk> =↑TYPE_<nk>;
```

Przypomnę, że źródłowa postać sekwencji generującej obiekt klasy jest następująca:

```
%% CREATE_CL<ref>;
[<inicjacja atrybutów>]
%% START <nk>(<ref> ....);
```

gdzie <ref> - zmienna typu REF_<nk> - przyjmuje wartość referencji tworzonego obiektu.

Preprocesor przekształca ją na postać wynikową:

```
{CREATE_CL} new (<ref>);
[<inicjacja atrybutów>]
{START} <nk>(<ref> ....);
```

Jak widać wykonanie instrukcji "CREARR_CL" sprowadza się do wywołania procedury "new". W efekcie zostaje utworzona zmienna dynamiczna typu TYPE_<nk> - obiekt klasy <nk>, a zmienna <ref> staje się jego referencją (wskaźnikiem). W tym momencie staje się jasne dlaczego zewnętrzny dostęp do atrybutów obiektu ma postać <ref>↑.<atrybut>.

W tym miejscu trzeba zanalizować wynikową postać treści klasy, która jest następująca (zob. treść źródłowa):

```
{CLASS} procedure <nk>(THIS : REF_<nk>....);
....
  {BEGIN_CL}
  begin with THIS↑ do begin
    XXXINITCL(_CL, MAIN_PGM);
    ....
    XXXTERMCL (_CL);
  end end;
  {END_CL}
```

gdzie: XXXINITCL - inicjator klasy, XXXTERMCL - terminator (zob. Mechanizmy quasi-równoległości). Ujęcie instrukcji klasy jako wnętrza instrukcji "with" realizuje wewnętrzny dostęp do atrybutów obiektu - za pośrednictwem identyfikatora. Istotne jest to, że parametr wskazujący obiektu (THIS) jest wołany przez wartość. Instrukcja "with" jest stosowana w identyczny sposób również w procedurach lokalnych klasy - stąd konieczność oznaczenia zewnętrznych nawiasów "begin" i "end" takiej procedury przez "%%".

Mechanizmy quasi - równoległości

Jak już wiemy, każdy obiekt dowolnej klasy ma standardowy atrybut _CL typu CLASS będący referencją programu tego obiektu, a dokładniej - referencją opisu programu. Z drugiej strony: Każdy program obiektu ma swoje pole robocze. W momencie zawieszenia jego wykonania przez wywołanie procedury - w szczególności DETACH lub RESUME - w polu roboczym zapamiętywany jest stan programu w chwili zawieszenia oraz adres powrotu do niego, który może być traktowany jako jego sterowanie lokalne - SL. Wynika stąd, że dla wznowienia zawieszonego programu obiektu wystarczy znać adres jego pola roboczego (work area address) - WAA. Konieczna jest również jakaś reprezentacja stanu SQR - określenie składowej nadrzędnej, bieżącej, sterowania systemu (SS). W tym celu wprowadza się definicje typów:

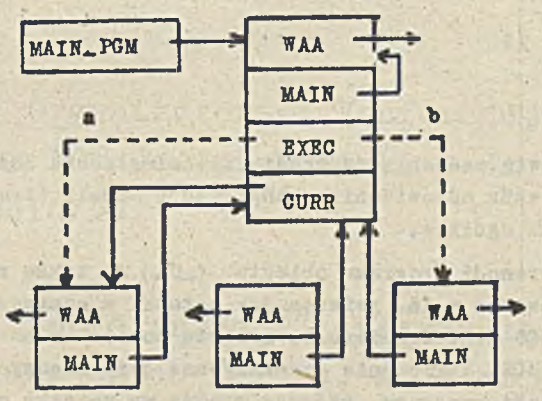
```
CLASSS = ↑XXXCLASS;
XXXCLASS = record
             WAA: integer;
             MAIN: CLASS;
             case boolean of
```

```
            true: (EXEC, CURR: CLASS);
            false:( )
        end;
```

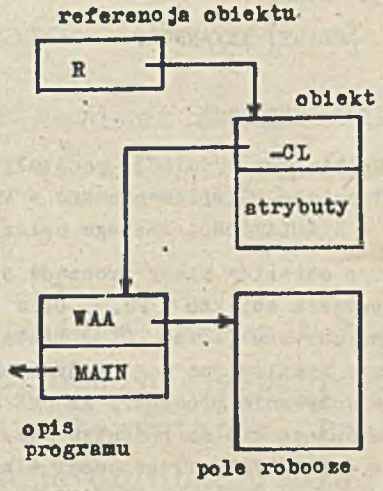
gdzie:

WAA - adres pola roboczego programu (obiektu lub składowej nadrzędnej),
MAIN - referencja programu głównego SQR,
EXEC - referencja wykonywanego programu(obiektu) - posiadającego sterowanie procesora (SP),
CURR - referencja bieżącej składowej SQR - programu posiadającego sterowanie systemu (SS).

Zmienna typu XXXCLASS stanowi opis programu w SQR. Wariant "true" przeznaczony jest dla składowej nadrzędnej, a "false" - dla programu obiektu. Opis programu obiektu zawiera więc jedynie pola WAA i MAIN. Stan SQR zapisany jest w opisie jego programu głównego. Dla dowolnego obiektu o referencji R zachodzi równość R↑._CL↑. MAIN = MAIN_PGM. Z powyższych rozważań wynika, że EXEC=CURR wtedy i tylko wtedy, gdy SP=SS. Jeśli więc EXEC<>CURR, to SP przebywa w programie obiektu przyłączonego - rys. 3. Zachodzi jednoznaczna zależność między obiektem, opisem jego programu i polem roboczym - R, R↑._CL, R↑._CL↑. WAA - rys. 4.



Rys. 3
a) SP przebywa w składowej,
b) SP przebywa w programie obiektu przyłączonego



Rys. 4

Mechanizmy quasi - równoległości działają na dwóch poziomach:

1) poziom Pascala - operacje na zmiennych MAIN, EXEC i CURR,
2) poziom języka maszynowego - operacje na stosie (listach itp.) pól roboczych.

Operacje poziomu 2) są uruchamiane z poziomu 1) przez wartość WAA. Nie będę ich szczegółowo opisywał, ponieważ są zależne od implementacji maszynowej.

SQR powstaje przez utworzenie zmiennej dynamicznej o referencji MAIN_PGM - wariant "true", zainicjowaniu MAIN, EXEC i CURR na wartość MAIN_PGM i "wydobyciu" z poziomu 2) wartości WAA programu głównego SQR.

Inicjator klasy (XXXINITCL) tworzy opis programu obiektu (wariant "false"), jego referencję podstawia na MAIN_PGM↑.EXEC i inicjuje WAA opisu programu.

Wywołanie DETACH (_CL) w programie obiektu przyłączonego (pierwsze po wygenerowaniu) polega na wznowieniu wykonania bieżącej składowej:

1) _CL↑.MAIN↑.EXEC := _CL↑.MAIN↑.CURR
2) sterowanie (SP) przekazuje się do programu o adresie pola roboczego _CL↑. MAIN↑.CURR↑. WAA.

Natomiast wywołanie RESUME (R↑._CL) polega na zmianie bieżącej składowej SQR:

1) _CL.MAIN↑.CURR := R↑._CL;
   _CL.MAIN↑.EXEC := R↑._CL;
2) sterowanie (SP)przekazuje się do programu o adresie pola roboczego R↑._CL↑.WAA.

Z kolei XXXTERMCL(_CL)powoduje:
1) wywołanie RESUME (_CL↑.MAIN);
2) likwidację pola roboczego o adresie _CL↑.WAA.

Działanie inicjatora i terminatora klasy związane jest z metodą przydziału pamięci dla klas i procedur. Ponieważ wywołanie procedury i rozpoczęcie wykonania programu obiektu (instrukcja "START") są składniowo identyczne, to może się okazać, że konieczne jest wcześniejsze przekazanie informacji odróżniającej do otoczenia programowego. Służy do tego procedura zapowiadająca rozpoczęcie wykonania programu obiektu XXXANNCL, której wywołanie generowane jest w instrukcji "START":

{START} XXXANNCL; <nk>(<ref> ....);

Mechanizmy symulacji

Mechanizmy symulacji powstały na podstawie mechanizmów quasi- równoległości. Zostały one całkowicie zaimplementowane w Pascalu na wzór odpowiednich konstrukcji Simuli (treść klas: SIMSET i SIMULATION). Dlatego opiszę je bardzo ogólnie.

