ПРАКТИЧЕСКАЯ РАБОТА № 7.

**Тема: *«РАЗРАБОТКА ПРОГРАММ РАЗВЕТВЛЯЮЩЕЙСЯ СТРУКТУРЫ С ИСПОЛЬЗОВАНИЕМ КОМАНД УСЛОВНЫХ И БЕЗУСЛОВНЫХ ПЕРЕХОДОВ. ВЫЧИСЛЕНИЕ ВЕЛИЧИНЫ СМЕЩЕНИЯ ПРИ ВЫПОЛНЕНИИ ПРЯМЫХ ОТНОСИТЕЛЬНЫХ ПЕРЕХОДОВ».***

**Цель:**

* научиться использовать команды условного и безусловного перехода при составлении программ разветвляющейся структуры на языке ассемблера;
* развить умения по использованию прямого относительного режима адресации памяти;
* научиться вычислять смещение при выполнении прямых относительных переходов;
* научиться кодировать и декодировать команды передачи управления.

**ХОД РАБОТЫ:**

* 1. Повторить приведенный ниже теоретический материал;
  2. вычислить значения операндов для своего варианта;
  3. выполнить указанные задания;
  4. ответить на контрольные вопросы;
  5. подготовить отчет о проделанной работе.

**СТРУКТУРА ОТЧЕТА**

1. Тема работы
2. Цель работы
3. Ответы на вопросы 10, 17, 18, 20 и 21. (*На остальные вопросы отвечать устно.*)
4. Листинг программ (*комментарии обязательны*)
5. Заполнить таблицу, исследуя состояние регистров и содержимое команд переходов при помощи отладчика:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Команда перехода | Код команды | Величина смещения | Состояние IP 1 | Состояние IP 2 |
|  |  |  |  |  |

*Комментарии:*

**IP 1** – состояние регистра Instruction Pointer при выполнении команды, предшествующей команде перехода

**IP 2** – состояние регистра Instruction Pointer при выполнении команды перехода

**ТЕОРЕТИЧЕСКИЙ МАТЕРИАЛ.**

Ассемблер, являясь языком низкого уровня, не содержит операторов ветвления, циклов, не поддерживает автоматического формирования адресов для структур данных, не обеспечивает автоматического выполнения преобразований при вводе-выводе данных. Все перечисленные операции программируются "вручную" с использованием имеющихся команд ассемблера.

***УСЛОВНЫЕ ПЕРЕХОДЫ***

Условные переходы проверяют флаги, которые были установлены предыдущей командой. Условия по каждой мнемонической записи даны ниже в таблице. Термины ***меньше*** и ***больше*** используются для сравнения знаковых целых операндов, а ***выше*** и ***ниже*** - беззнаковых целых.

Если заданное условие выполняется, то происходит переход на место, указанное операндом, в противном случае перехода не происходит, и процессор продолжает работу со следующей команды.

Команды условного перехода используют 8-разрядные непосредственные операнды, которые добавляются к текущему значению указателя команд IP для определения точного адреса перехода***. Переход может осуществляться как вперед, так и назад в диапазоне -128 …+127 байтов.***

В командах перехода непосредственно рассчитывать или определять смещение или сам адрес перехода программисту нет необходимости, программа ассемблера сделает это автоматически и, при необходимости, выдаст предупреждающее сообщение, если это сделать невозможно по тем или иным причинам. При написании программ целесообразно использовать метки, определяющие точку перехода и в тексте команд перехода использовать имена этих меток.

**Jcc rel8** – команда осуществляет условный переход. **сс –** условие, которое необходимо проверить; **Rel8** – относительный адрес в диапазоне от –128 до +127, следовательно, осуществляется переход к команде по адресу **IP+ Rel8**, возникающий при вычислении перенос не учитывается. (Необходимо помнить, что в регистре IP уже находится адрес команды, следующей за Jcc).

