Федеральное агентство по образованию
Российской Федерации

ГОУ ВПО «Владимирский Государственный Университет»

Кафедра Вычислительной Техники

ВЫЧИСЛИТЕЛЬНЫЕ СРЕДСТВА РАСПРЕДЕЛЕННЫХ АВТОМАТИЗИРОВАННЫХ СИСТЕМ

МЕТОДИЧЕСКИЕ УКАЗАНИЯ К ЛАБОРАТОРНЫМ РАБОТАМ

Составитель БЫКОВ В.И.

Владимир 2008

ЛАБОРАТОРНАЯ РАБОТА № 1

ОЦЕНКА ПРОИЗВОДИТЕЛЬНОСТИ ЭВМ И ЕЕ УСТРОЙСТВ И УЗЛОВ

Цель работы: Изучение методов, методик и способов оценки производительности компьютера и его устройств и узлов.

Теоретические сведения.

Оценка производительности ЭВМ

- *Проблемы:*
- Достоверность оценки
- Адекватность оценки
- Интерпретация оценки

Составляющие производительности компьютера:

1. Производительность процессора
2. Обращение к диску
3. Обращение к памяти
4. Производительность ввода/вывода
5. Дополнительные накладные затраты на работу ОС

Виды времени, могут оцениваться:

1. Время ЦП
2. Полное время выполнения программы
3. Пользовательское время ЦП включая время ожидания
4. Системное время ЦП, затраченное ОС на выполнение задания

Тестирование

Чаще всего производительность оценивается тестированием.

Существует три категории тестов:

1. Производительностей
2. Стандартные
3. Пользовательские, строятся на основе прикладных программ

Требования к тестам:

1. Тест должен получить признание широких кругов компьютерной общественности (это производители, пользователи и эксперты)
2. Методика измерения производительности не должна допускать оптимизации, т.е. поддерживать объективные результаты
3. Соответствие результатов тестирования представлению пользователей о типовых задачах
4. Тест должен обеспечивать простоту запуска на компьютере любого типа
5. Моделенезависимость

Способы толкования оценки:

1. Средняя производительность компьютера
2. Средняя производительность тестового пакета
3. Производительность при выполнении определенной задачи, т.е. натуральная
4. Пиковая производительность
5. Производительность по сравнению с, принятой за эталон производительностью некоторого компьютера

Оценка производительности процессора:

1. Количество и длительность тактов синхронизации
2. Частота синхронизации
3. Среднее количество тактов синхронизации на 1 команду процессора

Способы оценки:

1. MIPS – количество млн. команд в сек.
   "-" а) зависит от набора команд процессора
   б) зависит от набора команд в программе
   в) не всегда предсказуемо
2. MFLOPS – млн. операций с плавающей точкой
3. LINPACK – (ливенворские циклы)
4. SPEC – (standart performance evaluation corp.)
5. TPC – A/B/C (обработка и трансляция транзакций)
   A – кол-во транзакций в сек.
   B – тест БД
   C – обработка заказов

6. Bench mark – тестирование на многих пакетах. Пакеты включают задачи по работе с БД, дисками, математические задачки, обработку графики и звука и т.п.

Методические указания.

1. Изучить методы, методики и способы оценки производительности компьютера и его устройств и узлов, используя литературу и материалы лекций.

2. Разработать методику оценки производительности компьютера или его устройства или узла по заданию преподавателя.

3. Разработать алгоритм и программу для оценки производительности компьютера или его устройства или узла.

4. Выполнить программу с несколькими наборами исходных данных, определяя каждый раз время выполнения. Количество повторений заданных операций должно быть большим и изменяться от 100 или 1000 до 1млрд. с кратностью 10, т.е. в логарифмическом масштабе. Количество экспериментов, т.е. точек на графике – не менее 5. Для измерения времени использовать процедуры или функции, имеющиеся в языках программирования. Например в Паскале, процедура

GetTime(h,m,s,d) ,

дает значение текущего времени -- где h – часы; m - минуты; s – секунды; d – сотые доли сек (все параметры -- типа word).

Время выразить в сотых или тысячных долях сек.

5. Повторить эксперименты на другом компьютере, имеющем другую конфигурацию и параметры производительности.

6. Вычислить (или получить на компьютере) некоторые оценки производительности компьютера или его устройства или узла.

7. Составить таблицу и построить графики полученных зависимостей. При построении графиков на формате А4 по горизонтальной оси абсцисс должно быть количество повторений в логарифмическом масштабе (не менее 5 точек), по вертикальной оси ординат на всю высоту листа – время в сотых или тысячных долях секунды. При малом разрешении по вертикали можно построить 2 графика – для малых и больших значений времени.

Примечания:

1. При выполнении теста компьютер не должен быть загружен другими задачами.

2. При программировании необходимо использовать язык более низкого уровня (Ассемблер, Паскаль, Си, а не Delphi, Java).

3. При выполнении операций результаты должны иметь тот же тип, что и операнды, чтобы не было затрат на преобразование типов.

4. При выполнении операций результаты не должны вызывать переполнение.