Typ obiektów klasy procesów oprócz referencji programu obiektu (_CL) ma także referencję opisu procesu obiektu _PR. Opis procesu zawiera m.in. referencję: notatki w zdarzeniu i programu obiektu (równą _CL). Lista zdarzeń SQS jest dwukierunkową listą notatek o zdarzeniach. Procedury symulacyjne mogą modyfikować listę _SQS. Sterowanie przekazywane jest między procesami przez wywoływanie procedury RESUME od referencji programu, którego proces ma notatkę o zdarzeniu znajdującą się na początku SQS. Inicjator procesu wywołuje ponadto procedurę DETACH, a terminator - PASSIVATE. Preprocesor tłumaczy instrukcje aktywacji i reaktywacji na wywołanie wieloparametrowej procedury ACTIVATE.

Literatura

[1] Iglewski M., Madej J., Matwin St.: Pascal. Język wzorcowy. Pascal 360. WNT: Warszawa 1984.

[2] Oktaba H., Ratajczak W.: Simula 67. WNT: Warszawa 1980

[3] Pascal OS/JS - opis języka. Zakład Doświadczalny Instytutu Maszyn Matematycznych: Warszawa 1977

[4] Pascal OS/JS - przewodnik programisty. ZD IMM: Warszawa 1977

[5] Szewczuk A. "Quasi-równoległość i symulacja w Pascalu - język Simpas i jego implementacja w systemie OS/JS" (maszynopis). Warszawa 1984.

sprawozdania

IV Krajowa Konferencja Naukowo-Techniczna

ZASTOSOWANIE KOMPUTERÓW W PRZEMYŚLE

Szczecin, 11-13 września 1985 r.

Organizatorami Konferencji byli: Komitet Naukowo-Techniczny NOT ds Informatyki w Szczecinie oraz Politechnika Szczecińska.

Konferencje "Zastosowanie komputerów w przemyśle" organizowane są co 2 lata, a celem ich w założeniu jest przedstawienie różnorodnych zagadnień stosowania komputerów. Praktycznie jest to forum umożliwiające prezencję dorobku raczej placówek naukowych i przeważnie jest to raczej oferta niż relacja z wdrożeń. Niemniej oferta ciekawa, dowodząca, że istnieją w kraju liczne ośrodki i zespoły posiadające chęć, zdolności i umiejętność konstruowania zarówno sprzętu, jak i oprogramowania przeznaczonego dla różnorodnych, bardzo konkretnych zastosowań. Prezentowane rozwiązania odznaczały się w większości praktycznością, realnością. mało było przeważających dawniej wystąpień czysto teoretycznych, akademickich. Dominował konkret. Pomysły były poparte konstrukcjami wykonanymi, niestety, na ogół jedynie na skalę laboratoryjną, eksperymentalną.

Konferencji towarzyszyła wystawa sprzętu komputerowego, na której prezentowano mikrokomputerowe produkty firm polonijnych i zakładów doświadczalnych. Brakowało poważnych firm przemysłu komputerowego.

Ponadto zorganizowano giełdę oprogramowania, której celem była wymiana lub sprzedaż kompletnych produktów programowych.

Zgłoszone na konferencję referaty rozdzielono w dwojaki sposób. Po pierwsze wydzielono część referatów, które zaprezentowano na sesjach plenarnych. Na sesjach tych dodatkowo wystąpili przedstawiciele ICL i Hewlett-Packard, omawiając nowości swoich firm. Pozostałych 28 referatów było wygłoszonych na ośmiu sesjach sekcyjnych.

Drugi sposób podziału zgłoszonych referatów to: referaty wygłoszone na sesjach plenarnych lub sekcyjnych i opublikowane w całości w materiałach konferencji oraz materiały zakwalifikowane do prezentacji na sesjach plakatowych - 85 wystąpień, lub na giełdzie oprogramowania - 35 wystąpień. W tym wypadku w materiałach konferencji opublikowano jedynie streszczenia referatów, natomiast całe teksty autorzy rozwieszali na przydzielonych im "ścianach", mając możność prezentować swój materiał zainteresowanym uczestnikom konferencji. Materiały przeznaczone na sesje plakatowe zestawiono w grupy o tej samej nazwie co referyty sekcyjne.

I. Grafika komputerowa - 6 referatów oraz 9 wystąpień plakatowych

II. Systemy komputerowego wspomagania projektowania i produkcji - 5 referatów i 20 wystąpień plakatowych

III. Układy mikroprocesorowe, mikrokomputery, minikomputery w systemach pomiarów sterowania i automatyki: 3 referaty, 32 wystąpienia plakatowe

IV. Sieci komputerowe i ich oprogramowanie - 3 referaty i 5 wystąpień plakatowych

V. Standardowe systemy komputerowe i ich oprogramowanie: 3 referaty i 2 wystąpienia plakatowe

VII. Środowisko oprogramowania - 4 referaty i 9 wystąpień plakatowych

VIII. Szkolenie w informatyce - tylko 2 referaty.

Materiały z giełdy oprogramowania prezentowane były w trzech grupach:

- organizacja i sterowanie produkcją - obejmujące 13 ofert produktów programowych
- oprogramowanie użytkowe - 80 ofert
- systemy operacyjne, generatory, testy, projektowanie, programowanie - 14 ofert.

Z tej ogromnej ilości materiałów i informacji na wyróżnienie zasługują szczególnie plenarne referaty: doc. Jerzego Sołdka z Instytutu Okrętowego Politechniki Szczecińskiej.: "Komputerowe wspomaganie twórczej działalności w technice" oraz dr Tomasza Pawlaka (Sekretariat Komitetu Informatyki): "Zagadnienia rozwoju zastosowań komputerowych systemów techniczno-organizacyjnego przygotowania produkcji (topp)w przemyśle krajowym".

Doc. J. Sołdek przedstawił bogato ilustrowaną przezroczami, a nawet komputerowym filmem, wizję przyszłych zastosowań komputerów w pracy inżyniera-projektanta, w pracy naukowca, w skomputeryzowanym biurze, we wspomaganiu pracy menagera, plastyka, urbanisty, architekta, itp.

Dr T. Pawlak przedstawił aktualną i przewidywaną w najbliższym pięcioleciu sytuację w zakresie zastosowań komputerów w topp w przemyśle na tle możliwości dostaw sprzętu komputerowego.

Z referatów wygłoszonych na sekcjach na uwagę zasługują: dr inż. Marii Chałon z ITC Politechniki Wrocławskiej "Graficznie zorientowane struktury danych", mgr inż. Haliny Piotrowskiej-Gocławskiej z Instytutu Elektroniki Politechniki Łódzkiej "Metoda analizy statystycznej obrazów czarno-białych z preparatów mikrobiologicznych", doc. J. Sołdka i in. "Zintegrowany informatycznie system projektowo-produkcyjny, na przykładzie zakładu przemysłu okrętowego", dr A. Małachowskie-

go z Instytutu Informatyki Akademii Ekonomicznej we Wrocławiu "Wspomagane komputerem zarządzanie przedsiębiorstwem przemysłowym", grupa referatów na temat sieci komputerowych, z których wynika, że prowadzone są liczne prace nad sieciami SM-owskimi. Są to referaty: doc. dr hab. A. Gościńskiego i innych z Instytutu Informatyki AGH: "Wykorzystanie sieci komputerowej do łączenia środków komputerowych", mgr inż. Bieleninika i in. z Wrocławia pt.:"Minikomputer SM-4 jako mini-hos w sieci komputerowej MSK" oraz dr inż. A.Bending-Wielowiejskiego i mgr inż. J. Sarada z Instytutu Okrętowego Politechniki Szczecińskiej "Lokalna sieć komputerowa na minikomputerach serii SM".

