|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Министерство образования Республики Беларусь  Учреждение образования  Белорусский Государственный Университет Информатики и Радиоэлектроники | | | | |
| Факультет | компьютерных систем и сетей | | | |
| Кафедра | электронных вычислительных средств | | | |
|  | | |  | |
| **лАБОРАТОРНАЯ РАБОТА №1**    по курсу "Проектирование проблемно-ориентированных вычислительных средств"  Вариант № 3 | | | | |
| Студенты гр. 550701  Проверил: | |  | | Шимко М.Д.  Кудрявцев П.Д.  Богданович В.В.  Шемаров А. И. |
| Минск - 2017 | | | | |

**Цель:**

Знакомство с системой программирования и отладки программ Code Composer Studio, режимами расширения знака и коррекции переполнения в арифметическо-логическом устройстве и их влиянием на результат выполнения различных команд процессора TMS320VC5402. Разработка алгоритма и программы для проверки влияния битов состояния процессора на результат выполнения некоторых команд на ассемблере процессора TMS320VC5402 и отладка программы на лабораторном макете TMS320VC5402 DSP Starter Kit.

**Исходные данные:**

1. Сложить 32-разрядное X и 16-разрядное Y 16-ричные числа из памяти данных. Сохранить 32-разрядный результат Z в памяти данных.. Сравнить результат при разных значениях режима SXM (ssbx SXM или rsbx SXM).

а) X=4567C901, Y=7821, SXM=0/1; Z=????????, C=?, OV=?;

b) X=45677901, Y=9876, SXM=0/1; Z=????????, C=?, OV=?.

2. Сложить 32-разрядное X и 16-разрядное Y 16-ричные числа из памяти данных. Сохранить 32-разрядный результат Z в памяти данных. Сравнить результат при разных значениях режимов SXM и OVM (ssbx OVM или rsbx OVM).

а) X=80003456, Y=B762, SXM=0/1, OVM=0/1; Z=????????, C=?, OV=?.

3. Выполнить операции ABS и NEG над 32-разрядным 16-ричным числом X из памяти данных. Сохранить 32-разрядный результат Z в памяти данных. Сравнить результат при разных значениях режима OVM.

а) X=CE3A9725, OVM=0/1; Z=????????, C=?, OV=?;

b) X=80000000, OVM=0/1; Z=????????, C=?, OV=?.

4. Запишите в отчет программу и примеры с результатами с указанием режимов SXM и OVM, и признаков результата C и OVA или OVB. Для загрузки 32-разрядного числа в аккумулятор можно использовать пару команд LDU и ADD или LD и ADDS или LD и OR или LD и XOR. Сохранение младшей и старшей частей аккумулятора выполняется командами STL и STH. Для задания операндов в памяти процессора используется директива .WORD, а для указания номера текущей страницы – команда LD #k9, DP.

