ZESZYTY NAUKOWE POLITECHNIKI ŚLĄSKIEJ 1980

Seria: INFORMATYKA z. 1 Nr kol. 653

Thanasis KAMBURELIS
Instytut Komputerowych Systemów Automatyki i Pomiarów
Wrocław

# PROBLEMY OPTYMALIZACJI WIRTUALNEJ EMULACJI KOMPUTEROWEJ

Streszczenie. W artykule zaprezentowano krótko nowe podejście w zakresie organizacji środków technicznych przeznaczonych do celów emulacji komputerowej. Przedstawiona jest pewna nowa metoda, zwana Indeksową Metodą Emulacji Wirtualnej. Rozważana metoda zakłada, że zostanie zaprojektowany specjalny, mikroprogramowo sterowany autonomiczny Procesor Emulacji Wirtualnej (PE). Procesor PE ma możliwość współbieżnego wykonywania programów emulowanych (typu ODRA 1300) w czasie wykonywania programów w centralnym procesorze gościnnym (typu RIAD).

W artykule podano niektóre wyniki związane z określeniem optymalnego rozmiaru bloku informacji transmitowanych pomiędzy pamięciami roboczymi oraz informacje o symulacji programowej prezentowanej metody emulacji komputerowej.

Opisana w artykule metoda i algorytmy emulacji zostaną wdrożone w polskich maszynach typu RIAD w celu zapewnienia efektywnego przeniesienia programów polskiej rodziny komputerów ODRA 1300 na komputery systemu RIAD.

## 1. Wprowadzenie

Problemy konwersji programów są ważne zarówno dla producentów, jak i dla użytkowników maszyn cyfrowych, w momencie przejścia z jednego systemu na drugi. W przeszłości stosowano różne techniki konwersji programów, napisanych dla maszyny oryginalnej (starej), umożliwiające wykonanie tych programów przez maszynę gościnną (nową). Zastosowane techniki polegały zarówno na *przeprogramowaniu* lub *translacji* kodów, jak i na *symulacji* lub *emulacji* programów [1, 2, 3, 4].

Każda z wymienionych technik ma swoje zalety i wady (np. czas przeprogramowania danego problemu, czas wykonania programu, współczynnik kosztu itp.).

Emulacja, jak dobrze wiadomo, jest techniką interpretacyjną stosującą zarówno środki sprzętowe, jak i programowe. Technika emulacji jest, ogólnie mówiąc, przynajmniej o rząd szybsza od czystej symulacyjnej techniki i zapewnia efektywność przynajmniej w stosunku jeden do jednego w maszy-

nie gościnnej w odniesieniu do wydajności w maszynie oryginalnej. Jest na ogół rzeczą łatwą uzyskać taką efektywność, gdy maszyna gościnna jest od 5 do 10 razy szybsza od maszyny oryginalnej. W przypadku natomiast, kiedy maszyna gościnna nie jest znacznie szybsza od maszyny oryginalnej, jak to ma miejsce w przypadku maszyny ODRA 1300 i R-32 [7, 8], to niezbędne staje się wprowadzenie sprzętowej interpretacji (przeważnie mikroprogramowej) dla wszystkich rozkazów normalnych często spotykanych w programach emulowanych.

Konwencjonalne koncepcje emulacji, szeroko stosowane w przenoszeniu oprogramowania z maszyn drugiej (lub trzeciej) generacji na maszyny trzeciej generacji, polegają przeważnie na dodaniu dodatkowych mikroprogramów sterujących (i układów logicznych) do sterowania maszyny gościnnej w celu uzyskania efektywnej interpretacji rozkazów maszyny emulowanej.

Jedną z wad tych koncepcji jest to, że programy emulowane i programy własne maszyny emulującej nie mogą być wykonywane jednocześnie. Oznacza to w praktyce, że w czasie wykonywania programów maszyny emulowanej nie wykorzystuje się sporej części środków technicznych i nowych własności rozbudowanej przeważnie nowej maszyny gościnnej.

Drugą wadą jest niewątpliwie fakt, że takie podejście jest realizowalne tylko w nowo konstruowanych maszynach gościnnych. Ponadto producenci w praktyce emulują tylko swoje stare maszyny w nowo budowanych maszynach.

Powstaje zatem pytanie: jakie algorytmy i jaką organizację środków technicznych emulacji przyjąć w celu:

