北京郵電大學

实验报告

题目： **Linux 环境和 GCC 工具链**

班　　级： **2022211320**

学　　号： **2022211683**

姓　　名： **张晨阳**

学　　院： **计算机学院（国家示范性软件学院）**

2023 年 10 月 16 日

一、实验目的

1. 熟悉 linux 系统的常用命令；
2. 掌握 gcc 编译器的使用方法；
3. 掌握 gdb 的调试工具使用；
4. 掌握 objdump 反汇编工具使用；
5. 理解反汇编程序（对照源程序与 objdump 生成的汇编程序）。

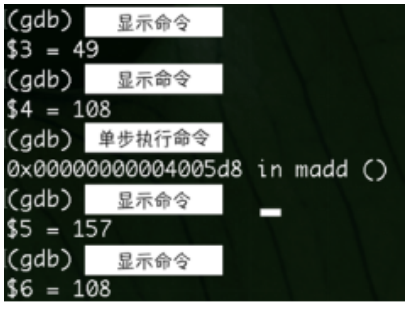
二、实验环境

1. Visual Studio Code 1.83.0
2. Compiler Explorer
3. Microsoft Edge 117.0.2045.60
4. Obsidian 1.4.16

三、实验内容

现有两个 int 型数组 $a[i]=i-50$，$b[i]=i+y$，其中 y 取自于学生本人学号 $2022211x*y$ 的个位。登录 bupt1服务器，在 linux 环境下使用 vi 编辑器编写 C 语言源程序，完成数组 a+b 的功能，规定数组长度为 100，函数名为 madd（），数组 a，b 均定义在函数内，采用 gcc 编译该程序（使用-g -no-pie -fno-pie -fno-stack-protector 选项），

1. 使用 objdump 工具生成汇编程序，找到 madd 函数的汇编程序，给出截图；
2. 用 gdb 进行调试，练习下列 gdb 命令，给出截图；
   gdb、file、kill、quit、break、delete、clear、info break、run、continue、nexti、stepi、disassemble、list、print、x、info reg、watch
3. 找到 $a[i]+b[i]$ 对应的汇编指令，指出 $a[i]$ 和 $b[i]$ 位于哪个寄存器中，给出截图；
4. 使用单步指令及 gdb 相关命令，显示 $a[xy]+b[xy]$ 对应的汇编指令执行前后操作数寄存器十进制和十六进制的值，其中 x，y 取自于学生本人学号 $2022211x*y$ 的百位和个位。
   学号2022211999，$a[99]+b[99]$ 单步执行前后的参考截图如下（实际命令未显示出）：



四、实验步骤及实验分析

1. **首先启动 powershell 并登录 bupt1 服务器。**
2. **实验准备需要使用 vi 编辑器编写 madd 函数程序并编译：**
   在学习了 PPT 上 vi 部分的内容后，使用命令 : e 新建了一个 Lab1.c 文件并写入了程序，最后使用命令:wq! 退出 vi 并保存文件。需要注意的是，当文件没有被写入时，是无法打开的。之后用 gcc 指令编译运行了 Lab1. c；但发现文件写入了两遍程序，并出现如图 0 展示的错误：

```
Lab1.c:18:6: error: redefinition of ‘madd’
 void madd()
      ^~~~
Lab1.c:2:6: note: previous definition of ‘madd’ was here
 void madd()
      ^~~~
Lab1.c:28:5: error: redefinition of ‘main’
 int main()
     ^~~~
Lab1.c:12:5: note: previous definition of ‘main’ was here
 int main()
     ^~~~
```