Содержание отчета:

1. Цель работы.
2. Методика оценки
3. Алгоритм
4. Программа
5. Условия экспериментов (детальное описание параметров и конфигурации ВС)
6. Таблицы и графики полученных зависимостей.
7. Выводы.

Темы индивидуальных заданий.

1. Накладные расходы при работе с векторами различных типов.
2. Производительность дисковых операций (чтение и запись).
3. Сравнительная производительность при выполнении операций с действительными (2вида) и целыми числами.
4. Сравнительная производительность при выполнении циклического оператора.
5. Сравнительная производительность при выполнении операций с различными видами действительных (3 вида) чисел.
6. Сравнительная оценка производительности двух различных ВС (машин) при выполнении различных операций, а именно:
   а) короткие операции с действительными (3 вида) числами;
   б) длинные операции с действительными (3 вида) числами;
   в) короткие операции с целыми (3 вида) числами.
   г) длинные операции с целыми (3 вида) числами

7. Сравнительная оценка накладных расходов на организацию циклов различного типа (for, while, repeat) в различных языках программирования^ Ассемблер (a), Паскаль (b), Си (c).

8. Сравнительная производительность при выполнении операций преобразования типов:

а) различные целые типы – в различные действительные;

б) различные действительные типы - в другие действительные.

9. Матричные операции с числами различных типов

Вопросы для контроля.

1. Оценка производительности ЭВМ (проблемы, требования)
2. Составляющие производительности ЭВМ
3. Методы (способы) оценки производительности ЭВМ.

ЛАБОРАТОРНАЯ РАБОТА № 2

РАСПАРАЛЛЕЛИВАНИЕ ЗАДАЧИ НА ОСНОВЕ РАСЩЕПЛЕНИЯ ЦИКЛА

Цель работы: Изучение методик и способов распараллеливания вычислений с использованием метода расщепления цикла.

Теоретические сведения.

Основной характеристикой любого компьютера является производительность, т.е. количество операций, выполняемых в единицу времени. Существуют два направления повышения производительности: повышение частоты процессора на основе достижений технологии и распараллеливание действий (операций, команд, микрокоманд) за счет увеличения аппаратных ресурсов. Повышение частоты процессора становится все более сложным исходя из физических принципов. Принцип распараллеливания применяется в процессорах уже давно на основе конвейеризации. Однако дальнейшее увеличение ступеней конвейера затруднительно в связи с многочисленными конфликтами.[1,2]. Поэтому в настоящее время переходят к мультипроцессорным компьютерам. Практически все новые процессоры являются многоядерными.

Мультипроцессорные компьютеры повышают производительность лишь в том случае, когда программа позволяет обеспечить распараллеливание выполнения операций или их групп. В противном случае производительность мультипроцессорного компьютера повышается незначительно [3]. Распараллеливание может быть выполнено на различных уровнях.

Уровни параллелизма.

1) Уровень заданий - наиболее легко осуществлять распараллеливание. Недостатки: небольшая надежность; ситуации, когда один процессор решает длинную задачу, а остальные процессоры простаивают.
2) Распараллеливание на уровне процедур. Требуется анализ возможностей распараллеливания.
3) Распараллеливание на уровне операций или команд. Пример распараллеливание циклических операций.
4) Распараллеливание на уровне битов (арифметический уровень).

а). на уровне арифметических операций за счет уменьшения высоты дерева вычислений;

б). параллельная обработка ряда битов (секционированные микропроцессоры К1804)

В реальных задачах наиболее трудоемкими являются задачи обработки векторов и матриц. Эти задачи сводятся в программе к циклическим операциям. Распараллеливание циклических операций возможно на основе расщепления цикла, т.е. выполнения операций тела цикла с различными индексами на разных процессорах. В простейшем случае, когда количество итераций цикла N равно количеству процессоров K, на каждом процессоре тело цикла будет выполняться 1 раз; в случае N<K K-N процессоров будут свободны; при N>K на каждом процессоре будет выполняться L=N/K повторов тела цикла (дробное значение L необходимо округлить до ближайшего *большего* целого). Расщепление цикла возможно на основе двух подходов к распределению индексов между процессорами – последовательном и поблочном. Для случая N=10 и K=3 распределение индексов представлено в таблице:

№ про-цес-сора	Последовательное распределение № повтора				Поблочное распределение № повтора			
	1	2	3	4	1	2	3	4
	№ индекса				№ индекса			
1	1	4	7	10	1	2	3	4
2	2	5	8		5	6	7	8
3	3	6	9		9	10		

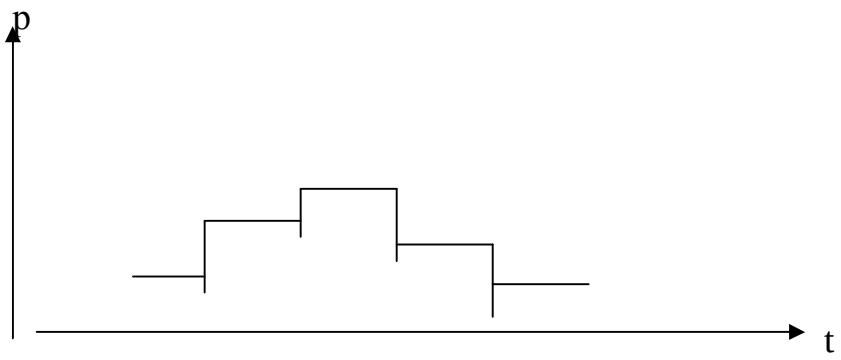
Выбор подхода к распределению зависит от алгоритма и возможностей дальнейшей загрузки процессоров. Пример распараллеливания на основе расщепления цикла приведен в приложении.

