Лабораторная работа № 3

ПРИНЦИПЫ ОРГАНИЗАЦИИ ПАРАЛЛЕЛЬНОГО ВЫПОЛНЕНИЯ КОМАНД

1. **ЦЕЛЬ РАБОТЫ:** освоение принципов построения приложений на языке ассемблера для системы Texas Instruments, ознакомление с командами и правилами построения программ в соответствии с особенностями конвейерного и параллельного выполнения команд.

2. ОСНОВНЫЕ ТЕОРЕТИЧЕСКИЕ ПОЛОЖЕНИЯ

Возможность одновременного выполнения группы команд обеспечивается за счет специальной архитектуры процессоров, при которой различные команды выполняются независимо функционирующими устройствами. Это позволило перейти к качественно новому, значительно более высокому уровню производительности.
Слово команды (длина 32 бита) содержит следующие поля:

Поле условия	Поле устройства	Поле операции	Поле операндов	Поле признака группировки

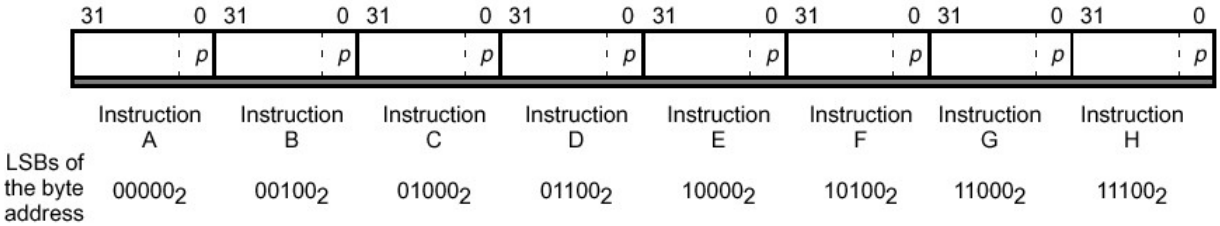
Поле условия хранит код условия выполнения операции, используется для указания условия выполнения команды. Передача управления различным точкам программы реализуется с помощью команд безусловного перехода, которые выполняются или не выполняются в зависимости от условия указываемого пользователем.

Поле операндов содержит указания на операнды. Операнды указываются именами регистров или непосредственно константой.

Поле устройства содержит код устройства, выполняющего операцию.

Поле признака группировки содержит код признака объединения данной команды со следующей для их совместного выполнения в группе.

Процессоры TMS320C6000 имеют в своем составе два одинаковых набора устройств, каждый из которых включает один умножитель (устройства M: M1, M2), два АЛУ (устройства L: L1,L2; S: S1, S2), и одно устройство генерации адреса (D: D1,D2).
Команды выбираются из памяти блоком из 8 команд, каждый блок состоит из 256 бит (восьми слов).

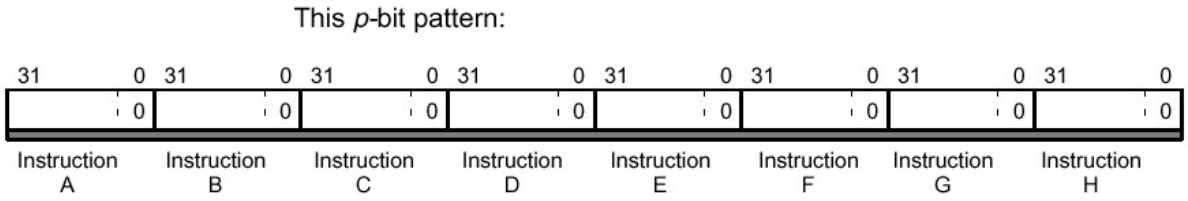


Выполнение каждой команды контролируется специальным битом в каждой команде, p-битом. P – бит устанавливается, когда команда выполняется параллельно с другой командой. P – биты читаются слева направо (от меньшего к большему адресу).
Когда p – бит команды i равен 1, то команда i+1 выполняется параллельно с командой i (в одном цикле). Если p – бит равен 0, то команда i+1 выполняется в следующем цикле после выполнения команды i.