图0

检查发现是无意间使用了两次写入命令，使用了 dd 命令修改文件并重新编译运行成功。

3. **实验内容第一项使用 objdump 工具生成汇编程序，找到 madd 函数的汇编程序，给出截图：**

使用命令 objdump -d -S Lab1 > Lab1.s 生成汇编程序，命名为 Lab1.s（注意不能使用-o，因为-o 不属于 objdump 的命令），通过 vi 程序进入 Lab1.s 文件，找到 madd 函数对应的汇编代码部分，截图如下：

```
void madd()
{
  401107:       55                      push   %rbp
  401108:       48 89 e5                mov    %rsp,%rbp
  40110b:       48 81 ec b8 02 00 00    sub    $0x2b8,%rsp
        int a[100], b[100];
        for(int i=0;i<100; i++){
  401112:       c7 45 fc 00 00 00 00    movl   $0x0,-0x4(%rbp)
  401119:       eb 28                   jmp    401143 <madd+0x3c>
                a[i] = i - 50;
  40111b:       8b 45 fc                mov    -0x4(%rbp),%eax
  40111e:       8d 50 ce                lea    -0x32(%rax),%edx
  401121:       8b 45 fc                mov    -0x4(%rbp),%eax
  401124:       48 98                   cltq
  401126:       89 94 85 60 fe ff ff    mov    %edx,-0x1a0(%rbp,%rax,4)
                b[i] = i + 3;
  40112d:       8b 45 fc                mov    -0x4(%rbp),%eax
  401130:       8d 50 03                lea    0x3(%rax),%edx
  401133:       8b 45 fc                mov    -0x4(%rbp),%eax
  401136:       48 98                   cltq
  401138:       89 94 85 d0 fc ff ff    mov    %edx,-0x330(%rbp,%rax,4)
        for(int i=0;i<100; i++){
  40113f:       83 45 fc 01             addl   $0x1,-0x4(%rbp)
  401143:       83 7d fc 63             cmpl   $0x63,-0x4(%rbp)
  401147:       7e d2                   jle    40111b <madd+0x14>
        }
        for(int i=0; i<100; i++)
  401149:       c7 45 f8 00 00 00 00    movl   $0x0,-0x8(%rbp)
  401150:       eb 2a                   jmp    40117c <madd+0x75>
                a[i] += b[i];
  401152:       8b 45 f8                mov    -0x8(%rbp),%eax
  401155:       48 98                   cltq
  401157:       8b 94 85 60 fe ff ff    mov    -0x1a0(%rbp,%rax,4),%edx
  40115e:       8b 45 f8                mov    -0x8(%rbp),%eax
  401161:       48 98                   cltq
  401163:       8b 84 85 d0 fc ff ff    mov    -0x330(%rbp,%rax,4),%eax
  40116a:       01 c2                   add    %eax,%edx
  40116c:       8b 45 f8                mov    -0x8(%rbp),%eax
  40116f:       48 98                   cltq
  401171:       89 94 85 60 fe ff ff    mov    %edx,-0x1a0(%rbp,%rax,4)
        for(int i=0; i<100; i++)
  401178:       83 45 f8 01             addl   $0x1,-0x8(%rbp)
  40117c:       83 7d f8 63             cmpl   $0x63,-0x8(%rbp)
  401180:       7e d0                   jle    401152 <madd+0x4b>
}
  401182:       90                      nop
  401183:       c9                      leaveq
  401184:       c3                      retq
```

图1-madd

4. **实验内容第二项用 gdb 进行调试，练习 gdb 命令，给出截图：**

```
2022211683@bupt1:~$ gdb
GNU gdb (Ubuntu 9.2-0ubuntu1~20.04.1) 9.2
Copyright (C) 2020 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
Type "show copying" and "show warranty" for details.
This GDB was configured as "x86_64-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
    <http://www.gnu.org/software/gdb/documentation/>.

For help, type "help".
Type "apropos word" to search for commands related to "word".
```

图2-gdb

```
(gdb) file Lab1
Reading symbols from Lab1...
(gdb) list
1
2       #include <stdio.h>
3       void madd()
4       {
5               int a[100], b[100];
6               for(int i=0;i<100; i++){
7                       a[i] = i - 50;
8                       b[i] = i + 3;
9               }
10              for(int i=0; i<100; i++)
(gdb) l
11                      a[i] += b[i];
12      }
13      int main()
14      {
15              madd();
16              return 0;
17      }
(gdb) l
Line number 18 out of range; Lab1.c has 17 lines.
```

图3-file, list

```
(gdb) break 11
Breakpoint 1 at 0x401152: file Lab1.c, line 11.
(gdb) run
Starting program: /students/2022211683/Lab1

Breakpoint 1, madd () at Lab1.c:11
11                      a[i] += b[i];
(gdb) watch i
Hardware watchpoint 2: i
(gdb) watch a[i]
Hardware watchpoint 3: a[i]
(gdb) continue
Continuing.

Hardware watchpoint 3: a[i]

Old value = -50
New value = -47
madd () at Lab1.c:10
10              for(int i=0; i<100; i++)
(gdb) continue
Continuing.

Hardware watchpoint 2: i

Old value = 0
New value = 1

Hardware watchpoint 3: a[i]

Old value = -47
New value = -49
0x000000000040117c in madd () at Lab1.c:10
10              for(int i=0; i<100; i++)
(gdb) continue
Continuing.

Breakpoint 1, madd () at Lab1.c:11
11                      a[i] += b[i];
```

图4-break, run, continue, watch

```
(gdb) continue
Continuing.

Breakpoint 1, madd () at Lab1.c:11
11                      a[i] += b[i];
(gdb) continue
Continuing.

Hardware watchpoint 3: a[i]

Old value = -49
New value = -45
madd () at Lab1.c:10
10              for(int i=0; i<100; i++)
(gdb) continue
Continuing.

Hardware watchpoint 2: i

Old value = 1
New value = 2

Hardware watchpoint 3: a[i]

Old value = -45
New value = -48
0x000000000040117c in madd () at Lab1.c:10
10              for(int i=0; i<100; i++)
(gdb) continue
Continuing.

Breakpoint 1, madd () at Lab1.c:11
11                      a[i] += b[i];
(gdb) continue
Continuing.

Hardware watchpoint 3: a[i]

Old value = -48
New value = -43
madd () at Lab1.c:10
10              for(int i=0; i<100; i++)
```

