УДК 681.06.03

ЗАВИСИМОСТЬ ВРЕМЕННОЙ ЭФФЕКТИВНОСТИ ВЫЧИСЛИТЕЛЬНЫХ ПРОГРАММ ОТ АРХИТЕКТУРЫ СУПЕРСКАЛЯРНЫХ ПРОЦЕССОРОВ

В.И. Шинкаренко

Днепропетровский национальный университет железнодорожного транспорта
им. академика В.Лазаряна
49010, Днепропетровск, ул. акад. Лазаряна,2
т.:(056) 776-19-52 e-mail: ccp@diit-70.dp.ua

Исследуется влияние архитектуры процессора на временную эффективность вычислительных программ. Предлагается критерий оценки эффективности параллельного выполнения команд процессором применительно к конкретным программным средствам. Представлены обоснование и методика для вычисления предложенного показателя эффективности.

The influence of architecture of the processor to temporary efficiency of the computing programs is investigated. The criterion of an estimation of efficiency of parallel performance of commands by the processor with reference to concrete software is offered. Are submitted a substantiation and technique for calculation of the offered parameter of efficiency

Введение

Значительный рост производительности современных процессоров не снимает остроты проблемы временной эффективности алгоритмов[1] и программ. Достаточно много практических задач сводятся к алгоритмам с временной эффективностью $O(n^2)$ и время выполнения соответствующих программ и сейчас является значительным. Конечно, кардинальным решением проблемы является применение эффективных "жадных" алгоритмов. Но при массовом производстве программ либо квалификация программистов, либо особенности конкретных задач либо другие причины не приводят к таким решениям.

Применение современных технологий программирования, наряду с повышением надежности и сокращением объемов и сроков работ по разработке ПО, приводят к неконтролируемому росту объемов кода программ и, соответственно, ухудшению временной эффективности.

В обоих приведенных случаях, сокращение времени выполнения программ, не прибегая к пересмотру алгоритмов и программ, возможно путем повышения характеристик производительности аппаратных средств и оптимизации программ.

Особенностью системы оптимизации программ (рис. 1) при выполнении на суперскалярных процессорах является наличие нескольких уровней оптимизации с применением как программных, так и аппаратных средств.

Отметим наличие двух механизмов процессора для повышения вычислительной эффективности программы: распараллеливания (один или несколько конвейеров) и оптимизации программы (переупорядочивание команд, переименование регистров, предсказание переходов и т.п.).

И если проблемы эффективности фазы оптимизации трансляторов внимание уделяется достаточно давно и много [например 2,3], проблема эффективности работы процессора также под контролем исследователей и производителей[4], то проблема исследования эффективности всей системы и отдельных ее частей применительно к конкретным алгоритмам и программам является практически неизученной[5].

Практическое значение таких исследований несомненно. Приведем, к примеру, задачу подбора техниче-

Рис. 1. Система оптимизации программы

ских средств наиболее соответствующих разработанным программным средствам, т.е. наиболее соответствующих друг другу и естественно повышающих эффективность: программно-аппаратную систему "операционная система" – "процессор" или "система управления базами данных" – "сервер данных".

Ввиду того ,что временная эффективность вычислительных алгоритмов и программ зависит от многих факторов [6,7]:

$$E = f(v, t, x, \psi, r), \text{ где} \tag{1}$$

v - объем входных данных, t – их тип, x – значения входных данных, ψ – программная среда функционирования и создания $*$.exe файла, r - архитектура ЭВМ, оценка эффективности программно-аппаратного оптимизирующего комплекса в целом, и отдельных его составляющих, по отношению к конкретным алгоритмам является задачей достаточно сложной.

Решению одной из частных задач этого направления и посвящена данная работа. Решается задача оценки эффективности конвейерной обработки и оптимизирующей системы суперскалярного процессора при выполнении многофункциональных систем.

1. Показатели эффективности конвейерной обработки

Эффективность конвейерной обработки по отношению к конкретной программе можно оценить по степени сокращения времени выполнения программы по отношению к без конвейерной обработке:

$$E(П \mid \Psi) = \frac{T_{-k} - T_{+k}}{T_{-k}}, \text{ где} \tag{2}$$

- $E(П \mid \Psi)$ - эффективность конвейерной обработки процессором Ψ алгоритма $П$;
- T_{+k}, T_{-k} - время выполнения программы с конвейерной и без конвейерной обработки соответственно.

