Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение
высшего профессионального образования
«Владимирский государственный университет
имени Александра Григорьевича и Николая Григорьевича Столетовых»

Кафедра вычислительной техники

АССЕМБЛЕР

Методические указания
к лабораторным работам

Составитель
И. А. КОЛОМИЕЦ

Владимир 2012

УДК 621.396
ББК 32.988-5я7
А91

Рецензент
Кандидат технических наук,
доцент кафедры радиотехники и радиосистем
Владимирского государственного университета
имени Александра Григорьевича и Николая Григорьевича Столетовых
В.А. Ефимов

Печатается по решению редакционно-издательского совета ВлГУ

А91 **Ассемблер :** метод. указания к лаборатор. работам / Владим. гос. ун-т имени Александра Григорьевича и Николая Григорьевича Столетовых ; сост. И. А. Коломиец. – Владимир : Изд-во ВлГУ, 2012. – 20 с.

Рассматриваются вопросы создания программ на языке ассемблера для вычислительных процессов различных структур. Приводятся примеры с детальным пояснением назначения и логики работы команд ассемблера, раскрываются приемы программирования.

Предназначены для студентов 1-го курса направления 230100 – информатика и вычислительная техника всех форм обучения.

Рекомендованы для формирования профессиональных компетенций в соответствии с ФГОС 3-го поколения

Библиогр.: 6 назв.

УДК 621.396
ББК 32.988-5я7

ВВЕДЕНИЕ

Лабораторные работы посвящены изучению архитектуры вычислительной машины и ее функционирования, принципов обработки машинных команд, особенностей реализации обработки различных типов данных. Рассматриваются принципы низкоуровневого программирования разветвляющихся и циклических вычислительных процессов, обработки массивов, подпрограмм с особенностями передачи в них параметров.

Основная цель лабораторных работ – сформировать у студентов глубокое понимание процессов, протекающих в ЭВМ при выполнении различного рода программ, что позволит значительно повысить качественный уровень и эффективность будущих разработок вычислительных систем.

Ассемблер – это язык символического программирования, сохраняющий все возможности машинного языка. В то же время Ассемблер – это системная программа, выполняющая перевод программы, написанной на языке ассемблера, на машинный язык. Сам процесс перевода называется ассемблированием. Ассемблер – это машинно-зависимый язык, т.е. язык ассемблера для различных архитектур вычислительных машин будет несколько различаться, хотя общие принципы будут теми же. Цикл лабораторных работ рассчитан на программирование для учебной цифровой вычислительной машины (ЦВМ) и для процессора x86. Для того чтобы отличить конкретный язык ассемблера учебной ЦВМ от языка ассемблера процессора x86, будем записывать первый с прописной буквы - Ассемблер.

Первые две лабораторные работы направлены на освоение приёмов разработки программ на языке Ассемблер и наблюдение за происходящими в учебной вычислительной машине процессами. В следующих двух работах те же задания выполняются с использованием языка ассемблера для x86-совместимых вычислительных машин. Последняя работа направлена на формирование навыков разработки разноязыковых многомодульных программ.

СОДЕРЖАНИЕ ОТЧЕТОВ

Каждая лабораторная работа включает в себя оформление соответствующей документации. В документации обязательно должны быть представлены следующие пункты:

1) текст индивидуального задания по варианту;

2) алгоритм решения задачи на языке высокого уровня C++;

3) спецификации всех разработанных процедур и/или функций (если есть);

4) схема алгоритма решения задачи;

5) текст программы на языке ассемблера;

6) тесты и результаты тестирования. Тесты должны включать входные и выходные данные и содержать все возможные варианты входных данных. Тесты должны покрывать все ветви алгоритма;

7) выводы по работе.

Все представленные пункты должны быть отражены в отчете по каждой лабораторной работе.

ПОДГОТОВКА К ЛАБОРАТОРНЫМ РАБОТАМ

Для выполнения первых двух лабораторных работ потребуется учебная цифровая вычислительная машина (УЦВМ) – файл TCom.exe.

