Санкт-Петербургский государственный университет

Изучение системы выполнения инструкций процессоров Intel

Курсовая работа

Выполнил:
Студент 243 группы
Чернявский А.А.

Научный руководитель
ст.п. кафедры СП
Баклановский М.В.

Американская корпорация Intel известна, прежде всего, как ведущий производитель микропроцессоров для персональных компьютеров. Долгое время определяющей характеристикой скорости процессора являлась его тактовая частота – количество операций в секунду, измеряемое в герцах. Причина заключалась в том, что все низкоуровневые команды, к которым сводились любые привычные для нас действия, занимали строго фиксированное число тактов процессора.
Но прогресс не стоял на месте, и понемногу в архитектуры процессоров внедрялись новые технологии, такие как:
Конвейерная обработка – организация вычислений, при которой инструкция делится на N независимых этапов (микрокоманд), позволяя в каждый момент времени выполнять микрокоманды идущей последовательности инструкций по схеме ниже, увеличивая таким образом скорость прохода в n раз.

Простой пятиуровневый конвейер в RISC-процессорах

Многоуровневое кэширование – ввод 1-3 уровней дополнительной памяти таким образом, что каждый более «низкий» уровень хранит более вероятно востребованные на данный момент данные, имеет меньший объем, но более быстрое время доступа
Внеочередное выполнение: инструкции помещаются в очередь и выполняются не в прямом порядке следования, а в зависимости от готовности их к исполнению (готовность к исполнению определяется по готовности операндов)
Всё это в совокупности, с одной стороны, повлекло заметное ускорение выполнения инструкций, но с другой – сильно уменьшило предсказуемость производительности процессора, т.е. стёрло понятие точного времени выполнения инструкций (далее – ВВИ). Таким образом, такие изменения в архитектуре повлекли за собой непредсказуемость поведения, и Intel взамен ввело понятие **прогнозируемого ВВИ,** прямо указав в своей документации, что реальное ВВИ может сильно превышать прогнозируемое.
В связи с этим возникла идея изучить систему выполнения инструкций процессоров Intel, сделать некоторые выводы относительно её поведения в реальных условиях.

Основным инструментом исследований в данной теме является программа t32, которая замеряет количество тактов, ушедших на выполнение определённой последовательности действий.
На вход ей подаются следующие аргументы:

