Министерство образования и науки Украины

Харьковский национальный университет радиоэлектроники

Кафедра ЭВМ

ОТЧЕТ

ПО ЛАБОРАТОРНОЙ РАБОТЕ 8

по предмету

«Архитектура компьютеров»

на тему:

«Архитектура прикладного уровня CPU x86-64»

Вариант №12

Выполнил: Принял:

студент гр. КИУКИ-18-4 Голубничий Д.Ю.

Кравченко Н.С. Дзюбенко В.Ф.

Харьков 2020

1. Цель работы

Углубление и закрепление знаний архитектуры прикладного уровня 32-разрядных процессоров семейства x86-64; получение практических навыков по составлению программ на языке ассемблера процессоров x86-64, их отладке и выполнению.

2. Постановка задачи

Выполнение работы состоит в разработке, отладке и выполнении ассемблерных программ в среде разработки Visual C++.

| №№  варианта | Комбинация вариантов заданий | | |
| --- | --- | --- | --- |
| № задания из пп.**8.5.1** | № задания из пп.**8.5.2** | № задания из пп.**8.5.3** |
| 12 | 9 | 7 | 6 |

Таблица 1 – Схема вариантов заданий

**8.5.1 - Задание 9.**   
Вычислить 6 значений функции Y = (2500 \* х – 8) / (3 \* х^2 + 20) (х изменяется от 4 с шагом 3). Результат округлить до целого и разместить в памяти.

**8.5.2 - Задание 7.**   
Найти целое значение аргумента, при котором функция Y = (7^x) / (5 \* x^2) превысит 300.

**8.5.3 - Задание 6.**В памяти задан массив из 12 элементов. Сохранить в регистре ЕАХ среднее арифметическое этих элементов. Результат округлить до целого.

3. Теоретическая часть

Под регистровой моделью понимается описание набора регистров процессора, их разрядности, способа организации, методов доступа, основных особенностей и пр.