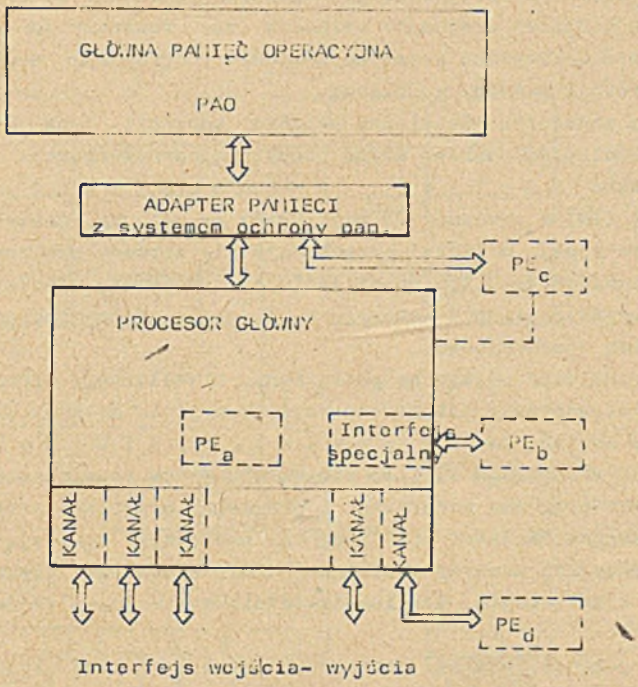
- uniknięcia potrzeby dokonywania zmian technicznych w środowisku maszyny gościnnej,
- Uzyskania maksymalnej łącznej wydajności wykonywania zarówno programów maszyny emulującej, jak i programów maszyny emulowanej,
- zapewnienia możliwości dołączania technicznych środków emulacji do większej klasy maszyn gościnnych (opracowanych na przykład przez różnych producentów).

Zaprezentowana w artykule metoda emulacji spełnia wyżej podane wymagania.

## 2. Koncepcja emulacji wirtualnej

W metodach emulacji zakłada się, że maszyna emulująca (gościnna) zostanie wyposażona w dodatkowe środki techniczne biorące bezpośredni udział w realizacji programów emulowanych. Środki te mogą być zorganizowane i zlokalizowane względem maszyny emulującej w różny sposób. Rysunek 1 pokazuje cztery takie lokalizacje (lub podejścia).

Rozwiązanie, które polega na dodaniu dodatkowych mikroprogramów do pamięci sterującej procesora gościnnego, zostało nazwane E m u l a c j ą K o n w e n c j o n a l n ą ($PE_a$). W rozważaniu tym korzysta się zarów-




Rys. 1. Rozważanie lokalizacyjne procesora emulacji względem procesora głównego komputera emulującego

$PE_a$ - emulacja konwencjonalna, $PE_b$ - emulacja częściowo-autonomiczna, $PE_c$ - emulacja autonomiczna, $PE_d$ - emulacja wirtualna

no z repertuaru mikrooperacji, jak i z arytmometru procesora gościnnego. W tym rozwiązaniu tylko operacje wejścia-wyjścia przebiegają jednocześnie z wykonywaniem programu emulowanego.

Drugim rozważanym podejściem jest zorganizowanie dodatkowych środków technicznych emulacji w formie niezależnego procesora do wykonywania programów emulowanych. Procesor ten jest podłączony do procesora gościnnego poprzez specjalnie określony interfejs (np. jednosłowowy).

Podejście to zostało nazwane C z ę ś c i o w o – A u t o n o m i c z n ą E m u l a c j ą ($PE_b$). Operacje procesora PE są inicjowane w tym podejściu tylko jedną instrukcją, która pojawia się w procesorze gościnnym. Natomiast przesyłanie rozkazów i argumentów, na poziomie pamięć operacyjna - procesor PE, odbywa się za pośrednictwem procesora gościnnego, w którym przebiegają niezależne operacje wejścia-wyjścia oraz procedury przerwań programowych. Przerwanie pracy procesora PE może nastąpić z inicjatywy programu emulowanego lub procesora gościnnego. Po przerwaniu pra-

cy programu emulowanego stany robocze akumulatorów procesora PE przenosi się automatycznie do tzw. rejestrów ogólnych procesora gościnnego. Dzięki temu funkcje bardziej złożonych rozkazów (np. rozkazów zmiennoprzecinkowych) mogą być wykonywane przez odpowiednie podprogramy napisane na poziomie instrukcji maszyny gościnnej.

W trzecim podejściu niezależny procesor emulacji jest podłączany do procesora gościnnego poprzez szynę pracy wieloprocesorowej. Podejście to zostało nazwane E m u l a c j ą  A u t o n o m i c z n ą ($PE_c$). Procesor główny (PG) i procesor PE mają tutaj niezależny dostęp do wspólnej pamięci operacyjnej, w której przechowuje się zarówno programy własne, jak i programy emulowane. W tym rozwiązaniu oba procesory funkcjonują jednocześnie i znajdują się pod kontrolą wspólnego niejednorodnego dwuprocesorowego systemu operacyjnego.

Czwarte podejście polega na podłączaniu niezależnego procesora emulacji poprzez standardowy interefejs wejścia-wyjścia maszyny emulującej (PG). Podejście to zostało nazwane E m u l a c j ą  W i r t u a l n ą ($PE_d$).