Z referatów zgrupowanych w pozostałych sekcjach trzeba wspomnieć o "Zastosowaniu systemu ELWRO-80 w sterowaniu i automatyce" zaprezentowanym przez mgr inż. K. Frączaka z Zakładów Projektowania Systemów ELWRO; "Oprogramowanie modelu użytkowego elektronicznej centrali telegraficznej - doświadczenia konstrukcyjne i technologiczne" przedstawiony przez dr inż. J. Bońskiego z Instytutu Informatyki Politechniki Gdańskiej oraz oba referaty sekcji "Szkolenie w informatyce": dr J. Mikiewicza i in. z OBP Politechniki Wrocławskiej "Komputeryzacja krajowego szkolnictwa wyższego, prognoza na lata 1986, 1990, 1995" i doc. dr hab. R. Swiniarskiego z Instytutu Sterowania i Elektroniki Przemysłowej PW-"Nauczanie systemów operacyjnych CP/M i RSX11", które to referaty są świadectwem zaawansowania problematyki komputerowo wspomaganej dydaktyki w krajowym szkolnictwie wyższym.

Podsumowując trzeba stwierdzić, że konferencja przyniosła bardzo duży materiał i stała się autentycznym forum wymiany doświadczeń w zakresie zastosowań komputerów.

dr inż. Stanisława Bonkowicz-Sittauer

nowości techniczne

Nowe superkomputery Fujitsu

Fujitsu Ltd. wprowadza do sprzedaży model Facom VP-50, który może być stosowany jako niezależy superkomputer lub komputer ogólnego użytku o bardzo dużych możliwościach. Szybkość procesora dla obliczeń wektorowych wynosi 140 mln operacji zmiennoprzecinkowych na sekundę, kilka razy więcej niż najszybszy jednoprocesorowy system tej firmy M-380. Jako maszyna ogólnego użytku Facom VP-50 ma prędkość porównywalną z M-380. Opłata miesięczna za dzierżawę wynosi 185 tys. dol. Ekonomicznie jest wykorzystywać Facom VP-50 jako system ogólnego użytku w dzień, a jako superkomputer nocą.

W grudniu 1985 r. rozpocznie się dostawa superkomputera VP-400 o szybkości 1,14 mld. operacji zmiennoprzecinkowych na sekundę (obecnie stosowane superkomputery VP-200 i VP-100 mają te szybkości odpowiednio 500 i 2500 mln/s). Opłata miesięczna za dzierżawę wynosi 318 tys.dol.

Electronics Week nr 16/85

Dynamiczne kostki pamięci 256 kbajtów

Wciąż spadające ceny kostek dynamicznej pamięci operacyjnej (DRAM) o pojemności 256 k osiągnęły już granicę 4 dolarów przy sprzedaży hurtowej na rynku japońskim. Ceny ko-

stek o pojemności 64k osiągnęły 1,2 dolara. Jednocześnie oczekuje się, że Nippon Electric Company, Hitachi i Fujitsu pójdą w ślady firmy Toshiba i będą oferować próbki kostek o pojemności 1 Mbitów do końca tego roku. Masowa produkcja tych kostek oczekiwana jest w 1987 roku, co wpłynie niewątpliwie na wielkość produkcji kostek 256 k.

Electronics Week nr 16/85

Nowe kostki 80286

Firma Adwanced Micro Devices Inc. w Austin rozpoczęła masową dostawę mikroprocesorów 80286 o częstotliwości zegara 10 MHz - prawie dwukrotnie szybszych od stosowanych obecnie w komputerze osobistym AT IBM. Są one wytwarzane w ceramicznych obudowach i partiach po 100 szt., sprzedawane są po 160 dol.

Również twórca tej kostki - firma INTEL Corp. wytwarza, zarówno kostki o częstotliwości zegara 10MHz, jak i 12,5 MHz. Osiąga to przy zastosowaniu najnowszej technologii nazwanej HMOSIII. Są one o 30% mniejsze niż dotychczas wytwarzane kostki o częstotliwości zegara 6 i 8 MHz. Sprzedaż ich rozpocznie się w czwartym kwartale 1985 r., przy czym cena kostki 10 MHz (w partiach po 100 szt.) wyniesie 176 dol., a kostki 12,5 MHz - 298 dol.

Electronics Week nr 17/85

Komputeryzacja mieszkań

Współczesne mieszkania pełne są urządzeń ułatwiających życie, lecz sterowanie tymi wszystkimi urządzeniami nie jest proste i dopiero wprowadzenie centralnych systemów sterowa-

nych komputerowo rzeczywiście ułatwiło i zabezpieczyło życie domowe.

Firma General Electric Co. zaprojektowała domowy system monitorujący o nazwie Homeminder. Jest to mikrokomputer małych rozmiarów, który dołącza się do telewizora. Można wówczas (zdalnie sterując) wywołać na ekran telewizora obraz wszystkich urządzeń, jakie mogą być programowane, wybrać te, których parametry chcemy ustalić czy zmienić. Inny obraz pokaże nam wykaz wszystkich pomieszczeń, w których te urządzenia się znajdują i znów wybieramy żądane pomieszczenie. Teraz w prosty sposób za pomocą kodu numerycznego ustalamy żądane parametry. Sygnały sterujące przesyłane są za pomocą standardowych przewodów elektrycznych do małych modułów dołączonych do poszczególnych urządzeń. System może włączać i wyłączać światło, klimatyzację, ogrzewanie i inne urządzenia wg ustalonego harmonogramu lub podanego czasu. Wyjeżdżając z domu można sterować systemem telefonicznie. Istnieje możliwość dołączenia urządzeń niestandardowych, jak np. zasilane z baterii urządzenie alarmowe reagujące na dym. Posługujemy się wówczas specjalnym językiem sygnalizującym - Homenet. W ten sposób można np. wyłączyć grzałkę pralki w wypadku uszkodzenia bębna itp.

Inne rozwiązanie zaproponowała japońska firma Mitsubishi, która wprowadziła urządzenie ISR (Invisible Silent Robot - niewidzialny milczący robot), które jest bardziej złożone od Homemindera. Znajduje to odbicie także w cenach, gdyż ISR kosztuje około 2000 dol. a Homeminder 450 dol. Posługując się ISR można przy wyłączaniu budzika rano powodować pobudzenie łańcucha odpowiednich czynności: zagrzanie wody w maszynce do kawy i na prysznic, włączenie telewizora na wiadomości poranne itp. Jest to jednocześnie system alarmowy. Czujniki przymocowane do ścian lub sufitów wykrywają ulatniający się gaz lub sygnalizują pożar przy podwyższeniu temperatury. Wykrycie takiej sytuacji w pustym domu powoduje zawiadomienie policji lub straży ogniowej przez uprzednio nagrane komunikaty. W podobny sposób czujniki ruchu mogą

spowodować zawiadomienie o włamaniu do zamkniętego pomieszczenia lub braku objawów działalności starszej osoby pozostawionej samej w domu.

Przypuszcza się, że w przyszłości systemy automatyzacji domu będą jeszcze bardziej inteligentne. NAHB (National Association of Home Builders - Krajowe Stowarzyszenie Budowniczych Domów) projektuje "sprytny" dom o znacznie obniżonym prawdopodobieństwie zwarcia elektrycznego czy pożaru. Wszystkie połączenia dokonywane są za pomocą jednolitej sieci kabli, dzięki czemu unika się szkodliwych oddziaływań wzajemnych. Centralny procesor doprowadza odpowiedni typ zasilania do każdego urządzenia, przy czym energia nie będzie dostarczona przed otrzymaniem odpowiedniego sygnału elektronicznego z urządzenia. Włączenie żelazka do prasowania np. mówi procesorowi, jaką moc ma posłać do urządzenia. System taki jest oszczędny - duże urządzenia, jak pralki i maszyny do zmywania naczyń korzystają z zasilania wysokonapięciowego, a odbiorniki radiowe i magnetofony ze źródeł niskiego napięcia bez użycia konwerterów. Jest to również bezpieczniejsze, gdyż przy zwarciu większa energia nie popłynie. Prototypowe domy tego typu mają być wybudowane w 1986 roku i do tego czasu będą przyjęte nowe standardy kablowania, a urządzenia wyposażone w dodatkowe kostki komputerow. W przyszłości można wyobrazić sobie, że światło wyłącza się samo po naszym wyjściu z pokoju, urządzenia włączają się na ustne rozkazy, a komputer domowy pisze na ekranie: "Zauważyłem, że wyjeżdżasz na weekendy, czy mam przyjąć to jako stałą pozycję w rozkładzie tygodnia?".

