**КПІ ім. Ігоря Сікорського**

**Інститут прикладного системного аналізу**

**Кафедра Системного проектування**

Лабораторна рoбота №2

«Дослідження роботи АЛУ»

Виконав:

Студент групи ДА-92

ННК «ІПСА»

Насікан Дмитро Юрійович

Варіант № 11

Київ – 2021 рік

**ЗАВДАННЯ**

1. Власноруч представити константи, задані в таблиці 1, у форматі двійкових чисел з фіксованою крапкою в двійковому і шістнадцятковому коді, а також в упакованому форматі десяткових чисел. Ознайомитись з синтаксисом введення чисельних констант в асемблері: EQU, 1.2, -12, 0123H, 0ABCDH, 01010b.

2. Пояснити призначення регістрів ЦПУ, що використовуються АЛУ, включаючи регістр флагів.

3. Вивчити команди виконання арифметичних операцій над даними з фіксованою комою ADD, ADC, SUB, SBB, CMP, MUL, IMUL, DIV, IDIV, CWD, CWB, NEG, INC, DEC.

4. Вивчити команди виконання логічних операцій AND, OR, XOR, TEST.

5. Вивчити команди виконання регістрових операцій SHL, SHR.

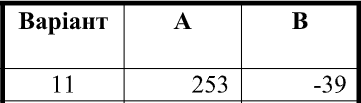
6. Продемонструвати їх роботу за допомогою демонстраційної програми виду:

С=(A/B+(A%B)\*(A%B)/3-A\*B)<<2; або подібного розрахунку, з використанням більшості досліджених команд.

7. Написати програму мовою С або С++, яка виконує ті самі дії, декларувавши операнди як “int register” (в цьому випадку, операнди розміщуються не в ОЗУ, а в регістрах процесора). Передивитись сгенерований компілятором код в режимі дебагеру як дезасемблований. Порівняти з власною реалізацією. В якості середовища можна використати будь-який, зручний Вам, компілятор. Наприклад, перевірено працездатність кросплатформеного середовища “CodeLite IDE” (https://codelite.org/) в режимі “simple console gcc project”.

8. Представити A/B та B/A як 16-бітові числа з фіксованою комою, виділивши достатню кількість розрядів для цілої частини (з запасом для складання і множення). Виконати операції складання та множення з цими числами.

9. Вивчити команди корекції результаів виконання операцій над двійково-десятковими числами в упакованому форматі (DAA, DAS). Написати і виконати програму на мові Асемблера для демонстрації складання і віднімання двійково-десяткових чисел для всіх можливих варіантів виконання корекції результату.



**ХІД РОБОТИ**

1. Представити константи, задані в таблиці 1, у форматі двійкових чисел з фіксованою крапкою в двійковому і шістнадцятковому коді, а також в упакованому форматі десяткових чисел:

У форматі двійкового коду:

25310 = 111111012

-3910 = 110110012 (доповнений код)

У форматі двійкових чисел з фіксованою крапкою:

25310 = 111111012

-3910 = 110110012

У шіснадцятковому форматі:

25310 = FD16

-3910 =FFD916

В упакованому форматі десяткових чисел:

25310 = 00100101 0011pBCD

-3910 = 100000111001pBCD

1. Призначення регістрів ЦПУ, що використовуються АЛУ, включаючи регістр флагів:

Регістри діляться на 3 категорії:

* Загальні регістри
* Регістри керування
* Сегментні регістри

Загальні регістри:

Регістри даних (EAX, EBX, ECX, EDX) – 4 32 бітних регістра, які використовуються для арефметичних, логічних та інших операцій.

AX – використовується для вводу/виводу даних під час арефметичних операцій.

BX – використовується при індексованій адресації.

CX – зберігає кількість циклів при ітераційному процесі.

DX – використовуюється у зв’язці з AX для виконання операцій множення та ділення великих значень.

Регістри-вказівники:

IP – зберігає зміщення адреси наступної команди, яка повинна бути виконана.

SP – забезпечує значення зміщення в програмному стеку.

BP – використовується при передачі параметрів у підпрограми.

Індексні регістри:

SI – використовується в якості вихідного індекса в рядковій операції.

DI – використовується в якості індекса призначення в рядкових операціях.

Регістри керування – 32 бітний регістр флагів, що показує поточний стан процесора.

Сегментні регістри – зберігають дані, код програми та стек.

CS – містить усі команди та інструкції які повинні бути виконані.

DS – містить дані, константи та робочі області.

SS – містить початкову адресу стека.

1. Продемонструємо роботу операторів за допомогою демонстраційної програми:

Лістинг:

%include "io.inc"

A EQU 253

B EQU -39

section .text

global CMAIN

CMAIN:

mov ebp, esp; for correct debugging

;write your code here

MOV AX, A

MOV BL, B

IDIV BL ; A/B

MOV CH, AL

MOV AL, AH

MUL AH ;(A%A)^2

MOV BH, 3

DIV BH ; (A%A)^2/3

MOV CL, AL

MOV AL, A

MOV BL, 39

MUL BL

ADD CH, CL

ADD AL, CH

SHL AX, 2

xor eax, eax

ret

1. Напишемо програму на C++ та дизасемблюємо її:

Лістинг:

int a = 253;

00C04148 mov dword ptr [a], FD

int b = -39;

