Лабораторная работа № 4
Использование команд условного перехода

Цель работы

Научиться выполнять операции с помощью команд условного перехода и безусловного перехода.

Порядок выполнения работы

1. Создать рабочую папку для текстов программ на ассемблере и записать в нее файлы tasm.exe, tlink.exe, rtm.exe и td.exe. из пакета tasm, а также файл с исходным текстом программы на ассемблере, который сохранить с именем prog4.asm.
2. Составить программу, использующую команды условных переходов, арифметических команды над целыми переменными и константами.
3. В центре чистого экрана: сформировать меню отражающее суть задания и предлагающее пользователь ввести символ 1,2 или 3 для расчета соответствующего выражения при заданных значениях в сегменте данных переменных a и b.
4. Протестировать программу, используя **однозначные, двухзначные и трёхзначные числа.**

Содержание отчета:

1. Цель работы.
2. Постановка задачи.
3. Блок схема.
4. Листинг программы.
5. Результат работы программы для вышеуказанных тестовых случаев.
6. Вывод.

Теоретическая часть

CMP (CoMPare operands)

Сравнение операндов

```
------------------------------------
          cmp   операнд1, операнд2
------------------------------------
```

Назначение:
Сравнение двух операндов.

Алгоритм работы:

- выполнить вычитание (операнд1-операнд2), при этом ни один из операндов не модифицируется;
- в зависимости от результата установить флаги, операнд 1 и операнд 2 не изменять (то есть результат не запоминать).

Состояние флагов после выполнения команды:

11	07	06	04	02	00
OF	SF	ZF	AF	PF	CF

Применение:

Данная команда используется для сравнения двух операндов методом вычитания, при этом операнды не изменяются. По результатам выполнения команды устанавливаются флаги. Команда cmp применяется с командами условного перехода и командой установки байта по значению setcc.

```
len     equ     10
...
        cmp     ax,len
        jne     m1      ;переход если (ax)<>len
              jmp     m2      ;переход если (ax)=len
```

JCC/JCXZ/JECXZ(Jump if condition)
(Jump if CX=Zero/ Jump if ECX=Zero)
Переход, если выполнено условие
Переход, если CX/ECX равен нулю

Схема команды:	jcc метка jcxz метка jecxz метка

Назначение: переход внутри текущего сегмента команд в зависимости от некоторого условия.

Алгоритм работы команд (кроме jcxz/jecxz):

Проверка состояния флагов в зависимости от кода операции (оно отражает проверяемое условие):

- если проверяемое условие истинно, то перейти к ячейке, обозначенной операндом;
- если проверяемое условие ложно, то передать управление следующей команде.

Алгоритм работы команды jcxz/jecxz:

Проверка условия равенства нулю содержимого регистра *ecx/cx*:

- если проверяемое условие истинно, то есть содержимое *ecx/cx* равно 0, то перейти к ячейке, обозначенной операндом метка;
- если проверяемое условие ложно, то есть содержимое *ecx/cx* не равно 0, то передать управление следующей за jcxz/jecxz команде программы.

Состояние флагов после выполнения команды:

11	07	06	05	04	03	02	01	00
OF	SF	ZF	0	AF	0	PF	1	CF
?	?	?		r		?		r

Применение (кроме jcxz/jecxz):

Команды условного перехода удобно применять для проверки различных условий, возникающих в ходе выполнения программы. Как известно, многие команды формируют признаки результатов своей работы в регистре eflags/flags. Это обстоятельство и используется командами условного перехода для работы. Ниже приведены перечень команд условного перехода, анализируемые ими флаги и соответствующие им логические условия перехода.

Команда	**Состояние проверяемых флагов**	**Условие перехода**
JA	CF = 0 и ZF = 0	если выше
JAE	CF = 0	если выше или равно
JB	CF = 1	если ниже
JBE	CF = 1 или ZF = 1	если ниже или равно
JC	CF = 1	если перенос
JE	ZF = 1	если равно
JZ	ZF = 1	если 0
JG	ZF = 0 и SF = OF	если больше
JGE	SF = OF	если больше или равно
JL	SF <> OF	если меньше
JLE	ZF=1 или SF <> OF	если меньше или равно
JNA	CF = 1 и ZF = 1	если не выше
JNAE	CF = 1	если не выше или равно
JNB	CF = 0	если не ниже
JNBE	CF=0 и ZF=0	если не ниже или равно
JNC	CF = 0	если нет переноса
JNE	ZF = 0	если не равно
JNG	ZF = 1 или SF <> OF	если не больше
JNGE	SF <> OF	если не больше или равно
JNL	SF = OF	если не меньше
JNLE	ZF=0 и SF=OF	если не меньше или равно
JNO	OF=0	если нет переполнения
JNP	PF = 0	если количество единичных битов результата нечетно (нечетный паритет)
JNS	SF = 0	если знак плюс (знаковый (старший) бит результата равен 0)
JNZ	ZF = 0	если нет нуля
JO	OF = 1	если переполнение
JP	PF = 1	если количество единичных битов результата четно (четный паритет)
JPE	PF = 1	то же, что и JP, то есть четный паритет
JPO	PF = 0	то же, что и JNP
JS	SF = 1	если знак минус (знаковый (старший) бит результата равен 1)
JZ	ZF = 1	если ноль