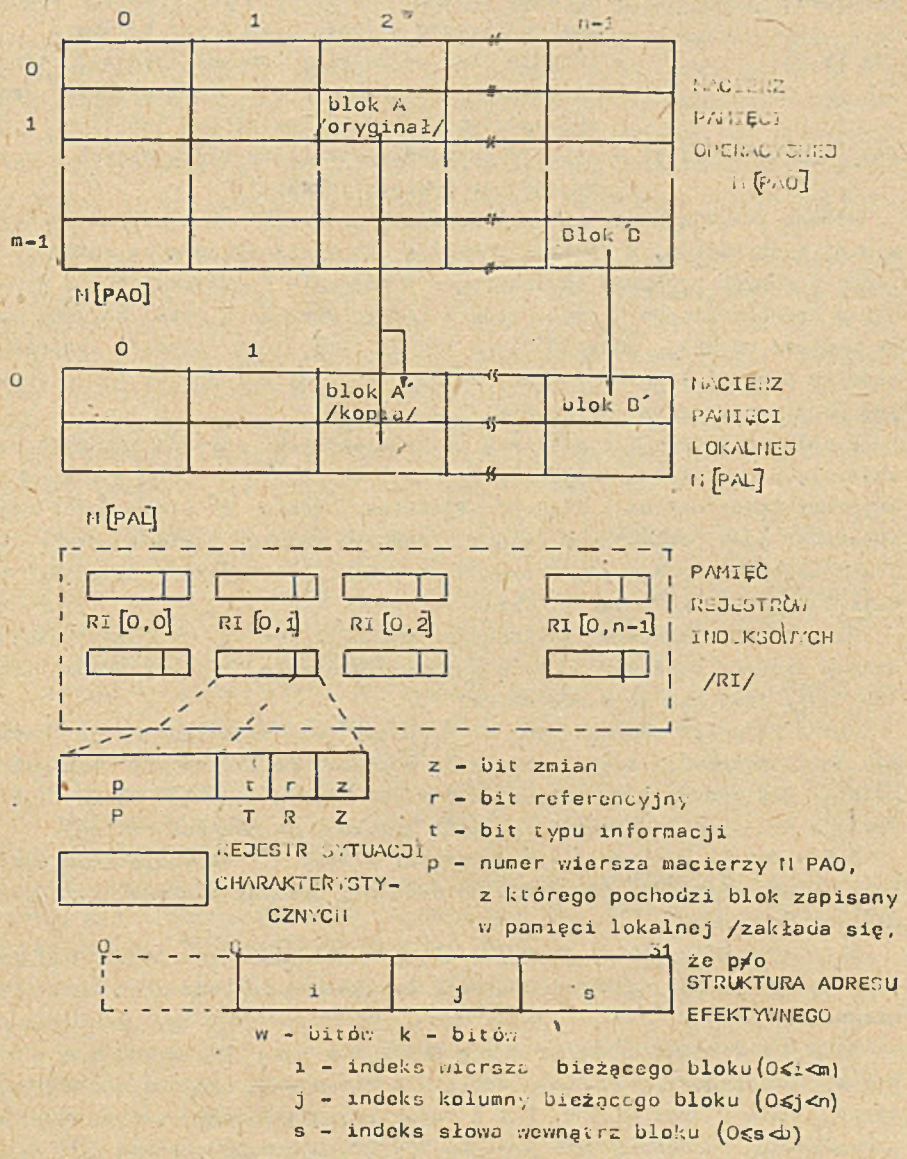
Współpraca procesorów PG i PE odbywa się w tym rozwiązaniu pod kontrolą specjalnego programu kanałowego i programu organizacyjnego o nazwie Emulator. Program Emulator oraz programy emulowane rezydują w całości w pamięci operacyjnej maszyny emulującej (PG), natomiast procesor PE wyposażony jest w małą pamięć lokalną (notatnikową), np. o pojemności 1 lub 2 K słów.

Zadaniem pamięci lokalnej jest przechowywanie w niej tylko aktywnych otoczeń rozkazów i danych programu emulowanego. Zakłada się, że transmisja bloków informacji pomiędzy pamięcią operacyjną (PAO) a pamięcią lokalną (PAL) procesora PE będzie się odbywać pod kontrolą specjalnych algorytmów minimalizujących liczbę transmisji bloków oraz łączny czas wykonywania programu emulowanego.

Szereze omówienie zalet i wad tych czterech wyżej wymienionych podejść w organizacji środków emulacji znajdzie czytelnik w pracach [6, 9, 10].

## 3. Indeksowa metoda emulacji wirtualnej

W metodzie indeksowej zakłada się, że cała pamięć realna (operacyjna) zostaje podzielona na pewne obszary. Zawartość takiego obszaru nazywa się blokiem danych lub po prostu blokiem, zaś adres takiego obszaru nazywa się adresem realnym bloku. Bloki danych traktuje się również jako elementy matrycy mającej m wierszy i n kolumn (rys. 2). Zakłada się przy tym, że n jest liczbą stałą, zaś m jest wielkością zmienną (zależną od wielkości zainstalowanej pamięci realnej).




Rys. 2. Struktura pamięci w emulacji wirtualnej opartej o metodę indeksową

Pamięć lokalna procesora PE jest również traktowana jako macierz mająca n kolumn i dwa wiersze. W dalszej części artykułu będziemy stosowali następujące notacje:

PAO [i,j] - określa element matrycy pamięci realnej, który należy do i-tego wiersza oraz j-tej kolumny,

PAL [k,j] - określa element matrycy pamięci lokalnej, który należy do j-tej kolumny oraz k-tego wiersza (rozważać będziemy tylko przypadki k=0 lub 1),

PAL [k,j]:=PAO [i,j] - opisuje operację przesłania bloku danych z pamięci realnej do pamięci lokalnej.

