ICS HW7

T1

我们使用编译器将高级语言（如 C/C++）编译为低级语言（如汇编语言）。考虑如下 C 语言代码段：

```c
int x;
scanf("%d", &x);
int y = 2;
int z = y * 3 + 1;
for(int i = 0; i < z; i++) {
    y = i + 3;
    if(i < 0)
        x--;
    else
        x++;
    printf("%d\n", x);
}
printf("%d %d %d", x, y, z);
```

1. 将上述代码段编译为 LC-3 汇编语言。方便起见，你可以使用 INPUT [reg]将控制台输入的数字读取到寄存器[reg]中，使用 OUTPUT [reg]将寄存器[reg]中的数字输出到控制台。

2. 编译出的 LC-3 汇编代码中，是否所有语句都是必要的？优化你编译出的 LC-3 代码，在不改变语义的前提下尽可能提高程序运行效率。

3. 编译器可能采用哪些技术来优化编译结果？

T2

在通常的算术表达式（即中缀表达式）中我们时常使用括号来保证运算优先级正确，这对计算机解析表达式是不利的。使用后缀表达式可以有效避免括号的使用，从而提高便利性。

后缀表达式的规则是：从前到后读取字符，若读取到的是数字，则将数字压入栈中；若读取到的是运算符，则将栈顶的 等同于运算符操作数 个数字弹出并进行运算，然后将运算结果压入栈中。重复此过程直到读取完整个表达式，此时栈中仅剩的一个数字即为表达式的运算结果。

例如考虑后缀表达式：

$$3\ 1-4*$$

计算该后缀表达式的步骤为：

1. 将数字3压入栈中

2. 将数字1压入栈中

3. 读取到操作符$-$，弹出栈顶的1与3，计算$3-1=2$，将2压入栈中

4. 将数字4压入栈中

5. 读取到操作符$*$，弹出栈顶的4与2，计算$2*4=8$，将 8 压入栈中

6. 表达式全部读取完毕，8 即为表达式的结果

不难看出，该后缀表达式等价于中缀表达式$(3-1)*4$。

1. 计算下列后缀表达式的值

(1) $3\ 1\ 4\ 5+ \ll 6 - *$，其中≪为左移运算符

(2) $9\ 3\ 1\ -\ 3\ *\ +10\ 2\ /\ +$

(3) $2\ 2\ 1 + *!9 -$，其中!为阶乘运算符

2. 将下列中缀表达式转化为后缀表达式

(1) $(A + B) * (C - D)$

(2) $(A\ \&\&\ B)\ ||\ (C\ \&\&\ !\ D)$

(3) $(A\ +\ B\ *\ (C\ -\ D\ \hat{}\ E)\ /\ F)\ *\ G\ -\ H$

3. 假设后缀表达式中共有n个数字与m个运算符，其中第i个运算符的操作数为opt_i，请问在合法的后缀表达式中n, m, opt_i应有怎样的数量关系？

T3

1. 函数在被调用时需要运行时栈 run-time stack 来保存必要的信息。当一个函数被调用时，哪些信息是必须被保存的？是否所有寄存器的值都必须被保存？

2. 使用递归解决问题时，在问题规模过大的情况下时常出现 stack overflow 错误，这是为什么？将递归改为递推，手动使用栈存储数据则不会出现这种情况，这是为什么？

T4

考虑以下两段 C 语言代码:

(a)

```
int a[]={0, 1, 2, 3, 4, 5, 6, 7, 8, 9};
int *b = a;
b[5] = 42;
printf("%d %d", a[5], b[5]);
```

(b)

```
int a[]={0, 1, 2, 3, 4, 5, 6, 7, 8, 9};
int *b = malloc(10 * sizeof(int));
for (int i = 0; i < 10; i++) {
    b[i] = a[i];
}
b[5] = 42;
printf("%d %d", a[5], b[5]);
```

1. 它们分别具有怎样的输出?

2. 这两段代码展示了两种数组拷贝的方式，它们分别具有怎样的优缺点?

3. 在代码(a)中，b = a 后，b 是否可以指向其他数组？a 是否可以被重新赋值（如 a=b）？

T5

众所周知，内存的读写速度远慢于寄存器，因此访问内存的指令将带来大量的时间消耗。高速缓存技术通过在寄存器与内存之间增加一个名为高速缓存的存储器来有效缓解由访问内存带来的性能问题。

假设在 LC-3 内存空间中存储着二维数组$A[4][4]$，数组的每个元素为 16 位整

型，数组的起始位置为内存 x3000 处。假设访问高速缓存需要 1 单位时间，访问内存需要 20 单位时间，忽略除读取数组外的时间消耗。

1. 不存在高速缓存的情况下，采用如下算法计算矩阵乘法$A*A$需要多少单位时间？（忽略寄存器的暂存功能）

```
for(int i = 0; i < 4; i++){
    for(int j = 0; j < 4; j++){
        for(int k = 0; k < 4; k++){
            calc A[i][k]*A[k][j]
        }
    }
}
```

2. 假设存在字长与内存相同的 4 位地址空间的高速缓存，每个内存块按地址后 4 位映射到对应位置的高速缓存块。访问内存时，首先访问对应位置的高速缓存块以查看其中是否为所需要的内存数据，若是则直接读取该数据，若不是则从内存中读取数据，然后用读取的数据填充对应的高速缓存位置。假设开始计算时高速缓存为空，在此情境下以同样算法计算矩阵乘法$A*A$需要多少单位时间？（忽略寄存器的暂存功能）

T6

```
#include<stdio.h>
struct Test{
    char c[3];
    int i;
}t;
int main(){
    scanf("%s",t.c);
    printf("%d\n",t.i);
    return 0;
}
```

考虑如上 C 语言程序，输入字符串"**abcdef**"时，在你的电脑上程序具有怎样的输出？由此推测结构体与整型数据是以何种方式在你电脑的内存中存储的。