Оценка параллельных мультипроцессоров:
(метрика параллельных вычислений)
1) степень параллелизма
D(t) – количество процессоров, участвующих в вычислениях в каждый момент времени.
2) В любой программе не может быть такой ситуации, что все вычисления могут сделать параллельно. f – процент обработки последовательной части.

3) Δ - производительность

4) n – общее число процессоров (процессоры одинаковые)

5) профиль параллелизма программы.



6) от профиля зависит общий объем вычислений за интервал времени

$W = \Delta \int_{t_1}^{t_2} D(t)dt = \Delta \sum_{i=1}^{m} i \cdot t_i$

Max параллелизм профиля – m

7) O(n) – общее количество операций, выполняемых на многопроцессорных ВС

8) T(n) – время, затраченное на выполнение общего количества операций в виде числа квантов времени.

В однопроцессорных ВС T(1)=O(1), если учесть что квант времени равен длительности одной операции. При n>=2 T(n)<O(n)

9) Ускорение – это отношение времени выполнения на однопроцессорной машине к времени выполнении на многопроцессорной $S(n) = \frac{T(1)}{T(n)}$ $S(n) < n$

10) Эффективность ускорения на однопроцессорной машине $E(n) = \frac{S(n)}{n} < 1 = \frac{T(1)}{n \cdot T(n)}$ $\frac{1}{n} \le E(n) \le n$

11) Избыточность $R(n) = \frac{O(n)}{O(1)} = \frac{1}{E(n)} - 1$, где $1 \le R(n) \le n$

12) Коэффициент полезного использования или утилизации U(n)=R(n)*E(n)

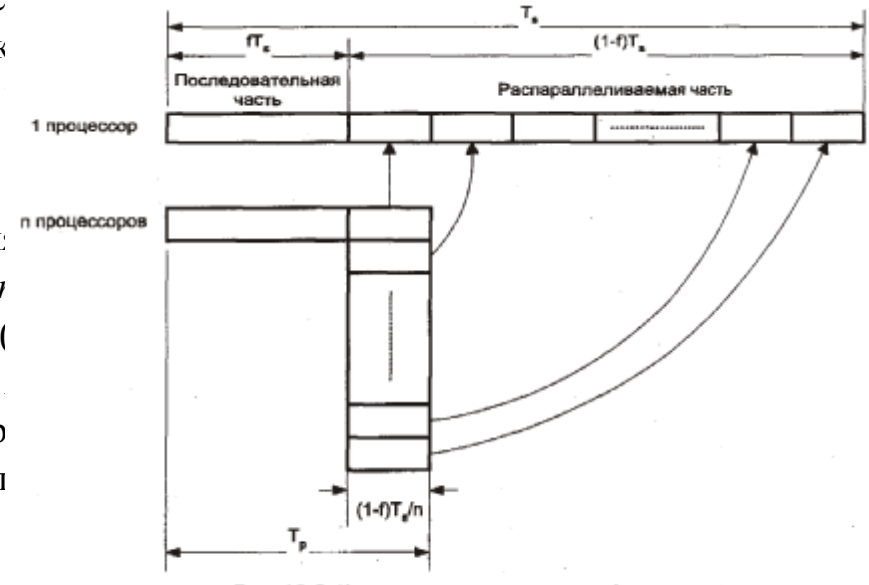
Издержки при использовании параллельных компьютеров по отношению к однопроцессорным:

1. Программные
   - дополнительные индексные вычисления при декомпозиции данных
2. Дисбаланс загрузки процессора
   - один процессор ждет завершения операции другими процессами
3. Коммуникационные издержки (время коммуникаций между процессорами).

- для уменьшении издержек желательно, чтобы вычислительные гранулы были по возможности больше

Оценка времени ожидания

Если с[illegible] ции P; количества к



При [illegible] торов:

$T_{ожс} = T_a + n$ [illegible] время прохождени[illegible]

Качес[illegible] $\cdot O(n)$,

где Q(n)<=S([illegible] f)

Закон [illegible] ре

T_s – вр[illegible]

f – дол[illegible]

Методические указания.

1. Изучить методы, методики и способы распараллеливания вычислений с использованием метода расщепления цикла, используя литературу и материалы лекций.

2. Разработать алгоритмы и программы для оценки затрат времени при последовательном и псевдопараллельном решении задачи (по вариантам).