Так как многие программы многофункциональны (какими являются, например, ОС Windows, MS Office, большинство современного прикладного программного обеспечения) встает вопрос что понимать под временем выполнения программы.

Время конкретного выполнения программы(КВП):

$$t_{КВП} = \frac{\sum_i^N n_i \bar{t}_i}{M} = \frac{\sum_i^N n_i (t_i - \tau_i)}{M}, \text{ где} \tag{3}$$

- i – номер команды процессора в некотором полном списке команд;
- n_i - количество выполненных i-х команд при КВП;
- $\bar{t}_i$ - среднее время выполнения i-й команды (в тактах);
- t_i - время процессора для выполнения i-й команды (без конвейера);
- τ_i - среднее сокращение времени выполнения i-й команды за счет конвейерной обработки и оптимизации порядка выполнения команд на конвейере;
- M – тактовая частота процессора.

Тогда:

$$E(КВП \mid \Psi) = \frac{\sum_i^N n_i t_i - \sum_i^N n_i (t_i - \tau_i)}{\sum_i^N n_i t_i} = \frac{\sum_i^N n_i \tau_i}{\sum_i^N n_i t_i} = \frac{\sum_i^N \frac{n_i}{N} \tau_i}{\sum_i^N \frac{n_i}{N} t_i} = \frac{\sum_i^N \tilde{p}(x_i) \tau_i}{\sum_i^N \tilde{p}(x_i) t_i} \text{ или}$$

$$E(КВП \mid \Psi) = \frac{\sum_i^N \tilde{p}(x_i) \tau_i}{\sum_i^N \tilde{p}(x_i) t_i}, \text{ где} \tag{4}$$

- $\tilde{p}(x_i)$ - вероятность выполнения i-й команды при конкретном выполнении программы.

В предположении, что вероятность появления команды в цикле и линейном участке одинакова можно считать, что вероятность выполнения команды равно вероятности появления команды в программе. Это ограничение является тем более слабым, чем больше команд в программе и разнообразнее функциональность при выполнении программы. При этом предположении эффективность конвейерной обработки можно оценить следующим образом:

$$E(КВП \mid \Psi) = \frac{\sum_{i}^{N} p(x_i)\tau_i}{\sum_{i}^{N} p(x_i)t_i}, \text{ где} \qquad (5)$$

- $p(x_i)$ - вероятность появления i-й команды в коде программы.

При этом, вероятность появления i-й команды в коде программы гораздо легче определить, чем вероятность выполнения i-й команды при конкретном выполнении программы. Кроме того, оценка для каждого КВП будет одинаковой и, следовательно, оценка (5) является оценкой программы в целом. Другими словами, оценкой эффективности конвейерной обработки для программы в целом будет:

$$E(П \mid \Psi) = \frac{\sum_{i}^{N} p(x_i)\tau_i}{\sum_{i}^{N} p(x_i)t_i} \qquad (6)$$

Далее рассмотрим порядок определения $p(x_i)$, t_i и τ_i.

2. Статистическое исследование кода программы

Для определения $p(x_i)$ выполняется исследование кода программы. Под ОС Windows разработана программа, функционально близкая дисассемблеру, которая распознает команды выполнимых файлов и выводит их полный список. Перечень подсчитываемых команд определен согласно [8,9 и др.] и данных официального сайта фирмы Intel. Обрабатываются все сегменты кода .text и .CODE всех *.exe и *.dll файлов PE формата [10] исследуемого программного средства (ПС). Программа анализа подсчитывает общее количество команд.

Во всех, без исключения, исследованных ПС вероятность появления команд (в отсортированном в порядке убывания списке) снижается более, чем экспоненциально.

Рассмотрим, для примера, вероятностные показатели для таких программных средств как ОС Windows 95 (выделено около 0,8 млн. команд в 2,1Мб кода 25 файлов), Windows 98(выделено около 1,6 млн. команд в 5,4Мб кода 30 файлов) и Windows 2000(выделено около 40 млн. команд в 117Мб кода 1211 файлов).

