ДОДАТКОВА ЛАБОРАТОРНА робота №6*.

**виконується групами КІ-307, КІ-308 та КІ-309 у разі несвоєчасного виконання основних лабораторних*

Укладачі: Мархивка В.С., ст. викладач,
Олексів М. В., асистент, к.т.н.,
Мороз І.В., ст. викладач, к.т.н.,
Акимишин О.І., доцент, к.т.н..

ВИКОРИСТАННЯ МАТЕМАТИЧНОГО СПІВПРОЦЕСОРА

Мета: ознайомитися з принципами роботи математичного співпроцесора і використати його можливості для обчислення арифметичних виразів з числами з плаваючою комою.

ТЕОРЕТИЧНІ ВІДОМОСТІ

~~Математичний співпроцесор запускається центральним процесором. Після запуску він виконує всі обрахунки самостійно і паралельно з роботою центрального процесора. Якщо центральний процесор направляє наступну команду співпроцесору в той момент, коли співпроцесор ще не закінчив виконання попередньої команди, центральний процесор переводиться в стан очікування. Якщо ж співпроцесор не зайнятий, центральний процесор, направивши команду співпроцесору, продовжує свою роботу, не очікуючи завершення обрахунку. Послідовне розташування команд співпроцесора і центрального процесора в коді програми створюють ілюзію послідовного їх виконання. Простіше кажучи можлива така ситуація, що коли центральний процесор звернеться до комірки пам'яті, куди арифметичний співпроцесор повинен був записати результат своїх обчислень і якщо співпроцесор ще не закінчив обрахунки то результату там не буде. Однак є спеціальні засоби синхронізації (команда FWAIT), що дозволять центральному процесорові дочекатися результатів роботи співпроцесора.~~

Префікс команд та адресація операндів

Команди які призначенні для виконання співпроцесором, записуються в програмі як звичайні машинні команди центрального процесора, але всі вони починаються з байта, який відповідає команді центрального процесора ESC. Зустрівши таку команду, процесор передає її співпроцесору, а сам продовжує виконання програми з наступної команди.

Асемблерна мнемоніка всіх команд співпроцесора починається з букви F, наприклад: FADD, FDIV, FSUB и так далі. Команди співпроцесора можуть адресувати операнди, аналогічно звичайним командам центрального

процесора. Операндами можуть бути або дані, які розміщуються в основній пам’яті комп’ютера, або внутрішні регістри співпроцесора.

Для команд арифметичного співпроцесора можливі всі види адресації даних, які використовуються центральним процесором.

Формати даних

Математичний співпроцесор може обробляти дійсні (у форматі з плаваючою комою) та цілі числа.

Дійсні числа. Дійсні числа в загальних обрахунках можна записати наступним чином:

(знак)(мантиса)*10(знак)(порядок)

Наприклад: $-1.35*10^5$ (мінус одна ціла тридцять п’ять сотих, помножені на десять в п’ятому степені). Тут знак: мінус, мантиса: 1.35, порядок: 5. Порядок теж може мати знак.

Важливим є таке поняття, як **нормалізоване представлення** чисел: якщо ціла частина мантиси числа складається з однієї цифри, не рівної нулю, то число з плаваючою крапкою називається нормалізованим. Перевага використання нормалізованих чисел полягає в тому, що для фіксованої розрядної сітки числа (тобто для фіксованої кількості цифр в числі) нормалізовані числа мають найбільшу точність. Крім того, нормалізоване представлення виключає неоднозначність, яка може виникнути через те, що кожне число з плаваючою крапкою може бути представлене різними (ненормалізованими) способами (наприклад: 123.5678*10^5 = 12.35678*10^6 = 1.235678*10^7 = 0.1235678*10^8).

При програмуванні на мовах високого рівня зустрічається наступне представлення **чисел з плаваючою крапкою:**

(знак)(мантиса)E(знак)(порядок)

Наприклад, -5.35E-2 означає число $-5.35*10^{-2}$ (мінус п’ять цілих тридцять п’ять сотих, помножені на десять в мінус другому ступені).

