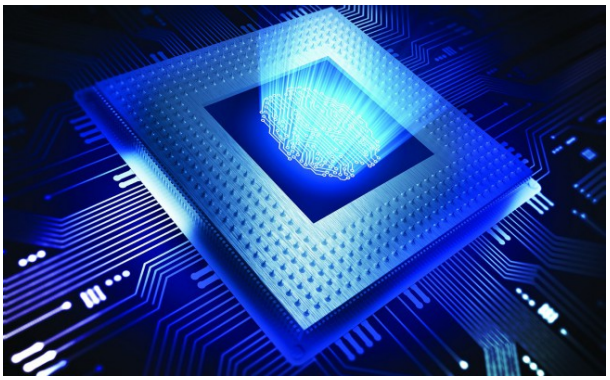
ГОУ ВПО «ДОНЕЦКИЙ НАЦИОНАЛЬНЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

МЕТОДИЧЕСКИЕ УКАЗАНИЯ К ЛАБОРАТОРНЫМ РАБОТАМ ПО КУРСУ "АРХИТЕКТУРА КОМПЬЮТЕРОВ"

(для студентов направления подготовки 09.04.04 «Программная инженерия»)

Составители: С.А. Зори, д.т.н., проф. кафедры ПИ ДонНТУ

У т в е р ж д е н о
на заседании кафедры ПИ
Протокол № 1 от 30.08.19



Донецк 2019

ОБЩИЕ СВЕДЕНИЯ

Предлагаемый курс лабораторных работ по дисциплине "Архитектура компьютеров" имеет своей целью развитие у студентов знаний и умений в области теории и средств организации архитектуры компьютеров и вычислительных систем, функционирования компьютера и его составных частей, разработки программ для управления компьютером на низком уровне.

Порядок выполнения лабораторных работ:

-- при самостоятельной подготовке к работе студент должен выполнить теоретическое задание либо написать программу в соответствии с вариантом и выполнить необходимые контрольные расчеты;

-- на занятии студент должен предъявить преподавателю подготовку и получить допуск к выполнению работы либо предъявить выполненное теоретическое задание на проверку;

-- отладив программу на ПК, студент должен пояснить преподавателю работу программы и обосновать правильность ее функционирования;

-- к следующей лабораторной работе студент должен оформить отчет по выполненной работе и защитить его у преподавателя.

Рекомендуется подписанные преподавателем отчеты хранить до конца обучения по курсу, так на экзамене разрешается пользоваться своими отчетами по лабораторным работам для решения экзаменационной задачи.

Составители: Зори Сергей Анатольевич, д.т.н., проф. кафедры программной инженерии ДонНТУ

ЛАБОРАТОРНАЯ РАБОТА 1

Представление информации в компьютере - системы счисления, перевод чисел из одной системы в другую.

Цель работы: изучение организации способов представления информации в компьютере.

Задания к лабораторной работе.

1. Перевести десятичные числа a, b, c в 2-ю, 8-ю, 16-ю системы счисления (с.с.) без использования промежуточных систем счисления. Для контроля правильности выполнить обратный перевод.
2. Перевести десятичное число x в двоичную систему счисления, используя в качестве промежуточной:
   а) 8-ю с.с.;
   б) 16-ю с.с.
3. Перевести 16-е число z в 10-ю, 8-ю, 2-ю с.с.

Исходные данные приведены в таблице 1.

Таблица 1

N	a	b	c	x	z
1.	4589	0.622	8634.56	399.77	D7.3B
2.	4577	0.464	9456.68	8945.9	3A.43
3.	4998	0.547	6345.67	9111.9	89.DF
4.	3456	0.534	9213.46	423.67	DE.45
5.	9232	0.345	9238.84	12.673	A5.FE
6.	3421	0.979	352.566	409.86	9F.6E
7.	3451	0.778	72.7085	564.69	D0.A7
8.	9236	0.779	858.922	456.88	54.28
9.	4556	0.897	869.999	890.98	4D.AF
10.	8778	0.649	767.908	678.89	DF.5E
11.	3219	0.921	9621.67	342.86	D9.67
12.	3493	0.723	162.245	95.853	4E.C8
13.	4034	0.556	835.083	67.368	75.EF
14.	9345	0.567	062.545	5673.9	ED.DE
15.	8869	0.732	954.786	50.898	A6.8F
16.	1923	0.789	845.663	566.43	E9.75
17.	4576	0.567	143.541	56.543	DE.A6
18.	4418	0.521	3766.32	455.64	8A.6B
19.	2345	0.478	847.435	56.754	96.D3
20.	7320	0.569	9432.89	43.439	DE.55
21.	3455	0.732	6467.76	55.568	DF.8F
22.	6889	0.779	846.907	677.08	DA.E8
23.	8468	0.888	967.769	6.4322	DE.56
24.	4567	0.234	957.907	566.75	67.AD
25.	9937	0.328	834.457	566.45	84.E5
26.	8875	0.345	467.874	36.566	E5.4A
27.	9344	0.768	737.459	778.87	66.AE
28.	1455	0.532	2467.76	585.68	EF.1F
29.	6219	0.479	986.207	67.648	D4.EA
30.	8345	0.812	917.219	86.432	CE.A6

ЛАБОРАТОРНАЯ РАБОТА 2

Логика и арифметика обработки двоичных чисел в компьютере

Цель работы: изучение алгоритмов выполнения арифметических и логических операций в компьютере.

Задания к лабораторной работе:

1. Используя данные таблицы 2, найти для модулей чисел a и b: $a \wedge b$, $a \vee b$, $a \oplus b$, $a \wedge \bar{a}$, $\bar{b} \vee b$, если длина разрядной сетки – L двоичных разрядов.

2. Используя данные таблицы 2 , найти алгебраические суммы a+b, a-b, -a+b, -a-b в обратном и дополнительном кодах, если длина разрядной сетки равна L двоичных разрядов. Сделать выводы о наличии или отсутствии переполнения.

Проверить результат переводом в десятичную систему счисления.

Оценить погрешность результатов.

3. Используя данные табл. 2 найти алгебраические суммы x-y, x+y, -x+y, -x-y в модифицированных обратном и дополнительном кодах, если длина разрядной сетки - L двоичных разрядов. Сделать выводы о наличии или отсутствии переполнения.

4. Оценить диапазон представления чисел с плавающей запятой с нормализованной и ненормализованной мантиссой, если для представления мантиссы отведено N двоичных разрядов, для порядка – M разрядов, основание характеристики равно S.

Исходные данные - в табл. 2.

