ИНСТИТУТ ПРИКЛАДНОЙ МАТЕМАТИКИ
Имени М. В. КЕЛДЫША
РОССИЙСКОЙ АКАДЕМИИ НАУК

В. М. Михелев

ПЕДАНТ
Архитектура многопроцессорного комплекса

МОСКВА
2001

УДК 681.31

Резюме.

Описывается вариант тредовой архитектуры для многопроцессорных вычислительных систем, когда поток вычислений организуется путем передачи данных между тредами. Рассматриваются проблемы связи процессоров с расщепленной памятью и синхронизации при параллельных вычислениях.

Abstract.

A variant of computer multithreading architecture is described. It has a good look at problem of connection processors and a shared memory as well as synchronization of parallel execution.

Введение.

Скорость вычислительной машины можно увеличить либо за счет повышения тактовой частоты, либо за счет объединения нескольких процессоров, способных одновременно участвовать в решении одной и той же задачи. Задача распараллеливания вычислений в таких многопроцессорных системах может выполняться либо программистом (транслятором), либо автоматически. Возможность автоматического распараллеливания на уровне отдельных инструкций – fine grain параллелизм существенно зависит от принятой модели вычислений. Например, фон-неймановская модель, требующая последовательного выполнения инструкций, допускает параллелизм лишь на линейном участке программы за счет объединения нескольких команд в одну так называемую команду с широким словом (VLIW), что, однако, дает максимум двукратное увеличение скорости.

Модель вычислений, принятая в dataflow машинах [1,2] – инструкция выполняется в тот момент, когда становятся известными ее операнды, потенциально позволяет использовать весь существующий в программе параллелизм. Однако необходимость приписывания уникального имени любому значению, участвующему в процессе вычислений, и использование ассоциативной памяти существенно затрудняют реализацию dataflow машин [3].

Можно представить себе другую, отличную от принятой в dataflow машинах модель вычислений, которой также внутренне присущ параллелизм – тредовую модель [4]. Под тредом (thread) обычно понимается последовательность инструкций, выполняемая отдельным процессором. Начало треда (адрес в программе) задается в команде, инициирующей его работу, а конец - командой останова выполнения треда. Внутри треда могут быть команды, инициирующие параллельную работу других тредов. Запуск треда может сопровождается передачей ему некоторых данных , например, значений регистров. Именно такое взаимодействие тредов отражено в названии архитектуры – ПЕДАНТ – ПЕредача ДАННых между Тредами.

Приведенные далее компоненты ПЕДАНТа являются дальнейшей разработкой системы, описанной в [5].

Работа выполнена при поддержке Российского Фонда Фундаментальных Исследований.

Проблемы.

Прежде чем переходить к детальному описанию предлагаемой архитектуры полезно, по крайней мере, перечислить решаемые проблемы.

Наличие многих процессоров допускает одновременное обращение к памяти сразу из нескольких. Следовательно, необходим специальный коммутатор, разрешающий такие конфликтные ситуации. Проблема усложняется при разбиении памяти на блоки и применении кэш памятей.

Очевидно также, что при обращении из тредов к объектам в памяти требуется специальный механизм, блокирующий в процессе изменения объекта одним тредом чтение или запись в него из других тредов.

В процессе изменения объекта в памяти при выполнении одного треда, например, при сложении или сдвиге может произойти обращение к этому же объекту из другого треда. Необходима, очевидно, защита от такого несвоевременного обращения путем прерывания выполнения второго треда.

Помимо приведенного, существует еще несколько случаев, требующих прерывания выполнения треда. К ним, в частности, относится “промах” при обращении к памяти, то есть случай, когда объект отсутствует в кэш и, следовательно, требуется обращение к основной памяти. Анализ показывает [6], что на это затрачивается до 50% времени.

Легко понять, что, например, при вычислении выражения в том случае, когда два треда вычисляют разные операнды одной и той же инструкции, их окончания сливаются и, следовательно, процесс вычисления может быть продолжен лишь одним из двух процессоров, очевидно, тем, который последним по времени подошел к выполнению этой инструкции. Следовательно, такие ситуации также требуют механизма синхронизации.

В многопроцессорных системах возникает проблема обмена данными между процессорами или, как в нашем случае, между тредами. Необходимость такого обмена возникает, например, в случае накопления данных в регистре одного процессора, получаемых в процессе выполнения тредов на других процессорах. Обмен данными может быть использован и для синхронизации.

В случае одновременного обращения к одной и той же процедуре из разных тредов возникает проблема распределения и базирования локальных областей. Очевидно, что для этой цели также требуется специальный механизм, поддерживающий соответствующие соглашения о связях между вызывающей и вызываемой процедурами.

Элементы архитектуры.

1. Процессор.

Прежде чем описывать архитектуру в целом рассмотрим компоненты отдельного процессора, схема которого изображена на рисунке. Здесь

Вычислитель – устройство для выполнения операций.

Регистр памяти – на этот регистр заносится команда обращения к памяти. Команда содержит код операции, например, чтение, адрес в памяти, номер процессора, выдавшего команду и в случае записи или передачи данных – значение.

Регистр результата – регистр, в который записывается результат выборки из памяти, предназначенный данному процессор.

