ARM-汇编指令集（总结）

王小波的博客 2017-01-04 原文

ARM汇编指令集

指令、伪指令

（汇编）指令： 是机器码的助记符，经过汇编器编译后，由CPU执行。

（汇编）伪指令：用来指导指令执行，是汇编器的产物，最终不会生成机器码。

有两种不同风格的ARM指令

1).ARM官方的ARM汇编风格：指令一般用大写，Windows中的IDE开发环境。

2).GNU风格的ARM汇编：指令一般用小写。

ARM汇编的特点

1. LDR/STR架构

1).ARM采用RISC架构，CPU本身不能直接读取内存，而需要先将内存中内容加载入CPU中通用寄存器中才能被CPU处理。
2).ldr（load register）指令将内存内容加载入通用寄存器。
3).str（store register）指令将寄存器内容存入内存空间中。
4).ldr/str组合用来实现 ARM CPU和内存数据交换。

2. 至此8种寻址方式

1).寄存器寻址mov r1, r2。
2).立即(立即数)寻址 mov r0, #0xFF00。
3).寄存器移位寻址 mov r0, r1, lsl #3。
4).寄存器间接寻址 ldr r1,
[r2] 表示内存，内存地址存在r2这个寄存器中，把内存地址里的值给r1。

5).基址变址寻址ldr r1, [r2, #4]内存地址在r2+4里面。

6).多寄存器寻址 ldmia r1!, {r2-r7, r12}一次访问多个寄存器。
7).堆栈寻址 stmfd sp!, {r2-r7, lr}。
8).相对寻址 beq flag。

3. 指令后缀

同一指令经常附带不同后缀，变成不同的指令。经常使用的后缀有：

B（byte）功能不变，操作长度变为8位

H（half word）功能不变，长度变为16位

S（signed）功能不变，操作数变为有符号

如 ldr ldrb ldrh ldrsb ldrsh

S（S标志）功能不变，影响CPSR标志位

如 mov和movs movs r0, #0

4. 条件执行后缀

条件后缀是否成立取决于当前代码的前面的代码。

条件后缀只影响当前代码的执行。

5. 多级指令流水线

为增加处理器指令流的速度，ARM使用多级流水线.，下图为3级流水线工作原理示意图。（S5PV210使用13级流水线，ARM11为8级）

允许多个操作同时处理，而非顺序执行。

1).PC指向正被取指的指令，而非正在执行的指令

数据传输与跳转指令详解

1. 数据处理指令

数据传输指令
mov mvn

算术指令 add sub rsb adc sbc rsc

逻辑指令 and orr eor bic

比较指令 cmp cmn tst teq

乘法指令 mvl mla umull umlal smull smlal

前导零计数 clz

2. cpsr访问指令

mrs & msr

mrs用来读psr，msr用来写psr

CPSR寄存器比**较特殊**，需要专门的指令访问，这就是**mrs和msr。**

3. 跳转(分支)指令

b & bl & bx

b 直接跳转（就没打开算返回）

bl branch and link，跳转前把返回地址放入lr中，以便返回，以便用于函数调用

bx跳转同时切换到ARM模式，一般**用于异常处理**的跳转。

4. 访存指令

ldr/str &
ldm/stm & swp

单个字/半字/字节访问 ldr/str

多字批量访问 ldm/stm

swp r1, r2, [r0]

swp r1, r1, [r0]

5. 软中断指令

swi（software interrupt）

软中断指令用来实现OS中系统调用

ARM汇编中的立即数

合法立即数与非法立即数

ARM指令都是32位，除了指令标记和操作标记外，本身只能附带很少位数的立即数。因此立即数有合法和非法之分。

合法立即数：经过任意位数的移位后非零部分可以用8位表示的即为合法立即数。

协处理器与协处理器指令集

6.协处理器cp15操作指令

mcr & mrc

mrc用于读取CP15中的寄存器

mcr用于写入CP15中的寄存器

7.arm寻址方式

1). 寄存器： MOV R1,R2 ; R2->R1
2). 立即数： SUBS R0,R1,#1; R0=R1-1
3). 寄存器移位：MOV R0,R2,LSL #3 ;R2左移三位->R0
4). 间接寻址： LDR R1,[R2] ; 装载R2指向的内存数值至R1
5). 基址寻址： LDR R2,[R3,#0x0F] ;R3+0x0F作为地址，将所

指向的置装入R2.R3的值不改变
6). 多寄存器寻址： LDMIA R1!,{R2-R7,R12} ; 将R1所指向的内

存块依次装入{}中的寄存器。
STMIA R0!,{R3-R6,R10} ;将{}列出的寄存器里的值依次填入R0所指向的内存块。
7). 相对寻址： BL XXX ;跳转
BEQ XXX ;条件跳转
协处理器解析：

