МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

НОВОСИБИРСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Факультет информационных технологий

Кафедра параллельных вычислений

ОТЧЕТ

О ВЫПОЛНЕНИИ ЛАБОРАТОРНОЙ РАБОТЫ

«ВВЕДЕНИЕ В АРХИТЕКТУРУ x86/x86-64»

студента 2 курса, 23209 группы
Инокова Семёна Шухратовича

Направление 09.03.01 – «Информатика и вычислительная техника»

Преподаватель:
А. Ю. Кудинов

Новосибирск 2024

СОДЕРЖАНИЕ

ЦЕЛЬ.....3
ЗАДАНИЕ.....3
ОПИСАНИЕ РАБОТЫ.....4
ЗАКЛЮЧЕНИЕ.....5
Приложение 1. *Код программы для работы*.....6
Приложение 2. *Ассемблерный листинг x86 с оптимизацией O0*.....6
Приложение 3. *Ассемблерный листинг x86 с оптимизацией O3*.....9
Приложение 4. *Ассемблерный листинг x86-64 с оптимизацией O0* 12

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:

- набор регистров,
- основные арифметико-логические команды,
- способы адресации памяти,
- способы передачи управления,
- работу со стеком,
- вызов подпрограмм,
- передачу параметров в подпрограммы и возврат результатов,
- работу с арифметическим сопроцессором,
- работу с векторными расширениями.

2. Для программы на языке Си (из лабораторной работы 1) сгенерировать ассемблерные листинги для архитектуры x86 и архитектуры x86-64, используя различные уровни комплексной оптимизации.

3. Проанализировать полученные листинги и сделать следующее.

- Сопоставьте команды языка Си с машинными командами.
- Определить размещение переменных языка Си в программах на ассемблере (в каких регистрах, в каких ячейках памяти).
- Описать и объяснить оптимизационные преобразования, выполненные компилятором.
- Продемонстрировать использование ключевых особенностей архитектур x86 и x86-64 на конкретных участках ассемблерного кода.
- Сравнить различия в программах для архитектуры x86 и архитектуры x86-64.

ОПИСАНИЕ РАБОТЫ

1. Код на языке Си был скомпилирован в несколько листингов на архитектурах x86 и x86-64 с различными уровнями оптимизации.
2. Результаты компиляции были проанализированы и сравнены.

ЗАКЛЮЧЕНИЕ

В ходе данной работы я познакомился с программными архитектурами x86 и x86-64, научился читать и анализировать ассемблерные листинги. Для оптимизации O3 были сделаны следующие выводы:

- *Тело функции piCalculation встраивается в main.*
- *Для хранения локальных переменных предпочтительно используются регистры (edi, esi, ebx, ecx).*
- *Используются более оптимизированные операции над числами, убираются лишние шаги в вычислениях.*

Приложение 1. *Код программы для работы*

```c
#include <stdio.h>
#include <stdlib.h>
#include <stdbool.h>
#include <time.h>
#include <math.h>

double piCalculation(long long n){
    double pi = 4;

    for (long long i = 1; i <= n; ++i)
    {
        double x = 4. / (2*i + 1);
        if (i % 2)
        {
            pi -= x;
        }
        else
        {
            pi += x;
        }
    }

    return pi;
}

int main() {
    long long n = 200;

    double pi = piCalculation(n);
    printf("Pi number: %.10lf\n", pi);

    return 0;
}
```