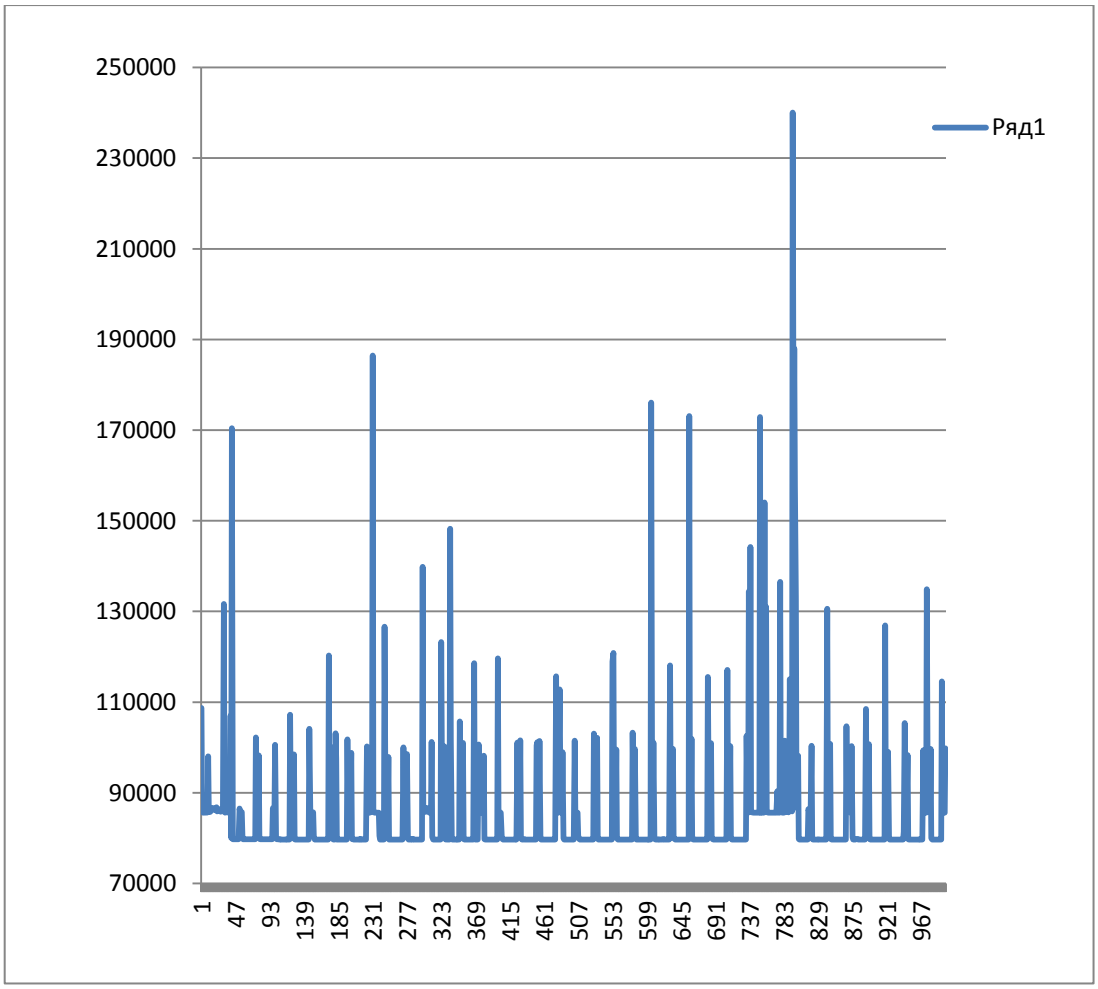
1) Тип барьера (m|x|mx|x: m – mfence, x – cpuid, mx – mfence&cpuid, n – none)
   Барьер – средство для борьбы с переупорядочиванием.
   mfence – инструкция-барьер, гарантирующая что все операции чтения и записи до неё будут завершены, по отношению к операциям, расположенным после неё.
   cpuid – инструкция получения данных о процессоре, также выполняет полный барьер памяти
   В дальнейшем для определённости будет использоваться аргумент mx, т.е. использование двух барьеров сразу.
2) Место хранения показателя счётчика тактов (m|p|x: m – память, p – стек, x – mmx-регистры)
   В дальнейшем для определённости будет использоваться аргумент m, т.е. запись счётчика тактов в память
3) Количество выполняемых замеров – последовательности действий, которые будут приведены ниже. Диапазон от 1 до 2^32 – 1 замеров.
4) Количество выполнений измеряемой инструкции и её мнемоника (может быть несколько инструкций). Верхний предел количества таких выполнений определяется длиной мнемоники инструкции.
   В результате будет выполнена следующая последовательность действий указанное в п.3 количество раз:
   1) Выполнение инструкции-барьера
   2) rdtsc – считывание показателя счётчиков тактов и их сохранение в место, указанное в п.2 выше
   3) Выполнение инструкции-барьера
   4) Выполнение инструкций (сама инструкция и её количество определяются из п.4 выше)
   5) Инструкция-барьер.
   6) rdtsc, и из последнего показателя счётчика тактов вычитаются сохранённые прошлые показатели.