3. При пседопараллельном выполнении задачи необходимо разбить операции на группы так, чтобы одна группа выполнялась как бы на одном процессоре, другая – на втором и т.д. В мультипроцессорной ВС эти группы операций будут выполняться параллельно, т.е. одновременно на разных процессорах. При пседопараллельном выполнении они должны выполняться последовательно на одном процессоре. При этом необходимо определять время выполнения каждой группы и принять максимальное из них как время выполнения при параллельном режиме.

4. Выполнить программу с несколькими значениями размера матрицы (вектора) N и с различным количеством процессоров K, определяя каждый раз время выполнения. Размеры матрицы N должны быть большими и изменяться от 10 или 100 до 1млн. с кратностью 10,

т.е. в логарифмическом масштабе; размеры вектора N должны изменяться от 1000 или 10000 до 1млрд. с кратностью 10. Количество процессоров К должно изменяться от 2 до 1024 с кратностью 2, т.е. должно быть 10 значений.

Для измерения времени использовать процедуры или функции, имеющиеся в языках программирования. Например в Паскале, процедура

GetTime(h,m,s,d) ,

дает значение текущего времени -- где h – часы; m - минуты; s – секунды; d – сотые доли сек (все параметры -- типа word).

Время выразить в сотых или тысячных долях сек.

5. Повторить эксперименты на другом компьютере, имеющем другую конфигурацию и параметры производительности.

6. Вычислить (или получить на компьютере) оценки затрат времени.

7. Составить таблицу и построить семейства графиков полученных зависимостей в 2-х системах координат:

1) t = f (N) при различных K;

2) t = f (K) при различных N; где N – размер матрицы (вектора); К - количество процессоров.

При построении графиков на формате А4 по горизонтальной оси абсцисс должно быть количество повторений в логарифмическом масштабе (не менее 5 точек), по вертикальной оси ординат на всю высоту листа – время в сотых или тысячных долях секунды. При малом разрешении по вертикали можно построить 2 графика – для малых и больших значений времени.

Примечания:

1. При выполнении теста компьютер не должен быть загружен другими задачами.

2. При программировании необходимо использовать язык, позволяющий организовать псевдопараллельную работу (например, создавать различные потоки) (Паскаль, Си++, Delphi, Java).

3. При выполнении операций результаты должны иметь тот же тип, что и операнды, чтобы не было затрат на преобразование типов.

4. При выполнении операций результаты не должны вызывать переполнение.

Содержание отчета:

1. Цель работы.
2. Методика оценки
3. Алгоритм
4. Программа
5. Условия экспериментов (детальное описание параметров и конфигурации ВС)
6. Таблицы и графики полученных зависимостей.
7. Выводы.

Темы индивидуальных заданий.

1. Умножение матриц.
2. Сложение матриц.
3. Расчет обратной матрицы.
4. Транспонирование матрицы.
5. Почленное сложение векторов.
6. Почленное умножение векторов.
7. Решение СЛАУ
8. Решение СНАУ
9. Решение СОДУ (заданным методом)
10. Расчет кратного интеграла методом Монте-Карло
11. Расчет определителя матрицы.
12. Расчет скалярного произведения векторов.

Вопросы и задания

1. Компьютеры параллельного. действия – уровни параллелизма.
2. Оценки компьютеров параллельного. действия.
3. Издержки компьютеров параллельного. действия.
4. Способы распараллеливания программ.
5. Два подхода к расщеплению цикла.

Пример

Распараллеливание на основе расщепления цикла.

Задача

Сложение векторов a и b

```
For i:=1 to M do
r[i]:= a[i]+b[i];
```

Распараллеливание
K – количество процессоров
L – количество повторов на 1 процессоре
L=M/K +1 (если M/K не целое число)
Npc -
L:=M div K;
If (M mod K) <> 0 then L:=L+1;

1-й вариант - последовательное исполнение

```
for mc:=1 to K do
begin
        {исполняется на каждом прц}
        j:=(mc-1)*K+npc;
        while j <= M do
        begin
                r[j]:= a[j] + b[j];
                j:=j+K;
        end;
end;
```

2-й вариант – поблочное исполнение с длиной блока L

```
for kc:=1 to K do
begin
        {исполняется на каждом прц}
        jn:=1+(kc-1)*L;
        jk:=jn+L-1;
        if jk > M then jk:=M;
        for j:=jn to jk do
                r[j]:= a[j] + b[j];
    end;
```

ЛАБОРАТОРНАЯ РАБОТА №3

КОНФЛИКТЫ ПРИ КОНВЕЙЕРНОЙ ОБРАБОТКЕ ДАННЫХ В ПРОЦЕССОРЕ

Цель работы: Ознакомление с принципами конвейерной обработки данных в процессоре. Изучение конфликтов при конвейерной обработке и методов и способов их предотвращения и разрешения.

Теоретические сведения

Конфликты при работе конвейеров

1. Структурные конфликты (риски), возникают из-за конфликтов по ресурсам при попытке нескольких команд одновременно обратиться к одному же ресурсу вычислительной машины (н/р памяти).
2. Конфликты по данным, возникают в случае, когда выполнение одной команды зависит от результата выполнения предыдущей команды.
3. Конфликты по управлению, возникают при конвейеризации команд переходов и других команд.