Регистр связи – содержит объект, переданный процессору другим процессором.

Локальная память команд – в эту память записывается та часть программы, к которой чаще всего обращаются в процессе вычислений.

Слово состояния процессора – регистр, содержащий

- текущий виртуальный номер процессора,
- статус процессора (свободный, занятый или ожидающий),
- счетчик команды,
- статус команды (в локальной или общей памяти),
- код условия,
- состояние операции.

Регистры – локальные регистры. Регистры именуются

S – регистр результата,

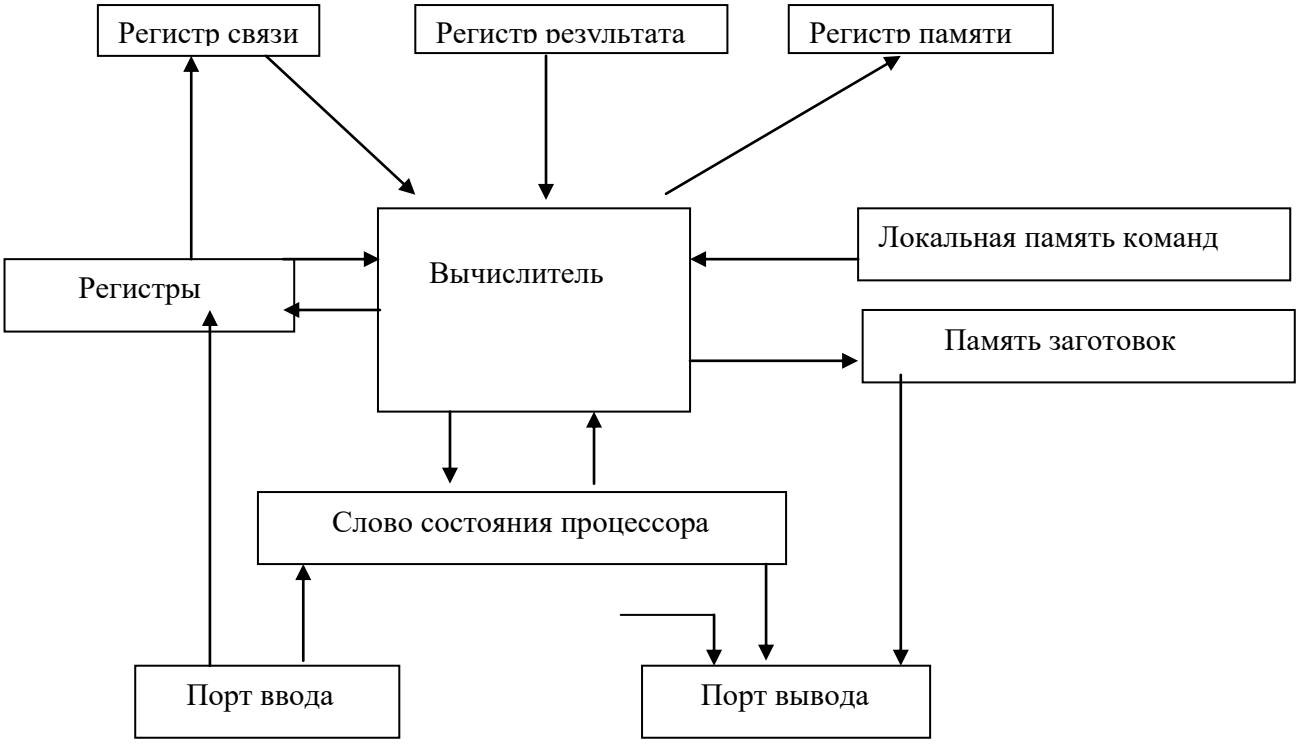
A,B,C,D,E,F,G,H – регистры, содержащие значения баз и промежуточные результаты.

Состояние процессора – состоит из слова состояния процессора и регистров.

Память заготовок – память, в которую записываются состояния процессоров. Память организована в виде очереди.

Порт ввода – содержит передаваемое процессору состояние. Порт ввода связан с кольцом виртуальных процессоров. В случае, когда этот порт свободен и в соответствующей ему позиции кольца находится виртуальный процессор, ищущий свободный физический, состояние виртуального передается в порт ввода.

Порт вывода – которому в случае, если он свободен и работа процессора для текущего треда прерывается, передается состояние процессора. Если порт вывода свободен и очередь заготовок не пуста, то первый элемент из этой очереди также передается порту вывода



.

Рассматриваемая архитектура подразумевает наличие двух видов процессоров – описанного выше физического и виртуального. Виртуальный процессор это всего лишь объект, содержащий состояние процессора. Он располагается в кольце виртуальных процессоров.

Если физический процессор свободен, то ему может быть передано либо состояние виртуального процессора из порта ввода, либо состояние из памяти заготовок. Приоритет - за портом ввода. Если при выполнении треда произошло прерывание, например, по несвоевременному доступу к памяти, то состояние процессора переписывается в порт вывода и процессор освобождается. Если выполнение треда завершается командой останова – все команды треда выполнены, то состояние завершения передается в порт вывода для освобождения соответствующего виртуального процессора. Виртуальный процессор, однако, не освобождается, если состояние процессора берется из памяти заготовок, то есть в этом случае очередной тред выполняется но том же процессоре.

