ZJEDNOCZENIE PRZEMYSŁU
AUTOMATYKI I APARATURY POMIAROWEJ
„MERA"

ELEKTRONICZNA TECHNIKA OBLICZENIOWA

NOWOŚCI

2/1973

INSTYTUT MASZYN MATEMATYCZNYCH
BRANŻOWY OŚRODEK INTE

ELEKTRONICZNA TECHNIKA OBLICZENIOWA

N O W O Ś C I

KWARTALNIK

Rok XII Nr 2 1973

S p i s t r e ś c i

str.

Wydaje

INSTYTUT MASZYN MATEMATYCZNYCH

Branżowy Ośrodek Informacji Naukowo-Technicznej
i Ekonomicznej

Adres Redakcji: Warszawa, ul. Krzywickiego 34,
tel. 28-37-29 lub 21-84-41 wewn. 431

Druk IMM z. 107/73 n. 1200 pap. offset. kl. III g. 70. R-30. INDEKS-35804

Inż. Jerzy SUKIENNIK
Stocznia im. Komuny Paryskiej
Gdynia

681.322-181.4.004.14:621.322.64

SEECHECK - SYSTEM PRZYGOTOWANIA DANYCH NA TAŚMIE MAGNETYCZNEJ

Celem artykułu jest opis koncepcji wykorzystania systemu SEECHECK produkcji firmy brytyjskiej REDIFON ELECTRONIC SYSTEMS LTD[1] w warunkach polskich. Proponowana koncepcja oparta jest na studiach porównawczych podobnych systemów przeprowadzonych zarówno przez Stocznię jak i inne ośrodki w Polsce. Wypracowany przez autora artykułu zakres zastosowania standardowego systemu SEECHECK oraz urządzeń dodatkowych znacznie wybiega poza dostępne opisy firmowe oraz przekracza krąg dotychczasowych zastosowań podobnych systemów na Zachodzie[2].

1. Wstęp

Obserwujemy bardzo szybki rozwój systemów komputerowych służących do automatycznego przetwarzania danych. Znacznie wolniejszy postęp techniczny i organizacyjny występuje w dziedzinie produkcji i wykorzystania urządzeń służących do przygotowania danych na nośnikach maszynowych [3]. Do końca lat sześćdziesiątych najszersze zastosowanie miały dziurkarki i sprawdzarki, czyli urządzenia do przygotowania danych na kartach lub rzadziej na taśmie papierowej.

---

1 System SEECHECK produkowany jest w USA pod nazwą ENTREX 480.

2 Opierając się na artykule przeglądowym pt. "Data Preparation-Machines and Techniques" opublikowanym w numerze 5-6 Data Processing: na Zachodzie zainstalowane są setki systemów typu "key-to-disc" wyposażonych w dziesiątki tysięcy stanowisk do przygotowania danych.

Dopiero w 1965 r. firma amerykańska Mohawk Data Sciences (MDS) wprowadziła na rynek urządzenia do zapisywania danych bezpośrednio na taśmie magnetycznej (1100 magnetic tape data recorders).W ślad za nią poszły inne firmy zachodnie produkujące urządzenia peryferyjne[1]. Początkowo były to systemy jednostanowiskowe składające się z klawiatury, pamięci buforowej oraz urządzenia do zapisu i odczytu danych z taśmy magnetycznej. Później wprowadzono systemy wielostanowiskowe, ale również do bezpośredniego zapisu danych na taśmie magnetycznej (MDS - system 9000, Singer - system 10, Datapoint - system 2200 itp.). Systemy powyższe nazywane są umownie "key-to-tape". Modyfikacją tych systemów są produkowane w ostatnich latach systemy "display-to-tape" lub "cassette-based video systems", które dodatkowo wyposażone są w monitory ekranowe.

W końcu lat sześćdziesiątych i na początku siedemdziesiątych wprowadzono na rynek nowy rodzaj systemów przygotowania danych zwanych umownie "key-to-disc". Lista systemów najpowszechniej stosowanych w świecie podana jest w tabeli nr 1 (kolejność wg liczby instalacji podanej w Data Processing [12]).

Nazwa umowna systemów wielostanowiskowych "key-to-disc" pochodzi z okresu, kiedy systemy te wyposażone były w tzw. ślepe klawiatury, które nie miały możliwości wyświetlenia zawartości całego buforu pamięci operacyjnej przeznaczonego na rejestrowane dane. Ostatnio systemy te wyposażone są w małe monitory ekranowe o pojemności 240-480 znaków. W konsekwencji tego właściwsze byłoby określenie nazwy tych systemów "key-display-to-disc".

Powstaje pytanie, czy szybki rozwój produkcji i zastosowań systemów "key-to-disc" przejawia tendencję długotrwałą. Istnieją prognozy świadczące o tym, że już w 1975 r. [13] produkcja i zastosowanie systemów przygotowania danych na taśmie magnetycznej znacznie wyprzedzi produkcję dziurkarek i sprawdzarek. Urządzenia do automatycznego odczytywania dokumentów nie znajdą w najbliższej przyszłości szerszego zastosowania ze względu na to, że masowo stosowane dokumenty źródłowe wypełniane są, przynajmniej częściowo, ręcznie. Z tego powodu dane muszą być nadal ręcznie przenoszone na różnego rodzaju nośniki maszynowe za pomocą sta-

---

[1] Stan liczbowy tych urządzeń zainstalowanych do końca 1972 r. oceniany jest na ponad 50.000 [12].

nowisk wyposażonych w klawiatury. W związku z tym nawet nowoczesne wielostanowiskowe systemy przygotowania danych nie rozwiązują w zasadniczym stopniu problemu wyeliminowania uciążliwej pracy ręcznej operatorek, stępiają tylko ostrość tego problemu. Niektórzy specjaliści uważają jednak [3], że "zastosowanie urządzeń do zapisu informacji na nośnikach magnetycznych nie stanowi prostej zamiany nośnika, lecz oznacza jakościową zmianę w organizacji systemu przygotowania danych".

Tab. I. Niektóre systemy przygotowania danych typu "key-to-disc" [1]

Nazwa systemu	Producent	Maksymalna liczba stanowisk w systemie
Key Processing	CMC - Computer Machinery Company (USA) [2]	CMC5 - do 12 CMC9 - do 32
Inforex 1301 1302	Extel Group (USA)	1301 - do 8 1302 - do 16
Key Display System 2404	MDS - Mohawk Data Sciences (USA)	do 20
Key Edit 50	ICL (Wielka Brytania)	do 16
Seecheck (Entrex 480 - USA)	Redifon Electronic Systems Ltd (Wielka Brytania)	do 32
Keycheck	Redifon Electronic Systems Ltd. (Wielka Brytania)	do 126
System 2100 (DT 2100)	General Computer Systems Inc. (USA)	do 30
VIDEO 3270	IBM (USA)	do 256 monitorów ekranowych dołączonych do centralnej jednostki sterującej IBM 370

[1] Według artykułu opublikowanego w [12] w końcu 1973 r. CMC kontrolować będzie 35% rynku, Extel [Inforex] 22% (ponad 800 systemów), MDS 19%. Pozostałych 24% dotyczy innych producentów.

[2] Firmy amerykańskie mają swoje filie w Europie Zachodniej.

2. Ogólne zasady pracy wielostanowiskowych systemów przygotowania danych

Wspólną cechą systemów wielostanowiskowych zwanych umownie "key-to-disc" jest użycie pamięci pośredniej - dysku magnetycznego lub rzadziej magnetycznej pamięci bębnowej połączonej z programowaną centralną jednostką sterującą (minikomputerem). Systemy te służą do pośredniego zapisu danych na taśmie magnetycznej, która pozostaje nadal podstawowym nośnikiem wyjściowym w tych systemach oraz nośnikiem wejściowym w komputerze głównym[1].

Genezą powstania tych systemów są opisane we wstępie wielostanowiskowe systemy do bezpośredniego zapisu danych na taśmie magnetycznej ("key-to-tape") oraz minikomputery, które stały się powszechnie dostępne w latach sześćdziesiątych. Systemy wielostanowiskowe typu "key-to-disc", sterowane przez odpowiednio oprogramowany uniwersalny minikomputer, miały usunąć wady systemów "key-to-tape":

- konieczność stosowania dodatkowych urządzeń do łączenia poszczególnych partii danych w zbiory jednokrążkowe;
- niedogodność zmieniania stanowisk przez operatorki dla zachowania zasady, że operację klawiszowego sprawdzania wykonuje inna osoba niż ta, która zarejestrowała dane (w innym wypadku trzeba przekładać szpule z taśmami magnetycznymi);
- małe możliwości wykrywania błędów w danych przez oprogramowanie "mikroprocesora" systemu "key-to-tape".

Systemy "key-to-disc" nazywane są również komputerowymi systemami przygotowania danych lub zintegrowanymi systemami przygotowania danych. Poszczególne moduły sprzętu tych systemów tworzą układ zintegrowany pracujący pod kontrolą specjalnego systemu operacyjnego oraz programów użytkowych opracowanych w celu wszechstronnej kontroli rejestrowanych danych. Pozwala to na przeniesienie wielu procedur wykonywanych dotychczas za pomocą komputera głównego na peryferyjny system przygotowania danych, który wyposażony został w kilka identycznych urządzeń oraz ma podobne możliwości programowe.

---

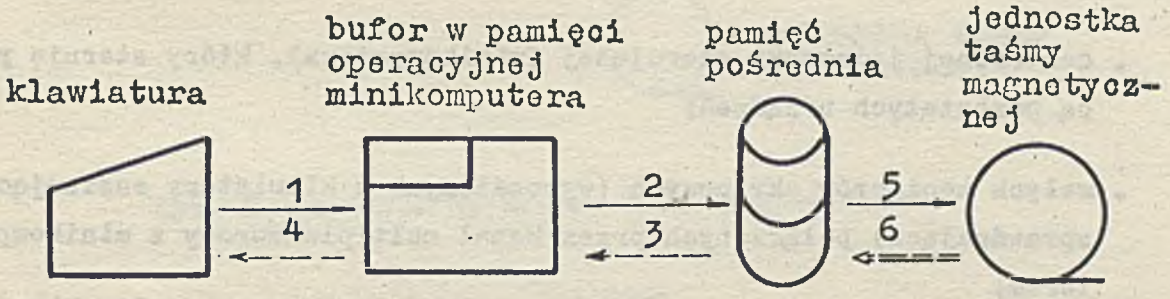
[1] Wyjątek stanowi wykorzystanie systemu do transmisji danych za pomocą łącz telefonicznych (zob. rys. 3).

Ogólnie istota rejestracji danych na taśmie magnetycznej w systemie "key-to-disc" może być scharakteryzowana następująco[1]:

1. Operator wprowadza dane z dokumentów źródłowych naciskając odpowiednio klawisze. Dane przechodzą znak po znaku do pamięci operacyjnej jednostki centralnej. Podczas wprowadzania kontrolowana jest programowo poprawność tych danych, a wszelkie błędy są natychmiast sygnalizowane operatorom w celu wyjaśnienia i skorygowania.

2. Po wprowadzeniu jednego zapisu (pozycji ewidencyjnej) z dokumentu dane przesłane są z pamięci operacyjnej do pamięci dyskowej.

3. Po zakończeniu wprowadzania wszystkich danych z dokumentów źródłowych (np. paczki - pliku) i zapisaniu ich na dysku następuje ręczne sprawdzanie za pomocą tej samej lub innej odpowiednio ustawionej klawiatury. Sprawdzanie nie zawsze jest konieczne w procesie przygotowania danych. W większości wypadków jest wykonywane ręcznie. Polega ono na powtórnym wprowadzaniu danych przez innego operatora oraz porównaniu ich z danymi poprzednio zapisanymi na dysku magnetycznym. Zakres i organizacja procesu sprawdzania (weryfikacji) zawarte są w systemie operacyjnym.



Rys. 1. Kolejność transferu danych

[1] Kolejność czynności wskazują cyfry arabskie na rys. 1

4. Po zakończeniu ręcznego sprawdzania wszystkich zapisów w danej partii (paczce) - lub bez sprawdzania ręcznego - dane są przygotowane do zapisania (transferu) blok po bloku na standardowej taśmie magnetycznej. Jednostką transferu jest w tym przypadku paczka, tzn. operować przepisywaniem można jedynie używając symbolu jednej lub kilku paczek (poza tym możliwy jest jednoczesny transfer wszystkich paczek). Dane tworzące paczkę zapisane zostaną w jednym lub kilku blokach na taśmie magnetycznej, w zależności od długości bloku. Po przepisaniu danych na taśmę magnetyczną dysk jest przygotowany do zapisu nowych danych.

5. W pewnych przypadkach dane zapisane na taśmie magnetycznej mogą być z powrotem załadowane na dysk magnetyczny, co umożliwia ponowne sprawdzenie i korektę danych. Ta sama procedura może być wykonywana również w przypadku taśmy magnetycznej zapisanej przez urządzenia taśmowe komputera głównego.

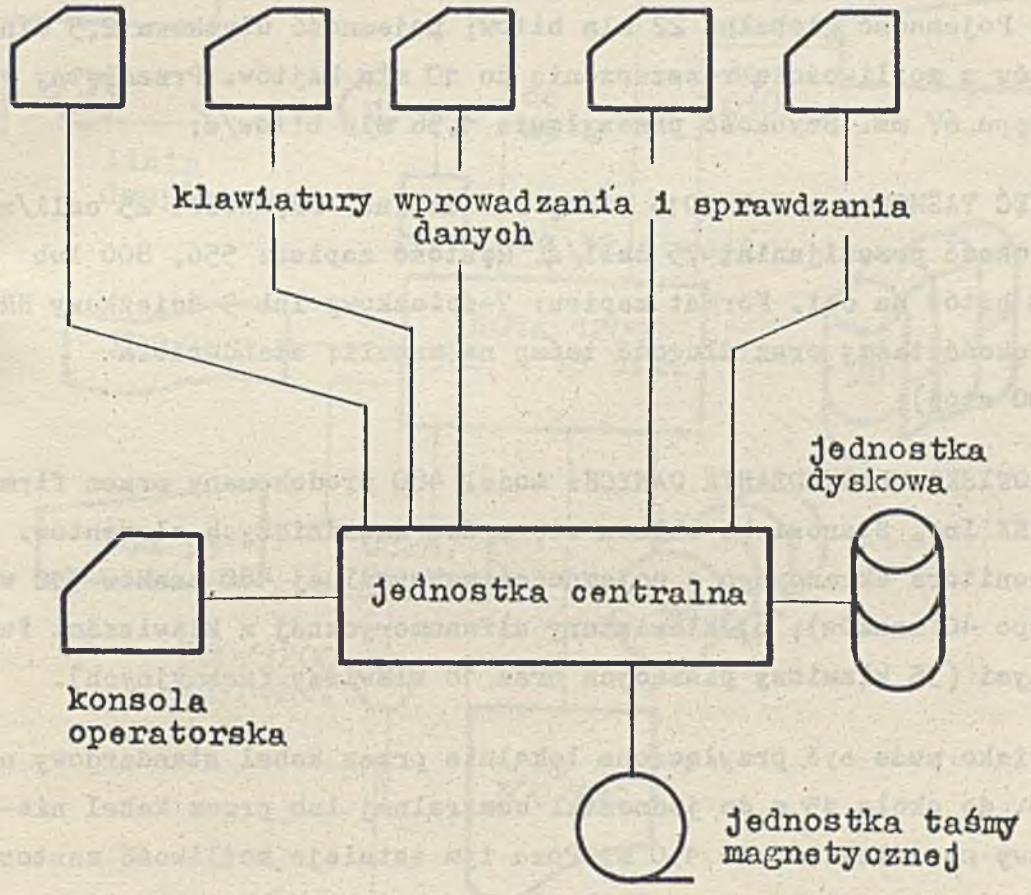
Przedstawione w tym punkcie ogólne zasady przygotowania danych za pomocą systemu "key-to-disc" nie wyczerpują zagadnienia. Dalsze rozwinięcie tej problematyki znajduje się w punkcie 4.

Podstawowy, tj. standardowy układ do przygotowania danych pokazany jest schematycznie na rys. 2. Układ ten składa się z następujących części:

- centralnej jednostki sterującej (minikomputera), który steruje pracą pozostałych urządzeń;
- małych monitorów ekranowych (wyposażonych w klawiatury zapisująco-sprawdzające) połączonych przez kanał multipleksorowy z minikomputerem;
- jednostki pamięci dyskowej lub bębnowej o dostępie bezpośrednim;
- jednostki pamięci taśmowej magnetycznej o dostępie sekwencyjnym;
- monitora dalekopisowego lub ekranowego służącego do operowania całym systemem.

Jednostka pamięci dyskowej oraz jednostka pamięci taśmowej magnetycznej są połączone z centralną jednostką sterującą przez odpowiednie jed-

nostki sterujące. Centralna jednostka sterująca wyposażona jest w pamięć operacyjną, w której przechowywany jest odpowiedni system operacyjny pozwalający na sterowanie pracą wszystkich urządzeń składowych za pomocą monitora dalekopisowego lub monitora ekranowego oraz zapewniający właściwą kontrolę wprowadzanych danych.



Rys. 2. Standardowy system typu "key-to-disc"

3. Ogólna charakterystyka systemu SEECHECK

System SEECHECK odpowiada amerykańskiemu systemowi ENTREX 480. Licencję na produkcję tego systemu i marketing na rynku europejskim zakupiła w 1972 r. firma brytyjska Redifon Electronic Systems Ltd., członek grupy Rediffusion Organisation.

Podstawowa konfiguracja SEECHECK obejmuje następujące urządzenia:

- CENTRALNA JEDNOSTKA STERUJĄCA: minikomputer programowany (uniwersalny), model NOVA 1200 produkowany przez firmę Data General Corp. (USA). Maksymalna pojemność pamięci operacyjnej 65.536 bajtów (32k słów 16 bitowych). Czas cyklu pamięci 1,2 μs. NOVA 1200 zbudowany jest na obwodach scalonych (LSI);

- PAMIĘĆ DYSKOWA: model 31 produkowany przez firmę Diablo Systems Inc. Pojemność globalna 22 mln bitów; pojemność użytkowa 2,5 mln bajtów z możliwością rozszerzenia do 10 mln bajtów. Przeciętny czas dostępu 67 ms. Szybkość przesyłania 1,56 mln bitów/s;

- PAMIĘĆ TAŚMOWA: model 2015 firmy Bucode Inc. Prędkość: 25 cali/s (prędkość przewijania: 75 cali/s). Gęstość zapisu: 556, 800 lub 1600 bitów na cal. Format zapisu: 7-ścieżkowy lub 9-ścieżkowy NRZI. Szerokość taśmy oraz długość taśmy na szpuli: standardowa (2400 stóp);

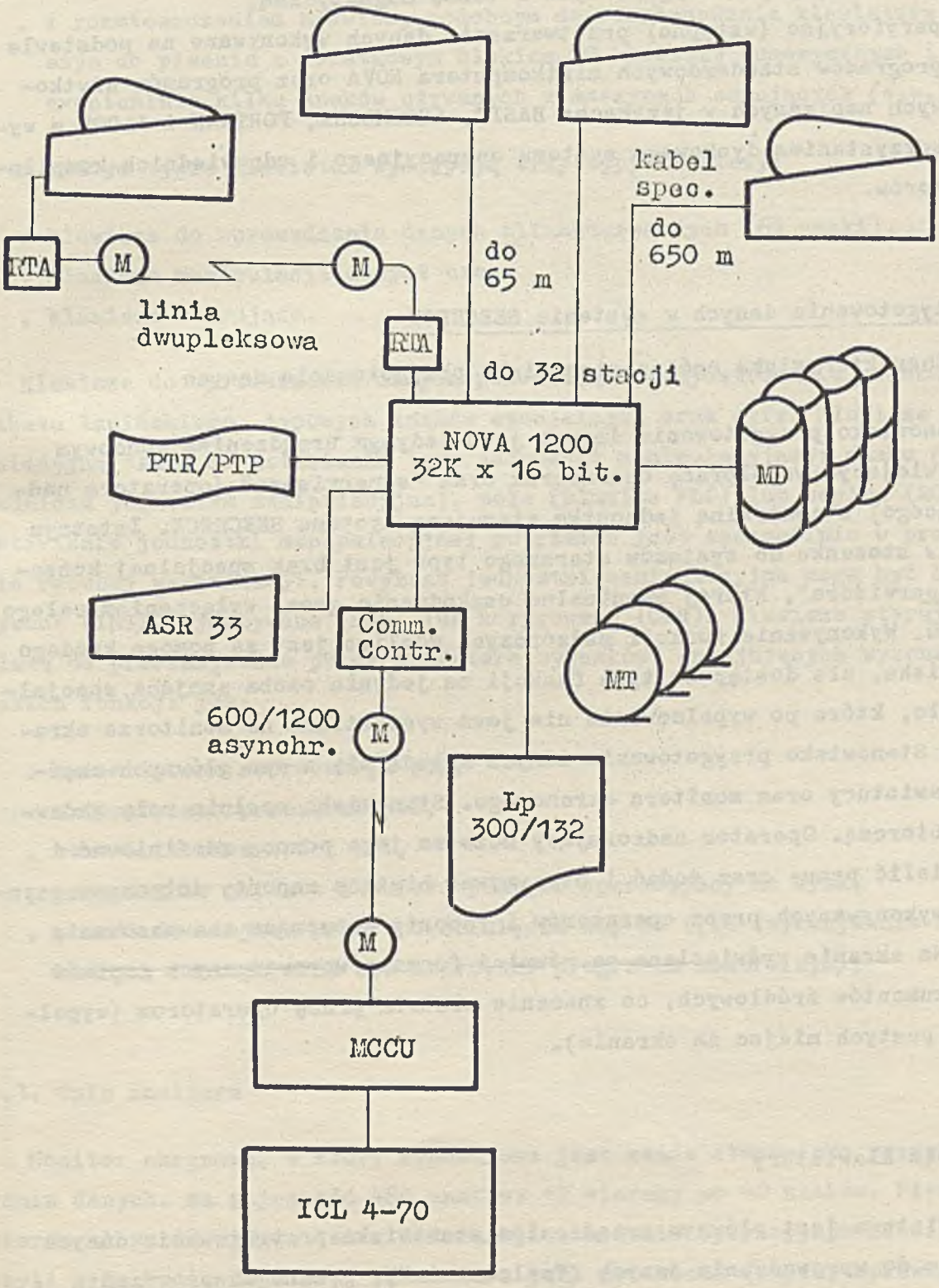
- STANOWISKO WPROWADZANIA DANYCH: model 480 produkowany przez firmę ENTREX Inc. Stanowisko składa się z dwu zasadniczych elementów: a) monitora ekranowego o pojemności maksymalnej 480 znaków (12 wierszy po 40 znaków); b) klawiatury alfanumerycznej z klawiszami funkcyjnymi (35 klawiszy piszących oraz 18 klawiszy funkcyjnych).

Stanowisko może być przyłączone lokalnie przez kabel standardowy na odległość do około 15 m do jednostki centralnej lub przez kabel niestandardowy na odległość do 150 m. Poza tym istnieje możliwość zastosowania modemów przy zdalnym przyłączeniu stanowisk. Maksymalna liczba stanowisk: 32.

Wymienione urządzenia stanowią konfigurację standardową stosowaną w większości przypadków zamiast poprzednio używanych urządzeń do przygotowania danych na kartach lub taśmie papierowej.

System SEECHECK może być dodatkowo wyposażony w jednostkę sterującą transmisją danych on-line do komputera głównego lub do drugiego systemu SEECHECK. Rozszerzoną konfigurację systemu obrazuje rys. 3. Pełna konfiguracja urządzeń systemu SEECHECK może spełniać następujące funkcje:

- ręczne wprowadzanie i sprawdzanie danych za pomocą klawiatur,
- programowa kontrola wprowadzania danych,



Rys. 3. Rozszerzona konfiguracja systemu SEECHECK

- transmisja danych on-line do komputera centralnego i odwrotnie,
- konwersja danych z jednych maszynowych nośników na inne, np. z kart lub taśmy papierowej na dysk i taśmę magnetyczną,
- peryferyjne (wstępne) przetwarzanie danych wykonywane na podstawie programów standardowych minikomputera NOVA oraz programów użytkowych napisanych w językach: BASIC, ASSEMBLER, FORTRAN i ALGOL z wykorzystaniem dyskowego systemu operacyjnego i odpowiednich kompilatorów.

4. Przygotowanie danych w systemie SEECHECK

4.1. Charakterystyka ogólna stanowiska przygotowania danych

Stanowisko przygotowania danych jest jedynym urządzeniem końcowym umożliwiającym współpracę operatorek oraz "supervisora" (operatora nadzorującego) z centralną jednostką sterującą systemu SEECHECK. Istotnym novum w stosunku do systemów starszego typu jest brak specjalnej konsoli "supervisora", której ewentualne uszkodzenie grozi wyłączeniem całego systemu. Wykonywanie funkcji nadzorczych możliwe jest za pomocą każdego stanowiska, ale dostęp do tych funkcji ma jedynie osoba znająca specjalne hasło, które po wypalcowaniu nie jest wyświetlane na monitorze ekranowym. Stanowisko przygotowania danych składa się z dwu głównych części: klawiatury oraz monitora ekranowego. Stanowisko spełnia rolę nadawczo-odbiorczą. Operator nadzorujący może za jego pomocą zdefiniować i przydzielić pracę oraz żądać i otrzymywać bieżące raporty dotyczące czynności wykonywanych przez operatorów i raporty dotyczące zaawansowania prac. Na ekranie wyświetlane są również formaty wprowadzanych zapisów lub dokumentów źródłowych, co znacznie ułatwia pracę operatorom (wypełnianie pustych miejsc na ekranie).

4.2. Opis klawiatury

Klawiatura jest głównym urządzeniem stanowiska przygotowania danych służącym do wprowadzania danych ("palcowania"), ręcznego sprawdzania danych (weryfikacji), badania i wyszukiwania danych oraz do wprowadzania programów kontrolnych i wykonywania funkcji nadzorczych ("supervisora"). System SEECHECK może mieć przyłączone stanowiska z dwoma typami klawiatur:

. z rozmieszczeniem klawiszy podobnym do rozplanowania klawiatury dziurkarki kart IBM 029 przedstawionym na rys. 4;

. z rozmieszczeniem klawiszy podobnym do rozplanowania klawiatury maszyn do pisania z dodatkowym blokiem 10 klawiszy numerycznych i ewentualnie kilku znaków używanych w maszynach sumujących (+,-.).

W każdym typie klawiatur występują trzy typy klawiszy:

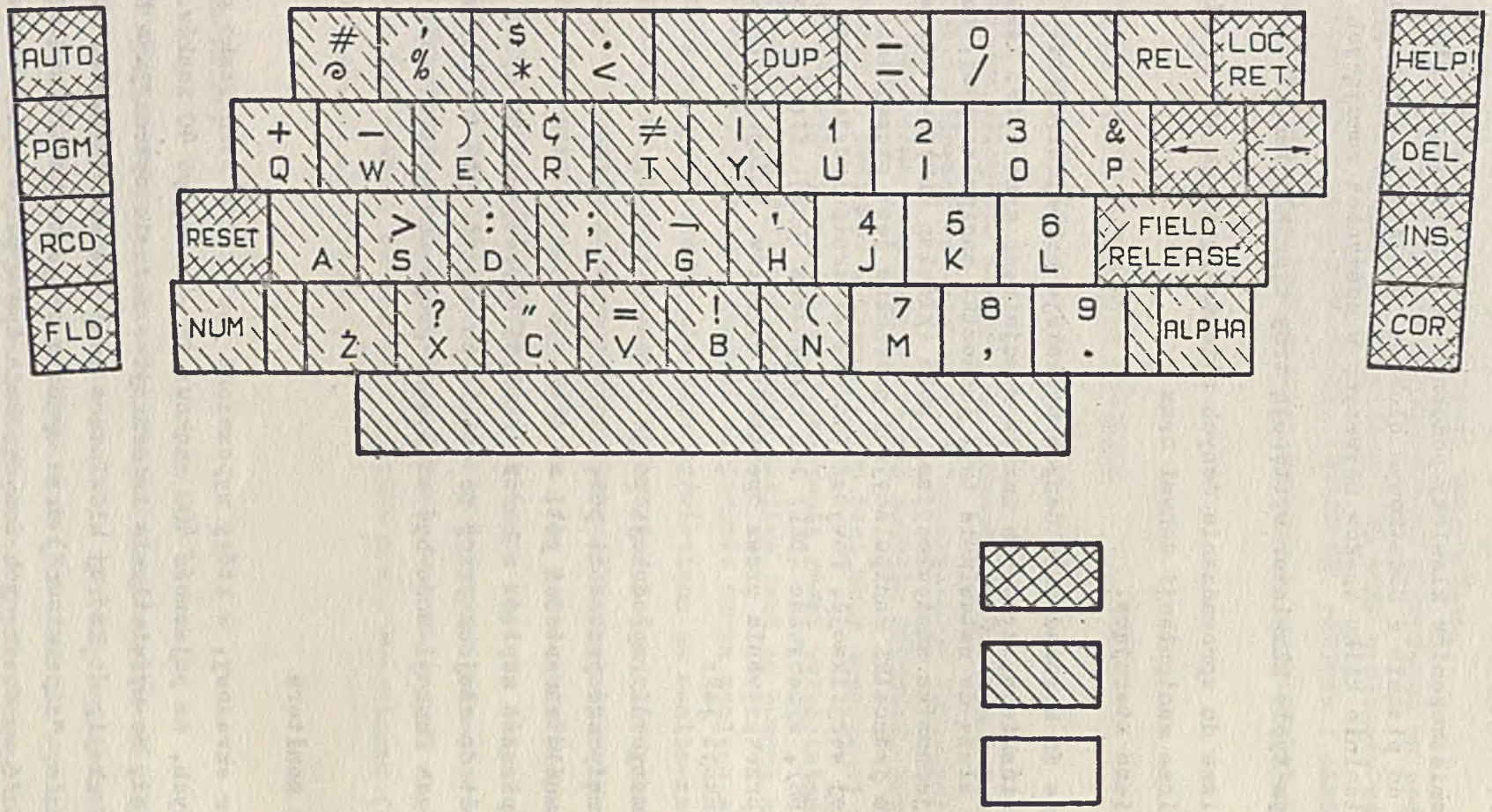
. klawisze do wprowadzania danych alfanumerycznych (64 znaki),
. klawisze manipulacji danymi oraz
. klawisze sterujące.

