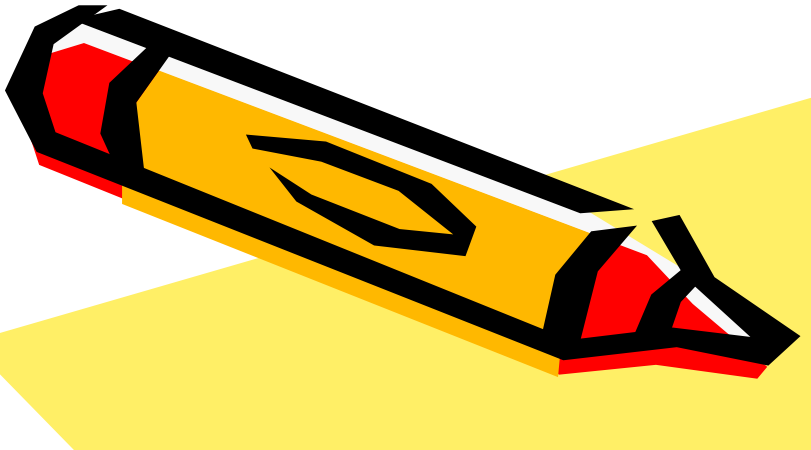
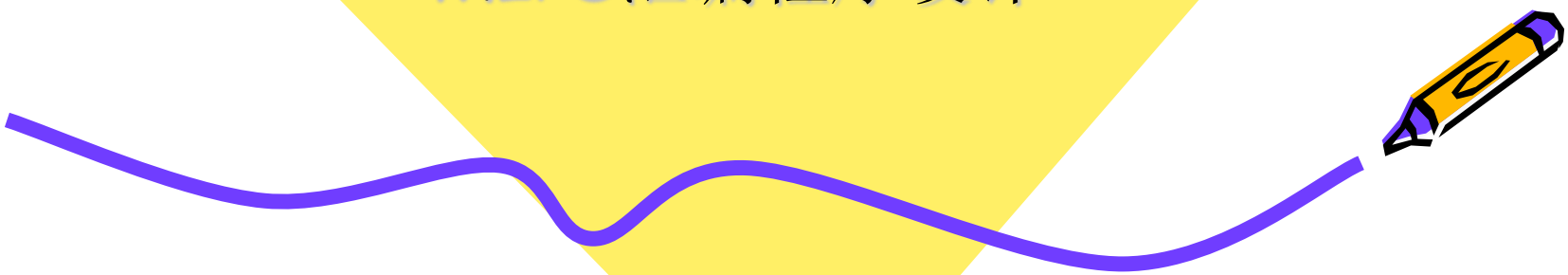


