COMET 虚拟机的设计与实现

柴树杉，刘福江，陈创，丁亮，田华
（中国地质大学 信息工程学院, 武汉 430074）

摘　要: COMET 虚拟机是一种为了教学目的而定义的虚拟机，其结构虽然简单，但功能完备，可作为一种教学辅助工具，帮助大学本科学习计算机课程的学生更好地理解现在计算机的结构和工作原理。作者基于原有 COMET 虚拟机定义，改进并实现了一个功能相对完备的虚拟机。COMET 虚拟机的开发主要突出两个原则：简单性和可移植性。本文将对 COMET 虚拟机的开发细节做一个完整的描述。

关键词: COMET 虚拟机; 设计; 实现

COMET Virtual Machine Design and Implementation

CHAI Shushan, LIU Fujiang, CHEN Chuang, DING Liang, TIAN Hua
(Faculty of Information Engineering, China University of Geosciences, Wuhan, 430074)

Abstract: This paper describes the design and implementation of COMET virtual machine. The COMET virtual machine implements an simple but full featured imaginary register based computer, which can be used as an assistant tool helping college level students understanding the structures and principles of the modern computers. The author improved and implemented complete virtual machine based on the original COMET virtual machine. The design of COMET virtual machine aims for two goals: simplicity and flexibility. This article will give complete development procedure of the COMET virtual machine.

Keywords: COMET virtual machine; design; implementation

1 引言

虚拟机就是一台假想的机器。一台虚拟机与一台真实存在的计算机不同之处在于前者只是一个技术规范。虚拟机技术从很早就开始研究，并且在软件技术中广为应用。目前，高度可移植的JAVA编程语言采用的就是虚拟机技术。虚拟机不仅有广泛的商业应用价值，同时也可作为一种很好的教学辅助工具。通过虚拟机，学生可以很好地学习理解计算机的结构和工作原理。因此，设计并实现一个简单的虚拟机有很好的教学意义[4，5]。

COMET虚拟机开始只是作为CASL汇编语言的运行平台而设计，并由中国计算机软件专业技术资格和水平考试大纲中给出的CASL汇编语言文本定义（下文将简称为文本）。作者在文本的基础上，对COMET虚拟机的功能进行了一定的改进和扩充，并实现了一个功能完善的虚拟机实例[1]。

2 COMET 虚拟机设计

2.1 逻辑结构

COMET 虚拟机是一台基于寄存器模型结构的计算机。COMET 虚拟机系统主要包括三个部分：中央处理器（CPU）、存储器和出入输出设备。COMET 计算机的逻辑结构如图 1 所示。

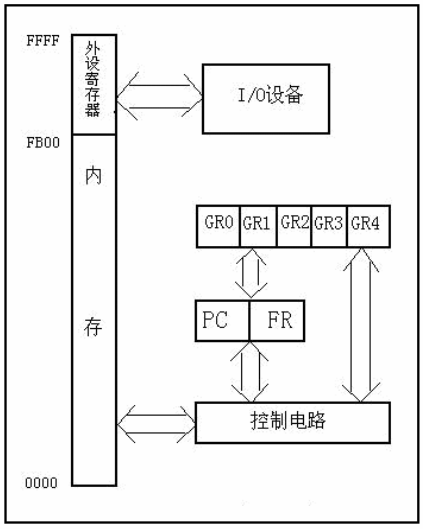

图 1 COMET 虚拟机结构

CPU是由寄存器、算术逻辑部件和控制

逻辑组成。寄存器在程序运行时存储计算过程的各种信息；算术逻辑部件的功能是进行所有的算术和逻辑运算。由于COMET计算机是虚拟的计算机，因此，它的控制逻辑对于我们是透明的[1, 4, 5]。

2.2 寄存器

COMET 虚拟机有 5 个通用寄存器 GR（16 位），一个指令计数器 PC（16 位）和一个标志寄存器 FR（2 位）。

GR（通用寄存器） 5 个通用寄存器的编号为 0、1、2、3、4，分别记为 GR0、GR1、GR2、GR3、GR4。这些通用寄存器用于算术、逻辑、移位等运算，其中 GR1、GR2、GR3、GR4 通用寄存器兼做变址寄存器。另外，GR4 还兼做栈指针（SP）用，栈指针是存放栈顶地址用的寄存器。

PC（指令计数器） 在执行指令的过程中，PC 中存放着正在执行的指令的第一个字的地址。当指令执行结束时，一般是把 PC 的内容加上当前指令的长度（不同指令的长度不同），只有在执行转移指令且条件成立时，才将转移指令地址置入 PC 中。