Newsweek nr 28/85

Nowe funkcje zegarków elektronicznych

Wraz z rozwojem techniki mikroprocesorowej zegarki ręczne wykonują coraz to nowe czynności. Firma Biotechnology Inc. z Miami opracowała urządzenie o nazwie Wrist Coach, które jest monitorem pulsu właściciela. Uderzenia serca przenoszone są cieniutkim przewodem z delikatnych elektrod na piersiach do małego mikroprocesora w zegarku, który przetwarza informację i wyświetla ją na monitorze ciekłokrystalicznym. Odpowiednim przyciskiem można uzyskać odczyt liczby uderzeń na minutę. Można też zaprogramować ostrzegawczy poziom prędkości uderzeń. Jeśli zostanie on przekroczony włącza się sygnał dźwiękowy. Urządzenie sprzedawane jest w cenie 99 dol.

Zegary mówiące, które pojawiły się ponad dwoma laty, osiągalne są obecnie w postaci zegarków ręcznych. Japońska firma Satoki Co Ltd. z Tokio oferuje Talking Space Watch, który mając postać standardowego zegarka cyfrowego zawiera kostki dźwiękowe i miniaturowy głośniczek podający czas zbliżonym do damskiego syntetyzowanym głosem. Zegarek może być zaprogramowany tak,aby ogłaszał czas automatycznie co pół godziny lub działał jako budzik, ustawiany co 5 minut. W czasie budzenia urządzenie podaje godzinę, po czym następuje 30 sekundowy sygnał alarmowy. Jeśli użytkownik przyciśnie klawisz "drzemka" będzie obudzony za pięć minut z podaniem czasu i komentarzem "Proszę pospiesz się". Zegarek mówi wyłącznie po angielsku i jest sprzedawany w USA przez firmę Hammacher Schlemmer po około 60 dolarów.

Kalkukatory połączone z zegarkami na rękę spełniają już z powodzeniem wiele funkcji, lecz kłopot sprawiają przyciski wielkości główki od szpilki, którymi trudno manipulować. Aby to ułatwić firma Casio wprowadziła dwa modele zegarków wykorzystujące czułe na dotknięcie kryształy zamiast przycisków. Model AT-552 ma analogowy wyświetlacz na 3/4 powierzchni zegarka. U góry jest wyświetlacz cyfrowy. Przyciśnięcie

przełącznika zmienia reżim pracy zegarka na kalkulator i cyfry wskazywane przez palce poprzez czuły na ciepło kryształ będą ukazywać się na wyświetlaczu wraz z wynikiem. Model TC-50 jest w pełni cyfrowy i klawiatura czuła na światło pojawia się na tarczy zegarka w reżimie kalkulatora. Cyfrowy odczyt wyświetlany jest na górnej części tarczy zegarka. Chromowa wersja AT-552 kosztuje 80 dolarów, a złocona 90, natomiast TC/50 - 50 dol.

Dwa lata temu firma Seiko pierwsza wprowadziła możliwość dokonywania zapisków w zegarku ręcznym ("notatnik"). Obecnie opracowane urządzenie o nazwie Datagraph RC-4000 umożliwia korzystanie z pojemności komputera domowego przy użyciu specjalnie zaprojektowanego oprogramowania. Zegarek może być dołączony bezpośrednio kablem lub przez modem przy długich odległościach, z niemal dowolnym typem komputera osobistego. Informacja może być przenoszona z pamięci komputera do zegarka. Z zegarka można również wprowadzić informację do pamięci. Układ może przechowywać 12 zbiorów takich, jak rozkład lotów lub spotkań, które mogą być wprowadzane lub wyświetlane w 2 wierszach po 12 znaków. Koszt RC-4000 wynosi około 200 dol. łącznie z oprogramowaniem i kablem.

Wreszcie dla tych, którzy wolą określać czas za pomocą położenia słońca i księżyca, firma Citigen oferuje Moon Phase Watch, który posiada na tarczy dwa krążki usytuowane powyżej i poniżej ramienia mocującego. Górny krążek przedstawia srebrny księżyc, który krąży zgodnie z kierunkiem obrotu wskazówek zegara od prawej do lewej (ze wschodu na zachód) z cyklem 24 godziny 51 minut. Obraca się on również z cyklem 30-dniowym, co powoduje zmianę jego faz.

Dolny krążek przedstawia małe złote słońce, które obraca się o 360° w ciągu 24 godzin. Te skomplikowane ruchy sterowane są z wbudowanej wielofunkcyjnej kostki. Zegarek jest pomocny dla nawigatorów i rybaków przy śledzeniu pływów. Dostarcza on ponadto informacji astrologicznych o perspektywach na dany dzień dotyczących zdrowia, bogactwa i przygód miłosnych.

Zamiast więc patrzeć na rozkład dnia właściciel może dowiedzieć się czy dzień przyniesie mu sławę i fortunę. Koszt tylko 70 dol.

Newsweek nr 27/85

Wśród firm komputerowych

Firma Mentor Graphics odgrywa czołową rolę w dziedzinie projektowania wspomaganego komputerowo (ok. 35% rynku, dochody 88 mld.dol.) Pod koniec 1985 roku oferować ona będzie stanowisko do projektowania pakietów (board station), które pozwoli skrócić cykl projektowania do jednego tygodnia. Cena systemu o średniej konfiguracji 79 tys.dolarów.

*

W roku obliczeniowym kończącym się 31 marca 1985 r. firma Dataproducts wykazała 471,8 mln.dol. wpływów (wzrost o 18% w stosunku do roku ubiegłego) i 27,6 mln. dol. zysku (wzrost o 6%). W ostatnim kwartale zanotowano pewien spadek zamówień.

*

W ciągu trzeciego, kolejnego roku firma Act Holdings, jedna z czołowych firm mikroinformatycznych w Wielkiej Brytanii, która stworzyła rodzinę maszyn Apricot, podwoiła swój zysk. Za okres roku kończącego się 31 marca 1985 dochody firmy przekroczyła 92 mln. funtów, a zysk przed zapłaceniem podatków wyniósł 10 mln. funtów.

*

Francuska firma usług i inżynierii informatycznej Dataid zaoferowała na giełdzie paryskiej swe akcje po 200 franków, których łączna wartość stanowi około 10% kapitału firmy. W roku 1984 wpływy firmy wyniosły 140 mln. franków, a czysty dochód 4,4 mln. franków.

*

W pierwszym półroczu 1985 r. dochody firmy Apple wzrosły o 45% w stosunku do analogicznego okresu w roku ubiegłym osiągając 1,1 mld dolarów. Czysty zysk wykazał wzrost o 9% i wyniósł 56,1 mld. dol.

*

Firma Appollo Computer specjalizująca się w 32-bitowych stanowiskach projektowych i naukowych podała, że w pierwszym kwartale 1985 r. jej dochody wyniosły 82 mln.dolarów, co oznacza wzrost o 124% w stosunku do analogicznego okresu ubiegłego roku. Zyski osiągnęły 9 mln. dol. wzrastając o 117%.

*

Wg przewidywań firmy ITT Corporation zyski w drugim kwartale 1985 r. powinny wzrosnąć o ponad 10% w stosunku do tego okresu w roku ubiegłym. Przewidywania te oparte są na wynikach wstępnych.

