Введение в архитектуру ARM.

Рассмотрим архитектуру ARM на примере процессора серии **ARMv7**. Ядро ARM7 является 32-х разрядным RISC процессором с максимальной производительностью до 80 миллионов операций в секунду (MIPS). Основными особенностями его архитектуры являются:

- 32-разрядная линейная адресация в непрерывном адресном пространстве размером 4 Гбайт, обеспечивающая простые механизмы обмена с внешней памятью.
- Наличие семи групп 32-разрядных регистров общего назначения (РОН), из которых выбирается 16 рабочих для текущего режима.
- Наличие одного рабочего и пяти теневых регистров состояния.
- Наличие устройство циклического сдвига
- Наличие аппаратного множителя 32x32 с 64-х разрядным результатом.
- Трехуровневый конвейер (выборка команды, ее декодирование и выполнение).
- Быстрый отклик на прерывания приложений реального времени.
- Поддержка систем виртуальной памяти.
- Простая, но мощная система команд.
- Способность ядра функционировать в режимах: ARM – с 32-разрядными командами и THUMB – с 16-разрядные команды.
- Аппаратная поддержка условного выполнения любой команды
- Поддержка функционирование внешнего сопроцессора.

Основным элементом архитектуры являются регистры процессора. В рабочем наборе программы содержатся 16 32-разрядных регистров общего назначения, наименованных от **R0 до R15**. Формально это равноправные регистры общего назначения. Однако у части из них есть выделенные функции

- R13 - используется в качестве указателя стека (Stack Pointer- SP).

- R14 - называется регистром связи (Link Register - LR). При вызове подпрограммы адрес возврата автоматически запоминается в регистре связи, откуда затем считывается при возврате. Такое решение позволяет быстро переходить к «концевым» функциям (функции, которые не вызывают других функций) и возвращаться из них. Если же функция входит в состав «ветви», т.е. вызывает другие функции, содержимое регистра связи необходимо сохранять в стеке (R13).
- R15 - выполняет функции счетчика команд (PC).

Необходимо отметить, что многие команды могут работать с регистрами R13...R15, как с обычными пользовательскими регистрами. Наряду с данными регистрами в ядре имеется дополнительный 32-битный регистр, который называется регистром текущего состояния программы (Current Program Status Register - CPSR). Регистр CPSRсодержит набор флагов, которые управляют функционированием ARM7 и отображают его состояние:

- Биты 31…28. Четыре старших битах регистра устанавливаются и сбрасываются АЛУ и отражают результат выполнения очередной команды обработки данных. Благодаря им можно узнать, не было ли получено в результате выполнения команды отрицательное или нулевое значение, а также, не произошел ли перенос или переполнение разрядной сетки АЛУ при выполнении команд обработки данных. Анализ состояния этих бит позволяет выполнять условные команды и переходы. Наличие двух флагов C и V позволяет выявлять переполнения АЛУ при выполнении арифметических операций в дополнительном коде. Значения этих бит не может изменятся программно, однако их изменение можно запретить установкой признака в коде команды.
- Биты 27…8 в ядреARM7 не используются.
- Биты 7 и 6 являются флагами обычных (I) и быстрых (F) прерываний. Эти флаги используются для разрешения и запрещения двух линий прерываний, являющихся внешними по отношению к ЦПУ ARM7. Все

периферийные модули микроконтроллеров LPC2000 подключены к этим двум линиям прерываний.
- Бит 5 признаком-указателем текущего набора команд. Ядро ARM7 поддерживает два набора команд - 32-битный набор команд ARM и 16-битный набор команд THUMB. Соответственно флаг T показывает, какой из наборов команд используется. Этот бит не может быть установлен или сброшен прикладной программой. Корректный механизм смены текущего набора команд рассмотрен ниже.
- Биты 4…0 являются флагами режима. В общей сложности ARM7 поддерживает 7 режимов работы, которые могут изменяться либо в зависимости от внутренних или внешних событий, возникающих при выполнении прикладной программы, либо при исполнении различных фрагментов прикладной программы с использованием соответствующих директив.

Биты 7…0 прикладная программа может изменять напрямую.

Команды набора ARM.

