Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

Факультет ИУ «Информатика и системы управления»

Кафедра ИУ-3 «Информационные системы и телекоммуникации»

Методические указания
по выполнению домашнего задания 1
«Команды микропроцессора»
по дисциплине «Микропроцессорные устройства обработки сигналов»

Для студентов, обучающихся по направлению 09.04.02

Составитель проф. каф. ИУ3, д.т.н. В.С. Выхованец

Москва, 2020

Содержание

1 Общие указания

Целью выполнения домашнего задания является изучение архитектуры и организации микропроцессора и выбор методов и средств реализации низкоуровневой функции, заданной в теме домашнего задания.

Для выполнения домашнего задания необходимо предварительно ознакомиться с учебным пособием [1, с. 6-68] и технической документацией [3], найти и изучить команды микропроцессора, позволяющие эффективно реализовать заданную функцию.

Следующим этапом выполнения домашнего задания является описание найденных команд и их настройка для реализации разрабатываемой низкоуровневой функции.

На последнем этапе выполнения домашнего задания описывается методика разработки низкоуровневой функции.

Итогом выполнения домашнего задания является оформление отчета. Срок выполнения домашнего задания – две недели.

2 Требования к отчету

Оформление отчета по домашнему заданию осуществляется в соответствии с ГОСТ 2.105-95 [4], список литературы по ГОСТ Р 7.0.5–2008 [5]. Отчет должен содержать:

- титульный лист;
- содержание;
- постановку задачи;
- описание алгоритма низкоуровневой функции;
- прототип низкоуровневой функции и описание аргументов;
- передачу аргументов и возврат результата;
- описание и форматы команд микропроцессора;
- методику разработки функции;
- список литературы.

3 Пример выполнения домашнего задания

3.1 Постановка задачи

Темой домашнего задания является разработка алгоритма низкоуровневой функции, выполняющей умножение комплексных чисел в формате Q1.15 для микропроцессора TMS320C5515.

Для выполнения домашнего задания необходимо:

– ознакомиться с рекомендованной литературой;

– разработать и описать алгоритм решения задачи;

– найти и изучить команды микропроцессора, необходимые для решения задачи;

– подготовить тестовые данные для проверки реализации алгоритма;

– оформить отчет по домашнему заданию.

Основным результатом домашнего задания является методика разработки низкоуровневой функции для умножения комплексных чисел.

3.2 Описание алгоритма низкоуровневой функции

Произведением комплексных чисел $x = a + jb$ и $y = c + jd$, записанных в алгебраической форме, называется комплексное число

$$z = (ac - bd) + j(ad + bc), \qquad (1)$$

где j – комплексная единица, $j^2 = -1$, а a, b, c, d – действительные числа.

На практике умножение комплексных чисел выполняют по правилу умножения двучленов, с последующей заменой j^2 на -1.

3.3 Прототип низкоуровневой функции

Прототип низкоуровневой функции умножения комплексных чисел на языке Си имеет следующий вид:

```
long _mpyc(long src1, long src2),                    (2)
```

где src1 и src2 – множимые, а возвращаемое значение функции – результат умножения src1 на src2.

Комплексные числа в (2) имеют длину 32 бита (тип long) и состоят из двух 16-разрядных дробных чисел в формате Q1.15: младшие 16 разрядов – действительная часть *a* числа *x*, старшие 16 разрядов – мнимая часть *b* числа *x* (рисунок 1).

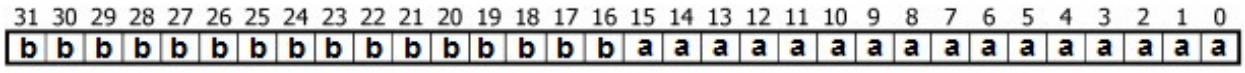

Рисунок 1 – Кодирование комплексных чисел

Формат представления дробных чисел Q1.15 показан на рис. 2, где указаны веса разрядов числа и их номера [1].

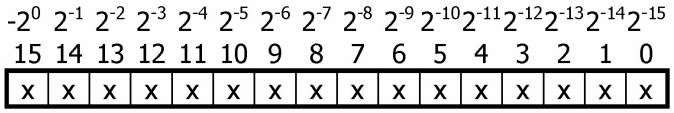

Рисунок 2 – Формат представления дробных чисел Q1.15

3.4 Передача аргументов и возврат результата

При вызове функции на языке Си передача аргументов осуществляется слева направо через регистры и через стек в следующем порядке [2, с. 125]:

– указатели разрядностью (24) 16 бит размещаются в регистрах (X)AR0, (X)AR1, (X)AR2, (X)AR3 и (X)AR4;

– данные разрядностью 16 бит размещаются в регистрах T0, T1, AR0, AR1, AR2, AR3, AR4;

– данные 32 или 40 бит размещаются в регистрах AC0, AC1, AC2;

– аргумент перед многоточием размещается в стеке;

– структуры длиной более 32 бит передаются через указатель.