数字系统II实验

MIPS汇编程序设计

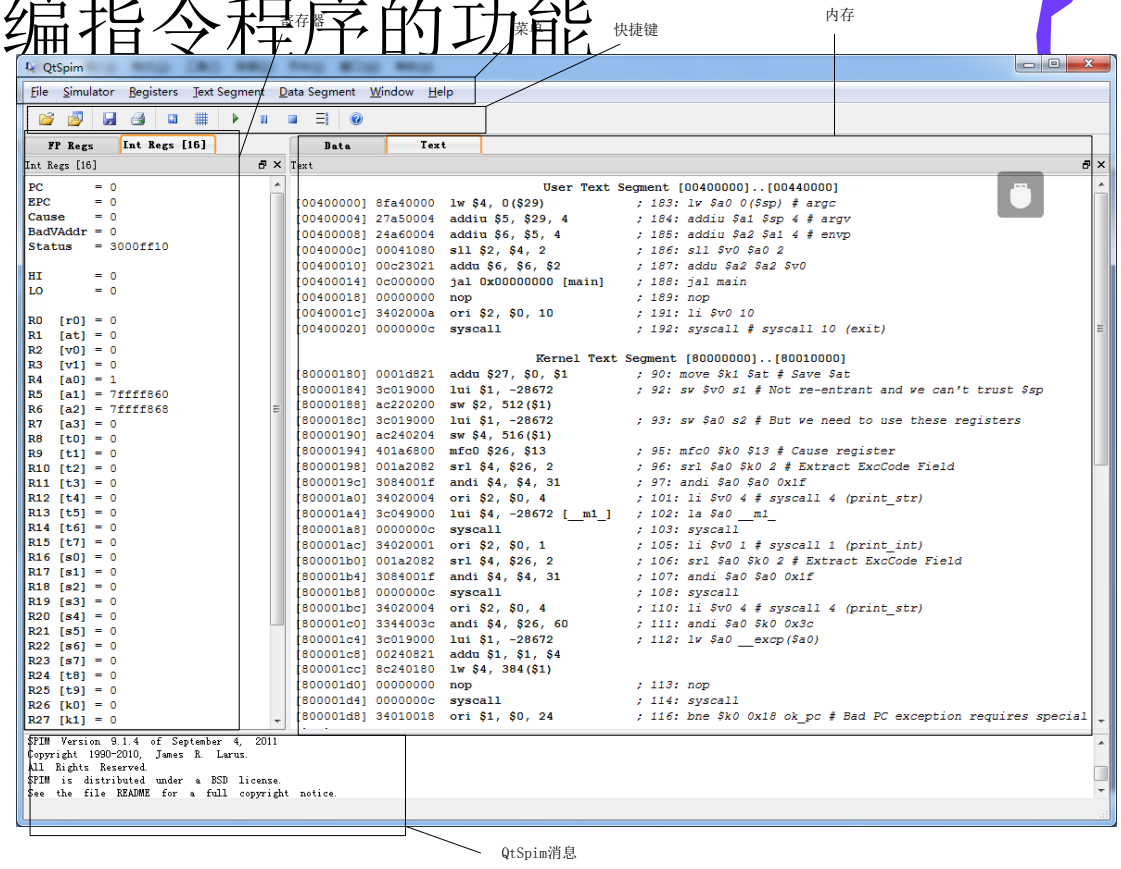


实验目的

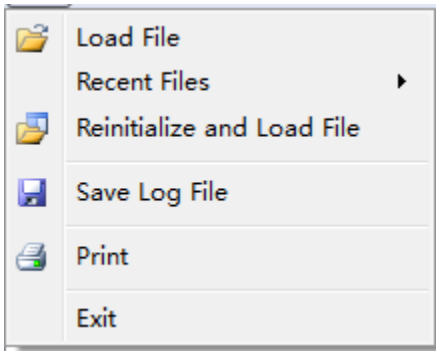
1. 掌握QTSPIM的调试技术
2. 了解MIPS汇编语言与机器语言之间的对应关系
3. 掌握MIPS汇编程序设计
4. 了解C语言语句与汇编指令之间的关系
5. 熟悉常见的MIPS汇编指令
6. 掌握程序的内存映像

