**华中科技大学**

**实验报告**

**课程名称：微机原理与接口技术实验**

**实验项目名称：汇编语言实验报告**

**指导老师：罗杰**

**专业/班级：电磁1802**

**学生学号：U201813405**

**学生姓名：吴叶赛**

**成绩/等级（批阅教师）：**

**实验执行日期： 2020年9月8日**

实验报告

**1，实验任务及要求**

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 ) ;

}

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

• 把Z[K]在屏幕上显示出来

1. **实验思路和代码**

思路；将伪代码翻译

################# Data segment #####################

.data

z:.space 200

str1:.asciiz "\nZ["

str2:.asciiz "]="

################# Code segment #####################

.text

main: # K $s0;Y $s1;Z $s2;$t2 zk的地址

addi $s1,$0,56 # y=56

add $s0,$0,$0 # k=0

la $s2,z # z的首地址

loop:

beq $s0,50,exit # k=50则退出

sll $t1,$s0,2 #z的偏移地址

add $t2,$t1,$s2 # zk的物理地址

srl $t3,$s0,2

addi $t3,$t3,210

sll $t3,$t3,4

sub $t3,$s1,$t3 #计算结果存入$t3

sw $t3,0($t2) #结果存入内存

li $v0,4

la $a0,str1

syscall

li $v0,1

add $a0,$0,$s0 #显示K

syscall

li $v0,4

la $a0,str2

syscall

li $v0,1

lw $a0,0($t2) #显示Z【k】

syscall

addi $s0,$s0,1 #K+1

j loop

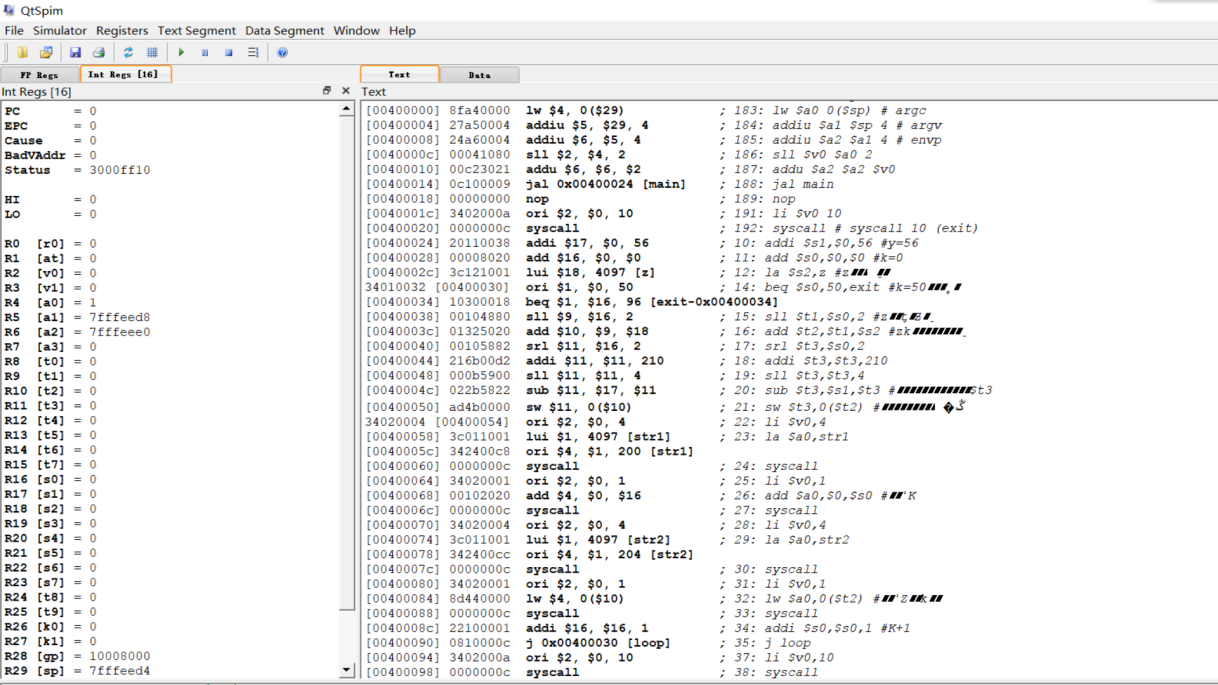
exit:

li $v0,10

syscall

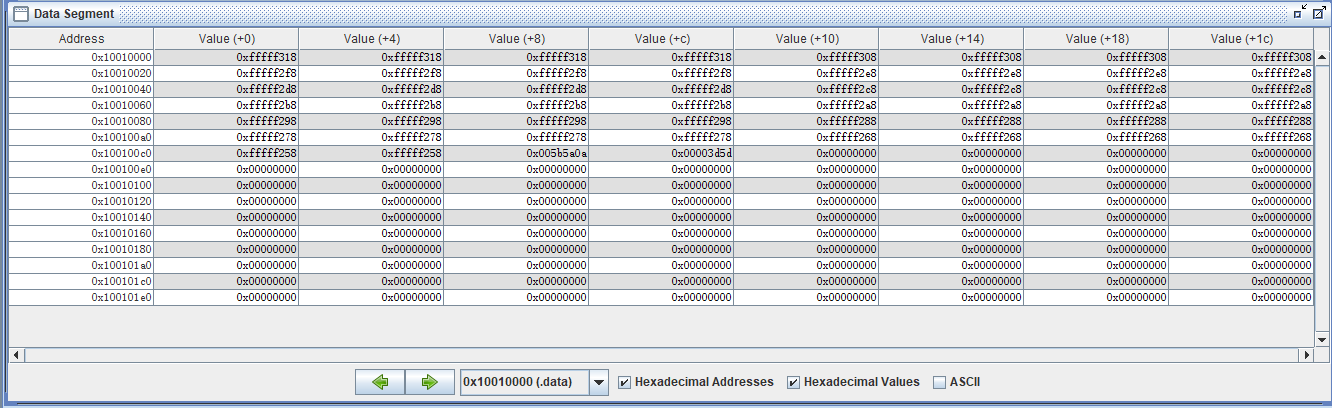
1. **测试过程**

（1）将写好的代码载入QtSpim，点击运行后可以看见汇编指令的机器码和寄存器的值，可以根据寄存器的最终值验证代码有无问题。



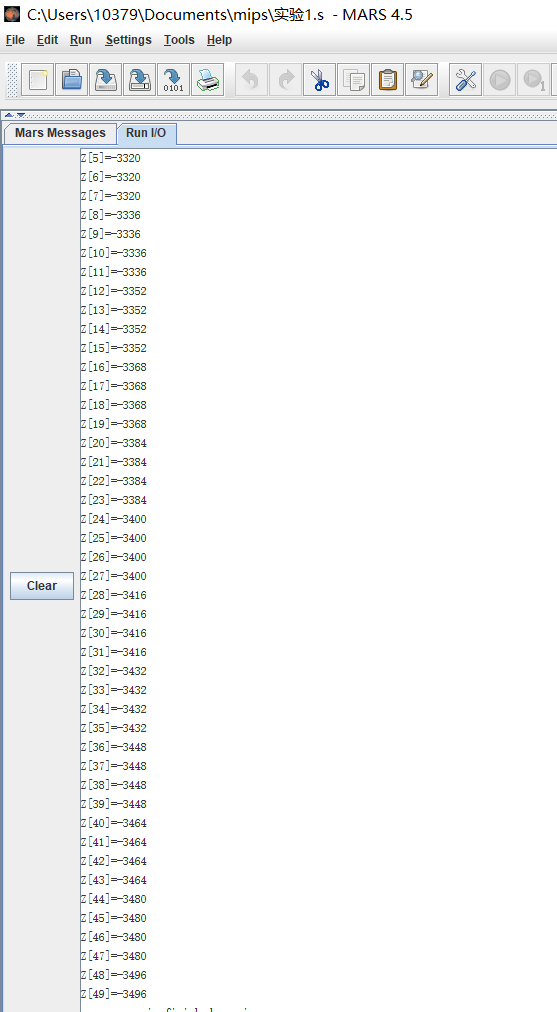
图一 代码段及寄存器情况

（2）随后将将代码在Mars上运行，可以看到数据段的结果。其中0x10010000-0x100100c4是数组Z[k]的值，而0x100100c8和0x100100cc中存放的是我提前定义的值。



图二 数据段

（3）在Mars的run I/O中查看结果，结果无误。



图三 实验结果

**4. 程序内存映像**

**数据段**

|  |  |  |  |
| --- | --- | --- | --- |
| 变量名 | 地址 | 数据 | 定义值 |
| Z[0] | 0x10010000 | 0xfffff318 | -3304 |
| Z[1] | 0x10010004 | 0xfffff318 | -3304 |
| Z[2] | 0x10010008 | 0xfffff318 | -3304 |
| Z[3] | 0x1001000c | 0xfffff318 | -3304 |
| Z[4] | 0x10010010 | 0xfffff308 | -3320 |
| Z[5] | 0x10010014 | 0xfffff308 | -3320 |
| Z[6] | 0x10010018 | 0xfffff308 | -3320 |
| Z[7] | 0x1001001c | 0xfffff308 | -3320 |
| Z[8] | 0x10010020 | 0xfffff2f8 | -3336 |
| Z[9] | 0x10010024 | 0xfffff2f8 | -3336 |
| Z[10] | 0x10010028 | 0xfffff2f8 | -3336 |
| Z[11] | 0x1001002c | 0xfffff2f8 | -3336 |
| Z[12] | 0x10010030 | 0xfffff2e8 | -3352 |
| Z[13] | 0x10010034 | 0xfffff2e8 | -3352 |
| Z[14] | 0x10010038 | 0xfffff2e8 | -3352 |
| Z[15] | 0x1001003c | 0xfffff2e8 | -3352 |
| Z[16] | 0x10010040 | 0xfffff2d8 | -3368 |
| Z[17] | 0x10010044 | 0xfffff2d8 | -3368 |
| Z[18] | 0x10010048 | 0xfffff2d8 | -3368 |
| Z[19] | 0x1001004c | 0xfffff2d8 | -3368 |
