Отчет

по лабораторной работе №3

по дисциплине “Архитектура компьютера”

Тема: “Изучение организации ветвлений в программах на языке Ассемблера”

Вариант №4

(Шифр 1.5.4)

Студент гр. 5383 \_\_\_\_\_\_\_\_\_\_\_\_\_\_ Допира В. Е.

Преподаватель \_\_\_\_\_\_\_\_\_\_\_\_\_\_ Кирьянчиков В.А.

Цель работы

Разработать на языке Ассемблер IBM PC программу, которая

по заданным целым значениям a, b, i, k, размером 1 слово, вычисляет:

а) значения i1 = f3(a, b, i) и i2 = f6(a, b, i);

b) значения res = f4(i1, i2, k),

где:

f1 = | 15-2\*i, если a > b

| 3\*i+4, если a ≤ b

f5 = | 20-4\*i, если a > b

| -(6\*i-6), если a ≤ b

f4 = | min(|i1 – i2|, 2), если k < 0

| max(-6, -i2), если k ≥ 0

Теоретические сведения

Безусловный переход — это переход, который выполняется всегда. Безусловный переход осуществляется с помощью команды JMP. У этой команды один операнд, который может быть непосредственным адресом (меткой), регистром или ячейкой памяти, содержащей адрес. Существуют также «дальние» переходы — между сегментами.

Примеры безусловных переходов:

*jmp metka* ;Переход на метку

*jmp bx* ;Переход по адресу в BX

*jmp word[bx]* ;Переход по адресу, содержащемуся в памяти по адресу в BX

Условный переход осуществляется, если выполняется определённое условие, заданное флагами процессора. Состояние флагов изменяется после выполнения арифметических, логических и некоторых других команд. Если условие не выполняется, то управление переходит к следующей команде.

Существует много команд для различных условных переходов. Также для некоторых команд есть синонимы (например, JZ и JE — это одно и то же).

|  |  |  |
| --- | --- | --- |
| Команда | Переход, если | Условие перехода |
| JZ/JE | нуль или равно | ZF=1 |
| JNZ/JNE | не нуль или не равно | ZF=0 |
| JC/JNAE/JB | есть переполнение/не выше и не равно/ниже | CF=1 |
| JNC/JAE/JNB | нет переполнения/выше или равно/не ниже | CF=0 |
| JP | число единичных бит чётное | PF=1 |
| JNP | число единичных бит нечётное | PF=0 |
| JS | знак равен 1 | SF=1 |
| JNS | знак равен 0 | SF=0 |
| JO | есть переполнение | OF=1 |
| JNO | нет переполнения | OF=0 |
| JA/JNBE | выше/не ниже и не равно | CF=0 и ZF=0 |
| JNA/JBE | не выше/ниже или равно | CF=1 или ZF=1 |
| JG/JNLE | больше/не меньше и не равно | ZF=0 и SF=OF |
| JGE/JNL | больше или равно/не меньше | SF=OF |
| JL/JNGE | меньше/не больше и не равно | SF≠OF |
| JLE/JNG | меньше или равно/не больше | ZF=1 или SF≠OF |

У всех этих команд один операнд — имя метки для перехода. Некоторые команды применяются для беззнаковых чисел, а другие — для чисел со знаком. Сравнения «выше» и «ниже» относятся к беззнаковым числам, а «больше» и «меньше» — к числам со знаком. Для беззнаковых чисел признаком переполнения будет флаг CF, а соответствующими командами перехода JC и JNC. Для чисел со знаком о переполнении можно судить по состоянию флага OF, поэтому им соответствуют команды перехода JO и JNO. Команды переходов не изменяют значения флагов.

Команда *CMP* сравнивает операнды и устанавливает флаги. Сравнение осуществляется путем вычитания источника (число, регистр или переменная) из приемника (регистр или переменная; приемник и источник не могут быть переменными одновременно), причем результат вычитания никуда не записывается, единственным результатом работы этой команды оказывается изменение флагов.

