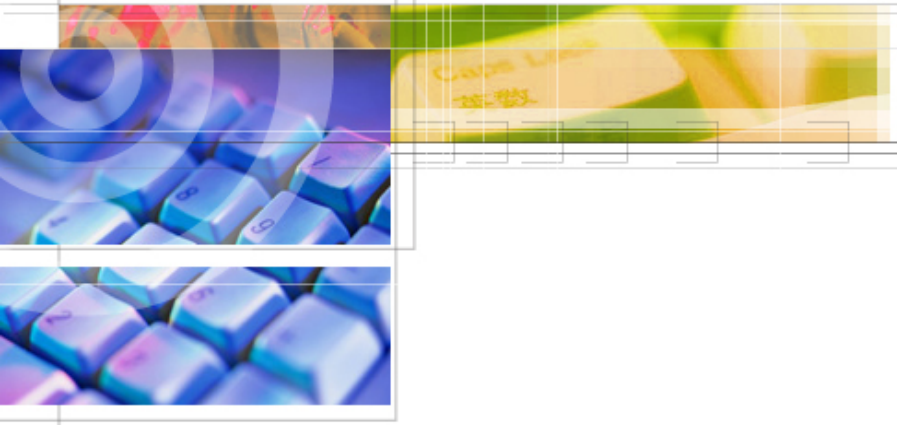


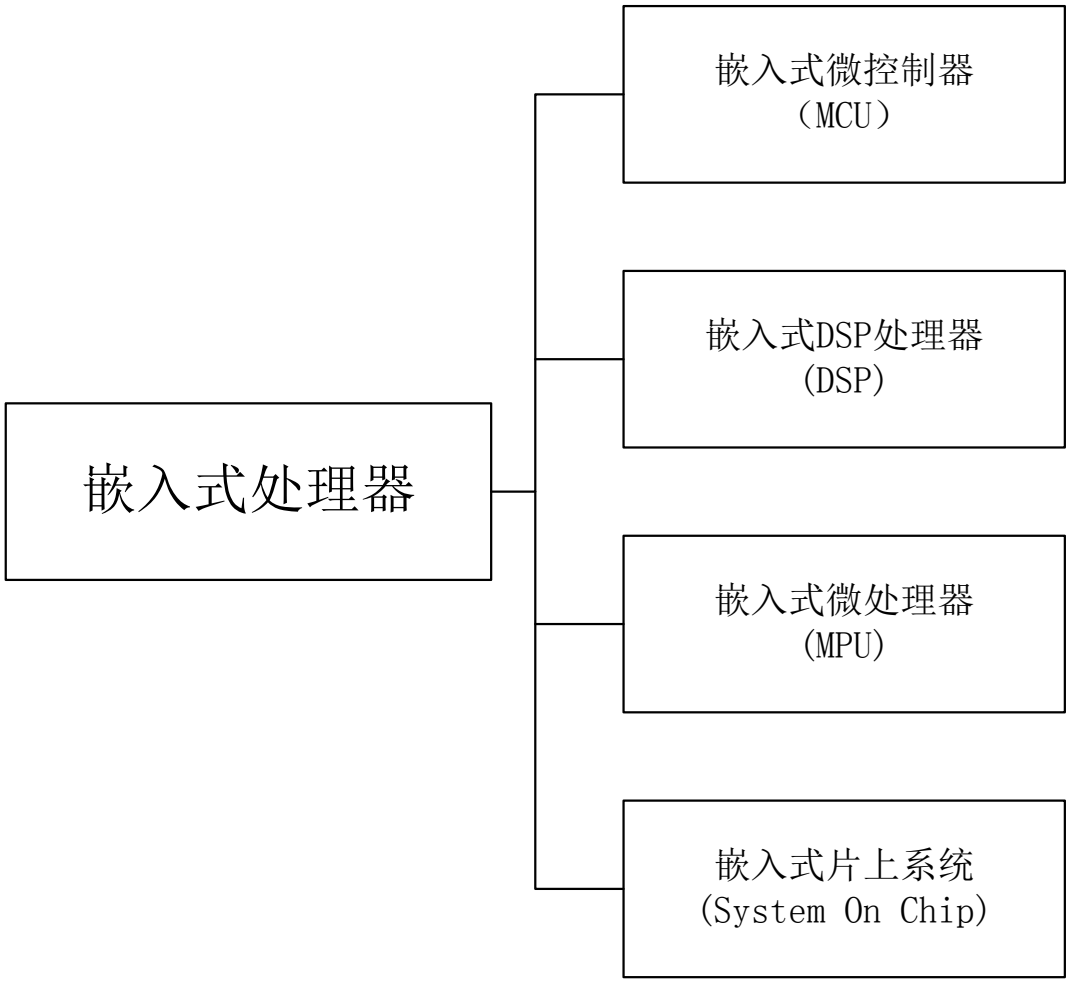
第二讲 嵌入式Linux开发环境构建

华中科技大学电信学院
鄢舒
E-mail: yan0shu@gmail.com

内容提纲(1/6)

