1991. – 94 с.

5. Семьянов П.В., Зегжда Д.П. Анализ средств противодействия исследованию программного обеспечения и методы их преодоления // КомпьютерПресс. – 1993. – № 11. – С. 34-36.

6. Середа С.А. Анализ средств преодоления систем защиты программного обеспечения // ИНФОРМОСТ: Радиоэлектроника и Телекоммуникации. – 2002. – № 4 (22). – С. 11-16.

7. Середа С.А. Оценка эффективности систем защиты программного обеспечения // КомпьЛог. – 2000. – № 2. – С. 56-60.

8. Safe and Secure firmware upgrade for AT91SAM microcontrollers // Atmel Corporation, 2006 / Document № 6253A-ATARM-07-sep-06.

9. Комплекс "РС-3000" // http://www.datarec.ru/company/devices.php

10. Аппаратная защита программного обеспечения // http://karman.com.ua/articles/_apparatnaja_zashchita_0_0_0_823_1.html

11. Мікропроцесорна техніка: Підручник / Ю.І. Якименко, Т.О. Терещенко, Є.І. Сокол та ін. – 2-е вид., переробл. та доповн. – К.: ІВЦ "Видавництво "Політехніка"; "Кондор", 2004. – 440 с.

Надійшла 4.2.2009 р.

УДК 004.78: 004.042

В.Я. ЛЯШКЕВИЧ, А.В. СИДОРУК
Чернівецький національний університет імені Юрія Федьковича

ПРОГРАМНА РЕАЛІЗАЦІЯ ВІРТУАЛЬНОГО ПРОЦЕСОРА ДЛЯ НАПИСАННЯ ПРОГРАМ З МОЖЛИВІСТЮ МОДИФІКАЦІЇ ЇХ КОДУ В ПРОЦЕСІ ВИКОНАННЯ

Розглянуто проблеми розширення гнучкості програмного коду за рахунок модифікації його у виконуваному файлі під час виконання. Розроблено мову програмування та віртуальний процесор, що дозволяють проводити зміни в області даних, коду і команд програми.

Вступ. Сучасні інструментальні засоби мов програмування надають можливість повноцінно використовувати можливості апаратного забезпечення комп'ютерних систем користувачам прикладного програмного забезпечення. Така взаємодія користувача із комп'ютерною системою можлива згідно з наперед визначеними і реалізованими алгоритмами у вигляді завантажувальних *.com чи *.exe файлів [1, 2] Це зумовлено тим, що не завжди є можливість вийти за рамки „безумовних" реакцій на вхідні дані, тобто обмеження полягає в тому, що неможливо в процесі виконання змінити виконавчий файл без перекомпіляції.

Постановка задачі. Системи обробки інформації, можуть змінювати свою структуру в процесі функціонування, але для цього необхідно виконати компіляцію, тривалість якої пропорційно залежить від величини досліджуваної системи. Також, ефективність використання інформаційних систем та можливість їх швидкої адаптації до користувача пов'язані з апаратною та програмною сумісністю, яку можна організувати за допомогою віртуального процесора тому, що фізична архітектура комп'ютерної системи накладає певні обмеження на програми й вимагає їх виконання за певних умов. Основне обмеження – програма являє собою кінцевий файл з незмінним кодом та набором команд, з можливістю проводити зміни тільки в області даних [2].

Для усунення зазначених недоліків пропонується забезпечити:

- можливість якісної динамічної зміни коду, з метою формування алгоритмів, які будуть володіти максимальною гнучкістю. На базі отриманих алгоритмів створювати принципово нові алгоритми, що зможуть змінювати свою структуру в процесі виконання. Зміна повинна відбуватись на рівні початкового алгоритму;
- зв'язок з іншими мовами програмування для використання модулів написаних на інших мовах програмування, наприклад, для забезпечення відповідної швидкодії певних модулів, взаємодії з операційною системою чи іншим прикладним програмним забезпеченням.