На рис.1. показаны дифференциальная и интегральная функции распределения команд для ОС Windows 95,98 и 2000. На рисунке а) и б) представлена дифференциальная функция распределения вероятностей с обычной и логарифмической шкалой соответственно, на в) – интегральная. Команды отсортированы в порядке убывания вероятностей в кодах Windows 2000.

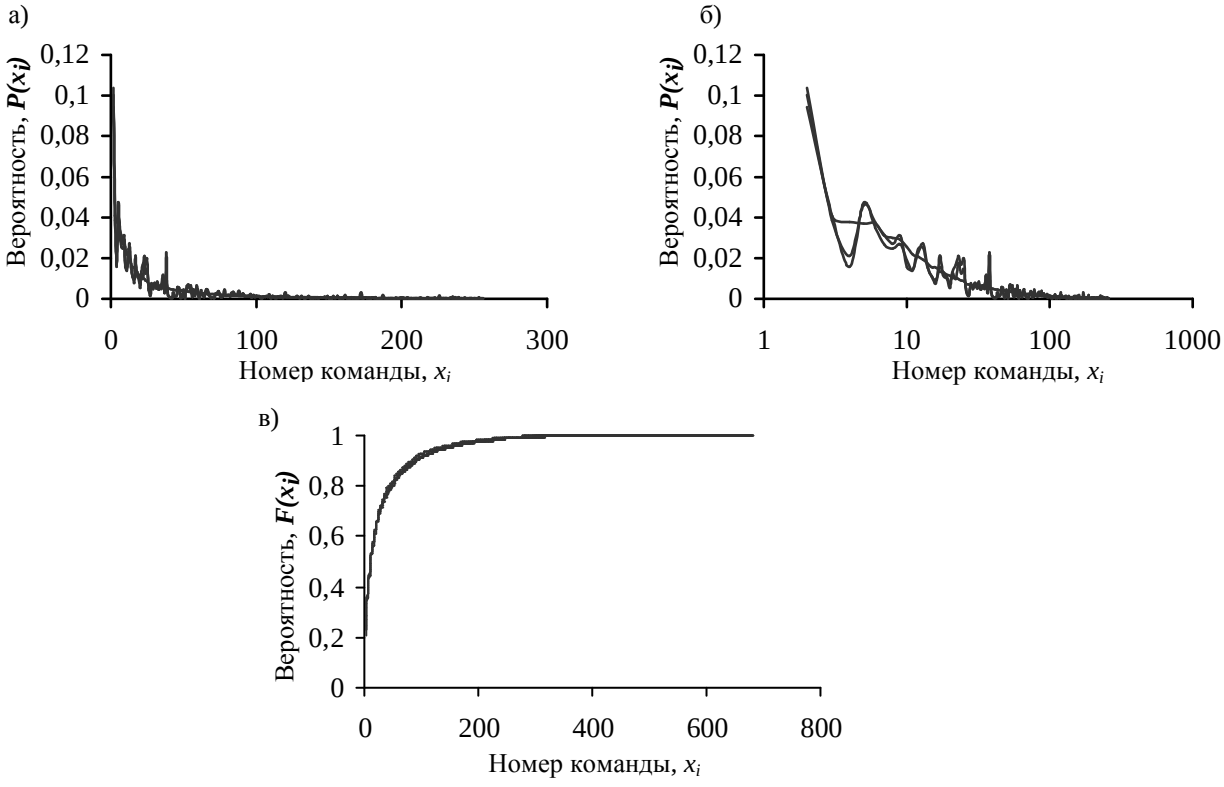

Рисунок 1. Распределение вероятности появления команд в коде программного обеспечения ОС Windows 95,98 и 2000.

Можно отметить, что из 680 команд и их модификаций суммарную вероятность 0,9 дают ~90 наиболее вероятных команд, 0,95 – ~140, 0,99 – ~250 команд. В табл.1 приведены вероятности появления 25 наиболее часто встречающихся команд в исследованных ПС.