Арифметичний співпроцесор може працювати з дійсними числами в трьох форматах:

- **одинарної точності** (4 байти);
- **подвійної точності** (8 байт);
- **розширеної точності** (10 байт).

При будь-якому представленні старший біт визначає знак дійсного числа: 0 - додатне, 1 – від’ємне. Числа, що рівні за модулем, відрізняються лише цим бітом, оскільки для представлення від’ємних чисел *доповняльний код не використовується*, на відміну від центрального процесора.

Арифметичний співпроцесор працює з нормалізованими двійковими числами. Двійкове число з плаваючою крапкою називається нормалізованим, якщо ціла частина мантиси рівна 1. З метою розширення розрядної сітки, ця одиниця не зберігається у форматах одинарної і подвійної точності. У форматі з розширеною точністю зберігається і "зайвий" біт цілої частини нормалізованого числа. Основна причина використання формату з розширеною точністю - запобігання можливій втраті точності обчислень при значній різниці в порядках чисел, що беруть участь в арифметичних операціях.

Поле порядку - це степінь числа 2, на який множиться мантиса, плюс зсув, рівний 127 для одинарної точності, 1023 - для подвійної точності і 16383 - для розширеної точності.

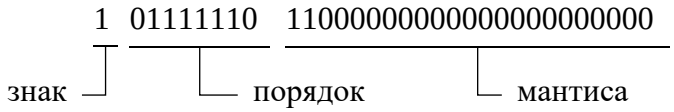
Для того, щоб визначити абсолютне значення числа з плаваючою крапкою, можна скористатися наступними формулами:

- **одинарна точність:** 1.(цифри мантиси)*2^{P-127};
- **подвійна точність**: 1.(цифри мантиси)* 2^{P-1023};
- **розширена точність**: 1.(цифри мантиси)* $2^{P-16383}$.

Знак числа визначається старшим бітом.

Наприклад:

Маємо таке число представлено з одинарною точністю:



Для цього числа знаковий біт рівний 1 (від'ємне число), порядок рівний 126, мантиса - 11 (у двійковій системі числення).

Значення цього числа рівне:

1.11 * 2(126-127)= -1.75 * 2^-1 = -0,875

Цілі числа. Арифметичний співпроцесор разом з дійсними числами здатний обробляти і цілі числа. Він має команди, що виконують перетворення цілих чисел в дійсні і назад.

Можливий чотири формати цілих чисел:

- **ціле число**;
- **коротке ціле число**;
- **довге ціле число**;
- **упаковане двійково-десяткове число**.

Ціле число займає два байти. Його формат повністю відповідає формату, що використовується центральним процесором. Для представлення від'ємних чисел використовується доповняльний код.

Коротке ціле і довге ціле мають аналогічні формати, але займають, відповідно 4 і 8 байт.

Упаковане двійково-десяткове число займає 10 байт. Це число містить 18 десяткових цифр, розташованих по дві в кожному байті. Знак упакованого BCD числа знаходиться в старшому біті найлівішого байта. Решта біт старшого байта повинна бути рівні 0.

Існують команди співпроцесора, які перетворюють числа у формат упакованих двійково-десяткових чисел з внутрішнього представлення в розширеному дійсному форматі. Якщо програма робить спробу перетворення в упакований формат ненормалізованих чисел, нечисел, нескінченності і т.ін., в результаті виходить невизначеність. Невизначеність в упакованому BCD форматі є числом, в якому два старші байти містять одиниці у всіх розрядах. Вміст решти восьми байтів довільний. При спробі використовувати таке упаковане число в операціях - фіксується помилка.

Регістри співпроцесора

Арифметичний співпроцесор має наступну систему регістрів:

- 8 регістрів загального призначення для роботи з даними (R0–R7, кожний розміром 10 байт), до яких можна звертатися тільки як до елементів стеку (де ST(0) – його вершина);
- регістр стану SR (два байта);
- регістр управління CR (два байта);
- регістр тегів TW (два байта);
- регістр вказівника команди FIP (шість байтів);
- регістр вказівника операнду команди FDIP (шість байтів).