图5-break, run, continue, watch

```
(gdb) info break
Num     Type           Disp Enb Address            What
1       breakpoint     keep y   0x0000000000401152 in madd at Lab1.c:11
        breakpoint already hit 3 times
2       hw watchpoint  keep y                      i
        breakpoint already hit 2 times
3       hw watchpoint  keep y                      a[i]
        breakpoint already hit 5 times
(gdb) disassemble $pc
Dump of assembler code for function madd:
   0x0000000000401107 <+0>:     push   %rbp
   0x0000000000401108 <+1>:     mov    %rsp,%rbp
   0x000000000040110b <+4>:     sub    $0x2b8,%rsp
   0x0000000000401112 <+11>:    movl   $0x0,-0x4(%rbp)
   0x0000000000401119 <+18>:    jmp    0x401143 <madd+60>
   0x000000000040111b <+20>:    mov    -0x4(%rbp),%eax
   0x000000000040111e <+23>:    lea    -0x32(%rax),%edx
   0x0000000000401121 <+26>:    mov    -0x4(%rbp),%eax
   0x0000000000401124 <+29>:    cltq
   0x0000000000401126 <+31>:    mov    %edx,-0x1a0(%rbp,%rax,4)
   0x000000000040112d <+38>:    mov    -0x4(%rbp),%eax
   0x0000000000401130 <+41>:    lea    0x3(%rax),%edx
   0x0000000000401133 <+44>:    mov    -0x4(%rbp),%eax
   0x0000000000401136 <+47>:    cltq
   0x0000000000401138 <+49>:    mov    %edx,-0x330(%rbp,%rax,4)
   0x000000000040113f <+56>:    addl   $0x1,-0x4(%rbp)
   0x0000000000401143 <+60>:    cmpl   $0x63,-0x4(%rbp)
   0x0000000000401147 <+64>:    jle    0x40111b <madd+20>
   0x0000000000401149 <+66>:    movl   $0x0,-0x8(%rbp)
   0x0000000000401150 <+73>:    jmp    0x40117c <madd+117>
   0x0000000000401152 <+75>:    mov    -0x8(%rbp),%eax
   0x0000000000401155 <+78>:    cltq
   0x0000000000401157 <+80>:    mov    -0x1a0(%rbp,%rax,4),%edx
   0x000000000040115e <+87>:    mov    -0x8(%rbp),%eax
   0x0000000000401161 <+90>:    cltq
   0x0000000000401163 <+92>:    mov    -0x330(%rbp,%rax,4),%eax
   0x000000000040116a <+99>:    add    %eax,%edx
   0x000000000040116c <+101>:   mov    -0x8(%rbp),%eax
--Type <RET> for more, q to quit, c to continue without paging--q
Quit
```

图6-info break, disassemble

```
(gdb) break 7
Breakpoint 4 at 0x40111b: file Lab1.c, line 7.
(gdb) info break
Num     Type           Disp Enb Address            What
1       breakpoint     keep y   0x0000000000401152 in madd at Lab1.c:11
        breakpoint already hit 3 times
2       hw watchpoint  keep y                      i
        breakpoint already hit 2 times
3       hw watchpoint  keep y                      a[i]
        breakpoint already hit 5 times
4       breakpoint     keep y   0x000000000040111b in madd at Lab1.c:7
(gdb) delete 4
(gdb) info break
Num     Type           Disp Enb Address            What
1       breakpoint     keep y   0x0000000000401152 in madd at Lab1.c:11
        breakpoint already hit 3 times
2       hw watchpoint  keep y                      i
        breakpoint already hit 2 times
3       hw watchpoint  keep y                      a[i]
        breakpoint already hit 5 times
(gdb) nexti

Hardware watchpoint 2: i

Old value = 2
New value = 3

Hardware watchpoint 3: a[i]

Old value = -43
New value = -47
0x000000000040117c in madd () at Lab1.c:10
10              for(int i=0; i<100; i++)
(gdb) nexti
0x0000000000401180      10              for(int i=0; i<100; i++)
(gdb) nexti

Breakpoint 1, madd () at Lab1.c:11
11                      a[i] += b[i];
```

图7-delete, nexti

```
(gdb) stepi
0x0000000000401155      11                      a[i] += b[i];
(gdb) stepi
0x0000000000401157      11                      a[i] += b[i];
(gdb) continue
Continuing.

Hardware watchpoint 3: a[i]

Old value = -47
New value = -41
madd () at Lab1.c:10
10              for(int i=0; i<100; i++)
(gdb) stepi

Hardware watchpoint 2: i

Old value = 3
New value = 4

Hardware watchpoint 3: a[i]

Old value = -41
New value = -46
0x000000000040117c in madd () at Lab1.c:10
10              for(int i=0; i<100; i++)
(gdb) stepi
0x0000000000401180      10              for(int i=0; i<100; i++)
(gdb) stepi

Breakpoint 1, madd () at Lab1.c:11
11                      a[i] += b[i];
(gdb) stepi
0x0000000000401155      11                      a[i] += b[i];
(gdb) stepi
0x0000000000401157      11                      a[i] += b[i];
(gdb) stepi
0x000000000040115e      11                      a[i] += b[i];
```