Таблица 2

N п/п	a	b	L	x	y	N	M	S
1	-0.37	0.68	9	0.15	0.96	6	5	2
2	0.83	-0.19	7	0.17	0.84	9	6	4
3	-0.27	0.99	8	0.35	0.88	10	7	8
4	0.95	-0.11	9	0.33	0.79	12	8	16
5	-0.35	0.90	7	0.47	0.69	14	9	16
6	0.79	-0.30	8	0.77	0.63	16	10	8
7	-0.07	0.97	9	0.81	0.45	18	5	4
8	0.81	0.29	7	0.56	0.99	20	6	2
9	-0.75	0.65	8	0.68	0.37	22	7	2
10	0.81	-0.26	9	0.19	0.83	24	8	4
11	-0.69	0.57	7	0.99	0.27	28	9	8
12	-0.45	0.72	8	0.11	0.95	30	10	16
13	0.93	-0.94	9	0.90	0.35	32	5	16
14	-0.81	0.59	7	0.30	0.79	36	6	8
15	0.97	-0.94	8	0.97	0.07	8	7	4
16	0.77	-0.79	9	0.28	0.82	9	8	2
17	-0.43	0.82	7	0.65	0.75	10	9	2
18	0.89	-0.16	8	0.26	0.81	11	10	4
19	-0.95	0.23	9	0.57	0.69	13	5	8
20	0.74	-0.39	7	0.77	0.79	15	6	16
21	-0.66	0.73	8	0.72	0.45	17	7	16
22	0.64	-0.85	9	0.94	0.93	19	8	8
23	-0.96	0.15	7	0.59	0.81	21	9	4
24	0.84	-0.17	8	0.64	0.97	23	10	2
25	-0.88	0.35	9	0.82	0.43	25	11	2
26	0.89	-0.40	7	0.98	0.08	21	11	4
27	-0.75	0.64	8	0.77	0.56	20	9	8
28	-0.74	0.49	9	0.35	0.79	30	6	8
29	0.97	-0.84	8	0.97	0.21	18	7	4
30	0.67	-0.59	9	0.29	0.62	19	8	2

ЛАБОРАТОРНАЯ РАБОТА 3

Ассемблер. Программирование разветвляющихся процессов.

Цель работы: изучение программирования разветвляющихся алгоритмов на Ассемблере.

Требования к программам на Ассемблере (для всех лаб. работ):

1. *Начальные значения счетчиков должны инициализироваться программой.*
2. *Результат вычислений должен сохраняться в переменных и быть выведен на экран либо доступен для просмотра в отладчике.*
3. *Сохранение и восстановление всех регистров, используемых подпрограммами.*
4. *Передача параметров в подпрограммы должна выполняться через стек и/или регистры. Использование в подпрограммах имен переменных из главной программы запрещено.*
5. *Программа должна выполняться корректно при любых входных данных, которые допустимы согласно условию задания.*
6. *Исходный код должен содержать комментарии.*

Отчет должен содержать:

1. *Титульный лист*
2. *Задание*
3. *Исходный код программы*
4. *Листинг, сгенерированный компилятором с двоичными кодами команд.*
5. *Расчет результата для всех необходимых наборов входных данных (должны быть проверены <u>все</u> логические ветки программы), выполненный вручную.*
6. *- Скриншоты вывода результатов для этих примеров входных данных.*

Задания к лабораторной работе:

1. Выполнить программирование на языке ассемблера заданных ниже согласно номера варианта вычислений с логическими ветвлениями. В заданиях к лабораторной работе значения исходных данных выбрать самостоятельно, исходя из того, что при выполнении программы должна быть проверена каждая возможная ветвь ветвлений.

2. Достоверность каждого решения проверить путем сравнения с контрольным просчетом.

3. В отчете представить исходный текст программы и листинг, сгенерированный компилятором с двоичными кодами команд, скриншоты выполнения программы для тестовых наборов, а также результаты контрольных просчетов.

Варианты заданий:

1.
```
        (x+y)**2/(4-x),          если x >= 3*y
z = {
        (7x + y)/(2 -5*x*y),     иначе
```

2.
```
         x**2+5  ,               если x > 2
y = {  (x + t )**3/(2*x-7),      если x = 2
         7x - 15 ,               если x < 2
```

```
            1+x+x**2+x**3/3,       если x < 20
3.    y = {
            (x+10)**2/(x-3),       если  x >= 20
             (a+b)/a+23b*a**3, если a > b
4.    y = {  (a-b)**4         , если a < b
              a               , если a = b

            abs((x*x)/(x+2)),  если  x <  0
5.    z = { x-y/2            ,  если  x >  0, y <  2
            (x+y)**3          ,  если  x >= 0, y >= 2


            xy**2           , если abs(y)  < x**2
6.    z = { 1+x             , если abs (y) > x**2
             y+x+x**2+x**3 , если abs (y) = x**2


            a*x+2/b  , если 0 < x <= 10, b /= 0
7.    y = {
            ax**3+2/b, если x < 0, x > 10, b /= 0


            t       ,если abs (t) < 1 или  2 < t < 4
8.    z = { t**2+1  ,если 4 < t <= 11
            1       ,в ост.сл.


            min(2a,0,b)    ,если a <= 1
9.    x = { a/5            ,если 5 < a < 25
            b+a            ,в ост.сл.


10.   t=(abs(x)+ min(x,y,z))/(abs(y)+ max(x,y,z))


11.   u=min(x+z,max(y-x,z*x))


            ax + by , если  c <=  ax+by <= d
12.   u = { x+y     , если  ax+by < c
            1-x-y   , иначе


13.   Заданы значения a,b,c.Найти max, min.


14.   z=max(min(a+b,b-2*c,a+c-3*b),(a+2*b)**3)


15.   z= abs(min(a-2*b,(b+16*c)/a,c**3-3*a*b))


                  (5*x+23*y)/(x-y),  если x > 0,y > 0
16.   z = 5xy+ {
                   x**2+y**3,        иначе
```

```
            1+x   , если min(x,y,z) = x
17.    k = { 2+x   , если min(x,y,z) = y
            3+x   , если min(x,y,z) = z
            -5x,     если abs(x) < 10
18.    y = { -3x**2,  если x <= 20
            -1200/x, в ост. сл.


                        ac  , если a > 0, c > 0
19.   y = abs(a-c) + { 0   , если a = 0, c = 0
                        1   , иначе


20.   z = max(a+b,12*b,c**3/(a-b))+ abs(a-b-c)


                         x**2/(7+z) ,если abs(x+z) < 100
21.   u = min(x,y,z) + {
                         12-x*z     , в ост.сл.


22.   z = max(x+y,y-2*x,min(5*x, y**3/(2-x), z+5))


23.   t = min(120*x/(y+x),y-3*x**2, max(x+12*z,y-2*x,z))


           abs(3*a+b**3-c/4),          если a> 0,b> 3,c> 20
24.   z = {
           min(20*a*b,(c-a*b)**4/(a+b)), иначе


             (4*x/y)**3     , если x > y
25.   z = {
              abs(34*x-y**2), иначе


26.   s = min(4a*b,(b+c)**3/(a-c*b),9*c)+ abs(a-2b)


27.   u = max(49*x,y**3,z) / min(20*x/(y-z),-25*x+c)


                            (x**3+y)/(12-y), если x>0,y>0
28.   u= abs( min(x,y,z))+ {
                            10+ 3*x + 17*x*y**2, иначе


                             (x**3+y)/(12-y*y),если x<0,y<0
29.  u = max(49*x,49*x+y) + {
                             10+ 3*x + 27*y*x**2, иначе


           (24*y/y+x)**3      , если x > y
30.   z = {
            abs(14*x**2+y**2) , иначе
```

