# ARM裸奔

## 开发板的介绍

s3c2440

处理器 2440

nor flash 2M

nand flash 256M

RAM 64M

lcd 3.5寸

烧写linux系统顺序先使用J-LINK在windows上烧写boot-loader,再通过linux中的usb和windows的串口配合（v-k-y）依次烧写boot-loader，kernel和镜像。

## 交叉编译

**一、常见交叉工具**

1、交叉编译器

arm-linux-gcc -o output-filename filename

arm-linux-gcc -g -o led.o -c led.c

arm-linux-gcc 是专门用来调试在arm平台下可运行文件交叉编译工具。

2、交叉链接器

arm-linux-ld -Tled.lds -o led.elf led.o

**扩展 连接器脚本**

每一个链接过程都是由链接脚本（一般以lds作为文件的后缀名）控制。链接器脚本主要用于规定如何把输入文件内的section放入输出文件内，并控制输出文件内各部分在程序地址空间内的布局。但你也可以用连接的命令做一些其他的事。

脚本格式

一下脚本将输出文件的text section 定位在0x10000,data section 定位在0x80000000;

SECTIONS

{

.=0X10000

.text ALIGN(4):

{

\*(.text)

}

.=0x8000000

.dataALIGN(4):

{

\*(.data)

}

.bssALIGN(4):

{

\*(.bss)

}

}

3、交叉转换器

arm-linux-objcopy被用来复制一个目标文件的内容到另一个文件中，可用于不用源文件之间的格式转换

例：arm-linux-objcopy -o binary elf\_file bin\_file

4、交叉文件工具

arm-linux-readelf 用来显示elf格式可执行文件的信息

5、交叉反汇编器

arm-linux-objdump 用来显示二进制文件信息，查看反汇编代码

例：arm-linux-objdump -D elf\_file >dis\_file 或

arm-linux-objdump -D -b binary -m arm bin\_file >dis\_file

**二、Makefile 工程管理**

一般来说，无论是C、C++、还是pas,首先要把源文件编译成中间代码文件，在Windows下也就是.obj文件，UNIX下是.o文件，即Object file,这个动作叫做编译（compile）。然后再把大量的Object File合成执行文件，这个动作叫做链接（link）。

编译时，编译器需要的是语法的正确，函数与变量的声明的正确。

链接时，主要是链接函数和全局变量。

1、介绍

make命令执行时，需要一个Makefile文件，以告诉make命令需要怎样的去编译和连接程序。

2、规则

目标（target）：依赖（prerequires）

命令（command）

注意：每个命令行前面必须是一个Tab字符进行缩减。

3、使用变量简化

edit:main.o command.o display.o insert.o editmain.o 或

object=main.o command.o display.o insert.o editmain.o

edit:$(object)

4、在目录中删除文件的规则

.PHONY :clean

clean:-rm edit$(objects)

5、makefile 注释

Makefile只有行注释，和UINX的shell脚本一样，其注释使用‘#’，可以用反斜杠进行转义。

## ARM工作模式

http://www.cnblogs.com/zzx1045917067/archive/2012/11/26/2789736.html

ARM7<ARM9<ARM11<Cortex-A5<Cortex-A8<Cortex-A9

ARM与Cortex-A5都是一个指令架构。

一、 存储格式

大端格式：高字节在低地址，低字节在高地址

小端格式：高字节在高地址，低字节在低地址

指令长度：

ARM微处理器的指令长度是32位，也可以为16位（thumb状态下）。Arm位处理器中支持字节（8位），半字（16位），字（32位）三种数据类型，其中，字需要4字节对齐，半字节需要2字节对齐。

ARM体系的CPU有两种工作状态

1 ARM状态：处理器执行32位的字对齐的ARM指令

2 Thumb状态：处理器执行16位的、半字对齐的Thumb指令

thumb指令集是arm指令集的一个子集，是针对代码密度问题而提出的，它具有16位的代码宽度。thumb不是一个完整的体系结构，不能指望处理器只执行thumb指令集而不支持arm集。

