计算机体系架构　LAB 01

范云潜 18373486

微电子学院 184111 班

日期：2020 年 10 月 9 日

作业内容：Lab01

目录

`.data` 表明使用了一个小节专门用于存储程序的数据。(.data Subsequent items stored in Data segment at next available address)

`.word` 表明定义的数据是以一个字（word ）为单位的，在这里定义了一个字大小对应的内存位置，存储了数字 9 ，并且可以使用标签 `n` 来引用对应的内存位置。(.word Store the listed value(s) as 32 bit words on word boundary)

`.text` 表示代码段开始。(.text Subsequent items (instructions) stored in Text segment at next available address)

SubProblem Breakpoint

在汇编之后，可以在执行顺序旁进行断点添加，类似于 gdb 的 `b` ，如**图 1**。直接运行会运行到下一个断点或直接完成程序，可以使用 Step 模式进行单条指令执行。

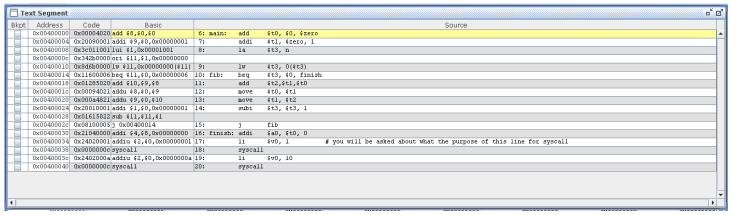


图 1: 断点添加界面

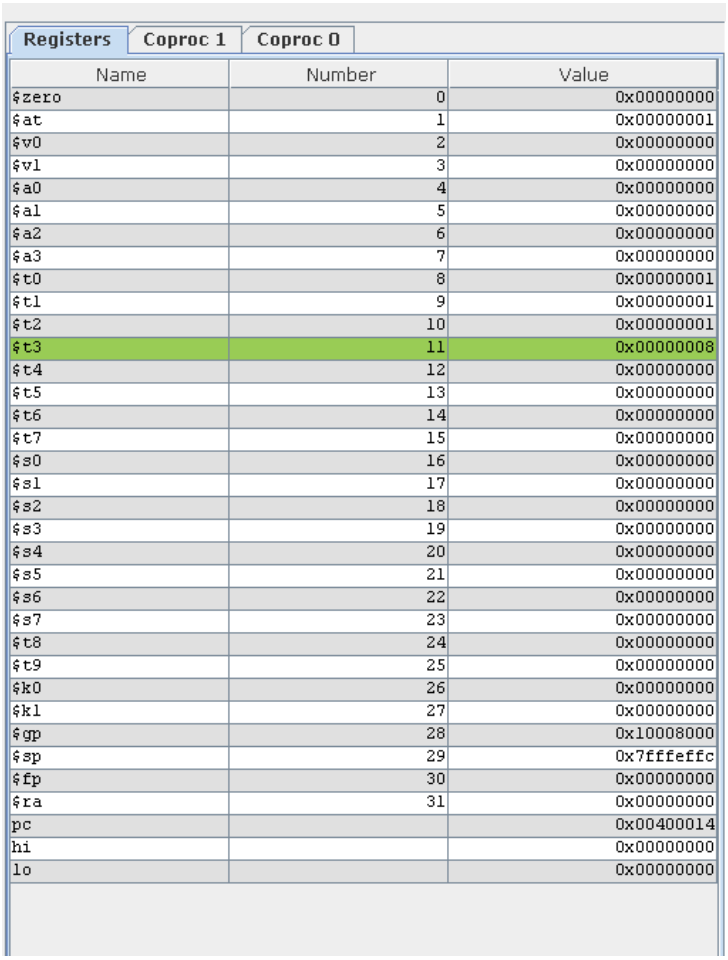
SubProblem Regs

寄存器的值可以从侧面寄存器面板直接获得，如**图 2**，双击即可强制修改。

SubProblem Mem

根据 Data Segment 面板，n 存储到 `0x10010000` 。

SubProblem syscall



Registers | Coproc 1 | Coproc 0

Name	Number	Value
$zero	0	0x00000000
$at	1	0x00000001
$v0	2	0x00000000
$v1	3	0x00000000
$a0	4	0x00000000
$a1	5	0x00000000
$a2	6	0x00000000
$a3	7	0x00000000
$t0	8	0x00000001
$t1	9	0x00000001
$t2	10	0x00000001
$t3	11	0x00000008
$t4	12	0x00000000
$t5	13	0x00000000
$t6	14	0x00000000
$t7	15	0x00000000
$s0	16	0x00000000
$s1	17	0x00000000
$s2	18	0x00000000
$s3	19	0x00000000
$s4	20	0x00000000
$s5	21	0x00000000
$s6	22	0x00000000
$s7	23	0x00000000
$t8	24	0x00000000
$t9	25	0x00000000
$k0	26	0x00000000
$k1	27	0x00000000
$gp	28	0x10008000
$sp	29	0x7fffeffc
$fp	30	0x00000000
$ra	31	0x00000000
pc		0x00400014
hi		0x00000000
lo		0x00000000

图 2: 寄存器面板

对于模拟器提供的 `syscall` 系列的函数，有着详细的文档说明[1]，这里使用的是 1 和 10 号 syscall，分别是 `print_int` 与 `exit`。

为了使用 syscall 需要在 `$vi` 中写入调用标号，之后使用 `syscall`。

Problem Exercise 2

实际上这个代码段只用到了 `add` 指令，分别用来加载寄存器内的值以及进行算数运算。为了验证代码的正确性，使用 python 完成了一个 `gold` 函数用于模拟行为，请见 lab1_ex2.py。

Problem Exercise 3

提炼出的 Bugs 如下：

1. 未对计数器 `$v0` 进行初始化而直接进行计数
2. 指向一个 word 的指针应当以 4 bytes 为单位，而不是 1 byte
3. 由于最后一个数据为 0，不应计数

[1]http://courses.missouristate.edu/kenvollmar/mars/help/syscallhelp.html

Problem Exercise 4

循环段落在，并且为每一行添加了注释进行说明

```
$L3:
    sw      $3,0($4)    # store the read word to 'dest'
    addiu   $6,$6,1     # add 1 to the counter of loop
    lw      $3,4($2)    # read next word of 'source'
    addiu   $4,$4,4     # advance the pointer of 'dest' by a word
    addiu   $2,$2,4     # advance the pointer of 'source' by a word
    bne     $3,$0,$L3   # loop condition: if read a '0' then stop
    nop
```

source 在

```
source:
    .word   3
    .word   1
    .word   4
    .word   1
    .word   5
    .word   9
    .word   0
    .ident  "GCC: (Ubuntu 7.5.0-3ubuntu1~18.04) 7.5.0"
```

dest 被定义为 40 个字节长，4 字节为单位的内存：`.comm  dest,40,4`