FR（标志寄存器） 在 ADD、SUB、MUL、DIV、MOD、AND、OR、EOR、CPA、CPL、SLA、SRA、SLL、SRL、LEA 等指令执行结束时，根据执行的结果，将 FR 设置成 00、01 或 10（大于、等于、小于或负数、零、正数）。它们不会因其他指令的执行而改变。

2.3 存储器

COMET虚拟机是一台 16 位的定点计算机，主存储器的基本存储单位是字，总容量为 65536（2^{16}=16k）字，各个字的地址按照 0000-FFFF（十六进制）编号。一个字的 16 位二进制采用从左到右次序编号，如图 2。

图 2 字节顺序

COMET虚拟机可以处理三种数据，即字符数据、带符号整数、地址数据。字符数据采用 16 位的Unicode字符；带符号整数采用二进制补码表示，可表示的数值范围为[-2^{15}，2^{15}-1]；地址数据为无符号的整数，范围是[0，2^{16}-1]。

在实现的COMET虚拟机中，内存的高1k字保留用于各种专用的外设寄存器[1, 3]。

2.4 指令系统

COMET 虚拟机和文本相比，新增加了 HALT、MUL、SUB、MOD 四条指令。下面只给出 HALT、MUL、SUB、MOD 四条指令的说明，其他的指令可以参考文本。

HALT 指令执行停机操作，占一个字长。MUL 指令执行乘法操作，操作规则和 ADD 等指令相似，占 2 个字长。SUB 指令执行除法操作，操作规则和 ADD 指令相似，占 2 个字长。MOD 指令执行取模操作，操作规则和 ADD 指令相似，占 2 个字长。

2.5 指令格式

COMET 虚拟机机器指令一般包含三种信息：OP，GR，XR，ADR。其中 OP 为机器指令的编号，对应第一个字的[0-7]位；GR 为通用寄存器编号，对应第一个字中的[8-11]位；XR 为变址寄存器编号，对应第一个字的[12-15]位，[12-15]位为 0 表示没有变址（GR0 不能作为变址寄存器）；ADR 为操作数，对应地二个字的[0-15]位。如果一个指令不含某种信息（例如，没有 ADR），则忽律该字段。OP、GR、XR、ADR 的存储细节如图 3 所示。

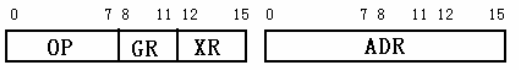

图 3 指令格式

根据指令操作数据的类型的不同，可将 COMET 虚拟机的机器指令分为四种：寄存器-存储器型，存储器型，寄存器型，空操作数型。如果一个 COMET 虚拟机指令不涉及主存储器操作（无 ADR），则为 1 个字长，其他均为 2 个字长。

2.6 字节码格式

字节码文件是COMET虚拟机的可执行文件。在启动虚拟机时，由命令行中的参数给出字节码文件名。字节码文件含有程序的长度、装载位置、程序指令等信息。虚拟机依据这些信息，将字节码中的指令从二级存储器装载到相应的虚拟机内存。这一系列操作和操作系统的程序加载器的工作情况类似[4]。

COMET 虚拟机的字节码文件是一种二

进制格式文件，格式如图 4 所示。

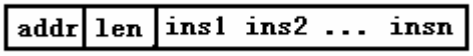

图 4 字节码结构

其中 addr 为字节码在虚拟机内存的装载地址，len 为字节码中指令的个数（不包含 len 和 addr），ins1、ins2、…、insn 为要载入虚拟机的指令。

2.7 输入输出设备

文本中并没有明确定义COMET虚拟机的输入输出设备，只是给出IN、OUT几个IO操作宏指令。在这里，我们采用将外部设备寄存器在内存中统一编址的方法来处理输入和输出设备[1, 3]。

COMET 虚拟机默认的 I/O 设备是键盘和显示器。利用操作系统提供的从定向功能用户也可以将它们重新定向到其他的设备。

COMET 虚拟机输入输出设备的两个专用寄存器为：IO_ADDR、IO_FLAG，分别对应 0xFD100、xFD10 地址内存。IO_ADDR 用于保存要传输数据的内存地址；IO_FLAG 为 IO 设备的标志位，其 8-15 位是要传输数据的个数(0 表示无 IO)，7 位表示输入或输出(1 表示输入，0 为输出)，6 位在出现 IO 错误时设置，3-5 位为传输的类型(有字符、八进制、十进制、十六进制等)，0-2 位保留。