Логические условия "больше" и "меньше" относятся к сравнениям целочисленных значений со знаком, а "выше и "ниже" — к сравнениям целочисленных значений без знака. Если внимательно посмотреть, то у многих команд можно заметить одинаковые значения флагов для перехода. Это объясняется наличием нескольких ситуаций, которые могут вызвать одинаковое состояние флагов. В этом случае с целью удобства ассемблер допускает несколько различных мнемонических обозначений одной и той же машинной команды условного перехода. Эти команды ассемблера по действию абсолютно равнозначны, так как это одна и та же машинная команда. Изначально в микропроцессоре i8086 команды условного перехода могли осуществлять только короткие переходы в пределах -128...+127 байт, считая от следующей команды. Начиная с микропроцессора i386, эти команды уже могли выполнять любые переходы в пределах текущего сегмента команд. Это стало возможным за счет введения в систему команд микропроцессора дополнительных машинных команд. Для реализации межсегментных переходов необходимо комбинировать команды условного перехода и команду безусловного перехода jmp. При этом можно воспользоваться тем, что практически все команды условного перехода парные, то есть имеют команды, проверяющие обратные условия.

Применение jcxz/jecxz:

Команда	Состояние флагов в eflags/flags	Условие перехода
JCXZ	не влияет	если регистр CX=0
JECXZ	не влияет	если регистр ECX=0

Команду jcxz/jecxz удобно использовать со всеми командами, использующими регистр *ecx/cx* для своей работы. Это команды организации цикла и цепочечные команды. Очень важно отметить то, что команда jcxz/jecxz, в отличие от других команд перехода, может выполнять только близкие переходы в пределах -128...+127 байт, считая от следующей команды. Поэтому для нее особенно актуальна проблема передачи управления далее чем в указанном диапазоне. Для этого можно привлечь команду безусловного перехода jmp. Например, команду jcxz/jecxz можно использовать для предварительной проверки счетчика цикла в регистре cx для обхода цикла, если его счетчик нулевой.

Пример1.

```
...
        jcxz    m1      ;обойти цикл, если cx=0
cycl:
;некоторый цикл
        loop    cycl
m1:     ...
```

JMP(JuMP)

Переход безусловный

```
-------------------------------------
          jmp   метка
-------------------------------------
```

Назначение:

Используется в программе для организации безусловного перехода как внутри текущего сегмента команд, так и за его пределы. При определенных условиях в защищенном режиме работы команда jmp может использоваться для переключения задач.

Алгоритм работы:

Команда jmp в зависимости от типа своего операнда изменяет содержимое либо только одного регистра eip, либо обоих регистров cs и eip:

- если операнд в команде jmp — метка в текущем сегменте команд (a8, 16, 32), то ассемблер формирует машинную команду, операнд которой является значением со знаком, являющимся смещением перехода относительно следующей за jmp команды. При этом виде перехода изменяется только регистр eip/ip;
- если операнд в команде jmp — символический идентификатор ячейки памяти (m16, 32, 48), то ассемблер предполагает, что в ней находится адрес, по которому необходимо передать управление. Этот адрес может быть трех видов:
    - значением абсолютного смещения метки перехода относительно начала сегмента кода. Размер этого смещения может быть 16 или 32 бит в зависимости от режима адресации;
    - дальним указателем на метку перехода в реальном и защищенном режимах, содержащим два компонента адреса — сегментный и смещение. Размеры этих компонентов также зависят от установленного режима адресации (use16 или use32). Если текущим режимом является use16, то адрес сегмента и смещение занимают по 16 бит, причем смещение располагается в младшем слове двойного слова, отводимого под этот полный адрес метки перехода. Если текущим режимом является use32, то адрес сегмента и смещение занимают, соответственно, 16 и 32 бит, — в младшем двойном слове находится смещение, в старшем — адрес сегмента;
    - адресом в одном из 16 или 32-разрядных регистров — этот адрес представляет собой абсолютное смещение метки, на которую необходимо передать управление, относительно начала сегмента команд.

Для понимания различий механизмов перехода в реальном и защищенном режимах нужно помнить следующее. В реальном режиме микропроцессор просто изменяет cs и eip/ip в соответствии с содержимым указателя в памяти. В защищенном режиме микропроцессор предварительно анализирует байт прав доступа AR в дескрипторе, номер которого определяется по содержимому сегментной части указателя. В зависимости от состояния байта AR микропроцессор выполняет либо переход, либо переключение задач.

