МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение

высшего образования

«УЛЬЯНОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

**Лабораторная работа №7**

по дисциплине: «ОРГАНИЗАЦИЯ ЭВМ И СИСТЕМ».

Вариант 1

Выполнил:

студент 3 курса, гр. ИВТАПбд-31

Казаков К.Г,

Вершинин Д. В.

Проверила:

Лылова А. В.

г. Ульяновск, 2018

**Введение**

Трудно представить себе программу без проверки условий и переходов. Для корректной обработки данных в ассемблере нужна разветвленность программы. С помощью условных переходов в программе реализуются различные управляющие конструкции, ветвления и даже циклы.

В систему команд ассемблера входит обычный набор команд перехода: безусловные и условные переходы, переходы с возвратами и др. Однако в различных архитектурах эти команды имеют некоторые особенности, которые в этой работе и рассматриваются.

Также в этой работе будут рассмотрены флаги, при помощи которых и работают условные переходы. Это является одной из главных особенностей Assembler’a, в отличие от высокоуровневых языков программирования.

1. **Безусловные переходы**
   1. **Безусловные переходы в архитектуре Intel**

В архитектуре Intel Команда безусловного перехода имеет следующий синтаксис:

JMP <операнд>

Операнд указывает адрес перехода. Существует два способа указания этого адреса, соответственно различают прямой и косвенный переходы.

*Прямой переход*: если в команде перехода указывается метка команды, на которую надо перейти, то переход называется прямым.

jmp L

.**..**

.**..**

L: mov eax, x

*Косвенный переход*: При косвенном переходе в команде перехода указывается не адрес перехода, а регистр или ячейка памяти, где этот адрес находится. Содержимое указанного регистра или ячейки памяти рассматривается как абсолютный адрес перехода. Косвенные переходы используются в тех случаях, когда адрес перехода становится известен только во время работы программы.

jmp ebx

* 1. **Безусловные переходы в архитектуре ARM**

Все версии архитектуры ARM поддерживают [команду перехода B](http://ru.osdev.wikia.com/wiki/%D0%98%D0%BD%D1%81%D1%82%D1%80%D1%83%D0%BA%D1%86%D0%B8%D0%B8_B,_BL_(ARM)), обеспечивающую переход в пределах 32 Мбайт вперёд или назад. Кроме того, поскольку счётчик команд PC является одним из [регистров общего назначения](http://ru.osdev.wikia.com/wiki/%D0%A0%D0%B5%D0%B3%D0%B8%D1%81%D1%82%D1%80%D1%8B_%D0%BE%D0%B1%D1%89%D0%B5%D0%B3%D0%BE_%D0%BD%D0%B0%D0%B7%D0%BD%D0%B0%D1%87%D0%B5%D0%BD%D0%B8%D1%8F_(ARM)), переход может быть выполнен записью в него нового значения, дальность перехода при этом не ограничена. Условные и безусловные переходы выполняются одними и теми же командами, поскольку все они включают в свой состав поле [кода условия](http://ru.osdev.wikia.com/wiki/%D0%9A%D0%BE%D0%B4%D1%8B_%D1%83%D1%81%D0%BB%D0%BE%D0%B2%D0%B8%D0%B9_(ARM)).

Для вызова подпрограмм используется специальная разновидность обычной команды перехода B — [инструкция BL](http://ru.osdev.wikia.com/wiki/%D0%98%D0%BD%D1%81%D1%82%D1%80%D1%83%D0%BA%D1%86%D0%B8%D0%B8_B,_BL_(ARM)). Она также осуществляет переход в пределах 32 Мбайт вперёд или назад, но дополнительно заносит адрес возврата в [регистр связи LR (R14)](http://ru.osdev.wikia.com/wiki/%D0%A0%D0%B5%D0%B3%D0%B8%D1%81%D1%82%D1%80%D1%8B_%D0%BE%D0%B1%D1%89%D0%B5%D0%B3%D0%BE_%D0%BD%D0%B0%D0%B7%D0%BD%D0%B0%D1%87%D0%B5%D0%BD%D0%B8%D1%8F_(ARM)). Специальной команды для возврата из подпрограммы не предусмотрено; для этого может использоваться, например, команда пересылки [MOV](http://ru.osdev.wikia.com/wiki/%D0%9A%D0%BE%D0%BC%D0%B0%D0%BD%D0%B4%D1%8B_%D0%BF%D0%B5%D1%80%D0%B5%D1%85%D0%BE%D0%B4%D0%BE%D0%B2_(ARM)) PC, LR.