Główną funkcją małej pamięci lokalnej procesora PE jest przechowywanie w niej tych bloków, z których pochodzą aktualnie wykonywane sekwencje rozkazów i danych programu emulowanego. W metodzie indeksowej bloki przesyła się w sposób "pionowy", tzn. blok z i-tego wiersza i j-tej kolumny matrycy pamięci realnej, gdzie $0 \leq i < m$ i $0 \leq j < n$, jest zawsze ładowany do obszaru pamięci lokalnej, który należy do j-tej kolumny i do pierwszego lub drugiego wiersza (patrz rys. 2).

W celu rejestracji indeksów wierszy blokowych adresów realnych wbudowuje się w procesorze emulacji 2n r e j e s t r ó w i n d e k s o w y c h. Rejestry indeksowe mają długość w+3-bitów, gdzie $2^w = m$. Każdy rejestr indeksowy jest powiązany z jednym elementem matrycy pamięci lokalnej. Rejestr indeksowy powiązany z elementem PAL[k,j] oznaczać będziemy przez RI[k,j], gdzie k=0 lub 1 oraz $0 \leq j < n$.

Bity pola P rejestru indeksowego, tj. RI[k,j]<p>, są porównywane z indeksem wiersza bieżącego adresu realnego w czasie, gdy nowe słowo (rozkaz lub dana) jest żądane w procesorze PE.

Bit Z rejestru indeksowego, RI[k,j]<z>, jest ustawiany w stan 1 zawsze, gdy nowa informacja zostaje zapisana w odpowiednim obszarze pamięci lokalnej. Bit ten jest wykorzystywany w operacjach usuwania bloków lub operacjach przesyłania bloków z pamięci lokalnej do pamięci realnej.

Bit R lub RI[k,j]<r>, jest ustawiany w stan 1 we wszystkich przypadkach, kiedy występuje odwołanie do dowolnej komórki odpowiedniego obszaru pamięci lokalnej.

Bit T lub RI[k,j]<t>, jest ustawiany w stan zero lub jeden, gdy do odpowiedniego obszaru pamięci lokalnej ładuje się rozkazy lub dane (odpowiednio).

Bity R i T stosuje się w a l g o r y t m i e w y m i a n y b l o k ó w, gdy żądany jest nowy blok i nie ma wolnego miejsca w pamięci lokalnej. Pole P oraz bity T, R i Z rejestru indeksowego tworzą czwórkę uporządkowaną oznaczoną symbolem <P,T,R,C>. Operacja postaci RI [k,j]:= =<i,0,0,0> oznacza, że zachodzi ładowanie indeksu i-tego wiersza w polu P rejestru RI[k,j] oraz zerowanie bitów T,R,Z tego rejestru.

3.1. Algorytm wymiany bloków

Inicjowanie wykonywania programów emulowanych w procesorze PE odbywa się za pomocą specjalnej komendy kanałowej (PISZ AKUMULATORY), która pojawia się w procesorze gościnnym. Powyższa komenda zapoczątkowuje również łańcuchową sekwencję komend w kanale procesora PG. Oznacza to, że do programu Emulator włączono pewną pętlę lub łańcuch słów sterujących kanału, który steruje przebiegiem wprowadzania-wyprowadzania sekwencji bloków danych o dowolnej długości. Wykonywanie programu emulowanego w procesorze PE jest kontynuowane aż do momentu pojawienia się rozkazu typu WEZWANIE SUPERWIZORA, który powoduje przesłanie sygnału przerwania do procesora gościnnego.

W czasie wykonywania programu emulowanego do odpowiednich obszarów pamięci lokalnej ładuje się żądane bloki danych oraz aktualizuje się stany rejestrów indeksowych; tzn. PAL[k,j]:= PAO[i,j] oraz RI[k,j]:=<i,t,0,0>, gdzie k=0 lub 1. Powyższy proces jest kontynuowany tak długo, dopóki w pamięci lokalnej istnieją wolne obszary. Jeśli jednak oba obszary, PAL[0,j] i PAL[1,j], pamięci lokalnej danej kolumny j są zajęte, to wówczas wymagana jest operacja wymiany bloków. W celach minimalizacji ilości transmisji bloków przyjęto następującą strategię wymiany bloków:

1. Jeśli oba obszary danej kolumny i pamięci lokalnej są wolne, tzn. RI[0,j]=<0,0,0,0> i RI[1,j]=<0,0,0,0>, to wówczas przyjmuje się k:=0; tzn. PAL[0,j]:= PAO[i,j] i RI[0,j]:=<i,t,0,0>, gdzie i oraz j są odpowiednio indeksem wiersza i kolumny bieżącego adresu realnego, zaś symbol t oznacza typ informacji (t=0 dla rozkazów i t=1 dla argumentów).

