操作系统lab0

班级：硬件一班

姓名：于华宇

学号：171491120

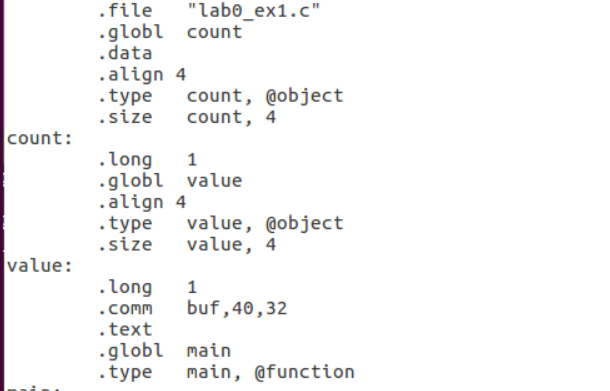
1. 实验目的：
2. 了解操作系统开发实验环境
3. 熟悉命令行方式的编译、调试工程
4. 掌握基于硬件模拟器的调试技术
5. 熟悉C语言编程和指针的概念
6. 了解X86汇编语言
7. 实验内容：
   1. 了解汇编

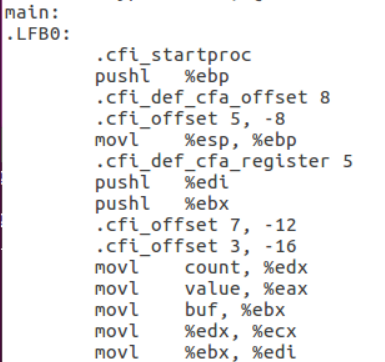
尝试理解下面的命令$gcc -S -m32 lab0\_ex1.c 接着我们将得到lab0\_ex1.s文件，请写出汇编代码与c代码之间的关系。

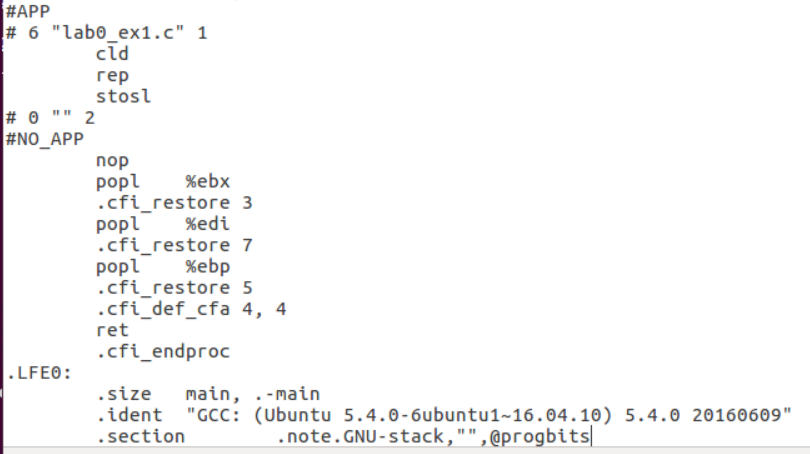
答：

执行命令后，生成.s文件（汇编代码文件）

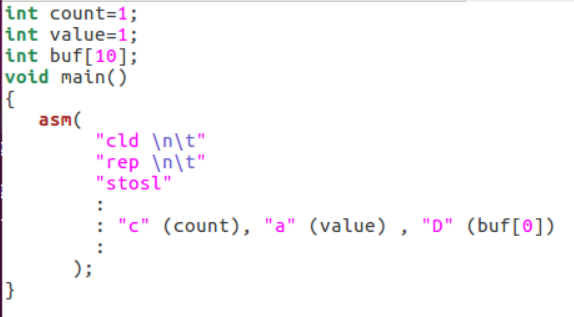
内容是：







c代码为：



根据学习，asm（）是C语言的gcc内联汇编函数。asm（）中的语句会转换成汇编语言。cld rep stosl这些命令的功能是向buf中写上count个value值。每行语句后面都加上\n\t是为了转变成汇编语言的格式的美观（换行缩进）。“c”（count），“a”（value），“D”（buf【0】）这个命令是向寄存机ecx存入count值，向eax中存入value值，向edi中存入buf【0】的值。最后生成的汇编代码文件中，内容与C语言内联汇编函数中内容一致。