Не оказывает влияния на флаги.

|  |  |  |
| --- | --- | --- |
| **Код команды**  ( 2байта ) | **Команда** | **Описание** |
| **77 disp8** | **JA rel8** | Короткий переход, если выше (CF=0 и ZF=0) |
| **73 disp8** | **JAE rel8** | Короткий переход, если выше или равно (CF=0) |
| **72 disp8** | **JB rel8** | Короткий переход, если ниже (CF=1) |
| **76 disp8** | **JBE rel8** | Короткий переход, если ниже или равно (CF=1 или ZF=1) |
| **72 disp8** | **JC rel8** | Короткий переход, если перенос (CF=1) |
| **E3 disp8** | **JCXZ rel8** | Короткий переход, если CX=0 |
| **74 disp8** | **JE rel8** | Короткий переход, если равно (ZF=1) |
| **74 disp8** | **JZ rel8** | Короткий переход, если нуль (ZF=1) |
| **7F disp8** | **JG rel8** | Короткий переход, если больше (ZF=0 и SF=OF) |
| **7D disp8** | **JGE rel8** | Короткий переход, если больше или равно (SF=OF) |
| **7C disp8** | **JL rel8** | Короткий переход, если меньше (SF<>OF) |
| **7E disp8** | **JLE rel8** | Короткий переход, если меньше или равно (ZF=1 или SF<>OF) |
| **76 disp8** | **JNA rel8** | Короткий переход, если не выше (CF=1 или ZF=1) |
| **72 disp8** | **JNAE rel8** | Короткий переход, если не выше и не равно (CF=1) |
| **73 disp8** | **JNB rel8** | Короткий переход, если не ниже (CF=0) |
| **77 disp8** | **JNBE rel8** | Короткий переход, если не ниже и не равно (CF=0 и ZF=0) |
| **73 disp8** | **JNC rel8** | Короткий переход, если нет переноса (CF=0 ) |
| **75 disp8** | **JNE rel8** | Короткий переход, если не равно (ZF=0) |
| **7E disp8** | **JNG rel8** | Короткий переход, если не больше (ZF=1 или SF<>OF) |
| **7C disp8** | **JNGE rel8** | Короткий переход, если не больше и не равно (SF<>OF) |
| **7D disp8** | **JNL rel8** | Короткий переход, если не меньше (SF=OF) |
| **7F disp8** | **JNLE rel8** | Короткий переход, если не меньше и не равно (SF=OF и ZF=0) |
| **71 disp8** | **JNO rel8** | Короткий переход, если нет переполнения (OF=0) |
| **7B disp8** | **JNP rel8** | Короткий переход, если нет четности (PF=0) |
| **79 disp8** | **JNS rel8** | Короткий переход, если положительно (SF=0) |
| **75 disp8** | **JNZ rel8** | Короткий переход, если не нуль (ZF=0) |
| **70 disp8** | **JO rel8** | Короткий переход, если переполнение (OF=1) |
| **7A disp8** | **JP rel8** | Короткий переход, если паритет (PF=1) |
| **7A disp8** | **JPE rel8** | Короткий переход, если сумма битов четная (PF=1) |
| **7B disp8** | **JPO rel8** | Короткий переход, если сумма битов нечетная (PF=0 ) |
| **78 disp8** | **JS rel8** | Короткий переход, если отрицательно (SF=1) |

**Примеры команд:**

CMP CX, 0 ; CX=0 ?

JE L1 ; если да (т.е. если ZF=1), перейти на метку L1

CMP AX, 1000 ; пусть АХ=8000h= 32768 (= - 32768h)

JA L2 ; 32768 > 1000, переход будет, т.к. CF=ZF=0.