Для выполнения последующих лабораторных работ потребуются:

1) любой текстовый редактор, в котором будут писаться программы на языке ассемблера; программы необходимо сохранять в файлах с расширением *.asm (стандартные соглашения);

2) TASM (Turbo Assembler) – программный пакет компании Borland, предназначенный для разработки программ на языке ассемблера для архитектуры x86.

В программный пакет TASM входят следующие файлы: TASM.exe – компилятор с ассемблера в машинные кода, TLink.exe – компоновщик, TD.exe – Turbo Debugger – среда для отладки кода, TDHelp.tdh – справочная информация по Turbo Debugger.

Кроме этого, для проработки и отладки алгоритма решения индивидуального задания каждой работы может пригодиться среда разработки на языке высокого уровня C++.

Лабораторная работа № 1

ОБРАБОТКА МАССИВОВ В УЧЕБНОЙ ЦВМ

1. Цель работы

Изучение принципов работы с одномерными и многомерными массивами; программирование разветвляющихся и циклических вычислительных процессов.

2. Порядок выполнения работы

2.1. Изучить систему команд Ассемблера и структуру программы, псевдокоманды *resb*, *word*, *start* и *end*.

2.2. Изучить процесс ассемблирования.

2.3. Изучить способы обработки массивов, индексную адресацию.

2.4. Изучить программные и аппаратные возможности УЦВМ для обработки массивов и организации разветвляющихся и циклических структур алгоритмов.

2.5. Изучить пример выполнения задания и на модели УЦВМ проследить ход его выполнения.

2.6. Разработать алгоритм решения индивидуального задания, удовлетворяющий требованиям п. 2.7, и записать его на языке С++.

2.7. Требования к алгоритму. Детализация алгоритма должна быть максимально приближена к командам языка Ассемблер.

2.8. Разработать программу на языке Ассемблер, реализующую алгоритм п. 2.6.

2.9. Загрузить программу в учебную ЦВМ и выполнить ее ассемблирование.

2.10. Выполнить тестирование программы несколькими наборами входных данных.

3. Пример выполнения задания

Переменной *max* присвоить значение максимального элемента одномерного массива.

Алгоритм решения задачи на языке C++

```
int main()
{
     unsigned int d[5]={5,2,8,3,1};
     unsigned int max;

     max=d[0];
     for(int i=1; i<5; i++)
          if (d[i]>max) max=d[i];

     return 0;
}
```

Текст программы на Ассемблере

```
      (АМ)
H Ex33        | Ex33  start  0
T 000 00002F  |       lda    d     ;max:=d[1]
T 003 0C0026  |       sta    max
T 006 040029  |       ldx    c3    ;i:=2
T 009 00802F  | rpt   lda    d,x   ;A:=d[i]
T 00C 280026  |       comp   max   ;A > max ?
T 00F 380018  |       jlt    m     ;Переход, если "меньше".
T 012 300018  |       jeq    m     ;Переход, если "равно".
T 015 0C0026  |       sta    max   ;max:=A
T 018 AC10    | m     rmo    x,a   ;i:=i+1
T 01A 180029  |       add    c3
T 01D AC01    |       rmo    a,x
T 01F 28002C  |       comp   c15   ;i <6 ?
T 022 380009  |       jlt rpt      ;Переход, если "да".
T 025 FF      |       hlt
              | ; данные
T 026         | max   resb 3      ;резервируем 3 байта
T 029 000003  | c3    word 3      ;константа 3
T 02C 00000F  | c15   word 15     ;константа 15
T 02F 000008  | d     word 8      ;задание элементов массива
T 032 000010  |       word 16
T 035 000004  |       word 4
T 038 00000C  |       word 12
T 03B 000007  |       word 7
E 000         |       end Ex33
```

4. Варианты индивидуальных заданий

Количество элементов в массиве следует считать фиксированным, равным 5 – 10. Если размерность массива не указана, то предполагается, что он одномерный.