ЛАБОРАТОРНАЯ РАБОТА 4

Ассемблер. Способы адресации данных. Программирование циклических процессов и работа с массивами.

Цель работы: изучение программирования циклов и способов адресации данных в памяти компьютера на Ассемблере.

Задание к лабораторной работе:

1. Составить на языке ассемблера программу, выполняющую заданные вычисления. При этом:

- массив элементов данных – размерность DD;
- в программе выполнять проверку деления на ноль;
- для выполнения накапливающих операций (или других необходимых вычислений) использовать подпрограмму.

2. Составить программу просчета задания на языке высокого уровня (любом) и убедиться, что результат совпадает с результатом, полученным на ассемблерной программе (в программе проверки учесть, что все вычисления на ассемблере – целочисленные!).

3. В отчете представить листинги программ и скриншоты их выполнения для тестового набора данных.

Варианты заданий:

1. $Y[k] = 4*X[k]/X[2n-k]+k*k*(X[n+1-k]**2)/(k+1)*X[k+1]$ $N=6 \quad k=1,2,...6$

2. $Y[k] = A[2k-1]*X[k]**2+A[2k]*X[k]+k*A[N+1-k]$ $N=4 \quad k=1,2,3.4$

3. $Y = \sum_{k=0}^{N} \{ k*A[k]/(k+B[k+1])+A[2k-1]/B[N+1-k] \}$ $N=4$

4. $Y[k] = 4*X[k]/X[2N-k] + k**2*X[N+1-k]/(k+1)/X[k+1]$ $N=5$ $k=1,2,...5$

5. $Z = \sum_{k=0}^{N} \{ k*X[k+1]/(k+1)-X[2k+1]*X[N+1-k]\}$ $N=4$

6. $Z[k] = (k+1)**2*X[k+1]**2+(N+1-k)**2*X[N+1-k]**2$ $N=6$ $k=1,2,...6$

7. $Z = \prod_{k=0}^{N} \{((X[k]**2+k)/(k+1) + X[N+1-k]**2\}$ $N=5$

8. $Y[k] = (k*A[k]**3-A[k+1]**2 + A[k+2])/ A[N+1-k]$ $N=6$ $k=1,2,...6$

9. $Z= \prod_{k=0}^{N} \{ (k+1)*A[k+1]**2- A[N+1-k]**2/(k+1) \}$ $N=5$

10. $Z = \prod_{k=0}^{N} \{ (k+1)*X[N-k] - X[N+1-k]/(k+1+X[k]) \}$ $N=4$

11. Z[k] = (B[k+1]*A[N+1-k])/(k+A[2k-1]*B[k]) N=5
k=1,2,...5

12. Y[k] = k*X[k]**2/M[k+1]+M[2k-1]*X[N+1-k] N=5
k=1,2,...5

13. $Z = \sum_{k=1}^{N} \{ (k-1)*X[k+1]**2/(2k+X[2N-k])+X[2k-1] \}$ k=1 N=4

14. $Z = \prod_{k=1}^{N} \{ k*(X[k]+k)/ X[N+1-k] + X[2k-2] \}$ k=1 N=5

15. Z[k] = (B[k]+k)*A[k]**2/(k+A[2k-1])+A[2N+2-2k] N=5 k=1,2,...5

16. $Y = \sum_{k=1}^{N} \{ k**2*A[k]**2 - A[2k]/(A[N+1-k]-B[k+1]) \}$ k=1 N=6

17. $Z = \sum_{k=1}^{N} \{ A[k]*B[N+1-k] + k**2*A[2k-1]*B[2k] \}$ k=1 N=4

18. Z[k] = k*M[k]-abs(A[k])/A[N+1-k]/M[N+2-k] N=5
k=1,2,...5

19. $Z = \sum_{k=1}^{N} \{ A[k]*B[2k+1]-A[N+1-k]/B[2N+2-2k] \}$ k=1 N=4

20. $V = \sum_{k=1}^{N} \{ k*A[k]**k+A[N+1-k]*B[k] \}$ k=1 N=5

21. Y[k] = (A[k]**2+k)*(A[2k+1]+A[2N+2-2k])
N=5 k=1,2,...5

22. Y[k] = A[k+1]*abs(B[[2k+1])+A[k]*B[2k]/B[N+1-k] N=4
k=1,2,...4

23. Z[k] = X[k]**2/A[2k-1]-abs(k*A[2k]/X[N+1-k]) N=5 k=1,2,...5

24. Y[k] = (A[k]-M[k+1])*(k*A[N+1-k]-M[k])*A[2k] N=6 k=1,2,...6

25. Y[k] = (k+1)*A[k]*B[N+1-k]+A[k+1]*(B[k]-A[2k]) N=5
k=1,2,...6

26. Z[k] = A[k]**2/(k+A[k+1])*A[2k+1]/B[2N+1-k] N=4
k=1,2,...4

27. Z[k] = 10*(k+1)**2/B[k]/(k+B[N+1-k])*(B[2k]-B[2k-1]) N=6 k=1,2,...6

28. Y[k] = A[N+k-1]*X[k]+B[2k+1]+A[N+k]/X[2N-k+2] N=5 k=1,2,...5

29. $Z = \sum_{k=1}^{N} \{ A[N+1-k]*X[k]+A[k]/X[2N+2-2k] \}$ k=1 N=5

30. Y[k] = (A[2k+1]-A[k])/X[2*(N-k)]+B[N+1-k]*X[2k] N=5
k=1,2,...5

Примечание:
Здесь S - сумма, а П - произведение, ** - степень

ЛАБОРАТОРНАЯ РАБОТА 5

Ассемблер. Подпрограммы. Способы адресации данных при обработке и вычислениях с матрицами

Цель работы: изучение представления и способов адресации данных при программировании вычислений и обработки матриц с помощью процедур на Ассемблере.

