# Теоретический метод для оценки и сравнения производительности процессоров на базе архитектуры ARM

А. А. Величко[1], А. А. Ракитский[2,*], Б. Я. Рябко[2]

[1]Сибирский государственный университет телекоммуникаций и информатики, Новосибирск, Россия

[2]Институт вычислительных технологий СО РАН, Новосибирск, Россия

*Контактный e-mail: rakitsky.anton@gmail.com

Предложен метод для теоретической оценки производительности популярных процессоров на базе архитектуры ARM путем определения вычислительной способности компьютера — характеристики, описанной Б.Я. Рябко в 2012 г. Проведен подробный анализ архитектуры ядер процессоров ARM, рассматриваются особенности, которые необходимо учитывать при определении вычислительной способности. На основе полученных результатов применения метода выполнено сравнение процессоров ARM и Intel. У этих процессоров принципиально разные подходы к организации архитектуры (разные наборы команд, принципы работы с памятью, устройство конвейера и т. д.). Кроме того, полученные результаты сравниваются с результатами общепризнанных бенчмарков. Сделаны выводы о применимости метода к реальным процессорам ARM, а также относительно возможности его применения к любым вычислительным системам с разными архитектурами.

*Ключевые слова*: процессор, вычислительная способность, производительность процессора, ARM.

## Введение

Большинство современных электронных устройств (мобильные телефоны, навигаторы, датчики, телевизоры, бытовая техника и т. д.) оснащается вычислительным процессором. Для компаний, занимающихся разработкой и производством вычислительных процессоров, актуальным является вопрос объективной оценки и сравнения их производительности, в том числе на этапе проектирования. Несмотря на стремительные темпы развития компьютерных технологий, производительность вычислительных процессоров уже долгое время оценивается только при помощи бенчмарков. Бенчмарк — это набор контрольных задач, которые запускаются и выполняются на устройстве для определения его сравнительных характеристик (время, требуемое для решения этих задач, объемы использованной памяти и т. д.). Главный недостаток этого подхода — необходимость наличия рабочей модели оцениваемого устройства, однако создание такой модели — крайне трудоемкий процесс. Кроме того, бенчмарк дает оценку работы устройства на ограниченном наборе конкретных задач, а не рассматривает все потенциально возможные задачи, что делает такой способ недостаточно объективным.

Этих недостатков можно избежать, если использовать теоретический метод оценки производительности, для применения которого достаточно лишь описания архитектуры вычислительного устройства.

Впервые метод теоретической оценки производительности компьютеров на основе определения их вычислительной способности опубликован Б.Я. Рябко в 2012 г. [1]. Метод базируется на основных идеях теории информации, и для его применения требуются только данные об архитектуре исследуемого компьютера (список инструкций процессора, время их выполнения, устройство конвейеров, объемы всех видов памяти и т. д.). В дальнейшем этот метод исследовался и развивался, была показана его применимость для реальных современных компьютеров на базе процессоров Intel и AMD, а также для построенных на их основе суперкомпьютеров [2 – 6]. В указанных работах в основном рассматриваются процессоры, применяемые в настольных компьютерах, ноутбуках и суперкомпьютерах, но современные устройства, использующие вычислительные процессоры, не ограничиваются этим перечнем. Вычислительные процессоры применяются в мобильных телефонах, планшетах, цифровых носителях и плейерах, портативных игровых консолях, калькуляторах, навигаторах и многих других устройствах. Важно отметить, что абсолютное большинство перечисленных устройств использует процессоры, основанные на архитектуре ARM.

Архитектура ARM — это семейство лицензируемых 32- и 64-битных микропроцессорных ядер, разработанных компанией ARM Limited. Лицензию на производство процессоров данной архитектуры имеют практически все ведущие производители устройств с вычислительными процессорами: AMD, Apple, Analog Devices, Atmel, Xilinx, Altera, Intel (до 27 июня 2006 г.), NXP, STMicroelectronics, Samsung, LG, MediaTek, MStar, Qualcomm, Sony, Texas Instruments, nVidia, Freescale, HiSilicon и др., а также российские: “Байкал Электроникс”, “Миландр”, “Модуль”, “Ангстрем”, “ЭЛВИС-НеоТек”. Уже в 2009 г. более 90 % встроенных 32-разрядных процессоров приходилось на процессоры архитектуры ARM.

В настоящей работе исследуются процессоры архитектуры ARM и определяется их вычислительная способность при помощи метода, предложенного в работе [1]. Результаты, полученные ранее в работах [2, 3], показали применимость метода к реальным процессорам Intel. Архитектура ARM принципиально отличается от архитектуры Intel, поэтому в работе проводится сравнение значений вычислительной способности процессоров ARM и Intel. Кроме того, полученные результаты сравниваются с аналогичными результатами общепризнанных бенчмарков. На основе этого сравнения делаются выводы о применимости метода к реальным процессорам ARM.

## 1. Основные определения и понятия

В работе [1] описана упрощенная модель компьютера, согласно которой он состоит из набора инструкций процессора $I$ и доступной памяти $M$. Каждая отдельная инструкция процессора — это совокупность имени команды и операндов, таких как индексы регистров и адреса ячеек памяти. То есть под инструкцией подразумевается не только имя команды, но и значение ее операндов, следовательно, инструкции, имеющие одинаковые имена команд, но разные значения операндов, будут входить в $I$ как разные инструкции.

**Определение.** *Пусть есть компьютер с набором инструкций $I$ и $\tau(x)$ — время выполнения инструкции $x \in I$. Тогда вычислительная способность $C(I)$ задается*

*следующим образом:*

$$C(I) = \lim_{T\to\infty} \frac{\log N(T)}{T} \ , \tag{1}$$

*где $N(T)$ — размер множества всех допустимых последовательностей инструкций, время выполнения которых равно $T$.*

**Утверждение.** *Предел* (1) *существует, если конечное множество $I$, время выполнения $\tau(x)$, $x \in I$, — целые числа и наибольший общий делитель всех $\tau(x)$, $x \in I$, равен 1.*

Рассмотрим компьютер с множеством инструкций $I$, время выполнения которых составляет $\tau(x)$, $x \in I$, и все последовательности инструкций являются разрешенными, т. е. набор $I$ рассматривается как алфавит, и все возможные слова над этим алфавитом можно представить как допустимые последовательности инструкций для компьютера. В этом случае оценить вычислительную способность (1) можно при помощи метода, предложенного Шенноном в [7], где показано, что $C(I)$ равна логарифму от наибольшего действительного решения $X_0$ следующего уравнения:

$$X^{-\tau(x_1)} + X^{-\tau(x_2)} + \ \dots \ + X^{-\tau(x_s)} = 1, \tag{2}$$

где $I = \{x_1, \dots, x_s\}$. Другими словами, $C(I) = \log X_0$ .

Более подробно теоретическое обоснование метода дано в работах [1, 5].