Разрешение конфликтов:

1. Структурные, возникают нечасто в связи с тем, что память кэшируется. Необходимость в некоторых случаях дублирования некоторых ресурсов.
2. Конфликты по данным, возникают при логической зависимости между командами. Для решения зависимости необходимо выбирать команды специальным образом.

Виды конфликтов по данным:

- чтение после записи (ЧПЗ - RAW)

Существует две команды и если одна команда пытается прочитать операнд используемых данных прежде, чем другая команда туда запишет.

- запись после чтения (ЗПЗ –WAR)

Одна команда пытается записать результат в приемник раньше, чем другая команда прочитывает эти данные.

- запись после записи

Одна команда пытается записать операнд прежде, чем будет записан результат другой команды, т.е. записи заканчиваются в неверном порядке.

Признак конфликта:

нарушение хотя бы одного условия Бернстайна, служащее критерием возможности возникновения конфликта по данным.

1) ЧПЗ

$O(i) \cap I(j)=0$;

O(i) – множество ячеек, при команде I;

I(j) – множество ячеек, при команде j, следующие за командой i.

2) ЗПЧ

$I(i) \cap O(j)=0$;

// I-input; O-output.

3) ЗПЗ

$O(i) \cap O(j)=0$;

Методы устранения конфликтов по данным:

1. Программные:

а) на этапе трансляции создается объектный код, в котором между командами, склонными к конфликту по данным, вставляются пустые операции;

б) на этапе трансляции создается объектный код, в котором команды i и j, склонные к конфликту по данным, переставляются так, чтобы между ними были команды, не связанные по данным с этими командами; при этом команда i может быть перенесена вперед – что более затруднительно, или команда j переставлена назад, что легче выполнимо.

2. Аппаратные

разрешают конфликт во время выполнения, т.е. j останавливается за несколько тактов, чтобы i успела завершить и миновать ступени конвейера, вызвавшие конфликт.

3. Ускоренное продвижение требуемой информации

forwarding data

Add R1, R2, R3	IF	ID	EX	MEM	WB				
SUB R4, R1 R5		IF	ID	EX	MEM	WB			

AND R6, R1, R7			IF	ID	EX	MEM	WB		
OR R8, R1, R9				IF	ID	EX	MEM	WB	
XOR R10, R1,R11					IF	ID	EX	MEM	WB

В регистр информация записывается упреждающим способом. Для этого существует специальные схемы АЛУ с цепями обхода и ускоренной пересылки.

Оптимизация с помощью компилятора: а = в + с; д = е + f;

Неоптимальное	Оптимальное

LW R_b, b
LW R_c, C
ADD R_a, R_b, R_c
SW a, R_a
LW R_e, e
LW R_f, f
SUB R_d, R_e, R_f
SW d, R_d
	LW R_b, b
	LW R_c, C
	LW R_e, e
	ADD R_a, R_b, R_c
	LW R_f, f
	SW a, R_a
	SUB R_d, R_e, R_f
	SW d, R_d

В простейшем алгоритме компиляции компилятор планирует распределение команд в одном и том же базовом блоке (это линейный участок кода, в котором нет переходов).

Использование временных регистровых результатов. Существуют логические регистры и физические регистры. Временные значения записываются в файловые регистры вместе с постоянными.

Временные значения становятся новым постоянным только после фиксации результата (5 степень конфликта). Завершение выполнение команды происходит тогда, когда все предыдущие команды завершились успешно в заданном программой порядке.

Такой подход называется методом переименования регистров.

Конфликты по управлению

Это неоднозначность при выборке следующей команды в случае перехода.

Методы борьбы:

1) Буферы предвыборки
2) Множественные потоки
3) Задержанный переход

4) Предсказание перехода

Наиболее часто используется предсказание перехода, для которого характерны следующие стратегии:

1) Статическое

Заранее выбирается направление.

- переход происходит всегда (ПВ), применяется при организации цикла, т.е. переход выполняемый;
- переход не происходит (ПНВ), т.е. переход невыполняемый;
- предсказание определяется по результатам профилирования (это тестовые прогоны программы);
- предсказание определяется кодом операции команды перехода;
- предсказание зависит от направления перехода (вперед или назад);
- при первом выполнении переход выполняется всегда.

2) Динамическое

Выполняется на основе прогноза.

Прогнозы различаются по глубине:

- однобитовый – т.е. такой, какой был на предыдущем шаге. Осуществляется на основе автомата.
- двухбитовый – т.е. на основе двух предыдущих выполнившихся переходов.

Методические указания.

1. Изучить виды конфликтов при конвейерной обработке данных в процессоре и методы и способы их предотвращения и разрешения, используя литературу и материалы лекций.

2. Разработать алгоритмы и программы, позволяющие обнаружить, и либо предотвратить конфликт, либо выйти из него.

3. Выполнить программу с несколькими (2 – 4) наборами исходных данных.

Примечания:

1. При программировании можно использовать любой язык.

2. Фрагмент программы, в котором возможен конфликт, должен быть написан на языке Ассемблера для RISC- процессора, т.е. с использованием только регистровых операндов и специальных операций работы с памятью.