В вариантах архитектуры, поддерживающих [систему команд Thumb](http://ru.osdev.wikia.com/wiki/%D0%9A%D0%BE%D0%BC%D0%B0%D0%BD%D0%B4%D1%8B_%D0%BF%D0%B5%D1%80%D0%B5%D1%85%D0%BE%D0%B4%D0%BE%D0%B2_(ARM)) (ARMv4T и последующие) появилась [инструкция BX](http://ru.osdev.wikia.com/wiki/%D0%98%D0%BD%D1%81%D1%82%D1%80%D1%83%D0%BA%D1%86%D0%B8%D1%8F_BX_(ARM)), предназначенная для перехода к выполнению кода Thumb. Она копирует в PC содержимое заданного регистра общего назначения и дополнительно устанавливает младший бит PC, что является признаком работы в режиме Thumb. Кроме того, для перехода от кода ARM к коду Thumb может использоваться и прямая загрузка PC значением из памяти или другого регистра.

Для вызова из кода ARM подпрограмм, использующих систему команд Thumb, может использоваться комбинация из команды, заносящей в LR правильный адрес возврата, и собственно [инструкции BX](http://ru.osdev.wikia.com/wiki/%D0%98%D0%BD%D1%81%D1%82%D1%80%D1%83%D0%BA%D1%86%D0%B8%D1%8F_BX_(ARM)). Начиная с процессоров версии ARMv5, появились команды BLX [первого](http://ru.osdev.wikia.com/wiki/%D0%98%D0%BD%D1%81%D1%82%D1%80%D1%83%D0%BA%D1%86%D0%B8%D1%8F_BLX_%D1%82%D0%B8%D0%BF%D0%B0_1_(ARM)) и [второго](http://ru.osdev.wikia.com/wiki/%D0%98%D0%BD%D1%81%D1%82%D1%80%D1%83%D0%BA%D1%86%D0%B8%D1%8F_BLX_%D1%82%D0%B8%D0%BF%D0%B0_2_(ARM)) типов, обеспечивающие занесение в LR адреса возврата и переход к подпрограмме Thumb, причём адрес перехода может либо задаваться смещением, что обеспечит переход в пределах 32 Мбайт вперёд или назад (подобно [команде BL](http://ru.osdev.wikia.com/wiki/%D0%98%D0%BD%D1%81%D1%82%D1%80%D1%83%D0%BA%D1%86%D0%B8%D0%B8_B,_BL_(ARM))), либо находиться в каком-либо регистре общего назначения (подобно [инструкции BX](http://ru.osdev.wikia.com/wiki/%D0%98%D0%BD%D1%81%D1%82%D1%80%D1%83%D0%BA%D1%86%D0%B8%D1%8F_BX_(ARM))).

В процессорах версии ARMv5, не имеющих поддержки Thumb, перечисленные выше инструкции, позволяющие переключиться в режим Thumb, могут всё-таки присутствовать и обеспечивать переход в режим Thumb. В этом случае после переключения произойдёт [прерывание](http://ru.osdev.wikia.com/wiki/%D0%9A%D0%BE%D0%BC%D0%B0%D0%BD%D0%B4%D1%8B_%D0%BF%D0%B5%D1%80%D0%B5%D1%85%D0%BE%D0%B4%D0%BE%D0%B2_(ARM)) по неопределённой инструкции.

