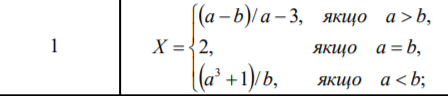
**Лабораторна робота № 6**

**ОРГАНІЗАЦІЯ УМОВНИХ ПЕРЕХОДІВ**

Мета: ознайомитися з основними ко- мандами мови Assembler дляорганізації умовних переходів; набути практичних навичок в написанні програм з організацією умовних переходів на мові Assembler.

**Хід роботи:**

1. Написати програму для обчислення заданого умовного цілочисельного виразу (табл. 6.3) для 8- бітних даних, використовуючи команди порівняння, умовного і безумовного переходів. Результат Х – теж цілочисельний і його діапазон (формат) залежить від специфіки вирішуваного умовного виразу. Провести тестові перевірки, відмітити нормальні та аномальні результати. Виконати покрокове виконання асем- блерного коду та навести значення регістрів при їх виконанні.



Лістинг програми:

#include <Windows.h>

#include <stdio.h>

int main() {

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

signed char a, b, res, dil = 0, per = 0, res\_asm;

printf("a[-128; 127] = "); scanf\_s("%hhi", &a);

printf("b[-128; 127] = "); scanf\_s("%hhi", &b);

if (a > b) {

if (a == 0) {

printf("Error:");

}

else {

res = (a - b) / a - 3;

}

}

else if (a == b) {

res = 2;

}

else if (a < b) {

if (b == 0) {

printf("Error:");

}

else {

res = (a \* a \* a + 1) / b;

}

}

\_\_asm {

mov al, a;

mov bl, b;

cmp al, bl;

jg mark1; //a > b

je mark2; //a = b

jl mark3; //a < b

mark1: //a > b

cmp al, 0;

je error1; //ділення на 0

sub al, bl;

cbw;

mov bl, a;

idiv bl;

jo error2;

sub al, 3;

mov res\_asm, al;

jmp ext;

mark2: //a = b

mov res\_asm, 2;

jmp ext;

mark3://a < b

cmp bl, 0;

je error1; //ділення на 0

imul al;

mov bl, 1;

idiv bl;

jo error2;

mov bl, a;

imul bl;

jo error2;

inc ax;

mov bl, b;

idiv bl;

mov res\_asm, al;

jmp ext;

//errors

error1: //ділення на 0

mov dil, 1;

jmp ext;

//exit

error2 :

mov per, 1;

jmp ext;

ext:

}

if (dil > 0) {

printf("Ділення на 0!\n");

} else if (per > 0) {

printf("Переповнення!\n");

} else if (dil == 0 && per == 0) {

printf("res = %hhi,\nres\_asm = %hhi\n", res, res\_asm);

}

system("pause");

return 0;

}

Результат виконанняпрограми:

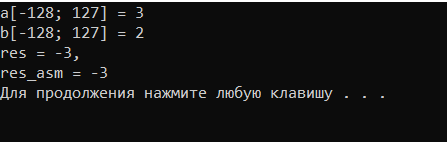


Рисунок 1.1 – Перевірка роботи програми для значен- ня a > b

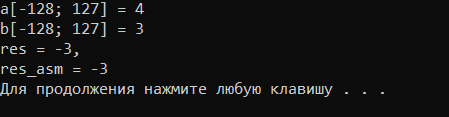


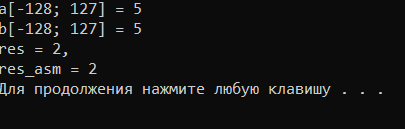
Рисунок 1.2 – Перевірка роботи програми для значен- ня a<b 

Рисунок 1.3 – Перевірка роботи програми для значен- ня a = b

Рис. 1. Результат виконанняпрограми

2. Написати програму для обчислення заданого умовного цілочисельного виразу (табл. 6.3) для 16- бітних даних, використовуючи команди порівняння, умовного і безумовного переходів. Результат Х – теж цілочисельний і його діапазон (формат) залежить від специфіки вирішуваного умовного виразу. Провести тестові перевірки, відмітити нормальні та аномальні результати. Виконати покрокове виконання асем- блерного коду та навести значення регістрів при їх виконанні.

Лістинг програми:

#include <Windows.h>

#include <stdio.h>

int main() {

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

short int a, b, res, dil = 0, per = 0, res\_asm;

printf("a[-128; 127] = "); scanf\_s("%hi", &a);

printf("b[-128; 127] = "); scanf\_s("%hi", &b);

if (a > b) {

if (a == 0) {

printf("Error:");

}

else {

res = (a - b) / a - 3;

}

}

else if (a == b) {

res = 2;

}

else if (a < b) {

if (b == 0) {

printf("Error:");

}

else {

res = (a \* a \* a + 1) / b;

}

}

\_\_asm {

mov ax, a;

mov bx, b;

cmp ax, bx;

jg mark1; //a > b

je mark2; //a = b

jl mark3; //a < b

mark1: //a > b

cmp ax, 0;

je error1; //ділення на 0

sub ax, bx;

cwd;

mov bx, a;

idiv bx;

jo error2;

sub ax, 3;

mov res\_asm, ax;

jmp ext;

mark2: //a = b

mov res\_asm, 2;

jmp ext;

mark3://a < b

cmp bx, 0;

je error1; //ділення на 0

imul ax;

mov bx, 1;

idiv bx;

jo error2;

mov bx, a;

imul bx;

jo error2;

inc eax;

mov bx, b;

idiv bx;

mov res\_asm, ax;

jmp ext;