Загальні підходи вирішення проблеми гнучкості коду. На сьогодні, вирішенням даної задачі – розроблений ряд віртуальних процесорів, а саме: віртуальна машина Java; Microsoft JScript; Microsoft Visual Basic Scripting; мова сценаріїв ASP; мова VCC та ін. мови (REXX, PERL, Python, LangMF). В їх основі використовуються наведені нижче підходи.

1. Зовнішнє програмування (рис. 1). У випадку невідомих вхідних даних програма робить запит користувачу для подальшого виконання й запису відповідної реакції.

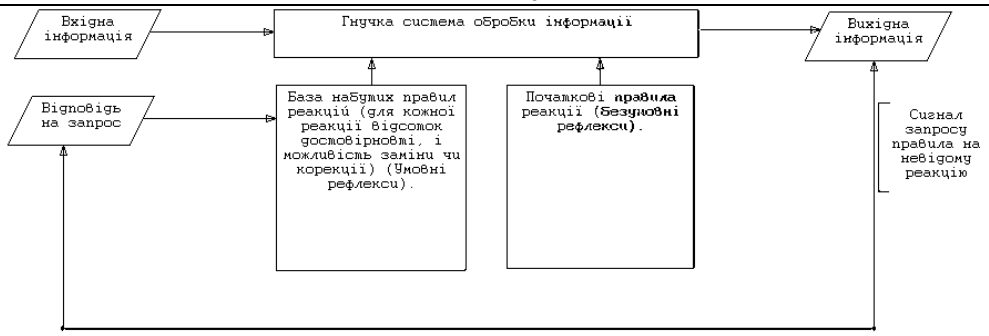

Рис. 1. Принципи зовнішнього програмування

2. Самопрограмування (рис. 2). При невідомих вхідних даних програма за конкретним алгоритмом, який використовує історію попередніх генерацій, видає результат, очікуючи певної реакції. Прийнявши наступним тактом реакцію, порівнює її з прогнозованою й записує результат у вигляді нової реакції, вже більш достовірної. Таким чином, за декілька проходів програма має достовірну реакцію на події, які записані в базі.