- 1. ARM处理器介绍
- 2. 三星S3C2410X(ARM9)芯片简介
- 3. GNU交叉工具链
- 4. 嵌入式Linux开发环境的设置
- 5. 嵌入式应用程序开发示例
- 6. 实验内容与要求

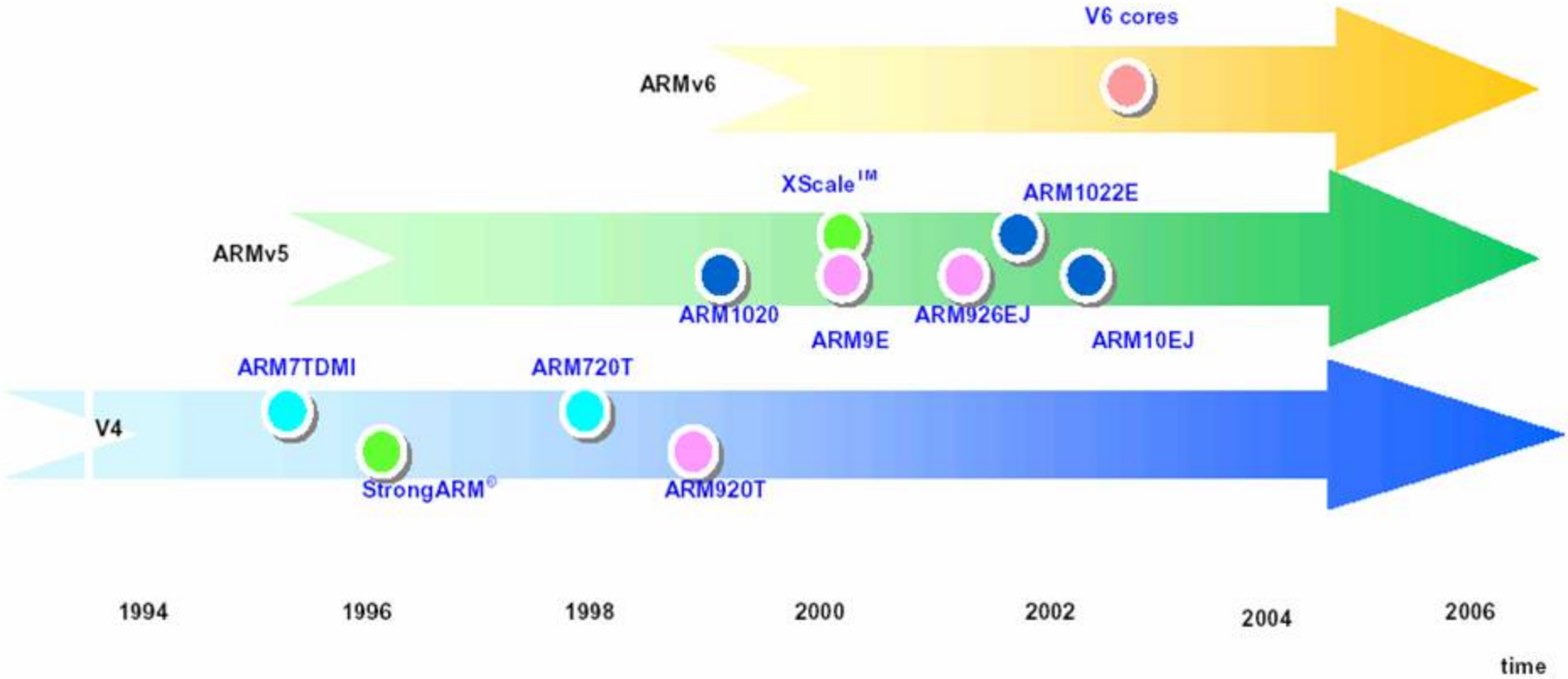
1.1 嵌入式处理器分类


嵌入式处理器
嵌入式微控制器
（MCU）
嵌入式DSP处理器
(DSP)
嵌入式微处理器
(MPU)
嵌入式片上系统
(System On Chip)

1.2 ARM的由来

- ARM（Advanced RISC Machines），既可以认为是一个公司的名字，也可以认为是对一类微处理器的通称，还可以认为是一种技术的名字。
- 1991年ARM公司成立于英国剑桥，主要出售芯片设计技术的授权。

1.3 ARM结构体系和处理器家族发展



1.4 ARM微处理器的特点(1/5)

- 低功耗、低成本、高性能
  - 采用RISC指令集
  - 使用大量的寄存器
  - ARM/THUMB指令支持
  - 三/五级流水线

1.4 ARM微处理器的特点(2/5)

- 采用RISC体系结构
  - 固定长度的指令格式，指令归整、简单、基 本寻址方式有2～3种；
  - 使用单周期指令，便于流水线操作执行；
  - 大量使用寄存器，数据处理指令只对寄存器进行操作，只有加载/存储指令可以访问存储器，以提高指令的执行效率。

