ЗНАКОМЬТЕСЬ, RISC-V

ПЕТР ПОЗДНЯКОВ, Pozdnyakov@actel.ru

В статье рассматриваются предпосылки появления новой процессорной архитектуры RISC–V и процесс создания проекта с использованием IP-ядра процессора RISC–V для ПЛИС новейшего семейства Microsemi PolarFire.

В ноябре 2016 г. корпорация Microsemi представила IP-ядро CoreRISCV_AXI4 процессора с новой архитектурой RISC–V. Инженеры, использующие ПЛИС и СнК компании Microsemi, получили возможность задействовать в проектах процессор с новейшей архитектурой RISC–V в т. ч. для новейшего семейства радиационно-стойких микросхем семейства RTG4. Предлагаемые Microsemi процессорные IP-ядра не зашифрованы, доступны в исходном коде, что существенно облегчает и удешевляет процесс сертификации изделий на их основе в сравнении с лицензированными процессорами. С течением времени количество доступных IP-ядер, в т. ч. с открытым исходным кодом, будет увеличиваться.

Количество предложений процессоров RISC–V в виде заказных микросхем (ASIC) также продолжает расти. В январе 2019 г. Microsemi анонсировала выпуск семейства СнК с четырьмя 64-бит и одним 32-бит процессорами RISC–V в одном корпусе.

Казалось бы, на рынке уже имеется широкий ряд процессорных архитектур, каждая из которых представлена целым набором процессоров различных производителей. Мысленно пробегая ряд существующих процессорных архитектур (x86, 8051, PowerPC, ARM Cortex-M0, ARM Cortex-A9), может показаться, что для каждого существующего и вновь создаваемого приложения уже выпускается процессор с требуемыми характеристиками.

Однако когда дело доходит до выбора процессора под конкретное приложение, оказывается, что дела обстоят не так радужно: одни процессоры недостаточно быстрые, питание других требует слишком большой электрической мощности, третьи предполагают покупку дорогостоящей лицензии для их производства, четвертые плохо документированы, у пятых дорогие инструменты разработки приложений, шестые не обладают необходимым уровнем радиационной стойкости и т. д. В результате, при всем кажущемся многообразии архитектур и их полупроводниковых воплощений разработчику приходится выбирать из пары–тройки процессоров, как правило, жертвуя какими-то ценными качествами создаваемого устройства из-за ограничений выбранного процессора.

Являются ли описанные трудности достаточным основанием для создания новой процессорной архитектуры? Скептики усомнятся: разработкой и производством процессоров в настоящее время занимаются такие солидные компании как Intel, AMD, Atmel, ARM Limited. Компаниями накоплен большой опыт в этом сложном деле, по всему миру собраны лучшие специалисты. Вероятно, этим компаниям удалось создать шедевральные процессоры, чьи характеристики к тому же постоянно улучшались с каждым новым поколением чипов. На это были потрачены миллиарды долларов, полученных от реализации процессоров предшествующих поколений. Можно ли за ограниченное время без такой солидной финансовой и материальной базы создать изделие, сопоставимое по качеству с продукцией перечисленных гигантов индустрии?

Что ж, для начала присмотримся к шедеврам внимательнее. Для примера рассмотрим инструкцию AAA процессора Intel x86. Она должна выполниться после команды процессора ADD, т. е. сложения операндов. Смысл производимых процессором действий при выполнении инструкции представлен во множестве источников, в частности [1–2]. Для удобства восприятия последовательность действий, производимых процессором в ходе выполнения инструкции, часто представляется на языке C (см. рис. 1).

```
if((AL & 0xF) > 9 || AF == 1) {
        AL = AL + 6;
        AH = AH + 1;
        AF = 1;
        CF = 1
}
else {
        AF = 0;
        CF = 0;
}
AL = AL & 0xF;
```

Рис. 1. **Инструкция AAA процессора Intel x86**

На человеческом языке это звучит примерно так: если в результате сложения младшая тетрада регистра AL содержит значение больше 9 и флаг AF равен 1, то необходимо:

1) к содержимому регистра AL прибавить 6;
2) к содержимому AH прибавить 1. Флаги AF и CF установить в 1.

