**ЛАБОРАТОРНА РОБОТА № 5**

ОБЧИСЛЕННЯ СКЛАДНИХ ЦІЛОЧИСЛЕНИХ ВИРАЗІВ НА МОВІ ASSEMBLER

**Мета заняття**: ознайомитися з основними командами мови Assembler для обчислення складних цілочисельних виразів; набути практичних навичок в написанні програм для обчислення складних цілочисельних виразів на мові Assembler.

**Хід роботи:**

1. Написати програму для обчислення заданого цілочисленого виразу(табл. 5.5) для початкових даних в знаковому форматі довжиною 8 біт, використовуючи арифметичні операції ADD, ADC, INC, SUB, SBB, DEC, NEG, IMUL, IDIV, CBW, CWD. Провести тестові перевірки, відмітити нормальні та аномальні результати. Виконати покрокове виконання асемблерного коду та навести значення регістрів при їх виконанні

Лістинг програми:

#include "stdafx.h"

#include <stdio.h>

#include <cstdlib>

int main()

{

signed char a, f, b, c, e, res\_c, res\_asm;

printf("a = "); scanf\_s("%d", &a);

printf("b = "); scanf\_s("%d", &b);

printf("c = "); scanf\_s("%d", &c);

printf("e = "); scanf\_s("%d", &e);

printf("f = "); scanf\_s("%d", &f);

res\_c = (b + c \* b - a / 4 - e) / (a \* b - 1 + f);

printf("Result C = %d\n", res\_c);

\_\_asm {

mov al, b;

mov bl, c;

imul bl;

mov bl, 1;

idiv bl; //al=c\*b

add al, b;

mov cl, al;

mov al, a;

cbw;

mov bl, 4;

idiv bl; //al=a/4

sub cl, bl;

sub cl, e;

mov al, a;

mov bl, b;

imul bl;

mov bl, 1;

idiv bl;

dec al;

add al, f;

xchg al, cl;

cbw;

idiv cl;

mov res\_asm, al;

}

printf("Result ASM = %d\n", res\_asm);

system("Pause");

return 0;

}Результат виконання програми:

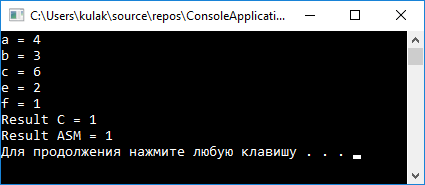


Рис. 1. Результат виконання програми

Значення регістрів при покроковому виконанні

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Крок** | **Команда** | **Значення регістру** | | | **EFLAGS/FLAGS (CF, OF)** |
| **al** | **cl** | **bl** |
| 1 | mov al, b | 3 | н/в | н/в | \_\_ |
| 2 | mov bl, c | 3 | н/в | 6 | \_\_ |
| 3 | imul bl | 18 | н/в | н/в | \_\_ |
| 4 | mov bl, 1 | 18 | н/в | 1 | \_\_ |
| 5 | idiv bl | 18 | н/в | 1 | \_\_ |
| 6 | add al, b | 21 | н/в | 1 | \_\_ |
| 7 | mov cl, al | 21 | 21 | 1 | \_\_ |
| 8 | mov al, a | 4 | 21 | 1 | \_\_ |
| 9 | mov bl, 4 | 4 | 21 | 4 | \_\_ |
| 10 | idiv bl | 1 | 21 | 4 | \_\_ |
| 11 | sub cl, al | 1 | 20 | 4 | \_\_ |
| 12 | sub cl, e | 1 | 20 | 4 | \_\_ |
| 13 | mov al, a | 4 | 20 | 4 | \_\_ |
| 14 | mov bl, b | 4 | 20 | 3 | \_\_ |
| 15 | imul bl | 12 | 20 | 3 | \_\_ |
| 16 | mov bl, 1 | 12 | 20 | 1 | \_\_ |
| 17 | idiv bl | 12 | 20 | 1 | \_\_ |
| 18 | dec al | 11 | 20 | 1 | \_\_ |
| 19 | add al, f | 12 | 20 | 1 | \_\_ |
| 20 | xchg al, cl | 20 | 12 | 1 | \_\_ |
| 21 | idiv cl | 1 | 12 | 1 | \_\_ |
| 22 | mov res\_asm, al | 1 | 12 | 1 |  |
|  | | | | | |

2. Написати програму для обчислення заданого цілочисленого виразу

(табл. 5.5) для початкових даних в знаковому форматі довжиною 16 біт, використовуючи арифметичні операції ADD, ADC, INC, SUB, SBB, DEC, NEG, IMUL, IDIV, CBW, CWD. Провести тестові перевірки, відмітити нормальні та аномальні результати. Виконати покрокове виконання асемблерного коду та навести значення регістрів при їх виконанні.