Если регистров для размещения аргументов недостаточно, то не поместившиеся аргументы заносятся в стек. Аргументы в стеке размещаются в порядке, обратном вызову и выравниваются по двойному слову (по границе 4 байт): в ячейке *SP(0) размещается адрес возврата из функции, в *SP(1) – 1-й аргумент функции, не поместившийся в регистрах, и т.д., где SP – указатель стека операндов, а в круглых скобках указано смещение относительно этого указателя в двойных словах.

Возврат результата выполнения функции осуществляется через регистры:

– данные разрядностью 16 бит размещаются в регистре T0;

– данные разрядностью 32 или 40 бита размещаются в регистре AC0;

– указатели разрядностью (24) 16 бит размещаются в регистре (X)AR0;

– структура длиной более 32 би возвращается через первый аргумент функции – указатель на структуру, если он 0x0, то структура не возвращается.

Вызываемая функция при использовании должна сохранять и восстанавливать регистры T2, T3, (X)AR5, (X)AR6 и (X)AR7, а регистры T0, T1, (X)AR0, (X)AR1, (X)AR2, (X)AR3, (X)AR4, AC0, AC1, AC2, AC3 может использовать без сохранения.

Все другие специальные регистры микропроцессора при использовании подлежат обязательному сохранению и восстановлению, а именно: RETA, BKx, BRCx, BRS1, BSAx, RSAx, REAx, RPTC, CSR, TRNx, (X)DP, (X)CDP, STx_55, где вместо x подставляется номер соответствующего регистра.

Сохранение регистров производится в автоматической (локальной) памяти функции, выделяемой в стеке перед адресом возврата путем уменьшения указателя стека SP на число двойных слов выделяемой памяти. Локальная память освобождается перед возвратом из функции путем увеличения указателя стека SP на число выделенных двойных слов. После выделения локальной памяти доступ к аргументам осуществляется с учетом длины выделенной памяти. Например, после выделения двух двойных слов адрес возврата из функции может быть адресован как *SP(2), первый аргумент в стеке – как *SP(3), и т.д.

Очистку стека с аргументами производит вызывающая функция после вызова вызываемой.

В соответствии с вышеописанным для функции с прототипом (2) аргументы размещаются в порядке следования в регистрах AC0 и AC1, а результат возвращается в регистре AC0. Стек для размещения аргументов не используется.

3.5 Команды микропроцессора

Комплексные числа в разрабатываемом алгоритме кодируются парой дробных чисел, представленных в формате Q1.15 (рисунок 2) и хранящиеся в 32-разрядном целом числе (рисунок 1).

Из формулы (1) следует, что для умножения двух комплексных чисел необходимо выполнить шесть операций: четыре операции умножения и по одной операции сложения и вычитания дробных чисел.

3.5.1 Выбор команд

В наборе команд микропроцессора TMS320C55x [3] имеются следующие команды, использующие двойной умножитель-аккумулятор и позволяющие реализовать умножение комплексных чисел наиболее эффективным образом:

– MPY::MPY – две параллельные операции умножения операндов, выполняемые за один такт;

– MAS::MAC – две параллельные операции умножения с вычитанием и умножения со сложением, выполняемые за один такт.

3.5.2 Синтаксис команд

Команды MPY::MPY и MAS::MAC имеют следующий синтаксис [3, с. 5-471 и с. 5-468]:

MPY[R][40] [uns(]Smem[)], [uns(]HI(Cmem)[)], ACy

:: MPY[R][40] [uns(]Smem[)], [uns(]LO(Cmem)[)], ACx, (3)

MAS[R][40] [uns(]Smem[)], [uns(]HI(Cmem)[)], ACy

:: MAC[R][40] [uns(]Smem[)], [uns(]LO(Cmem)[)], ACx, (4)

где :: – признак встроенного параллелизма, MPY – мнемоника команды умножения, MAS – мнемоника команды умножения с вычитанием, MAC – мнемоника команды умножения со сложением, R – признак округления результата операции, 40 – признак использования 40 разрядов аккумулятора, uns – признак беззнакового кодирования операндов, Smem – метод адресации первого операнда в памяти, Cmem – метод косвенной адресации второго операнда с использованием регистра-указателя коэффициентов CDP, HI – признак использования старшего слова двойного слова, LO – признак использования младшего слова двойного слова, ACy и ACy – регистр аккумулятор AC0-AC3.

Необязательные признаки команд в (3) и (4) заключены в квадратные скобки.