1.4 ARM微处理器的特点(3/5)

- 大量使用寄存器
- ARM 处理器共有37个寄存器，被分为若干个组，这些寄存器包括：
  - 31个通用寄存器，包括程序计数器（PC 指针），均为32位的寄存器；
  - 6个状态寄存器，用以标识CPU的工作状态及程序的运行状态，均为32位。

1.4 ARM微处理器的特点(4/5)

- 高效的指令系统
- ARM微处理器支持两种指令集：ARM指令集和Thumb指令集。
- ARM指令为32位的长度，Thumb指令为16位长度。Thumb指令集为ARM指令集的功能子集，但与等价的ARM代码相比较，可节省30％～40％以上的存储空间，同时具备32位代码的所有优点。

1.4 ARM微处理器的特点(5/5)

- 除此以外，ARM体系结构还采用了一些特别的技术，在保证高性能的前提下尽量缩小芯片的面积，并降低功耗。
- 所有的ARM指令都可根据前面的执行结果决定是否被执行，从而提高指令的执行效率。
- 可用加载/存储指令批量传输数据，以提高数据的传输效率。
- 可在一条数据处理指令中同时完成逻辑处理和移位处理。
- 在循环处理中使用地址的自动增减来提高运行效率。

1.5 ARM微处理器系列

- ARM7系列
- ARM9系列
- ARM9E系列
- ARM10E系列
- SecurCore系列
- Inter的Xscale
- Inter的StrongARM
- 其中，ARM7、ARM9、ARM9E和ARM10E为4个通用处理器系列，每一个系列提供一套相对独特的性能来满足不同应用领域的需求。SecurCore系列专门为安全要求较高的应用而设计。

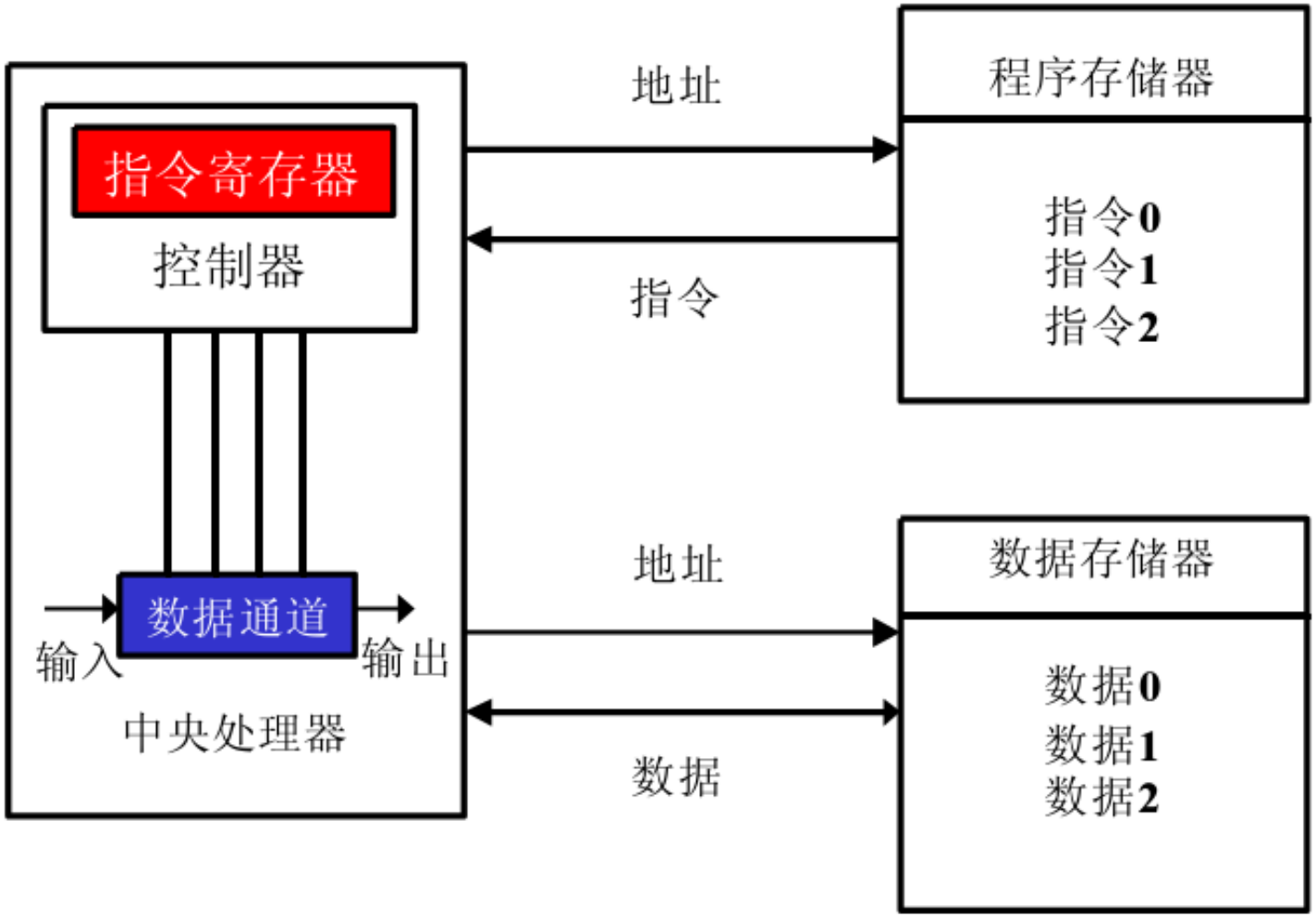
内容提纲(2/6)

