парных уравнений (9), (10), а следовательно, искомое решение для изображения $\bar{\theta}(r, z, s)$ (7). Оригинал $\theta(r, z, \tau) = L^{-1}[\bar{\theta}(r, z, s)]$ находится известными способами, либо с использованием обширных таблиц обратного преобразования Лапласа [3, 4].

1. Уфлянд Я. С. Метод парных уравнений в задачах математической физики. Л., 1977.
2. Seddon I. Mixed boundary value problems in potentional theory. Amsterdam, 1966.
3. Бейтман Г., Эрдейи А. Таблицы интегральных преобразований. Преобразования Фурье, Лапласа, Меллина. М., 1969.
4. Прудников А. П., Брычков Ю. А., Маричев О. И. Интегралы и ряды специальных функций. М., 1983.

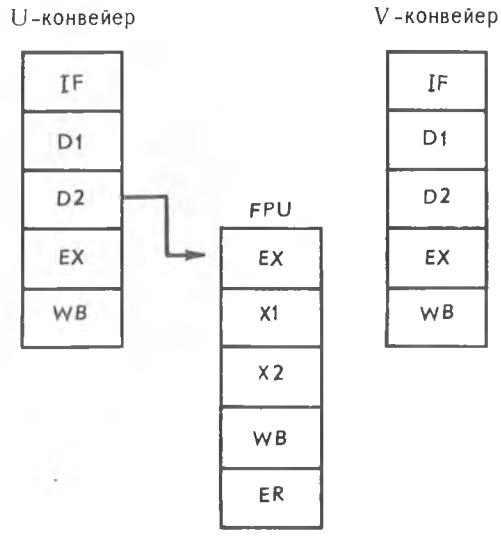
Поступила в редакцию 06.03.95.

УДК 681.3

С. С. ЗМАЧИНСКИЙ, Г. И. ШПАКОВСКИЙ,
Н. В. СЕРИКОВА, О. А. БЕЛЕЦКИЙ

АНАЛИЗ ПРОИЗВОДИТЕЛЬНОСТИ СУПЕРСКАЛЯРНОГО МИКРОПРОЦЕССОРА

A program model of the Pentium superscalar processor is described. The pipeline performances are studied with respect to optimization level of compilers. Some recomendations on programming methods and microprocessor structure improvement are suggested.



Конвейеры процессора Пентиум

Пентиум — первый суперскалярный микропроцессор (МП), разработанный фирмой Intel, вслед за i386 и i486. Его система команд включает команды i486 с некоторыми расширениями. Все программы, написанные для МП младших моделей, выполняются и на Пентиуме. Для повышения быстродействия в Пентиум внесены существенные изменения: два целочисленных конвейера (U и V) и конвейерное устройство плавающей запятой (FPU), которые могут работать независимо. Ступени конвейеров приведены на рисунке: IF — предвыборка; D1—декодирование; D2 — генерация адреса; EX, X1, X2 — вычисление; WB — запись результата; ER — сообщение об ошибке. Двойной конвейер может запускать в каждом такте две целочисленные команды или одну плавающую. Конвейер U выполняет все целочисленные команды, а также начальные ступени плавающих команд. V-конвейер может выполнять параллельно с U-конвейером простые целые инструкции и плавающую команду FXCH.

В связи с возможностью запуска параллельных команд большое значение приобретает особая оптимизация кода, позволяющая максимально загрузить ступени всех конвейеров. Однако для этого нужны специальные компиляторы. Поэтому значительный интерес представляют вопросы: насколько эффективно будут выполняться на Пентиуме уже существующие программы, что дает использование оптимизирующих компиляторов, какое максимальное ускорение может дать оптимизация? В статье делается попытка ответить на эти вопросы путем измерения процесса выполнения прикладных программ на программной модели процессора Пентиум.

