

## ПРИНЦИПЫ ОРГАНИЗАЦИИ ПРОГРАММ

**1. ЦЕЛЬ РАБОТЫ:** освоение принципов построения приложений на языке ассемблера для системы Texas Instruments, ознакомление с командами и правилами построения программ, ознакомление с методикой проектирования программ в среде программирования.

## 2. ОСНОВНЫЕ ТЕОРЕТИЧЕСКИЕ ПОЛОЖЕНИЯ

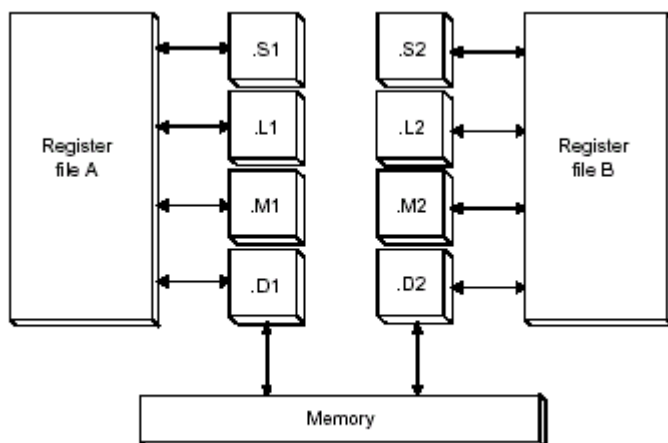
Регистровые файлы - наборы регистров, предназначенных для выполнения различных функций. Количество наборов и количество регистров в наборе в различных процессорах меняется в достаточно широком диапазоне и имеет тенденцию увеличиваться в современных процессорах.

В любых процессорах существуют команды различных типов: “регистр, регистр -> регистр”, “память, память -> память”, “память, память -> регистр”, “память -> регистр” и другие. Тип команды “регистр, регистр -> регистр” означает, что источником двух операндов при ее выполнении являются регистры, и результат операции помещается также в регистр.

Применение регистровых файлов в качестве источников/получателей позволяет широко использовать команды типа “регистр, регистр -> регистр” и строить быстродействующие процессоры.

ЦПУ имеет восемь операционных модулей L, S, M, D разбитые на две идентичные группы 1 и 2. Источниками операндов и получателями результатов являются два набора 32-разрядных регистров A и B соответственно для операционных модулей группы 1 и 2.

Схема операционных модулей.



L1, S1, M1, D1 – регистры общего назначения A0-A15

L2, S2, M2, D2 – регистры общего назначения B0-B15

Модули L (L1, L2)

32/40 – разрядные арифметические операции и операции сравнения;

32-разрядные логические операции;

операции нормализации

Модули S (S1 , S2)

- 32–разрядные арифметические операции;
- 32/40 – разрядные операции сдвига и операции с отдельными битами;
- 32–разрядные логические операции;

Модули M (M1 , M2)

- Операции умножения 32х32 с фиксированной точкой;
- Операции умножения 32х32 с плавающей точкой;

Модули D (D1 , D2)

- 32–разрядные операции по вычислению адресов, в том числе адресов циклических и линейных буферов.

### 3. ПРИМЕР ВЫПОЛНЕНИЯ ЛАБОРАТОРНОЙ РАБОТЫ:

Информация о том как создать проект, подключить файл с программой и запустить на выполнение можно просмотреть в разделе **Описание среды программирования**.

Пример программы:

Формат данных: 32 бита знаковые (int signed)

;Простая программа:

;вычислить  $A+ABS(BxC-D)$ , результат в регистре A1, где ABS – модуль числа

.ref \_c\_int00 ;точка входа  
\_c\_int00:

.text

MVK .S1 -8,A0	;A = -8
MVK .S1 2,A1	;B = 2
MVK .S1 6,A2	;C = 6
MVK .S1 20,A3	;D = 20
MPY .M1 A1,A2,A1	;умножение BxC, результат в A1;
NOP 2	
SUB .L1 A1, A3,A3	;вычисляем разность BxC-D, результат в A3;
ABS .L1 A3,A1	;вычисляем модуль A1 =  A3  - модуль; A3=BxC-D;
ADD .L1 A0,A1,A1	; складываем значение по модулю A+ABS(BxC-D) ; результат в A1;