Klawisze do wprowadzania danych pozwalają na rejestrowanie znaków alfabetu łacińskiego, typowych znaków specjalnych oraz cyfr. Klawisze manipulacyjne służą do ustawiania tzw. jednostki manipulacyjnej: znaku (zasadnicza jednostka manipulacyjna), pola (klawisz FLD) lub zapisu (RCD). Ustawienie jednostki manipulacyjnej potrzebne jest szczególnie w procesie ręcznej weryfikacji. Powyższe jednostki manipulacyjne mogą być dopisywane (INS), wymazywane (DEL) lub korygowane (COR). Klawisze sterujące służą do przekazywania przez operatora sygnałów inicjujących wykonanie takich funkcji jak:

. automatyczna reprodukcja pól,
. automatyczne przeskoki pól,
. kasowanie zawartości pól,
. przepisanie zapisów z buforu pamięci operacyjnej na dysk,
. powrót do miejsca pracy po cofnięciu się do tyłu (wykonywanie niektórych funkcji może być sterowane programem kontrolnym).

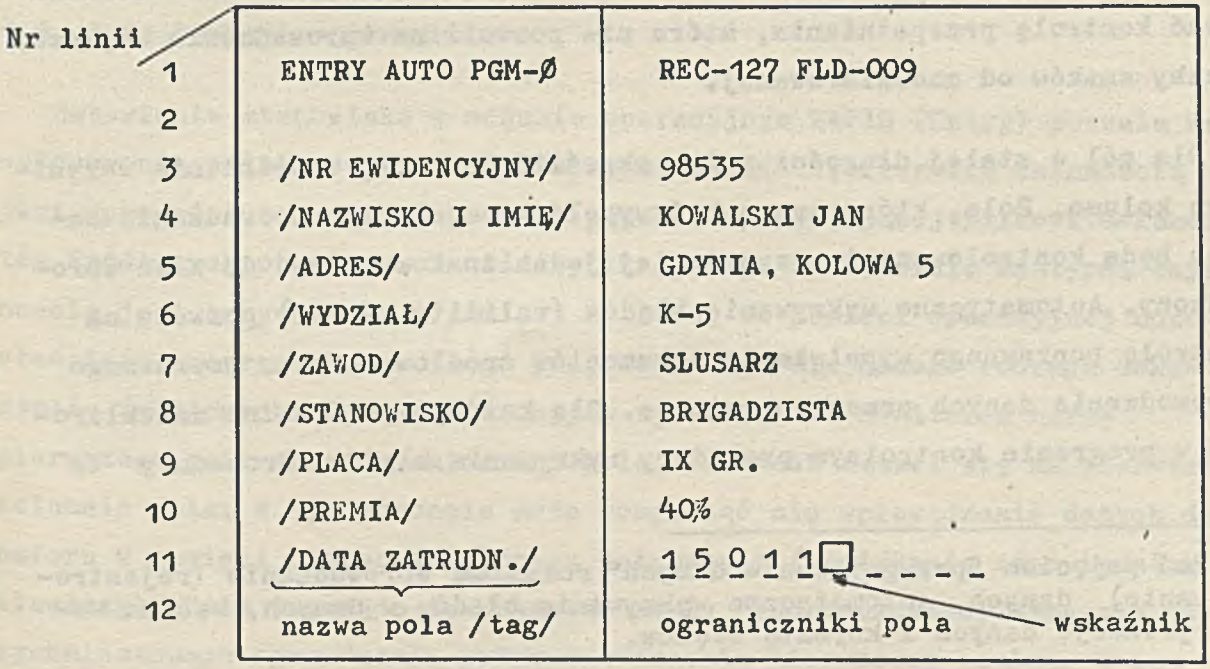
4.3. Opis monitora

Monitor ekranowy, w który wyposażone jest każde stanowisko przygotowania danych, ma pojemność 480 znaków: 12 wierszy po 40 znaków. Pierwszy wiersz służy do wyświetlania informacji o modusie operacyjnym klawiatury, czyli o funkcji, do której klawiatura jest ustawiona (zapis, sprawdzanie, badanie, "supervisor") oraz symbolu programu kontrolującego format i poprawność wprowadzanych danych. Poza tym w pierwszym wierszu wyświetlany jest bieżący numer pola. Drugi wiersz służy do wyświetlania infor-



Rys. 4. Schemat rozplanowania klawiszy SEECHECK na klawiaturze typu dziurkarka kart

macji diagnostycznych dla operatora nadzorującego (funkcja "supervisor") lub dla operatora przygotowującego dane. Sygnały wyświetlane są w języku angielskim lub innym, np. polskim - po wprowadzeniu odpowiednich modyfikacji do systemu operacyjnego. Pozostałych 10 wierszy (400 znaków) służy do wyświetlania danych oraz nazw pól (tags). Maksymalna długość zapisu może przekroczyć 400 znaków, ponieważ zapis dłuższy wyświetlany jest partiami na monitorze na zasadzie przesuwania obrazu: gdy górny wiersz znika, pozostałe przesuwają się do góry i pojawia się wiersz następny. W specjalnych przypadkach cały ekran może być zajęty na wyświetlenie listy instrukcji operowania (tzw. lista HELP) dla operatorki lub "supervisora" w celu podania czynności, które powinny być wykonane. Pozwala to prowadzić swoistą konwersację operatorek lub "supervisora" z systemem metodą pytań i odpowiedzi. Lista HELP jest bardzo istotnym udogodnieniem, zarówno z punktu widzenia przyuczenia operatorek, jak również w czasie normalnej pracy, gdy operatorka zapomni kolejności wykonywanych czynności. Nie musi w takim przypadku szukać pomocy poza systemem. W czasie wprowadzania danych bardzo pomocne są następujące możliwości monitora ekranowego: wskaźnik (cursor) następnej pozycji znaku, który ma być wprowadzony, ograniczniki pola, które jest aktualnie wypełniane. Elementy te są pódane schematycznie na rys. 5. Innym istotnym udogodnieniem jest możliwość cofnięcia wskaźnika o kilka znaków, pól lub zapisów do tyłu w celu skorygowania lub zbadania danych bez wymazywania znaków pomijanych. Po wykonaniu niezbędnych czynności wskaźnik wraca na odpowiednie miejsce w zapisie, co umożliwia natychmiastową kontynuację pracy.



Rys. 5. Przykład formatu monitora ekranowego

4.4. Opis programu kontrolnego

Program kontrolny (record format) jest dokładnym opisem sposobu przygotowania[1] poprawnego zapisu danych. Występuje tutaj pewna analogia do dziurkarek kart. Karta perforowana ma swój stały format w postaci nadruku pól. Operatorka zgodnie z instrukcją perforacji musi ustawić maszynę przed rozpoczęciem pracy: "nastaw stały" (reprodukcja pól), przeskoki itp. Program kontrolny określa dla każdego dokumentu źródłowego podział zapisu na pola, podaje nazwy i długość tych pól oraz pozycję poszczególnych pól w zapisie. Dla każdego pola podaje się dodatkowo dokładną charakterystykę i sposób jego kontroli przez system operacyjny oraz sposób ręcznej weryfikacji. Do jednego zbioru może być wprowadzanych do 9 różnych formatów zapisów. Liczba programów kontrolnych przechowywanych na dysku w systemie operacyjnym nie jest ograniczona. Program kontrolny zawiera deklaracje funkcji edycyjnych oraz procedur automatycznego wykrywania błędów w danych. Do funkcji edycyjnych (editing) należy automatyczne manipulowanie (handling) polami danych (przy jednoczesnym włączeniu klawisza sterującego AUTO): reprodukcja pól, przeskoki, rozmnażanie pól, uzupełnianie (powiększanie) zawartości pól o stałą wielkość. Dla pól zmiennej długości można zadeklarować tzw. automatyczne przesuwanie w prawo, co pozwala na wprowadzanie danych od pierwszej kolumny pól, ponieważ system operacyjny ustawi odpowiednio zawartość pola w stosunku do prawego ogranicznika. Kolumny z lewej strony zostaną po przesunięciu wypełnione zerami, spacjami lub "blankami". Dla wielu pól można zadeklarować kontrolę przepełnienia, która nie pozwoli na wprowadzenie większej liczby znaków od zadeklarowanej.

Dla pól o stałej długości można określić kontrolę wypełnienia wszystkich kolumn. Pola, które muszą być wypełnione podczas wprowadzania danych będą kontrolowane i przynajmniej jeden znak musi być do nich wprowadzony. Automatyczne wykrywanie błędów (validity checks) pozwala na kontrolę poprawnego wypełnienia dokumentów źródłowych oraz poprawnego wprowadzenia danych przez operatorkę. Dla każdego pola można zadeklarować w programie kontrolnym procedury wykrywania błędów. Procedury te

---

[1] Pod pojęciem "przygotowanie danych" rozumiem wprowadzanie (rejestrowanie) danych, automatyczne wykrywanie błędów w danych, ręczną weryfikację danych i korektę błędów.

wykonywane są jednocześnie z wprowadzaniem danych. Wykrycie błędu powoduje blokadę klawiatury, rozlega się sygnał dźwiękowy, a odpowiednia nazwa błędu wyświetlana jest na monitorze ekranowym. Lista procedur kontroli danych przedstawia się następująco:

- numeryczna lub alfabetyczna "czystość" znaków w polu;
- zakres wartości danego pola, np. maks. 80 godzin nadliczbowych powinno wystąpić w dokumencie źródłowym;
- zawartość pola, np. czas nominalny może wynosić jedynie 200 godz.;
- wzrastające wartości pola na kolejnych dokumentach, np. numery uporządkowanych kwitów;
- cyfry (znaki) kontrolne;
- sumy kontrolne (do 20 pól) - uzgadnianie odbywa się automatycznie, tzn. sumy sporządzone ręcznie są wprowadzane do systemu jako zapis pierwszy w paczce dokumentów źródłowych;
- tablice wartości pozwalają na porównywanie wybranych pól, np. nr magazynu z dopuszczalnymi symbolami podanymi w tablicy.

Tablice "negatywne" zawierają wartości stałe niedopuszczalne, a tablice "pozytywne" zawierają wartości dopuszczalne. Opisane sposoby kontroli zapewniają znaczne zwiększenie poprawności rejestrowanych danych, ponieważ wykrywają zarówno błędy dokumentów źródłowych, jak i błędy operatorek. Błędy operatorek mogą być usunięte natychmiast.

4.5. Opis funkcji wprowadzania

Ustawienie stanowiska w modusie operacyjnym ZAPIS (Entry) pozwala na wykonywanie podstawowej funkcji systemu SEECHECK. Pierwszą czynnością jest wprowadzenie nazwy (symbolu) paczki odpowiadającej plikowi dokumentów źródłowych oraz symbolu identyfikacyjnego operatorki. Następną czynnością jest przywołanie z pamięci dyskowej do pamięci operacyjnej odpowiedniego programu kontrolnego przygotowanego dla danego rodzaju dokumentów źródłowych. System operacyjny wyświetli na monitorze nazwę pierwszego pola oraz ograniczniki pola. Wskaźnik ustawi się na pierwszej kolumnie pola. W tym momencie może rozpocząć się wprowadzanie danych do buforu w pamięci operacyjnej przez palcowanie (naciskanie odpowiednich klawiszy). Pola tworzące tzw. nastaw stały (reprodukowane) wymagają natychmiastowego sprawdzenia podczas tworzenia pierwszego zapisu. W tym

celu system operacyjny przełączy automatycznie stanowisko na modus operacyjny SPRAWDZANIE (Verify). Pola te po sprawdzeniu będą automatycznie reprodukowane we wszystkich dalszych zapisach w danej paczce. Inny wariant postępowania musi być przyjęty w przypadku stosowania kontroli za pomocą tzw. sum kontrolnych. W tym przypadku pierwszym zapisem wprowadzanym są sumy kontrolne sporządzone ręcznie przez kontrolerów dokumentów. Sumy te wprowadzone są do odpowiednich akumulatorów. Wartości zadeklarowanych pól będą w nich sukcesywnie odejmowane po wprowadzeniu kolejnych zapisów. Po wprowadzeniu ostatniego zapisu w danej paczce oraz po jej "zamknięciu" następuje sprawdzenie, czy zawartość akumulatorów jest zerowa. Jeżeli nie jest, zapisy paczki muszą być wydrukowane na drukarce lub wyświetlone (zbadane) na monitorach ekranowych przez kontrolerów dokumentów, ponieważ istnieje prawdopodobieństwo podania błędnych sum kontrolnych. Bez stosowania sum kontrolnych dane z pierwszego dokumentu są wprowadzane znak po znaku. Po wypełnieniu pola następuje jego dokładne sprawdzenie zgodnie z procedurami zadeklarowanymi w programie kontrolnym. W przypadku wykrycia błędu następuje blokada klawiatury uniemożliwiająca operatorkom wprowadzanie dalszych pól do czasu bądź skorygowania popełnionego przez siebie błędu, bądź ustawienia w odpowiednim polu tzw. znaku błędu (error flag) - jeżeli błąd dotyczy dokumentu źródłowego. Zapisy posiadające znak błędu muszą być skorygowane przed zapisaniem na taśmie magnetycznej, ponieważ system operacyjny nie pozwoli na ich zapisanie. Software systemu SEECHECK pozwala na wyszukanie i wydrukowanie lub wyświetlenie wszystkich zapisów ze znakiem błędu w celu wyjaśnienia i skorygowania przez kontrolerów dokumentów. Jeśli pole zostanie wprowadzone poprawnie, na ekranie zostaną wyświetlone nazwa oraz ograniczniki następnego pola. Następne pole może być wyświetlone w tym samym lub kolejnym wierszu. Po zakończeniu wprowadzania wszystkich pól dla danego zapisu następuje automatyczne przeniesienie zawartości buforu w pamięci operacyjnej jednostki centralnej do pamięci dyskowej. W przypadku konieczności korygowania aktualnie wprowadzonego zapisu lub zapisu przeniesionego do pamięci dyskowej operatorka może cofnąć się o kilka znaków pól lub rekordów, skorygować znaki poprzednio wprowadzone i wrócić na poprzednie miejsce.

4.6. Opis funkcji sprawdzania

Każde stanowisko przygotowania danych może być ustawione w modusie operacyjnym SPRAWDZANIE (Verify). Ręcznemu sprawdzaniu podlegają jedynie niektóre pola zadeklarowane w programie kontrolnym. Część pól nie musi być sprawdzana ręcznie, ponieważ są one sprawdzone przez program kontrolny za pomocą: sum kontrolnych, cyfry kontrolnej, badania zakresu itd. Niektóre pola mogą być sprawdzane wizualnie, np. nazwisko i imię pracownika w przypadku wprowadzania danych do kartoteki osobowej. Sprawdzanie ręczne (klawiszowe) rozpoczyna się - podobnie jak wprowadzanie danych - od wprowadzenia nazwy paczki sprawdzanych danych, podania symbolu identyfikującego operatorkę oraz przywołania symbolu odpowiedniego programu kontrolnego. Nazwa i ograniczniki pierwszego sprawdzanego pola zostaną wyświetlone na ekranie. Operatorka wprowadza ponownie odpowiednie dane znak po znaku, które są porównywane przez system operacyjny ze znakami poprzednio wprowadzonymi (przeniesionymi z dysku do buforu w pamięci operacyjnej). W przypadku niezgodności następuje blokada klawiatury a nazwa błędu wyświetlana jest na ekranie. Operatorka wyjaśnia, czy nacisnęła klawisz zgodnie z treścią dokumentu. O ile występuje niezgodność, koryguje swój błąd przez cofnięcie się o jeden znak i wprowadzenie poprawnego znaku. W innym przypadku powoduje wyświetlenie znaku zapisanego w pamięci operacyjnej i koryguje ten znak.

5. Zalety systemów typu "key-to-disc"

Zarówno system SEECHECK, jak i inne wielostanowiskowe systemy przygotowania danych tworzą zintegrowany układ urządzeń pozwalających na realizowanie wielu funkcji. Najbardziej istotnymi funkcjami są:

- wprowadzanie danych z dokumentów źródłowych,
- ręczna i automatyczna kontrola poprawności wprowadzonych danych oraz
- korekta błędów.

Wszystkie wymienione funkcje mogą być wykonywane za pomocą tego samego, uniwersalnego stanowiska przygotowania danych (zanik podziału na "dziurkarki" i "sprawdzarki"). Pozwala to na bardziej intensywne wykorzystanie zarówno stanowisk, jak i operatorek, które wykonywać mogą

wszystkie funkcje. Funkcje klawiatury wykonywane są szybciej, ponieważ wyeliminowana jest mechaniczna bezwładność urządzeń. Klawiatura pracuje bezgłośnie i niezawodnie. Błędy są błyskawicznie sygnalizowane. Automatyczne sterowanie wieloma funkcjami eliminuje znacznie liczbę uderzeń (palcowanie) klawiszy, co umożliwia osiągnięcie znacznie wyższej wydajności pracy operatorek: wg doświadczeń amerykańskich [1] najlepsza operatorka dziurkarki kart osiąga 16.000 uderzeń na godzinę; za pomocą systemu "key-to-disc" osiąga 22.000 uderzeń na godzinę; wzrost o około 37% przy znacznie mniejszej liczbie przepuszczonych błędów operatorek. Dzięki automatycznemu wykrywaniu błędów można znacznie skrócić proces ręcznego sprawdzania. Mając dokładne raporty sporządzane przez system o ilości i jakości pracy operatorek można na nie oddziaływać za pomocą precyzyjnie ustawionych systemów płac i premii.

Większość błędów występujących w dokumentach źródłowych oraz błędów popełnianych przez operatorki jest wykrytych na etapie przygotowania danych. Powoduje to skrócenie cyklu przygotowania danych. Szereg programów kontroli danych wykonywanych dotychczas za pomocą komputera głównego może być wyeliminowanych, ponieważ przygotowana jest "czysta" taśma magnetyczna. Oszczędności czasu komputera głównego mogą wynosić około 30% w stosunku do przygotowania danych na kartach dziurkowanych. Znaczne skrócenie cyklu przygotowania danych pozwala na skrócenie całego cyklu niezbędnego do uzyskania zestawień wynikowych. System SEECHECK może być również stosowany do wstępnego przetwarzania danych. Standardowy system operacyjny pozwala na sortowanie danych na dysku magnetycznym oraz na wykonywanie działań matematycznych. Istnieje również możliwość zmiany formatu danych na wyjściu: przed zapisaniem na taśmie magnetycznej lub wydrukowaniem na drukarce wierszowej. Możliwości te pozwalają zastosować system SEECHECK do systemów tworzenia (emisji) dokumentacji technologicznej i produkcyjnej oraz jako system zdecentralizowanego zbierania danych (za pomocą stanowisk połączonych z systemem liniami transmisji danych). System może stanowić również stację transmisji danych do oddalonego komputera głównego. Zastosowanie systemu SEECHECK możliwe jest nie tylko w dużych ośrodkach przetwarzania danych do scentralizowanego przygotowania danych, ale również w komórkach przygotowania produkcji emitujących dokumentację lub w magazynach centralnych do zdecentralizowanego zbierania danych z poszczególnych magazynów branżowych. Nowoczesne systemy typu "key-to-disc" nie wymagają klimatyzacji, mogą więc pracować w normalnych

warunkach. Powinny się nimi zainteresować szczególnie przedsiębiorstwa nie posiadające dotychczas żadnych urządzeń do zbierania, przygotowania, przetwarzania i transmisji danych.

Literatura

[1] Converting to key-to-disc for Data Entry. Data Processing Mag. 1971, nr 9.

[2] SUKIENNIK J.: Key Edit - zintegrowany system przygotowania danych. Informatyka 1973, nr 4.

[3] WALCZAK T.: Unowocześnianie wprowadzania danych. Informatyka 1972, nr 11.

[4] The SEECHECK Manual. RDS-261.

[5] Supervisors Reference Manual. RDS.

[6] SEECHECK input/output Operations. RDS.

[7] SEECHECK Formatting Techniques. RDS.

[8] KEYEDIT Supervisor Operating. ICL.

[9] KEYEDIT Keystation Operating. ICL.

[10] SYSTEM 2400 General Description. MDS.

[11] An Introduction to the IBM 3270 Information Display System. IBM.

[12] Data Preparation - Machines and Techniques. Data Processing, 1972, nr 5/6.

[13] Boom v periferních - rychlý vzestup trhu periferních zařízeni. Mechanizace a automatyzace administrativy, 1972, nr 8.

[14] Computer Controlled Data Preparation. Data Processing, 1970, nr 7/8.

[15] Wprowadzanie danych. Europejski Program Badawczy Diebolda, z. 35, Warszawa 1972, OBRI.

ETO NOWOŚCI Nr 2/1973

Mgr Maria ŁĄCKA
Mgr Jerzy SWIANIEWICZ

Instytut Maszyn Matematycznych

681.322.63/.64

STRUKTURA PROGRAMÓW ZŁOŻONYCH W SYSTEMACH IBM OS I DOS

Opracowanie jest fragmentem bardziej wszechstronnego opracowania mającego na celu porównanie dwóch systemów operacyjnych maszyn IBM/360, a mianowicie systemów OS (Operating System)i DOS (Disk Operating System). Ogólnie wiadomo, że system OS/360 jest znacznie bardziej rozbudowany i uniwersalny od systemu DOS/360. System DOS natomiast stawia znacznie mniejsze wymagania co do zestawu maszyny. Może on być stosowany na maszynach z mniejszą pamięcią operacyjną, poczynając już od 16k bytów, podczas gdy system OS w swojej najprymitywniejszej wersji wymaga co najmniej 32k[1]. Również mniejsze wymagania dotyczą wyposażenia maszyny w urządzenia zewnętrzne.

Jest zatem rzeczą pożyteczną uzmysłowienie sobie jakie nowe cechy systemu uzyskujemy decydując się na bardziej rozbudowany, ale zarazem kosztowniejszy system operacyjny OS/360.

Zajmiemy się tutaj jednym aspektem powyższego zagadnienia. Omówimy mianowicie strukturę programów złożonych w obydwu systemach.

Czytelnika należy uprzedzić, że praca ta nie jest wyczerpującym wykładem zasad konstruowania programów złożonych w obu omawianych systemach. Szczegółowe informacje na ten temat znajdują się w dokumentach firmy IBM wymienionych w literaturze w zakończeniu pracy. Zamierzeniem autorów było syntetyczne omówienie środków dostarczanych przez systemy OS i DOS w omawianej dziedzinie oraz pewne naświetlenie ich celowości oraz określenie głównych idei jakimi kierowali się projektanci systemów operacyjnych

---

[1] Podane wymiary pamięci są zresztą bardzo zaniżone. Praktycznie, winny one wynosić 32k dla DOS i 128k dla OS.

w IBM. Dlatego też, praca może zainteresować zarówno osoby przystępujące do studiowania literatury IBM jak również te, które z tą literaturą zawarli już wstępną znajomość.

1. Ogólna problematyka projektowania programów złożonych

Stworzenie środków ułatwiających konstruowanie dużych programów stanowiących nieraz całe systemy, jest jednym z podstawowych zadań systemów operacyjnych. Jest to zresztą problem, który staje również przed twórcami systemu operacyjnego w związku z opracowywaniem tak złożonego programu jakim jest sam system operacyjny.

Podstawową cechą takich programów-systemów jest ich modułowa struktura. Poszczególne moduły są zwykle niezależnymi programami spełniającymi pewne określone funkcje. Powinny one stanowić pewnego rodzaju elementy standardowe, dające się łatwo wymieniać i łatwo podłączać do innych systemów. Pełną elastyczność w operowaniu modułami, bądź całymi systemami modułów, uzyskuje się w systemach IBM/360 za pomocą standaryzacji formy i treści.

Standaryzacja formy - polega na tym, że każdy program, w wyniku przetworzenia go przez odpowiedni translator, zostaje sprowadzony do pewnej standardowej formy, której struktura nie zależy od postaci źródłowej (języka programowania) tego programu. Jest to żądanie nałożone przez system na translatory.

Standaryzacja treści - polega na żądaniu przestrzegania pewnych wymagań, odnoszących się do wewnętrznej konstrukcji programów. Mamy tu na myśli pewne konwencje dotyczące przekazywania sterowania między programami, przekazywania danych i wyników między programem nadrzędnym (głównym) i podrzędnym (podprogramem), przechowywania stanu rejestrów itp.

Standaryzację treści w przypadku programowania w języku maszyny Assembler realizuje programista. W przypadku języków wyższych - jest to zadanie translatora. Podstawowym celem, który ma być osiągnięty przez stosowanie się do różnych reguł i konwencji jest, aby każdy program niezależnie od jego złożoności był standardowo zbudowanym podprogramem.

Proces przetwarzania programu źródłowego na program wykonywalny odbywa się, w obydwu omawianych systemach, w dwóch etapach. Formą pośrednią jest zestandaryzowana postać, na którą wszystkie translatory tłumaczą dostarczany im tekst źródłowy. Efektem działania translatora jest tzw. moduł wynikowy (object module), który na ogół nie jest samodzielnym programem. Jest to raczej cegiełka, która dopiero po połączeniu z innymi modułami wynikowymi utworzy element programu wykonywalnego. Nie ma istotnej różnicy między modułami wynikowymi w systemach OS i DOS. Moduły wynikowe mogą być włączane do biblioteki.

Z modułów wynikowych za pomocą programu łączącego (linkage editor), zgodnie z przepisem zapisanym w języku programu łączącego, buduje się elementy programu wykonywalnego.

Zasadnicza różnica między systemami OS i DOS w omawianej dziedzinie polega na różnym efekcie działania programu łączącego. W systemie DOS elementy programu wykonywalnego, tzw. fazy (phase) są przystosowane do wykonywania w określonym miejscu pamięci, podczas gdy w OS elementy programu wykonywalnego, tzw. moduły ładowalne (load modules) przystosowują się do działania w wyznaczonym miejscu pamięci już w trakcie wykonywania programu. Z tego więc wynika znacznie większa swoboda w konstruowaniu programów z elementów zawartych w bibliotece modułów ładowalnych w systemie OS.

W dalszych częściach pracy zasady konstruowania programów w obydwu systemach będą omówione bardziej szczegółowo.

2. Faza a moduł ładowalny

Organizacja maszyny IBM/360 umożliwia stosunkowo łatwe pisanie w języku maszyny programów, które w postaci binarnej są niezależne od położenia w pamięci. Przyczyniają się do tego następujące cechy tej maszyny:

a) struktura części adresowej rozkazu w postaci baza/przesunięcie; każdy adres pamięci operacyjnej wyliczany jest dynamicznie podczas realizacji rozkazu, na podstawie zawartości wskazanego rejestru bazy oraz przesunięcia wyszczególnionego w samym rozkazie;

b) rozkazy pozwalające załadować rejestr bazy adresem miejsca pamięci operacyjnej, w którym ten rozkaz się znajduje.

Istnieją jednak sytuacje kiedy zachodzi potrzeba umieszczenia w programie adresów pewnych wyróżnionych miejsc tego programu tzw. stałych adresowych (address constants). Wartości stałych adresowych mogą być określone dopiero po powzięciu decyzji w jakim obszarze pamięci dany program będzie wykonywany.

Liczne różnice między systemami operacyjnymi OS i DOS wynikają z faktu, że w obu systemach moment powzięcia wyżej wymienionej decyzji jest różny. W systemie DOS decyzja ta musi być powzięta podczas przygotowywania programu do wykonania. Przekazuje się ją programowi łączącemu za pomocą zdań języka tego programu. Program łączący przystosowuje program do działania w określonym obszarze pamięci. Tak spreparowany program może być od razu wykonywany bądź umieszczony w bibliotece programów przygotowanych do wykonania w tzw. bibliotece faz (core image library). Elementami biblioteki faz są fazy. Z każdą fazą związany jest opis zawierający: nazwę przypisaną tej fazie, adres pamięci operacyjnej, od którego należy umieszczać fazę w celu jej wykonania, punkt wejścia - adres pierwszego wykonywanego rozkazu fazy, parametry określające położenie fazy w bibliotece faz. Fazy włączone do biblioteki sprowadzane są do pamięci operacyjn j za pomocą systemowego programu ładującego (system loader) wchodzącego w skład programu sterującego w DOS. Faza może być wywołana do wykonania przez podanie jej nazwy w czołówce jako parametru zdania określającego etap pracy (job step) tzw. zdania EXEC. Program działający może przekazać sterowanie innej fazie przez odwołanie się do programu sterującego za pomocą makrorozkazu FETCH.

W systemie OS decyzja o przystosowaniu programu do określonego obszaru pamięci operacyjnej odsunięta jest do momentu pobrania tego programu w celu jego wykonania.Elementy biblioteki programów przygotowanych do wykonania, tj. moduły ładowalne, zawierają oprócz treści programu tzw. listę RLD (Relocation List Dictionary), na której opisane są elementy programu wymagające przeadresowania. Lista ta wykorzystywana jest przez program ładujący (program fetch) będący częścią programu sterującego w OS.

W dalszym ciągu omówimy dokładniej ogólne zasady oraz środki przeznaczone do realizowania programów złożonych w systemie OS, a następnie opiszemy analogiczne środki dostarczane przez DOS.

3. Dynamiczna struktura programu złożonego w systemie OS

Przesunięcie momentu ostatecznego przystosowania programu do określonego obszaru pamięci do momentu wykonywania programu umożliwiło (w systemie OS) przekazanie funkcji przydzielania programom miejsca w pamięci operacyjnej programowi sterującemu. Umożliwiło to zrealizowanie opisanej poniżej dynamicznej struktury programu złożonego.

