# Лабораторная работа 8. Архитектура прикладного уровня CPU x86-64

## 8.1 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.

## 8.2 Самостоятельная работа студентов

Перед выполнением лабораторной работы студенты должны изучить регистровую модель, базовые типы и форматы данных, ознакомиться с системой команд языка ассемблера СPU х86-64 ([приложение А](file:///G:\Users\Dima\0501-KI%D0%B7%20-%2013%2014%20%D0%90%D0%9A\2020%20%D0%BE%D1%81%D0%B5%D0%BD%D1%8C\CA_appendix.pdf#%5B%7B%22num%22%3A15%2C%22gen%22%3A0%7D%2C%7B%22name%22%3A%22XYZ%22%7D%2C54%2C790%2C0%5D)).

При подготовке к выполнению практических заданий следует обратить особое внимание на группы команд целочисленной арифметики, команды сдвигов и команды условных переходов.

Также необходимо иметь представление о работе с программной средой Visual C++ и ее функциональных возможностях в контексте создания и отладки ассемблерных программ, встраиваемых в тело программы на языке C++.

## 8.3 Примеры выполнения практических заданий

8.3.1 Вычислить 7 значений функции Y = (15 \* х^2 – 12) / (4 \* x + 5) (х изменяется от 3 с шагом 3). Результат округлить до целого и разместить в памяти.

void main () // начало программы на языке С++

{

long X=3; // ячейка памяти для аргумента

long REZ[7]; // 7 ячеек памяти для результатов

\_asm

{ ; начало ассемблерной вставки

lea EBX, REZ ; загрузка адреса результатов в регистр EBX

mov ECX, 7 ; счетчик количества повторений цикла

m1: mov EAX, 4 ; EAX = 4

imul X ; EAX = 4 \* x

add EAX, 5 ; EAX = 4 \* x + 5

mov EDI, EAX ; пересылка знаменателя в регистр EDI

mov EAX, 15 ; EAX = 15

imul X ; EAX = 15 \* x

imul X ; EAX = 15 \* x^2

sub EAX, 12 ; EAX = 15 \* x2 – 12

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

} // окончание ассемблерной вставки

}

8.3.2 Определить номер (n) элемента последовательности an = 8^n – 5 \* n, при котором сумма элементов последовательности превысит 10000.

void main () // начало программы на языке С++

{

long N=0; // ячейка памяти для аргумента

long S=0; // ячейка для хранения суммы

long P=1; // ячейка для накопления 8n

\_asm

{ ; начало ассемблерной вставки

m1: inc N ; увеличение аргумента

mov EAX, 8 ; EAX = 8

mul P ; умножение – 8^n

mov P, EAX ; пересылка 8^n в ячейку памяти Р

add S, EAX ; накопление суммы

mov EAX, 5 ; EAX = 5

mul N ; EAX = 5 \* n

sub S, EAX ; накопление суммы

cmp S, 10000 ; сравнение суммы с 10000

jc m1 ; переход, если сумма меньше 10000

} // окончание ассемблерной вставки

}

8.3.3 В памяти задан массив из 5 элементов. Поместить в регистр EAX максимальный элемент массива, а в регистр EDI - его адрес в памяти.

void main () {

long x[5]={23, 56, 84, 15, 74} // массив в памяти

\_asm

{ ; начало ассемблерной вставки

lea EBX, x ; начальный адрес массива – в ЕВХ

mov ECX, 4 ; счетчик повторений

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

m2: add EBX, 4 ; увеличение адреса

cmp EAX, dword ptr[EBX] ; сравнение со следующим элемент.

jnc m1 ; переход, если больше

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

mov EDI, EBX ; адрес элемента – в EDI

m1: loop m2 ; зацикливание по счетчику

}

}

8.3.4 В памяти задан массив из 8 элементов. Отсортировать элементы массива по возрастанию. Пример «пузырьковой сортировки».

void main () {

long x[8]={7, 23, 56, 33, 84, 15, 11, 74};