Задание к лабораторной работе:

1. Написать подпрограмму, выполняющую заданное действие над матрицей.

Подпрограмму оформить в стандарте MASM Win-32 и организовать ее выполнение в любой соответствующей стандарту среде.

Подпрограмма должна получать адрес и размеры матрицы (произвольные) из основной программы и возвращать результат, указанный в задании, который затем может быть просмотрен тем или иным способом.

Размеры матрицы задавать переменными, программа должна корректно работать при любых размерах матрицы от 1x1 до 100x100.

Для доступа к элементам матрицы обязательно использовать адресацию с масштабированием и (или) индексированием.

Столбцы и строки нумеруются с 0.

Подпрограмма не должна обращаться напрямую к глобальным переменным.

В случае, если подпрограмма вычисляет массив значений, его адрес должен передаваться выходным параметром (указателем).

2. Написать программу на произвольном языке высокого уровня, выполняющую заданное вариантом действие над матрицей, и сравнить результаты ее работы с ассемблерной программой.

3. В отчете представить листинги программ и скриншоты их выполнения.

Варианты заданий:

1. Подпрограмма суммирования слов с четными номерами в нечетных строках.
2. Подпрограмма суммирования двойных слов с нечетными номерами в нечетных столбцах.
3. Подпрограмма суммирования четных двойных слов в четных строках.
4. Подпрограмма суммирования слов, делящихся на 3, в четных столбцах.
5. Подпрограмма суммирования байт, не лежащих на побочной диагонали.
6. Подпрограмма нахождения минимального элемента среди тех, которые не лежат на побочной диагонали (элемент – двойное слово).
7. Подпрограмма подсчета байт с четным числом единиц в нечетных столбцах.
8. Подпрограмма подсчета слов с нечетным числом единиц в четных строках.
9. Подпрограмма суммирования слов в четных столбцах, содержащих в разрядах 1, 5, 11 и 13 код 0101.
10. Подпрограмма суммирования байт нечетных строках, содержащих в разрядах 2-5 код 1011.
11. Подпрограмма суммирования элементов главной диагонали матрицы двойных слов, являющихся нечетными числами.
12. Подпрограмма суммирования элементов побочной диагонали матрицы слов, являющихся четными числами.
13. Подпрограмма вычисления сумм по строкам матрицы (элемент - слово).

14.Подпрограмма вычисления сумм по столбцам матрицы (элемент - байт).

15.Подпрограмма нахождения минимального элемента матрицы двойных слов, исключая угловые элементы.

16.Подпрограмма поиска минимального элемента среди максимальных в строках матрицы (элемент – двойное слово).

17.Подпрограмма поиска максимального элемента среди минимальных в столбцах матрицы (элемент – байт).

18.Подпрограмма транспонирования квадратной матрицы, где каждый элемент - байт.

19.Подпрограмма суммирования минимальных элементов в строках матрицы (элемент – байт).

20.Подпрограмма суммирования максимальных элементов в столбцах матрицы (элемент – слово).

21. Подпрограмма суммирования всех граничных элементов матрицы байт.

22. Подпрограмма поиска максимального элемента среди всех граничных элементов в матрице двойных слов.

23. Подпрограмма суммирования всех элементов матрицы слов, исключая граничные элементы.

24. Подпрограмма поиска минимального элемента в матрице байт, исключая граничные элементы.

25. Подпрограмма умножения элементов четных столбцов матрицы двойных слов на 2 и деления элементов нечетных столбцов на 2.

26. Подпрограмма поиска минимального элемента среди нечетных строк матрицы байт и максимального элемента среди четных строк.

ЛАБОРАТОРНАЯ РАБОТА 6

Ассемблер. Обработка чисел. Арифметические операции с числами в BCD-формате.

Цель работы: изучение программирования арифметических операций и обработки чисел в BCD-формате на Ассемблере.

Методические указания к лабораторной работе:

ДВОИЧНО-ДЕСЯТИЧНЫЙ ФОРМАТ (BCD)

Пусть в некотором примере деления в ASCII-формате было получено частное 00090204. Если сжать это значение, сохраняя только правые цифры каждого байта, то получим 0924. Такой формат называется двоично- десятичным (BCD - Binary Coded Decimal) (или упакованным). Он содержит только десятичные цифры от 0 до 9. Длина двоично-десятичного представления в два раза меньше ASCII-представления.

Заметим, однако, что десятичное число 0924 имеет основание 10 и, будучи преобразованным в основание 16 (т.е. в шест. представление), даст шест.039C.

1. ПРЕОБРАЗОВАНИЕ ASCII-ФОРМАТА В ДВОИЧНЫЙ ФОРМАТ

Выполнение арифметических операций над числами в ASCII или BCD форматах удобно лишь для коротких полей. В большинстве случаев для арифметических операций используется преобразование в двоичный формат. Практически проще преобразование из ASCII-формата непосредственно в двоичный формат, чем преобразование из ASCII- в BCD-формат и, затем, в двоичный формат:

Метод преобразования базируется на том, что ASCII-формат имеет основание 10, а компьютер выполняет арифметические операции только над числами с основанием 2. Процедура преобразования заключается в следующем:

1. Начинают с самого правого байта числа в ASCII-формате и обрабатывают справа налево.
2. Удаляют тройки из левых шестн. цифр каждого ASCII-байта.
3. Умножают ASCII-цифры на 1, 10, 100 (шестн. - 1, A, 64) и т.д. и складывают результаты.

Для примера рассмотрим преобразование числа 1234 из ASCII-формата в двоичный формат:

	Десятичное	Шестнадцатеричное
4 x 1 =	4	4
3 x 10 =	30	1E
2 x 100 =	200	C8
1 x 1000 =	1000	3E8
Результат (сумма):	1234	04D2

Из этого примера видно, что шестнадцатеричное число 04D2 действительно соответствует десятичному 1234.

2. АРИФМЕТИЧЕСКИЕ ИНСТРУКЦИИ

2.1. Форматы арифметических данных

Арифметические операции процессоров 8086/8088 могут выполняться над операндами 4-х типов (таблица 2.1):

1. Двоичные без знака.
2. Двоичные со знаком (целые).
3. Упакованные десятичные без знака.
4. Распакованные десятичные без знака.