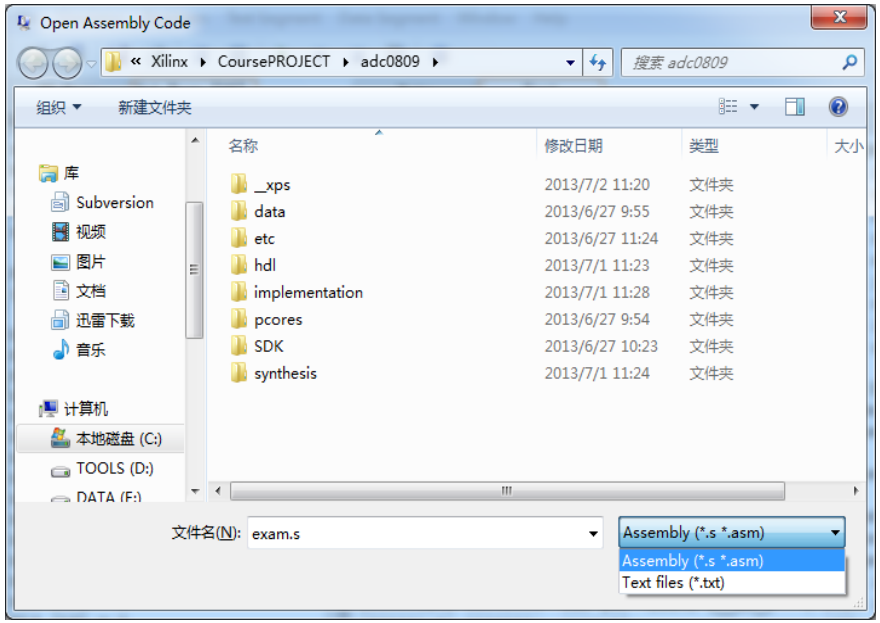
实验环境简介——QTSPIM

- QTSPIM:
  - 运行在windows操作系统下的支持MIPS32指令集的MIPS微处理器仿真器，具备调试、运行MIPS32汇编指令程序的功能

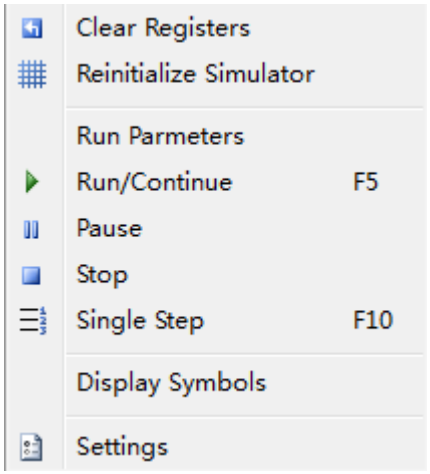


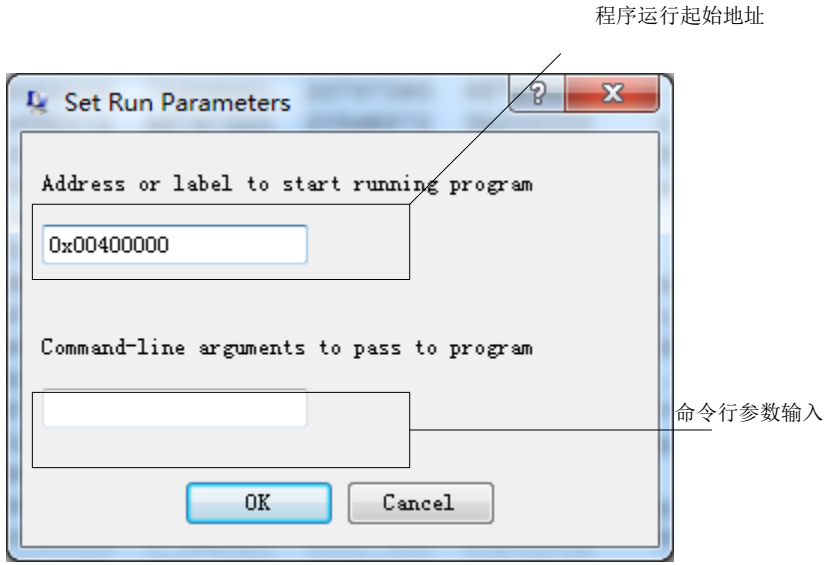
File菜单


Load File
Recent Files
Reinitialize and Load File
Save Log File
Print
Exit

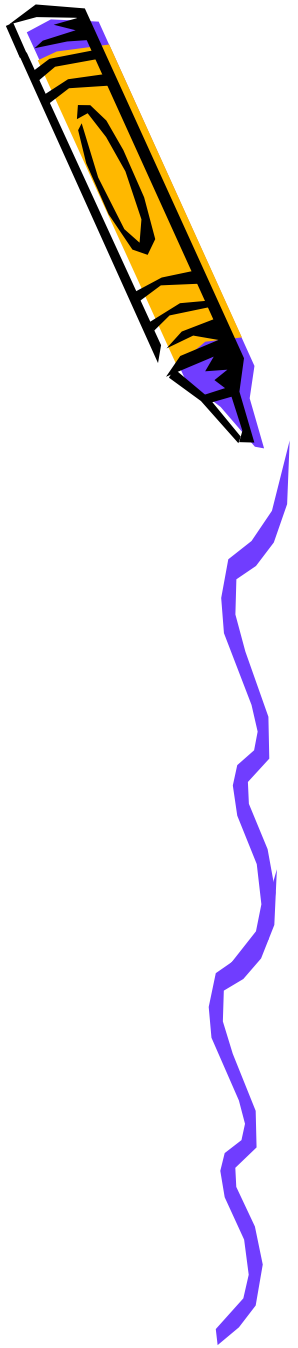

Open Assembly Code
Xilinx ▸ CoursePROJECT ▸ adc0809 ▸
搜索 adc0809
组织
新建文件夹
名称
修改日期
类型
大小
库
Subversion
视频
图片
文档
迅雷下载
音乐
计算机
本地磁盘 (C:)
TOOLS (D:)
_xps 2013/7/2 11:20 文件夹
data 2013/6/27 9:55 文件夹
etc 2013/6/27 11:24 文件夹
hdl 2013/7/1 11:23 文件夹
implementation 2013/7/1 11:28 文件夹
pcores 2013/6/27 9:54 文件夹
SDK 2013/6/27 10:23 文件夹
synthesis 2013/7/1 11:24 文件夹
文件名(N): exam.s
Assembly (*.s *.asm)
Assembly (*.s *.asm)
Text files (*.txt)

Simulator菜单





汇编程序结构

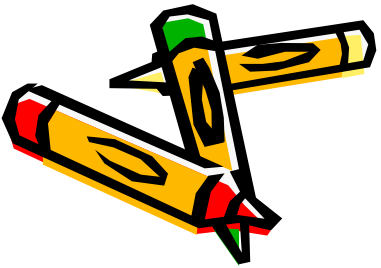


```
    .data
      prompt: .asciiz "\n Please Input a value for N = "
      result: .asciiz " The sum of the integers from 1 to N is"
