**КПІ ім. Ігоря Сікорського**

**Інститут прикладного системного аналізу**

**Кафедра Системного проектування**

Лабораторна рoбота №3

«Дослідження роботи арифметичного співпроцесору.»

Виконав:

Студент групи ДА-92

ННК «ІПСА»

Насікан Дмитро Юрійович

Варіант № 11

Київ – 2021 рік

**ЗАВДАННЯ**

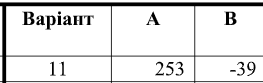
1. Дослідити структуру регістрів арифметичного сопроцесору та намалювати її діаграму.

2. Вивчити формати представлення чисел з плаваючою комою. Представити задані константи у форматі 4-байтових чисел з плаваючою комою. Звернути увагу на особливі випадки (нескінченність, не число і тому подібне).

3. Завантажити константи до FPU за допомогою команд: команди: FINIT, FLD, FST, FLI.

4. Виконати над константами опреації додавання, ділення, множення, перетворення типів даних.

5. 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”.

6. Внести до складу протоколу результати роботи програм з вікна дебагера. 

**ХІД РОБОТИ**

1. Дослідимо структуру регістрів арифметичного сопроцесору та намалювати її діаграму:

У FPU можна виділити три групи регістрів:

**Стек процесора**: регістри ST0..ST7. Розмірність кожного регістру: 80 біт.

службові регістри

**Регістр стану процесора SWR** (Status Word Register) – зберігає інформацію про поточний стан співпроцесора. Розмірність: 16 біт.

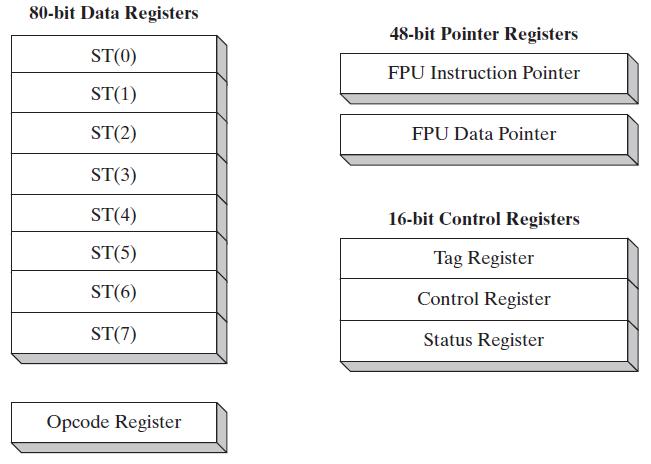
**Керуючий регістр співпроцесора CWR** (Control Word Register) - управління режимами роботи співпроцесора. Розмірність: 16 біт.

**Регістр слова тегів TWR** (Tags Word Register) - контроль над регістрами R0..R7 (наприклад, для визначення можливості запису). Розмірність: 16 біт.

регістри покажчиків

**Покажчик даних DPR** (Data Point Register). Розмірність: 48 біт.

Покажчик команд IPR (Instruction Point Register). Розмірність: 48 біт.



1. Представимо задані константи у форматі 4-байтових чисел з плаваючою комою. Звернемо увагу на особливі випадки (нескінченність, не число і тому подібне):

Плаваюча точка (floating point) - метод представлення дійсних чисел, при якому число зберігається у вигляді мантиси і показника ступеня, а значення числа обчислюється за формулою:

x = (-1)^{sign} \times mant \times base^{exp}

Форми мантиси:

Нормальною називається форма подання числа, при якій абсолютне значення мантиси десяткового числа знаходиться на [0,1)

Нормалізованою називається форма подання числа, при якій абсолютне значення мантиси десяткового числа лежить на [1,10), а двійкового – [1,2).

Число з плаваючою точкою зберігається в нормалізованій формі і складається з трьох частин (в дужках вказано кількість біт, що відводяться на кожну секцію в форматі double):

1. знак (1)

2. експонента (показник степені) (у вигляді цілого числа в коді із зсувом) (11)

3. мантиса (в нормалізованій формі) (53)

За базу (основу степені) використовується число 2.