二、 ARM处理器的7种工作模式

用户模式（usr）,快速中断模式（fiq）,外部中断模式（irq）,管理模式（svc）,中止模式（abt），未定义模式（und）,系统模式（sys）。

处理器模式的切换方式：

软件控制进行切换（被动切换）

通过外部中断和异常进行切换（主动切换）

三、ARM寄存器概述

ARM处理器v4及以上版本有37个32位的寄存器，其中31个为通用寄存器，6个为状态寄存器。

R13----栈指针寄存器，用于保存**堆栈**指针

栈区（Stack）--由编译器自动分配释放，存放函数的参数值，局部变量的值等。其操作方式方式类似于数据结构中的栈。

堆区（heap）--一般有程序员分配释放，若程序员不释放，程序结束时可能由OS回收。注意它与数据结构中的堆是两回事，分配方式类似于链表。

R14---程序连接寄存器，当执行BL子程序调用指令时，R14中得到R15的备份，而当发生中断或异常时，R14保存R15的返回值。

R15---程序计数器

快速中断模式有七个备份寄存器R8-R14，这使得进入快速中断模式执行很大部分程序时，甚至不需要保存任何寄存器；其他特权模式都含有两个独立的寄存器副本R13、R14，这样可以令每个模式都拥有自己的堆栈指针和连接寄存器。

当前程序状态寄存器（CPSR）

CPSR中各位意义如下：

T位：1---CPU处于Thumb状态，0---CPU处于ARM状态

I、F(中断禁止位)：1---禁止中断，0---中断使能

四、模式切换

当异常发生，CPU进入相应的异常模式时，以下工作是由CPU自动完成的

1. 在异常模式的R14中保存前一工作模式的下一条即将执行的指令地址
2. 将CPSR的值复制到异常模式的SPSR中；
3. 将CPSR的工作模式设为该异常模式对应的工作模式；
4. 令PC值等于这个异常模式在异常向量表中的地址，即跳转去执行异常向量表中的相应指令；

从异常工作模式退回到之前的工作模式时，需要软件来完成以下工作：

1. 将异常模式的R14减去一个适当的值（4或8）后赋给PC寄存器；
2. 将异常模式SPSR的值赋给CPSR。

## ARM汇编

1. 汇编程序框架

一般框架

.section .data

<初始化的数据>

.section .bss

<为初始化的数据>

.section .text

.global \_start

\_start:

<汇编代码>

简化之后

.text

.global \_start

start:

<汇编代码>

## ARM指令

1. 算术和逻辑指令

mov 目的操作数,源操作数

将源操作数（立即数或者地址上的数据）传送给目的操作数（地址）

mvn 目的操作数,源操作数

将源操作数（立即数或者地址上的数据）取反传送给目的操作数（取反）

sub ax,bx

就是ax中的值减bx中的值 ，把结果放在ax中

add ax,bx

将ax与bx相加，然后把结果放在ax中

and ax,bx

将ax与bx相与，然后把结果放在ax中

bic Rd,Rn,operand2

将Rn的值与操作数opreand2的反码按位逻辑“与”，结果存放在目的寄存器Rd中

1. 比较指令

cmp 操作对象1，操作对象2

计算操作对象1-操作对象2，但不保存结果，只是根据结果修改相应的标志位

tst 操作数1，操作数2

数据处理指令，用于把一个寄存器的内容和另一个寄存器的内容或立即数进行按位的与运算，并根据运算的结果更新CPSR中条件标志位的值

1. 跳转指令

bl是arm汇编中用来调用子程序的指令，它把BL后面的一条指令的地址放在R14寄存器里，R15寄存器（PC当前指针地址）就设置成要跳往的地址。这样在这个子程序返回时，再mov PC,R14 就可以返回到BL后面的地址了。B是直接mov PC，目标地址；而不保存其后的地址到R14。

1. 移位指令

lsl

mov r0,r1,lsl#2 ;将R1中的内容左移两位后传送到R0中

ror

mov r0,r1,ror#2;将r1中的内容循环右移两位后传送到r0中

1. 程序状态字访问指令

mrs 与msr配合使用，作为更新psr的读-修改-写序列的一部分

mrs r0,cpsr ;将cpsr中的内容传送至r0

mrs cpsr, r0 ;将cpsr中的内容传送至r0

1. 存储器访问指令

ldr r0,0x12344563 ;就是把0x12344563这个地址中的值存放在r0中

str 源寄存器，<存储器地址>

str指令用于从源寄存器中将一个32位的字数据传送到存储器中。

例： str r0,[r1],#8 ;将r0中的字数据写入以r1为地址的存储器中，并将新地址r1+8写入r1

str r0,[r1,#8] ;将r0中的字数据写入以r1+8为地址的存储器中

## ARM伪指令

ARM机器码

汇编语言-(汇编器)>机器码-（运行在）>芯片

1. 定义类伪指令

global 定义全局变量，该伪指令的含义是让global过的符号对链接器可见，也就是说，一个函数或者变量，通常情况下只在本文件内有效，当需要在外部引用该文件里的某一个函数或变量时，必须首先将该函数或变量使用global伪指令进行声明。