В противном случае сбросить флаги AF и CF в 0, после чего обнулить старшую тетраду AL.

Еще раз оговоримся: описанная последовательность действий и представленный на рисунке 1 код – не программа, написанная уставшим программистом, а инструкция процессора, т. е. элемент базовой системы команд процессора, которой, по все видимости, должна быть присуща некоторая простота и атомарность.

Однако это еще не все. С каждым новым процессором архитектуры x86 Intel добавляла в систему команд все новые команды – в среднем, по одной команде каждые две недели на протяжении нескольких десятилетий. Ситуация с системой команд x86 оказалась столь запутанной, что даже профессионалы не могут точно сказать, сколько именно команд включает ISA процессора. Оценки колеблются в диапазоне от 981 до 3683.

Является ли система команд, включающая такое количество инструкций, шедевром технической мысли? Можно ли несколько упростить систему команд процессора? С технической точки зрения, ответы на эти вопросы давно известны. Однако кроме технической точки зрения имеются еще и экономические соображения. Исторически сложилось так, что компаниям IBM и Intel удалось в нужный момент представить на рынок удачное системное решение – персональный компьютер и таким образом «поймать волну» компьютеризации в мире. После того как большое количество пользователей привыкли к IBM PC, а производители программного обеспечения начали массово писать ПО для процессоров x86, недостатки «отдельных команд» процессорной

архитектуры уже мало кого интересовали. Пока удвоение производительности происходило каждые 1,5 года, а привязанность компьютера к розетке в стене была нормой, никто не думал менять привычный x86 на что-то иное.

Однако все когда-нибудь заканчивается. Продажи ПК на основе x86 падают из года в год (см. рис. 2) [3]. Аналитики уже подвели черту под эрой ПК и объявили о наступлении эры планшетных компьютеров. Для планшетного компьютера, в отличие от персонального, потребление является ключевым параметром, и наличие в системе команд процессора x86 инструкций типа AAA, которые для выполнения требуют наличия на полупроводниковом кристалле лишних тысяч транзисторов, потребляющих драгоценные ватты мощности батарей, привело к тому, что среди устройств новой эры места для процессоров x86 не нашлось. И это при том, что ниша портативных устройств оказалась в сотни раз просторнее, чем ниша персональных компьютеров. Продажи всех портативных устройств с процессором архитектуры ARM на борту подобрались к 16 млрд (планшетные компьютеры и смартфоны на основе ARM-процессоров). Сравните эту цифру с жалкими 365 млн ПК в самый удачный для ПК 2011 г.!

Но, может быть, архитектура RISC процессоров ARM, столь популярная в носимых устройствах, устранила все недостатки x86 и является пределом совершенства? С технической точки зрения, все действительно неплохо – процессоры с архитектурой ARM обладают высокой производительностью при довольно низком потреблении и занимают не очень много места на кристалле. С экономической точки зрения, для крупных производителей электроники и для нас, потребителей, все опять «не слава богу»! Дело в том, что данная процессорная архитектура является проприетарной, т. е. для изготовления процессоров с этой архитектурой производителю необходимо приобрести лицензию у правообладателя. А правообладатель стремится увеличить свою прибыль и держит цену на оптимальном с точки зрения его бизнес-модели уровне, который часто оказывается высоким и даже непреодолимым порогом для новых проектов интернета вещей, требующих недорогого процессора.

Например, небольшая компания разработала для интернета вещей «умную» лампу, выполняющую голосовые команды, или «умную» подушку, анализирующую ритм дыхания и сердцебиения

Рис. 2. **Объем продаж персональных компьютеров с процессорами архитектуры Intel x86**

во время сна [8], и планирует открыть серийное производство этого инновационного продукта. Экономический расчет проекта показал, что если стоимость процессора, применяемого в устройстве, будет меньше некоторой величины, обозначим ее условно в 1 долл., то стартап окажется успешным – изделия разойдутся по миру миллионными тиражами, проект окупится и принесет прибыль. Если же цена процессора окажется больше доллара, проект не сможет начаться.