Приложение 2. *Ассемблерный листинг x86 с оптимизацией O0*

```asm
piCalculation:
    push    ebp
    mov     ebp, esp
    sub     esp, 48
    mov     eax, DWORD PTR [ebp+8]
    mov     edx, DWORD PTR [ebp+12]
    mov     DWORD PTR [ebp-40], eax
    mov     DWORD PTR [ebp-36], edx
    fld     QWORD PTR .LC0
    fstp    QWORD PTR [ebp-8]
    mov     DWORD PTR [ebp-16], 1
    mov     DWORD PTR [ebp-12], 0
    jmp     .L2
.L5:
    mov     eax, DWORD PTR [ebp-16]
    mov     edx, DWORD PTR [ebp-12]
```

```
    add     eax, eax
    adc     edx, edx
    add     eax, 1
    adc     edx, 0
    movd    xmm0, eax
    movd    xmm1, edx
    punpckldq       xmm0, xmm1
    movq    QWORD PTR [ebp-48], xmm0
    fild    QWORD PTR [ebp-48]
    fld     QWORD PTR .LC0
    fdivrp  st(1), st
    fstp    QWORD PTR [ebp-24]
    mov     eax, DWORD PTR [ebp-16]
    mov     edx, DWORD PTR [ebp-12]
    and     eax, 1
    mov     edx, 0
    mov     ecx, eax
    or      ecx, edx
    je      .L3
    fld     QWORD PTR [ebp-8]
    fsub    QWORD PTR [ebp-24]
    fstp    QWORD PTR [ebp-8]
    jmp     .L4
.L3:
    fld     QWORD PTR [ebp-8]
    fadd    QWORD PTR [ebp-24]
    fstp    QWORD PTR [ebp-8]
.L4:
    add     DWORD PTR [ebp-16], 1
    adc     DWORD PTR [ebp-12], 0
.L2:
    mov     eax, DWORD PTR [ebp-16]
    mov     edx, DWORD PTR [ebp-12]
    mov     ecx, DWORD PTR [ebp-36]
    cmp     DWORD PTR [ebp-40], eax
    sbb     ecx, edx
    jge     .L5
    fld     QWORD PTR [ebp-8]
    leave
    ret
.LC2:
    .string "Pi number: %.10lf\n"
main:
    lea     ecx, [esp+4]
    and     esp, -16
    push    DWORD PTR [ecx-4]
    push    ebp
    mov     ebp, esp
    push    ecx
    sub     esp, 20
    mov     DWORD PTR [ebp-16], 200
```

```
    mov     DWORD PTR [ebp-12], 0
    push    DWORD PTR [ebp-12]
    push    DWORD PTR [ebp-16]
    call    piCalculation
    add     esp, 8
    fstp    QWORD PTR [ebp-24]
    sub     esp, 4
    push    DWORD PTR [ebp-20]
    push    DWORD PTR [ebp-24]
    push    OFFSET FLAT:.LC2
    call    printf
    add     esp, 16
    mov     eax, 0
    mov     ecx, DWORD PTR [ebp-4]
    leave
    lea     esp, [ecx-4]
    ret
.LC0:
    .long   0
    .long   1074790400
```

