ЦЕЛЬ

1. Знакомство с программной архитектурой x86/x86-64.

2. Анализ ассемблерного листинга программы для архитектуры x86/x86-64

ЗАДАНИЕ

1. Изучить программную архитектуру x86/x86-64.

2. Для программы на языке Си (из лабораторной работы) сгенерировать ассемблерные листинги для архитектуры x86 и архитектуры x86-64, используя различные уровни комплексной оптимизации.

3. Проанализировать полученные листинги и сделать следующее.

ОПИСАНИЕ РАБОТЫ

Код выбранной мной программы на Си выглядит следующим образом:

```c
#include <stdio.h>

double Power(double number, double pow) {
double result = 1;
for (int i = 1; i <= pow; i++) {
result *= number;
}
return result;
}

int main() {
double n = 150;
double x = 0.5;
double sum = 0;
for (double i = 1; i <= n; i++) {
sum += (Power(-1, i + 1) * Power(x, i) / i);
}
printf("%f\n", sum);

return 0;

}
```

Листинг x86-64 -m32 -O0 gcc 13.2

```
1       Power:
2       pushl   %ebp
3       movl    %esp, %ebp
4       subl    $32, %esp
5       movl    8(%ebp), %eax
6       movl    12(%ebp), %edx
7       movl    %eax, -24(%ebp)
8       movl    %edx, -20(%ebp)
9       movl    16(%ebp), %eax
10      movl    20(%ebp), %edx
11      movl    %eax, -32(%ebp)
12      movl    %edx, -28(%ebp)
13      fld1
14      fstpl   -8(%ebp)
15      movl    $1, -12(%ebp)
16      jmp     .L2
17      .L3:
18      fldl    -8(%ebp)
19      fmull   -24(%ebp)
20      fstpl   -8(%ebp)
```

