实验报告

实验名称：MIPS 汇编程序设计

院系：

数据科学与计算机学院

软件工程(移动信息工程)

班级： 1518

姓名： 张镓伟

学号：15352408

指导老师：郭雪梅

一、 实验目的：

1.熟悉 MIPS 汇编程序开发环境，学习使用 MARs 工具。知道如何查看内存空间分配。

2. 了解 C 语言语句与汇编指令之间的关系。

3. 掌握 MIPS 汇编程序设计， 掌握 MARs 的调试技术。

4. 了解 MIPS 汇编语言与机器语言之间的对应关系 。

5. 熟悉常见的 MIPS 汇编指令

6. 掌握程序的内存映像。

二、实验内容

1.用汇编程序实现以下伪代码：要求使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);
}
```

说明：Int Z[n];要求每个同学按照自己的学号末尾 2 位选择 n,末尾 40-50 之间不变，其他用末尾个位数加 40.

三、程序设计及分析

1.C 语言分析：

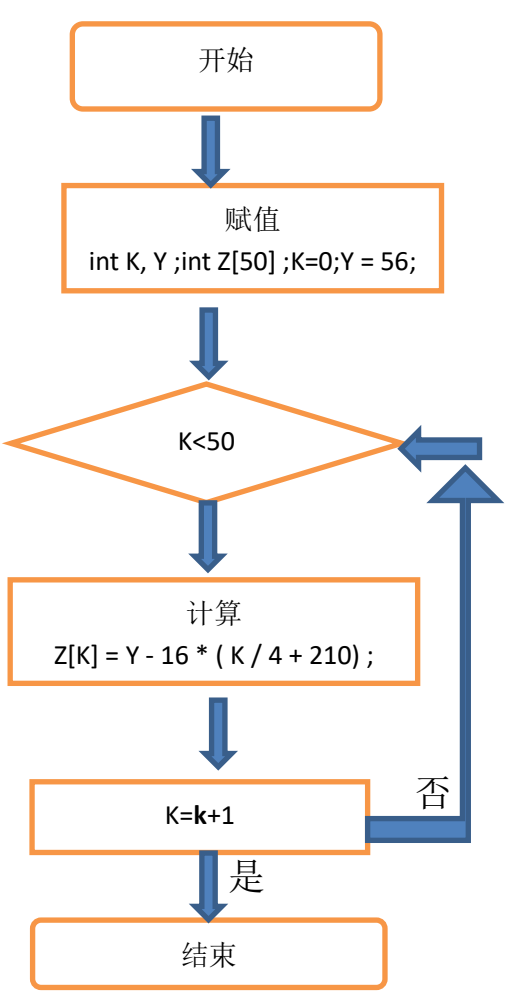
有两个变量是 int 型，一个数组型；还有一个循环执行过程。

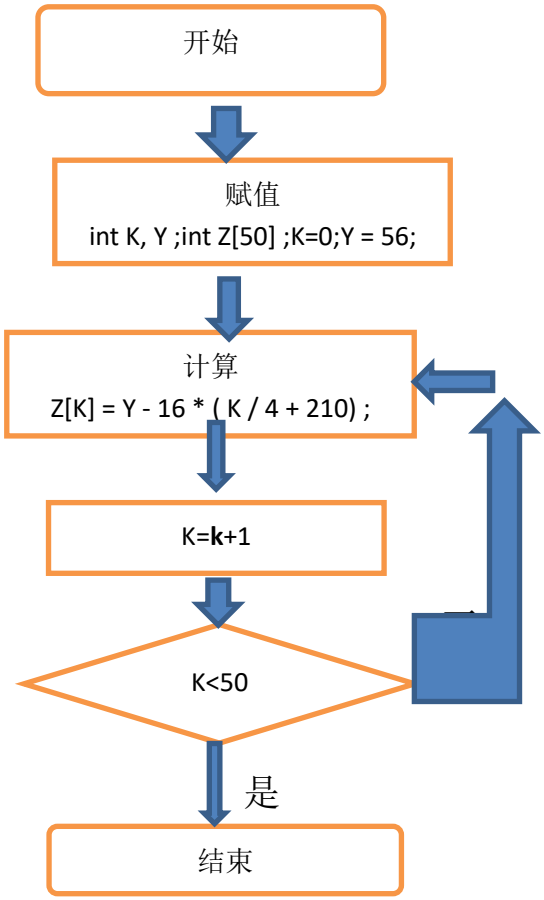
2.汇编程序实现分析：

首先需要定义用户数据段，获得一个内存空间作为数组空间。再选定几个寄存器作为 K，Y 以及输出，其中输出和 Y 可以合用一个寄存器。

3.设计思路：

分配完空间地址后，最重要的是完成循环控制。循环控制有两个思路：可以是先判断后循环；或者是先循环后判断即如图





slti $t2,$t0,50 #判断 k 是否小于 50

beq $t2,$0（$t2=1 循环，否则结束。）

slti $t2,$t0,50 #判断 k 是否小于 50，
beq $t2,$0，#是则结束
#否,循环

四、程序实现及调试分析

1. 汇编程序代码实现：

方法一