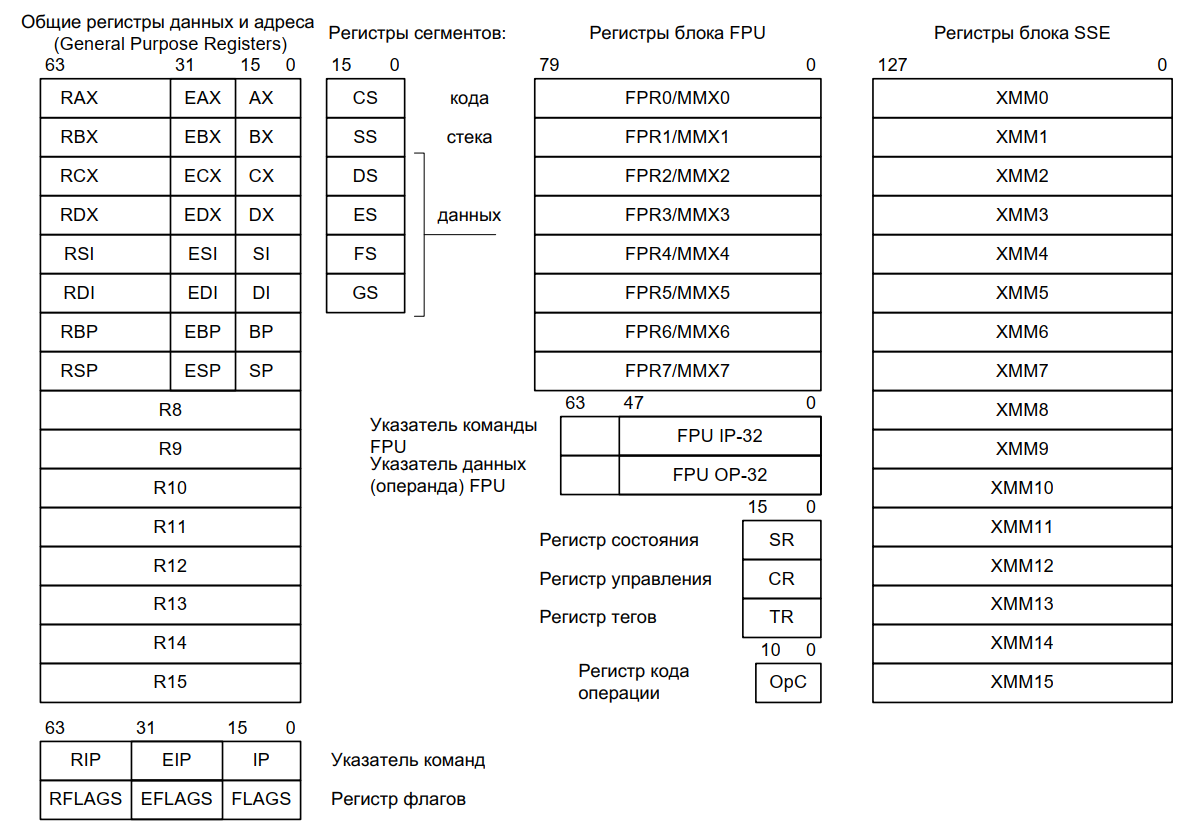


Рисунок 1 – Регистровая модель микропроцессоров

На рисунке 1 показаны регистры, доступные программисту при использовании машинных кодов (или языка ассемблера).

Наиболее универсальными являются регистры общего назначения (РОН), которые в 8086 обозначаются с использованием символов латинского алфавита A, B, C, D. Регистры имеют 16 разрядов, их младшая часть дополнительно обозначается символом L (Low), а старшая ­– H (High). Весь регистр обозначается дополнительным символом X. Таким образом, AX соответствует всем 16 разрядам первого регистра общего назначения, а доступ к двум его 8-разрядным половинам производится по именам AH, AL.

Для операций с объектами в памяти используются индексные регистры. Их имена являются сокращениями от SourceIndex (SI) и DestinationIndex (DI), т.е. «индекс источника» и «индекс приемника» соответственно.

Также в 8086 присутствуют указатели на области памяти стека и стекового кадра SPи BP, сводный регистр флагов (особых признаков результата последней операции) и сегментные регистры.

В систему команд языка ассемблера СPU х86-64 входят следующие группы команд:

* общего назначения;
* блока FPU;
* управления состоянием блоков FPU и SIMD;
* технологии MMX;
* расширений SSE, SSE2, SSE3, SSSE3, SSE4, Intel AVX;
* системного управления;
* режима IA-32e (64-битные команды);
* AESNI и PCLMULQDQ;
* VMX (virtual-machine extension);
* SMX (safer mode extension).

4. Экспериментально-практическая часть

**8.5.1 - Задание 9.**   
Вычислить 6 значений функции Y = (2500 \* х – 8) / (3 \* х^2 + 20) (х изменяется от 4 с шагом 3). Результат округлить до целого и разместить в памяти.

Теоретические расчеты:

*x = 4: 146.94*

*x = 7: 104.74*

*x = 10: 78.1*

*x = 13: 61.65*

*x = 16: 50.75*

*x = 19: 43.06*

Так как используется формат целочисленных данных, результат округлен до целого.

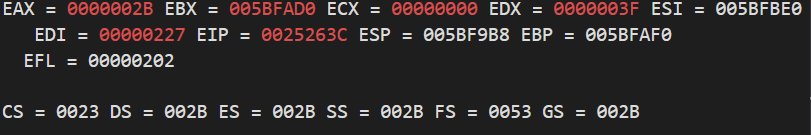


Рисунок 2 – Окно отладки «Registers»

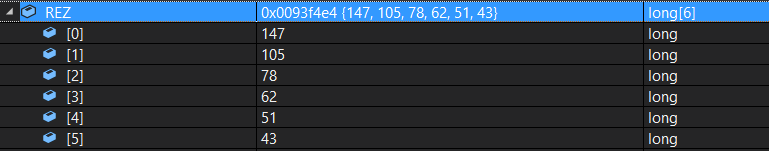


Рисунок 3 – Результат работы программы

Код программы:

long X = 4; // ячейка памяти для аргумента

long REZ[6]; // 6 ячеек памяти для результатов

\_asm

{

lea EBX, REZ ; загрузка адреса результатов в регистр EBX

mov ECX, 6 ; счетчик количества повторений цикла

m1 :

mov EAX, 3 ; EAX = 3

imul X ; EAX = 3 \* x

imul X ; EAX = 3 \* x ^ 2

add EAX, 20 ; EAX = 3 \* x ^ 2 + 20

mov EDI, EAX ; пересылка знаменателя в регистр EDI

mov EAX, 2500 ; EAX = 2500

imul X ; EAX = 2500 \* x

sub EAX, 8 ; EAX = 2500 \* x – 8

cdq ; расширение операнда - делимого в EAX - EDX

div EDI ; частное – EAX, остаток – EDX

shr EDI, 1 ; деление знаменателя(делителя) на 2

cmp EDI, EDX ; сравнение половины делителя с остатком

adc EAX, 0 ; добавление к частному заема от сравнения

mov dword ptr[EBX], EAX ; пересылка результата в память

add EBX, 4 ; увеличение адреса результатов

add X, 3 ; увеличение аргумента

loop m1 ; зацикливание по счетчику в ЕСХ

}

for (auto element : REZ)

std::cout << element << " ";

std::cout << std::endl;