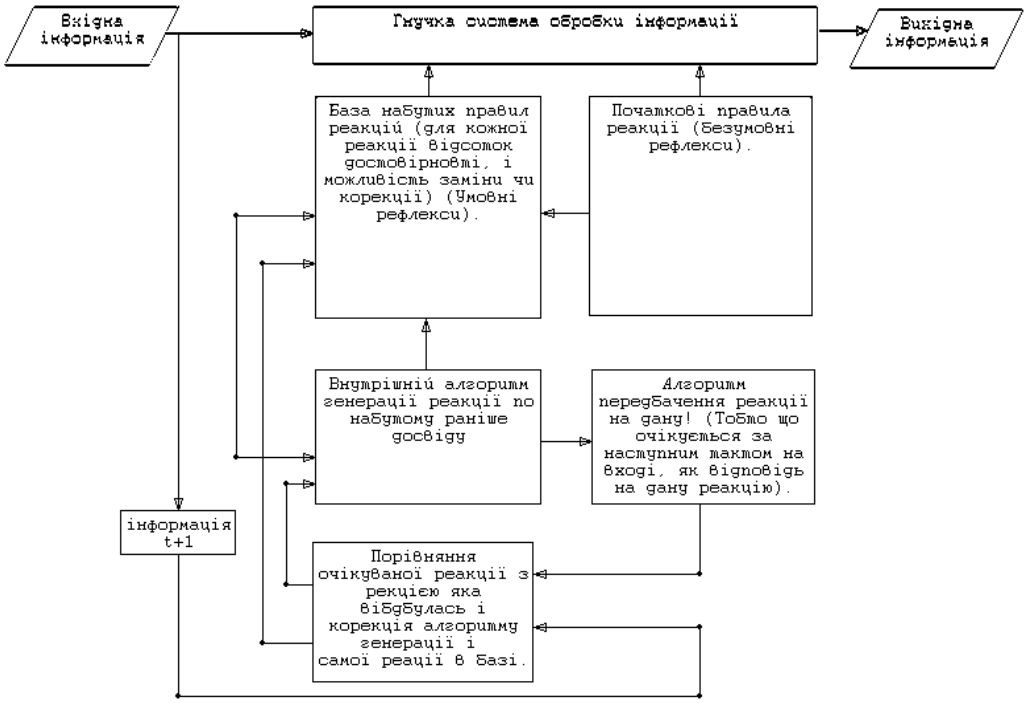

Рис. 2. Принципи самопрограмування

3. Комбінований метод (рис. 3). Даний метод базується на застосуванні принципів зовнішнього програмування та самопрограмування.

Програмна структура віртуального процесора. Основні частини віртуального процесора (рис. 4) – матриця та ядро. Весь код збережено в чотиривимірній матриці. Вона зберігає, як код, так і систему команд, які його обробляють. Власне, вся обробка відбувається в даній матриці. Код в матрицю завантажується частинами, і зберігається також частинами, що дає можливість не утримувати постійно весь код великої програми в динамічній пам'яті, а користуватись лише необхідними частинами коду в певний час роботи. Для цього створено багатосекторний код, який дозволяє розбивати загальну структуру на частини, кожній з яких виділяється окремий потік. Код потоку зберігається в третьому вимірі матриці. Ядро, при необхідності, завантажує і зберігає сектори коду. Команда (точка входу) стартового сектору активує процес виконання команд.