Полученная разность и будет результатом выполнения программы.
Таким образом, в результате можно получить число (или последовательность чисел) тактов, ушедших на сохранение показателя счётчика тактов, прохождение барьеров и выполнение определённого количества инструкций.
Например, если дать команду
t32 mx m 10 1000 43 >output.txt
то в файле output.txt будет 10 чисел, каждое из которых обозначает количество тактов, пошедших на прохождение барьеров mx, считывание и запись в память результатов rdtsc, выполнение 1000 раз подряд инструкции, соответствующей мнемонике 43 (inc eax). Уточним сразу: результаты не в полной мере характеризуют ВВИ инструкции, т.к., запись показаний rdtsc (п.2) и

прохождение инструкций-барьеров также занимает определённое количество времени.

В качестве вспомогательных инструментов по мере писались скрипты на Perl и командные файлы для увеличения автоматизации возникающих исследований.

В первую очередь было решено проверить, а действительно ли ВВИ перестало быть точным значением в общем случае. В качестве примера было выбрано 1000 измерений из 100 тыс. выполнений инструкции inc eax. Ниже приведён график зависимости получившегося количества тактов (вертикальная ось) от номера эксперимента (горизонтальная ось):



Результаты можно назвать противоречивыми.

С одной стороны: ВВИ оказалось явно неустойчивым, и на первый взгляд, природа таких колебаний не очень понятна. С другой стороны - в этих колебаниях можно выделить периоды, также по графику видна линия устойчивого минимума. Но, опять же, с первой стороны, даже этот устойчивый минимум колеблется.

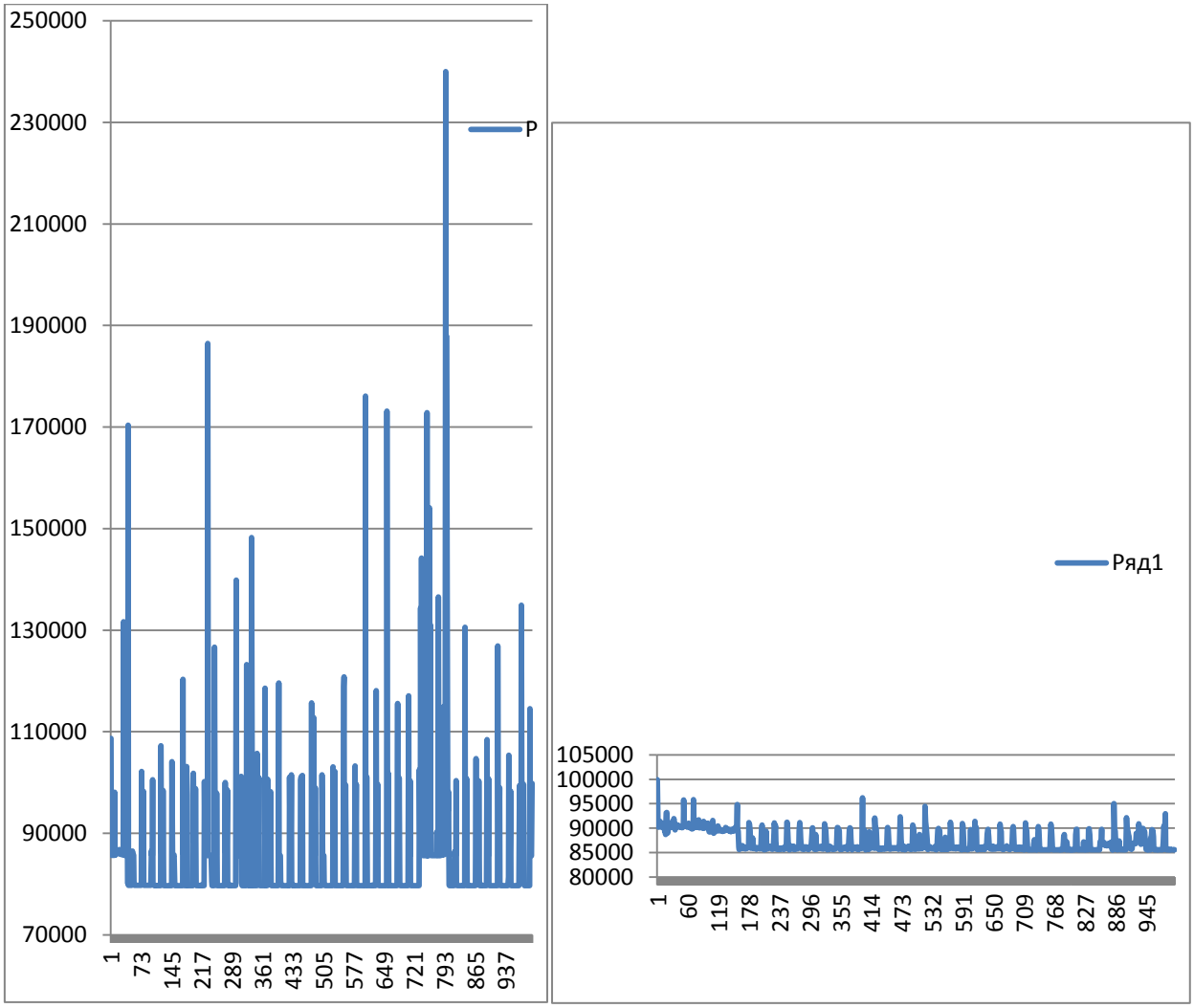
Таким образом, все опасения по поводу непредсказуемости поведения процессора подтвердились. Есть моменты, которые можно объяснить на уровне гипотез (к примеру, периодические колебания – влиянием фоновых процессов ОС), и есть явления, которые