Таблица 2.1. Арифметическая интерпретация 8-битовых чисел.

16-ричное	Битовое двоичное	Без знака десятич-ное	Со знаком десятич-ное	Распак. десятич-ное	Упак. десятич-ное
07 89 C5	00000111 10001001 11000101	7 137 197	+7 -119 -59	7 некорр. некорр.	7 89 некор.

Двоичные числа могут занимать 1 или 2 байта.

Десятичные числа хранятся побайтно по 2 десятичные цифры на байт для упакованного формата или по 1 десятичной цифре на байт для распакованного формата.

Процессор предполагает, что определенные в арифметических инструкциях операнды содержат данные, представляющие корректные для данной инструкции числа. Некорректные данные могут привести к непредсказуемым результатам.

Двоичные числа без знака могут занимать 8 или 16 бит; все биты значимы. Диапазон значений 8-битового числа - от 0 до 255, 16-битового - от 0 до 65535. Над двоичными числами без знака можно выполнять операции сложения, вычитания, умножения и деления.

Двоичные числа со знаком (целые) могут занимать также 8 или 16 бит. Значение старшего бита (самого левого) задает знак числа: 0 - положительное, 1 - отрицательное. Отрицательные числа представляются в дополнительном коде. Поскольку один разряд отведен под знак, диапазон изменения 8-битового числа - от -127 до +127, 16-битового - от - 32768 до +32767. Число 0 имеет положительный знак. Над двоичными числами со знаком могут быть выполнены операции умножения и деления, сложения и вычитания. Для обнаружения переноса в знаковый разряд в результате операции можно использовать инструкции условного перехода.

Упакованные десятичные числа хранятся как беззнаковые байтовые величины. Каждый байт содержит 2 десятичные цифры, занимающие по 4 бита каждая. Цифра в старшем полубайте более значима. В каждом полубайте допустимы только 16-ричные значения от 0 до 9; соответственно пределы изменения десятичного числа - от 0 до 99. Сложение и вычитание таких чисел выполняются в 2 стадии.

1. Сначала применяется обычная беззнаковая двоичная инструкция, которая формирует в регистре AL промежуточный результат.

2. Затем выполняется операция настройки (инструкция DAA или DAS), преобразующая содержимое AL в корректный упакованный десятичный результат.

Умножение и деление упакованных десятичных чисел невозможно.

Распакованные десятичные числа хранятся как беззнаковые байтовые величины. Десятичная цифра располагается в младшем полубайте. Допустимы и интерпретируются как десятичные числа 16-ричные значения от 0 до 9.

Для выполнения операций умножения и деления старший полубайт должен быть заполнен нулями; для сложения и вычитания он может содержать любое значение.

Арифметические операции над распакованными десятичными числами выполняются в 2 стадии.

1. Сначала используются обычные беззнаковые инструкции сложения, вычитания или умножения, которые формируют в регистре AL промежуточный результат.

2. Затем выполняется операция настройки (инструкция AAA, AAS или AAM), преобразующая содержимое AL в результирующее корректное распакованное десятичное число.

Деление выполняется аналогично, за исключением того, что сначала следует настроить числитель в AL (инструкция AAD), а затем выполнить инструкцию беззнакового двоичного деления, результатом которого будет корректное распакованное десятичное число.

Формат десятичных распакованных чисел подобен представлению десятичных цифр в коде ASCII. При этом для числа в коде ASCII старший полубайт содержит 16-ричное значение 3. Возможное содержимое старшего полубайта для распакованного формата приведено выше. Преобразование из одного вида в другой сложности не представляет.

2.2. Сложение

AAA – ASCII-НАСТРОЙКА ДЛЯ СЛОЖЕНИЯ

Эта инструкция преобразует содержимое регистра AL в корректное распакованное десятичное число; старший полубайт обнуляется. AAA модифицирует флаги AF и CF; состояния флагов OF, PF, SF и ZF после AAA не определены.

DAA – ДЕСЯТИЧНАЯ НАСТРОЙКА ДЛЯ СЛОЖЕНИЯ

Эта инструкция корректирует результат предшествующего сложения 2- х правильных упакованных десятичных чисел, содержащийся в регистре AL. Содержимое AL преобразуется в пару корректных упакованных десятичных чисел. DAA модифицирует флаги AF, CF, PF, SF и ZF; состояние флага OF после DAA не определено.

2.3. Вычитание

AAS – ASCII-НАСТРОЙКА ДЛЯ ВЫЧИТАНИЯ

Эта инструкция преобразует находящийся в регистре AL результат предшествующей операции вычитания 2-х корректных десятичных распакованных чисел в корректное десятичное распакованное число, остающееся также в AL. Старший полубайт регистра AL обнуляется. AAS модифицирует флаги AF и CF; состояния флагов OF, PF и ZF после AAS не определены.

DAS – ДЕСЯТИЧНАЯ НАСТРОЙКА ДЛЯ ВЫЧИТАНИЯ

Эта инструкция преобразует находящийся в регистре AL результат предшествующей операции вычитания 2-х корректных десятичных упакованных чисел в пару корректных десятичных упакованных цифр, остающихся также в AL. DAS модифицирует флаги AF, CF, PF, SF и ZF; состояние флага OF не определено.

2.4. Умножение

AAM – ASCII-НАСТРОЙКА ДЛЯ УМНОЖЕНИЯ

Инструкция AAM корректирует результат предшествующей операции умножения 2-х корректных десятичных распакованных операндов. Корректное десятичное распакованное

число, состоящее из 2-х цифр, извлекается из регистров AH и AL, и результат возвращается в регистры AH и AL. Старшие полубайты перемножаемых операндов должны быть обнулены, что необходимо AAM для формирования правильного результата. AAM модифицирует флаги PF, SF и ZF; состояния флагов AF, CF и OF после AAM не определены.

2.5. Деление

AAD – ASCII-НАСТРОЙКА ДЛЯ ДЕЛЕНИЯ

Инструкция AAD модифицирует числитель в регистре AL перед делением 2-х корректных десятичных распакованных операндов таким образом, чтобы частное от деления было также корректным десятичным распакованным числом. Для того, чтобы последующая инструкция DIV сформировала правильный результат, регистр AH должен содержать нули. Частное остается в регистре AL, остаток - в регистре AH; оба старших полубайта обнуляются. AAD модифицирует флаги PF, SF и ZF; состояния флагов AF, CF и OF после AAD не определены.

Задание к лабораторной работе:

Разработать программу на ассемблере, выполняющую арифметические операции над BCD-числами. Обязательным условием является использование команд коррекции арифметических операций.

