

电子科技大学

软件开发环境实验一：流程控制语句反汇编

学生姓名：任振华 学号：2017060801023

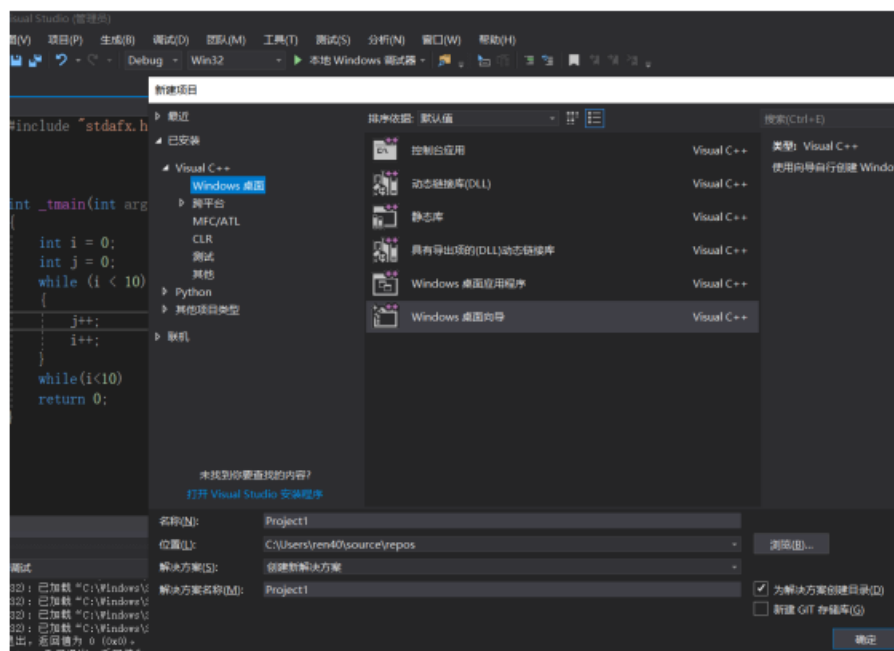
指导教师：李林 实验时间：2019/12/18

1 实验目的

本实验总体目的是，通过使用 Visual Studio 2008 查看 if、if/else、do/while/for 等类型语句的反汇编代码，以达到掌握流程控制语句识别的目的。

本实验学时数为 2 学时。

2.1 工程创建



2.2 if 语句的反汇编

代码清单 1:

以下是 if 语句源代码

```
int _tmain(int argc, _TCHAR* argv[])
{
    int i = 3;
    if (i > 3) {
        i = 4;
    }

    return 0;
}
```

反汇编代码如下

```
int i = 3;
0040145E mov     dword ptr [i], 3
    if (i > 3) {
00401465 cmp     dword ptr [i], 3
00401469 jle     wmain+32h (0401472h)
        i = 4;
0040146B mov     dword ptr [i], 4
    }

    return 0;
00401472 xor     eax, eax
}
```

mov dword ptr [i], 3 的意思是 *i = 3, ptr 是属性控制字, dword 是双字, 也就是四个字节, 这条指令就是将 3 送到 i 指向的地址
cmp 相当于减法运算, 判断两个操作数大小, 用来作 jle 跳转指令的执行条件
jle 是小于等于的时候跳转, 就是第一个操作数小于或等于第二个操作数时跳转, 跳转到 0401472h 地址, 也就是 return 0 下面左边写的地址
mov ptr [i], 4 同理, 是将 4 送到 i 所指向的地址
xor eax, eax 异或, 对 eax 寄存器清 0

代码清单 3

```
int _tmain(int argc, _TCHAR* argv[])
{
    int i = 3;
    if (i < 3) {
        i = 4;
    }

    return 0;
}
```

```
    int i = 3;
0040145E  mov     dword ptr [i],3
    if (i < 3) {
00401465  cmp     dword ptr [i],3
00401469  jge     wmain+32h (0401472h)
        i = 4;
0040146B  mov     dword ptr [i],4
    }

    return 0;
00401472  xor     eax, eax
}
```