Приложение 3. *Ассемблерный листинг x86 с оптимизацией O3*

```
piCalculation:
    push    edi
    xor     eax, eax
    push    esi
    push    ebx
    sub     esp, 24
    mov     ecx, DWORD PTR [esp+40]
    mov     ebx, DWORD PTR [esp+44]
    cmp     eax, ecx
    sbb     eax, ebx
    jge     .L6
    add     ecx, 1
    mov     esi, 3
    mov     eax, 1
    fld     DWORD PTR .LC0
    adc     ebx, 0
    mov     DWORD PTR [esp+8], ecx
    xor     edi, edi
    xor     edx, edx
    mov     DWORD PTR [esp+12], ebx
    jmp     .L5
.L10:
    fsubp   st(1), st
.L4:
    add     eax, 1
    mov     ecx, DWORD PTR [esp+8]
    mov     ebx, DWORD PTR [esp+12]
    adc     edx, 0
    add     esi, 2
    adc     edi, 0
    xor     ecx, eax
    xor     ebx, edx
    or      ecx, ebx
    je      .L1
.L5:
    mov     ecx, eax
    movd    xmm0, esi
    movd    xmm1, edi
    xor     ebx, ebx
    and     ecx, 1
    mov     DWORD PTR [esp+4], ebx
    mov     ebx, DWORD PTR [esp+4]
    punpckldq       xmm0, xmm1
    mov     DWORD PTR [esp], ecx
    mov     ecx, DWORD PTR [esp]
    movq    QWORD PTR [esp+16], xmm0
    fild    QWORD PTR [esp+16]
    fdivr   DWORD PTR .LC0
```

```
    or      ecx, ebx
    jne     .L10
    faddp   st(1), st
    jmp     .L4
.L6:
    fld     DWORD PTR .LC0
.L1:
    add     esp, 24
    pop     ebx
    pop     esi
    pop     edi
    ret
.LC2:
    .string "Pi number: %.10lf\n"
main:
    lea     ecx, [esp+4]
    and     esp, -16
    mov     eax, 1
    xor     edx, edx
    push    DWORD PTR [ecx-4]
    push    ebp
    mov     ebp, esp
    push    edi
    push    esi
    push    ebx
    xor     ebx, ebx
    push    ecx
    mov     ecx, 3
    sub     esp, 24
    fld     DWORD PTR .LC0
.L15:
    mov     esi, eax
    movd    xmm0, ecx
    movd    xmm1, ebx
    xor     edi, edi
    and     esi, 1
    mov     DWORD PTR [ebp-36], edi
    mov     edi, DWORD PTR [ebp-36]
    punpckldq       xmm0, xmm1
    mov     DWORD PTR [ebp-40], esi
    mov     esi, DWORD PTR [ebp-40]
    movq    QWORD PTR [ebp-32], xmm0
    fild    QWORD PTR [ebp-32]
    fdivr   DWORD PTR .LC0
    or      esi, edi
    je      .L12
    add     eax, 1
    fsubp   st(1), st
    adc     edx, 0
    add     ecx, 2
    adc     ebx, 0
```

```
    jmp     .L15
.L12:
    add     eax, 1
    faddp   st(1), st
    adc     edx, 0
    mov     esi, eax
    add     ecx, 2
    adc     ebx, 0
    xor     esi, 201
    or      esi, edx
    jne     .L15
    sub     esp, 12
    fstp    QWORD PTR [esp]
    push    OFFSET FLAT:.LC2
    call    printf
    add     esp, 16
    lea     esp, [ebp-16]
    xor     eax, eax
    pop     ecx
    pop     ebx
    pop     esi
    pop     edi
    pop     ebp
    lea     esp, [ecx-4]
    ret
.LC0:
    .long   1082130432
```

Приложение 4. *Ассемблерный листинг x86-64 с оптимизацией O0*

```
piCalculation:
    push    rbp
    mov     rbp, rsp
    mov     QWORD PTR [rbp-40], rdi
    movsd   xmm0, QWORD PTR .LC0[rip]
    movsd   QWORD PTR [rbp-8], xmm0
    mov     QWORD PTR [rbp-16], 1
    jmp     .L2
.L5:
    mov     rax, QWORD PTR [rbp-16]
    add     rax, rax
    add     rax, 1
    pxor    xmm1, xmm1
    cvtsi2sd        xmm1, rax
    movsd   xmm0, QWORD PTR .LC0[rip]
    divsd   xmm0, xmm1
    movsd   QWORD PTR [rbp-24], xmm0
    mov     rax, QWORD PTR [rbp-16]
    and     eax, 1
    test    rax, rax
    je      .L3
    movsd   xmm0, QWORD PTR [rbp-8]
    subsd   xmm0, QWORD PTR [rbp-24]
    movsd   QWORD PTR [rbp-8], xmm0
    jmp     .L4
.L3:
    movsd   xmm0, QWORD PTR [rbp-8]
    addsd   xmm0, QWORD PTR [rbp-24]
    movsd   QWORD PTR [rbp-8], xmm0
.L4:
    add     QWORD PTR [rbp-16], 1
.L2:
    mov     rax, QWORD PTR [rbp-16]
    cmp     rax, QWORD PTR [rbp-40]
    jle     .L5
    movsd   xmm0, QWORD PTR [rbp-8]
    pop     rbp
    ret
.LC1:
    .string "Pi number: %.10lf\n"
main:
    push    rbp
    mov     rbp, rsp
    sub     rsp, 16
    mov     QWORD PTR [rbp-8], 200
    mov     rax, QWORD PTR [rbp-8]
    mov     rdi, rax
    call    piCalculation
```

```
    movq    rax, xmm0
    mov     QWORD PTR [rbp-16], rax
    mov     rax, QWORD PTR [rbp-16]
    movq    xmm0, rax
    mov     edi, OFFSET FLAT:.LC1
    mov     eax, 1
    call    printf
    mov     eax, 0
    leave
    ret
.LC0:
    .long   0
    .long   1074790400
```