Лабораторная работа № 4 «Конвейерная обработка»

Цель работы: получить навыки ручного написания кода конвейерной обработки.

- 1. Этапы написания конвейерного кода цикла
- 1.1 Построение графа зависимости

В графе зависимости (рис. 1), узлы обозначают инструкции и символические переменные. Дорожки показывают поток данных и комментируются с задержкой их родительских узлов.



Рис. 1 – Граф зависимости

После того, как основной граф зависимости нарисован назначается функциональный модуль на каждый узел или инструкцию. Потом, рисуется линия, чтобы разделить рабочую нагрузку между А- и В-сторонами на равные части.

Пример графа зависимости для операции свертки

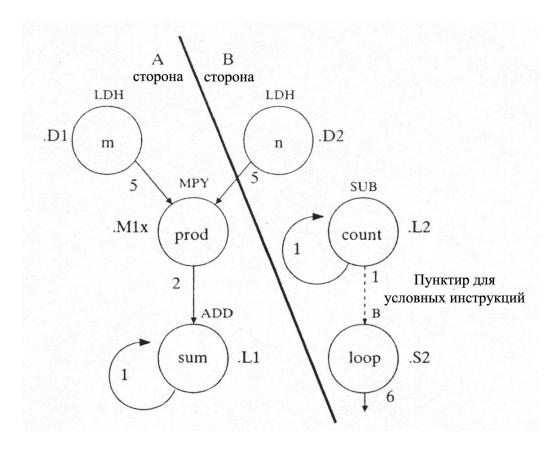


Рисунок 2 – Граф зависимости для операции свертки 1.2 Составление таблицы планирования

Чтобы построить таблицу планирования необходимо найти самую длинную дорожку, посчитав количество задержек. Таким образом, определяется длина таблицы планирования. Для примера самая длинная дорожка равна 8. Это означает, что 7 колонок Пролога требуются перед входом в ядро цикла. Таблица планирования состоит из 15 колонок (7 для Пролога, 1 для ядра цикла, 7 для эпилога) и восьми рядов (один ряд для каждой функциональной единицы). Эпилог и Пролог имеют одинаковую длину.

Таблица 1 – Пример таблицы планирования

	Пролог						Ядр	Эпилог							
								0							
Цикл	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Модул															
Ь															
.D1	<u>1</u> LD	LDH	LD	LD	LD	LDH	LDH	LDH							
	Н		Н	Н	Н										
.D2	<u>4</u> LD	LDH	LD	LD	LD	LDH	LDH	LDH							
	Н		Н	Н	Н										
.L1								3ADD	ADD	ADD	ADD	ADD	ADD	AD	AD
														D	D
.L2		₆ SU B	SUB	SUB	SUB	SUB	SUB	SUB							
.S1															
.S2			<u>3</u> B	В	В	В	В	В							
.M1						2MP Y	MP Y	MPY	MP Y	MP Y	MP Y	MP Y	MP Y		
.M2															

1.3 Написание конвейерного кода

Пользуясь таблицей планирования, набирается код конвейерной обработки. Пример кода конвейерной обработки:

```
cycle 1:
                                            || [B0] sub .L2 B0, 1, B0
      ldh .D1 *A1++, A2
                                            || [B0] B .S2 loop
                                                mpy .M1x A2, B2, A3
     ldh .D2 *B1++, B2
cycle 2:
                                           cycles 8 to n: Single-cycle loop
                                                 ldh .D1 *A1++, A2
      ldh .D1 *A1++, A2
     ldh .D2 *B1++, B2
                                                 ldh .D2 *B1++, B2
 || [B0] sub .L2 B0, 1, B0
                                            || [B0] sub .L2 B0, 1, B0
                                            || [B0] B .S2 loop
                                                 mpy .M1x A2, B2, A3
cycles 3, 4 and 5:
      ldh .D1 *A1++, A2
                                                 add .L1 A4, A3, A4
     ldh .D2 *B1++, B2
 || [B0] sub .L2 B0, 1, B0
                                           cycles n+1 to n+5:
 || [B0] B .S2 loop
                                                 mpy .M1x A2, B2, A3
                                                 add .L1 A4, A3, A4
cycles 6 and 7:
      ldh .D1 *A1++, A2
                                           cycles n+6 to n+7:
     ldh .D2 *B1++, B2
                                                 add .L1 A4, A3, A4
```

2. Задание лабораторной работы

Создайте проект, включите в него два исполняемых файла *.c (основная программа) и *.asm (код конвейерной обработки).

В качестве шаблона для файла основной программы можно использовать шаблон из предыдущей лабораторной работы (лабораторная работа №3).

Файл *.asm должен содержать реализацию функции из табл. 1 согласно варианту с использованием конвейерной обработки.

Для этого необходимо переписать реализацию данной функции на классическом ассемблере, полученную в результате прошлой лабораторной работы (лабораторная работа \mathfrak{N}_{2} 3), пользуясь методикой, описанной в пункте 1. Составьте граф зависимости, таблицу планирования и напишите код конвейерной обработки. Откомпилируйте проект, запустите на исполнение. Убедитесь, что полученное значение у совпадает с тем, что было получено в предыдущей лабораторной работе. Проведите профилирование кода конвейерной обработки.

В материалах к лабораторной работе даны файлы с исходными данными (data1.h, data2.h, data3.h), а также файл dotp3.asm, который содержит пример конвейерной реализации функции свертки двух массивов.

Вариант	Функция	Входные данные
1	$y = \sum_{k=0}^{n} ((x_k - a_k)^2 * 2)$	data1.h
2	$y = \sum_{k=0}^{n} (x_k * a_k - x_k * 2)$	data2.h
3	$y = \sum_{k=0}^{n} ((x_k^2 - a_k) * 2)$	data3.h

3. Содержание отчета

- 1) Цель работы и исходные данные.
- 2) Граф зависимости, таблица планирования, код конвейерной обработки.
- 3) Графики сигнала на входе (массивы а и х) во временной области и их спектры и значение переменной у.
- 4) Результаты профилирования кода (минимальное число циклов, потраченное на выполнение функции (cycle.Total Incl. или cycle.Total: Excl.)).
 - 5) Выводы по работе.