В процессорах, начиная с ARMv6, а также в J-версиях ARMv5 присутствует [команда BXJ](http://ru.osdev.wikia.com/wiki/%D0%98%D0%BD%D1%81%D1%82%D1%80%D1%83%D0%BA%D1%86%D0%B8%D1%8F_BXJ_(ARM)), обеспечивающая переход к выполнению кода Java.

Список команд перехода:

[B, BL](http://ru.osdev.wikia.com/wiki/%D0%98%D0%BD%D1%81%D1%82%D1%80%D1%83%D0%BA%D1%86%D0%B8%D0%B8_B,_BL_(ARM)) — переход или переход с возвратом

BLX [первого](http://ru.osdev.wikia.com/wiki/%D0%98%D0%BD%D1%81%D1%82%D1%80%D1%83%D0%BA%D1%86%D0%B8%D1%8F_BLX_%D1%82%D0%B8%D0%BF%D0%B0_1_(ARM)) и [второго](http://ru.osdev.wikia.com/wiki/%D0%98%D0%BD%D1%81%D1%82%D1%80%D1%83%D0%BA%D1%86%D0%B8%D1%8F_BLX_%D1%82%D0%B8%D0%BF%D0%B0_2_(ARM)) типов — переход с возвратом к коду Thumb

[BX](http://ru.osdev.wikia.com/wiki/%D0%98%D0%BD%D1%81%D1%82%D1%80%D1%83%D0%BA%D1%86%D0%B8%D1%8F_BX_(ARM)) — переход к коду Thumb

[BXJ](http://ru.osdev.wikia.com/wiki/%D0%98%D0%BD%D1%81%D1%82%D1%80%D1%83%D0%BA%D1%86%D0%B8%D1%8F_BXJ_(ARM)) — переход к коду Java

1. **Условные переходы**
   1. **Условные переходы в архитектуре Intel**

Условный переход осуществляется, если выполняется определённое условие, заданное флагами процессора (кроме одной команды, которая проверяет CX на равенство нулю). Состояние флагов изменяется после выполнения арифметических, логических и некоторых других команд. Если условие не выполняется, то управление переходит к следующей команде.

Существует много команд для различных условных переходов. Также для некоторых команд есть синонимы (например, JZ и JE — это одно и то же). Для наглядности все команды условных переходов приведены в таблице:

|  |  |  |
| --- | --- | --- |
| **Команда** | **Переход, если** | **Условие перехода** |
| JZ/JE | нуль или равно | ZF=1 |
| JNZ/JNE | не нуль или не равно | ZF=0 |
| JC/JNAE/JB | есть переполнение/не выше и не равно/ниже | CF=1 |
| JNC/JAE/JNB | нет переполнения/выше или равно/не ниже | CF=0 |
| JP | число единичных бит чётное | PF=1 |
| JNP | число единичных бит нечётное | PF=0 |
| JS | знак равен 1 | SF=1 |
| JNS | знак равен 0 | SF=0 |
| JO | есть переполнение | OF=1 |
| JNO | нет переполнения | OF=0 |
| JA/JNBE | выше/не ниже и не равно | CF=0 и ZF=0 |
| JNA/JBE | не выше/ниже или равно | CF=1 или ZF=1 |
| JG/JNLE | больше/не меньше и не равно | ZF=0 и SF=OF |
| JGE/JNL | больше или равно/не меньше | SF=OF |
| JL/JNGE | меньше/не больше и не равно | SF≠OF |
| JLE/JNG | меньше или равно/не больше | ZF=1 или SF≠OF |
| JCXZ | содержимое CX равно нулю | CX=0 |

У всех этих команд один операнд — имя метки для перехода. Некоторые команды применяются для беззнаковых чисел, а другие — для чисел со знаком. Сравнения «выше» и «ниже» относятся к беззнаковым числам, а «больше» и «меньше» — к числам со знаком. Для беззнаковых чисел признаком переполнения будет флаг CF, а соответствующими командами перехода JC и JNC. Для чисел со знаком о переполнении можно судить по состоянию флага OF, поэтому им соответствуют команды перехода JO и JNO. Команды переходов не изменяют значения флагов.