2. Jeśli x-ty wiersz (x=0 lub 1) danej kolumny j pamięci lokalnej jest wolny, zaś $\bar{x}$-ty wiersz tej samej kolumny jest zajęty, to k:=x; tzn. PAL[x,j]:= PAO[i,j] i RI[x,j]:=<i,t,0,0>.

3. Jeśli oba obszary danej kolumny pamięci lokalnej są zajęte oraz zawierają ten sam typ informacji, to wymienia się zawartość tego obszaru, którego bit R ma mniejszą wartość.

4. Jeśli oba obszary danej kolumny j pamięci lokalnej są zajęte oraz zawierają różne typy informacji, to wymienia się zawartość obszaru mającego ten sam typ informacji co blok pobierany.

W ostatnich dwóch przypadkach bit Z rejestru indeksowego związanego z obszarem ładowanym jest testowany. Jeśli RI[k,j]<z>=1, to wykonuje się operację pamiętania PAO[p,j]:= PAL[k,j] przed wykonaniem operacji pobrania PAL[k,j]:= PAO[i,j]; gdzie p oraz i są odpowiednio starym i nowym indeksem wiersza adresu realnego.

Po zakończeniu operacji wymiany bloków oraz zrealizowaniu żądanego odwołania zeruje się oba bity referencyjne danej kolumny, tj. RI[0,j]<r>:= 0 i RI[1,j]<r>:= 0, po czym kontynuuje się wykonywanie programu emulowanego.

Opisany algorytm wymiany bloków jest realizowany sprzętowo w procesorze PE.

3.2. Operacje inicjacji, przesyłania i zakończenia

Wykonywanie programu emulowanego w procesorze PE jest inicjowane za pomocą komendy kanałowej PISZ AKUMULATORY. Komenda ta powoduje przesłanie zawartości akumulatorów i innych rejestrów sterujących (np. licznika rozkazów, wskaźników programowych itp.) maszyny emulowanej z pamięci realnej do procesora PE.

Tabela 1

| KOMENDY KANAŁOWE |
|---|
| Nazwa i funkcja komendy |
| 1. ZERUJ REJESTRY INDEKSOWE /ZRI/:<br>$\forall$(RI [0,j] :=0 $\wedge$ RI [1,j] :=0)<br>$0 \leqslant j < n$ |
| 2. CZYTAJ REJESTRY INDEKSOWE /CRI/ DO TABLICY ADRESOW REALNYCH /TAR/:<br>TAR [4j] <0> :=RI [0,j] <z>;<br>TAR [4j+1:4j+3] <i> :=RI [0,j] <p>;<br>TAR [4j+4n] <0> :=RI [1,j] <z>;<br>TAR [4j+4n+1:4j+4n+3] <i> :=RI [1,j] <p>;<br>Wejście tablicy TAR są 4-bajtowe |
| 3. PISZ DANE /PID/:<br>PAL [k,j] :=PAO [i,j]; |
| 4. Czytaj DANE /CZD/:<br>PAO [p,j] :=PAL [k,j]; |
| 5. PISZ AKUMULATORY /PIA/ Z POLA AKUMULATOROW EMULOWANYCH /PAE/:<br>AKUM [0:a-1] :=PAE [0:a-1];<br>gdzie a - liczba akumulatorów |
| 6. CZYTAJ AKUMULATORY /CZA/:<br>PAE [0:a-1] :=AKUM [0:a-1]; |
| 7. CZYTAJ ADRES KOPII /CAK/ DO POLA PROGRAMU KANAŁOWEGO /PPK/:<br>PPK [s+25:s+27] :=RI [k,j] <p>+N <j>;<br>gdzie N - bieżący adres realny |
| 8. CZYTAJ ADRES ORYGINAŁU /CAO/:<br>PPK [s+41:s+43] :=N <i>+N <j>; |
| 9. CZYTAJ SYTUACJĘ CHARAKTERYSTYCZNĄ /CZS/:<br>PPK [s-5] :=KOD SYTUACJI (jeśli nowa sytuacja) |
| 10. CZYTAJ KOPIE /CZK/ |
| 11. PRZEŁĄCZ W KANAŁ /PŁK/ |
| 12. ZWOLNIJ KANAŁ /ZWK/ |

Sterowanie przebiegiem wykonywania programów emulowanych jest oparte na technice programu kanałowego. Program kanałowy, który znajduje się w programie Emulator, jest pewnym łańcuchem słów sterujących pracą kanału i składa się z dwunastu różnych komend zdefiniowanych dla potrzeb emulacji

wirtualnej. Nazwy i funkcje tych komend podaje tabela 1. Struktura programu kanałowego jest tak pomyślana, aby możliwe było przesyłanie dowolnej ilości bloków programu emulowanego. Adresy realne nowych żądanych bloków wyznacza się w procesorze PE, po czym przesyła się je do odpowiednich pól programu kanałowego. Jeśli bit Z rejestru indeksowego, związanego z obszarem podlegającym ładowaniu, znajduje się w stanie jeden, to przesyła się do programu kanałowego także adres realny starego bloku przechowywanego w tym obszarze. Program kanałowy wybiera odpowiednie drogi dalszej pracy, stosownie do zaistniałej sytuacji w procesorze PE. W procesorze tym mogą, ogólnie mówiąc, pojawić się trzy wyróżnione sytuacje:

1. W programie emulowanym pojawiła się instrukcja typu Wezwanie Supervizora (kod sytuacji 0).

2. W procesorze PE żądany jest nowy blok, a stary musi być przechowany (kod 16).

3. W procesorze PE żądany jest nowy blok oraz istnieje wolny obszar dla przyjęcia nowego bloku lub bit Z odpowiedniego rejestru indeksowego jest równy zeru (kod 32).

