Практическое занятие 4_2022

Тема: Функционирование *скалярного конвейерного* (*суперскалярного*) *процессора* на примере вычисления выражения

Рассмотрим функционирование *скалярного конвейерного процессора* на примере вычисления выражения.

Процессор содержит **несколько конвейерных** АЛУ. Это позволяет одновременно исполнять смежные арифметико-логические операции, что соответствует реализации не только параллелизма служебных операций, но и локального параллелизма. Для разных операций АЛУ имеют различную длину конвейера.

В процессоре используются команды двух классов:

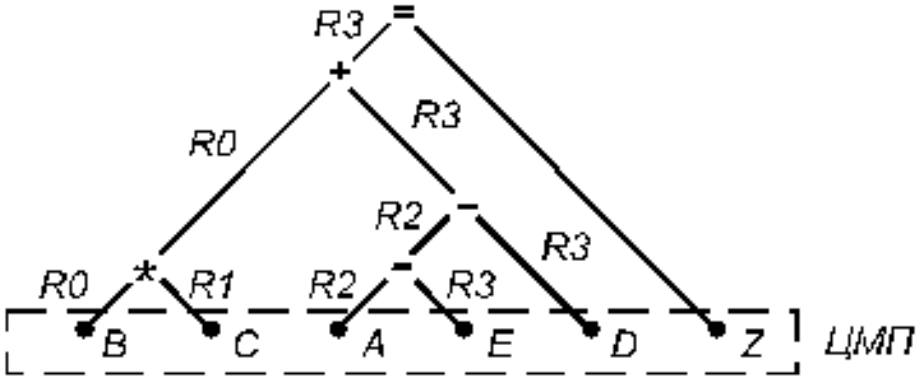
- команды обращения в память и
- регистровые команды для работы с РОН (регистры общего назначения).

Буфер команд имеет многостраничную структуру, что позволяет во время работы УУ с одной страницей производить заранее смену других страниц.

Рассмотрим отрезок программы, соответствующий вычислению выражения:

$$Z = A - (B * C + D) - E$$

Информационный граф процесса выполнения выражения имеет вид:



Фрагмент программы имеет вид:

1 LD R0, B
2 LD R1, C
3 MP R0, R1
4 LD R2, A
5 LD R3, E
6 SUB R2,R3
7 LD R3, D
8 SUB R3, R2
9 ADD R3, R0
10 SW Z, R3

В программе: *LD* – команда загрузки операнда из памяти в регистр; *MP*, *SUB*, *ADD* – команды умножения, вычитания и сложения соответственно; *SW* – команда

записи операнда из регистра в память. Для разных операций АЛУ имеют различную длину конвейера (длина **для аддитивных операций – 6** ступеней, **для умножения – 7** и **14** для операции **деления**).

Любая операция может быть запущена только после того, как подготовлены соответствующие операнды. Это достигается путем запрета доступа в определенные РОН до окончания операции, в которой участвуют данные РОН. Состояния РОН отражены в специальном блоке состояний (БС) РОН.

Состояние некоторых регистров при выполнении программы показано в таблице. В последней колонке таблицы приведен порядок запуска команд на исполнение и сами команды. В частности, видно, что некоторые команды могут опережать по запуску команды, находящиеся в программе выше запущенной. Например, команды 4 и 5 выполняются ранее команды 3. *Это возможно благодаря наличию в программе локального параллелизма и нескольких АЛУ в структуре процессора.* Однако подобные "обгоны" не должны нарушать логики исполнения программы, задаваемой ее информационным графом.

В таблице приведено также описание нескольких тактов работы процессора. Принято, что **выборка операнда** из ЦМП занимает **четыре такта**. Кроме того, считается, что **за один такт** процессора устройство управления з**апускает на исполнение одну команду** или просматривает в программе до четырех команд.