32-разрядная система команд ядра ARM7 содержит одиннадцать базовых типов команд:

- Два типа используют встроенное арифметико-логическое устройство, циклическое сдвиговое устройство и умножитель при операциях над данными в банке из 31 регистра, форматом по 32 разряда каждый;
- Три класса команд управления перемещением данных между памятью и регистрами, один оптимизированный на обеспечение гибкости адресации, другой под быстрое контекстное переключение и третий под подкачку данных;
- Три команды управляют потоком и уровнем привилегии выполнения;
- Три типа предназначены для управления внешними сопроцессорами, что позволяет расширить функциональные возможности системы команд за пределами ядра.

Необходимо отметить, что программы, подготовленные даже для довольно эффективной 32-разрядной ARM системы команд, требуют памяти значительного объема, что в свою очередь приводит к росту общей стоимости системы. Специалисты фирмы ARM предложили решение этой проблемы, разработав и внедрив технологию **Thumb**, технологию, позволяющую существенно сократить объем кодов, необходимых для реализации той же программы, что выполняется на 32-разрядной ARM системе команд. До настоящего времени эта технология считается лучшей из технологий, использующих сжатые системы команд. Фактически Thumb выполняет внутреннюю перекодировку из 32-битный команд ARM в 16-битные команды Thumb. Соответственно, не все команды ARM имеют свои аналоги, но в целом код получается более компактный.
Команды выполняются в 3-ёх ступенчатом конвейере

1. загрузка
2. декодирование
3. выполнение

однако для ряда команд 3 шаг конвейера может повторятся несколько раз.

Список базового набора команд ARMv7 :

Мнемоника	Команда	Действие
ADC	Сложение с переносом	Rd := Rn + Op2 + перенос
ADD	Сложение	Rd := Rn + Op2
AND	Логическое И	Rd := Rn AND Op2
B	Переход	R15 := адрес
BIC	Очистить бит	Rd := Rn AND NOT Op2
BL	Переход со ссылкой	R14 := R15, R15 := адрес
BX	Переход и переключение режима ядра	R15 := Rn, T бит := Rn[0]
CDP	Обработать данные сопроцессором	(зависит от типа сопроцессора)

CMN	Сравнить с отрицательным операндом	CPSR флаги := Rn + Op2
CMP	Сравнение	CPSR флаги:= Rn - Op2
EOR	Исключающее ИЛИ	Rd := (Rn AND NOT Op2) OR (op2 AND NOT Rn)
LDC	Загрузить в сопроцессор из памяти	Загрузить в сопроцессор
LDM	Загрузить сразу несколько регистров	Манипуляции со стеком (Pop)
LDR	Загрузить регистр из памяти по указанному адресу	Rd := (адрес)
MCR	Скопировать регистр CPU в регистр сопроцессора	cRn := rRn {<op>cRm}
MLA	Умножение со сложением	Rd := (Rm * Rs) + Rn
MOV	Загрузить в регистр константу	Rd : = Op2
MRC	Скопировать регистр сопроцессора в регистр CPU	Rn := cRn {<op>cRm}
MRS	Переместить регистр статуса/флагов PSR в регистр Rn	Rn := PSR
MSR	Загрузить в PSR статус/флаги указанный регистр	PSR := Rm
MUL	Умножение	Rd := Rm * Rs
MVN	Загрузить регистр отрицательной константой	Rd := 0xFFFFFFFF EOR Op2
ORR	Логическое ИЛИ	Rd := Rn OR Op2
RSB	Обратное вычитание	Rd := Op2 - Rn
RSC	Обратное вычитание с	Rd := Op2 - Rn - 1 +

	переносом	Перенос
SBC	Вычитание с переносом	Rd := Rn - Op2 - 1 + Перенос
STC	Сохранить регистр сопроцессора в памяти	адрес := CRn
STM	Сохранить сразу несколько регистров	Манипуляции со стеком (Push)
STR	Сохранить регистр в памяти	<адрес> := Rd
SUB	Вычитание	Rd := Rn - Op2
SWI	Программное прерывание	Вызывается операционной системой
SWP	Обменять местами содержимое регистра и памяти	Rd := [Rn], [Rn] := Rm
TEQ	Побитовая проверка на равенство	CPSR флаги := Rn EOR Op2
TST	Проверка битов	CPSR флаги := Rn AND Op2