1. Найти значение минимального элемента массива.
2. Найти индекс максимального элемента массива.
3. Найти скалярное произведение двух векторов.
4. Найти сумму элементов главной диагонали квадратной матрицы.
5. Найти сумму положительных элементов массива.
6. Найти индекс элемента с заданным значением. Предполагается, что в массиве есть такой элемент и он единственный.
7. Определить количество положительных и отрицательных элементов массива.
8. Определить количество элементов массива, которым предшествуют элементы с меньшими значениями.
9. Каждому элементу массива, начиная со второго, присвоить значение максимального элемента из числа ему предшествующих и его самого.
10. Дана прямоугольная матрица. Найти сумму элементов строки с заданным номером.
11. Определить, какие два последовательных элемента массива наименее отличаются друг от друга. Найти индекс первого элемента пары.
12. Построить массив, элементы которого суть суммы последовательных пар элементов исходного массива.
13. Определить количество элементов массива, значения которых превышают заданное.
14. Массив, элементы которого принадлежат множеству {0,1}, рассматривается как представление целого числа. Определить значение числа, заданного таким способом.
15. Дан массив, элементы которого принадлежат множеству {0,1}. Определить длину первой последовательности рядом стоящих единиц.

5. Контрольные вопросы

1. Форматы команд учебной ЦВМ.
2. Как производится ассемблирование исходного кода на Ассемблере в машинные коды?
3. Какие архитектурные средства предусмотрены в УЦВМ для обработки массивов? В чем состоит их назначение?
4. Какие программы называются реентерабельными?
5. Каким образом массивы представляются в памяти УЦВМ?
6. Каким образом вычисляется исполнительный адрес при прямой и индексной адресациях?

Лабораторная работа № 2

ПОДПРОГРАММЫ В УЧЕБНОЙ ЦВМ

1. Цель работы

Изучение основных принципов организации вычислительного процесса с помощью подпрограмм; изучение организации передачи параметров в подпрограммы.

2. Порядок выполнения работы

2.1. Изучить организацию подпрограмм, особенности организации связи по управлению и связи по данным.

2.2. Изучить программные и аппаратные возможности УЦВМ для организации подпрограмм.

2.3. Изучить пример выполнения задания и на модели УЦВМ проследить ход его выполнения.

2.4. Разработать алгоритм решения индивидуального задания, удовлетворяющий требованиям п. 2.5, и записать его на языке С++.

2.5. Требования к алгоритму. Детализация алгоритма должна быть максимально приближена к командам языка Ассемблер. Решение задачи

организовать в виде *подпрограммы*, обязательно все данные передавать в подпрограмму *списком параметров*. В основном коде предусмотреть организацию таблицы адресов и значений (ТАЗ) для передачи фактических параметров и вызов разработанной подпрограммы.

2.6. Разработать программу на языке Ассемблер, реализующую алгоритм п. 2.4.

2.7. Загрузить программу в учебную ЦВМ и выполнить ее ассемблирование.

2.8. Выполнить тестирование программы несколькими наборами входных данных.

3. Пример выполнения задания

Составить подпрограмму копирования переменной.

Алгоритм решения задачи на языке С++

```
void copy(unsigned int x, unsigned int &y)
{ y = x; }

int main()
{
const unsigned int u = 15;
     unsigned int v, w;

copy(u,v); copy(v,w);

return 0;
}
```

Текст программы на Ассемблере

```
      (AM)
H Ex43        | Ex43  start  0
              | ; вызов copy(u,v)
T 000 00001F  |       lda    u
T 003 0C0009  |       sta    p11
T 006 480028  |       jsub   copy
                                  ;ТАЗ:
T 009         | p11   resb   3    ;u - по значению
T 00C 000022  | p12   word   v    ;v - по ссылке
              | ; вызов copy(v,w)
```

```
T 00F 000022 |       lda    v
T 012 0C0018 |       sta    p21
T 015 480028 |       jsub   copy
T 018        | p21   resb   3
T 01B 000025 | p22   word   w
T 01E FF     |       hlt
             | ; данные
T 01F 00000F | u     word   15
T 022        | v     resb   3
T 025        | w     resb   3
             | ; подпрограмма copy
T 028 AC21   | copy  rmo    l,x  ;X := "адрес ТАЗ"
T 02A 008000 |       lda    0,x  ;A := p1
T 02D 048003 |       ldx    3,x  ;X := p2
T 030 0C8000 |       sta    0,x  ;p2 := A
T 033 AC20   |       rmo    l,a  ;Корректировка
T 035 18003B |       add    c6   ;адреса возврата
T 038 AC02   |       rmo    a,l  ;на длину ТАЗ.
T 03A 4C     |       rsub
             | ; данные п/п copy
T 03B 000006 | c6    word   6
E 000        |       end    Ex43
```