2. Формат команды.

Ниже приведен формат команды. Команда состоит из четырех полей и может содержать один или два операнда.

Операция	Индик	Рег	Адрес/Регистр/Литерал

Рег – регистры – S,A,B,C,D,E,F,G,H. Регистр А обычно используется в качестве базы программы, если она записана в основной памяти, регистр В – база данных, а регистр S – выполняет функцию первого операнда и регистра результата в однооперандных командах.

Индик – индикатор способа адресации. От способа адресации зависит и количество заданных операндов в команде. Команды с индикаторами **a,c,d,w –** однооперандные, а с индикаторами **b**,**r,p,l –** двухоперандные.

a - признак автоматического базирования Если это адрес в команде передачи данных, то в качестве базы берется регистр А, если адрес данных, то регистр В. Исполнительный адрес вычисляется следующим образом (База)+(Рег)+Адрес. В командах с индикатором **a** первым операндом служит результат предыдущей выполненной операции, а вторым – операнд по адресу.

b – также считается признаком автоматического базирования, но в этом случае Рег используется как первый операнд, а исполнительный адрес вычисляется как (База)+Адрес.

c – для команд передачи данных считается, что это адрес в локальной памяти команд процессора. Этот адрес не базируется. Если это адрес данных, то исполнительный адрес вычисляется как (Рег)+Адрес.

d – косвенная адресация в командах передачи данных, использующих дескриптор. Адрес самого дескриптора задается аналогично **a.** Дескриптор – три элемента памяти. В первом элементе задается адрес треда, которому передаются данные, во втором – база той части программы, которой тред принадлежит, а в третьем – база соответствующих программе данных. При передаче данных по дескриптору устанавливаются базы вызываемой программы, то есть содержимое второго элемента, если он не нулевой устанавливается в регистр **A**, а содержимое третьего регистра заносится в регистр **B.**

w – при выполнении команды с таким индикатором адресуемый объект рассматривается как рабочий, используемый для реализации рандеву двух операндов. Исполнительный адрес вычисляется как в случае **a**.

l – в поле Адрес записывается константное или адресное выражение.

r – оба операнда заданы в регистрах: первый в поле Рег, второй – в поле Регистр. Результат остается в первом операнде. Команды с таким индикатором не изменяют регистр **S** как регистр результата. Однако, он может использоваться в качестве операнда.

p – также как и в случае **r** оба операнда заданы в регистрах: первый в поле Рег, а второй в регистре связи. В поле Регистр задается имя регистра-счетчика. Команды с таким индикатором работают следующим образом. Команда не выполняется до тех пор, пока не будет получено значение в регистре связи. По получению значения команда выполняется, из счетчика вычитается единица и проверяется его значение. Если значение равно нулю, то выполняется следующая команда, в противном случае цикл выполнения команды повторяется. Если имя регистра в поле Регистр отсутствует, то команда выполняется лишь один раз.

3. Элементы памяти.

Объекты в памяти адресуются с точность до слова. Каждое слово сопровождается тегом, указывающим на наличии в памяти значения и допустимости его изменения. Размер тега – два разряда.

Значение тега:

00 – объект не содержит значения (Cl),
01 – разрешена запись (Wr),
10 – разрешено чтение (Rd),
11 – значение в процессе изменения (Up).

По-существу, тег Up служит тегом блокировки. Поскольку изменение объекта может выполняться несколькими командами одного треда, то на это время доступ к объекту из другого треда должен быть заблокирован. Примером может служить изменение части слова.

4. Команды процессора.

Ниже приведен список команд, используемых для решения тестовых примеров на модели. В дальнейшем список этот будет расширен. Все команды делятся на семь групп.

1. Выполняющие действия над двумя операндами. Значение результата остается в первом операнде.

add, sub, mult, div, mod – арифметические команды с целыми аргументами.

or, and, not – логические команды.

eq, ge, ne, le, lt, gt – команды, устанавливающие отношения между операндами. Команды вырабатывают код условия – true или false.

2. Передающие данные – значения всех регистров и код условия - треду по заданному в команде адресу. Команды этой группы позволяют распараллеливать вычисления.

fork - разветвление. Команда используется для запуска выполнения треда по указанному в команде адресу.

gener – генератор значений. Команда используется для параллельного выполнения команд тела цикла с параметром, изменяющимся в интервале – исходное значение регистра C .. 0. Команда выполняется следующим образом. Если значение в регистре C больше или равно нулю, то данные передаются треду, адрес которого на единицу больше адреса команды gener и значение регистра C уменьшается на единицу. Когда значение в регистре C становится отрицательным, то выполняется команда по адресу, заданному в команде gener.

3. Передающие значение другому треду.

conn – передача значения первого операнда другому треду, выполняемому на другом процессоре. Номер этого процессора задается во втором операнде.

4. Передающие управление. Команды выполняются внутри одного треда.

br - безусловная передача управления команде по адресу.

cond – передача управления по условию. Если условие true, то следующей выполняется команда по адресу, если false, то текстуально следующая.

