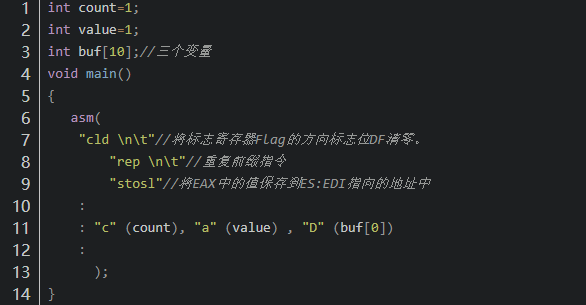
##### 1、了解汇编

运行 gcc -S -m32 lab0\_ex1.c ，生成S汇编语言文件。

-S表示仅仅编译，不进行链接或汇编

-m32表示生成32位机器的汇编代码

得到lab\_ex1.S文件，下面对比理解C文件和S文件：

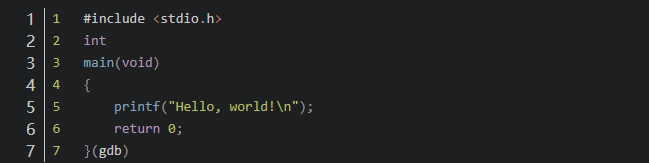


##### ２、用gdb调试

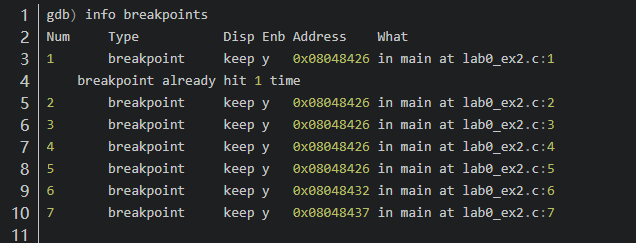
gcc -g -m32 lab0\_ex2.c // -g表示调试

gdb a.out

先用 gdb l 查看代码，发现是helloworld：



给每一行都加入断点，输入 info breakpoints 查看，有虽然每行都有断点，但从地址上看实际上只有三处有断点：



按 r （run）从头运行，停止在第一个断点。这里：

* 按 c （continue）继续运行，直到遇见下一个断点
* 按 s （step）表示执行下一语句，会进入函数内部，类似VS的 F11
* 按 n （next）表示执行下一语句，但不会进入函数内部，类似VS的 F10

按下 s 后屏幕输出 hello,world! 停止在了断点6，再按 s 停止在断点7反花括号那里，最后一步程序结束运行。按 q退出。

##### 3、掌握指针和类型转换相关的Ｃ编程

实验给出了一个C文件，让debug:

#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);

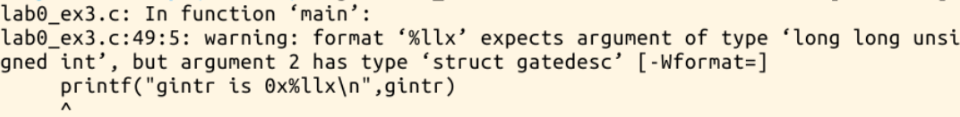
printf("intr is 0x%x\n",intr);

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

return 0;

}

首先编译报错为：



题目的意思应该是输出更改后的gintr这个结构体变量的内容 ，参照上面处理方法我把它强制类型转换成llu型： \* ( long long unsigned \*) & gintr 就OK了。

解释：首先是取gintr这个变量的地址，强制转换为指向llu型变量的指针，再引用这个地址就得到了llu型的变量。

输出结果为：



这个程序目的应该是用setgate函数修改结构体内容，不知道我这样改对不对。。。

另外注意到 gintr 调试输出的高位为0x00020001 ，高位放在了低地址，说明该机器是大端模式。

##### 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。

查看lab0\_ex４.c代码：