图8-stepi

```
(gdb) continue
Continuing.

Hardware watchpoint 2: i

Old value = 4
New value = 5

Hardware watchpoint 3: a[i]

Old value = -39
New value = -45
0x000000000040117c in madd () at Lab1.c:10
10              for(int i=0; i<100; i++)
(gdb) print i
$1 = 5
(gdb) print a[i]
$2 = -45
(gdb) print b[i]
$3 = 8
(gdb) info reg
rax            0x4                 4
rbx            0x4011a0            4198816
rcx            0x4011a0            4198816
rdx            0xffffffd9          4294967257
rsi            0x7fffffffebe8      140737488350184
rdi            0x1                 1
rbp            0x7fffffffeae0      0x7fffffffeae0
rsp            0x7fffffffe828      0x7fffffffe828
r8             0x0                 0
r9             0x7ffff7fe0d60      140737354009952
r10            0xf                 15
r11            0x2                 2
r12            0x401020            4198432
r13            0x7fffffffebe0      140737488350176
r14            0x0                 0
r15            0x0                 0
rip            0x40117c            0x40117c <madd+117>
eflags         0x206               [ PF IF ]
cs             0x33                51
ss             0x2b                43
ds             0x0                 0
es             0x0                 0
fs             0x0                 0
gs             0x0                 0
```

图9-print, info reg

```
(gdb) x 0x1
0x1:    Cannot access memory at address 0x1
(gdb) x 0x7fffffffe828
0x7fffffffe828: 0x00000021
(gdb) x
0x7fffffffe82c: 0x00000022
(gdb) x/3uh
0x7fffffffe830: 35      0       36
(gdb) x/4uh
0x7fffffffe836: 0       37      0       38
(gdb) x/8uh
0x7fffffffe83e: 0       39      0       40      0       41      0       42
(gdb) x/3dh
0x7fffffffe84e: 0       43      0
(gdb) continue
Continuing.

Hardware watchpoint 3: a[i]

Old value = -45
New value = -37
madd () at Lab1.c:10
10              for(int i=0; i<100; i++)
```

图10-x

```
(gdb) break 7
Breakpoint 4 at 0x40111b: file Lab1.c, line 7.
(gdb) info break
Num     Type           Disp Enb Address            What
1       breakpoint     keep y   0x0000000000401152 in madd at Lab1.c:11
        breakpoint already hit 6 times
2       hw watchpoint  keep y                      i
        breakpoint already hit 5 times
3       hw watchpoint  keep y                      a[i]
        breakpoint already hit 11 times
4       breakpoint     keep y   0x000000000040111b in madd at Lab1.c:7
(gdb) clear 4
No breakpoint at 4.
(gdb) clear 7

(gdb) info break
Deleted breakpoint 4 Num     Type           Disp Enb Address            What
1       breakpoint     keep y   0x0000000000401152 in madd at Lab1.c:11
        breakpoint already hit 6 times
2       hw watchpoint  keep y                      i
        breakpoint already hit 5 times
3       hw watchpoint  keep y                      a[i]
        breakpoint already hit 11 times
(gdb) kill
Kill the program being debugged? (y or n) y
[Inferior 1 (process 1790386) killed]
(gdb) continue
The program is not being run.
(gdb) quit
2022211683@bupt1:~$
```

图11-clear, kill,quit

5. **实验内容第三项找到 $a[i]+b[i]$ 汇编指令，指出 $a[i]$ 和 $b[i]$ 位于哪个寄存器中，给出截图：**

分析思路 1：在 $a[i]+=b[i]$ 代码行设置断点，然后执行至程序停止在断点处。此时断点语句（$a[i]+=b[i]$）还未执行，使用命令 disassemble 查看此时的汇编指令并记录停止位置，然后使用命令 nexti 逐步执行至该语句完成，再次使用命令 disassemble 查看汇编指令并记录停止位置。两次停止位置所夹区间即为 $a[i]+b[i]$ 汇编指令，具体操作过程如下：

```
(gdb) break 11
Breakpoint 1 at 0x401152: file Lab1.c, line 11.
(gdb) run
Starting program: /students/2022211683/Lab1

Breakpoint 1, madd () at Lab1.c:11
11                  a[i] += b[i];
(gdb) disassemble
Dump of assembler code for function madd:
   0x0000000000401107 <+0>:     push   %rbp
   0x0000000000401108 <+1>:     mov    %rsp,%rbp
   0x000000000040110b <+4>:     sub    $0x2b8,%rsp
   0x0000000000401112 <+11>:    movl   $0x0,-0x4(%rbp)
   0x0000000000401119 <+18>:    jmp    0x401143 <madd+60>
   0x000000000040111b <+20>:    mov    -0x4(%rbp),%eax
   0x000000000040111e <+23>:    lea    -0x32(%rax),%edx
   0x0000000000401121 <+26>:    mov    -0x4(%rbp),%eax
   0x0000000000401124 <+29>:    cltq
   0x0000000000401126 <+31>:    mov    %edx,-0x1a0(%rbp,%rax,4)
   0x000000000040112d <+38>:    mov    -0x4(%rbp),%eax
   0x0000000000401130 <+41>:    lea    0x3(%rax),%edx
   0x0000000000401133 <+44>:    mov    -0x4(%rbp),%eax
   0x0000000000401136 <+47>:    cltq
   0x0000000000401138 <+49>:    mov    %edx,-0x330(%rbp,%rax,4)
   0x000000000040113f <+56>:    addl   $0x1,-0x4(%rbp)
   0x0000000000401143 <+60>:    cmpl   $0x63,-0x4(%rbp)
   0x0000000000401147 <+64>:    jle    0x40111b <madd+20>
   0x0000000000401149 <+66>:    movl   $0x0,-0x8(%rbp)
   0x0000000000401150 <+73>:    jmp    0x40117c <madd+117>
=> 0x0000000000401152 <+75>:    mov    -0x8(%rbp),%eax
   0x0000000000401155 <+78>:    cltq
   0x0000000000401157 <+80>:    mov    -0x1a0(%rbp,%rax,4),%edx
   0x000000000040115e <+87>:    mov    -0x8(%rbp),%eax
   0x0000000000401161 <+90>:    cltq
   0x0000000000401163 <+92>:    mov    -0x330(%rbp,%rax,4),%eax
   0x000000000040116a <+99>:    add    %eax,%edx
   0x000000000040116c <+101>:   mov    -0x8(%rbp),%eax
--Type <RET> for more, q to quit, c to continue without paging--q
Quit
(gdb) nexti
0x0000000000401155      11                  a[i] += b[i];
(gdb) ni
0x0000000000401157      11                  a[i] += b[i];
(gdb) ni
0x000000000040115e      11                  a[i] += b[i];
(gdb) ni
0x0000000000401161      11                  a[i] += b[i];
(gdb) ni
0x0000000000401163      11                  a[i] += b[i];
(gdb) ni
0x000000000040116a      11                  a[i] += b[i];
(gdb) ni
0x000000000040116c      11                  a[i] += b[i];
(gdb) ni
0x000000000040116f      11                  a[i] += b[i];
(gdb) ni
0x0000000000401171      11                  a[i] += b[i];
(gdb) ni
10              for(int i=0; i<100; i++)
```