| Z[20] | 0x10010050 | 0xfffff2c8 | -3384 |
| Z[21] | 0x10010054 | 0xfffff2c8 | -3384 |
| Z[22] | 0x10010058 | 0xfffff2c8 | -3384 |
| Z[23] | 0x1001005c | 0xfffff2c8 | -3384 |
| Z[24] | 0x10010060 | 0xfffff2b8 | -3400 |
| Z[25] | 0x10010064 | 0xfffff2b8 | -3400 |
| Z[26] | 0x10010068 | 0xfffff2b8 | -3400 |
| Z[27] | 0x1001006c | 0xfffff2b8 | -3400 |
| Z[28] | 0x10010070 | 0xfffff2a8 | -3416 |
| Z[29] | 0x10010074 | 0xfffff2a8 | -3416 |
| Z[30] | 0x10010078 | 0xfffff2a8 | -3416 |
| Z[31] | 0x1001007c | 0xfffff2a8 | -3416 |
| Z[32] | 0x10010080 | 0xfffff298 | -3432 |
| Z[33] | 0x10010084 | 0xfffff298 | -3432 |
| Z[34] | 0x10010088 | 0xfffff298 | -3432 |
| Z[35] | 0x1001008c | 0xfffff298 | -3432 |
| Z[36] | 0x10010090 | 0xfffff288 | -3448 |
| Z[37] | 0x10010094 | 0xfffff288 | -3448 |
| Z[38] | 0x10010098 | 0xfffff288 | -3448 |
| Z[39] | 0x1001009c | 0xfffff288 | -3448 |
| Z[40] | 0x100100a0 | 0xfffff278 | -3464 |
| Z[41] | 0x100100a4 | 0xfffff278 | -3464 |
| Z[42] | 0x100100a8 | 0xfffff278 | -3464 |
| Z[43] | 0x100100ac | 0xfffff278 | -3464 |
| Z[44] | 0x100100b0 | 0xfffff268 | -3480 |
| Z[45] | 0x100100b4 | 0xfffff268 | -3480 |
| Z[46] | 0x100100b8 | 0xfffff268 | -3480 |
| Z[47] | 0x100100bc | 0xfffff268 | -3480 |
| Z[48] | 0x100100c0 | 0xfffff258 | -3496 |
| Z[49] | 0x100100c4 | 0xfffff258 | -3496 |