Стоимость лицензии самого доступного процессора ARM Cortex-M0 с архитектурой ARM IP составляет 40000 долл. Не у каждой молодой инновационной компании найдется такая сумма для запуска в производство первой партии устройств. Конечно, есть коммерческие банки, интернет-сообщества и проекты, позволяющие получить деньги на стартап, но очевидным остается факт, что отсутствие на рынке свободного, не проприетарного процессора тормозит появление новых гаджетов, развитие новых проектов интернета вещей и, таким образом, замедляет технический прогресс в целом. Причем, закрытость архитектуры пагубно сказывается не только на развитии рынка аппаратуры, но и на рынке программного обеспечения и создания интегрированных сред разработки для создания ПО гаджетов. Проприетарность процессорной архитектуры ARM ведет к тому, что приобрести необходимый комплект документации для создания средств разработки ПО могут только очень состоятельные компании. В результате на мировом рынке существует очень небольшое число производителей интегрированных сред разработки. Конкуренция между ними крайне низка. Они создают платные и довольно дорогие инструменты разработки ПО, что также является барьером для развития инноваций.

Эта проблему отчетливо осознают крупные производители коммерческой электроники. Для них проприетарность процессоров оборачивается миллионами невыпущенных устройств и миллиардами долларов недополученной прибыли.

Проблемы с проприетарными ISA[1] испытывают не только компании, производящие коммерческую электронику, но и выпускающие аппаратуру промышленного и даже космического назначения. Предположим, корпорация Microsemi хочет применить процессор ARM в модуле нового космического аппарата. Для этого необходимо повысить стойкость процессора к воздействию космической радиации. Специалисты компании определили, какие узлы и блоки процессора наиболее подвержены негативному воздействию ионизирующих излучений, выяснили, как повысить их стойкость. Для проведения необходимых работ у корпорации имеются необходимые высококвалифицированные инженеры и финансовые средства. Однако, даже обладая всеми необходимыми знаниями и средствами, Microsemi не может ничего сделать. Лицензия ARM не позволяет разработать собственное радиационно-стойкое ядро процессора на основе существующего, а лишь дает право на использование оригинального не стойкого к воздействию радиации ядра.

Таким образом, какую область электроники ни возьми, начиная с бытовой электроники и заканчивая космической, проприетарность ISA тормозит развитие техники. Поэтому, как только в 2010 г. появился коллектив разработ-

[1] ISA (instruction set architecture) – архитектура системы команд.

чиков, поставивший себе цель создать процессор с открытой архитектурой, его поддержали многие мировые гиганты, например Google, IBM, SAMSUNG, Nvidia, NXP, Western Digital и многие другие [11].

Несмотря на то, что в исследовательскую группу вошли создатели RISC-архитектуры, в частности Дэвид Паттерсон, группа изначально не отдавала предпочтение RISC, а начала свою работу с анализа достоинств и недостатков всех имевшихся на тот момент процессорных архитектур, начав с простейшей RISC и завершив анализ VLIW, и обнаружили следующую особенность: архитектура набора команд (ISA) довольно незначительно влияет на производительность процессора. Т.е., если пересчитать производительность процессоров с разными ISA к одному уровню технологического процесса, используемого при производстве чипов, и к уровню потребления, то окажется, что производительность отличается несущественно. А если ISA не влияет на этот важнейший параметр процессора, зачем такое разнообразие ISA? Нельзя ли создать одну архитектуру ISA для всех процессоров, выпускаемых в мире?