      bye: .asciiz "\n **** Adios Amigo - Have a good day ****"
    .globl main
    .text
main:
    li $v0, 4 # 打印字符串spim系统功能掉用号存入$v0
    la $a0, prompt # 将消息prompt地址存入 $a0
    syscall # 打印消息
    li $v0, 5 # 读取整数spim系统功能掉用号存入$v0
    syscall # 从console读入整数存入 $v0
    blez $v0, end # 如果 $v0 < = 0跳转到end标号处
    li $t0, 0 # 清除 $t0 为0
loop:
    add $t0, $t0, $v0 # 将和存入 $t0
    addi $v0, $v0, -1 # 输入数据减1
    bnez $v0, loop #  $v0!= 0，继续循环
    li $v0, 4 #     打印字符串spim系统功能掉用号存入$v0
    la $a0, result # 将消息result地址存入 $a0
    syscall # 打印消息
    li $v0, 1 #打印整数spim系统功能掉用号存入$v0
    move $a0, $t0 #整数值存入 $a0
    syscall # 打印整数
    b main # 跳转到main标号
end: li $v0, 4 # 打印字符串spim系统功能掉用号存入$v0
    la $a0, bye # 将消息bye地址存入 $a0
    syscall #打印消息
    li $v0, 10 # 退出程序系统功能调用
    syscall #返回到spim内核
```

数据段

代码段

空白行表示程序的结束



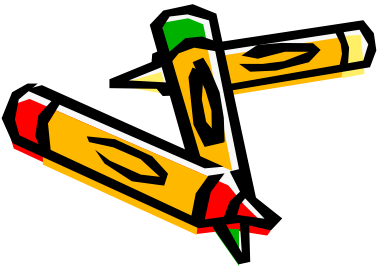
QtSpim系统功能调用

功能描述	功能号（$v0）	输入参数	输出参数
显示整数	1	$a0:整数值	
显示字符串直到字符串结束符0	4	$a0:字符串首地址	
读入整数	5		$v0:输入的整数值
读入字符串	8	$a0:内存空间首地址 $a1:内存空间长度	
退出	10		

