POLSKA AKADEMIA NAUK
CENTRUM BADAŃ NAUKOWYCH W WOJ. KATOWICKIM
ZAKŁAD SYSTEMÓW AUTOMATYKI KOMPLEKSOWEJ

PODSTAWY STEROWANIA

KWARTALNIK

Tom 6 — Zeszyt 4

GLIWICE — KRAKÓW — WARSZAWA 1976

TREŚĆ

T. Kamburelis, Współbieżna emulacja komputerowa 339
P. Bąk, Porównanie własności różnych metod ochrony przed wzajemną blokadą 359
W. Marek, T. Traczyk, Stochastyczne systemy informacyjne 383
R. Jakubowski, J. Rosek, Zastosowanie grafów funkcyjnych do opisu cząsteczek związków organicznych dla potrzeb automatycznego ich syntezowania 393
R. Jakubowski, A. Szelc, Grafy funkcyjne jako rozszerzenie sieci Petriego. I. Procesy markowania nad grafami funkcyjnymi 401
A. Mrózek, R. Pozowski, Synteza modeli matematycznych obiektów drogą analizy stanów charakterystycznych 413
K. M. Przyłuski, Obserwatory dla stacjonarnych układów z opóźnieniem . . 423
J. A. Barchanski, E Staicut, O implementacji Programu Kontroli Sieci Komputerowej 431

CONTENTS

T Kamburelis, Parallel Computer Emulation 339
P. Bąk, A Comparative Study of Anti-Deadlock Methods 359
W. Marek, T. Traczyk, Stochastic Informational Systems 383
R. Jakubowski, J. Rosek, Application of Functional Graphs for Description of Organic Chemical Molecules for Machine Generation of Synthetic Pathways 393
R. Jakubowski, A. Szelc, Functional Graphs as an Extension of Petri Nets. I. Marking Processes over Functional Graphs 401
A. Mrózek, R. Pozowski, Mathematical Models Synthesis by the Analysis of Plant Characteristic States 413
K. M. Przyłuski, An Observer Theory for Time Delay Systems 423
J. A. Barchanski, E. Staicut, Implementation of the Network Control Program for a Computer Network 431

СОДЕРЖАНИЕ

Т. Камбурелис, Параллельная компютерная емуляция 339
П. Бонк, Сравнение разных метод предупреждения взаимной блокаде 359
В. Марек, Т. Трачык, Стохастические информационные системы 383
Р. Якубовски, Я. Росек, Применение функционального графа для описания химической молекулы в целях её автоматического синтезирования. 393
Р. Якубовски, А. Шельц, Функциональные графы, как расширение сетей Петри. I. Маркировочные процессы над функциональными графами. 401
А. Мрузек, Р. Позовски, Синтез математических моделей объектов анализируя их характеристические состояния 413
К. М. Пшылуски, Теория наблюдателя для систем с запаздыванием. 423
Ю. А. Бархански, Е. Станцут, Об имплементации контрольной программы компютерной сети 431

POLSKA AKADEMIA NAUK
CENTRUM BADAŃ NAUKOWYCH W WOJ. KATOWICKIM
ZAKŁAD SYSTEMÓW AUTOMATYKI KOMPLEKSOWEJ

PODSTAWY STEROWANIA

KWARTALNIK

tom 6 — zeszyt 4

GLIWICE — KRAKÓW — WARSZAWA — 1976

PAŃSTWOWE WYDAWNICTWO NAUKOWE

Adres Redakcji

Zakład Systemów Automatyki Kompleksowej PAN
44-100 Gliwice, ul. Zwycięstwa 21, tel. 91-05-85

PAŃSTWOWE WYDAWNICTWO NAUKOWE
ODDZIAŁ W KRAKOWIE, Kraków, ul. Smoleńsk 14

Nakład 480+110 egz. Ark. wyd. 7,0; ark. druk. $7\,^{4}/_{16}$. Papier druk. sat. kl. III, 70×100, 80 g. Do składania oddano w sierpniu 1976 r. Podpisano do druku w grudniu 1976 r. Druk ukończono w grudniu 1976 r.
Zam. 837/76 P-23 Cena zł 15.—

DRUKARNIA UNIWERSYTETU JAGIELLOŃSKIEGO
Kraków, ul. Czapskich 4

PODSTAWY STEROWANIA
Tom 6 (1976), z. 4
pp. 339—358

Współbieżna emulacja komputerowa

THANASIS KAMBURELIS

(Maszynopis wpłynął 15 czerwca 1976)

1. Wstęp

Komputery budowane w przeszłości miały na ogół różne organizacje logiczne, czyli były wzajemnie niezgodne. Oznacza to, że posiadały różne struktury danych i rozkazów oraz odmienne zasady działania. Oprogramowanie systemowe i użytkowe różnych starych komputerów — pisane zwykle w językach niskiego poziomu — było także wzajemnie niezgodne. W związku z tym przy przejściu z jednego typu komputera na drugi (często tego samego producenta) zachodziła konieczność całkowitej zmiany oprogramowania, co pociągało za sobą istotny wzrost kosztów zastosowań. Takimi wzajemnie niezgodnymi komputerami w Polsce były np. maszyny: UMC-1, ODRA 1003, ODRA 1103, ODRA 1204, ODRA 1304, ZAM-41.

W celu zmniejszenia kosztów oprogramowania przy przejściu z jednego typu komputera na drugi, wprowadzono koncepcję rodziny komputerów, która składa się zwykle z kilku do kilkunastu modeli o zróżnicowanej wydajności (np. od 0,01 do 10 mln operacji na sekundę) i mających wzajemnie zgodne struktury danych oraz zgodne zasady działania. Dzięki temu oprogramowanie rodziny komputerów jest również wzajemnie zgodne. Znanymi w Polsce rodzinami komputerów są: ODRA 1300 (modele 1304, 1325 i 1305), ICL 1900, IBM 360 i 370, RIAD I i RIAD II.

Przyjęcie koncepcji rodziny komputerów pozwoliło producentom zachować przez około 10—15 lat pewną stabilizację w oprogramowaniu komputerów danej rodziny i wszechstronnie przygotować te komputery do różnych zastosowań. Jednakże koncepcje architektury logicznej rodzin komputerowych, które wprowadzono w życie w pierwszych latach dziesięciolecia 1960—69, stają się stopniowo przestarzałe i dlatego pojawiają się nowe rodziny komputerowe posiadające zupełnie nowe struktury i nowe cechy funkcjonalne.

Zatem problem przeniesienia dotychczasowego oprogramowania powstaje teraz przy przejściu z komputerów „starej" rodziny do komputerów „nowej" rodziny. W takim przypadku mamy do czynienia z oprogramowaniem nie jed-

nego komputera, lecz z oprogramowaniem całej rodziny komputerów o wielomiliardowej wartości.

Tak np. przeniesienie bogatego i zweryfikowanego przez praktykę oprogramowania komputerów serii ODRA 1300 do komputerów Jednolitego Systemu RIAD staje się zadaniem wielkiej wagi. Przeniesienie to powinno się odbywać metodami efektywnymi zarówno ze względu na dużą wartość użytkową oprogramowania maszyn ODRA 1300, jak i ze względu na długi i wieloletni okres potrzebny na przygotowanie równoważnego oprogramowania dla Elektronicznych Maszyn Cyfrowych Jednolitego Systemu RIAD (JS EMC RIAD).

Niniejszy artykuł jest próbą podania kilku efektywnych — zdaniem autora — metod organizacji logicznej środków technicznych przenoszenia programów z EMC ODRA 1300 do EMC Jednolitego Systemu.

Techniki przenoszenia programów

W problemach przenoszenia oprogramowania z jednego systemu komputerowego na drugi rozpatruje się zarówno zagadnienia przeniesienia samych programów, jak i zagadnienia przeniesienia danych.

W celu przeniesienia istniejących programowych systemów zastosowań stosuje się na ogół jedną z następujących ogólnych technik [2]:

1) Bezpośrednie przeniesienie programów na poziomie instrukcji maszynowych (np. zachowanie kompatybilności architektury logicznej, metody symulacji i emulacji). Technikę tę określa się również jako technikę modelowania działania starego komputera w nowym.

2) Translacja programów źródłowych z języka starego systemu komputerowego na zbliżony lub na ten sam język nowego komputera.

3) Przeprogramowanie danego systemu programowego, w którym zachowuje się logiczną strukturę systemu oraz jego algorytmy.

4) Przeprojektowanie danego systemu programowego, w którym zachowuje się jedynie pewne ogólne algorytmy działania systemu.

Powyższe techniki różnią się między sobą pracochłonnością przeniesienia danego programu, szybkością wykonywania programów w nowym systemie oraz wielkością wbudowanych środków technicznych przeznaczonych do interpretowania i wykonywania programów starej maszyny. Każda z tych technik ma swoje zalety i wady, które muszą być wzięte pod uwagę przy wyborze techniki przeniesienia danego systemu programowego z jednego komputera na drugi.