Исследования группы показали, что можно и даже нужно! С технической, коммерческой и потребительской точек зрения переход на единую ISA в планетарном масштабе вполне оправдан, поскольку ведет к унификации средств разработки программного обеспечения, повышению конкуренции и качеству программного кода. Если прежде разработка ПО для бортовых компьютеров космических аппаратов или самолетов была уделом узкого круга посвященных, чья исключительность основывалась на знании недочетов конкретных процессорных архитектур, допущенных к использованию в отрасли, а также умении обходить эти недочеты, то при переходе к единой ISA разработка ПО перестает быть делом избранных. Программист, еще вчера разрабатывавший встраиваемое ПО для стиральных машин и кофеварок, сможет быстро переключиться на разработку ПО для смартфона, космического аппарата и бортового компьютера авиалайнера.

Несмотря на кажущийся слишком смелым отказ от множества существующих процессорных архитектур и переход к единой ISA в глобальном масштабе, история показывает, что аналогичные переходы к единым стандартам уже не раз происходили. В качестве примера можно привести историю создания единой системы железных дорог. Первые железные дороги появлялись на промышленных и горнорудных предприятиях. Параметры дорог, например ширина колеи, размеры и даже материал рельс и шпал были адаптированы под условия выпуска, определенные конкретным предприятием. Однако общая цель производителей и потребителей – перемещение грузов на дальние расстояния – при создании сети железных дорог вынудила перейти на единые стандарты, по крайней мере, в рамках государств и их союзов. Еще одним примером создания глобальных стандартов служит создание Международной системы единиц (СИ).

Другой важный вывод, который был сделан командой исследователей при разработке открытой ISA, заключался в том, что поскольку ISA не является основным фактором, определяющим производительность или потребление процессора, то для новой процессорной архитектуры необходимо использовать или придумать архитектуру с минимальным набором команд. Воплощение такой ISA в полупроводнике приведет к уменьшению занимаемого места на кристалле, что удешевит конечное изделие и положительно скажется на электропотреблении.

Третий фундаментальный принцип, который был положен в основу нового ISA, – расширяемость и масштабируемость. Следуя ему, мы получим семейство процессоров, начиная с кристаллов с сокращенным набором 32-бит регистров общего назначения для встраиваемых приложений с очень малым потреблением и относительно низкой производительностью, и заканчивая x64-бит и 128-бит процессорами со средней и высокой вычислительной мощностью. При работе со всеми чипами линейки можно будет использовать единую среду разработки, программатор и единые приемы программирования.

Заметим, что целью организации RISC–V Foundation является не реализация еще одного нового процессора, а создание спецификации архитектуры системы команд, открытой и доступной для использования всеми разработчиками оборудования и ПО в мире. При этом конкретные реализации процессора, к которым относятся и ASIC, и IP-ядра для ПЛИС, созданные на основе спецификации, могут быть как платными, так и бесплатными, и существенно различаться по своим характеристикам. Одни представители семейства будут отличаться невысокой производительностью и очень малым потреблением, другие, напротив, будут обладать высокой вычислительной мощностью и соответствующим потреблением.

Итак, перечислим все плюсы и минусы, которые ожидаются от перехода к единой открытой архитектуре набора команд. Положительными следствиями перехода к единой ISA являются:

1. Появление промышленного стандарта для вычислительных устройств всех типов.
2. Снижение стоимости процессора и возможность сделать его подходящим для все более широкого круга задач и устройств.
3. Повышение уровня конкуренции на рынке аппаратного и программного обеспечения и уменьшение стоимости продукции для конечного потребителя.
4. Увеличение доступности библиотек открытых проектов Open Core Designs, благодаря чему увеличиваются возможности повторного использования кода.
5. Упрощение процесса обучения разработке ПО и аппаратуры.
6. Ускорение поиска и устранения ошибок в коде ПО.
7. Сокращение времени выхода на рынок новой продукции.
8. Увеличение количества инновационной продукции в повседневной жизни.
9. Появление дополнительных трудностей для правительственных организаций разных государств сделать «потайные двери» в коммерческих устройствах.
10. Увеличение «продолжительности жизни» ПО бытовых приборов. Пока для коммерческих устройств на основе RISC–V указывается срок службы ПО ориентировочно 50 лет.

