# MIPS架构概念和实验环境搭建

## 处理器体系架构和体系结构的概念

* **体系架构：**

 CPU架构是CPU厂商给属于同一系列的CPU产品定的一个规范，主要目的是为了区分不同类型CPU的重要标示。

 目前市面上的CPU主要分有两大阵营，一个是INTEL系列CPU，另一个是AMD系列CPU。

* **体系结构：**

在计算世界中, "体系结构"一词被用来描述一个抽象的机器,而不是一个具体的机器实现。 一般而言，一个CPU的体系结构有一个指令集加上一些寄存器而组成。“指令集”与“体系结构”这两个术语是同义词。

## 二、MIPS架构基本概念

MIPS架构（Microprocessor without Interlocked Pipeline Stages），是一种采取精简指令集（RISC）的处理器架构，1981年出现，由MIPS科技公司开发并授权，广泛被使用在许多电子产品、网络设备、个人娱乐装置与商业装置上。最早的MIPS架构是32位元，最新的版本已经变成64位元。

发展历史：

1981年，斯坦福大学教授约翰·轩尼诗领导他的团队，实作出第一个MIPS架构的处理器。他们原始的概令是透过指令管线化来增加CPU运算的速度。

1984年，约翰·轩尼诗教授离开斯坦福大学，创立MIPS科技公司。于1985年，设计出R2000芯片，1988年，将其改进为R3000芯片。

2002年，中国科学院计算所开始研发龙芯处理器，采用MIPS架构，但未经MIPS公司的授权，遭到侵权的控告。2007年，中国科学院与MIPS公司达成和解，得到正式授权。

## 三、常见的CPU架构比较——X86、ARM、MIPS

* **X86：**

X86采用了CISC指令集。在CISC指令集的各种指令中，大约有20%的指令会被反复使用，占整个程序代码的80%。而余下的80%的指令却不经常使用，在程序设计中只占20%。

* **总线接口部件BIU**

总线接口部件由4个16位段寄存器（DS,ES,SS,CS）、一个16位指令指针寄存器(IP)、20位物理地址加法器、6字节指令队列（8088为4字节）及总线控制电路组成，负责与存储器及I/O端口的数据传送。

* **执行部件EU**

执行部件由ALU、寄存器阵列(AX,BX,CX,DX,SI,DI,BP,SP)、标志寄存器(PSW)等几个部分组成，其任务就是从指令队列流中取出指令，然后分析和执行指令，还负责计算操作数的16位偏移地址。

* **寄存器的结构**

1）数据寄存器AX、BX、CX、DX均为16位的寄存器，它们中的每一个又可分为高字节H和低字节L。即AH、BH、CH、DH及AL、BL、CL、DL可作为单独的8位寄存器使用。不论16位寄存器还是8位寄存器，它们均可寄存操作数及运算的中间结果。有少数指令指定某个寄存器专用，例如，串操作指令指定CX专门用作记录串中元素个数的计数器。

2）段寄存器组：CS、DS、SS、ES。8086/8088的20位物理地址在CPU内部要由两部分相加形成的。SP、BP、SI、DI是用以指明其偏移地址，即20位物理地址的低16位；而CS、DS、SS、ES是用以指明20位物理地址的高16位的，故称作段寄存器。

4个存储器使用专一，不能互换，CS识别当前代码段，DS识别当前数据段，SS识别当前堆栈段；ES识别当前附加段。一般情况下，DS和ES都须用户在程序中设置初值。

3）控制寄存器组：IP和FLAG。指令指针IP用以指明当前要执行指令的偏移地址（段地址由CS提供）。标志寄存器FLAG有16位，用了其中的九位，分两组：状态标志和控制标志。前者用以记录状态信息，由6位组成，后者用以记录控制信息由3位组成。6位状态标志，包括CF、AF、OF、SF、PF和ZF，它反映前一次涉及ALU操作的结果，对用户它“只读不写”。控制标志包括方向标志DF，中断允许标志IF及陷阱标志TF，中断允许标志IF及陷阱标志TF，可通过指令设置。

* **MIPS：**
* 采用RISC指令集
* 所有指令都是32位编码；
* 有些指令有26位供目标地址编码；有些则只有16位。因此要想加载任何一个32位值，就得用两个加载指令。16位的目标地址意味着，指令的跳转或子函数的位置必须在64K以内（上下32K）；
* 所有的动作原理上要求必须在1个时钟周期内完成，一个动作一个阶段；
* 有32个通用寄存器，每个寄存器32位（对32位机）或64位（对64位机）；
* 本身没有任何帮助运算判断的标志寄存器，要实现相应的功能时，是通过测试两个寄存器是否相等来完成的；
* 所有的运算都是基于32位的，没有对字节和对半字的运算（MIPS里，字定义为32位，半字定义为16位）；
* 没有单独的栈指令，所有对栈的操作都是统一的内存访问方式。因为push和pop指令实际上是一个复合操作，包含对内存的写入和对栈指针的移动；
* 由于MIPS固定指令长度，所以造成其编译后的二进制文件和内存占用空间比x86的要大，（x86平均指令长度只有3个字节多一点，而MIPS是4个字节）；
* 寻址方式：只有一种内存寻址方式。就是基地址加一个16位的地址偏移；
* 内存中的数据访问必须严格对齐（至少4字节对齐）；
* 跳转指令只有26位目标地址，再加上2位的对齐位，可寻址28位的空间，即256M；
* 条件分支指令只有16位跳转地址，加上2位的对齐位，共18位寻址空间，即256K；
* MIPS默认不把子函数的返回地址（就是调用函数的受害指令地址）存放到栈中，而是存放到$31寄存器中；这对那些叶子函数有利。如果遇到嵌套的函数的话，有另外的机制处理；
* 高度的流水线：