5. Команды, организующие обращение к процедурам.

call - по адресу в команде записывается дескриптор возврата. Первый операнд – адрес точки возврата.

par – запись шаблона дескриптора параметров. Первый операнд – адрес процедуры записывается во второе слово дескриптора. В третье слово дескриптора записывается регистр D.

6. Команды, присваивающие значение. Напомним, что в одноопернадных командах первый операнд – регистр S.

rdv – первому операнду присваивается значение второго.

rdup – аналогично rdv. Второму операнду, если он в памяти, присваивается тег up, защищающий его от несвоевременного чтения командой из другого треда.

wrv – второму операнду присваивается значение первого. Если второй операнд - объект в памяти, то ему присваивается тег rd.

wrup – снимается блокировка, установленная rdup. Далее аналогично wrv.

prnm – виртуальный номер процессора присваивается объекту по адресу.

crd – чтение кода условия.

cwr – запись кода условия.

push - значение регистров C,D,E,F,G,H,S сохраняются в области, адрес которой задается во втором операнде.

pop – восстанавливает значения регистров.

7. Команда останова.

stop - прекращение выполнения треда.

5. Общая схема взаимодействия процессоров с памятью.

Ниже на рисунке приведена общая схема многопроцессорного комплекса с архитектурой ПЕДАНТ.

Здесь:

Физические процессоры – совокупность процессоров. Их порты ввода и вывода связаны с кольцом виртуальных процессоров, регистры памяти - с устройством выбора процессора для обмена, а регистр результата с памятью. Регистр связи получает значение непосредственно от устройства выбора.

Выбор процессора для обмена с памятью – устройство для разрешения конфликтных ситуаций, возникающих при одновременном обращении к памяти или другому процессору из нескольких процессоров одновременно.

Память данных и команд – совокупность памятей, включая кэш памяти.

Синхронизатор – память, сохраняющая запросы к основной памяти, которые не были реализованы из-за отсутствия объекта в кэш или из-за несоответствия тегов, либо значения, передаваемые виртуальным процессорам, которым в момент передачи не соответствовали физические. Размер памяти равен количеству виртуальных процессоров.

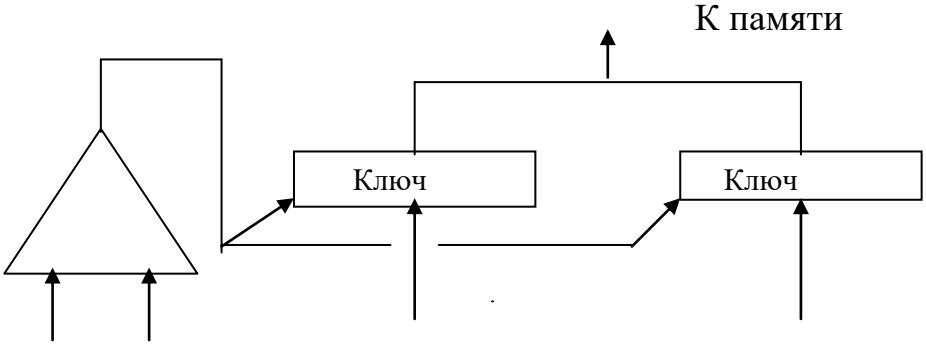
Кольцо виртуальных процессоров – кольцевая память, каждый элемент которой содержит виртуальный процессор. Виртуальные процессоры заносятся в кольцо через порты вывода физических процессоров как результат выполнения команд, передающих данные. Напомним, что в этом случае передаются данные треду, для выполнения которого и выделяется виртуальный процессор.

Контроль присутствия – устройство, позволяющее определить, соответствует ли в данный момент физический процессор виртуальному, которому передается значение. При несоответствии значение попадает в память синхронизации.

Рассмотрим теперь более подробно работу каждого из устройств.

6. Выбор процессора при обращении к памяти.

В многопроцессорных системах принципиальной является проблема разрешения конфликтной ситуации при обращении к одной и той же ячейки памяти одновременно из разных процессоров. Ниже рассматривается один из вариантов решения этой проблемы. Назовем такое устройство “выбор с привилегиями”. Устройство состоит из двух частей – выбор номера процессора и ключей, позволяющих по этому номеру выбрать соответствующий регистр памяти.