Содержание отчета:

1. Цель работы.
2. Теоретические сведения о возникновении и разрешении конфликта.
3. Алгоритм обнаружения и предотвращения конфликта.
4. Программа обнаружения и предотвращения конфликта.
5. Условия экспериментов (детальное описание параметров и конфигурации ВС)
6. Фрагменты исходных программ и программ, полученных в результате преобразования, в которых конфликты устранены.
7. Выводы.

Темы индивидуальных заданий.

Вид конфликта, который надо разрешить, и способ разрешения:

1. Конфликт по данным типа ЧПЗ.
2. Конфликт по данным типа ЗПЧ.
3. Конфликт по данным типа ЗПЗ.
4. Конфликт по управлению – статическое предсказание выполняемого перехода.
5. Конфликт по управлению – статическое предсказание невыполняемого перехода.
6. Конфликт по управлению – статическое предсказание выполняемого перехода
7. Конфликт по управлению – динамическое предсказание на основе однобитового прогноза перехода;
8. Конфликт по управлению – динамическое предсказание на основе двухбитового прогноза перехода;

Вопросы и задания

4. Конвейерная организация. Типы конфликтов в конвейере.
5. Конвейерная организация. Структурные конфликты и способы минимизации потерь при них.
6. Конвейерная организация. Конфликты по данным и способы их разрешения.

7. Конвейерная организация. Конфликты по данным – классификация.

8. Конвейерная организация. Конфликты по данным – методика планирования компилятора для их устранения. Средства динамической оптимизации.

9. Конвейерная организация. Методы снижения потерь на выполнение команд перехода.

10. Конвейерная и суперскалярная обработка. Аппаратное прогнозирование направления переходов и снижение потерь на организацию переходов.

11. Виды конфликтов по данным при работе конвейера и способы их устранения.

12. Виды конфликтов по управлению при работе конвейера и способы их устранения.

13. Автомат двухбитового предсказания.

ЛАБОРАТОРНАЯ РАБОТА №4

ПРОГРАММНАЯ РЕАЛИЗАЦИЯ АЛГОРИТМА ТОМАСУЛО

1. Цель работы:

Исследовать алгоритм Томасуло и реализовать его программно.

2. Обобщенное словесное описание алгоритма

Программа представляет собой программную модель архитектуры устройства ПТ, разработанную Томасуло.
Устройство состоит из:

- очередь операций ПТ
- буфера загрузки из памяти
- буфера записи в память
- регистры ПТ
- станции резервирования сложения
- станции резервирования умножения
- устройство сложения
- устройство умножения
- ОШД (общая шина данных)

Структурная схема приведена на рис. 1.

Шаг прохода алгоритма Томасуло

На каждом шаге происходит просмотр состояния каждого из устройств.

1. Просмотр очереди команд

Из очереди забирается наиболее возможное число команд при условии наличия свободных буферов загрузки, записи, станций резервирования сложения, умножения.

2. Просмотр буфера загрузки

Если есть активный буфер загрузки, то он продолжает читать память еще один такт, если он закончил чтение, то результат выдается на ОШД, и буфер освобождается. Активного буфера загрузки теперь нет.

Если нет активного буфера загрузки, то он ищется. Если находится буфер, готовый к выполнению (с приоритетом 0), то он начинает выполняться и становится активным.

3. Просмотр устройства сложения

Так как у нас буферное устройство сложения, то оно параллельно может выполнять несколько команд сложения/вычитания. Поэтому каждая операция, выполняемая в нем, продолжает выполняться еще один такт. Если какая-то операция выполнена, то результат выдается на ОШД, если она свободна, а если занята – то результат сохраняется в устройстве сложения. Освобождаются станции резервирования сложения, где хранились операции, завершившиеся в этом такте.

4. Просмотр устройства умножения

Так как у нас буферное устройство умножения, то оно параллельно может выполнять несколько команд умножения/деления. Поэтому каждая операция, выполняемая в нем, продолжает выполняться еще один такт. Если какая-то операция выполнена, то результат выдается на ОШД, если она свободна, а если занята – то результат сохраняется в устройстве умножения. Освобождаются станции резервирования умножения, где хранились операции, завершившиеся в этом такте.

5. Просмотр ОШД

Если на ОШД есть результат выполнения какой-либо операции, то все устройства, ждущие операнда, забирают значение с ОШД, если это ожидаемый операнд.

6. Просмотр станции резервирования сложения

Если после просмотра ОШД какие-либо операции получили ожидаемый операнд и готовы к выполнению, то они отправляются на выполнение в устройство сложения.

7. Просмотр станции резервирования умножения

Если после просмотра ОШД какие-либо операции получили ожидаемый операнд и готовы к выполнению, то они отправляются на выполнение в устройство умножения.

