**Программирование микропроцессора Intel 8086.**

**Инструкции передачи управления**

Цель работы

а) Инструкции прыжка.

б) Инструкции тестирования.

в) Итерационные инструкции.

**1. Инструкции прыжка**

Операторы прыжка могут быть безусловным и условным.

***1.1. Инструкции безусловного прыжка***

Инструкция безусловного прыжка имеет вид:

 Позиция JMP

где позиция может быть меткой, регистром, переменной и т. д.

Его действие заключается в загрузке в регистр IP и, возможно, в регистр CS новый адрес (полученный из позиции) следующей инструкции, которая будет выполнена.

***1.2. Инструкции условного прыжка***

Операторы условного прыжка имеют вид:

Позиция Jconditie

где после проверки условия прыжок выполняется, только если выполняется условие.

Конкретные формы инструкции условного прыжка:

1. Для любого типа значений:

а) Условный прыжок CX = 0 (регистр счетчика имеет нулевое содержимое):

Положение JCXZ

б) Условный прыжок CF = 1 (переход к предыдущей операции):

Позиция JC

в) Условный прыжок CF = 0 (без перехода к предыдущей операции):

Позиция JNC

г) Условный прыжок ZF = 1 (нулевой результат в предыдущей операции):

Позиция JE

Позиция JZ

д) Условный прыжок ZF = 0 (нулевой результат в предыдущей операции):

Позиция JNE

Позиция JNZ

е) Условный прыжок PF = 1 (четный результат в предыдущей операции):

Позиция JP

Позиция JPE

ж) Условный прыжок PF = 0 (нечетный результат в предыдущей операции):

Положение JNP

Позиция JPO

2. Для беззнаковых (натуральных значений)

a) Условный прыжок (CF = 0 AND ZF = 0), т.е. "более высокий" результат (> 0) на предыдущей операции:

Позиция JA

Позиция JNBE

б) Условный прыжок CF = 0, т. е. результатом "больше или равно" (> = 0) к предыдущей операции:

Позиция JAE

Положение JNB

в) Условный прыжок CF = 1, т.е. "меньший" результат (<0) на предыдущей операции:

Позиция JB

Позиция JNAE

г) Условный прыжок (CF = 1 AND ZF = 1), то есть результатом «меньше или равно» (<= 0) к предыдущей операции:

Позиция JBE

Позиция JNA

3. Для значений со знаком (целых):

a) Условный прыжок (SF = OF AND ZF = 0), т. е. «более высоким» результатом (> 0) к предыдущей операции:

Позиция JG

Позиция JNLE

б) Условный прыжок SF = OF, т.е. результат "больше или равно" (> = 0) к предыдущей операции:

Позиция JGE

Положение JNL

в) Условный прыжок SF <> OF, т.е. "меньший" результат (<0) к предыдущей операции:

Позиция JL

Позиция JNGE

г) Условный прыжок (SF <> OF AND ZF = 1), то есть результатом "меньше или равно"

(<= 0) к предыдущей операции:

Позиция JLE

Положение JNG

д) Условный прыжок SF = 1 (отрицательный результат предыдущей операции):

Позиция JS

е) Условный прыжок SF = 0 (положительный результат предыдущей операции):

Позиция JNS

ж) Условный прыжок OF = 1 (превышение диапазона предыдущей операции):

Позиция JO

з) Условный прыжок OF = 0 (без выхода за пределы предыдущей операции):

Положение JNO

С помощью этих инструкций в программе могут быть созданы структуры принятия решений, аналогичные инструкциям if в языках высокого уровня (C, Pascal и т. Д.).

Пример 1 (расчет модуля суммы):

ADD AX, DX ; количество

JGE et1 ; если AX> = 0, он остается неизменным

NEG AX ; в противном случае AX = - AX

et1:MOV rez, AX ; переменное значение ответа AX

Пример 2 (обозначение "O" нулевого результата и "N" ненулевого результата суммы):

MOV rez, 'O' ; предполагаемый нулевой результат

ADD AX, DX ; количество

JZ et1 ; если AX = нулевой прыжок

MOV rez, 'N' ; в противном случае результат будет ненулевой

et1: ; следующий оператор

**2. Инструкции тестирования**

Если вы хотите протестировать условия, которые будут использоваться в операторе условного прыжка, без загрузки результата теста в регистр или сохранения в памяти, вы можете использовать тестовые операторы CMP и TEST.

Эти инструкции имеют эффект, аналогичный инструкциям SUB и AND, соответственно, но без загрузки результата в регистр или сохранения в памяти.