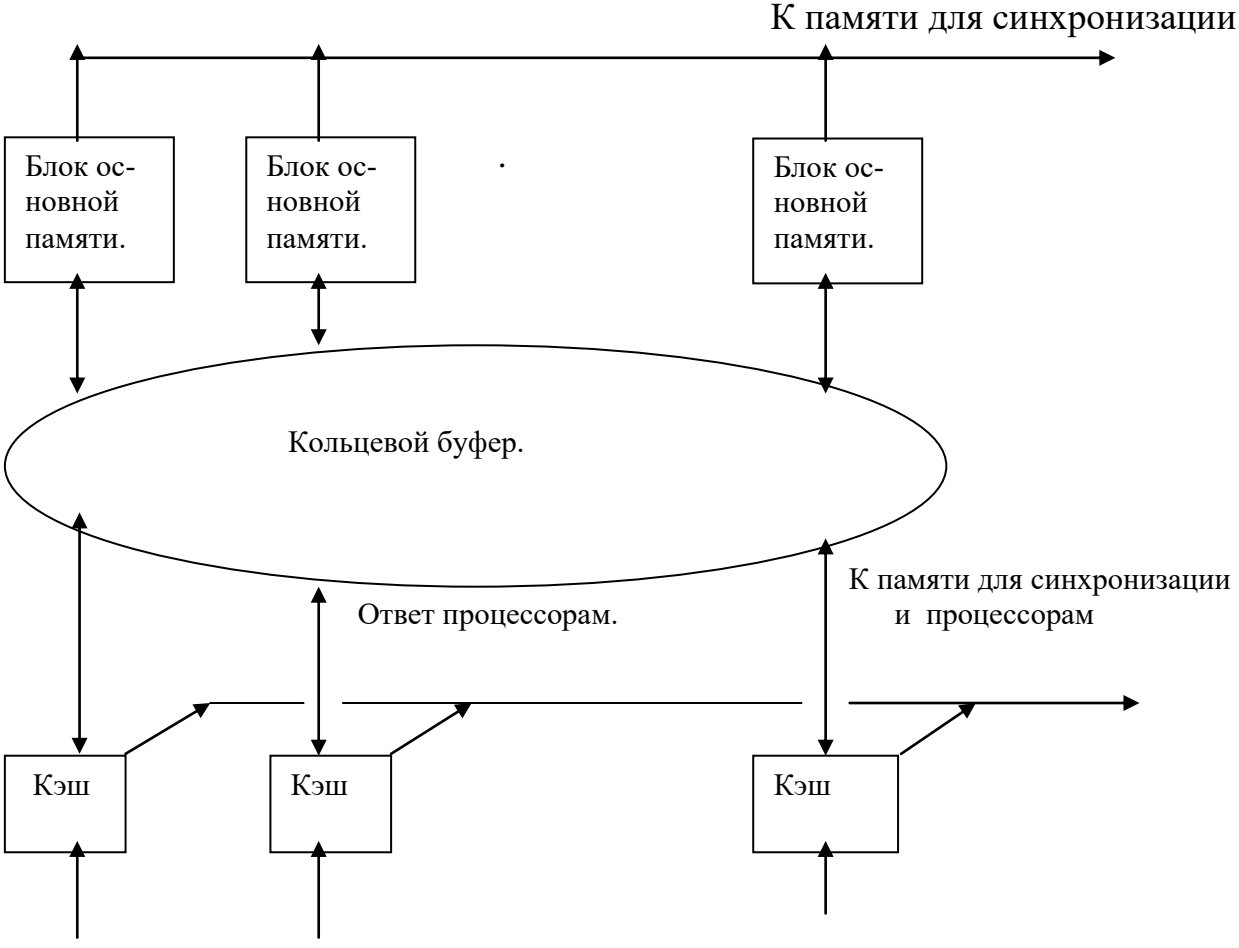
Здесь “Номер процессора” - виртуальный номер процессора, записанный в слове состояния физического процессора. Количество номеров, из которых производится выбор, очевидно, равно числу физических процессоров. Возможны различные схемы выбора, например, шкала номеров

или древовидная структура из компонент, разрешающих конфликтные ситуации, или их сочетание. Критерием выбора является, конечно, скорость выбора. После выбора номера содержимое соответствующего регистра памяти передается памяти, а сам регистр - сбрасывается.

Устройство выбора задает очередность обращения процессоров к памяти. Задержка из-за очереди увеличивает время выборки. Однако, это время можно уменьшить за счет разбиения памяти на блоки, в которых адреса группируются, например, по модулю (четные-нечетные и т. п.). В этом случае устройство выбора тиражируется в соответствии с количеством модулей, допуская обращение к памяти одновременно в нескольких модулях.

7. Память данных.

Будем считать, что память данных состоит из нескольких блоков с непересекающимися адресами. Рассмотрим один из возможных способов организации такой памяти – память с кольцевым буфером.



От блоков выбора процессора при обращении к памяти.

Когда при обращении к кэш памяти происходит “промах”, то есть объект в кэш отсутствует, то команда обращения к памяти передается в кольцевой буфер. В нем команда циркулирует до тех пор, пока необходимый блок памяти не завершит предыдущее задание и не начнет выполнять данную команду. Ответы из блоков также попадают в кольцевой буфер и из него подкачиваются в кэш.

Команда обращения к кэш может быть как выполненной, так и не выполненной. Последнее бывает в двух случаях – несовпадение тегов и “промах”. И в том и в другом случае происходит прерывание выполнения треда командой ответа с одновременной записью запроса в память синхронизации.

8. Синхронизатор.