Негативными факторами на текущий момент можно считать неразвитость инфраструктуры разработки проектов RISC–V и связанные с этим трудности перехода на новую архитектуру.

Отчасти этот недостаток уже преодолен. Компания Microsemi предлагает четыре IP-ядра процессора RISC–V и средство разработки проектов встроенного программного обеспечения SoftConsole 6.0. На популярном среди разработчиков сайте демонстрационных проектов GitHub представлена целая коллекция дизайнов для процессора с новой ISA [10]. Коллекция так разрослась, что для проектов RISC–V, выполненных на основе ПЛИС и СнК, компании Microsemi пришлось создать отдельную ветку [11].

Имплементация IP-ядра CORERISCV_AXI4 требует приблизительно 10 тыс. логических элементов 4LUT-DFF, а с необходимой для взаимодействия с внешним миром периферией (coreUART, coreIO) – не менее 12 тыс. Из семейств Igloo2/SmartFusion2 подойдут чипы, начиная с 25 kLE, т. е. с M2GL025/M2S025 и большей логиче-

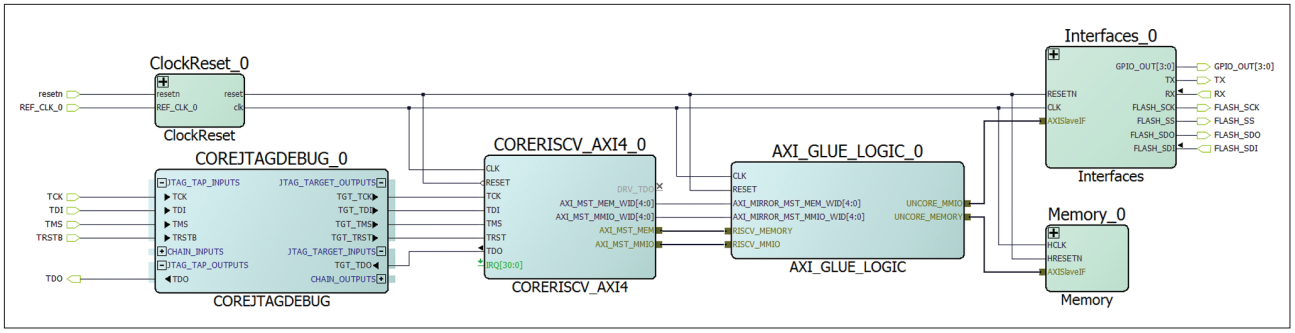

Рис. 3. **Верхний уровень проекта Libero SoC с использованием процессорного ядра coreRISCV_AXI4**

ской емкости. В линейке ПЛИС новейшего семейства PolarFire для реализации ядра процессора можно использовать любой чип. Причем, уже в самом младшем чипе MPF100 можно установить до 10 экземпляров процессорного ядра.

Поскольку проекты с использованием IP-ядер RISC–V содержат многоступенчатое соединение процессорных шин AXI – AHBLight – APB3, они сложны для самостоятельной сборки «с нуля». Гораздо проще и быстрее создать собственный проект, имея в качестве образца готовый работающий. Для первого знакомства с RISC–V можно воспользоваться готовым проектом TU0775: PolarFire FPGA: Building a Mi-V Processor Subsystem Tutorial [12]. Для изучения достаточно открыть проект СнК в среде разработки Libero SoC 12.0, а проект встроенного ПО – в среде SoftConsole 6.0.

Верхний уровень проекта СнК на основе процессорного ядра coreRISCV_AXI4 представлен на рисунке 3. Для улучшения читабельности схемы IP-ядра, отвечающие за получение сигналов сброса и тактирования, обращение к оперативной памяти и интерфейсам, сгруппированы в отдельные блоки (см. рис. 4–6).

