计算机体系结构实验

QtSpim 软件

可运行32位MIPS汇编代码的MIPS模拟器

孙晓光

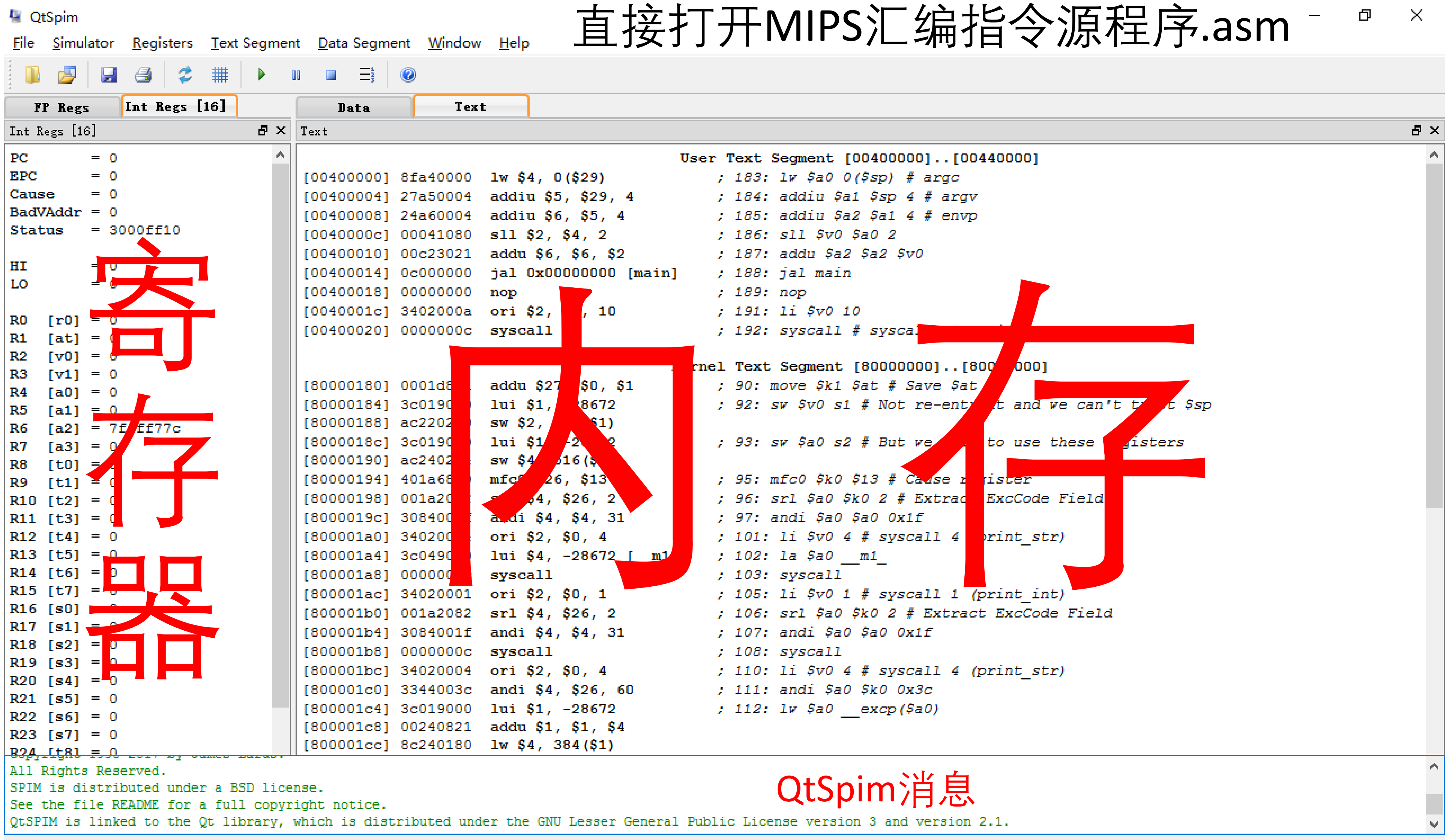
xgsun@fudan.edu.cn

2024-11-10

MIPS 仿真器 QtSpim (简称 Spim)

QtSpim： 支持32位MIPS指令集的MIPS微处理器模拟器。

直接打开MIPS汇编指令源程序.asm


QtSpim
File Simulator Registers Text Segment Data Segment Window Help
FP Regs
Int Regs [16]
Data
Text
寄存器
内存
QtSpim消息

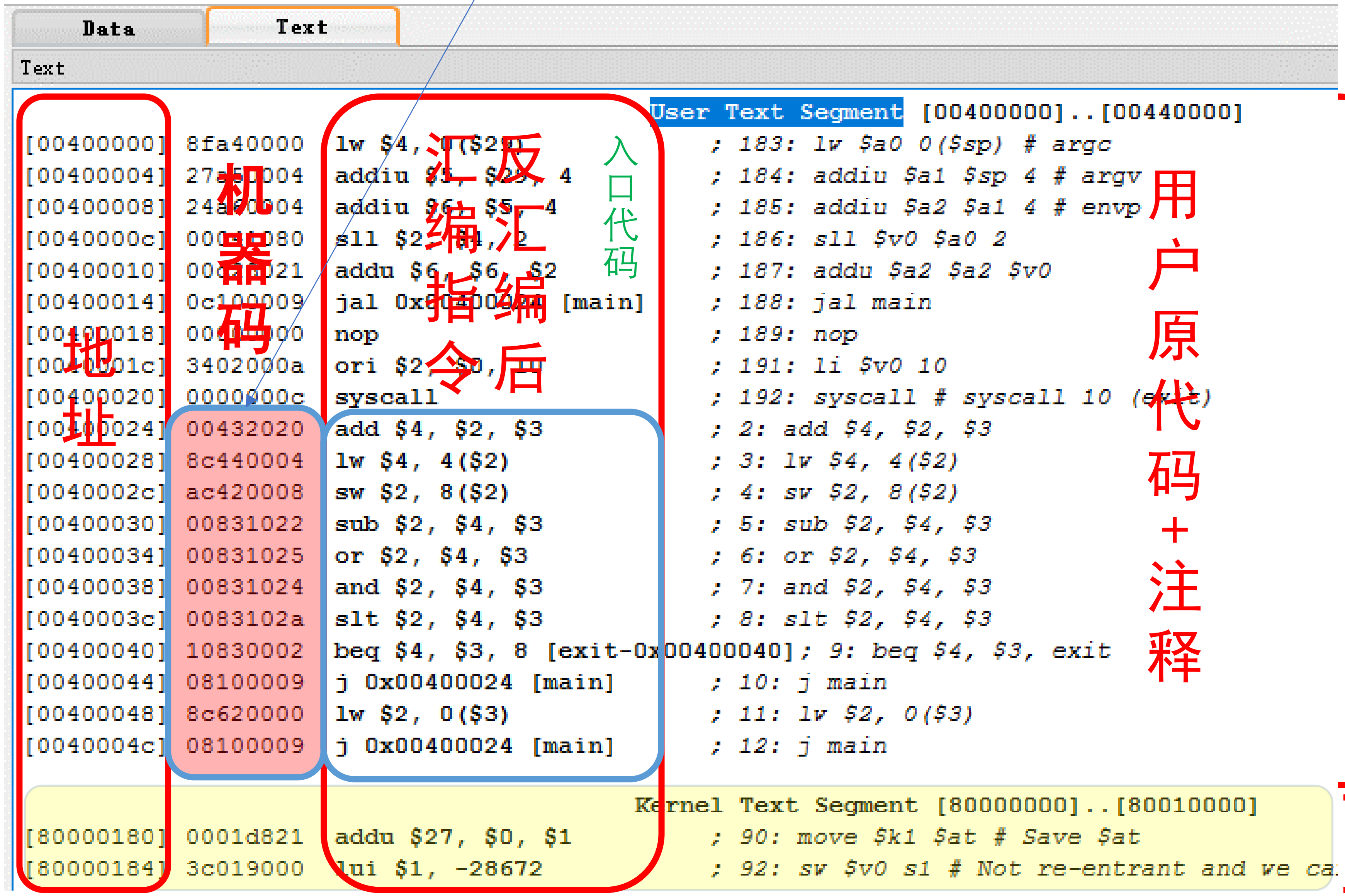
```
Int Regs [16]
PC       = 0
EPC      = 0
Cause    = 0
BadVAddr = 0
Status   = 3000ff10

HI       = 0
LO       = 0

R0  [r0] = 0
R1  [at] = 0
R2  [v0] = 0
R3  [v1] = 0
R4  [a0] = 0
R5  [a1] = 0
R6  [a2] = 7f[illegible]ff77c
R7  [a3] = 0
R8  [t0] = 0
R9  [t1] = 0
R10 [t2] = 0
R11 [t3] = 0
R12 [t4] = 0
R13 [t5] = 0
R14 [t6] = 0
R15 [t7] = 0
R16 [s0] = 0
R17 [s1] = 0
R18 [s2] = 0
R19 [s3] = 0
R20 [s4] = 0
R21 [s5] = 0
R22 [s6] = 0
R23 [s7] = 0
R24 [t8] = 0
```

```
Text
                          User Text Segment [00400000]..[00440000]
[00400000] 8fa40000  lw $4, 0($29)             ; 183: lw $a0 0($sp) # argc
[00400004] 27a50004  addiu $5, $29, 4          ; 184: addiu $a1 $sp 4 # argv
[00400008] 24a60004  addiu $6, $5, 4           ; 185: addiu $a2 $a1 4 # envp
[0040000c] 00041080  sll $2, $4, 2             ; 186: sll $v0 $a0 2
[00400010] 00c23021  addu $6, $6, $2           ; 187: addu $a2 $a2 $v0
[00400014] 0c000000  jal 0x00000000 [main]     ; 188: jal main
[00400018] 00000000  nop                       ; 189: nop
[0040001c] 3402000a  ori $2, [illegible], 10   ; 191: li $v0 10
[00400020] 0000000c  syscall                   ; 192: syscal[illegible]

                       [illegible]rnel Text Segment [80000000]..[800[illegible]000]
[80000180] 0001d8[illegible]  addu $27[illegible]$0, $1   ; 90: move $k1 $at # Save $at
[80000184] 3c019[illegible]  lui $1, [illegible]8672     ; 92: sw $v0 s1 # Not re-ent[illegible]t and we can't t[illegible]t $sp
[80000188] ac2202[illegible]  sw $2, [illegible]$1)
[8000018c] 3c019[illegible]  lui $1[illegible]-2[illegible]2     ; 93: sw $a0 s2 # But we [illegible] to use these [illegible]gisters
[80000190] ac2402[illegible]  sw $4[illegible]16($[illegible]
[80000194] 401a68[illegible]  mfc[illegible]26, $13       ; 95: mfc0 $k0 $13 # Ca[illegible]se r[illegible]ister
[80000198] 001a20[illegible]  [illegible]$4, $26, 2       ; 96: srl $a0 $k0 2 # Extra[illegible] ExcCode Field
[8000019c] 308400[illegible]  a[illegible]di $4, $4, 31    ; 97: andi $a0 $a0 0x1f
[800001a0] 34020[illegible]  ori $2, $0, 4             ; 101: li $v0 4 # syscall 4 [illegible]rint_str)
[800001a4] 3c049[illegible]  lui $4, -28672 [__m1[illegible]  ; 102: la $a0 __m1_
[800001a8] 00000[illegible]  syscall                   ; 103: syscall
[800001ac] 34020001  ori $2, $0, 1             ; 105: li $v0 1 # syscall 1 (print_int)
[800001b0] 001a2082  srl $4, $26, 2            ; 106: srl $a0 $k0 2 # Extract ExcCode Field
[800001b4] 3084001f  andi $4, $4, 31           ; 107: andi $a0 $a0 0x1f
[800001b8] 0000000c  syscall                   ; 108: syscall
[800001bc] 34020004  ori $2, $0, 4             ; 110: li $v0 4 # syscall 4 (print_str)
[800001c0] 3344003c  andi $4, $26, 60          ; 111: andi $a0 $k0 0x3c
[800001c4] 3c019000  lui $1, -28672            ; 112: lw $a0 __excp($a0)
[800001c8] 00240821  addu $1, $1, $4
[800001cc] 8c240180  lw $4, 384($1)
```

```
Copyright 1990-2017 by James Larus.
All Rights Reserved.
SPIM is distributed under a BSD license.
See the file README for a full copyright notice.
QtSPIM is linked to the Qt library, which is distributed under the GNU Lesser General Public License version 3 and version 2.1.
```