QtSpim汇编、调试程序示例

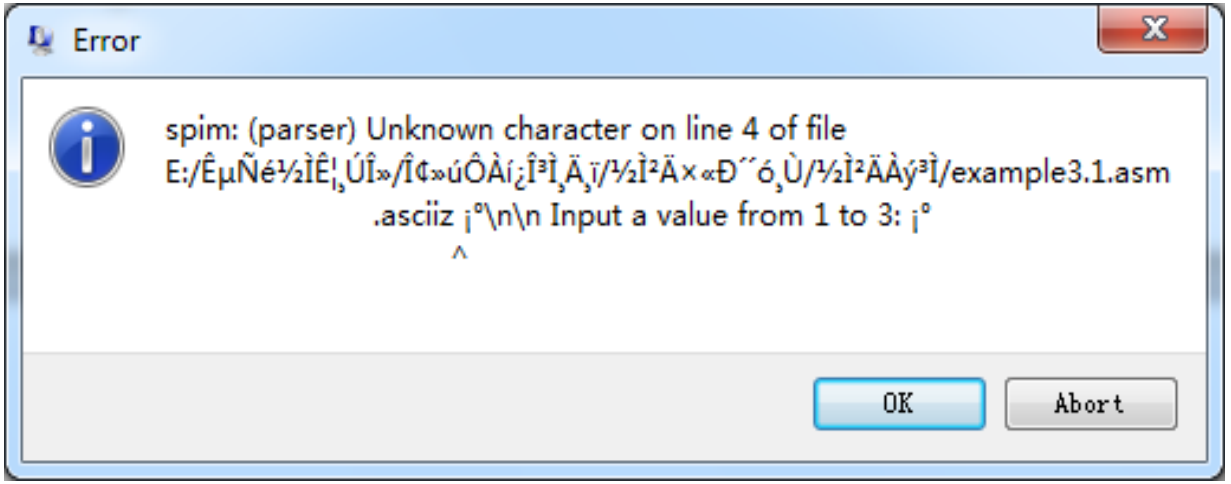
- 用户程序入口

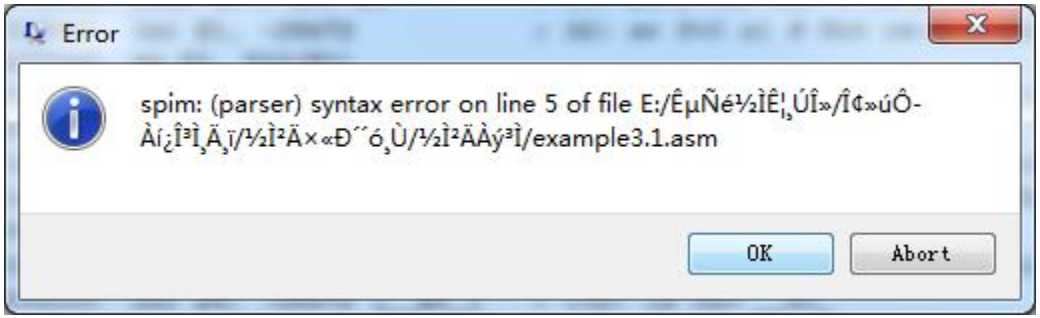
```
.text
.globl __start
__start:
lw $a0, 0($sp) # argc
addiu $a1, $sp, 4 # argv
addiu $a2, $a1, 4 # envp
sll $v0, $a0, 2
addu $a2, $a2, $v0
jal main
li $v0,10
syscall # 退出
```



汇编提示信息

- 装载汇编源程序时完成





查看内存映像

- （MIPS)大字节序、
- (PC)小字节序

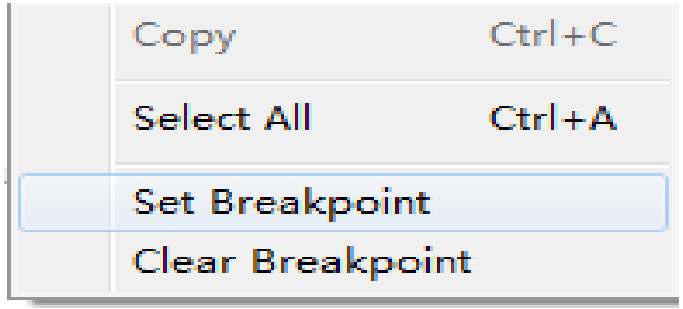
```
.data
        .align 2
jumptable: .word main, case1, case2, case3
prompt : .asciiz "\n\n Input a value from 1 to 3: "
```

变量名	地址	数据	定义值
jumptable	0x10010000	0x24	main
		0x00	
		0x40	
		0x00	
		0x60	case1
prompt	0x10010010	0x0a	\n
		0x0a	\n
		0x20	space
		0x49	I

```
User data segment [10000000]..[10040000]
[10000000]..[1000ffff]  00000000
[10010000]    00400024  00400060  00400068  00400070    $ . @ . ` . @ . h . @ . p .
[10010010]    49200a0a  7475706e  76206120  65756c61    . .   I n p u t   a   v a l
[10010020]    6f726620  2031206d  33206f74  0000203a      f r o m   1   t o   3 :
[10010030]..[1003ffff]  00000000
```