图12

```
(gdb) nexti
0x0000000000401155      11                  a[i] += b[i];
(gdb) ni
0x0000000000401157      11                  a[i] += b[i];
(gdb) ni
0x000000000040115e      11                  a[i] += b[i];
(gdb) ni
0x0000000000401161      11                  a[i] += b[i];
(gdb) ni
0x0000000000401163      11                  a[i] += b[i];
(gdb) ni
0x000000000040116a      11                  a[i] += b[i];
(gdb) ni
0x000000000040116c      11                  a[i] += b[i];
(gdb) ni
0x000000000040116f      11                  a[i] += b[i];
(gdb) ni
0x0000000000401171      11                  a[i] += b[i];
(gdb) ni
10              for(int i=0; i<100; i++)
(gdb) disassemble
Dump of assembler code for function madd:
   0x0000000000401107 <+0>:     push   %rbp
   0x0000000000401108 <+1>:     mov    %rsp,%rbp
   0x000000000040110b <+4>:     sub    $0x2b8,%rsp
   0x0000000000401112 <+11>:    movl   $0x0,-0x4(%rbp)
   0x0000000000401119 <+18>:    jmp    0x401143 <madd+60>
   0x000000000040111b <+20>:    mov    -0x4(%rbp),%eax
   0x000000000040111e <+23>:    lea    -0x32(%rax),%edx
   0x0000000000401121 <+26>:    mov    -0x4(%rbp),%eax
   0x0000000000401124 <+29>:    cltq
   0x0000000000401126 <+31>:    mov    %edx,-0x1a0(%rbp,%rax,4)
   0x000000000040112d <+38>:    mov    -0x4(%rbp),%eax
   0x0000000000401130 <+41>:    lea    0x3(%rax),%edx
   0x0000000000401133 <+44>:    mov    -0x4(%rbp),%eax
   0x0000000000401136 <+47>:    cltq
   0x0000000000401138 <+49>:    mov    %edx,-0x330(%rbp,%rax,4)
   0x000000000040113f <+56>:    addl   $0x1,-0x4(%rbp)
   0x0000000000401143 <+60>:    cmpl   $0x63,-0x4(%rbp)
   0x0000000000401147 <+64>:    jle    0x40111b <madd+20>
   0x0000000000401149 <+66>:    movl   $0x0,-0x8(%rbp)
   0x0000000000401150 <+73>:    jmp    0x40117c <madd+117>
   0x0000000000401152 <+75>:    mov    -0x8(%rbp),%eax
   0x0000000000401155 <+78>:    cltq
   0x0000000000401157 <+80>:    mov    -0x1a0(%rbp,%rax,4),%edx
   0x000000000040115e <+87>:    mov    -0x8(%rbp),%eax
   0x0000000000401161 <+90>:    cltq
   0x0000000000401163 <+92>:    mov    -0x330(%rbp,%rax,4),%eax
   0x000000000040116a <+99>:    add    %eax,%edx
   0x000000000040116c <+101>:   mov    -0x8(%rbp),%eax
--Type <RET> for more, q to quit, c to continue without paging--c
   0x000000000040116f <+104>:   cltq
   0x0000000000401171 <+106>:   mov    %edx,-0x1a0(%rbp,%rax,4)
=> 0x0000000000401178 <+113>:   addl   $0x1,-0x8(%rbp)
   0x000000000040117c <+117>:   cmpl   $0x63,-0x8(%rbp)
   0x0000000000401180 <+121>:   jle    0x401152 <madd+75>
   0x0000000000401182 <+123>:   nop
   0x0000000000401183 <+124>:   leaveq
   0x0000000000401184 <+125>:   retq
End of assembler dump.
```

