林州電子科找大学

计算机组成原理(甲)实验报告

学	院	网络空间安全学院
专	业	网络工程
班	级	19272401
学	号	19061440
学生姓名		F001
教师姓名		袁理峰
完成日期		2020.12.18
成	绩	

实验六 MIPS 汇编器与模拟器实验

一、实验目的

- (1) 学习 MIPS 指令系统,熟悉 MIPS 指令格式及其汇编指令助记符,掌握机器指令编码方法。
- (2) 学习 MIPS 汇编程序设计,学会使用 MIPS 汇编器将汇编语言程序翻译成二进制文件。
- (3) 了解使用 MIPS 教学系统模拟器运行程序的方法。

二、实验原理

计算机硬件只能识别 0、1 代码,因为电子器件只能在电信号下才能工作,能让及 其理解的最简单的信号就是"开"和"关"信号。计算机硬件设计者设计出的计算机, 对计算机设计者的命令言听计从,计算机设计者的"言"就是机器指令,"命令"就 是机器指令。换言之,机器指令就是能被计算机硬件识别并直接执行的 0、1 代码串。

使用 0、1 代码编写机器语言程序,不仅烦琐枯燥,而且要求程序员对机器指令格式和硬件了如指掌。为了将机器指令用人类容易理解的符号来表达, 出现了汇编语言——用助记符来表示机器指令,这种指令又叫汇编指令。显而易见,汇编语言程序虽然易于程序员理解,但是计算机硬件并不能识别。因此,需要将汇编语言程序翻译成计算器语言程序,能完成这种翻译功能的程序叫汇编程序,又叫汇编器。汇编程序将用符号表示的汇编指令翻译成二进制的机器指令。

虽然汇编语言对程序员来说是一个巨大的进步,但是他仍然是面向机器的语言,程序员需要按照机器的思维进行变成。高级程序设计语言的出现突破了汇编语言的局限性。高级语言更接近人类的思维方式,他是面向用户的自然语言,使得程序员的编程效率更加高效,并且可以执行使得程序独立于开发环境。高级语言程序通过编译器翻译成为二进制的机器语言程序。

三、 实验环境

所用电脑的软硬件配置:自己的笔记本电脑、Windows10操作系统实验所用的软件: PCSpim

四、主要操作步骤及实验结果记录(不能光截图,要有相应的文字说明)

(对实验过程中的主要操作步骤进行描述,并随时记录实验过程中观察到的结果,必要时可辅助截图) 任务一:使用记事本程序或者任何纯文本编辑器编辑 test.asm 文件,输入相关内容。

(编程五分钟,注释两小时==)

III test.asm - 记事本 文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H) main:li \$v0,5 #main表示主程序部分开始, li表示将一常量 (后面的数) 放入指定寄存器中 (前面的) v0寄存器常用于控制 #表示暂停,在v0为5的情况下表示读入一个数,并存在v0中,1输出,5输入,10暂停 svscall move \$t0,\$v0 #move表示把后面哪个寄存器里面的值挪到前面的寄存器里,这里把v0,即刚刚输入的值放到t0里 #v0再存5 li \$v0,5 syscall #懂得都懂,我再输入move \$t1,\$v0 #懂得都懂,我再存一个t1 bgt \$t0,\$t1,t0_bigger #bgt表示分支,若t0>t1,则执行第三个参数表示的地方里的东西,不然就执行下面的东西 move \$t2,\$t1 #跑到这儿就是指t1<=t0了, 然后把t1的值给t2 b endif #这个if就结束了 t0 bigger:move \$t2,\$t0 #如果是,就跑到这里执行,然后把t0的值给t2 endif:move \$a0,\$t2 #分支结束了,继续干活。把t2的值给a0 li \$v0,1 #v0给个1 syscall #输出了,默认输出a0的东西 #不是很清楚,好像是把ra的地址给pc,相当于返回地址/相关值 ir \$ra: #打印时满足的条件: v0=1, 且只打印a0 任务二:运行 PCSpim 程序,在其中打开 test.asm,先连续执行,输入起始地址 0x0040000,再单步执 行,按照需要再控制台输入2个数据,观察结果。 第一次输入数据故意输入了一个特别大的数,发现其作为整数最多存 9999999999999999999999999 储32位有符号整数,使其溢出了,这是正常情况。 2147483647 Console 第二次输入故意再第一组数据中间加了空格,发现其自动忽略了空格 50 28 之后的数据,与 c 语言中的 scanf 很像。此时返回结果也是正常的。 20 50 设置了断点之后重新输入数据,发现结果错误,返回了 38。经过询 S Console 问之后得知是某个参数没有配好,但是具体是什么尚不清楚。之后尝试了 38 几次之后均是返回第一个参数。在一步步进行步进的过程中发现,在判断 250 语句是正常的,之后的 move 语句没有正常执行,仅这一条语句错误,其 38 余均正常。 任务三: 使用记事本程序或者任何纯文本编辑器编辑 R CPU Test.asm 文件,输入相关内容。 ■ R CPU Test.asm - 记事本 文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H) nor \$1.\$0.\$0: #\$1=FFFF FFFF 没什么好说的, 这边全部都是对于其相关的预测值。而且均为三目运算符, 均是将后两个值进行运算后返回给最前面的值。 sltu \$2.\$0.\$1: #\$2=0000 0001 nor是或非运算,add是加法运算,sltu的意思为如果R2的值小于R3,那么设置R1的值为1,否则设置R1的值为0 add \$3.\$2.\$2: #\$3=0000 0002 sllv是第二个参数向左移动第三个参数位,结果保留在第一个参数里 add \$4.\$3.\$2: #\$4=0000 0003 add \$5,\$4,\$3; #\$5=0000 0005 xor是异或运算, or是或运算, and是与运算 add \$6,\$5,\$3; #\$6=0000 0007 sllv \$7,\$6,\$2; #\$7=0000 000E add \$9,\$5,\$6; #\$9=0000 000C sllv \$8,\$6,\$9; #\$8=0000 7000 xor \$9,\$1,\$8; #\$9=FFFF_8FFF0 add \$10,\$9,\$1; #\$10=FFFF 8FFE sub \$11,\$8,\$7; #\$11=0000 6FF2 sub \$12,\$7,\$8; #\$12=FFFF 900E and \$13,\$9,\$12; #\$13=FFFF_800E or \$14,\$9,\$12; #\$14=FFFF_9FFF or \$15,\$6,\$7; #\$15=0000_000F nor \$16,\$6,\$7; #\$16=FFFF_FFF0