Лістинг програми:

#include "pch.h"

#include <stdio.h>

#include <cstdlib>

int main()

{

short int a, f, b, c, e, res\_c, res\_asm;

printf("a = "); scanf\_s("%d", &a);

printf("b = "); scanf\_s("%d", &b);

printf("c = "); scanf\_s("%d", &c);

printf("e = "); scanf\_s("%d", &e);

printf("f = "); scanf\_s("%d", &f);

res\_c = (b + c \* b - a / 4 - e) / (a \* b - 1 + f);

printf("Result C = %d\n", res\_c);

\_\_asm {

mov ax, b;

mov bx, c;

imul bx;

mov bx, 1;

idiv bx; //ax=c\*b

add ax, b;

mov cx, ax;

mov ax, a;

mov bx, 4;

idiv bx; //ax=a/4

sub cx, ax;

sub cx, e;

mov ax, a;

mov bx, b;

imul bx;

mov bx, 1;

idiv bx;

dec ax;

add ax, f;

xchg ax, cx;

idiv cx;

mov res\_asm, ax;

}

printf("Result ASM = %d\n", res\_asm);

system("Pause");

return 0;

}Результат виконання програми:

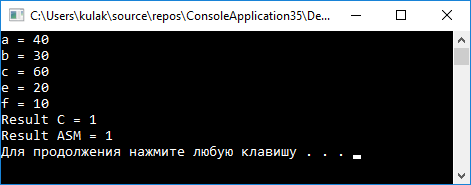


Рис. 2. Результат виконання програми

Значення регістрів при покроковому виконанні

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Крок** | **Команда** | **Значення регістру** | | | **EFLAGS/FLAGS (CF, OF)** |
| **ax** | **cx** | **bx** |
| 1 | mov ax, b | 30 | н/в | н/в | \_\_ |
| 2 | mov bx, c | 30 | н/в | 60 | \_\_ |
| 3 | imul bx | 180 | н/в | н/в | \_\_ |
| 4 | mov bx, 1 | 180 | н/в | 10 | \_\_ |
| 5 | idiv bx | 180 | н/в | 10 | \_\_ |
| 6 | add ax, b | 210 | н/в | 10 | \_\_ |
| 7 | mov cx, ax | 210 | 210 | 10 | \_\_ |
| 8 | mov ax, a | 40 | 210 | 10 | \_\_ |
| 9 | mov bx, 4 | 40 | 210 | 40 | \_\_ |
| 10 | idiv bx | 10 | 210 | 40 | \_\_ |
| 11 | sub cx, ax | 10 | 200 | 40 | \_\_ |
| 12 | sub cx, e | 10 | 200 | 40 | \_\_ |
| 13 | mov ax, a | 40 | 200 | 40 | \_\_ |
| 14 | mov bx, b | 40 | 200 | 30 | \_\_ |
| 15 | imul bx | 120 | 200 | 30 | \_\_ |
| 16 | mov bx, 1 | 120 | 200 | 10 | \_\_ |
| 17 | idiv bx | 120 | 200 | 10 | \_\_ |
| 18 | dec ax | 110 | 200 | 10 | \_\_ |
| 19 | add ax, f | 120 | 200 | 10 | \_\_ |
| 20 | xchg ax, cx | 200 | 120 | 10 | \_\_ |
| 21 | idiv cx | 1 | 120 | 10 | \_\_ |
| 22 | mov res\_asm, ax | 1 | 120 | 10 |  |
|  | | | | | |

3. Написати програму для обчислення заданого цілочисленого виразу(табл. 5.5) для початкових даних в знаковому форматі довжиною 32 біт, використовуючи арифметичні операції ADD, ADC, INC, SUB, 12 SBB, DEC, NEG, IMUL, IDIV, CBW, CWD. Провести тестові перевірки, відмітити нормальні та аномальні результати. Виконати покрокове виконання асемблерного коду та навести значення регістрів при їх виконанні.