- 1. ARM处理器介绍
- 2. 三星S3C2410X(ARM9)芯片简介
- 3. GNU交叉工具链
- 4. 嵌入式Linux开发环境的设置
- 5. 嵌入式应用程序开发示例
- 6. 实验内容与要求

2.1 ARM9微处理器系列

- ARM9系列微处理器在高性能和低功耗特性方面提供最佳的表现。具有以下特点
  - 5级流水线，指令执行效率更高。
  - 提供1.1MIPS/MHz的哈佛结构。
  - 支持32位ARM指令集和16位Thumb指令集。
  - 支持32位的高速AMBA总线接口。
  - 全性能的MMU，支持Windows CE、Linux、Palm OS等多种主流嵌入式操作系统。
  - MPU支持实时操作系统。
  - 支持数据Cache和指令Cache，具有更高的指令和数据处理能力。

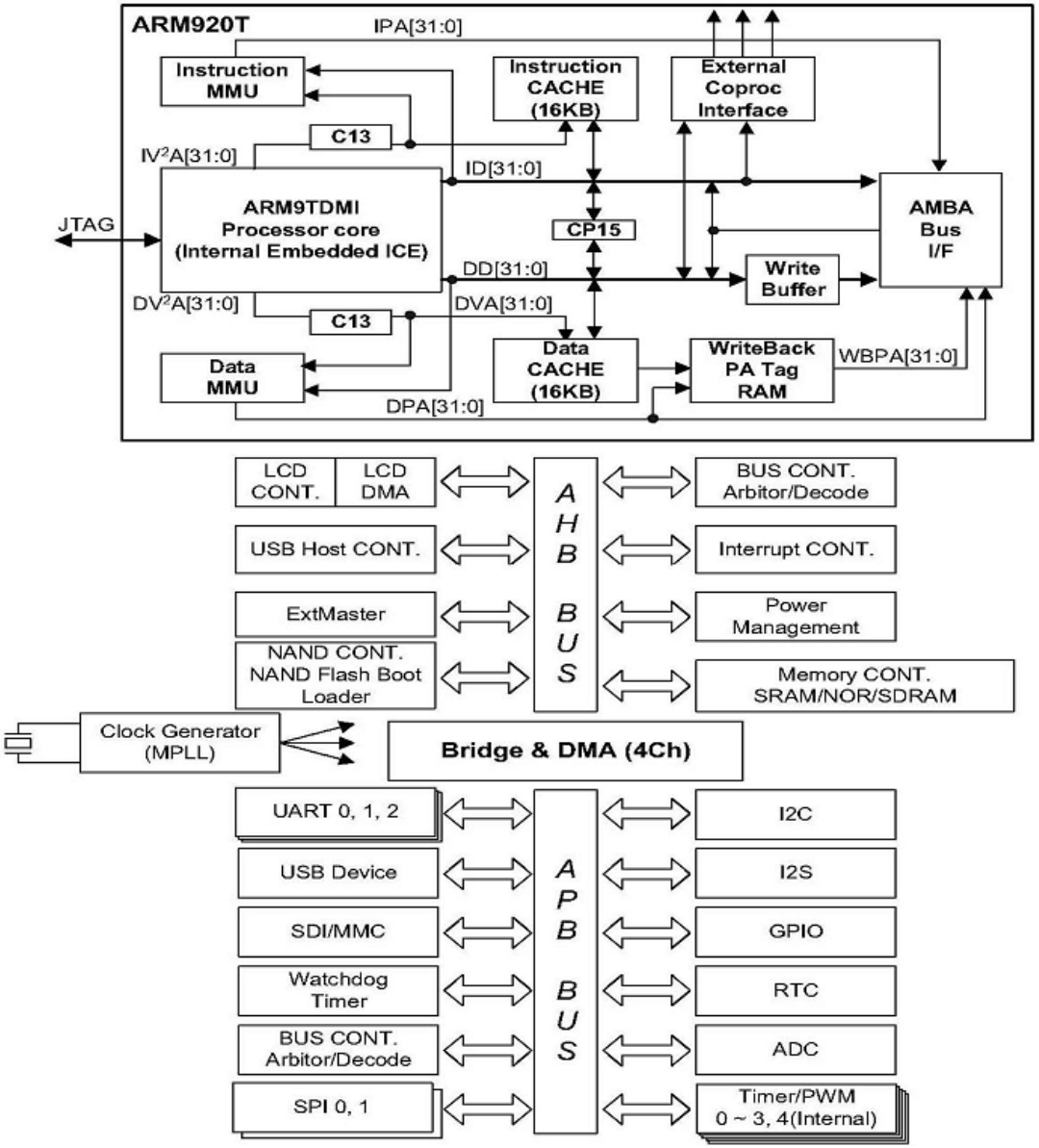
2.2 哈佛体系结构


指令寄存器
控制器
数据通道
输入
输出
中央处理器
地址
指令
地址
数据
程序存储器
指令0
指令1
指令2
数据存储器
数据0
数据1
数据2

2.3 哈佛体系结构的特点

- 程序存储器与数据存储器分开
- 提供了较大的数存储器带宽
- 适合于数字信号处理
- 大多数DSP都是哈佛结构
- ARM9是哈佛结构

2.4 S3C2410X内部结构图



2.5 S3C2410X片上资源

- 16KB 指令Cache，16KB数据Cache；
- 存储器控制器，支持NAND FLASH启动，4KB用于启动的内部缓存区；
- 55个中断源，24个外部中断口；
- 4通道16bit带PWM的定时器及1通道16bit内部定时器；
- 3通道UART、1个多主I2C总线控制器、1个IIS总线控制器，一个SPI接口；
- LCD控制器，支持黑白、STN、TFT显示器；触摸屏接口支持；
- 支持SD卡/MMC卡；
- 两个USB主、一个USB从；
- 4个DMA控制器， 8通道10位ADC；
- 实时时钟等。

2.6 S3C2410X特性

- 内核:1.8V　I/O : 3.3 V
- 最高为203MHz
- 272脚的FBGA封装

内容提纲(3/6)

- 1. ARM处理器介绍
- 2. 三星S3C2410X(ARM9)芯片简介
- 3. GNU交叉工具链
- 4. 嵌入式Linux开发环境的设置
- 5. 嵌入式应用程序开发示例
- 6. 实验内容与要求

3.1 DENX ELDK开发套件

- DENX的ELDK（Embedded Linux Development Kit）是在GPL协议下发行的公开全部源代码的自由软件，它可以提供一套完整的嵌入式系统开发环境，支持ARM，PowerPC和MIPS处理器，包括以下组件：