## 2. Пример практического применения метода

Рассмотрим описанный выше метод на примере упрощенного процессора ARM. Пусть наш процессор имеет 4 четырехразрядных регистра: три регистра общего назначения (назовем их r0–r2) и регистр r3 для адреса выбираемой из памяти инструкции. Регистры имеют размер 4 бита, инструкции — 1 байт. Набор инструкций содержит команды обработки данных и команды обмена данными. Формат команд обработки данных представлен на рис. 1, формат команд обмена данными — на рис. 2.

Команды обработки данных формируют результат, который помещается в регистр Rd, выполняя указанную в поле “Код операции” арифметическую или логическую операцию с одним или двумя операндами, при этом первый операнд — регистр, указанный

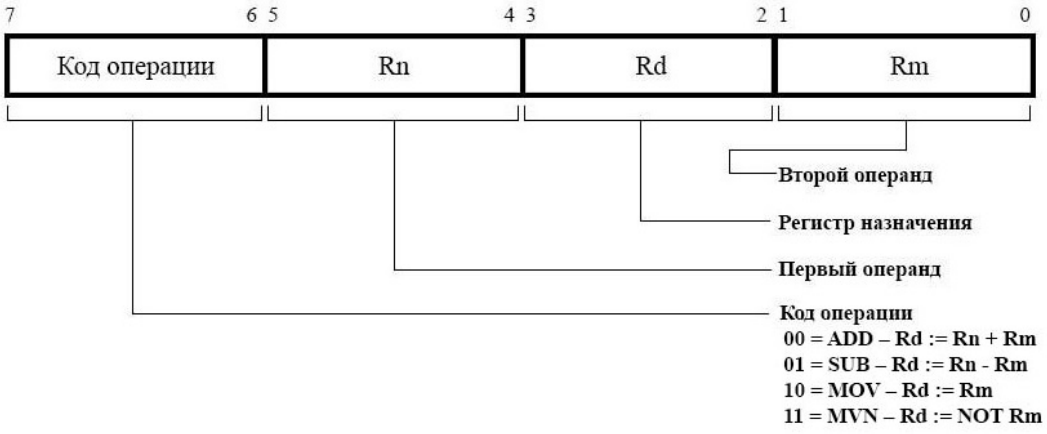


Рис. 1. Формат команд обработки данных

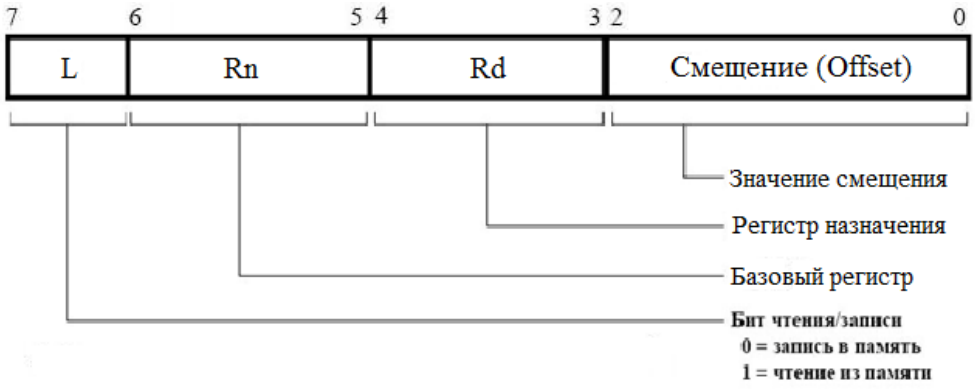


Рис. 2. Формат команд обмена данными

в поле Rn, второй операнд — регистр, указанный в поле Rm. Важно заметить, что при записи результата в регистр r3 время выполнения инструкции увеличивается на два такта, так как происходит сброс конвейера.

Подсчитаем количество инструкций $Q$, которые используют оба операнда и не записывают результат в регистр r3, по следующей формуле:

$$Q = Q_{OpCode} Q_{Rn} Q_{Rd} Q_{Rm},$$

где $Q_i$ — количество вариантов значений для поля $i$.

Так как команд, использующих два операнда, две (команды ADD и SUB), то $Q_{OpCode} = 2$. В качестве регистров Rn и Rm можно использовать все регистры, поэтому $Q_{Rn} = Q_{Rm} = 4$. Результат выполнения операции не записывается в регистр r3, значит, $Q_{Rd} = 3$. Таким образом, $Q = 2 \cdot 4 \cdot 3 \cdot 4 = 96$. Количество остальных инструкций обработки данных подсчитывается аналогично (табл. 1).

Команды обмена данными используются для записи/чтения байта данных между регистром Rd и памятью. Адрес памяти, использующийся при обмене, формируется добавлением содержимого поля “Смещение” к содержимому регистра Rn. Поле “Смещение” содержит трехразрядное беззнаковое значение. Количество инструкций обмена данными (табл. 2) определяется по формуле

$$Q = Q_L Q_{Rn} Q_{Rd} Q_{Offset}.$$

Т а б л и ц а 1. Команды обработки данных

| Тип обработки | Тип команды | Число инструкций | Время выполнения, такт |
|---|---|---|---|
| Обычная обработка данных | Используются 2 операнда | $2 \cdot 4 \cdot 3 \cdot 4 = 96$ | 1 |
| | Используется 1 операнд | $2 \cdot 3 \cdot 4 = 24$ | 1 |
| Обработка данных с записью в регистр PC | Используются 2 операнда | $2 \cdot 4 \cdot 1 \cdot 4 = 32$ | 3 |
| | Используется 1 операнд | $2 \cdot 1 \cdot 4 = 8$ | 3 |

Т а б л и ц а 2. Команды обмена данными

| Инструкция | Число инструкций | Время выполнения, такт |
|---|---|---|
| Запись в память | $4 \cdot 4 \cdot 2^3 = 128$ | 2 |
| Чтение из памяти | $4 \cdot 4 \cdot 2^3 = 128$ | 3 |

Подсчитываем количество инструкций определенного типа, чтобы объединить в одно слагаемое инструкции одного типа с одинаковым временем выполнения. На основе полученных значений построим уравнение (2):

$$\frac{96}{X} + \frac{24}{X} + \frac{32}{X^3} + \frac{8}{X^3} + \frac{128}{X^2} + \frac{128}{x^3} = 1.$$

Решив это уравнение, получим $C(I) \approx \log_2 121.069 \approx 6.9$ бит/такт.

Так как количество слагаемых в уравнениях для реальных процессоров измеряется сотнями и даже тысячами, в рамках исследования потребовалось разработать программу, которая при помощи метода бисекции находит решение уравнения (2).

## 3. Архитектура процессоров ARM

ARM — это архитектура микропроцессоров с набором инструкций RISC, разрабатываемая компанией ARM Limited. Преимущество архитектуры заключается в небольшом наборе простых инструкций, которые обрабатываются с минимальными затратами энергии, что позволяет ARM-процессорам находить широкое применение: недорогие и энергоэффективные чипы используются в мобильных устройствах, сетевом оборудовании и измерительных приборах, во встраиваемых системах.

