ЛАБОРАТОРНАЯ РАБОТА № 2

Цель работы:

приобрести умения и практические навыки для работы с командами сравнения, ветвления и командами передачи управления.

Теоретическая часть:

Команда сравнения – *cmp приемник, источник* – выполняется как команда вычитания. Но, в отличии от команды вычитания, она устанавливает флаги, но не сохраняет результат. Эта команда используется совместно с командами условной передачи управления для организации ветвления в программе.

Цикл – повторение одного или группы операндов. Для организации циклов в языке ассемблера используется команда *loop метка*. Эта команда уменьшает значение регистра *cx* (*ecx*) на единицу и выполняет переход типа *short* на метку, то есть *cx* (*ecx*) в данном случае играет роль счетчика.

Например, если мы хотим, чтобы команда сложения выполнилась 3 раза, фрагмент кода программы может выглядеть следующим образом:

```
…
mov cx, 3 ; занесение в cx числа повторений цикла
m:
add ax, 3 ; команда сложения, которую необходимо было повторить
loop m ; команда цикла
…
```

Повторение при этом будет выполняться с метки, указанной в команде цикла. Внутри цикла также могут быть организованы ветвления, но рекомендуется при их выполнении не выполнять переход на метки вне цикла без необходимости.

Для обработки элементов массивов также рекомендуется применять циклы. При обработке элементов массива переход к следующему элементу выполняется при помощи увеличения значения регистра *bx* или *si* (смотря какой регистр был использован для обращения к элементам массива). В зависимости от размера элементов массива (байт, слово, двойное слово и т.д.)

изменяется и значение, на которое необходимо увеличивать. Увеличение, как правило, прописывается в самом конце цикла перед командой *loop*.

Например, если требуется просмотреть в регистре *ax* все элементы массива слов из 10 элементов поочередно, фрагмент программы может иметь следующий вид:

```
…
mov cx, 10 ; занесение в cx числа повторений цикла
xor bx, bx ; обнуление регистра bx, можно просто занести 0
m:
mov ax, mas[bx] ; занесение очередного элемента массива в регистр ax для просмотра
add bx, 2 ; увеличение значения регистра bx на 2 для перехода к следующему элементу
loop m ; команда цикла
…
```

Команды передачи управления

Команды передачи управления делятся на команды условной и безусловной передачи управления. Первый тип команд выполняет переход на метку только при выполнении определенного условия, второй тип выполняет переход всегда.

Команда безусловной передачи управления – jmp операнд. Операнд в данном случае может быть меткой, регистром или ячейкой памяти, хранящими адрес перехода. Команда выполняет передачу управления по указанному адресу, не сохраняя информацию для возврата. Различают *короткий (short)* переход – на расстояние от -128 до +127 байт от команды, *близкий (near)* – адрес перехода находится в том же сегменте, что и команда, и *дальний (far)* – адрес перехода находится в другом сегменте.

Группа команд условной передачи управления используется для организации ветвлений в программе. Эти команды проверяют значение флагов, установленных предшествующими командами, и выполняют передачу управления на метку при выполнении соответствующего условия. Можно выделить следующие команды условной передачи управления:

– *jg метка* (переход выполнится в случае, если $a > b$);

– *jge метка* (переход выполнится в случае, если $a \geq b$);

– *jl метка* (переход выполнится в случае, если $a < b$);

– *jle метка* (переход выполнится в случае, если $a \leq b$);

– *je метка* (переход выполнится в случае, если $a = b$);

– *jne метка* (переход выполнится в случае, если $a \neq b$);

– *jz метка* (переход выполнится в случае, если содержимое аккумулятора равно 0);

– *jnz метка* (переход выполнится в случае, если содержимое аккумулятора не равно 0).

Команды условной передачи управления функционируют следующим образом: при выполнении условия управление передается на метку, указанную в команде, иначе – на следующую команду после команды условной передачи управления.

Например, необходимо сравнить значение двух переменных *x* и *y* и выполнить переход на метку в случае, если значение первой переменной больше значения второй. Тогда фрагмент кода может выглядеть следующим образом:

```
…
mov ax, x ; занесение в ax значения переменной x
cmp ax, y ; сравнение переменных x и y
jg m ; указание выполнить переход на метку m в случае, если первое число больше второго
jmp k ; в противном случае выполнить переход на метку k
m: …
k: …
…
```

При выполнении сложных ветвлений рекомендуется придерживаться следующей методики: программу пишут по ветвям «нет» до конца программы, а затем дописываются оставшиеся ветви. При выполнении данной лабораторной работы рекомендуется построить блок-схему для удобного ориентирования в ветвлениях.