Числові регістри

Числові регістри, як правило, в технічній літературі позначаються ST0 - ST7. Вони організовані за принципом стеку.

Регістр стану в полі ST містить номер числового регістра, що є вершиною стеку. При виконанні команд як операнди можуть виступати числові регістри. У цьому випадку номер вказаного в команді регістра додається до вмісту поля ST регістра стану і таким чином визначається регістр, що використовується. Більшість команд після виконання збільшують поле ST регістра стану, записуючи результати своєї роботи в стек числових регістрів.

Система команд математичного співпроцесора

Можливі три формати команд співпроцесора, аналогічні форматам команд центральних процесорів фірми Intel. Це команди зі звертанням до оперативної пам’яті, команди зі звертанням до одного з числових регістрів і команди без операндів, заданих явним чином. Команди зі звертанням до пам’яті можуть займати від двох до чотирьох байт, залежно від способу адресації. Всі

асемблерні мнемоніки команд співпроцесора починаються з букви F, тому їх легко відрізнити від команд центрального процесора.

Команди співпроцесора можна розділити на декілька груп:

- команди пересилки даних;

Таблиця 3.1.

Основні команди математичного співпроцесора

Команда	*Функція*	*Операнд 1*	*Операнд 2*	*Результат*
Команди пересилки даних				
FLD	Завантажити змінну дійсного типу до стеку співпроцесора	відправник	–	ST(0)
FILD	Завантажити змінну цілого типу до стеку співпроцесора	відправник	–	ST(0)
FST	Копіювати значення зі стеку (регістр ST(0)) у змінну дійсного типу	приймач	–	приймач
FSTP	Перенести значення зі стеку (регістр ST(0)) у змінну дійсного типу	приймач	–	приймач
FIST	Копіювати значення зі стеку (регістр ST(0)) до приймача (змінна цілого типу)	приймач	–	приймач
FISTP	Перенести значення зі стеку (регістр ST(0)) у змінна цілого типу	приймач	–	приймач
FXCH	Обмін значеннями регістрів ST(0) та відправника (інший регістр)	відправник	–	–
FCMOVxx	Група команд умовної пересилки даних	приймач	відправник	ST(0)

Група команд умовної пересилки даних FCMOVxx		
Команда	*Реальна умова*	*Умова для команди FCOM*
FCMOVE	ZF = 1	Якщо рівні
FCMOVNE	ZF = 0	Якщо нерівні
FCMOVB	CF = 1	Якщо менше
FCMOVBE	CF = 1 та ZF = 1	Якщо менше або рівні
FCMOVNB	CF = 0	Якщо не менше
FCMOVNBE	CF = 0 та ZF = 0	Якщо не менше або рівні

Команди управління FPU				
FINIT	Ініціювання роботи математичного співпроцесора FPU	–	–	–
FSTCW	Копіювання вмісту регістру CR до приймача (змінна розміром 2 байта або регістр)	приймач	–	змінна

Команда	*Функція*	*Операнд 1*	*Операнд 2*	*Результат*
FLDCW	Копіювання відправника (змінна розміром 2 байта) до регістру CR	відправник	–	CR
FSTSW	Копіювання вмісту регістру SR до приймача (змінна розміром 2 байта або регістр AX)	приймач	–	змінна або AX

- арифметичні команди;
- команди порівнянь чисел;
- трансцендентні команди;
- команди керування.

Команди пересилки даних призначені для завантаження чисел з оперативної пам’яті в числові регістри, записи даних з числових регістрів в оперативну пам’ять, копіювання даних з одного числового регістра в інший.

Арифметичні команди виконують такі операції, як додавання, віднімання, множення, ділення, знаходження квадратного кореня, знаходження часткового залишку, округлення і т.п.

Команди порівняння порівнюють дійсні і цілі числа, виконують аналіз чисел.

Трансцендентні команди призначені для обчислення різних тригонометричних, логарифмічних, показникових і гіперболічних функцій - sin, cos, tg і т.ін.

Команди керування забезпечують установку режиму роботи арифметичного співпроцесора, його скидання і ініціалізацію, перехід співпроцесора в захищений режим роботи і т.д.

