Министерство образования Республики Беларусь
Учреждение образования
«Белорусский государственный университет информатики и радиоэлектроники»
Филиал
«Минский радиотехнический колледж»

Учебная дисциплина «Встраиваемые микропроцессорные системы»

Инструкция

по выполнению лабораторной работы
«Программирование на языке Ассемблер. Изучение команд перехода»

Минск

2017

Лабораторная работа № 10

Тема работы: «Программирование на языке Ассемблер. Изучение команд перехода»

1. **Цель работы**:

Получение практических навыков работы с команд переходов и передачи управления.

2. Задание

Изучить команды переходов и передачи управления. Написать на языке ассемблера, выполнить и исследовать с помощью отладчика программу для работы с командами переходов и передачи управления.

3. Оснащение работы

Техническое задание, ПК, эмулятор DOSBox.

4. Основные теоретические сведения

4.1 Безусловные переходы.

Основной инструкцией перехода в наборе инструкций МП является инструкция JMP. Эта инструкция указывает МП, что в качестве следующей за JMP инструкцией нужно выполнить инструкцию по целевой метке. Например, после завершения выполнения фрагмента программы:

```
mov         ax,1
jmp         AddTwoToAX
AddOneToAx:
inc         ax
jmp         AXIsSet
AddTwoToAX:
            add  ax,2
AXIsSet:
```

регистр AX будет содержать значение 3, а инструкции ADD и JMP, следующие за меткой AddOneToAX, никогда выполнены не будут. Здесь инструкция:

```
jmp  AddTwoToAX
```

указывает процессору, что нужно установить указатель инструкций IP в значение смещения метки AddTwoToAX; поэтому следующей выполняемой инструкцией будет инструкция:

```
add  ax,2
```

Иногда совместно с инструкцией JMP используется операция SHORT. Для указания на целевую метку инструкция JMP обычно использует 16-битовое смещение. Операция SHORT указывает Ассемблеру, что нужно использовать

не 16-битовое, а 8-битовое смещение (что позволяет сэкономить в инструкции JMP один байт). Например, последний фрагмент программы можно переписать так, что он станет на два байта короче:

```
        movax,1
        jmpSHORT AddTwoToAX
AddOneToAx:
inc         ax
jmp         SHORT AXIsSet
AddTwoToAX:
inc         ax

AXIsSet:
```

Недостаток использования операции SHORT (короткий) состоит в том, что короткие переходы могут осуществлять передачу управления на метки, [illegible] некоторых случаях Ассемолер может сообщать вам, что метка недостижима с помощью короткого перехода. К тому же операцию SHORT имеет смысл использовать для ссылок вперед, поскольку для переходов назад (на предшествующие метки) Ассемблер автоматически использует короткие переходы, если на метку можно перейти с помощью короткого перехода, и длинные в противном случае.

4.2 Условные переходы.

Описанные инструкции переходов – это только часть того, что вам потребуется для написания полезных программ. В действительности необходима возможность писать такие программы, которые могут принимать решения. Именно это можно делать с помощью операций условных переходов.

Инструкция условного перехода может осуществлять или нет переход на целевую (указанную в ней) метку, в зависимости от состояния регистра флагов.

Рассмотрим следующий пример:

```
mov ah,1                ;функция DOS ввода с клавиатуры
int 21h                 ; получить следующую нажатую клавишу
cmp al,'A'                ;была нажата буква "A"?
je AWasTyped            ; да, обработать ее
mov [TampByte],al  ; нет, сохранить символ
.
.
.
AWasTyped:
push ax                 ; сохранить символ в стеке
```

Сначала в данной программе с помощью функции операционной системы

DOS воспринимается нажатая клавиша. Затем для сравнения введенного символа с символом A используется инструкция CMP. Эта инструкция аналогична инструкции SUB, только ее выполнение ни на что не влияет, поскольку назначение данной инструкции состоит в том, чтобы можно было сравнить два операнда, установив флаги так же, как это делается в инструкции SUB. Поэтому в предыдущем примере флаг нуля устанавливается в значение 1 только в том случае, если регистр AL содержит символ A.

Инструкция JE представляет инструкцию условного перехода, которая осуществляет передачу управления только в том случае, если флаг нуля равен 1. В противном случае выполняется инструкция, непосредственно следующая за инструкцией JE (в данном случае – инструкция MOV). Флаг нуля в данном примере будет установлен только в случае нажатия клавиши A; и только в этом случае процессор 8086 перейдет к выполнению инструкции с меткой AWasTyped, то есть инструкции PUSH.

Набор инструкций процессора предусматривает большое разнообразие инструкций условных переходов, что позволяет вам осуществлять переход почти по любому флагу или их комбинации. Можно осуществлять условный переход по состоянию нуля, переноса, по знаку, четности или флагу переполнения и по комбинации флагов, показывающих результаты операций чисел со знаками.

Перечень инструкций условных переходов приводится в таблице 10.1.