4. Варианты индивидуальных заданий

1. Даны три переменные. Определить наибольшее значение.

2. Даны три переменные. Результату присвоить значение 1, если переменные упорядочены по возрастанию; иначе - 0.

3. Определить n-й член ряда Фибоначчи: $F[1] = 1$, $F[2] = 1$, $F[n] = F[n - 1] + F[n - 2]$.

4. Вычислить $y = 1 + x + 2x^2 + 3x^3 + 4x^4 + 5x^5$.

5. Известно, что две переменные имеют одинаковые значения, а третья - отличное от них. Найти ее значение.

6. Составить подпрограмму перемножения двух чисел без применения команды умножения.

7. Составить подпрограмму целочисленного деления двух чисел без применения команды деления. Если деление невозможно, то переменной, называемой индикатором ошибки, присвоить значение 1, иначе - 0.

8. Составить подпрограмму для нахождения наибольшего общего делителя двух чисел.

9. Вычислить факториал числа n, где $n \geq 1$.

10. Вычислить n-ю степень числа, где $n \geq 1$.

11. Определить максимальное число, факториал которого может быть представлен в машине.

12. Вычислить $y = \min(x^2, \max(y,10))$.

13. Вычислить число размещений из m по n по формуле $A(n, m) = m(m - 1)(m - 2), ..., (m - (n - 1))$, где $m \geq n$.

14. Вычислить $y = \max(x - 1, y^2, z \text{ div } 3)$.

15. Даны три переменные с неравными значениями, рассматриваемые как координаты точек на числовой оси. Определить длину минимального отрезка, включающего все три точки.

5. Контрольные вопросы

1. Какие специальные архитектурные средства предусмотрены в УЦВМ для организации подпрограмм? Каково их назначение?

2. Объясните алгоритм выполнения команд *jsub* и *rsub*.

3. Перечислите основные способы связывания программных единиц по данным и объясните, как они реализуются.

4. Как в Ассемблере организуется передача параметров по значению, по ссылке?

Лабораторная работа № 3

ОБРАБОТКА МАССИВОВ

1. Цель работы

Изучение средств обработки массивов языка ассемблера микропроцессора (МП) Intel 8086.

2. Порядок выполнения работы

2.1. Изучить принципы индексной адресации и способы доступа к элементам массива, работу команды *loop*, функции прерывания *21h* (4Ch, 01h, 09h).

2.2. Изучить пример выполнения задания.

2.3. Разработать алгоритм решения индивидуального задания, удовлетворяющий требованиям п. 2.4 и записать его на языке C++.

2.4. Требования к алгоритму. Детализация алгоритма должна быть максимально приближена к командам языка ассемблера. Обязательно предусмотреть *вывод результатов на экран*.

2.5. Разработать программу на языке ассемблера, реализующую алгоритм п. 2.3.

2.6. Выполнить сборку разработанной программы в исполняемый файл (*.exe) (см. порядок трансляции и компоновки).

2.7. Выполнить тестирование программы несколькими наборами входных данных.

3. Порядок трансляции и компоновки.

Для сборки написанного кода на ассемблере (*.asm) в исполняемый файл (*.exe) необходимо выполнить следующее:

1) из командной строки запустить команду: tasm <имя_файла>.asm; в результате будет выполнена компиляция исходного файла и создан объектный файл с тем же именем, но с расширением *.obj (<имя_файла>.obj);

2) из командной строки запустить команду: tlink <имя_файла>.obj; в результате будет выполнена компоновка объектного файла и создан исполняемый файл с тем же именем, но с расширением *.exe (<имя_файла>.exe).