图13

如图 12，第一次停在断点处时，指在了<+75>处，接着执行了 10 个 nexti 至 $a[i]+=b[i]$ 完成；再次查看汇编指令，发现指在<+113>处；故语句 $a[i]+=b[i]$ 所对应的汇编指令就是<+75>至<+106>处，即图 13 中高亮部分。

分析思路 2：若要确定 $a[i]$ 和 $b[i]$ 所在寄存器，可利用这两变量的值的变化来确定。使用 nexti 命令逐步执行，通过 print 打印 $a[i]$ 的值来确定其是否变化，若发生变化，查看汇编指令，通过此时指示行附近的 mov 或 add 等指令来猜测寄存器，并使用命令 info register 查看其中的值来验证，具体操作过程如下：

```
Breakpoint 1, madd () at Lab1.c:11
11                      a[i] += b[i];
(gdb) disas
Dump of assembler code for function madd:
   0x0000000000401107 <+0>:     push   %rbp
   0x0000000000401108 <+1>:     mov    %rsp,%rbp
   0x000000000040110b <+4>:     sub    $0x2b8,%rsp
   0x0000000000401112 <+11>:    movl   $0x0,-0x4(%rbp)
   0x0000000000401119 <+18>:    jmp    0x401143 <madd+60>
   0x000000000040111b <+20>:    mov    -0x4(%rbp),%eax
   0x000000000040111e <+23>:    lea    -0x32(%rax),%edx
   0x0000000000401121 <+26>:    mov    -0x4(%rbp),%eax
   0x0000000000401124 <+29>:    cltq
   0x0000000000401126 <+31>:    mov    %edx,-0x1a0(%rbp,%rax,4)
   0x000000000040112d <+38>:    mov    -0x4(%rbp),%eax
   0x0000000000401130 <+41>:    lea    0x3(%rax),%edx
   0x0000000000401133 <+44>:    mov    -0x4(%rbp),%eax
   0x0000000000401136 <+47>:    cltq
   0x0000000000401138 <+49>:    mov    %edx,-0x330(%rbp,%rax,4)
   0x000000000040113f <+56>:    addl   $0x1,-0x4(%rbp)
   0x0000000000401143 <+60>:    cmpl   $0x63,-0x4(%rbp)
   0x0000000000401147 <+64>:    jle    0x40111b <madd+20>
   0x0000000000401149 <+66>:    movl   $0x0,-0x8(%rbp)
   0x0000000000401150 <+73>:    jmp    0x40117c <madd+117>
=> 0x0000000000401152 <+75>:    mov    -0x8(%rbp),%eax
   0x0000000000401155 <+78>:    cltq
   0x0000000000401157 <+80>:    mov    -0x1a0(%rbp,%rax,4),%edx
   0x000000000040115e <+87>:    mov    -0x8(%rbp),%eax
   0x0000000000401161 <+90>:    cltq
   0x0000000000401163 <+92>:    mov    -0x330(%rbp,%rax,4),%eax
   0x000000000040116a <+99>:    add    %eax,%edx
   0x000000000040116c <+101>:   mov    -0x8(%rbp),%eax
   0x000000000040116f <+104>:   cltq
   0x0000000000401171 <+106>:   mov    %edx,-0x1a0(%rbp,%rax,4)
   0x0000000000401178 <+113>:   addl   $0x1,-0x8(%rbp)
   0x000000000040117c <+117>:   cmpl   $0x63,-0x8(%rbp)
   0x0000000000401180 <+121>:   jle    0x401152 <madd+75>
   0x0000000000401182 <+123>:   nop
   0x0000000000401183 <+124>:   leaveq
--Type <RET> for more, q to quit, c to continue without paging--q
Quit
(gdb) print a[i]
$1 = -50
(gdb) ni
0x0000000000401155      11                      a[i] += b[i];
(gdb) print a[i]
$2 = -50
```