Процесс разработки встраиваемого программного обеспечения для нового процессорного ядра практически не отличается от такового для процессора ARM Cortex-M3 СнК SmartFusion2, описанного в статьях, посвященных этому семейству [13–15]. В результате работы нашей программы в окне терминала появляется приветственное сообщение, а светодиоды начинают мигать в соответствии с алгоритмом, реализованном в программе ВПО (см. рис. 7).

Описанные в данной статье проекты СнК и ВПО см. на [16].

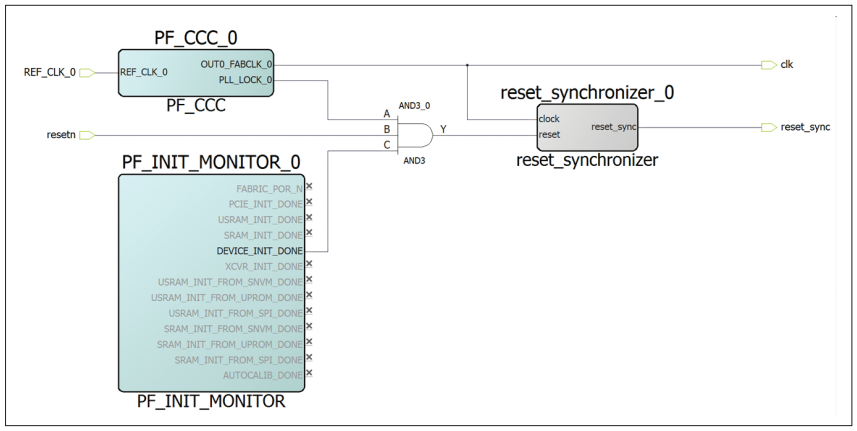

Рис. 4. **Блок синтеза сигналов сброса и тактирования**

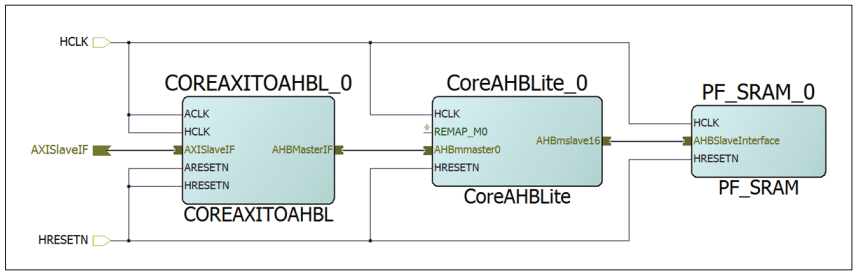

Рис. 5. **Блок обращения к оперативной памяти**

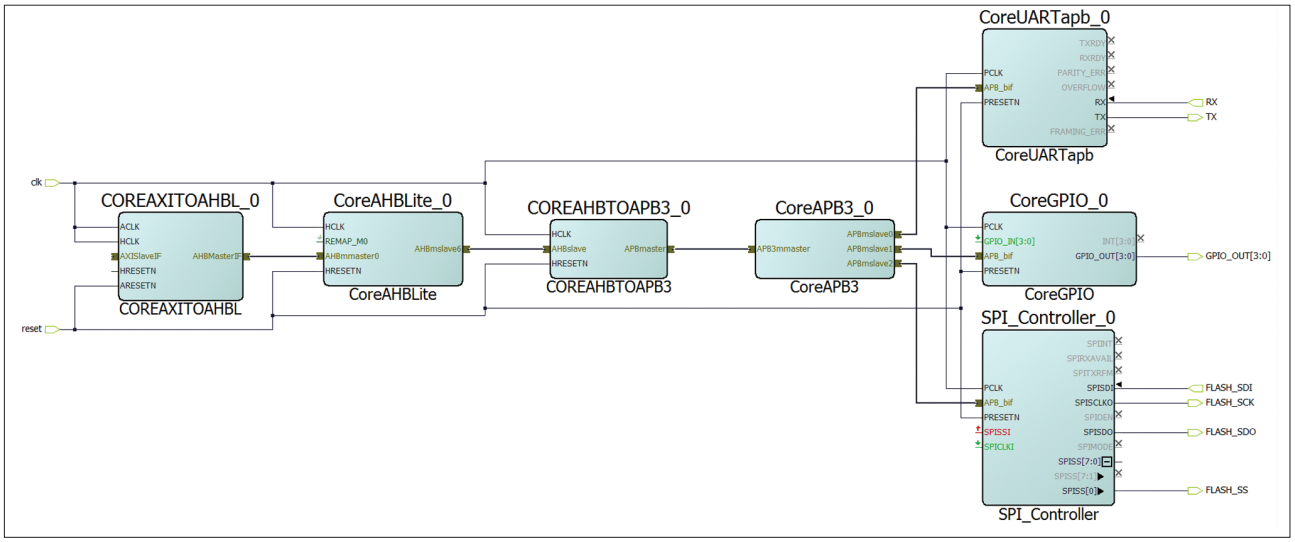

Рис. 6. **Блок интерфейсов**

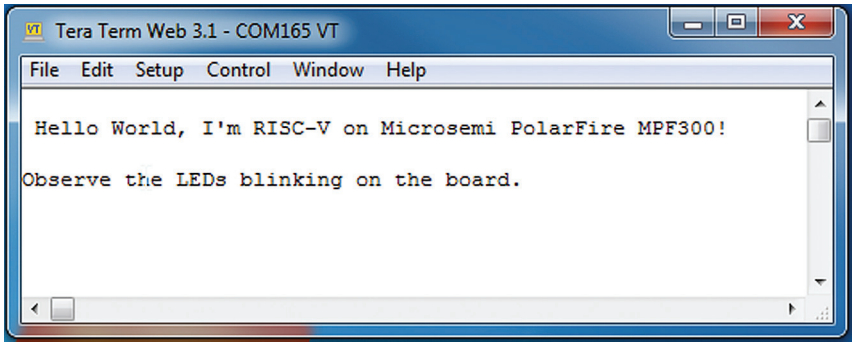

Рис. 7. **Результат работы программы**

ЛИТЕРАТУРА

1. www.club155.ru/x86cmd/AAA.

2. www.fermimn.gov.it/linux/quarta/x86/aaa.htm.

3. ru.wikipedia.org/wiki/Meltdown_ (уязвимость).

4. ru.wikipedia.org/wiki/Spectre_ (уязвимость).

