Учреждение образования

«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ»

Кафедра информатики

Отчет по лабораторной работе №1

Основные конструкции. Написание простейшей программы

Студент: гр. 753502

Саттарова Александра Сергеевна

Руководитель: ассистент Шиманский В.В.

Минск 2018

СОДЕРЖАНИЕ

1. Введение
2. Постановка задачи
3. Программная реализация
4. Выводы

Литература

Приложение 1. Текст программы

1. Введение
   1. Язык ассемблера — система обозначений, используемая для представления в удобочитаемой форме программ, записанных в машинном коде. Язык ассемблера позволяет программисту пользоваться алфавитными мнемоническими кодами операций, по своему усмотрению присваивать символические имена регистрам ЭВМ и памяти, а также задавать удобные для себя схемы адресации (например, индексную или косвенную). Кроме того, он позволяет использовать различные системы счисления (например, десятичную или шестнадцатеричную) для представления числовых констант и даёт возможность помечать строки программы метками с символическими именами с тем, чтобы к ним можно было обращаться (по именам, а не по адресам) из других частей программы (например, для передачи управления).  
       Перевод программы на языке ассемблера в исполнимый машинный код (вычисление выражений, раскрытие макрокоманд, замена мнемоник собственно машинными кодами и символьных адресов на абсолютные или относительные адреса) производится *ассемблером* — программой-транслятором, которая и дала языку ассемблера его название.
   2. Цель лабораторной работы — освоить основы синтаксиса компилятора NASM в операционной системе Linux и особенности написания программы в размерности 64-бит. NASM (*Netwide Assembler*) — это свободный (LGPL и лицензия BSD) ассемблер для архитектуры Intel x86, использующийся для написания 16-, 32- и 64-разрядных программ. В NASM используется Intel-синтаксис записи инструкций.   
       Задачи лабораторной работы также освоить основные операции над регистрами (ADD, SUB, MUL, DIV, CMP) и работу с метками.
2. Постановка задачи
   1. Условие задания

if (a\*c + b\*d = a\*d + b\*c)

if (a > b)

result = a^2

else if (a > c)

result = (c && b) + min(a, b, c)/(a^2 - c)

else result = a - (b OR c)

else result = a/d

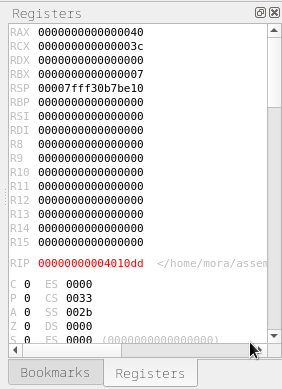
2.2. Текст задания

В каждом из заданий переменные a, b, c, d определяются в сегменте данных и имеют размерность слово. Необходимо выполнить над ними заданные арифметические и логические операции, а результат поместить в регистр AX.

При выполнении умножения считаем, что результат вмещается в слово. При выполнении деления считаем, что оно целочисленное, и делитель не равен нулю.

1. Программная реализация
   1. Значения переменных устанавливаются при объявлении сегмента данных. Программа разбита при помощи меток на несколько логических частей, каждая из которых выполняет определенную ветку условия.
   2. Результат можно видеть в отладчике в регистре AX
   3. Примеры:
      1. Тест для ветки «Результат = a ^ 2»

а=8 b=7 c=4 d=4



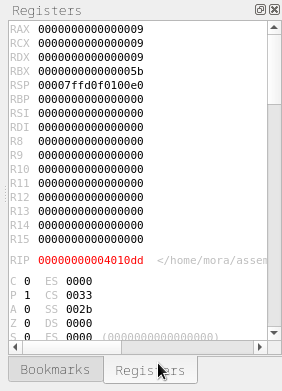
Ответ: 64 (0x40)

3.3.2

Тест для ветки

«Результат = (c && b) + min(a, b, c)/(a^2 — c)»

а=10 b=11 c=9 d=9

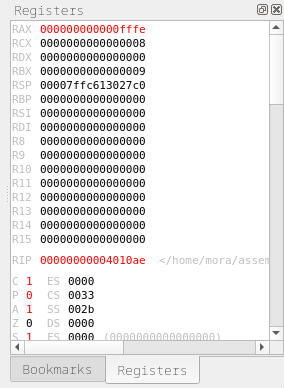


Ответ: 9 (0x9)