можно объяснить далеко не сразу (к примеру, неустойчивость минимального, «опорного» минимума).

Далее возник вопрос: каким же способом можно получить наиболее точные данные о ВВИ для каждой инструкции?

Первым вариантом было введения понятия **среднего ВВИ**, которое было бы подсчитано на основе цикла измерений. Возвращаясь к прошлым (mx m 1000 100000 43) вышло среднее значение примерно в 89 тыс. тактов на 100 тыс. инструкций, т.е. 0.89 такта на инструкцию. Считать ли результат правдоподобным, если в документации для данной инструкции было указано прогнозируемое ВВИ 0.33 такта?

Возникла следующая гипотеза: хаотичность данного распределения может зависеть от текущего состояния загрузки процессора. Создать «идеальные» условия не представлялось на тот момент возможным, но в какой-то степени приблизиться к ним можно путём очистки всех процессов, каких только можно, с сохранением работоспособности ПК, и в таких условиях происходят абсолютно аналогичные замеры. Результат представлен на странице ниже, для удобства сравнения масштабы были приведены.



Слева изображён график работы в «разгруженном» режиме работы ПК, справа – в обыкновенном режиме работы. Как видно по графику, количество и диапазон колебаний уменьшились. Таким образом, гипотеза о влиянии сторонних процессов ОС