Przenoszenie programów na poziomie instrukcji maszynowych

Jedną z technik przeniesienia oprogramowania na poziomie instrukcji maszynowych są różne metody translacji programu, podobnie jak w przypadkach programów napisanych w językach wyższego poziomu; inną techniką są metody modelowania starego komputera w nowym komputerze. Celem translacji programu jest automatyczne wygenerowanie nowego programu, który

realizuje ten sam maszynowy algorytm liczenia w nowym komputerze. Natomiast w przypadku modelowania, maszyna modelująca zachowuje się w czasie wykonywania programu źródłowego tak jak maszyna modelowana. Zatem nie zachodzi potrzeba wcześniejszego przetłumaczenia programu źródłowego starej maszyny cyfrowej.

Metody translacji uwzględniają tylko początkową, tj. statyczną strukturę programu i na tej podstawie montują nowy program. Zatem montaż ten odbywa się bez uwzględnienia struktury dynamicznej programu wyjściowego. Z tego też względu wykonanie tak przetłumaczonego programu może przebiegać inaczej w nowej maszynie i tym samym może prowadzić do różnych wyników obliczeń. Oprócz wyżej wymienionych trudności, metoda translacji jest mało efektywna, gdyż w praktyce każdą instrukcję wyjściowej maszyny zmienia się przeważnie na kilka lub kilkanaście instrukcji nowej maszyny.

Technika modelowania jednej maszyny cyfrowej w drugiej na poziomie instrukcji maszynowych jest dokonywana za pomocą jednej z trzech metod:

a) metodą symulacji, gdzie wszystkie instrukcje starej maszyny mają programową interpretację;

b) metodą czystej emulacji, gdzie wszystkie funkcje starej maszyny mają sprzętową realizację;

c) metodą zwykłej emulacji, tj. zarówno środkami sprzętowymi, jak i programowymi.

Metoda symulacji zakłada, że zarówno interpretacja instrukcji, jak i wykonanie podstawowej funkcji dowolnej instrukcji symulowanej maszyny, odbywa się wyłącznie za pomocą odpowiednich procedur programowych. Metoda ta jest łatwa do realizacji w dowolnej EMC, lecz daje ogromne zwolnienie szybkości wykonania programu symulowanej EMC.

Natomiast metoda czystej emulacji zakłada, że interpretacja i wykonanie wszystkich instrukcji oraz innych funkcji emulowanej EMC (np. operacje wejścia-wyjścia) są realizowane w maszynie emulującej wyłącznie środkami sprzętowymi. Zatem jest to na ogół droga i mało elastyczna metoda.

Metoda zwykłej emulacji zakłada, że wykonywanie instrukcji i innych funkcji maszyny emulowanej odbywa się zarówno środkami sprzętowymi, jak i programowymi w emulującej maszynie. Wyboru instrukcji realizowanych środkami sprzętowymi dokonuje się, przeprowadzając odpowiednie analizy dotyczące częstotliwości stosowania tych instrukcji w programach.

Metoda zwykłej emulacji (zwana dalej po prostu emulacją) daje nie tylko zadowalające szybkości wykonania programu emulowanej maszyny, lecz także istotnie obniża ogólne koszty przeniesienia oprogramowania oraz w pełni wykorzystuje wszystkie możliwości nowej EMC.

Emulacja a systemy operacyjne

Systemy operacyjne pierwszych maszyn emulujących były specjalnie projektowane do sterowania pracą tylko programów emulowanych, natomiast

systemy operacyjne współczesnych maszyn emulujących sterują pracą zarówno programów własnych jak i emulowanych, w oparciu o zasady wieloprogramowości [1].

2. Emulacja zanurzeniowa EMC ODRA 1300 w EMC JS

Koncepcja emulacji zanurzeniowej polega na mikroprogramowej realizacji wszystkich podstawowych rozkazów maszyn cyfrowych serii ODRA 1300 w wybranym modelu JS EMC. Zatem jest to konwencjonalna mikroprogramowa metoda emulacji [7, 8, 9], w której oprócz rozszerzenia pamięci mikroprogramów o mikroprogramy emulacji, przewidziano dodatkowo specjalny blok sprzętowy zwany Podprocesorem Emulacji, którego celem jest zapewnienie możliwości opracowania efektywnych mikroprogramów emulacji.

Konieczność wprowadzenia Podprocesora Emulacji (PE) do struktury Procesora Głównego (PG), tj. do struktury emulującej maszyny JS EMC, wynikła z istotnie odmiennej struktury informacji i zasad działania systemów ODRA 1300 i JS EMC. Podprocesor PE jest przeznaczony głównie do szybkiego przekształcania struktury rozkazów, adresów i danych z postaci obowiązującej w EMC ODRA 1300 na postać wewnętrzną procesora PG.

Podprocesor PE został w pewien sposób „zanurzony" w strukturze procesora PG, przez zdefiniowanie zasad współpracy sprzętu podprocesora PE z pozostałymi blokami procesora PG. W ten sposób mikroprogramy emulacji korzystają ze wszystkich operacji i mechanizmów (np. sumatorów, rejestrów, sterowania mikroprogramowanego) procesora głównego oraz dodatkowo z operacji podprocesora PE.

W metodzie emulacji zanurzeniowej zakłada się istnienie następujących środków w emulującej maszynie Jednolitego Systemu:

— Mikroprogramowana struktura maszyny emulującej, której pamięć mikroprogramów (PAM) została powiększona o niezbędną pojemność (około 600 mikroinstrukcji) dla zapisania mikroprogramów emulacji. W ten sposób PAM zawiera jednocześnie zarówno mikroprogramy maszyny emulującej, jak i mikroprogramy maszyny emulowanej. Zatem maszyna emulująca posiada zdolność wykonywania zarówno programów własnych, jak i programów maszyny emulowanej (np. na zasadzie podziału czasu).

— Procesor główny o sterowaniu mikroprogramowym, którego repertuar mikrooperacji został rozszerzony o specjalne mikrooperacje dla emulacji (tzw. mikrooperacje emulacji).

— Podprocesor emulacji, tj. wydzielony blok układowy, który wyposażony jest we własne rejestry i we własne lokalne układy sterujące. Główną funkcją podprocesora PE jest rozszyfrowanie i wykonanie mikrooperacji emulacji, których kody dostarcza centralne sterowanie mikroprogramowe (tj. sterowanie procesora PG).

— Mikroprogramy dla interpretacji i wykonania funkcji podstawowych roz-

kazów maszyny emulowanej (w przypadku EMC ODRA 1300 zakłada się mikroprogramową realizację prawie całej listy rozkazów użytkowych).

— Rozszerzony repertuar instrukcji maszyny emulującej o nowe instrukcje (nieuprzywilejowane) dla wywoływania odpowiednich mikroprogramów realizujących pewne operacje często występujące w podprogramach interpretacyjnych programu Emulator.

— Wyróżniona nieuprzywilejowana instrukcja należąca do zbioru instrukcji maszyny emulującej — dla inicjowania pracy programu emulowanego, tj. dla rozpoczęcia wykonania sekwencji rozkazów według stanu licznika rozkazów maszyny emulowanej.

— Mechanizmy przerywania programu emulowanego i przejścia do specjalnego programu Emulator, gdy w bieżącym programie emulowanym pojawi się rozkaz nie mający realizacji technicznej w środowisku maszyny emulującej. Innymi słowy zakłada się, że te rozkazy będą wykonywane przez odpowiednie podprogramy interpretacyjne należące do programu Emulator.

Powyższe ogólne wymagania mogą być zrealizowane bez większych trudności w każdej nowo opracowanej mikroprogramowanej maszynie Jednolitego Systemu.

Przedstawiona szkicowo metoda emulacji zanurzeniowej znajduje się obecnie na etapie opracowań logiczno-konstrukcyjnych w zastosowaniu do maszyn ODRA 1300-R45 [3, 4, 5, 6].

3. Emulacja autonomiczna i współbieżna

Przedstawiona koncepcja emulacji zanurzeniowej polega na zrealizowaniu środkami technicznymi w maszynie „gościnnej" funkcji maszyny emulowanej. W szczególności oznacza to wykorzystanie tego samego procesora i to samo sterowanie pamięcią mikroprogramów.