В данной работе исследуются процессоры:

- LPC2468 — 32-разрядный микроконтроллер, разработанный компанией NXP на базе ядра ARM7TDMI-S (ARM7TDMI-S — 32-разрядный микропроцессор с архитектурой ARMv4T) [8].
- AT91RM9200 — микроконтроллер компании Atmel на базе ядра ARM920T.

Рассмотрим некоторые особенности архитектуры процессоров ARM, необходимые для определения вычислительной способности. (Документация с подробным ее описанием выложена на сайте компании ARM в открытом доступе [9 – 11].)

Конвейер — это механизм, позволяющий обрабатывать несколько (в зависимости от количества ступеней) инструкций одновременно за счет разбиения процесса выполнения инструкции на этапы, что снижает время выполнения инструкций в потоке и увеличивает производительность процессора. В ядре ARM7TDMI-S используется трехступенчатый конвейер, инструкции выполняются в три этапа: выборка, дешифрация и исполнение. В ядре ARM920T к основным трем ступеням добавляются ступень доступа к памяти и ступень записи в регистр. Наличие дополнительных ступеней позволяет некоторым инструкциям процессора выполняться быстрее, чем в ARM7TDMI-S. На каждом этапе инструкции могут выполняться разное время. И если на каком-то этапе инструкция выполняется долго, то на всех предыдущих ступенях уже выполнившиеся инструкции простаивают в ожидании. Поэтому под временем выполнения инструкции в конвейерных системах понимается задержка, которая произошла в конвейере по ее вине.

Еще одна особенность подобных систем — сброс конвейера. В процессорах ARM такая ситуация может произойти при выполнении инструкций перехода. Конвейер сбрасывается, и на его повторное заполнение требуется время, что снижает общую производительность. Однако следует заметить, что, во-первых, ситуация со сбросом конвейера довольно редка, во-вторых, из-за малого числа ступеней конвейера (3 ступени) задержка составляет от одного до трех тактов. Поэтому в данной работе решено пренебречь сбросом конвейера.

Стоит отметить, что помимо основного пользовательского режима работы процессора ARM процессоры имеют пять дополнительных привилегированных режимов, которые используются для обработки исключительных ситуаций. Но в рамках данной работы рассматривается только обычный режим работы.

Процессоры ARM имеют архитектуру "load and store" (загрузка и сохранение), поэтому для выполнения любой операции над данными они из памяти загружаются в определенные регистры, выполняется инструкция обработки данных и затем полученные значения записываются обратно в память. Доступ к данным в памяти могут осуществлять только инструкции чтения, записи и обмена.

Процессоры ARM содержат 31 32-разрядный регистр общего назначения (РОН) и шесть 32-разрядных регистров статуса программы. Из них одновременно пользователю доступны только 16 РОН и один или два регистра статуса программы. Остальные регистры предназначены для привилегированных режимов, с их помощью ускоряется обработка исключений. Регистры R0...R13 используются для хранения данных или адресов, а регистры R14 и R15 имеют дополнительные функции. R15 используется в качестве счетчика инструкций (PC), R14 (LR) содержит адрес возврата после вызова подпрограммы. Текущее состояние программы хранится в регистре CPSR. CPSR содержит: флаги кода условий (4 флага АЛУ) и управляющие разряды: 2 разряда запрета прерываний, разряд, указывающий на используемый набор инструкций (ARM или Thumb), и пять разрядов, кодирующих текущий режим работы процессора. Все пять привилегированных режимов содержат регистр статуса SPRS, в который копируется значение CPRS до начала обработки исключительной ситуации.

## 4. Наборы команд

Процессоры ARM поддерживают два набора инструкций: 32-разрядный набор ARM и 16-разрядный набор Thumb. Рассмотрим набор инструкций ARM, разбив его на пять групп: команды обработки данных, команды умножения, команды ветвления, команды передачи данных и команды передачи блоков данных.

Состав и архитектура сопроцессоров определяются конкретным производителем процессора и не зависят от архитектуры ARM, поэтому инструкции для обращения к сопроцессору не рассматриваются. Не рассматривается и команда программного прерывания, которая переводит процессор в один из привилегированных режимов, так как в данной работе рассматривается только основной режим работы процессора.

Перед выполнением каждой инструкции из набора ARM старшие четыре бита инструкции сравниваются с соответствующими флагами условий в регистре CPSR. Если их значения не совпадают, инструкция не выполняется и проходит через конвейер как NOP (нет операции). Этот случай обрабатывается за счет добавления в список инструкций операции NOP с временем выполнения 1 такт.

Набор инструкций Thumb является выборкой наиболее часто используемых инструкций ARM. Каждая из инструкций Thumb 16-разрядная и имеет соответствующую ей 32-разрядную инструкцию ARM, в которую распаковывается во время выполнения в режиме реального времени. Несмотря на то что инструкции Thumb обеспечивают меньшую производительность по сравнению с инструкциями ARM, благодаря им достигается более высокая плотность кода. Инструкции Thumb не поддерживают условного выполнения (за исключением инструкций условных переходов) и не имеют полного доступа ко всем регистрам регистрового файла. С регистрами R0...R7 (так называемыми

младшими регистрами) могут работать все команды обработки данных, а к регистрам R8. . . R12 (старшим регистрам) могут обращаться только три из них: MOV, ADD, CMP. Отдельно рассматривать в статье этот набор инструкций не имеет смысла, так как процесс их включения в характеристическое уравнение аналогичен инструкциям ARM.

Рассмотрим на нескольких примерах, как происходит подсчет числа инструкций в режиме ARM.

### 4.1. Команды обработки данных

Команды обработки данных (рис. 3) формируют результат, выполняя указанную в поле OpCode арифметическую или логическую операцию на одном или двух операндах, который помещается в регистр Rd.

Первый операнд — это всегда регистр (Rn), а вторым операндом может быть сдвинутый регистр (Rm) или повернутая 8-разрядная константа (Imm) в зависимости от значения разряда $I$ инструкции. Коды условий в регистре CPSR могут оставаться без изменений или могут быть обновлены в результате выполнения команды в соответствии со значением разряда $S$ инструкции. Стоит отметить, что если выставлен разряд $S$ и в качестве Rd используется регистр R15, то регистр SPSR текущего режима копируется в регистр CPSR, поэтому такой вариант инструкции недопустимо применять в пользовательском режиме работы процессора, так как в нем регистр SPSR недоступен.

Когда второй операнд определен как сдвинутый регистр, тип сдвига, который будет выполнен (логический влево или вправо, арифметический вправо или циклический вправо), определяет поле Shift в инструкции. Число разрядов, на которое будет сдвинуто содержимое регистра, содержится либо непосредственно в инструкции (5-разрядное целое), либо в старшем байте другого регистра (кроме R15) (рис. 4). Если же второй операнд определен как константа, то 32-разрядный операнд формируется поворотом вправо значения поля Imm, дополненного нулями до 32 разрядов, на удвоенное значение поля Rotate.