**Листинг программы к заданию 1 и 2 с комментариями:**

.def \_c\_int00 ; определение начала программы

.text ; начало области программы

\_c\_int00: ; метка начала программы

LD #X, DP ; текущая страница данных

RSBX OVM ; нет коррекции переполнения

RSBX SXM ; нет расширения знака

RSBX OVA ; сброс бита переполнения

xor A,A ; очистка аккумулятора

dld X,A ; загрузка двойного слова Х

nop ; задержка

add Y,A ; прибавление слова Y к X

dst A,Z ; сохранение двойного слова в Z

RSBX OVM ; нет коррекции переполнения

SSBX SXM ; режим расширения знака

RSBX OVA ; сброс бита переполнения

xor A,A ; очистка аккумулятора

dld X,A ; загрузка двойного слова Х

nop ; задержка

add Y,A ; прибавление слова Y к X

dst A,Z ; сохранение двойного слова в Z

; в задании 2

SSBX OVM ; коррекция переполнения к заданию 2

RSBX SXM ; нет расширения знака

RSBX OVA ; сброс бита переполнения

xor A,A ; очистка аккумулятора

dld X,A ; загрузка двойного слова Х

nop ; задержка

add Y,A ; прибавление слова Y к X

dst A,Z ; сохранение двойного слова в Z

SSBX OVM ; коррекция переполнения к заданию 2

SSBX SXM ; режим расширения знака

RSBX OVA ; сброс бита переполнения

xor A,A ; очистка аккумулятора

dld X,A ; загрузка двойного слова Х

nop ; задержка

add Y,A ; прибавление слова Y к X

dst A,Z ; сохранение двойного слова в Z

nop ; завершение операции в конвейере

nop ; завершение операции в конвейере

nop ; точка остановки

.data ; начало области данных

X: .long 0x4567C901 ; 1a

Y: .word 0x7821 ; 1a

X: .long 0x45677901 ; 1b

Y: .word 0x9876 ; 1b

X: .long 0x80003456 ; 2

Y: .word 0xB762 ; 2

Z: .long 0

.end

**Листинг программы к заданию 3**

.def \_c\_int00 ; определение начала программы

.text ; начало области программы

\_c\_int00: ; метка начала программы

LD #X, DP ; текущая страница данных

RSBX SXM ; нет расширения знака

RSBX OVM ; нет коррекции переполнения

RSBX OVA ; сброс бита переполнения

xor A,A ; очистка аккумулятора

dld X,A ; загрузка двойного слова Х

nop ; задержка

ABS A ; получение модуля

dst A,Z ; сохранение двойного слова в Z

SSBX OVM ; коррекция переполнения

RSBX OVA ; сброс бита переполнения

xor A,A ; очистка аккумулятора

dld X,A ; загрузка двойного слова Х

nop ; задержка

ABS A ; получение модуля

dst A,Z ; сохранение двойного слова в Z

RSBX OVM ; нет коррекции переполнения

RSBX OVA ; сброс бита переполнения

xor A,A ; очистка аккумулятора

dld X,A ; загрузка двойного слова Х

nop ; задержка

NEG A ; изменение знака

dst A,Z ; сохранение двойного слова в Z

SSBX OVM ; коррекция переполнения

RSBX OVA ; сброс бита переполнения

xor A,A ; очистка аккумулятора

dld X,A ; загрузка двойного слова Х

nop ; задержка

NEG A ; изменение знака

dst A,Z ; сохранение двойного слова в Z

SSBX SXM ; режим расширения знака

RSBX OVM ; нет коррекции переполнения

RSBX OVA ; сброс бита переполнения

xor A,A ; очистка аккумулятора

dld X,A ; загрузка двойного слова Х

nop ; задержка

ABS A ; получение модуля

dst A,Z ; сохранение двойного слова в Z

SSBX OVM ; коррекция переполнения

RSBX OVA ; сброс бита переполнения

xor A,A ; очистка аккумулятора

dld X,A ; загрузка двойного слова Х

nop ; задержка

ABS A ; получение модуля

dst A,Z ; сохранение двойного слова в Z

RSBX OVM ; нет коррекции переполнения

RSBX OVA ; сброс бита переполнения

xor A,A ; очистка аккумулятора

dld X,A ; загрузка двойного слова Х

nop ; задержка

NEG A ; изменение знака

dst A,Z ; сохранение двойного слова в Z

SSBX OVM ; коррекция переполнения

RSBX OVA ; сброс бита переполнения

xor A,A ; очистка аккумулятора

dld X,A ; загрузка двойного слова Х

nop ; задержка

NEG A ; изменение знака

dst A,Z ; сохранение двойного слова

nop ; завершение операции в конвейере

nop ; завершение операции в конвейере

nop ; точка остановки

.data ; начало области данных

;X: .long 0xCE3A9725 ; двойное слово Х в 3а)

X: .long 0x80000000 ; двойное слово Х в 3b)

Z: .long 0 ; двойное слово Z

.end ; конец программы

**Результаты работы программы:**

1.

a)

RSBX: Z = 0x0045684122; C = 0; OVA = 0

SSBX: Z = 0x0045684122; C = 0; OVA = 0

b)

RSBX: Z = 0x0045681177; C = 0; OVA = 0

SSBX: Z = 0x0045671177; C = 1; OVA = 0

2.

OVM = 0, SXM = 0: Z = 0x008000EBB8; C = 0; OVA = 1

OVM = 0, SXM = 1: Z = 0xFF7FFFEBB8; C = 1; OVA = 1

OVM = 1, SXM = 0: Z = 0x007FFFFFFF; C = 0; OVA = 1

OVM = 1, SXM = 1: Z = 0xFF80000000; C = 1; OVA = 1

3.

a)

ABS: SXM = 0, OVM = 0: Z = 0x00CE3A9725; C = 0; OVA = 1

ABS: SXM = 0, OVM = 1: Z = 0x007FFFFFFF; C = 0; OVA = 1

NEG: SXM = 0, OVM = 0: Z = 0xFF31C568DB; C = 0; OVA = 1

NEG: SXM = 0, OVM = 1: Z = 0xFF80000000; C = 0; OVA = 1

ABS: SXM = 1, OVM = 0: Z = 0x0031C568DB; C = 0; OVA = 0

ABS: SXM = 1, OVM = 1: Z = 0x0031C568DB; C = 0; OVA = 0

NEG: SXM = 1, OVM = 0: Z = 0x0031C568DB; C = 0; OVA = 0

NEG: SXM = 1, OVM = 1: Z = 0x0031C568DB; C = 0; OVA = 0

b)

ABS: SXM = 0, OVM = 0: Z = 0x0080000000; C = 0; OVA = 1

ABS: SXM = 0, OVM = 1: Z = 0x007FFFFFFF; C = 0; OVA = 1

NEG: SXM = 0, OVM = 0: Z = 0xFF80000000; C = 0; OVA = 0

NEG: SXM = 0, OVM = 1: Z = 0xFF80000000; C = 0; OVA = 0

ABS: SXM = 1, OVM = 0: Z = 0x0080000000; C = 0; OVA = 1

ABS: SXM = 1, OVM = 1: Z = 0x007FFFFFFF; C = 0; OVA = 1

NEG: SXM = 1, OVM = 0: Z = 0x0080000000; C = 0; OVA = 1

NEG: SXM = 1, OVM = 1: Z = 0x007FFFFFFF; C = 0; OVA = 1

**Вывод:** Мы познакомились с системой программирования и отладки программ Code Composer Studio, режимами расширения знака и коррекции переполнения в арифметическо-логическом устройстве и их влиянием на результат выполнения различных команд процессора TMS320VC5402. Разработали программу для проверки влияния битов состояния процессора на результат выполнения некоторых команд на ассемблере процессора TMS320VC5402.