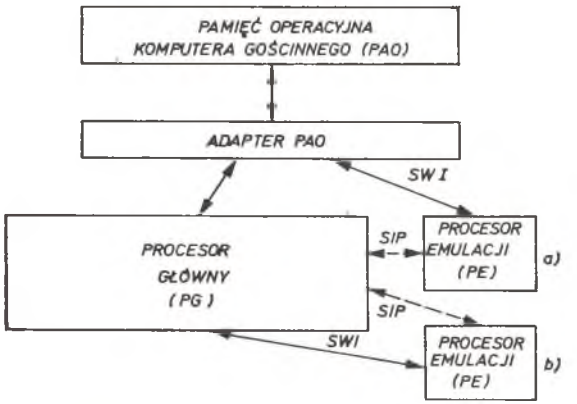
Zaletą emulacji zanurzeniowej jest to, że nie wymaga ona dużych dodatkowych środków sprzętowych. Z projektu emulacji EMC ODRA 1300, według zaproponowanej metody zanurzeniowej, w nowo opracowanej EMC JS-1045 wynika, że podprocesor PE zajmie tylko dwa standardowe pakiety układów logicznych oraz około 600 mikroinstrukcji.

Wadą natomiast metody emulacji zanurzeniowej jest konieczność opracowania każdorazowo odrębnego projektu i dokumentacji emulacji danej starej maszyny dla wybranego modelu procesora nowego systemu komputerowego.

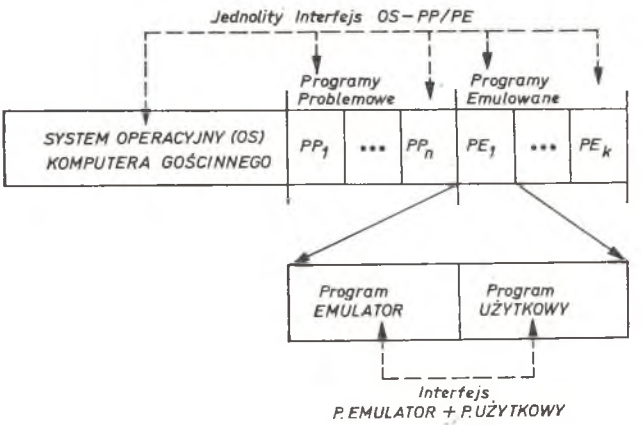
Powstaje zatem zagadnienie, czy nie można opracować autonomicznego, prostego i uniwersalnego procesora (PE) do wykonania funkcji emulacji, który byłby przyłączalny do większej klasy jednostek centralnych (PG) nowego systemu komputerowego? W pracy [3] podano projekty organizacji współpracy takiego procesora z jednostkami centralnymi rodziny EMC Jednolitego Systemu.

Jeden z projektów przewiduje, że procesor PE funkcjonuje niezależnie i jedno-

cześnie z procesorem głównym PG komputera gościnnego (rys. 1a). Oznacza to możliwość współbieżnego wykonania dwóch niezależnych sekwencji rozkazów (lub instrukcji). Jedna sekwencja sterowana będzie licznikiem rozkazów procesora PE (pobieranych z PAO z pominięciem układów PG), a druga licznikiem instrukcji procesora PG (by pozostać w terminologii systemów ODRA 1300 i JS EMC). Czyli współbieżność pracy procesorów PE i PG prowadzi do



Rys. 1. Ogólny układ procesorów PE i PG. SWI — Szyna Wymiany Informacji (Danych i Rozkazów), SIP — Szyna Inicjacji i Przerywania pracy programu emulowanego



Rys. 2. Ogólne rozmieszczenie n programów problemowych komputera gościnnego i k programów emulowanych

zorganizowania specjalnego niejednorodnego systemu dwuprocesorowego zarówno w rozwiązaniach architektury logicznej, jak i w rozwiązaniach programowych. W ten sposób dwa programy byłyby jednocześnie w czasie wykonywane (jeden program typu maszyny emulowanej i jeden maszyny emulującej).

Drugi projekt zakłada, że procesor PE funkcjonuje autonomicznie, tzn. realizuje niezależną sekwencję rozkazów według własnego licznika rozkazów,

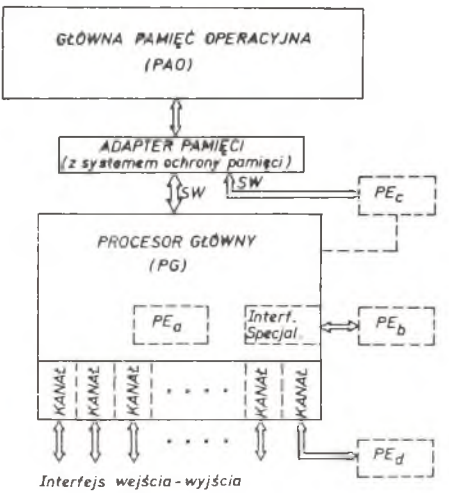
ale nie współbieżnie w czasie z procesorem PG, gdyż pobiera rozkazy i dane za pośrednictwem procesora PG (rys. 1b).

Zakłada się, że procesory PE i PG realizowałyby na przemian funkcje programu emulowanego (rys. 2). Program Emulator, który jest pisany w instrukcjach komputera gościnnego, realizowany jest w procesorze PG, zaś program użytkowy (tj. właściwy program emulowany) w procesorze PE. W rozważaniach emulacji współbieżnej i autonomicznej największe trudności napotkano przy zdefiniowaniu takiej architektury logicznej i takich zasad współpracy procesorów PE i PG — z jednej strony — oraz procesora PE i PAO z drugiej strony, które to zasady synchronizowałyby współfunkcjonowanie dwóch systemów komputerowych (np. ODRA 1300 i JS EMC) o różnych zasadach działania i o różnych koncepcjach rozwiązań programowych.

4. Emulacja wirtualna

4.1. Wprowadzenie

Metody emulacji zanurzeniowej, autonomicznej i współbieżnej (opisane w poprzednich rozdziałach) zakładały zawsze, że główny komputer emulujący PG zostanie wyposażony w dodatkowe środki techniczne biorące bezpośredni udział w realizacji programu emulowanego (rys. 3).



Rys. 3. Rozważane lokalizacje procesora emulacji względem procesora głównego komputera emulującego. SW — Szyna systemu Wieloprocesorowego, PE — Procesor Emulacji, PE$_a$ — emulacja zanurzeniowa, PE$_b$ — autonomiczna, PE$_c$ — współbieżna, PE$_d$ — wirtualna

W przypadku metody emulacji zanurzeniowej tymi dodatkowymi środkami technicznymi były: podprocesor emulacji oraz mikroprogramy realizujące funkcje rozkazów maszyny emulowanej i rezydujące w pamięci sterującej ma-

szyny głównej. W tej metodzie dodatkowe środki techniczne były silnie zintegrowane z podstawowym sprzętem procesora PG (np. przez stosowanie tych samych rejestrów roboczych, tych samych zespołów przetwarzających oraz tego samego repertuaru mikrooperacji).

W przypadku natomiast metody emulacji autonomicznej dodatkowymi środkami technicznymi były:

— mikroprogram instrukcji inicjacji, asystowania i przerywania programu emulowanego oraz

— nowo zdefiniowany specjalny interfejs 24-bitowy współpracy procesorów PG i PE. Wielkość sprzętu dodatkowego (w procesorze PG) potrzebnego w metodzie emulacji autonomicznej jest istotnie mniejsza w porównaniu z emulacją zanurzeniową, jednakże sprzęt ten jest dalej ściśle zintegrowany ze sprzętem procesora PG.

W przypadku emulacji współbieżnej środkami dodatkowymi byłyby dodatkowa szyna współpracy procesora PE z adapterem wspólnej pamięci operacyjnej.

W ostatnim przypadku powiązania pomiędzy PE i PG są luźniejsze. Ponadto tą dodatkową szyną, przez którą pobiera się rozkazy i dane programu emulowanego do procesora PE, jest często szyna standardowa przewidziana do pracy wieloprocesorowej. Jednakże przydzielenie tej szyny procesorowi emulacji oznacza często zmniejszenie lub nawet zlikwidowanie możliwości jednorodnej pracy wieloprocesorowej na poziomie procesorów głównych.

Powstaje zatem pytanie czy nie można opracować takiej metody efektywnej emulacji, która by nie wymagała dokonania jakichkolwiek zmian technicznych w środowisku komputera głównego. Wielkie zalety użytkowe tej metody są oczywiste, gdyż pozwalają w dowolnej chwili wyposażyć komputery będące w eksploatacji w zdolność emulacji efektywnej.