В зависимости от значений бита p в словах различных команд одного пакета выборки возможны следующие ситуации:

- Последовательное выполнение
- Параллельное выполнение
- Последовательное и параллельное выполнение команд одновременно

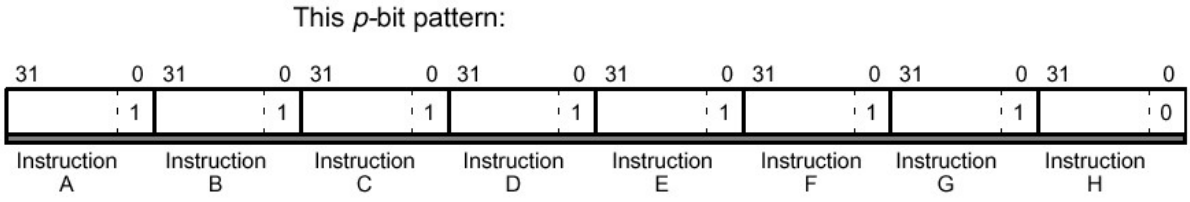
Последовательное выполнение команд:



Cycle/Execute Packet	Instructions
1	A
2	B
3	C
4	D
5	E
6	F
7	G
8	H

Здесь все команды выполняются последовательно, так как значение всех p – битов равны 0.

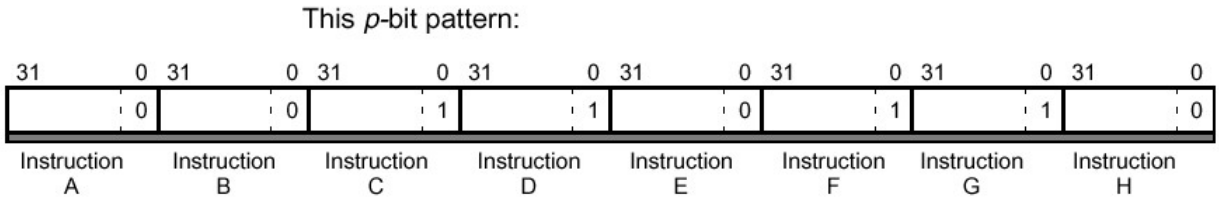
Параллельное выполнение команд:



Cycle/Execute Packet	Instructions							
1	A	B	C	D	E	F	G	H

Здесь все команды выполняются параллельно (в течении одного цикла), так как значения всех p - битов равны 1.

Смешанное выполнение команд:



results in this execution sequence:

Cycle/Execute Packet	Instructions		
1	A		
2	B		
3	C	D	E
4	F	G	H

Здесь команды выполняются как последовательно, так и параллельно.

Слова команд операций с данными и команд управления в процессорах TMS320C6000 имеют одинаковую структуру и содержат поля условия, операции, операндов, устройства и признака группировки. Согласно этой структуре мнемонический синтаксис команд имеет вид:

[||] [Условие] КОП .Устройство [список операндов],
где || - признак объединения команд в группу для одновременного выполнения различными устройствами процессора; **Условие,** если оно есть должно указываться в квадратных скобках.

Пример:
4 одновременно выполняемых команды:

```
        [B0]   ADD  .L2   B0,1,B0      ; инкремент (B0), если (B0) <> 0
||      [B0!]  ADD  .S2   B8,B7,B7     ; (B8)+(B7) – B7, если (B0) =0
||      [A1]   ADD  .L1   A5,A4,A5     ; (A5)+(A4) – A5, если (A1) <> 0
||             B    .S1   LOOP         ; безусловный переход к метке LOOP
```

3. ПРИМЕР ВЫПОЛНЕНИЯ ЛАБОРАТОРНОЙ РАБОТЫ:

Задание берется из лабораторной работы №2.
Задание: Дано два массива необходимо поэлементно их перемножить и сложить полученные произведения.
Пример:
A: {1,2,3}
B: {1,2,3}
1x1+2x2+3x3=1+4+9=14;

Конвейерные вычисления

```
;поэлементное произведение элементов 2 массивов
;регистр A0 содержит конечную сумму произведений
;используются конвейерные вычисления

 .ref _c_int00     ;точка входа
_c_int00:

;//////////////////////////////////////
 .data   ;секция данных

array1: .int 1,2,3;                    ;создаем массив 32 разрядных чисел
array2: .int 1,2,3;                    ;создаем массив 32 разрядных чисел
size    .set 3                         ;размер массива(>1)(препроцессорная константа)

;//////////////////////////////////////
 .text   ;секция кода

      ;Инициализация:
            MVKL .S1 array1,A3        ;загружаем адрес массива1 в A3
            MVKH .S1 array1,A3
            MVKL .S1 array2,A7        ;загружаем адрес массива2 в A7
            MVKH .S1 array2,A7
            MVK .S1 size,A2           ;загружаем колво элементов массива в A2
            MVK .S1 0,A4               ;произведение текущих элементов массива
            MVK .S1 0,A0               ;сумма произведений
            MVK .S2 0,B2              ; тек. элемент выбираемый из массива 1
            MVK .S1 0,A5              ; тек. элемент выбираемый из массива 1
            LDW   .D1  *A3++, B2      ;загружаем текущий элемент в B2
            LDW   .D1  *A7++, A5      ;загружаем текущий элемент в A2
LOOP:

            [A2] B .S1 LOOP           ;переход если A2 <> 0
            SUB .L1 A2,1,A2           ;A2 := A2 - 1
            LDW   .D1  *A3++, B2      ;загружаем текущий элемент в B2
            LDW   .D1  *A7++, A5      ;загружаем текущий элемент в A2
            MPY .M1X A5,B2,A4         ;умножение элементов массивов
            ADD .L1 A0, A4,A0         ;сумма результатов умножения
```

