ИНСТИТУТ ПРИКЛАДНОЙ МАТЕМАТИКИ
имени М.В. Келдыша
РОССИЙСКОЙ АКАДЕМИИ НАУК

В.М.Михелев

ВИКТОР М
Архитектура многопроцессорного комплекса

МОСКВА
2004

УДК 681.31

Резюме

Описывается вариант архитектуры для многопроцессорных вычислительных комплексов. Рассматриваются проблемы разбиения программы на параллельно выполняемые части – трассы, связи процессоров с расщепленной памятью и автоматизации синхронизации при параллельных вычислениях.

Abstract.

A variant of computer multithreading architecture is described. It has a good look at problem of dividing program into parts (traces), connection processors and a shared memory as well as synchronization of parallel execution.

Введение.

Предлагаемая ниже работа служит очередным этапом в проводимом автором исследовании по эффективной реализации многопроцессорных систем [1,2]. Аббревиатура **ВИКТОР** расшифровывается как **ВИ**ртуальные процессоры в **К**омпьютере с **Т**рассовой **ОР**ганизацией вычислений. Пока, не вдаваясь в подробности, можно рассматривать предлагаемую архитектуру как архитектуру **MIMD** компьютера с разделяемой памятью, то есть компьютера с несколькими процессорами, в котором одновременно существуют несколько потоков команд и несколько потоков данных .

Проблемы, которые обычно решаются при разработке архитектур таких компьютеров, можно разбить на две группы -

(1) разделение программы на части и синхронизация их совместной работы и

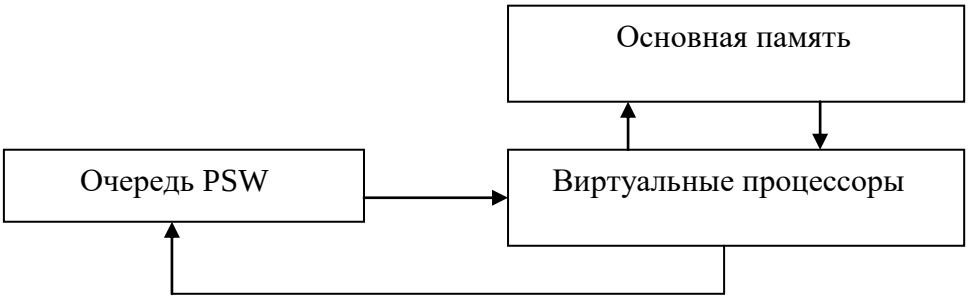
(2) реализация связи процессоров и памяти, минимизирующая неизбежные потери, возникающие из-за появления очередей к общим ресурсам.

Уже в самом названии работы отражены наиболее существенные элементы рассматриваемой архитектуры – использование *виртуальных процессоров* (не путать с виртуальными компьютерами) и предлагаемый способ разбиения программы на части – *трассы.* Считается, что каждая трасса выполняется на выделенном ей виртуальном процессоре до момента ее завершения. После чего на этом процессоре начинает выполняться другая трасса.

Описание архитектуры **ВИКТОР** разделено на две части – логическую, описывающую выполнение трасс на виртуальных компьютерах, и схемотехническую, в которой описываются схемы, реализующие отдельные элементы архитектуры. Первую назовем **работа компьютера**, а вторую – **реализация компьютера**.

1 Работа компьютера.

Ниже приведена схема компьютера с точки зрения его логического функционирования.



Работа компьютера состоит из совместной работы отдельных виртуальных процессоров. Каждый такой процессор выполняет последовательность команд переданной ему трассы.

1.1 Трасса.

Трасса это часть программы. Начало трассы указывается в PSW – слове состояния виртуального процессора. Оканчивается трасса по достижению команды stop. После окончания виртуальный процессор освобождается и может быть загружен новой трассой из очереди PSW. Очередь PSW пополняется в процессе выполнения трасс, командами, порождающими трассы. Именно эти команды позволяют распараллеливать вычисления. Таких команд три.

gnr – команда, которая порождает трассы - ветви цикла,

fork - безусловное порождение новой трассы с заданным в команде её начальным адресом и

cfork - порождение новой трассы при наличии выработанного предыдущими командами условия.

PSW, соответствующее началу программы, записывается в очередь при запуске задачи операционной системой.