Program taki składa się z pewnej liczby modułów ładowalnych, które są na ogół samodzielnymi fragmentami programu. Moduły ładowalne muszą być elementami biblioteki i są identyfikowane za pomocą nazw. Oprócz nazwy z modułem ładowalnym związanych jest wiele atrybutów, o których będzie jeszcze mowa dalej.

Pierwszy moduł programu wywoływany jest za pomocą zdania EXEC[1] określającego wykonywany program jako etap w ramach pracy (job). W czasie wykonywania programu moduł ładowalny może w dwojaki sposób przekazywać sterowanie innemu modułowi wskazując jego nazwę. Pierwszy sposób określimy jako "odwołanie się do podprogramu", drugi - "oddanie sterowania".

W pierwszym przypadku wywołujący moduł ładowalny przekazuje sterowanie podprogramowi (być może składającemu się z wielu modułów lub odwołującemu się do innych podprogramów), spodziewając się powrotu z tego podprogramu po wykonaniu przewidzianych dla niego czynności.

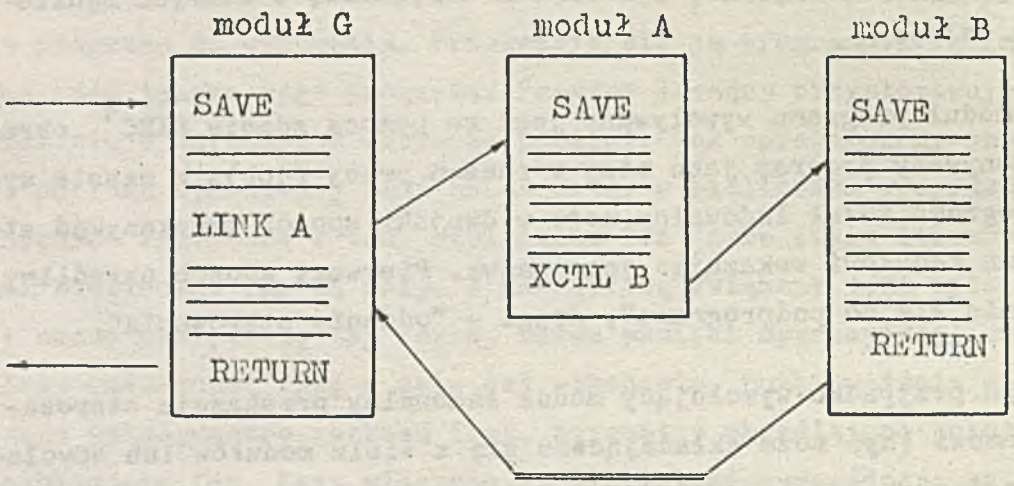
W drugim przypadku sterowanie zostaje oddane innemu modułowi kontynuującemu wykonywanie czynności zapoczątkowanych przez moduł wywołujący. Tego typu przekazanie sterowania zawiera supozycję, że wywołujący moduł przestaje być potrzebny w pamięci maszyny.

Obydwa typy przekazywania sterowania między modułami ładowalnymi realizowane są przez odwołanie się do programu sterującego za pomocą odpowiedniego rozkazu wywołania supervisora (SVC - z odpowiednim parametrem). Ze względu na konieczność przekazywania supervisorowi wielu parametrów, tego typu odwołania przyjmują postać pewnych standardowych sekwencji rozkazowych, dla których w języku Macro-Assemblera zostały zdefiniowane specjalne wieloparametrowe makrorozkazy. Makrorozkaz LINK realizuje prze-

[1] Zdanie EXEC jest jednym ze zdań języka Job Control Language służącego do opisywania prac przekazywanych maszynie do realizacji.

kazanie sterowania pierwszego rodzaju, zaś makrorozkaz XCTL - drugiego. Podstawowym parametrem powyższych makrorozkazów jest nazwa modułu ładowalnego, któremu przekazuje się sterowanie. Trzecim makrorozkazem występującym w powyżej opisanym modelu jest makrorozkaz RETURN używany do przekazania sterowania z powrotem do modułu wywołującego tj. tego, który ostatnio oddał sterowanie za pośrednictwem LINK.

Opisanie powiązania modułów ilustruje rys. 1. Występujący na wejściu do każdego modułu makrorozkaz SAVE realizuje standardowe czynności wywołania programu polegające w szczególności na przechowywaniu stanu rejestrów programu wywołującego w obszarze określonym przez tenże program.



Rys. 1. Powiązania modułów

Przyjmując powyższy model programista bierze na siebie minimum kłopotów związanych z zaplanowaniem struktury programu. Właściwie, jedynym chyba jego zadaniem w tej dziedzinie jest dokonanie podziału programu na moduły, co zresztą wynika z metodyki konstruowania programów złożonych i jest niezależne od maszyny, dla której taki program jest przygotowywany. Takie funkcje jak odnajdywanie odpowiednich modułów w bibliotece, przydzielanie im miejsca w pamięci, ładowanie i przekazywanie sterowania są funkcjami systemu (dokładniej supervisora).

4. Środki zwiększania efektywności czasowej programu złożonego w OS

Opisany w poprzednim paragrafie model programu złożonego wykorzystuje tylko nieznaczny procent udogodnień i środków proponowanych w systemie OS/360 programiście opracowującemu strukturę złożonego programu. Tezą

dalszych rozważań będzie stwierdzenie, że wszystkie pozostałe mechanizmy i środki mają służyć programiście do zrobienia swojego programu bardziej efektywnym.

Zauważmy przede wszystkim, że główne straty na efektywności programu mogą wyniknąć z następujących przyczyn:

- zbędnego sprowadzania modułu z biblioteki (znajdującej się w pamięci pomocniczej) do pamięci operacyjnej,
- zbędnych czynności organizacyjnych wykonywanych przez program sterujący (supervisor) przy przekazywaniu sterowania od modułu do modułu.

Poniżej omówimy środki, którymi dysponuje system OS dla unikania ww strat w efektywności programów.

4.1. Atrybut regeneracji modułów

Włączając moduł ładowalny do biblioteki programista może go określić jako:

- jednokrotny (non reusable),
- powtarzalny seryjnie (serially reusable),
- powtarzalny równolegle (reenterable).

Wymienione cechy są wartościami atrybutu regeneracji modułów. Zajmiemy się tutaj dwiema pierwszymi z tych cech (trzecia dotyczy tzw. programów równoległych, których nie będziemy tutaj rozważali). Moduł winien dostać cechę "jednokrotny" jeżeli z jego wykonaniem związane jest "samouszkodzenie" powodujące, że nie nadaje się on do powtórnego wykonania. Moduł taki, aby mógł być powtórnie wykonany, musi być ponownie sprowadzony z biblioteki do pamięci.

Moduł z cechą powtarzalności seryjnej musi się regenerować w pamięci, a więc może być wykonywany wielokrotnie po sprowadzeniu z biblioteki.

Jeżeli zatem programista zadba o to, aby opracowane przez niego moduły miały cechę powtarzalności seryjnej wtedy supervisor (który prowadzi rejestr modułów znajdujących się w pamięci) realizując funkcje LINK i

XCTL nie będzie sprowadzał z biblioteki modułów, które już znajdują się w pamięci.

4.2. Makrorozkazy LOAD i DELETE

Supervisor prowadząc rejestr modułów ładowalnych znajdujących się w pamięci oraz gospodarując pamięcią przez przydzielanie jej modułom nowo sprowadzanym z biblioteki i zwalnianie obszarów zajmowanych przez moduły wg niego już niepotrzebne - działa wg pewnego algorytmu allokacji (różnego zresztą w różnych wersjach systemu). Jak wiadomo, nie istnieje optymalny algorytm allokacji pamięci, w związku z czym może się zdarzyć, że zgodnie z algorytmem zostanie usunięty z pamięci moduł tuż przed jego ponownym wywołaniem. Takim niekorzystnym sytuacjom może zapobiec programista przez sprowadzanie modułów do pamięci "do odwołania". Takiej możliwości dostarczają makrorozkazy LOAD i DELETE, z których pierwszy każe supervisorowi sprowadzić do pamięci moduł (jeżeli jeszcze go tam nie ma) i oznaczyć jako "nieusuwalny", drugi - powoduje zlikwidowanie cechy nieusuwalności oraz usunięcie modułu z pamięci (ściślej zwolnienie pamięci przez niego zajmowanej). Bliższe szczegóły związane z funkcjonowaniem makrorozkazów LOAD i DELETE podane są w punkcie 4.4.

4.3. Makrorozkaz CALL

Funkcje LOAD i DELETE pozwalają jeszcze bardziej zwiększyć efektywność przekazywania sterowania między modułami. Mianowicie, po sprowadzeniu modułu do pamięci za pomocą odwołania typu LOAD mamy pewność, że moduł ten pozostanie w pamięci aż do zwolnienia go za pomocą makrorozkazu DELETE. Jednocześnie możemy przekazywać mu sterowanie bez pośrednictwa supervisora obciążającego program całym skomplikowanym mechanizmem gospodarki pamięcią operacyjną. Funkcję taką spełnia makrorozkaz CALL, który realizuje funkcje analogiczne do makrorozkazu LINK nie angażując przy tym supervisora.

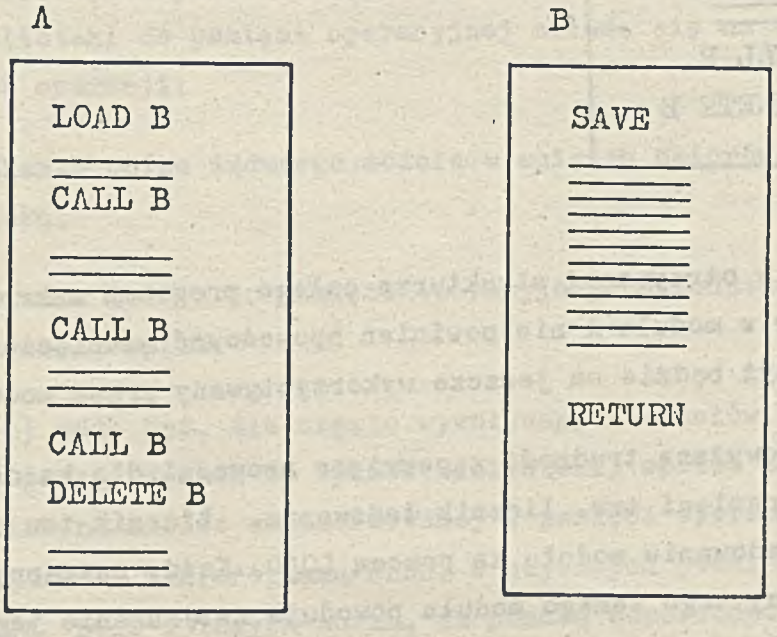
Interesujący jest fakt, że powrót do modułu wywołującego (niezależnie od tego czy wywołany moduł otrzymał sterowanie za pośrednictwem makrorozkazu LINK czy CALL) odbywa się za pomocą makrorozkazu RETURN.

A więc ta sama sekwencja rozkazów, odpowiadająca makrorozkazowi RETURN, w w przypadku wywołania typu LINK powoduje przekazanie sterowania supervisorowi (który musi być poinformowany o zakończeniu pracy modułu), zaś w przypadku wywołania typu CALL - bezpośrednio do modułu wywołującego. Sekręt polega na różnym ustawieniu adresu powrotu w przeznaczonym do tego rejestrze (zgodnie z pewną konwencją).

Ustalone ściśle konwencje dotyczące wykorzystania rejestrów, przekazywania parametrów do podprogramów oraz sposobu przechowywania i odnawiania stanu rejestrów powodują, że możliwa jest pełna niezależność programów (w szczególności modułów ładowalnych) od sposobu ich wywoływania. Podstawową zasadą przy tym jest, że każdy program powinien być tak napisany, aby mógł być wywołany jako podprogram przez inny program nadrzędny.

4.4. Licznik ładowania

Typowe zastosowanie makrorozkazów LOAD, DELETE i CALL ilustruje rys. 2:

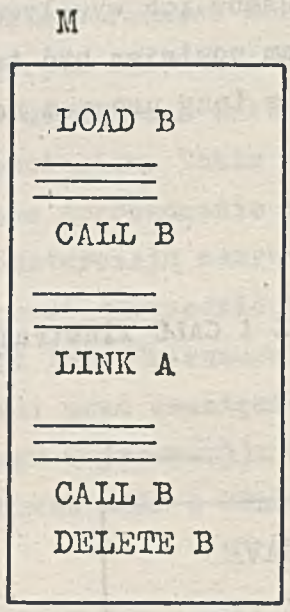


Rys.2. Zastosowanie makrorozkazów LOAD,DELETE i CALL

Makrorozkaz LOAD gwarantuje istnienie modułu B w pamięci, podając jego adres w jednym z rejestrów dzięki czemu można wielokrotnie odwoływać

się do niego za pośrednictwem makrorozkazu CALL. Po ostatnim makrorozkazie CALL makrorozkaz DELETE "zwalnia" moduł B (pozwala supervisorowi usunąć go z pamięci).

Stosując zasadę, że każdy program winien mieć strukturę podprogramu (zasada ta jest niezwykle istotna przy konstruowaniu dużych systemów programowych) możemy sobie wyobrazić, że program pokazany na rys. 2 został potraktowany jako podprogram programu nadrzędnego zawartego w module M. Przypuśćmy dalej, że program M korzysta również bezpośrednio z modułu B jako swego podprogramu. Możemy zatem otrzymać strukturę zilustrowaną na rys. 3.



Rys.3. Podwójne ładowanie i zwalnianie modułu

Zauważmy, że w otrzymanej strukturze całego programu makrorozkaz DELETE B zawarty w module A nie powinien spowodować usunięcia tego modułu z pamięci, gdyż będzie on jeszcze wykorzystywany przez moduł M.

Aby pokonać powyższą trudność supervisor prowadzi dla każdego modułu załadowanego do pamięci tzw. licznik ładowania. Licznik ten otrzymuje wartość 1 po załadowaniu modułu za pomocą LOAD. Każdy następny makrorozkaz LOAD dotyczący tego samego modułu powoduje zwiększanie jego licznika ładowania o 1 zaś każdy makrorozkaz DELETE powoduje zmniejszenie tego licznika o 1. Gdy w wyniku wykonania makrorozkazu DELETE licznik ładowania określonego modułu stanie się zerem to moduł ten zostaje usunięty z rejestru modułów załadowanych do pamięci i obszar pamięci przez niego zajmowany zostaje zwolniony.

Nieodzowność opisanego wyżej mechanizmu jest jeszcze bardziej oczywista w systemach pozwalających na pracę wielozadaniową, kiedy możliwe jest równoległe wykonywanie kilku zadań (sekwencji rozkazów) w ramach jednego programu.

4.5. Makrorozkaz BLDL

Makrorozkaz BLDL jest środkiem na przyspieszenie czynności sprowadzania modułu z biblioteki. Biblioteka modułów ładowalnych jest zbiorem danych (bądź zespołem kilku zbiorów danych) o dwupoziomowej strukturze tzw. zbiorów rozczłonowanych (partitioned data sets).

Zbiór o tej strukturze jest zespołem zbiorów sekwencyjnych zaopatrzonym w ich spis (directory). Te zbiory składowe będziemy nazywali podzbiorami (members) zbioru rozczłonowanego. Spis zbioru rozczłonowanego zawiera, oprócz nazw podzbiorów, ich adresy w pamięci pomocniczej oraz informacje opisujące własności tych podzbiorów.

Moduły ładowalne, wchodzące w skład biblioteki systemu, są podzbiorami zbiorów rozczłonowanych tworzących tę bibliotekę. Zatem sprowadzenie modułu z biblioteki do pamięci operacyjnej składa się na ogół z dwóch czasochłonnych operacji:

a) z odszukania opisu żądanego modułu w spisach zbiorów tworzących bibliotekę,

b) sprowadzenia modułu do pamięci operacyjnej na podstawie informacji uzyskanych ze spisu.

Operacja a) może być, dla często wywoływanych modułów, wyeliminowana przez wcześniejsze wybranie ze spisów biblioteki, opisów tych modułów i umieszczenie ich na liście skonstruowanej w pamięci operacyjnej. Dokonujemy tego za pomocą makrorozkazu BLDL. W używanych później makrorozkazach LINK, XCTL, LOAD i innych można, za pomocą odpowiedniego parametru, wskazywać pozycje listy utworzonej za pomocą BLDL, unikając czasochłonnego przeglądania spisów zawartych w pamięci pomocniczej (na dyskach).

5. Efektywność czasowa a wymagania przestrzenne

Dążenie do zwiększenia efektywności czasowej programów na ogół prowadzi do zwiększenia zapotrzebowania na pojemność pamięci operacyjnej wykorzystywanej przez program. W dalszych dwóch paragrafach omówione będą środki, które oferuje system OS dla złagodzenia konfliktu pojawiającego się przy równoczesnym dążeniu do zwiększania efektywności zarówno czasowej jak i przestrzennej programu.

5.1. Logiczna struktura programu

W systemie maszyny IBM/360 program, niezależnie od tego w jakim języku był pisany, składa się z pewnych jednostek logicznych. W języku Assemblera jednostkami tymi są sekcje kontrolne (control sections), w języku FORTRAN będą to program główny i podprogramy, w COBOL-u cały program. Po przetworzeniu programu za pomocą odpowiedniego translatora, wymienione wyżej jednostki logiczne programu, zostają, jak już wspomniano, zamienione na elementy o zestandaryzowanej, niezależnej od języka źródłowego, strukturze. Elementy te nazwiemy sekcjami wynikowymi[1]. Stosunek sekcji wynikowej do modułu wynikowego, o którym była mowa we wstępie, jest następujący: moduł wynikowy jest to zespół kilku sekcji wynikowych powstały jako rezultat jednego odwołania się do któregoś z translatorów. Moduł wynikowy jest jednostką biblioteczną tj. posiada własny identyfikator w ramach określonej biblioteki.

Sekcja wynikowa składa się z kodu wynikowego w postaci binarnej oraz dwóch list:

- słownika relokacji zawierającego spis stałych adresowych programu, które wymagają przeadresowania zależnego od położenia danej sekcji oraz sekcji z nią powiązanych,

- listy symboli komunikacyjnych, do których należą:
  - symbole zewnętrzne tj. zdefiniowane w innych sekcjach, a do których dana sekcja się odwołuje,

---

[1] W opisach IBM nie ma rozróżnienia terminologicznego między sekcją kontrolną w języku Assembler a sekcją wynikową. W obu przypadkach używany jest termin "control section". Sekcja wynikowa jest wewnętrzną postacią sekcji kontrolnej.

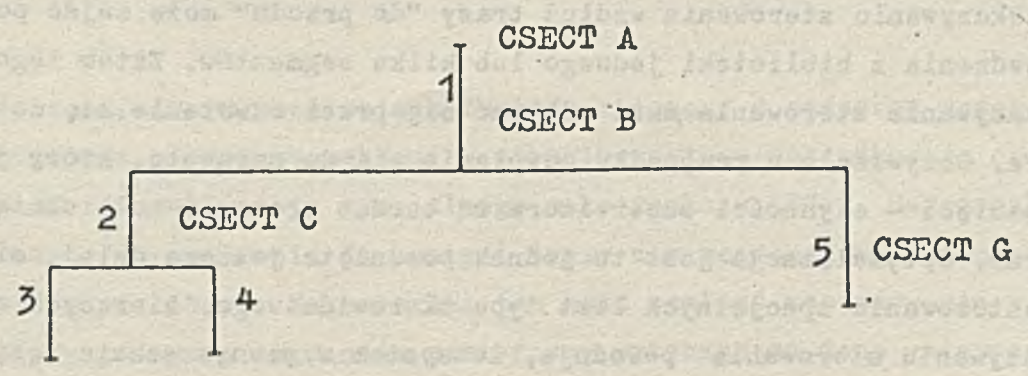
• symbole wejściowe (entry points) tzn. takie, które są zdefiniowane w danej sekcji ale mogą być używane w innych sekcjach.

Sekcje wynikowe są elementami, z których program łączący (linkage-editor) tworzy moduły ładowalne.

5.2. Moduły typu overlay

Dążąc do uzyskania maksymalnie efektywnego pod względem czasowym programu programista może, jako krańcowe rozwiązanie, zażądać od programu łączącego połączenia wszystkich sekcji wynikowych w jeden moduł ładowalny. Wtedy, cały program będzie się równocześnie znajdował w pamięci i przekazywanie sterowania między poszczególnymi sekcjami nie będzie wymagało żadnej ingerencji supervisora. Może się jednak okazać, że otrzymany w ten sposób moduł ładowalny jest za duży i bądź po prostu nie mieści się w pamięci maszyny, bądź przekracza pewne z góry narzucone ograniczenia dotyczące wymiarów. W takim przypadku wyjściem z sytuacji może być zmiana struktury modułu na tzw. strukturę typu nakładkowego (overlay). Jeżeli przekazywanie sterowania od sekcji do sekcji realizowane było zgodnie z pewnymi określonymi konwencjami (jak np. przez zastosowanie makrorozkazu CALL) to zmiana struktury modułu ładowalnego może być dokonana na etapie programu łączącego. Wystarczy zmienić program opisujący sposób łączenia sekcji nie dokonując żadnych lub bardzo nieliczne zmiany w samym programie łączonym.

W strukturze nakładkowej moduł składa się z pewnej liczby tzw. segmentów, które utworzone są z sekcji tworzących moduł. Segmenty te uporządkowane są wg pewnej struktury w postaci drzewa.



Rys.4. Moduł o strukturze nakładkowej

Rys. 4 pokazuje przykład takiej struktury. Segment 1 złożony z sekcji A, B stanowi tzw. segment główny (root segment). Segment ten musi znajdować się stale w pamięci podczas wykonywania programu zawartego w module. Segmenty 2 (sekcja C) i 5 (sekcja G) są segmentami wykluczającymi się - tzn. w pamięci może się znajdować jeden z nich, przy czym mają ten sam adres początku. Podobnie wykluczającymi się są nie tylko segmenty 3 i 4 (mające ten sam adres początkowy) ale również segmenty 3 i 5 oraz 4 i 5.

Wyrażając się niezbyt precyzyjnie - trasą (path) nazwiemy ciąg segmentów, przez które należy przejść na schemacie drzewa aby dojść do określonego segmentu. W naszym przykładzie (rys. 4) mamy następujące trasy:

1 - trasa segmentu głównego
1, 2 - trasa segmentu 2
1, 2, 3 - trasa segmentu 3
1, 2, 4 - trasa segmentu 4
1, 5 - trasa segmentu 5

W każdym momencie wykonywania modułu w pamięci znajduje się jedna z tras. Zasadniczo przekazywanie sterowania odbywa się wzdłuż tras. W przypadku przekazania sterowania do segmentu nie znajdującego się w danym momencie w pamięci - do pamięci zostają sprowadzone oprócz żądanego segmentu, wszystkie brakujące segmenty z jego trasy. Tak więc wywołanie segmentu 4 z segmentu 1 spowoduje (w razie potrzeby) sprowadzenie do pamięci również segmentu 2.

W strukturze nakładkowej przy zmniejszaniu zapotrzebowania na pamięć operacyjną można osiągnąć bardzo dobrą efektywność czasową programu. Straty związane z funkcjami supervisora sprowadzone są tutaj do minimum. Przy przekazywaniu sterowania wzdłuż trasy "do przodu" może zajść potrzeba sprowadzenia z biblioteki jednego lub kilku segmentów. Zatem tego typu przekazywanie sterowania musi odbywać się przez odwołanie się do supervisora. Oczywiście w przypadku odwołania się do segmentu, który już jest w pamięci - czynności supervisora są bardzo proste i mało obciążają program. Optymalizacja jest tu jednak posunięta jeszcze dalej, mianowicie zastosowanie specjalnych list typu skorowidzowego, biorących udział w przekazywaniu sterowania powoduje, że system w pewnym sensie uczy się. W efekcie każde następne przejście między tymi samymi punktami w różnych

segmentach danej trasy (aż do momentu usunięcia segmentu późniejszego) odbywa się już bez udziału supervisora. Jest to szczególnie istotne w przypadku występowania pętli obejmujących kilka segmentów w jednej trasie. Odwołanie się "wstecz" po trasie zawsze odbywa się bez udziału supervisora, zawsze bowiem segmenty poprzedzające dany segment muszą znajdować się w pamięci.

Powyższe fakty możliwe są dzięki złożonym czynnościom programu łączącego, realizowanym podczas formowania modułu o strukturze nakładkowej. Do jego zadań należy m.in.

- skonstruowanie odpowiedniej tablicy segmentów (umieszczonej na początku modułu ładowalnego) w której dynamicznie rejestrowane są informacje o stanie poszczególnych segmentów;

- przeanalizowanie wszystkich przejść między sekcjami tworzącymi moduł i przetwarzanie tych, które mogą zawierać w sobie żądanie sprowadzenia segmentu (przejścia do przodu wzdłuż trasy) tak aby przechodziły przez odpowiednią tablicę wejść do segmentów.

Powyższe tablice wejść zostają umieszczone na końcu segmentów stanowiących węzły struktury drzewowej. Są one wykorzystywane przez supervisor w trakcie wykonywania programu. Prawidłowe skonstruowanie takich tablic jest możliwe pod warunkiem ścisłego przestrzegania przez programistę pewnych zasad przekazywania sterowania między sekcjami. W przypadku dostosowania się do tych przepisów - w listach związanych z każdą sekcją kontrolną zostaną zarejestrowane wszystkie punkty wejściowe do danej sekcji jak również miejsca w sekcji, w których znajdują się stałe adresowe określające wejścia do innych sekcji - informacje niezbędne dla właściwego skonstruowania przez program łączący odpowiednich tablic wejść.

Część supervisora realizująca funkcje związane z nadzorem przekazywania sterowania między segmentami w strukturze typu nakładkowej, obsługuje przy okazji realizację specjalnego makrorozkazu SEGWT. Makrorozkaz ten powoduje sprowadzenie do pamięci trasy dla segmentu wskazanego w parametrze makrorozkazu. Makrorozkaz ten może znaleźć zastosowanie w przypadku gdy w jednym z segmentów zamierzamy wykorzystać dane umieszczone w segmencie, o którym nie mamy pewności czy znajduje się w pamięci. Sto-

sowanie jednak tego makrorozkazu implikuje stosowanie struktury nakładkowej już podczas pisania programu. Jeżeli zatem chcemy odłożyć decyzję o wyborze struktury programu do momentu formowania modułu ładowalnego - musimy zrezygnować zarówno z tego makrorozkazu jak i wspomnianej techniki dowolnego rozmieszczania danych w segmentach.

W rzeczywistości wiele innych ograniczeń dla piszącego program pojawia się przy próbie napisania takiego programu, który dawałby się formować w dowolne struktury. Są one na ogół oczywiste i łatwe do spełnienia pod warunkiem znajomości techniki przekazywania sterowania w strukturze nakładkowej. Przy okazji można wysunąć przypuszczenie, że często złożone techniki programowania dają się prosto wyjaśnić przez ujawnienie ich realizacji.

6. Proces konstruowania programu złożonego w OS. Podsumowanie

Napisane w różnych językach programowania fragmenty opracowywanego programu zostają przetworzone za pomocą odpowiednich translatorów na tzw. sekcje wynikowe. Sekcje te niezależnie od tego w jakim języku był napisany program źródłowy, mają jednolitą strukturę. Są one elementami, z których budowane są tzw. moduły ładowalne. Operacji łączenia sekcji wynikowych w moduły ładowalne dokonuje program łączący zgodnie ze wskazówkami dostarczonymi przez programistę. Moduły ładowalne zostają włączone do biblioteki i stamtąd mogą być wywoływane wg nadanych im nazw bibliotecznych. Możliwe jest dynamiczne przekazywanie sterowania między modułami za pomocą makrorozkazów LINK, XCTL bądź sprowadzanie ich do pamięci za pomocą makrorozkazu LOAD. Ponieważ technika przekazywania sterowania między modułami ładowalnymi jest różna od techniki przekazywania sterowania między sekcjami zawartymi w tym samym module, podział sekcji między moduły ładowalne musi być uwzględniony już podczas pisania programu źródłowego.

Z punktu widzenia budowy wewnętrznej moduł ładowalny może mieć strukturę prostą bądź typu "overlay". W strukturze prostej moduł jest w całości sprowadzany do pamięci operacyjnej. Wszystkie sekcje tworzące moduł równocześnie znajdują się w pamięci w trakcie wykonywania programu.

W strukturze "overlay" sekcje tworzące moduł ładowalny pogrupowane są w zespoły zwane segmentami. Wyróżniony jest segment główny, który w trakcie wykonywania modułu stale znajduje się w pamięci. Pozostałe segmenty tworzą strukturę, w której wyróżnione pary segmentów wzajemnie się wykluczają i są w razie potrzeby sprowadzane do pamięci na to samo miejsce. Celowość struktury "overlay" polega na oszczędności pamięci przy stosunkowo niedużym zmniejszeniu efektywności. Ponieważ przy ścisłym stosowaniu się do ustalonych konwencji pisania programów, sposób przekazywania sterowania między sekcjami będzie z punktu widzenia programisty identyczny w obu strukturach - decyzja o wyborze struktury może być podjęta już po napisaniu programu.

7. Dynamiczna struktura programu złożonego w DOS

Jak już wiadomo, program złożony w systemie DOS składa się z faz umieszczanych w bibliotece faz. Fazy identyfikowane są za pomocą nadanych im nazw. Wprowadzając fazę do biblioteki można jej przypisać atrybut przemieszczalności. Oznacza to, że program tworzący daną fazę może być wykonywany w dowolnym obszarze pamięci operacyjnej. Inaczej mówiąc taki program albo nie zawiera przemieszczalnych stałych adresowych albo też w jego skład wchodzi sekwencja rozkazów dynamicznie ustawiająca stosowane przemieszczalne stałe adresowe. Fazie nie mającej cechy przemieszczalności zostaje przyporządkowany adres początkowy tj. absolutny adres pamięci operacyjnej, od którego poczynając powinna być ona umieszczona w celu wykonania. Według tego adresu właśnie są ustalane wartości przemieszczalnych stałych adresowych występujących w programie.