任务四: 再 PCSpim 程序中打开 R_CPU_Test.asm, 单步执行, 记录执行结果。

add \$17,\$7,\$3; #\$17=000_0010 sllv \$18,\$8,\$17; #\$18=7000_0000 sllv \$19,\$3,\$17; #\$19=0002_0000 sllv \$20,\$19,\$7; #\$20=8000_0000 add \$21,\$20,\$1; #\$21=7FFF FFFF

```
(r0) = 00000000
                    R8
                        (t0) = 00007000
                                         R16 (s0) = fffffff0
                                                              R24 (t8) = 00000000
R1
          ffffffff
                    R9
                        (t1) = ffff8fff
                                         R17
                                             (s1) = 00000010
                                                              R25
                                                                  (t9)
                                                                        00000000
    (at) =
                    R10 (t2) =
                               ffff8ffe
                                         R18 (s2) = 70000000
                                                                        00000000
R2
    (v0) = 00000001
                                                                  (k0)
                                                              R26
    (v1) = 00000002
                    R11 (t3) = 00006ff2
                                         R19 (s3) = 00020000
R3.
                                                              R27
                                                                        00000000
                                                                  (k1)
    (a0) = 00000003
                                                                      = 00000000
R4
                    R12
                        (t4)
                            = ffff9NNe
                                         R20
                                             (s4) = 80000000
                                                              R28
                                                                  (gp)
R5.
    (a1) = 000000005
                    R13 (t5) = ffff800e
                                         R21 (s5) = 000000000
                                                              R29
                                                                      = 7ffff664
                                                                  (sp)
        = 00000007
                        (t6) = ffff9fff
                                         R22
                                             (s6) = 00000000
                                                              R30
                                                                  (s8) = 00000000
R6
    (a2)
                    R14
                    R15 (t7) = 00000000f
                                         R23 (s7) = 000000000
    (a3) = 0000000e
                                                             R31 (ra) = 000000000
这是全部的处理结果,当文件执行到以下这一行时出现了错误:
             0x0281a820 add $21, $20, $1
                                                   ; 22: add $21,$20,$1; #$21=7FFF_FFFF
经分析得, r1 已经是 ffffffff 了, 此时再进行加法操作, 会溢出, 因此报错, 自动跳转到 kernel 中进行
计算。
```

五、 实验分析总结及心得

(结合所学知识对实验过程中观察到的实验结果进行分析总结,以便加深对知识的理解,并总结通过实验学到的知识或技术)

我认为,之前学的 ISE 对于实际理解计算机组成原理的帮助不大。直到这一章的 PCSpim,才算是有趣了不少。汇编语言可以说是计算机的灵魂,它是人和机器沟通的最快的桥梁。因此,对于 PCSpim 的学习尤为必要。

在这章中,我查阅了相关博客,了解了每一条指令的原理以及应用方式。对于实验中产生的奇奇怪怪的现象,也进行了相应的错误分析与调试。我认为这样是最好的学习方式,我也希望有机会一直学习汇编语言。在实验结果上,两个实验都不算成功。第一个实验错误位置,但是表现在分支结构后的 move语句出现了问题,查阅后是某个参数配置错误,但是目前尚未找到解决办法。第二个实验则是给定的程序文件存在问题。出现了溢出的错误。因此之后的程序直接被忽略。

```
(r^0) = 00000000
                           (t0) = 00007000
R0
                                             R16 (s0) = fffffff0
                                                                   R24 (t8) = 000000000
                      P8
R1
    (at)
         = ffffffff
                      R9
                           (t1) =
                                  ffff8fff
                                             R17
                                                 (s1)
                                                     = 00000010
                                                                   R25
                                                                       (t9) = 00000000
R2
    (v0) = 00000001
                      R10 (t2) =
                                  ffff8ffe
                                             R18 (s2) =
                                                        70000000
                                                                   R26
                                                                       (k0) =
                                                                               00000000
R3
                      R11 (t3) =
    (v1)
         = 00000002
                                  00006ff2
                                            R19 (s3)
                                                        00020000
                                                                   R27
                                                                       (k1)
                                                                               00000000
R4
    (a0) = 00000003
                      R12 (t4) =
                                  ffff900e
                                            R20 (s4) =
                                                        80000000
                                                                   R28 (gp) =
                                                                               00000000
                      R13 (t5) = ffff800e
R14 (t6) = ffff9fff
    (a1) = 00000005
                                            R21 (s5) = 00000000
                                                                   R29 (sp) = 7ffff664
R5
    (a2) = 00000007
IR6
                                            R22
                                                 (s6) = 000000000
                                                                   R30 (s8) = 00000000
    (a3) = 0000000e R15 (t7) = 0000000f R23 (s7) = 00000000
                                                                   R31 (ra) = 000000000
```

上述的全部内容均出现在这张图片里。

我将会继续关注指令以及相关汇编语言。