Трасса может быть помечена командой **lbm**. Это позволяет завершить выполнение всех помеченных трасс при возникновении каких-либо исключительных ситуаций, например, окончания вычислений, выявленного при работе одной из таких трасс. Для этой цели служат две команды

brk, которая заносит в регистр прерывания номера задачи и метки, и

cut, которая при совпадении метки трассы и метки в регистре передает управление по адресу в команде.

1.2 Слово состояния виртуального процессора – PSW.

PSW задает начальное состояние виртуального процессора и содержит следующие поля

- номера задачи,
- метка трассы,
- адреса начала трассы и
- состояния регистров.

Заметим, информация от вызывающей трассы к вызываемой передается на регистрах. Такой информацией может быть, в частности, значения баз программы и данных в основной памяти. На языке ассемблера регистры именуются

S – регистр результата,

A,B,C,D,E,F,G,H – регистры баз и промежуточных результатов.

1.3 Синхронизация работы трасс.

Одним из важнейших аспектов работы многопроцессорных является синхронизация работы трасс. Действительно, в процессе вычисления трассы могут изменять значение общей для всех трасс переменной. Очевидно, что здесь существенен порядок записи и чтения. Второй вариант необходимости синхронизации возникает в случае, когда трассы вычисляют операнды одной и той же команды, например, при сложении двух подвыражений, вычисляемых разными трассами. В этом случае та трасса, которая приходит к точке синхронизации первой, заканчивается фиктивной командой останова. Рассмотрим соответствующие аппаратные средства, используемые в системе ВИКТОР .

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

Значение тега:

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

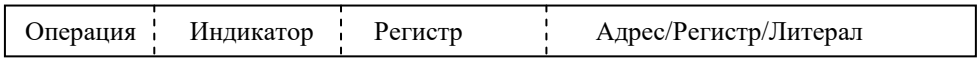
При обращении к памяти происходит проверка тега на соответствие требованию команды обращения к памяти. Если условие не выполняется, то команда не выполняется. Тег Up используется как тег блокировки - при изменение объекта командами одной трассы доступ к объекту из другой трассы должен быть заблокирован. Примерами могут служить изменение части слова или накопление суммы операндов, взятых из разных трасс.

Некоторые команды после их выполнения изменяют значение тега:

- запись **wrv** заменяет тег **Wr** на тег **Rd**,
- чтение с блокировкой **rdup** заменяет тег **Rd** на **Up**,
- запись со снятием блокировки **wru**p заменяет тег **Up** на тег Rd,
- запись, игнорирующая тег, **wrdr** записывает значение с тегом **Wr**,
- запись тега **wrtg** безусловно записывает тег **Wr,** не изменяя значения.

1.4 Формат команды.

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



Операция – код операции команды.

Индикатор – индикатор способа адресации. От способа адресации зависит, в частности, и количество заданных в команде операндов. Команды с индикаторами **a,c,d,w –** однооперандные, а с индикаторами **b**,**r,l –** двухоперандные. Значения индикаторов приведены в Приложении 1.

Регистр – поле, содержащее один из регистров S,A,B,C,D,E,F,G,H. Регистр A обычно используется в качестве базы программы, регистр B – базы данных, а регистр S – выполняет функцию первого операнда и регистра результата в однооперандных командах. В командах, где оба операнда – регистры, это поле первого операнда.

Адрес/Регистр/литерал – в этом поле в зависимости от индикатора может быть записан либо адрес второго операнда, выбираемого из памяти, либо второй операнд – регистр, либо константа в команде.

Описание команд дано в Приложении 2.

1.5 Примеры программ.

Ниже приведены три программы, демонстрирующие порождение трасс и синхронизацию их работы. Все приведенные примеры записываются на языке ассемблера. Поля в предложениях ассемблера называются:

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

1.5.1 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
```

В приведенном ниже примере вторая команда gnr порождает трассы, каждая из которых вычисляет один элемент матрицы результата.