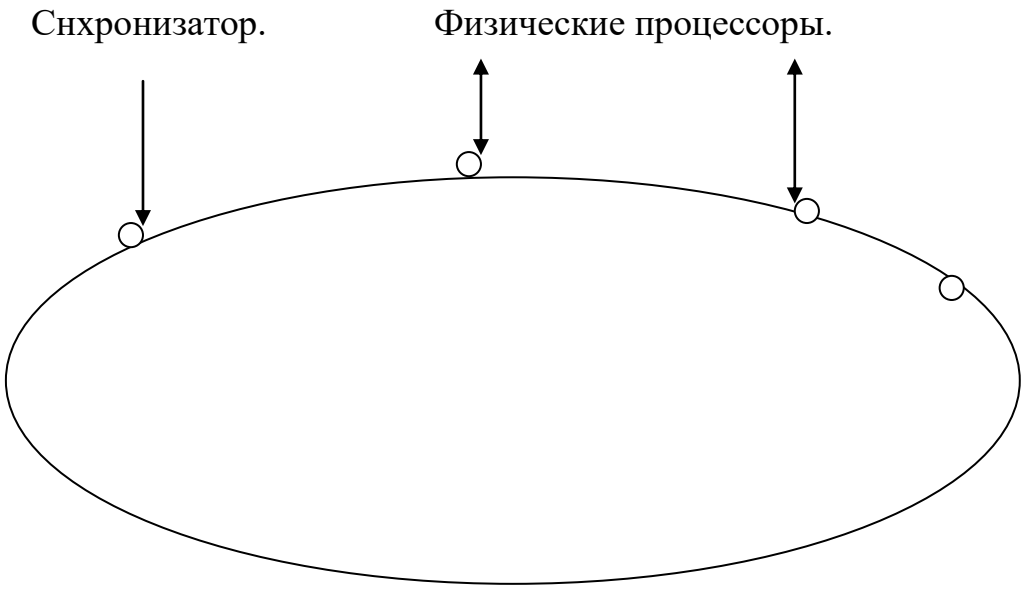
По-существу, синхронизатор это специальная память, в которой хранятся неудовлетворенные запросы при обращении процессора к основной памяти и при передаче значения другому процессору. Количество объектов в памяти равно числу виртуальных процессоров. Неудовлетворенный запрос записывается в тот объект памяти, номер которого равен номеру процессора, выдавшего запрос или не получившего значение.

Синхронизатор используется для реализации запросов при изменении ситуация в памяти. Рассматриваются два варианта изменения – подкачка после “промаха” и перемена тега. При изменении ситуации происходит обращение ко всем объектам синхронизатора, в которых хранящийся запрос имеет адрес, равный адресу того объекта в основной памяти, который был подкачен или у которого изменился тег. Удовлетворенные таким образом запросы передаются в кольцо виртуальных процессоров процессорам с соответствующими номерами.

Синхронизатор можно рассматривать как полуассоциативную память, в которую записывается по адресу, а выбирается по ассоциации.

9. Кольцо виртуальных процессоров.

Кольцо виртуальных процессоров – это кольцевая память, объектами которой являются виртуальные процессоры. Каждый такт изменяет положение виртуальных процессоров на одну позицию. На рисунке кружочками обозначены виртуальные процессоры.



.

Физические процессоры связаны с кольцом через порты ввода и вывода. Каждому занятому физическому процессору соответствует в кольце его виртуальный.

Виртуальные процессоры могут находиться в одном из следующих четырех состояний:

- свободен,
- поиск свободного физического процессора,
- ожидание информации от синхронизатора,
- занят физическим процессором.

В свою очередь порт ввода может находиться в состоянии:

- свободен,
- захвачен.

Состояниями порта вывода могут быть:

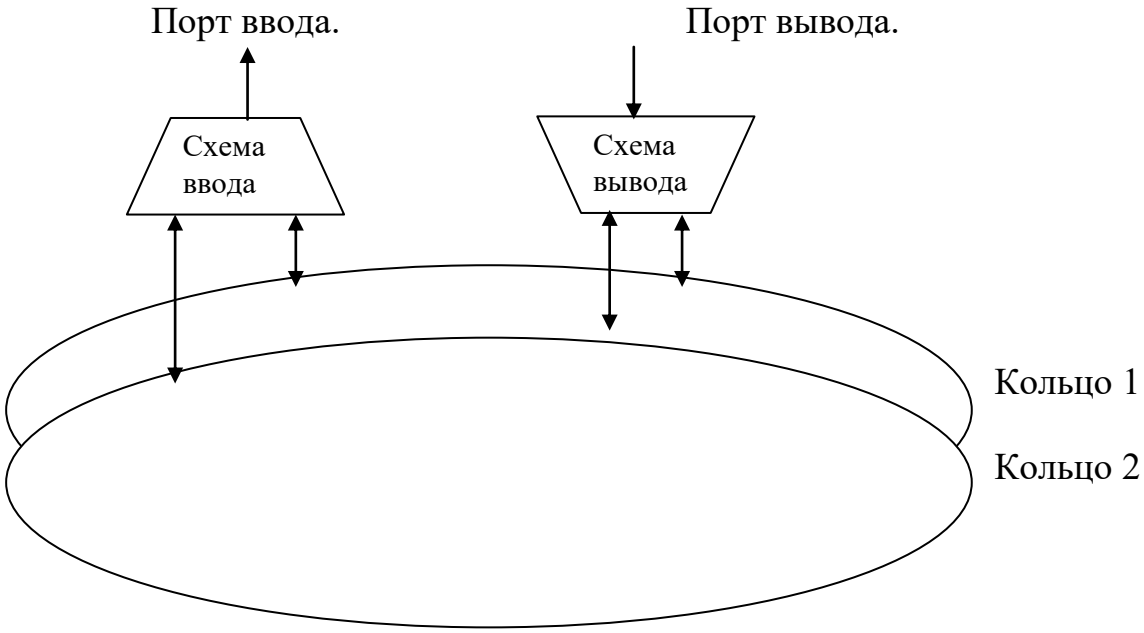
- свободен, - ожидание свободного виртуального процессора для передачи данных,
- ожидание виртуального процессора, соответствующего по номеру текущему физическому, для сохранения состояния в случае прерывания исполнения треда или его освобождения

Передача данных между физическими процессорами и находящимися в кольце виртуальными происходит в соответствии с их состояниям в момент совпадения позиции кольца с позицией процессора.