CMP AX, 1000h ; пусть АХ=8000h= - 32768

JG L3 ; - 32768 = 1000h, перехода не будет, т.к. ZF=1

Обратите внимание, что перед командами условных переходов обычно используется команда сравнения операндов – CMP. Необходимо помнить, что при сравнении операндов:

А) если операнды равны, то ZF=1, иначе ZF=0;

Б) для беззнаковых операндов:

* если уменьшаемое выше (больше) вычитаемого, то CF=0;
* если уменьшаемое ниже (меньше) вычитаемого, то CF=1 (т.е.осуществляется заем для старшего разряда);

В) для знаковых операндов:

* если уменьшаемое больше вычитаемого, то устанавливаются значения SF=OF=0 или SF=OF=1;
* если уменьшаемое меньше вычитаемого, то SF=0, OF=1 или SF=1, OF=0 (т.о. флаг OF изменяется – устанавливается в 1, если результат сложения 2-х операндов с **одинаковыми знаками** превышает диапазон допустимых значений: для положительных больше **7FFFh,** для отрицательных меньше **8000h**, иначе OF=0);

***БЕЗУСЛОВНЫЕ ПЕРЕХОДЫ***

Абсолютный адрес команды, которая должна быть выполнена следующей, определяется парой **CS:IP**, поэтому выполнение перехода к другой команде программы означает изменение этих регистров, обоих или только одного (IP).

Если изменяется **только IP**, то такой переход называется **внутрисегментным** или ***ближним*** (управление остается в том же сегменте команд), а если меняются оба регистра **CS и IP**, то это **межсегментный** или ***дальний*** переход (начинают выполняться команды из другого сегмента команд).

По способу изменения регистра **IP** переходы делятся на ***абсолютные*** и ***относительные.***

**Если в команде перехода указан адрес (смещение) той команды, которой надо передать управление, то это *абсолютный переход***.

**Если в команде указана величина, которую надо добавить к текущему значению регистра IP, чтобы получился адрес перехода, тогда это будет *относительный переход*, при этом сдвиг может быть положительным или отрицательным (т.е. возможен переход вперед или назад).**

По величине сдвига **относительные переходы** делятся на:

* ***короткие*** - сдвиг задается байтом;
* ***ближние (near) -*** сдвиг задается машинным словом.

**Абсолютные** переходы делятся на:

* ***прямые –*** адрес перехода задается в самой команде;
* ***косвенные –*** в команде указывается регистр или ячейка памяти, в котором (ой) находится адрес перехода.

Все команды безусловного перехода обозначаются одинаково: **JMP *op***, но в зависимости от типа операнда ***op*** ассемблер формирует разные машинные команды. Команда передает управление в указанную точку того же или другого программного сегмента. Не воздействует на флаги CPU. Адрес точки вызова теряется и **вернуться к ней нельзя**!

**1) Прямой короткий (short) переход (внутрисегментный)**:

**JMP <offset8>** ; IP = IP + <offset8>

**Код команды EB.**

**Прямым называется переход**, в команде которого в явной форме указывается метка, на которую нужно перейти. Эта метка должна присутствовать в том же программном сегменте, при этом помеченная ею команда может находиться как до, так и после команды **JMP**. Команда короткого перехода занимает лишь 2 байта памяти: в первом байте записывается код операции **Ebh,** во втором – смещение к точке перехода. <**offset8>** иобозначает смещение размером в байт, которое интерпретируется как знаковое целое от –129 до +127. Команда **прибавляет это число к текущему значению регистра IP**, получая в нем адрес (смещение от начала сегмента команд) той команды, которая должна быть выполнена следующей. Значение регистра СS при этом не меняется. **Помните!!!**, что величина <offset> прибавляется к содержимому регистра IP, хранящего уже адрес команды следующей за JMP. Поэтому, например, команда **JMP 0** – это переход на следующую команду программы.

В командах на ассемблере всегда указывается метка той команды на которую надо передать управление, и ассемблер сам вычисляет сдвиг, который он и записывает в машинную команду. Отсюда следует, что в ассемблере команда перехода по метке воспринимается как **относительный переход (**смещаемся относительно команды, следующей за JMP**).** Имеем **прямую относительную адресацию.**

Итак, по короткому переходу можно передать управление только на ближайшие команды программы – отстоящие от команды, следующей за командой перехода, до 128 байтов назад (-128) или до 127 (+127) байтов вперед.