Realizacja programu złożonego rozpoczyna się od sprowadzenia fazy startowej programu do pamięci operacyjnej i zainicjowania jej wykonywania. Nazwę tej fazy podaje się w zdaniu EXEC określającym etap (job step) w ramach wykonywanej pracy (job). Faza startowa jest ładowana od adresu początkowego w przypadku, gdy faza nie posiada cechy przemieszczalności bądź od pierwszego dostępnego dla programu adresu obszaru pamięci (background, foreground 2, foreground 1), w którym ma być wykonywana.

Poniżej podajemy sposoby przekazywania sterowania z fazy do fazy, w systemie DOS określone w p. 3.

Oddanie sterowania innej fazie jest realizowane za pomocą makrorozkazu FETCH, odwołującego się za pośrednictwem supervisora do systemowego programu ładującego, który na podstawie nazwy określonej w parametrach makrorozkazu wyszukuje fazę w bibliotece faz. W przypadku, gdy faza nie ma przypisanej cechy przemieszczalności, to jest ona ładowana do pamięci operacyjnej od adresu ustalonego podczas działania programu łączącego, inaczej - od adresu określonego w parametrach FETCH. Sterowanie zostaje przekazane zgodnie z wartością punktu wejścia podaną w parametrach FETCH (parametr opcjonalny) albo w bibliotecznym opisie fazy.

Przekazanie sterowania typu "odwołania się do podprogramu" można realizować tylko między programami znajdującymi się jednocześnie w pamięci operacyjnej. Wskutek tego zaszła potrzeba zdefiniowania makrorozkazu, którego zadaniem byłoby sprowadzenie wskazanej fazy z biblioteki programów wykonywalnych do pamięci. Makrorozkazem tym jest LOAD, odwołuje się on, tak jak FETCH, do systemowego programu ładującego w celu wyszukania fazy o nazwie wymienionej w parametrach LOAD, w bibliotece faz i sprowadzenia jej do pamięci operacyjnej. Adres, od którego ładuje się fazę do pamięci jest określany w parametrach makrorozkazu LOAD (parametr opcjonalny) albo w bibliotecznym opisie fazy. Po sprowadzeniu fazy do pamięci sterowanie zostaje zwrócone fazie, w której zastosowano LOAD. Cecha przemieszczalności przypisana fazie nie ma żadnego wpływu na wykonanie tego makrorozkazu.

Przekazywanie sterowania typu "odwołania się do podprogramu" między programami umieszczonymi jednocześnie w pamięci operacyjnej jest organizowane za pomocą makrorozkazów CALL, SAVE i RETURN. Dla programów o strukturze podprogramu pierwszą wykonywaną sekwencją rozkazów maszyny musi być sekwencja odpowiadająca makrorozkazowi SAVE, a ostatnią - sekwencja odpowiadająca makrorozkazowi RETURN. Z programu wywołującego (w szczególnym przypadku z podprogramu) sterowanie przekazuje się podprogramowi za pośrednictwem makrorozkazu CALL.

W systemie DOS można również konstruować programy o tzw. strukturze nakładkowej overlay. Program taki powstaje wówczas gdy programowi łączącemu (linkage editor) opiszemy wzajemne rozmieszczenie w pamięci faz będących wynikiem jego działania. Język programu łączącego pozwala na układanie faz w struktury drzewowe analogiczne do struktury modułu typu over-

lay w systemie OS. W tym przypadku zadaniem programu łączącego jest jedynie odpowiednie zaadresowanie (tzn. przystosowanie do działania w odpowiednim miejscu pamięci) poszczególnych faz zgodne z ich rozmieszczeniem w ogólnej strukturze. Przekazywanie sterowania między fazami w programie o strukturze nakładkowej nie różni się od opisanego powyżej, tzn. stosuje się do tego omówione już makrorozkazy FETCH bądź LOAD i CALL.

8. Podsumowanie

Przedstawione w tym opracowaniu zestawienie możliwości IBM-owskich systemów OS i DOS w dziedzinie konstrukcji programów złożonych wskazuje wyraźnie, że realizatorzy faworyzowali system OS wyposażając go w tak bogaty i elastyczny aparat. Czytelnik z pewnością zauważył, że większość opisanych elementów zawartych w OS nie ma swoich odpowiedników w DOS. Wynikło to zapewne w głównej mierze z przyjętych założeń zakresu stosowania tych systemów w celu najekonomiczniejszego wykorzystania dostępnego sprzętu. I tak DOS ma zapewniać skuteczne funkcjonowanie instalacji wyposażonych w pamięć operacyjną o pojemności 128 k bajtów, a powyżej tej granicy OS opcja MFT albo MVT.

Nieekonomiczność systemu OS dla instalacji o pojemności pamięci operacyjnej mniejszej niż 128 k bajtów wynika przede wszystkim:

- z ilości czasu przeznaczonego na wykonanie funkcji supervisora, oraz
- z obszaru pamięci operacyjnej zajmowanego przez supervisor.

Dlatego też przy realizacji DOS minimalizowano funkcje realizowane przez supervisor.

Główną funkcją systemu OS, która pozwoliła zrealizować opisany aparat w OS jest przydzielanie modułom miejsca w pamięci.

9. Literatura

[1] IBM System/360 DOS. System Control and System Service Programs. Form GC24-5036

[2] IBM System/360 DOS. System Programmers Guide. Form: GC24-5073

[3] IBM System/360 OS. Supervisor and Data Management Services. Form GC28-6646

[4] IBM System/360 OS. Supervisor and Data Management Macro-Instructions. Form GC28-6647

[5] IBM System/360 Disk and Tape Operating Systems. Concepts and Facilities. Form GC24-5030

[6] BENDER G., FREEMAN D.N., SMITH J.D.: Function and Design of DOS/360 and TOS/360. IBM Systems Journal 6, 1967, nr 2, s. 2-21

[7] IBM System/360 Operating System. Concepts and Facilities. Form GC28-6535

[8] MEALY G.H., WITT B.I., CLARK W.A.: The Functional Structure of OS/360. IBM Systems Journal 5, 1966, nr 1

681.322.06

SYSTEMY OPERACYJNE[1]

Wstęp

W niniejszym artykule, przeznaczonym dla fachowców z dziedziny techniki obliczeniowej nie będących specjalistami w zakresie systemów operacyjnych, przedstawiono obraz istniejącej sytuacji oraz przewidywane kierunki rozwoju w dziedzinie funkcjonalności (performance) systemów operacyjnych. Ocena każdej nowej dynamicznie rozwijającej się dziedziny musi być częściowo subiektywna.

Aby treść artykułu była przejrzysta ograniczymy się do omówienia kilku wybranych problemów. Załączamy wykaz publikacji, z których kilka zaopatrzonych jest w obszerne bibliografie. Mam nadzieję, że podana literatura ułatwi czytelnikowi skorzystanie z nagromadzonych do tej pory doświadczeń i umożliwi dokładniejsze zaznajomienie się z interesującymi go zagadnieniami.

Ponadto założeniem autora było niewdawanie się w miarę możliwości w dyskusję z tezami przedstawionymi w innych artykułach. Pominięto więc całkowicie zagadnienia zabezpieczeń, ochrony i niezawodności, jak również problem struktury systemów operacyjnych, chyba że ma on związek z funkcjonalnością systemu.

Doświadczenia wykazują, że zachowanie się systemów przeznaczonych do rozwiązywania pewnych konkretnych zagadnień i przy stosunkowo stałym obciążeniu, jest względnie dobrze znane. Trudniej jest skonstruować system w przypadku, gdy nie ma ścisłej specyfikacji obciążenia i gdy jest ono

---

[1] Lynch W.C.: Operating System Performance. Communications of the ACM, t. 15, 1972, nr 7, s. 579. Opracowanie: mgr inż.Jolanta Krauze,Zakłady Wytwórcze Przyrządów Pomiarowych "ERA".

zmienne. Okazuje się, że o takich systemach jeszcze bardzo mało wiemy.

W ciągu ostatnich kilku lat zbudowano wiele modeli elementów składowych systemów operacyjnych oraz nagromadzono liczne doświadczenia w tej dziedzinie. Jednakże występowanie wielu oddziaływań wzajemnych między modułami w wielu dużych systemach pociąga za sobą konieczność skonstruowania kompleksowego modelu całego systemu. Badania zachowania się systemów operacyjnych zmierzają właśnie w kierunku uzyskania możliwości wykonania takiego kompleksowego projektu całego systemu. Potrzeba skonstruowania modelu kompleksowego wpływa na zagadnienie modularyzacji systemów, zwłaszcza sposób w jaki systemy są modularyzowane decyduje o możliwości zbudowania takiego modelu. W każdym razie koncepcja kompleksowego projektu systemu ujawnia wiele luk występujących w naszej obecnej wiedzy, jak również wytycza cele, do których powinniśmy zmierzać.

W artykule tego typu nie można nie wspomnieć o roli i pozycji jaką zajmują handlowe systemy operacyjne a zwłaszcza OS/360. Z drugiej strony byłoby niemożliwe i niepotrzebne podawanie tutaj długiego spisu zalet i wad wszystkich tych systemów handlowych. Zdecydowaliśmy więc ograniczyć się jedynie do kilku ogólnych uwag na temat roli i istoty systemu OS/360. Uwagi te odnoszą się oczywiście również do większości systemów konkurujących na rynku z systemem OS/360.

Punkt widzenia programów systemu operacyjnego

Z dotychczasowych doświadczeń wynika m.in., że suma składowych systemu nie reprezentuje go w sposób dostateczny. Innymi słowy, opis funkcjonalności poszczególnych elementów lub modułów systemu operacyjnego, ogólnie biorąc, da niewiele przesłanek do oceny funkcjonalności całego systemu. Odpowiedź systemu operacyjnego na różne obciążenia jest zwykle w dużym stopniu nieliniowa, występuje silne oddziaływanie wzajemne modułów systemu oraz liczne różnego rodzaju sprzężenia zwrotne. Użytkowników systemu należy traktować z zasady jako integralną część systemu, uczestniczącą we wzajemnych oddziaływaniach podczas korzystania z usług systemu.

Rozpatrzymy kilka przykładów. Weźmy znane zjawisko "wąskiego gardła"; jest to jakiś obiekt lub zasób wprowadzający ograniczenia. Często słyszy się o próbach usunięcia takiego "wąskiego gardła" z systemu. W takim przypadku liczy się na to, że uwolnienie się od ograniczeń narzuconych przez krytyczny obiekt poprawi funkcjonalność całego systemu operacyjnego. Jednakże w systemie może istnieć więcej niż jedno "wąskie gardło" wprowadzające ograniczenia. Jeżeli występują dwa takie "wąskie gardła", odblokowanie jednego z nich w niewielkim stopniu poprawi funkcjonowanie systemu; dopiero odblokowanie ich obu równocześnie da w efekcie poprawę. Opisana sytuacja jest właśnie przykładem nieliniowości i oddziaływań wzajemnych, występujących w czasie funkcjonowania systemu.

W ciągu ostatnich kilku lat wprowadzono wiele ulepszeń taktycznych w sposobach szeregowania obsługi (scheduling) urządzeń wejścia/wyjścia szczególnie w przypadku pamięci rotacyjnych jak dyski i bębny [1], [8], [10], [38], [9], [36], [20], [2], [19], [30], [32], [37]. Zajmijmy się rozpatrzeniem jednego z takich ulepszeń, mianowicie modelu szeregowania sektorów (sector) w pamięci bębnowej opisanego przez Coffmana [8]. Istotą tej techniki jest szeregowanie odwołań do pamięci bębnowej według możliwości szybkiej ich realizacji a nie w kolejności zgłoszeń. Dało to w efekcie skrócenie czasu oczekiwania a więc zwiększyło liczbę operacji wykonywanych w ciągu sekundy. W przypadku gdy liczba odwołań do pamięci, a więc i kolejka odwołań oczekujących na realizację nie jest duża, wspomniana wyżej technika nie daje żadnych korzyści w porównaniu z konwencjonalną techniką FCFS (pierwsze zgłoszone - pierwsze obsługiwane). Jednakże przy wzroście liczby odwołań oraz długości kolejki oczekujących odwołań regulamin ten wybierze to żądanie (spośród oczekujących w kolejce), które może być wykonane najwcześniej. Tak więc opisywany regulamin zwiększa szybkość realizacji odwołań w miarę wzrostu długości kolejki oczekujących żądań. Przy dużej liczbie odwołań i długiej kolejce, technika ta daje duże korzyści w porównaniu z konwencjonalną techniką FCFS. Według teorii sterowania jest to ujemne sprzężenie zwrotne: w miarę wzrostu kolejki odwołań, szybkość obsługi wykonywanej przez bęben wzrasta, co powoduje skrócenie kolejki dzięki szybszemu wykonywaniu żądań. W przypadku skrócenia się kolejki, szybkość obsługi wykonywanej przez bęben maleje. Takie ujemne sprzężenie zwrotne stabilizuje więc długość kolejki zadań.

Mechanizm ten może zostać poważnie zakłócony przez "wąskie gardło" znajdujące się w zupełnie innej części systemu operacyjnego. Na przykład

często się zdarza, że system daje możliwość tylko jednorazowego odwołania się do pamięci pomocniczej na jedno zadanie (job) lub na jeden sektor pamięci operacyjnej (partition).

Tak więc długość kolejki nie może być większa niż liczba programów w pamięci operacyjnej lub liczba sektorów pamięci operacyjnej. Długość kolejki odwołań do pamięci bębnowej jest więc ograniczona przez "wąskie gardło" w pamięci rdzeniowej w taki sposób, że kolejka nie może osiągnąć takich rozmiarów aby algorytm szeregowania sektorów (sector) działał efektywnie [5]. Wystąpienie takiego oddziaływania wzajemnego w systemie zmieni całkowicie funkcjonalność systemu, w stosunku do oczekiwań opartych na przebadaniu poszczególnych jego składowych.

W systemach operacyjnych występuje również często dodatnie sprzężenie zwrotne. W takim przypadku szybkość obsługi wykonywanej przez dany obiekt w ramach systemu operacyjnego będzie się zmniejszała w miarę wzrostu ilości żądań obsługi. Zjawisko takie może upośledzić funkcjonowanie systemu w nieznacznym stopniu, może jednak prowadzić do niestabilności, która spowoduje nagłe załamanie się funkcjonalności systemu. Przykładem takiej sytuacji jest dobrze znane zjawisko migotania stron (trashing) w systemach z paginacją. Występowanie migotania oznacza całkowite załamanie się pracy systemu wynikające z nadmiernego wykorzystania pamięci wirtualnej i wieloprogramowania [12], [3]. Według Denninga [12] mechanizm tego zjawiska wygląda w przybliżeniu następująco. System operacyjny zauważa, że funkcjonowanie jest nieefektywne, ponieważ procesor centralny nie jest w pełni wykorzystany i występuje mało transmisji stron z bębna. Wnioskuje, że sytuację tę można poprawić przez zwiększenie poziomu wieloprogramowania, tak żeby procesor centralny wykorzystał swój wolny czas na wykonanie dodatkowych programów. Jednakże załadowanie takiego dodatkowego programu może co prawda łatwo zwiększyć wykorzystanie pamięci bębnowej, ale powiększając liczbę zbędnych transmisji bloków zmniejsza w rezultacie sumaryczne wykorzystanie procesora centralnego. W takiej sytuacji system operacyjny wzmacnia swój poprzedni rozkaz i załadowuje jeszcze więcej programów. Oczywiście, niedługo potem następuje całkowite załamanie się funkcjonalności systemu. Opisane sprzężenie zwrotne jest przykładem sprzężenia dodatniego.

W innym przykładzie dodatniego sprzężenia zwrotnego w pętlę sprzężenia włączony jest również użytkownik systemu. Mam tu na myśli szeregowa-

nie danych wyjściowych przeznaczonych do wydrukowania. Z zasady, im krótszy jest czas odpowiedzi systemu lub czas wykonania pełnego cyklu obliczeń, tym mniej danych wyjściowych na jednostkę obliczeniową będą wymagali użytkownicy. I odwrotnie, im dłuższy czas odpowiedzi lub większa długość cyklu obliczeniowego, tym większa ilość danych wyjściowych będzie potrzebna użytkownikom. Jednakże czas odpowiedzi, czy też czas wykonania pełnego cyklu obliczeń może zależeć w dużym stopniu od ilości danych wyjściowych, które muszą przejść przez system. Istnieje więc potencjalne niebezpieczeństwo wytworzenia się niestabilnej pętli sprzężenia zwrotnego, w której zwiększenie czasu odpowiedzi systemu powoduje zwiększenie wymagań użytkownika co do ilości danych wyjściowych, co z kolei prowadzi do dalszego przedłużenia czasu odpowiedzi [24].

Z ostatniego przykładu wynika dodatkowe wymaganie, a mianowicie konieczność uwzględnienia cech charakterystycznych użytkowników systemu. Postulat ten ma szeroki zakres, ponieważ obejmuje zarówno cechy psychiczne użytkownika jak i jego programy. Należy wziąć pod uwagę właściwości statyczne użytkowników i ich programów oraz ich właściwości dynamiczne tzn. przewidywać jak postąpią i jak zmienią swoje programy w odpowiedzi na sytuację, którą zaprezentuje im system. Łatwo uświadomić sobie, że w większości systemów występuje zależność między ładunkiem przedstawianych im zadań a funkcjonalnością systemu, i oczywiście odwrotnie.

Modularność

Obecne systemy operacyjne są z zasady modularyzowane poziomo, tzn. poszczególne moduły odpowiadają odrębnym częściom systemu. Niewiele uwagi poświęcono jednak wzajemnym oddziaływaniom funkcjonalnym (performance interactions) występującym między modułami systemu. Głównym czynnikiem określającym charakter modularyzacji w obecnie stosowanych systemach operacyjnych jest prawo Conway'a [27], które stwierdza, że organizacja systemu operacyjnego jest dokładną kopią organizacji zespołu, który go opracował. Porozumienie między zespołami projektującymi odbywa się za pośrednictwem koordynatora mającego niewiele wspólnego z poszczególnymi zespołami. Taka metoda postępowania prowadzi do konieczności dopasowywania oddziaływań między modułami post factum, co jest bardzo trudne. Gdyby samoloty były projektowane w ten sposób nawet pilot oblatywacz nie miałby odwagi na nich latać.

Obecnie zaleca się nowocześniejsze podejście do zagadnienia a mianowicie programowanie strukturalne [39], [15], [21], [14]. Podejście to polega na zastosowaniu modularyzacji pionowej tzn. rozwarstwienia, gdzie funkcja jednej warstwy jest opisana w kategoriach funkcji realizowanych przez warstwę poprzednią, przy czym należy przyjąć kierunek od góry do dołu. Rozpoczyna się od sformułowania funkcji całego systemu, po czym określa się funkcje pomocnicze potrzebne do uzyskania funkcji nadrzędnej. Proces ten jest następnie powtarzany dając ciąg warstw lub inaczej modułów pionowych, aż do momentu gdy system zostanie całkowicie zrealizowany w terminach dostępnych operacji elementarnych zwykle badanych składowymi hardware'u. Przy dużych systemach należy oczywiście stosować strategię mieszaną. W miarę przechodzenia do coraz niższych warstw ilość nagromadzonego materiału staje się tak duża, że pojedynczy zespół nie może sobie z nim poradzić. Tak więc zgodnie z prawem Conway'a niezbędny jest również pewien procent modularyzacji poziomej.

Na jakiej zasadzie można system podzielić na moduły z punktu widzenia jego funkcjonalności? Z poprzedniego rozdziału wynika, że silne oddziaływanie wzajemne występujące między modułami narzuca konieczność określenia a priori jak będzie funkcjonował system złożony z tych modułów. Jest to zgodne z przyjętą praktyką przy projektowaniu złożonych systemów w innych dziedzinach techniki. Na przykład przy projektowaniu samolotu jego całkowity ciężar jest określany na początku procesu projektowego. Z całą pewnością nie ustala go się w ten sposób, że zaprojektowany samolot stawia się na wadze. Taka zasada odpowiada modularyzacji pionowej, w której specyfikacja funkcjonalności stanowi integralną część projektu każdej warstwy. Nie trzeba chyba wspominać, że przy projektowaniu każdej warstwy należy przeprowadzić odpowiednią analizę i pomiary celem upewnienia się, że jej funkcjonalność jest zgodna z założeniami projektu.

Nie chcielibyśmy aby czytelnik odniósł wrażenie, że modularyzacja pionowa stanowi panaceum na nieudolność przy projektowaniu aspektu funkcjonalności systemów operacyjnych. Obecnie warstwowych systemów operacyjnych jest jeszcze niewiele. Na pierwszym miejscu trzeba wymienić system THE [16]. Doświadczenia, których dostarczają te systemy są bardzo interesujące. Technika rozwarstwiania sprawdza się, jeżeli poszczególne warstwy są dobrze zdefiniowane zarówno funkcjonalnie jak i w aspekcie

dynamicznym. Zwłaszcza stałe czasowe reakcji danej warstwy powinny być w dostatecznym stopniu mniejsze od stałych czasowych sąsiadującej z nią wyższej warstwy. Innymi słowy, funkcje warstwy niższej przyjętej za podstawową muszą być spełniane w czasie dostatecznie krótkim aby można go było pominąć w odniesieniu do skali czasowej sąsiadującej z nią wyższej warstwy. Powszechnie uważa się, jakkolwiek nie jest to jeszcze dowiedzione, że jest to warunek wystarczający aby system mógł być rozłożony na warstwy pionowe. O ile wiemy, nikt jeszcze dotychczas nie wypowiedział się czy ten warunek jest niezbędny.

Jako ilustrację do tej zasady rozważmy koncepcję pamięci wirtualnej [13]. Ostatnio stwierdzono [3], że stała czasowa dla operacji stronicowania (translacja pamięci fizycznej na pamięć wirtualną), jest około kilku sekund. Na przykład program z 20-stronicowym zbiorem roboczym nie jest rzeczą niecodzienną. Jednak wiele stronicowanych pamięci bębnowych może dla przypadku pojedynczego programu przesyłać dane z prędkością 20 do 30 stron na sekundę (można uzyskać znacznie większą globalną prędkość transmisji). Tak więc wprowadzenie programu zawierającego 20-stronicowy zbiór roboczy może wymagać sekundy lub więcej na ustabilizowanie (equalize) dystrybucji pamięci między programy. Taka stała czasowa jest znacznie większa niż czas trwania operacji wejścia/wyjścia i jest bliska czasowi reakcji szybszych użytkowników. W związku z tym pojawiają się poważne zakłócenia pracy systemu wynikające z oddziaływań wzajemnych programów wchodzących i wychodzących z pamięci rdzeniowej oraz częstych reakcji urządzeń wejścia/wyjścia i użytkownika. Funkcjonalność systemu może być w wyniku tych zakłóceń gorsza niż można by się spodziewać. Wynika z tego, że koncepcja pamięci wirtualnej jaką dysponuje użytkownik jest koncepcją słuszną tylko dla procesów o stałych czasowych rzędu kilkudziesięciu sekund. W systemie o konstrukcji warstwowej, wymagającym zastosowania szybkiej pamięci wirtualnej, będą więc występowały zakłócenia funkcjonalności.

Stałą czasową można obniżyć przez zwiększenie szybkości transmisji danych z bębna. Wspomniana wyżej prędkość 20 do 30 stron na sekundę odnosi się do pamięci bębnowej z szeregowaniem sektorów w trybie "demand paging", w którym na jeden obrót bębna przypada transmisja jednego bloku. Zwiększenie rozmiaru strony lub skrócenie czasu przygotowania stron do transmisji da w wyniku zmniejszenie stałej czasowej. Nowsze systemy

operacyjne pracujące z paginacją, jak np. Tenex [4] oraz BCC [5] stosują operacje typu "roll-in-roll-out" celem zmniejszenia tej stałej czasowej.

Stan aktualny

Przejdziemy obecnie do rozpatrzenia istniejącej aktualnie sytuacji oraz prawdopodobnych kierunków jej rozwoju w niedalekiej przyszłości. Rozpoczniemy od omówienia obecnie istniejącej sytuacji, koncentrując się na głównych kierunkach. Szczegółowe omówienie poruszanych zagadnień znajdzie czytelnik w źródłach podanych w bibliografii.

W ciągu ostatnich pięciu lat skonstruowano wiele modeli elementów składowych systemów operacyjnych. W niektórych posłużono się metodami stochastycznymi oraz teorią kolejek celem przebadania zachowania się systemu przy różnych algorytmach szeregowania (scheduling). Istnieją modele sterowania pamięciami rotacyjnymi, do szeregowania przydziału pamięci rdzeniowej oraz do szeregowania pracy procesorów. Ten pokaźny zbiór modeli służy zarówno jako katalog badań standardowych sytuacji jak również stanowi podstawę do dalszych badań rozwojowych. Materiał dotyczący tego tematu jest zbyt obszerny aby można było go tutaj omówić. Zainteresowanych odsyłamy do wykazu bibliografii [25], [9] oraz [40].

Z systemów istniejących, najlepsze wyniki z punktu widzenia funkcjonalności osiągnięto w przypadku systemów o niezmiennym obciążeniu, a zwłaszcza systemów stosowanych do rezerwacji miejsc w komunikacji lotniczej. W systemach tych rozszerzono opisaną wyżej technikę modelowania poszczególnych składowych systemu w kierunku budowania modeli przepływowych, wykorzystywanych do prognozowania obciążenia i funkcjonalności każdej składowej oraz rodzajów oddziaływań wzajemnych, które wystąpią w różnych częściach systemu. Było to możliwe, ponieważ ogólnie biorąc obciążenie systemów obsługujących rezerwację miejsc w komunikacji lotniczej jest stałe, możliwe do przewidzenia i ze statystycznego punktu widzenia stabilne. Projektanci systemów podkreślają, że obciążenie systemu powinno być dokładnie przeanalizowane [29], [26]. Systemy te są projektowane z tak ścisłą tolerancją, że odchylenie obciążenia o 10% w stosunku do wielkości założonej może spowodować pogorszenie funkcjonowa-

nia systemu w stopniu uniemożliwiającym jego wykorzystanie. Należy ponadto zwrócić uwagę, że najpierw konstruuje się model całego systemu a dopiero potem przydziela się szczegółowe funkcje.

Ponieważ zakres wymagań użytkowników jest bardzo szeroki i nie dający się dokładnie określić, większość systemów operacyjnych jest zbudowanych na zasadzie modularyzacji funkcjonalnej czyli poziomej. Dobrym przykładem takiej struktury jest system OS/360 [27], [28]. OS/360 był początkowo pomyślany, a w dużym stopniu jest nim nadal, jako podręczny zbiór narzędzi, technik i operacji, z którego to zbioru można wybrać i zestawić system spełniający konkretne, określone wymagania użytkownika. W takim systemie istnieją różne sposoby osiągnięcia tego samego celu i różne urządzenia hardware'u, które mogą spełniać tę samą funkcję. Użytkownik musi wybrać zestaw urządzeń hardware'u oraz modułów software'u, zestawić je ze sobą tak aby system optymalnie spełniał postawione wymagania. System OS/360 oraz System 360 dysponuje tak obszernym zbiorem operacji, części i elementów, że w zasadzie jest to z pewnością możliwe. Jednak analiza i optymalizacja całego systemu (które to czynności użytkownik musi przeprowadzić) wykraczają często poza możliwości aktualnego stanu wiedzy. Nieznane jest bowiem wymagane obciążenie oraz złożone oddziaływania wzajemne w systemie. Nikt nie jest w stanie (nawet ekspert w tej dziedzinie a już na pewno nie kierownik ośrodka obliczeniowego) zbudować takiego modelu i przeprowadzić takiej analizy, co jest niezbędne do optymalizacji systemu o modularyzacji poziomej.

Pewne jest, że powodzenie projektu zależy w znacznym stopniu od dokładności z jaką można scharakteryzować obciążenie projektowanego systemu. W dziedzinie tej nastąpił ogromny postęp [22], [6], [23] i [33] . Pozostało jednak tak dużo do nauczenia się, że przyjmując skalę porównawczą można powiedzieć, że pozostajemy nadal w stanie całkowitej ignorancji. Nadal brak nam jeszcze wiedzy o takich wielkościach jak rozkład czasów oddziaływań (interaction times), ilość danych wejściowych i wyjściowych przypadających na oddziaływanie, ilość obliczeń wymaganych przy jednym oddziaływaniu itd. Co więcej, konieczna jest znajomość pewnych danych szczegółowych na temat użytkowania systemu. Dane dotyczące lokalizacji odwołań do pamięci, a zwłaszcza charakterystyk odwołań wewnątrz poszczególnych pól adresowych, zarówno w pamięci głównej jak i pomocniczej, mają zasadnicze znaczenie przy układaniu algorytmów przydziału pola w pamięci oraz algo-

rytmów dostępu. Takie dane nawet w odniesieniu do systemów, które istnieją i już od kilku lat są wykorzystywane, są bardzo skąpe. Oczywiste jest, że jeżeli nie wiemy w jaki sposób użytkownicy wykorzystują aktualnie istniejące systemy, tym bardziej trudno jest przewidzieć w jaki sposób będą wykorzystywali nowe prezentowane im systemy. Jedno jest tylko pewne, a mianowicie to, że użytkownik, jeżeli tylko będzie miał okazję, będzie dostosowywał swój tryb użytkowania do okoliczności w jaki się znajduje. Fakt ten komplikuje dodatkowo i tak już trudną sytuację.

