## «ВВЕДЕНИЕ В АРХИТЕКТУРУ x86/x86-64»

## Отчёт по лабораторной работе №3

## Выполнил: Ланин Д.М. ФИТ НГУ 2 курс

## Преподаватель: Артюхов А. А.

## Цель лабораторной работы:

Ознакомиться с программной архитектурой x86/x86-64, а также провести 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.

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

* Сопоставить 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.

**Для вычислений применяется алгоритм вычисления числа Пи методом Монте-Карло.**

*Далее подробно об алгоритме* : cначала в квадрат с центром в начале координат и со стороной два вписывается круг с единичным радиусом. Затем в этом квадрате случайным образом с равномерным распределением генерируются N точек. Точка может попасть в окружность или нет (условие попадания x2 + y 2 ≤ 1). Далее определяется число M точек, попавших в круг. При достаточно большом числе бросков N, по значениям M и N вычисляется число Пи: Pi = 4 \* M / N.

**Описание работы**

1.Ознакомление с теоретической частью.

2.Для программы на языке Си (из лабораторной работы 1) сгенерируем

ассемблерные листинги для архитектуры x86 и x86-64,

используя различные уровни комплексной оптимизации (O0, O1, O3).

3. Проанализировать полученные листинги.

## Результат:

Мы научились анализировать ассемблерный код, при этом заметили, что переменные, в зависимости от применённого уровня оптимизации при компиляции, хранились либо в стеке, либо в регистрах общего назначения, преимущественно регистрах rax, eax, xmm0. Основные различия архитектур заключались в размерах передаваемых данных: если же в x86 использовались 4х-байтные значения, то в x86-64 преимущественно 8-байтные значения.