Таблица 1. Наиболее часто встречающиеся команды процессора

Команда процессора	Вероятность появления команды $p(x_i)$		
	Windows 95	Windows 98	Windows 2000
MOV r32,r/m32	0,1029	0,1263	0,1371
PUSH r32	0,1004	0,1037	0,0943
POP r32	0,0408	0,0386	0,0409
ADD r/m8,r8	0,0379	0,0209	0,0157
CALL rel32	0,0369	0,0459	0,0471
MOV r/m32,r32	0,0369	0,0375	0,0349
LEA r32,m	0,0310	0,0311	0,0265
JE rel8	0,0299	0,0271	0,0247
PUSH r/m32	0,0288	0,0311	0,0268
CALL r/m32	0,0255	0,0156	0,0189
INC r32	0,0217	0,0145	0,0139
PUSH imm8	0,0208	0,0226	0,0244
TEST r/m16,r16	0,0192	0,0272	0,0249
JNE rel8	0,0174	0,0160	0,0152
PUSH imm32	0,0154	0,0107	0,0110
DEC r32	0,0153	0,0095	0,0080
MOV r32,imm32	0,0134	0,0156	0,0213
XOR r32,r/m32	0,0131	0,0130	0,0117
JMP rel8	0,0119	0,0107	0,0101
CMP r32,r/m32	0,0113	0,0074	0,0053
RET imm16	0,0104	0,0123	0,0134
JMP rel32	0,0102	0,0097	0,0168
CMP r/m32,imm8	0,0098	0,0175	0,0210
MOV r/m32,imm32	0,0090	0,0124	0,0168

В табл.1 приняты следующие обозначения: r-регистр, r/m – регистр или память, rel – относительный адрес, imm – непосредственный операнд. Число рядом с мнемоникой показывает соответствующую разрядность.

3. Методика определения времени выполнения команд

Время выполнения команд процессором без конвейерной обработки может быть получено из соответствующих таблиц фирмы разработчика. Однако, во-первых, не для любого процессора такую информацию можно получить, во-вторых, таблицы достаточно громоздки и могут содержать технические ошибки, в-третьих, речь может идти о средних значениях при конвейерной обработке. Поэтому такие характеристики желательно получать путем измерений. С другой стороны, опытным путем определить t_i сложно, т.к. “отключить” конвейерную обработку невозможно. Предлагается вместо времени выполнения команд без конвейерной обработки определять максимальное время выполнения команды на конвейере.

Методика измерений строится на том, что измеряется время выполнения цикла с командой (рис.2) и без нее. Так как многозадачные операционные системы могут вносить погрешности в измерения и изменять работу конвейера при переключении задач, исследования следует проводить либо без операционной системы либо в однозадачной ОС. Наши исследования проведены в MS-DOS 6.22.

Измерительная система построена следующим образом. Вход в цикл происходит непосредственно после прерывания от таймера, условием завершения цикла является следующее прерывание от таймера. Определяется число полных циклов между прерываниями от таймера. Так как прерывания от таймера поступают 18.206497192 раз в секунду несложно вычислить время выполнения цикла в секундах, а зная частоту процессора и в тактах. Маскировались все прерывания, а обработчик прерывания от таймера заменялся таким образом, что он заменял некоторую переменную (в нашем примере – Start=1).

Конвейер до выполнения команды не будет “чистым”, команды организации цикла, инкрементации счетчика, сохранения/восстановления регистров образуют некоторую “предысторию”. При этом влияют команды как до вставленной в цикл исследуемой команды, так и после нее. Чтобы снизить это влияние, строится несколько таких циклов с включением 1, 2, 3 и т.д. исследуемых команд. И, чтобы максимально задержать команду на конвейере, операнды повторяемой исследуемой команды полностью одинаковы.

Для определения t_i строится линейная регрессионная зависимость времени выполнения цикла от количества вставленных в цикл исследуемых команд. Коэффициенты регрессии определяются методом наименьших квадратов. Коэффициент при количестве команд и определяет время выполнения одной исследуемой команды.