Kierunki rozwojowe

Jak wynika z poprzednich rozdziałów, inżynieria funkcjonalna systemów operacyjnych zmierza w kierunku, który można by nazwać kompleksowym projektowaniem systemów. Taka metoda projektowania wymaga rozpoznania istotnych dla projektowania parametrów systemu, ich wzajemnych powiązań i zależności ograniczających oraz określenia zbioru cech obiektywnych (wynikających z wymagań stawianych systemowi), które można przyjąć jako miarę wartości systemu. Spróbujmy zidentyfikować niektóre z tych parametrów. Takimi parametrami, których wartość należy określić są: konfiguracja systemu, wielkość pamięci rdzeniowej, szybkość procesora centralnego, urządzenia końcowe, urządzenia transmisyjne, rodzaj i powierzchnia pamięci rotacyjnych. Podobnie istnieją parametry części software'owej systemu, na przykład rozmiary strony w systemach z paginacją, rodzaj algorytmów szeregowania transmisji danych z pamięci rotacyjnych, liczba transmisji przewidywana dla każdej z tych pamięci, wielkość bloków, które będą transmitowane do urządzeń wejścia/wyjścia, organizacja plików a zwłaszcza w odniesieniu do odwołań pamięciowych, sposób zorganizowania słowników w pamięci pomocniczej, poziom wieloprogramowania oraz inne parametry. Stosunkowo łatwo jest ułożyć listę parametrów mających wpływ na funkcjonalność systemu; trudniej jest wskazać parametry krytyczne. Jeszcze trudniejsze, a zarazem istotniejsze, jest określenie wzajemnych powiązań i zależności, które wiążą ze sobą parametry systemu. Nie wszystkie wartości parametrów mogą być wybrane dowolnie. Na przykład parametry takie jak poziom wieloprogramowania, ilość danych wejściowych i wyjściowych, wielkość bloków transmitowanych przez urządzenia wejścia/wyjścia oraz liczba kanałów transmitujących są powiązane określonymi zależnościami. Tutaj użyteczny staje się katalog modeli podsystemów, o którym wspomnieliśmy powyżej.

Dysponujemy już obecnie katalogiem modeli obejmującym wszystkie zależności, które mogą nas interesować. W każdym przypadku określenie ilościowych zależności między istotnymi parametrami systemu jest bardzo trudnym zadaniem, które jednak trzeba wykonać przy kompleksowym projektowaniu systemu. Rozwiązanie tego problemu jest dodatkowo utrudnione przez brak informacji co do zachowania się użytkownika. Na przykład przy projektowaniu urządzeń wejścia/wyjścia należy wiedzieć jaka ilość danych będzie transmitowana. Jak wspomniano wyżej, wielkość ta jest często funkcją czasu reakcji systemu. Może ona być również funkcją urządzeń wejścia/wyjścia dostępnych dla użytkownika. W każdym razie do zbudowania kompleksowego modelu systemu niezbędne jest uprzednie założenie i sprawdzenie zależności ilościowych.

Ustalenie parametrów obiektywnych (objective functions) jest nieco łatwiejsze niż określenie wewnętrznych powiązań w systemie, nie jest jednak w żadnym przypadku zadaniem zbyt prostym. Na przykład, załóżmy, że postanawiamy zbudować tani system obejmujący 100 stanowisk dalekopisowych i pracujący w trybie podziału czasu. Jeżeli przyjmiemy to za parametr obiektywny możemy łatwo spełnić te wymagania łącząc 100 dalekopisów razem za pomocą sieci kablowej. Bez centralnego procesora, pamięci rdzeniowej oraz przekładni transmisyjnej czas odpowiedzi będzie dążył do nieskończoności. Założone wymaganie obiektywne będzie jednak spełnione. W rzeczywistości, bez użycia procesora centralnego i pamięci, niezbędne będzie zatrudnienie więcej niż 100 dalekopisów. Tak więc zakwalifikowanie danego wymagania do grupy parametrów obiektywnych lub zależnych powinno być dokonywane ostrożnie.

Formułując powyższe wnioski należy powiedzieć, (nie wdając się w szczegóły), że ustalenie parametrów projektowanego systemu określa przestrzeń potencjalnych projektów. Zależności między tymi parametrami zawężają tę przestrzeń do przestrzeni projektów realnych. Miejmy nadzieję, że zależności te nie wykluczają wykonania co najmniej jednego lub kilku projektów. Po ilościowej ocenie sytuacji, z uwzględnieniem analizy ilościowej parametrów obiektywnych, możemy przystąpić do formalnej optymalizacji systemu. Przykład takiego podejścia podany jest przez Foley'a [17] oraz Stimlera i Bionsa [35]. Muszę tutaj zaznaczyć, że jeszcze nie znamy dobrej metody zdefiniowania parametrów i związków między nimi. W tym kierunku zmierza, jak mi się wydaje, dziedzina badania funk-

cjonalności systemów operacyjnych. Opisane podejście nie jest czymś zupełnie nowym odkrytym dopiero w związku z rozwojem komputerów i systemów operacyjnych. Metoda ta została w mniejszym lub większym stopniu przyjęta w innych dyscyplinach inżynierskich, w których zgromadzono już wystarczająco dużą wiedzę. Technika ta jest już od dawna stosowana w inżynierii lądowej i lotniczej [31], [18]. Jest to naturalne w każdej dyscyplinie inżynieryjnej, w której zgromadzona wiedza pozwala na przyjęcie niezbędnych założeń ilościowych. Jak wykazano wyżej, baza niezbędnej wiedzy podstawowej w interesującej nas dziedzinie jest właśnie w stadium powstawania.

Przyjęcie tego punktu widzenia stawia w nowym świetle problem pomiarów i technik pomiarowych. W wielu przypadkach pomiary prowadzone obecnie mają na celu wyjaśnienie mechanizmu zjawisk, które były dotychczas zupełnie nieznane [34]. Jedynie w nielicznych przykładach pomiary te wykonuje się aby zgromadzić informacje niezbędne do zbudowania modeli elementów składowych systemu. W ramach tendencji do kompleksowego projektowania systemów, celem pomiarów powinna być przede wszystkim kontrola jakościowa. Metodologia kompleksowego projektowania systemów wskazuje na konieczność określenia z góry funkcjonalności systemu a następnie skonstruowania systemu tak aby spełniał założone warunki. Fakt, że systemy liczące i operacyjne wykazują duże nieliniowości oraz duży stopień zależności wewnętrznych (interactive) oznacza, że na najogólniejszym poziomie szczegółowości będą nam potrzebne modele symulujące funkcjonowanie całego systemu. Aby modele te były operatywne, nie powinny być zbyt skomplikowane. W miarę przechodzenia na niższe poziomy szczegółowości potrzebne będą coraz dokładniejsze modele analityczne. W trakcie projektowania poszczególnych warstw pionowych systemu będziemy potrzebować technik pomiarowych umożliwiających sprawdzenie czy funkcjonalność konstruowanego systemu spełnia założone wymagania w dostatecznym stopniu.

Powyższe uwagi dostarczają sporo wniosków dotyczących wymaganych technik pomiarowych. Najważniejszymi wielkościami, które powinny być zmierzone są oczywiście parametry projektowe. Musimy bowiem sprawdzić czy ich wartości są zgodne z założonymi i czy w konstruowanym systemie występują takie same ograniczenia i zależności jak w projekcie. Musimy znać długość kolejek, czasy przebiegu zadania, czasy oczekiwania, szybkości przepływu dla poszczególnych mechanizmów systemu itp. [5], [7].

Do takich pomiarów potrzebny będzie raczej zestaw rejestratorów taśmowych niż skomplikowany system rejestracji cyfrowej. Tzw. cyfrowy rejestrator o pracy ciągłej (event-trace logger)[32], który rejestruje każde ważniejsze posunięcie maszyny jest użyteczny przede wszystkim do analizy mechanizmów występujących w systemie, których zachowanie się jest trudne do przewidzenia. Określenie parametrów systemu za pomocą rejestratora cyfrowego wymaga dużej ilości danych wyjściowych z systemu, co może zakłócać pracę samego systemu i spowodować zmniejszenie ilości danych użytkowych. Tak więc technika ta nie wydaje się szczególnie użyteczna do wyznaczania wartości globalnych parametrów systemu.

W pracach projektowych pomocna jest metoda symulacji. Potrzebne jest wczesne uzyskanie informacji dotyczących modułów i ich oddziaływań wzajemnych niezgodnych z pierwotnym modelem lub takich, które nie były modelowane. Jedynie symulacja lub rzeczywisty system mogą ujawnić nieprzewidziane trudności. Aby symulacja była skuteczna musi być na tyle prawdziwa aby z dużym prawdopodobieństwem odwzorowała nieoczekiwane mechanizmy i na tyle nieskomplikowana w porównaniu z rzeczywistym systemem, aby opłacało się ją stosować.

W wyniku zastosowania wyżej opisanej techniki projektowania otrzymamy przybliżenia, które dostarczą wskazówek co do dekompozycji i modularyzacji systemu. Badania te powinny być połączone z analizą systemu, która wskaże w jakich warunkach przybliżenie autonomiczności modułu może zastąpić szczegółową analizę całego systemu. Otrzymamy stąd zbiór zasad wskazujących jak podzielić system na moduły bez równoczesnego wprowadzenia nieprzewidzianych i niepożądanych oddziaływań w systemie. Jedną z takich zasad może być opisana wyżej zasada dotycząca stałych czasowych. W każdym razie jako podstawową, należy przyjąć następującą zasadę inżynierską: "jeżeli nie rozumiesz tego, co masz zaprojektować, zaprojektuj to, co rozumiesz".

Omawiając parametry obiektywne systemu przyjęliśmy, że mogą być one łatwo i zwięźle określone. Zagadnienie to jest znacznie trudniejsze w przypadku konstruowania systemu uniwersalnego jak np. OS/360, zaprojektowanego w ten sposób, że jego podsystemy powinny pracować efektywnie w środowiskach różnych użytkowników. Byłoby pożądane opracowanie zestawu technik, które mogłyby być zastosowane przez kierownika ośrodka obli-

czeniowego oraz inżyniera obsługującego system do określenia elementów składowych i organizacji wybranego podsystemu (np. systemu OS i/lub z systemem 360), które zapewniłyby optymalną konfigurację hardware'u i software'u. Kiedy to będzie osiągalne czytelnik może sam próbować ocenić, biorąc pod uwagę nasz obecny stan wiedzy oraz omówione wyżej tendencje rozwojowe.

Znaczenie rozwoju hardware'u

Podczas gdy my zajmujemy się planowaniem przyszłości i zastanawiamy się jak wykorzystać System 360, koledzy zajmujący się projektowaniem hardware'u nie pozostają bezczynni. Metoda kompleksowego projektowania systemu dotyczy zarówno hardware'u jak i software'u. Inżynierowie wiedzą jaki sprzęt można zaprojektować i jak to najlepiej zrobić. Nie są oni jednak w lepszym położeniu niż projektanci software'u, kiedy dochodzi do określenia jakie elementy trzeba skonstruować i jak one powinny być zorganizowane. Określenie rozmiaru strony, geometria pamięci rotacyjnych oraz znaczenie zwiększenia szybkości procesorów są to zagadnienia, których nie można rozpatrywać w oderwaniu od software'u i projektowania systemu operacyjnego. Projektowanie hardware'u i software'u powinno, i z pewnością będzie, rozwijać się razem.

Wiele modeli elementów składowych systemów operacyjnych wymaga obszernych obliczeń uwzględniających skończony czas odpowiedzi [20]. W wielu przypadkach potrzebne są również informacje o zachowaniu się podsystemów mechanicznych w funkcji czasu, np. dotyczące sygnałów pozycyjnych określających położenie dysków. Gdy algorytmy te będą lepiej znane, ustalone i zaakceptowane należałoby je włączyć bezpośrednio do hardware'u jako część kompleksowego projektu systemu. Dysponujemy już komputerami zbudowanymi techniką obwodów scalonych, byłoby więc dobrze z ekonomicznego punktu widzenia aby akcje zarządzania były również zrealizowane w ramach tej technologii.

Z drugiej strony mamy już pamięci zewnętrzne o pojemności rzędu biliona (10^3 miliardów) bitów. Pamięć taka wprowadza nowy niezbadany poziom w hierarchii pamięci. Brak ustalonych metod w zakresie odwołań do pamięci i charakterystyk tych odwołań odnosi się również i do tych no-

wych urządzeń. Optymalny tryb organizacji tych pamięci pozostaje jeszcze do opracowania. Wiadomo jednak, że możliwości systemów współpracujących z pamięcią o tak dużej pojemności są ogromne. Gdy w niedalekiej przyszłości wejdą one szerzej w użycie, stosowane obecnie fragmentaryczne podejście do zagadnienia ich wykorzystania może doprowadzić do katastrofy.

Wnioski

Występowanie silnych oddziaływań wzajemnych między modułami systemu oraz między użytkownikiem i systemem powoduje, że problem projektowania i oceny systemów operacyjnych może być rozwiązany jedynie metodą zbudowania kompleksowego projektu systemu uwzględniającego wszystkie parametry hardware'u i systemu operacyjnego. Ponieważ dotychczas zrobiono w tej dziedzinie bardzo mało, można przypuszczać, że w najbliższej przyszłości nastąpi rozwój badań tego zagadnienia. Niezbędne będzie również zgromadzenie szczegółowych informacji o dynamice oddziaływań między modułami oraz cechach charakterystycznych użytkownika. Zebrano już sporo takich informacji, wciąż jednak istnieją tutaj poważne luki, których wypełnienie będzie wymagało dodatkowego wysiłku. Przyjęcie opisanej metody postępowania z pewnością wpłynie na sposób modularyzacji i projektowania systemów.

Literatura

[1] ABATE J., DUBNER H.; Optimizing the Performance of a Drum-like Storage. IEEE Trans. Computers C-18, 1969, nr 11, s. 992-997.

[2] ABATE J., DUBNER H., WEINBERG S.B.: Queueing Analysis of the IBM 2314 Disk Storage Facility. J. ACM, 1968, nr 4, s. 577-589.

[3] ALDERSON A., LYNCH W.C., RANDELL B.: Thrashing in a Multiprogrammed Paging System. International Seminar on Operating Systems Techniques, Queens U., Belfast, Northern Ireland, 1971.

[4] BOBROW D.G. i in.: A Paged Time Sharing System for the PDP-10, Proc. Third Symposium on Operating System Principles, Stanford U., 1971, s. 1-10.

[5] BOSKETL F.: The Dependence of Computer System Queues upon Processing Time Distribution and Central Processor Scheduling. Proc. Third Symposium on Operating System Principles, Stanford U., 1971, s. 109-113.

[6] BROWN B.S., GUSTAVSON F.G., MARKIN E.S.: Sorting in Paging Environment. Comm. ACM, 1970, nr 8, s. 483-494.

[7] CAMPBELL D.J., HEFFNER W.J.: Measurement and Analysis of Large Operating Systems During System Development, Proc. AFIPS 1968 FJCC, t. 33, Pt. 1, AFIPS Press, Montvale, N.J., s. 903-914.

[8] COFFMAN E.G.: Analysis of a Drum Input/Output Queue under Scheduled Operation in a Paged Computer System. J. ACM, 1969, nr 1, s. 73-90.

[9] RIVET P.H.: Data Traffic Models for the Performance of Modular Computer Systems. Ph.D.Th., Case Western Reserve U., 1971.

[10] DENNING P.J.: Effects of Scheduling on File Memory Operations. Proc. AFIPS 1967 SJCC, t. 30, AFIPS Press, Montvale, N.J., s. 9-21.

[11] DENNING P.J.: The Working Set Model for Program Behavior. Comm. ACM, 1968, nr 5, s. 323-333.

[12] DENNING P.J.: Thrashing: Its Causes and Preventions. Proc. 1968 AFIPS FJCC, t. 33, Pt. 1, AFIPS Press, Montvale, N.J., s. 915-922.

[13] DENNING P.J.: Virtual Memory. Computing Surveys, 1970, nr 3, s. 153-190.

[14] DIJKSTRA E.W.: Complexity Controlled by Hierarchical Ordering of Function and Variability. NATO Conf. on Software Engineering, Garmisch, Germany, 1968, s. 181-185.

[15] DIJKSTRA E.W.: Notes on Structured Programming. EWD249, Technical U. Eindhoven, The Netherlands, 1969.

[16] DIJKSTRA E.W.: The Structure of the T.H.E. Multiprogramming System. Comm. ACM, 1968, nr 5, s. 341-346.

[17] FOLEY James D.: An Approach to the Optimum Design of Computer Graphics Systems. Comm. ACM, 1971, nr 6, s. 380-390.

[18] FOX R.L.: Optimization Methods for Engineering Design. Addison-Wesley, Reading, Mass., 1971.

[19] FRANK H.: Analysis and Optimization of Disk Storage Devices for Time-Sharing Systems. J. ACM, 1969, nr 4, s. 602-620.

[20] GREENBERG M.L.: An Algorithm for Drum Storage Management in Time-Sharing Systems. Proc. Third Symposium on Operating System Principles, Stanford U, 1971, s. 141-149.

[21] HANSEN P.B.: The Nucleus of a Multiprogramming System. Comm. ACM, 1968, nr 4, s. 74-84.

[22] LOWE Thomas C.: The Influence of Data Base Characteristics and Usage on Direct Access File Organization. J. ACM, 1968, nr 4, s. 535-548.

[23] LUM V.Y., YUEN P.S.T., DODD M.: Key to Address Transform Techniques: a Fundamental Performance Study on Large Existing Formatted Files. Comm. ACM, 1971, nr 4, s. 228-239.

[24] LYNCH W.P.: Evolution of Computer Operating Systems. 1967, IEEE International Convention Record, Pt. 10, s. 18-22.

[25] McKINNEY J.M.: A Survey of Analytical Time-sharing Models. Computing Surveys, 1969, nr 2, s. 105-116.

[26] MARTIN J.: Design of Real-Time Computer Systems. Prentice-Hall, Englewood Cliffs, N.J., 1967.

[27] MEALY G.H.: The System Design Cycle. Proc. Second Symposium on Operating Systems Principles, Princeton U., 1969, s. 1-7.

[28] MEALY G.H., WITT B.I., CLARK W.A.: The Functional Structure of OS/360. IBM Systems J., 1966, nr 1, s. 2-51.

[29] MELBOURN A.J.: Response Time Considerations. IBM European Systems Research Institute. Geneva.

[30] MERTEN A.G.: Some Quantitative Techniques for File Organization, Tech. Rep. No. 15, U. of Wisconsin Computing Center, 1970.

[31] MIURA H.: An Optimal Configuration Design of Lifting Surface Type Structures under Dynamic Constraints. Ph.D. Thesis, Case Western Reserve U., 1971, Cleveland, Ohio.

[32] PINKERTON T.B.: Program Behavior and Control in Virtual Storage Computer Systems. CONCOMP Project Rep. No. 4, U. of Michigan, 1968.

[33] RODRIGUEZ-ROSELL J.: Experimental Data and How Program Behavior Affects the Choice of Schedular Parameters. Proc. Third Symposium on Operating Systems Principles, Stanford U., 1971, s. 156-163.

[34] SATTYER J.H., GIOTELL J.W.: The Instrumentation of Multics. Comm. ACM, 1970, nr 8, s. 495-500.

[35] STIMLER S., BIONS K.A.: A Methodology for Calculating and Optimizing Real-Time System Performance. Comm. ACM, 1968, nr 7, s. 509-516.

[36] TEOREY T.J., PINKERTON T.B.: A Comparative Analysis of Disk Scheduling Policies. Proc. Third Symposium on Operating System Principles, Stanford U., 1971, s. 114-121.

[37] WEINGARTEN A.: The Analytical Design of Real-Time Disk Systems, Proc. IFIP Congr. 1968, North Holland Pub. Co., Amsterdam, s. D131-137.

[38] WEINGARTEN A.: The Eschenback Drum Scheme. Comm. ACM, 1966, nr 7, s. 509-512.

[39] WIRTH N.: Program Development Stepwise Refinement. Comm. ACM, 1971, nr 4, s. 221-226.

[40] BANT R., TSICHRITZES D.: A Selective Annotated Bibliography. Tech. Rep. 24, 1971, Dept. of Computer Science, U. of Toronto.

681.322.06

TWORZENIE LEPSZEGO OPROGRAMOWANIA MATEMATYCZNEGO[1]

Wstęp

Oprogramowanie matematyczne mogłoby i powinno być zdecydowanie lepsze aniżeli jest. W artykule niniejszym zamieszczono pewne sugestie na temat działań zmierzających do poprawy istniejącej sytuacji. Dotyczą one takich zagadnień jak: dokumentacja, standardy, poprawność oraz konstrukcja. Sprawą zasadniczą jest to, że wiele spośród tych zagadnień zazębia się, toteż w celu dokonania poważniejszego postępu niezbędny jest prawidłowo skoordynowany wysiłek. Zbieranina dobrego, ale niejednolitego oprogramowania nie jest rozwiązaniem pożądanym. Aby więc taką koordynację osiągnąć postuluje się utworzenie ośrodka, który zająłby się pracami w tej dziedzinie.

Uważny przegląd literatury komputerowej pokazuje, że zagadnieniom konstrukcji maszyn cyfrowych, analizy numerycznej, języków formalnych i teorii automatów poświęcono, jako przedmiotom poważnych opracowań, znacznie więcej uwagi niż algorytmom komputerowym. Innym sposobem odnotowania braku należytego zainteresowania dla tych spraw jest przegląd biblioteki oprogramowania w większości ośrodków komputerowych, a także porównanie nakładów środków finansowych i siły roboczej na rozwój tego oprogramowania z nakładami na inne rodzaje prac ośrodka. Nawet w wiodących laboratoriach naukowych do oprogramowania matematycznego nie przywiązuje się należytej wagi. Można jednak również przytoczyć fakty świadczące o tym, że sytuacja ta ulega zmianie. Publikuje się coraz więcej dobrych algorytmów, a także dużo więcej uwagi poświęca się sprawie uzasadnienia poprawności i testowaniu algorytmów komputerowych. Firma IMSL podejmuję próby wpro-

---

[1] Lloyd D.Fosdick: The Production of Better Mathematical Software. ACM,. 1972, nr 7, s.611-617. Opracowanie: mgr Ewa Zawisza, Zakłady Elektronicznej Techniki Obliczeniowej,Warszawa.

wadzenia na rynek oprogramowania matematycznego wysokiej jakości. Jest to częściowo wynikiem zdania sobie sprawy z faktu, że wysokie koszty wiążą się właśnie z kiepskimi programami, takimi, które z powodu błędów nieoczekiwanie "zacinają się", takimi, które z powodu ubogiej dokumentacji nie pozwalają rozszyfrować, jak należy z nich korzystać oraz takimi, które w sposób zdecydowany a niejasny uzależnione są od jakiegoś określonego systemu tak, że podstawowym zadaniem staje się ich adaptacja do innego systemu. Coraz powszechniej uznaje się też fakt, że musimy rozumieć, jak należy budować bardzo duże programy, jeżeli chcemy efektywnie wykorzystywać przyszłe komputery. W tej chwili poważnym wysiłkiem jest napisanie programu zawierającego od 1000 do 5000 instrukcji. Możemy spodziewać się, że trzeba będzie zajmować się programami, których obszerność przewyższy dzisiejsze programy o kilka rzędów wielkości.

Niektóre problemy z zakresu oprogramowania wydają się nieciekawe od strony intelektualnej. Na przykład przygotowanie dobrej dokumentacji i określanie standardów nie wzbudza zainteresowania większości naukowców od spraw komputerów. Natomiast konstrukcja programu, jego analiza i określenie poprawności stwarzają większe pole do popisu niż dokumentacja i standardy. Czytelnik znajdzie dalej opis zagadnień i proponowanych rozwiązań zmierzających do rozwoju tych dziedzin.

Problemy, wobec których stajemy, powiązane są nawzajem w sposób oczywisty. Nie ulega wątpliwości, że użytkownicy oprogramowania widzą je inaczej niż producenci, którzy często woleliby nie dostrzegać praktycznych potrzeb użytkowników. Użytkownicy natomiast pragnęliby mieć zestaw programów, na którym można polegać, tj. obszerny, jednolity wewnętrznie i łatwy w użyciu. Zaspokojenie tej potrzeby wymaga dużego i skoordynowanego wysiłku, obejmującego prace we wszystkich wymienionych kierunkach. W dalszej części niniejszego artykułu przedstawiono propozycję utworzenia krajowego ośrodka oprogramowania, który mógłby pomóc w zaspokojeniu tej potrzeby.

Dokumentacja programów

Większość z nas zna tradycyjną dokumentację towarzyszącą programowi wraz ze wszystkimi jej niedoskonałościami. Wyprodukowano już najróżno-

rodniejsze poradniki sporządzania dokumentacji. Większość z nich pojawia się w postaci opracowań wewnętrznych, ale niektóre zostały opublikowane [1][2]. Pewne algorytmy opublikowane w czasopiśmie Communications of the ACM (Algorithms Department of Communications) mają wyjątkowo dobrą dokumentację, na przykład algorytmy 343 [3] i 395 [4]. Inne przykłady dobrej dokumentacji można znaleźć w zbiorze podprogramów algebry liniowej, przygotowywanym w ramach projektu NATS [5][1]. Zamiast jednak zajmować się stereotypowymi aspektami tego zagadnienia, poszukajmy innych pożytecznych rozwiązań. Po pierwsze zauważmy, że istnieją dwa rodzaje informacji, których użytkownik może potrzebować w związku z programem. Jeden z nich to te informacje, które można wydedukować z samego programu oraz z warunków, w których będzie on wykonywany, np. sposób wywoływania programu; drugi zaś rodzaj to te, których nie można w ten sposób wydedukować, np. fakt, że program oparto na algorytmie podanym przez J. Kowalskiego i omówionym w określonym czasopiśmie. Powinno się dążyć do tego, aby automatycznie uzyskiwać możliwie największą ilość informacji pierwszego rodzaju. Jednakże, poza automatycznym tworzeniem schematów blokowych, czemu poświęcono dużo uwagi w pracach [6][7][8][9] wydaje się, że niewiele więcej uczyniono w tym zakresie. Do daleko poważniejszych osiągnięć można i należałoby zmierzać w sprawie automatycznego wydobywania informacji z programu; należałoby zbadać możliwość uzyskiwania informacji dotyczących dopuszczalnych ograniczeń dla parametrów wejścia i wyjścia, czasu wykonania programu i warunków błędu. Automatyczne określanie dokładnych ograniczeń dla parametrów programu jest prawdopodobnie trudne do uzyskania, a w większości przypadków wręcz niemożliwe, ale jakaś pożyteczna informacja o tych ograniczeniach mogłaby być dostępna. Automatyczne precyzowanie informacji na temat czasu wykonania nie zawiera automatycznego oszacowania zbieżności procesów iteracyjnych; powinno jednak być możliwe uzyskanie informacji o czasie wykonania każdego cyklu iteracji.

---

[1] Celem tego projektu jest tworzenie i rozprowadzanie wysokiej jakości oprogramowania matematycznego z wybranych dziedzin. Projekt jest realizowany wspólnie przez wiele instytucji, przede wszystkim: Argonne National Laboratory, Standford University i University of Texas. Wiele uniwersytetów i laboratoriów, m.in. również w Stanach Zjednoczonych służy jako miejsca testowania dla programów będących wynikiem tych przedsięwzięć.

Różni użytkownicy często żądają różnych informacji o programie; z tej przyczyny wydaje się pożyteczne podjęcie próby dostarczenia informacji o programie za pomocą systemu "pytanie - odpowiedź", Przy takim systemie dostarczano by tylko tych informacji,które służą do identyfikacji programu i stanowią przedmiot ogólnego zainteresowania ze strony wszystkich potencjalnych użytkowników. Inne informacje byłyby uzyskiwane na podstawie pytań zgłaszanych do systemu z urządzenia końcowego. Prawidłowo skonstruowany system tego rodzaju pozwalałby indywidualnemu użytkownikowi na zadawanie bardzo szczegółowych pytań o programie, np. pytań o częstotliwość wykonywania części programu w przypadku zastosowania wybranych danych wejściowych, o ścieżki wykonania, o odwołania do pamięci, itp. Użytkownik mógłby, przy pewnych udogodnieniach redakcyjnych, wprowadzać do programu zmiany odpowiadające jego indywidualnym potrzebom.

Z systemu takiego mogliby również korzystać autorzy dokumentacji w celu gromadzenia i redagowania informacji drugiego rodzaju, np. zbiór pytań utworzonych przez komputer, takich jak: Imię i nazwisko autora?, Data ostatniej korekty? Odsyłacze? itd. oraz właściwe odpowiedzi udzielane przez człowieka przy urządzeniu końcowym spowodować musiałyby ujęcie dokumentacji w określone szablony, czyniąc ją w ten sposób bardziej jednolitą. Prowadzenie ewidencji pytań zgłaszanych przez użytkowników za pośrednictwem urządzeń końcowych mogłoby posłużyć do wyeksponowania problemów związanych z dokumentacją.

Jest oczywiste, że stworzenie takiego systemu wymagałoby poważnego wysiłku; na to, aby uzasadnić koszt jego opracowania, trzeba by znaleźć dla niego szerokie zastosowanie oraz zapewnić ciągłe utrzymywanie go w dobrym stanie.

Standardy

Jedną z największych przeszkód w rozprowadzaniu i wymianie oprogramowania jest brak standardów tak w zakresie sprzętu, jak i oprogramowania. Na przykład przeniesienie programu z jednego zestawu maszyny do innego za pomocą nośnika w postaci taśmy magnetycznej jest często nadmiernie skomplikowane z powodu niedopasowania sprzętu bądź oprogramowania (lub jednego i drugiego). Zarówno z tego jak i z innych powodów brak standar-

dów powoduje stratę czasu i pieniędzy oraz hamuje postęp. Prace zmierzające do rozwiązania problemów związanych ze standardami w zakresie sprzętu i oprogramowania prowadzi Komitet Sekcyjny X3 Amerykańskiego Instytutu Standardów Krajowych (American National Standards Institute Sectional Committee X3). Sprawozdania podkomitetów Komitetu X3 są publikowane w czasopismie Communications of the ACM. Przegląd prac z tego zakresu do 1967 r. można znaleźć w artykule T.B. Steela [10]. Standardy nie przynoszą efektów, jeżeli nie są używane. Wydaje się, że w chwili obecnej trudno zmusić producentów, aby przystosowali się do standardów, co szczególnie odnosi się do tych, którzy zapewniają zgodność konkurujących ze sobą produktów.