\_asm

{

mov EDX, 7 ; счетчик внешнего цикла - на 1 меньше

; количества элементов массива

m3: lea EBX, x ; начальный адрес массива

mov ECX, EDX ; счетчик внутреннего цикла

m2: mov EAX, dword ptr[EBX] ; элемент массива – в ЕАХ

add EBX, 4

cmp EAX, dword ptr[EBX] ; сравнение соседних элементов

jc m1 ; переход, если меньше

xchg dword ptr[EBX], EAX ; обмен элементов массива

mov dword ptr[EBX-4], EAX

m1: loop m2 ; окончание внутреннего цикла

dec EDX ; уменьшение счетчика внешнего цикла

jnz m3 ; окончание внешнего цикла

}

}

## 

## 8.4 Порядок выполнения работы

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

После запуска программы Visual C++ в меню «File» необходимо выбрать команду «New». В открывшемся окне выбрать закладку «Projects», на которой:

* выбрать «Win32 Console Application»;
* в поле «Project name» записать имя проекта (например: lab\_1\_1 для задания 1, lab\_1\_2 для задания 2 и т.д.);
* в поле «Location» выбрать папку для записи проекта.

После формирования папки проекта необходимо ввести программу на языке «С++» с ассемблерной вставкой. Для этого в меню «File» выбрать команду «New». В открывшемся окне выбрать закладку «Files», на которой:

* выбрать «C++ Source File»;
* в поле «File name» записать имя файла.

Для компиляции программы необходимо нажать клавишу «F5».

Для пошаговой отладки программы необходимо:

* установить курсор в начале первой строки ассемблерной вставки;
* в меню «Build» выбрать команду «Start Debug» и вариант «Run to Cursor» (или нажать «Ctrl-F10»).

Каждый шаг отлаживается нажатием на кнопку «F10».

При пошаговой отладке промежуточные результаты необходимо контролировать в регистрах CPU или ячейках RAM. С этой целью следует открыть окна «Registers» и «Memory» из меню «Debug»->«Windows». Данная возможность становится доступной после того, как отладка была начата.

В окне «Registers» выбор отображаемых элементов программной модели CPU производится посредством контекстного меню данного окна (рисунок 8.1).

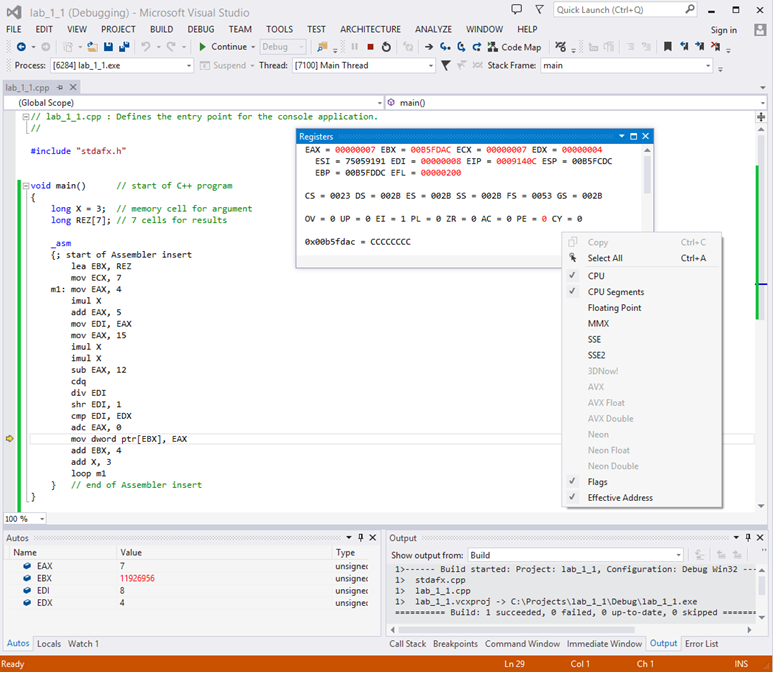


Рисунок 8.1 – Окно отладки «Registers»

В процессе отладки возможно иметь доступ к четырем независимым областям памяти (элементы «Memory n» меню «Memory» на рисунке 8.2).

