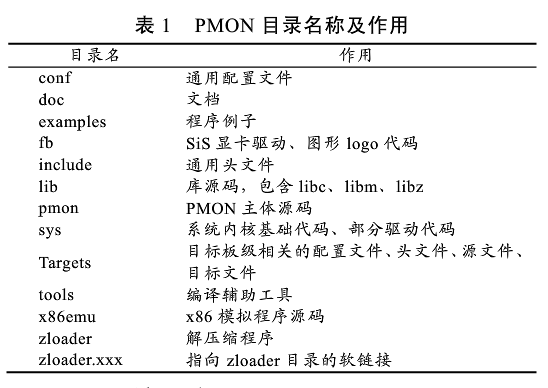
# PMON详解

## 一、PMON的目录结构

PMON的目录结构非常清晰，如下表1所示。其中tools目录下有很多工具，我们只需要用到里面的pmoncfg，该程序是用来根据配置文件为PMON主程序的生成做准备工作。

最后一行的“zloader.xxx”是指存在于源码根目录下的软链接，比如“zloader.ls1a”。这些软链接都指向同一个目录zloader，编译时会根据路径名使用不同的makefile。



## 二、PMON的配置和编译

### 2.1 PMON的配置

总的配置文件： ./1A-PMON-BSP/conf/files

开发板的配置文件： ./ 1A-PMON-BSP/Targets/Bonito/conf/Bonito

其他的配置文件还有： ./ 1A-PMON-BSP/sys/dev/pci/files.pci

./ 1A-PMON-BSP/ sys/dev/ata/files.ata

其中最重要的配置文件是：./ 1A-PMON-BSP/Targets/Bonito/conf/Bonito，它主要是对串口的波特率，文件系统，显示器和键盘模拟终端，自动引导内核，实时钟，环境变量支持，软驱，网卡驱动，硬盘驱动，光驱驱动等进行设置。

### 2.2 PMON的编译

cd ./1A-PMON-BSP/zloader(我们是切换在zloader.ls1a下，它是zloader的一个软链接)

make cfg 根据配置文件重新生成makefile

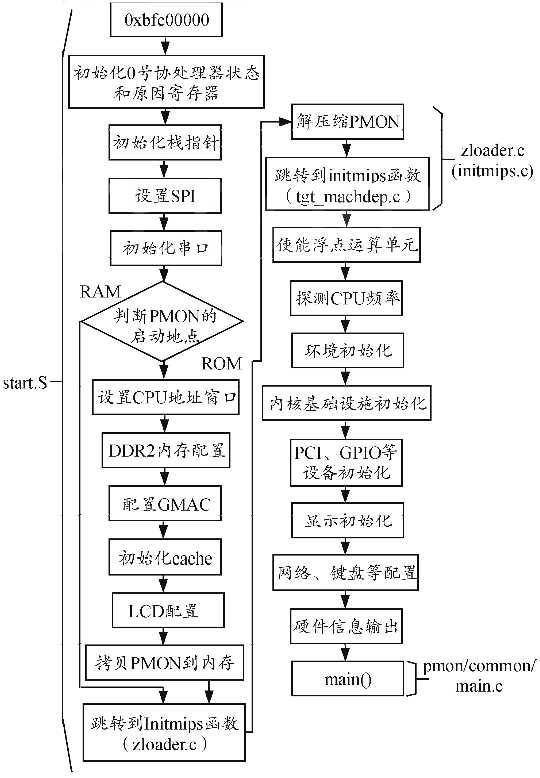
make tgt=rom 生成rom bin文件gzrom.bin

make tgt=ram 生成网络加载文件gzram，通过tftp重新加载pmon调试pmon程序的时候很有用，可以直接放入内存执行

我们采用的编译方式是：make cfg all tgt=rom

## 三、PMON代码分析

按照龙芯开发板的架构要求，0x10000000~0xbfffffff(kseg1段)是唯一在系统重启时能正常工作的内存映射空间，并且地址0xbfc00000是重新启动时的入口变量。首先靠start.s所包含的源代码来处理初始化工作，然后跳到C代码的入口继续执行，如下图所示：