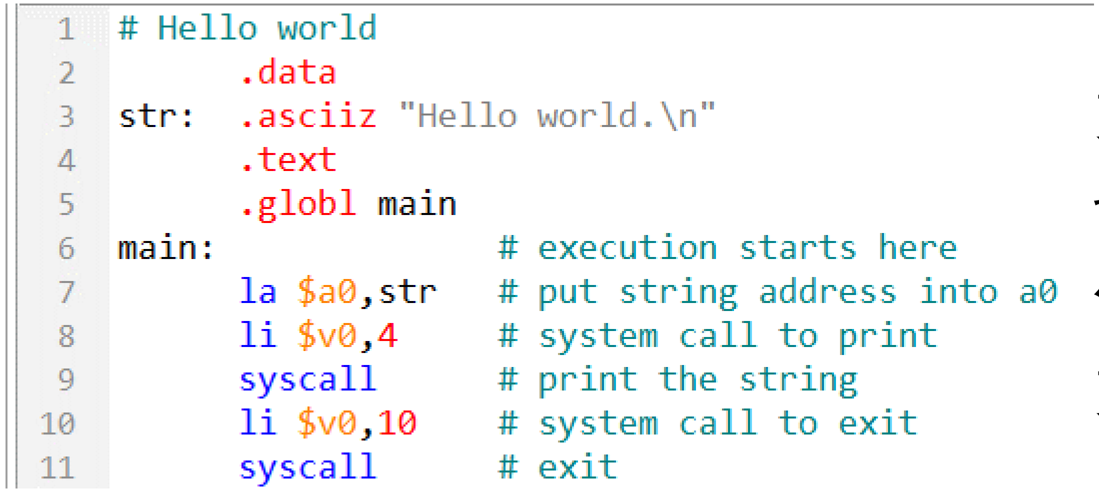
MIPS 仿真器 QtSpim

- 可生成ROM/RAM的初始化文件.COE



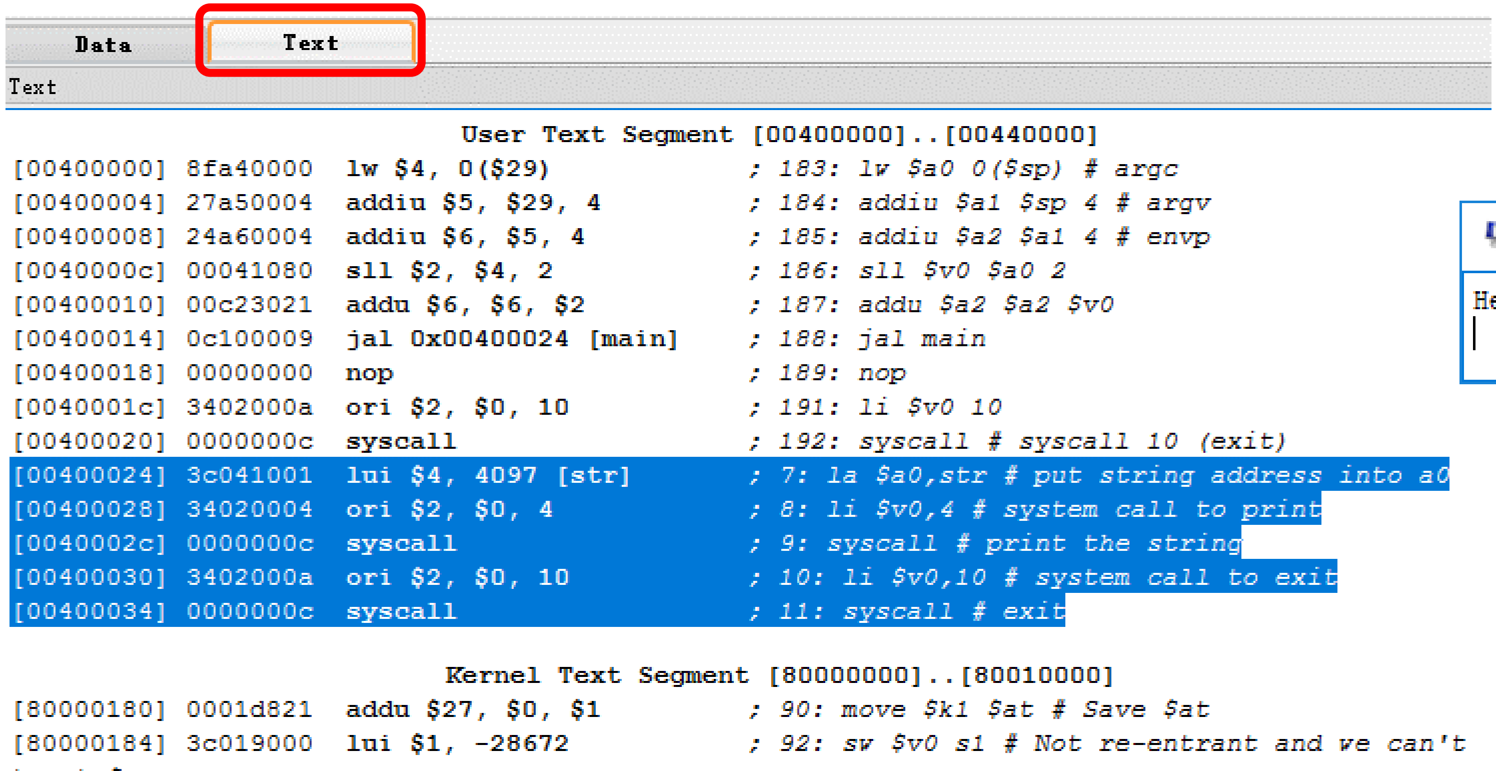
```
Text
                                                User Text Segment [00400000]..[00440000]
[00400000] 8fa40000  lw $4, 0($29)                 ; 183: lw $a0 0($sp) # argc
[00400004] 27a50004  addiu $5, $29, 4              ; 184: addiu $a1 $sp 4 # argv
[00400008] 24a60004  addiu $6, $5, 4               ; 185: addiu $a2 $a1 4 # envp
[0040000c] 00042080  sll $2, $4, 2                 ; 186: sll $v0 $a0 2
[00400010] 00c23021  addu $6, $6, $2               ; 187: addu $a2 $a2 $v0
[00400014] 0c100009  jal 0x00400024 [main]         ; 188: jal main
[00400018] 00000000  nop                           ; 189: nop
[0040001c] 3402000a  ori $2, $0, 10                ; 191: li $v0 10
[00400020] 0000000c  syscall                       ; 192: syscall # syscall 10 (exit)
[00400024] 00432020  add $4, $2, $3                ; 2: add $4, $2, $3
[00400028] 8c440004  lw $4, 4($2)                  ; 3: lw $4, 4($2)
[0040002c] ac420008  sw $2, 8($2)                  ; 4: sw $2, 8($2)
[00400030] 00831022  sub $2, $4, $3                ; 5: sub $2, $4, $3
[00400034] 00831025  or $2, $4, $3                 ; 6: or $2, $4, $3
[00400038] 00831024  and $2, $4, $3                ; 7: and $2, $4, $3
[0040003c] 0083102a  slt $2, $4, $3                ; 8: slt $2, $4, $3
[00400040] 10830002  beq $4, $3, 8 [exit-0x00400040]; 9: beq $4, $3, exit
[00400044] 08100009  j 0x00400024 [main]           ; 10: j main
[00400048] 8c620000  lw $2, 0($3)                  ; 11: lw $2, 0($3)
[0040004c] 08100009  j 0x00400024 [main]           ; 12: j main

                                                Kernel Text Segment [80000000]..[80010000]
[80000180] 0001d821  addu $27, $0, $1              ; 90: move $k1 $at # Save $at
[80000184] 3c019000  lui $1, -28672                ; 92: sw $v0 s1 # Not re-entrant and we ca
```

例1. Hello world.