图14-第一次断点时的值为 -50

```
(gdb) ni
0x0000000000401171      11                      a[i] += b[i];
(gdb) p a[i]
$10 = -50
(gdb) ni
10              for(int i=0; i<100; i++)
(gdb) p a[i]
$11 = -47
(gdb) disas
Dump of assembler code for function madd:
   0x0000000000401107 <+0>:     push   %rbp
   0x0000000000401108 <+1>:     mov    %rsp,%rbp
   0x000000000040110b <+4>:     sub    $0x2b8,%rsp
   0x0000000000401112 <+11>:    movl   $0x0,-0x4(%rbp)
   0x0000000000401119 <+18>:    jmp    0x401143 <madd+60>
   0x000000000040111b <+20>:    mov    -0x4(%rbp),%eax
   0x000000000040111e <+23>:    lea    -0x32(%rax),%edx
   0x0000000000401121 <+26>:    mov    -0x4(%rbp),%eax
   0x0000000000401124 <+29>:    cltq
   0x0000000000401126 <+31>:    mov    %edx,-0x1a0(%rbp,%rax,4)
   0x000000000040112d <+38>:    mov    -0x4(%rbp),%eax
   0x0000000000401130 <+41>:    lea    0x3(%rax),%edx
   0x0000000000401133 <+44>:    mov    -0x4(%rbp),%eax
   0x0000000000401136 <+47>:    cltq
   0x0000000000401138 <+49>:    mov    %edx,-0x330(%rbp,%rax,4)
   0x000000000040113f <+56>:    addl   $0x1,-0x4(%rbp)
   0x0000000000401143 <+60>:    cmpl   $0x63,-0x4(%rbp)
   0x0000000000401147 <+64>:    jle    0x40111b <madd+20>
   0x0000000000401149 <+66>:    movl   $0x0,-0x8(%rbp)
   0x0000000000401150 <+73>:    jmp    0x40117c <madd+117>
   0x0000000000401152 <+75>:    mov    -0x8(%rbp),%eax
   0x0000000000401155 <+78>:    cltq
   0x0000000000401157 <+80>:    mov    -0x1a0(%rbp,%rax,4),%edx
   0x000000000040115e <+87>:    mov    -0x8(%rbp),%eax
   0x0000000000401161 <+90>:    cltq
   0x0000000000401163 <+92>:    mov    -0x330(%rbp,%rax,4),%eax
   0x000000000040116a <+99>:    add    %eax,%edx
   0x000000000040116c <+101>:   mov    -0x8(%rbp),%eax
   0x000000000040116f <+104>:   cltq
   0x0000000000401171 <+106>:   mov    %edx,-0x1a0(%rbp,%rax,4)
=> 0x0000000000401178 <+113>:   addl   $0x1,-0x8(%rbp)
   0x000000000040117c <+117>:   cmpl   $0x63,-0x8(%rbp)
   0x0000000000401180 <+121>:   jle    0x401152 <madd+75>
   0x0000000000401182 <+123>:   nop
   0x0000000000401183 <+124>:   leaveq
--Type <RET> for more, q to quit, c to continue without paging--c
   0x0000000000401184 <+125>:   retq
End of assembler dump.
```

图15-值恰好更新为-47

观察图 14 到图 15，$a[i]$ 的值从-50 更新至-47（$-47 = -50 + 3$），观察<+75>至<+106>这段汇编指令，发现寄存器%eax,%edx 可能储存了 $a[i]$ 和 $b[i]$ 的值，分别打印其中的值如图 16。不难发现，%edx 是存储 $a[i]$ 的寄存器，而%eax 存储的却是循环里的 i；那么猜测是 $b[i]$ 所在的寄存器的值发生了更新，覆盖了原先的 $b[i]$，接下来验证这个猜测：

```
(gdb) info register eax
eax            0x0                 0
(gdb) info register edx
edx            0xffffffd1          -47
(gdb) ni

Breakpoint 1, madd () at Lab1.c:11
11                      a[i] += b[i];
(gdb) info register eax
eax            0x0                 0
(gdb) info register edx
edx            0xffffffd1          -47
(gdb) ni
0x0000000000401155      11                      a[i] += b[i];
(gdb) info register eax
eax            0x1                 1
(gdb) info register edx
edx            0xffffffd1          -47
(gdb) ni
0x0000000000401157      11                      a[i] += b[i];
(gdb) info register eax
eax            0x1                 1
(gdb) info register edx
edx            0xffffffd1          -47
(gdb) ni
0x000000000040115e      11                      a[i] += b[i];
(gdb) info register eax
eax            0x1                 1
(gdb) info register edx
edx            0xffffffcf          -49
```

图16-猜测寄存器并验证

```
(gdb) ni
0x0000000000401161      11                      a[i] += b[i];
(gdb) info register eax
eax            0x1                 1
(gdb) ni
0x0000000000401163      11                      a[i] += b[i];
(gdb) info register eax
eax            0x1                 1
(gdb) ni
0x000000000040116a      11                      a[i] += b[i];
(gdb) info register eax
eax            0x4                 4
(gdb) ni
0x000000000040116c      11                      a[i] += b[i];
(gdb) info register eax
eax            0x4                 4
(gdb) ni
0x000000000040116f      11                      a[i] += b[i];
(gdb) info register eax
eax            0x1                 1
(gdb) ni
0x0000000000401171      11                      a[i] += b[i];
(gdb) info register eax
eax            0x1                 1
(gdb) info register edx
edx            0xffffffd3          -45
```

图17-验证对%eax的猜测

通过继续执行 nexti 并观察%eax 的值的变化，发现%eax 在执行过程中既存储 i 的值（图 17 中的 1），也存储 $b[i]$ 的值（图 17 中的 4）。故 $a[i]$ 位于寄存器%edx 中，$b[i]$ 位于寄存器%eax 中，同时%eax 还具有存储循环变量 i 的作用。

6. **实验内容第四项使用单步指令及 gdb 相关命令，显示 $a[xy]+b[xy]$ 对应的汇编指令执行前后操作数寄存器十进制和十六进制的值：**

其中 x，y 取自于学生本人学号 2022211$x*y$ 的百位和个位，我的学号是 2022211683，因此 x=6，y=3，需显示 $a[63]+=b[63]$ 前后对应寄存器的值。