Алгоритм:

В начале работы программы производится инициализация используемых в работе регистров начальными значениями. В основном цикле программы уменьшается счетчик количества элементов массива на 1, затем производится загрузка значения элемента первого и второго массива в соответствующие регистры. Затем производится перемножение элементов массивов, и полученное произведение суммируется с предыдущим. После происходит переход на метку при условии, что счетчик элементов массива не равен 0.

В отличие от программы приведенной в лабораторной работе №2, в этой программе сокращено количество пустых операторов NOP, и соответственно количество тактов работы процессора в этой программе будет меньше.

В программе используется следующий алгоритм: команда условного перехода занимает 6 тактов работы процессора, и поэтому к тому моменту как произойдет переход, процессор выполнит 5 команд.

Перед входом в основной цикл программы производится чтение первых элементов массивов, счетчик элементов массива не изменяется. На первой итерации сумма произведений будет равна произведению первых элементов массива. Значения данных в регистрах после выполнения команд чтения из памяти и умножения будут обновляться на следующих итерациях, поэтому количество итераций должно быть на одну больше. В программе это достигнуто путем считывания и суммирования первого произведения элементов массива без уменьшения счетчика до входа в основной цикл программы. Считывание первых элементов массива производится на этапе инициализации.

На этом закачивается первая часть лабораторной работы №3.

Параллельное выполнение команд

```
;Текст программы
;поэлементное произведение элементов 2 массивов
;регистр A0 содержит конечную сумму произведений
;используется параллельное выполнение команд

;//////////////////////////////////////
 .data   ;секция данных

array1: .int 1,2,3;                    ;создаем массив 32 разрядных чисел
array2: .int 1,2,3;                    ;создаем массив 32 разрядных чисел
size    .set 3                         ;размер массива(>1)(препроцессорная константа)


;//////////////////////////////////////


.text   ;секция кода


      ;Инициализация:
            MVKL .S2 array1,B4          ;загружаем адрес массива1 в A3
            MVKH .S2 array1,B4

            MVKL .S1 array2,A7          ;загружаем адрес массива2 в A7
            MVKH .S1 array2,A7

            MVK .S2 size,B0             ;загружаем колво элементов массива в B0
            MVK .S1 0,A4                ;ПРОИЗВЕДЕНИЕ ТЕКУЩИХ элементов массива

            MVK .S1 0,A0                ;сумма произведений
            MVK .S2 0,B2                ; тек. элемент выбираемый из массива 1
            MVK .S1 0,A5                ; тек. элемент выбираемый из массива 2
            LDW   .D2  *B4++, B2        ;загружаем текущий элемент в B2

            LDW   .D1  *A7++, A5        ;загружаем текущий элемент в A2
            NOP 4
            MPY .M1X A5,B2,A4


LOOP:
            ADD .L1 A0, A4,A0           ; Суммируем произведения
      ||    MPY .M1X A5,B2,A4           ; Вычисляем произведения
      ||    LDW   .D2 *B4++, B2         ; Выбираем из массива A элемент
      ||    LDW .D1  *A7++, A5          ; Выбираем из массива B элемент
      ||    [B0] ADD .S2 -1,B0,B0       ; Уменьшим счетчик на 1, если (B0)<>0
      ||    [B0] B .S1 LOOP             ; Безусловный переход на метку , если (B0)<>0
            NOP 5
```

Алгоритм:

В начале работы программы производится инициализация используемых в работе регистров начальными значениями. В основном цикле программы уменьшается счетчик количества элементов массива на 1, затем производится загрузка значения элемента первого и второго массива в соответствующие регистры. Затем производится перемножение элементов массивов, и полученное произведение суммируется с предыдущим. После происходит переход на метку при условии, что счетчик элементов массива не равен 0.

Во второй части лабораторной работы №3 следует модернизировать программу из первой части под выполнение команд в параллельном режиме.