Команды обработки данных можно разбить на три группы:

- команды, использующие оба операнда (AND, EOR, SUB, RSB, ADD, ADC, SBC, RSC, ORR, BIC);
- команды, использующие один операнд (MOV, MVN);
- команды, использующиеся для обновления флагов регистра CPSR (не записывают результат в Rd, $S = 1$) (TST, TEQ, CMP, CMN).

Время выполнения инструкций обработки данных увеличивается при записи результата в регистр PC, а также, если число сдвигов в поле Shift указано в регистре. Таким образом, можно говорить о четырех типах обработки данных:

- обычная обработка данных;
- обработка данных с числом сдвигов, указанным в регистре;
- обработка данных с записью в регистр PC;
- обработка данных с числом сдвигов, указанным в регистре, и с записью в регистр PC.

Учитывая ограничения на операнды, подсчитаем количество инструкций $Q$ для команд, которые используют оба операнда и записывают результат в PC, по следующей формуле:

$$Q = Q_S Q_{Rn} Q_{Rd} (Q_{Op2_0} + Q_{Op2_1}),$$

$$Q_{Op2_0} = (Q_{Sa} + Q_{Sr}) Q_{Rm}, \quad Q_{Op2_1} = Q_{Rotate} Q_{Imm},$$

где $Q_i$ — количество вариантов поля $i$ инструкции.

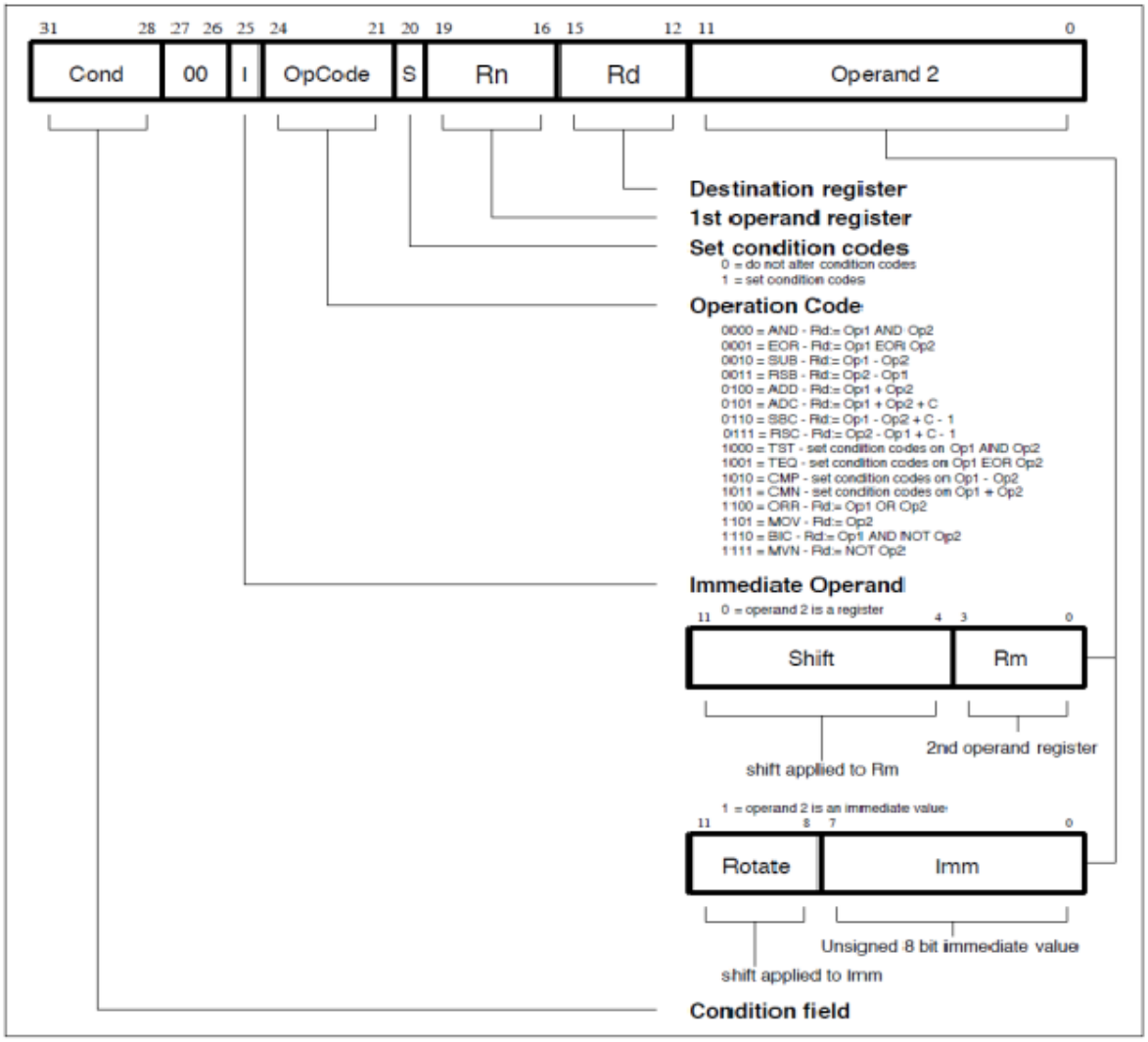


Рис. 3. Команды обработки данных

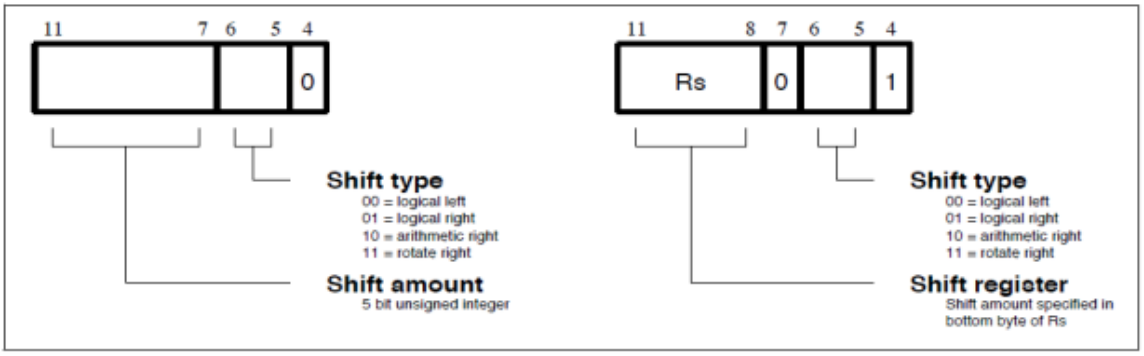


Рис. 4. ARM-операции сдвига

Так как результат работы данных команд записывается в один регистр, $Q_{Rd} = 1$. Сочетание $S = 1$ и Rd = R15 дает некорректные варианты инструкций в непривилегированном режиме, поэтому $Q_S = 1$. В качестве регистров Rn и Rm можно использовать все РОН, значит, $Q_{Rn} = 16$, $Q_{Rm} = 16$. Количество типов сдвига, указывающихся