1. Инструкция CMP имеет вид:

CMP операнд1, операнд2

и эффект: операнд1 - операнд2

Пример (изначально AX = 1000 H и BX = 800 H):

CMP AX, BX

Эффект:

0001 0000 0000 0000 B= 1000 H -

0000 1000 0000 0000 B= 0800 H

-------------------------------------------------

0000 1000 0000 0000 B= 0800 H

После выполнения инструкции: AX = 1000 H, BX = 800 H, OF = 0, SF = 0, ZF = 0 и CF = 0.

2. Инструкция TEST имеет вид:

TEST операнд1, операнд2

и эффект: операнд1 И операнд2

Пример (изначально AX = 5555 H и BX = 6666 H):

TEST AX, BX

Эффект:

0101 0101 0101 0101 B = 5555 H x

0110 0110 0110 0110 B = 6666 H

------------------------------------------------

0100 0100 0100 0100 B = 4444 H

После выполнения инструкции: AX = 5555 H, BX = 6666 H, SF = 0 и ZF = 0.

**3. Итерационные инструкции**

***3.1. Безусловная итеративная инструкция***

Простая итеративная инструкция имеет форму:

LOOP pozitie

где позиция может быть меткой, регистром, переменной и т. д.

Первой операцией, выполняемой в этом случае, является уменьшение регистра CX (счетчик или числитель). Если его содержимое не становится 0 после уменьшения, эффект идентичен инструкции JMP. Если после уменьшения CX он становится 0, инструкция LOOP не имеет другого эффекта.

Таким образом, действие инструкции:

LOOP etich

эквивалентна следующей последовательности:

DEC CX

JNZ etich

Оператор LOOP используется в типичной структуре программы:

MOV CX, nr

etich: ...; последовательность

...; инструкции

LOOP etich

что позволяет многократно выполнять последовательность инструкций.

Эта инструкция позволяет реализовать итерационные программные структуры.

Пример (вычисление в DX количества битов 1 слова cuv, вариант по биту Carry

(CF), финальный цикл испытаний, фиксированное количество = 16 итераций):

MOV AX, cuv

XOR DX, DX

MOV CX, 16 ; для CX от 16 до 1

et1: SHLAX, 1 ; CF = MSb

JNC et 2 ; если CF = 0 скачок

INC DX ; в противном случае (CF = 0) считать 1

et2: LOOP et1 ; повторение

***3.2. Условные итерационные инструкции***

Условные итерационные инструкции имеют следующие формы:

а) Условный цикл с нулевым результатом (ZF = 1):

LOOPZ etich

Или LOOPE etich

б) Цикл, обусловленный ненулевым результатом (ZF = 0):

LOOPNZ etich

Или LOOPNE etich

Типичная структура, в которой используется оператор LOOPcondition:

MOV CX, nr

etich: ...; последовательность

...; инструкции

LOOPcondition etich

допускает повторное выполнение последовательности инструкций не более одного раза, потому что повторение прерывается, если выполняется условие.

Таким образом, действие инструкции:

LOOPcondition etich

эквивалентна следующей последовательности:

Jconditie etich

DEC CX

JNZ etich

Пример (поиск наиболее значимого бита 0 в cuv, последний цикл тестирования с двойным условием):

MOV CX, 15 ; CX = позиция тестируемого долота

MOV BX, 1 ; BX = маска

и: ROR BX, 1 ; вращает BX с позицией на dr.

ТЕСТ cuv, BX ; И по крупицам без резерва памяти.

LOOPNZ et ; до ((бит проверяется = 0) ИЛИ (CX = 0))

**4. Примеры программ**

***1. Программа для вычисления произведения двух шестнадцатеричных цифр, считанных с клавиатуры, с отображением результата на экране.***

data segment

numeprog db 25 dup(0ah),'Вычисление произведения двух значений $'

cifra1 db 2 dup(0ah),0dh,' 'Первое значение: $'

cifra2 db 2 dup(0ah),0dh,' Второе значение: $'

eroarecifra db 2 dup(0ah),0dh, 'Значения не являются шестнадцатеричными цифрами!$'

rezultat db 2 dup(0ah),0dh,' Результат: $'

data ends

assume cs:cod, ds:data

cod segment

start:

mov ax, data

mov ds, ax

mov dx, offset numeprog ; отображение строки символов

mov ah, 9 ; (название программы)

int 21h

mov dx, offset cifra1 ; отображение строки символов

mov ah, 9 ; (сообщение №1)

int 21h

mov ah, 1 ; чтение символов не повторяется

int 21h ; (чтение первой цифры в al)

; преобразование символов -> цифра

cmp al, 30h; caracter >= '0' ?

jl er1 ; нет => ошибка

cmp al, 3ah ; да => символ <= '9'?

jnl urm11 ; нет => перейти к следующему тесту

sub al, 30h ; да => настройка (символ - это цифра 0..9)

jmp sf1 ; прыжок к концу преобразования

 urm11:cmp al, 41h ; символ> = 'A'?

jl er1 ; нет => ошибка

cmp al, 47h ; дa => буква <= 'F'?