Результат операции записать в ASCII-строку и вывести на экран.

В отчете необходимо указать содержимое регистров на каждом шаге вычисления результата. Представить листинги программы и скриншоты ее выполнения для заданных операций.

Варианты заданий:

№	A	B	X	Y	Операции*
1	4534	42	8600	09	1, 3, 5
2	272334	4634	947774	05	1, 2, 6
3	18	54234323	633444	02	2, 3, 5
4	362342	5334	9234	04	1, 2, 5
5	32	344	926544	03	2, 3, 5
6	2144	37	3533	09	2, 5, 6
7	51	480989	7234	06	1, 4, 5
8	3689	2788	85	05	1, 3, 6
9	56	3909	8623	09	3, 4, 6
10	183454	54	7643	07	2, 5, 6
11	1934	6234535	9610	04	1, 2, 5
12	33	723534	1643	05	2, 3, 6
13	3478	5590	8334	07	1, 2, 5
14	45	3690	6223	06	2, 3, 6
15	69	1334	95	05	1, 3, 6
16	342344	61	84	06	1, 3, 5
17	76	2645	1445	06	3, 5, 6

18	1823	52	3723	05	1, 4, 6
19	45123	47	8442	06	1, 4, 5
20	201232	5645	94	03	1, 2, 6
21	251234	732345	6465	05	1, 3, 5
22	49	41	84	07	1, 4, 6
23	68	28	96	04	2, 5, 6

* Операции:

1. Сложение двух упакованных BCD-чисел A и B.
2. Сложение двух неупакованных BCD-чисел A и B.
3. Вычитание двух упакованных BCD-чисел A и B.
4. Вычитание двух неупакованных BCD-чисел A и B.
5. Умножение двух неупакованных BCD-чисел X и Y
6. Деление двух неупакованных BCD-чисел X и Y

ЛАБОРАТОРНАЯ РАБОТА 7

Ассемблер. Арифметический сопроцессор

Цель работы: изучение представления и способов обработки данных в арифметическом сопроцессоре при программировании вещественных вычислений на Ассемблере.

Задание к лабораторной работе:

1. Составить программу, выполняющую вычисление вещественной переменной *r* по формуле, указанной для каждого варианта. Арифметические операции и сравнение чисел должны выполняться на арифметическом сопроцессоре. Исходные данные представлены вещественными переменными *a*, *b*, *d* и целой переменной *z*.

Если при некоторых значениях исходных данных значение формулы не определено, является комплексным числом или равно бесконечности (например, из-за деления на ноль), программа должна записать в переменную *r* ноль, корректно завершить работу и вывести соответствующее сообщение (из основной программы). В каждом задании необходимо выполнить как минимум одно сравнение входных или промежуточных данных или проверку флагов исключений сопроцессора.

Если в задании не указано обратного, считать, что углы заданы в радианах.

При выполнении задания минимизировать число операций и обращений к памяти.

В каждой строчке программы, меняющей значение стека регистров сопроцессора, в комментарии необходимо указать текущее состояние регистров сопроцессора (для каждого регистра указать его значение в виде формулы).

Перед выполнением задания необходимо представить исходное выражение в обратной польской нотации (ОПН). Также привести оптимизированную ОПН с использованием команд сопроцессора.

Пример:

- исходное выражение: *b /(sin (a) * d + 1) + a*
- ОПН (один из возможных вариантов): *b a sin d * 1 + / a +*
- оптимизированная ОПН (одни из вариантов): *a sin d * 1 + b FDIVR a +*

В комментарии к каждой строке, меняющей содержимое регистров данных сопроцессора, должно быть приведено содержимое стека сопроцессора после выполнения этой команды (для каждого занятого регистра показано его значение в аналитическом виде).

2. Предусмотреть отработку программой «особых ситуаций» вычислений.

3. Подготовить два контрольных примера, вычисленных вручную (на калькуляторе), которым соответствуют различные значения проверяемых в программе условий.

4. В отчете представить листинг программы и скриншоты их выполнения для тестовых данных с результатами контрольного просчета.

Варианты заданий:

1. $r=\frac{10}{a+b}-(a+b)*2^z$

2. $r=\begin{cases} a^b+4.1*d, & b\in[-1,1] \\ a*b+z/2.4, & \text{иначе} \end{cases}$

3. $r=\begin{cases} a^b, & b>3 \\ a^b+3.7*z, & \text{иначе} \end{cases}$

4. $r=a*\arccos(b)+\frac{\sqrt{b-1}}{a+1}$

5. $r=\begin{cases} (z+b)*\log_2 b, & b>0 \\ b/d-1.3*a, & \text{иначе} \end{cases}$

6. $$r=\frac{z}{x^2*tg\left(\frac{x}{y}\right)+y^2*ctg\left(\frac{x}{y}\right)}$$

7. $$r=z+\left(\frac{x*y+y*\log_2 x}{arctg\left(\frac{x}{y}\right)}\right)$$

8. $$r=|x-y|*\cos\left(\frac{x}{y}+\frac{y}{x}\right)$$

9. $r=x*y+y*\frac{\log_2 x}{arctg\left(\frac{x}{d}\right)}+z$

10. $r=x/y+z*\frac{cos(x+y)}{sin(x-d)}$

11. $r=x^2*tg\left(\frac{x}{y}\right)+y^2*ctg\left(\frac{x}{y}\right)+z$

12. $r=y*x^2*ctg\left(\frac{x}{y}\right)+x*y^2*z*tg\left(\frac{x}{y}\right)+z$

13. $r=\sin(y)*x^3*z+\cos(x)*y^2*z+tg\left(\frac{x}{y}\right)$

14. $r=(\sin(y))^2*\sqrt{z}+(\cos(x^2))^2+tg\left(\frac{d}{y}\right)$

15. $r=\sin(y)/\sqrt{z}+(\cos(x+y))^2*z$

16. $r=\frac{(\sin(y))^2}{\sqrt[4]{z}}-\cos(x-y)/y+\log_{10} z$

17. $r=\frac{(\cos(x+y))^2}{10}-\frac{\cos(x-y)}{d}-\log_2 z$

18. $r=\frac{e^z}{5}-\frac{\cos(x/y)}{3}-\log_{10}(z+y)$

19. $r = x^3 * d + |x| * y^3 + \frac{z}{d+2}$
20. $r = \frac{x^3}{d} - |z| * y^3 + \frac{\sqrt{d}}{z+2}$
21. $r = \frac{e^z}{4} - \frac{\sqrt{\cos\left(\frac{y}{z}\right) * (x+y)}}{2} - \ln(z-y)$
22. $r = e^{-z} + \frac{\sqrt{(1-\cos^2(x*z)} + (x-y)}{6} - \ln(x+y)$
23. $r = 1 + \frac{\sqrt{(1-\sin^2(y)} + \sqrt[4]{z}}{\pi} - \ln(z)$
24. $r = \pi * x^3 * tg\left(\frac{x}{y} + \pi\right) - ctg\left(\frac{x}{y}\right) + |1-z|$

ЛАБОРАТОРНАЯ РАБОТА 8

Ассемблер. Возможности параллельных вычислений на многоядерных процессорах (SSE,AVX).