В качестве примера, демонстрирующего работу целочисленных конвейеров, рассмотрим фрагмент программы на Фортране:

```
      DO 1 I = 1, N
        J = J + M
    1 A(I) = A(I) + B(I) — C(J).
```

Компилятор NDP Fortran сгенерирует для него следующий код:

```
L41:  mov   edx, [ebp] + (-8)              ;  чтение M
      add   esi, [edx]                     ;  j = j + M
      mov   eax, [edi]                     ;  чтение B(i)
      add   eax, [ecx]                     ;  A(i) + B(i)
      mov   edx, [ebp] + (-4)
      sub   eax, [edx][esi * 4] + (-4)     ;  -C(j)
      mov   [ecx], eax                     ;  запись A(i)
      add   ecx, 4
      add   edi, 4
      dec   ebx                            ;  уменьшение счетчика
      jne   L41 short
```

В табл. 1 показано, каким образом эти команды будут выполняться на ступени EX в U- и V-конвейере. Одна итерация цикла выполняется за 12 тактов. Из 11 команд цикла 7 выполнится в U-конвейере, а 4 команды — в V-конвейере. Команды ADD и SUB с обращением к памяти в АЛУ выполняются за 2 такта (при условии, что все необходимые данные находятся в КЭШе). Однотактная команда MOV в V-конвейере на ступени EX начинает выполняться только в 4-м такте (в последнем такте параллельной двухтактной команды ADD).

Таблица 1

Выполнение цикла на языке Fortran

U-конвейер	V-конвейер	Такты
mov edx, [ebp] + (-8)	. . . . .	1
.(AGI).	.(AGI).	2
add esi, [edx]	. . . . .	3
(add — 2-й такт)	mov eax, [edi]	4
add eax, [ecx]	. . . . .	5
(add — 2-й такт)	mov edx, [ebp] + (-4)	6
.(AGI).	.(AGI).	7
sub eax, [edx] [esi * 4] + (-4)	. . . . .	8
(sub — 2-й такт)	. . . . .	9
mov [ecx], eax	add ecx, 4	10
add edi, 4	dec ebx	11
jne L41 short	. . . . .	12

В процессе выполнения ступени конвейеров могут простаивать как из-за многотактных команд, так и из-за возникновения зависимости при генерации адреса на ступени D2 (AGI — Adress Generation Interlock). В рассматриваемом примере 2-й такт является тактом простоя из-за AGI, так как команда ADD использует содержимое регистра EDX, которое должно быть изменено предыдущей командой MOV. Также простоем из-за AGI является и 7-й такт. В этом случае зависимость по регистру EDX возникает между командами MOV и SUB, находящимися в разных конвейерах.

Для проведения измерений, на основании доступной информации [1, 2], была разработана и реализована модель процессора Пентиум (MP5). Некоторые детали функционирования были получены при помощи экспериментов на реальном процессоре. Модель позволяет измерять степень загрузки и простоев ступеней различных конвейеров (U, V и FPU), а также получать некоторые интегральные характеристики как для отдельных программ, так и для целого пакета.

Модель была построена со следующими допущениями:

1) процессор всегда правильно предсказывает ветвления;

2) все необходимые данные и команды находятся в КЭШе.

Плавающие инструкции частично занимают целочисленный U-конвейер, так как они проходят ступени IF, D1 и D2. Для проверки того, насколько это снижает производительность, была реализована модель MP6 суперскалярного процессора, в котором FPU-конвейер имеет независимый вход. В MP6 одновременно могут запускаться три независимые команды: две целочисленные и одна плавающая.

Таблица 2

Результаты измерений для NDP Fortran

Ступень	Модель MP5				Модель MP6			
	Загрузка, %	Простой, %			Загрузка, %	Простой, %		
		Незагрузка	AGI	Плавающие команды		Незагрузка	AGI	Зависимость в FPU
U EX	40,3	—	6,6	53,1	50,9	22,1	4,3	22,7
V EX	13,4	26,8	6,6	43,1	22,4	50,6	4,3	22,7
		Незагрузка	Зависимость в FPU					
FPU EX	31,2	50,3	18,5		37,5	35,5	4,3	22,7
FPU X1	29,5	60,5	10,0		32,1	55,8	—	12,1
FPU X2	11,0	85,6	3,4		12,1	84,2	—	3,7