jnl urm12 ; нет => перейти к следующему тесту

sub al, 37h ; да => настройка (символ - цифра A..F)

jmp sf1 ; прыжок к концу преобразования

 urm12:cmp al, 61h ; символ >= 'a' ?

jl er1 ; нет => ошибка

cmp al, 67h ; да => буква <= 'f' ?

jnl er1 ; нет => перейти к следующему тесту

sub al, 57h ; да => настройка (символ - цифра a..f)

jmp sf1

 er1:mov dx, offset eroarecifra

 mov ah, 9 ; отображение строки символов

int 21h ; (сообщение об ошибке)

 sf1:

mov bl, al ; цифра1 в bl

mov dx, offset cifra2 ; afisare sir caractere

mov ah, 9 ; (сообщение №2)

int 21h

mov ah, 1 ; чтение символов без повторения

int 21h ; (чтение второй цифры в al)

; преобразование символов -> цифра

cmp al, 30h ; символ >= '0' ?

jl er2 ; нет => ошибка

cmp al, 3ah ; да => символ <= '9' ?

jnl urm21 ; нет => перейти к следующему тесту

sub al, 30h ; да => настройка (символ - цифра 0..9)

jmp sf2 ; прыжок к концу преобразования

 urm21:cmp al, 41h ; символ >= 'A' ?

jl er2 ; нет => ошибка

cmp al, 47h ; да => буква <= 'F' ?

jnl urm22 ; нет=> перейти к следующему тесту

sub al, 37h ; да => настройка (символ - цифра A..F)

jmp sf2 ; прыжок к концу преобразования

urm22:cmp al, 61h ; символ >= 'a' ?

jl er2 ; нет => ошибка

cmp al, 67h ; да => буква <= 'f' ?

jnl er2 ; нет => перейти к следующему тесту

sub al, 57h ; да => настройка (символ - цифра a..f)

jmp sf2

 er2:mov dx, offset eroarecifra

 mov ah, 9 ; отображение строки символов

int 21h ; (сообщение об ошибке)

 sf2:

; расчет

mov dl, bl ; цифра 2

mul dl ; (цифра 2 в al)

mov bx, ax ; результат в bx

 mov dx, offset rezultat

mov ah, 9 ; отображение строки символов

int 21h ; (сообщение результата)

 mov cl, 4; separare cifra 1

mov al, bl

 shr al, cl

; преобразование цифры 1 -> символ

cmp al, 0ah ; цифра = 0..9?

jl urm3 ; да => настройка(urm4)

cmp al, 10h ; нет=> цифра = A..F ?

jnl er3

add al, 37h ; да => настройка

jmp sf3

 urm3:add al, 30h

jmp sf3

 er3:

 sf3:

 mov dl, al

mov ah, 2 ; отображение первой цифры

int 21h

mov al, bl; separare cifra 2

 shl al, cl

 shr al, cl

; преобразование цифры 2 -> символ

cmp al, 0ah ; цмфра = 0..9 ?

jl urm4 ; да => настройка (urm4)

cmp al, 10h ; нет => цифра = A..F ?

jnl er4

add al, 37h ; да => настройка

jmp sf4

 urm4:add al, 30h

jmp sf4

 er4:

 sf4:

 mov dl, al

mov ah, 2 ; отображение второй цифры

int 21h

mov ah, 8 ; чтение символов без повторения

int 21h ; (Enter)

mov ah, 4ch ; выход

int 21h

cod ends

end start

***2. Программа упорядочивания 2-х значений из ряда натуральных чисел по возрастанию.***

DATA SEGMENT

Vect DW 8766 H, 5678 H, 0ABC3 H, 0B44 H

DATA ENDS

ASSUME CS: COD, DS: DATA

COD SEGMENT

START:

MOV AX, DATA

MOV DS, AX

MOV BX, OFFSET VECT

MOV AX, [BX] ; AX принимает значение vect [0]

CMP AX, [BX+2] ; AX (vect [0]) сравнивается с vect [2]

JB inord ; если vect [0] <vect [1], то

; по возрастанию (прыжок)

XCHG AX, [BX+2] ; в противном случае vect[0] заменяется на

MOV [BX], AX ; vect[2]

inord:MOV AH, 4CH

INT 21H

CODENDS

END START

***3. Программа упорядочивания по возрастанию 2-х значений из строки целых чисел со знаком.***

DATASEGMENT

Vect DW 8766 H, 5678 H, 0ABC3 H, 0B44 H

DATA ENDS

ASSUME CS: COD, DS: DATA

CODSEGMENT

START:

MOV AX, DATA

MOV DS, AX

MOV BX, OFFSET VECT

MOV AX, [BX] ; AX принимает значение vect [0]

CMP AX, [BX+2] ; AX (vect [0]) сравнивается с vect [2]

JL inord ; если vect [0] <vect [1], то

; по возрастанию (прыжок)

XCHG AX, [BX+2] ; в противном случае vect[0] заменяется на

MOV [BX], AX ; vect[2]

inord:MOV AH, 4CH

INT 21H

CODENDS

END START

***4. Программа для вычисления количества битов, равных 1 слова cuv в регистре DX.***

a)Программа для вычисления количества битов, равных 1 слова cuv в регистре DX.