Struktura programu kanałowego pokazana jest w tabeli 2.

Tabela 2

STRUKTURA PROGRAMU KANAŁOWEGO

```
s-32: [ZRI, 0, 0, 1];
s-24: [PIA, PAE, 4a, 1];
s-16: [CZS, s-5, 1, 1];
s-8 : [PŁK, s+, 0, 0];
s+0 : [CZA, PAE, 4a, 1];   s+0
s+8 : [CRI, TAR, 8n, 0];
s+16: [CAK, s+25, 3, 1];   s+16
s+24: [CZD, 0+, b, 1];
s+32: [CAO, s+41, 3, 1];   s+32
s+40: [PID, 0+, b, 1];
s+48: [ZWK, 0, 0, 1];
s+56: [PŁK, s-16, 0, 0];
```

Gdzie czwórka [K, D, L, Ł] oznacza 8-bajtowe Słowo Sterujące Kanału [6].

Oznaczenia:

K: Kod Komendy Kanałowej
D: Adres Danych
L: Licznik Bajtów
Ł: Łańcuch Komend (1 lub 0)
s: Dowolny adres realny podzielny przez 256
b: Rozmiar bloku w bajtach
n: Ilość kolumn pamięci PAL

## 4. Określenie optymalnego rozmiaru bloku

Czas wykonania programu emulowanego w procesorze PE zależy zarówno od struktury wewnętrznej programu, jak i od przyjętego rozmiaru bloku informacji transmitowanych pomiędzy pamięcią operacyjną maszyny emulującej a pamięcią lokalną procesora PE. Przyjęcie zbyt małego rozmiaru bloku pociąga za sobą konieczność zorganizowania wielu transmisji blokowych. Zorganizowanie i inicjacja danej transmisji w kanale wymagają zawsze określonego czasu, który jest na ogół niezależny od rozmiaru bloku transmitowanego. Przyjęcie natomiast zbyt dużego rozmiaru bloku może często prowadzić do pobierania do pamięci lokalnej niepotrzebnych fragmentów programów (lub danych) i tym samym do częstych przeładowywań małej pamięci lokalnej.

W celu znalezienia optymalnego rozmiaru bloku dla przyjętej metody wirtualnej emulacji oraz dla przyjętej struktury maszyny emulującej i maszyny emulowanej, przeprowadzono pewne rozważania teoretyczne, w wyniku których wyprowadzono prosty wzór wyznaczający rozmiar bloku. Jeśli rozmiar bloku (liczony w słowach) oznaczymy przez s, to zachodzi następująca zależność:

$$T_s = \frac{1}{-2 \ln p} \sqrt{\frac{2T_k + 3T_p}{T_p(1 - v)}} = \frac{1}{-2 \ln p \sqrt{t(1 - v)}}$$

gdzie:

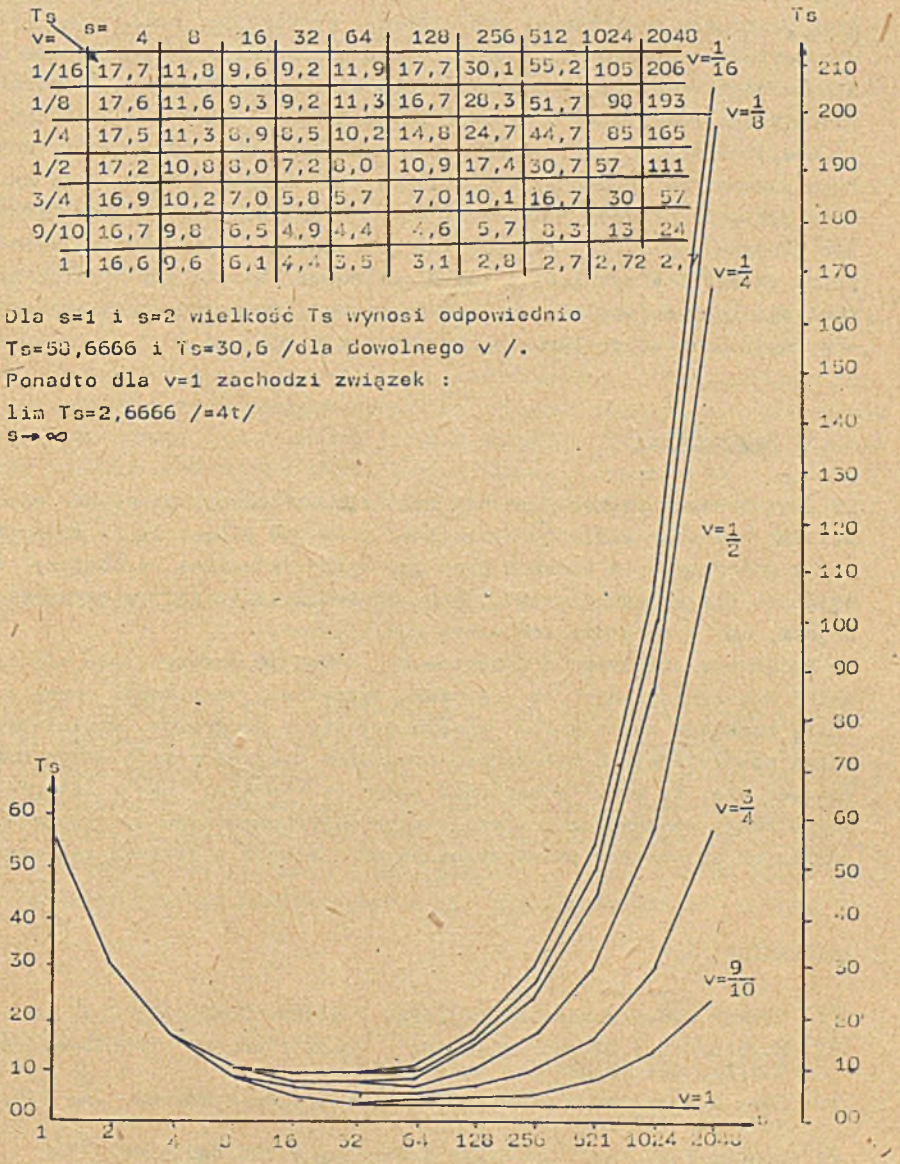
- p – oznacza stopień sekwencyjności programu emulowanego (lub prawdopodobieństwo, z jakim adresowa zmienna losowa przybiera adres A+1, po przyjęciu adresu A),
- v – oznacza stosunek wielkości pamięci lokalnej do pojemności pamięci operacyjnej zajmowanej przez dany program emulowany,
- $T_k$ – czas inicjacji pracy kanału maszyny emulującej (czas ten wynosi 27 mikrosekund dla maszyny R-32),
- $T_p$ – czas przesłania jednego bajtu informacji przez interfejs wejścia-wyjścia (równy 0,666 mikrosekundy dla maszyny R-32).

Jeśli zatem znamy parametry konstrukcji maszyny emulującej (tj. czas inicjacji pracy kanału i szybkość interfejsu wejścia-wyjścia) oraz parametry p i v programu emulowanego, to można w prosty sposób oszacować optymalny rozmiar bloku. Rysunek 3, sporządzony dla średniej wartości parametru p=0,82 (oszacowanej na podstawie kilku mieszanek statystycznych instrukcji), pokazuje, że funkcja $T_s$ średniego czasu transmisji, w przeliczeniu na jedno przetworzone słowo programu emulowanego, osiąga minimum dla rozmiaru bloku s=32 i dla v od 1/16 do 1/2.

Rysunek 3 pokazuje również, że jeśli parametr v wzrasta, to wzrasta również optymalny rozmiar bloku i odwrotnie, rozmiar ten maleje wraz ze zmniejszeniem pojemności pamięci lokalnej.

| Ts v= \ s= | 4 | 8 | 16 | 32 | 64 | 128 | 256 | 512 | 1024 | 2048 |
|---|---|---|---|---|---|---|---|---|---|---|
| 1/16 | 17,7 | 11,8 | 9,6 | 9,2 | 11,9 | 17,7 | 30,1 | 55,2 | 105 | 206 |
| 1/8 | 17,6 | 11,6 | 9,3 | 9,2 | 11,3 | 16,7 | 28,3 | 51,7 | 98 | 193 |
| 1/4 | 17,5 | 11,3 | 8,9 | 8,5 | 10,2 | 14,8 | 24,7 | 44,7 | 85 | 165 |
| 1/2 | 17,2 | 10,8 | 8,0 | 7,2 | 8,0 | 10,9 | 17,4 | 30,7 | 57 | 111 |
| 3/4 | 16,9 | 10,2 | 7,0 | 5,8 | 5,7 | 7,0 | 10,1 | 16,7 | 30 | 57 |
| 9/10 | 16,7 | 9,8 | 6,5 | 4,9 | 4,4 | 4,6 | 5,7 | 8,3 | 13 | 24 |
| 1 | 16,6 | 9,6 | 6,1 | 4,4 | 3,5 | 3,1 | 2,8 | 2,7 | 2,72 | 2,7 |

Dla s=1 i s=2 wielkość Ts wynosi odpowiednio
Ts=58,6666 i Ts=30,6 /dla dowolnego v /.
Ponadto dla v=1 zachodzi związek :
$\lim_{s\to\infty} Ts = 2{,}6666$ /=4t/