***Ассемблерный листинг для архитектуры x86-64 -O0***

|  |  |
| --- | --- |
| .LC4:          .string "%lf\n"  .LC5:          .string "Total time: %lf seconds\n"  main:          pushq   %rbp          movq    %rsp, %rbp          subq    $96, %rsp          movl    $0, %edi          call    time          movl    %eax, %edi          call    srand          movq    $100000000, -24(%rbp)          movq    $0, -8(%rbp)          call    clock          movq    %rax, -32(%rbp)          movl    $0, -12(%rbp)          jmp     .L2  .L5:          call    rand          pxor    %xmm0, %xmm0          cvtsi2sdl       %eax, %xmm0          movsd   .LC0(%rip), %xmm1          divsd   %xmm1, %xmm0          movsd   %xmm0, -88(%rbp)          fldl    -88(%rbp)          fstpt   -64(%rbp)          call    rand          pxor    %xmm0, %xmm0          cvtsi2sdl       %eax, %xmm0          movsd   .LC0(%rip), %xmm1          divsd   %xmm1, %xmm0          movsd   %xmm0, -88(%rbp)          fldl    -88(%rbp)          fstpt   -80(%rbp)          fldt    -64(%rbp)          fld     %st(0)          fmulp   %st, %st(1)          fldt    -80(%rbp)          fmul    %st(0), %st          faddp   %st, %st(1)          fld1          fcomip  %st(1), %st          fstp    %st(0)          jb      .L3          addq    $1, -8(%rbp)  .L3:          addl    $1, -12(%rbp)  .L2:          movl    -12(%rbp), %eax          cltq          cmpq    -24(%rbp), %rax          jb      .L5          call    clock          movq    %rax, -40(%rbp)          movq    -40(%rbp), %rax          subq    -32(%rbp), %rax          pxor    %xmm0, %xmm0          cvtsi2sdq       %rax, %xmm0          movsd   .LC2(%rip), %xmm1          divsd   %xmm1, %xmm0          movsd   %xmm0, -48(%rbp)          movq    -8(%rbp), %rax          testq   %rax, %rax          js      .L6          pxor    %xmm0, %xmm0          cvtsi2sdq       %rax, %xmm0          jmp     .L7  .L6:          movq    %rax, %rdx          shrq    %rdx          andl    $1, %eax          orq     %rax, %rdx          pxor    %xmm0, %xmm0          cvtsi2sdq       %rdx, %xmm0          addsd   %xmm0, %xmm0  .L7:          movsd   .LC3(%rip), %xmm1          mulsd   %xmm0, %xmm1          movq    -24(%rbp), %rax          testq   %rax, %rax          js      .L8          pxor    %xmm0, %xmm0          cvtsi2sdq       %rax, %xmm0          jmp     .L9  .L8:          movq    %rax, %rdx          shrq    %rdx          andl    $1, %eax          orq     %rax, %rdx          pxor    %xmm0, %xmm0          cvtsi2sdq       %rdx, %xmm0          addsd   %xmm0, %xmm0  .L9:          divsd   %xmm0, %xmm1          movq    %xmm1, %rax          movq    %rax, %xmm0          movl    $.LC4, %edi          movl    $1, %eax          call    printf          movq    -48(%rbp), %rax          movq    %rax, %xmm0          movl    $.LC5, %edi          movl    $1, %eax          call    printf          movl    $0, %eax          leave          ret  .LC0:          .long   -4194304          .long   1105199103  .LC2:          .long   0          .long   1093567616  .LC3:          .long   0          .long   1074790400 | Добавление rbp в стек  Rbp = rsp  Выделение памяти в размере 96 байт на локал переменные  Edi = 0  Вызов time  edi = eax  Вызор srand  индекс смещён на количество байт, выделенное на данную переменную  Вызов clock  …  …  Безусловный переход в блок L2  Вызов rand  Xor xmm0 xmm0 (зануление)  Преобразование в double из DWord  Xmm1 \*= LC0(rip – регистр указатель инструкции, смещение на след. регистр)  Xmm0 /= xmm1  rbp-88 = xmm0  Загрузить вещественное число в rbp-88  Вытащить число со стека в rbp-88  Вызвать rand  Xor xmm0 xmm0  Преобразование в double  Xmm1 = LC0  Xmm0 /= xmm1  rbp-88 = xmm0  Загрузить вещественное число в rbp-88  Записать и вытолкнуть вещественное число  Загрузить вещественное число в rpb-64  Загрузить вещественное число в st(0) стэк сопроцессора  Умножение вещественного числа с выталкиванием  Загрузить вещественное число в rbp-80  Умножение вещественного числа с выталкиванием  Сложение с вещественным числом с выталкиванием  Загрузить число +1.0  Вещественное сравнение с установкой EFLAGS и выталкиванием Записать и вытолкнуть вещественное число  Условный переход  rbp-8 += 1  rbp-12 += 1  …  Преобразование DWord в Qword  проверка  Условный переход  Вызов clock  rbp-40 = rax  Rax = rbp-40  Rax -= rbp-32  Упакованное логическое исключающее ИЛИ(MMX)  Преобразование скалярного DWord в Double  Перемещение строки двойных слов  Скалярное деление Double  rbp-48 = xmm0  Rax = rbp-8  Rax == rax  Условное перемещение в .L6  Упакованный xor  Преобразование DWord в Double  Безусловный перезод в L7  Rdx = rax  Логический сдвиг вправо. Rdx >>= 1;  Eax &= 1  Rdx |= rax  Упакованный xor  Преобразование скалярного DWord в Double  Скалярное сложение Double  Xmm0 += xmm0  Перемещение строки двойных слов(rip -регистр указателя инструкции, смещение на след. регистр)  Умножение Double  Rax = rbp-24  Rax == rax  Jump L8  xor  Преобразование из DWord в Double  Переход в L9  Rdx = rax  Rdx >>= 1  Eax &= 1  Rdx |= rax  Упакованный xor  Преобразование из DWord в Double  Сложение Double  Деление Double  Move Qword rax = xmm1  Rax = xmm1  Xmm0 = rax  Edi = LC4 (“lf\n”)  Eax = 1  Вызов printf  Rax = rbp-48  Xmm0 = rax  Edi = "Total time: %lf seconds\n"  Eax = 1  Вызов printf  Eax = 0  Высокоуровневый выход из процедуры  Возврат из подпрограммы |

***Ассемблерный листинг для архитектуры x86 -m32 -O0***

.LC4:

        .string "%lf\n"

.LC5:

        .string "Total time: %lf seconds\n"

main:

        leal    4(%esp), %ecx

        andl    $-16, %esp

        pushl   -4(%ecx)

        pushl   %ebp

        movl    %esp, %ebp

        pushl   %ecx

        subl    $84, %esp

        subl    $12, %esp

        pushl   $0

        call    time

        addl    $16, %esp

        subl    $12, %esp

        pushl   %eax

        call    srand

        addl    $16, %esp

        movl    $100000000, -20(%ebp)

        movl    $0, -12(%ebp)

        call    clock

        movl    %eax, -24(%ebp)

        movl    $0, -16(%ebp)

        jmp     .L2

.L5:

        call    rand

        movl    %eax, -84(%ebp)

        fildl   -84(%ebp)

        fldl    .LC0

        fdivrp  %st, %st(1)

        fstpt   -52(%ebp)

        call    rand

        movl    %eax, -84(%ebp)

        fildl   -84(%ebp)

        fldl    .LC0

        fdivrp  %st, %st(1)

        fstpt   -64(%ebp)

        fldt    -52(%ebp)

        fld     %st(0)

        fmulp   %st, %st(1)

        fldt    -64(%ebp)

        fmul    %st(0), %st

        faddp   %st, %st(1)

        fld1

        fcomip  %st(1), %st

        fstp    %st(0)

        jb      .L3

        addl    $1, -12(%ebp)

.L3:

        addl    $1, -16(%ebp)

.L2:

        movl    -16(%ebp), %eax

        cmpl    -20(%ebp), %eax

        jb      .L5

        call    clock

        movl    %eax, -28(%ebp)

        movl    -28(%ebp), %eax

        subl    -24(%ebp), %eax

        movl    %eax, -84(%ebp)

        fildl   -84(%ebp)

        fldl    .LC2

        fdivrp  %st, %st(1)

        fstpl   -40(%ebp)

        movd    -12(%ebp), %xmm0

        movq    %xmm0, -80(%ebp)

        fildq   -80(%ebp)

        fldl    .LC3

        fmulp   %st, %st(1)

        movd    -20(%ebp), %xmm0

        movq    %xmm0, -80(%ebp)

        fildq   -80(%ebp)

        fdivrp  %st, %st(1)

        subl    $4, %esp

        leal    -8(%esp), %esp

        fstpl   (%esp)

        pushl   $.LC4

        call    printf

        addl    $16, %esp

        subl    $4, %esp

        pushl   -36(%ebp)

        pushl   -40(%ebp)

        pushl   $.LC5

        call    printf

        addl    $16, %esp

        movl    $0, %eax

        movl    -4(%ebp), %ecx

        leave

        leal    -4(%ecx), %esp

        ret

.LC0:

        .long   -4194304

        .long   1105199103

.LC2:

        .long   0

        .long   1093567616

.LC3:

        .long   0

        .long   1074790400

***Ассемблерный листинг для архитектуры x86-64 -O1***

.LC4:

        .string "%lf\n"

.LC6:

        .string "Total time: %lf seconds\n"

main:

        pushq   %r12

        pushq   %rbp

        pushq   %rbx

        subq    $32, %rsp

        movl    $0, %edi

        call    time

        movl    %eax, %edi

        call    srand

        call    clock

        movq    %rax, %r12

        movl    $100000000, %ebx

        movl    $0, %ebp

.L4:

        call    rand

        pxor    %xmm0, %xmm0

        cvtsi2sdl       %eax, %xmm0

        divsd   .LC0(%rip), %xmm0

        movsd   %xmm0, (%rsp)

        fldl    (%rsp)

        fstpt   (%rsp)

        call    rand

        pxor    %xmm0, %xmm0

        cvtsi2sdl       %eax, %xmm0

        divsd   .LC0(%rip), %xmm0

        movsd   %xmm0, 24(%rsp)

        fldl    24(%rsp)

        fldt    (%rsp)

        fmul    %st(0), %st

        fxch    %st(1)

        fmul    %st(0), %st

        faddp   %st, %st(1)