Pewna działalność z zakresu standardów, odnosząca się bezpośrednio do oprogramowania matematycznego, mogłaby i powinna być aktywnie prowadzona. Jest nią tworzenie, w postaci czytelnej dla maszyny, tablic wartości funkcji elementarnych i nieelementarnych. Przy sprawdzaniu algorytmu numerycznego, np. dla obliczania wartości funkcji Bessela określonego rodzaju, trzeba posługiwać się tablicami w celu zweryfikowania liczb. Z takiego rozwiązania wynikają oczywiste kłopoty. Można się upierać, że tablice takie nie są niezbędne, bowiem zamiast nich można używać bardzo dokładnie sprawdzonych programów standardowych. Trudność polega na tym, że wynik obliczenia zależy od maszyny i związanego z nią oprogramowania, a czynniki te nie są stałe. Kto wie, czy nie będzie możliwe tworzenie programów, dla których można by zagwarantować wyniki niezależne od tych czynników; lecz nawet jeżeli odpowiedź na to pytanie jest twierdząca - nie nastąpi to szybko. Miały miejsce pewne odosobnione próby stworzenia tablic na taśmie magnetycznej: jedne podjęte w Krajowym Biurze Standardów (National Bureau of Standards) [12], drugie zaś prowadzone w związku z projektem NATS [5]. Sens ma jednak dopiero wysiłek podejmowany szerszym frontem. Sukces wymaga, aby w celu zapewnienia jednolitości materiału projekt taki był dobrze skoordynowany. Pożądany byłby wspólny nakład pracy, podobny do tego, który dał w wyniku "Handbook of Mathematical Functions" [13], stanowiący podręczny zbiór funkcji matematycznych.

Rozwiązania wymaga wiele zagadnień związanych z tworzeniem standardowych tablic: rząd dokładności wyników, wybór argumentów, system zapisu (np. dwójkowy, dziesiętny) itd. Wydaje się, że właściwy rząd dokład-

ności, to odpowiednik ok ło pięćdziesięciu miejsc dziesiętnych oraz, że należałoby przyjąć zapis dwójkowy, a nie dziesiętny. Można oczekiwać, że wybór argumentów będzie zależeć od zmienności funkcji i w ten sposób może zostać powiązany z jej modułem ciągłości. Pewne argumenty funkcji, np. argumenty odpowiadające jej wartościom ekstremalnym i zerowym, bywają szczególnie interesujące. Te i podobne im zagadnienia - podstawowe dla utworzenia takich tablic na taśmie magnetycznej, wymagają dokładnych badań, przy czym pożądane byłyby w tym zakresie uzgodnienia pomiędzy potencjalnymi użytkownikami.

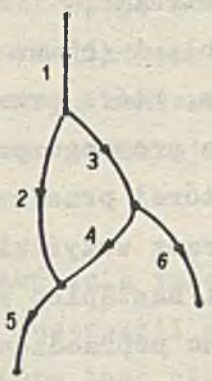
Poprawność programów

Użytkownikom programów komputerowych trzeba zapewnić sensowną gwarancję tego, że mogą na programie polegać. Prowadzi się wiele badań z zakresu dowodzenia poprawności programów; jedne z najbardziej znanych prac z tej dziedziny opublikowali autorzy: McCarthy [14], McCarthy i Painter [15], Floyd [16], Hoare [17], Dijkstra [18] oraz Naur [19] [20]. Bibliografię prac z tego zakresu sporządził London [21] [22]. Większość tych prac ma charakter teoretyczny i jakkolwiek daje się zauważyć poważny postęp, to dzieli nas jeszcze sporo od praktycznego ich celu - uzasadnienia poprawności, np. programu rozwiązującego równania różniczkowe. Praca Hulla [23] na temat poprawności oprogramowania matematycznego jest wyjątkiem. Hull zajmuje się bowiem bezpośrednio zagadnieniem poprawności programów z dziedziny algebry liniowej i równań różniczkowych.

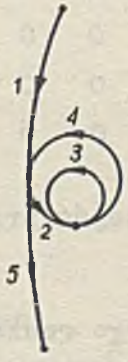
W odniesieniu do wszystkich programów można by podjąć pewne, stosunkowo proste kroki, które pomogłyby zapewnić ich niezawodność. Jednym z takich kroków byłoby zagwarantowanie, że każda instrukcja w programie była wykonana co najmniej raz podczas sprawdzania danego programu za pomocą testów. Przy założeniu, że informacje tego rodzaju podawane będą w sposób jasny, konieczne staje się tylko zidentyfikowanie wszystkich rozgałęzień w programie i dołączenie do niego rozkazu ustalającego wartość odpowiedniego wskaźnika. Program, który wykonuje to zadanie dla programów w Fortranie, używając liczników zamiast wskaźników, omówiono w pracy [24]. Do krótkich programów takie instrukcje ustalające wskaźniki - można łatwo wprowadzić ręcznie, lecz może to spowodować popełnienie błędu przez człowieka. Mimo jednak, że potrzeba jakiegoś minimalnego

sprawdzianu wydaje się czymś oczywistym i mimo uwag, które poświęcił temu zagadnieniu Wegstein [25], żadne opracowanie spośród ponad dwustu otrzymanych przez dział algorytmów w Communications of the ACM w okresie przeszło dwóch lat nie zawierało informacji o wykonaniu tego rodzaju testów.

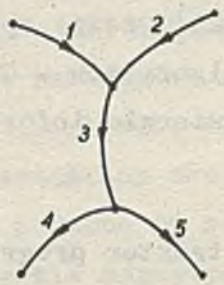
W przypadku prostych struktur programowych wykonanie każdej instrukcji co najmniej raz podczas testowania programu sugeruje, że nastąpiło przejście przez wszystkie jego ścieżki. Przypadek taki zilustrowano na rys. 1. Inny przypadek, w którym przyjmujemy założenie, że nie rozróżniamy między jednym a większą liczbą przejść przez ścieżkę, natomiast przejście przez pętlę musi nastąpić przynajmniej raz (tak jak to ma miejsce w odniesieniu do pętli DO w Fortranie ANSI), przedstawiono na rys. 2. Powinno być oczywiste, że właściwość tę posiada struktura drzewa. Z drugiej strony prosta struktura programowa - taka, jaką pokazano na rys. 3, nie ma tej właściwości: jest oczywiste, że przejście przez ścieżki (a), (b) i (c) ustali wartości wszystkich wskaźników.



Rys. 1. Struktura programu o ścieżkach: (a) 1, 2, 5; (b) 1, 3, 6; (c) 1, 3, 4, 5



Rys. 2. Struktura programu o ścieżkach (a) 1, 5; (b) 1, 2, 4, 5; (c) 1, 2, 3, 4, 5



Rys. 3. Struktura programu o ścieżkach: (a) 1, 3, 4; (b) 2, 3, 5; (c) 1, 3, 5; (d) 2, 3, 4

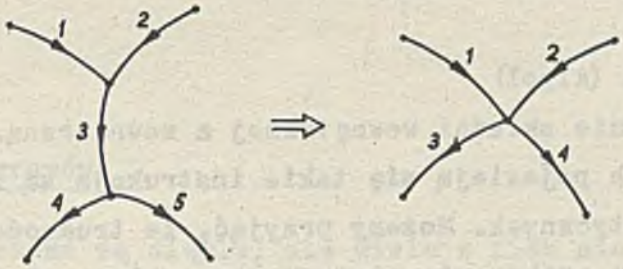
Inna metoda stosowana przy sprawdzaniu programu polega na identyfikacji przejść przez pary gałęzi. Tak więc ścieżka (a) na rys. 3 prowadzi przez pary [1, 3] i [3, 4], ścieżka (b) - przez pary [2, 3] i [3, 5], itd. Przejścia te mogą być identyfikowane za pomocą macierzy przejść tak, jak to pokazano na rys. 4, gdzie elementami macierzy są liczby 0 lub 1, przy czym liczba 1 na pozycji [i, j] reprezentuje przejście przez parę gałęzi [i, j]. Jest zrozumiałe, że ograniczenie się do identyfikacji przejść przez pary gałęzi na ogół nie pozwoli sprawdzić przejść przez wszystkie ścieżki struktury. Przejścia przez pary gałęzi wygenerowane ze ścieżek (a), (b), i (c) na rys. 3 dałyby w rezultacie taką samą macierz przejść jak ta, którą przedstawiono na rys. 4. Należy jednak zauważyć, że po dokonaniu prostego przekształcenia pokazanego na rys. 5 otrzymamy strukturę, w której przejście przez wszystkie pary gałęzi daje gwarancję przejścia przez wszystkie ścieżki. Zmianę taką sugeruje oczywisty fakt, że jeżeli nastąpiło przejście przez gałęzie 1 i 4 na rys. 3, to musiało być ono poprzedzone przejściem przez gałąź 3.

0	0	1	0	0
0	0	1	0	0
0	0	0	1	1
0	0	0	0	0
0	0	0	0	0

Rys. 4. Macierz przejść dla przejść przez pary gałęzi: [1, 3] [2, 3], [3, 4], [3, 5]

Wydaje się zrozumiałe, że tego rodzaju próby mogą być w różny sposób rozszerzane oraz można z nich korzystać w celu sprawdzania programów. Byłoby dobrze z krótkiej informacji: przejścia przez pary gałęzi, itp.

- móc uzyskać pełną informację, na przykład taką, że testy przeszły przez wszystkie ścieżki programu. W jakim stopniu tego rodzaju podejście może okazać się skuteczne w praktyce, zależy od struktury rzeczywistego programu, o której bardzo niewiele wiadomo. Jeden z eksperymentów tego rodzaju, opisany przez Knutha [26], związany był przede wszystkim z częstotliwością użycia instrukcji i wyrażeń w Fortranie.



Rys. 5. Struktura z rys. 3 przekształcona przez usunięcie gałęzi 3

Inna próba rozwiązania tego samego problemu polega na połączeniu czynności określania algorytmu ze sprawdzaniem jego poprawności. Dijkstra [18] próbował wykorzystać tę koncepcję do skonstruowania pewnego systemu programowania. Bardziej sformalizowane ujęcie tego zagadnienia przedstawił Floyd [27], opisując metodę konwersacyjną układania poprawnego programu na przykładzie programu odszukującego symbol w posortowanej tablicy.

Analizę programu, służącą do sprawdzenia jego poprawności, można by uprościć wprowadzając rozróżnienie instrukcji na te, które sterują ścieżką wykonania programu i te, które tego nie robią. Wilkes [28] nazywa instrukcje pierwszej grupy składnią zewnętrzną, instrukcje zaś grupy drugiej - składnią wewnętrzną. Stąd instrukcje takie jak:

GO TO 55 (Fortran)

lub

go to L; (Algol)

oraz

DO 15 J = 1,25 (Fortran)

lub

for j: = 1 step 1 until 25 do (Algol)

należą do składni zewnętrznej, instrukcje zaś takie jak

X = Y + 5.0 x Z (Fortran)
lub
x: = y + 5.0 x z; (Algol)
należą do składni wewnętrznej. Rozróżnienie to pozwala podzielić analizę programu na potencjalnie łatwiejsze do opracowania fragmenty. Jednakże instrukcje typu
IF (X.LT.12.5) GO TO 15 (Fortran)
lub
if x < 12.5 then do (Algol)
wprowadzają połączenie składni wewnętrznej z zewnętrzną. Te miejsca w programie, w których pojawiają się takie instrukcje są szczególnie trudne do badań diagnostycznych. Możemy przyjąć, że trudności występujące podczas analizowania zachowania się programu zależą od liczby takich punktów.

Podczas zajmowania się składnią wewnętrzną wypływa sprawa analizy błędu zaokrąglenia. Analizę tę można podzielić na dwie części. W jednej z nich operacje maszyny $\oplus$, $\ominus$ itd. są zastępowane przez operacje dokładne + , − , itd. zgodnie z formalnymi regułami, np.
$x \oplus y = (x + y)\ (1 + \varrho)$
gdzie ϱ jest parametrem, którego dokładna wartość jest nieznana, ale spełnia nierówność

$$|\varrho| < \varepsilon$$

gdzie liczba ε jest znana i zależy od błędu zaokrągleń danej maszyny. Za pomocą tych reguł i kilku kluczowych twierdzeń można wyrażenia zawierające operacje maszyny przekształcać na wyrażenia zawierające dokładne operacje oraz czynniki zależne liniowo od parametru ϱ , którego dokładna wartość znowu nie jest znana, ale jest ograniczona, podobnie jak w nierówności przedstawionej powyżej. Przekształcanie to jest niezwykle proste, choć zdecydowanie nudne i mogłoby być wykonywane przez maszynę. Druga część analizy błędów dotyczy ustalania górnych ograniczeń wartości wyników pośrednich, po to aby na podstawie pewnego wyboru twierdzeń można było uzyskać ograniczenie błędu dla ostatecznego wyniku obliczeń maszynowych. Nie jest w chwili obecnej sprawą oczywistą, jak można by tę drugą część zrealizować w sposób automatyczny; nie ulegą jednak wątpliwości, że praca, której wymaga cały przedstawiony proces analizy, mogłaby być uproszczona, gdyby przynajmniej jego pierwsza część została

przekazana do wykonania maszynie. W celu szacowania błędów wtórnych można by pominąć drugą część analizy, a zamiast tego zlecić maszynie tworzenie zapisów wartości ekstremalnych dla kluczowych wyników pośrednich podczas wykonywania obliczeń. Takie postępowanie wymagałoby mniejszego nakładu pracy podczas wykonywania programu niż stosowanie arytmetyki przedziałowej lub arytmetyki cyfr znaczących; posiada ono ponadto cenną zaletę w postaci tego, że obejmuje co najmniej część wstępnej analizy błędów.

Konstrukcja programów

Programy tworzone są ciągle, ale wiele z nich nie wnosi niczego nowego pod żadnym względem. Zbyt często ich autorzy nie interesują się osiągnięciami innych twórców programów. Można tę wypowiedź potraktować jako uwagę na temat autorów; ale z całą pewnością jest to uwaga na temat mechanizmów przekazywania informacji o programach. Myślę tu nie tyle o publikowaniu programów, ile o sposobach do jakich uciekamy się wówczas, gdy mówimy o programach i opisujemy je dla potrzeb innych osób, zainteresowanych nimi z różnych względów. Oczywiście w celu zrozumienia programu posiadanie jego tekstu jest sprawą zasadniczą, ale zazwyczaj potrzeba jeszcze czegoś więcej. Nasza niedoskonałość w przekazywaniu informacji o programach znajduje widoczny wyraz w fakcie, że programiści często wolą napisać własny program niż podejmować próby wykorzystania i zrozumienia czyjegoś programu. Wiele z tego, co można i należałoby zrobić w tej dziedzinie omówiono powyżej, poruszając zagadnienia związane z dokumentacją. Z tematem tym wiąże się też przenośność programu, czyli możliwość użycia go w odmiennych warunkach: inni ludzie, inne oprogramowanie, inny sprzęt.

Istnieją proste, dość elementarne metody, których stosowanie podczas pisania programów w językach standardowych zwiększa możliwości ich przenoszenia. Jedna z nich polega na zgrupowaniu w jednym miejscu w programie wszystkich parametrów zależnych od maszyny, zidentyfikowaniu tego ich charakteru i podaniu zasad ich zmiany w przypadku zmiany warunków maszynowych. Programy często mają parametry sterujące przydziałem pamięci, czasem wykonania i dokładnością. Parametry te również powinny być zebrane razem, zidentyfikowane i opatrzone informacją podającą zasady ich zmiany. Mogłoby to stanowić pomoc dla użytkownika wówczas, gdyby

chciał jedną z tych właściwości poświęcić na rzecz drugiej, na przykład szybkość - na rzecz dokładności.

Powinna istnieć możliwość układania niektórych programów w taki sposób, aby podstawowe parametry szybkości, pamięci i dokładności można było dopasowywać automatycznie. Nie wydaje się jednak, aby podejmowane były jakiekolwiek poważne próby w tym kierunku. Bezspornym miejscem, gdzie zastosowanie metody tego rodzaju wydawałoby się pożyteczne jest aproksymacja funkcji. Załóżmy, że jakaś funkcja $f(x)$ ma być aproksymowana przez inną funkcję $\bar{f}(x)$ z zadanym błędem $\epsilon(x)$; wobec tego $f(x) = \bar{f}(x) + \epsilon(x)$.
$\bar{f}(x)$ mogłaby być na przykład podane jako funkcja wymierna bądź w postaci tablicy lub też w jednej i drugiej postaci. Wyobraźmy sobie teraz formalny opis algorytmu obliczania wartości $\bar{f}(x)$, w którym nie określono szczegółów odnoszących się do konkretnej realizacji. Na przykład gdyby $\bar{f}(x)$ była funkcją wymierną, wówczas wielomiany licznika i mianownika mogłyby pozostać nieokreślone. Taki formalny opis algorytmu będziemy nazywali metaalgorytmem. Program obliczania wartości $\bar{f}(x)$ za pomocą konkretnej maszyny i przy określonych parametrach szybkości, pamięci i dokładności można uzyskać z metaalgorytmu w sposób następujący: specjalny rodzaj translatora - nazwijmy go mapistą - przeczyta metaalgorytm razem z danymi precyzującymi warunki dotyczące szybkości, pamięci i dokładności; następnie albo utworzy program obliczania wartości $f(x)$ przy spełnieniu podanych warunków, albo poda komunikat, że nie mógł wygenerować programu spełniającego te warunki. Tę ambitną metodę można zrealizować tylko częściowo. Jej najwłaściwsza realizacja może mieć miejsce w procesie współdziałania człowieka z maszyną w taki sposób, że człowiek rozstrzyga okoliczności nieprzewidziane, bądź takie, w przypadku których aktualnie nie istnieje rozwiązanie w postaci algorytmu.

Rozwój nowej arcnitektury maszynowej, takiej jak procesory równoległe, procesory szeregowe i różnorodne rodzaje struktur pamięciowych, powoduje tworzenie nowych algorytmów uwzględniających korzyści, które ta architektura daje. Więcej wysiłku powinno się jednakże poświęcać działaniom w kierunku odwrotnym, a więc znajdowania najlepszej architektury dla określonych algorytmów. W rzeczywistości, powinniśmy elastycznie traktować zarówno architekturę, jak i algorytmy - poszukując najlepszej ich kombinacji dla jakiejś klasy problemów. I tutaj znowu poważną przeszkodą jest słaba znajomość programów będących w powszechnym użyciu. Nie jest najbardziej racjonalnym rozwiązaniem zastosowanie dużego procesora równoległego do prac nad obliczeniami dla reaktora,

jeżeli tylko 30% tych obliczeń dotyczy rozwiązywania równań różniczkowych na pewnej siatce metodą różnic skończonych, a pozostałe 70% wiąże się z wyprowadzaniem wyników w sposób nie wykorzystujący sprawności procesora równoległego.

Wracając teraz do konstrukcji określonych rodzajów algorytmów, zbadanie indeksu CALGO [29] ukazuje widoczne obszary słabości, np. wykaz algorytmów dla rozwiązywania równań całkowych i różniczkowych jest zadziwiająco krótki. Algorytmy dla aproksymacji funkcji, o więcej niż jednej zmiennej, prawie nie istnieją. Algorytmy dla funkcji nieelementarnych mogłyby być udoskonalone. W związku z tym rozważana była sprawa opracowania zbioru algorytmów dla funkcji nieelementarnych sporządzonego analogicznie czy też równolegle do podręcznika Handbook of Mathematical Functions [13].

Ośrodek oprogramowania

W artykule niniejszym podano kilka propozycji w sprawie przyszłych prac w dziedzinie oprogramowania, zwłaszcza oprogramowania matematycznego. Są one podzielone na cztery grupy: dokumentacja, standardy i projektowanie. Między tymi grupami istnieją silne powiązania; każdy, kto zamierza poważny nakład pracy przeznaczyć w celu stworzenia dobrego oprogramowania, nie może tego nie dostrzegać. Ze względu na to, że tworzenie dobrego oprogramowania jest trudne, kosztowne i wymaga długotrwałych prac, właściwym posunięciem byłoby utworzenie ośrodka, który zająłby się tą dziedziną. Pomysł ten nie jest niczym nowym; propozycję taką wysunął ostatnio Rice [30], przed nim zaś J. Schwartz proponował utworzenie ośrodka badań w dziedzinie informatyki. Rice wspominał, że prace nad oprogramowaniem są działalnością o niskim statusie zawodowym i dlatego nie przyciągają one zdolnych ludzi. Jest to prawda, ale istnieje też inny powód tego, że dobre kadry nie interesują się tymi pracami. Naukowcy chcieliby wiedzieć, że prace ich wywierają jakiś wpływ, z założenia pozytywny. Niestety brak koordynacji i standardów w tej dziedzinie pociąga duże prawdopodobieństwo tego, że wysiłek ludzki skierowany na stworzenie dobrego oprogramowania pozostanie nieznany i nie wykorzystany, z wyjątkiem, być może, ośrodków lokalnych. I to właśnie wpływa hamująco na wysiłki fachowców. Ośrodek skupiający całość działalności związanej z opro-

gramowaniem mógłby pomóc w przezwyciężeniu tego problemu. Na przykład gdyby jednym z zadań ośrodka było opracowywanie, utrzymywanie i upowszechnienie biblioteki oprogramowania matematycznego, wówczas zaistniałyby potencjalne warunki do uznania i wykorzystania efektów prac twórców oprogramowania.

Nie wydaje się, aby trzeba było wyliczać wszystkie korzyści, jakie mieliby z takiego ośrodka użytkownicy oprogramowania, ale kilka z nich przytaczamy. Centralna biblioteka oprogramowania matematycznego wysokiej jakości pozwoliłaby zmniejszyć ogromne straty spowodowane powielaniem tych samych prac zarówno w prywatnych, jak i ogólnie dostępnych ośrodkach. Można by wówczas przeznaczać większe nakłady na ulepszanie metod sprawdzania poprawności oraz na dokumentację, które są obecnie tak bardzo zaniedbywane. Do ośrodka można by się zwracać w celu sprawdzania, czy w oprogramowaniu wytworzonym gdzie indziej uwzględniono standardy testowania i zgodności. Warto też wziąć pod uwagę potężny potencjalnie czynnik wpływu takiego ośrodka na wszystkie prace z zakresu oprogramowania, wynikający z jego wiodącej roli.

Istnieją też pewne praktyczne aspekty roli takiego ośrodka, które zgodnie z moim odczuciem należą do istotnych. Ośrodek powinien umożliwić łatwą wymianę informacji i ludzi. W związku z tym jego praca nie powinna być w żaden sposób utrudniana przez zastrzeżoną prawnie tajemnicę, jaka związana jest z interesami prywatnymi. Co więcej, ośrodek musi być nasycony tym rodzajem stymulacji intelektualnej, jaką znaleźć można na uniwersytetach. Nikt nie może dać recepty na zagwarantowanie tego stanu rzeczy, ale szanse na to, że stymulacja intelektualna znajdzie tam swoje miejsce można znacznie zwiększyć, jeżeli dokona się konkretnego wysiłku w kierunku zapewnienia łatwej wymiany ludzi. Około 50% lub więcej pracowników takiego ośrodka mogłoby być zatrudnionych na zasadzie jednorazowych kontraktów długoterminowych, np. dwuletnich, bądź krótkoterminowych, np. jednomiesięcznych. Pracownicy ci pochodziliby z uniwersytetów, laboratoriów państwowych i z przemysłu prywatnego. Istnieje jeszcze jedno uzasadnienie dla prowadzenia takiej polityki. Należy przewidywać, że możliwości omawianego ośrodka byłyby dużo bardziej rozległe i potężne niż możliwości osiągalne gdzie indziej. Pracownicy zaproszeni z innych ośrodków mogliby zatem korzystać z udogodnień niełatwo dostępnych im w inny sposób. Oczywiście niektóre z tych możliwości mogłyby i powinny być osią-

gane za pomocą sieci maszyn. Sieci maszyn są jednak bardzo kosztowne, a to uniemożliwi przez pewien czas w sieci o dużej liczbie stacji końcowych instalowanie urządzeń dużo lepszych niż urządzenia końcowe o powolnym przesyłaniu danych, jakimi są dalekopisy.

Wydaje się mało prawdopodobne, aby fundusze na finansowanie takiego ośrodka mogły pochodzić z innych źródeł niż rząd. Prawdopodobnie można by czerpać pewne fundusze ze źródeł prywatnych na stypendia oraz wspomniane jednorazowe kontakty. Można przypuszczać, że oprogramowanie stworzone przez ośrodek będzie sprzedawane w celu pokrycia pewnych kosztów. Nie jest jednak oczywiste, czy powinno to mieć miejsce.

Proponowany ośrodek byłby związany z jedną dziedziną obliczeń, tj. z oprogramowaniem matematycznym. Można by włączyć do jego działalności inne dziedziny lub też można by stworzyć sieć ośrodków, z których każdy specjalizowałby się w innej dziedzinie. Utworzenie ośrodka oprogramowania matematycznego wydaje się być w każdym razie rozsądnym krokiem.

Literatura

[1] HOWARTH R.J., LIM A.L.: An Approach to Program Documentation. Comp. Bull. 13, 1969, s. 291-295.

[2] WALSH D.: A Guide for Software Documentation. Advanced Computer Techniques Corp., 1969.

[3] GRAD J., BREBNER M.A.: Algorithm 343, Eigenvalues and Eigenvectors of a Real General Matrix. Comm. ACM, 11, 1968, nr 12, s. 820-826.

[4] HILL G.W.: Algorithm 395, Student's t-distribution. Comm. ACM, 13, 1970, nr 10, s. 617-619.

[5] NATS Project. SIGNUM Newsletter, 6, 1971, nr 3, s. 5.

[6] KNUTH D.: Computer-drawn flowcharts. Comm. ACM, 6, 1963, nr 9, s. 555-563.

[7] SHERMAN P.M.: Flowtrace, a Computer Program for Flowcharting Programs. Comm. ACM, 9, 1966, nr 12, s. 845-854.

[8] OBRIEN F.; BECKWITH R.C.: A Technique for Computer Flow Chart Generation. Comp. J., 11, 1968, s. 138-140.

[9] AUTFLOW. Applied Data Research, Princeton, N.J.

[10] STEEL T.B.: Standards for Computers and Information Processing. W: Advances in Computers, t.8, New York, Academic Press, s. 47-152.

[11] BROOKS J.: Letter to Charles L. Schultze, Director Bureau of the Budget. Comm. ACM, 11, 1968, nr 1, s. 55-56.

[12] SADOWSKI W.L., MAXIMON L., LOZIER D.W.: A Bit Comparison Program for Algorithm Testing. Approximators Workshop, Argonne National Lab., 1971.

[13] ABRAMOWITZ M., STEGUN I.: Handbook of Mathematical Functions. AMS, 55, National Bureau of Standards, Washington, D.C.

[14] McCARTHY J.: A Basis for a Mathematical Theory of Computation. W: Computer Programming and Formal Systems, Amsterdam 1963, North-Holland Pub. Co., s. 33-70.

[15] McCARTHY J., PAINTER J.: Correctness of a Compiler for Arithmetic Expressions. W: Proceedings of Symposia in Applied Mathematics, t. 19, 1966, s. 33-41.

[16] FLOYD R.W.: Assigning Meanings to Programs. W: Proceeding of Symposia in Applied Mathematics, t. 19, 1966, s. 19-32.

[17] HOARE C.A.R.: An Axiomatic Basis for Computer Programming. Comm. ACM 12, 1969, nr 10, s. 576-580; 583.

[18] DIJKSTRA E.W.: A Constructive Approach to the Problem of Program Correctness. BIT 1968, nr 8, s. 174-186.

[19] NAUR P.: Proof of Algorithms by General Snapshots. BIT 1966, nr 6, s. 310-316.

[20] NAUR P.: Programming by Action Clusters. BIT 1969, nr 9, s. 250-258.

[21] LONDON Ralph L.: Bibliography on Proving the Correctness of Computer Programs. W: Machine Intelligence, 5, 1970, s. 569-580.

[22] LONDON Ralph L.: Bibliography on Proving the Correctness of Computer Programs - Addition no 1. U. Wisconsin, Computer Science Dept., Report No 104, 1970, s. 1-8.

[23] HULL T.J., ENRIGHT W.H., SEDGWICK A.E.: The Correctness of Numerical Algorithms, SIGPLAN Notices 7, 1 and SIGACT News, 14, 1972, s. 66-73.

[24] INGALLS D.H.H.: FETE, a Fortran Execution Time Estimator. Stanford, U., Dept. of Computer Science, Report No 204, 1971, s. 1-10.

[25] WEGSTEIN J.H.: Announcement of Algorithms Department. Comm. ACM, 3, 1960, nr 1, s. 73.

[26] KNUTH D.E.: An Empirical Study of Fortran Programs. Software 1971, nr 1, s. 105-133.

[27] FLOYD R.W.: Toward Interactive Design of Correct Programs. Stanford U., Dept. of Computer Science, Report No 235, 1971, s. 1-12.

[28] WILKES M.V.: The Outer and Inner Syntax of a Programming Language Computer J., 1968, nr 11, s. 260-263.

[29] CALGO, Collected Algorithms from CACM, ACM, New York.

[30] RICE J.R.: The Distribution and Sources of Mathematical Software. W: Mathematical Software, New York 1971, Academic Press, s. 27-41.