分析思路：设置条件为 $i=63$ 的条件断点，使得 i=63 时程序暂停，然后使用命令 nexti 逐步执行，同时跟踪寄存器%eax 和%edx 的值。具体操作过程如下：

```
(gdb) break 11 if i==63
Breakpoint 1 at 0x401152: file Lab1.c, line 11.
(gdb) run
Starting program: /students/2022211683/Lab1

Breakpoint 1, madd () at Lab1.c:11
11                      a[i] += b[i];
(gdb) p i
$1 = 63
(gdb) p a[i]
$2 = 13
(gdb) info register edx
edx            0x4d               77
(gdb) info register eax
eax            0x3e               62
(gdb) ni
0x0000000000401155      11                      a[i] += b[i];
(gdb) info register edx
edx            0x4d               77
(gdb) info register eax
eax            0x3f               63
(gdb) ni
0x0000000000401157      11                      a[i] += b[i];
(gdb) info register edx
edx            0x4d               77
(gdb) info register eax
eax            0x3f               63
(gdb) ni
0x000000000040115e      11                      a[i] += b[i];
(gdb) info register edx
edx            0xd                13
(gdb) info register eax
eax            0x3f               63
(gdb) ni
0x0000000000401161      11                      a[i] += b[i];
(gdb) info register edx
edx            0xd                13
(gdb) info register eax
eax            0x3f               63
(gdb) ni
0x0000000000401163      11                      a[i] += b[i];
(gdb) info register edx
edx            0xd                13
(gdb) info register eax
eax            0x3f               63
```

图18

```
(gdb) ni
0x000000000040115e      11                      a[i] += b[i];
(gdb) info register edx
edx            0xd                13
(gdb) info register eax
eax            0x3f               63
(gdb) ni
0x0000000000401161      11                      a[i] += b[i];
(gdb) info register edx
edx            0xd                13
(gdb) info register eax
eax            0x3f               63
(gdb) ni
0x0000000000401163      11                      a[i] += b[i];
(gdb) info register edx
edx            0xd                13
(gdb) info register eax
eax            0x3f               63
(gdb) ni
0x000000000040116a      11                      a[i] += b[i];
(gdb) info register edx
edx            0xd                13
(gdb) info register eax
eax            0x42               66
(gdb) ni
0x000000000040116c      11                      a[i] += b[i];
(gdb) info register edx
edx            0x4f               79
(gdb) info register eax
eax            0x42               66
(gdb) ni
0x000000000040116f      11                      a[i] += b[i];
(gdb) info register edx
edx            0x4f               79
(gdb) info register eax
eax            0x3f               63
(gdb) ni
0x0000000000401171      11                      a[i] += b[i];
(gdb) info register edx
edx            0x4f               79
(gdb) info register eax
eax            0x3f               63
(gdb) ni
10              for(int i=0; i<100; i++)
(gdb)
```

图19

如图 18、图 19，一开始寄存器%edx 的值为 77，寄存器%eax 的值为 62，因为此时的%edx 和%eax 仍存储着上一次循环的结果（%edx 存储 $a[62]$,%eax 存储 i）；执行 nexti 之后，%eax 更新为 63（即 i++），继续执行，%edx 更新为 13（即 $a[63]=63-50=13$）；继续执行，%eax 更新为 66（即 $b[63]=63+3=66$），%edx 更新为 79（即 $a[63]+b[63]=13+66=79$）；执行完完整的 $a[i]+=b[i]$ 后，%eax 更新为 63（即 i），%edx 仍存储 79（即 $a[63]+b[63]$）。

五、总结体会

1. **实验内容总结：** 在本次实验中，初步掌握了 vi 编辑器的操作语句以及 gdb 调试工具的使用方法，为后续的计算机系统实验打下了基础。虽然在实验过程中遇到了许多问题，但是通过不懈努力，成功地解决了。
2. **所遇问题及解决：** 主要困难之一是对工具的不熟练使用。因为平时都使用集成开发环境来编写代码，所以一开始对于 vi 编辑器的使用感到非常不适应。编辑器的操作和错误修改都显得相当繁琐，尤其是光标没有办法像集成开发环境那样随意移动。为了克服这个困难，我花了很多时间练习使用 vi，逐渐熟悉了其中的指令，并且努力记忆了这些操作，以提高我的编辑效率，同时搜集了一些 vim 的资料以供日后继续提升。

另外，与集成开发环境相比，gdb 更接近程序运行的底层，因此要求对程序的运行过程有更深入的了解。为了克服这个困难，我学习尝试了 gdb 的各种调试指令，逐步掌握了它的使用方法。通过多次调试，我逐渐发现 gdb 的强大之处，它可以帮助我查看程序中任何位置的变量值，并且可以逐行分析汇编代码，相比 vscode 那种固定好的调试方式，gdb 很显著地提高了我对程序运行的理解以及调试程序的能力。

3. **报告总结：** 详细的实验报告对于实验经历的总结非常重要。在实验中，我曾尝试过多次预实验，认为自己已经理解了所有内容。然而，在撰写实验报告的过程中，我发现需要仔细地解释每一个细节，这使得我经常重复同一个操作来获取第一次没获取到的细节。这个繁琐的过程使我意识到，详细的实验报告不仅仅是对实验内容的总结，更促使我思考每一个细节，也让我养成了一边实验一边记录的习惯，而不再是以前一次性做完然后凭记忆撰写的方式。
4. **意见和建议：** 对于未来的实验，我建议将学生在实验结束后遇到的各种细节问题进行汇总，并且组织讨论或公开讲解，以便大家共同学习好的解决办法。我认为这样的做法不仅能够帮助我们更好地掌握实验知识，也能够促使我们在解决问题的过程中提高自己的分析和解决问题的能力。