Micro Systemes
nr 9/85

Mikroprocesory 32-bitowe

Niewątpliwie systemy 32-bitowe będą rozwijać się i w przyszłości stanowić będą znaczną część rynku komputerowego. Ocenia się, że w drugiej połowie lat dziewięćdziesiątych zastosowane będą w samych tylko jednostkach centralnych mikroprocesory 32-bitowe za kilka miliardów dolarów, a licząc urządzenia peryferyjne, oprogramowanie i systemy wspomagania - rynek ten wynosić będzie dziesiątki miliardów.

Początkowo największe zastosowania będą miały miejsce w automatyzacji biur i stanowisk pracy oraz sterowania numerycznego, gdzie obecnie używane są złożone układy 16-bitowe. Drugi rodzaj zastosowań dla mikroprocesorów 32-bitowych obejmuje systemy ekspertowe, roboty, grafikę komputerową, przetwarzanie transakcji i sygnałów oraz rozpoznawanie mowy. Tutaj obecnie wykorzystywane są duże maszyny i superminikomputery. Ostatnim wreszcie i prawdopodobnie najszerszym zastosowaniem tych mikroprocesorów będzie rozległy rynek komputerów domowych i osobistych, które, aczkolwiek bardzo złożone, powinny być łatwe w obsłudze.

Jak dotychczas dużo mówi się o mikroprocesorach 32-bitowych, ale z opracowanych kilkudziesięciu tych układów tylko trzy są osiągalne na rynku. Wiele firm opracowało takie mikroprocesory do swych własnych systemów i wytwarza je tylko wewnętrznego użytku, lecz są one bardzo ostrożne z udostępnianiem ich do ogólnego użytku.

Firma Dataquest z San Jose przeprowadziła badania rynku i opracowała prognozy dla tych mikroprocesorów. Obecnie (1985 r.) zastosowania w automatyzacji biur i komputerach osobistych stanowią około 1/3 rynku, następnie idą zastosowania związane ze wspomaganym komputerowo projektowaniem i zarządzaniem (1/4 rynku) oraz automatyzacją wytwarzania i robotami (1/5); reszta to zastosowania telekomunikacyjne, wojskowe i inne. Natomiast w roku 1990 automatyzacja biur i komputery osobiste stanowić będą aż 84% wszystkich zastosowań mikroprocesorów 32-bitowych, wspomagane projektowanie tylko 8%, a automatyka produkcji i roboty zaledwie 4%. Przy czym obecnie rynek ten jest w ogóle nieznaczny. Pierwsi dostawcy oferują układy o częstotliwości zegara w zakresie 6 ÷ 10 MHz. Jest to obecnie za mało. Potencjalni użytkownicy chcieliby budować jednostki centralne o częstotliwości zegara 12 ÷ 16 MHz, a docelowo 25 MHz. Uzyskanie jednak tego nie jest łatwe. Dlatego początkowe przewidywania są obecnie oceniane jako zbyt optymistyczne. Mówiło się o sprzedaży 100-150 tys. mikroprocesorów 32-bitowych w 1985 roku. W rzeczywistości liczba ta będzie 10-15% niższa. Jeszcze trudniejsza do oceny jest wartość tych mikroprocesorów, gdyż większość cen jest sztucznie zawyżona. Zakładając średnią cenę 250-500 dolarów uzyskuje się sumę 20-50 milionów dol. co w znacznej mierze pokryje nakłady na badania i rozwój w tej dziedzinie wg Mel Thomsena, eksperta wspomnianej firmy. Ocenia on, że wzrost produkcji tych układów jednak nie będzie tak szybki jak niektórzy oczekują. W roku 1889 lub 1990 powinno się zużywać milion tych mikroprocesorów rocznie, co odpowiada 200 mln. dol. Łącznie z urządzeniami peryferyjnymi, oprogramowaniem i systemami wspomagania wyniesie to około miliarda dol. Niektórzy przedstawiciele przemysłu oceniają tę wartość na 2-3 miliardy, lecz nie wydaje się to pewne, gdyż mikroprocesory 32-bitowe są zbyt złożone by stosować je w dużych ilościach poza jednostkami centralnymi.

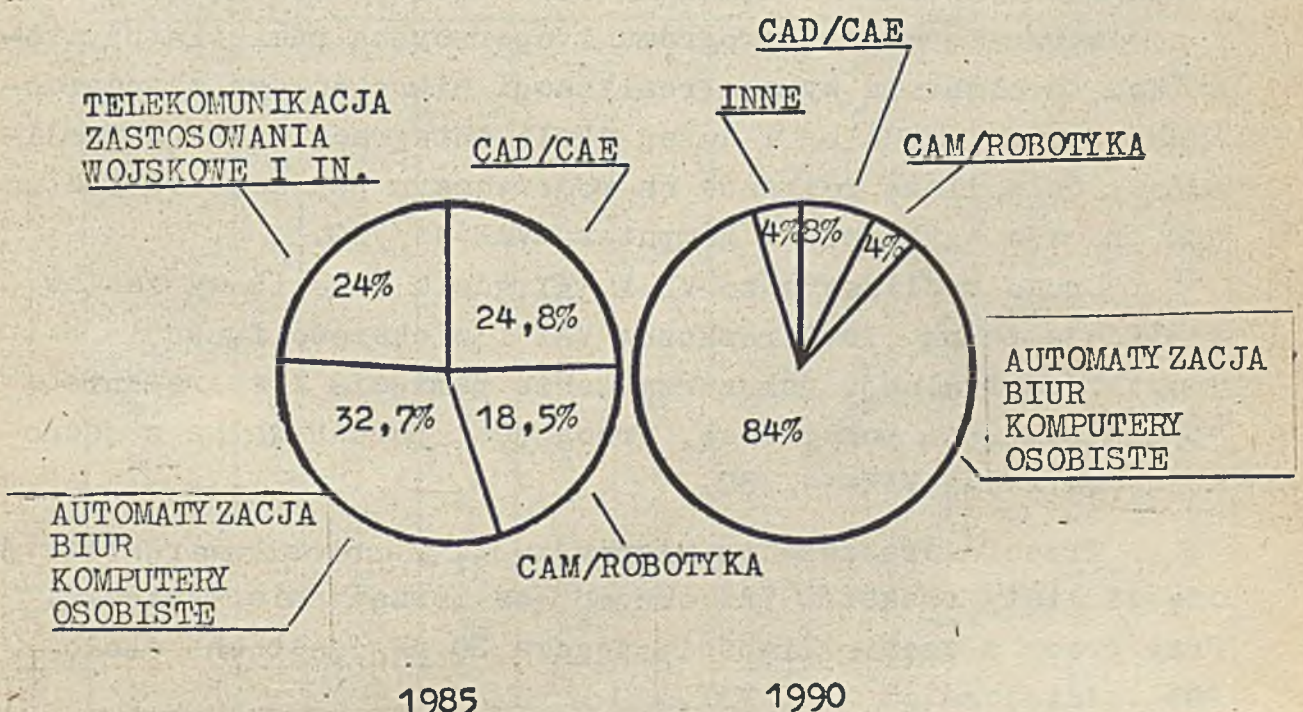
Obecnie mamy do wyboru mikroprocesory trzech firm. National Semiconductor Corp. oferuje swój układ 32032 będący

próbą pełnej realizacji potokowej architektury minikomputera dla języka wysokiego poziomu. Zarządzanie pamięcią i procesory zmiennoprzecinkowe są tu dołączane z zewnątrz, a adresowanie pamięci operacyjnej sięga do 16 megabajtów. Lista rozkazów zapewnia symetryczny dostęp do każdego rodzaju danych, a także komórek rejestrów i pamięci. Układ ten pojawił się w końcu 1983 roku i można go traktować za najbardziej popularny mikroprocesor 32-bitowy, gdyż zastosowano go już w prawie 1500 różnych systemów. Rozwiązanie technologiczne jest nie najnowocześniejsze - wytwarza się go w procesie n-MOS, a dostęp ścieżek wynosi 3,5µm. Przygotowywana jest nowa wersja w technologii CMOS z odstępem ścieżek 2 µm. Obecnie wytwarzane są wersje o częstotliwości zegara 6 i 10 MHz. Przygotowywana jest poprawiona konstrukcja w technologii n-MOS o częstotliwości zegara 15MHz, a docelowa wersja CMOS ma mieć 25MHz. W drugiej połowie 1985 r. ta sama firma ma rozpocząć dostawę układu 32132, który ma ponadto układ przydziału szyn dla ściśle sprzężonego układu dwuprocesorowego. Natomiast w drugiej połowie 1986 roku ma pojawić się układ 32C132, który będzie jednostką centralną w technologii CMOS z zarządzaniem pamięcią i szybką pamięcią pomocniczą (cache).