Арифметичні команди математичного співпроцесора

Співпроцесор використовує шість основних форматів арифметичних команд:

Fxxx	перший операнд - вершина стеку, другий - наступний елемент стеку, результат операції записується у вершину стеку
Fxxx пам’ять	перший операнд - пам’ять, другим та приймачем є вершина стеку ST(0). Покажчик стеку ST не змінюється, команда для дійсних чисел з одинарною і подвійною точністю
Fixxx пам’ять	аналогічно попередньому формату команди, але операндами можуть бути 16- або 32-розрядні цілі числа
Fxxx ST, ST(i)	для цього формату регістр ST(i) є джерелом, а ST(0) - верхівка стеку - приймачем. Покажчик стеку не змінюється
Fxxx ST(i), ST	для цього типу регістр ST(0) є джерелом, а ST(i) -

	приймачем. Покажчик стеку не змінюється
FxxxP ST(i), ST	регістр ST(i) - приймач, регістр ST(0) - джерело. Після виконання команди джерело ST(0) витягується із стеку

Символи "xxx" можуть приймати наступні значення:
ADD - додавання;
SUB - віднімання;
SUBR - зворотне віднімання, тобто операнди міняються місцями;
MUL - множення;
DIV – ділення;
DIVR - зворотне ділення, ділене і дільник міняються місцями.
Окрім основних арифметичних команд є додаткові арифметичні команди:
FSQRT - знаходження квадратного кореня;
FSCALE - масштабування на ступінь числа 2;
FPREM - обчислення часткової остачі;
FRNDINT - заокруглення до цілого;
FXTRACT - виділення порядку числа і мантиси;
FABS - знаходження абсолютного значення (модуля) числа;
FCHS - зміна знаку числа.

КОНТРОЛЬНІ ПИТАННЯ

1. Для чого використовується математичний співпроцесор?
2. Які формати даних може обробляти математичний співпроцесор?
3. Які числові регістри має співпроцесор?
4. Які регістри керування має співпроцесор?
5. Які групи команд має співпроцесор?
6. Які команди пересилки даних співпроцесора ви знаєте?
7. Які команди управління співпроцесора ви знаєте?
8. Які арифметичні команди FPU ви знаєте?

ЛІТЕРАТУРА

1.Березко Л.О., Троценко В.В. Особливості програмування в турбо-асемблері. - Київ, НМК ВО, 1992.
2.Абель П.Язык ассемблера для IBM PC и программирования. Пер. з англ.- М.,"Высшая школа",1992.
3. Intel® 64 and IA-32 Architectures Software Developer’s Manual Volume 2 (2A & 2B): Instruction Set Reference, A-Z. – 2011. Режим доступу:

http://www.intel.com/content/www/us/en/architecture-and-technology/64-ia-32-architectures-software-developer-vol-2a-2b-instruction-set-a-z-manual.html

ЗАВДАННЯ

1. Створити *.exe програму, яка реалізовує обчислення, заданого варіантом виразу. Вхідні дані повинні вводитися з клавіатури, під час виконання програми, як дійсні числа зі знаком. Програма повинна складатися з двох модулів:
   головний модуль – створюється мовою С і має забезпечити ввід необхідних даних, виклик асемблерної процедури для обчислення виразу та вивід результату обчислень;
   модуль безпосередніх обчислень – здійснює всі необхідні арифметичні дії з використанням математичного співпроцесора.
2. Переконатися у правильності роботи кожного модуля зокрема та програми загалом.
3. Скласти звіт про виконану роботу з приведенням тексту програми та коментарів до неї.
4. Дати відповідь на контрольні запитання.

Таблиця 3.1.

