基于远程设备的汇编源码调试器的设计与实现

包 磊,姚放吾
(南京邮电大学 计算机学院,江苏 南京 210003)

摘 要:基于远程设备的汇编语言调试器为用户提供了一个友好的汇编语言软件调试平台,能够让用户方便的对运行在没有操作系统的远程设备上的汇编程序进行调试。文中对整个调试系统做了简要的介绍,重点阐述了运行在远程设备上的监控程序的设计。监控程序巧妙地运用了Intel处理器单步中断等特点,准确高效地实现了单步运行、运行到断点以及显示和更改远程设备上内存和存储器值等符号化调试功能,能够让用户清晰地了解到汇编程序运行的机理,极大地方便了远程设备上的汇编语言的开发。

关键词:源码调试;远程设备;调试信息
中图分类号:TP313 **文献标识码**:A **文章编号**:1673-629X(2009)11-0155-04

Design and Realization of Source Level Assembler Language Debugger Based on Remote Device

BAO Lei, YAO Fang-wu
(College of Computer, Nanjing University of Posts & Telecommunication, Nanjing 210003, China)

Abstract: Assembler language source level debugger environment provides a friendly software platform for the assembler language, which can conveniently debug assembler language running on a remote device without operation system. Briefly introduce the whole debugging system, and focus on the monitor program on the remote device. The monitor program made good things out of the features of Intel processer of single step interruption, such as single step running, running to the breakpoint, and showing & verifying the contents of memory and registers of the remote device. The system helps the developers understand the mechanism of the assembler language clearly, and makes a very easy development for the assembler language on the remote device.

Key words: source level debugging; remote device; debugging information

0 引 言

汇编语言是面向机器的程序设计语言,它是一种能够利用计算机所有硬件特性并能直接控制硬件的语言。在编写和运行汇编程序的过程中,经常会遇到一些问题,需要对程序进行分析和调试。现在很多的编译器都会集成汇编语言调试器,还有专门的汇编调试器。如运行于DOS下的DEBUG就是一款专为汇编语言设计的调试器[1,2],它能使程序设计者观察和修改寄存器和存储单元的内容,并能监视目标程序的执行情况,使用户真正接触到CPU内部,与计算机产生最紧密的工作联系。但是这些编译器都有一个共同的缺点:必须运行于独特的操作系统环境之中。若在一个裸机(如没有系统的嵌入式设备)中调试汇编语言,这些调试器就无能为力了。

基于如上问题,笔者设计了基于远程设备的汇编语言源码调试器[3,4],这种调试器主要运用于没有操作系统的远程设备环境中,图1为其用例图。用户在上位机集成开发环境中编译汇编代码,并将可执行文件下载给远程设备,用户通过窗口发送各种调试命令来控制远程设备中代码的运行,远程设备将存储器和内存值等调试信息及时的返回给用户,这样用户就能

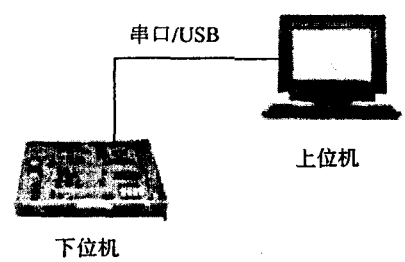

图1 基于远程设备的汇编源码调试器的用例图

收稿日期:2009-03-21;**修回日期**:2009-06-07
基金项目:江苏省企校合作项目(2008发04)
作者简介:包 磊(1984-),男,江苏淮安人,硕士研究生,研究方向为嵌入式技术及其在通信中的应用;姚放吾,硕士生导师,研究方向为并行计算机及其体系结构、嵌入式技术和计算机在通信中的应用。

够很方便的对汇编语言进行调试。

1 汇编源码调试器的总体结构设计

汇编源码调试器涉及到上位机和下位机两个部分,这两个部分的软件有着密切的分工和合作。上位机部分主要负责用户代码的编辑、编译、与下位机的通信、显示调试信息等功能。下位机主要负责符号化调试、调试信息的采集、与上位机的通信等。