Анализ показывает, что зависимость фактически не является линейной, но близка к ней и неплохо аппроксимируется линейной. Пример такой зависимости для команды mov reg16,reg16 приведен на рис. 3. Маркерами помечены экспериментальные точки.

```
  MVC    word ptr [Start],0    ;  обнуляется переменная, которая изменяется нашим обработчиком
                               ;  прерывания от таймера
Li:
  CMP    word ptr [Start],0    ;  цикл ожидания прерывания от таймера
  JZ     Li

   MVC    word ptr [Start],0
Li+1:
  CMP    word ptr [Start],0    ;  выполнение цикла до следующего прерывания от таймера
  JNZ    Li+2
     mov oldSP, SP             ;  сохранение регистров
     mov oldBP, BP
```

Исследуемая команда: SHR byte ptr [m4],01h

СМЕСЬ КОМАНД:

```
CMP
CMP cx,01h
ADD dl,byte ptr [m15]
PUSH si
MOV byte ptr [m16],01h
PUSH ax
LEA bx,word ptr [m17]
PUSH cx
PUSH dx
CMP word ptr [m18],ax

MOV bl,byte ptr [m19]
PUSH cx
ADD byte ptr [m1],dl
SHR si,01h
POP ax
PUSH word ptr [m2]
OR byte ptr [m3],bl
MOV cl,byte ptr [m4]
ADD byte ptr [m5],dl
SAL si,01h
```

```
     mov SP, oldSP             ;  восстановление регистров
     mov BP, oldBP
   ADD    word ptr [Counti],1    ;  изменение соответствующего счетчика
   ADC    word ptr [Counti],0    ;
   JMP    Li+1:
Li+2:
   end;
```

Рисунок 2. Цикл для измерения времени выполнения команды.

Рисунок 3. Зависимость времени выполнения цикла от количества вставленных исследуемых команд

Далее определяется время выполнения команды $t_i - \tau_i$ в условиях приближенных к ее использованию в исследуемом ПС. Для этого в цикл (рис.2) вставляется смесь команд. Моделирование смеси осуществляется в соответствии с вероятностями появления команды в коде ПС и вероятностями модификации команды: операнд регистр/память, длина операнда, разрядность команды.

Маловероятные команды - команды, которые дают суммарную вероятность менее 1%, для снижения размерности задачи отбрасываются, и в дальнейших исследованиях не используются. Однако, количество команд остается достаточно большим. Строить для каждого эксперимента отдельный EXE модуль накладно. Подготовлена программа на языке высокого уровня (C++ Bilder 6.0), которая строит ассемблерный код для многих исследуемых команд сразу. Эта программа готовит EXE модули для выполнения под DOS:

- для каждой исследуемой команды отдельный модуль с циклами с включением 1, 2, 3 и т.д. исследуемых команд;
- модули с циклом со вставленной смесью команд(20 команд) и циклами со вставленной той же смесью команд и поочередно вставленными в середину смеси каждой из исследуемых команд. Таких модулей с разными смесями команд готовится достаточно много (200).

Время выполнения команды в смеси команд, характерной для исследуемого ПС, определяется как разность времени выполнения цикла со смесью команд со вставленной в середину исследуемой командой и такого же цикла без этой команды. Так как смесь команд строится случайным образом, время $t_i - \tau_i$ определяется как среднее по разным смесям.

В связи с трудностями при выполнении смоделированной смеси команд предлагается определять упрощенную оценку $E\_(\Pi \mid \Psi)$. Трудности связаны с тем, что случайная и по сути бессмысленная последовательность команд должна выполняться. Однако, выполнение некоторых команд должно предваряться вполне определенными другими командами и состоянием технических и прораммныхсредств.

Упрощенная оценка учитывает ограниченный список команд. При этом из списка исследуемых исключаются:

- команды управления процессором и процессами (HLT, LGDT, LIDT и т.п.). Для выполнения таких команд необходима достаточно сложная дополнительная индивидуальная подготовка;
- команды ввода/вывода (IN, OUT и т.п.). В данном исследовании речь идет о вычислительных программах;
- команды переходов (CALL, LEAVE, IRET , JMP и т.п.). Они изменяют состояние конвейера. Отдельно следует изучать возможности предсказания переходов. Моделирование многих из них затруднительно (например, многие модификации CALL);
- могут исключаться специфические команды (команды MMX, сопроцессора и т.п.).

При этом может быть отброшено 20-25% команд кода ПС. Несмотря на такое усечение, упрощенная оценка дает представление о качественном использовании конвейерной обработки конкретного процессора применительно в исследуемому ПС.