### Описание команд:

#### Записать 16 разрядную константу в регистр

MVK (.unit) cst, dst ; где cst – константа; dst – регистр  
unit = .S1 или .S2

Пример:

MVK .S1 0,A4 ;Запишем 0 в регистр A4;

Количество тактов : 1

#### Записать 16 разрядную константу в верхние биты регистра

MVKH (.unit) cst, dst ; где cst – константа; dst – регистр  
unit = .S1 или .S2

Количество тактов : 1

#### Записать 16 разрядную константу в нижние биты регистра

MVKL (.unit) cst, dst ; где cst – константа; dst – регистр  
unit = .S1 или .S2

Количество тактов : 1

### Знаковое вычитание

SUB (.unit) src1, src2, dst  
src1 вычитает src2, результат размещается в dst.

Формат данных 32 бита

.unit = .L1, .L2, .S1, .S2

Количество тактов : 1

### Знаковое сложение

ADD (.unit) src1, src2, dst  
src1 складывается с src2, результат размещается в dst.

Формат данных 32 бита

.unit = .L1, .L2, .S1, .S2

Количество тактов : 1

### Знаковое умножение

MPY (.unit) src1, src2, dst  
src1 умножается на src2, результат в dst

Формат данных 16 бит

.unit = .M1 или .M2

Количество тактов : 2; 1 такт на саму команду, 1 такт задержка выполнения. Команда считается выполненной после того как произошла запись значения в dst.

### Абсолютное значение числа (модуль)

ABS (.unit) src2, dst, где src2 – регистр , dst – регистр где будет размещен результат  
.unit = .L1, .L2  
Модуль src2 размещается в dst;  
Количество тактов : 1;

### Описание программы:

В самом начале с помощью директивы ассемблера **.ref** объявляется метка **\_c\_int00**. Директива **.ref** используется для объявления символов, используемых в данном модуле, но определенных в других модулях. Символ **\_c\_int00** определен глобально и представляет собой точку входа, т.е. выполнение программы начинается с метки **\_c\_int00**. По умолчанию ассемблер транслирует программу в секцию кода, поэтому метка **\_c\_int00** указывает на начало секции кода.

Директива **.text** выбирает в качестве текущей секции секцию кода.

Команда **MVK .S1 -8,A0** записывает значение **-8** в регистр **A0**, через операционный модуль **.S1**, и затем в регистры записываются начальные значения для вычисления по формуле.

Если пользователю необходимо записать значение в старшие или младшие половины регистров, то для этого можно воспользоваться следующими командами: Команда **MVKL** загружает 16 разрядную константу в регистр, заполняя старшую половину регистра значением знакового бита константы. Команда **MVKH** загружает 16 разрядную константу в старшую часть регистра.

После записи начальных данных для вычисления производится умножение регистров **A1** и **A2** командой **MPY .M1 A1,A2,A1**. После команды умножения следует пустая команда **NOP** . Эта команда ничего не делает, заставляя процессор простаивать один такт. По прошествию одного такта, произведение значений регистров запишется в регистр **A1**, и команда умножения будет считаться выполненной.

После команды умножения следует команда вычитания **SUB .L1 A1, A3,A3**, из значения регистра **A1** произойдет вычитание значения регистра **A3**, результат будет располагаться в регистре **A3**.

Затем произойдет определение модуля числа хранящегося в регистре **A3** командой **ABS .L1 A3,A1** результат будет записан в регистр **A1** .

Программа завершается командой **ADD .L1 A0,A1,A1**. Производится сложение значение регистра **A0** со значением регистра **A1**, результат будет записан в регистр **A1**. На этом программа завершается.

#### 4. ЗАДАНИЕ НА ЛАБОРАТОРНУЮ РАБОТУ:

В ходе выполнения лабораторной работы необходимо разработать программу, в соответствии с заданием. Отчет по лабораторной работе должен содержать описание индивидуального задания, граф схемы алгоритмов с их описанием, текст программы с соответствующими комментариями и пример результатов работы.