3.5.3 Операции команд

Команда MPY::MPY выполняет две параллельные операции умножения (MPY) в одном цикле:

ACy = Smem * HI(Cmem)

::ACx = Smem * LO(Cmem)),

а команда MAS::MAC – две параллельные операции умножения с вычитанием (MAS) и умножения со сложением (MAC) в одном цикле:

ACy = ACy – (Smem * HI(Cmem))

::ACx = ACx + (Smem * LO(Cmem)).

На выполнение команд влияют флаги FRCT (флаг дробного режима), M40 (флаг 40-разрядного аккумулятора, кодируется в команде), RDM (флаг округления, кодируется в команде), SATD (флаг насыщения арифметико-логического устройства), SMUL (флаг насыщения умножителя), SXMD (флаг расширения знака, кодируется в команде).

3.5.4 Форматы команд

Команды MPY::MPY и MAS::MAC имеют следующие форматы соответственно:

1111 1101 AAAA AAAI 0000 00mm DDDD uug%

1111 1101 AAAA AAAI 0101 11mm DDDD uug%,

где двоичными цифрами показаны поля кодов команд, AAAA AAAI – поле метода адресации первого операнда Smem, mm – поле метода адресации второго операнда Cmem, DD – поле регистра-аккумулятора ACy, DD – поле регистра-аккумулятора ACx, u – поле признака беззнакового формата первого операнда, u – поле признака беззнакового формата второго операнда, g – поле использования 40-разрядного аккумулятора, % – поле округления.

Восьмиразрядное поле AAAA AAAI адресации операнда в памяти Smem кодируется следующим образом:

– AAAA AAA0 – 7 разрядов смещения при прямой адресации операнда относительно регистра XDP или XSP в зависимости от флага CPL в регистре статуса микропроцессора ST1_55,

– AAAA AAA1 – 7 разрядов кода способа косвенной адресации операнда через регистры CDP, AR0-AR7 и T0-T1.

Поле адресации операнда в памяти AAAA AAA1 кодируется следующим образом [3, с. 6-18]:

– 0001 0001 – ABS16(#k16),
– 0011 0001 – *(#k23),
– 0101 0001 – port(#k16),
– 0111 0001 – *CDP,
– 1001 0001 – *CDP+,
– 1011 0001 – *CDP−,
– 1101 0001 – *CDP(#K16),
– 1111 0001 – *+CDP(#K16),
– PPP0 0001 – *ARn,
– PPP0 0011 – *ARn+,
– PPP0 0101 – *ARn−,
– PPP0 0111 – *(ARn + T0),
– PPP0 1001 – *(ARn – T0),
– PPP0 1011 – *ARn(T0),
– PPP0 1101 – *ARn(#K16),
– PPP0 1111 – *+ARn(#K16),
– PPP1 0011 – *(ARn + T1) при ARMS = 0 и *ARn(#1) при ARMS = 1,
– PPP1 0101 – *(ARn – T1) при ARMS = 0 и *ARn(#2) при ARMS = 1,
– PPP1 0111 – *ARn(T1) при ARMS = 0 и *ARn(#3) при ARMS = 1,
– PPP1 1001 – *+ARn при ARMS = 0 и *ARn(#4) при ARMS = 1,
– PPP1 1011 – *−ARn при ARMS = 0 и *ARn(#5) при ARMS = 1,
– PPP1 1101 – *(ARn + T0B) при ARMS = 0 и *ARn(#6) при ARMS = 1,
– PPP1 1111 – *(ARn – T0B) при ARMS = 0 и *ARn(#7) при ARMS = 1,

где ABS16 – абсолютная адресация операнда в памяти путем задания смещения в виде 16-разрядной константы k16, размещаемой непосредственно после кода команды, PPP – поле

номера n дополнительного регистра ARn; k16 (k23, K16) – беззнаковая 16-разрядная (беззнаковая 23-разрядная, знаковая 16-разрядная) константа, хранящаяся в памяти непосредственно после команды; port – ключевое слово для обращения в адресное пространство ввода-вывода; ARMS – флаг режима косвенной адресации (3-й разряд регистра статуса микропроцессора ST2); B – признак бит-реверсивной адресации.

Двухразрядное поле адресации операнда в памяти mm кодируется следующим образом [3, с. 6-24]:

– 00 – *CDP
– 01 – *CDP+
– 10 – *CDP−
– 11 – *(CDP + T0).

Двухразрядное поле регистра-аккумулятора DD кодируется следующим образом:

– 00 – регистр-аккумулятор AC0,
– 01 – регистр-аккумулятор AC1,
– 10 – регистр-аккумулятор AC2,
– 11 – регистр-аккумулятор AC3.