3.3.3

Тест для ветки «Результат = a - (b OR c)»

а=7 b=9 c=8 d=8

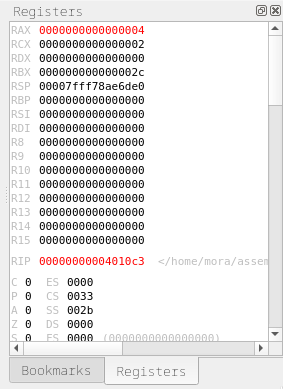


Ответ: -2 (0xfffe)

3.3.4.

Тест для ветки «Результат = a/d»

а=8 b=6 c=4 d=2



Ответ: 4 (0x4)

4. Выводы

В ходе выполнения лабораторной работы были освоены основы синтаксиса компилятора NASM в операционной системе Arch Linux c версией ядра 4.18.9 и особенности написания программы в размерности 64-бит. Также были разобраны основные операции над регистрами и работа с метками. Среди них операции ADD (сложение значения одного регистра со значением второго), SUB (вычитание значения одного регистра из значения второго), MUL (умножение значения одного регистра на значение второго), DIV (деление значения первого на значение второго регистра). Освоены операция сравнения CMP и связанные с ней флаги ZF, CF, а также инструкции (JE\JZ, JNE\JNZ, JG, JL).

Литература

1. Learn Assembly Language - <http://asmtutor.com/>

2. Столяров А. В. Программирование на языке ассемблера NASM для ОС Unix: Уч. пособие. - 2-е изд. - М.: МАКС Пресс, 2011. - 188 с.

Приложение 1. Текст программы

section .data

;result = (c && b) + min(a, b, c)/(a^2 -c)

;a: dw 10

;b: dw 11

;c: dw 9

;d: dw 9

;result = a^2

;a: dw 8

;b: dw 7

;c: dw 4

;d: dw 4

;result = a - (b OR c)

a: dw 7

b: dw 9

c: dw 8

d: dw 8

;result = a/d

;a: dw 8

;b: dw 6

;c: dw 4

;d: dw 2

section .text

global \_start

\_start:

mov ax, [a]

mov cx, [c]

mul cx ;ax = a\*c

mov bx, ax ;bx = a\*c

mov ax, [b]

mov cx, [d]

mul cx ;ax = b\*d

add bx, ax ;bx = a\*c + b\*d

push bx

mov ax, [a]

mul cx ;ax = a\*d

mov bx, ax ;DX = a\*d

mov ax, [b]

mov cx, [c]

mul cx ;ax=b\*c

add bx, ax ;DX = a\*d + b\*c

mov ax, [a]

mov cx, bx

pop bx

cmp bx, cx ;if (a\*c + b\*d == a\*d + b\*c)

jnz else1 ;if they are not equal, go to label else1

mov bx, [b]

cmp ax, bx

jng else2 ;if a < b, go to label else2

imul ax ;print a^2

jmp end

else2:

mov cx, [c]

cmp ax, cx

jng else3 ;if a < c, go to label else3

;print ((c AND b) + min (a, b, c)/(a^2-c))

call findmin

push ax

mov ax, [a]

mul ax

sub ax, cx ;ax = a^2-c

and cx, bx ;cx = c and b

mov bx, ax

pop ax ;findmin

div bx

add ax, cx

jmp end

else3:

or bx, cx

sub ax, bx ;a-(b or c)

jmp end

else1:

mov ax, [a]

mov cx, [d]

div cx ;a/d

jmp end

findmin:

cmp ax, bx

jg findmin.comp\_b\_c ;if a > b, go to label comp\_b\_c

cmp ax, cx

jg findmin.min\_c ;if a > c, go to min\_c

ret

.comp\_b\_c:

cmp bx, cx

jg findmin.min\_c ;if b > c go to min\_c

mov ax, bx

ret

.min\_c:

mov ax, cx

ret

end:

mov eax, 1

xor ebx, ebx

int 80h