  - 交叉编译工具（Cross Development Tools）：编译器、汇编器和链接器等，可在宿主机上开发在目标板上运行的软件；
  - 本地开发工具（Native Tools）：Shell、常用命令和库文件等，可在目标板上提供一套标准Linux开发环境；
  - 固件（Firmware）：能过很容易的移植到新的板子和处理器上；
  - Linux内核（Linux Kernel）：包括所有设备驱动和板级支持功能的完整内核源代码；
  - RTAI（Real Time Application Interface）：可将系统扩展为满足硬实时响应；
  - SELF（Simple Embedded Linux Framework）：可以做为构建你的嵌入式系统的基础。

下载网址：http://www.denx.de/ftp/pub/eldk/

3.2 ELDK的安装方法

1. 下载ELDK的ISO文件：http://ftp.denx.de/pub/eldk/4.1/arm-linux-x86/iso/arm-2007-01-21.iso或者最新版的ISO
2. 建立安装目录：bash$ mkdir /opt/eldk
3. 挂载ISO文件：bash$ mount –o loop arm-2007-01-21.iso /mnt/cdrom
4. 运行安装脚本：bash$ /mnt/cdrom/install –d /opt/eldk
5. 设置环境变量：bash$ export CROSS_COMPILE=arm-linux-
6. 设置搜索路径：bash$ PATH=$PATH:/opt/eldk/usr/bin:/opt/eldk/bin

3.3 GNU交叉工具链介绍

名称	归属	作用
arm-linux-as	binutils	编译ARM汇编程序
arm-linux-ar	binutils	把多个.o合并成一个.o或静态库(.a)
arm-linux-ranlib	binutils	为库文件建立索引，相当于arm-linux-ar -s
arm-linux-ld	binutils	链接器，把多个.o或库文件链接成一个可执行文件
arm-linux-objdump	binutils	查看目标文件(.o)和库(.a)的信息
arm-linux-objcopy	binutils	转换可执行文件的格式
arm-linux-strip	binutils	去掉elf文件的符号信息，使其体积变小
arm-linux-readelf	binutils	读elf文件的信息
arm-linux-gcc	gcc	编译C程序(.c)或汇编程序(.S)
arm-linux-g++	gcc	编译C++程序