При передаче данных от синхронизатора рассматривается готовность объекта с номером, равным номеру виртуального процессора, находящегося

в позиции синхронизатора, и состояние самого этого процессора. Передача данных, очевидно, меняет состояние.

Время поиска в кольце виртуальных процессоров процессора, находящегося в нужном состоянии, пропорционально количеству позиций кольца. Нетрудно видеть, что можно уменьшить в несколько раз длину кольца при сохранении количества виртуальных процессоров. Для этого нужно лишь заменить одно кольцо несколькими, усложнив схемы выбора. Ниже приведен пример двойного кольца.



Примеры и приемы программирования.

Все приведенные примеры записываются на языке ассемблера. Поля в предложениях ассемблера называются:

Опер – операция,
И – индикатор способа адресации,
Р – идентификатор регистра,
Адр- поле адреса.

1. Перемножение матриц.

Рассмотрим следующий пример.

```
MM,NN,RR: array[1..20] of integer;
                Sum: integer;

For I in 1..20
   Loop
      For j in 1..20
         Loop  Sum:=0;
                  For k in 1..20
                     Loop Sum+=MM[i,k]*NN[k,j]
                End loop;
              RR[i,j]:=Sum
        End loop
   End loop
```

Опер	И	Р	Адр	Комментарий
rdv	l	C	19	19 => C
gener	c		end	Конец перемножения матриц. →
mult	l	C	20	
wtv	r	C	D	i*20 => D
rdv	l	C	19	
gener	c		Line	→
wtv	r	C	E	j => E
rdv	l	C	19	k => C
prnm	r		H	Виртуальный номер процессора => H
gener	c		Summa	→
wrv	r	C	F	
add	r	F	D	i*20+k => F
rdv	a	F	MM	
wrv	r	S	F	MM[j,k] => F
mult	l	C	20	k*20 => C
add	r	C	E	k*20+j => C
rdv	a	C	NN	NN[k,j] => S
mult	r	S	F	
conn	r	S	H	Передача MM[i,k]*NN[k,j] процессору c

```
                                        номером в H
        stop

Summa   rdv   l    G   20          В G количество слагаемых.
        rdv   l    S   0
        add   p    S   G           В S накапливается сумма значений из
                                     регистра связи.
        add   r    E   D           i*20+j => E
        wrv   a    E   RR          Запись значения в RR[I,j]
        stop

Line    stop

End     stop
```

В приведенном примере строки из пробелов отделяют части программы, соответствующие тредам. Стрелки → указывают на продолжение треда.

2. Упорядочение массива целых чисел.

Рассмотрим пример, демонстрирующий защиту объекта, который изменяется одним тредом, от несвоевременного доступа со стороны другого треда. С этой целью рассмотрим задачу упорядочения массива целых чисел методом пузырька. Допустим, что задача решается независимо, но одновременно сразу несколькими процессорами, и используется стандартный алгоритм сортировки. Понятно, что задача будет решаться быстрее, поскольку процессоры будут помогать друг другу. Для определенности выберем 10 виртуальных процессоров и массив MM из 1000 слов по два байта.

```
        Rdv   l   D    1          D = 0 - признак упорядоченности
        Rdv   l   C    9
        Gener c       End         Генерация 10 тредов  →

Cycl    eq    l   D    0
        cond  c       Stp         Окончание сортировки
        rdv   l   D    0
        rdv   l   E    0          E – номер элемента в массиве
        rdup  a   E    MM         Блокировка обращения из другого треда
```

```
        wrv     r   S    F          В F – левый элемент пары
Chck add    l   E    2
        rdup    a   E   MM
        wrt     r   S    G         В G – правый элемент пары
        le      r   F    G
        cond    c        Right
        wtv     r   G    H         Обмен значениями
        wrv     r   F    G
        wrv     r   H    F
        rdv     r   D    1         Сортировка не закончена
Right sub       l   E    2
        rdv     r   S    F
        wrup    a   E   MM
        add     l   E    4
        wrv     r   F    G          Новое значение левого  элемента пары
        lt      l   E   2000
        cond    c       Chck
        br      c       Cycl
Stp   stop

End   stop
```

В этом примере тред, содержащий команду gener, передает данные десяти другим тредам, каждый из которых на своем собственном процессоре выполняет сортировку массива. Здесь тред – целая программа со своими циклами.

3. Вызов процедуры.