Таблица 10.1 - Инструкции условных переходов

Название	Значение	Проверяемые флаги
JB/JNAE	Перейти, если меньше / перейти, если не больше или равно	CF = 1
JAE/JNB	Перейти, если больше или равно / перейти, если не меньше	CF = 0
JBE/JNA	Перейти, если меньше или равно / перейти, если не больше	CF = 1 или ZF = 1
JA/JNBE	Перейти, если больше / перейти, если не меньше или равно	CF = 0 и ZF = 0
JE/JZ	Перейти, если равно	ZF = 1
JNE/JNZ	Перейти, если не равно	ZF = 0
JL/JNGE	Перейти, если меньше чем / перейти, если не больше чем или равно	SF = OF
JGE/JNL	Перейти, если больше чем или равно/перейти, если не меньше чем	SF = OF
JLE/JNLE	Перейти, если меньше чем или равно / перейти, если не больше, чем	ZF = 1 или SF = OF
JG/JNLE	Перейти, если больше чем / перейти, если не меньше чем или равно	ZF = 0 или SF = OF

JP/JPE	Перейти по четности	PF = 1
J NP/JPO	Перейти по нечетности	PF = 0
J S	Перейти по знаку	SF = 1
J NS	Перейти, если знак не установлен	SF = 0
J C	Перейти при наличии переноса	CF = 1
J NC	Перейти при отсутствии переноса	CF = 0
J O	Перейти по переполнению	OF = 1
J NO	Перейти при отсутствии переполнения	OF = 0
J CXZ	Переход, если cx равно 0	CX= 0

CF – флаг переноса, SF – флаг знака, OF – флаг переполнения, ZF – флаг нуля, PF – флаг четности.

Несмотря на свою гибкость, инструкции условного перехода имеют также серьезные ограничения, поскольку переходы в них всегда короткие. Другими словами, целевая метка, указанная в инструкции условного перехода, должна отстоять от инструкции перехода не более, чем на 128 байт. Например, Ассемблер не может ассемблировать:

```
Ju

.

.DB             1000 DUP (?)
.
.
.
dec             Ax
Jnz             JumpTarget
```

так как метка JumpTarget отстоит от инструкции JNZ более чем на 1000 байт. В данном случае нужно сделать следующее:

```
Ju
D

d
jn
```

```
jmp  JumpTarget

SkipJump:
```

где условный переход применяется для того, чтобы определить, нужно ли выполнить длинный безусловные переход.

Командам условного перехода может предшествовать любая команда, изменяющая состояние флагов.

Рассмотрим простейший пример с использованием ввода, вывода и различных видов переходов. Необходимо ввести с клавиатуры значение двух переменных a и x. Если a<x, то из x отнять a, а иначе сложить их значения.

```
MODEL      TINY
STACK 100h
DATASEG
        a db ?
        x db ?
        InputX db 0ah,0dh,"Please? input X =$"
        InputA db 0ah,0dh,"Please? input A =$"
        AnswerOne db 0ah,0dh,"a + x = $"
        AnswerTwo db 0ah,0dh,"x - a = $"
CODESEG
start:
        mov ax,@data
        mov ds,ax

        xor ax,ax
        lea dx,InputA;Вывод сообщения
        mov ah,09h
        int 21h

        mov ah,01h ;read(a) ->al
        int 21h
        mov a,al
        sub a,30h ;Перевести в число

        xor ax,ax
        lea dx,InputX;Вывод сообщения
        mov ah,09h
        int 21h

        mov ah,01h ;read(x) ->al
        int 21h
```

```
mov x,al
sub x,30h ;Перевести в число
mov al,a      ;перенесем a в регистр
cmp al,x      ;команда сравнения 1й аргумент должен быть регистром
jl AlowerX
XlowerA:
lea dx,answerTwo;Вывод сообщения
mov ah,09h
int 21h

xor ax,ax;Вычисления по заданию
mov al,x
sub al,a

mov dl,al ;вывод результата
add dl,30h
mov ah,02h
int 21h
jmp exit

AlowerX:
lea dx,answerOne;Вывод сообщения
mov ah,09h
int 21h

xor ax,ax
mov al,a;Вычисления по заданию
add al,x

mov dl,al ;вывод результата
add dl,30h
mov ah,02h
int 21h

exit:
mov ax,04c00h
int 21h

end start
```

5. Порядок выполнения работы

Написать на языке ассемблера, выполнить и исследовать с помощью

отладчика программу для работы с командами переходов и передачи управления. Значение a, x ввести с клавиатуры, а значение y вывести на экран. Если оно отрицательное, то впереди вывести «-», а y числа изменить знак. Ответ должен быть в пределах от -9 до 9. Для защиты подобрать 4 примера, которые демонстрируют работу всех веток программы. Варианты заданий приведены в приложении А.

6. Форма отчета о работе

Лабораторная работа № ___
Номер учебной группы _____
Фамилия, инициалы учащегося _____
Дата выполнения работы _____
Тема работы: _____
Цель работы: _____
Оснащение работы: _____
Индивидуальное задание на работу
Указание имен исходного и исполняемого файлов
Результат выполнения работы: _____
_____
_____

Отчет представляется в виде текстового файла. К отчету должны прилагаться файл исходного кода Программы и рабочий исполняемый файл.

7. Контрольные вопросы и задания

1. Что такое флаги и когда они изменяются?
2. Можно ли напрямую изменить флаг?
3. Флаги признаки.
4. Флаги управления.
5. Условные и безусловные переходы.

8. Рекомендуемая литература

Финогенов, К. Г. Основы языка Ассемблера [Текст] / К. Г. Финогенов. – М.: Радио и связь, 2000.
Финогенов, К. Г. Использование языка Ассемблера [Текст]: учеб. пособие для вузов / К.Г. Финогенов. – М.: Горячая линия Телеком, 2004.
Юров, В. И. Assembler [Текст]: учеб. пособие для вузов / В. И. Юров. 2-е изд. – СПб.: Питер, 2007.