2 通信功能的实现

图 2 显示了调试器的总体框架,从图中可以看出,通信模块在上位机和下位机的联系中起着重要的作用。为了通信的方便,制定了通信协议,图 3 是通信数据帧格式[5]。

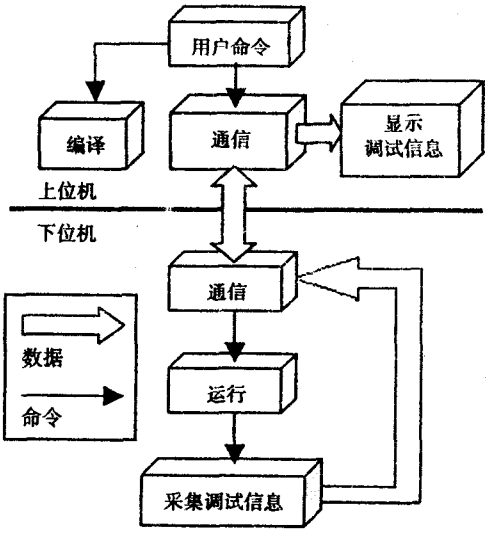

图 2 调试器的软件框架图

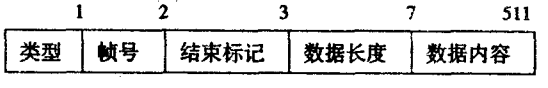

图 3 数据帧格式

一帧数据共有 512 个字节,分为 5 个部分,每个部分的含义如下:

类型:占 1 个字节,表明这帧数据的用途,该系统定义了 17 种命令,如需扩展功能,可以再定义新的命令,实现新的功能。

帧号:占 1 个字节,表明该数据帧在某次数据传输中序号。

结束标志:占 1 个字节,标志该数据帧是否是最后一帧,0 表示非最后一帧,1 表示是最后一帧。

数据长度:占 4 个字节,表明该数据帧所含的数据内容有效长度。

数据内容:占 502 个字节,表明该数据帧要传输的数据内容。

3 编译功能的实现

上位机的集成开发环境,需要对用户的源代码进行编译。MASM 是一款优秀的汇编语言编译器,它除了能生产可执行文件外,还能够根据用户的需求,生成 .OBJ,.MAP,.LST 等调试文件,给用户带来很大的方便。于是上位机集成开发环境就选择 MASM 作为编译引擎。

在该工程中编写了一个中介程序,这个中介程序调用 MASM 并随时获取其输出的编译信息,然后直接将编译信息用约定的进程间通讯方式(比如匿名管道)传回 GUI 程序显示,如图 4 所示。

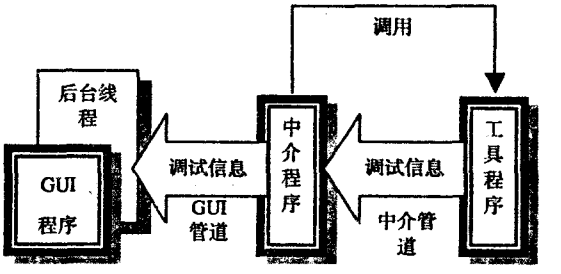

图 4 编译功能的实现

图 4 中,工具程序和中介程序都是以隐藏的方式运行。工具程序中原本输出到标准输出的信息被重定向到中介程序开辟的管道中,中介程序再利用 GUI 程序创建的管道将信息即时传递到 GUI 程序的一个后台线程里,后台线程负责刷新 GUI 程序的用户界面,及时的将编译信息显示给用户。

工具程序(MASM)编译链接后生成的是 EXE 文件,由于目标环境中没有 DOS 系统,所以不能直接加载 EXE 文件。这就需要对 EXE 文件进行信息提取,以手工方式加载 EXE 文件,需要提取其中的一些信息,包括:代码段、数据段、重定位信息以及指令偏移地址信息等。当将这些信息正确地传递到下位机后,下位机就可以正确地执行这个程序了。

4 汇编源码调试器的设计与实现

调试环境与目标机之间是以信令形式的数据流联系在一起的松耦合结构,根据系统的需要,调试环境被设计成一个用时间驱动的状态转换机[6,7],图 5 是调试环境的状态转换图。从图 5 中可以看出,在没有任何用户事件输入时,系统处于空闲状态。当串口或 USB 端口有数据,表示此时有用户的命令输入。完整地读取命令后、进行命令解析,根据不同的命令,进行不同的操作。命令完成之后,系统又返回到空闲状态,等待下一个命令。整个系统就在信息事件驱动中运行。