Rys. 3. Wykresy funkcji $T_s$ dla p=0,82 i dla v = $\frac{1}{16}$, $\frac{1}{8}$, $\frac{1}{4}$, $\frac{1}{2}$, $\frac{3}{4}$, $\frac{9}{10}$, 1

Proponowana metoda emulacji wirtualnej została również przebadana na drodze symulacji programowej. W tym celu opracowano szczegółowe procedury symulacji wszystkich algorytmów metody i wykonano specjalny program symulacji działający na maszynie ODRA 1305. Po czym wykonano ponad 30 testowych programów emulowanych pod kontrolą symulatora, obliczając jednocześnie wartości szeregu parametrów dobrze charakteryzujących badane algorytmy emulacji wirtualnej. Otrzymane średnie wyniki symulacji potwierdziły teoretyczne rozważania dotyczące optymalnego rozmiaru bloku informacji transmitowanych. Wyniki symulacji pozwoliły także na optymalny dobór szeregu parametrów konstrukcji procesora (PE) emulacji wirtualnej (np. pojemność pamięci lokalnej, liczba rejestrów indeksowych, rozmiar bloku itp.). Wyniki symulacji pozwoliły również na dokonanie porównania zalet i wad metod emulacji wirtualnej z innymi metodami, tj. metody czystej symulacji, emulacji konwencjonalnej lub zanurzeniowej, emulacji realizującej sprzętowo tylko proces pobierania i modyfikacji rozkazu.

## 5. Zakończenie

Zastosowanie opisanej metody emulacji wirtualnej ma szereg zalet. Niezależny procesor emulacji wirtualnej może być dołączony do różnych komputerów gościnnych, w których przyjęto tylko jednakowe procedury wejścia-wyjścia. Nie wymaga się tutaj jakichkolwiek zmian ani w systemie operacyjnym, ani w sprzęcie komputera emulującego.

Relatywna efektywność dyskutowanej metody jest porównywalna z efektywnością metod konwencjonalnych, w przypadku sprzętowej realizacji prawie pełnej listy rozkazów normalnych. Ponadto wykonanie programów emulowanych przebiega współbieżnie z wykonaniem programów komputera gościnnego. Można spodziewać się, że nowe technologie zespołów wielkiej skali integracji będą stymulowały zastosowania niezależnych mikrokomputerów do wykonywania niezależnych procesorów przetwarzaniowych w nowych systemach komputerowych.

LITERATURA

[1] Tucker S.G.: Emulation of Large Systems. Comm. ACM 8,12 (Dec. 1965).

[2] Husson S.S.: Microprogramming - Principles and Practices. Englewood Cliffs, NJ: Prentice - Hall, 1970.

[3] Cary R.: System 370 Integration Emulation under OS and DOS. AFIPS, vol. 38, 1971.

[4] Schoen T.A., Belsole M.R.: A Burroughs 220 Emulator for the IBM 360/25. IEEE Trans.Comp. vol. C-20, 7, July 1971.

[5] Salisbury A.B.: The evaluation of Microprogram Implemented Emulators. Stanford Universyty, AD-768883/1, July 1973.

[6] Emulacja EMC ODRA 1300 w EMC JS. OBR Mera-Elwro (raport wewnętrzny), Wrocław, styczeń 1977.

[7] Kamburelis T.: Architektura logiczna m.c. ODRA 1305. Mera-Elwro, Wrocław, 1974.

[8] Kamburelis T.: Architektura logiczna EMC J.S. Problemy Informatyki, OBRI, Warszawa, 1976.

[9] Kamburelis T.: Współbieżna emulacja komputerowa. Podstawy Sterowania, tom 6(1976), z. 4.

[10] Kamburelis T., Zasada A.: Współbieżna emulacja komputerowa - metoda stronicowa. Podstawy Sterowania, tom 7, 1977, z. 1.

ПРОБЛЕМЫ ОПТИМАЛИЗАЦИИ ВИРТУАЛЬНОЙ КОМПЬЮТЕРНОЙ ЭМУЛЯЦИИ

Р е з ю м е

В статье излагается краткий новый подход в области организации технических средств предназначенных для компютерной эмуляции. Приводится новый метод, называемый Индексный метод виртуальной эмуляции. Рассуждаемый метод предполагает, что будет запроектированный специальный,автономный микропрограммно управляемый Процессор виртуальной эмуляции PE .Процессор PE имеет возможность одновременного решения эмулированных программ типа ODRA 1305 во время решения программ в центральном гостеприимном процессоре типа RIAD .

SOME OPTIMALIZATION ASPECTS OF THE COMPUTER VIRTUAL EMULATION

S u m m a r y

The new approach to the organization of the technical means for the computer emulation is briefly stated in the paper. A new method, named the Index Virtual Emulation Method is presented. The method considered assumes the special design construction of the microprogram controlled autonomous Virtual Emulation Processor (PE). The PE Processor provides the availability of the concurrent realization of the programs emulated (ODRA 1300, type) at the time of central host processor programs execution (RIAD type).