SoC内部另一处理核心，协助主CPU实现某些功能，被主CPU调用执行一定任务。

ARM设计上支持多达16个协处理器，但是一般SoC只实现其中的CP15.（cp：coprocessor）

协处理器和MMU、cache、TLB等处理有关，功能上和操作系统的虚拟地址映射、cache管理等有关。

MRC & MCR的使用方法

mcr{<cond>} p15, <opcode_1>, <Rd>, <Crn>, <Crm>, {<opcode_2>}

opcode_1：对于cp15永远为0

Rd：ARM的普通寄存器

Crn：cp15的寄存器，合法值是c0～c15

Crm：cp15的寄存器，一般均设为c0

opcode_2：一般省略或为。

ldm/stm与栈的处理

为什么需要多寄存器访问指令

ldr/str每周期只能访问4字节内存，如果需要批量读取、写入内存时太慢，解决方案是stm/ldm

ldm(load register mutiple)

stm（store register mutiple）

举例（uboot start.S 537行）

stmia
sp, {r0 - r12}

将r0存入sp指向的内存处（假设为0x30001000）；然后地址+4（即指向0x30001004），将r1存入该地址；然后地址再+4（指向0x30001008），将r2存入该地址……直到r12内容放入（0x3001030），指令完成。

一个访存周期同时完成13个寄存器的读写

后缀的种类：

ia（increase after）先传输，再地址+4

ib（increase before）先地址+4，再传输

da（decrease after）先传输，再地址-4

db（decrease before）先地址-4，再传输

fd（full decrease）满递减堆栈

ed（empty decrease）空递减堆栈

fa（……）满递增堆栈

ea（……）空递增堆栈

四种栈解析：

空栈：栈指针指向空位，每次存入时可以直接存入然后栈指针移动一格；而取出时需要先移动一格才能取出。

满栈：栈指针指向栈中最后一格数据，每次存入时需要先移动栈指针一格再存入；取出时可以直接取出，然后再移动栈指针。

增栈：栈指针移动时向地址增加的方向移动的栈。

减栈：栈指针移动时向地址减小的方向移动的栈。

!的作用:

ldmia
r0, {r2 - r3}

ldmia r0！, {r2 - r3}

感叹号的作用就是r0的值在ldm过程中发生的增加或者减少最后写回到r0去，也就是说ldm时会改变r0的值。

^的作用:

ldmfd
sp!, {r0 - r6, pc}

ldmfd sp!, {r0 - r6, pc}^

^的作用：在目标寄存器中有pc时，会同时将spsr写入到cpsr，一般用于从异常模式返回。

总结：批量读取或写入内存时要用ldm/stm指令。

各种后缀以理解为主，不需记忆，最常见的是stmia和stmfd。

谨记：操作栈时使用相同的后缀就不会出错，不管是满栈还是空栈、增栈还是减栈。

常用gun伪指令：

global _start
@ 给_start外部链接属性

.section .text @ 指定当前段为代码段

.ascii .byte .short .long .word

.quad .float .string @ 定义数据

.align 4 @ 以4字节对齐

.balignl 16 0xabcdefgh @ 16字节对齐填充

.equ @ 类似于C中宏定义

偶尔会用到的gun伪指令

.end
@标识文件结束

.include @ 头文件包含

.arm / .code32 @声明以下为arm指令

.thumb / .code16 @声明以下为thubm指令

重要的几个伪指令

ldr
大范围的地址加载指令

adr 小范围的地址加载指令

adrl 中等范围的地址加载指令

nop 空操作

ARM中有一个ldr指令，还有一个ldr伪指令

一般都使用ldr伪指令而不用ldr指令

adr与ldr

adr编译时会被1条sub或add指令替代，而ldr编译时会被一条mov指令替代或者文字池方式处理；

adr总是以PC为基准来表示地址，因此指令本身和运行地址有关，可以用来检测程序当前的运行地址在哪里

ldr加载的地址和链接时给定的地址有关，由链接脚本决定。

@笔记记得有点不清晰，希望在学习ARM在座的各位指出哪有错，大家一起进步，共同学习。@

ARM-汇编指令集（总结）的更多相关文章

1. ARM汇编
   ARM汇编 ISA ISA即指指令集架构(Instruction Set Architecture)是与程序设计有关的计算机架构的一部分,包括本地数据类型.指令.寄存器.地址模式.内存架构.中断和意外 ...
2. GNU风格 ARM汇编语法指南

汇编源程序一般用于系统最基本的初始化:初始化堆栈指针.设置页表.操作 ARM的协处理器等.这些初始化工作完成后就可以跳转到C代码main函数中执行. 1. GNU汇编语言语句格式 任何Linux汇编 ...

