**实验零**

（1）汇编代码与c代码之间的关系：

1、它们都是让人与机器交流的语言

2、C语言更接近自然语言，代码比较容易阅读

3、汇编更接近机器，掌握它可以从更深层次去理解机器

4、它们经过编译程序处理后得到的都是二进制指令数据，对机器来说无区别。

（2）并写出设置断点、单步执行及查看变量的过程：

**设置断点**

break 命令设置断点，简写b ;

break main ，在main()函数的入口处设置断点；

break 5，在源代码的第5行设置断点；

break hello.c:5 ，指定源码文件的代码第5行设置断点；

**单步调试**

next，n，越过 函数调用（函数会在背地里自己悄悄运行完），单步执行；

step，s，进入 函数体内部，单步执行；

**查看变量**

disp，使得每次有暂停，都会输出指定的变量的值；

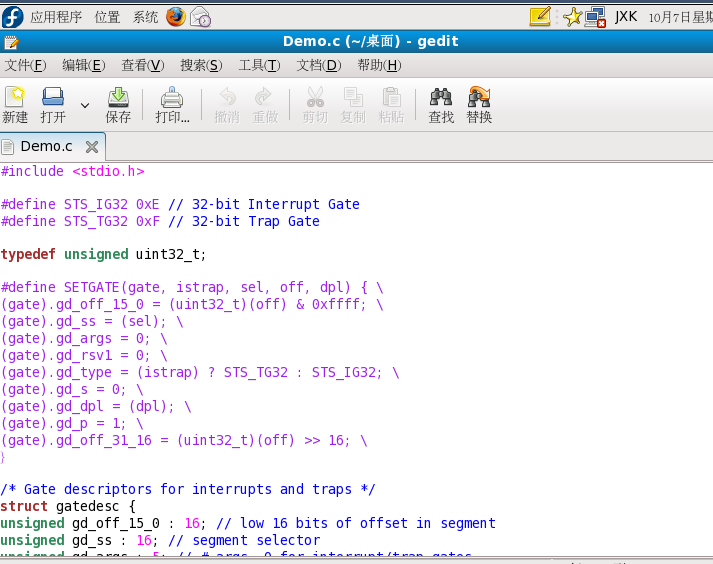
print，p，只显示一次变量的值；

1. （3）掌握指针和类型转换相关的Ｃ编程

将代码粘贴，运行执行后结果为：

Intr is 0x10002

Intr is 0xee0000010002





（4）掌握通用链表结构相关的Ｃ编程

查看list.h和lab0\_ex4.c，编写一个程序，利用list.h中的链表结构，将26个英文字母存入链表中，并逆序打印出来

#include<stdio.h>  
#include<stdlib.h>   
  
struct list\_entry

 {  
    struct list\_entry \*prev, \*next;  
};  
  
typedef struct list\_entry list\_entry\_t;  
  
struct entry

 {  
    list\_entry\_t node;  
    int num;  
};  
  
void list\_init(list\_entry\_t \*elm)

{  
    elm->prev = elm->next = elm;  
}  
  
bool list\_empty(list\_entry\_t \*list)

{  
    return list->next == list;  
}  
  
void \_\_list\_add(list\_entry\_t \*elm, list\_entry\_t \*prev, list\_entry\_t \*next)

{  
    prev->next = next->prev = elm;  
    elm->next = next;  
    elm->prev = prev;  
}  
  
void list\_add\_after(list\_entry\_t \*listelm, list\_entry\_t \*elm)

 {  
    \_\_list\_add(elm, listelm, listelm->next);  
}  
  
void list\_add(list\_entry\_t \*listelm, list\_entry\_t \*elm)

{  
    list\_add\_after(listelm, elm);  
}  
  
void list\_add\_before(list\_entry\_t \*listelm, list\_entry\_t \*elm)

{  
    \_\_list\_add(elm, listelm->prev, listelm);  
}  
  
list\_entry\_t \* list\_next(list\_entry\_t \*listelm) {  
    return listelm->next;  
}  
  
int main()

{  
    struct entry head;  
    list\_entry\_t\* p = &head.node;  
    list\_init(p);  
    head.num = 0 ;  
  
for (int i = 26; i != 0; i --) {  
        struct entry \* e = (struct entry \*)malloc(sizeof(struct entry));  
        e->num = i;  
        list\_add(p, &(e->node));  
        p = list\_next(p);  
          
        printf("%c",i+&apos;A&apos;-1);  
    }  
     
return 0;  
}