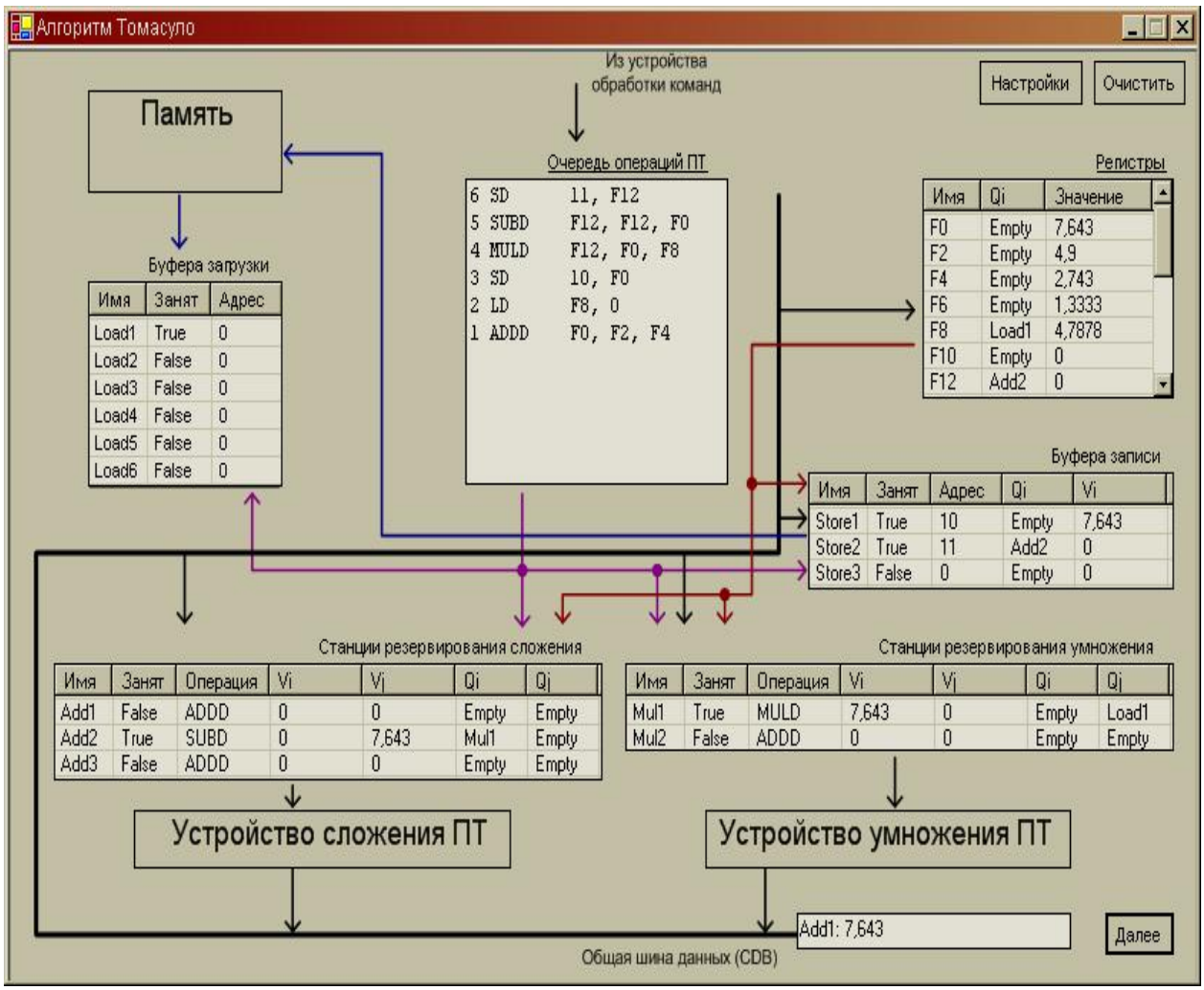
8. Просмотр буферов записи

Если есть активный буфер записи, то он продолжает записывать память еще один такт, если он закончил запись, то буфер освобождается. Активного буфера записи теперь нет.
Если нет активного буфера записи, то он ищется. Если находится буфер, готовый к выполнению (имеющий операнд, с приоритетом 0), то он начинает выполняться и становится активным.

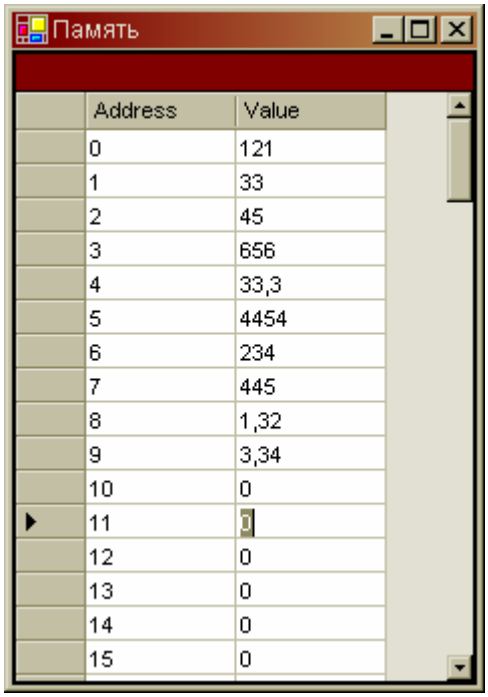
Примечание: Блок схемы работы программы и текст программы приведены в электронном варианте методических указаний.