W tym artykule zostanie przedstawiona pewna metoda takiej emulacji (współbieżnej), którą umownie nazwiemy emulacją wirtualną, ze względu na sposób organizacji pamięci roboczej zawierającej dane i programy emulowane. Trzeba od razu wyjaśnić, że termin emulacja wirtualna nie oznacza, że komputer główny musi być wyposażony w konwencjonalną pamięć wirtualną [10]. W emulacji wirtualnej zakłada się, że:

a) zostanie opracowany prosty i niezależnie działający procesor (PE) do wykonywania programów maszyny emulowanej,

b) procesor PE zostanie wyposażony w małą pamięć lokalną PAL (np. 1÷4 K słów), w której przechowywać się będzie tylko aktywne fragmenty wykonywanych programów (i danych) maszyny emulowanej,

c) procesor PE będzie współpracował z procesorem głównym PG za pośrednictwem standardowego interfejsu wejścia-wyjścia maszyny emulującej (tj. za pośrednictwem szybkiego kanału),

d) zostanie opracowany algorytm minimalizujący liczbę transmisji bloków informacji przesyłanych pomiędzy dwoma poziomami pamięci roboczej — tj. głównej pamięci operacyjnej procesora PG i pamięci lokalnej procesora PE —

oraz minimalizujący łączny czas wykonania programu emulowanego przy zadanych ograniczeniach (szybkość procesorów PG i PE, szybkość kanału, pojemność PAL, itp.),

e) nie dokona się żadnych zmian zarówno w sprzęcie, jak i w systemie operacyjnym maszyny emulującej. Opracuje się natomiast tylko program Emulator (na poziomie programów problemowych), który spełnia te same funkcje co w innych metodach emulacji. Program Emulator organizuje również tzw. program kanałowy (tj. informacje sterujące przesyłaniem danych przez kanał).

Organizacja pamięci operacyjnej i lokalnej

W kolejnych artykułach rozważy się trzy warianty organizacji pamięci lokalnej (PAL) procesora PE oraz głównej pamięci operacyjnej (PAO) procesora PG. Rozważane warianty noszą następujące nazwy:
metoda przydziału elementów macierzy (lub metoda macierzowa),
metoda dynamicznego przydziału stronic (lub metoda stronicowa),
metoda przydziału stałych stref dla rozkazów i danych oraz stałego stosu dla wyników obliczeń.

4.2. Metoda macierzowa

W metodzie macierzowej zakłada się, że cała pamięć operacyjna procesora PG jest zorganizowana w formie macierzy M[PAO] składającej się z m wierszy i n kolumn (rys. 4). Zawartość jednego pola tej macierzy nazywamy blokiem. Ilość słów bloku oznaczymy przez b. Zatem rozważana maksymalna pojemność PAO wynosi mnb-słów.

W przypadku EMC Jednolitego Systemu słowo składa się z 32 bitów informacyjnych (tj. z czterech bajtów), w którym zanurzono 24-bitowe słowo EMC ODRA 1300.

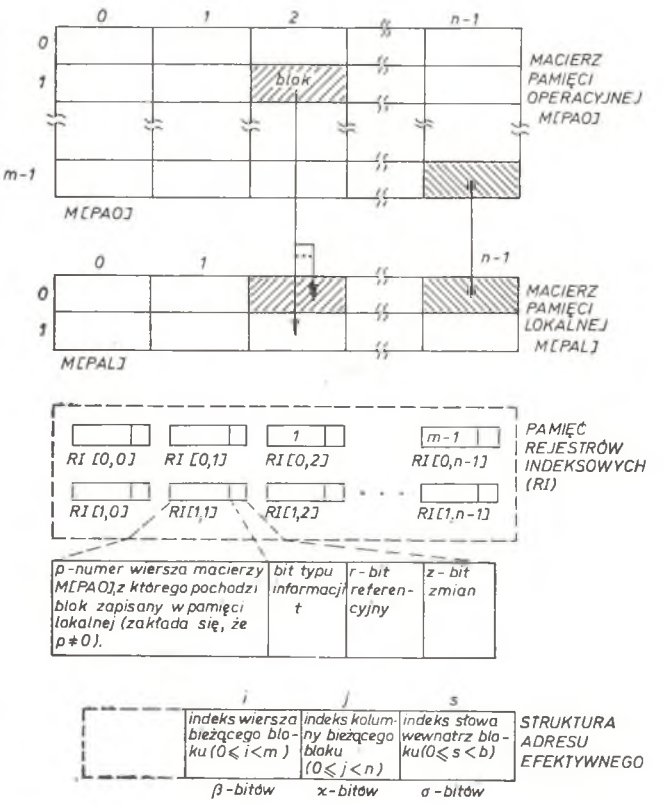
Zakłada się, że macierz M[PAO] jest ogólnie macierzą niekwadratową ($m \neq n$), gdzie wielkość n jest stała w danej maszynie, zaś m jest zmienna (zależnie od wielkości aktualnie zainstalowanej pamięci operacyjnej).

Pamięć lokalna procesora emulacji PE jest również zorganizowana w formie prostokątnej macierzy M[PAL] mającej dwa wiersze i n kolumn. Zatem macierz M[PAL] ma tyle samo kolumn co macierz M[PAO]. Zawartość jednego pola (bloku) macierzy M[PAL] wynosi również b słów.

Podstawową funkcją małej pamięci lokalnej PAL jest przechowywanie w niej tych bloków programu emulowanego (rezydującego w PAO), z których pochodzą aktualnie wykonywane (w procesorze PE) sekwencje rozkazów i danych (czyli bloków aktywnych). Zakłada się w tej metodzie, że bloki rozkazów i danych są pobierane z PAO do PAL w sposób „pionowy", tzn. blok i-tego wiersza oraz j-tej kolumny pamięci PAO, gdzie $0 \leqslant i < m$ oraz $0 \leqslant j < n$, zostaje wpisany w polu k-tego wiersza i j-tej kolumny pamięci PAL (gdzie $k = 0$ lub 1). Tak określone pobieranie bloku informacji będziemy skrótowo oznaczać przez

zapis PAL$[k,j]$: = PAO$[i,j]$. Analogicznie zapis PAO$[i,j]$: = PAL$[k,j]$ będzie oznaczać zapamiętanie bloku informacji z pamięci PAL do pamięci PAO.

W przyjętej strukturze procesora emulacji PE wprowadzamy dodatkową tzw. Pamięć Indeksową (PAI), która składa się z $2n$-rejestrów $l+3$-bitowych, gdzie $2^l = m$. Każdy rejestr pamięci indeksowej zwany Rejestrem Indeksu RI, jest jednoznacznie związany z jednym elementem macierzy M[PAL]. Rejestr pamięci indeksowej związany z blokiem PAL$[k,j]$ pamięci lokalnej oznaczać będziemy przez RI$[k,j]$.



Rys. 4. Struktura pamięci w emulacji wirtualnej w oparciu o metodę macierzową

Rolą rejestrów RI jest przechowywanie w lewych l-bitach indeksów wierszy bloków informacji pamięci operacyjnej aktualnie zapisanych w pamięci lokalnej procesora emulacji oraz rejestracja zmian (bit z) informacji, odwoływań do informacji (bit r) lub rejestracji typu przechowywanej informacji (bit t) w bloku. Zatem zawartość rejestru RI$[k,j] = \langle i,t,r,z\rangle$ oznacza, że blok PAO$[i,j]$ z pamięci operacyjnej został zapisany w polu PAL$[k,j]$ pamięci lokalnej. Ponadto jeśli parametr z równa się 0, to informacja pola PAL$[k,j]$ jest zgodna z zawartością pola PAO$[i,j]$; jeśli zaś $z = 1$, to rozważane pola mają różne informacje. Parametr r przyjmuje zero przy ładowaniu danego bloku do PAL,

zaś jedynkę po odwołaniu się procesora PE do dowolnego słowa tego bloku (tj. po odczycie lub zapisie słowa). Natomiast parametr t równa się 0, gdy dany blok został pobrany jako blok rozkazów (tj. według Licznika Rozkazów) lub 1 w przeciwnym przypadku. Operacja np. pobrania rozkazów $\mathrm{PAL}[k,j] := \mathrm{PAO}[i,j]$ powoduje wpisanie czwórki liczb $\langle i,0,0,0\rangle$ do rejestru $\mathrm{RI}[k,j]$, zaś operacja zapisu w polu $\mathrm{PAL}[k,j]$ przez procesor PE powoduje wpisanie jedynki w bitach r i z rejestru $\mathrm{RI}[k,j]$. Informacje t,r,z są wykorzystywane w algorytmach ładowania pamięci PAL oraz przechowywania stanu tej pamięci.

Inicjowanie właściwego programu emulowanego przez zastartowanie procesora PE jest dokonywane przy pomocy programu Emulator. Program Emulator przygotowuje również informacje sterujące związane z transmisją bloku informacji programu emulowanego z PAO do PAL (lub z PAL do PAO). Współrzędne pola macierzy $M[\mathrm{PAO}]$, do którego (lub z którego) ma być transmitowany dany blok informacji, określa procesor PE. Procesor PE określa również numer wiersza pola pamięci PAL, w którym dany blok ma być zapisany (lub z którego ma być odczytany), według specjalnego algorytmu realizowanego sprzętowo.