Листинг x86-64 -m32 -Ofast gcc 13.2

```
1       Power:
2       subl    $12, %esp
3       movsd   .LC0, %xmm0
4       movsd   24(%esp), %xmm2
5       movsd   16(%esp), %xmm3
6       comisd  %xmm0, %xmm2
7       jb      .L1
8       movl    $1, %eax
9       .L4:
10      mulsd   %xmm3, %xmm0
11      addl    $1, %eax
12      pxor    %xmm1, %xmm1
13      cvtsi2sdl       %eax, %xmm1
14      comisd  %xmm1, %xmm2
15      jnb     .L4
16      .L1:
17      movsd   %xmm0, (%esp)
18      fldl    (%esp)
19      addl    $12, %esp
20      ret
```

```
        addl    $1, -12(%ebp)
        .L2:
        fildl   -12(%ebp)
        fldl    -32(%ebp)
        fcomip  %st(1), %st
        fstp    %st(0)
        jnb     .L3
        fldl    -8(%ebp)
        leave
        ret
        .LC6:
        .string "%f\n"
        main:
        leal    4(%esp), %ecx
        andl    $-16, %esp
        pushl   -4(%ecx)
        pushl   %ebp
        movl    %esp, %ebp
        pushl   %ecx
        subl    $52, %esp
        fldl    .LC2
        fstpl   -32(%ebp)
        fldl    .LC3
        fstpl   -40(%ebp)
        fldz
        fstpl   -16(%ebp)
        fld1
        fstpl   -24(%ebp)
        jmp     .L6
        .L7:
        fldl    -24(%ebp)
        fld1
        faddp   %st, %st(1)
        leal    -8(%esp), %esp
        fstpl   (%esp)
        fld1
        fchs
        leal    -8(%esp), %esp
        fstpl   (%esp)
        call    Power
        addl    $16, %esp
        fstpl   -48(%ebp)
        pushl   -20(%ebp)
        pushl   -24(%ebp)
        pushl   -36(%ebp)
        pushl   -40(%ebp)
        call    Power
        addl    $16, %esp
        fmull   -48(%ebp)
        fdivl   -24(%ebp)
```

```
        .LC5:
        .string "%f\n"
        main:
        leal    4(%esp), %ecx
        andl    $-16, %esp
        movl    $150, %edx
        pushl   -4(%ecx)
        pushl   %ebp
        movl    %esp, %ebp
        pushl   %ecx
        subl    $20, %esp
        movl    $0, -16(%ebp)
        movsd   .LC0, %xmm7
        movq    .LC3, %xmm6
        movl    $0, -12(%ebp)
        movsd   .LC4, %xmm5
        movapd  %xmm7, %xmm4
        .L17:
        movapd  %xmm4, %xmm3
        pxor    %xmm1, %xmm1
        addsd   %xmm7, %xmm4
        movl    $1, %eax
        comisd  %xmm1, %xmm3
        movapd  %xmm7, %xmm0
        jb      .L12
        .L14:
        addl    $1, %eax
        pxor    %xmm1, %xmm1
        movapd  %xmm0, %xmm2
        cvtsi2sdl       %eax, %xmm1
        xorpd   %xmm6, %xmm2
        movapd  %xmm2, %xmm0
        comisd  %xmm1, %xmm4
        jnb     .L14
        comisd  %xmm7, %xmm3
        movl    $1, %eax
        movapd  %xmm7, %xmm1
        jb      .L12
        .L16:
        mulsd   %xmm5, %xmm1
        addl    $1, %eax
        pxor    %xmm0, %xmm0
        cvtsi2sdl       %eax, %xmm0
        comisd  %xmm0, %xmm3
        jnb     .L16
        movapd  %xmm2, %xmm0
        mulsd   %xmm1, %xmm0
        .L12:
        divsd   %xmm3, %xmm0
        addsd   -16(%ebp), %xmm0
```

```
71      fldl    -16(%ebp)
72      faddp   %st, %st(1)
73      fstpl   -16(%ebp)
74      fldl    -24(%ebp)
75      fld1
76      faddp   %st, %st(1)
77      fstpl   -24(%ebp)
78      .L6:
79      fldl    -24(%ebp)
80      fldl    -32(%ebp)
81      fcomip  %st(1), %st
82      fstp    %st(0)
83      jnb     .L7
84      subl    $4, %esp
85      pushl   -12(%ebp)
86      pushl   -16(%ebp)
87      pushl   $.LC6
88      call    printf
89      addl    $16, %esp
90      movl    $0, %eax
91      movl    -4(%ebp), %ecx
92      leave
93      leal    -4(%ecx), %esp
94      ret
95      .LC2:
96      .long   0
97      .long   1080213504
98      .LC3:
99      .long   0
100     .long   1071644672
```

```
71      movsd   %xmm0, -16(%ebp)
72      subl    $1, %edx
73      jne     .L17
74      subl    $12, %esp
75      movsd   %xmm0, (%esp)
76      pushl   $.LC5
77      call    printf
78      movl    -4(%ebp), %ecx
79      addl    $16, %esp
80      xorl    %eax, %eax
81      leave
82      leal    -4(%ecx), %esp
83      ret
84      .LC0:
85      .long   0
86      .long   1072693248
87      .LC3:
88      .long   0
89      .long   -2147483648
90      .long   0
91      .long   0
92      .LC4:
93      .long   0
94      .long   1071644672
```

Листинг x86-64 -O0 gcc 13.2

```
1       Power:
2       pushq   %rbp
3       movq    %rsp, %rbp
4       movsd   %xmm0, -24(%rbp)
5       movsd   %xmm1, -32(%rbp)
6       movsd   .LC0(%rip), %xmm0
7       movsd   %xmm0, -8(%rbp)
8       movl    $1, -12(%rbp)
9       jmp     .L2
10      .L3:
11      movsd   -8(%rbp), %xmm0
12      mulsd   -24(%rbp), %xmm0
13      movsd   %xmm0, -8(%rbp)
14      addl    $1, -12(%rbp)
15      .L2:
16      pxor    %xmm1, %xmm1
17      cvtsi2sdl       -12(%rbp), %xmm1
18      movsd   -32(%rbp), %xmm0
19      comisd  %xmm1, %xmm0
```