3.4 GNU交叉工具链使用示例（1/2）

- 首先编辑一个c文件，比如hello.c：

```
#vi hello.c

#include <stdio.h>
int main()
{
  printf(“Hello world!\n”);
  return 0;
}
```

3.4 GNU交叉工具链使用示例 （2/2）

- 使用交叉编译工具编译一个hello的arm可执行程序:

  #arm-linux-gcc -Wall -o hello hello.c

  -c:只编译不链接，-o:编译且链接，-Wall:显示出错信息

- 读elf文件的文件头

  #arm-linux-readelf –h hello

  ELF Header:

  Magic: 7f 45 4c 46 01 01 01 61 00 00 00 00 00 00 00 00

  Class: ELF32

  Data: 2's complement, little endian

  Version: 1 (current)

  OS/ABI: ARM

  ABI Version: 0

  Type: EXEC (Executable file)

  Machine: ARM

  Version: 0x1

  Entry point address: 0x82b4

  。 。 。

3.5 Makefile文件

- 只要Makefile写得够好，make命令会自动智能地根据当前的文件修改的情况来确定哪些文件需要重编译，从而自己编译所需要的文件和链接目标程序。比如：
  1. 如果这个工程没有编译过，那么所有C文件都要编译并被链接。
  2. 如果这个工程的某几个C文件被修改，那么只编译被修改的C文件，并链接目标程序。
  3. 如果这个工程的头文件被改变了，那么需要编译引用了这几个头文件的C文件，并链接目标程序。

3.6 其他现成的交叉编译工具链

- 实验室中还安装好了另外一套现成的交叉编译工具链，主编译器armv4l-unknown-linux-gcc在路径/opt/host/armv4l/bin/中，如果在~/.bash_profile文件中PATH变量设为PATH=$PATH:$HOME/bin:/opt/host/armv4l/bin/，那么armv4l-unknown-linux-gcc会自动搜索到，可以在终端上输入arm，然后按tab键，会自动显示armv4l-unknown-linux-

- 注意：armv4l中是字母L的小写，表示armv4核的小端模式；该交叉编译工具链也是GNU的，只是采用的gcc版本不同。

内容提纲(4/6)

- 1. ARM处理器介绍
- 2. 三星S3C2410X(ARM9)芯片简介
- 3. GNU交叉工具链
- 4. 嵌入式Linux开发环境的设置
- 5. 嵌入式应用程序开发示例
- 6. 实验内容与要求

4.1 Linux基本操作环境

- Linux系统可以在两种环境下操作：一种是在X Window的桌面环境下操作，它与Microsoft Windows环境下的操作十分相似；另一种是在控制台的字符屏幕下操作，它通过键盘输入命令来实现对系统的操作。
- shell是一种命令行解释程序（Command-Language Interpreter），负责用户和操作系统的沟通。
- 在shell中键入startx命令可启动X Window。

4.2 登录系统

- 在字符终端界面环境下，终端屏幕上出现如下登录提示行 login:

  在它的后面输入自己的登录名。本系统可使用普通用户arm登录。
- 输入登录名、并按【Enter】键后，在屏幕上出现

  password:
- 要求在其后输入口令。本系统root的口令为：arm。Sudo命令可以管理员身份执行，口令也为：arm。输入的口令字符串并不在屏幕上显示，以利于保密。输入完口令并按【Enter】键后，系统就对登录名和口令进行验证。如果确认无误，则在屏幕上显示若干行信息，最后一行出现：$
- $是一般用户的shell提示符，其后是闪烁的光标条（root用户的提示符是“#”）。
- 注意：请勿修改实验室机器上任何用户的密码。

4.3 重新启动和关闭系统

- 只有超级管理员用户才能重启系统，在提示符下输入：#shutdown –r now
- 同样只有超级管理员用户才能关闭系统，在提示符下输入：#shutdown –h now

4.4 一些常用操作命令

- 查看系统中的磁盘情况：#fdisk –l
- 挂载U盘：#mount /dev/sda1 /mnt
- 查看磁盘空间：#df –h
- 查看目录容量：#du -h
- 查看网络配置：#ifconfig -a
- 配置网络IP：#ifconfig eth0 192.168.0.10
- 配置网关：#route add default gw 192.168.0.254
- 测试网络：#ping –c 4 192.168.0.11

4.5 minicom的设置(1/3)

- 在linux 终端下运行

  # minicom

  会出现如下界面：

```
Welcome to minicom 2.1

OPTIONS: History Buffer, F-key Macros, Search History
Buffer, I18n
Compiled on Mar 29 2005, 09:39:09.

Press CTRL-A Z for help on special keys
```

4.5 minicom的设置(2/3)