Т а б л и ц а 3. Команды обработки данных, использующие два операнда

| Тип обработки | Число инструкций | Время выполнения, такт |
|---|---|---|
| Обычная обработка данных | $2 \cdot 16 \cdot 15 \cdot (128 \cdot 16 + 4096) = 2\,949\,120$ | 1 |
| Обработка данных с числом сдвигов, указанным в регистре | $2 \cdot 16 \cdot 15 \cdot 60 \cdot 16 = 460\,800$ | 2 |
| Обработка данных с записью в регистр PC | $1 \cdot 16 \cdot 1 \cdot (128 \cdot 16 + 4096) = 98\,304$ | 3 |
| Обработка данных с числом сдвигов, указанным в регистре, и записью в регистр PC | $1 \cdot 16 \cdot 1 \cdot 60 \cdot 16 = 15\,360$ | 4 |

Т а б л и ц а 4. Команды обработки данных, использующие один операнд

| Тип обработки | Число инструкций | Время выполнения, такт |
|---|---|---|
| Обычная обработка данных | $2 \cdot 15 \cdot (128 \cdot 16 + 4096) = 184\,320$ | 1 |
| Обработка данных с числом сдвигов, указанным в регистре | $2 \cdot 15 \cdot 60 \cdot 16 = 28\,800$ | 2 |
| Обработка данных с записью в регистр PC | $1 \cdot 1 \cdot (128 \cdot 16 + 4096) = 6144$ | 3 |
| Обработка данных с числом сдвигов, указанным в регистре, и записью в регистр PC | $1 \cdot 1 \cdot 60 \cdot 16 = 960$ | 4 |

Т а б л и ц а 5. Команды обработки данных, использующиеся для обновления флагов регистра CPSR

| Тип обработки | Число инструкций | Время выполнения, такт |
|---|---|---|
| Обычная обработка данных | $1 \cdot 16 \cdot (128 \cdot 16 + 4096) = 98\,304$ | 1 |
| Обработка данных с числом сдвигов, указанным в регистре | $1 \cdot 16 \cdot 60 \cdot 16 = 15\,360$ | 2 |

в поле Shift, равно четырем, число целых значений, на которое можно произвести сдвиг, равно $2^5$. Варианты, в которых число сдвигов содержится в регистре для данных команд, не рассматриваются. Таким образом, $Q_{Sa} = 2^5 \cdot 4 = 128$, $Q_{Op2_1} = 2^4 \cdot 2^8 = 4096$. Итого получаем $Q = 1 \cdot 16 \cdot 1 \cdot (128 \cdot 16 + 4096) = 98\,304$.

Аналогично подсчитывается количество инструкций для всех групп команд с учетом типа обработки данных. Результаты подсчетов приведены в табл. 3 – 5.

### 4.2. Команды умножения

Формат команд умножения показан на рис. 5. Команда умножения MUL формирует результат, выполняя умножение содержимого регистров Rm и Rs, который помещается в регистр Rd. Регистр Rn игнорируется. Команда умножения MLA со сложением выполняет умножение содержимого регистров Rm и Rs, прибавляет к полученному значению

содержимое регистра Rn и помещает результат в регистр Rn. Установка флагов CPSR регулируется разрядом $S$.

Ограничения на операнды команд умножения следующие:

- регистр назначения Rd не должен совпадать с регистром операнда Rm;
- R15 не должен использоваться в качестве Rd.

Выполняются длинное умножение и длинное умножение со сложением. Формат команд длинного умножения показан на рис. 6. Команды длинного умножения выполняют умножение двух 32-разрядных операндов и получают 64-разрядный результат. Знаковое или беззнаковое умножение (разряд U), каждое из которых может выполняться со сложением (разряд A), формирует четыре варианта команд длинного умножения.

Команды умножения UMULL и SMULL формируют 64-разрядный результат, выполняя умножение двух 32-разрядных операндов (содержимое регистров Rm и Rs), 32 младших разряда которого помещаются в регистр RdLo, 32 старших — в регистр RdHi.

Команды умножения со сложением UMLAL и SM LAL выполняют умножение содержимого регистров Rm и Rs, прибавляют к полученному значению содержимое регистров RdHi, RdLo и помещают результат в регистры RdHi, RdLo. Младшие разряды прибавляемого значения и результат содержатся в регистре RdLo, старшие — в регистре RdHi.

