

《计算机系统》

实验一实验报告

班级：计科 2104

学号：202108010426

姓名：冷长佼

基础实验：原型机

【实验目的】

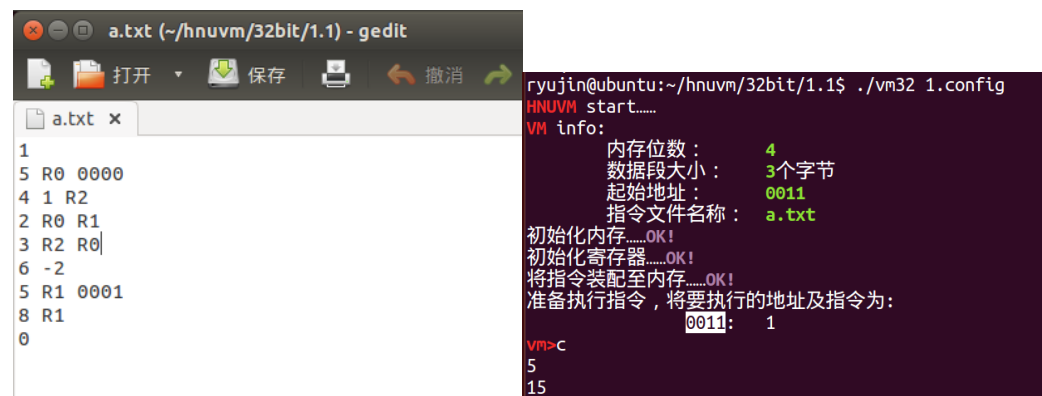
- (1) 了解冯诺伊曼体系结构；
- (2) 理解指令集结构及其作用；
- (3) 理解计算机的运行过程，就是指令的执行过程，并初步掌握调试方法。

【实验准备】

- (1) 阅读教材，掌握冯诺伊曼体系的相关内容；
- 学习课程《最小系统与原型机 I》。

实验 1.1 原型机 I

Config1



The screenshot shows a gedit window with a file named a.txt. The file contains the following assembly code:

```
1
5 R0 0000
4 1 R2
2 R0 R1
3 R2 R0
6 -2
5 R1 0001
8 R1
0
```

Next to it is a terminal window showing the execution of the program. The terminal output is as follows:

```
ryujin@ubuntu:~/hnuvm/32bit/1.1$ ./vm32 1.config
HNUVM start.....
VM info:
    内存位数:      4
    数据段大小:    3个字节
    起始地址:      0011
    指令文件名称:  a.txt
初始化内存.....OK!
初始化寄存器.....OK!
将指令装配至内存.....OK!
准备执行指令, 将要执行的地址及指令为:
    0011:  1
VM>C
5
15
```

功能：输入 n，输出 $1+2+\dots+n$

Config2



The screenshot shows a gedit window with a file named b.txt. The file contains the following assembly code:

```
1
5 R0 0000
5 R0 R1
1
5 R0 0001
3 R1 R0
6 3
5 0000 0010
7 2
5 0001 0010
5 0010 R0
8 R0
0
```

Next to it is a terminal window showing the execution of the program. The terminal output is as follows:

```
ryujin@ubuntu:~/hnuvm/32bit/1.1$ ./vm32 2.config
HNUVM start.....
VM info:
    内存位数:      4
    数据段大小:    3个字节
    起始地址:      0011
    指令文件名称:  b.txt
初始化内存.....OK!
初始化寄存器.....OK!
将指令装配至内存.....OK!
准备执行指令, 将要执行的地址及指令为:
    0011:  1
VM>C
3
5
5
VM>x 7 0000
0000:  00000011
0001:  00000101
0010:  00000101
0011:  1
0100:  5 R0 0000
0101:  5 R0 R1
0110:  1
```

```

ryujin@ubuntu:~/hnuvm/32bit/1.1$ ./vm32 2.config
HNUVM start.....
VM info:
    内存位数:      4
    数据段大小:    3个字节
    起始地址:      0011
    指令文件名称:  b.txt
初始化内存.....OK!
初始化寄存器.....OK!
将指令装配至内存.....OK!
准备执行指令, 将要执行的地址及指令为:
    0011:  1

VM>C
5
3
5
VM>X 5 0000
0000:  00000101
0001:  00000011
0010:  00000101
0011:  1
0100:  5 R0 0000