- 按下 CTRL－A Z 出现如下界面：

```
                    Minicom Command Summary

          Commands can be called by CTRL-A <key>

               Main Functions                 Other Functions

Dialing directory..D  run script (Go)....G | Clear Screen.......C
Send files.........S  Receive files......R | cOnfigure Minicom..O
comm Parameters....P  Add linefeed.......A | Suspend minicom....J
Capture on/off.....L  Hangup.............H | eXit and reset.....X
send break.........F  initialize Modem...M | Quit with no reset.Q
Terminal settings..T  run Kermit.........K | Cursor key mode....I
lineWrap on/off....W  local Echo on/off..E | Help screen........Z
                                           | scroll Back........B

      Select function or press Enter for none.

            Written by Miquel van Smoorenburg 1991-1995
            Some additions by Jukka Lahtinen 1997-2000
            i18n by Arnaldo Carvalho de Melo 1998
```

4.5 minicom的设置(3/3)

- 按P出现设置界面，把它设置成下图：

```
┌─────────[Comm Parameters]──────────┐
│                                    │
│ Current: 115200 8N1                │
│                                    │
│   Speed          Parity       Data │
│                                    │
│ A: 300           L: None      S: 5 │
│ B: 1200          M: Even      T: 6 │
│ C: 2400          N: Odd       U: 7 │
│ D: 4800          O: Mark      V: 8 │
│ E: 9600          P: Space          │
│ F: 19200                   Stopbits│
│ G: 38400                     W: 1  │
│ H: 57600                     X: 2  │
│ I: 115200        Q: 8-N-1          │
│ J: 230400        R: 7-E-1          │
│                                    │
│ Choice, or <Enter> to exit?        │
└────────────────────────────────────┘
```

4.5 minicom的设置(4/4)

- minicom 串口参数设置正确，按下开发板上的Reset 按键，会得到以下输出：

```
This bootloader is based www.mizi.com vivi bootloader.
***             ***    ***            ***             ****   ***************
***             ***    ***            ***          ****      ***************
***             ***    ***            ***      ****                 ****
***             ***    ***            ***    ****                   ****
***             ***    ***            ***    ****                   ****
****************    ***            ***    ****                   ****
****************    ***            ***       *****               ****
***             ***    ***            ***             ****        ****
***             ***    ***            ***                ****     ****
***             ***    ***            ***             ****        ****
***             ***    ***            ***          *****          ****
***             ***    ****************       *****               ****
***             ***    ****************    *****                  ****

                                                                MMU table base addreO
NAND device: Manufacture ID: Oxec, Chip ID: Ox76 (Samsung K9D1208VOM)
Found saved vivi parameters.
Number of parameters: 11
name                          :            hex                integer
--------------------------------------------------------------------
mach_type                     :        000000c1                    193
media_type                    :        00000003                      3
boot_mem_base                 :        30000000              805306368
baudrate                      :        0001c200                 115200
xmodem_one_nak                :        00000000                      0
xmodem_initial_timeout        :        000493e0                 300000
xmodem_timeout                :        000f4240                1000000
ymodem_initial_timeout        :        0016e360                1500000
boot_delay                    :        00100000                1048576
distype                       :        00000000                      0
ostype                        :        00000001                      1
Linux command line: noinitrd root=/dev/mtdblock/3 init=/linuxrc console=ttyS0 ic
mtdpart info.  (5 partitions)
name                offset          size          flag
-------------------------------------------------------
vivi              : 0x00000000      0x00020000      0   128k
param             : 0x00020000      0x00010000      0    64k
kernel            : 0x00030000      0x001c0000      0     1M+768k
root              : 0x00200000      0x00200000      0     2M
usr               : 0x00400000      0x03cf8000      0    60M+992k
Press Return to start the OS   now, other key for vivi shell
type "help" for help.
 HUSTEI>
```

4.6 配置NFS服务器(1/2)

- 设PC机ip地址是192.168.0.10，它将作为NFS服务器。2410实验箱ip地址设为192.168.0.11。

  修改/etc/exports文件如下：

  #more /etc/exports

  /home/nfs 192.168.0.11(rw,sync,no_root_squash)

- 该文件的意思是允许IP为192.168.0.11的机器来装载/home/nfs这个目录。rw表示192.168.0.11的机器以读写权限来挂接该文件系统；no_root_squash表示192.168.0.11的机器以主机上的root身份挂接该文件系统。

4.6 配置NFS服务器(2/2)

- 启动NFS服务器运行

  # exportfs –rav

  # /etc/init.d/nfs restart(或start)

- 验证NFS服务器是否正常，将PC机的/home/nfs挂载到实验箱的/tmp目录，在实验箱上执行：

  #mount -t nfs 192.168.0.10:/home/nfs /tmp

  然后就可以如同本地磁盘一样使用主机上nfs目录了。

内容提纲(5/6)

- 1. ARM处理器介绍
- 2. 三星S3C2410X(ARM9)芯片简介
- 3. GNU交叉工具链
- 4. 嵌入式Linux开发环境的设置
- 5. 嵌入式应用程序开发示例
- 6. 实验内容与要求