KRÓTKIE INFORMACJE

Z KRAJU

• II SYMPOZJUM KLUBU UŻYTKOWNIKÓW ELEKTRONICZNYCH MASZYN CYFROWYCH JEDNOLITEGO SYSTEMU

14 marca br. w Katowicach odbyło się II Sympozjum Klubu Użytkowników JS EMC, zorganizowane przez Klub Użytkowników JS EMC, Oddział Śląski Instytutu Maszyn Matematycznych w Katowicach oraz Oddział Wojewódzki Polskiego Komitetu Automatycznego Przetwarzania Informacji. Było to drugie z kolei Sympozjum na temat Jednolitego Systemu Elektronicznych Maszyn Cyfrowych ze szczególnym uwzględnieniem EMC R-30; pierwsze odbyło się we Wrocławiu 12.XII.1972 r., ponieważ jednak liczba zainteresowanych przekroczyła znacznie możliwość jednorazowej ich obsługi, wydało się pożyteczne powtórzenie tego samego tematu.

Około 250 uczestników wysłuchało informacji o celach pracy Klubu Użytkowników oraz sześciu referatów problemowych (w tym pięciu przygotowanych i wygłoszonych przez przedstawicieli WZE ELWRO/)o tytułach takich samych jak na I Sympozjum i zawierających jednak w porównaniu z tamtymi wiele nowych i istotnych szczegółów.

Dotyczy to przede wszystkim informacji o systemach operacyjnych DOS EMC JS R-30 (referował mgr R. Tryba), OS EMC JS (referat wygłosił dr A. Bukowy z IMM OŚ1) oraz emulacji MC serii ODRA 1300 w Jednolitym Systemie (mgr Thanasis Kamburelis).

W dyskusji poruszono m.in. sprawy możliwości dzierżawy sprzętu u producenta (WZE ELWRO), nowych form organizacji obsługi sprzętu, dostaw i aktualizacji oprogramowania, standardowego oprogramowania JS itp.

Dalsze plany Klubu Użytkowników przewidują zorganizowanie w IV kwartale br. sympozjum na temat systemów wielodostępnych realizowanych na komputerach JS.

(ab)

• CYFROWA JEDNOSTKA STERUJĄCA TYP CJS-72

Cyfrowa jednostka sterująca CJS-72 przeznaczona jest do sterowania urządzeń technologicznych i obrabiarek specjalnych. Realizuje ona w sposób programowy ustawienie stołu krzyżowego, uruchamia i steruje pracą głowicy roboczej oraz steruje pracą mechanizmów pomocniczych.

Schematy logiczne zbudowane są całkowicie na układach scalonych, zasilacze i układy wyjściowe na elementach krzemowych. Jednostka może sterować stołem napędzanym silnikami krokowymi lub silnikami o ruchu ciągłym w układzie otwartym lub w pętli zamkniętej z impulsowymi przetwornikami analogowo-cyfrowymi.

W jednostkę wbudowany jest rewersyjny czytnik taśmy RCT-250 - fotoelektryczny z tarciowym napędem taśmy i możliwością dwukierunkowej pracy. Czytnik wyposażony jest w szpule oraz mechanizmy automatycznego podawania i zwijania taśmy.

Pulpit operacyjny umieszczony na przedniej ścianie jednostki zawiera cyfrowe wskaźniki współrzędnych stołu oraz klawiaturę do ręcznego sterowania, kontroli i początkowych nastaw rodzaju pracy.

Zasilacze silników stanowią samodzielny zespół dostarczany wraz z jednostką i wykonany w jednym z odpowiednich wariantów. Modułowe rozwiązanie układów sterujących daje możliwość łatwego wprowadzania zmian i uzupełnień w celu spełnienia różnorodnych postulatów użytkownika.

Podstawowe dane techniczne:

• cyfrowy zakres współrzędnych X i Y		- 5 dekad
• czytnik taśmy - liczba ścieżek		- 8 + 1 prowadząca
•	- prędkość czytania	- 250 zn/s
	- pojemność szpul	- 100 metrów taśmy
• kody wejściowe		- ISO ASCII lub EIA
• zasilanie		- 220V 50 Hz 300W

CJS-72 zbudowano w Zakładzie Automatyzacji Produkcji Urządzeń Cyfrowych Instytutu Maszyn Matematycznych.

(jg)

• REWERSYJNY CZYTNIK TAŚMY RCT-250

Przeznaczenie i zasada działania

Rewersyjny czytnik taśmy RCT-250 przeznaczony jest do odczytu informacji zapisanych na papierowej taśmie dziurkowanej z możliwością odczytu przy dwóch kierunkach przesuwu taśmy. Czytnik RCT-250 przystosowany jest do wbudowania w szafy jednostek sterujących lub innych urządzeń automatyki cyfrowej.

Odczyt realizuje się metodą fotoelektryczną. Posuw taśmy dokonuje się za pomocą rolki napędowej na zasadzie tarcia bez wykorzystania perforacji ścieżki prowadnikowej. W czasie odczytu taśma porusza się ruchem ciągłym. Uruchomienie i zatrzymanie następuje na sygnały START i STOP z zewnętrznych współpracujących układów elektronicznych. Po sygnale STOP taśma zatrzymuje się na drodze mniejszej niż odległość do następnej dziurki prowadnikowej - niezależnie od kierunku przesuwu. Czytnik wyposażony jest w szpule i mechanizmy do nawijania odczytanej taśmy i do szybkiego przewijania.

Układy elektroniczne czytnika zbudowane są na elementach krzemowych. Wejścia i wyjścia przystosowane są do współpracy ze sterowaniem rozwiązanym na układach scalonych TTL.

Czytnik odznacza się bardzo cichą pracą mechanizmów. Jest prosty w budowie, łatwy w obsłudze i konserwacji.

Podstawowe dane techniczne:

- liczba ścieżek informacyjnych - 8 + 1 prowadząca
- prędkość czytania max. - 250 zn/s
- kanały wyjściowe: dziurka - 0 - 0,5 V
  brak dziurki - + 5V
  obciążalność wyjść wzmacniaczy - 10 wejść układu TTL
- kanały wejściowe
  a) kanał posuwu START - + 5 V
  STOP - 0 V
  b) kanał kierunku W PRZÓD - + 5 V
  W TYŁ - 0 V
- sygnał wejściowy jest obciążony sześcioma jednostkami obciążenia dla układów TTL
- impulsy dziurki prowadzącej ustawione symetrycznie względem impulsów ścieżek informacyjnych

czas trwania - min. 0,5 ms
- max. 10 ms

. przewijanie taśmy
prędkość - 2,5 m/s
pojemność szpuli - 100 m
. zasilanie - +5V ± 2% 1,2 A
- +12V 3 A
. wymiary
płyta czołowa - 390 x 180
głębokość za płytą czołową - 180
przed płytą czołową - 45
wkręty mocujące M5 na płycie czołowej - 380 x 120

nadaje się do wbudowania w szafy standardowe o module szerokości 19".

Urządzenie skonstruowano w Zakładzie Automatyzacji Produkcji Urządzeń Cyfrowych Instytutu Maszyn Matematycznych.

(jg)

• AUTOMATYCZNY STÓŁ KRZYŻOWY typ ASK-500

Automatyczny stół krzyżowy ze sterowaniem programowym jest zespołem o uniwersalnym przeznaczeniu i przewidziany jest jako główna część składowa różnorodnych urządzeń technologicznych i pomiarowych dla przemysłu informatyki, elektroniki, mechaniki precyzyjnej i innych.

Mechanizm krzyżowy x-y zamocowany jest na sztywnej podstawie. Członem ruchomym jest wózek służący do umieszczenia przedmiotu obrabianego lub głowicy roboczej. W rozwiązaniu zespołów kinematycznych wykorzystano podzespoły toczne - prowadnice i śruby kulkowe.

Stół napędzany jest silnikami krokowymi lub silnikami prądu stałego o ruchu ciągłym. Zasilacze silników wbudowane są w podstawę stołu. W podstawie przewidziane jest również wolne miejsce na elementy i układy zasilające zainstalowanej na stole głowicy roboczej.

Sterowanie realizuje cyfrowa jednostka sterująca CJS-72.

Podstawowe dane techniczne:

. powierzchnia stołu - 640 x 920 mm
. pole robocze zakresy ruchu wózka - 400 x 500 mm

- wymiary wózka - 150 x 140 mm
- działka elementarna - 0,025 mm
- prędkość - 3 m/min
- zasilanie - 220V 50 Hz

Przykłady zastosowań

- wiercenie płytek drukowanych
- wiercenie otworów w płaskich elementach metalowych
- montaż elementów w płytkach drukowanych
- programowane trasowanie wymiarów
- mikroobróbka lub mikromontaż
- pomiary precyzyjne elementów płaskich

Urządzenie skonstruowano w Zakładzie Automatyzacji Produkcji Urządzeń Cyfrowych Instytutu Maszyn Matematycznych.

(jg)

• PROGRAMOWANE URZĄDZENIE DO MONTAŻU OKABLOWANIA typ PSM-500

Programowane urządzenie montażowe typu PSM-500 przeznaczone jest do półautomatycznego wykonywania połączeń kablowych na płytkach montażowych, panelach i ramach urządzeń elektronicznych.

Urządzenie działa na zasadzie bezpośredniego wskazywania punktu montażowego (szpilki) przez ruchomy celownik, zamocowany na wózku stołu krzyżowego. Jednocześnie z ustawianiem celownika zapalana jest lampka sygnalizacyjna nad odpowiednim pojemnikiem w magazynie przewodów. Połączenie wykonuje ręcznie operator metodą owijania.

Podstawowe dane techniczne:

- pole montażowe - 400 x 500 mm
- działka elementarna - 0,05 mm
  wykonanie specjalne - 0,01 mm
- prędkość przemieszczania celownika - 5 m/min
- liczba pojemników na przewody - 38
- zasilanie - 220V 50Hz 600W

Sterowanie przebiega automatycznie wg programu wprowadzanego na 8-ścieżkowej taśmie dziurkowanej - za pomocą czytnika taśmy RCT-250 wbudowanego w jednostkę sterującą. Jest to czytnik fotoelektryczny z możliwością dwukierunkowej pracy i mechanizmami szpulowymi.

Część sterująca jednostki zbudowana całkowicie na układach scalonych. Pulpit operacyjny zawiera cyfrowe wskaźniki współrzędnych stołu oraz klawiaturę do ręcznego sterowania, kontroli i początkowych nastaw rodzaju pracy. Programowanie współrzędnych w bezwzględnych wartościach w liczbach dziesiętnych dodatnich. Przykład bloku rozkazów na jedno połączenie:

. owinięcie początku przewodu - S142, t25, x2450, y1840
. owinięcie końca przewodu - x3200, y1250

Urządzenie PSM-500 podnosi wydajność pracy przy montażu i prawie całkowicie eliminuje błędne połączenia; opracowano je w Zakładzie Automatyzacji Produkcji Urządzeń Cyfrowych Instytutu Maszyn Matematycznych.

(jg)

• POWOŁANIE SEKCJI SPRZĘTU KOMITETU INFORMATYKI PAN

Na plenarnym posiedzeniu Komitetu Informatyki PAN, które odbyło się w Warszawie w dniach 23-24 marca br. powołano Sekcję Sprzętu. Zakres działania tej sekcji obejmować będzie następujące dziedziny:

. podstawy fizykalne informatyki
. struktura i architektura maszyn matematycznych
. struktura i architektura urządzeń peryferyjnych
. metody wspomaganego (automatycznego) projektowania
. metody wytwarzania sprzętu informatycznego

Cele działania sekcji

Zasadniczym celem sekcji, w pierwszym okresie jej działania będzie zestawienie listy problemów względnie tematów badawczych, których rozwiązanie może w istotny sposób wpływać na zbudowanie w latach osiemdziesiątych systemów liczących, zwanych umownie maszynami IV generacji. Problemy te mogą być podstawą do rozpoczęcia prac podstawowych, o znaczeniu wykraczającym poza wymieniony cel czysto utylitarny, a równocześnie w przypadku powodzenia takiej akcji mogą zapewnić silniejszą integrację tej części środowiska informatyków, którą interesują przede wszystkim prace o charakterze teoretycznym, z tą jego częścią, która uczestniczy

przede wszystkim w działaniach ukierunkowanych na nowoczesne konstrukcje komputerów.

Następnym etapem po sporządzeniu listy problemów będzie inwentaryzacja sił, polegająca na określeniu, które środowiska informatyków mogą się podjąć rozwiązania poszczególnych problemów, które z tematów mogą się stać przedmiotem polskiej specjalizacji w ramach współpracy krajów socjalistycznych, w których miejscach konieczne jest uzyskanie pomocy technicznej, np. przez zakup odpowiednich licencji. Etap ten będzie się charakteryzował stopniowym precyzowaniem tematyki.

W wyniku tych działań związanych z jednej strony z potrzebami wynikającymi z zamierzeń perspektywicznych, a z drugiej z możliwościami aktualnej polskiej kadry informatyków, zostaną stworzone podstawy do sformułowania nowych problemów węzłowych, obejmujących naukowe problemy informatyki, przez pogrupowanie tematów i problemów badawczych z listy pierwotnej, wzbogaconej wynikami dyskusji.

Kolejnym etapem działania, uzależnionym od powodzenia poprzednich etapów będzie ocena (typu ekspertyzy) zrealizowanych etapów badań, ocena ich przydatności dla wykorzystania przemysłowego, ocena kierunków badań teoretycznych, które np. w wyniku postępów technologii tracą swe znaczenie i nie zasługują na intensywne rozwijanie, uzupełnianie listy problemów badawczych nowymi pozycjami. Funkcje te będą wykonywane przez:

- okresowe organizowanie lub współorganizowanie konferencji, sympozjów i spotkań roboczych (sesji wyjazdowych Komitetu Informatyki PAN)
- publikowanie sprawozdań o różnym stopniu szczegółowości w czasopismach informatycznych
- przedstawianie bardziej interesujących opracowań na forum współpracy krajów socjalistycznych w ramach Jednolitego Systemu EMC.

Niektóre problemy badawcze związane z potrzebami maszyn perspektywicznych

Niżej zamieszczona lista jest próbą wstępnego sformułowania pewnych problemów, a raczej ich hasłowym omówieniem, o których już w tej chwili

wiadomo, że będą odgrywały poważną rolę w perspektywicznych systemach liczących.

1. Problemy projektowania, budowy i oprogramowania specjalizowanych systemów liczących przeznaczonych do projektowania złożonych struktur cyfrowych, zintegrowanych z systemami sterowania produkcją modułów cyfrowych i systemami automatycznej kontroli jakości. Problemy banku danych dla takiego systemu, zapewniającego automatyczne przekazywanie dokumentacji wytworzonej w fazie projektowania do części systemu sterującymi produkcją i kontrolą. Problemy współpracy systemu z projektantami - inżynierami.

2. Problemy określenia podstawowej struktury ("szkieletu komunikacyjnego" listy rozkazowej, interface´u oraz podstawowych jednostek przetwarzających i sterujących) eksperymentalnego systemu obliczeniowego dużej mocy, spełniającego równocześnie rolę systemu umożliwiającego szybkie przeprowadzanie eksperymentów z nowymi systemami przetwarzania informacji, np. wykorzystującymi zasady asocjacyjnego przechowywania i przetwarzania informacji, urządzenia do przetwarzania informacji obrazowej (picture processing), rozpoznawania obrazów i mowy, złożone eksperymenty symulacyjne z wykorzystaniem pracy wieloprocesorowej i inne metody nie dające się w tej chwili dokładnie określić.

3. Problemy sieci komputerowych złożonych z maszyn o zróżnicowanej strukturze, listach rozkazowych i systemach operacyjnych. Problemy realizowania zdecentralizowanych (rozłożonych) banków danych jedno- i wielodziedzinowych, potrzebnych w takich sieciach.

4. Problemy nowych struktur logicznych systemów liczących szczególnie przydatnych dla realizowania wzajemnie powiązanych procesów przetwarzania informacji przebiegających równolegle. Problemy komunikacji procesów przebiegających w różnych środowiskach hardware´owych (np. wspólna pamięć - pamięć rozdzielana między poszczególne procesory, synchronizacja liczników rozkazów, wykorzystanie i "usprzętownie" pewnych koncepcji oprogramowania: np. semafory Dijkstry).

5. Problemy interface´ów elektroniczno-optycznych dla przetwarzania informacji dwuwymiarowej. Sposoby reprezentacji informacji wizualnej (obrazy tonalne, barwne)w pamięciach nowego typu - zarówno elektronicznych

jak i optycznych (np. holograficznych).

6. Problemy wykorzystania regularnych sieci logicznych (cellular networks) dla rozwiązywania specjalizowanych problemów np. projektowanie układów logicznych, wykonywanie masek obwodów scalonych średniej i dużej integracji, projektowanie architektoniczne itp.

7. Problemy projektowania hierarchicznych systemów pamięciowych, wykorzystujących możliwości zawarte np. w pamięciach holograficznych (bardzo duże strumienie informacji - przy dowolnym dostępie) zarówno dla systemów uniwersalnych, jak i do procesorów specjalnych - np. przetwarzanie list.

W innych problemach badawczych, również o wielkim znaczeniu dla maszyn perspektywicznych, sekcja sprzętu będzie współpracowała z pozostałymi sekcjami Komitetu. Niżej wymienione problemy mają charakter zagadnień z dziedziny oprogramowania. Jednakże obserwowana obecnie tendencja przejmowania licznych funkcji oprogramowania przez sprzęt (tzw. petryfikacja software'u) wymaga coraz większego udziału specjalistów - sprzętowców w takich pracach, aby podział funkcji między sprzęt i oprogramowanie dla przyszłych systemów był dokonany najracjonalniej z punktu widzenia wykorzystania wszystkich możliwości proponowanych przez współczesne technologie. Wstępna lista takich problemów obejmuje następujące pozycje:

1. Problemy wspomaganego (zautomatyzowanego) przejmowania oprogramowania przez maszyny nowo budowane i włączanie maszyn (wraz z oprogramowaniem) w skład nowo budowanych systemów liczących.

2. Problemy budowy języków dynamicznie definiowanych (rozszerzalnych) i metody realizacji translatorów takich języków - realizacja sprzętowa typowych funkcji w takich językach i ich translatorach.

3. Metody analizy syntaktycznej, umożliwiające samokorekcję trywialnych błędów, metody budowy bibliotek środków diagnostycznych dla "run time" - zagadnienia te nabiorą szczególnego znaczenia w miarę postępowania procesu eliminacji profesjonalnych programistów w przygotowaniu programów zastosowaniowych.

(jd)

PRZEGLĄD DOKUMENTACYJNY

ETO NOWOŚCI Nr 2/1973

Na podstawie kart dokumentacyjnych nadesłanych przez ELWRO i Meramat - opracował Jerzy Klamborowski

621.382.049.7-181.4 Układy scalone ELWRO ang.

Rapp A.K., Ross E.C.: Silicon-on-sapphire substrates overcome MOS limitations. Podłoża krzemu na szafirze pozwalają wyeliminować ograniczenia układów MOS. Electronics 1972 nr 20, s.113-116, rys.4.

Budowa i zasada działania układów MOS z podłożem na szafirze. Dzięki wynikającej z właściwości szafiru odmiennej technice izolacji układy te dorównują pod względem szybkości układom bipolarnym. W dodatku układy te są zgodne z układami TTL, czym różnią się od konwencjonalnych układów MOS.

Treter A.

621.382.049.7-181.4 Układy scalone ELWRO
621.3.011.3.001.24 Indukcyjność, obliczanie ang.

Ruehli A.E.: Inductance calculations in a complex integrated circuit environement. Obliczenia indukcyjności torów w złożonych układach scalonych. IBM J.Res.Dev. 1972 nr 5, s.470-481, rys.13, bibliogr., poz.10.

Teoria obliczania indukcyjności torów przewodzących w złożonych wzorach połączeń stosowanych w półprzewodnikowych układach scalonych. Teoria ta nazywana teorią indukcyjności cząstkowych polega na obliczaniu indukcyjności zamkniętych pętli przewodzących na podstawie określenia indukcyjności segmentów tych pętli.

Treter A.

681.327.17 Urządzenia kontrolujące i sprawdzające ELWRO ang.
681.382.049.7-181.4.001.5 Układy scalone, badania naukowo-techniczne

Runyon S.: On MSI/LSI testers. Testery obwodów scalonych MSI/LSI. Electron. Des. 1972 nr 17, s.60-69, rys.9.

Praktyczne możliwości i zalety automatycznych, programowanych testerów układów scalonych MSI/SSI. Przedstawiono parametry i wielkości elektryczne sprawdzanych układów za pomocą mikroprogramowanych testerów produkowanych przez różne firmy zachodnie. Sposoby pracy układów testera zbudowanego w technice MOS.

Urbanek A.

681.322.004.15.001.36 Elektroniczne maszyny cyfrowe, wydajność techniczna,p.widz. porównawczy ELWRO ang.
681.322.001.6 Elektroniczne maszyny cyfrowe, praca nad rozwojem

Flynn M.J.: Some computer organizations and their effectiveness. Organizacja komputerów i ich efektywność.IEEE Trans.Comp. 1972 nr 9, s.948-960,rys.11,bibliogr.,poz.34.

Porównanie efektywności i wydajności komputerów przy uwzględnieniu różnych systemów organizacyjnych tych maszyn.Przedstawiono sugestie w jakim kierunku powinien postępować rozwój typów organizacji wewnętrznych systemów komputerowych (jednoprocesorowe i wieloprocesorowe, szeregowe i równoległe przesyłanie danych, itp.) w celu osiągnięcia maksymalnych korzyści (czas obliczeń) przy stosunkowo niewielkich nakładach na sprzęt techniczny.

Urbanek A.

681.322 Elektroniczne maszyny cyfrowe ELWRO
681.327.02 Pamięci ang.

Morris J.B.: Demand paging through utilization of working sets on the MANIAC II. Stronicowanie pamięci w komputerze MANIAC II. Communic.ACM 1972 nr 10,s.867-872, rys.3,bibliogr., poz.20.

Struktura wewnętrzna komputera MANIAC II,konfiguracja tego systemu łącznie z urządzeniami zewnętrznymi oraz metoda stronicowania pamięci. Zalety tego typu stronicowania pamięci, układy blokowe mechanizmu stronicowania i rozwiązania techniczno-logiczne zapewniające efektywną pracę modelu MANIAC II. Przeprowadzono analizę optymalnej wielkości i liczby stronic pamięci w celu organizacji efektywnej pracy maszyny.

Urbanek A.

681.322 Elektroniczne maszyny cyfrowe ELWRO
621.382.049.7-181.4 Układy scalone ang.
621.3.049.73 Oprzewodowanie

Balph T.:Avoid ECL-10000 wiring problems. Jak uniknąć problemów z połączeniami ECL-10000. Electron.Des. 1972 nr 18,s.48-52, rys.9, bibliogr., poz.1.

Problem połączeń w sieci mikroukładów scalonych ECL-10000 firmy Motorola. Ze względu na dużą szybkość tych układów,sposób ich łączenia stanowi problem konstrukcyjno-technologiczny.Uwzględniowo takie aspekty jak czas propagacji sygnału, tłumienie sygnału, przesłuchy między sąsiednimi torami i odbicia wywołane niedopasowaniem impedancji.Uwzględniono połączenia wykonywane pojedynczymi drutami, parami przewodów skręconych oraz połączenia owijane.

Treter A.

681.322 Elektroniczne maszyny cyfrowe ELWRO
621.315.68 Łączenie przewodów i kabli ang.

Sidney D.: Connector evaluation for computer equipment applications. Ocena złączy stosowanych w sprzęcie komputerowym.Computer Des. 1972 nr 10, s.89-95, rys.4, tabl.2.

Aktualny stan i możliwości techniczne przemysłu produkującego złącza dla potrzeb techniki komputerowej. Wymagania ogólne dla złączy i łączówek w zależności od zastosowań. Podano tablice klasyfikacyjne dla wszelkich typów złączy elektrycznych.Najważniejsze cechy materiałów dielektrycznych i przewodzących stosowanych w złączach elektrycznych oraz rozpatrzono problem elastycznych złączy kablowych dla zastosowań w komputerach.

Urbanek A.

681.322-181.4.001.13 Elektroniczne maszyny cyfrowe małe, projekt — ELWRO ang.
681.382.049.7-181.4 Układy scalone

Breedlove P.S.: ECL/MOS for optimum minicomputer systems. Układy logiczne w technice ECL/MOS dla optymalnych systemów minikomputerowych. Computer Des. 1972 nr 8, s.61-66, rys.9, tabl.3.

Podano klasyczną strukturę logiczną systemu minikomputera, w której uwzględniono: system sterowania, układy arytmometru, układy pamięciowe, urządzenia zewnętrzne oraz współpracę z jednostką sterującą. W strukturze logicznej przedstawiono schematy blokowe mikroprogramowanej pamięci stałej oraz schematy blokowe arytmometru i jednostki sterującej urządzeniami zewnętrznymi. Charakterystyczne parametry techniczne minikomputera opartego na ww strukturze, zbudowanego w technice MOS.

Urbanek A.

681.322 Elektroniczne maszyny cyfrowe — IMM
518.5 Liczenie za pomocą maszyn do liczenia — ang.
519.281 Teoria błędów

Richman P.L.: Automatic error analysis for determining precision. Automatyczna analiza błędu dla określenia dokładności obliczeń. Communic.ACM 1972 nr 9, s.813-817, bibliogr., poz.2.

Obliczanie z określoną dokładnością wyrażeń zawierających liczby naturalne, na komputerze ze zmiennym przecinkiem o zmiennej dokładności. Automatyczna analiza błędu pozwala na podstawie badania obliczeń w przedziale niskiej dokładności określić wymaganą dokładność obliczeń i danych.

Kawa M.

681.322.004.14 Elektroniczne maszyny cyfrowe, zastosowanie — ELWRO ang.
802.0 Język angielski
801.56 Składnia
801.54 Semantyka

Simmons R., Slocum J.: Generating English discourse from semantic networks. Tworzenie zdań w języku angielskim za pomocą zbiorów semantycznych. Communic.ACM 1972 nr 10, s. 891-905, rys.4, tabl.7, bibliogr.,poz.23.

Metoda tworzenia zdań w języku angielskim za pomocą sieci semantycznej. Podano definicję zbioru semantycznego, zasady kojarzenia sylab w zdania, zasady kojarzenia zdań, reguły gramatyczne w języku angielskim oraz algorytm tworzenia zdań przez komputer. Podano przykłady tworzenia zdań.

Urbanek A.

681.322.06 Elektroniczne maszyny cyfrowe, programy i programowanie — MERAMAT ros.

Safonov I.V.: Ob odnom algoritme unifikacii operacionnych blokov cifrovych masin. O pewnym algorytmie unifikacji operacyjnych bloków komputerów. Avtom.Vycisl.Tech. 1972 nr 4, s.69-73, bibliogr., poz.3.

Propozycja formalnego sposobu unifikacji elementów procesorów komputerów na etapie projektowania algorytmu. Wprowadzenie funkcji pierwszeństwa pozwala na zmniejszenie nadmiaru wariantów.

Zwierzyk L.A.

681.322.06 Elektroniczne maszyny cyfrowe programy i programowanie — MERAMAT ang.

McCracken D.D.,Weinberg G.M.: How to write a readable FORTRAN program. Jak pisać czytelne programy w języku FORTRAN. Datamation 1972 nr 10, s.73-77, rys.1.

Większość programów w okresie ich stosowania wymaga wnoszenia zmian. Podstwę ich może stanowić jedynie dokładnie wykonana dokumentacja. Zasady umieszczania komentarzy w programach pisanych w języku FORTRAN. Wymagania dotyczące zestawu dokumentacji programu.

Kawa M.

681.322.06 Elektroniczne maszyny cyfrowe, programy i programowanie — MERAMAT ang.
681.327.8 Urządzenia do transmisji danych cyfrowych

Bowie J.A.: Software for telecommunications. Oprogramowanie systemów telekomunikacyjnych. Data Systems 1972 nr 10, s. 22-25, rys.2.

Podstawowe informacje o budowie systemów telekomunikacyjnych i wymagania stawiane ich oprogramowaniu. Architektura oprogramowania systemów. Ocena wymagań sprzętowych.

Kawa M.

681.322.06 Elektroniczne maszyny cyfrowe, programy i programowanie — ELWRO ang.
681.3.056 Operacje logiczne
681.3.058 Generowanie funkcji

Dathe G.: Conversion of decision tables by rule mask method without rule mask. Konwersja tablic decyzyjnych z maskowaniem i bez maskowania. Communic.ACM 1972 nr 10, s.906-909, rys.11, bibliogr., poz.7.

Dwie metody konwersji tablic decyzyjnych stosowanych w programach i systemach komputerowych. Jedna z metod polega na generowaniu tablic za pomocą drzewa logicznego, druga za pomocą maski tablicy. Przedstawiono dwa algorytmy konwersji tablic za pomocą maskowania. Oszacowano korzyści wynikające ze stosowania tej metody.

Urbanek A.

681.322 Elektroniczne maszyny cyfrowe — ELWRO ang.
681.327.8 Urządzenia do transmisji danych cyfrowych

Hobbs L.C.: Terminals. Urządzenia końcowe. Proc.IEEE 1972 nr 11, s.1273-1284.

Opis używanych obecnie urządzeń końcowych. Analiza możliwości tych urządzeń w odniesieniu do wymagań stawianych przez system łączności, z którym te urządzenia współpracują. Aspekty ekonomiczne i niezawodnościowe wyposażenia tych urządzeń w autonomiczne możliwości programowego przetwarzania informacji.

Treter A.

681.322 Elektroniczne maszyny cyfrowe ELWRO
681.327.8.001.13 Urządzenia do transmisji danych cyfrowych, projekt ang.

Chandy K.M. i in.: The design of multipoint linkages in a teleprocessing tree network. Projektowanie wielopunktowych sieci abonenckich dla przetwarzania danych. IEEE Trans.Comp. 1972 nr 10, s.1062-1072, rys.4, bibliogr.,poz.12.