2.8 调试器

COMET调试器是一个内嵌在虚拟机里的机器级的调试器。当需要调试一个COMET虚拟机的程序时，只需要在启动COMET虚拟机的时给出相应的命令参数就启动调试功能了[2, 4]。

COMET 调试器的基本功能有：显示帮助（help），运行程序直到停止（go），分步执行（step n），跳转程序（jump），显示寄存器内容（regs），显示内存数据（dMem），显示内存指令（iMem），修改内存数据（alter），遍历指令（trace），指令记数功能（print），重新装载字节码（clear），退出调试器（quit）。每个调试命令的具体用法可以参考 COMET 虚拟机的帮助文件。

3 COMET 虚拟机实现

3.1 虚拟机数据结构

```
struct comet
{
    off_t pc;
    short fr;
    short gr[5];
    short mem[MEMSIZE];
} cmt;
```

虚拟机结构变量 cmt 是一个全局变量，成员分别为：指令计数器（pc）、标志寄存器（fr）、通用寄存器（gr）、存储器（mem）。将cmt设计为全局变量的优点是个函数不用传递复杂的结构体参数，缺点是每个进程同时只能有一个虚拟机实例。

3.2 主函数

```
int
main(int argc, char *argv[])
{
    init(argc, argv);
    if(debug) comet_debug();
    else while(comet_step());
    fclose(source);
    return 0;
}
```

函数init首先初始化COMET虚拟机并装载字节码，如果发生错误则停止。然后根据调试器状态，选择运行虚拟机的方式。如果调试开关（debug）被设置，则调用comet_debug 函数在调试状态下运行COMET虚拟机。如果没有打开调试开关，则循环调用单步执行函数comet_step，直到程序结束[1, 2]。

3.3 字节码载入

```
void
comet_load(void)
{
    unsigned short n, flag[2];
    fseek(source, 0, SEEK_SET);
    n = fread(flag,
        sizeof(off_t), 2, source);
    n = fread(&cmt.mem[flag[0]],
        sizeof(off_t),tmp[1],source);
    /* 其他处理代码 */
}
```

变量 n 用于记录读取字节码的数目，如

果n小于相应的值，则发生字节码装载错误。变量 flag 用于保存字节码装载信息，分别字节码装载地址和字节码大小。

COMET字节码设计比较简单，也存在很多不足。例如，没有标志文件格式的魔数，没有更完善的错误检测措施。我们的目的是让读者了解字节码的工作原理，因此只给出了一种最简单的实现[1, 2, 4]。

3.4 指令解析

指令的解析一般包含这个几个过程：取指令，解码，执行。其中解码对虚拟机的执行效率有很大的影响[5]。这里采用下标索引技术来解码指令。具体代码如下：

```
void comet_ld(void);      /* LD 指令*/
void comet_ld(void);      /*ST 指令 */
void comet_ld(void);      /* LEA 指令 */
/* 其他指令函数声明 */
int comet_step(void)
{
    static void (*comet_op)() = {
        comet_ld, comet_st, comet_lea,
        /* 其他指令执行函数指针 */
    };
    /* 解析指令，存放在 op 中 */
    short op = get_op();
    /* 执行 op 对应的代码 */
    (*comet_op [op])();
    /* 返回执行状态 */
    return val;
}
```

例如，有指令 LEA，其对应的机器码为031，那么将通过函数指针数组 comet_op 直接定位到（*comet_op[031]）函数，即并执行相应的 comet_lea 函数。

3.5 输入输出设备

COMET虚拟机在解析每个指令前，先读取IO设备状态寄存器IO_FLAG中的值，如果IO_FLAG被设置，则执行相应的IO操作[3]。具体代码如下：

```
void
comet_io(void)
{
    switch(cmt.mem[IO_FLAG]&IO_TYPE)
    {
        case IO_NULL:
            /* 无 IO 操作 */
        case IO_OCT & IO_IN:
            /* 八进制输入 */
        case IO_DEC & IO_IN:
            /* 十进制输入 */
        case IO_HEX & IO_IN:
            /* 十六进制输入 */
        case IO_OCT & IO_OUT:
            /* 八进制输出 */
        case IO_DEC & IO_OUT:
            /* 十进制输出 */
        case IO_HEX & IO_OUT:
            /* 十六进制输出 */
        default:
            /* 未知 IO 类型 */
    }
    /* 重置 IO 状态寄存器 IO_FLAG */
}
```