mov dword ptr [i], 3 将 3 送到 i 指向的地址

cmp dword ptr [i], 3 比较 i 指向地址里的数和 3 的大小

jge 大于等于的时候跳转，如果跳转，回到 0401472h 地址

mov dword [i], 4 将 3 送到 i 指向的地址

xor eax, eax 异或，对 eax 寄存器清 0

代码清单 4

```
int _tmain(int argc, _TCHAR* argv[])
{
    int i = 3;
    if (i == 3) {
        i = 4;
    }

    return 0;
}
```

```
int i = 3;
0040145E mov     dword ptr [i], 3    已用时间 <= 1ms
    if (i == 3) {
00401465 cmp     dword ptr [i], 3
00401469 jne     wmain+32h (0401472h)
        i = 4;
0040146B mov     dword ptr [i], 4
    }

    return 0;
00401472 xor     eax, eax
}
```

mov dword ptr [i], 3 将 3 送到 i 指向的地址

cmp dword ptr [i], 3 比较 i 指向地址里的数和 3 的大小

jne ZF=0 的时候跳转, ZF=0 的条件是 cmp 指令比较的两个操作数不相等

mov dword [i], 4 将 3 送到 i 指向的地址

xor eax, eax 异或, 对 eax 寄存器清 0

条件判断语句的反汇编规则

条件判断语句常用 cmp, 和 jg, jl, jge, jle, jne 等指令结合。

cmp 指令是比较两个操作数的指令, 往往是 jg, jl, jge, jle, jne 等指令的执行条件。

2.3if/else 语句的反汇编

代码清单 5

```
6
7
8  int _tmain(int argc, _TCHAR* argv[])
9  {
10     int i = 3;
11     if (i < 3) {
12         i = 4;
13     }
14     else
15     {
16         i = 5;
17     }
18     return 0;
19 }
20
```

```
    int i = 3;
0040145E mov     dword ptr [i],3    已用时间 <= 1ms
    if (i < 3) {
00401465 cmp     dword ptr [i],3
00401469 jge     wmain+34h (0401474h)
        i = 4;
0040146B mov     dword ptr [i],4
    }
    else
00401472 jmp     wmain+3Bh (040147Bh)
    {
        i = 5;
00401474 mov     dword ptr [i],5
    }
    return 0;
0040147B xor     eax,eax
}
```

0040145E mov dword ptr [i],3

将3送到i所指向的那个地址

00401465 cmp dword ptr [i],3

是比较*i和3的大小

00401469 jge wmain+34h (0401474h)

jge是大于等于的时候跳转, *i > 3时跳转到(0401474h)

```
0040146B  mov     dword ptr [i], 4
```

将4送到i所指向的那个地址

```
00401472  jmp     wmain+3Bh (040147Bh)
```

跳转到(040147Bh)的那个地址

```
00401474  mov     dword ptr [i], 5
```

将5送到i所指向的那个地址

```
0040147B  xor     eax, eax
```

xor 是异或指令, 将 eax 清 0

if-else 反汇编代码规律

if, else 语句主要有 cmp, jg, jl, jge, jle, jne 等指令结合, 还有 jmp 无条件跳转指令。

if 语句往往由 cmp, jg, jl, jge, jle, jne 等指令实现, else 指令有 jmp 指令实现。

代码清单 7

```
int _tmain(int argc, _TCHAR* argv[])
{
    int i = 3;
    if (i > 30) {
        i = 4;
    }
    else if (i >= 20)
    {
        i = 5;
    }
    else if (i <= 5)
    {
        i = 6;
    }
    else if (i < 10)
    {
        i = 7;
    }
    else if (i == 12)
    {
        i = 8;
    }
    else
    {
        i = 9;
    }
    return 0;
}
```

```

    int i = 3;
0040187E  mov     dword ptr [i],3
    if (i > 30) {
00401885  cmp     dword ptr [i],1Eh
00401889  jle     wmain+34h (0401894h)
        i = 4;
0040188B  mov     dword ptr [i],4
00401892  jmp     wmain+77h (04018D7h)
    }
    else if (i >= 20)
00401894  cmp     dword ptr [i],14h
00401898  jl      wmain+43h (04018A3h)
    {
        i = 5;
0040189A  mov     dword ptr [i],5
004018A1  jmp     wmain+77h (04018D7h)
    }

