МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение
высшего профессионального образования
«МИРЭА – Российский технологический университет»
РТУ МИРЭА

Институт информационных технологий
Кафедра вычислительной техники

Отчет по лабораторной работе №3

по дисциплине

«Архитектура процессоров и микропроцессоров»

Выполнил: студент группы ИВБО-02-19 К. Ю. Денисов

Принял: старший преподаватель кафедры ВТ Ю. М. Скрябин

Работа выполена «____» _____ 202__
«Зачтено» «____» _____ 202__

Москва 2021

Цель работы

Целью работы является изучение структуры эмулятора, системы команд. Необходимо построить для программы временную диаграмму работы конвейера. Пояснить, что происходит в конвейере в каждом такте, какие возникают конфликты, указать причину конфликта.

Описание работы

Задание 1. Изучить работу команд условных переходов данной программы:

0000 MOV 00, 0003
0001 DECR 00
0002 JP 0001
0003 JMP 0001

Решение. Приведем временную диаграмму (см. таблицу 3).

Задание 2. Разработать программу для вычисления суммы первых десяти членов натурального ряда ($n = 10$), ввести в эмулятор, исследовать ее выполнение, выявить конфликты по данным. Построить временную диаграмму работы конвейера. Пояснить возникающие конфликты, указав № такта.

Решение. Опишем программу, реализующую алгоритм нахождения суммы членов натурального ряда (см. таблицу 1).

Таблица 1 — Программа для нахождения суммы ряда

Команда	Описание
MOV 00, #000A	Запись в РОН 00 числа 10
ADD 01, 00, 01	К РОН 01 прибавляем содержимое регистра 00
DECR 00	Вычитаем 1 из РОН 00
JP 0001	Если содеримое РОН 00 положительное, то повторяем цикл

Описание алгоритма:

1. В РОН 00 записываем длину арифметической последовательности, т.е. 10;
2. К РОН-аккумулятору, в котором будет накапливаться сумма последовательности прибавляем текущее значение РОН 00;

3. Уменьшаем значение РОН текущего индекса в арифметической последовательности на 1;
4. Если результат положительный, повторяем цикл, иначе конец алгоритма.

Построим временную диаграмму данной программы (см. таблицу 4).

Опишем конфликты, возникающие при выполнении данной программы конвейером:

1. На 4 такте мы наблюдаем *структурный конфликт*, так как команда ADD использует тот же РОН что и команда MOV, но команда MOV ещё не закончила своё выполнение, поэтому мы не можем обратиться к одному и тому же РОН и для чтения и для записи;
2. На 9 такте мы наблюдаем *конфликт по данным*, так как команды ADD и DECR используют один и тот же операнд из РОН 00, но команада DECR ещё не закончила своё выполнение;

Опишем варианты избежания конфликтов:

1. Можно избежать конфликта, на такте 4 если поменять местами команды, чтобы чтение происходило в другом такте, но в данном случае это невозможно из-за небольшого количества команд в программе;
2. Можно избежать конфликта на такте 9, если использовать обходную цепь, но так как между командами есть ещё команда JP, обходную цепь использовать не представляется возможности.

Задание 3. Разработать программу для организации инкремента содержимого регистра РОН от 0 до 10.

Решение. Опишем программу, реализующую алгоритм организации инкремента содержимого регистра РОН от 0 до 10. (см. таблицу 2).

Таблица 2 — Программа для инкрементирования значения регистра

Команда	Описание
MOV 00, #000A	Записываем значение 10 в регистр 00
INCR 01	Инкрементируем значение регистра 01
SUB 02, 01, 00	Вычитаем из значения РОН 00 значение РОН 01 и результат записываем в РОН 02
JP 0001	Если результат предыдущей операции положительный, то переходим на первый шаг

Описание алгоритма:

1. В РОН 00 записываем значение до которого происходит инкрементирование переменной, т.е. 10. Сама переменная будет находиться в РОН 01;
2. На каждой итерации цикла увеличиваем значение РОН переменной на один.
3. Вычитаем это значение из 10;
4. Если результат положительный, повторяем цикл, иначе конец алгоритма.

Построим временную диаграмму данной программы (см. таблицу 5).

Опишем конфликты, возникающие при выполнении данной программы конвейером:

На 4 такте мы наблюдаем *структурный конфликт*, так как команда INCR использует тот же РОН что и команда MOV, но команда MOV ещё не закончила своё выполнение, поэтому мы не можем обратиться к одному и тому же РОН и для чтения и для записи.

Опишем варианты избежания конфликта:

Можно избежать конфликта, на такте 4 если поменять местами команды, чтобы чтение происходило в другом такте, но в данном случае это невозможно из-за небольшого количества команд в программе.

Таблица 3 — Задние 1. Временная диаграмма

Ст/Т	1	2	3	4	5	6	7	8	9	10	11	12	13
1	MOV	DECR	JP	JP	JP	DECR	JP	JP	DECR	JP	JP	DECR	JP
2		MOV	DECR	DECR	DECR	JP	DECR	DECR	JP	DECR	DECR	JP	DECR
3			MOV			DECR	JP		DECR	JP		DECR	JP
4				MOV			DECR	JP		DECR	JP		DECR
5					MOV			DECR	JP		DECR	JP	

Ст/Т	14	15	16	17	18	19	20	21	22	23	24	25	26
1		JMP	DECR	JP	JMP	DECR		DECR	JP	DECR	JP	JP	DECR
2			JMP	DECR	JP	JMP			DECR	JP	DECR	DECR	JP
3				JMP	DECR	JP				DECR	JP		DECR
4	JP				JMP	DECR	JP				DECR	JP	
5	DECR	JP				JMP	DECR	JP				DECR	JP

Таблица 4 — Задание 2. Временная диаграмма

Ст/Т	1	2	3	4	5	6	7	8	9	10	11	12
1	MOV	ADD	DECR	DECR	DECR	JP	ADD	DECR	DECR	JP	ADD	DECR
2		MOV	ADD	ADD	ADD	DECR	JP	ADD	ADD	DECR	JP	ADD
3			MOV			ADD	DECR	JP		ADD	DECR	JP
4				MOV			ADD	DECR	JP		ADD	DECR
5					MOV			ADD	DECR	JP		ADD

Таблица 5 — Задание 3. Временная диаграмма

Ст/Т	1	2	3	4	5	6	7	8	9	10	11
1	MOV	INCR	SUB	SUB	SUB	JP	INCR	SUB	JP	INCR	SUB
2		MOV	INCR	INCR	INCR	SUB	JP	INCR	SUB	JP	INCR
3			MOV			INCR	SUB	JP	INCR	SUB	JP
4				MOV			INCR	SUB	JP	INCR	SUB
5					MOV			INCR	SUB	JP	INCR

Вывод: в ходе данной лабораторной работы мы ознакомились со структурой эмулятора RISC конвейера, изучили его систему команд, режимы работы, описали алгоритмы и реализовали программы согласно варианту, построили временные диаграммы работы конвейера, идентифицировали конфликты и указали способы их устранения.