MIPS指令的五级流水线：（每条指令都包含五个执行阶段）

第一阶段：从指令缓冲区中取指令。占一个时钟周期；

第二阶段：从指令中的源寄存器域（可能有两个）的值（为一个数字，指定$0~$31中的某一个）所代表的寄存器中读出数据。占半个时钟周期；

第三阶段：在一个时钟周期内做一次算术或逻辑运算。占一个时钟周期；

第四阶段：指令从数据缓冲中读取内存变量的阶段。从平均来讲，大约有3／4的指令在这个阶段没做什么事情，但它是指令有序性的保证。占一个时钟周期；

第五阶段：存储计算结果到缓冲或内存的阶段。占半个时钟周期；

所以一条指令要占用四个时钟周期；

* **ARM：**

ARM处理器是一个32位元精简指令集(RISC)处理器架构,其广泛地使用在许多嵌入式系统设计。

* 采用RISC指令集
* 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：

1）所有的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。

2）可用加载/存储指令批量传输数据，以提高数据的传输效率。

* 寄存器结构

ARM处理器共有37个寄存器，被分为若干个组（BANK），这些寄存器包括：

1）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。

2）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，目前只使用了其中的一部分。

* 指令结构：

ARM微处理器的在较新的体系结构中支持两种指令集：ARM指令集和Thumb指令集。其中，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。

## 四、实验环境搭建(实验用到的操作系统 ubuntu10.04 32位)

* **安装并使用交叉编译器：**
* 交叉编译就是在一个平台上生成另一个平台上的可执行代码。这里需要注意的是所谓平台，实际上包含两个概念：体系结构(Architecture)、操作系统(Operating System)。
* 为什么要安装交叉编译器：
* 目的平台上不允许或不能够安装我们所需要的编译器，而我们又需要这个编译器的某些特征。
* 目的平台上的资源贫乏，无法运行我们所需要编译器。
* 安装过程：
* 下载mips-2007-01-21.iso
* 下载地址：http://ftp.sunet.se/pub/Linux/distributions/eldk/4.1/mips-linux-x86/iso/
* 解压mips-2007-01-21.iso，解压命令：sudo mount -o loop mips-2007-01-21.iso /home/mipsiso/
* 在目录/opt下建目录eldk ：sudo mkdir /opt/eldk
* 把解压目录设为当前目录，输入命令：sudo ./install -d /opt/eldk 来运行安装文件install。其中，参数-d指定安装目录为/opt/eldk
* 为了以后方便，把路径添加到环境变量中，命令是：gedit ~/.bashrc

加入如下语句：#Mips 4KC toolchain

export PATH=/opt/eldk/usr/bin:$PATH

最后source ~/.bashrc

* 检查安装是否成功，查看安装路径/usr/bin/mips\_4KC-\*，是否包含所有常用的gcc工具集。并且可以利用mips\_4KC-gcc和mips\_4KC-objdump来检验工具是否可用。
* **安装并使用仿真器：**
* 了解仿真器：

由于实验的操作系统内核是运行在 Mips 体系结构上的，而我们平常使用的是基于 x86 体系结构的 PC，所以需要使用仿真器让我们的操作系统内核能够运行在 Mips 体系结构上。在这个实验中我们使用的是 GXemul 仿真器。

* 安装GXemul仿真器：
* 运行sudo apt-get install gxemul即可完成安装，或者去网站上下载，下载地址：http://gxemul.sourceforge.net/src/（注意，使用的版本是GXemul 0.4.6，如果版本不对，则需要重新下载）
* 解压gxemul-0.4.6.tar.gz，解压命令：sudo tar -zxvf gxemul-0.4.6.tar.gz
* 编译：进入解压目录，./configure

make

make install

cp gxemul /bin

* 测试是否安装成功：进入r3000\_test文件所在的目录，然后在命令行输入gxemul @r3000\_test，应该出现如下界面：

GXemul 0.4.6 Copyright (C) 2003-2007 Anders Gavare

Read the source code and/or documentation for other Copyright messages.

Creating emulation from configfile "r3000\_test":

name: "MALTA R3000"

machine "SCSE-1 Testing":

memory: 64 MB

cpu0: R3000 (I+D = 4+4 KB)

machine: MIPS test machine

loading test

starting cpu0 at 0x80010000

-------------------------------------------------------------------------------

此时应该弹出另一窗口，中间显示

main.c: main is start ...

init.c: mips\_init() is called

panic at init.c:14: ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^

仿真器既为正确