```

```

    else if (i <= 5)
004018A3  cmp     dword ptr [i],5
    }
    else if (i <= 5)
004018A7  jg      wmain+52h (04018B2h)
    {
        i = 6;
004018A9  mov     dword ptr [i],6
004018B0  jmp     wmain+77h (04018D7h)
    }
    else if (i < 10)
004018B2  cmp     dword ptr [i],0Ah
004018B6  jge     wmain+61h (04018C1h)
    {
        i = 7;
004018B8  mov     dword ptr [i],7
004018BF  jmp     wmain+77h (04018D7h)
    }
    else if (i == 12)

```

```

        i = 7;
004018B8  mov     dword ptr [i], 7
004018BF  jmp     wmain+77h (04018D7h)
    }
    else if (i == 12)
004018C1  cmp     dword ptr [i], 0Ch
004018C5  jne     wmain+70h (04018D0h)
    {
        i = 8;
004018C7  mov     dword ptr [i], 8
    }
    else
004018CE  jmp     wmain+77h (04018D7h)
    {
        i = 9;
004018D0  mov     dword ptr [i], 9
    }
    return 0;
004018D7  xor     eax, eax
    }

```

```

int i = 3;
0040187E  mov     dword ptr [i], 3
将3移动到i指向的地址
    if (i > 30) {
00401885  cmp     dword ptr [i], 1Eh
比较i指向地址里的数值和30的大小
00401889  jle     wmain+34h (0401894h)
jle表示小于等于的时候跳转到 (0401894h)
        i = 4;
0040188B  mov     dword ptr [i], 4
将4移动到i指向的地址
00401892  jmp     wmain+77h (04018D7h)
jmp是无条件跳转指令，直接跳转到 (04018D7h)地址
    }
    else if (i >= 20)
00401894  cmp     dword ptr [i], 14h
比较i指向地址里的数值和20的大小
00401898  jl      wmain+43h (04018A3h)
jl表示小于的时候跳转
    {
        i = 5;
0040189A  mov     dword ptr [i], 5
将5移动到i指向的地址

```



```

004018A1  jmp          wmain+77h (04018D7h)
jmp是无条件跳转指令，直接跳转到 (04018D7h)地址
}
    else if (i <= 5)
004018A3  cmp          dword ptr [i],5
cmp是比较指令，比较i指向地址里的数值和5的大小
}
    else if (i <= 5)
004018A7  jg          wmain+52h (04018B2h)
jg表示大于的时候发生跳转
{
    i = 6;
004018A9  mov          dword ptr [i],6
将6移动到i指向的地址
004018B0  jmp          wmain+77h (04018D7h)
jmp是无条件跳转指令，直接跳转到 (04018D7h)地址
}
    else if (i < 10)
004018B2  cmp          dword ptr [i],0Ah
cmp是比较指令，比较i指向地址里的数值和10 (0Ah) 的大小
004018B6  jge          wmain+61h (04018C1h)
jge表示大于等于的时候跳转
{
    i = 7;
004018B8  mov          dword ptr [i],7
将7移动到i指向的地址
004018BF  jmp          wmain+77h (04018D7h)
jmp是无条件跳转指令，直接跳转到 (04018D7h)地址
}
    else if (i == 12)
004018C1  cmp          dword ptr [i],0Ch
cmp是比较指令，比较i指向地址里的数值和12 (0Ch) 的大小
004018C5  jne          wmain+70h (04018D0h)
jne表示ZF=0，即两个操作数不相等的时候跳转
{
    i = 8;
004018C7  mov          dword ptr [i],8
将8移动到i指向的地址
}
    else
004018CE  jmp          wmain+77h (04018D7h)
jmp是无条件跳转指令，直接跳转到 (04018D7h)地址
{
    i = 9;