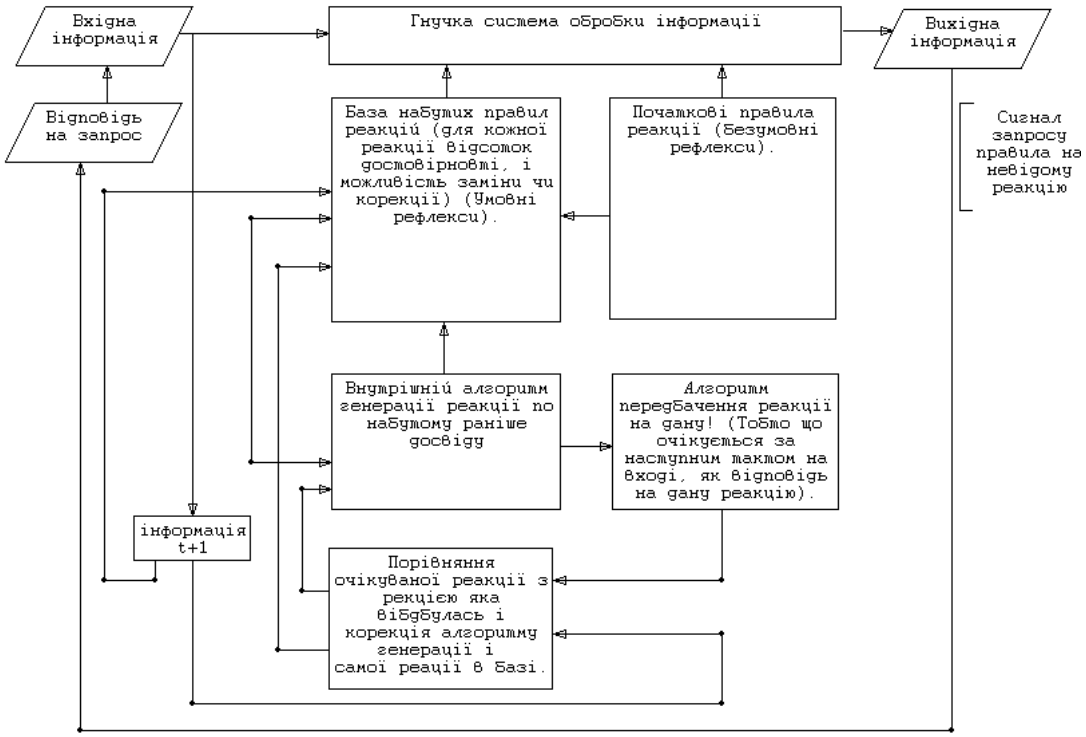

Рис. 3. Принципи комбінованого методу

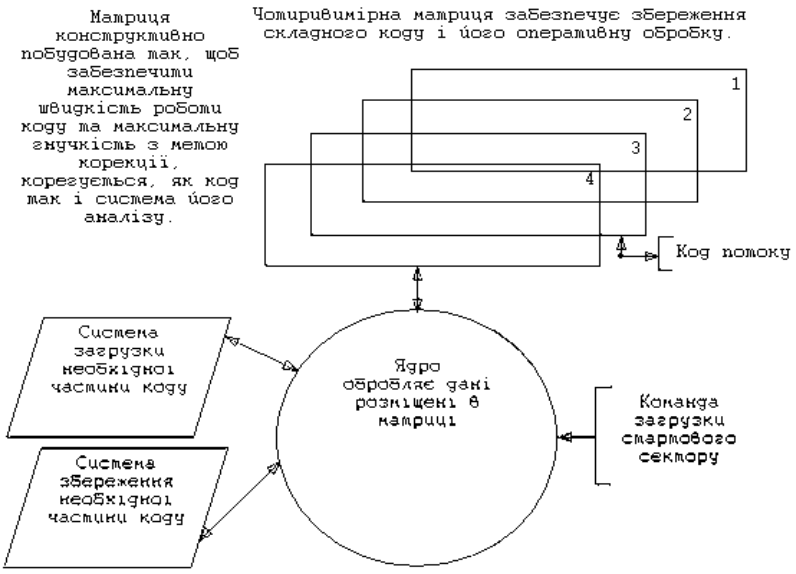

Рис. 4. Програмна структура віртуального процесора BERS

Ядро (рис. 5) використовується всіма потоками процесора, також при завантаженні, збереженні секторів (для синхронізації між потоками). Ядро – основна сила, яка створює рух по коду і призводить до конструктивних змін у матриці, яка використовує для кожного потоку як його код, так і систему команд, прив'язану до даного коду. Тобто ядро тісно взаємодіє з матрицею, проводячи в ній реконструкцію й відповідно дублює зміни в оригінальному файлі коду.

Матриця складається з 4 площин і являє собою складну структуру зі складними внутрішніми зв'язками:

- перша площина. Рядок – елементарні значущі одиниці;
- друга площина. Сторінка – послідовність рядків;
- третя площина. Сектор – набір двомірних сторінок із зв'язками між секторами та кодом потоку;
- четверта площина. Об'єднання внутрішніх наборів команд, кожній з яких відповідає набір секторів із міжмовними зв'язками.

Для написання програмного коду, що виконується на розробленому віртуальному процесорі, розроблено оболонку, вигляд якого зображено на рис. 6. Деякий фрагмент програмного коду наведено на рис. 7.