Цель работы: изучение представления и способов параллельной обработки данных инструкциями SSE на Ассемблере.

Задание к лабораторной работе:

1. Используя SSE- команды перестановки, арифметические и редукционные операции, написать программу на ассемблере, производящую необходимые вычисления согласно ниже представленным вариантам.

Продемонстрировать время выполнения SSE-вычислений программой.

2. Запрограммировать выполнение тех же вычислений без использования SSE-вычислений на любом языке высокого уровня.

Продемонстрировать время выполнения вычислений программой.

3. Сравнить результирующие временные характеристики вычислений обоими реализациями.

Варианты 1- 12.

Имеется 1024 числа в памяти машины от 1 до 1024 (массив a).

Используя SSE- команды перестановки и арифметические операции, написать программу на ассемблере, производящую необходимые вычисления массива B согласно ниже представленным вариантам:

1. B[i]=a[2*i]+a[2*i+2]
2. B[i]=a[2*i]*a[2*i+2]
3. B[i]=a[2*i]-a[2*i+2]
4. B[i]=a[2*i]+a[2*i+3]
5. B[i]=a[2*i]*a[2*i+3]
6. B[i]=a[2*i]-a[2*i+3]
7. B[i]=a[2*i]+a[2*i+4]
8. B[i]=a[2*i]*a[2*i+4]
9. B[i]=a[2*i]-a[2*i+4]
10. B[i]=a[2*i]+a[2*i+1]
11. B[i]=a[2*i]*a[2*i+1]
12. B[i]=a[2*i]-a[2*i+1]

Варианты 13- 25.

Пусть в памяти компьютера расположено RGB изображение 32x32 пикселей в виде одномерного построчного массива (R-байт+G-байт+B-байт для каждого пикселя).

13. Выполнить перевод изображения в яркостную серую шкалу (среднее арифметическое значение из исходных R, G, B каждого пикселя записывается в R, G, B результирующего пикселя изображения).

14. Выполнить перевод изображения в яркостную серую шкалу по R- компоненте (значение исходного R каждого пикселя записывается в R, G, B результирующего пикселя изображения).

15. Выполнить перевод изображения в яркостную серую шкалу по G - компоненте (значение исходного G каждого пикселя записывается в R, G, B результирующего пикселя изображения).
16. Выполнить перевод изображения в яркостную серую шкалу по B - компоненте (значение исходного B каждого пикселя записывается в R, G, B результирующего пикселя изображения).
17. Используя редукцию посчитать максимальное значение R- компоненты для изображения.
18. Используя редукцию посчитать максимальное значение B- компоненты для изображения.
19. Используя редукцию посчитать максимальное значение G- компоненты для изображения.
20. Используя редукцию посчитать минимальное значение R- компоненты для изображения.
21. Используя редукцию посчитать минимальное значение B- компоненты для изображения.
22. Используя редукцию посчитать минимальное значение G- компоненты для изображения.
23. Используя редукцию посчитать среднее значение R- компоненты для изображения.
24. Используя редукцию посчитать среднее значение B- компоненты для изображения.
25. Используя редукцию посчитать среднее значение G- компоненты для изображения.

Методические указания к выполнению работы:

Современные многоядерные процессоры фирм Intel, AMD и др. совместимые предоставляют возможность обрабатывать одной командой 4 и более стандартных 32-разрядных операндов.

Таким образом при обработке массива 32-битных чисел:
- за 1 итерацию обрабатывается 4 элемента,
- число элементов кратно 4,
- число итераций в 4 раза меньше длины массива.

Эти возможности связаны с современными аппаратными потоковыми SIMD расширениями SSE (Streaming SIMD Extensions) и AVX (Advanced Vector Extensions).

Рассмотрим примеры ассемблерного кода с комментариями

SSE-команды ассемблера и терминология

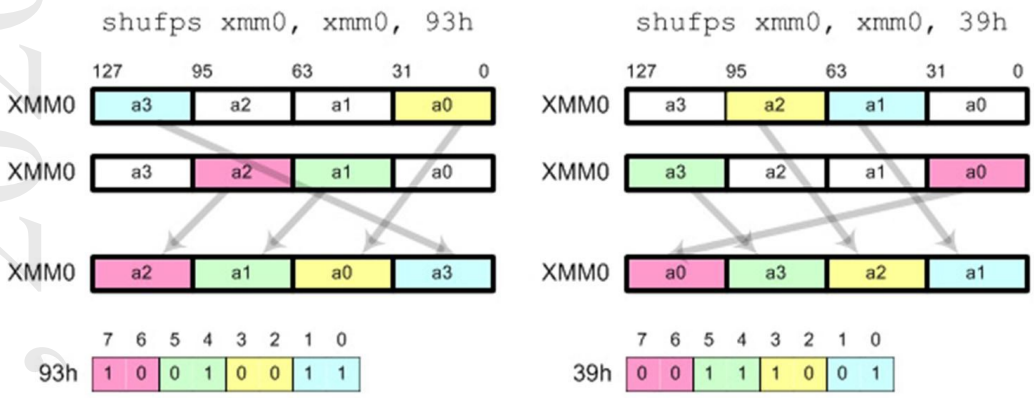
Команды перестановки

SHUFPS dst, src, code

Заполняет младшую часть dst – числами из dst
старшую часть dst – числами из src
Номера чисел в регистрах указаны в code

размер code – 1 байт
номер чисел кодируется парой бит

Пример:



Вертикальные вычисления

– в них операнды – элементы разных векторов с одинаковыми индексам

Не горизонтальная инструкция SSE3

инструкция	результат (A=**dst**, B=**src**)			
	A3	A2	A1	A0
ADDSUBPS (Add Subtract Packed)	A3+B3	A2–B2	A1+B1	A0–B0

Горизонтальные вычисления

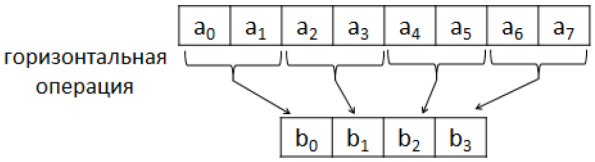
– в них операнды – элементы одного вектора с разными индексами

инструкция	результат (A=**dst**, B=**src**)			
	A3	A2	A1	A0
HADDPS (Horizontal Add Packed)	B2+B3	B0+B1	A2+A3	A0+A1
HSUBPS (Horizontal Subtract Packed)	B2-B3	B0-B1	A2-A3	A0-A1

Горизонтальные вычисления

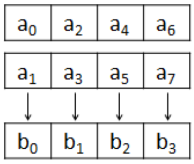
Пример 1.