```

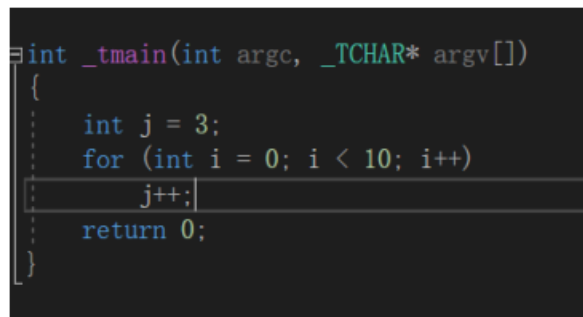
```
004018D0  mov             dword ptr [i],9
将9移动到i指向的地址
    }
    return 0;
004018D7  xor             eax,eax
xor 是异或指令，将 eax 清 0
```

if-else 构成的多分支流程的反汇编代码的规律

else 出现时往往有 jmp 指令，并且 jmp 指令跳转的地方都是同一个地方。
if 指令出现的时候往往时 cmp, jg, jl, jge, jle, jne 等指令配合

2.4 循环的反汇编

代码清单 8

A screenshot of a code editor showing a C function `_tmain`. The function takes `argc` and `argv` as parameters. Inside the function, there is a loop: `int j = 3;` followed by `for (int i = 0; i < 10; i++)`. The body of the loop contains `j++;` and `return 0;`. The code is displayed with syntax highlighting on a dark background.

```
int _tmain(int argc, _TCHAR* argv[])
{
    int j = 3;
    for (int i = 0; i < 10; i++)
    {
        j++;
    }
    return 0;
}
```

```

    int j = 0;
0040187E mov     dword ptr [j],0
    for (int i = 0; i < 10; i++)
00401885 mov     dword ptr [ebp-14h],0
0040188C jmp     wmain+37h (0401897h)
0040188E mov     eax,dword ptr [ebp-14h]
00401891 add     eax,1
00401894 mov     dword ptr [ebp-14h],eax
00401897 cmp     dword ptr [ebp-14h],0Ah
0040189B jge     wmain+48h (04018A8h)
    j++;
0040189D mov     eax,dword ptr [j]
004018A0 add     eax,1
004018A3 mov     dword ptr [j],eax
004018A6 jmp     wmain+2Eh (040188Eh)
    return 0;
004018A8 xor     eax,eax

```

```

int j = 0;
0040187E mov     dword ptr [j],0
将0送到j指向的地址
    for (int i = 0; i < 10; i++)
00401885 mov     dword ptr [ebp-14h],0
将[ebp-14h]即i设置为0
0040188C jmp     wmain+37h (0401897h)
jmp是无条件跳转指令，直接跳转到(0401897h)
0040188E mov     eax,dword ptr [ebp-14h]
将[ebp-14h]即i送到eax寄存器
00401891 add     eax,1
add是加法指令，将1和eax相加并把结果送回eax寄存器
00401894 mov     dword ptr [ebp-14h],eax
把将[ebp-14h]即i设置为eax寄存器的值
00401897 cmp     dword ptr [ebp-14h],0Ah
cmp是比较指令，比较[ebp-14h]即i与10（0Ah）的大小
0040189B jge     wmain+48h (04018A8h)
jge是大于等于的时候跳转，则意味这i大于等于10的时候，跳转到(04018A8h)
    j++;
0040189D mov     eax,dword ptr [j]
004018A0 add     eax,1
004018A3 mov     dword ptr [j],eax
上面三条指令是对j做+1运算
004018A6 jmp     wmain+2Eh (040188Eh)
jmp是无条件跳转指令，直接跳转到(040188Eh)
    return 0;
004018A8 xor     eax,eax

```

xor 是异或操作，使 eax 寄存器清 0

for 循环反汇编规律

for 循环执行完初始化条件以后，经过判断后进入循环体后，需要注意跳转地址，再执行完循环体后需要再次跳转回来进行判断。

这里面用的一个 cmp, jg, jl, jge, jle, jne 等指令配合的判断条件。

用到了 mov, add 指令结合的自增条件。

用到了多处 jmp 指令

代码清单 10

```
int _tmain(int argc, _TCHAR* argv[])
{
    int i = 0;
    int j = 0;
    while (i < 10)
    {
        j++;
        i++;
    }
    return 0;
}
```

```

    int j = 0;
00401885 mov     dword ptr [j],0
    while (i < 10)
0040188C cmp     dword ptr [i],0Ah
00401890 jge     wmain+46h (04018A6h)
    {
        j++;
00401892 mov     eax,dword ptr [j]
00401895 add     eax,1
00401898 mov     dword ptr [j],eax
        i++;
0040189B mov     eax,dword ptr [i]
0040189E add     eax,1
004018A1 mov     dword ptr [i],eax
    }
004018A4 jmp     wmain+2Ch (040188Ch)
    return 0;
004018A6 xor     eax,eax
}