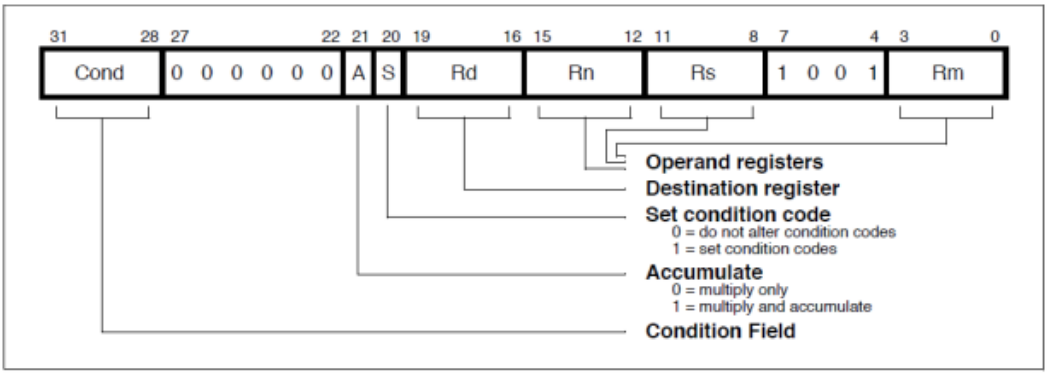


Рис. 5. Команды умножения

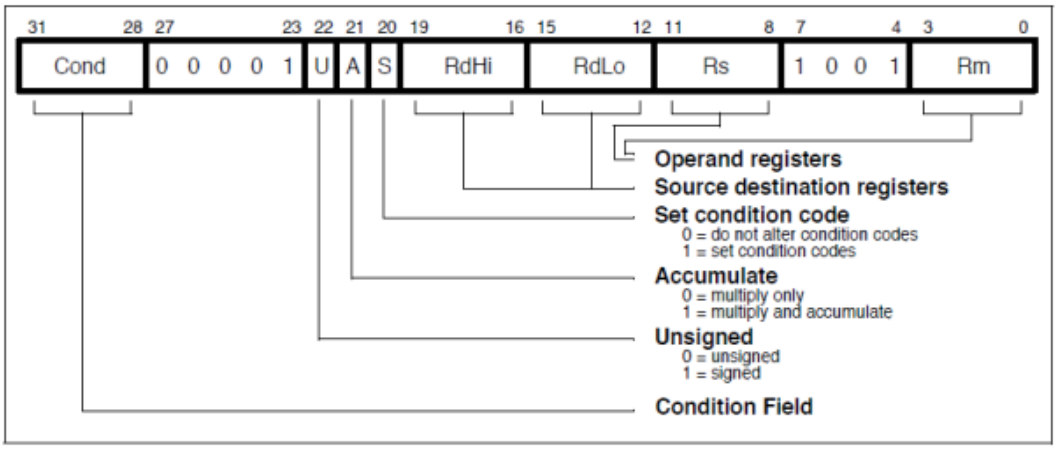


Рис. 6. Команды длинного умножения

Т а б л и ц а 6. Команды умножения

| Команда | Число инструкций | Время выполнения, такт |
|---|---|---|
| $MUL_m$ | $2 \cdot 15 \cdot 1 \cdot 15 \cdot 14 = 6300$ | $m+1$ |
| $MLA_m$ | $2 \cdot 15 \cdot 15 \cdot 15 \cdot 14 = 94\,500$ | $m+1$ |
| $MULL_m$ | $2 \cdot 2 \cdot 15 \cdot 14 \cdot 15 \cdot 13 = 163\,800$ | $m+1$ |
| $MLAL_m$ | $2 \cdot 2 \cdot 15 \cdot 14 \cdot 15 \cdot 13 = 163\,800$ | $m+1$ |

Установка флагов CPSR регулируется разрядом $S$.

Ограничения на операнды команд умножения следующие:

- R15 не должен использоваться в качестве операнда или регистра назначения;
- поля RdHi, RdLo и Rm должны указывать на разные регистры.

Количество тактов, требующихся для выполнения алгоритма умножения, зависит от значения множителя (содержимого регистра Rs): один такт, если разряды [31:8] только нули или только единицы, два такта, если разряды [31:16] множителя только нули или единицы, три такта, если разряды [31:24] множителя только нули или единицы, четыре такта — во всех других случаях. Таким образом, можно говорить о четырех видах команд умножения (назовем $MUL_m$, $MLA_m$ и т. д., где $m = 1 \ldots 4$), каждый из которых определяется значением множителя.

Количество инструкций умножения и количество инструкций умножения со сложением определяются по формуле

$$Q = Q_S Q_{Rd} Q_{Rn} Q_{Rs} Q_{Rm}.$$

Количество инструкций длинного умножения и количество инструкций длинного умножения со сложением определяются по формуле

$$Q = Q_U Q_S Q_{RdHi} Q_{RdLo} Q_{Rs} Q_{Rm}.$$

Результаты подсчетов приведены в табл. 6, где $m = 1 \ldots 4$ — вид команды умножения ($m = 1$, если у регистра Rs разряды [31:8] только нули или только единицы, $m = 2$, если разряды [31:16] только нули или единицы, и т. д.).

### 4.3. Команды для работы с памятью

Команды процессоров ARM работают в основном со встроенными регистрами. Тем не менее обращения к внешней памяти (кэш-памяти, оперативной памяти) все же происходят. Всего в рассматриваемой архитектуре существует не так много команд для работы с памятью:

- команды для записи и чтения одиночных слов/полуслов LDR, STR, LDRH, LDRSB, LDRSH, STRH;
- команды передачи блоков данных LDM и STM (формат этих команд представлен на рис. 7).

Рассмотрим команды передачи блоков данных, которые позволяют загружать данные из памяти одновременно в любое подмножество регистров и соответственно записывать данные из любого подмножества регистров в память. Принцип работы с кэш-памятью аналогичен описанному ранее в работе [2] для процессоров Intel, поэтому рассмотрим подробнее другие особенности этих команд. Список регистров (Register list) —

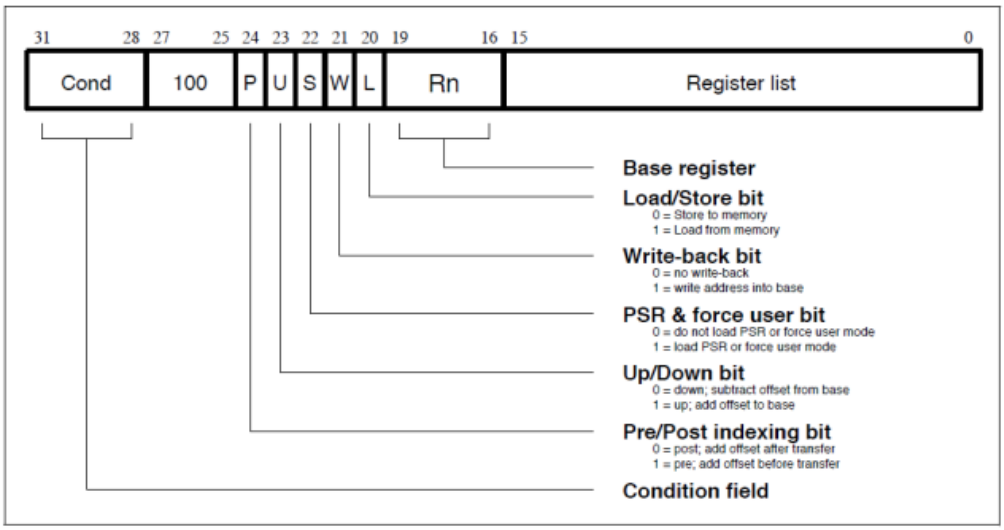


Рис. 7. Формат команд передачи блоков данных

это 16-разрядное поле инструкции, каждый разряд которого соответствует одному из РОН (значения 1 разряда 0 списка инструкций означает, что R0 будет использован, значение 0 — что не будет и т. д.). Адрес передачи определяется содержимым регистра Rn. Разряды P, U, W, L отвечают за специфику исполнения команды (L, например, определяет, будут загружены данные из памяти в регистры или нет). Если инструкция выполняется в непривилегированном режиме, разряд $S$ может быть равен только нулю. Регистр R15 не должен использоваться в качестве базового. Количество инструкций передачи блоков данных подсчитывается по формуле

$$Q = Q_P Q_U Q_W Q_{Rn} Q_{Rl}.$$

Время выполнения инструкций передачи блоков данных увеличивается от количества переданных слов, поэтому разобьем команды на следующие группы: команды записи/чтения одного слова, команды записи/чтения двух слов и т. д. Обозначим количество возможных вариантов списка регистров для группы команд, записывающей/считывающей $i$ слов как $Q_{Rl_i}$. Тогда для команд записи блоков данных количество возможных вариантов списка регистров определяется как

$$Q_{Rl_i} = C_{16}^i = \frac{16!}{i!(16-i)!},$$

где $C_{16}^i$ — биномиальный коэффициент, $i = 1 \ldots 16$. Для команд чтения данных необходимо учесть увеличение количества тактов, требующихся на их выполнение, при записи в регистр R15. Тогда для чтения блоков данных, не записывающих данные в регистр R15, $Q_{Rl_i}$ определяется как

$$Q_{Rl_i} = C_{15}^i = \frac{15!}{i!(15-i)!},$$

где $i = 1 \ldots 15$. Для команд, записывающих данные в регистр R15, $Q_{Rl_i}$ определяется как

$$Q_{Rl_i} = C_{15}^i = \frac{15!}{(i-1)!(15-(i-1))!},$$

где $i = 1 \ldots 16$.