3.6 Методика разработки функции

Исходные данные представлены в формате Q1.15 и упакованы в двойные слова, следовательно, для умножения комплексных чисел необходимо использовать следующие варианты команд (3) и (4):

MPY Smem, HI(Cmem), ACy
:: MPY Smem, LO(Cmem), ACx
MAS Smem, HI(Cmem), ACy
:: MAC Smem, LO(Cmem), ACx,

где операнд в памяти команды MPY::MPY, адресуемый методом Smem, указывает на 16-разрядный операнд *a* из (1), операнд в памяти команды MPY::MPY, адресуемый методом Cmem, указывает на операнд *c* (младшая часть двойного слова) и операнд *d* (старшая часть двойного слова), операнд в памяти команды MAS::MAC, адресуемый методом Smem, указывает на 16-разрядный операнд *b*, а операнд в памяти команды MAS:: MAC, адресуемый методом Cmem, указывает на все тот же операнд *c* (младшая часть двойного слова) и операнд *d* (старшая часть двойного слова).

Так как комплексные числа поступают в функцию в регистрах-аккумуляторах (см. 3.4), то необходимо выполнить их размещение в локальной памяти функции. Для выделения и

освобождения локальной памяти используются следующие команды адресной арифметики [3, с. 5-6]:

```
AADD    #-4, SP                 ; Выделение в стеке четырех слов
AADD    #4, SP                  ; Восстановление стека
```

Запись в локальную память регистров AC0 и AC1 с аргументами функции осуществляется командами [3, с. 5-410]:

```
MOV     AC1, dbl(*SP(0))        ; Сохранение второго операнда
MOV     AC0, dbl(*SP(2))        ; Сохранение первого операнда
```

Для выполнения команд необходимо установить флаг дробного режима FRCT, предварительно сохранив в стеке старое значение регистра статуса ST1_55, что осуществляется следующей командой [3, с. 5-506, 5-118]:

```
PSH     mmap(ST1_55)
BSET    FRCT
```

Так как команды умножения используют регистр-указатель CDP, то его надо сохранить в стеке и загрузить адресом второго аргумента – адресом src2 [3, с. 5-513, 5-424 и 5-340]:

```
PSHBOTH XCDP
MOV     SP, mmap(@CDP)
```

В итоге с учетом использования в стеке трех дополнительных ячеек памяти для сохранения регистра статуса ST1_55 и регистра-указателя XCDP, команды умножения комплексных чисел будут иметь следующий вид:

```
MPY     *SP(5), LO(*CDP), AC0   ; AC0 = Re(src1)*Re(src2)
::MPY   *SP(3), LO(*CDP), AC1   ; AC1 = Im(src1)*Re(src2)
MAS     *SP(3), HI(*CDP), AC0   ; AC0 -= Im(src1)*Im(src2)
::MAC   *SP(5), HI(*CDP), AC1   ; AC1 += Re(src1)*Im(src2)
```

Для формирования результата функции необходимо выполнить преобразование чисел из формата Q1.31, размещенных в регистре AC0 – действительная часть результата, и регистре в AC1 – мнимая часть результата, в формат Q1.15 и записать их в двойное слово регистра-аккумулятора AC0 как показано на рисунке 1. Последнее можно сделать командами [3, с. 5-417 и с. 5-349]:

```
MOV pair(LO(AC0)), dbl(*CDP)
MOV dbl(*CDP), AC0
```

где первая команда записывает двойное слово, состоящее из младших слов пары регистров-аккумуляторов AC0 и AC1 в локальную память по адресу второго аргумента, а вторая команда извлекает двойное слово по этому адресу и записывает его в регистр-аккумулятор AC0.

Последней частью функции является восстановление регистра статуса ST1_55, регистра-указателя XCDP, освобождение локальной памяти и возврат из функции:

```
POP     mmap(ST1_55)
POPBOTH XCDP
AADD    #4, SP
```

```
RET
```

Список литературы

[1] Выхованец, В.С. Микропроцессорные устройства обработки сигналов [Текст] / В.С. Выхованец, Н.А. Демин, Е.И. Мозговая, С.И. Назарова, Д.А. Рожкова, Е.С. Шапкина; Под ред. В.С. Выхованца. – М.: МГТУ им. Н.Э. Баумана, 2014. – 177 с.

[2] TMS320C55x. Optimizing C-C++ Compiler. User's Guide. – Texas Instruments, 2011. – 181 p.

[3] TMS320C55x Mnemonic Instruction Set: Reference Guide [Текст]. – Texas Instruments, 2009. – 863 p.

[4] ГОСТ 2.105-95. Единая система конструкторской документации. Общие требования к текстовым документам [Текст]. – М.: Изд-во стандартов, 2012. – 26 с.

[5] ГОСТ Р 7.0.5–2008. Библиографическая ссылка. Общие требования и правила составления [Текст]. – М.: Изд-во стандартов, 2009. – 23 с.