5.1 编辑源代码

- 首先在PC机的/home/nfs目录下编辑一个c文件，比如hello.c：

  #vi hello.c

- hello.c的源代码如下：

```
#include <stdio.h>
 int main(void)
 {
    printf ("Hello world, Linux programming!\n");
    return 0;
 }
```

5.2 编写Makefile文件(1/2)

```
CC = /opt/host/armv4l/bin/armv4l-unknown-linux-gcc
EXEC = hello
OBJS = hello.o
CFLAGS +=-Wall -g
LDFLAGS += -static

all: $(EXEC)
$(EXEC): $(OBJS)
    $(CC) $(LDFLAGS) -o $@ $^

clean:
    rm -f $(EXEC) *.elf *.gdb *.o
```

5.2 编写Makefile文件(2/2)

- 以上是本例用到的Makefile文件，它有几个主要部分：
  CC 指明编译器
  EXEC 表示编译后生成的执行文件名称
  OBJS 目标文件列表
  CFLAGS 编译参数
  LDFLAGS 连接参数
  all: 编译主入口
  clean: 清除编译结果

- 注意：“$(CC) $(LDFLAGS) -o $@ $^”和“rm -f $(EXEC) *.elf *.gdb *.o”前空白由一个Tab制表符生成，不能单纯由空格来代替。另外还要注意缺省规则的使用。

5.3 编译应用程序

- 在上面的步骤完成后，我们将hello.c和Makefile文件放在同一目录下，运行make来编译程序。程序进行修改需重新编译，则运行make clean，然后再make。
- 若不用Makefile，直接用gcc编译也行

/opt/host/armv4l/bin/armv4l-unknown-linux-gcc –o hello hello.c

- 注意：编译、修改程序都是在宿主机（本地PC机）上进行，不能在minicom下进行。

5.4 通过NFS调试运行程序

- 前面配置开发环境时已在宿主PC上启动了NFS服务，并设置好了共享目录，可以在终端中输入minicom，建立实验箱与宿主PC机之间的通讯了。

  #mount -t nfs 192.168.0.10:/home/nfs /tmp

- 成功挂接宿主PC的nfs目录（假定编写了hello.c和Makefile都在此目录下）后，在实验箱上进入/tmp目录便相应进入宿主PC的/nfs目录。在/tmp目录下运行编译好的hello程序：#./hello

  可以看到结果：Hello world, Linux programming!

- 注意：实验箱挂接宿主机目录只需挂接一次，只要实验箱没有重启，就能一直保持连接。这样可以反复 修改、编译、调试，不需要下载到实验箱。

内容提纲(6/6)

- 1. ARM处理器介绍
- 2. 三星S3C2410X(ARM9)芯片简介
- 3. GNU交叉工具链
- 4. 嵌入式Linux开发环境的设置
- 5. 嵌入式应用程序开发示例
- 6. 实验内容与要求

6.1 本次实验要求

- 编写数组排序程序
  - 在Linux下编写主程序和函数库，然后编译连接成ARM可运行的二进制文件，最后把该二进制文件下载到目标机（实验箱平台）上运行，查看运行结果是否正确。
  - 在C程序main函数中，接收用户输入（用户任意输入多个整数），然后在main中调用另一个C源文件中编写的函数（单独存储成一个文件，在该函数中完成对这多个整数的排序功能），然后再在C程序main函数中输出这几个排好顺序的整数。

6.2 主程序的示例

```
#include <stdio.h>   #include <stdlib.h>   #include “lib.h”

int main()
{
    int i=0;   int num=0;   int *array=NULL;
    while(num <= 0){
        printf(“\nPlease enter the number of elements:\n");
        scanf("%d",&num);
        if(num > 0) break;}
    if(NULL == (array = (int *)malloc(num*sizeof(int)))) {
        printf("malloc failed!\n");
        exit(-1);}
    printf(“Please enter the elements(total %d):\n“, num);
    for(i = 0; i<num; i++){
        printf("\nNo.%d:\t", i);
        scanf("%d", array+i);}
    sort(array, num);
    printf("========================\nthe result is:\n");
    for(i = 0; i<num; i++) printf("%d:\t%d\n", i, *(array+i));
    free(array);   return 0;
}
```

6.3 Makefile文件示例

```
CC = /opt/host/armv4l/bin/armv4l-unknown-linux-gcc
EXEC = example
OBJS = example.o lib.o
CFLAGS +=  -Wall –g
LDFLAGS +=

all: $(EXEC)
$(EXEC): $(OBJS)
    $(CC) $(LDFLAGS) -o $@ $^
example.o: example.c
    $(CC) $(CFLAGS) -o $@ -c $^
lib.o: lib.c
    $(CC) $(CFLAGS) -o $@ -c $^
clean:
    rm -f $(EXEC) *.o
```