### 3.1 start.s文件

MIPS cpu约定cpu执行的第一条指令位于虚拟地址0xbfc00000处（物理地址的0x1fc00000），而pmon的二进制代码是以load –r –f 0xbfc00000 gzrom.bin这个命令烧入0xbfc00000这个地址开始的flash中，start.s这段代码放在地址0xbfc00000处（也就是gzrom.bin的开头）。 它的核心作用是把pmon的二进制文件复制到内存，并初始化cache，内存控制器，内存和南桥的部分信号。这个代码执行之后会执行C代码，解压在二进制中压缩的bin文件，跳到解压后的代码继续执行。具体分为：

1. 初始化cpu内的寄存器，清TLB
2. 初始化北桥中的cpu接口寄存器
3. 初始化串行口IO用于诊断输出
4. 初始化SDRAM
5. 初始化cache
6. 拷贝pmon的代码到内存
7. 调用initmips函数进入内存执行

C:\Users\hj014\Desktop\initmips.bmp

如上图所示，start.s最后的任务是跳转到initmips函数中执行，这个函数在./1A-PMON-BSP/zloader.ls1a/initmips.c中，initmips.c是在编译过程中产生的，在编译的最后阶段会看到：./genrom ../Targets/LS1X/compile/ls1a/pmon > initmips.c这条语句。

### 3.2 从汇编到C

**Initmips函数有三个参数，分别代表：内存大小，tgt\_putchar函数在rom中的地址，stringserial函数在rom中的地址。enable\_cache函数的功能就是把config寄存器低三位设为011，然后调用run\_unzip解压boisdata数组里的文件，接着调用flush\_cache2函数把二级缓存置为无效，最后把msize作为参数进入realinitmips函数，这个函数在Targets/Bonito/Bonito/tgt\_machdep.c中**



### 3.3 initmips

initmips函数位于Targets/Bonito/Bonito/tgt\_machdep.c中，这个函数是PMON的主要初始化函数，初始化的最后阶段是进入main函数，循环读取命令行输入的数据。下面主要介绍initmips函数的功能和执行流程：

1. 首先执行的是tgt\_cpufreq();函数，实际上是执行\_probe\_frequencies();函数用于探测cpu的频率；
2. 然后执行dbginit(NULL);函数，这个函数的调用层次很深，几乎大部分工作都是在这个函数里面完成的，在pmon/common/main.c中，下面对dbginit函数进行详细介绍：

a 首先调用的是\_init函数，这个函数就是执行所有包含constructor关键字属性的函数，函数中只要声明了\_\_attribute\_\_((constructor))，这个函数的地址就会被放在构造列表中，其完成的功能主要包括：命令初始化，文件系统初始化，可执行文件的初始化，如：初始化load,g命令，初始化ext2,fat文件系统，bin格式文件的初始化等；

b 然后调用envinit ();函数，完成环境变量的初始化；

c 接着执行tgt\_devinit函数，主要完成了以下两个工作：

一是开启cache，进行cache初始化配置，并检查内存是否有错，检查方法为逐字写入并读出比较；

二是通过调用\_pci\_businit函数驱动pci总线和设备，其中\_pci\_hwinit函数是初始化pci总线数据结构和地址寄存器，\_pci\_scan\_dev和\_setup\_pcibuses函数初始化pci设备；

d 接下来执行init\_net (1);函数。

1）执行paraminit初始化系统要用的全局参数；

2）执行vminit函数取一块长度为1M的虚存空间，并记住它的基址；

3）执行kmeminit为kernel内存中的每页建立数据结构并把他们存储在kernel内存中，同时确立kernel内存的基址，偏移量，上限等变量；

4）初始化callouts，在内存中为以后会出现的函数调用建立数据结构链表，它的域包括函数名，参数，执行最大时间，next指针；

5）初始化mbuf，并用mbinit更新中断优先级，分配一个页并插入freelist中；

6）初始化网络设备和网络协议；

7）调用init\_proc初始化进程表；最后根据程序要求看是否需要配置默认的以太网接口；

e 再执行hisinit，初始化命令的历史记录，只有一句话，histon=1；表示现在历史列表中有一个命令；

f 接着执行syminit，建立符号表；

g 调用ioctl(STDIN, TCGETA, &consterm);建立控制终端初始状态；

h 最后输出本程序和机器相关的版本和其他信息；

1. 最后进入pmon/common/main.c里面的main函数。首先他会取环境变量update\_usb的值，经测试，它的值是”no”；然后再取环境变量al的值，如果取得了值就自动加载内核，没有就进入while循环读取用户输入的命令并执行。