Jeśli żądany blok informacji leży w polu $\mathrm{PAO}[i,j]$ i ma być przesłany do pamięci lokalnej PAL, to wykonuje się następujące funkcje:

1. Procesor PE zgłasza i przesyła do procesora PG (tj. do programu Emulator) współrzędne i oraz j.

2. Program Emulator organizuje transmisję do procesora PE bloku informacji (b-słów) z i-tego wiersza oraz j-tej kolumny pamięci PAO.

3. Procesor PE zapisuje transmitowany blok w j-tej kolumnie pamięci PAL i w k-tym wierszu tej pamięci, gdzie k jest określony następującym algorytmem przydziału wiersza pamięci lokalnej:

a) jeśli oba wiersze j-tej kolumny pamięci lokalnej są wolne, to przyjmuje się, że $k=0$. Dokładnie: jeśli $\mathrm{RI}[0,j]=\mathrm{RI}[1,j]=\langle 0,0,0,0\rangle \Rightarrow \mathrm{PAL}[0,j] := \mathrm{PAO}[i,j] \wedge \mathrm{RI}[0,j] := \langle i,t,0,0\rangle$, gdzie $t=0$ lub 1 zależnie od rodzaju pobranej informacji z PAO ($t=0$ dla rozkazów lub $t=1$ dla danych).

b) Jeżeli wiersz x (gdzie $x=0$ lub 1) pamięci PAL jest wolny, zaś wiersz $\tilde{x}$ jest zajęty, to przyjmuje się, że $k=x$. Dokładnie: jeśli $\mathrm{RI}[x,j] = \langle 0,0,0,0\rangle \wedge \mathrm{RI}[\tilde{x},j] = \langle \neq 0, t_{\tilde{x}}, r_{\tilde{x}}, z_{\tilde{x}}\rangle \Rightarrow \mathrm{PAL}[x,j] := \mathrm{PAO}[i,j] \wedge \mathrm{RI}[x,j] := \langle i,t,0,0\rangle$.

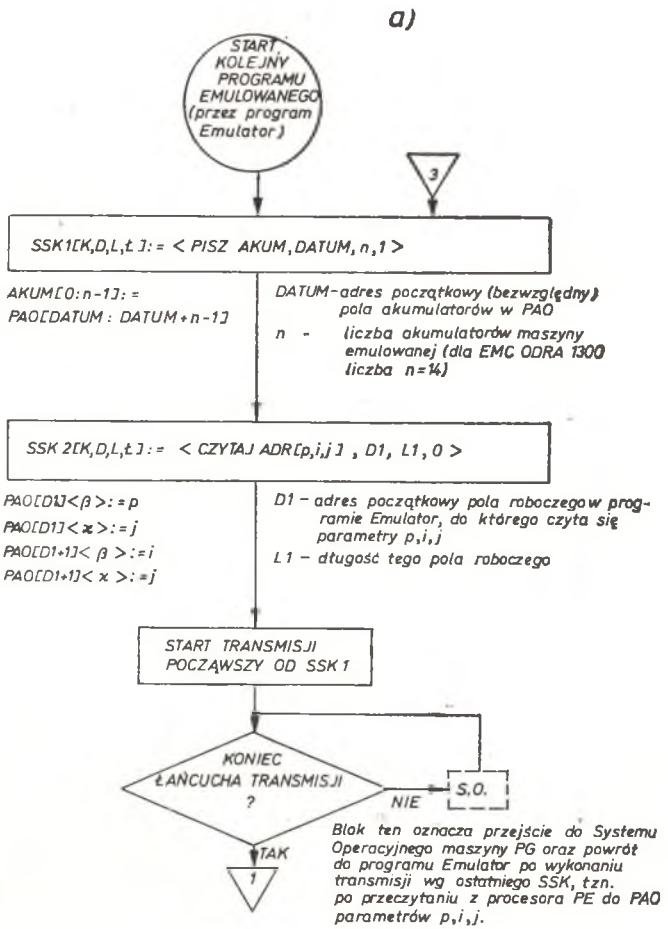
c) Jeśli oba wiersze pamięci PAL są zajęte i zawierają jednakowy typ informacji, to zmienna k przyjmuje numer wiersza dla którego bit referencyjny ma mniejszą wartość. Dokładnie: jeśli $\mathrm{RI}[0,j]=\langle \neq 0, t_0, r_0, z_0\rangle \wedge \mathrm{RI}[1,j] = \langle \neq 0, t_1, r_1, z_1\rangle \wedge t_0 = t_1 \Rightarrow \mathrm{PAL}[0,j] := \mathrm{PAO}[i,j] \wedge \mathrm{RI}[0,j] := \langle i,t,0,0\rangle$ dla $r_0 \leqslant r_1 \vee \mathrm{PAL}[1,j] := \mathrm{PAO}[i,j] \wedge \mathrm{RI}[1,j] := \langle i,t,0,0\rangle$ dla $r_0 > r_1$.

d) Jeśli oba wiersze pamięci PAL są zajęte i zawierają różne typy informacji, to k przyjmuje numer wiersza, w którym znajduje się zgodny typ informacji z typem pobieranej informacji z PAO (czyli nowy blok rozkazów usuwa stary blok rozkazów, zaś nowy blok danych usuwa stary blok danych z PAL). Do-

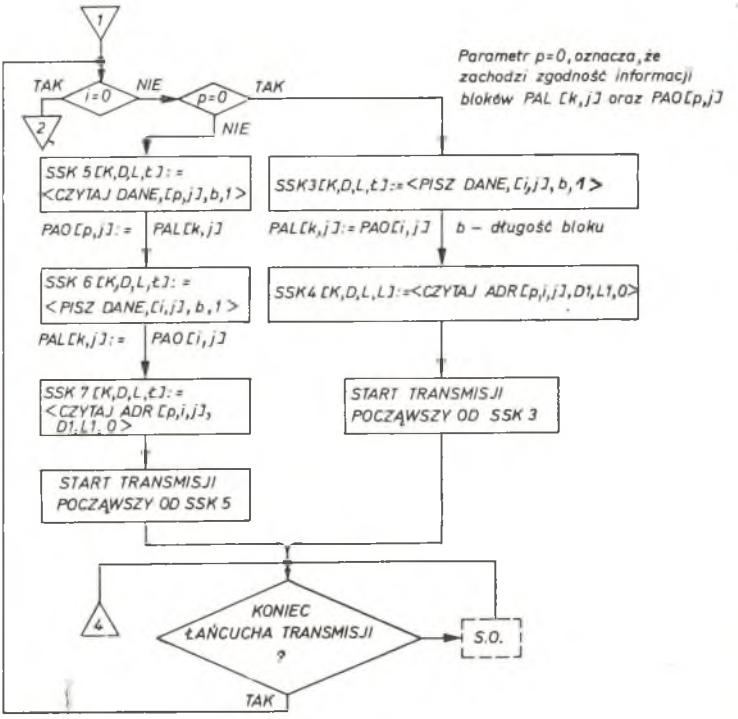
kładnie: jeśli $\mathrm{RI}[0,j] = \langle \neq 0, t_0, r_0, z_0 \rangle \wedge \mathrm{RI}[1,j] = \langle \neq 0, t_1, r_1, z_1 \rangle \wedge t_0 \neq t_1 \Rightarrow \mathrm{PAL}[0,j] := \mathrm{PAO}[i,j] \wedge \mathrm{RI}[0,j] := \langle i, t, 0, 0 \rangle$ dla $t = t_0 \vee \mathrm{PAL}[1,j] := \mathrm{PAO}[i,j] \wedge \mathrm{RI}[1,j] := \langle i, t, 0, 0 \rangle$ dla $t = t_1$ (gdzie przez t oznaczono typ bieżącego bloku informacji). Powyższe rozwiązanie oparto o założenie sekwencyjności pobierania rozkazów i danych programu emulowanego.

4. Procesor PE zeruje bity referencyjne r_0 i r_1 rejestrów indeksowych $\mathrm{RI}[0,j]$ i $\mathrm{RI}[1,j]$, po czym kontynuuje wykonywanie programu emulowanego.