Состояние флагов после выполнения команды (за исключением случая переключения задач): выполнение команды не влияет на флаги

Применение:

Команду jmp применяют для осуществления ближних и дальних безусловных переходов без сохранения контекста точки перехода.

Варианты

1. $$F=\begin{cases} ax^2+bx+c, & x<0, b\neq 0 \\ \dfrac{x+a}{x-c}, & x>0, b=0 \\ \dfrac{x}{c}, & \text{иначе} \end{cases}$$

2. $$F=\begin{cases} \dfrac{1}{ax}-b, & x+5<0, c=0 \\ \dfrac{x-a}{x}, & x+5>0, c\neq 0 \\ \dfrac{10x}{c-4}, & \text{иначе} \end{cases}$$

3. $$F=\begin{cases} ax^2+bx+c, & a<0, c\neq 0 \\ \dfrac{-a}{x-c}, & a>0, c=0 \\ a(x+c), & \text{иначе} \end{cases}$$

4. $$F=\begin{cases} -ax+b, & c<0, x\neq 0 \\ \dfrac{x-a}{-c}, & c>0, x=0 \\ \dfrac{bx}{c-a}, & \text{иначе} \end{cases}$$

5. $$F=\begin{cases} a-\dfrac{x}{10+b}, & x<0, b\neq 0 \\ \dfrac{x-a}{x-c}, & x>0, b=0 \\ 3x+\dfrac{2}{c}, & \text{иначе} \end{cases}$$

6. $$F=\begin{cases} ax^2+b^2x, & c<0, b\neq 0 \\ \dfrac{x+a}{x+c}, & c>0, b=0 \\ \dfrac{x}{c}, & \text{иначе} \end{cases}$$

7. $$F=\begin{cases} -ax^2-b, & x<5, c\neq 0 \\ \dfrac{x-a}{x}, & x>5, c=0 \\ -\dfrac{x}{c}, & \text{иначе} \end{cases}$$

9. $$F=\begin{cases} ax^2+b^2x, & a<0, x\neq 0 \\ x-\dfrac{a}{x-c}, & a>0, x=0 \\ 1+\dfrac{x}{c}, & \text{иначе} \end{cases}$$

10. $$F=\begin{cases} ax^2-bx+c, & x<3, b\neq 0 \\ \dfrac{x-a}{x-c}, & x>3, b=0 \\ \dfrac{x}{c}, & \text{иначе} \end{cases}$$

11. $$F=\begin{cases} ax^2+bc, & x<1, b\neq 0 \\ \dfrac{x-a}{x-c}, & x>1.5, b=0 \\ \dfrac{x}{c}, & \text{иначе} \end{cases}$$

12. $$F=\begin{cases} ax^3+b^2+c, & x<0.6, b+c\neq 0 \\ \dfrac{x-a}{x-c}, & x>0.6, b+c=0 \\ \dfrac{x}{c}+\dfrac{x}{a}, & \text{иначе} \end{cases}$$

13. $$F=\begin{cases} ax^2+b, & x-1<0, b-x\neq 0 \\ \dfrac{x-a}{x}, & x-2>0, b+x=0 \\ \dfrac{x}{c}, & \text{иначе} \end{cases}$$

14. $$F=\begin{cases} -ax^3-b, & x+c<0, a\neq 0 \\ \dfrac{x-a}{x-c}, & x+c>0, a=0 \\ \dfrac{x}{c}+\dfrac{c}{x}, & \text{иначе} \end{cases}$$

15. $$F=\begin{cases} -ax^2+b, & x<0, b\neq 0 \\ \dfrac{x}{x-c}+5.5, & x>0, b=0 \\ \dfrac{-x}{c}, & \text{иначе} \end{cases}$$

16. $$F=\begin{cases} a(x+c)^2-b, & x=0, b\neq 0 \\ \dfrac{x-a}{-c}, & x>0, b=0 \\ a+\dfrac{x}{c}, & \text{иначе} \end{cases}$$

17. $$F=\begin{cases} ax^2-cx+b, & x+10<0, b\neq 0 \\ \dfrac{x-a}{x-c}, & x+10>0, b=0 \\ \dfrac{-x}{a-c}, & \text{иначе} \end{cases}$$

18. $$F=\begin{cases} ax^3+bx^2, & x<0, b\neq 0 \\ \dfrac{x-a}{x-c}, & x>0, b=0 \\ \dfrac{x+5}{c(x-10)}, & \text{иначе} \end{cases}$$

19. $$F=\begin{cases} a(x+7)^2-b, & x<5, b\neq 0 \\ \dfrac{x-ac}{ax}, & x>5, b=0 \\ \dfrac{x}{c}, & \text{иначе} \end{cases}$$

20. $$F=\begin{cases} -\dfrac{2x-c}{cx-a}, & x<0, b\neq 0 \\ \dfrac{x-a}{x-c}, & x>0, b=0 \\ -\dfrac{x}{c}+\dfrac{-c}{2x}, & \text{иначе} \end{cases}$$