Zagadnienia projektowania sieci abonenckich dla systemów EPD. Podano algorytm tworzenia optymalnych sieci abonenckich z uwzględnieniem minimalnej liczby połączeń przy zadanej liczbie punktów abonenckich. Porównano efektywność wykonywanych obliczeń w różnych systemach abonenckich w stosunku do optymalnego rozwiązania podanego w artykule.

Urbanek A.

681.322 Elektroniczne maszyny cyfrowe ELWRO
681.327.8 Urządzenia do transmisji danych cyfrowych ang.
513.83.004.14 Topologia, zastosowania

Franck H., Chou W.: Topological optimization of computer network. Topologiczna optymalizacja sieci komputerowej.Proc. IEEE 1972 nr 11, s.1385-1397, rys.13, tabl.1,bibliogr., poz.59.

Problem optymalizacji sieci komputerowej. Podstawowy model sieci dla teorii obsługi i analizy niezawodności. Podano kilka metod optymalizacji. Dziedziny, w których niezbędne są dalsze badania.

Treter A.

681.382.049.7-181.4.004.14 Układy scalone, zastosowanie MERAMAT ros.
681.323.001.13 Specjalizowane systemy i maszyny cyfrowe

Majorov S.A., Li Si Ken, Starodubcev E.Y.: Nekotorye voprosy proektirovania SCVM na bolsich integral nych schemach. Niektóre zagadnienia projektowania specjalizowanych komputerów na układach scalonych o dużej integracji. Priborostroenie 1972 nr 8, s.67-69, rys.1, bibliogr., poz.3.

Jeden ze sposobów realizacji obliczenia funkcji dwóch zmiennych za pomocą typowych macierzy, na podstawie których konstruuje się specjalizowane komputery na układach scalonych o dużej integracji.

Zwierzyk L.

681.323 Specjalizowane systemy i maszyny cyfrowe, zastosowanie IMM ang.
519.14 Grafy

Levitt K.N., Kautz W.H.: Cellular arrays for the solution of graph problems. Zastosowanie układów komórek dla rozwiązywania zadań z teorii grafów. Communic.ACM 1972 nr 9, s.789-801, rys.11, bibliogr., poz.13.

Dwuwymiarowa tablica złożona z identycznych modułów (komórek) zawierających kilkubitową pamięć i nieskomplikowane układy logiczne, połączenia między sąsiadującymi modułami w tablicy. Możliwość osiągnięcia znacznego zwiększenia prędkości obliczeń dzięki jednoczesnemu ich wykonywaniu. Zastosowanie układu komórek obliczeniowych do rozwiązywania zadań z dziedziny teorii grafów.

Kawa M.

681.32.001.13 Cyfrowe układy, maszyny i urządzenia,projekt — ELWRO ang.
621.322.049.7-181.4.001.13 Układy scalone, projekt
513.83.004.14 Topologia, zastosowanie

Patel S., Barry J.N.: Custom designed MOS aarrays for use in digital systems. Matryce MOS wykonywane na zamówienie do systemów cyfrowych. Electron.Eng 1972 nr 536, s.64-66, rys.3.

Zagadnienie zlecania przez producenta dużych systemów cyfrowych - produkcji specjalizowanych matryc MOS producentowi półprzewodnikowych układów scalonych. Opisano sposoby formułowania wymagań dotyczących bramki podstawowej i jej układu topologicznego. Racjonalniejszym sposobem postępowania jest opracowywanie topologii przez zleceniodawcę i przekazywanie do producenta dopiero mozaik układów.

Treter A.

621.322.049.7-181.4.001.13 Układy scalone, projekt — ELWRO
681.32.001.13 Cyfrowe układy,maszyny i urządzenia,projekt — ang.

Patel S., Barry J.N.: MOS custom design meets digital systems requirements. Produkcja układów MOS na zamówienie spełnia wymagania stawiane przez systemy cyfrowe. Electron.Eng 1972 nr 537, s.62-64, rys.6, bibliogr., poz.2.

Opisano sposoby uzyskiwania pojemności między węzłami układów oraz podano ogólne wskazówki projektowania układów MOS o dużej skali integracji.

Treter A.

681.32.001.5 Cyfrowe układy, maszyny i urządzenia, badania naukowo-techniczne — MERAMAT ros.
681.327.17 Urządzenia kontrolujące i sprawdzające

Marinov M., Vladkov E., Aleksandrova Z.: Apparat dla avtomatičeskoj diagnostiki diskretnych schem i ustrojstv. Aparat do automatycznej diagnostyki dyskretnych układów i urządzeń. Avtom.Vycisl. Tecan. 1972 nr 5, s.85-91, rys.5.

Zagadnienie syntezy urządzeń diagnostyki układów logicznych i złożonych funkcjonalnych węzłów techniki cyfrowej. Proponuje się metodę kolejnej kontroli elementarnych układów badanych węzłów drogą porównania sprawdzanej informacji wyjściowej i wzorcowego układu. Przytoczono schemat realizacji urządzenia diagnostycznego dla elektronicznego kalkulatora bułgarskiego typu "ELKA".

Zwierzyk L.A.

681.3.004.14.001.36 Maszyny matematyczne, zastosowanie, p.widz.porównawczy — ELWRO ang.
681.327.12 Urządzenia odczytujące

Preston K.: A comparison of analog and digital techniques for pattern recognition. Porównanie techniki analogowej i cyfrowej w zastosowaniu do rozpoznawania wzorów. Proc.IEEE 1972 nr 10, s.1216-1231, rys.21, tabl.1, bibliogr., poz.31.

Dokonano porównania przydatności komputerów analogowych i cyfrowych do rozpoznawania wzorów. Stwierdzono, że w ograniczonym lecz ważnym zakresie zastosowań, maszyny analogowe przewyższają maszyny cyfrowe pod względem szybkości i taniości sprzętu. Jednak należy się spodziewać, że rozwój konstrukcji maszyn cyfrowych odwróci tę sytuację w najbliższych latach.

Treter A.

681.325.65 Układy funktorów logicznych ELWRO ang.

Miles T.E.; Schotky TTL vs ECL for high speed logic. Porównanie układów logicznych TTL na diodach Schotky ego z szybkimi układami ECL. Computer Des. 1972 nr 10, s.79-86, rys.17, tabl.3.

Metody zwiększające szybkość przełączania stanu "jedynki" i "zera" w układach logicznych z zastosowaniem obwodów techniki TTL i ECL. Zasada działania układów TTL z zastosowaniem układów diodowych z progiem Schotky ego o typowym opóźnieniu ok.3 ns oraz zasada działania układów ECL z opóźnieniem 2÷3,5 ns. Podano schematy ideowe układów klasycznych. Charakterystyki prądowo-napięciowe, czasy opóź - nień, pobierana moc, zakłócenia i zgodność z innymi układami.

Urbanek A.

681.327.17 Urządzenia kontrolujące i sprawdzające ELWRO ang.
681.325.65.001.5 Układy funktorów logicznych, badania naukowo-techniczne

Schertz D.R.: A new representation for faults in combina - tional digital circuits. Detekcja błędów w kombinacyjnych układach cyfrowych. IEEE Trans.Comp. 1972 nr 8, s.858-866, rys.8, tabl.2, bibliogr., poz.13.

Problem detekcji błędów w układach komputerów. Strukturę logiczną podzielono na segmenty związane z poszczególnymi mikroukładami, przedstawiono wykresy graficzne takiego układu, zastosowano do niego analizę matematyczną. Metoda powyższa pozwala na detekcję trzech typów błędów często spotykanych przy projektowaniu układów logicznych komputerów.

Urbanek A.

681.326 Środki programowania ELWRO
681.325.65 Układy funktorów logicznych ang.

Jump J.R., Pritsche D.R.: Microprogrammed arrays. Układy mikroprogramowania. IEEE Trans. on Comp. 1972 nr 9, s.974-984, rys.20, bibliogr., poz.14.

Problem mikroprogramowania zbiorów składających się z identycznych komórek, z których każda może przyjmować kilka ściśle określonych stanów logicznych. Omówiono szczegółowo stany tych komórek, możliwe połączenia wewnętrzne w zbiorach, realizację typowych funkcji arytmetycznych i logicznych przez pojedyncze komórki sieci logicznej, połączenia kaskadowe i równoległe komórek w celu realizacji mikrorozkazów komputera.

Urbanek A.

681.382.049.7-181.4.001.13 Układy scalone projekt ELWRO ang.
681.325.65.001.13 Układy funktorów logicznych, projekt

Balph T.: Use ECL 10.000 layout rules. Zasady projektowania pakietów dla techniki ECL 10.000. Electron Des. 1972 nr 17, s.72-76, rys.5, tabl.3, bibliogr.,poz.2.

Zasady projektowania pakietów wielowarstwowych stosowanych w komputerach. Zwrócono szczególną uwagę na dopasowanie układów scalonych w technice ECL mające na celu zwiększenia szybkości działania przez stosowanie właściwej oporności. Podano tabele opóźnień wnoszonych przez linie długie w zależności od obciążeń układów wyjściowych.

Urbanek A.

681.325.65 Układy funktorów logicznych ELWRO
621.382.049.7-181.4 Układy scalone ang.

Berger H.H., Wiedmann S.K.: Merged transistor logic (MTL) - A low cost bipolar logic concept. Połączone tranzystorowe układy logiczne (MTL) - tanie układy bipolarne. IEEE J. Solid St.Circuits 1972 nr 5, s.340-346, rys.14, bibliogr., poz.7.

Opisano nową logikę bipolarną wykorzystującą bezpośrednio wstrzykiwanie nośników mniejszościowych do tranzystora przełączającego. Występuje tu bezpośrednie współdziałanie tranzystorów komplementarnych - stąd nazwa (Merged Transistor Logic). Opisano zasadę budowy i działania, charakterystyki urządzeń MTL i przedyskutowano zagadnienie dopasowania układów MTL do układów pośredniczących.

Treter A.

681.325.65 Układy funktorów logicznych ELWRO
621.382.049.7-181.4 Układy scalone ang.
621.382.33 Tranzystory bipolarne

Hart K., Slab A.: Integrated injection logic: a new approach to LSI. Scalone układy logiczne oparte na wstrzykiwaniu nośników: nowe podejście do układów scalonych o dużym stopniu integracji. IEEE J.Solid St.Circuits 1972 nr 5, s.346-351, tabl.2, bibliogr., poz.3.

Technika wykonywania układów scalonych bipolarnych o dużym stopniu integracji. Polega ona na wykorzystaniu tranzystora wielokolektorowego z wtryskiwaniem nośników. Opisano dwa typy aktywacji wstrzykiwania: za pośrednictwem światła lub złącza p-n. Opisano technologię wykonywania układów IIL i uzyskane parametry. Zagadnienie szumów i czasu propagacji oraz dopasowanie do układów pośredniczących. Możliwość współpracy układów IIL z obwodami liniowymi.

Treter A.

681.325.59 Urządzenia do wyciągania pierwiastka kwadratowego ELWRO
ang.
681.3.042 Systemy dwójkowe

Kostopulos G.K.: Computing the square root of binary numbers. Obliczanie pierwiastka kwadratowego liczb binarnych. Computer Des. 1972 nr 8, s.53-57, rys.6, bibliogr., poz.1.

Ogólna teoria i algorytm wyciągania pierwiastka kwadratowego z zastosowaniem do liczb binarnych nie wymagająca wnoszenia poprawek. Podano przykłady obliczeń na liczbach binarnych oraz sprzęt techniczny niezbędny do realizacji tego algorytmu. Realizację algorytmu przedstawiono w dwóch wariantach: dla układów szeregowych oraz równoległych. Metoda pozwala na układową bądź programową realizację.

Urbanek A.

681.325.59 Urządzenia do wyciągania pierwiastka kwadratowego ELWRO
ang.
681.3.055 Operacje mnożenia arytmetycznego

Ramamorthy C.V., Goodman J.R., Kim K.H.: Some properties of iterative square - rooting methods using high - speed multiplication. Pewne właściwości iteracyjnej metody pierwiastkowania kwadratowego z zastosowaniem mnożenia o dużej szybkości. IEEE Trans. Comp. 1972 nr 8, s.837-847, rys.6, tabl.3, bibliogr., poz.12.

Przeprowadzono klasyfikację algorytmów obliczania pierwiastka kwadratowego za pomocą komputerów. Podano trzy metody obliczeń ze szczególnym uwzględnieniem metody iteracyjnej Newtona-Raphsona. Zalety stosowania odpowiedniego algorytmu oraz matematyczne uzasadnienie każdego z nich.

Urbanek A.

681.325.6 Urządzenia,maszyny lub elementy dla operacji logicznych — MERAMAT ros.

Jakubajtis E.A.: Struktura i effektivnost' mnogofunkcjonal'nogo logiceskogo elementa. Budowa i efektywność wielofunkcyjnego elementu logicznego. Avtom.Vycisl.Tech. 1972 nr 5, s.1-8, tabl.4, rys.4.

Rozpatrzono budowę wielofunkcyjnego elementu logicznego i efektywność jego wykorzystania w końcowym automacie,przeznaczonego dla realizacji przetwarzania logicznego.

Zwierzyk L.A.

681.327.54'11 Drukarki wierszowe — MERAMAT ang.

Blee M.: Printer line-up. Drukarki wierszowe. Data Systems 1972 nr 9,s.27-28, rys.2.

Przegląd właściwości stosowanych obecnie drukarek wierszowych. Perspektywy rozwoju i metody zastosowań. Wzrost zastosowań drukarek łańcuchowych i seryjnych.

Kawa M.

681.327.54'11.001.13 Drukarki, projekt
681.327.11.001.13 Urządzenia zapisujące i rejestrujące,projekt
744.32 Maszyny rysujące — ELWRO ang.

Bakey T.F.: Hardware design of an electrostatic printer/plotter. Projektowanie układów elektrostatycznych drukarek i pisaków X-Y. Computer Des. 1972 nr 9, s.83-89, rys.14.

Zasada działania elektrostatycznych urządzeń wyprowadzających dane z maszyny cyfrowej: drukarek i pisaków X-Y. Schemat blokowy urządzenia, układów sterujących oraz zapisujących informację na papierze, model generatora znaku, układy do sterowania mechanizmem przesuwu oraz system barwienia papieru metodą elektrostatyczną przy dużej szybkości przesuwu nośnika.

Urbanek A.

681.327.54'11.048 Drukarki, systemy alfanumeryczne
537.228.1.004.14 Piezoelektryczność,zastosowanie
538.652.004.14 Magnetostrykcja,zastosowanie — ELWRO ang.

Ernbo A.: Application of intensity - modulated ink jets to alphanumeric printing devices. Zastosowanie pisaka o modulowanym zaczernieniu do alfanumerycznych urządzeń drukujących. IEEE Trans.Comp. 1972 nr 9, s.942-947, rys.9,bibliogr., poz.12.

Nowa metoda szybkiego wydruku informacji na papierze wykorzystująca zjawisko piezoelektryczności lub magnetostrykcji.Wydruk znaków alfanumerycznych przy szybkości 50 znaków/s oraz 24 linii/s następuje nie za pomocą młotków lecz przez modulację zaczernienia strumienia farby oraz odpowiednie jego ukształtowanie. Schematy blokowe układu modulującego strumień farby oraz generatory znaków alfanumerycznych.Układ charakteryzuje się prostotą konstrukcji i brakiem części mechanicznych przy zadowalającej rozdzielczości wydruku.

Urbanek A.

681.326.3.004.17 Urządzenia sterujące i programujące, sprawność ekonomiczna — MERAMAT ang.
621.382.049.7-181.4.004.14.004.17 Układy scalone, zastosowanie, sprawność ekonomiczna

Davidow W.H.: General-purpose microcontrollers. Part I: economic considerations. Uniwersalne jednostki sterujące. Część I - ocena ekonomiczna. Computer Des. 1972 nr 7, s.75-79, rys.12.

Mikroprogramowanie pozwala zredukować koszty konstruowania i produkcji jednostek sterujących o 80%. Zmniejszają się także koszty obsługi i dokumentacji jednostek sterujących zbudowanych na zasadzie mikroprogramowania. Ocena przydatności konkretnych typów elementów scalonych w budowie jednostek sterujących.

Kawa M.

681.327 Pamięci — MERAMAT
681.327.17 Urządzenia kontrolujące i sprawdzające — ros.

Plitman A.D.: Ob odnom metode obnaružženia odinočnych neispravnostej v schemach s pamiat ju. O pewnym sposobie wykrywania pojedynczych uszkodzeń w układach z pamięcią. Avtom.Telemech. 1972 nr 10, s.166-173, rys.1, tabl.1, bibliogr., poz.6.

Metody konstrukcji ciągu impulsów sprawdzających dla danego pojedynczego uszkodzenia. Układy synchroniczne z przerzutnikami jako elementami pamięci, bez wewnętrznych sprzężeń zwrotnych. W układzie dopuszcza się pojedyncze stałe uszkodzenie typu "0" i "1". Przy określonych ograniczeniach elementów pamięci i ich stanów wyjściowych metoda ta pozwala otrzymać ciąg impulsów sprawdzających o minimalnej długości.

Zwierzyk L.A.

681.327.12 Urządzenia odczytujące — MERAMAT
681.327.4 21 Wejścia na taśmy dziurkowane — ros.

Kutuzov V.I., Petrov A.G.; Revenko V.S.: Vvod zapisannoj na perfolente informacii v BESM-4. Wprowadzanie informacji zapisanej na taśmie perforowanej do BESM-4. Prib.i Tech.Eksper. 1972 nr 4, s.85-89, rys.4, bibliogr., poz.2.

Przedstawiono urządzenie pośredniczące dla wprowadzenia do komputera BESM-4 informacji zapisanej na taśmie perforowanej. Taśma odczytywana jest za pomocą mechanizmu z odczytem fotooptycznym FSM-5 ze średnią szybkością 1000 wierszy/s.

Zwierzyk L.A.

681.326.3.001.13 Urządzenia sterujące i programujące, projekt — ELWRO ang.
681.382.049.7-181.4.004.14 Układy scalone
681.327.66.004.14 Urządzenia pamięci ze stałym nośnikiem

Davidow W.H.: General-purpose microcontrollers: design and applications. Uniwersalne mikroukłady sterujące: projektowanie i zastosowanie. Computer Des. 1972 nr 8, s.69-75, rys.9.

Problemy związane z projektowaniem układów sterujących dla kilku urządzeń zewnętrznych współpracujących z komputerem. Omówiono strukturę logiczną mikroukładów sterujących, układy rejestrów sterowanych pamięcią stałych, elementy scalone typu ROM i RAM, wykresy czasowe współpracy z urządzeniami zewnętrznymi, formaty mikroinstrukcji oraz metody synchronizacji w przypadku współpracy za pomocą multipleksora.

Urbanek A.

681.327.64.001 Urządzenia pamięci magnetycznej taśmowej, projekt — ELWRO ang.

Frank M.: Coaxial cartridge concept for miniature tape decks. Koncepcja współosiowego mechanizmu dla miniaturowych taśm magnetycznych. Computer Des. 1972 nr 10, s.96-100,rys.2.

Koncepcja miniaturowych taśm magnetycznych z zastosowaniem jednoosiowego napędu dla obydwu krążków taśmowych. Podano podstawowe obliczenia dla naciągu taśmy, przykłady obliczeń oraz zastosowanie takiego zestawu jako końcowego urządzenia do zbierania i zapisywania danych w systemie komputerów.

Urbanek A.

681.327.66 Urządzenia pamięci ze stałym nośnikiem informacji — ELWRO ang.
621.382 Przyrządy półprzewodnikowe

Engeler W.E., Tiemann J.J., Baertsch R.D.: A surface charge random-access memory system. System pamięci o dostępie natychmiastowym na elementach o ładunku powierzchniowym. IEEE J.Solid St.Circuits 1972 nr 5, s.330-335, rys.11, bibliogr., poz.11.

Budowa i działanie komórki pamięci działającej na zasadzie wykorzystania zjawiska ładunku powierzchniowego z kanałem typu -p. Komórka taka nadaje się do systemu pamięci z dostępem natychmiastowym, dynamicznej i z wybieraniem liniowym. Opisano również wzmacniacz odczytu i regeneracji zapisu. Wyniki doświadczeń wykonanych na matrycy pamięciowej 4 x 8. Symulacja maszynowa tego typu pamięci pokazuje, że przy pojemności 4096 bitów i gęstości zapisu 2,5 mils^2/bit, czas cyklu, przy opisanym wzmacniaczu powinien wynieść 250 ns, czas dostępu 150 ns.

Treter A.

681.327.66 Urządzenia pamięci ze stałym nośnikiem informacji — MERAMAT ros.

Samofalov K.G., Seligej A.M.,Trostjaneckij D.S.: Voprosy optimizacii struktury transformatornych postojannych zapominajuščich ustrojstv. Zagadnienia optymalizacji struktury stałych pamięci transformatorowych. Priborostroenie 1972 nr 9,s.69-73, rys.3, tabl.3, bibliogr.,poz.2.

Zagadnienia konstrukcji stałych pamięci transformatorowych z uwzględnieniem optymalizacji struktury części adresowej i szybkości działania. Dla spełnienia tych wymagań proponuje się zastosowanie systemów zliczania o podstawie $p > 2$, w szczególności systemu czwórkowego.

Zwierzyk L.A.

681.327.66-416 Urządzenia pamięci ze stałym nośnikiem informacji,bardzo cienkie przedmioty płaskie — ELWRO ang.

Torok E.J.: Film only a few atoms thich promises very large mass memories. Warstwa o grubości zaledwie kilku atomów może służyć jako duża pamięć masowa. Electronics 1972 nr 23, s.106-112, rys.8, bibliogr.,poz.14.

Opisano pamięć, której nośnikiem informacji jest bardzo cienka warstwa permalojowa. Warstwa ma grubość nie przekraczającą 50 średnic atomowych i jest nazywana oligatomową. Pamięć ma czas cyklu ok.3 µs.natomiast pozwala na ok. 100-krotnie gęściejszy zapis niż na konwencjonalnych cienkich warstwach magnetycznych. Dokonano zestawień porównawczych dla różnych rodzajów pamięci masowych.

Treter A.

681.327.001.13 Pamięci, projekt
681.325.67 Urządzenia do sortowania i wybierania
519.14.004.14 Grafy, zastosowanie

ELWRO ang.

Patt N.Y.: Minimum search tree structures for data partitioned into pages. Optymalny algorytm wyszukiwania danych w zbiorach dzielonych na strony. IEEE Trans.Comp. 1972 nr 9, s.961-967, rys.3, bibliogr.,poz.8.

Matematyczne uzasadnienie wydajnej (szybkiej) metody przeszukiwania zbiorów danych w celu znalezienia właściwej strony danego zbioru dla komputerów ze stronicowaniem pamięci. Analizę oparto na strukturze drzew o podwójnym łańcuchu i jest ona podstawowa w zastosowaniu do komputerów przy stronicowaniu danych lub wyszukiwaniu zbiorów.

Urbanek A.

681.327.06 Pamięci,programy i programowanie

ELWRO ang.

Thorington J.M., Irwin J.D.: An adaptive replacement algorithm for paged memory computer systems. Zastępczy algorytm dla stronicowania pamięci w systemach komputerowych. IEEE Trans.Comp. 1972 nr 10, s.1053-1061, rys.10, tabl.3,bibliogr., poz.20.

Uproszczony dwykres blokowy stronicowania pamięci w komputerach oraz odpowiadający mu algorytm logiczny stronicowania pamięci za pomocą programu. Wyniki tej symulacji oraz dyskusja nad teoretycznym, rzeczywistym i optymalnym modelem stronicowania pamięci w komputerach.

Urbanek A.

681.327.63 Urządzenia pamięci magnetycznej bębnowej i dyskowej
681.327.613 Głowice magnetyczne zapisujące i odczytujące
681.325.65 Układy funktorów logicznych

ELWRO ang.

Symons C.V.: Logic sparing of fixed heads on drums and disc. Automatyczne przełączanie uszkodzonych głowic w pamięciach dyskowych i bębnowych. Computer Des. 1972 nr 9, s.91-94, rys.3.

Logiczna metoda przełączania układu wybierania głowic w pamięciach zewnętrznych komputerów w przypadku uszkodzenia którejś głowicy. Podano blokowe układy typowego wybierania głowic oraz ulepszony system wybierania z możliwością korekcji uszkodzenia. Ww system wybierania oznacza się prostotą konstrukcji oraz maksymalnym wykorzystaniem urządzeń pamięciowych przez zmniejszenie do minimum czasu napraw i wymiany głowic.

Urbanek A.

681.327.63 Urządzenia pamięci magnetycznej dyskowej
681.327.17 Urządzenia kontrolujące i sprawdzające w pamięciach komputerów

ELWRO ang.

Lignos D.: Error detection and correction in mass storage equipment. Detekcja i korekcja błędów w pamięciach masowych. Computer Des. 1972 nr 10, s.71-75, rys.4, bibliogr.,poz.3.

Problem detekcji i korekcji błędów w pamięciach dyskowych wykorzystywanych w komputerach. Teoria korekcji błędów oparta na redundancji przesyłanej informacji oraz kryteria wyboru optymalnego algorytmu korekcji w celu uzyskania minimalnej stopy błędu. Przykłady technicznej realizacji ww algorytmów.

Urbanek A.

WYDAWNICTWA PRZEMYSŁU MASZYNOWEGO "WEMA"
oferują usługi wydawnicze

Od 5 lat działa w Warszawie specjalne wydawnictwo resortowe powołane do świadczenia usług wydawniczych na rzecz jednostek organizacyjnych resortu przemysłu maszynowego.

Do szczególnych zadań Wydawnictw Przemysłu Maszynowego "WEMA" należy:

- prowadzenie działalności wydawniczej zgodnie z potrzebami resortu,
- koordynacja działalności wydawniczej w jednostkach organizacyjnych resortu,
- koordynacja i nadzór nad prawidłowym wykorzystaniem maszyn i urządzeń poligraficznych,
- prowadzenie własnego ośrodka poligraficznego,
- prowadzenie ośrodka informacji wydawniczej.

Od ubiegłego roku Wydawnictwo znacznie rozszerzyło zakres usług i obecnie wydaje:
- katalogi branżowe i karty katalogowe

oraz na zlecenie przedsiębiorstw przemysłowych różnego rodzaju literaturę firmową, jak:
- katalogi zakładowe,
- katalogi części wymiennych,
- informatory techniczno-handlowe,
- dokumentacje techniczno-ruchowe, instrukcje obsługi i instrukcje naprawcze,
- dokumentacje techniczne kapitalnych remontów,
- wydawnictwo reklamowe, jak prospekty, foldery, ulotki itp.

Katalogi branżowe wydaje się w porozumieniu i we współpracy z właściwymi gestyjnie zjednoczeniami.

Sprzedażą katalogów WPM "WEMA" zajmują się następujące księgarnie:
Księgarnie "WSPÓLNEJ SPRAWY":
Warszawa, ul. Marszałkowska 28, tel. 21-66-60
Warszawa, ul. Marchlewskiego 35, tel. 20-49-69

"DOM KSIĄŻKI":
Główna Księgarnia Techniczna, Warszawa, ul. Świętokrzyska 14,
tel. 26-63-38.

Księgarnie te prowadzą sprzedaż odręczną i wysyłkową.

Literaturę firmową WPM "WEMA" wykonują na konkretne zamówienie przedsiębiorstw przemysłowych.

WPM "WEMA" znacznie skróciły cykle wydawnicze i zapewniają obecnie terminową realizację zamówień.

Wszelkich informacji na temat warunków przyjmowania i realizacji zamówień wydawniczych udziela Sekretariat Wydawnictwa, Warszawa, ul. Daniłowiczowska 18, pokój nr 7, tel. 27-49-47, skr. poczt. 90.

WYDAWNICTWA IMM

Branżowy Ośrodek Informacji Naukowo-Technicznej i Ekonomicznej Instytutu Maszyn Matematycznych wydaje:

ALGORYTMY - półrocznik; zawiera artykuły na temat teorii programowania i zastosowania elektronicznych maszyn cyfrowych. Do nabycia w księgarni ORWN PAN oraz w Domach Książki. Cena zeszytu 40,- zł.

PRACE IMM - 3 numery w roku, zawierają publikacje naukowe i badawcze pracowników IMM w zakresie projektowania i budowy elektronicznych maszyn cyfrowych oraz systemów przetwarzania informacji. Do nabycia w księgarni ORWN PAN oraz w Domach Książki. Cena zeszytu 60,- zł.

Elektroniczna Technika Obliczeniowa - NOWOŚCI - kwartalnik, zawiera artykuły przeglądowe z dziedziny maszyn matematycznych, opracowane na podstawie najnowszej literatury światowej. Prenumeratę prowadzi Centrala Kolportażu Prasy i Wydawnictw "RUCH". Cena prenumeraty krajowej 240,- zł rocznie.

Automatyzacja Przetwarzania Informacji - INFORMACJA EKSPRESOWA - miesięcznik. Prenumeratę prowadzi Centrala Kolportażu Prasy i Wydawnictw "RUCH". Cena prenumeraty krajowej 240,- zł rocznie.

Warunki prenumeraty

Cena prenumeraty krajowej:

rocznie - zł 240,-

Prenumerata przyjmowana jest do dnia 10 grudnia na rok następny.

Prenumeratę krajową dla czytelników indywidualnych przyjmują urzędy pocztowe oraz listonosze.

Czytelnicy indywidualni mogą dokonywać wpłat również na konto PKO Nr 1-6-100020 - Centrala Kolportażu Prasy i Wydawnictw "Ruch" Warszawa, ul. Wronia 23.

Wszystkie instytucje państwowe i społeczne mogą zamawiać prenumeratę wyłącznie za pośrednictwem Oddziałów i Delegatur "Ruch".

Prenumeratę ze zleceniem wysyłki za granicę, która jest o 40% droższa od krajowej, przyjmuje Biuro Kolportażu Wydawnictw Zagranicznych "Ruch" Warszawa, ul. Wronia 23 konto PKO Nr 1-6-100024 tel.20-46-88.

Cena zł 60,-