Przedstawiony algorytm ładowania pamięci PAL może spowodować zniszczenia starej informacji w PAL — przypadek 3c i 3d — której kopia w PAO jest różna (wówczas bit $z = 1$). W związku z tym przed wykonaniem operacji typu $\mathrm{PAL}[k,j] := \mathrm{PAO}[i,j]$ wykonuje się operację postaci $\mathrm{PAO}[p,j] := \mathrm{PAL}[k,j]$ w przypadku $z = 1$, gdzie p jest starą zawartością rejestru $\mathrm{RI}[k,j]$. W tym celu zakłada się, że procesor PE przesyła do procesora PG (wraz z żądaniem transmisji nowego bloku do PAL) dwie pary współrzędnych macierzy $M[\mathrm{PAO}]$, tj. współrzędne $[p,j]$ bloku do przechowania w PAO z PAL oraz współrzędne $[i,j]$ bloku do przesłania z PAO do PAL.



b)



Algorytmy inicjacji, wymiany bloków i zakończenia programu emulowanego przedstawiono na (rys. 5 i 6). W opisie proponowanego algorytmu przyjmujemy następujące oznaczenia:

SSK[K, D, L, $Ł$] — oznacza słowo sterujące kanału, tj. opis pola pamięci operacyjnej PAO o początkowym adresie bezwzględnym D i długości L słów, z którego (do którego) przysyła się blok informacji do (z) pamięci PAL. Litera K oznacza rodzaj komendy wprowadzanie-wyprowadzanie informacji i przyjmuje operacje:

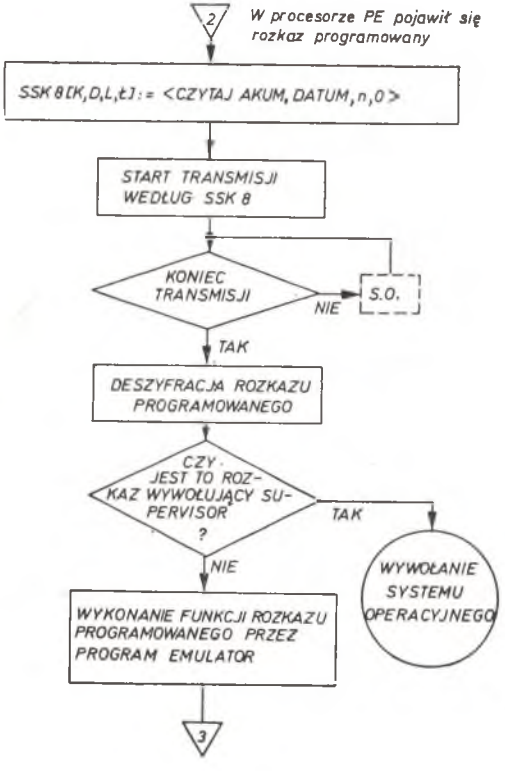
K:: = ⟨PISZ DANE⟩|⟨CZYTAJ DANE⟩|⟨PISZ AKUM⟩|⟨CZYTAJ AKUM⟩|⟨CZYTAJ ADR⟩|⟨ZERUJ RI⟩|⟨CZYTAJ RI⟩|⟨CZYTAJ PAL⟩

Litera $Ł$ oznacza Łańcuch Komend, tj, ciąg operacji wejścia-wyjścia (wówczas $Ł = 1$). W tym przypadku po wykonaniu operacji określonej przez bieżące SSK przechodzi się automatycznie do wykonania operacji opisanej przez kolejne SSK. $Ł$:: = ⟨0⟩|⟨1⟩.

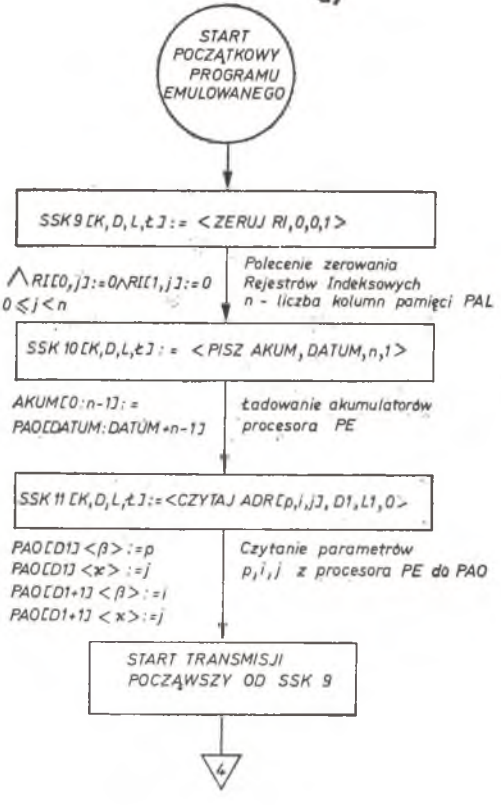
AKUM[0 : n] — oznacza zestaw wyróżnionych rejestrów AKUMulatorowych procesora PE ponumerowanych od 0 do n. W maszynach ODRA 1300 [4] rejestrami akumulatorowymi są:

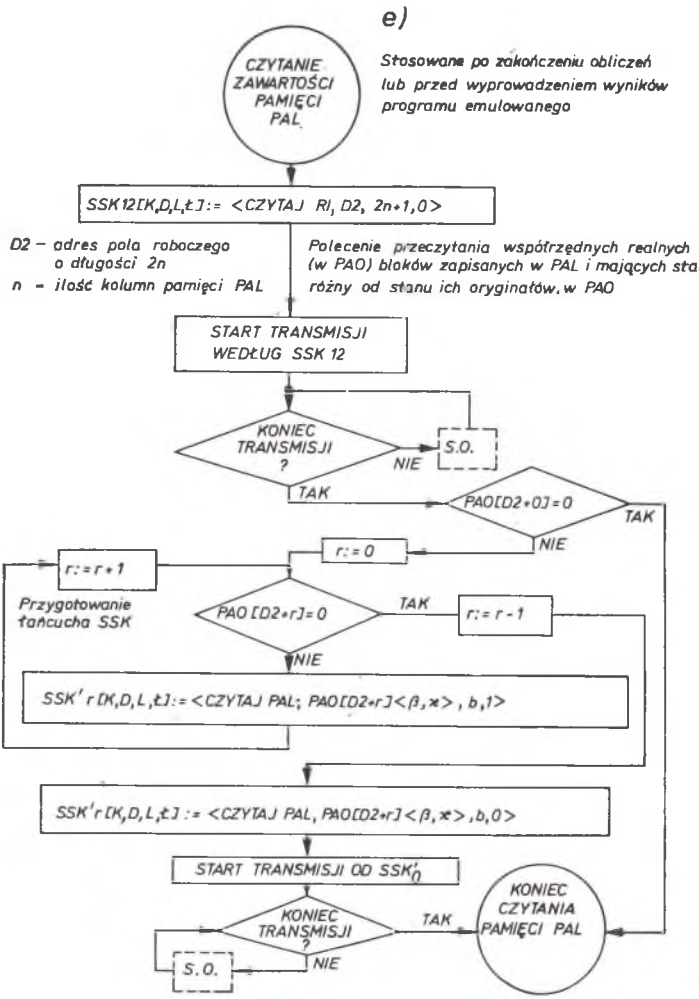
AKUM[0 : 7] — akumulatory XO ÷ X7
AKUM[8] — licznik rozkazów (LR)

c)


2
W procesorze PE pojawił się rozkaz programowany
SSK 8 [K,D,L,t] := <CZYTAJ AKUM, DATUM, n, 0>
START TRANSMISJI WEDŁUG SSK 8
KONIEC TRANSMISJI
NIE
S.O.
TAK
DESZYFRACJA ROZKAZU PROGRAMOWANEGO
CZY JEST TO ROZKAZ WYWOŁUJĄCY SUPERVISOR ?
TAK
NIE
WYWOŁANIE SYSTEMU OPERACYJNEGO
WYKONANIE FUNKCJI ROZKAZU PROGRAMOWANEGO PRZEZ PROGRAM EMULATOR
3

d)


START POCZĄTKOWY PROGRAMU EMULOWANEGO
SSK 9 [K, D, L, t] := <ZERUJ RI, 0, 0, 1>
∧ RI[0, j] := 0 ∧ RI[1, j] := 0
0 ≤ j < n
Polecenie zerowania Rejestrów Indeksowych
n - liczba kolumn pamięci PAL
SSK 10 [K, D, L, t] := <PISZ AKUM, DATUM, n, 1>
AKUM[0 : n-1] :=
PAO[DATUM : DATUM + n-1]
ładowanie akumulatorów procesora PE
SSK 11 [K, D, L, t] := <CZYTAJ ADR [p, i, j], D1, L1, 0>
PAO[D1] <β> := p
PAO[D1] <α> := j
PAO[D1+1] <β> := i
PAO[D1+1] <α> := j
Czytanie parametrów p, i, j z procesora PE do PAO
START TRANSMISJI POCZĄWSZY OD SSK 9
4



Rys. 5a, b, c, d, e. Algorytmy inicjowania programu emulowanego i wymiany bloków pamięci pamięciami PAO i PAL (funkcje realizowane w procesorze PG)