1. **Прямой ближний (near) или внутрисегментный переход:**

**JMP <offset16>** ; IP = IP + <offset16>

**Код команды E9.**

**<offset16>** обозначает смещение размером в слово, который рассматривается как знаковое целое от –32768 до +32767. Этот переход аналогичен прямому короткому переходу.

Встретив команду перехода с меткой, которой была помечена одна из предыдущих команд программы, **ассемблер вычисляет разность между адресом этой метки и адресом команды перехода** и по этому сдвигу определяет, какой переход – короткий или ближний – надо сформировать.

Но если метка еще не встречалась в тексте программы, т.е. делается переход вперед, тогда ассемблер, не зная еще адреса метки, не может определить, какую именно машинную команду прямого перехода формировать, поэтому он на всякий случай выбирает команду ближнего перехода. Однако эта машинная команда занимает 3 байта, а команда короткого перехода – 2 байта. Поэтому, если известно, что должна выполняться команда короткого перехода, с целью экономии памяти необходимо заранее сообщить об этом ассемблеру, чтобы он сформировал команду короткого перехода. Такое указание дается с помощью оператора SHORT, например: **JMP SHORT L1** (зависит от используемого транслятора). Для переходов назад оператор SHORT не нужен, т.к. уже зная адрес метки, ассемблер сам определит вид команды прямого перехода.

1. **Косвенный ближний (внутрисегментный) переход:**

**JMP Reg16** ; IP = [reg]

или

**JMP Mem16** ; IP = [mem16]

**Код команды FF (расширение в поле reg/OPC – 100).**

Здесь **Reg16** обозначает любой из 16-разрядных регистров AX, BX, CX, DX, SI, DI, SP, BP, а Mem16 – адрес слова памяти, задаваемый одним из режимов адресации. В этом регистре (или слове памяти) должен находиться адрес, по которому и будет произведен переход. Например, по команде **JMP BX** осуществляется переход по адресу, находящемуся в регистре BX.

1. **Прямой дальний (far), или межсегментный переход:**

**JMP seg:ofs** ; CS=seg, IP = ofs

**Код команды EA.**

Здесь **seg** – начало (старшие 16 разрядов начального адреса) сегмента памяти, а **ofs** – смещение в этом сегменте.

В командах эта пара **всегда задается конструкцией** **FAR PTR <метка>**, которая указывает, что надо сделать переход по указанной метке, причем эта метка – "дальняя", т.е. находится в другом сегменте. Ассемблер сам определяет какой нужен сегмент. И сам подставляет в машинную команду его начало, т.е. значение **seg**.

1. **Косвенный дальний (межсегментный) переход:**

**JMP mem32** ; CS=[mem32+2], IP = [mem32]

**Код команды FF (расширение в поле reg/OPC – 101).**

Здесь **mem32** – адрес двойного слова памяти, в котором находится пара **seg:ofs,** задающая абсолютный адрес, по которому данная команда должна выполнить переход. Помните, что в ПК величины размером в двойное слово хранятся в "перевернутом" виде, поэтому смещение **ofs** находится в младшем слове двойного слова mem32, а смещение **seg –** в старшем слове ( по адресу mem32+2).

**Команды межсегментного перехода** используются тогда, когда команды программы размещены не в одном сегменте памяти, а в нескольких (например, команд столько, что они не помещаются в 64 Кбайтах). При переходе из одного такого сегмента в другой необходимо менять не только счетчик команд **IP**, но и содержимое регистра **CS**, загружая в последний начальный адрес второго сегмента. Такое одновременное изменение обоих этих регистров и делают команды межсегментного перехода.

Команда **JMP A** может восприниматься ассемблером неоднозначно: что такое **А** метка или имя переменной? Если **метка** – это должен быть относительный переход, если **имя переменной** – это должен быть косвенный переход внутрисегментный или межсегментный в зависимости от типа переменной. Чтобы избежать неприятностей, желательно конкретно указать, что необходимо выполнить, ***например***:

**JMP SHORT A ;** прямой короткий (внутрисегментный) переход по метке А

**JMP word ptr A ;** косвенный ближний переход, в переменной А находится 16-разрядный адрес, который становится содержимым регистра IP.