3.6 调试器

调试程序是建立和单步执行的 COMET 虚拟机之上的。当没有打开调试功能时，循环执行 COMET 虚拟机字节码程序，直到停止。当打开了调试功能时，调试函数 debug 根据调试命令，执行相应步的指令、显示或操作相关的数据。

```
/* 各种调试命令 */
typedef enum
{
    GO, STEP, JUMP, REGS,
    IMEM, DMEM, ALTER,
    TRACE, PRINT, CLEAR,
    HELP, QUIT
} DbType;
/* 调试函数 */
void
comet_debug(void)
{
    int cmd;    /* 保存调试命令 */
    while(1) {
        /* 读调试命令 */
        switch(cmd) {
            case HELP: /* 帮助文件 */
            case GO  : /* 执行程序 */
```

```
            case STEP: /* 分步执行 */
            /* 其他调试命令 */
            default  : /* 未知命令 */
        }
    }
}
```

调试函数comet_debug根据不同的调试命令执行相应的操作，并显示虚拟机状当前的状态信息。

4 运行虚拟机

下面通过一个求（1 + 2 + … + n）的程序，来介绍其在 COMET 虚拟机上的执行的流程。程序的字节码由相关的工具生成，保存为 sum.comet 文件（后缀为 comet）。

4.1 普通运行

输入命令：comet sum

在 COMET 虚拟机获得 sum 参数后，会自动识别为 sum.comet 字节码文件。

输入 100，表示求 1+2+…+100 的和。

COMET 虚拟机输出：

```
===============
COMET 虚拟计算机
===============

100
5050[root@knuth /root]#
```

表示 1+2+…+100 等于 5050，结果正确。

4.2 调试运行

输入命令：comet –d sum

```
================
COMET 虚拟计算机
================

调试 （帮助输入 help）...

输入命令: t
指令显示功能打开
输入命令: p
指令计数功能打开
输入命令: r
显示寄存器数据
GR[0] =      0      PC =      0
GR[1] =      0      SP = fc00
GR[2] =      0      FR =     01
GR[3] =      0
输入命令: d
显示内存数据
mem[0     ] = 1200
输入命令: i
显示内存指令
mem[0]: JMP   5
输入命令:g
mem[0]: JMP   5
mem[5]: ST    GR0, fe00
mem[7]: PUSH fe00
mem[9]: LEA   GR0, 4
mem[b]: ST    GR0, fd10
mem[d]: LEA   GR0, c01
mem[f]: ST    GR0, fd11
mem[11]: POP   GR0
3
mem[12]: LD    GR0, 4
mem[14]: ST    GR0, 2
……
6mem[4f]: JMP 51
mem[51]: HALT
执行指令数目 = 71
输入命令: q
退出调试...
```

其中 t、p、d、r、g 分别是调试命令 trace、print、dMem、regs、go 的缩写；3（mem[11] 的下一行）是用户输入的数据，不是调试命令。6（mem[4f]行）表示 1+2+3 的和为 6。

COMET 虚拟机内嵌的调试器功能非常强大，熟练掌握后对 COMET 虚拟机的程序开发会有很大的帮助。调试命令细节可以通过帮助命令 help 获得。

5 结语

本文给出了COMET虚拟机的硬件设计方案。该方案阐述了虚拟机各个基本部分的详细设计思路及需要注意的问题。目前已经实现了COMET虚拟机、CASL汇编器和TINY编译器。我们将基于COMET虚拟机开发一个小型的文件系统，使得其应用于实际教学任务[1, 4, 5]。

参考文献

[1] CASL 汇编语言文本[M].中国计算机软件专业技术资格和水平考试大纲.
[2] (美)Kenneth C. Louden. 编译原理及实践[M].北京:机械工业出版社,2003.
[3] (澳)John Lions.莱昂氏 UNIX 源代码分析[M].北京:机械工业出版社,2000.
[4] (美)Bill Blunden.虚拟机的设计与实现——C/C++[M].北京:机械工业出版社,2003 .
[5] 王亚平,陈甫舟.基于 Linux 平台上的 BOST 虚拟机的硬件设计与实现[J].现代电子技术.2004,15:8-10.

作者简介：
柴树杉（1983-），男，江苏沭阳县人，中国地质大学 2002 级地理信息系统本科应届毕业生。
刘福江（1973-），男，博士研究生，河南柘城，讲师，研究方向为 GIS 技术、遥感技术及其应用。