Firma Motorola dostarcza od połowy 1984 roku układ 68020 będący pełną, 32-bitową wersją popularnego mikroprocesora 16-bitowego tej firmy 68000. Układ ten realizuje istniejące programy binarne. Stosowana jest tu technologia HCMOS, a częstotliwość zegara wynosi 16,67 MHz. Układ wyposażony jest też w szybką pamięć pomocniczą rozkazów, 4 Gbitową pamięć wirtualną, dynamiczny wybór szyn i przesuwnik dla przyspieszenia pracy. Czterokrotne zrównoleglenie (pipelining) umożliwia jednoczesne wykonanie rozkazów w różnych fazach. Układ ma 32 linie adresowe i 32 linie danych. Wersja 25 MHz ma być dostarczana w końcu 1986 roku. Mikroprocesor też ma już ponad 500 realizacji i liczba ta szybko rośnie ze względu na popularność 6800. Układ ten wytwarzany jest też przez francuską firmę Thomson - CSF.

Do połowy 1986 roku mają pojawić się na rynku trzy nowe mikroprocesory 32-bitowe. Każdy z nich stanowi możliwość zagrożenia pozycji omówionych powyżej ze względu na korzystniejsze parametry. Można tu nawet mówić o drugiej generacji tych mikroprocesorów. Czołowym osiągnięciem jest tu układ firmy Zilog Z80000 stanowiący znaczny postęp w stosunku do istniejących procesorów 32-bitowych, zarówno jeśli chodzi o architekturę, jak i o złożoność procesu wytwarzania. W przeciwieństwie do większości wytwórców układów 32-bitowych, opracowano tu specjalny proces n-MOS o dostępie ścieżek 2μm pozwalający na uzyskanie różnych tranzystorów i wielopoziomowych połączeń, co umożliwiło uzyskanie częstotliwości zegara 25 MHz. Proces ten umożliwił uzyskanie za niższą cenę produktu o mniejszych wymiarach i szybszego niż większość realizacji CMOS, przy czym zużycie mocy jest takie samo jak w układach konkurencyjnych. Układ wyposażony jest w pomocniczą szybką pamięć, zarówno dla danych, jak i rozkazów (68020 mają tylko dla rozkazów, a 80386 i 32032 nie mają wcale). Taka pamięć w kostce minimalizuje dostęp do pamięci i tym samym ogranicza ruch na szynach. Daje to 20% poprawy działania, co jest istotne w zastosowaniach wojskowych i naukowych. Drugim istotnym czynnikiem jest zastosowanie architektury o sześciu poziomach zrównoleglenia, a ostatnim zawarte w kostce zarządzanie pamięcią pozwalające na 4 gigabajtowy obszar pamięci wirtualnej. Jednostka centralna może generować adresy fizyczne eliminując opóźnienia występujące przy sterowaniu pamięcią z zewnątrz. Mając w układzie rejestr sterujący połączeniami między sprzętem i interfejsem oraz sześć programowanych generatorów opóźnień, można określić własności sprzętu wokół jednostki centralnej uwzględniając prędkość przesyłania po szynach, liczbę linii danych i opóźnienia. Można spowodować, że przesyłanie po szynach zewnętrznych odbywać się będzie z prędkością działania jednostki centralnej lub dwa razy wolniej, gdy jednostka ta pracuje z zegarem 25 MHz. Wynikiem usprawnień technicznych są uzyskane najlepsze parametry na rynku 32-bitowym, ale nie wiadomo czy to wystarczy.

Układ 80386 firmy Intel jest w zasadzie rozszerzeniem architektury 16-bitowego 80286. Próbki tego układu mają być dostępne w pierwszym kwartale 1986 roku. Wyposażony jest on w stronicowaną na żądanie pamięć wirtualną, zakres adresowania do 32 Mbajtów, zarządzanie pamięcią do 4 gigabajtów oraz zrównoleglanie operacji przez kolejkowanie rozkazów. Na ma natomiast szybkiej pamięci pomocniczej. Technologicznie różni się od 80286 (proces HMOS) zastosowaniem procesu HC MOS-III i oczywiście ma 32 szyny pamięciowe i 32-bitową architekturę wewnętrzną. Częstotliwość zegara wynosi 16 MHz, jest więc trzykrotnie szybszy od swego poprzednika 16-bitowego. Daje to przetwarzanie 3,5-4 milionów rozkazów na sekundę. Firma Intel wskazuje tutaj, że jej układ 16-bitowy 286 ma lepsze parametry pracy od układów 32032 i 68020. Wykonuje on bowiem 1,4-1,5 mln. rozkazów na sekundę w porównaniu 1 mln. dla 32032 przy częstotliwości zegara 6 MHz. W systemie Unix, 10 MHz układ 80286 wykonuje 2,2 mln. rozkazów na sekundę - mniej więcej tyle samo co 68020 o częstotliwości zegara 16.67 MHz. Przy częstotliwości zegara 12,5 MHz 286 wykonuje 2,7 mln. rozkazów na sekundę.



Na początku tego roku firma AT&T wprowadziła układ WE 32100 wytwarzany w technologii CMOS przy odstępie ścieżek 1,5 um. Obecnie osiągalne są wersje o częstotliwości zegara 10 i 14 MHz. Układy te mają wytwarzać też firmy Mostek i GE/Intersil.

Oprócz AT&T co najmniej trzy inne firmy wytwarzające duże systemy opracowały 32-bitowe jednostki centralne na własny użytek. Data General zrealizowała architekturę swego komputera Eclipse na pięciu kostkach bardzo dużej skali integracji. Zastosowano tu proces n-MOS z 2 µm odstępami. Układ pracuje o połowę wolniej niż superminikomputer MV/8000 tej firmy i może posługiwać się 4 Gbajtami pamięci wirtualnej. Jednostka centralna jest w pełni 32-bitowa i ma 3 poziomy zrównoleglenia. Zawiera ona jednostkę arytmetyczno-logiczną, 4 akumulatory, 8 rejestrów ogólnego użytku i 12 rejestrów specjalnych, wszystkie o długości 32 bitów. Układ translacji adresów zapewnia zarządzanie pamięcią w stronicach 2 kbajtowych o obszarze adresów fizycznych 128 Mbajtów. Lista rozkazów komputera Eclipse jest zrealizowana na oddzielnej kostce (microsequencer).

Firma DEC opracowała 3 oddzielne realizacje architektury superminikomputera VAX. Pierwsza z nich to micro VAX-1, trzykostkowy zestaw obejmujący 32-bitową jednostkę centralną z zewnętrzną pamięcią programu i operacyjną pamięcią użytkownika. Ta ostatnia wymaga realizacji mikrokodu ze standardowych układów średniej i dużej skali integracji. Układ realizujący całą listę rozkazów na pojedynczym pakiecie 50 kostkowym pracuje szybciej od komputera VAX-11/730.

Druga realizacja to VLSI VAX, gdzie 9-kostkowy zestaw realizuje pełną listę rozkazów VAX i większość funkcji jednostki centralnej, jak zarządzenie pamięcią i sterowanie szybką pamięcią pomocniczą. Praca jest porównywalna z jednostką centralną VAX-11/780.

Trzecia wreszcze to Microvax-32, jednokostkowa realizacja części listy rozkazów VAX obejmująca zarządzanie pamięcią. Przy pracy z częstotliwością zegara 20 MHz jest ona około 20% wolniejsza od VLSI VAX.