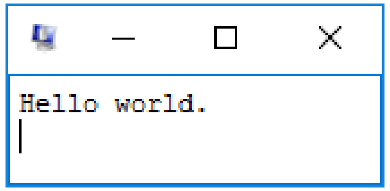


```
# Hello world
        .data
str:    .asciiz "Hello world.\n"
        .text
        .globl main
main:                   # execution starts here
        la $a0,str      # put string address into a0
        li $v0,4        # system call to print
        syscall         # print the string
        li $v0,10       # system call to exit
        syscall         # exit
```

支持MIPS汇编指令程序调试，也支持MIPS宏汇编指令。但不支持在线编辑，也不支持直接装载二进制程序。

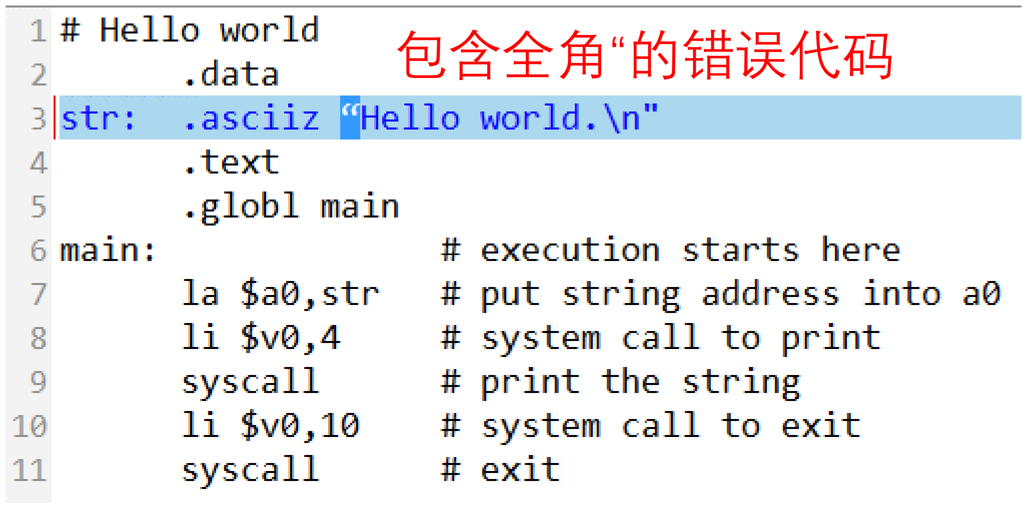


```
                         User Text Segment [00400000]..[00440000]
[00400000] 8fa40000  lw $4, 0($29)              ; 183: lw $a0 0($sp) # argc
[00400004] 27a50004  addiu $5, $29, 4           ; 184: addiu $a1 $sp 4 # argv
[00400008] 24a60004  addiu $6, $5, 4            ; 185: addiu $a2 $a1 4 # envp
[0040000c] 00041080  sll $2, $4, 2              ; 186: sll $v0 $a0 2
[00400010] 00c23021  addu $6, $6, $2            ; 187: addu $a2 $a2 $v0
[00400014] 0c100009  jal 0x00400024 [main]      ; 188: jal main
[00400018] 00000000  nop                        ; 189: nop
[0040001c] 3402000a  ori $2, $0, 10             ; 191: li $v0 10
[00400020] 0000000c  syscall                    ; 192: syscall # syscall 10 (exit)
[00400024] 3c041001  lui $4, 4097 [str]         ; 7: la $a0,str # put string address into a0
[00400028] 34020004  ori $2, $0, 4              ; 8: li $v0,4 # system call to print
[0040002c] 0000000c  syscall                    ; 9: syscall # print the string
[00400030] 3402000a  ori $2, $0, 10             ; 10: li $v0,10 # system call to exit
[00400034] 0000000c  syscall                    ; 11: syscall # exit

                        Kernel Text Segment [80000000]..[80010000]
[80000180] 0001d821  addu $27, $0, $1           ; 90: move $k1 $at # Save $at
[80000184] 3c019000  lui $1, -28672             ; 92: sw $v0 s1 # Not re-entrant and we can't
```



```
Hello world.
```

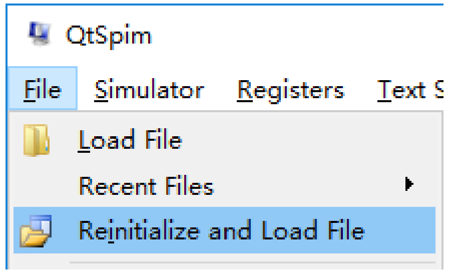
QtSpim 错误调试

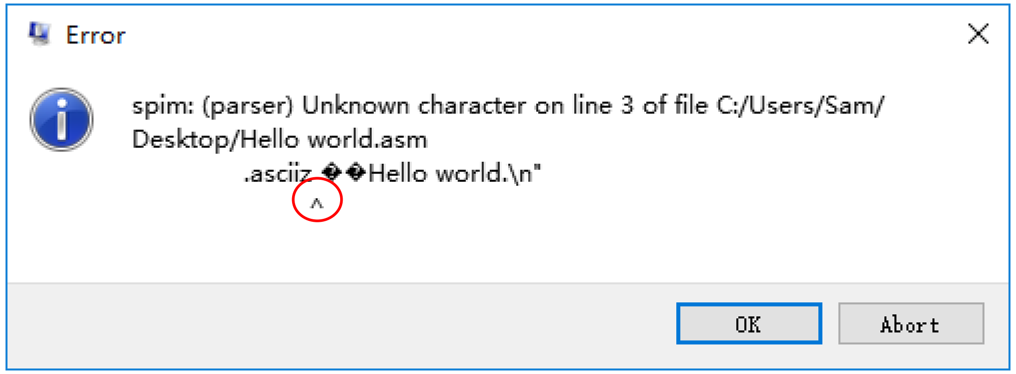


```
 1 # Hello world
 2         .data
 3 str:    .asciiz “Hello world.\n"
 4         .text
 5         .globl main
 6 main:                # execution starts here
 7         la $a0,str   # put string address into a0
 8         li $v0,4     # system call to print
 9         syscall      # print the string
10         li $v0,10    # system call to exit
11         syscall      # exit
```

包含全角“的错误代码

在QtSpim中打开





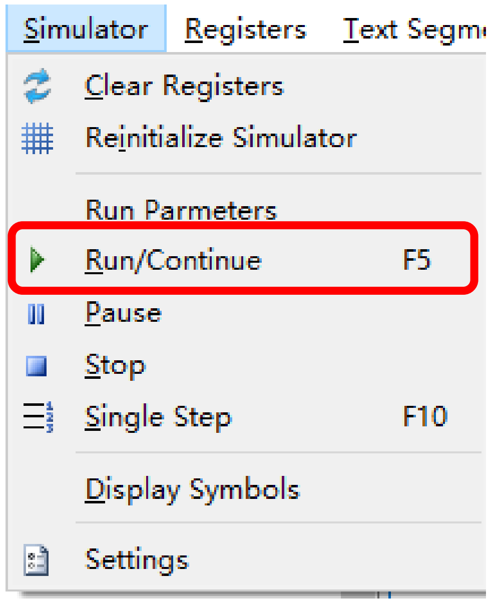
```
spim: (parser) Unknown character on line 3 of file C:/Users/Sam/
Desktop/Hello world.asm
           .asciiz ��Hello world.\n"
                 ^
```

- QtSpim装载汇编源程序后，如果源程序有错，只报第1个错误。

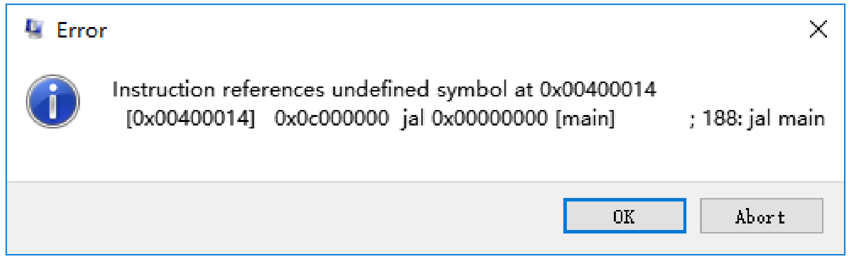
文件夹名称也不能用中文！

- 汉字显示为乱码
- 错误之处用“^”标注
- 右侧代码引号为中文全角，应改为半角双引号。

QtSpim运行代码 (F5)



将所有的通用寄存器清零
重新初始化仿真器
运行程序，并输入参数
运行/继续运行程序
暂停运行
停止运行
单步运行
将代码中含有的标号及对应的地址显示在消息窗口
设置参数



```
Instruction references undefined symbol at 0x00400014
  [0x00400014]  0x0c000000  jal 0x00000000 [main]        ; 188: jal main
```

- 程序没有定义main标号

有错误的代码

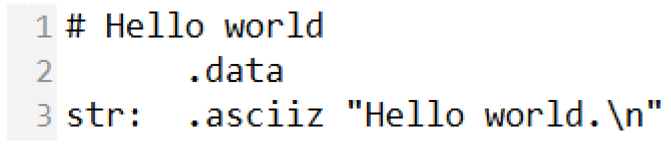
```
# Hello world
        .data
str:  .asciiz "Hello world.\n"
        .text
        .globl top
top:                    # execution starts here
        la $a0,str  # put string address into a0
        li $v0,4       # system call to print
        syscall        # print the string
        li $v0,10     # system call to exit
        syscall        # exit
```

修改后的代码

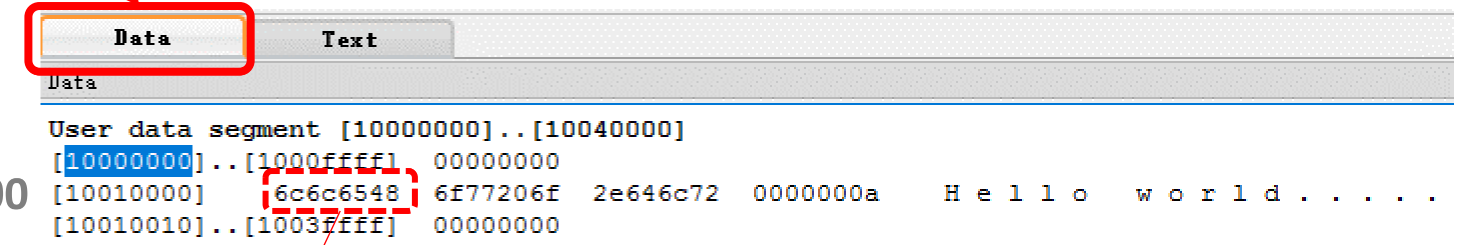
```
# Hello world
        .data
str:  .asciiz "Hello world.\n"
        .text
        .globl main
main:                   # execution starts here
        la $a0,str  # put string address into a0
        li $v0,4       # system call to print
        syscall        # print the string
        li $v0,10     # system call to exit
        syscall        # exit
```