Таблица 3

Результаты измерений для Watcom C

Ступень	Модель MP5				Модель MP6			
	Загрузка, %	Простой, %			Загрузка, %	Простой, %		
		Незагрузка	AGI	Плавающие команды		Незагрузка	AGI	Зависимость в FPU
U EX	47,2	—	1,7	51,1	58,5	13,7	1,4	26.4
V EX	20,9	26,3	1,7	51,1	28,9	43,2	1,4	26,4
		Незагрузка	Зависимость в FPU					
FPU EX	34,5	48,9	16,6		39,3	32,9	1,4	26,4
FPU X1	29,5	58,3	12,1		29,7	56,5	—	13,8
FPU X2	11,3	84,5	4,2		12,3	83,1	—	4,6

Эксперимент проводился путем имитации выполнения на моделях MP5 и MP6 пакетов научных программ. На вход модели поступает программа на языке Ассемблера IBM PC. Известно, что основной объем вычислений сосредоточен во внутренних циклах. Поэтому имитировалось выполнение только внутренних циклов и при этом накапливались данные по загрузке и простоям ступеней конвейеров. Моделируемые циклы не должны были содержать команд CALL и RET. Также не рассматривались циклы, в которых встречались многотактные плавающие команды (F2XM1, FSQRT и др.).

На моделях MP5 и MP6 были проведены измерения для двух пакетов

программ. В качестве «стандартного» был взят компилятор NDP Fortran версии 2.1 с оптимизацией кода для процессора i386/387. Пакет на Фортране состоял из 57 научных подпрограмм, взятых из стандартных библиотек NAG Fortran и SSP. В качестве «оптимизирующего» был взят один из лучших компиляторов Watcom C версии 9.5, имеющий опцию оптимизации кода для Пентиума. Пакет на Си состоял из 15 научных программ (системы уравнений, анализ Фурье и т. п.).

В табл. 2 и 3 приводятся результаты измерений для пакетов на Фортране и на Си: загрузка и простой (в процентах от суммарного времени выполнения) исполнительных ступеней U- V-конвейера (U EX и V EX), а также загрузка и простой исполнительных ступеней FPU (FPU EX, FPU X1 и FPU X2).

Следует отметить, что простои возникают по разным причинам. Ступень U EX может простаивать из-за AGI и выполнения плавающих команд в FPU. Ступень V EX, кроме этого, простаивает из-за незагрузки команд в V-конвейер. Ступени FPU простаивают по двум причинам: непоступление команд в FPU (незагрузка) и из-за зависимости плавающих команд через вершину стека.

Простои в модели MP6 несколько отличаются от MP5. Так, ступени U EX и V EX простаивают из-за незагрузки (например, если идут только плавающие команды), AGI и из-за возникновения зависимостей на ступени FPU EX. В свою очередь, ступень FPU EX может простаивать из-за возникновения AGI в целочисленных конвейерах.

Одной из основных интегральных оценок производительности МП является среднее количество команд, выполняемых за один такт (архитектурная скорость):

$$R = \frac{N_{команд}}{N_{тактов}}. \qquad (1)$$

Для оценки потенциальных возможностей оптимизации можно вычислить значение «максимальной» производительности для случая, когда ступени всех конвейеров максимально загружены и сбалансированы. В этом случае все целочисленные команды поровну распределены между U- и V-конвейерами и отсутствуют задержки из-за зависимости данных в FPU. Количество тактов в этом случае минимально, и для модели MP5 его можно вычислить как сумму усредненной загрузки U- и V-конвейеров, задержки из-за AGI и времени плавающих операций:

$$N_{min} = \frac{T_{u\,ex} + T_{v\,ex}}{2} + T_{agi} + T_{fpu\,ex}.$$

Для модели MP6 минимальное количество тактов равно:

$$N_{min} = \min\left\{ \frac{T_{u\,ex} + T_{v\,ex}}{2} + T_{agi},\ T_{fpu\,ex} \right\}.$$

Таблица 4

Сравнительные характеристики пакетов

Компилятор	Программ	Циклов	Команд	Тактов	Модель MP5			Модель MP6		
					R	R_{max}	ΔR, %	R	R_{max}	ΔR, %
NDP Fortran	57	537	10460	17988	0,58	0,90	55	0,63	1,51	139
Watcom C	15	83	907	1087	0,83	1,18	41	0,91	2,00	119

Воспользовавшись значениями N_{min}, можно по (1) для каждой модели вычислить предельную скорость R_{max} и определить ее относительное увеличение ΔR (табл. 4). Значение ΔR показывает рост производительности в случае идеальной оптимизации, что позволяет оценить резервы дальнейшей оптимизации кодов.