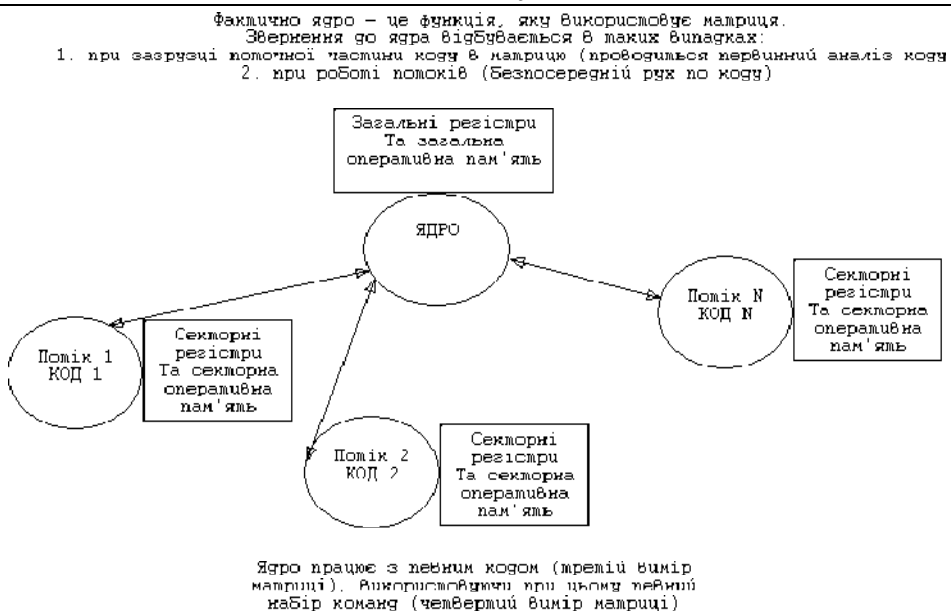

Рис. 5. Структурна організація ядра

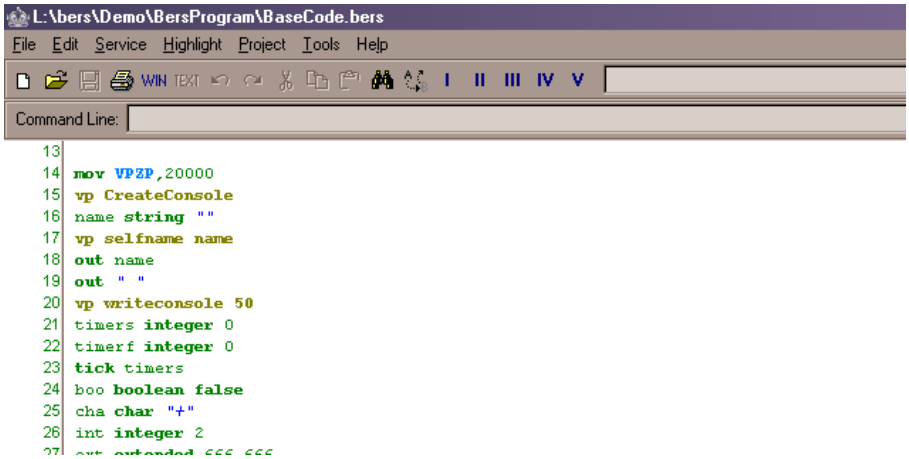

Рис. 6. Середовище для написання програмного коду

```
26  int integer 2
27  ext extended 666.666
28  str string "str"
29  mov boo,true
30  if boo = true
31    dinamlab string ""
32    mov dinamlab,"din_1"
33    jmp din_test
34    din_2 label
35    jmp din_exit
36    din_test label
37    jmp dinamlab
38    din_1 label
39    mov dinamlab,"din_2"
40    jmp din_test
41    din_exit label
42    mov str,"is"
43    case str
44      ifc "-str-": out "error"
45      ifc "123": out "error"
46      ifc 4: out "error"
47      ifc "-": out "error"
48      ifc "is":
49        pusha
50        push boo
51        mov vstr0,"str_s_s_s"
52        repeat
53          sub vstr0,"_s"
54          mov vpeax,6
55          while vpeax = 0
56            dec vpeax
57          endw
58        endr vstr0 = "str"
59        pop boo
60        popa
61      ifc 9.9: out "error"
62    endc
63  else
64  endi
65  tick timerf
66  sub timerf,timers
67  s string ""
68  mov s,"time = "
69  add s,timerf
70  add s," ms"
71  out s
72  vp consoleend
73  ret
```

Рис. 7. Приклад програмного коду на мові BERS, який демонструє синтаксис мови

Можливості віртуального процесора. Розроблений віртуальний процесор є основою, яка дозволяє створити алгоритми, що змінюють код програми, тобто присутня автокорекція. Фактично, результат, якого

можна досягти залежить лише від уяви програміста. Алгоритми в часі можуть як регресувати, так і прогресувати. Система, що володіє автокорекцією, дозволяє реалізувати алгоритми, які вимагають зміни коду в часі (наприклад – адаптація до нових вхідних даних). Для зв'язку з операційною системою, прикладним програмним забезпеченням віртуальний процесор звертається до динамічних бібліотек.