5. ru.wikipedia.org/wiki/Ошибка_Pentium_FDIV.

6. ru.wikipedia.org/wiki/F0_0F_C7_C8.

7. rationalnumbers.ru/tags/mir.

8. www.kickstarter.com/projects/modem/the-sunrise-smart-pillow-sleep-smart-wake-naturall.

9. riscv.org/members-at-a-glance.

10. github.com/riscv.

11. github.com/RISCV-on-Microsemi-FPGA.

12. github.com/RISCV-on-Microsemi-FPGA/M2S090-Security-Eval-Kit.

13. Поздняков П. Разработка приложений для СнК SmartFusion2 с использованием средств разработки Libero SoC и SoftConsole//Компоненты и технологии. 2016. № 1.

14. Поздняков П. Разработка приложений для СнК SmartFusion2 с использованием средств разработки Libero SoC и SoftConsole. Часть 2. Создание исполняемой версии прошивки//Компоненты и технологии. 2016. № 2.

15. Поздняков П. Разработка приложений для СнК SmartFusion2 с использованием средств разработки Libero SoC и SoftConsole. Часть 3. Разработка встраиваемого ПО в среде SoftConsole 4.0//Компоненты и технологии. 2016. № 3.

16. https://drive.google.com/file/d/1AyFr3CRriLitmj81iXd8OdYh2kz5aZxo/view?usp=sharing