冒泡排序实验报告

PB20000328 杨博涵

一．实现思路

根据 C 语言中的冒泡排序法如下

```
int main()
{
    int i,j,v[N]={4,3,6,3,6,8,2,5,7,2,2},m;
    for(i=0;i<N;i++)
    {
        for(j=0;j<N-i;j++)
            if(v[j]<v[j+1]) {m=v[j+1];v[j+1]=v[j];v[j]=m;}
    }
}
```

将其手动编译成汇编即可。

二．实验代码

```
.data
dp: .word 20,28,14,17,24,30,40,11,23,24
newline:    .string         "\n"
delimiter:  .string         ", "
.text
#打印
    la s2,dp
    li t3,10
    and t5,t5,zero
print:
    li a7,1
    lw a0,0(s2)
    ecall
    li a7, 4
    la a0, delimiter
    ecall
    addi t5,t5,1
    addi s2,s2,4
    bge t5,t3,over
    jal print
over:
    jal printNewline

#s0,s1 存储循环变量,s2 存 dp
```

```
        and s0,s0,zero
big:    and s1,s1,zero
            la s2,dp
        small:      lw t1,0(s2)
                    lw t2,4(s2)
                    bge t1,t2,br
                        sw t1,4(s2)
                        sw t2,0(s2)
            br: addi s2,s2,4
                    addi s1,s1,1
                    add t4,s0,s1
                    bge t4,t3,tran
                    jal small

tran:addi s0,s0,1
            bge s0,t3,p
            jal big

#打印
p:      la s2,dp
        li t3,10
        and t5,t5,zero
print2:
        li a7,1
        lw a0,0(s2)
        ecall
        li a7, 4
        la a0, delimiter
        ecall
        addi t5,t5,1
        addi s2,s2,4
        bge t5,t3,end
        jal print2
end:
        jal printNewline

        li a7,10
        ecall



printNewline:
        la a0, newline
        li a7, 4
```

ecall
jr x1

我们需要用到两个循环，分别用 S0，S1 寄存器存放。S2 作为数据指针。其余寄存器均为临时寄存。

我们选择 20,28,14,17,24,30,40,11,23,24 这串数字进行降序排序。截图如下。

Console

```
20, 28, 14, 17, 24, 30, 40, 11, 23, 24,
40, 30, 28, 24, 24, 23, 20, 17, 14, 11,

Program exited with code: 0
```

Address	Word	Byte 0	Byte
0x10000034	X	X	X
0x10000030	X	X	X
0x1000002c	0x0000202c	0x2c	0x2
0x10000028	0x0000000a	0x0a	0x0
0x10000024	0x0000000b	0x0b	0x0
0x10000020	0x0000000e	0x0e	0x0
0x1000001c	0x00000011	0x11	0x0
0x10000018	0x00000014	0x14	0x0
0x10000014	0x00000017	0x17	0x0
0x10000010	0x00000018	0x18	0x0
0x1000000c	0x00000018	0x18	0x0
0x10000008	0x0000001c	0x1c	0x0
0x10000004	0x0000001e	0x1e	0x0
0x10000000	0x00000028	0x28	0x0
0x0ffffffc	X	X	X

在编程时比较容易把一些寄存器初始化和归位的步骤遗忘，比如 S2 在大循环之后需要复位重新指向，但是如果遗忘复位会直接溢出，问题已解决。