00C0414F mov dword ptr [b], 7FD9

float c = a / b + (a % b) \* (a % b) / 3 - a \* b;

00C04156 mov eax,dword ptr [a]

00C04159 cdq

00C0415A idiv eax,dword ptr [b]

00C0415D mov ecx,eax

00C0415F mov eax,dword ptr [a]

00C04162 cdq

00C04163 idiv eax,dword ptr [b]

00C04166 mov esi,edx

00C04168 mov eax,dword ptr [a]

00C0416B cdq

00C0416C idiv eax,dword ptr [b]

00C0416F imul esi,edx

00C04172 mov eax,esi

00C04174 cdq

00C04175 mov esi,3

00C0417A idiv eax,esi

00C0417C add ecx,eax

00C0417E mov edx,dword ptr [a]

00C04181 imul edx,dword ptr [b]

00C04185 sub ecx,edx

00C04187 cvtsi2ss xmm0,ecx

00C0418B movss dword ptr [c],xmm0

Як бачимо, код майже не відрізняється від написаного в демонстраційній програмі, окрім того, що тут використовуються змінні а не констранти, та проводяться деякі службові операції.

1. Представимо A/B та B/A як 16-бітові числа з фіксованою комою та виконаємо операції додавання й множення з ними:

Формат 1.3:

-6.487 \* 10^3 = -6487 = E6A9

-0.154 \* 10^3 = -154 = FF66

E6A9 + FF66 = E60F = -6.641 \* 10^3

E6A9\*FF66 = F3E56 = - 0. 998998 \* 10^6

%include "io.inc"

A\_B EQU 0hE6A9

B\_A EQU 0hFF66

section .text

global CMAIN

CMAIN:

MOV EBP, ESP; for correct debugging

;write your code here

MOV AX, A\_B

ADD AX, B\_A

XOR EAX, EAX

MOV AX, A\_B

MOV DX, B\_A

IMUL DX

XOR EAX, EAX

Ret

1. Вивчити команди корекції результатів виконання операцій над двійково-десятковими числами в упакованому форматі (DAA, DAS). Написати і виконати програму на мові Асемблера для демонстрації складання і віднімання двійково-десяткових чисел для всіх можливих варіантів виконання корекції результату:

DAA інструкція:

Якшо останні 4 біти числа в регістрі АL після додавання більші за 9(у двійковому форматі) або якщо встановлено флаг AF, додає до числа в регістрі AL 6 для відновлення структури упакованого двійково-десяткового числа.

Якшо перші 4 біти числа в регістрі АL після додавання більші за 9(у двійковому форматі) або якщо встановлено флаг CF, додає до числа в регістрі AL 60h для відновлення структури упакованого двійково-десяткового числа.

Приклад 1:

%include "io.inc"

A EQU 29h

B EQU 69h

section .text

global CMAIN

CMAIN:

mov ebp, esp

MOV AL, A

ADD AL, B

DAA

xor eax, eax

ret

Після виконання операції додавання число в AL має вигляд: 92 (10010010) та в регістрі флагів піднятий флаг AF.

Після виконання DAA число в AL має вигляд: 98 (10011000), що є правильним результатом додавання.

Приклад 2:

%include "io.inc"

A EQU 27h

B EQU 34h

section .text

global CMAIN

CMAIN:

mov ebp, esp

MOV AL, A

ADD AL, B

DAA

xor eax, eax

ret

Після виконання операції додавання число в AL має вигляд: 5B (01011101) та в регістрі флагів піднятий флаг AF.

Після виконання DAA число в AL має вигляд: 61 (01100001), що є правильним результатом додавання.

Приклад 3:

%include "io.inc"

A EQU 52h

B EQU 61h

section .text

global CMAIN

CMAIN:

mov ebp, esp

MOV AL, A

ADD AL, B

DAA

xor eax, eax

ret

Після виконання операції додавання число в AL має вигляд: B3 (10110010) та в регістрі флагів піднятий флаг AF.

Після виконання DAA число в AL має вигляд: 13 (00010011), що є правильним результатом додавання, враховуючи флаг CF.

DAS інструкція:

Якшо останні 4 біти числа в регістрі АL після віднімання більші за 9(у двійковому форматі) або якщо встановлено флаг AF, віднімає від числа в регістрі AL 6 для відновлення структури упакованого двійково-десяткового числа.

Якшо перші 4 біти числа в регістрі АL після відніманн більші за 9(у двійковому форматі) або якщо встановлено флаг CF, віднімає від числа в регістрі AL 60h для відновлення структури упакованого двійково-десяткового числа.

Приклад:

%include "io.inc"

A EQU 71h

B EQU 43h

section .text

global CMAIN

CMAIN:

mov ebp, esp

MOV AL, A

SUB AL, B

DAA

xor eax, eax

ret

Після виконання операції віднімання число в AL має вигляд: 2Е та в регістрі флагів піднятий флаг AF.

Після виконання DAS число в AL має вигляд: 28 (00101000), що є правильним результатом віднімання.

**ВИСНОВКИ**

У ході даної роботи мною було освоєно основні арефметичні та логічні операції ALU. Я попрактикувався переводити числа в різні системи числення та формати, працювати з різними представленнями числа в мові асемблеру. Також, була продемонстрована робота основних арефметичних операцій на мові асемблеру, скаладена й дизасембльована така ж програма на мові програмування С.