Команды условной передачи управления в реальном режиме выполняют короткий переход на метку, то есть нельзя передать управление на расстояние

больше 128 байтов (чаще встречается в программах с большим количеством строк). Для обхода этого ограничения можно писать код следующим образом:

```
cmp ax, 0
jl m1          ; ax < 0
jmp m     ; иначе, то есть ax > 0
m1: < участок, который необходим>
m: <…>
```

Варианты заданий:

Задание 1. Ветвления:

1. Даны *a*, *b*, *c* – значения типа *word*. Присвоить переменной *c* значение наибольшего из трех.
2. Даны *a*, *b*, *c* – значения типа *word*. Присвоить переменной *c* значение наименьшего из трех.
3. Даны *a*, *b*, *c* – значения типа *byte*. Присвоить переменной *c* значение наибольшего из трех.
4. Даны *a*, *b*, *c* – значения типа *byte*. Присвоить переменной *c* значение наименьшего из трех.

Задание 2. Работа с массивами:

1. Дан массив *X* из 8 значений типа *byte*. Увеличить в два раза все элементы массива.
2. Дан массив *X* из 8 значений типа *byte*. Уменьшить в два раза все элементы массива.
3. Дан массив *K* из 8 значений типа *word*. Найти сумму отрицательных элементов массива.
4. Дан массив *K* из 8 значений типа *word*. Найти сумму положительных элементов массива.
5. Дан массив *A* из 8 значений типа *word*. Найти произведение положительных элементов массива.
6. Дан массив *A* из 8 значений типа *word*. Найти произведение отрицательных элементов массива.
7. Дан массив *M* из 10 значений типа *byte*. Найти количество положительных элементов массива.

8. Дан массив *M* из 10 значений типа *byte*. Найти количество отрицательных элементов массива.

9. Дан массив *M* из 10 значений типа *word*. Найти количество положительных элементов массива.

10. Дан массив *M* из 10 значений типа *word*. Найти количество отрицательных элементов массива.

11. Дан массив *E* из 8 значений типа *word*. Увеличить положительные элементы в 2 раза, обнулить отрицательные элементы.

12. Дан массив *E* из 8 значений типа *word*. Увеличить отрицательные элементы в 2 раза, обнулить положительные элементы.

13. Дан массив *X* из 8 значений типа *word*. Найти количество четных элементов массива.

14. Дан массив *X* из 8 значений типа *word*. Найти количество нечетных элементов массива.

15. Дан массив *X* из 8 значений типа *word*. Найти количество элементов массива, кратных 3.

16. Дан массив *X* из 8 значений типа *word*. Найти количество элементов массива, кратных 5.

17. Даны два массива *A* и *B*, заполненные 8 значениями типа *byte* каждый. Создать новый массив *C*, составленный по правилу: $C_i = max\ (A_i, B_i)$.

18. Даны два массива *A* и *B*, заполненные 8 значениями типа *byte* каждый. Создать новый массив *C*, составленный по правилу: $C_i = min\ (A_i, B_i)$.

19. Даны два массива *A* и *B*, заполненные 8 значениями типа *word* каждый. Создать новый массив *C*, составленный по правилу: $C_i = max\ (A_i, B_i)$.

20. Даны два массива *A* и *B*, заполненные 8 значениями типа *word* каждый. Создать новый массив *C*, составленный по правилу: $C_i = min\ (A_i, B_i)$.

Порядок работы:

1. Открыть программу, указанную преподавателем, после объяснения принципов работы с ней.
2. Создать файл с расширением, указанным преподавателем.
3. Ввести текст программы (с крайней левой позиции):

```
.model tiny
.code
N:   push cs
     pop ds
     ; ввести команды программы
     mov ax, 4c00h
     int 21h
.data
     ; ввести директивы распределения данных
end N
```

4. Сохранить программу.
5. Выполнить компиляцию.
6. Если ошибок нет, то запустить эмуляцию программы и пошагово выполнить ее.
7. Подготовить отчет о проделанной работе.

Вопросы к теоретическому материалу

1. Перечислите регистры общего назначения и их названия.
2. Охарактеризуйте регистр ax.
3. Охарактеризуйте регистр bx.
4. Охарактеризуйте регистр cx.
5. Охарактеризуйте регистр dx.
6. Опишите строение регистров общего назначения.
7. Из скольки полей состоит строка кода на языке Ассемблера? Как они называются?
8. Какое поле является обязательным?
9. Какие поля являются необязательными?
10. Охарактеризуйте поле метки.
11. Охарактеризуйте поле кода операции.
12. Охарактеризуйте поле операндов.
13. Охарактеризуйте поле комментариев.
14. Что указывает при написании программы на место, где должны располагаться данные?
15. Перечислите псевдокоманды определения данных.
16. Что может быть указано в поле операндов для псевдокоманд определения данных?
17. Верна ли следующая запись: x db 01100111? Почему?
18. Верна ли следующая запись: x db 01100001b? Почему?
19. Верна ли следующая запись: x dw 01h? Почему?
20. Верна ли следующая запись: x db FFFh? Почему?
21. Верна ли следующая запись: x dd 0B1h? Почему?
22. Верна ли следующая запись: x db ? ? Почему?
23. Назовите команду для сравнения и укажите ее синтаксис.
24. Совместно с какими командами используется команда сравнения?
25. Какие два вида команд передачи управления существуют?