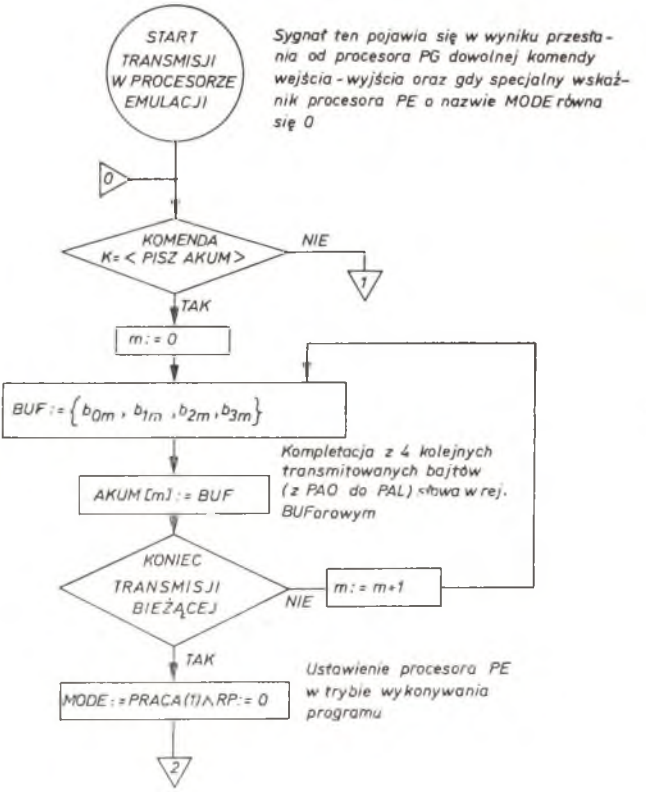
AKUM[9] — adres bazowy programu (DATUM) określający początek programu emulowanego w PAO

AKUM[10] — słowo rozkazu programowanego (ekstrakodowego)

AKUM[11] — adres efektywny rozkazu programowanego

AKUM[12 : 13] — akumulator zmiennoprzecinkowy (A).

ADR$[p, i, j]$ — oznacza dwa różne ADResy wierszy pól (p-ty oraz i-ty) leżących w j-tej kolumnie pamięci PAO. Adresy te ustala procesor emulacji PE i wykorzystywane są w operacjach transmisji postaci PAO$[p, j] :=$ PAL$[k, j]$ oraz PAL$[k, j] :=$ PAO$[i, j]$, gdzie k jest ustalany algorytmem przydziału wiersza pamięci lokalnej. Adres p jest określany zawartością rejestru indekso-



Rys. 6a. Funkcja operacji PISZ AKUM (realizowana sprzętowo w procesorze PE)

wego RI[k, j], gdy bit z (zmian) tego rejestru równa się 1. Jeśli zaś bit $z = 0$, to przyjmuje się, że $p = 0$. Ponadto zakłada się, że jeśli $i \neq 0$, to oznacza numer wiersza macierzy M[PAO]. Jeśli natomiast $i = 0$, oznacza to, że w procesorze PE pojawił się rozkaz programowany (tzn. rozkaz nie mający realizacji technicznej w procesorze PE).

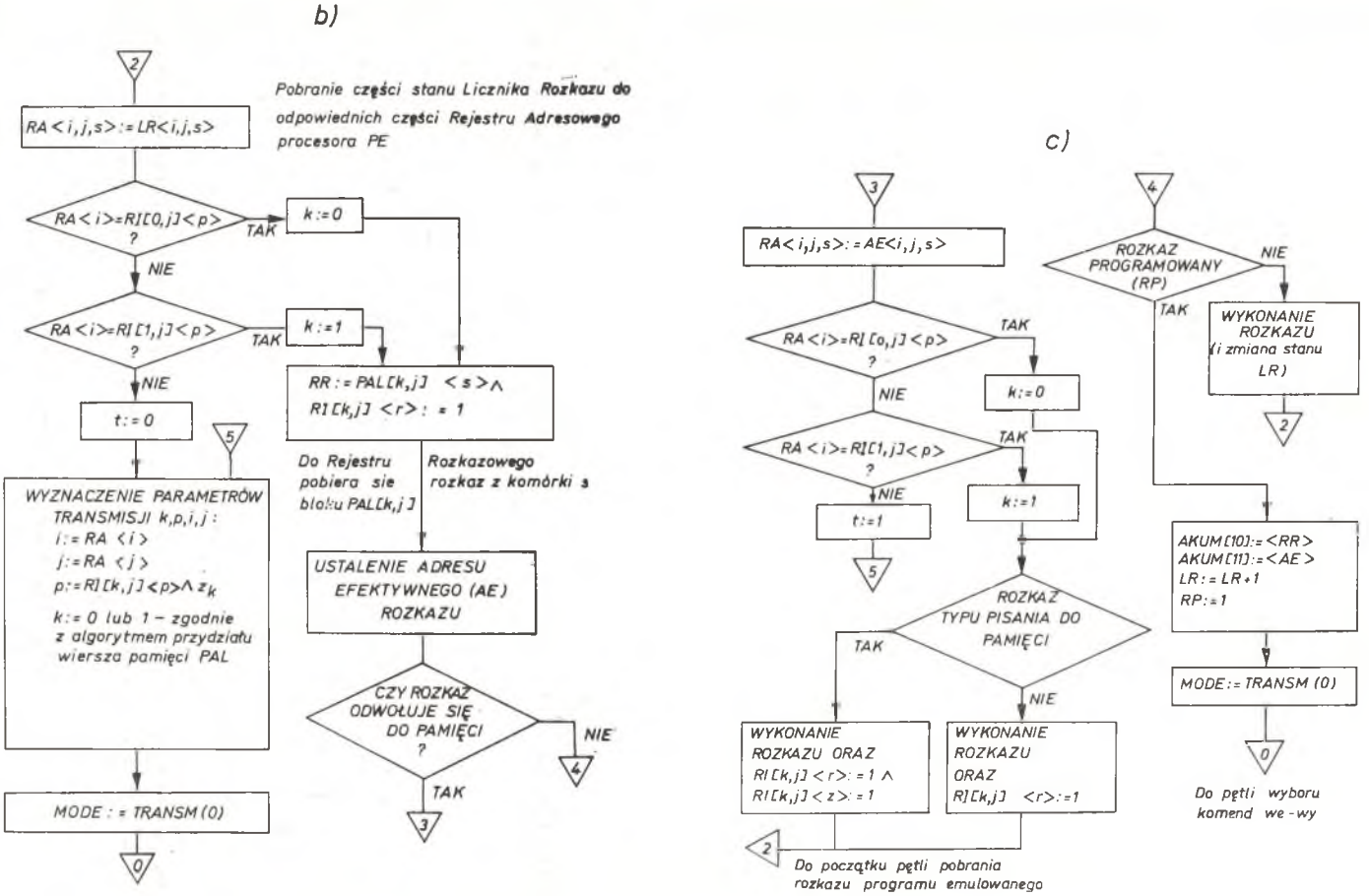
RI[k, j]$\langle p \rangle$ — zawartość rejestru RI[k, j] zapisana w części p tego rejestru

PAL[k, j]$\langle s \rangle$ — zawartość słowa o adresie s bloku PAL[k, j]

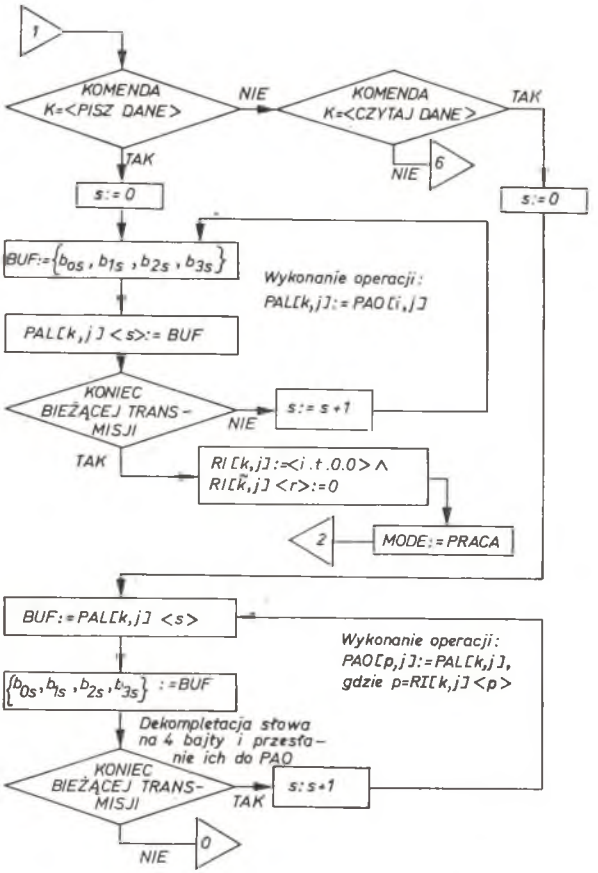
BUF$\langle x \rangle := y$ — oznacza, że wielkość y zostaje wpisana w polu x rejestru BUF (jeśli ładowany rejestr składa się z kilku części oraz ładuje się jednocześnie wszystkie jego części, to opuszcza się ich oznaczenia).