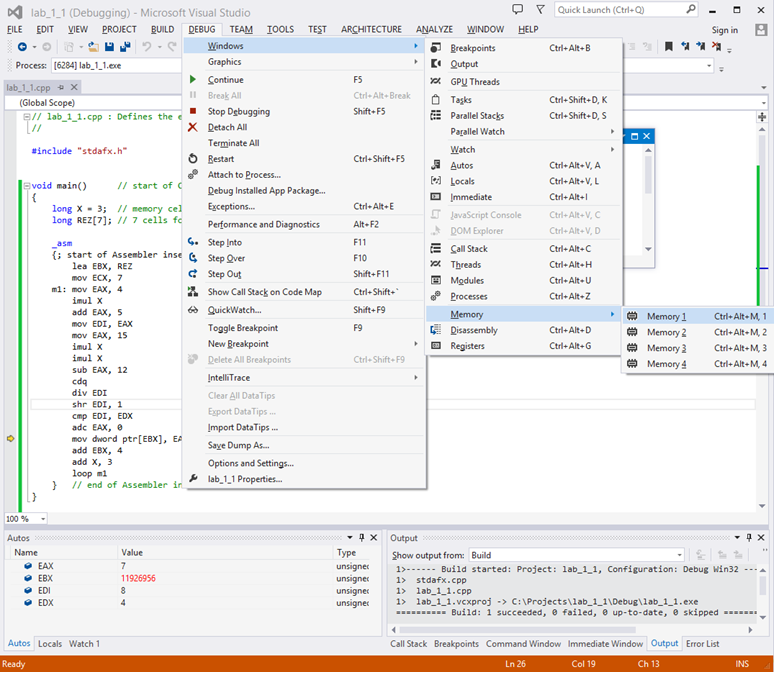


Рисунок 8.2 – Выбор областей памяти

В окне «Memory n», отображающем выбранную область памяти (например «Memory 1», как показано на рис. 8.3), посредством контекстного меню и поля ввода «Columns» рекомендуется установить соответствующие параметры отображения. Установку начального адреса области памяти следует выполнить с помощью поля ввода «Address».

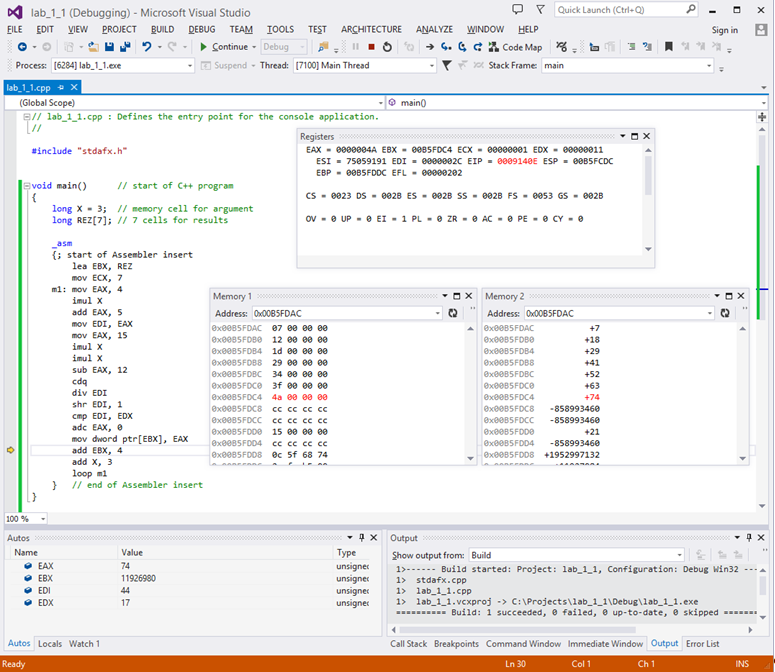


Рисунок 8.3 – Окно отладки «Memory 1»

Результаты выполнения программы представить в десятичном и шестнадцатиричном форматах.

## 8.5 Варианты индивидуальных заданий

#### 8.5.1 Исследование выполнения арифметических операций.

Задание 1. Вычислить 7 значений элементов последовательности an = 3 \* n^2 + 11 (для n от 4 с шагом 1). Результат разместить в памяти.

Задание 2. Вычислить 6 значений функции Y = (6^х + 12) /(4\* x^2 – 3) (х изменяется от 1 с шагом 1). Результат округлить до целого и разместить в памяти.

Задание 3. Вычислить 5 значений элементов последовательности an = 2 \* n^2 + 5 (для n от 4 с шагом 1). Результат разместить в памяти.

Задание 4. Вычислить 6 значений элементов последовательности an = (3^n) / (n + 5) (для n от 1 с шагом 1). Результат округлить до целого и разместить в памяти.

Задание 5. Вычислить 7 значений функции Y = 7500 / (2 \* х^2 + 15) (х изменяется от 3 с шагом 5). Результат округлить до целого и разместить в памяти.