```

功能：输入两个数，输出较大的那个数

Config3

```

c.txt (~/.hnuvm/32bit/1.1) - gedit
1
5 R0 00000
1
5 00000 00001
5 00001 R1
3 R0 R1
6 2
7 7
5 00010 R2
4 1 R3
2 R3 R2
5 R2 00010
5 R1 00001
7 -9
5 00001 R1
5 R0 R2
3 R1 R2
6 5
5 00010 R2
4 1 R3
2 R3 R2
5 R2 00010
5 00010 R1
8 R1
0

```

```

ryujin@ubuntu:~/hnuvm/32bit/1.1$ ./vm32 3.config
HNUVM start.....
VM info:
    内存位数:      5
    数据段大小:    3个字节
    起始地址:      00011
    指令文件名称:  c.txt
初始化内存.....OK!
初始化寄存器.....OK!
将指令装配至内存.....OK!
准备执行指令, 将要执行的地址及指令为:
    00011:  1

VM>C
9
3
3

```

功能：实现两个数之间的除法

思考问题

- (1) 如果基于这些指令实现两个整数的乘法与除法？

```

1Mul.txt (~/.hnuvm/32bit/1.2) - gedit
1Mul.txt x
1
5 R0 0000
1
4 1 R2
5 0000 R3
2 R3 R1
3 R2 R0
6 -3
8 R1
0

```

```

ryujin@ubuntu:~/hnuvm/32bit/1.2$ ./vm32 1.config
HNUVM start.....
VM info:
    内存位数:      4
    数据段大小:    3个字节
    起始地址:      0011
    指令文件名称:  1Mul.txt
初始化内存.....OK!
初始化寄存器.....OK!
将指令装配至内存.....OK!
准备执行指令, 将要执行的地址及指令为:
    0011:  1

VM>C
3
7
21

```

基于指令实现乘法



基于指令实现除法

(2) 原型机 I 的指令集是否完备? 如果是, 那么如何证明 (提示: 搜索并阅读“可计算性理论”)? 如果不是, 那么要增加哪些指令? 不完备, 对于算数运算指令, 还需要增加加一, 减一, 比较三个指令。

实验 1.2 原型机 II-扩充指令集

【实验目的】

- (1) 理解指令集结构及其作用;
- (2) 理解计算机的运行过程, 对指令集进行修改;

【实验准备】

- (1) 阅读教材, 掌握冯诺伊曼体系的相关内容;
- (2) 学习《最小系统与原型机 I》内容, 完成实验 1.1

基于 cpu.c 文件修改, 得到乘法和除法的指令 9 和 f 乘法:

```

void ExecuteMul(char source[], char dest[], int *result)
{
    char op;
    if(0==strcmp(source, "R0"))
        op=R0;
    else if(0==strcmp(source, "R1"))
        op=R1;
    else if(0==strcmp(source, "R3"))
        op=R3;
    else
        *result=-1;
    if(0==strcmp(dest, "R0\n"))
        R0*=op;
    else if(0==strcmp(dest, "R1\n"))
        R1*=op;
    else if(0==strcmp(dest, "R2\n"))
        R2*=op;
    else if(0==strcmp(dest, "R3\n"))
        R3*=op;
    else
        *result=-1;
}

```

```

        case '9':
            split(instruction_buffer, " ", revbuf, &num);
            if(3>num)
                *result=-1; //出错
            else
                ExecuteMul(revbuf[1], revbuf[2], result);
            if(*result!=-1) *result=2;
            PC++;
            break;