Аналогичный подробный анализ для каждой команды (как ARM, так и Thumb) ядра процессора ARM7TDMI-S представлен в электронном документе [12].

*Процессор LPC2468.* При помощи описанной ранее программы построено характеристическое уравнение (2) для данного процессора. Уравнение имеет вид

$$\frac{30279421}{X}+\frac{20545169}{X^2}+\frac{36159597}{X^3}+\frac{675414}{X^4}+\frac{779197}{X^5}+\frac{690972}{X^6}+\frac{1334760}{X^7}+$$
$$+\frac{2028408}{X^8}+\frac{2480559}{X^9}+\frac{2505426}{X^{10}}+\frac{2162188}{X^{11}}+\frac{1656728}{X^{12}}+\frac{1154401}{X^{13}}+$$
$$+\frac{722400}{X^{14}}+\frac{500760}{X^{15}}+\frac{167520}{X^{16}}+\frac{54840}{X^{17}}+\frac{12600}{X^{18}}+\frac{1800}{X^{19}}+\frac{120}{X^{20}}=1.$$

Решение данного уравнения $X_0 = 30\,279\,421.68$. Таким образом, вычислительная способность $C(I) = 24.85$ бит/такт. Частота процессора LPC2468 72 МГц, следовательно, $C = 1789.33$ Мбит/с.

*Процессор AT91RM9200.* Характеристическое уравнение (2) для данного процессора имеет вид

$$\frac{61856327}{X}+\frac{4920679}{X^2}+\frac{33783795}{X^3}+\frac{816076}{X^4}+\frac{2326261}{X^5}+\frac{1992412}{X^6}+\frac{2485428}{X^7}+$$
$$+\frac{2371296}{X^8}+\frac{2137257}{X^9}+\frac{1681708}{X^{10}}+\frac{1288568}{X^{11}}+\frac{1045201}{X^{12}}+\frac{852000}{X^{13}}+$$
$$+\frac{616800}{X^{14}}+\frac{362400}{X^{15}}+\frac{163920}{X^{16}}+\frac{54600}{X^{17}}+\frac{12600}{X^{18}}+\frac{1800}{X^{19}}+\frac{120}{X^{20}}=1.$$

Решение уравнения $X_0 = 61\,856\,327.08, C(I) = 25.88$ бит/с. Частота процессора 180 МГц. Таким образом, вычислительная способность $C = 4658.84$ Мбит/с.

## 5. Анализ полученных результатов

В табл. 7 приведены технические характеристики процессоров ARM и рассмотренных ранее в работах [2, 3] процессоров Intel. Их производительность оценивается при помощи бенчмарка CoreMark и характеристики “вычислительная способность”. Архитектура процессоров Intel [2, 3] существенно отличается от рассматриваемой в работе архитектуры ARM (принципиально разные подходы к организации вычислений, работе с памятью и выбору списка команд процессора, а также к его физической структуре).

Так как единицы измерения сравниваемых величин различны, будем производить сравнение относительных величин. На рис. 8 представлено попарное сравнение различных процессоров (сравнивается рост характеристики процессора относительно предыдущего в таблице). На графике результаты бенчмарка CoreMark и вычислительной

Т а б л и ц а 7. Характеристики процессоров

| Процессор | Частота, МГц | Вычислительная способность, Мбит/с | CoreMark |
|---|---|---|---|
| NXP LPC2468 | 72 | 1789.33 | 107.14 |
| Intel Pentium | 100 | 2556.00 | 213.83 |
| Atmel AT91RM9200 | 180 | 4658.84 | 292.59 |
| Intel Pentium III | 866 | 36391.35 | 1945.53 |
| Intel Pentium M | 1 300 | 66557.44 | 4213.78 |

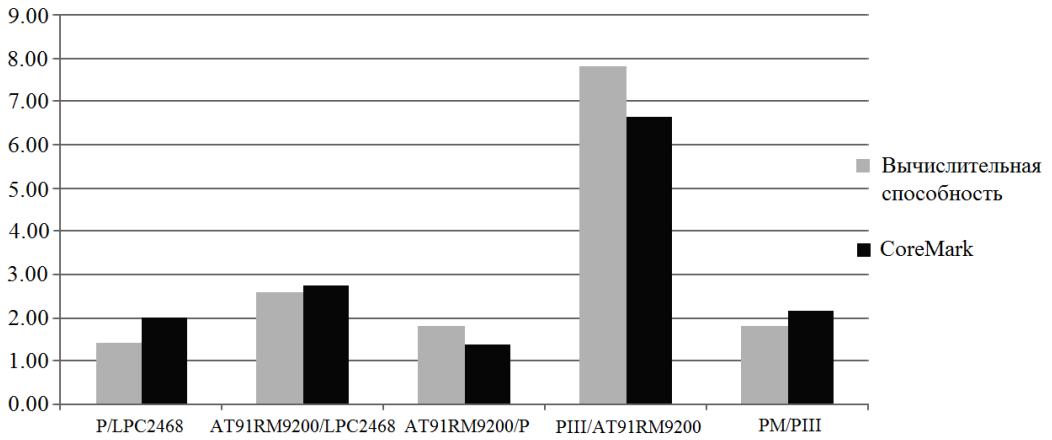


Рис. 8. Попарное сравнение процессоров ARM и Intel

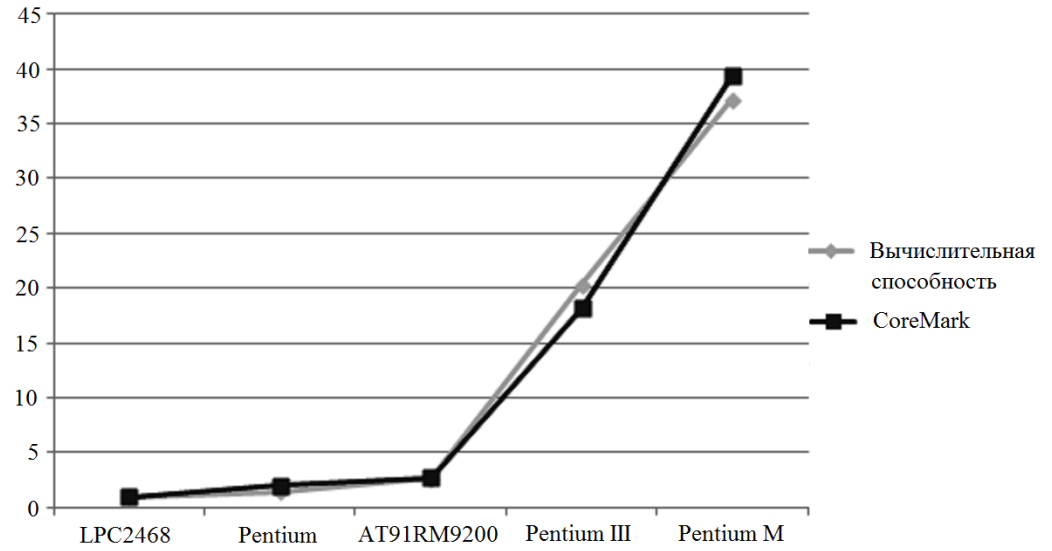


Рис. 9. Сравнительная диаграмма процессоров ARM и Intel

способности показывают схожую тенденцию к росту, но с наличием некоторой погрешности. Это объясняется тем, что бенчмарк оценивает производительность процессора при решении конкретного набора задач, которые его создатели посчитали наиболее важными. Метод оценки вычислительной способности, наоборот, направлен на оценку верхнего порога производительности для всех возможных задач, которые может решать процессор. Из-за этого появляются разногласия между значениями этих характеристик.