3. 常用ARM汇编指令
   常用ARM汇编指令 [日期:2012-07-14] 来源:Linux社区 作者:xuyuanfan77 [字体:大 中 小] 在嵌入式开发中,汇编程序常常用于非常关键的地方,比如系统启动时初 ...
4. ARM 汇编的一些规范
   A.5.1 文件格式 ARM 源程序文件(即源文件)为文件格式,可以使用任一文本编辑器编写程序代码. 在一个项目中,至少要有一个汇编源文件或C 程序文件,可以有多个汇编 ...
5. ARM 汇编指令
   ARM汇编程序特点: l 所有运算处理都是发生通用寄存器(一般是R0~R14)的之中.所有存储器空间(如C语言变量的本质就是一个存储器空间上的几个BYTE).的值的处理,都是要传送到通 ...
6. iOS 逆向之ARM汇编
   最近对iOS逆向工程很感兴趣. 目前iOS逆向的书籍有: <Hacking and Securing IOS Applications>, <iOS Hacker's Handboo ...
7. 经常使用ARM汇编指令
   一面学习,一面总结,一面记录. 以下是整理在网上找到的一些资料,简单整理记录一下,方便以后查阅. ARM处理器的指令集能够分为跳转指令.数据处理指令.程序状态寄存器(PSR)处理指令.载入/存储指令. ...
8. 生成ARM汇编
   使用ndk即可生成arm汇编 1.首先写好hello.c 2.编写makefile #ndk根目录 NDK_ROOT=E:\Android\android-ndk-r10b #编译器根目录 TOOLC ...
9. ARM汇编指令调试方法
   学习ARM汇编时,少不了对ARM汇编指令的调试.作为支持多语言的调试器,gdb自然是较好的选择.调试器工作时,一般通过修改代码段的内容构造trap软中断指令,实现程序的暂停和程序执行状态的监控.为了在 ...
10. 3.1 ARM汇编编程概述
    1. 汇编编程 为什么要学习汇编 1). Bootloader初始化 2). Linux kernel 3). 高效 2. ARM汇编分类 1. ARM标准汇编:ARM公司得汇编器适合在Windows ...

随机推荐

1. JavaSript模块化-AMD规范与CMD规范
   JavaScript模块化 在了解AMD,CMD规范前,先来简单地了解下什么是模块化,模块化开发. 模块化是指在解决某一个复杂问题或者一系列的杂糅问题时,依照一种分类的思维把问题进行系统性的分解以之处 ...
2. 安卓奇葩问题之：返回按键监听，使Dialog不消失
   本文出处:http://bbs.9ria.com/thread-204406-1-1.html 在做自动更新的时候,弹出Dialog提示,要求是只能点击更新或者取消更新时Dialog才会消失.但是在这 ...
3. 自制简单实用IoC
   IoC是个好东西,但是为了这个功能而使用类似 Castle 这种大型框架的话,感觉还是不大好 代码是之前写的,一直没详细搞,今天整理了一下,感觉挺实用的. IoC定义接口: using System; ...
4. WebDataGrid设置某行某列的值
   <ig:WebDataGrid ID="grid" OnRowSelectionChanged="grid_RowSelectionChanged" O ...
5. DOM--4 响应用户操作和事件（2）
   自定义事件 //旧的方法 //创建 var event = document.createEvent('Event'); //初始化 event.initEvent('build', true, tr ...
6. javascript问题积累
   今天在写网页时碰到了几个js可以解决的小问题,很好用,很简便 1.鼠标移动到图片上时可更换图片,比如用到给图片加颜色,去颜色. <img src="../img/02.gif" ...
7. 欢快的使用Unity JSON吧
   0x01:前言 Unity 5.3加入了UnityUtility类,意味着Unity终于有了自己原生态的JSON库.Unity主要用来游戏开发,JSON做为游戏开发中最受欢迎的配置文件.在官方没有库支 ...
8. Swift 程序流程控制
   Swift采用类同c语言的流程控制语句,if, for, for-in, while, do-while , switch, break, continue .Swift语言的Switch语句自动 ...
9. 【转】 Linux chmod命令

在Unix和Linux的各种操作系统下,每个文件(文件夹也被看作是文件)都按读.写.运行设定权限.例如我用ls -l命令列文件表时,得到如下输出:-rw-r--r-- 1 apple users 22 ...

10. 一步一步理解GB、GBDT、xgboost

    GBDT和xgboost在竞赛和工业界使用都非常频繁,能有效的应用到分类.回归.排序问题,虽然使用起来不难,但是要能完整的理解还是有一点麻烦的.本文尝试一步一步梳理GB.GBDT.xgboost,它们 ...