Задание 6. Вычислить 6 значений функции Y = (6 \* х^2 + 12) / (5 \* x – 8) (х изменяется от 2 с шагом 4). Результат округлить до целого и разместить в памяти.

Задание 7. Вычислить 7 значений функции Y = 7 \* x^2 + 12 \* x – 32 (х изменяется от 3 с шагом 4). Результат разместить в памяти.

Задание 8. Вычислить 8 значений функции Y = 5 \* x^2 + 2 \* x – 14 (х изменяется от 2 с шагом 4). Результат разместить в памяти.

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.

Задание 10. Вычислить 8 значений функции Y = (8 \* х^2 + 12 \* х – 7) / (3 \* x + 25) (x изменяется от 2 с шагом 3). Результат округлить до целого и разместить в памяти.

#### 8.5.2 Исследование выполнения операций сравнения.

Задание 1. Найти целое значение аргумента, при котором функция Y = 9 \* x^2 – 8 \* x + 15 станет больше 1000.

Задание 2. Найти целое значение аргумента, при котором функция Y = 300 \* х / (8^x + 14) станет меньше 5.

Задание 3. Определить номер (n) элемента последовательности an = 5^n + 8 \* n, при котором сумма элементов последовательности превысит 20000.

Задание 4. Найти целое значение аргумента, при котором функция Y = 20000 / (8 \* x^2 + 25) станет меньше 20.

Задание 5. Найти целое значение аргумента, при котором функция Y = 7 \* x^2 + 25 \* x – 27 станет больше 3000.

Задание 6. Определить номер (n) элемента последовательности an = n^2 +6 \* n + 28, при котором сумма элементов последовательности превысит 1000.

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

Задание 8. Определить номер (n) элемента последовательности an = 3\* n^2 – 5 \* n + 12, при котором сумма элементов последовательности превысит 1500.

Задание 9. Найти целое значение аргумента, при котором функция Y = (2000 + х) / (8 \* x^2 + 25) станет меньше 10.

Задание 10. Найти целое значение аргумента, при котором функция Y = 15 \* x^2 + 11 \* x – 16 станет больше 2000.

#### 8.5.3 Исследование выполнения операций над массивами в памяти.

Задание 1. В памяти задан массив из 10 элементов. Сохранить в регистре ESI количество единичных битов во всех элементах.

Задание 2. В памяти задан массив из 11 элементов. Отсортировать элементы массива по убыванию.

Задание 3. В памяти задан массив из 10 элементов. Поместить в регистр EAX минимальный элемент массива, а в регистр EDX его адрес в памяти.

Задание 4. В памяти задан массив из 10 элементов. Сохранить в регистре ESI количество отрицательных элементов.

Задание 5. Рассчитать и сохранить в памяти элементы массива, заданные функцией Y = n! (для n от 1 до 8).

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

Задание 7. В памяти задан массив из 10 элементов. Заменить эти числа произведением их старшего и младшего слова.

Задание 8. В памяти задан массив из 8 элементов. Поместить в регистр EAX максимальный элемент массива, а в регистр ESI его адрес в памяти.

Задание 9. В памяти задан массив из 9 элементов. Отсортировать элементы массива по возрастанию.

Задание 10. В памяти задан массив из 10 элементов. Сохранить в регистре ESI количество нечетных элементов.