QtSpim查看程序内存映像-1

- 通过**数据窗口**查看**用户数据段内存映像**



```
# Hello world
        .data
str:    .asciiz "Hello world.\n"
```



```
User data segment [10000000]..[10040000]
[10000000]..[1000ffff]  00000000
[10010000]    6c6c6548  6f77206f  2e646c72  0000000a    H e l l o   w o r l d . . . . .
[10010010]..[1003ffff]  00000000
```

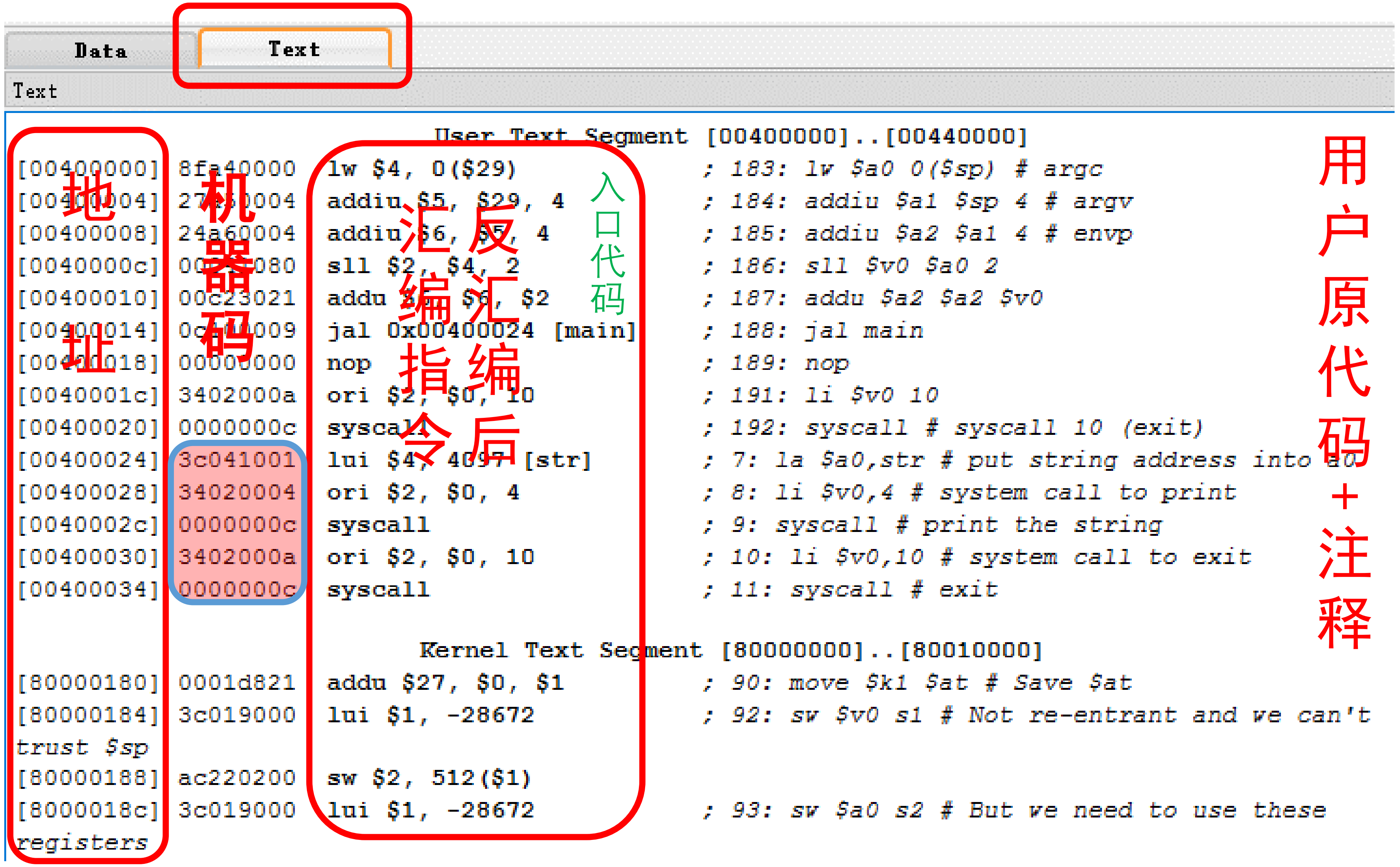
起始地址：0x**10010000**

变量名	地址	数据	定义值
str	0x10010010	0x**48**	***H***
		0x**65**	***e***
		0x**6c**	***l***
		0x**6c**	***l***
		0x**6f**	***o***
		0x**20**	
		0x**77**	***w***
		0x**6f**	***o***

【注意】MIPS微处理器原本采用**大字节**顺序存放数据，但由于仿真器运行在PC（Intel微处理器）上，因此实际数据的存储采用**小字节**顺序。

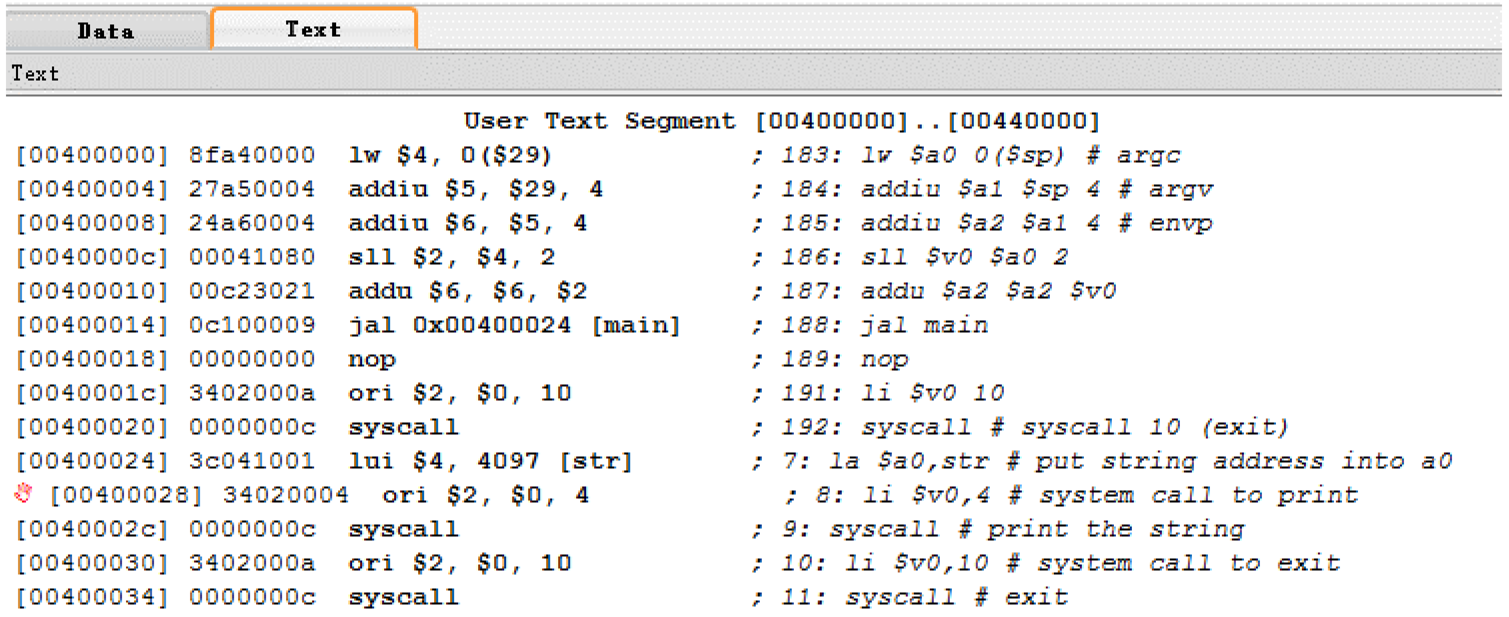
QtSpim查看程序内存映像-2

通过**代码窗口**查看**用户代码段内存映像**



```
                      User Text Segment [00400000]..[00440000]
[00400000] 8fa40000  lw $4, 0($29)              ; 183: lw $a0 0($sp) # argc
[00400004] 27a50004  addiu $5, $29, 4           ; 184: addiu $a1 $sp 4 # argv
[00400008] 24a60004  addiu $6, $5, 4            ; 185: addiu $a2 $a1 4 # envp
[0040000c] 00041080  sll $2, $4, 2              ; 186: sll $v0 $a0 2
[00400010] 00c23021  addu $6, $6, $2            ; 187: addu $a2 $a2 $v0
[00400014] 0c100009  jal 0x00400024 [main]      ; 188: jal main
[00400018] 00000000  nop                        ; 189: nop
[0040001c] 3402000a  ori $2, $0, 10             ; 191: li $v0 10
[00400020] 0000000c  syscall                    ; 192: syscall # syscall 10 (exit)
[00400024] 3c041001  lui $4, 4097 [str]         ; 7: la $a0,str # put string address into a0
[00400028] 34020004  ori $2, $0, 4              ; 8: li $v0,4 # system call to print
[0040002c] 0000000c  syscall                    ; 9: syscall # print the string
[00400030] 3402000a  ori $2, $0, 10             ; 10: li $v0,10 # system call to exit
[00400034] 0000000c  syscall                    ; 11: syscall # exit


                      Kernel Text Segment [80000000]..[80010000]
[80000180] 0001d821  addu $27, $0, $1           ; 90: move $k1 $at # Save $at
[80000184] 3c019000  lui $1, -28672             ; 92: sw $v0 s1 # Not re-entrant and we can't
trust $sp
[80000188] ac220200  sw $2, 512($1)
[8000018c] 3c019000  lui $1, -28672             ; 93: sw $a0 s2 # But we need to use these
registers
```