```

```

1.config x Din.txt x Mul.txt x
1
5 R0 R1
1
9 R0 R1
8 R1
0

```

```

ryujin@ubuntu:~/hnuvm/32bit/1.2$ ./vm32 2.config
HNUVM start.....
VM info:
    内存位数:      4
    数据段大小:    3个字节
    起始地址:      0011
    指令文件名称:  Mul.txt
初始化内存.....OK!
初始化寄存器.....OK!
将指令装配至内存.....OK!
准备执行指令, 将要执行的地址及指令为:
    0011: 1
VM>C
3
34
102

```

除法:

```
void ExecuteDin(char source[],char dest[],int *result)
{
    char op;
    if(0==strcmp(source,"R0"))
        op=R0;
    else if(0==strcmp(source,"R1"))
        op=R1;
    else if(0==strcmp(source,"R3"))
        op=R3;
    else
        *result=-1;
    if(0==strcmp(dest,"R0\n"))
        R0/=op;
    else if(0==strcmp(dest,"R1\n"))
        R1/=op;
    else if(0==strcmp(dest,"R2\n"))
        R2/=op;
    else if(0==strcmp(dest,"R3\n"))
        R3/=op;
    else
        *result=-1;
}
```

```
        break;
    case 'f':
        split(instruction_buffer," ",revbuf,&num);
        if(3>num)
            *result=-1; //出错
        else
            ExecuteDin(revbuf[1],revbuf[2],result);
        if(*result!=-1) *result=2;
        PC++;
        break;
```

2. 思考问题

(1) 原型机 I 与原型机 II 完成乘法和除法操作的方式有何不同?

原型机 1 利用基础的加减法来实现整数的乘除法, 原型机 2 是基于 cpu.c 的 C 语言代码来实现乘除法。

(2) 在指令集中增加乘法、除法等指令时, 原型机中需要增加代码, 那么硬件实现上需要增加什么样的部件?

需要增加晶体管和电子元件。

(3) 如果一台计算机只支持加法、减法操作, 那么能否计算三角函数, 对数函数? (提示: 搜索并阅读“泰勒级数展开”等内容)

支持, 利用泰勒级数展开, 可以将三角函数和对数函数转化为只有加减乘除的算术运算, 而乘除法能用加减法表示, 故只支持加减法的计算机能计算三角函数和对数函数。

$$\sin x = x - \frac{x^3}{6} + \dots = \sum_{n=0}^{\infty} (-1)^n \frac{x^{2n+1}}{(2n+1)!}, x \in (-1,1)^{\infty}$$

$$\cos x = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} + \dots = \sum_{n=0}^{\infty} (-1)^n \frac{x^{2n}}{(2n)!}, x \in (-1,1)^{\infty}$$

$$\tan x = x + \frac{x^3}{3} + \dots = \sum_{n=1}^{\infty} \frac{(2^{2n}-1)2^{2n}B_n}{(2n)!} x^{2n-1}, x \in (-1,1)^{\infty}$$

$$\ln(1+x) = x - \frac{x^2}{2} + \frac{x^3}{3} + \dots = \sum_{n=0}^{\infty} (-1)^n \frac{x^{n+1}}{n+1}, x \in (-1,1]$$

(4) 对于某个需要完成的功能，如果既可以通过硬件上增加电路来实现，也可以通过其他已有指令的组合来实现，那么如何判断哪一种比较合适？（提示：搜索并阅读 RISC 与 CISC）。

RISC 为精简指令集计算机，取使用频率较高的一些简单指令以及一些很有用但不复杂的指令，让复杂指令的功能由使用频率高的简单指令的组合来实现。

CISC 为杂指令集计算机，指令系统复杂庞大，指令数目一般多达 200~300 条，指令长度不固定，指令格式种类多，寻址方式种类多。

比较而言，RISC 比 CISC 更能提高计算机运算速度，便于设计，可降低成本，提高可靠性。但是 CISC 的指令系统比较丰富，有专用指令来完成特定的功能，因此处理特殊任务效率高。

对于不特殊的指令，可以选择用已有指令组合，特殊的指令，选择增加电路实现。