Особливі випадки:

1. Число 0:

Знак – 0/1

Експонента – усі нулі.

Мантиса – усі нулі

1. Безкінченість зі знаком:

Знак – 0/1

Експонента – усі одиниці

Мантиса – усі нулі

1. Невизначеність:

Знак – 0/1

Експонента – усі 1

Мантиса – усі 0/1

Представимо задані варіантом числа у формі плаваючої коми:

**253:**

25310 = 111111012

Нормалізуємо:

11111101 = 1,1111101 \*

Знайдемо зсунутий порядок:

0111111 + 00000111 = 01000110

Отже, 253 у 32розрядному записі матиме вигляд:

0 01000110 11111010000000000000000

**-39:**

25310 = 1001112

Нормалізуємо:

100111= 1,00111\*

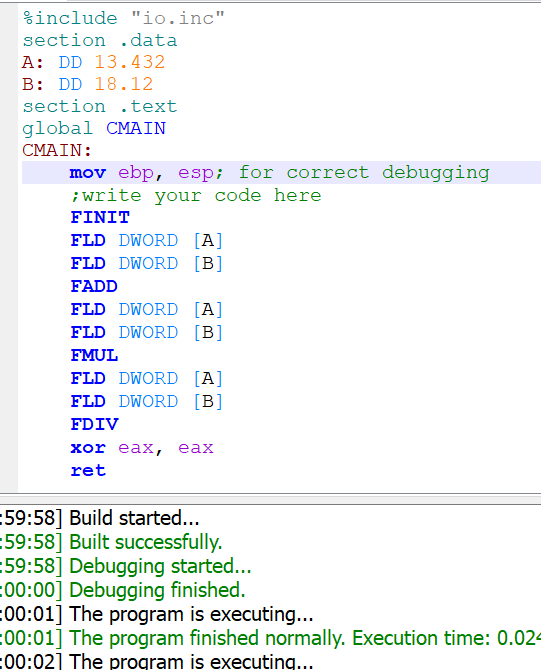
Знайдемо зсунутий порядок:

0111111 + 00000101 = 01000100

Отже, 253 у 32розрядному записі матиме вигляд:

1 01000100 00111000000000000000000

1. Завантажимо константи до FPU за допомогою команд: FINIT, FLD, FST, FLI.



1. Складемо програму на мові програмування C, яка виконує ті самі дії, та подивимося її код у режимі дезасемблювання:

int main(){

float A = 13.432;

float B = 18.12;

float res = A + B;

res = A \* B;

res = A / B;

return 0;

}

Код на асемблері:

push rbp

mov rbp, rsp

mov DWORD PTR [rbp-20], edi

movss xmm0, DWORD PTR .LC0[rip]

movss DWORD PTR [rbp-4], xmm0

movss xmm0, DWORD PTR .LC1[rip]

movss DWORD PTR [rbp-8], xmm0

movss xmm0, DWORD PTR [rbp-4]

addss xmm0, DWORD PTR [rbp-8]

movss DWORD PTR [rbp-12], xmm0

movss xmm0, DWORD PTR [rbp-4]

mulss xmm0, DWORD PTR [rbp-8]

movss DWORD PTR [rbp-12], xmm0

movss xmm0, DWORD PTR [rbp-4]

divss xmm0, DWORD PTR [rbp-8]

movss DWORD PTR [rbp-12], xmm0

mov eax, 0

pop rbp

ret

**ВИСНОВКИ**

У ході цієї лабораторної роботи було досліджено роботу арифметичного співпроцесора FPU, його регістрів. Було перетворено задані числа у формат представлення чисел з плаваючою комою та основні команди асемблера для роботи з цими числами: їхнього завантаження у стек процесора, основні математичні операції, команди FINIT, FLD, FST. Ці команди було використано в прикладі, який був написаний на асемблері для тестування роботи FPU. Також, була розроблена програма, яка виконує ті самі дії на мові C, й дизасембльована. Лістинги програм наведені вище.