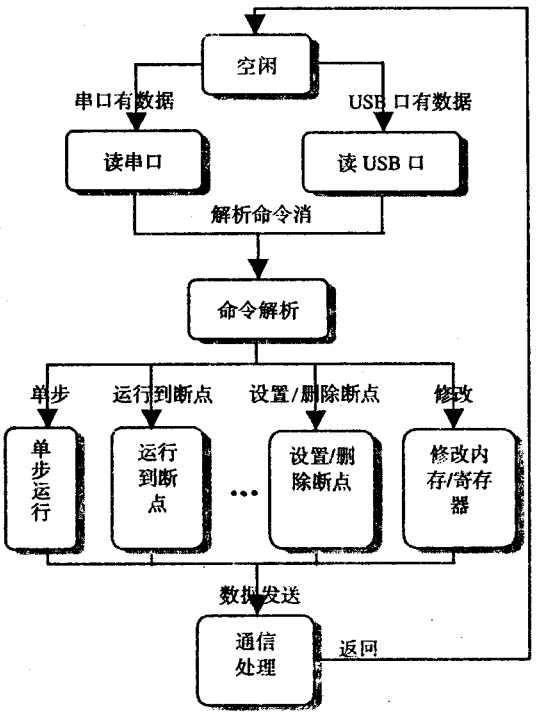

图 5　调试环境的状态转换图

5　符号化调试器实现的关键技术

符号化调试技术[8]的特点就是能够支持在源文件的任意程序语句上设置断点,并在这个断点位置打印和设置程序中所有变量、寄存器和内存的值。要想目标机在执行到断点时能够停下来,并将内存和寄存器的值返回给上位机,就必须有一套切换和保存环境的技术,这些技术主要包括运行到断点和单步运行。

符号化调试器的实现是目标机的主要功能。在目标机中运行着一个监控程序,负责实现符号化调试。在目标机中,用户代码和数据分别存放在一个固定的区域,当执行用户代码时,只需将用户当前的指令地址、数据段地址和堆栈地址分别送给 CPU 对应的寄存器就可实现从监控程序到用户程序的转换。当需要暂停用户程序时,只要先将当前指令地址、数据段地址和堆栈地址保存,再将 CPU 寄存器的值换成监控程序对应的值即可。其中应用了 INT 3(对应的十六进制为 CC)指令,在 3 型中断服务程序中实现了从用户程序到监控程序的转换、断点的撤销等功能。

5.1　运行到断点的实现

运行到断点是指实现程序运行到用户设置的断点处暂停运行,查看寄存器,内存数据等信息的一种调试方法。完成此功能需要两个操作:

1) 设置断点 用户在需要暂停的代码处设置断点(在目标机中的具体表现为在相应的用户代码地址处,将原代码替换为 INT 3 指令)。目标机在接收到用户设置断点命令时,把该断点地址存放于监控程序的内存缓冲区中,以便调试时使用。

2) 运行到断点 下位机接收到运行到断点命令后,监控程序从内存缓冲区中读取断点地址,并将所有断点处的代码保存,替换为 CC(INT 3),下位机保存监控程序的运行现场后,切换到用户程序执行。当用户程序运行到 INT 3 指令时,就会将用户代码的变量、内存、寄存器值返回给上位机,并且切换回监控程序。

图 6 展示了在设置断点前后内存源码镜像的变化。用户在 ADD DX, 2 和 JZ L1 这两条指令处设置了断点,在目标机的内存缓冲区中记录了这两条指令在代码段的偏移值,当用户执行"运行到断点"指令时,监控程序会将缓冲区中记录的偏移地址处的代码替换为 CC,切换到用户环境执行用户代码,当运行到 CC 时就会执行 3 型中断服务程序。3 型中断服务会将所有在缓冲区中记录的偏移地址处的 CC 替换为原来的代码,实现了运行到断点的命令。

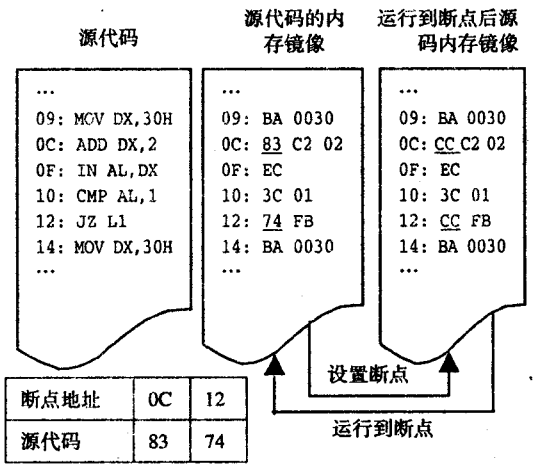

图 6　运行到断点的实现示意图

5.2　单步运行的实现

单步运行是指实现程序的单步运行后,查看寄存器、内存数据等信息的一种调试方法。单步运行有两种方式:单步跳跃和单步进入。这两种方式主要是针对过程调用型指令(如 CALL 指令)而言。当前指令为调用型指令时,单步跳跃执行后,会将整个过程执行完毕,代码停止在过程转移指令的下一条指令处。而单步进入执行后,代码停在过程体的第一条指令处。

单步运行的实现要借助于单步中断(设置 T 标志位)。在单步中断服务程序中,实现了用户程序切换到监控程序,以及调试信息的收集功能。

单步运行的具体实现方法如下:

1)非调用型指令。

监控程序保存监控现场,恢复用户环境,并将用户环境的 T 标志置 1。当切换到用户代码,并执行了一

条指令时，便发生单步中断，实现单步运行。

2)调用型指令。

(1)单步跳跃的实现。

执行调用型指令的单步跳跃时，上位机会将该指令的下一条指令的起始地址传输给目标机，目标机将该地址处的用户代码替换为 INT3 指令，然后切换到用户程序运行。当用户程序执行到 INT3 指令时，就表明过程体已经执行完毕，执行完 3 型中断服务程序后，便实现了过程性转移指令的单步跳跃功能。

(2)单步进入的实现。

调用型指令的单步进入实现方法与非调用指令的单步运行方式大体相同，只是在执行软中断指令时不同。因为软中断服务程序在执行的过程中会将 T 标志置 0，禁止了单步中断，所以对于软中断指令的单步进入就不能用此方法来实现，而是将其转化为单步跳跃去处理，即将整个中断服务程序执行完毕后再返回监控程序。

6 结束语

基于远程设备的汇编语言源码调试器采用基于目标文件的实现方案，使用异步通信协议和自行设计的宿主机/目标机通讯协议，具有很大的扩充性，可扩展为多种语言的多机器调试器，这正是下一步的研究内容。

参考文献：

[1] Rosenberg J. How Debuggers Work[M]. New York: Wiley Computer Publishing, 1996.

[2] 仇玉章. 32 位微型计算机原理与接口技术[M]. 北京：清华大学出版社，2000.

[3] Lablance T, Mellor－Crummey J M. Debugging parallel programs with instant replay[J]. IEEE Trans Comput, 1987,36(4):471－482.

[4] 王亚磊，姚放吾，罗 威，等. 基于 386EX 的嵌入式软件调试器的设计与实现[J]. 计算机技术与发展，2007,17(s):264－266.

[5] 郝凤琦，程广河，张让勇. 嵌入式通讯协议栈的构件化研究[J]. 计算机技术与发展，2006,16(10):72－74.

[6] 曲玉昭，陆华奇，于忠清 基于事件和周期任务的嵌入式系统设计[J]. 计算机技术与发展，2006,16(10):175－177.

[7] 陈定君，郭晓东. 嵌入式软件仿真开发系统的研究[J]. 电子学报，2000,28(3):137－139.

[8] Azzerini B, Lopriore L. Program debugging environments: design and utilization[M]. New York: Ellis Horwood, 1992.

(上接第 3 页)

果图，分别对应植被、干草落叶、牧场和高速路。

从图 3 可以看出，文中算法在降低数据量的同时，目标检测几乎未受到影响。由于原始数据受到噪声的影响，目标检测的效果并不理想，而文中算法利用 PCA 降维后，去除了噪声了影响，在一定程度上提高了检测性能。在 bpp=0.5 时，目标检测结果仍优于原始图像，验证了算法的有效性。

3 结束语

针对高光谱图像目标检测的后续应用，提出了面向目标检测的高光谱图像压缩算法。从实验结果来看，利用主成分进行目标检测的效果要优于原始图像的检测效果，并且压缩对主成分的影响较小，这有利于保持图像中的目标信息。在实际应用中，可以在编码端直接对主成分进行目标检测，将检测结果进行压缩传输即可。目前，独立分量分析也广泛地应用于高光谱图像的目标提取，下步研究可将独立分量分析用于高光谱图像的压缩，进一步提高小目标的检测质量。

参考文献：

[1] 童庆喜，张 兵，郑芬兰. 高光谱遥感——原理、技术与应用[M]. 北京：高等教育出版社，2006.

[2] 孙 蕾，罗建书，谷德峰. 基于谱间预测和码流预分配的高光谱图像压缩算法[J]. 光学精密工程，2008,16(4): 752－757.

[3] Lee H S, Younan N H, King R K. Hyperspectral image cube compression combining JPEG－2000 and spectral decorrelation[C]//Proceedings of the International Geoscience and Remote Sensing Symposium. Toronto, Canada: [s. n.], 2002: 3317－3319.

[4] 吴家骥，吴成柯. Karhunen－Loeve 和小波变换的多光谱图像三维集合嵌入块编码压缩算法[J]. 电子与信息学报，2005,27(8): 1244－1247.

[5] Du Q, Fowler J E. Hyperspectral image compression using JPEG2000 and principal component analysis[J]. IEEE Geoscience and Remote Sensing Letters, 2007,4(2):201－205.

[6] Chang C I, Du Q. Estimation of number of spectrally distinct signal sources in hyperspectral imagery[J]. IEEE Trans. on Geoscience and Remote Sensing, 2004,42(3):608－619.

[7] 吴 波，张良培，李平湘. 非监督正交子空间投影的高光谱混合像元自动分解[J]. 中国图像图形学报，2004,9(11):1392－1396.

[8] 耿修瑞，赵永超. 高光谱遥感图像小目标探测的基本原理[J]. 中国科学，2007,37(8):1081－1087.