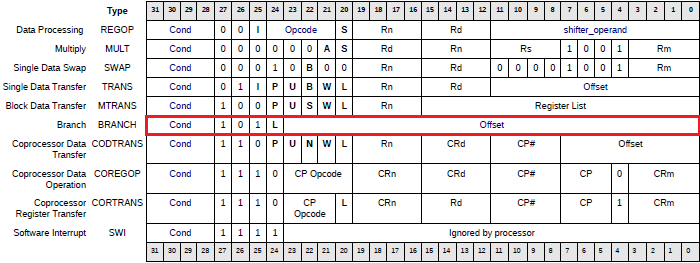
Часто для формирования условий переходов используются команды CMP и TEST. Команда CMP предназначена для сравнения чисел. Она выполняется аналогично команде SUB: из первого операнда вычитается второй, но результат не записывается на место первого операнда, изменяются только значения флагов. Например:

|  |
| --- |
| **cmp** **al**,5 *;Сравнение AL и 5*  **jl** c1 *;Переход, если AL < 5 (числа со знаком)* |
| **cmp** **al**,5 *;Сравнение AL и 5*  **jb** c1 *;Переход, если AL < 5 (числа без знака)* |

Команда TEST работает аналогично команде AND, но также результат не сохраняется, изменяются только флаги. С помощью этой команды можно проверить состояние различных битов операнда. Например:

|  |
| --- |
| **test** **bl**,00000100b *;Проверить состояние 2-го бита BL*  **jz** c2 *;Переход, если 2-й бит равен 0* |

* 1. **Условные переходы в архитектуре ARM**

В процессоре Amber ARM v2a есть всего две команды: b (от слова Branch – ветка, переход) и bl (Branch with Link – переход с сохранением адреса возврата).  
  


Синтаксис команд очень прост: b{cond} label, bl{cond} label.

Понятно, что любые переходы могут быть условными, то есть в программе могут встретиться вот такие, образованные от корней «b» и «bl» и суффиксов условия {Cond}, странные слова: beq, bne, bcs, bhs, bcc, blo, bmi, bpl, bvs, bvc, bhi, bls, bge, bgt, ble, bal и bleq, blne, blcs, blhs, blcc, bllo, blmi, blpl, blvs, blvc, blhi, blls, blge, blgt, blle, blal.

В команде перехода содержится 24-х битное смещение Offset. Адрес перехода вычисляется как сумма текущего значения указателя pc и сдвинутого на 2 бита влево числа Offset, интерпретируемого как знаковое число: New pc = pc + Offset\*4. Таким образом, диапазон переходов составляет 32Мб вперед или назад.

Рассмотрим, что такое переход с сохранением адреса возврата bl. Эта команда используется для вызова подпрограмм. Интересной особенностью этой команды является то, что адрес возврата из процедуры при вызове процедуры сохраняется не в стеке, как у процессоров Интел, а в обычном регистре r14. Тогда для возврата из процедуры не нужна специальная команда ret, как у тех же процессоров Интел, а можно просто скопировать значение r14 назад в pc. Теперь понятно, почему регистр r14 имеет альтернативное название lr (Link Register).

**Заключение**

Мы познакомились с условными и безусловными переходами в архитектурах Intel и Arm. Рассмотрели некоторые особенности, связанные с их применением. Научились строить ветвления на языке Ассемблера.

**Использованные ресурсы**

1. <http://ru.osdev.wikia.com//>- ARM Wiki
2. <http://asmworld.ru/uchebnyj-kurs/016-uslovnye-i-bezuslovnye-perexody/> - Учебный курс по программированию на Ассемблере
3. <https://sites.google.com/site/sistprogr/lekcii1/lek9> - Лекции по программированию на языке Ассемблер
4. <http://asmcourse.cs.msu.ru/wp-content/uploads/2012/02/study7.pdf> - курс "Архитектура ЭВМ и язык ассемблера"
5. <http://www.gaw.ru/html.cgi/txt/doc/micros/arm/asm/arm_thumb/16.htm> - Архитектура и система команд RISC-процессоров системы ARM