подтверждена. Один из выводов: при измерении ВВИ следует каким-то образом учитывать загруженность ПК на данный момент.
Следующий вариант основывался на идее, что получаемое значение в тактах складывается из «чистого» ВВИ т.н. «мусора», к коим можно отнести те же сторонние процессы. Заключается идея в двух кардинальных изменениях.
Первое изменение, позволяющее приблизить результат к истине – рассмотрение не среднего значения на определённом количестве замеров, а лишь его минимального значения, как наименее подверженного внешним помехам.
Второе изменение, позволяет в большой степени нейтрализовать влияние таких факторов, как данное состояние ПК или неточность измерений, о которой упоминалось после описания инструмента t32. Изменение состоит в том, что перед началом измерений выполняется измерение без инструкций, т.н. «измерение барьеров», и тогда полученный результат позволит нам узнать, какое количество тактов пойдёт на прохождение инструкций-барьеров и на запись показаний rdtsc.
Таким образом, расчётная формула для отдельно взятой инструкции выглядит следующим образом:
Пусть bar – время выполнения барьера;
t – результат, выдаваемый t32 (выполнение инструкций вместе с прохождением барьеров);
n – количество выполнения этой инструкции в памяти (т.е. между барьерами);
то ВВИ рассчитывается по формуле:
(t – bar)/n.
Формула применяется для каждой инструкции. Чтобы полученные данные в большей мере можно было назвать точными, в качестве bar (прохождение барьеров) берётся минимальное значение по результатам некоторого количества опытов, точно также поступаем с t; n варьируем от 100 до 100 тыс. с шагом *10.
Предполагается, что весь процесс будет автоматизирован внутри одного скрипта для дальнейшей возможности запуска на разных ПК, поэтому, чтобы скрипт работал быстро, было взято 1000-кратное повторение цикла измерений, т.е. значения bar и t выбирались минимальными на основании 1000 циклов.
Список инструкций брался наиболее типичный: операции mov, inc/dec, add/sub, xor, nop; для некоторых варьируются аргументы: непосредственные аргументы, регистры, их разрядность. В итоге для каждой из 24 инструкций получены 4 числа, подсчитанные при фиксированном n. Весь набор действий выполнялся примерно за 15-20 секунд.

Полная таблица с результатами для процессора i7 2630QM приведена ниже:

instruction\n	100	1 000	10 000	100 000
add ax 1	2	0.89	0.7813	0.70657
add bx 1	0.56	0.663	0.6723	0.70529
add eax 1	2	0.89	0.7813	0.72391
add ebx 1	0.77	0.769	0.7493	0.69671
dec ax	1.76	0.868	0.7203	0.71643
dec bx	0.84	0.766	0.7215	0.71501
dec eax	1.97	0.89	0,7813	0,71513
dec ebx	0.53	0.645	0.6968	0.71403
inc ax	1.76	0.868	0.7813	0.69061
inc bx	0.74	0.766	0.769	0.68945
inc eax	1.76	0.868	0.7791	0.71493
inc ebx	0.53	0.663	0.6831	0.68921
mov ax 0	2.53	2.511	2.3559	2.30037
mov bx 0	2.31	2.489	2.3539	2.29803
mov eax 0	0.25	0.297	0.6548	0.70611
mov ebx 0	0.25	0.241	0.5867	0.67831
nop	0	0.155	0.2004	0.20155
sub ax 1	2	0.889	0.7813	0.73305
sub bx 1	0.56	0.748	0.7671	0.70587
sub eax 1	2.22	0.914	0.7834	0.69953
sub ebx 1	0.25	0.769	0.7693	0.72411
xor ax ax	0.53	0.745	0.6829	0.69613
xor bx bx	0	0.176	0.2592	0.30345
xor eax eax	-0.21	0.176	0.2634	0.29967

Как видно, результаты вышли неожиданные: если числа на большем количестве выполнений (n=10k, n=100k) ещё приближённо соответствовали ожиданиям и здравому смыслу, то числа первого столбика (n=100) зачастую сильно отклонялись в обе стороны, давая даже в некоторых случаях погрешность более чем в целый такт. Особенно неадекватными выглядели такие результаты как 0 тактов (команды nop, xor ebx ebx), и даже -0,21 такта (команда xor eax eax).
Гипотеза, объясняющая это явления: даже 1000 измерений цикла явно недостаточны для выявления ВВИ, причём как для барьеров, так и для инструкций.
Для проверки гипотезы проведём простейший эксперимент: высчитаем минимальное ВВИ для барьера mx (т.е. без инструкции) для 1000, 10 000, 100 000, 500 000 и 1 000 000 измерений. Результаты вышли такими:

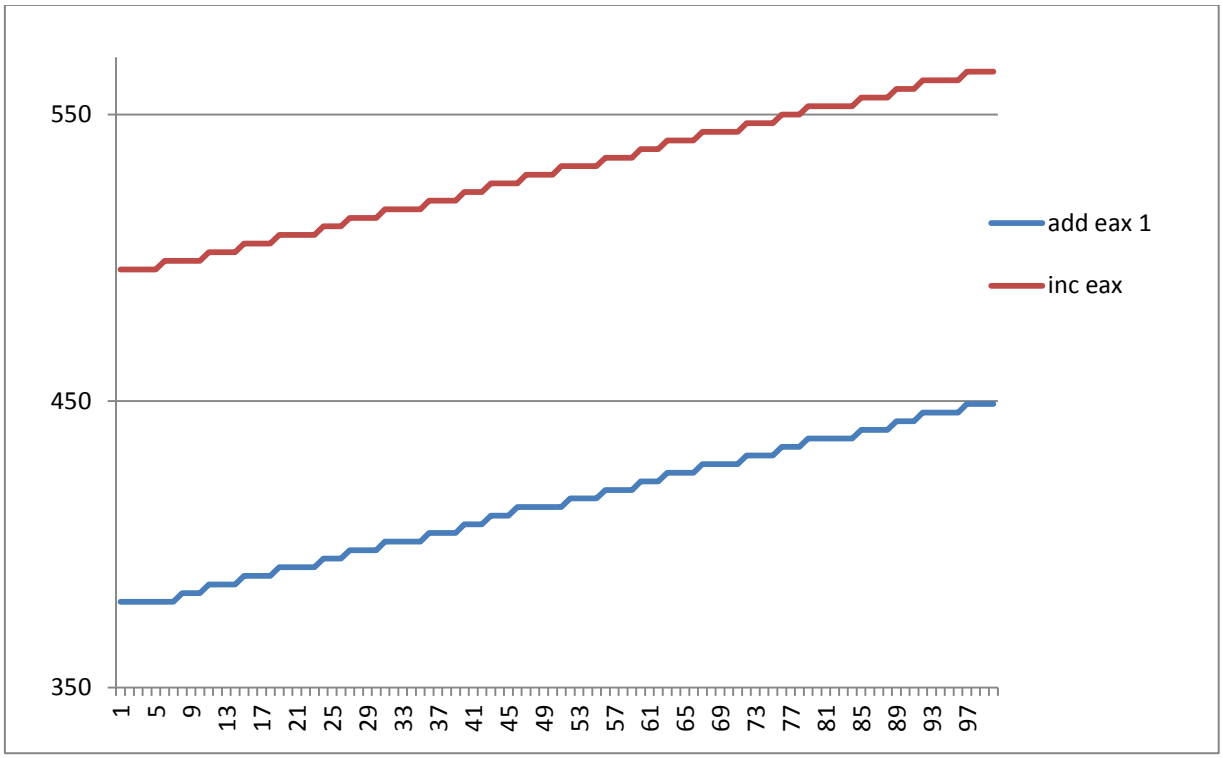
Кол-во замеров	1 000	10 000	100 000	500 000	1 000 000
Результат	452	396	377	376	376

Разница в 76 тактов, при делении на n=100 получаем неприемлемую погрешность в 0,76 такта в среднем.
Из этих результатов делаем вывод: оптимальнее всего искать минимальное значение среди 100 тыс. измерений.
В связи с увеличением в 1000 раз количества измерений, время работы скрипта увеличилось с 15-20 секунд более чем до 30 мин.
Финальная версия алгоритма дала результаты, на основании которых можно делать выводы. Скрипт работал на нескольких процессорах семейства Sandy Bridge, с частичными результатами можно ознакомиться по таблице ниже.

Instruction\Source	official manual	i5 2410M	i7 2630QM
inc eax	0.33	0.69	0.69
dec eax	0.33	0.69	0.69
add eax 1	0.33	0.69	0.69
sub eax 1	0.33	0.69	0.69
xor ax ax	0.33	0.69	0.69
nop	0.25	0.28	0.28

Полную таблицу со всеми участвующими процессорами и инструкциями нет смысла приводить, т.к. в общем и целом картина не меняется.
Таким образом, после долгих поисков методов непосредственного измерения, был получен логичный и предсказуемый результат: действительно, реальное ВВИ отличается от заявленных в большую сторону, и это разница может быть даже более чем 2-кратной.