Tab. 1

Procesor	National Semiconductor NS32032	Motorola MC68020	Zilog Z80000	Intel 80386	AT&T WE32100
1.	2	3	4	5	6
Technologia	n-MOS	CMOS z jamą n	n - MOS	CMOS z jamą n	Domino CMOS
Odstęp ścieżek (µm)	3,5	2,25	2	1,5	1,5
Częstotliwość zegara [MHz]	4/6/10	16,67	10/18/25	12/16	10/14
Pobór mocy [W]	1,5	2	2	1,5÷2,5	0,7
Pomocnicza pamięć rozkazów (bajty)	0	256	256	0	0
Pomocnicza pamięć danych (bajty)	0	0	256	0	256
Zakres adresowania (bity)	24	32	32	32	32
Linie danych (adresowe)	multiplekowane	niemultipleksowane	multiplekowane	multiplekowane	niemultiplekowane
Ilość sposobów adresowania	9	20	9	24	9

1	2	3	4	5	6
Zarządzanie pamięcią	stronico-wanie	stronico-wanie/ segmenta-cja	stronico-wanie (w kostce)	stronico-wanie/ segmen-tacja (w kostce)	stronico-wanie/ segmenta-cja
Ilość rozkazów	86	65	110	111	169
Kolejkowanie rozkazów	8 bajtów	2 słowa	6 stopni	4 pozio-my	4 stopnie
Blokowe przesyłanie do pamięci	nie ma	nie ma	istnieje	nie ma	nie ma
Rejestry ogólnego przeznaczenia	8	16	16	16	16
Rejestry specjalne	8	8	0	0	0
Termin sprzedaży	VI.85	VI.85	1986	1986	VI.85

Prawdopodobnie jedną z najbardziej zaawansowanych spośród dotychczas opracowanych realizacji 32-bitowych jest jednostka centralna Focus firmy Hewlett-Packard o procesie n-MOS i odstępie 1 µm, stosowana tylko w urządzeniach tej firmy, przede wszystkim w minikomputerze 9000. Charakteryzuje się ona stronicowaniem pamięci na żądanie, zakresem adresów do 500 Mbajtów, zegarem 18 MHz i 38 bitowym słowem rozkazowym. Lista rozkazów zawiera 230 tych rozkazów. Kostka ta wytwarzana od końca 1980 r. jest już przestarzała. Jej następcą ma być procesor 32-bitowy o zredukowanej liście rozkazów o nazwie Spectrum.

Wiele firm opracowało własne 32-bitowe jednostki centralne oferując mikrokomputerową wersję swych minikomputerów i dużych komputerów. Jeśli nawet służą do użytku wewnętrznego stanowią one konkurencję dla systemów mających kupowane jednostki centralne. W rezultacie wzrastają inwestycje na badania i aby sprostać wydatkom, firmy te "ujawniają" swe mikroprocesory tak, jak to zrobiła AT&T. Przewiduje się, że stopniowo opracowanie wewnętrzne odgrywać będą mniejszą rolę. Początkowo dostawcy układów musieli dostosowywać się do opracowań różnych firm. Później jednak oni będą decydować w większym stopniu. Tylko te z własnych opracowań mogą odegrać rolę, które będą kompatybilne z istniejącymi systemami. W rezultacie, o ile własne opracowania są obecnie decydujące, to na początku lat dziewięćdziesiątych stosunek ten gwałtownie się zmieni. Do roku 1995 prawdopodobnie 80% całego rynku mikroprocesorów pochodzić będzie z zakupu.

Co najmniej trzy firmy zdecydowały się nie uczestniczyć we współzawodnictwie nad projektami uniwersalnymi, ale każda z nich opracowała inny układ specjalistyczny. Układ NCR/32 firmy NCR jest zestawem 4 kostek segmentowych o bardzo dużej skali integracji wyrabianych w procesie n-MOS. Obecnie myśli się o wersji CMOS. Są to: 32-bitowa jednostka centralna, kostka translacji zakresu, kostka zmiennoprzecinkowa i sterownik interfejsu i systemu. Ponadto zewnętrzna pamięć stała

o pojemności 128 kbajtów, zwana pamięcią rozkazów, zawiera zewnętrzny mikrokod potrzebny do dostosowania zestawu kostek do wymagań użytkownika. Jednostka centralna wykonuje 179 prostych rozkazów w trzypoziomowym zrównolegleniu. Przy 13,3 MHz NCR/32 realizuje oprogramowanie IBM MVS z taką samą prędkością jak IBM 4331. Emulując środowisko DEC VMS jest ona 5 razy wolniejsza od VAX-11/780, lecz używając własnego kodu jest trzy razy szybsza od VAX.

Jeszcze ambitniejsze zadania ma układ firmy AMD 29300/29400. Jest to zestaw kostek segmentowych mający dać wydajność 3-10 razy większa niż VAX-11/780. Jest on realizowany tak jak AMD 2900 tylko ma 32-bitowe dane. W końcu 1985 roku ma być osiągalny układ 29300 o wewnętrznych układach logicznych ze sprzężeniem emiterowym i interfejsem TTL. Cykl tego układu wynosi 70-80 ns. Układ 29400 jest szybszy cykl (40-55 ns) i wyposażony jest w układy ECL. Firma przygotowuje wersje CMOS.

Układ firmy Inmos zwany transputerem T424 jest realizowany w technologii CMOS z odstępem 2 µm. Jest to 32-bitowa jednostka centralna o zredukowanej liście rozkazów z szybką pamięć pomocniczą 4k, zaprojektowana do stosowania w systemach wieloprocesorowych. Przy częstotliwości zegara 5 MHz T424 wykonuje jeden format rozkazów stanowiący prawie 80% typowych wykonywanych rozkazów w przeciętnym systemie. Każda jednostka centralna ma wielokrotne kanały szybkiej komunikacji pozwalające jej na dołączenie do innych podzespołów przy minimalnych układach interfejsu. Każdy transputer ma w kostce co najmniej 4 pełnoduplexowe linie, co pozwala na tworzenie wielotransputerowych systemów o dowolnej wielkości. Wśród innych układów tego typu należy wymienić: Clipper firmy Fairchild CMOS , który ma pracować z oszałamiającą częstotliwością 40 MHz, ROMP i 801 IMP firmy IBM i pierwszą jednostkę centralną z arsenku galu firmy Mc Donnell Douglas Corp. oraz 32-bitową kostkę o zredukowanej liście rozkazów firmy TRW opracowaną w programie Bardzo Szybkich

Układów Scalonych Departamentu Obrony. Również firmy japońskie jak: Hitachi, NEC, Fujitsu, Matsushita, Toshiba i Mitsubishi opracowują 32-bitowe jednostki centralne. Najbardziej zaawansowane są Micro 32 Hitachi i seria V NEC. Pierwsza z nich zawiera 32-bitowe szyny adresowe i danych, jednostkę arytmetyczno-logiczną i 200 kbitów pamięci stałej. Jest kompatybilna z 68020 i ma wykonawać 5 mln rozkazów na sekundę. Firma ma nadzieję dostarczyć próbki w końcu 1986 roku i rozpocząć produkcję w 1987 r.

Seria V ma być kompatybilna z kostkami Intela 286/386. W pierwszej połowie 1987 projektowana jest jednostka centralna V-70.

Przewiduje się, że rynek układów 32-bitowych rozwinie się bardziej niż łączny rynek układów 8- i 16-bitowych, ale wzrost ten będzie powolny. Widzimy to na przykładzie takich procesorów 16-bitowych jak Z8000 i 8086, których produkcja rozpoczęta w 1978 roku nie zwiększała się gwałtownie aż do 1983 r. I nawet w końcu 1984 r. procesory 16-bitowe reprezentowały tylko 12% wszystkich sprzedanych procesorów. Analogiczny, ewolucyjny wzrost można przewidzieć dla procesorów 32-bitowych. Prawdziwy rozwój zacznie się 4-5 lat po ich wprowadzaniu, a więc w latach 1988-89, a zaczną dominować na rynku w połowie lat dziewięćdziesiątych. Do tej pory projektanci będą porównywać i wybierać spośród różnych rozwiązań.