```

```

int i = 0;
0040187E mov     dword ptr [i],0
使i=0
    int j = 0;
00401885 mov     dword ptr [j],0
使j=0
    while (i < 10)
0040188C cmp     dword ptr [i],0Ah
cmp是比较指令，比较i和10（0Ah）的大小
00401890 jge     wmain+46h (04018A6h)
    {
        j++;
00401892 mov     eax,dword ptr [j]
00401895 add     eax,1
00401898 mov     dword ptr [j],eax
上面三条指令是j++
        i++;
0040189B mov     eax,dword ptr [i]
0040189E add     eax,1
004018A1 mov     dword ptr [i],eax
上面三条指令i++
    }
004018A4 jmp     wmain+2Ch (040188Ch)
jmp无条件指令，直接跳转到(040188Ch)
    return 0;

```

004018A6 xor eax, eax
xor 是异或指令，是将 eax 清 0

代码清单 11

```
int _tmain(int argc, _TCHAR* argv[])
{
    int i = 0;
    int j = 0;
    while (i < 10)
    {
        j++;
        i++;
    }
    while(i<10)
    return 0;
}
```

```
int j = 0;
00401885 mov     dword ptr [j], 0
    while (i < 10)
0040188C cmp     dword ptr [i], 0Ah
00401890 jge     wmain+46h (04018A6h)
    {
        j++;
00401892 mov     eax, dword ptr [j]
00401895 add     eax, 1
00401898 mov     dword ptr [j], eax
        i++;
0040189B mov     eax, dword ptr [i]
0040189E add     eax, 1
004018A1 mov     dword ptr [i], eax
    }
004018A4 jmp     wmain+2Ch (040188Ch)
    while(i<10)
004018A6 cmp     dword ptr [i], 0Ah
004018AA jge     wmain+52h (04018B2h)
    return 0;
004018AC xor     eax, eax
    return 0;
004018AE jmp     wmain+58h (04018B8h)
004018B0 jmp     wmain+46h (04018A6h)
```

```
int i = 0;
0040187E mov     dword ptr [i], 0
使i=0
    int j = 0;
```

```

00401885  mov             dword ptr [j],0
使j=0
    while (i < 10)
0040188C  cmp             dword ptr [i],0Ah
00401890  jge             wmain+46h (04018A6h)
cmp是比较指令，jge是大于等于时跳转，上面两条指令共同使i<10
{
    j++;
00401892  mov             eax,dword ptr [j]
00401895  add             eax,1
00401898  mov             dword ptr [j],eax
上面三条指令让j++
    i++;
0040189B  mov             eax,dword ptr [i]
0040189E  add             eax,1
004018A1  mov             dword ptr [i],eax
上面三条指令让i++
}
004018A4  jmp             wmain+2Ch (040188Ch)
jmp是无条件跳转指令
    while(i<10)
004018A6  cmp             dword ptr [i],0Ah
004018AA  jge             wmain+52h (04018B2h)
上面两条指令让i<10,cmp是比较指令，比较i和10的大小，jge是大于等于转移指令
    return 0;
004018AC  xor             eax,eax
xor是异或指令，使eax寄存器清0
    return 0;
004018AE  jmp             wmain+58h (04018B8h)
004018B0  jmp             wmain+46h (04018A6h)
jmp 是无条件跳转指令

```

while 语句的反汇编代码规律

```

while()
A : cmp 操作数 1 操作数 2   while 循环结束条件做比较
jxx B           若不符合条件则，跳转到 B 处继续执行，若符合，
则顺序执行循环体
{               循环体指令
.....
.....

```

```

.....

}
    jmp A    跳转到 A 处继续执行
B: .....

```

do-while 循环的反汇编代码规律:

```

do
{   循环体指令代码
    A:.....
    .....
    .....
}while( );
cmp 操作数1 操作数2 do-----while循环 条件判断 语句
jxx A 如果符合条件的话,就跳转到A处的循环体部分开始执行,否则继续向下
执行

```