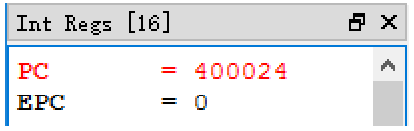
QtSpim断点、调试



```
                          User Text Segment [00400000]..[00440000]
[00400000] 8fa40000  lw $4, 0($29)             ; 183: lw $a0 0($sp) # argc
[00400004] 27a50004  addiu $5, $29, 4          ; 184: addiu $a1 $sp 4 # argv
[00400008] 24a60004  addiu $6, $5, 4           ; 185: addiu $a2 $a1 4 # envp
[0040000c] 00041080  sll $2, $4, 2             ; 186: sll $v0 $a0 2
[00400010] 00c23021  addu $6, $6, $2           ; 187: addu $a2 $a2 $v0
[00400014] 0c100009  jal 0x00400024 [main]     ; 188: jal main
[00400018] 00000000  nop                       ; 189: nop
[0040001c] 3402000a  ori $2, $0, 10            ; 191: li $v0 10
[00400020] 0000000c  syscall                   ; 192: syscall # syscall 10 (exit)
[00400024] 3c041001  lui $4, 4097 [str]        ; 7: la $a0,str # put string address into a0
 [00400028] 34020004  ori $2, $0, 4            ; 8: li $v0,4 # system call to print
[0040002c] 0000000c  syscall                   ; 9: syscall # print the string
[00400030] 3402000a  ori $2, $0, 10            ; 10: li $v0,10 # system call to exit
[00400034] 0000000c  syscall                   ; 11: syscall # exit
```

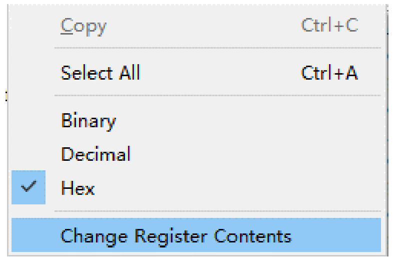
鼠标右键设置**断点**

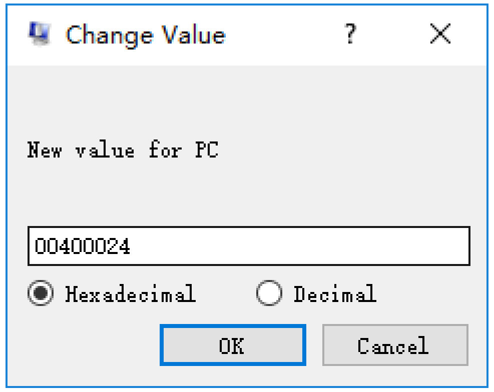
观察寄存器的值



```
PC        = 400024
EPC       = 0
```

改变寄存器的值





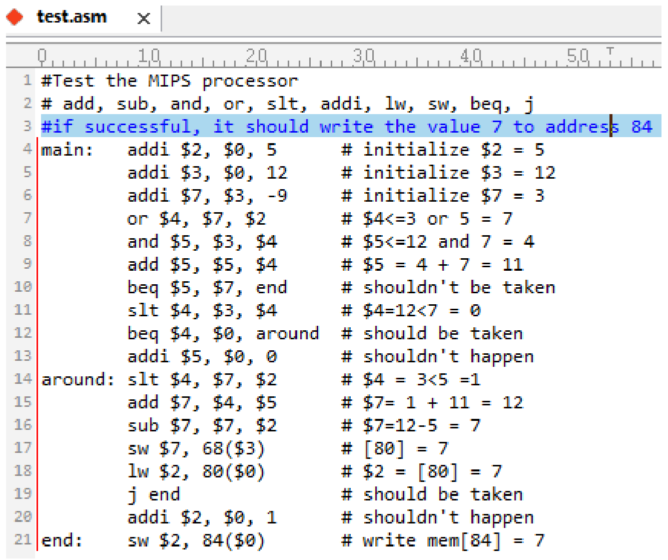
ROM/RAM初始化文件.COE的制作

或Notepad++

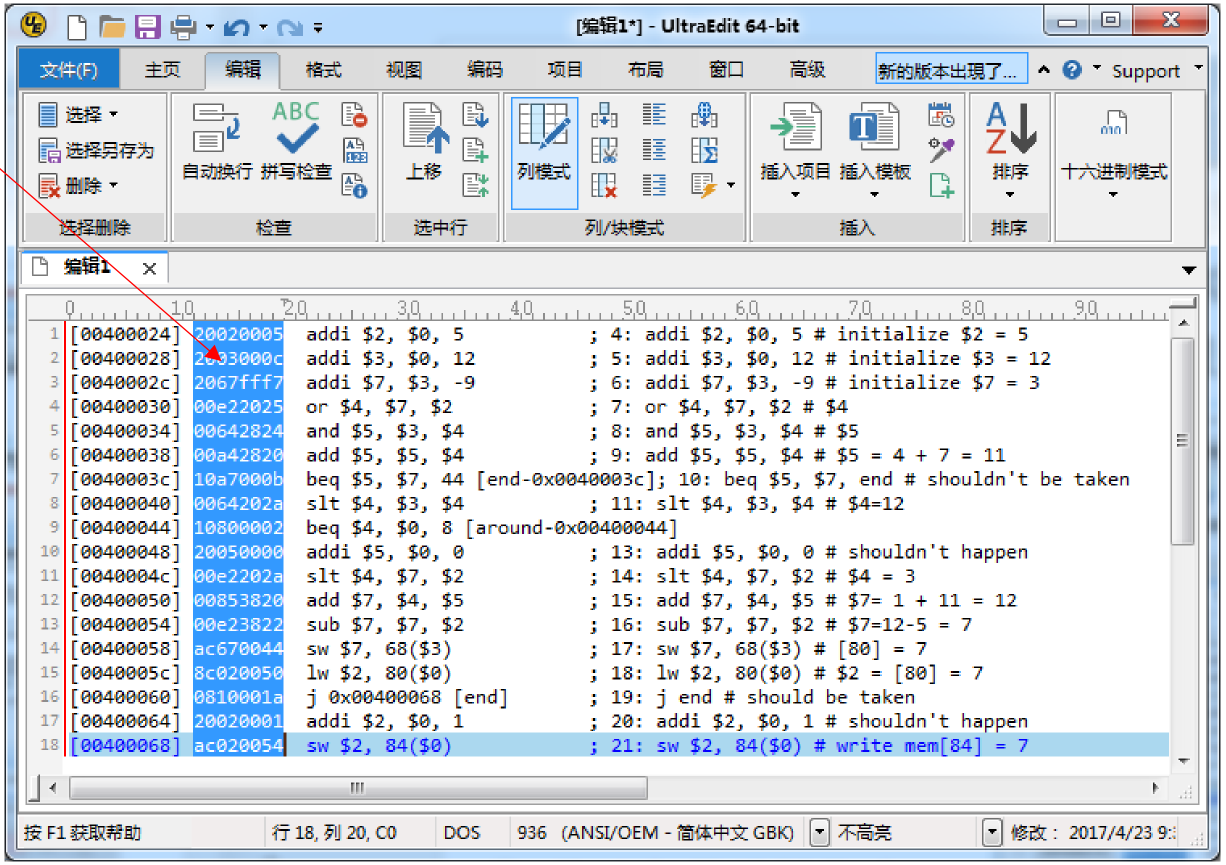
① 用ultraEdit编辑汇编源程序代码。(见左下角图)

② 用QtSpim装载test.asm， 同时测试功能是否正常。(见上页图)

③ 复制QtSpim中的用户代码段， 拷贝到ultraEdit中， 并设置为**列模式**，提取**机器码**。



```
#Test the MIPS processor
# add, sub, and, or, slt, addi, lw, sw, beq, j
#if successful, it should write the value 7 to address 84
main:   addi $2, $0, 5      # initialize $2 = 5
        addi $3, $0, 12     # initialize $3 = 12
        addi $7, $3, -9     # initialize $7 = 3
        or $4, $7, $2       # $4<=3 or 5 = 7
        and $5, $3, $4      # $5<=12 and 7 = 4
        add $5, $5, $4      # $5 = 4 + 7 = 11
        beq $5, $7, end     # shouldn't be taken
        slt $4, $3, $4      # $4=12<7 = 0
        beq $4, $0, around  # should be taken
        addi $5, $0, 0      # shouldn't happen
around: slt $4, $7, $2      # $4 = 3<5 =1
        add $7, $4, $5      # $7= 1 + 11 = 12
        sub $7, $7, $2      # $7=12-5 = 7
        sw $7, 68($3)       # [80] = 7
        lw $2, 80($0)       # $2 = [80] = 7
        j end               # should be taken
        addi $2, $0, 1      # shouldn't happen
end:    sw $2, 84($0)       # write mem[84] = 7
```



```
[00400024] 20020005  addi $2, $0, 5            ; 4: addi $2, $0, 5 # initialize $2 = 5
[00400028] 2003000c  addi $3, $0, 12           ; 5: addi $3, $0, 12 # initialize $3 = 12
[0040002c] 2067fff7  addi $7, $3, -9           ; 6: addi $7, $3, -9 # initialize $7 = 3
[00400030] 00e22025  or $4, $7, $2             ; 7: or $4, $7, $2 # $4
[00400034] 00642824  and $5, $3, $4            ; 8: and $5, $3, $4 # $5
[00400038] 00a42820  add $5, $5, $4            ; 9: add $5, $5, $4 # $5 = 4 + 7 = 11
[0040003c] 10a7000b  beq $5, $7, 44 [end-0x0040003c]; 10: beq $5, $7, end # shouldn't be taken
[00400040] 0064202a  slt $4, $3, $4            ; 11: slt $4, $3, $4 # $4=12
[00400044] 10800002  beq $4, $0, 8 [around-0x00400044]
[00400048] 20050000  addi $5, $0, 0            ; 13: addi $5, $0, 0 # shouldn't happen
[0040004c] 00e2202a  slt $4, $7, $2            ; 14: slt $4, $7, $2 # $4 = 3
[00400050] 00853820  add $7, $4, $5            ; 15: add $7, $4, $5 # $7= 1 + 11 = 12
[00400054] 00e23822  sub $7, $7, $2            ; 16: sub $7, $7, $2 # $7=12-5 = 7
[00400058] ac670044  sw $7, 68($3)             ; 17: sw $7, 68($3) # [80] = 7
[0040005c] 8c020050  lw $2, 80($0)             ; 18: lw $2, 80($0) # $2 = [80] = 7
[00400060] 0810001a  j 0x00400068 [end]        ; 19: j end # should be taken
[00400064] 20020001  addi $2, $0, 1            ; 20: addi $2, $0, 1 # shouldn't happen
[00400068] ac020054  sw $2, 84($0)             ; 21: sw $2, 84($0) # write mem[84] = 7
```