ascii 该伪指令用于在内存中定义字符串，我们要输出到串口中的字符串“helloworld\n”就是由它定义的。

byte 定义一个字节的变量

word 定义一个字的变量

data 将一个内部RAM的地址赋给指定的符号名

equ 相当于C中的宏定义

align 更新位置计数器的值，使代码对齐到某一边界。

**指令与伪指令的区别**

**指令是在执行阶段发挥作用的，由cpu(Intel、AMD等)来执行。**

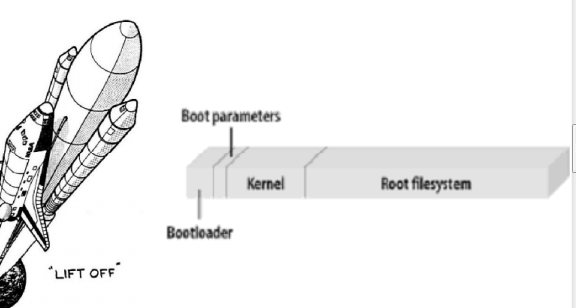
**伪指令是在编译阶段发挥作用的，由汇编器（MASM、TASM等）来解释**

**协处理器访问指令**

协处理器用于执行特定的处理任务。ARM可支持多达16个协处理器，其中CP15是最重要的一个。

## Bootloader设计蓝图

1. bootloader作用



U-Boot是用于多种嵌入式CPU（MIPS X86 ARM等）的bootloader程序，U-boot不仅支持嵌入式linux系统的引导，还支持VxWorks,QNX等多种嵌入式操作。

二、U-boot的设计框架

---- 以2440为例

第一阶程序设计包括核心初始化（设置中断向量表、设置处理器为svc模式、关闭看门狗、关闭所有中断、关闭mmu和cache）C语言编程环境设置（设置堆栈和清除bss段）、led初始化、初始化系统时钟、进行内存初始化、复制nand flash 中的bl到内存中（简单初始化nand flash、复制代码到内存、跳转到第二阶段入口）

第二阶程序设计包括mmu初始化、中断初始化（中断初始化、按键初始化）、初始化串口（串口初始化、移植printf函数）、网卡初始化、LCD初始化（触摸屏初始化、LCD初始化）、解析执行用户命令（移植tftp命令、移植bootm命令）



核心初始化

1. 异常向量表
2. 概念解析

异常定义：因为内部或者外部的一些事件导致处理器停下正在处理的工作，转而去处理这些发生的事件。

1. ARM处理器支持7种类型的异常

用户模式、快速中断模式、外部中断模式、特权模式、数据访问中止模式、未定义指令终止模式、系统模式。

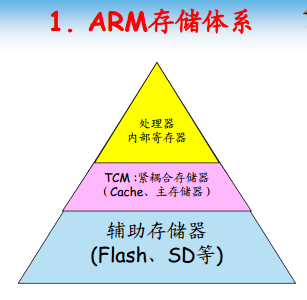
1. 异常向量

当一种异常发生的时候，ARM处理器会跳转到相应异常的固定地址去执行异常处理程序，而这个固定的地址，就称为异常向量。

二、看门狗

watchdog一般是一个硬件模块，其作用就是在系统死机时，帮助系统实现自动重启。watchdog在硬件上实现了计时功能，启动计时后，用户（软件）必须在计时结束后重新开始计时，俗称“喂狗”，如果超时的时候还没有重新开始计时，那么它认为系统是死机了，就自动重启系统。

三、mmu与cache



cache是一种容量小但存取速度非常快的存储器，它保存在近用到的存储器中的数据的拷贝。

对于程序员来说，Cache是透明的。它自动决定保存那些数据，覆盖那些数据。按照功能划分：

I-Cache：指令cache，用于存放指令

D-Cace：数据cache，用于存放数据

mmu是Memory Management Unit的缩写，中文名是内存管理单元，他是中央处理器中用来管理虚拟存储器、物理存储器的控制线路，同时也负责虚拟地址映射为物理地址，以及提供硬件机制的内存访问授权，多用户多进程系统。