Тестирование

|  |  |  |
| --- | --- | --- |
| Входные данные | Выходные данные | Ожидаемый результат |
| a = 1  b = 2  i = 3  k = 4 | i1 = 000D = 13  i2 = FFF4 = -12  res = 000C=12 | i1 = 13  i2 = -12  res = 12 |
| a = 2  b = 1  i = 1  k = 1 | i1 = 000D = 13  i2 = 10  res = FFFA = -6 | i1 = 13  i2 = 10  res = -6 |
| a = -1= FFFF  b = -2= FFFE  i = 1  k = 1 | i1 = 000D = 13  i2 = 10  res = FFFA = -6 | i1 = 13  i2 = 10  res = -6 |
| a = -4= FFFC  b = 1  i = 6  k = 0 | i1 = 16  i2 = FFE2=-30  res = 001E=30 | i1 = 16  i2 = -30  res = 30 |
| a = 5  b = 1  i = 4  k = -4= FFFC | i1 = 7  i2 = 4  res = 2 | i1 = 7  i2 = 4  res = 2 |
| a = 1  b = 5  i = 4  k = -4= FFFC | i1 = 10  i2 = FFEE=-18  res = 2 | i1 = 10  i2 = -18  res = 2 |

**Код программы**

AStack SEGMENT STACK

DW 12 DUP(?)

AStack ENDS

DATA SEGMENT

a DW 1

b DW 2

i DW 3

k DW 4

i1 DW 0

i2 DW 0

res DW 0

DATA ENDS

CODE SEGMENT

ASSUME CS:CODE, DS:DATA, SS:AStack

Main PROC FAR

push ds

xor ax, ax

push ax

mov ax, DATA

mov ds,ax

mov ax, a

; F1

; si = i

; di = 2i

mov si, i

mov di, si

shl di, 1

cmp ax, b

jg F3\_L

; a <= b

; cx = 3i

add di, si

mov cx, di

add cx, 4

jmp F1\_STO

F3\_L:

; a > b

mov cx, 15

sub cx, di

F1\_STO:

; store result

mov i1, cx

; F5

; di \*= 2

shl di, 1

; di = -di

neg di

cmp ax, b

jg F5\_L

; a <= b

add di, 6

jmp F5\_STO

F5\_L:

; a > b

add di, 20

F5\_STO:

; store result

mov i2, di

; F4

; ax = -i2

xor ax, ax

sub ax, i2

cmp k, 0

jge F4\_L

; k < 0

; ax = i1 - i2

add ax, i1

; ax = |i1 - i2|

cmp ax, 0

jg MODULO\_END

neg ax

MODULO\_END:

; min

cmp ax, 2

; min = ax

jl F4\_STO

mov ax, 2

; min = 2

jmp F4\_STO

F4\_L:

; else

; k >= 0

cmp ax, -6

; max = ax

jg F4\_STO

; max = -6

mov ax, -6

F4\_STO:

mov res, ax

ret 2

Main ENDP

CODE ENDS

END Main

Файл листинга

0000 AStack SEGMENT STACK

0000 000C[ DW 12 DUP(?)

????

]

0018 AStack ENDS

0000 DATA SEGMENT

0000 0001 a DW 1

0002 0002 b DW 2

0004 0003 i DW 3

0006 0004 k DW 4

0008 0000 i1 DW 0

000A 0000 i2 DW 0

000C 0000 res DW 0

000E DATA ENDS

0000 CODE SEGMENT

ASSUME CS:CODE, DS:DATA, SS:AStack

0000 Main PROC FAR

0000 1E push ds

0001 33 C0 xor ax, ax

0003 50 push ax

0004 B8 ---- R mov ax, DATA

0007 8E D8 mov ds,ax

0009 A1 0000 R mov ax, a

; F1

; si = i

; di = 2i

000C 8B 36 0004 R mov si, i

0010 8B FE mov di, si

0012 D1 E7 shl di, 1

0014 3B 06 0002 R cmp ax, b

0018 7F 0A jg F3\_L

; a <= b

; cx = 3i

001A 03 FE add di, si

001C 8B CF mov cx, di

001E 83 C1 04 add cx, 4

0021 EB 06 90 jmp F1\_STO

0024 F3\_L:

; a > b

0024 B9 000F mov cx, 15

0027 2B CF sub cx, di

0029 F1\_STO:

; store result

0029 89 0E 0008 R mov i1, cx

; F5

; di \*= 2

002D D1 E7 shl di, 1

; di = -di

002F F7 DF neg di

0031 3B 06 0002 R cmp ax, b

0035 7F 06 jg F5\_L

; a <= b

0037 83 C7 06 add di, 6

003A EB 04 90 jmp F5\_STO

003D F5\_L:

; a > b

003D 83 C7 14 add di, 20

0040 F5\_STO:

; store result

0040 89 3E 000A R mov i2, di

; F4

; ax = -i2

0044 33 C0 xor ax, ax

0046 2B 06 000A R sub ax, i2

004A 83 3E 0006 R 00 cmp k, 0

004F 7D 16 jge F4\_L

; k < 0

; ax = i1 - i2

0051 03 06 0008 R add ax, i1

; ax = |i1 - i2|

0055 3D 0000 cmp ax, 0

0058 7F 02 jg MODULO\_END

005A F7 D8 neg ax

005C MODULO\_END:

; min

005C 3D 0002 cmp ax, 2

; min = ax

005F 7C 0E jl F4\_STO

0061 B8 0002 mov ax, 2

; min = 2

0064 EB 09 90 jmp F4\_STO

0067 F4\_L:

; else

; k >= 0

0067 3D FFFA cmp ax, -6

; max = ax

006A 7F 03 jg F4\_STO

; max = -6

006C B8 FFFA mov ax, -6

006F F4\_STO:

006F A3 000C R mov res, ax

0072 CA 0002 ret 2

0075 Main ENDP

0075 CODE ENDS

END Main

Segments and Groups:

N a m e Length Align Combine Class

ASTACK . . . . . . . . . . . . . 0018 PARA STACK

CODE . . . . . . . . . . . . . . 0075 PARA NONE

DATA . . . . . . . . . . . . . . 000E PARA NONE

Symbols:

N a m e Type Value Attr

A . . . . . . . . . . . . . . . L WORD 0000 DATA

B . . . . . . . . . . . . . . . L WORD 0002 DATA

F1\_STO . . . . . . . . . . . . . L NEAR 0029 CODE

F3\_L . . . . . . . . . . . . . . L NEAR 0024 CODE

F4\_L . . . . . . . . . . . . . . L NEAR 0067 CODE

F4\_STO . . . . . . . . . . . . . L NEAR 006F CODE

F5\_L . . . . . . . . . . . . . . L NEAR 003D CODE

F5\_STO . . . . . . . . . . . . . L NEAR 0040 CODE

I . . . . . . . . . . . . . . . L WORD 0004 DATA

I1 . . . . . . . . . . . . . . . L WORD 0008 DATA

I2 . . . . . . . . . . . . . . . L WORD 000A DATA

K . . . . . . . . . . . . . . . L WORD 0006 DATA

MAIN . . . . . . . . . . . . . . F PROC 0000 CODE Length = 0075

MODULO\_END . . . . . . . . . . . L NEAR 005C CODE

RES . . . . . . . . . . . . . . L WORD 000C DATA

@CPU . . . . . . . . . . . . . . TEXT 0101h

@FILENAME . . . . . . . . . . . TEXT lab3

@VERSION . . . . . . . . . . . . TEXT 510

129 Source Lines

129 Total Lines

23 Symbols

47974 + 461317 Bytes symbol space free

0 Warning Errors

0 Severe Errors

Карта памяти

Start Stop Length Name Class

00000H 00017H 00018H ASTACK

00020H 0002DH 0000EH DATA

00030H 000A4H 00075H CODE

Program entry point at 0003:0000

Вывод

В данной лабораторной работе мы изучили операции сравнения и условного перехода. Безусловный переход осуществляется с помощью команды JMP.

Условный переход осуществляется, если выполняется определённое условие, заданное флагами процессора. Состояние флагов изменяется после выполнения арифметических, логических и некоторых других команд. Если условие не выполняется, то управление переходит к следующей команде.

Команда *CMP* сравнивает операнды и устанавливает флаги. Сравнение осуществляется путем вычитания источника из приемника.