Вызов процедуры передачей данных отличается тем, что любая передача параметра приводит к началу выполнения треда в вызываемой программе. Второе отличие – процедуры должны быть реентерабельными. Это означает наличие собственной локальной памяти для каждого вызова процедуры. Если это условие не соблюдено, то потребуется дополнительная синхронизация для установления последовательности вызова процедур.

Очевидно также, что в случае процедуры-функции команда, принимающая значение, должна иметь индикатор **w**, чтобы обеспечить синхронизацию операндов. Кроме того, необходимо обеспечить восстановление тех значений регистров, которые они имели до обращения к процедуре.

Вообще говоря, команды, реализующие вызов и возврат из процедуры, должны выполнять соглашение о связи вызывающей и вызываемой процедур. Один из возможных вариантов такого соглашения реализован в приведенном ниже примере. В дальнейшем будем считать, что

- программа расположена в основной памяти,
- все процедуры адресуются с нуля, то есть значение базы (регистр **A**) процедуры устанавливается на ее начало,
- адреса меток задаются по отношению к началу процедуры, а адрес самой процедуры устанавливается загрузчиком по отношению к началу памяти,
- при каждом вызове процедуры ей выделяется локальная память данных, база которой находится в регистре **D,**
- локальная память начинается с области сохранения регистров, за которой следуют дескриптор возврата и дескриптор параметров,
- параметр передается по значению командой передачи данных треду.

Рассмотрим следующий простой пример.

```
Proc RlSign(x,y) returns integer
   If x*y>0
     then return 0
     else  return 1
   end if;


… p*RlSign(m,n) …       (* вызов RlSign *)
```

Пусть аргумент m записан в локальной памяти вызывающей процедуры по адресу MM, аргумент n – по адресу NN, а p – по адресу PP. Как обычно, локальная память данных в вызывающей процедуре базируется регистром B. Будем считать, что размер слова в этой памяти равен четырем байтам.

Локальная память данных начинается областью сохранения регистров – C,D,E,F,G,H,S, состоящей из семи слов. За ней следуют дескриптор возврата и шаблон дескриптора передачи параметров. В шаблоне в момент передачи параметра первое слово заменяется адресом входа, по которому этот параметр передается.

Ниже приведены фрагменты текста программы.

Процедура RlSign.

```
RlSign     proc
RlSign#1 nop
RlSign#2 mult  w        WRlSign    WRlSign– объект для синхронизации,
                                    который находится в локальной
          gt     l   S   0          памяти данных процедуры RlSign.
          pop    c   B   0          Восстановление регистров
          rdv    l   S   0
          cond d   D   0          D снова указывает на начало области
          rdv    l   S   1
          br     d   D   0
         endp
```

Передача параметров и обращение к процедуре RlSign.

```
          push   c  D   0          Сохранение регистров
          rdv    l  S   Rt
          call    c  D  28         Запись дескриптора возврата
          rdv    l  S   RlSign     База процедур
          par    c  D   40         Запись шаблона для дескриптора параметра
          rdv    l  S   RlSign#1
          wrv    c  D   40
          rdv    a      MM
          fork   d  D   40         Передача первого параметра
          rdv    l   S   RlSign#2
          wrv    c   D   40
          rdv    a       NN
          fork   d   D   40        Передача второго параметра
          rdv    a       PP
Rt        mult  w        Wmlt      Wmlt – объект для синхронизации, который
                                    расположен  а локальной памяти данных.
           …
```

Заключение.

Предлагаемая архитектура многопроцессорного комплекса решает большинство проблем, возникающих при совместной работе нескольких процессоров над одной задачей в стиле fine grain параллелизма. Заметим, что используемая тредовая модель вычислений позволяет рассматривать в качестве треда любую часть программы от нескольких команд до отдельных процедур. На наш взгляд существенным в предлагаемой архитектуре является использование специальной памяти для синхронизации, которая значительно облегчает решение таких проблем как попытка использование значения до его получения, ожидание подкачки из основной памяти в кэш или передачи значения от одного процессора другому. Использование очереди заготовок существенно ускоряет переключение физического процессора при замене тредов. В настоящее время проводится анализ работы системы на программной модели. Анализ результатов, по-видимому, приведет к дальнейшему усовершенствованию системы.

Литература.

1. Dennis, Jack B. Data flow supercomputers.
   Computer 13, 11 (Nov. 1980).
2. Gard, J. R., Kirkham, C. C., Watson, I.
   The Manchester prototype dataflow computer.
   CACM 28, 1 (January 1985).
3. G. M. Papandoupulos, K. R. Traub.
   Multithreading: A Revisionist View of Dataflow Architectures.
   ACM SIGARCH V. 19, N 3 (May 1991)
4. R. Alverson, D. Callahan, D. Cummings, B. Koblenz, A. Portenfield, B. Smith.
   The Tera computer system.
   Proceedings of the 1990 ACM International Conference on Supercomputing, june 1990.
5. В.М. Михелев
   ПРОК Архитектура многопроцессорного комплекса, препринт ИПМ им. Келдыша РАН, 2000, N 59.
6. Cvetanonic.
   Alpha Server 4100 Performance Characterization.
   Digital Technical Jornal 8(4) 1996.

Оглавление.