26. Назовите команду безусловной передачи управления и укажите ее синтаксис.

27. Сохраняется ли информация для возврата при выполнении перехода при помощи команды безусловной передачи управления?

28. Какие виды перехода различают?

29. Назовите команду условной передачи управления, которая передает управление на метку в случае, если при сравнении первый операнд больше второго. Укажите ее синтаксис.

30. Назовите команду условной передачи управления, которая передает управление на метку в случае, если при сравнении первый операнд больше или равен второму. Укажите ее синтаксис.

31. Назовите команду условной передачи управления, которая передает управление на метку в случае, если при сравнении первый операнд меньше второго. Укажите ее синтаксис.

32. Назовите команду условной передачи управления, которая передает управление на метку в случае, если при сравнении первый операнд меньше или равен второму. Укажите ее синтаксис.

33. Назовите команду условной передачи управления, которая передает управление на метку в случае, если при сравнении первый операнд равен второму. Укажите ее синтаксис.

34. Назовите команду условной передачи управления, которая передает управление на метку в случае, если при сравнении первый операнд не равен второму. Укажите ее синтаксис.

35. Назовите команду условной передачи управления, которая передает управление на метку в случае, если содержимое аккумулятора равно 0. Укажите ее синтаксис.

36. Назовите команду условной передачи управления, которая передает управление на метку в случае, если содержимое аккумулятора не равно 0. Укажите ее синтаксис.

37. Каким образом функционируют команды условной передачи управления? Куда передается управление в случае, если условие, выполнение которого проверяет команда, не выполняется?

38. Для чего предназначены команды условной передачи управления?

ПРОЦЕСС СДАЧИ ЛАБОРАТОРНОЙ РАБОТЫ

По итогам выполнения каждой лабораторной работы студент:

1. демонстрирует преподавателю правильно работающие программы;
2. демонстрирует приобретенные знания и навыки, отвечая на несколько небольших вопросов преподавателя по составленной программе, возможностям ее доработки и теме лабораторной работы в целом;
3. демонстрирует отчет по выполненной лабораторной работе.

Итоговая оценка складывается из оценок по трем указанным составляющим.

Отчет по лабораторной работе оформляется по шаблону, представленному в приложении 1. Требования к отчету представлены в приложении 2.

ПРИЛОЖЕНИЕ 1. ШАБЛОН ОТЧЕТА

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Казанский национальный исследовательский технический университет им. А.Н. Туполева – КАИ»

Институт компьютерных технологий и защиты информации

Отделение СПО ИКТЗИ (Колледж информационных технологий)

ЛАБОРАТОРНАЯ РАБОТА №__

по дисциплине

СИСТЕМНОЕ ПРОГРАММИРОВАНИЕ

Работу выполнил

Студент гр.43__

Фамилия И.О.

Принял

Преподаватель Григорьева В.В.

Казань 2024

ВАРИАНТ __

1. **Цель работы**
2. **Задание на лабораторную работу** – вставляется задание на лабораторную работу, соответствующее индивидуальному варианту студента.
3. **Результат выполнения работы** – формируется описание хода выполнения работы (разработанных подпрограмм, классов, переменных, структур данных и т.п.) и вставляются скриншоты с результатами работы разработанных программ (скриншоты должны быть подписаны).
4. **Листинг программы** – вставляется код разработанной программы.

ПРИЛОЖЕНИЕ 2. ТРЕБОВАНИЯ К ОФОРМЛЕНИЮ ОТЧЕТА

Лист документа должен иметь книжную ориентацию, поля документа должны составлять: левое – 3 см, правое – 1,5 см, верхнее – 2 см, нижнее 2 см.

Нумерация страниц – внизу страницы, по центру, особый колонтитул для первой страницы.

Междустрочный интервал – 1,5 (полуторный), отступ первой строки – 1,25.

Текст документа должен быть выполнен с использованием шрифта Times New Roman, размер – 14, выравнивание – по ширине. Заголовки выполняются тем же шрифтом, но размера 16, полужирное начертание, размещение – по центру.

Рисунки должны размещаться по центру, они нумеруются по порядку. Перед рисунком в тексте на него должна быть ссылка. Подпись рисунка должна располагаться по центру и быть выполнена шрифтом Times New Roman, размер – 12. Сначала происходит нумерация рисунка, а затем пишется его название.