3. Интерфейс

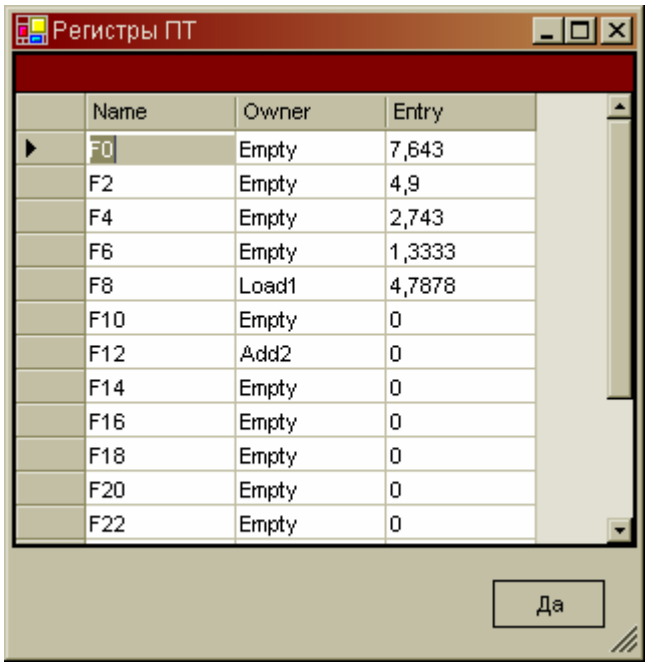
Окошко «Алгоритм Томасуло»



Окошко «Память»



Окошко «Регистры»



Окошко «Код программы»

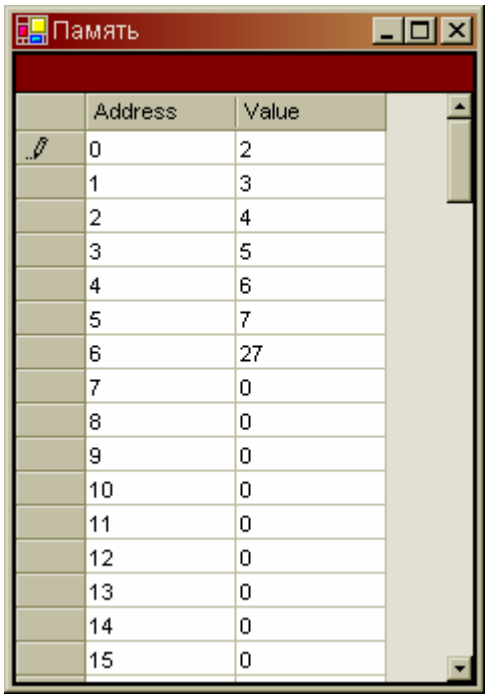
4. Тесты

Код

```
; проверка конфликта типа RAW, WAW (регистры);
; задача: найти сумму первых 6-ти ячеек памяти
; и записать результат в 7-ую;
; исходные данные: ненулевые значения
;                  0-5 ячеек памяти,
;                  f2 = 0
ld     f0, 0
addd   f2, f2, f0
```

```
ld      f0, 1
addd    f2, f2, f0
ld      f0, 2
addd    f2, f2, f0
ld      f0, 3
addd    f2, f2, f0
ld      f0, 4
addd    f2, f2, f0
ld      f0, 5
addd    f2, f2, f0
sd      6,  f2
```

После выполнения



	Address	Value
	0	2
	1	3
	2	4
	3	5
	4	6
	5	7
	6	27
	7	0
	8	0
	9	0
	10	0
	11	0
	12	0
	13	0
	14	0
	15	0

5. Выводы

В процессе выполнения лабораторной работы мы исследовали алгоритм Томасуло и реализовали его программно. Надо отметить, что так как мы реализовывали программно, то алгоритм, который сам по себе предусматривает параллельную обработку устройств, пришлось преобразовать в последовательный. Также обнаружилось слабое место алгоритма – ОШД, так как она единственна, но результат на нее выдают буфера загрузки, устройства сложения, умножения и этим устройствам приходится ждать освобождения ОШД,

Вопросы для контроля.

1. Конвейерная организация. Конфликты по данным и способы их разрешения.
2. Конвейерная организация. АЛУ с цепями обхода и ускоренной пересылки.
3. Конвейерная организация. Конфликты по данным – классификация.
4. Конвейерная и суперскалярная обработка. Идеи динамической оптимизации и ее реализация с централизованной схемой обнаружения конфликтов.
5. Алгоритм Томасуло.

Литература

1. Танненбаум Э. Архитектура компьютера. 4-е изд. – СПб.: Питер. 2003. –704 с.
2. Цилькер Б.Я., Орлов С.А. Организация ЭВМ и систем: Учебник для вузов. – СПб.: Питер. 2004. – 668с.
3. Организация ЭВМ. 5-е изд./ К. Хамахер, З. Вранешич, С. Заки. – Спб.: Питер, 2003. – 848с.
4. Распределенные системы. Принципы и парадигмы / Э. Таненбаум, М. ван Стеен. – СПб.. Питер. 2003. – 877с.

Содержание