Порожденный таким образом код можно отлаживать с помощью Turbo Debugger.

4. Пример выполнения задания

Переменной *Max* присвоить значение максимального элемента одномерного массива.

Алгоритм решения задачи на языке C++

```
int main()
{
     unsigned int A[5]={5,2,8,3,1};
     unsigned int Max;
     unsigned char MaxStr[2] = {' ','\0'};
```

```
     Max=A[0];
     for(int i=1; i<5; i++)
          if (A[i]>Max) Max=A[i];
     MaxStr = Max + 48;
     printf("%s\n", MaxStr)
     return 0;
}
```

Текст программы на языке ассемблера

```
.model small   ; задание модели памяти
.stack 200h    ; задание сегмента стека размером 512 байт
.data               ; начало сегмента данных

; выделение 10 байт под массив из 5 элементов (по 2 байта
; каждый) с начальной инициализацией
A          dw   5 dup (5,2,8,3,1)

Max        dw   ?   ; выделение 2 байт под переменную

MaxStr     dw   ?   ; значение Max в символьном виде
           db   '$' ; символ конца строки

.code              ; начало сегмента кода
Begin:
           mov ax, @Data ;  настройка  регистра  сегмента
данных
           mov ds, ax

           mov  ax, A
           mov  Max, ax

; в регистр si заносится смещение элемента массива относительно
; базового адреса массива(т.к. один элемент массива занимает
; 2 байта, то для второго элемента смещение равно 2 байтам,
; для третьего элемента - 4 байтам и т.д.)

           mov  si, 2

; команда цикла loop работает с регистром cx, в который
; заносится необходимое количество итераций

           mov  cx, 4

for_cycle: mov  ax, A[si]; команде дано символическое имя -
                          ; метка for_cycle
           cmp  ax, Max
           jle  do_else
           mov  Max, ax
```

```
do_else:   add  si, 2
           loop for_cycle

;вывод значения переменной Max на экран (для примера считаем
;Max числом, состоящим из одного знака)
           add ax, 48     ; получаем код символа значения Max
           mov MaxStr, ax

           mov dx, offset MaxStr
           mov ax, 09h
           int 21h

           mov ax, 4C00h ; корректное завершение программы
           int 21h

           end Begin
```

5. Варианты индивидуальных заданий

В качестве индивидуального задания для выполнения работы следует использовать варианты заданий из лаб. работы 1.

6. Контрольные вопросы

1. Программно-доступные регистры МП Intel 8086, их назначение.
2. Система команд МП Intel 8086: типы операций.
3. Режимы адресации; вычисление исполнительного адреса.
4. Структура оператора языка ассемблер МП INTEL 8086.
5. Директивы компилятора.
6. Процедуры трансляции и компоновки.
7. Особенности организации циклов.
8. Объявление массивов.
9. Индексная адресация.
10. Доступ к элементам массивов.
11. Линеаризация многомерных массивов.

Лабораторная работа № 4

ОРГАНИЗАЦИЯ ПОДПРОГРАММ

1. Цель работы

Изучение способов организации подпрограмм и передачи параметров по ссылке и по значению; приобретение навыков работы со стеком.

2. Порядок выполнения работы

2.1. Изучить принципы организации подпрограмм: описание, связь по управлению, связь по данным; работу со стеком. Изучить режимы адресации.

2.2. Изучить работу команд *call* и *ret*, *push* и *pop*.

2.3. Изучить пример выполнения задания. Обратить внимание на передачу параметров в подпрограмму через стек и работу со стеком в теле подпрограммы, на использованные режимы адресации.

2.4. Разработать алгоритм решения индивидуального задания, удовлетворяющий требованиям п. 2.5 и записать его на языке C++.