**代码段**

|  |  |  |
| --- | --- | --- |
| 00400000 | lw $4, 0($29) | 8fa40000 |
| 00400004 | addiu $5, $29, 4 | 27a50004 |
| 00400008 | addiu $6, $5, 4 | 24a60004 |
| 0040000c | sll $2, $4, 2 | 00041080 |
| 00400010 | addu $6, $6, $2 | 00c23021 |
| 00400014 | jal 0x00400024 | 0c100009 |
| 00400018 | nop | 00000000 |
| 0040001c | ori $2, $0, 10 | 3402000a |
| 00400020 | syscall | 0000000c |
| 00400024 | addi $17, $0, 56 | 20110038 |
| 00400028 | add $16, $0, $0 | 00008020 |
| 0040002c | lui $18, 4097 [z] | 3c121001 |
| 00400030 | ori $1, $0, 50 | 34010032 |
| 00400034 | beq $1, $16, 96 [exit-0x00400034] | 10300018 |
| 00400038 | sll $9, $16, 2 | 00104880 |
| 0040003c | add $10, $9, $18 | 01325020 |
| 00400040 | srl $11, $16, 2 | 00105882 |
| 00400044 | addi $11, $11, 210 | 216b00d2 |
| 00400048 | sll $11, $11, 4 | 000b5900 |
| 0040004c | sub $11, $17, $11 | 022b5822 |
| 00400050 | sw $11, 0($10) | ad4b0000 |
| 00400054 | ori $2, $0, 4 | 34020004 |
| 00400058 | lui $1, 4097 [str1] | 3c011001 |
| 0040005c | ori $4, $1, 200 [str1] | 342400c8 |
| 00400060 | syscall | 0000000c |
| 00400064 | ori $2, $0, 1 | 34020001 |
| 00400068 | add $4, $0, $16 | 00102020 |
| 0040006c | syscall | 0000000c |
| 00400070 | ori $2, $0, 4 | 34020004 |
| 00400074 | lui $1, 4097 [str2] | 3c011001 |
| 00400078 | ori $4, $1, 204 [str2] | 342400cc |
| 0040007c | syscall | 0000000c |
| 00400080 | ori $2, $0, 1 | 34020001 |
| 00400084 | lw $4, 0($10) | 8d440000 |
| 00400088 | syscall | 0000000c |
| 0040008c | addi $16, $16, 1 | 22100001 |
| 00400090 | j 0x00400030 [loop] | 0810000c |
| 00400094 | ori $2, $0, 10 | 3402000a |
| 00400098 | syscall | 0000000c |

**5.心得体会**

汇编语言编程，因为不熟悉的原因，首先要把思路写成c代码然后再从转变成汇编原因。另外汇编语言比更加底层也更繁琐。比如在显示的Z[k]的指令，c只需要printf一行代码；而汇编语言则需要调用3次系统功能函数。但是它可以查看寄存器的状态和值，这是c语言做不到的。

并且，通过这次实验，我对汇编语言的认识加深了。