**JMP dword ptr A ;** косвенный дальний переход, в 32-разрядной переменной А находится базовый адрес нового сегмента кодов seg (старшее слово) и новое содержимое регистра IP – ofs (младшее слово).

Переход по метке **А,** находящейся в другом сегменте команд, всегда должен указываться с помощью **far ptr**.

**JMP far ptr A ;** прямой дальний переход по метке. Метка А расположена в другом сегменте кодов.

***Например*:**

**1) JMP L1 ;** прямой короткий переход к команде с меткой L1 (определяется смещение)

**2) JMP SHORT L1** ; переход к команде с меткой L1 и явно указывается, что используется прямой короткий переход в пределах – 128 … + 127 байт, используют, если переход вперед.

**3) MOV AX, 1400h**

**JMP AX** ; переход ближний косвенный по содержимому регистра АХ, в IP заносится число 1400h

**4) М1 dw 1400h ;** переменная М1 описана как слово

**JMP M1 ;** эта команда работает так же, как и предыдущая.

**5) MOV M1, offset L1** ; в переменную М1 заносится смещение метки L1

**JMP M1** ; выполняется уже ближний косвенный (а не прямой!) безусловный переход

**6) М2 dd 14002500h** ; описание 32-разрядной переменной

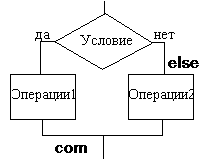
**JMP M2** ; косвенный дальний переход по адресу, записанному в памяти в переменной М2. в IP будет величина 2500h, а в CS - 1400h.

**7) MOV DI, offset LL ;** в регистр DI пересылается адрес метки LL (5Fh)

**JMP [DI]** ; короткий косвенный переход по адресу из **DI.**

**8) JMP far ptr GO ; GO –** метка, расположенная в другом сегменте кода, следовательно, осуществляется прямой дальний переход.

1. **Программирование ветвлений.**



Ветвления программируются с использованием команд условной и безусловной передачи управления.

**cmp ...**

**j**<условие> **else**

**<**операции 1**>**

**jmp com**

**else: <**операции 2**>**

**com: <**продолжение**>**

***Пример 1.***

Вычислить **X=max(A,B)**:

**…**

**mov ax, A**

**cmp ax, B** ; сравнение **A** и **B**

**jl less** ; переход по «меньше»

**mov X, ax**

**jmp continue** ; переход на конец ветвления

**less: mov ax, B**

**mov X, ax**

**continue: ret**

**…**

*Пример 2.*

Определите, равны ли два числа вводимые пользователем с клавиатуры. Определить равенство чисел можно используя вычитание, если разность исследуемых чисел равна 0, то они равны.

***model small***

***.stack 100h***

***.data***

*s1 db 'числа равны$'*

*s2 db 'числа не равны$'*

*const1 = 10*

*const2 = 8*

***.code***

***start:***

***mov ax,@data***

***mov ds,ax***

*mov ax, const1 ; в ax занесли первое число*

*mov dx, const2 ;посылаем в dx второе число*

*sub ax,dx ;сравнили числа*

*jnz m1 ;если получили не 0 результат, то на метку m1*

*mov dx, offset s1 ;иначе выводим строку s1, о том что числа равны.*

*jmp m2*

*m1: mov dx, offset s2 ;числа не равны, выводим строку s2*

*m2: mov ah,09h*

*int 21h ;вывод информационную строку*

***mov ax,4c00h***

***int 21h***

***end start***

Пример 3.

Даны три числа, найти среди них максимальное.