2.5. Требования к алгоритму. Детализация алгоритма должна быть максимально приближена к командам языка ассемблера. Обязательно выделить часть алгоритма в виде *подпрограммы с параметрами* (необходимо использовать различные варианты передачи параметров: по ссылке, по значению). Предусмотреть вывод результатов на экран.

2.6. Разработать программу на языке ассемблера, реализующую алгоритм п. 2.4.

2.7. Разработанную программу транслировать в исполняемый файл (*.exe).

2.8. Выполнить тестирование программы несколькими наборами входных данных.

3. Пример выполнения задания

Переменной *Max* присвоить значение максимального элемента одномерного массива.

Текст программы на языке C++

```
void   MaxMas(unsigned   int*   mas,   unsigned   int   &max,
              unsigned int num)
{
     max=mas[0];
     for(int i=1; i<num; i++)
          if (mas[i]>max) max=mas[i];
}

int main()
{
     unsigned int A[5]={5,2,8,3,1};
     unsigned int Max;
     unsigned char MaxStr[2] = {' ','\0'};

     MaxMas(A, Max, 5);
     MaxStr[0] = Max + 48;
     printf("%s\n", MaxStr)
     return 0;
}
```

Текст программы на языке ассемблера

```
.model small
.stack 200h
.data

A          dw   5 dup (5,2,8,3,1)

Max        dw   ?

MaxStr     dw   ?
           db   '$'

.code

MaxMas     proc near ; объявление процедуры ближнего типа

; сохраняем содержимое регистра bp в стеке,
; т.к. будем его изменять
           push bp

; настраиваем bp на вершину стека
           mov  bp, sp
```

```
; загружаем в bx содержимое ячейки памяти с адресом bp+8,
; т.к. bp (см. предыд. команду) содержит адрес вершины стека,
; то в bx загружается из стека пятое, считая с вершины, слово
; таким образом, в bx загружается адрес массива A
      mov  bx, bp[8]

;загружаем в ax содержимое ячейки памяти с адресом, хранящимся
; в bx, т.е. в ax загружается первый элемент массива A
      mov  ax, [bx]

; загружаем в di адрес переменной Max
      mov  di, [bp+6]

; сохраняем значение первого элемента массива в переменной Max
      mov  [di], ax

      mov  si, 2
; загружаем в cx количество итераций из стека
      mov  cx, bp[4]

; выполняем действия, аналогичные предыдущему примеру
for_cycle: mov  ax, [bx+si]
      cmp  ax, [di]
      jle  do_else
      mov  [di], ax
do_else: add si, 2
      loop for_cycle

;восстанавливаем сохраненное ранее в стеке значение bp
go_exit: pop bp

;команда возврата из подпрограммы изменяет значение регистра
;ip на адрес возврата, вынимая его из вершины стека,
;и вынимает из стека 6 байт (освобождает)
      ret 6

MaxMas endp ; конец процедуры

Begin: mov ax, @Data
       mov ds, ax

; помещаем в стек фактические параметры процедуры:
; 1)помещаем в стек адрес массива A (его смещение относительно
; начала сегмента данных) - передача по ссылке
      mov  ax, offset A
      push ax

; 2)помещаем в стек адрес Max (его смещение относительно
; начала сегмента данных) - передача по ссылке
```

```
        mov  ax, offset Max
        push ax

; 3)помещаем в стек число 4 – передача по значению
        mov  ax, 4
        push ax

;вызываем подпрограмму: содержимое регистра ip (у нас ближний
; вызов) помещается в стек, затем ip присваивается адрес
; первой команды процедуры MaxMas
        call MaxMas

        mov ax, Max
        add ax, 48
        mov MaxStr, ax

        mov dx, offset MaxStr
        mov ah, 09h
        int 21h

        mov ax, 4C00h
        int 21h

        end Begin
```

4. Варианты индивидуальных заданий

В качестве индивидуального задания для выполнения работы следует использовать варианты заданий из лаб. работы 2.