4. Расчет показателей временной эффективности

Исследовалось качество конвейерной обработки процессоров Intel Pentium III E model 8 stepping 3 и AMD Athlon model 8 stepping 1 при работе ОС Windows 95/98/2000. В табл. 2 для некоторых команд приведены максимальное время выполнения одной команды на конвейере и среднее время ее выполнения в смеси команд, характерной для Windows 2000 при выполнении на Intel Pentium III E процессоре. Отметим, что максимальное время выполнения команды может превышать время приводимое в справочных таблицах[8].

Таблица 2. Время выполнения команды

Команда процессора	Максимальное время выполнения команды процессором	Среднее время выполнения команды в смеси команд, характерной для Windows 2000
MOV r16,r16	0,358004	0,362773
PUSH r16	1,723991	1,612239
POP r16	1,070719	-1,04873
MOV m16,r16	1,849523	0,76846
LEA r16,m	0,576157	0,302028
INC r16	0,501376	1,138526
PUSH imm8	1,723719	1,614384
TEST r16,r16	0,365928	0,578639
PUSH imm32	1,724152	1,520562
DEC r16	0,499769	1,047758
MOV r16,imm16	0,378911	0,284271
XOR r16,r16	0,482732	0,813029
CMP r16,r16	0,365409	0,497182

В табл. 3 приводятся значения упрощенной оценки эффективности распараллеливания при выполнении команд процессором для двух конкретных процессоров.

Таблица 3. Упрощенная оценка эффективности конвейерной обработки

Процессор	Оценка эффективности в исследуемом ПС		
	Windows 95	Windows 98	Windows 2000
Intel Pentium III E model 8 stepping 3 650 Mh	0,28	0,34	0,30
AMD Athlon model 8 stepping 1 1400 Mh	0,06	0,10	0,25

Согласно полученным результатам эффективность конвейерной обработки $E\_(Windows95/97/2000 \mid Intel\ Pentium\ III\ E\ model\ 8\ stepping\ 3)$ составляет 0,28-0,34. За счет параллельного выполнения команд на конвейере время выполнения программ ОС Windows в части реорганизации данных и целочисленных вычислений сокращается примерно на треть. Значительно меньше соответствующая эффективность для процессора AMD Athlon model 8 stepping 1.

Заключение

Предлагаемый показатель оценки эффективности конвейерной обработки и методика его определения дают возможность оценить эффективность конкретной архитектуры конвейера для конкретных программных средств.

Литература.

1. *Кормен Т., Лейзерсон Ч., Ривест Р.* Алгоритмы: построение и анализ. М.: МЦНМО, 2001 - 960с.
2. Корнеев В.В., Киселев А.В. Современные микропроцессоры. М.:НОЛИДЖ, 1998 –240с.
3. Ахо А., Ульман Дж. Теория синтаксического анализа, перевода и компиляции. т.2. М.: Мир, 1978 – 487с.
4. Скворцов С.В. Оптимизация кода для суперскалярных процессоров с использованием дизъюнктивных графов // Программирование. – 1996. -№2. – с. 41-52.
5. *Шинкаренко В.И.* Об оценке эффективности алгоритмов с учетом архитектуры ЭВМ // Компьютерное моделирование. Международная научно-методическая конференция. Тезисы докладов. Днепродзержинск: ДГТУ, -2000 – с. 268-269.
6. *Шинкаренко В.И.* Особенности оценки эффективности вычислительных алгоритмов // Проблемы программирования. –2001 - № 1-2. – с.23-29.
7. *Шинкаренко В.И.* Сравнительный анализ временной эффективности функционально эквивалентных алгоритмов // Проблемы программирования. –2001 - № 3-4. – с.31-39.
8. Hyde R. The Art of Assembly Launguage. San Francisco: NoStarchPress, 1996 – 928pp.
9. Рудаков П.И., Финогенов К.Г. Язык ассемблера: уроки программирования. М.:ДИАЛОГ-МИФИ, 2001 – 640с.
10. Пирогов В.Ю. Ассемблер для Windows.М.: Издатель Молгачева С.В., 2002 – 552с.