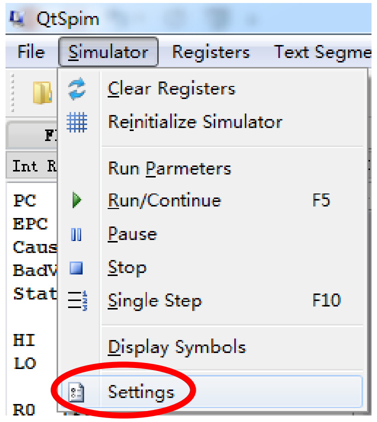
按 F1 获取帮助 | 行 18, 列 20, C0 | DOS | 936 (ANSI/OEM - 简体中文 GBK) | 不高亮 | 修改： 2017/4/23 9:

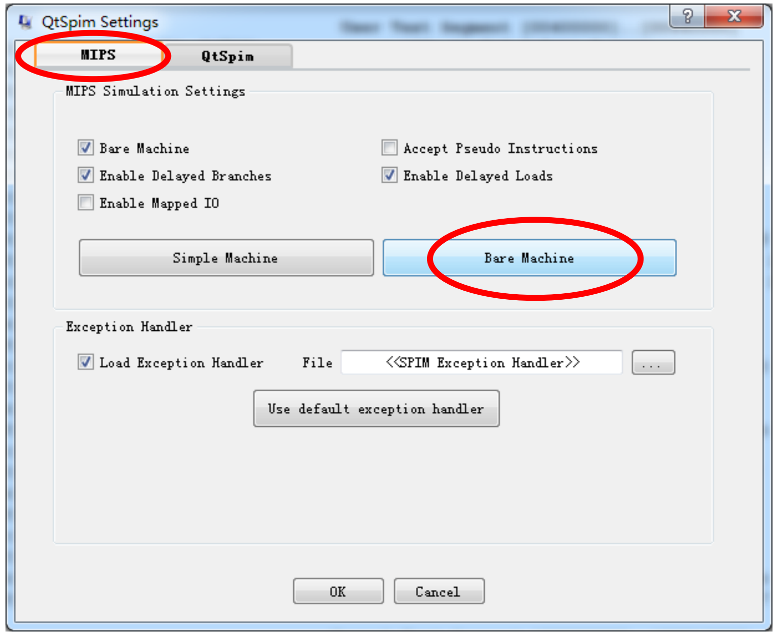
ROM/RAM初始化文件.COE的制作-2

④ 有条件跳转指令机器码

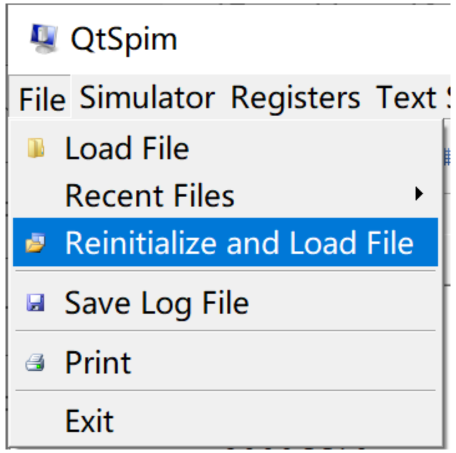
如 beq、bne等指令

方法一：设置QtSpim参数为 **Bare Machine**





- [✓] Bare Machine
- [ ] Accept Pseudo Instructions
- [✓] Enable Delayed Branches
- [✓] Enable Delayed Loads
- [ ] Enable Mapped IO
- [✓] Load Exception Handler

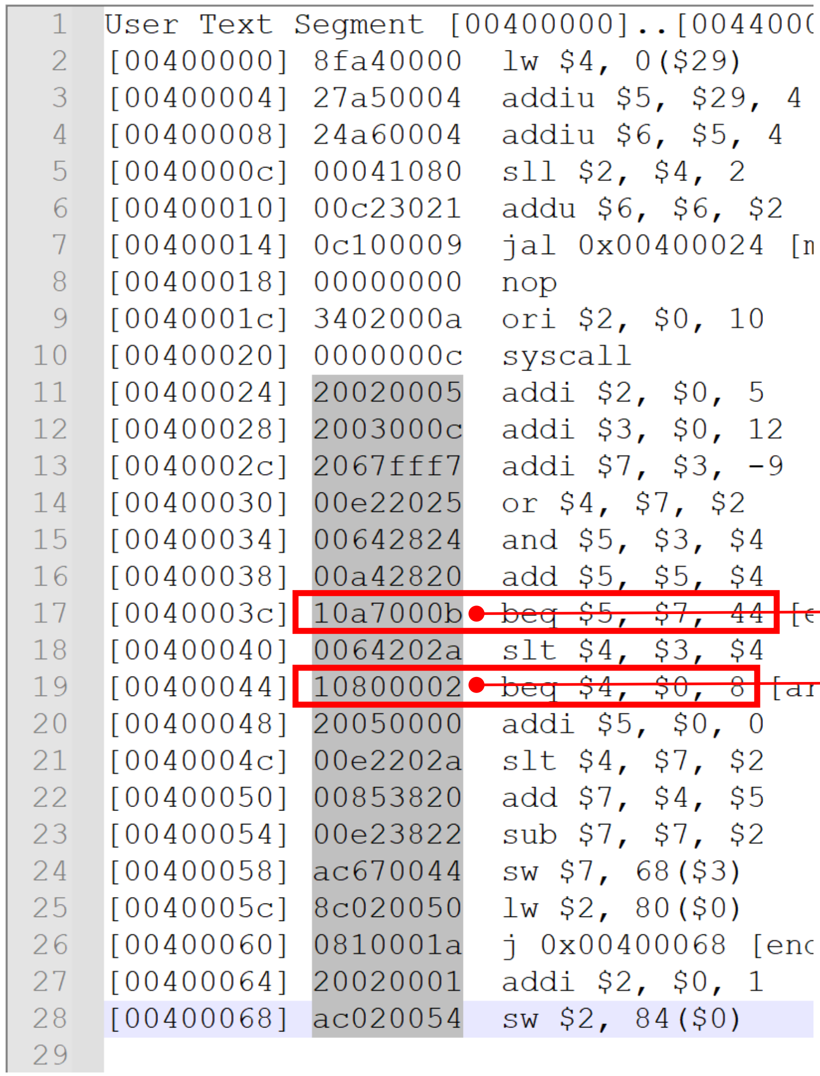


方法二：自行修改为：**条件跳转指令地址 – 1**

参见：教材P396附录B 注脚

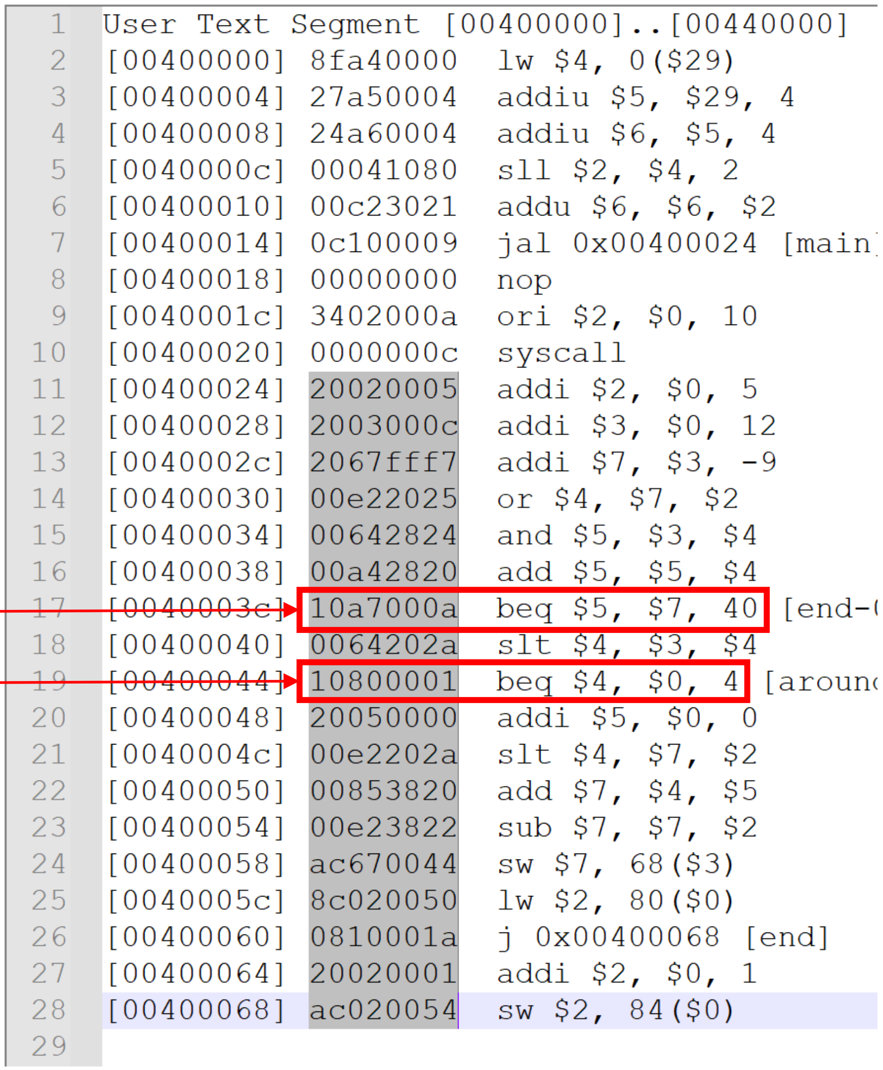
方法一: 设置QtSpim参数为 Bare Machine

Simple Machine



```
User Text Segment [00400000]..[0044000
[00400000] 8fa40000  lw $4, 0($29)
[00400004] 27a50004  addiu $5, $29, 4
[00400008] 24a60004  addiu $6, $5, 4
[0040000c] 00041080  sll $2, $4, 2
[00400010] 00c23021  addu $6, $6, $2
[00400014] 0c100009  jal 0x00400024 [m
[00400018] 00000000  nop
[0040001c] 3402000a  ori $2, $0, 10
[00400020] 0000000c  syscall
[00400024] 20020005  addi $2, $0, 5
[00400028] 2003000c  addi $3, $0, 12
[0040002c] 2067fff7  addi $7, $3, -9
[00400030] 00e22025  or $4, $7, $2
[00400034] 00642824  and $5, $3, $4
[00400038] 00a42820  add $5, $5, $4
[0040003c] 10a7000b  beq $5, $7, 44 [e
[00400040] 0064202a  slt $4, $3, $4
[00400044] 10800002  beq $4, $0, 8 [ar
[00400048] 20050000  addi $5, $0, 0
[0040004c] 00e2202a  slt $4, $7, $2
[00400050] 00853820  add $7, $4, $5
[00400054] 00e23822  sub $7, $7, $2
[00400058] ac670044  sw $7, 68($3)
[0040005c] 8c020050  lw $2, 80($0)
[00400060] 0810001a  j 0x00400068 [enc
[00400064] 20020001  addi $2, $0, 1
[00400068] ac020054  sw $2, 84($0)
```