##### Варианты преобразований:

1.  $(A * B - C) + D$
2.  $A + C - C * B$
3.  $(A + B) * C - D$
4.  $C * 2 - (A - B)$
5.  $D * A + (A + B)$
6.  $(A \text{ or } B) * D + (C - 1)$
7.  $(A - 1) * 2 - (C * 2 + B)$
8.  $D * 4 + ((A - C) * (B - 1))$
9.  $B * (A + B) - (C - D - 1)$
10.  $(C \text{ and } D) \text{ xor } (A \text{ and } B) * D + (A + B)$
11.  $((C * 4) \text{ xor } B) + (A * D)$
12.  $(C - 10 * 2 + (D + B - A))$
13.  $(A * B - C) + D$
14.  $A + C - C * B$
15.  $(A + B) * C - D$
16.  $C * 2 + (A - B)$
17.  $D * A + (A + B)$
18.  $(A \text{ or } B) * D + (C - 1)$
19.  $(A - 1) * 2 \text{ and } (C * 2 + B)$
20.  $D * 4 \text{ or } ((A - C) * (B - 1))$
21.  $B * (A + B) + (C - D - 1)$
22.  $(C \text{ and } D) \text{ xor } (A \text{ and } B) * D + (A + B)$
23.  $((C * 4) \text{ xor } B) + (A * D)$
24.  $(C - 10 * 2) - (D + B - A)$
25.  $D * A \text{ or } (A + B)$
26.  $(D * A - 1) \text{ xor } (A * B)$
27.  $(10 * 2 + D * A) + (A - B)$
28.  $D * A * 3 + (A \text{ xor } B)$
29.  $D * 4 \text{ or } ((A - C) + (B - 10))$
30.  $(C \text{ or } 10 * 3 + (D - B + A))$
31.  $((A - 1) * 2 + B) * C - D$
32.  $((C * 4) + B) - (A * D)$

Код варианта задания определяется тремя компонентами:

- ◆ Номер задания
- ◆ Формат данных (byte(b) – 1 байт; short(s) – 2 байта; int(i) – 4 байта; )
- ◆ Со знаком или без знака (signed(s) / unsigned(u))

Таблица вариантов

№ вар.	1	2	3	4	5	6	7	8
Код вар.	1bu	2ss	3is	4bs	5ss	6su	7iu	8bu

№ вар.	9	10	11	12	13	14	15	16
Код вар.	9ss	10is	11bu	12ss	13iu	14bu	15is	16bs

№ вар.	17	18	19	20	21	22	23	24
Код вар.	17su	18is	19bu	20ss	21iu	22iu	23is	24bu

№ вар.	25	26	27	28	29	30	31	32
Код вар.	25is	26ss	27bu	28bu	29su	30ss	31iu	32su

### Список литературы:

#### 1. Общие теоретические сведения.

Перевод - файл “Процессор TMS 320C6000.doc”  
 Английский вариант - файл “spru189f.pdf”

#### 2. Описание команд

Английский вариант - файл “spru189f.pdf”  
 раздел TMS320C62x/C64x/C67x Fixed-Point Instruction Set  
 Электронная документация - раздел Instruction Set Summary

#### 3. Структура ассемблерного кода

Перевод -раздел Structure of Assembly Code(\*\*\*)

Английский вариант -файл spru198f.pdf  
 раздел Structure of Assembly Code

Электронная документация - раздел  
TMS320C6000 Programmer’s Guide/ Structure of Assembly Code

#### 4. Директивы ассемблера

Перевод -раздел Assembler Directives(\*\*\*)

Английский вариант -файл spru186i.pdf  
 раздел Assembler Directives

Электронная документация - раздел  
Code Generation Tool OnLine Documentaion/ Using the Assembler/ Assembler Directives

## **5. Описание ассемблера**

Перевод -раздел Assembler Description(\*\*\*)

Английский вариант -файл spru186i.pdf  
раздел Assembler Description

Электронная документация - раздел  
Code Generation Tool OnLine Documentaion/ Using the Assembler/ Assembler Description

## **6. Руководство по среде программирования (\*\*\*)**

## **7. Приложение к методическому пособию (\*\*\*)**