Вычислить $b_i = a_{2*i} * a_{2*i+1}$



Пример 1

- Нет горизонтальной инструкции умножения
  => расположить данные вертикально (перестановкой)



- За одну итерацию
  – загружать 2 раза по 4 элемента *a*
  – получать 4 элемента *b*

```
…
    xor esi, esi
fori:

    ; загрузка
movaps xmm0, a[2*esi]
movaps xmm4, a[2*esi+16]

    ; xmm0 = a3, a2, a1, a0
    ; xmm4 = a7, a6, a5, a4
```

Пример 1 (продолжение)

```
    ; xmm0 = a3, a2, a1, a0
    ; xmm4 = a7, a6, a5, a4

; подготовка данных 1
 movaps xmm1, xmm0
 shufps xmm1, xmm4, 088h

    ; 88 = 10 00 10 00
    ; xmm1 = a6, a4, a2, a0
```

Пример 1 (продолжение)

```
; вертикальная операция
mulps  xmm1, xmm0
     ; xmm1 = a6*a7, a5*a4, a3*a2, a1*a0

movaps   b[esi], xmm1

add esi, 16
loop fori
```

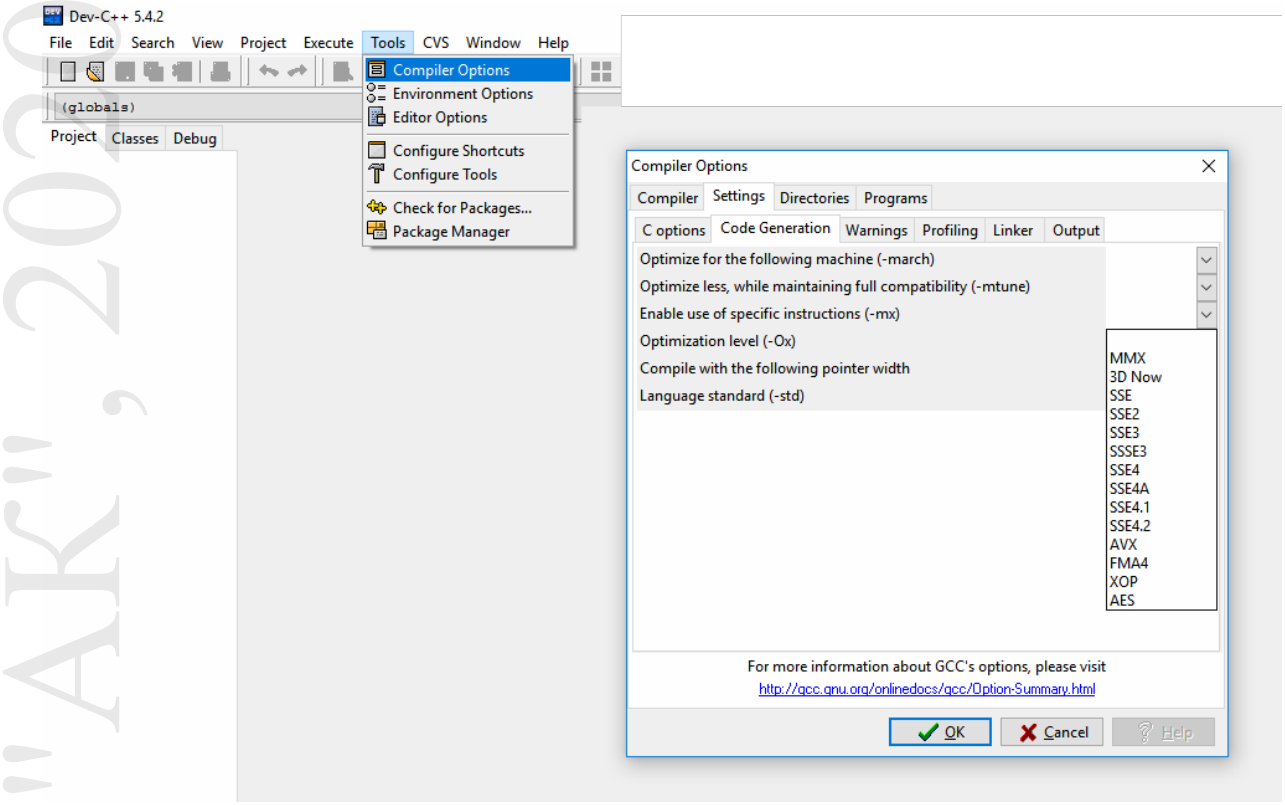
Пример:

Даны массивы ***a, b***

Вычислить $b_i = a_{2*i} + a_{2*i+1}$ с использованием команды **HADDPS**

```
……..
xor esi, esi
fori:
; загрузка
movaps xmm0,   a[2*esi] -
movaps xmm4, a[2*esi+16]
; xmm0 = a3, a2, a1, a0  ; xmm4 = a7, a6, a5, a4
haddps xmm0, xmm4  : одна команда-сложение 4 чисел
; xmm0 = a7+a6, a5+a4, a3+a2, a1+a0
; конец цикла –
movaps b[esi], xmm0
add esi, 16
loop fori
```

Для облегчения работы можно использовать компилятор DEV C++ с выбором соответствующих опций для генерации кода с использованием SSE,AVX команд.



Окно Disassembly (Debug > Windows > Disassembly) показывает команды ассемблер для данного объектного файла.
Код, который написали на C++, показывается черным цветом. Disassembled code показывается серым после соответствующего ему кода на C++/ассемблер.
Окно Disassembly позволяет отлаживать код и осуществлять stepping по нему.
Окно регистров (Debug > Windows > Registers) позволяет посмотреть значение регистров.
Окно памяти (Debug > Windows > Memory) позволяет посмотреть дамп памяти, слева мы видим шестнадцатеричные адрес, справа шеснадцатеричные значения соответствующих ячеек памяти, можно перемещаться, вводя адрес в соответствующее поле