Из результатов эксперимента видно, что загрузка U-АЛУ для обоих пакетов составляет 40 — 50 %, а V-АЛУ — 10 — 20 %. Плавающее АЛУ было загружено примерно на 30 % на ступени FPU EX. Однако общая производительность оказалась невысокой и составила 0,6 — 0,8 команд

за такт. Переход к модели MP6 увеличивает загрузку целочисленных АЛУ и производительность в среднем на 10 %.

Компилятор Watcom обеспечивает более высокие характеристики. Это выражается в лучшей загрузке V-конвейера, меньших затратах из-за AGI и увеличении производительности почти на 40 %. Широко используются сложные адресные выражения, которые вычисляются за один такт на ступени D2. Для сравнения рассмотрим код, который сгенерирует для рассмотренного выше цикла, написанного на языке Си, компилятор Watcom:

```
L1      add     EBX, EDI                              ;  j = j + M
        mov     EDX, dword ptr [ESP + EAX]            ;  чтение A(i)
        mov     ECX, dword ptr + 320h[ESP + EAX]      ;  чтение B(i)
        mov     EBP, dword ptr + 190h[ESP + EBX*4]    ;  чтение C(j)
        add     ECX, EDX                              ;  A(i) + B(i)
        add     EAX, 00000004H                        ;  счетчик
        sub     ECX, EBP                              ;  – C(j)
        mov     dword ptr — 4h[ESP + EAX], ECX        ;  запись A(i)
        cmp     EAX, ESI
        jne     short L1
```

Этот цикл из 10 команд будет выполняться всего за 7 тактов вместо 12 у NDP (табл. 5). Остается только один такт простоя из-за AGI. Кроме этого, используются однотактные арифметические команды формата регистр — регистр, а NDP использует команды формата регистр —память. Практически компилятор Watcom для генерации кодов использует load/store модель, которая, как известно, дает преимущество для суперскалярных и RISC-микропроцессоров.

Таблица 5

Выполнение цикла на языке Си

U-конвейер	V-конвейер	Такты
add EBX, EDI	mov EDX, dw. p. [ESP + EAX]	1
(AGI).	. . . . .	2
mov ECX, dw. p. + 320h[ESP + EAX]	mov EBP, dw. p. + 190 h [ESP + EBX*4]	3
add ECX, EDX	add EAX, 00000004h	4
sub ECX, EBP	. . . . .	5
mov dw. p. – 4h[ESP + EAX], ECX	cmp EAX, ESI	6
jne short L1	. . . . .	7

Потенциальные резервы оптимизации составляют для модели MP5 40 — 50 %, что может обеспечить предельную производительность около одной команды за такт. Для модели MP6 резерв значительно выше и максимальная производительность может доходить до 1,5 — 2 команд за такт.

Выполненная работа показывает, что программные модели суперскалярных процессоров являются основным инструментом исследования их характеристик. В отношении CISC МП типа Пентиум модель позволила установить, что имеются значительные резервы для увеличения архитектурной скорости как за счет совершенствования оптимизирующих компиляторов, так и за счет изменения структуры микропроцессора. Получены количественные оценки реальной и максимальной архитектурной скорости для моделей P5 и P6. Построенная модель может быть использована для обучения программированию на Пентиуме, для оценки качества программ, а также для ручной оптимизации программ на языке Ассемблера.

1. Pentium Processor User's Manual; Architecture and Programming Manual — Intel Literature Sales. 1993. V. 1 — 3. P. 870.
2. S c h m i t M. // Dr. Dobb's Journal. 1994. January. P. 40.
3. С т э м Н. // P. C. Magazine. 1993. № 3. С. 35. (пер. с англ.)

Поступила в редакцию 04.07.94.