***model small***

***.stack 100h***

***.data***

*s1 db 'максимальное число',10,13,'$'*

*x1 db 34*

*x2 db 56*

*x3 db 45*

***.code***

***start:***

***mov ax,@data***

***mov ds,ax***

*mov dx, offset s1*

*mov ah,09h*

*int 21h ;вывод информационную строку*

*;находим максимальное число*

*mov dl,x1 ;dl:=x1*

*cmp dl,x2 ;сравниваем х1 и х2*

*ja m1 ;если х1>х2, то на m1*

*mov dl,x2 ;иначе dl:=x2*

*m1: cmp dl,x3 ;сравниваем dl и х3*

*ja m2 ;если dl>х3 то на m2*

*mov dl,x3*

*;в dl находится максимальный элемент*

*m2: mov ah,02h*

*int 21h ;выводим максимальный элемент*

***mov ax, 4c00h***

***int 21h***

***end start***

**КОНТРОЛЬНЫЕ ВОПРОСЫ**

1. Объясните различие между командами JMP и JZ.
2. Чем отличаются абсолютные безусловные переходы от относительных?
3. На какие виды делятся абсолютные безусловные переходы?
4. На какие виды делятся относительные безусловные переходы?
5. Какой условный переход использует для проверки содержимое регистра общего назначения?
6. Какое различие между командами JA и JNBE?
7. Какое различие между командами JB и JL?
8. Можно ли при переходе по условию сделать переход на метку в любом месте сегмента?
9. После выполнения команды JB какой флаг должен быть установлен?
10. Предположим, что команда CMP сравнивает два операнда. Для каждой из команд условного перехода из таблицы 4 определено состояние 4-х флагов. Отметьте в последнем столбце будет ли осуществляться переход.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Команда** | **OF** | **SF** | **ZF** | **CF** | **Переход** |
| 1. JNZ | 0 | 0 | 1 | 0 |  |
| 1. JA | 1 | 0 | 1 | 0 |  |
| 1. JNB | 1 | 0 | 1 | 0 |  |
| 1. JBE | 0 | 0 | 1 | 0 |  |
| 1. JOE | 1 | 1 | 0 | 0 |  |
| 1. JNLE | 0 | 1 | 0 | 1 |  |
| 1. JNS | 0 | 0 | 1 | 0 |  |
| 1. JNG | 1 | 1 | 0 | 1 |  |
| 1. JE | 1 | 0 | 1 | 0 |  |
| 1. JNAE | 1 | 0 | 1 | 0 |  |

1. Для чего нужен префикс PTR?
2. В чем отличие команд MOV AX, OFFSET MASS и LEA AX, MASS?
3. В чем отличие команд MOV AX, BX и MOV AX, [BX]?
4. В чем отличие команд MOV AX, [BP] и MOV AX, [BX]?
5. В чем отличие команд MOV AX, [BX+2] и MOV AX, [BX]+2?
6. В чем отличие команд MOV AX, [BX][SI] и MOV AX, [SI][BX]?
7. Как организовать межсегментную передачу управления?
8. Напишите фрагмент программы условного перехода к метке, лежащей от самого перехода на расстоянии 259 байт.
9. Даны два числа. Записать в регистр SI меньшее, а в регистр DI - большее из них.
10. Дано целое число. Если оно чётное, то записать в регистр DI значение FFFFh, если нечётное - FF00h.

**ПРАКТИЧЕСКИЕ ЗАДАНИЯ**

**Практическое задание 1.**

Записать машинные коды всех команд, их 16-ричное представление, определить будет ли выполняться переход.

CMP AL, FFh ; (AL) → -1

JG lable1 ; смещение метки 20h

SUB DX, 0C56Fh ; (DX) → 4470h

JA lable2 ; смещение метки 5Ah

CMP SI, BX ; (SI) → B200h; (BX) → 0050h

JL lable3 ; смещение метки BDh.

ADD CL, 57h ; (CL) → 32h

JP lable4 ; смещение метки 125.

OR DL, 77h ; (DL) → 1Fh

JS lable5 ; смещение метки - 95.

XOR CX, CX ; (CX) → 8A6Bh

JZ lable6 ; смещение метки 68h.