На відміну від традиційних мов програмування, робота віртуального процесора зводиться до надання можливості під час виконання програми змінювати код і правила його обробки за алгоритмом, що являє собою частину того ж коду. Слід відзначити, що вся інформація про код і команди його обробки розміщена в спеціальній матриці.

Віртуальний процесор підтримує можливість змінювати свій код в довільному секторі й передавати керування в необхідну точку на рівні базових команд. В процесі роботи можуть створюватись нові набори команд, що забезпечується розробленими механізмами. Даний підхід дозволяє змінювати тільки команди коду, утворюючи складну систему обробки. Спочатку зміни відбуваються в матриці й далі фіксуються в оригінальному коді.

Швидкодія розробленої програми для віртуального процесора має деякі обмеження, оскільки робота відбувається безпосередньо з текстом. На аналіз і обробку тексту витрачається значно більше часу, ніж на обробку байт-кодів (Java) чи бінарного файлу (*.exe, *.com). Крім того, весь код програми розміщено в динамічній пам'яті віртуального процесора. Результати дослідження швидкодії віртуального процесора наведені на рис. 8.

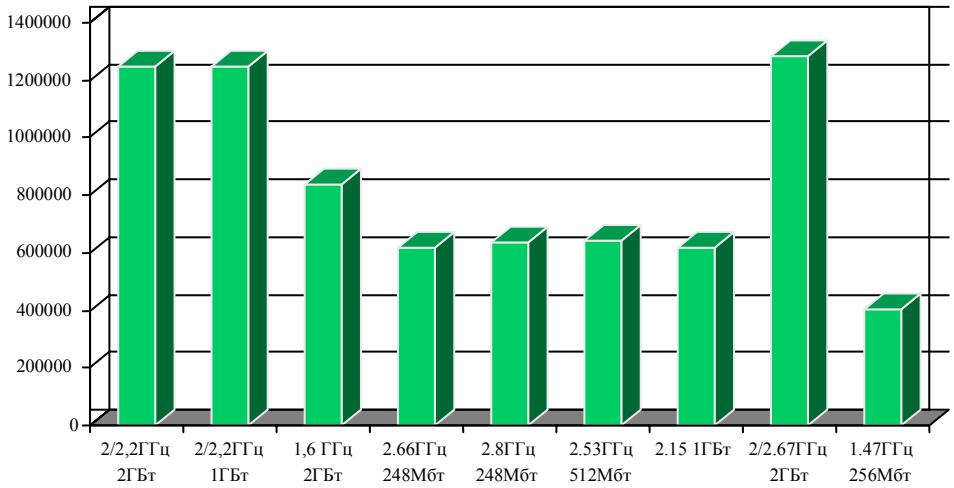

Рис. 8. Результати дослідження швидкості виконання команд програми різними комп'ютерними системами

Висновки. Основні конструкції розробленої мови програмування збігаються із загальноприйнятими, тобто лінійний рух по коду, підтримка умовних та безумовних переходів, цикли й ін.

Запропонована мова програмування має ряд недоліків:

- обмежена швидкодія;
- строгість написання коду;
- необхідність віртуального процесора під кожну апаратну платформу.

До переваг можна віднести:

- можливість створення програм реального часу;
- можливість подальшого розвитку динамічної автокорекції в реальному часі;
- поліпшене використання в навчальних цілях, через те що непотрібно перетворення у двійковий код;
- можливість працювати на довільній платформі.

Розроблений віртуальний процесор надає можливість реалізувати ідею самокорегування коду програми в процесі виконання й дозволяє будувати віртуальну модель архітектури системи з можливістю програмам змінювати себе не лише в області даних, а й в області коду та команд.

Література

1. Вирт Н. Алгоритмы и структуры данных. – М.: Мир, 1989. – 360 с.

2. Гуц А. К. Математическая логика и теория алгоритмов: Уч. пособие. – Омск: Диалог-Сибирь, 2003. – 108 с.

Надійшла 19.3.2009 р.