Bare Machine



```
User Text Segment [00400000]..[00440000]
[00400000] 8fa40000  lw $4, 0($29)
[00400004] 27a50004  addiu $5, $29, 4
[00400008] 24a60004  addiu $6, $5, 4
[0040000c] 00041080  sll $2, $4, 2
[00400010] 00c23021  addu $6, $6, $2
[00400014] 0c100009  jal 0x00400024 [main
[00400018] 00000000  nop
[0040001c] 3402000a  ori $2, $0, 10
[00400020] 0000000c  syscall
[00400024] 20020005  addi $2, $0, 5
[00400028] 2003000c  addi $3, $0, 12
[0040002c] 2067fff7  addi $7, $3, -9
[00400030] 00e22025  or $4, $7, $2
[00400034] 00642824  and $5, $3, $4
[00400038] 00a42820  add $5, $5, $4
[0040003c] 10a7000a  beq $5, $7, 40 [end-
[00400040] 0064202a  slt $4, $3, $4
[00400044] 10800001  beq $4, $0, 4 [aroun
[00400048] 20050000  addi $5, $0, 0
[0040004c] 00e2202a  slt $4, $7, $2
[00400050] 00853820  add $7, $4, $5
[00400054] 00e23822  sub $7, $7, $2
[00400058] ac670044  sw $7, 68($3)
[0040005c] 8c020050  lw $2, 80($0)
[00400060] 0810001a  j 0x00400068 [end]
[00400064] 20020001  addi $2, $0, 1
[00400068] ac020054  sw $2, 84($0)
```

ROM/RAM初始化文件.COE的制作-3

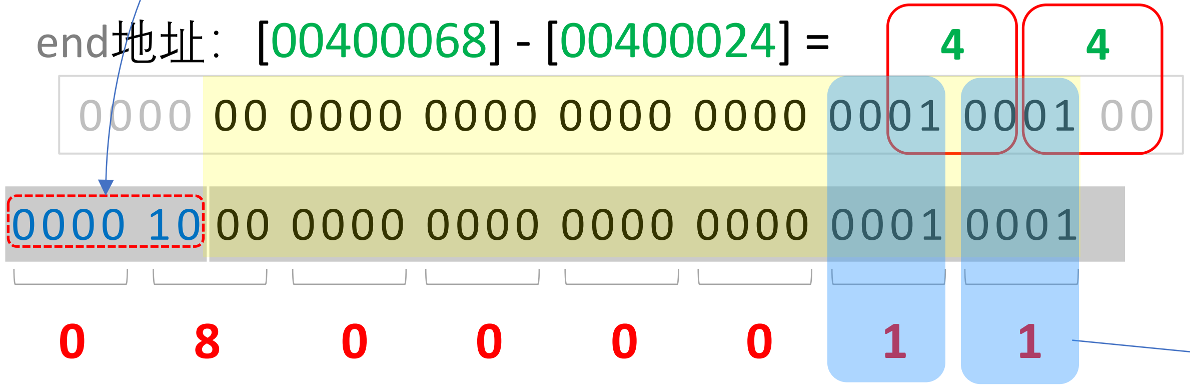
⑤ 修改**无条件跳转指令**中地址。

操作码
26位地址操作数

例 j end

先将32位地址转为26位，
再与6位操作码合为机器码。

end地址：[00400068] - [00400024] =



j的opcode =$(000010)_2$，参见教材附录B 396页

规律：跳转相对地址/4

	地址	拷贝后的机器码	修改后的机器码
[00400024]	0	20020005	20020005
	4	2003000C	2003000C
	8	2067FFF7	2067FFF7
	C	00E22025	00E22025
	10	00642824	00642824
	14	00A42820	00A42820
	18	10A7000A	10A7000A
	1C	0064202A	0064202A
	20	10800001	10800001
	24	20050000	20050000
	28	00E2202A	00E2202A
	2C	00853820	00853820
	30	00E23822	00E23822
	34	AC670044	AC670044
	38	8C020050	8C020050
j end	3C	0810001A	08000011
	40	20020001	20020001
end: [00400068]	44	AC020054	AC020054

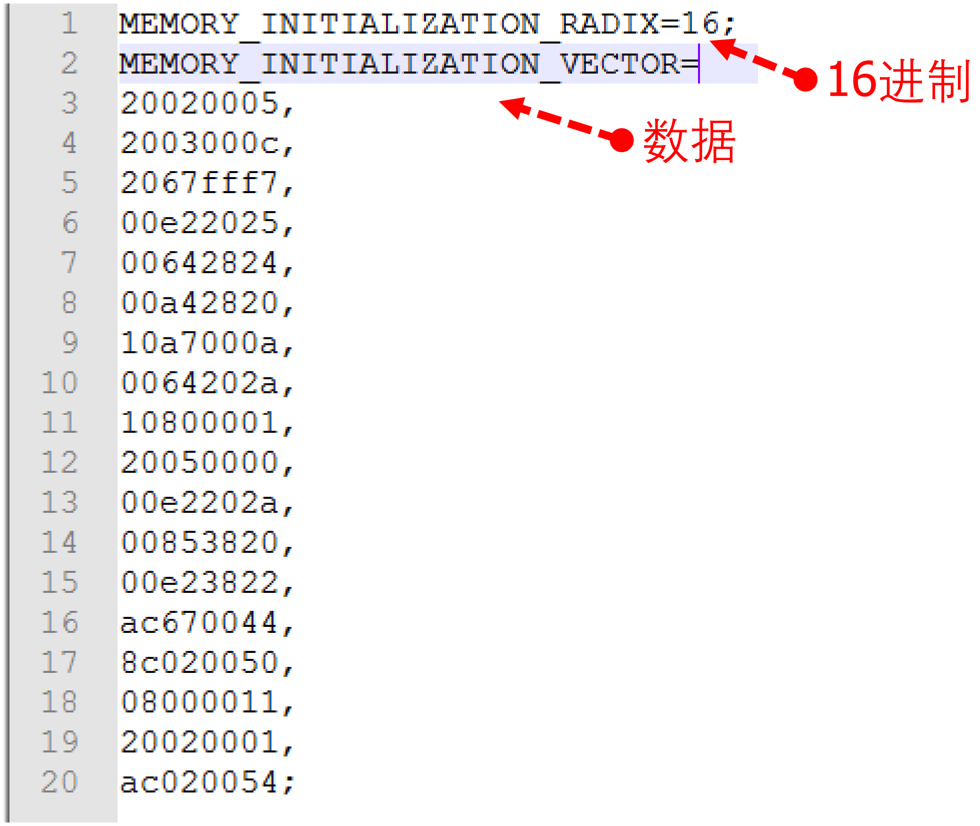
ROM/RAM初始化文件.COE的制作-4

⑥ 添加coe文件头描述语句。

并在每行机器指令后加”,”号

最后一行结尾加”;”号

⑦ 保存为test.coe文件。



```
MEMORY_INITIALIZATION_RADIX=16;
MEMORY_INITIALIZATION_VECTOR=
20020005,
2003000c,
2067fff7,
00e22025,
00642824,
00a42820,
10a7000a,
0064202a,
10800001,
20050000,
00e2202a,
00853820,
00e23822,
ac670044,
8c020050,
08000011,
20020001,
ac020054;
```