Лістинг програми:

#include "pch.h"

#include <stdio.h>

#include <cstdlib>

int main()

{

int a, f, b, c, e, res\_c, res\_asm;

printf("a = "); scanf\_s("%d", &a);

printf("b = "); scanf\_s("%d", &b);

printf("c = "); scanf\_s("%d", &c);

printf("e = "); scanf\_s("%d", &e);

printf("f = "); scanf\_s("%d", &f);

res\_c = (b + c \* b - a / 4 - e) / (a \* b - 1 + f);

printf("Result C = %d\n", res\_c);

\_\_asm {

mov eax, b;

mov ebx, c;

imul ebx;

mov ebx, 1;

idiv ebx; //ax=c\*b

add eax, b;

mov ecx, eax;

mov eax, a;

mov ebx, 4;

idiv ebx; //ax=a/4

sub ecx, eax;

sub ecx, e;

mov eax, a;

mov ebx, b;

imul ebx;

mov ebx, 1;

idiv ebx;

dec eax;

add eax, f;

xchg eax, ecx;

idiv ecx;

mov res\_asm, eax;

}

printf("Result ASM = %d\n", res\_asm);

system("Pause");

return 0;

}Результат виконання програми:

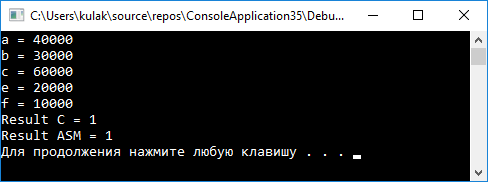


Рис. 3. Результат виконання програми

Значення регістрів при покроковому виконанні

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Крок** | **Команда** | **Значення регістру** | | | **EFLAGS/FLAGS (CF, OF)** |
| **eax** | **ecx** | **ebx** |
| 1 | mov eax, b | 30000 | н/в | н/в | \_\_ |
| 2 | mov ebx, c | 30000 | н/в | 60000 | \_\_ |
| 3 | imul ebx | 180000 | н/в | н/в | \_\_ |
| 4 | mov ebx, 1 | 180000 | н/в | 10000 | \_\_ |
| 5 | idiv ebx | 180000 | н/в | 10000 | \_\_ |
| 6 | add eax, b | 210000 | н/в | 10000 | \_\_ |
| 7 | mov ecx, eax | 210000 | 210000 | 10000 | \_\_ |
| 8 | mov eax, a | 40000 | 210000 | 10000 | \_\_ |
| 9 | mov ebx, 4 | 40000 | 210000 | 40000 | \_\_ |
| 10 | idiv ebx | 10000 | 210000 | 40000 | \_\_ |
| 11 | sub ecx, eax | 10000 | 200000 | 40000 | \_\_ |
| 12 | sub ecx, e | 10000 | 200000 | 40000 | \_\_ |
| 13 | mov eax, a | 40000 | 200000 | 40000 | \_\_ |
| 14 | mov ebx,b | 40000 | 200000 | 30000 | \_\_ |
| 15 | imul ebx | 120000 | 200000 | 30000 | \_\_ |
| 16 | mov ebx, 1 | 120000 | 200000 | 10000 | \_\_ |
| 17 | idiv ebx | 120000 | 200000 | 10000 | \_\_ |
| 18 | dec eax | 110000 | 200000 | 10000 | \_\_ |
| 19 | add eax, f | 120000 | 200000 | 10000 | \_\_ |
| 20 | xchg eax,ecx | 200000 | 120000 | 10000 | \_\_ |
| 21 | idiv ecx | 1 | 120000 | 10000 | \_\_ |
| 22 | mov res\_asm, ax | 1 | 120000 | 10000 |  |
|  | | | | | |

Рис 4. Помилка при введенні чисел поза діапазоном

***Висновки:*** в ході виконання лабораторної роботи було ознайомлено з основними командами мови Assembler для обчислення складних цілочисельних виразів; набули практичних навичок в написанні програм для обчислення складних цілочисельних виразів на мові Assembler