## 8.6 Схема вариантов заданий

| №№  варианта | Комбинация вариантов заданий | | |
| --- | --- | --- | --- |
| № задания из пп.**8.5.1** | № задания из пп.**8.5.2** | № задания из пп.**8.5.3** |
|  | 1 | 1 | 1 |
|  | 2 | 2 | 5 |
|  | 3 | 5 | 3 |
|  | 4 | 8 | 1 |
|  | 9 | 8 | 4 |
|  | 7 | 2 | 7 |
|  | 5 | 2 | 5 |
|  | 10 | 3 | 2 |
|  | 6 | 9 | 8 |
|  | 1 | 4 | 5 |
|  | 8 | 6 | 2 |
|  | 9 | 7 | 6 |
|  | 7 | 8 | 5 |
|  | 1 | 8 | 6 |
|  | 9 | 4 | 2 |
|  | 6 | 4 | 6 |
|  | 5 | 9 | 4 |
|  | 8 | 2 | 10 |
|  | 10 | 5 | 4 |
|  | 9 | 10 | 3 |
|  | 4 | 3 | 9 |
|  | 2 | 1 | 10 |
|  | 3 | 6 | 7 |
|  | 5 | 7 | 8 |
|  | 6 | 10 | 9 |
|  | 10 | 9 | 1 |
|  | 3 | 7 | 10 |
|  | 2 | 10 | 8 |
|  | 4 | 6 | 9 |
|  | 7 | 1 | 4 |
|  | 8 | 3 | 7 |
|  | 1 | 5 | 7 |
|  | 5 | 4 | 3 |
|  | 10 | 4 | 5 |
|  | 9 | 5 | 1 |
|  | 7 | 4 | 2 |
|  | 4 | 1 | 3 |
|  | 8 | 10 | 6 |
|  | 6 | 7 | 9 |
|  | 2 | 9 | 8 |
|  | 3 | 6 | 10 |
|  | 2 | 3 | 6 |
|  | 3 | 7 | 1 |
|  | 1 | 2 | 9 |
|  | 5 | 9 | 10 |
|  | 10 | 6 | 7 |
|  | 6 | 5 | 8 |
|  | 7 | 5 | 3 |
|  | 9 | 2 | 5 |
|  | 10 | 5 | 7 |
|  | 1 | 7 | 6 |
|  | 2 | 4 | 9 |
|  | 4 | 4 | 8 |
|  | 9 | 1 | 3 |
|  | 8 | 7 | 10 |
|  | 6 | 8 | 4 |
|  | 4 | 3 | 2 |
|  | 8 | 1 | 4 |
|  | 3 | 10 | 1 |
|  | 7 | 3 | 2 |
|  | 5 | 6 | 4 |
|  | 4 | 9 | 1 |
|  | 10 | 7 | 3 |
|  | 7 | 6 | 6 |
|  | 1 | 1 | 8 |
|  | 9 | 3 | 1 |
|  | 2 | 8 | 9 |
|  | 3 | 10 | 7 |
|  | 6 | 9 | 10 |
|  | 8 | 2 | 5 |
|  | 5 | 8 | 2 |
|  | 3 | 10 | 4 |
|  | 4 | 8 | 5 |
|  | 9 | 8 | 7 |
|  | 8 | 9 | 4 |
|  | 1 | 10 | 8 |
|  | 5 | 6 | 10 |
|  | 2 | 1 | 3 |
|  | 7 | 3 | 6 |
|  | 6 | 2 | 9 |
|  | 10 | 5 | 2 |
|  | 7 | 9 | 1 |
|  | 5 | 3 | 4 |
|  | 6 | 7 | 3 |
|  | 10 | 1 | 8 |
|  | 8 | 4 | 2 |
|  | 2 | 5 | 6 |
|  | 4 | 10 | 5 |
|  | 1 | 9 | 3 |
|  | 3 | 6 | 9 |
|  | 6 | 2 | 10 |
|  | 4 | 5 | 7 |
|  | 1 | 6 | 5 |
|  | 2 | 4 | 10 |
|  | 8 | 7 | 6 |
|  | 10 | 8 | 7 |
|  | 7 | 1 | 2 |
|  | 3 | 2 | 1 |
|  | 9 | 3 | 8 |
|  | 5 | 10 | 9 |

## 8.7 Содержание отчета

Отчет о лабораторной работе должен содержать:

* цель работы;
* краткое описание теоретических основ;
* постановку задачи;
* экспериментально-практическую часть (включая анализ результатов);
* выводы по работе.

В экспериментально-практической части необходимо привести исходные тексты разработанных программ (на языке C++ с ассемблерными вставками) и результаты их выполнения, полученные в регистрах CPU и (или) ячейках памяти.

## 8.8 Контрольные вопросы

1. Что представляет собой программная модель 32-разрядных процессоров х86-64?
2. Перечислите форматы данных процессоров х86-64.
3. Для чего нужны форматы двоично-десятичных чисел?
4. Перечислите методы (способы) адресации данных в процессорах х86-64.
5. Перечислите команды пересылки данных и варианты расположения операндов-приемников и операндов-источников.
6. Перечислите арифметические и логические команды.
7. Где могут располагаться операнды команд умножения и деления, и куда могут записываться их результаты?
8. Какие команды позволяют обрабатывать десятичные данные без перевода их в двоичный формат?
9. Как выполняются команды сдвигов?
10. Перечислите цепочечные (строковые) команды и особенности их выполнения.
11. Как выполняются команды условных и безусловных переходов?
12. Чем отличаются команды JMP и CALL?