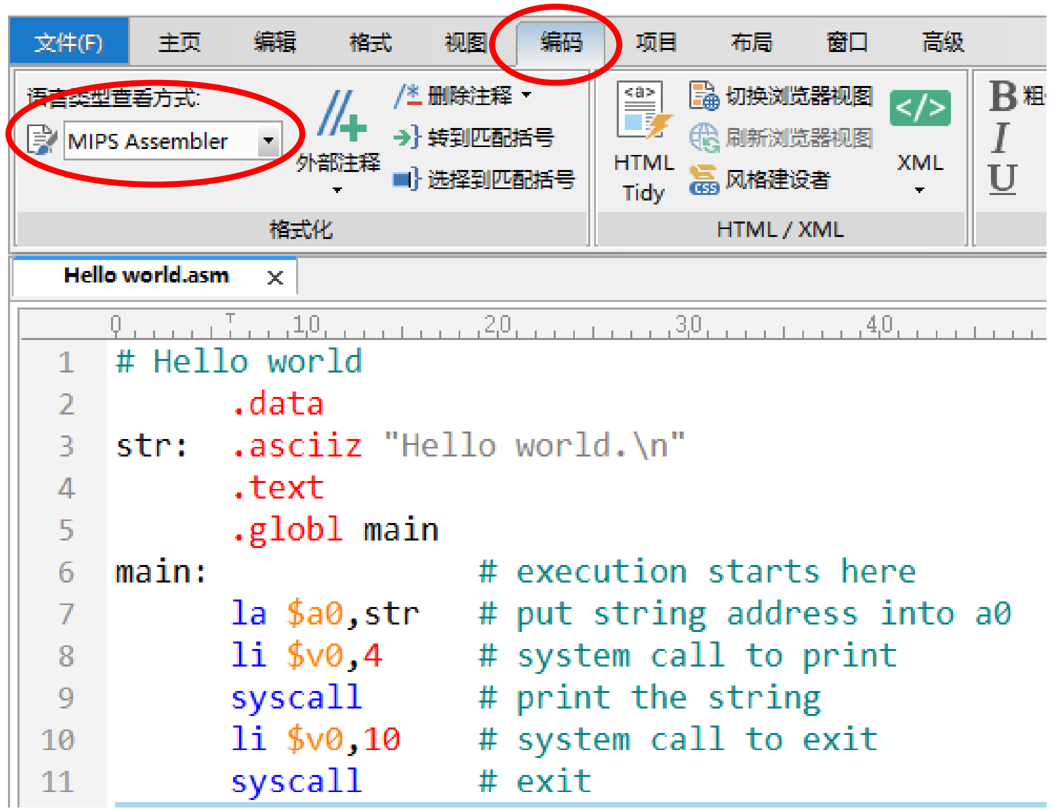
Ultraedit软件加亮显示MIPS汇编文件

- 下载相应的格式显示配置文件（.uew）

  http://www.ultraedit.com/downloads/extras/wordfiles.html#wordfiles

- 将该文件放到路径：　　【注】IDMComp为隐藏子目录

  C:\Users\你的用户名\AppData\Roaming\IDMComp\UltraEdit\wordfiles

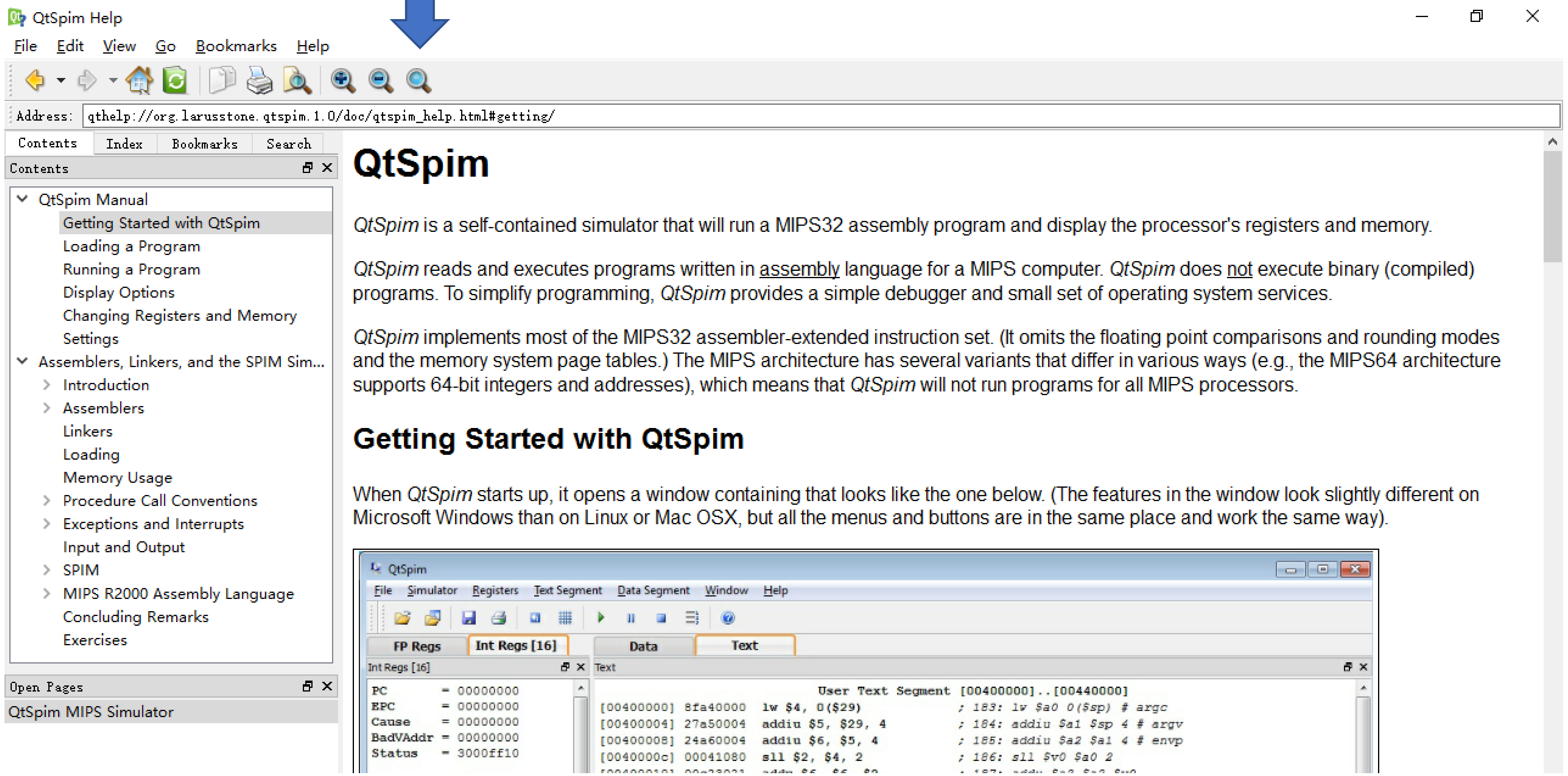


```
# Hello world
        .data
str:    .asciiz "Hello world.\n"
        .text
        .globl main
main:                   # execution starts here
        la $a0,str      # put string address into a0
        li $v0,4        # system call to print
        syscall         # print the string
        li $v0,10       # system call to exit
        syscall         # exit
```

参考资料

- QtSpim Help

http://pages.cs.wisc.edu/~larus/spim.html


QtSpim Help
File Edit View Go Bookmarks Help
Address: qthelp://org.larusstone.qtspim.1.0/doc/qtspim_help.html#getting/
Contents Index Bookmarks Search
Contents
Open Pages
QtSpim MIPS Simulator

QtSpim

QtSpim is a self-contained simulator that will run a MIPS32 assembly program and display the processor's registers and memory.

QtSpim reads and executes programs written in assembly language for a MIPS computer. *QtSpim* does not execute binary (compiled) programs. To simplify programming, *QtSpim* provides a simple debugger and small set of operating system services.

QtSpim implements most of the MIPS32 assembler-extended instruction set. (It omits the floating point comparisons and rounding modes and the memory system page tables.) The MIPS architecture has several variants that differ in various ways (e.g., the MIPS64 architecture supports 64-bit integers and addresses), which means that *QtSpim* will not run programs for all MIPS processors.

Getting Started with QtSpim

When *QtSpim* starts up, it opens a window containing that looks like the one below. (The features in the window look slightly different on Microsoft Windows than on Linux or Mac OSX, but all the menus and buttons are in the same place and work the same way).

QtSpim
File Simulator Registers Text Segment Data Segment Window Help
FP Regs
Int Regs [16]
Data
Text

```
Int Regs [16]
PC       = 00000000
EPC      = 00000000
Cause    = 00000000
BadVAddr = 00000000
Status   = 3000ff10
```

```
Text
                        User Text Segment [00400000]..[00440000]
[00400000] 8fa40000  lw $4, 0($29)          ; 183: lw $a0 0($sp) # argc
[00400004] 27a50004  addiu $5, $29, 4       ; 184: addiu $a1 $sp 4 # argv
[00400008] 24a60004  addiu $6, $5, 4        ; 185: addiu $a2 $a1 4 # envp
[0040000c] 00041080  sll $2, $4, 2          ; 186: sll $v0 $a0 2
```

类型	指令	指令举例	含义	备注
算术运算	加法	add $s1,$s2,$s3	$s1=$s2+$s3	三个寄存器操作数
	减法	sub $s1,$s2,$s3	$s1=$s2-$s3	三个寄存器操作数
	加立即数	addi $s1,$s2,20	$s1=$s2+20	用来加立即数
数据传送	读取字	lw $s1,20($s2)	$s1=mem[$s2+20]	从内存读字到寄存器
	存储字	sw $s1,20($s2)	mem[$2+20] =$s1	从寄存器写字到内存
	读取半字	lh $s1,20($s2)	$s1=mem[$s2+20]	从内存读半字到寄存器
	读取无符号半字	lhu $s1,20($s2)	$s1=mem[$s2+20]	从内存读半字到寄存器
	存储半字	sh $s1,20($s2)	mem[$s2+20] =$1	从寄存器写半字到内存
	读取字节	lb $s1,20($s2)	$s1=mem[$s2+20]	从内存读字节到寄存器
	读取无符号字节	lbu $s1,20($s2)	$s1=mem[$s2+20]	从内存读字节到寄存器
	存储字节	sb $s1,20($s2)	mem[$s2+20] =$s1	从寄存器写字节到内存
	读取链接字	ll $s1,20($s2)	$s1= mem[$s2+20]	读字作为原子交换的第一半
	条件存储字	sc $s1,20($s2)	mem[$s2+20] =s$1;$s1=0或1	写字作为原子交换的第二半
	读取立即数到高半字	lui $s1,20	$s1=20*2^{16}	读取一个常数到高16位
逻辑操作	与	and $s1,$s2,$s3	$s1=$s2&$s3	三个寄存器，位与
	或	or $s1,$s2,$s3	$s1=$s2|$s3	三个寄存器，位或
	或非	nor $s1,$s2,$s3	$s1=~($s2|$s3)	三个寄存器，位或非
	与立即数	andi $s1,$s2,20	$s1=$s2&20	寄存器与立即数位与
	或立即数	ori $s1,$s2,20	$s1=$s2|20	寄存器与立即数位或
	逻辑左移	sll $s1,$s2,10	$s1=$s2<<10	左移常数次
	逻辑右移	srl $s1,$s2,10	$s1=$s2>>10	右移常数次
条件跳转	相等转移	beq $s1,$s2,25	If ($s1=$s2) goto PC+4+25*4	相等测试，转移
	不相等转移	bne $s1,$s2,25	If ($s1!=$s2) goto PC+4+25*4	不相等测试，转移
	小于设置	slt $s1,$s2,$s3	If($s2<$s3) $s1=1 else $s1=0	比较小于设置$s1=1
	低于设置	sltu $s1,$s2,$s3	If($s2<$s3) $s1=1 else $s1=0	比较低于设置$s1=1
	小于常数设置	slti $s1,$s2,20	If($s2<20) $s1=1 else $s1=0	和常数比较小于设置$s1=1
	低于常数设置	sltiu $s1,$s2,20	If($s2<20) $s1=1 else $s1=0	和常数比较低于设置$s1=1
无条件跳转	直接跳转	j 2500	goto 2500*4	跳转到目标地址
	间接跳转	jr $ra	goto $ra	用在分支和子程序返回
	跳转并链接	jal 2500	$ra=PC+4; goto 2500*4	用在子程序调用
系统功能调用	系统功能调用	syscall		实现人机对话