**8.5.2 - Задание 7.**   
Найти целое значение аргумента, при котором функция Y = (7^x) / (5 \* x^2) превысит 300.

Теоретические расчеты:

*x = 5: 134*

*x = 6: 653,6*

Как можно проследить, функция *Y = (7^x) / (5 \* x^2)* превысит значение 300 при *x=6*.

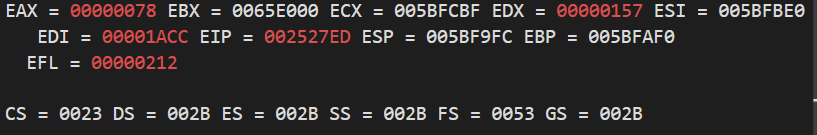


Рисунок 4 – Окно отладки «Registers»



Рисунок 5 – Результат работы программы

Код программы:

long X = 0; // ячейка памяти для аргумента

long P = 1; // ячейка для накопления 8n

\_asm

{

m1: inc X ; увеличение аргумента

mov EAX, 140 ; EAX = 5

imul X ; EAX = 5 \* x

imul X ; EAX = 5 \* x ^ 2

mov EDI, EAX ; пересылка знаменателя в регистр EDI

mov EAX, 7 ; EAX = 7

mul P ; умножение – 7 ^ n

mov P, EAX ; пересылка 7 ^ n в ячейку памяти Р

cdq

div EDI ; частное – EAX, остаток – EDX

cmp AL, 300 ; сравнение функции с 300

jc m1 ; переход, если функция меньше 300

}

std::cout << X << std::endl;

**8.5.3 - Задание 6.**В памяти задан массив из 12 элементов. Сохранить в регистре ЕАХ среднее арифметическое этих элементов. Результат округлить до целого.

Теоретические расчеты:

Входные данные: { 1,2,3,4,5,6,7,8,9,10,11,12 }

Так как используется формат целочисленных данных, результат округлен до целого.

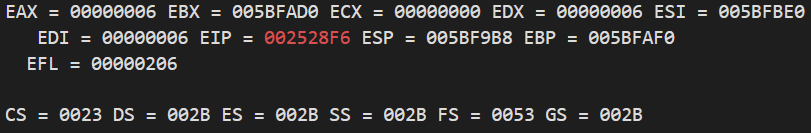


Рисунок 6 – Окно отладки «Registers»



Рисунок 7 – Результат работы программы

Код программы:

long const x\_count = 12;

long x[x\_count] = { 1,2,3,4,5,6,7,8,9,10,11,12 };

long S = 0;

long res;

\_asm

{

lea EBX, x ; начальный адрес массива – в ЕВХ

mov ECX, 12 ; счетчик повторений

cycle :

mov EAX, dword ptr[EBX] ;первый элемент – в ЕАХ

add S, EAX ; накопление суммы

mov EDI, EBX ; адрес элемента – в EDI

add EBX, 4 ; увеличение адреса

loop cycle ; зацикливание по счетчику

mov EAX, S

mov EDI, x\_count

cdq

div EDI

shr EDI, 1 ; деление знаменателя(делителя) на 2

cmp EDI, EDX ; сравнение половины делителя с остатком

adc EAX, 0 ; добавление к частному заема от сравнения

mov res, EAX

}

std::cout << res << std::endl;

5. Анализы и выводы по работе

В ходе выполнения лабораторной работы ознакомился с системой команд языка ассемблера СPU х86-64 и получил практические навыки по составлению ассемблерных вставок, встраиваемых в тело программы на языке C++ и отладке приложения.

Выполнение заданий проходило в среде разработки Visual C++, которая содержит удобный функционал режима отладки для контроля значений переменных и регистров.

В процессе работы были решены 3 задачи, в следствие чего получен опыт вычисления арифметических операций, округления чисел, вычисления среднего арифметического, использования команд условных переходов и сохранения результатов в регистр или память.

Результаты выполнение отличаются от теоретических вычислений из-за погрешности, вызванной использованием целочисленного формата данных (откидыванием дробной части или округлением).