北京师范大学2018-2019学年第一学期 “汇编程序设计”

上机作业八

姓 名	学 号	日 期
刘源	201611210134	2018.11.19

实验目的

- **学会g++编译工具的使用：编译汇编代码和目标文件，并链接成可执行文件。**
- **理解多文件的编译和链接时函数接口规范。**

实验总结：

不同版本的g++要使用的汇编代码进行联合编译时，所处的标准也不一样，也要注意add、sub指令使用的时候，谁先谁后的问题。

第 1 题

编写三个C++文件：main.cpp add.hpp add.cpp

```
/*main.cpp*/
#include <iostream>
#include "add.hpp"
using namespace std;

int main()
{
    int a=1;
    int b=2;
    int c=0;

    c = add(a, b);

    cout << "c=" << c << endl;
    return 0;
}

/*add.hpp*/
```

```
int add(int a, int b);

/*add.cpp*/
int add(int a, int b)
{
    return a+b;
}
```

1. 手动编译汇编文件
2. 为main.s和add.s文件中的汇编语句添加注释

```
/*main.s*/
    .file   "main.cpp"
.lcomm __ZStL8__ioinit,1,1
    .def    ___main;    .scl    2;  .type   32; .endef
    .section .rdata,"dr"
LC0:
    .ascii "c=\0"
    .text
    .globl  _main
    .def    _main;  .scl    2;  .type   32; .endef
_main:
    leal    4(%esp), %ecx
    andl    $-16, %esp
    pushl   -4(%ecx)
    pushl   %ebp
    movl    %esp, %ebp
    pushl %ecx
    subl    $36, %esp               # 开辟栈空间
    call    ___main
    movl    $1, -12(%ebp)           # int a = 1
    movl    $2, -16(%ebp)           # int b = 2
    movl    $0, -20(%ebp)           # int c = 0
    movl    -16(%ebp), %eax         # ax = b
    movl    %eax, 4(%esp)           # 将b压入栈
    movl    -12(%ebp), %eax         # ax = a
    movl    %eax, (%esp)            # 将a压入栈
    call    __Z3addii
    movl    %eax, -20(%ebp)
    movl    $LC0, 4(%esp)
    movl    $__ZSt4cout, (%esp)
    call    __ZStlsISt11char_traitsIcEERSt13basic_ostreamIcT_ES5_PKc
    movl    %eax, %edx
    movl    -20(%ebp), %eax
    movl    %eax, (%esp)
    movl    %edx, %ecx
    call    __ZNSolsEi
    subl    $4, %esp
    movl    $__ZSt4endlIcSt11char_traitsIcEERSt13basic_ostreamIT_T0_ES6_, (%esp)
    movl    %eax, %ecx
    call    __ZNSolsEPFRSoS_E
    subl    $4, %esp
    movl    $0, %eax
```

```
    movl    -4(%ebp), %ecx
    leave
    leal    -4(%ecx), %esp
    ret
    .def    ___tcf_0;   .scl    3;  .type   32; .endef
___tcf_0:
    pushl   %ebp
    movl    %esp, %ebp
    subl    $8, %esp
    movl    $__ZStL8__ioinit, %ecx
    call    __ZNSt8ios_base4InitD1Ev
    leave
    ret
    .def    __Z41__static_initialization_and_destruction_0ii;   .scl    3;  .type   32;
.endef
__Z41__static_initialization_and_destruction_0ii:
    pushl   %ebp
    movl    %esp, %ebp
    subl    $24, %esp
    cmpl    $1, 8(%ebp)
    jne L4
    cmpl    $65535, 12(%ebp)
    jne L4
    movl    $__ZStL8__ioinit, %ecx
    call    __ZNSt8ios_base4InitC1Ev
    movl    $___tcf_0, (%esp)
    call    _atexit
L4:
    leave
    ret
    .def    __GLOBAL__sub_I_main;   .scl    3;  .type   32; .endef
__GLOBAL__sub_I_main:
    pushl   %ebp
    movl    %esp, %ebp
    subl    $24, %esp
    movl    $65535, 4(%esp)
    movl    $1, (%esp)
    call    __Z41__static_initialization_and_destruction_0ii
    leave
    ret
    .section    .ctors,"w"
    .align 4
    .long   __GLOBAL__sub_I_main
    .ident  "GCC: (tdm-1) 4.9.2"
    .def    __Z3addii;  .scl    2;  .type   32; .endef
    .def    __ZStlsISt11char_traitsIcEERSt13basic_ostreamIcT_ES5_PKc;   .scl    2;
.type   32; .endef
    .def    __ZNSolsEi; .scl    2;  .type   32; .endef
    .def    __ZSt4endlIcSt11char_traitsIcEERSt13basic_ostreamIT_T0_ES6_;    .scl    2;
.type   32; .endef
    .def    __ZNSolsEPFRSoS_E;  .scl    2;  .type   32; .endef
    .def    __ZNSt8ios_base4InitD1Ev;   .scl    2;  .type   32; .endef
    .def    __ZNSt8ios_base4InitC1Ev;   .scl    2;  .type   32; .endef
```

```
    .def    _atexit;    .scl    2;  .type   32; .endef

/*add.s*/
    .file   "add.cpp"           # 源程序名称
    .text
    .globl  __Z3addii           # 定义子程序名称
    .def    __Z3addii;  .scl    2;  .type   32; .endef
__Z3addii:
    pushl   %ebp                # 将bp寄存器的值push进栈
    movl    %esp, %ebp          # bp = sp
    movl    8(%ebp), %edx       # dx = 8(%ebp) = a
    movl    12(%ebp), %eax      # ax = 12(%ebp) = b
    addl    %edx, %eax          # ax = ax + dx
    popl    %ebp                # 还原bp的值，即指向源代码的位置
    ret
    .ident  "GCC: (tdm-1) 4.9.2"
```

第 2 题

自己手动编写一个add2.s 文件，执行加法的任务是计算a到b（默认第1个参数小于第2个参数，且包含b）。然后将自己编写的add2.s生成目标文件，并和原来的main.o进行连接生成可执文件。

```
/*add2.s*/
    .file   "add.cpp"           # 源程序名称
    .text
    .globl  __Z3addii           # 定义子程序名称
    .def    __Z3addii;  .scl    2;  .type   32; .endef
__Z3addii:
    pushl   %ebp                # 将bp寄存器的值push进栈
    movl    %esp, %ebp          # bp = sp
    movl    8(%ebp), %edx       # dx = 8(%ebp) = a
    movl    12(%ebp), %eax      # ax = 12(%ebp) = b
    addl    %edx, %eax          # eax = b+a
    leal    -1(%eax), %edx
    movl    12(%ebp), %eax
    subl    8(%ebp), %eax
    imull   %edx, %eax
    shr     $1, %eax
    popl    %ebp                # 还原bp的值，即指向源代码的位置
    ret
    .ident  "GCC: (tdm-1) 4.9.2"
```

实验测试:（成功）