5. Podsumowanie i wnioski

Poruszone w tej pracy problemy emulacji komputerowej wynikają z bardzo istotnych potrzeb dalszego rozwoju naszej informatyki. Warto przypomnieć, że komputery serii ODRA 1300 są obecnie najbardziej rozpowszechnionymi



Rys. 6b, c. Algorytmy wykonywania programu zapisanego w PAL



Rys. 6d. Funkcje operacji PISZ DANE i CZYTAJ DANE (realizowane sprzętowo w procesorze PE)

w Polsce maszynami. Będzie istniało zatem duże zapotrzebowanie na przeniesienie danych i oprogramowania tych maszyn na EMC Jednolitego Systemu.

W związku z tym przygotowuje się w Centrum MERA-ELWRO wiele opracowań programowych i technicznych, których celem jest zapewnienie przenoszenia danych i programów z EMC serii ODRA 1300 na EMC Jednolitego Systemu [3].

Parallel Computer Emulation

The paper presents effective methods of logic organization of hardware means that provide transfer of programs from ODRA 1300 series computers to unifed system RIAD computers.

Параллельная компютерная эмуляция

В работе представлено некоторые эффективные методи логической организации технических средств переноса мат-обеспечения вычислительных машин серии ODRA 1300 в Единую Систему ЭВМ Риад (РИАД).

Literatura

[1] R. Cary, *System/370 Integration Emulation under OS and DOS*, AFIPS, vol. 38, 1971.
[2] S. Husson, *Programming, Principles and Practices*, Prentice-Hall, Inc. 1970.
[3] Th. Kamburelis, *Projekty organizacji logicznej emulacji EMC ODRA 1300 w EMC Jednolitegb Systemu*, MERA-ELWRO (raport wewnętrzny), Wrocław 1976.
[4] Th. Kamburelis, *Architektura logiczna maszyny cyfrowej ODRA 1305*, MERA-ELWRO, Wrocław 1974.
[5] Th. Kamburelis, *Architektura logiczna EMC JS*, Seria Problemy Informatyki, OBRI, Warszawa 1976.
[6] Projekt Techniczny EMC JS-1045, MERA-ELWRO (Opracowanie wewnętrzne), Wrocław 1974.
[7] S. Tucker, *Emulation of Large Systems*, Communications of the ACM, vol. 8, Dec. 1965.
[8] V. Lesser, *An Introduction to the Direct Emulation of Control Structures by a Parallel Microcomputer*, IEEE Trans. on Computers, vol. C-20, July 1971.
[9] T. Schoen, M. Belsole, *A Burroughs 220 Emulator for the IBM 360/25*, IEEE Trans. on Computers, vol. C-20, July 1971.
[10] H. Dryzek, *Organizacja pamięci w systemach liczących. Problemy podziału*, PWN, Warszawa 1970.

MGR T. KAMBURELIS
OŚRODEK BADAWCZO-ROZWOJOWY CENTRUM MERA-ELWRO
Ostrowskiego 30, 53-238 WROCŁAW, POLSKA

INFORMACJE DLA AUTORÓW

Komitet Redakcyjny w celu skrócenia cyklu wydawniczego prosi autorów o opracowywanie materiałów przeznaczonych do druku w Podstawach Sterowania zgodnie z podanymi poniżej wytycznymi:

1. Prace winny być pisane na maszynie jednostronnie, na pojedynczych arkuszach formatu A4, z podwójną interlinią, z marginesem 3,5 cm, z lewej strony. Stronice numerowane. Prace należy nadsyłać w dwóch egzemplarzach.

2. Wzory i oznaczenia należy wpisać czytelnie, używając jedynie liter łacińskich i greckich. Wskaźniki, niżej liter i wykładniki potęg pisać należy dokładnie i wyraźnie.

3. Prace mogą być nadsyłane w jednym z pięciu języków: polskim, angielskim, francuskim, niemieckim i rosyjskim. Każda praca powinna być zaopatrzona w krótkie streszczenie (do 25 wierszy maszynopisu) oraz w obszerniejsze streszczenia (ok. 20% objętości artykułu) w języku angielskim i rosyjskim, jeśli praca jest w języku polskim, natomiast w polskim i rosyjskim w innych przypadkach.

4. Rysunki i wykresy należy wykonać na oddzielnych arkuszach i co najmniej dwukrotnie większe niż mają być w druku. Na każdym rysunku musi być zaznaczony kolejny numer i nazwisko autora. Oprócz tego należy dołożyć wykaz podpisów pod rysunkami.

5. Wszystkie tablice, podobnie jak rysunki, należy wykonać na oddzielnych arkuszach i numerować je kolejno liczbami arabskimi. U góry tablicy podać tytuł objaśniający.

6. Spis literatury należy umieścić na końcu wymieniając w następującej kolejności: pierwsze litery imion i nazwisko autora, pełny tytuł dzieła lub artykułu, tytuł czasopisma, tom, numer zeszytu, rok, miejsce wydania oraz numery stron. Pozycje powinny być ponumerowane. Odsyłacze do literatury w tekście należy oznaczać kolejnymi cyframi umieszczonymi w nawiasach kwadratowych.

7. Na końcu pracy po nazwisku autora powinna być podana nazwa zakładu pracy i adres.

8. Autorowi przysługuje bezpłatnie 25 egzemplarzy odbitek pracy. Dodatkowe egzemplarze autor może zamówić w redakcji na własny koszt przy przesyłaniu korekty swej pracy lub na koszt zakładu pracy.

Uwaga: Autora obowiązuje korekta autorska, którą należy zwracać w ciągu 3 dni pod adresem: Państwowe Wydawnictwo Naukowe, 31-112 Kraków, ul. Smoleńsk 14, Dział Czasopism.

9. Prace nadsyłać należy bezpośrednio do redaktorów na następujące adresy:

z zakresu: **Podstawowe problemy struktur sterowania**

PROF. JERZY BROMIRSKI

Instytut Cybernetyki Technicznej
Politechnika Wrocławska

ul. Janiszewskiego 11/17, 50-372 Wrocław

z zakresu: **Matematyczne podstawy teorii sterowania**

PROF. CZESŁAW OLECH

Instytut Matematyczny PAN
Warszawa

ul. Śniadeckich 8

z zakresu: **Podstawowe problemy algorytmów sterowania i oprogramowanie jednostek centralnych**

PROF. STEFAN WĘGRZYN

Zakład Systemów Automatyki Kompleksowej PAN
ul. Zwycięstwa 2, 44-100 Gliwice

PODSTAWY STEROWANIA
(kwartalnik)

Warunki prenumeraty czasopisma

Cena prenumeraty krajowej

rocznie zł 60.—
półrocznie zł 30.—

Prenumeratę na kraj przyjmują Oddziały RSW „Prasa—Książka—Ruch" oraz urzędy pocztowe i doręczyciele w terminach:

— **do dnia 25 listopada** na styczeń, I kwartał, I półrocze roku następnego i cały rok następny,

— **do dnia 10 miesiąca** poprzedzającego okres prenumeraty.

Jednostki gospodarki uspołecznionej, instytucje i organizacje społeczno-polityczne składają zamówienia w miejscowych Oddziałach RSW „Prasa—Książka—Ruch".

Zakłady pracy w miejscowościach, w których nie ma Oddziałów RSW oraz prenumeratorzy indywidualni, zamawiają prenumeratę w urzędach pocztowych lub u doręczycieli.

Prenumeratę ze zleceniem wysyłki za granicę, która jest o 50% droższa od prenumeraty krajowej, przyjmuje RSW „Prasa—Książka—Ruch", Centrala Kolportażu Prasy i Wydawnictw, ul. Towarowa 28, 00-958 Warszawa, konto PKO 1531-71 w terminach podanych dla prenumeraty krajowej.

Bieżące i archiwalne numery można nabyć lub zamówić we Wzorcowni Wydawnictw Naukowych PAN — Ossolineum — PWN, Pałac Kultury i Nauki (wysoki parter) 00-901 Warszawa oraz w księgarniach naukowych „Domu Książki".

Tylko prenumerata zapewnia regularne otrzymywanie czasopisma

Information for subscribers

Published quarterly, one volume a year. A subscription order stating the period of time, subscriber's name and address can be sent to any subscription agent or directly to the Foreign Trade Enterprise ARS POLONA — RUCH, 00-068 Warszawa, Krakowskie Przedmieście 7, P. O. Box 1001, Poland.

Please sent payments (annual subscription US $ 8.0) to the account of ARS POLONA — RUCH, Bank Handlowy S. A., Traugutta 7, 00-067 Warszawa, Poland.

Indeks 36881