Листинг x86-64 -Ofast gcc 13.2

```
1       Power:
2       movapd  %xmm0, %xmm3
3       movsd   .LC0(%rip), %xmm0
4       movl    $1, %eax
5       comisd  %xmm0, %xmm1
6       jb      .L1
7       .L4:
8       mulsd   %xmm3, %xmm0
9       addl    $1, %eax
10      pxor    %xmm2, %xmm2
11      cvtsi2sdl       %eax, %xmm2
12      comisd  %xmm2, %xmm1
13      jnb     .L4
14      .L1:
15      ret
16      .LC4:
17      .string "%f\n"
18      main:
19      subq    $8, %rsp
```

```
        jnb     .L3
        movsd   -8(%rbp), %xmm0
        movq    %xmm0, %rax
        movq    %rax, %xmm0
        popq    %rbp
        ret
        .LC5:
        .string "%f\n"
        main:
        pushq   %rbp
        movq    %rsp, %rbp
        subq    $48, %rsp
        movsd   .LC1(%rip), %xmm0
        movsd   %xmm0, -24(%rbp)
        movsd   .LC2(%rip), %xmm0
        movsd   %xmm0, -32(%rbp)
        pxor    %xmm0, %xmm0
        movsd   %xmm0, -8(%rbp)
        movsd   .LC0(%rip), %xmm0
        movsd   %xmm0, -16(%rbp)
        jmp     .L6
        .L7:
        movsd   -16(%rbp), %xmm1
        movsd   .LC0(%rip), %xmm0
        addsd   %xmm1, %xmm0
        movq    .LC4(%rip), %rax
        movapd  %xmm0, %xmm1
        movq    %rax, %xmm0
        call    Power
        movsd   %xmm0, -40(%rbp)
        movsd   -16(%rbp), %xmm0
        movq    -32(%rbp), %rax
        movapd  %xmm0, %xmm1
        movq    %rax, %xmm0
        call    Power
        mulsd   -40(%rbp), %xmm0
        divsd   -16(%rbp), %xmm0
        movsd   -8(%rbp), %xmm1
        addsd   %xmm1, %xmm0
        movsd   %xmm0, -8(%rbp)
        movsd   -16(%rbp), %xmm1
        movsd   .LC0(%rip), %xmm0
        addsd   %xmm1, %xmm0
        movsd   %xmm0, -16(%rbp)
        .L6:
        movsd   -24(%rbp), %xmm0
        comisd  -16(%rbp), %xmm0
        jnb     .L7
        movq    -8(%rbp), %rax
        movq    %rax, %xmm0
```

```
        pxor    %xmm8, %xmm8
        movsd   .LC0(%rip), %xmm7
        movq    .LC2(%rip), %xmm6
        movsd   .LC3(%rip), %xmm5
        movl    $150, %edx
        movapd  %xmm8, %xmm9
        movapd  %xmm7, %xmm4
        .L16:
        movapd  %xmm4, %xmm3
        movl    $1, %eax
        addsd   %xmm7, %xmm4
        comisd  %xmm9, %xmm3
        movapd  %xmm7, %xmm0
        jb      .L11
        .L13:
        addl    $1, %eax
        pxor    %xmm1, %xmm1
        movapd  %xmm0, %xmm2
        cvtsi2sdl       %eax, %xmm1
        xorpd   %xmm6, %xmm2
        movapd  %xmm2, %xmm0
        comisd  %xmm1, %xmm4
        jnb     .L13
        comisd  %xmm7, %xmm3
        movl    $1, %eax
        movapd  %xmm7, %xmm1
        jb      .L11
        .L15:
        mulsd   %xmm5, %xmm1
        addl    $1, %eax
        pxor    %xmm0, %xmm0
        cvtsi2sdl       %eax, %xmm0
        comisd  %xmm0, %xmm3
        jnb     .L15
        movapd  %xmm2, %xmm0
        mulsd   %xmm1, %xmm0
        .L11:
        divsd   %xmm3, %xmm0
        addsd   %xmm0, %xmm8
        subl    $1, %edx
        jne     .L16
        movapd  %xmm8, %xmm0
        movl    $.LC4, %edi
        movl    $1, %eax
        call    printf
        xorl    %eax, %eax
        addq    $8, %rsp
        ret
        .LC0:
        .long   0
```

```
70      movl    $.LC5, %edi
71      movl    $1, %eax
72      call    printf
73      movl    $0, %eax
74      leave
75      ret
76      .LC0:
77      .long   0
78      .long   1072693248
79      .LC1:
80      .long   0
81      .long   1080213504
82      .LC2:
83      .long   0
84      .long   1071644672
85      .LC4:
86      .long   0
87      .long   -1074790400
```

```
70      .long   1072693248
71      .LC2:
72      .long   0
73      .long   -2147483648
74      .long   0
75      .long   0
76      .LC3:
77      .long   0
78      .long   1071644672
```

ЗАКЛЮЧЕНИЕ

В результате проделанной работы я научилась сопоставлять команды языка Си с машинными командами, рассмотрела, как оптимизационные преобразования влияют на ассемблерный листинг, подробно изучила ассемблерные листинги архитектур x86 и x86-64.