Wśród procesorów 8 bitowych było wprowadzonych 10 istotnych odmian architektonicznych, z których 5 stanowi dziś 85% rynku. Wśród układów 16-bitowych 4 spośród 6 głównych jednostek centralnych stanowi dziś też 85%. Łącznie z opracowaniami wewnętrznymi, japońskimi i różnymi jednostkami centralnymi z uniwersytetów i prywatnych laboratoriów badawczych, istnieje około 50 układów 32-bitowych. Można przypuszczać, że stopniowo prawdopodobnie 5 czołowych rozwiązań stanowić będzie 80% rynku, a reszta zmieści się w pozostałych 20%, lecz biorąc pod uwagę rozmiary tego rynku kilka z nich

może przetrwać. Najwięcej będą mieli do powiedzenia wytwórcy układów 8- i 16-bitowych, lecz tak jak przy układach 16-bitowych pojawili się nowi wytwórcy, tak będzie i z 32-bitowymi. Największe szanse mają National Semiconductor ze względu na wczesną ofertę i rozległy rynek, IBM i Intel przez dominację ich komputerów osobistych i Zilog z zaletami technicznymi układu Z80000. Następnie należy wymienić AT&T m.in. ze względu na system operacyjny, którym dla systemów 32-bitowych będzie w znacznej mierze UNIX, opracowany właśnie w tej firmie.

Dla pewnych zastosowań wystarczają układy 8-bitowe i będą one nadal wytwarzane ewentualnie ze zwiększoną szybkością. Mikroprocesory 16-bitowe dominują dziś w stanowiskach do automatycznego programowania, automatyzacji biur i bardziej złożonych komputerach osobistych. Najbardziej znane jednostki centralne to 68000, 68010, 16016 i 16032, nieco mniej jest Z8000 i 8026. Te zastosowania stanowić będą obszar, gdzie w początkowym okresie pojawią się układy 32-bitowe, głównie 68020 i 80386.

Trzecią grupę stanowią tradycyjni użytkownicy minikomputerów, mający duże doświadczenie w oprogramowaniu. Rozumieją oni inowacje wprowadzane w układach 32-bitowych takie, jak zrównoleglanie, zabezpieczenie pamięci, pamięć wirtualna i szybka pamięć pomocnicza. Tutaj znajdą zastosowania bardziej złożone z istniejących układów oraz nowe projekty jak Z80000, jednostka centralna Fairchilda i niektóre architektoniczne nowości japońskie.

Ostatnią wreszcie kategorią będą zupełnie nowe rozwiązania, które nie występują w istniejących systemach, a znajdą zastosowanie w różnych dziedzinach. W komputerach osobistych i domowych systemy takie łączyć będą możliwości 32-bitowego przetwarzania z nowym oprogramowaniem dla systemów ekspertowych, przetwarzania obrazów i sygnałów, syntezy mowy i dużych pojemności pamięci, przy czym koszt będzie niższy.

Wówczas nastąpi gwałtowny rozwój do znacznej części rynku masowego, który obecnie nie istnieje. Będzie to miało miejsce w końcu tego wieku i trudno obecnie przewidzieć kto zwycięży w tym wyścigu. Zależeć to będzie od tego, kto zostanie w poprzednich cyklach i kto poczyni niezbędne inwestycje w opracowanie oprogramowania, które umożliwi realizację tych zastosowań.

Opracował J. Ryżko na podstawie
Electronics Week z 1985.06.03

Informacja o cenach i warunkach prenumeraty na 1986 r.
- dla czasopism Instytutu Maszyn Matematycznych

● Cena prenumeraty rocznej

Techniki Komputerowe - Biuletyn Informacyjny	1560.-	dwum.
Przegląd Dokumentacyjny - Nauki i Techniki Komputerowe	1260.-	dwum.
Informacja Ekspresowa - Nauki i Techniki Komputerowe	2400.-	mies.
Prace naukowo-badawcze Instytutu Maszyn Matematycznych	660.-	3x w roku

○ Warunki prenumeraty

1/ dla osób prawnych - instytucji i zakładów pracy:
- instytucje i zakłady pracy zlokalizowane w miastach wojewódzkich i pozostałych miastach, w których znajdują się siedziby oddziałów RSW "Prasa-Książka-Ruch" zamawiają prenumeratę w tych oddziałach;
- instytucje i zakłady pracy zlokalizowane w miejscowościach, gdzie nie ma oddziałów RSW "Prasa-Książka-Ruch" i na terenach wiejskich opłacają prenumeratę w urzędach pocztowych i u doręczycieli;

2/ dla osób fizycznych - prenumeratorów indywidualnych:
- osoby fizyczne zamieszkałe na wsi i w miejscowościach, gdzie nie ma oddziałów RSW "Prasa-Książka-Ruch" opłacają prenumeratę w urzędach pocztowych i u doręczycieli;
- osoby fizyczne zamieszkałe w miastach - siedzibach oddziałów RSW "Prasa-Książka-Ruch" opłacają prenumeratę wyłącznie w urzędach pocztowych nadawczo-oddawczych właściwych dla miejsca zamieszkania prenumeratora. Wpłaty dokonują używając "blankietu wpłaty" na rachunek bankowy miejscowego oddziału RSW "Prasa-Książka-Ruch";

3/ Prenumeratę ze zleceniem wysyłki za granicę przyjmuje RSW "Prasa-Książka-Ruch", Centrala Kolportażu Prasy i Wydawnictw, ul. Towarowa 28, 00-958 Warszawa, konto NBP XV Oddział w Warszawie nr 1153-201045-139-11. Prenumerata ze zleceniem wysyłki za granicę pocztą zwykłą jest droższa od prenumeraty krajowej o 50% dla zleceniodawców indywidualnych i o 100% dla zlecających instytucji i zakładów pracy.

● Terminy przyjmowania prenumeraty na kraj i za granicę:

- do dnia 11 listopada na I kwartał, I półrocze roku następnego oraz na cały rok następny,
- do dnia 1-każdego miesiąca poprzedzającego okres prenumeraty roku bieżącego.

Zamówienia na prenumeratę "Prac naukowo-badawczych Instytutu Maszyn Matematycznych przyjmuje Dział Sprzedaży Wysyłkowej Ośrodka Rozpowszechniania Wydawnictw Naukowych PAN, Warszawa, Pałac Kultury i Nauki, tel. tel.20-02-11 w.2516. Egzemplarze pojedyncze Prac są do nabycia w księgarni ORWN PAN, Warszawa, Pałac Kultury i Nauki, tel.20-02-11 w.2105.

Druk IMM zam. 87/85 nakł. 400 egz.

Cena zł. 260.-

informacja ekspresowa

NAUKI
I TECHNIKI
KOMPUTEROWE

Instytut Maszyn Matematycznych zawiadamia, że od 1984 r., po dwuletniej przerwie, wznawia wydawanie miesięcznika "Informacja ekspresowa - Nauki i Techniki Komputerowe". W czasopiśmie zamieszczamy opisy bibliograficzne /wraz z krótkimi notatkami objaśniającymi/ dokumentów źródłowych, które znajdują się w bibliotece IMM - najlepiej zaopatrzonej w branży komputerowej.

Dokumentujemy ok. 600 pozycji książkowych rocznie /krajowych, i zagranicznych/ oraz 184 tytuły czasopism /około 2000 zeszytów/ w językach: polskim, angielskim, rosyjskim, niemieckim, czeskim; katalogi i in.

Informacja ekspresowa NiTK informuje o najnowszych publikacjach z zakresu branży komputerowej i dziedzin pokrewnych oraz nauk związanych z branżą /monografie, słowniki, podręczniki, materiały szkoleniowe, artykuły w czasopismach, przyczynki, któtkie notatki o najnowszych zdobyczach techniki komputerowej na świecie itp./ jest więc podstawowym i niezbędnym narzędziem pracy każdego pracownika naukowego, studenta, inżyniera - praktyka, projektanta i in.

Nasi Czytelnicy mogą zamawiać mikrofilmy i kserokopie dokumentów, których opisy znajdują się w Informacji ekspresowej.