ADD AX, 03FFh ; (AX) → FD00h

JNO lable7 ; смещение метки -120.

INC BH ; (BH) → FFh

JNZ lable7 ; смещение метки -54.

***Пример выполнения:***

CMP DX, 0057h ; (DX) → 1200h

JG lable1 ; 1200h > 0057h, OF=SF=0 and ZF=0, сл., переход будет.

Поясним, как определить флаги OF и SF:

- сведем операцию вычитания к операции сложения 1200h – 0057h = 1200h + ( - 0057h);

- найдем дополнительный код числа ( - 0057h). Он равен 10000h – 0057h = FFA9h;

- сложим оба числа: 1200h + FFA9h = 1)11A9h – отделили единицу переноса ;

- OF=0, т.к. складывали числа с разными знаками, SF=0, т.к. число положительное.

Машинный код команды: 0111 1111 0101 1010 ; команда занимает 2 байта

16-ричный код команды: 7F5Ah, т.к. смещение метки lable1 равно 5Ah.

**Практическое задание 2.**

Записать машинные коды всех команд, их 16-ричное представление, определить какой тип безусловного перехода будет выполняться.

**.data**

B1 db 40h

W1 dw 1234h

DW1 dd 12345678h

W2 dw ?

|  |  |  |
| --- | --- | --- |
| JMP short L1  INC BX  L1: ADD AX, 1 | JMP DW1 | MOV DI, -560  JMP DI |
| MOV AX, W1  JMP AX | JMP word ptr DW1 | JMP B1 |
| MOV W2, offset L1  JMP W2 | MOV SI, 0A6h  JMP SI | JMP far ptr DL1  ;(CS)→88A0h, offset DL1→12h |
| JMP W1 | JMP word ptr [SI]  ; [SI]→ 2208h | L2: SUB CX, 2  MOV [BP][DI], CL  JMP L2 |

**Практическое задание 3.**

Разработать программу реализации циклического процесса в соответствии с вариантом задания. Вычисляется выражение типа

⎧ y1, x<0

Y = ⎨ y2 , x=0

* y3 , x>0

при этом используется целочисленное деление, а параметром цикла является переменная x

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Вариант | y1 | y2 | y3 | a | b | c | d | x |
| 1 | (a2+x)  c | a+b  d | 3ax | 12 | -3 | 4 | 3 | [-3;4] |
| 2 | 2ax+5 | a-b  d | a2-x  c+d | -6 | 4 | 8 | 2 | [-2;5] |
| 3 | x(c+7) | b-d  a | cx2  c+d | 5 | -6 | 2 | 1 | [-4;5] |
| 4 | (d2+x)bc | c+b  d | x2  a+b | 1 | -3 | 12 | 4 | [-3;4] |
| 5 | cx  a | -b2d | (b+c)x | 2 | -7 | 11 | 5 | [-5;2] |
| 6 | b+cx  a | -(5+bc) | d2x  c+3 | 4 | -3 | 6 | 10 | [-4;4] |
| 7 | ab-c  x | d-ab | b2x  c+d | 14 | 2 | 1 | -6 | [-6;2] |
| 8 | a2-x  c | bd-a2 | (a2+x)b | 5 | -3 | 4 | 16 | [-2;6] |
| 9 | c2 -x  a-c | a+b  d | 4ax | 12 | -3 | 4 | 3 | [-3;4] |
| 10 | 2a+5x | a-b  d | a2+x  c-d | -6 | 4 | 8 | 2 | [-3;3] |
| 11 | x(c+5) | 2b-d  2a | x2  c+d | 5 | -6 | 2 | 1 | [-4;5] |
| 12 | (d2+x)b | c+b  d | x2  a-b | 1 | -3 | 12 | 4 | [-3;4] |
| 13 | c+x  a | -b2d | (b+c)-x | 2 | -7 | 11 | 5 | [-5;2] |
| 14 | b+x  a | (5+bc) | d2x  c+3 | 4 | -3 | 6 | 10 | [-4;4] |