```
.data 0x0
z:.space 192 #学号15352408，则int z[48],占内存4*48=192
str:.asciiz " " #输出间隔
.text 0x3000
main:
    la $s0,z #$s0，表示z[k]的地址
    li $t0,0 #$t0为k，初始为0
    li $t1,56 #$t2为Y，初始值56
loop:
    slti $t2,$t0,48 #判断k是否于48
    beq $t2,$0,exit #当k大于等于50,循环结束,退出
    srl $t3,$t0,2  #k/4
    addi $t3,$t3,210 #k/4+210
    sll $t3,$t3,4 #16*(k/4+210)
    sub $t3,$t1,$t3 #y-16*(k/4+210)
    sw  $t3,0($s0) #写进z[k]

    li $v0,1  #输出
    addi $a0,$s0,0
    syscall
```

```
    li $v0,4  #输出间隔
    la  $a0,str
    syscall

    addi $s0,$s0,4 #地址下一位
    addi $t0,$t0,1 #k++
    j loop  #循环
exit:
    li $v0,10
    syscall
```

方法二

```
.data 0x0
z:.space 192 #学号15352408，则int z[48],占内存4*48=192
str:.asciiz " " #输出间隔
.text 0x3000
main:
    la $s0,z #$s0，表示z[k]的地址
    li $t0,0 #$t0为k，初始为0
    li $t1,56 #$t2为Y，初始值56
loop:
    srl $t3,$t0,2  #k/4
    addi $t3,$t3,210 #k/4+210
    sll $t3,$t3,4 #16*(k/4+210)
    sub $t3,$t1,$t3 #y-16*(k/4+210)
    sw  $t3,0($s0) #写进z[k]

    li $v0,1  #输出
    addi $a0,$s0,0
    syscall
```

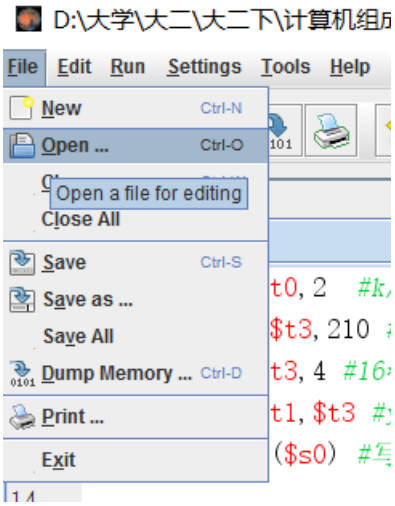
```
    li $v0,4  #输出间隔
    la  $a0,str
    syscall