Таблица

Порядок исполнения программы в скалярном конвейере

NN такта	Состояние РОН				Номер команды/ команда
	*R*0	*R*1	*R*2	*R*3	
1	4	–	–	–	***1 LD R0, B***
2	3	4	–	–	***2 LD R1, C***
3	2	3	4	–	***4 LD R2, A***
4	1	2	3	4	***5 LD R3, E***
5	x	1	2	3	**–**
6	7	–	1	2	***3 MP R0, R1***
7	6	–	x	1	**–**
8	5	–	6	–	***6 SUB R2,R3***
9	4	–	5	4	***7 LD R3, D***
10	3	–	4	3	–
. . .					. . .

Символ **x** – блокировка регистра
Символ “**–**“ – регистр свободен

Пояснения к таблице.

Такт 1. Анализ БС РОН показывает, что все РОН свободны, поэтому запускается для исполнения 1-ая команда. В столбец *R*0 записывается 4, что означает: *R*0 будет занят четыре такта. После исполнения каждого такта эта величина уменьшается на единицу.

Такт 2. Запускается команда 2 и блокируется регистр *R*1.

Такт 3. Просматривается команда 3, которая не может быть выполнена, так как нужные для ее исполнения регистры *R*0 и *R*1 заблокированы. Команда 3 пропускается, а ее номер записывается в указатель номера пропущенной команды (УПК). Производится анализ условий запуска следующей (по состоянию СчК) команды. Запускается команда 4.

Такт 4. Просмотр блока команд начинается с номера команды, записанной в УПК. Команда 3 не может быть запущена, поэтому запускается команда 5.

Такт 5. Команда 3 не может быть запущена, так как занят регистр *R*1, однако регистр *R*0 освободился и может использоваться командой 3, но он снова блокируется (символ **x**). Просмотр четырех следующих команд показывает, что они не могут быть запущены, поэтому в такте 5 для исполнения выбирается новая команда.

Такт 6. Запускается команда 3.

В дальнейшем процесс происходит аналогично. За 10 тактов, описанных в таблице, в процессоре запущено 7 команд, что соответствует 10/7 = 1,5 такта на команду.

Задание по вариантам:

Представить процесс вычисления заданного выражения информационным графом, на котором определить порядок занятия 4-х РОН. Записать программу для вычисления выражения на условном ассемблерном языке, представить процесс ее реализации в скалярном конвейерном процессоре с использованием не более 4-х РОН, определить CPI. При выборе порядка загрузки РОН необходимо стремиться к равномерности загрузки регистров и к минимальному в итоге значению CPI.

Варианты заданий:

1. F=(a+b/c)*d+c*g-t*s
2. F=(a*b-c)/d+g-s*t/w
3. F=(n+m-c)*d-a*g+b*t
4. F=(a*b-c)*(n+d)+g*t-w
5. F=(a/b-c)*d-g*q*t-w
6. F=(k*l-m*c)*n+d*t
7. F=(a+b/c)*d+c*g-w*t
8. F=(a*b-c)/d+g*n/w+y
9. F=(n+m-c)*d-a*g /s+y
10. F=(a*b-c)*(n+d)+g*t+w
11. F=(a/b-c)*d-g*q/t+e
12. F=(k*l-m*c)*(n+d)/t*g
13. F=(a+b/c)*d+c*g+e/r
14. F=(a*b-c)/d+g-s*w+g
15. F=(n+m-c)*d-a*g*(e-t)
16. F=(a*b-c)*(n+d)+g/t
17. F=(a/b-c)*d-g*q/s+w
18. F=(k*l-m*c)*(n-d)/w+q
19. F=(a+b/c)*d+c*g/w+q
20. F=(a*b+c)/d+g-s/w+q
21. F=(n+m-c)*d-a*g/b+c
22. F=(a*b-c)*(n+d)+g/b+c
23. F=(a/b-c)*d-g*q/(b+c)
24. F=(k*l-m*c)*(n+d)/e+g

25.F=(k*l-m*c)*(n-d)+t/v
26.F=(a+b-c)*d+c*g-t+s
27.F=(a*b-c)/d+g-s*t/w+z
28.F=(n+m-c*d)/a*g+b*t
29.F=(a*b-c)*(n+d)+g*t-w*z
30.F=(a/b-c)/d-g*q*t-w*z