DATA SEGMENT

cuv DW 0e360 H

DATA ENDS

ASSUME CS: COD, DS: DATA

CODSEGMENT

START:

MOV AX, DATA

MOV DS, AX

MOV AX, cuv ; загрузка cuv в регистр батареи

XOR DX, DX ; сброс счетчика (DX = 0)

MOV CX, 16 ; для CX от 16 до 1

 et1:SHL AX, 1 ; перемещение влево на одну позицию

JNC et2 ; прыжок, если транспорта не было

INC DX ; приращение счетчика, если это было CF.

 et2:LOOP et1 ; повторение

MOV AH, 4CH

INT 21H

CODENDS

END START

b) Вариант через бит Carry, цикл с начальным тестом, фиксированное количество итераций:

DATA SEGMENT

cuv DW 0e360 H

DATA ENDS

ASSUME CS: COD, DS: DATA

COD SEGMENT

START:

MOV AX, DATA

MOV DS, AX

MOV AX, cuv ; загрузка cuv в регистр батареи

XOR DX, DX ; сброс счетчика (DX = 0)

MOV CX, 17 ; для CX от 17 до 1

 et1:DEC CX ; CX = CX - 1

JCXZ et2 ; если CX = 0 перейти к et2 (готово)

SHL AX, 1 ; в противном случае переместитесь влево на одну позицию

JNC et1 ; прыжок, если транспорта не было

INC DX ; приращение счетчика, если это было CF.

JMP et1 ; повторение

et2: MOV AH, 4CH

INT 21H

COD ENDS

END START

c) Вариант через бит Carry, цикл с финальным тестом, переменное количество итераций:

DATA SEGMENT

cuv DW 0e360 H

DATA ENDS

ASSUME CS: COD, DS: DATA

COD SEGMENT

START:

MOV AX, DATA

MOV DS, AX

MOVAX, cuv ; загрузка cuv в регистр батареи

XOR DX, DX ; сброс счетчика (DX = 0)

 et1:SHL AX, 1 ; повторить сдвиг влево с одной позицией

JNC et2 ; прыжок, если транспорта не было

INC DX ; приращение счетчика, если это было CF.

 et2: OR AX, AX ; тестирование AX

JNZ et1 ; пока что AX=0

MOV AH, 4CH

INT 21H

COD ENDS

END START

d) Вариант через бит Carry, цикл с начальным тестом, переменное количество итераций:

DATA SEGMENT

Cuv DW 0e360 H

DATA ENDS

ASSUME CS: COD, DS: DATA

COD SEGMENT

START:

MOV AX, DATA

MOV DS, AX

MOVAX, cuv ; загрузка cuv в регистр батареи

XOR DX, DX ; сброс счетчика (DX = 0)

 et1:JCXZ et2 ; в то время как CX <> 0

SHL CX, 1 ; двигаться влево на одну позицию

JNC et1 ; прыжок, если транспорта не было

INC DX ; приращение счетчика, если это было CF.

JMP et1 ; повторение

 et2:MOV AH, 4CH

INT 21H

COD ENDS

END START

e) Вариант по маскировке и фиксированному количеству итераций:

DATA SEGMENT

cuv DW 0e360 H

DATA ENDS

ASSUME CS: COD, DS: DATA

COD SEGMENT

START:

MOV AX, DATA

MOV DS, AX

MOVAX, cuv ; загрузить cuv в регистр батареи

XOR DX, DX ; сброс счетчика (DX = 0)

MOV BX, 1 ; маска = BX = 1

 et1: TEST cuv, BX ; повторить тест (cuv AND BX)

JZ et2

INC DX ; если результат теста отличен от нуля, он

; нашел 1 и увеличил счетчик

 et2: SHL BX, 1 ; переместить маску в левое положение.

JNC et1 ; прыжок, если транспорта не было

MOV AH, 4CH

INT 21H

COD ENDS

END START

**5. Домашние задания и упражнения.**

1. Напишите программу для вычисления суммы двух строк двойной точности с двойной точностью слова по памяти.

2. Напишите программу, которая считывает с клавиатуры цифру N и вычисляет N! (N

факториал) и отобразить результат на экране.

3. Напишите программу, которая считывает с клавиатуры значения x, y и z, чтобы вычислить выражение: E = x! + 0.25y + 5z и вывести результат на экран.