# 汇编实验二

完成实验后，需用实验报告纸撰写实验报告。

一、实验报告包含以下内容

- 实验序号
- 实验内容
- 算法描述
- 汇编程序
- 运行结果

二、实验目的

1. 掌握分支与循环结构的汇编表示。
2. 掌握使用子程序或函数，参数栈传递方式
3. 熟悉调试器 ollydbg 的使用。

三、课堂练习

1.试编程实现找出 k 个完美数,正整数 n 为完美数是指 n 等于其所有真因子的和。

如 6=1+2+3, 28=1+2+4+7+14

算法如下：

```
void printPerfNumbers(int k) {
    int count=0;
    int n=6;
    while (count < k) {
        if (isPerfNumber(n))   {
            print(n);
            count++;
        }
        n++;
    }
}
int isPerfNumber(int n) {
    int sum=1;
    int factor=2;
    while (factor < = n/2) {
        if ( n% factor ==0 ) {
             sum = sum + factor;
        }
        factor=factor+1;
    }
    if (sum==n)  return 1;
    return  0;
}
```

2.编程实现下列选择排序算法(参数以栈方式传递)

算法描述

```
void slectSort(int *arr,int n) {
    for(int i=n;i>1;i--) {
          j=maxIndex(arr, i);
    int  temp=arr[j]
    arr[j]=arr[i-1];
    arr[i-1]=temp;
    }
}
int maxIndex(int*arr, int n){
     int index;
     index=0;
     for(int i=0; i<n; i++) {
         if( arr[index] < arr[i]) {
                index=i
      }
   }
  return index;
}
```

四、实验内容

1.试编程实现正整数的素数分解。例如：　72=2^3*3^2

2.判断元素是否属于集合 M,其中 M 是这样生成：

(1) 已知 k 是集合 M 的元素；

(2) 如果 y 是 M 的元素，那么， 2y+1 和 3y+1 都是 M 的元素；

(3) 除了上述二种情况外，没有别的数能够成为 M 的一个元素。

试编程实现任意给定 k 和 x，请判断 x 是否是 M 的元素。

如果是，则输出 YES，否则，输出 NO

3. 试编程实现快速排序。

4.从键盘读取一串字符 S,该字符串包含一个简单算术表达式（含两个正整数（如：3434*45）的四则运算），试编程解析该字符串，并实现其语义。

提示：

1. 算法

```
int factorNumber(int *array,int n)
len=0;
if isPrime(n) then {
    save(n,1) to array;
    return len++;
}
p=2;
while (p<=n/2) {
    If  isPrime(p) then {
        if k= maxExp (n, p) > 0 then {
       save(p,k) to array;
       len++;
```

```
        }
          }
    p=p+1;
}
return len;
}
```

算法 isPrime(n)：

```
if  n == 1 then  return 0;
for (i=2; i*i<n; i++)
    if n%i ==0 then return 0;
}
return 1;
```

算法 maxExp (n, p)：

```
k=0;
while ( n%p==0) {
    k=k+1;
    n=n/p;
}
return k;
```

2. 算法 c 描述

```
int pd(int k,int x){
    if(k>x) return 0;
    else if(k==x) return 1;
        return (pd(2*k+1,x)||pd(3*k+1,x));
}
```

或者：

```
int pd(int k,int x){
    if(k>x) return 0;
    else if(k==x) return 1;
        if (pd(2*k+1,x)==1) return 1;
     return pd(3*k+1,x);
}
```

3. 算法 C 描述

```
void swap(int *a, int *b){
    int tmp = *a;
    *a = *b;
    *b = tmp;
}
```

```
int partition(int a[], int low, int high){
    int Key = a[low]; //基准元素
    while(low < high){ //从表的两端交替地向中间扫描
        while(low < high && a[high] >= Key) --high;
        swap(&a[low], &a[high]);
        while(low < high && a[low] <= Key ) ++low;
        swap(&a[low], &a[high]);
    }
    return low;
}

void quickSort(int a[], int low, int high){
    if(low < high){
        int Loc = partition(a, low, high); //将表一分为二
        quickSort(a, low, Loc -1); //递归对低子表递归排序
        quickSort(a, Loc + 1, high); //递归对高子表递归排序
    }
}
```