//errors

error1: //ділення на 0

mov dil, 1;

jmp ext;

//exit

error2:

mov per, 1;

jmp ext;

ext:

}

if (dil > 0) {

printf("Ділення на 0!\n");

}

else if (per > 0) {

printf("Переповнення!\n");

}

else if (dil == 0 && per == 0) {

printf("res = %hi,\nres\_asm = %hi\n", res, res\_asm);

}

system("pause");

return 0;

}Результат виконанняпрограми:

Рисунок 2.1 – Перевірка роботи програми для значен- ня a > b

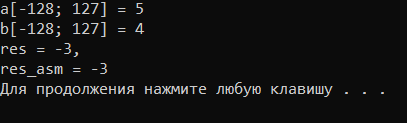


Рисунок 2.2 – Перевірка роботи програми для значен- ня a < b

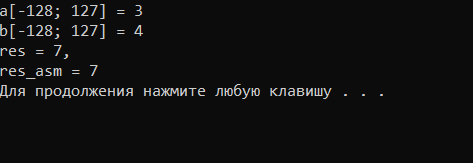
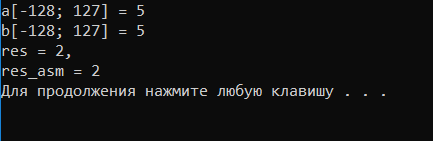


Рисунок 2.3 – Перевірка роботи програми для значен- ня a = b 

3. Написати програму для обчислення заданого умовного цілочисельного виразу (табл. 6.3) для 32- бітних даних, використовуючи команди порівняння, умовного і безумовного переходів. Результат Х – теж цілочисельний і його діапазон (формат) залежить від специфіки вирішуваного умовного виразу. Провести тестові перевірки, відмітити нормальні та аномальні результати. Виконати покрокове виконання асем- блерного коду та навести значення регістрів при їх виконанні.

Лістинг програми:

#include <Windows.h>

#include <stdio.h>

int main() {

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

int a, b, res, dil = 0, per = 0, res\_asm;

printf("a[-128; 127] = "); scanf\_s("%d", &a);

printf("b[-128; 127] = "); scanf\_s("%d", &b);

if (a > b) {

if (a == 0) {

printf("Error:");

}

else {

res = (a - b) / a - 3;

}

}

else if (a == b) {

res = 2;

}

else if (a < b) {

if (b == 0) {

printf("Error:");

}

else {

res = (a \* a \* a + 1) / b;

}

}

\_\_asm {

mov eax, a;

mov ebx, b;

cmp eax, ebx;

jg mark1; //a > b

je mark2; //a = b

jl mark3; //a < b

mark1: //a > b

cmp eax, 0;

je error1; //ділення на 0

sub eax, ebx;

cdq;

mov ebx, a;

idiv ebx;

jo error2;

sub eax, 3;

mov res\_asm, eax;

jmp ext;

mark2: //a = b

mov res\_asm, 2;

jmp ext;

mark3://a < b

cmp ebx, 0;

je error1; //ділення на 0

imul eax;

mov ebx, 1;

idiv ebx;

jo error2;

mov ebx, a;

imul ebx;

jo error2;

//inc edx;

mov ebx, b;

idiv ebx;

mov res\_asm, eax;

jmp ext;

//errors

error1: //ділення на 0

mov dil, 1;

jmp ext;

//exit

error2:

mov per, 1;

jmp ext;

ext:

}

if (dil > 0) {

printf("Ділення на 0!\n");

}

else if (per > 0) {

printf("Переповнення!\n");

}

else if (dil == 0 && per == 0) {

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

}

system("pause");

return 0;

}

Результат виконанняпрограми:

Рисунок 3.1 – Перевірка роботи програми для значен- ня a > b

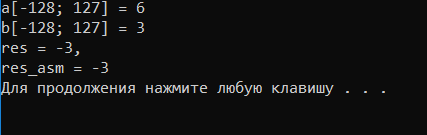


Рисунок 3.2 – Перевірка роботи програми для значен- ня a < b

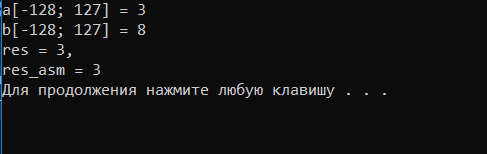
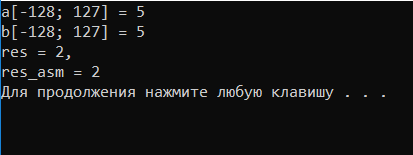


Рисунок 3.3 – Перевірка роботи програми для значен- ня a = b 

**Висновок:**Ознайомився з основними ко- мандами мови Assembler дляорганізації умовних переходів; набути практичних навичок в написанні програм з організацією умовних переходів на мові.Переповнення не можливо виконати через нову версію [Visual Studio](https://visualstudio.microsoft.com/)