Алгоритм решение такой же как и в первой части, за исключением того, что 5 команд основного цикла программы будут выполняться одновременно, после будет загружаться команда безусловного перехода, которая занимает 6 тактов работы процессора, поэтому после нее стоит пустой оператор.

Параллельное выполнение команд осуществляется за счет того, что команды выполняются на разных операционных модулях.

Приведенная выше программа описывает синтаксические особенности организации параллельного выполнения команд. Эффективность программы по количеству тактов работы процессора не слишком отличается от программы с использованием конвейерного выполнения команд. Поэтому программу стоит использовать для ознакомления, но не для использования в качестве образца для выполнения лабораторной работы.

Более универсальные и действенные методы организации параллельного выполнения команд можно изучить в Приложении к методическому пособию. На основе программ из Приложения и программы приведенной выше строится вторая часть лабораторной работы.

На этом закачивается вторая часть лабораторной работы №3.

4. ЗАДАНИЕ НА ЛАБОРАТОРНУЮ РАБОТУ:

В ходе выполнения лабораторной работы необходимо разработать программу, в соответствии с заданием, программа должна быть выполнена с применением конвейерных вычислений и параллельного выполнения команд. Отчет по лабораторной работе должен содержать описание индивидуального задания, граф схемы алгоритмов с их описанием, текст программы с соответствующими комментариями и пример результатов работы, а также количество тактов программы. Привести данные о количестве тактов с использованием конвейерного выполнения команд и параллельного выполнения команд.

Варианты заданий:

1. Разработать программу, подсчитывающую количество нулевых элементов массива.
2. Разработать программу вычисления суммы векторов. Сложить элементы двух массивов, результат записать в третий.
3. Разработать программу вычисления разности векторов. Произвести вычитание элементов двух массивов, результат записать в третий.
4. Разработать программу вычисления повторяемости заданного значения в массиве.
5. Подсчитать количество отрицательных элементов в массиве.
6. Разработать программу, переставляющую элементы массива в обратном порядке.
7. Вычисление факториала.
8. Проверить является ли число палиндромом. Палиндром – это число, которое читается слева направо так же как и справа налево (напр. число 121).
9. Подсчитать сумму всех элементов массива, имеющих положительные значения.
10. Заменить все отрицательные значения элементов массива на модули их значений.
11. Найти количество вхождений в массив заданной комбинации двух чисел.
12. Подсчитать сумму элементов массива следующий образом: пусть имеется массив состоящий из 5 элементов, необходимо просуммировать элементы в следующем порядке 1 элемент суммируется с 5, 2 суммируется с 4.
13. Разработать программу вычисляющую сумму элементов массива между двумя заданными. Задаются индексы массива.
14. Разработать программу, сдвигающую элементы массива на две позиции влево. Освободившиеся ячейки левой части массива заполняются элементами правой части.
15. Разработать программу, переставляющую элементы массива в обратном порядке между двумя заданными индексами массива.
16. Проверить является ли массив упорядоченным по возрастанию. Результат проверки записать в регистр.
17. Проверить является ли последовательность элементов между двумя заданными индексами массива упорядоченной по убыванию. Результат проверки записать в регистр.
18. Разработать программу увеличения значения элементов массива в два раза, если элемент имеет нечетный индекс, если элемент имеет четный индекс, то к его значению следует прибавить число 10.
19. Разработать программу подсчета суммы элементов массива у которых нечетные индексы.
20. Проверить являются ли значения элементов массива имеющие нечетные индексы упорядоченными по возрастанию.
21. Проверить являются ли элементы массива имеющие нечетный индекс упорядоченными по возрастанию, а элементы массива имеющие четный индекс упорядоченными по убыванию.

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	1iu	2is	3bu

№ вар.	25	26	27	28	29	30	31	32
Код вар.	4is	5ss	6bu	7bu	8su	9ss	10iu	11su

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. **Общие теоретические сведения о конвейерном выполнении команд**
   Перевод -раздел Using the Assembly Optimizer (***)

   Английский вариант -файл spru187i.pdf
   раздел Using the Assembly Optimizer

   Электронная документация - раздел
   Code Generation Tool Online Documentation/ Using the Assembly Optimizer

7. **Дополнительные сведения о параллельном выполнении команд**
   Перевод -раздел Optimizing Assembly Code Via Linear Assembly (***)

   Английский вариант - файл “spru198f.pdf”
   раздел Optimizing Assembly Code Via Linear Assembly

   Электронная документация - раздел
   TMS320C6000 Programmer’s Guide / Optimizing Assembly Code Via Linear Assembly

8. **Руководство по среде программирования (***)**

9. **Приложение к методическому пособию (***)**