    addi $s0,$s0,4 #地址下一位
    addi $t0,$t0,1 #k++

    slti $t2,$t0,48 #判断k是否于48
    beq $t2,$0,exit #当k大于等于50,循环结束,退出
    j loop  #循环
exit:
    li $v0,10
    syscall
```

2.调试过程

1.编写程序：详细见代码

2.装载程序

点开 MARS 左上角的 File，选择 open，找到后缀为 asm 的写好的汇编源代码文件打开即可。



3.如果没有错误，便运行。

运行之后点击不同的窗口便可得到我们想要的结果。详细结果如下图

内存占用情况映像

经比较，两种方法内存占用情况一样

Data:

Data Segment

Address	Value (+0)	Value (+4)	Value (+8)	Valu
0x00000000	0xfffff318	0xfffff318	0xfffff318	
0x00000020	0xfffff2f8	0xfffff2f8	0xfffff2f8	
0x00000040	0xfffff2d8	0xfffff2d8	0xfffff2d8	
0x00000060	0xfffff2b8	0xfffff2b8	0xfffff2b8	
0x00000080	0xfffff298	0xfffff298	0xfffff298	
0x000000a0	0xfffff278	0xfffff278	0xfffff278	
0x000000c0	0x00000020	0x00000000	0x00000000	

e (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0xfffff318	0xfffff308	0xfffff308	0xfffff308	0xfffff308
0xfffff2f8	0xfffff2e8	0xfffff2e8	0xfffff2e8	0xfffff2e8
0xfffff2d8	0xfffff2c8	0xfffff2c8	0xfffff2c8	0xfffff2c8
0xfffff2b8	0xfffff2a8	0xfffff2a8	0xfffff2a8	0xfffff2a8
0xfffff298	0xfffff288	0xfffff288	0xfffff288	0xfffff288
0xfffff278	0xfffff268	0xfffff268	0xfffff268	0xfffff268

Text:

Address	Value (+0)	Value (+4)	Value (+8)	Value
0x00003000	0x20100000	0x24080000	0x24090038	
0x00003020	0x24020001	0x22040000	0x0000000c	
0x00003040	0x290a0030	0x11400001	0x08000c03	
0x00003060	0x00000000	0x00000000	0x00000000	
0x00003080	0x00000000	0x00000000	0x00000000	

(+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x00085882	0x216b00d2	0x000b5900	0x012b5822	0xae0b0000
0x24020004	0x200400c0	0x0000000c	0x22100004	0x21080001
0x2402000a	0x0000000c	0x00000000	0x00000000	0x00000000

分析：由图可知数组地址从 0Xfffff318—0Xfffff268;每行有四个是一样的，总共 48 个地址。这是因为数组含有 48 个元素，而 int 型数据占 4 个字节空间，所以连续四个地址是相同的.

数据段内存映像

Data Segment

Address	Value (+0)	Value (+4)	Value (+8)	Valu
0x00000000	0xfffff318	0xfffff318	0xfffff318	
0x00000020	0xfffff2f8	0xfffff2f8	0xfffff2f8	
0x00000040	0xfffff2d8	0xfffff2d8	0xfffff2d8	
0x00000060	0xfffff2b8	0xfffff2b8	0xfffff2b8	
0x00000080	0xfffff298	0xfffff298	0xfffff298	
0x000000a0	0xfffff278	0xfffff278	0xfffff278	
0x000000c0	0x00000020	0x00000000	0x00000000	

e (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0xfffff318	0xfffff308	0xfffff308	0xfffff308	0xfffff308
0xfffff2f8	0xfffff2e8	0xfffff2e8	0xfffff2e8	0xfffff2e8
0xfffff2d8	0xfffff2c8	0xfffff2c8	0xfffff2c8	0xfffff2c8
0xfffff2b8	0xfffff2a8	0xfffff2a8	0xfffff2a8	0xfffff2a8
0xfffff298	0xfffff288	0xfffff288	0xfffff288	0xfffff288
0xfffff278	0xfffff268	0xfffff268	0xfffff268	0xfffff268

运行结果显示：

两种方法的运行结果一致如下图(由于一行太长，所以我截图成两行显示)

```
— program is finished running —

0 4 8 12 16 20 24 28 32 36 40 44 48 52 56 60 64 68 72 76 80 84 88 92 96 100
— program is finished running —
```