Варіанти завдань

№	Вираз	K
1.	$X=A_4+B_2*C_1-D_2/E_1+K$	1254021
2.	$X=A_4/B_2+C_3-D_1*E_1-K$	202
3.	$X=K-B_4+C_2/D_1-E_1*F_2$	37788663
4.	$X=A_1*(B_2+C_1)-D_1/E_2+K$	45694
5.	$X=A_2*B_2-A_2*C_1-D_1/E_2+K$	505
6.	$X=K+B_2/C_1-D_2*F_2-E_1$	6DD02316
7.	$X=A_4/B_2-C_1*(D_1+E_2-K)$	717
8.	$X=A_2-B_1+K-D_2/E_1+F_1*B_2$	88
9.	$X=A_1*B_2-A_1*C_2+D_2/(E_1+K)$	29
10.	$X=A_2-B_1/C_2+K+E_2*F_1$	2310
11.	$X=(A_2-B_1-K)*D_1+E_4/F_2$	311
12.	$X=K+B_1/C_2-D_2*F_2-E_1$	7055E0AC
13.	$X=A_2/B_1+C_1*(D_1+E_2-K)$	2513
14.	$X=A_2-B_1-K-D_2/E_1+F_1*B_1$	614
15.	$X=A_2+(B_1*C_2)-D_4/E_2+K$	4569600F
16.	$X=A_1/B_2+C_3-D_1*E_1+K$	616

17.	$X=A_1-K+C_1/D_2-E_1*F_1$	1017
18.	$X=A_1*(B_2-C_1)+D_2/E_1+K$	56987018
19.	$X=A_2*B_2+A_2*C_1-D_2/E_1+K$	4019
20.	$X=K+B_1/C_2-D_1*F_1-E_2$	18932020
21.	$X=A_1/B_2+C_1*(D_2 – E_1+K)$	21
22.	$X=K-B_2-C_1-D_2/E_1+F_2*B_1$	45781022
23.	$X=A_2*B_2-C_1+D_1/E_2+K$	7AA02023
24.	$X=K-B_2/C_1+D_1+E_2*F_2$	74569024
25.	$X=(K-B_2-C_1)*D_1+E_4/F_2$	2B05025
26.	$X=A_2+K+C_2/D_1-E_1*F_1$	6C26
27.	$X=A_2*B_1+C_1/(K-E_1*F_1)$	A77627
28.	$X=K+B_1/C_2+D_2-E_2/F_1$	3FF28
29.	$X=K-B_1*C_1+D_2-F_2/E_1$	12A0C029
30.	$X=K+B_2-D_2/C_1+E_1*F_2$	25630

A, B, C, D, E, F - дійсні числа, де індекс визнає точність внутрішнього подання (1 – одинарної, 2, 3, 4 – подвійної точності). Константу К подано у 16-му форматі.

ПРИКЛАД ВИКОНАННЯ

Завдання: Написати програму, яка обчислює арифметичний вираз над дійсними числами, введеними в десятковій системі і результат виводить на екран в десятковій формі зі знаком.

Заданий вираз:

X=A+B

Оскільки дані, що вводяться повинні бути знаковими і зберігати заданий розмір, проаналізуємо які значення можуть бути введені, а отже і отримані в результаті обчислень.

Текст програми, яка повністю реалізовує завдання, приведений нижче.Програма передбачає ввід/вивід знакових даних з використанням функції С.

Далі, застосовуються можливості арифметичного співпроцесора для обчислень всіх арифметичних дій. Для цього в сегменті даних передбачаємо змінні, що будуть використовуватися для обчислень. Вони повинні бути 4 або 8 байтними (згідно форматів даних для співпроцесора).

Програма з взаємодією C-ASM, що виконує додавання двох дійсних чисел з використанням математичного співпроцесора

main.cpp **calc.asm**

```
#include <stdio.h>

extern "C" float
calc(float,float);

int main()
{
  float a=0;
  float b=0;
  float res=0;
  printf("Enter numbers:\n");
  printf("A = ");
  scanf("%f",&a);
  printf("B = ");
  scanf("%f",&b);
  res=calc(a,b);
  printf("Result is: %f\n",res);
  return 0;
}
```

```
.386
.model flat, c
.data
.code
calc PROC
  push ebp
  mov ebp,esp
  fld dword ptr [ebp+8]
  fadd dword ptr [ebp+12]
  pop ebp
  ret
calc ENDP
END
```