5. Контрольные вопросы

1. Архитектура МП Intel 8086.
2. Специальные архитектурные средства для организации подпрограмм в МП Intel 8086, назначение этих средств.
3. Основные способы связи программных единиц.
4. Режимы адресации; вычисление исполнительного адреса.
5. Модели памяти и сегментация.
6. Команды работы со строками.
7. Префикс *rep*, его функция.
8. Типы исполняемых модулей.

Лабораторная работа № 5

ИСПОЛЬЗОВАНИЕ АССЕМБЛЕРА В ЯЗЫКАХ ВЫСОКОГО УРОВНЯ

1. Цель работы

Изучение особенностей построения разноязыковых программ; изучение взаимосвязи разных уровней представления программы.

2. Порядок выполнения работы

2.1. Изучить стандартные соглашения о регистрах, соглашения о метках и типах, соглашения о передаче параметров. Изучить процедуры пролога и эпилога.

2.2. Разработать процедуру на языке высокого уровня C++, реализующую вычисление функции из индивидуального задания.

2.3. Разработать процедуру вычисления функции на языке ассемблера, используя встроенные средства языка C++.

2.4. Разработать отдельную процедуру вычисления функции на языке ассемблера и подключить её к проекту на языке C++.

2.5. Выполнить анализ реализованных процедур вычисления заданной функции по быстродействию, для чего разработать соответствующую главную функцию на языке C++.

Все данные разработанных процедур обязательно должны передаваться списком параметров.

3. Варианты индивидуальных заданий

1. $(2c - d + 23) / (a/4 - 1)$.
2. $(c + 4d - 123) / (1 - a/2)$.
3. $(-2c + 82d) / (a/4 - 1)$.
4. $(2c + d - 52) / (a/4 + 1)$.
5. $(c/4 - 62d) / (a \cdot a + 1)$.
6. $(-2c - d + 53) / (a/4 - 1)$.
7. $(2c - d/4) / (a \cdot a + 1)$.
8. $(2 + c - 23d) / (2a \cdot a - 1)$.
9. $(2c - d/3) / (b - a/4)$.
10. $(4c + d - 1) / (c - a/2)$.
11. $(2c - 42d) / (c + a - 1)$.
12. $(25/c - d + 2) / (b + a \cdot a - 1)$.
13. $(c - d/2 + 33) / (2a \cdot a - 1)$.
14. $(4c - d/2 + 23) / (a \cdot a - 1)$.
15. $(c \cdot d + 23) / (a/2 - 4d - 1)$.

4. Контрольные вопросы

1. Какие существуют формы комбинирования программ на языке высокого уровня с ассемблером? Дайте им характеристику.

2. Процедуры пролога и эпилога.

3. Соглашения о регистрах, соглашения о метках и типах, соглашения о передаче параметров.

4. Возможности встроенных средств C++ для использования языка ассемблера.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. *Пирогов, В. Ю.* Ассемблер на примерах / В. Ю. Пирогов. – СПб. : БХВ-Петербург, 2005. – 416 с. ISBN 5-94157-745-1. – ISBN 978-5-94157-745-3.

2. *Он же*. Ассемблер для Windows / В. Ю. Пирогов. – 2-е изд., перераб. и доп. – СПб. : БХВ-Петербург, 2003. – 648 с. – ISBN 5-94157-329-4.

3. *Голубь, Н. Г.* Искусство программирования на Ассемблере. Лекции и упражнения / Н. Г. Голубь. – 2-е изд., испр. и доп. – СПб. : ДиаСофтЮП, 2002. – 656 с. – ISBN 5-93772-056-3.

4. *Шилдт, Г.* C++: базовый курс / Г. Шилдт. – М. : Вильямс, 2008. – 614 с. – ISBN 5-8459-0768-3. – ISBN 0-07-222897-0.

5. Assembler: учеб. для вузов. / В. И. Юров. – 2-е изд. – СПб. : Питер, 2003. – 637 с. – ISBN 5-94723-581-1.

6. *Пильщиков, В. Н.* Программирование на языке ассемблера IBM PC / В. Н. Пильщиков. – М. : Диалог-МИФИ, 2003. – 288 с. – ISBN 5-85404-051-7.