Для построения графика на рис. 9 характеристики всех процессоров сравниваются относительно значений соответствующих характеристик процессора NXP LPC2468. Данная диаграмма хорошо отражает общую тенденцию к росту, на ней видно, что бенчмарк и вычислительная способность ведут себя схожим образом.

Исходя из представленных данных можно сделать вывод, что метод оценки вычислительной способности компьютеров применим для оценки производительности устройств на базе процессоров ARM. Полученные результаты однозначно показывают, что метод позволяет сравнивать и оценивать компьютеры с принципиально различной архитектурой (Intel и ARM). Это говорит об универсальности метода и позволяет утверждать, что он применим для оценки производительности любых вычислительных систем.

**Благодарности.** Работа выполнена при финансовой поддержке РФФИ (грант № 15-07-01851).

## Список литературы / References

[1] **Ryabko, B.** An information-theoretic approach to estimate the capacity of processing units // Performance Evaluation. 2012. Vol. 69. P. 267–273.

[2] **Ракитский А.А.** Практическое применение методов теоретической оценки вычислительной способности для процессоров Intel P5 // Вест. СибГУТИ. 2012. № 4. С. 50–61.
**Rakitskiy, A.A.** Practical application of theoretical estimation methods of computational power for Intel processor P5 series // Vestnik SibGUTI. 2012. No. 4. P. 50–61. (In Russ.)

[3] **Ракитский А.А.** Теоретическая оценка вычислительной способности процессоров Intel // Вест. СибГУТИ. 2013. № 3. С. 29–45.
**Rakitskiy, A.A.** Theoretical evaluation of computational power of Intel processors // Vestnik SibGUTI. 2013. No. 3. P. 29–45. (In Russ.)

[4] **Ракитский А.А.** Использование вычислительной способности как характеристики для оценки и сравнения суперкомпьютеров // Вест. СибГУТИ. 2013. № 4. С. 67–84.
**Rakitskiy, A.A.** Application of computer capacity as a characteristic for evaluation and comparison of super-computers // Vestnik SibGUTI. 2013. No. 4. P. 67–82. (In Russ.)

[5] **Ракитский А.А., Рябко Б.Я., Фионов А.Н.** Аналитический метод сравнения и оценки производительности компьютеров и вычислительных систем // Вычисл. технологии. 2014. Т. 19, № 4. С. 84–98.
**Rakitskiy, A.A., Ryabko, B.Ya., Fionov, A.N.** The analytical method for comparing and evaluating the performance of computers and computer systems // Computat. Technologies. 2014. Vol. 19, No. 4. P. 84–98. (In Russ.)

[6] **Ракитский А.А.** Разработка и исследование аналитического метода оценки вычислительной способности компьютеров: Дис. ... канд. техн. наук. Новосибирск: СибГУТИ, 2015. 132 с.
**Rakitskiy, A.A.** Development and research of the analytical method for estimating the computing capacity of computers: Dissertation for degree of candidate of technical sciences. Novosibirsk: SibGUTI, 2015. 132 p. (In Russ.)

[7] **Shannon, C.E.** A mathematical theory of communication // The Bell System Technical J. 1948. Vol. 27 (July). 379–423; Vol. 27 (October) P. 623–656.

[8] **Мартин Т.** Микроконтроллеры ARM7 семейств LPC2300-2400. Вводный курс разработчика. М.: Изд. дом “Додэка-XXI”, 2010.
**Martin, T.** ARM7 microcontrollers of LPC2300-2400 families. Introductory course for developers. Moscow: Izdatel’skiy Dom “Dodeka-XXI”, 2010. (In Russ.)

[9] ARM technical reference manual. Available at: http://infocenter.arm.com (accessed: 09.04.2015).

[10] ARM7TDMI-S data sheet. Available at: http://infocenter.arm.com (accessed: 09.04.2015).

[11] ARM architecture reference manual. Available at: http://infocenter.arm.com (accessed: 17.05.2015).

[12] Список инструкций ARM7TDMI-S для построения характеристического уравнения. Адрес доступа: https://goo.gl/7YtJr2 (дата обращения: 04.11.2015).
ARM7TDMI-S instruction list for building the characteristic equation. Available at: https://goo.gl/7YtJr2 (accessed: 04.11.2015).

*Поступила в редакцию 25 декабря 2015 г.,*
*с доработки — 23 марта 2016 г.*

**The theoretical method for evaluating and comparing the performance of ARM-based processors**

Velichko, Anna A.[1], Rakitskiy, Anton A.[2,*], Ryabko, Boris Ya.[2]

[1]Siberian State University of Telecommunication and Informatics, Novosibirsk, 630102, Russia
[2]Institute of Computational Technologies SB RAS, Novosibirsk, 630090, Russia

*Corresponding author: Rakitskiy, Anton A., e-mail: rakitsky.anton@gmail.com

The main purpose of this paper is to develop a theoretical method for estimating the performance of ARM-based processors. This method is based on the fundamental ideas of information theory and was initially proposed by B. Ryabko in 2012. The main idea of the method is its ability to evaluate the performance by using the "Computer capacity" characteristic. To determine the "Computer capacity" it is necessary to solve the characteristic equation which construction just needs the information of the architecture of investigated processor (instruction list, execution time of instructions, specification of all types of used memory etc.). During the work we investigate the ARM processors based on ARM7TDMI-S and ARM920T cores. The various features of implementation of ARM instructions which are accounted in construction of characteristic equations are described in detail. The results of applying the method to ARM processors are compared with those corresponding results for Intel processors obtained in previous papers and with CoreMark performance evaluation results. The result of applying the method shows the numbers similar to the CoreMark values. In the course of comparison, we show that the method may be used equally with benchmarks and allows us to compare processors with fundamentally different architectures (Intel and ARM), it does not require the presence of a working model of the processor, and needs only the descriptions of its architecture.

*Keywords*: processor, computer capacity, processor performance, ARM.

**Acknowledgements.** The work has been supported by the RFBR (grant No. 15-07-01851).

*Received 25 December 2015,*
*Received in revised form 23 March 2016*