* 1. 用gdb调试

尝试下面的命令，$gcc -g -m32 lab0\_ex2.c 接着我们会得到a.out文件，请用gdb调试，并写出设置断点、单步执行及查看变量的过程。

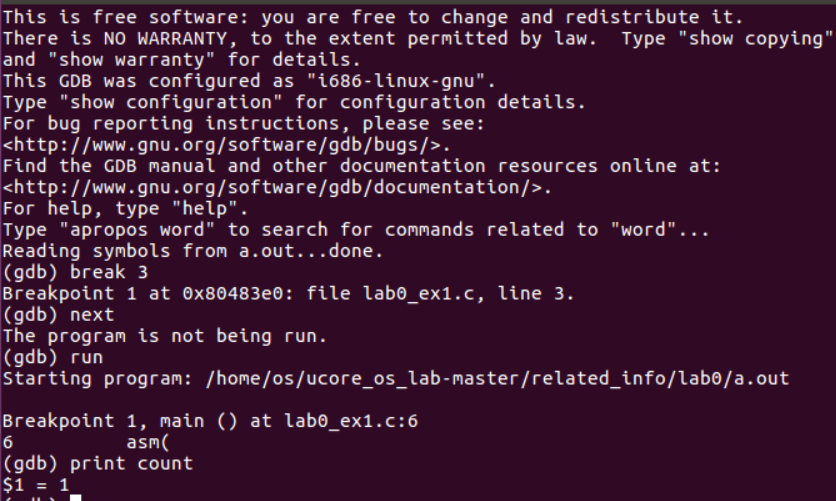
答：

调试：gdb a.out

设置断点：(gdb) break 3

单步执行：(gdb) next

用 print 命令查看 string 的值：(gdb) run (gdb) print string (gdb) $1=1



* 1. 掌握指针和类型转换相关的Ｃ编程

分析如下代码段，

#include<stdio.h>

#define STS\_IG32 0xE // 32-bit Interrupt Gate

#define STS\_TG32 0xF // 32-bit Trap Gate

typedef unsigned uint32\_t;

#define SETGATE(gate, istrap, sel, off, dpl) { \

(gate).gd\_off\_15\_0 = (uint32\_t)(off) & 0xffff; \

(gate).gd\_ss = (sel); \

(gate).gd\_args = 0; \

(gate).gd\_rsv1 = 0; \

(gate).gd\_type = (istrap) ? STS\_TG32 : STS\_IG32; \

(gate).gd\_s = 0; \

(gate).gd\_dpl = (dpl); \

(gate).gd\_p = 1; \

(gate).gd\_off\_31\_16 = (uint32\_t)(off) >> 16; \

}

/\* Gate descriptors for interrupts and traps \*/

struct gatedesc {

unsigned gd\_off\_15\_0 : 16; // low 16 bits of offset in segment

unsigned gd\_ss : 16; // segment selector

unsigned gd\_args : 5; // # args, 0 for interrupt/trap gates

unsigned gd\_rsv1 : 3; // reserved(should be zero I guess)

unsigned gd\_type : 4; // type(STS\_{TG,IG32,TG32})

unsigned gd\_s : 1; // must be 0 (system)

unsigned gd\_dpl : 2; // descriptor(meaning new) privilege level

unsigned gd\_p : 1; // Present

unsigned gd\_off\_31\_16 : 16; // high bits of offset in segment

};

int main(void)

{

unsigned before;

unsigned intr;

unsigned after;

struct gatedesc gintr;

intr=8;

before=after=0;

gintr=\*((struct gatedesc \*)&intr);

SETGATE(gintr, 0,1,2,3);

intr=\*(unsigned \*)&(gintr);

printf("intr is 0x%x\n",intr);

printf("gintr is 0x%llx\n", gintr);

return 0;

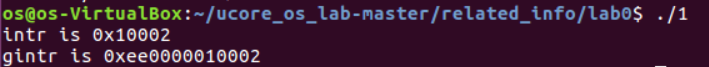
}

写出gintr和intr的结果，试着编译这段代码，如果遇到错误进行改正，并分析错误原因。

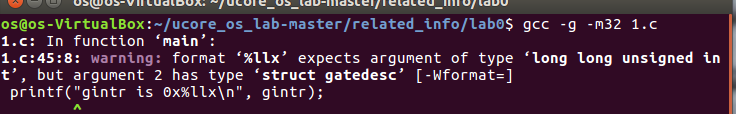
答：

在代码中，gintr等于被强制转换成struct gatedesc类型的指针，指向intr的地址，intr等于被强制转换成unsigned int类型的指针，指向gintr。所以intr与gintr的值应该相同。但是intr是32位，gintr是64位。

结果是：



编译时遇到的问题：



gintr是64位的结构体，%llx是64位整型（16进制），所以打印时将gintr的类型强制转换成long long usigned就会正确。但是结构体类型不能直接强制转换成数量类型，应该转换成指针类型再找到其指向的内容。

即，将printf（“gintr is 0x%llx\n”,gintr）;

改为printf（“gintr is 0x%llx\n”,（\*（long long unsigned \*）&gintr））;

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>

#include "list.h"

#include"defs.h"

struct entry {

list\_entry\_t node;

char letter;

};

int main() {

struct entry head;

list\_entry\_t\* p = &head.node;

list\_init(p);

head.letter = 'a';

int i;

for (i = 0; i <= 26; i ++) {

struct entry \* e = (struct entry \*)malloc(sizeof(struct entry));

e->letter = 'a'+i;

list\_add(p, &(e->node));

p = list\_next(p);

}

//reverse list all node

while ((p = list\_prev(p)) != &head.node){

printf("%c\t", ((struct entry \*)p)->letter);

}

return 0;

}