        fld1

        fcomip  %st(1), %st

        fstp    %st(0)

        sbbq    $-1, %rbp

        subq    $1, %rbx

        jne     .L4

        call    clock

        movq    %rax, %rbx

        testq   %rbp, %rbp

        js      .L5

        pxor    %xmm0, %xmm0

        cvtsi2sdq       %rbp, %xmm0

.L6:

        mulsd   .LC2(%rip), %xmm0

        divsd   .LC3(%rip), %xmm0

        movl    $.LC4, %edi

        movl    $1, %eax

        call    printf

        subq    %r12, %rbx

        pxor    %xmm0, %xmm0

        cvtsi2sdq       %rbx, %xmm0

        divsd   .LC5(%rip), %xmm0

        movl    $.LC6, %edi

        movl    $1, %eax

        call    printf

        movl    $0, %eax

        addq    $32, %rsp

        popq    %rbx

        popq    %rbp

        popq    %r12

        ret

.L5:

        movq    %rbp, %rax

        shrq    %rax

        andl    $1, %ebp

        orq     %rbp, %rax

        pxor    %xmm0, %xmm0

        cvtsi2sdq       %rax, %xmm0

        addsd   %xmm0, %xmm0

        jmp     .L6

.LC0:

        .long   -4194304

        .long   1105199103

.LC2:

        .long   0

        .long   1074790400

.LC3:

        .long   0

        .long   1100470148

.LC5:

        .long   0

        .long   1093567616

***Ассемблерный листинг для архитектуры x86-64 -O3***

.LC4:

        .string "%lf\n"

.LC6:

        .string "Total time: %lf seconds\n"

main:

        pushq   %r12

        xorl    %edi, %edi

        pushq   %rbp

        xorl    %ebp, %ebp

        pushq   %rbx

        movl    $100000000, %ebx

        subq    $32, %rsp

        call    time

        movl    %eax, %edi

        call    srand

        call    clock

        movq    %rax, %r12

.L4:

        call    rand

        pxor    %xmm0, %xmm0

        cvtsi2sdl       %eax, %xmm0

        divsd   .LC0(%rip), %xmm0

        movsd   %xmm0, 8(%rsp)

        fldl    8(%rsp)

        fstpt   16(%rsp)

        call    rand

        pxor    %xmm0, %xmm0

        cvtsi2sdl       %eax, %xmm0

        divsd   .LC0(%rip), %xmm0

        movsd   %xmm0, 8(%rsp)

        fldl    8(%rsp)

        fldt    16(%rsp)

        fmul    %st(0), %st

        fxch    %st(1)

        fmul    %st(0), %st

        faddp   %st, %st(1)

        fld1

        fcomip  %st(1), %st

        fstp    %st(0)

        sbbq    $-1, %rbp

        subl    $1, %ebx

        jne     .L4

        call    clock

        movq    %rax, %rbx

        testq   %rbp, %rbp

        js      .L5

        pxor    %xmm0, %xmm0

        cvtsi2sdq       %rbp, %xmm0

.L6:

        mulsd   .LC2(%rip), %xmm0

        movl    $.LC4, %edi

        movl    $1, %eax

        subq    %r12, %rbx

        divsd   .LC3(%rip), %xmm0

        call    printf

        pxor    %xmm0, %xmm0

        movl    $.LC6, %edi

        movl    $1, %eax

        cvtsi2sdq       %rbx, %xmm0

        divsd   .LC5(%rip), %xmm0

        call    printf

        addq    $32, %rsp

        xorl    %eax, %eax

        popq    %rbx

        popq    %rbp

        popq    %r12

        ret

.L5:

        movq    %rbp, %rax

        andl    $1, %ebp

        pxor    %xmm0, %xmm0

        shrq    %rax

        orq     %rbp, %rax

        cvtsi2sdq       %rax, %xmm0

        addsd   %xmm0, %xmm0

        jmp     .L6

.LC0:

        .long   -4194304

        .long   1105199103

.LC2:

        .long   0

        .long   1074790400

.LC3:

        .long   0

        .long   1100470148

.LC5:

        .long   0

        .long   1093567616