После получения такого результата у меня возник вопрос: всё это время расчёты велись на основе многократного подряд выполнения какой-либо инструкции. А как же поведение одиночных инструкций? Чтобы рассмотреть поведение системы на таком уровне, были произведены измерения инструкций inc eax и add eax 1, прокручивающихся в памяти от 1 до 100 раз. Возник вопрос: а действительно ли график зависимости минимального значения от количества повторов инструкции в памяти будет линейным? Т.е. действительно ли каждое следующее приращение количество выполняемых инструкций будет давать каждый раз фиксированное приращение количеству затраченных на это тактов с точностью до округления до целого такта? Минимальное значение будет выбираться среди результатов 1 млн. измерений., чтобы исключить погрешности.
Оказалось, что нет. Рассмотрим графики для обеих инструкций: горизонтальная ось – количество выполнений инструкций в памяти, вертикальная – соответствующее количество тактов.



В обоих случаях вместо ожидаемого линейного графика или ступенчатого графика с шагом 1, мы видим ступенчатые графики с шагом 3. Более того, длина этих «ступенек» не постоянна, а колеблется между 3 и 6.

Объяснить это явление можно, к примеру, работой оптимизирующих технологий, описанных в начале.

На основе этих результатов возникло следующее предположение: а что если время выполнения совокупности инструкций на малых повторениях в большей степени определяется их сочетаниями, а не складывается из их ВВИ «по умолчанию»?

Воспользовавшись тем, что инструмент t32 позволяет сделать измерение вида C1 C2 C2 … C2 (n раз) вводом комбинации C1 n C2, замерим минимальное выполнение каждой из инструкций или их последовательности:

opcode	<empty>	43	40	43 1 40	43 2 40	43 3 40	43 10 40	43 100 40
result	377	380	570	567	496	469	502	579

Результат: подобие линейности можно наблюдать только лишь при многократном повторении одной и той же инструкции.

Если смотреть на уровне взаимодействия различных команд, то получаем явление, которое, казалось бы, противоречит здравому смыслу: в некоторых случаях получается, что добавление команд к последовательности УМЕНЬШАЕТ её минимальное время выполнения.

Объяснить это, опять же, можно лишь гипотетически, к примеру: для разных случаев работают разные оптимизирующие технологии.

Главный же вывод сформулировать можно так: время выполнения набора команд в реальных условиях в большей степени определяется взаимодействием этих команд, нежели их индивидуальным временем выполнения.
Таким образом, компания Intel путём ввода в архитектуру своих процессоров многочисленных оптимизирующих технологий стёрла само понятие **общего случая** для выполняемых команд.

Итак, все выводы:

1) При измерении ВВИ следует учитывать текущее состояние ПК.
2) Один из способов подходящих способов сделать это – замерить время прохождения барьера и вычесть его.
3) Чтобы убрать из рассмотрения ВВИ, на которые повлияли сторонние процессы, необходимо для каждой измеряемой последовательности выбирать минимальное значение для определённого количества измерений. Самый оптимальный вариант – 100000 измерений
4) Реальные ВВИ намного превышают заявленные Intel в своей документации.
5) Из-за особенностей архитектуры зависимость минимального ВВИ от её повторений в памяти не линейна, а ступенчата.
6) В реальных условиях время работы последовательности команд в большей степени определяется особенностями их взаимодействия, нежели их отдельными ВВИ.

Использованная литература:

1) Intel 64 and IA-32 Architectures Software Developer’s Manual
2) Torbjorn_Granlund - Instruction latencies and throughput for AMD and Intel x86 processors
3) Agner Fog. Copenhagen University College of Engineering – Instruction tables