## 时钟初始化

1. 时钟脉冲信号定义

按一定的电压幅度 ，一定的时间间隔连续发出的脉冲信号 。时钟脉冲信号是时序逻辑的基础 ，它用于决定逻辑单元中的状态何时更新 。数字芯片中众多的晶体管都工作在开关状态 ，它们的导通和关断动作无 不是按照时钟信号的节奏进行的。

1. 时钟脉冲信号频率

在单位时间（如一秒）内产生的时钟脉冲个数。

1. 信号的产生
2. 晶振
3. 锁相环PLL，PLL合成器是一种更为复杂的系统时钟源。通常PLL合成器需要一个外部晶体并包含一个能够对晶体的特定频率加倍或者分频的集成锁相环（PLL）电路。

4、2440时钟体系

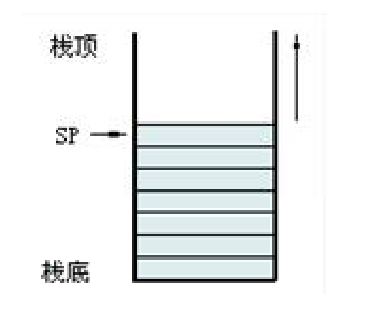
S3C 2440可以使用外部晶振（XTIpll）（默认为12MHZ）和外部时钟（EXTCLK）两种方式输入时钟信号。它由跳线OM[3：2]决定。S3C2440默认主频为12MHZ。

S3C2440使用的内存为SDRAM。

## C语言环境初始化

C语言所需关键点就是堆与栈。

栈是一种具有先进后出的数据组织方式。栈低是第一个进站栈的数据所处的位置，栈顶是最后一个进栈的数据所处的位置。如下图所示。



根据SP指针所指向的位置，栈可以分为满栈和空栈。

1. 满栈：当堆栈指针SP总是指向最后压入堆栈的数据。
2. 空栈：当堆栈指针SP总是指向下一个将要放入数据的空位置。
3. ARM采用满栈。

栈作用:保存局部变量、参数传递、保存寄存器值。

**内存分配中的堆与栈**

一般情况下程序放在Rom或Flash中，运行时需要拷贝到内存中执行。内存中的栈区处于相对较高的地址并以地址的增长方向为上的话，栈地址是向下增长的。栈中分配局部变量空间，堆区是向上增长的用于分配程序员申请的内存空间。另外还有静态区分配静态变量，全局变量空间的；只读区是分配常量和程序代码空间的；以及其他的一些分区。

来看一个网上很流行的经典例子

main.cpp

int a=0;//全局初始化区

char \*p1;//全局未初始化区

main()

{

int b;//栈

char s[]=”abc”;//栈

char\*p2;//栈

char \*p3=“1234567”；//1234567\0在常量区,p3在栈上

static int c=0;//全局（静态）初始化区

p1 = (char\*)malloc(10);//堆

p2 = (char\*)malloc(20);//堆

}

**堆与栈有以下几点不用**

1、申请方式与回收方式不同

栈区（Stack）--由编译器自动分配释放，存放函数的参数值，局部变量的值等。其操作方式方式类似于数据结构中的栈。

堆区（heap）--一般有程序员分配释放，若程序员不释放，程序结束时可能由OS回收。注意它与数据结构中的堆是两回事，分配方式类似于链表。

2、申请后系统的响应

栈：只要栈的剩余空间大于所申请空间，系统将为程序提供内存，否则将报异常提示栈溢出。

堆：首先应该知道操作系统有一个记录空闲内存地址的链表，当系统受到程序的申请时，会遍历该链表，寻找第一个空间大于所申请空间的堆。

3、申请大小的限制

在windows下，栈是向低地址扩展的数据结构，是一块连续的内存的区域。这句话的意思是栈顶地址和栈的最大容量是系统预先规定好的，在windows下，栈的大小是2M（也有人说是1M）。由于栈的大小有限，所以子函数还是有物理意义的，而不仅仅是逻辑意义。

## C与汇编混合编程

为什么需要混合编程

1. 执行效率
2. 能够更直接的控制处理器

C与汇编混合编程有三种方式

1. 汇编调用C函数
2. C调用汇编函数
3. C内嵌汇编

C内嵌汇编—格式

\_\_asm\_\_(

汇编语句部分

：输出部分

：输入部分

：破坏描述部分

);

例：

void write\_p15\_c1 (unsigned long value)

{

\_\_asm\_\_ volatile(

“mcr p15, 0, %0, c1, c0, 0\n”

:

:

: “r” (value)

: "memory");

}

使用volatile来告诉编译器不要对接下来这部分代码进行优化。