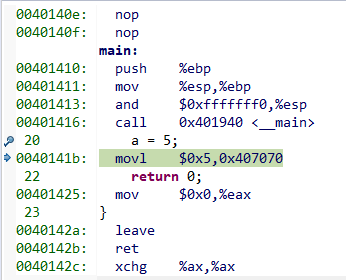
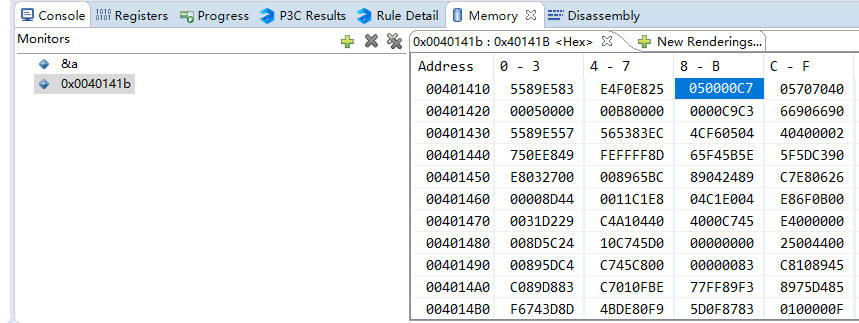
# 实验三

备注：编译器是gcc，内存在eclipse中查看的

1. 实验第三题
2. 程序1





该指令的机器代码是050000C7

1. 程序2

mov 0x404008,%edx

mov %edx,%eax

add %eax,%eax

add %eax,%edx

mov 0x404004,%eax

add %edx,%eax

mov %eax,0x407070

使用了mov、add指令，乘法用多次add来实现，赋值用move来实现，用到了两个寄存器，edx和eax，edx：基地址寄存器，内存寻址时，用来存放地址；eax：累加寄存器，用于暂时存放计算结果。

1. 程序3

mov 0x404004,%edx

mov 0x404008,%eax

and %edx,%eax

mov %eax,0x407020

1. 程序4

mov 0x404004,%eax

test %eax,%eax

je 0x401437 <main+39>

mov 0x404008,%eax

test %eax,%eax

je 0x401437 <main+39>

mov $0x1,%eax

jmp 0x40143c <main+44>

mov $0x0,%eax

mov %eax,0x407020

1. 程序5

mov 0x407070,%eax

test %eax,%eax

jle 0x40143d <main+45>

框架如下：

mov //装值，可以是edx、 eax两种寄存器

cmp/test //比较，改变

je/jle/jne…… //跳转到某处

其他指令 //完成满足条件的事

跳转的位置 //不满足条件跳到此处

1. 程序6

mov 0x407070,%eax

test %eax,%eax

jle 0x40143f <main+47>

movl $0x405044,(%esp)

call 0x403ac0 <puts>

jmp 0x40144b <main+59>

movl $0x405049,(%esp)

call 0x403ac0 <puts>

框架如下：

mov //装值，可以是edx、 eax两种寄存器

cmp/test //比较，改变

je/jle/jne…… //跳转到某处

其他指令 //满足if之后执行

jmp //无条件跳转，跳到程序结束处

其他指令 //不满足if跳转到的地方

1. 程序7

mov 0x407070,%eax

cmp $0x2,%eax

je 0x401445 <main+53>

cmp $0x3,%eax

je 0x40144f <main+63>

cmp $0x1,%eax

jne 0x40145b <main+75>

movl $0x1,0x407070

jmp 0x401465 <main+85>

movl $0x2,0x407070

movl $0x3,0x407070

jmp 0x401465 <main+85>

movl $0x5,0x407070

框架如下：

mov //装值

cmp //数个比较、跳转，用于实现case子句，第一个case子句写到最后用jne，跳转到//default，紧接着第一个case子句的内容。

je/jne…… //重复这两条指令

其他指令 //实现case子句的内容

jmp //无条件跳转实现break;

指令 //实现default的内容

1. 程序8

movl $0x5,0x407070

jmp 0x40143c <main+44>

addl $0x1,0xc(%esp)

mov 0x407070,%eax

sub $0x1,%eax

mov %eax,0x407070

mov 0x407070,%eax

test %eax,%eax

jg 0x40142a <main+26>

框架如下：

mov //装值

jmp //无条件跳转到判断部分

其他指令 //实现循环体内部需要做的事情

mov //装入值

cmp/test //循环判断的条件

jg/je…… //跳到循环体内部执行处

1. 程序9

movl $0x5,0x407070

addl $0x1,0xc(%esp)

mov 0x407070,%eax

sub $0x1,%eax

mov %eax,0x407070

mov 0x407070,%eax

test %eax,%eax

jg 0x401428 <main+24>

框架如下：

其他指令 //负责do内容的实现

mov //装值

cmp/test //循环判断比较

jmp/jg…… //满足条件跳转到do内容开头处

1. 程序10

movl $0x5,0x407070

jmp 0x40144b <main+59>

mov 0x407074,%eax

add $0x1,%eax

mov %eax,0x407074

mov 0x407070,%eax

ub $0x1,%eax

mov %eax,0x407070

mov 0x407070,%eax

test %eax,%eax

jg 0x401431 <main+33>

框架如下：

mov //装值

jmp //跳转到对参数实行操作的地方

其他指令 //for循环内部内容实现

mov //装值进入eax寄存器

add/sub //对值进行+1或-1操作

test/cmp //循环条件的判定

jg/jl…… //满足条件，跳转到其他实现for内容指令处

1. 程序11

add:

push %ebp //备份当前帧指针

mov %esp,%ebp //调整当前帧指针到栈指针的位置

sub $0x10,%esp //调用时需给临时变量预留内存

mov 0x8(%ebp),%edx //从栈中取出参数的值

mov 0xc(%ebp),%eax //从栈中取出参数的值

add %edx,%eax

mov %eax,-0x4(%ebp)

mov -0x4(%ebp),%eax //函数返回值

leave //一共做两件事，第一件是将栈指针指向帧指针，第二件是弹出备份的原帧指针到%ebp

ret //做两件事第一件是将栈顶的返回地址弹出到PC，第二件事则是按照PC此时指示的指令地址继续执行程序

mov 0x18(%esp),%eax

mov %eax,0x4(%esp)

mov 0x1c(%esp),%eax

mov %eax,(%esp)

call 0x401410 <add> //做两件事，第一件是将返回地址（也就是call指令执行时PC的值）压入栈顶，第二件是将程序跳转到当前调用的方法的起始地址

框架如下：

函数调用之初，会将参数压入栈中，即上述call指令之前得四个mov指令。

函数传递是用call来实行跳转，此时还要将PC值压入栈顶。

函数内部，首先两行指令，备份帧指针，并将栈指针值赋给帧指针，使程序切换到函数处执行。之后leave与ret执行收尾操作，leave负责栈指针的重新定位和帧指针的还原，ret负责PC还原和继续执行。

函数返回值存储在eax中来传递这个值。

1. 实验第五题

**.globl** \_main

.def \_main; .scl 2; **.type** 32; .endef

**\_main:**

movl $1, **%ecx**

movl $2, **%edx**

call xaddy

leave

ret

**xaddy:**

pushl **%ebp**

movl **%esp**, **%ebp**

movl **%ecx**, **%eax**

addl **%edx**, **%eax**

leave

ret

1. 实验第六题

机器代码为：050000A1 FF3560FE 2800C3B8（具体见下图：）