```
104 108 112 116 120 124 128 132 136 140 144 148 152 156 160 164 168 172 176 180 184 188
```

代码段内存映像

方法一：

Address	Code	Basic	
0x00003000	0x20100000	addi $16,$0,0x00000000	5: la $s0,z #$s0，表示z[k]的地址
0x00003004	0x24080000	addiu $8,$0,0x00000000	6: li $t0,0 #$t0为k，初始为0
0x00003008	0x24090038	addiu $9,$0,0x00000038	7: li $t1,56 #$t2为Y，初始值56
0x0000300c	0x290a0030	slti $10,$8,0x00000030	9: slti $t2,$t0,48 #判断k是否于48
0x00003010	0x1140000e	beq $10,$0,0x0000000e	10: beq $t2,$0,exit #当k大于等于50,循环结束,退出
0x00003014	0x00085882	srl $11,$8,0x00000002	11: srl $t3,$t0,2 #k/4
0x00003018	0x216b00d2	addi $11,$11,0x000000d2	12: addi $t3,$t3,210 #k/4+210
0x0000301c	0x000b5900	sll $11,$11,0x00000004	13: sll $t3,$t3,4 #16*(k/4+210)
0x00003020	0x012b5822	sub $11,$9,$11	14: sub $t3,$t1,$t3 #y-16*(k/4+210)
0x00003024	0xae0b0000	sw $11,0x00000000($16)	15: sw $t3,0($s0) #写进z[k]
0x00003028	0x24020001	addiu $2,$0,0x00000001	17: li $v0,1 #输出
0x0000302c	0x22040000	addi $4,$16,0x00000000	18: addi $a0,$s0,0
0x00003030	0x0000000c	syscall	19: syscall
0x00003034	0x24020004	addiu $2,$0,0x00000004	21: li $v0,4 #输出间隔
0x00003038	0x200400c0	addi $4,$0,0x000000c0	22: la $a0,str
0x0000303c	0x0000000c	syscall	23: syscall
0x00003040	0x22100004	addi $16,$16,0x00000004	25: addi $s0,$s0,4 #地址下一位
0x00003044	0x21080001	addi $8,$8,0x00000001	26: addi $t0,$t0,1 #k++
0x00003048	0x08000c03	j 0x0000300c	27: j loop #循环
0x0000304c	0x2402000a	addiu $2,$0,0x0000000a	29: li $v0,10
0x00003050	0x0000000c	syscall	30: syscall

方法二：

Address	Code	Basic	
0x00003000	0x20100000	addi $16,$0,0x00000000	5: la $s0,z #$s0，表示z[k]的地址
0x00003004	0x24080000	addiu $8,$0,0x00000000	6: li $t0,0 #$t0为k，初始为0
0x00003008	0x24090038	addiu $9,$0,0x00000038	7: li $t1,56 #$t2为Y，初始值56
0x0000300c	0x00085882	srl $11,$8,0x00000002	9: srl $t3,$t0,2 #k/4
0x00003010	0x216b00d2	addi $11,$11,0x000000d2	10: addi $t3,$t3,210 #k/4+210
0x00003014	0x000b5900	sll $11,$11,0x00000004	11: sll $t3,$t3,4 #16*(k/4+210)
0x00003018	0x012b5822	sub $11,$9,$11	12: sub $t3,$t1,$t3 #y-16*(k/4+210)
0x0000301c	0xae0b0000	sw $11,0x00000000($16)	13: sw $t3,0($s0) #写进z[k]
0x00003020	0x24020001	addiu $2,$0,0x00000001	15: li $v0,1 #输出
0x00003024	0x22040000	addi $4,$16,0x00000000	16: addi $a0,$s0,0
0x00003028	0x0000000c	syscall	17: syscall
0x0000302c	0x24020004	addiu $2,$0,0x00000004	19: li $v0,4 #输出间隔
0x00003030	0x200400c0	addi $4,$0,0x000000c0	20: la $a0,str
0x00003034	0x0000000c	syscall	21: syscall
0x00003038	0x22100004	addi $16,$16,0x00000004	23: addi $s0,$s0,4 #地址下一位
0x0000303c	0x21080001	addi $8,$8,0x00000001	24: addi $t0,$t0,1 #k++
0x00003040	0x290a0030	slti $10,$8,0x00000030	26: slti $t2,$t0,48 #判断k是否于48
0x00003044	0x11400001	beq $10,$0,0x00000001	27: beq $t2,$0,exit #当k大于等于50,循环结束,退出
0x00003048	0x08000c03	j 0x0000300c	28: j loop #循环
0x0000304c	0x2402000a	addiu $2,$0,0x0000000a	30: li $v0,10
0x00003050	0x0000000c	syscall	31: syscall

实验总结

1.本次实验我对汇编代码的编写有了更进一步的了解，对汇编的各种指令有了更清晰的理解。

2.本次实验主要练习的是 C 与汇编的转换。至此我发现高级语言

实在是方便许多，将 C 翻译成汇编代码一定要很清楚地明白各个寄存器的意思，逻辑和思维要清晰，否则很容易陷入逻辑混乱而不知所措。这个时候，适当地注释显得极其重要，我也是因为写了注释才没有被搞晕。