设置断点、单步执行，修改寄存器值

- 光标处

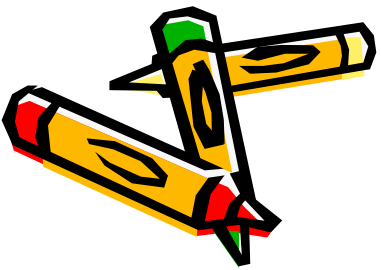



```
[00400068] 00108080  sll $16, $16, 2          ; 22: sll $s0, $s0, 2 #×óÒÆ2Î»
[0040006c] 04010002  bgez $0 8 [output-0x0040006c]; 23: b output
```

实验任务

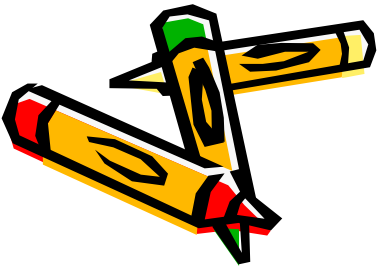
用汇编程序实现以下伪代码：要求采用移位指令实现乘除法运算。

```
int main()
{ int K, Y ;
int Z[50] ;
Y = 56;
for(K=0;K<50;K++)
Z[K] = Y - 16 * ( K / 4 + 210) ;
     }
```



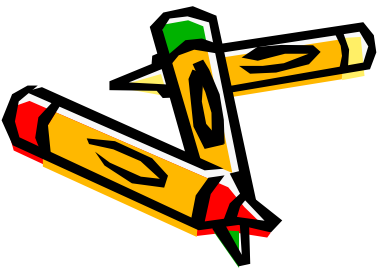
实验要求

- 完成汇编语言程序设计、调试、测试全过程
- 指出用户程序的内存映像，包括代码段和数据段
- 完成软件实验报告



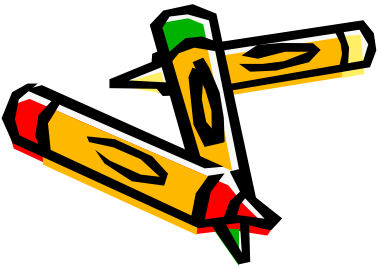
实验报告要求

1. 实验要求
2. **汇编**源程序设计思路（算法）、源代码及注释
3. 详细的调试、测试过程（可以截图并附加说明）
4. 程序内存映像（最后结果Z[k]（k=0~49）的内存映像需验收）
5. 心得体会



```
.DATA
Z:  .SPACE 200
.TEXT
MAIN:
LA $S3,Z
LI $T0,0
LI $T1,56

LI $V0,10
SYSCALL
```



Z	0x0012fe60
[0x0]	0xffffff318
[0x1]	0xffffff318
[0x2]	0xffffff318
[0x3]	0xffffff318
[0x4]	0xffffff308
[0x5]	0xffffff308
[0x6]	0xffffff308
[0x7]	0xffffff308
[0x8]	0xffffff2f8
[0x9]	0xffffff2f8
[0xa]	0xffffff2f8
[0xb]	0xffffff2f8
[0xc]	0xffffff2e8
[0xd]	0xffffff2e8
[0xe]	0xffffff2e8
[0xf]	0xffffff2e8
[0x10]	0xffffff2d8
[0x11]	0xffffff2d8
[0x12]	0xffffff2d8
[0x13]	0xffffff2d8
[0x14]	0xffffff2c8
[0x15]	0xffffff2c8
[0x16]	0xffffff2c8
[0x17]	0xffffff2c8
[0x18]	0xffffff2b8
[0x19]	0xffffff2b8
[0x1a]	0xffffff2b8
[0x1b]	0xffffff2b8
[0x1c]	0xffffff2a8
[0x1d]	0xffffff2a8
[0x1e]	0xffffff2a8
[0x1f]	0xffffff2a8
[0x20]	0xffffff298
[0x21]	0xffffff298
[0x22]	0xffffff298
[0x23]	0xffffff298