```
        Опер  И   Р   Адр       Комментарий

        rdv   l   S   20-1        19 => S
        gnr   a       end       Конец перемножения матриц.      →

        mult  l   S   20
        wtv   r   S   D         i*20 => D
        rdv   l   S   19
        gnr   a       Line                                      →

        wrv   r   S   E         j => E
        rdv   l   F   0         k:=0 => F
        rdv   l   C   0         Сумма C=0
Cycle   eq    r   F   20
        cond  a       Summa
        rdv   r   H   F
        add   r   H   D         i*20+k => H
        rdv   a   H   MM
        wrv   r   S   G         MM[j,k] => G
        rdv   r   H   20
        mult  r   H   F
        add   r   H   E         k*20+j => H
        rdv   a   H   NN        NN[k,j] => S
        mult  r   S   G         MM[i,k]*NN[k,j] => S
        add   r   C   S
        add   l   F   1         k:=k+1
        br    a       Cycle
Summa   add   r   D   E         i*20+j => E
        Rdv   r   S   C
        wrv   a   D   RR        Запись значения в RR[I,j]
        stop

Line    stop

End     stop
```

В приведенном примере строки из пробелов отделяют части программы, соответствующие трассам. Стрелки → указывают на продолжение трассы. В комментариях => означает “содержится в регистре”.

1.5.2 Упорядочение массива целых чисел.

Рассмотрим пример, демонстрирующий защиту объекта, который изменяется одной трассой, от несвоевременного доступа со стороны другой трассы. 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 запросим 60 процессоров и массив MM из 1000 слов. Защищаемым объектом здесь является элемент массива. Поэтому запись и чтение элемента выполняется командами **rdup** и **wrup.**

В этом примере трасса, содержащая команду gnr, передает данные десяти другим трассам, каждая из которых на своем собственном виртуальном компьютере выполняет сортировку массива. Однако, если компьютер содержит лишь 10 виртуальных машин, то 50 из запрошенных в виде PSW перейдут в очередь PSW и будут запрашиваться по мере освобождения виртуальных процессоров. Когда сортировка закончится какой-либо одной из трасс, то команда **brk** завершит работу всех других трасс, поскольку все они помечены одной и той же меткой, а также выбросит незатребованные PSW из очереди.

```
 proc sort
  local list: array[1..1000] of integer;
       nonsort: boolean:=true;
       temp,max: integer;
     while nonsort
      loop
        for current,next as list(1..1000)
          loop nonsort:=false;
               if current>next
                 then
                   temp:=next;
                   next:=current;
                   current:=temp;
                   nonsort:=true
               end if
         end loop
       end loop
```

Ниже приведена соответствующая программа на языке ассемблера.

```
rdv  1 D  1      конец сортировки
lbm  1 no 77     метка для всех порождаемых трасс
rdv  1 S  59      количество запрашиваемых процессоров – 1 (трасс)
```

```
        gnr   c  no end

cycle  cut     c  no  end    завершение выполнения по прерыванию
        eq      l  D  0
        cond  c  no stp
        rdv    l  D   0
        rdv    l  E   0
        rdup   c  E  MM     текущий
        wrv    r  S   F
check add     l  E   1
        rdup  c  E   MM       следующий
        wrv    r  S   G
        le      r  F   G
        cond  c  no  right
        wrv    r  G  H
        rdv    r  G  F
        wrv    r  H  F
        rdv    l  D  1
 right  rdv    r  S   F
        wrup c  E   MM-1
        rdv    r  F   G
        lt      l  E   999         размер массива -1
        cond  c  no check
        rdv    r  S   G
        wrup  c  E  MM          последний
        br      c  no cycle

 stp    brk     l  no 77           призыв к завершению выполнения всех трасс
        stop    r  no no

 end    stop    r  no no
```

1.5.3 Синхронизация при помощи рабочих объектов.

В процессе вычисления выражения промежуточные результаты, присваиваемые рабочим объектам, обычно используются для объединения независимо вычисляемых частей выражения. В случае, когда эти части вычисляются параллельно на разных процессорах, рабочие объекты могут быть применены и для синхронизации.

Синхронизация выполняется командой с двумя операндами, вторым из которых является рабочий объект. В команде устанавливается индикатор **w.** В исходном состоянии рабочий объект имеет тег **wr**. По сути дела к команде с синхронизацией обращаются дважды. Трасса, которая первая обращается к такой команде , записывает в рабочую ячейку результат. Запись разрешена,

поскольку тег равен **wr.** После записи работа трассы завершается фиктивной командой stop. Тег рабочей ячейки становится равным **rd**. Трасса, выполняющая эту команду второй, завершает ее выполнение, поскольку присутствуют оба операнда. Операнд, полученный первым, очевидно, может быть прочтен, поскольку его тег – **rd.** Порядок операндов определяется кодом последней выполненной команды. Если это команда разветвления, то переданный ей операнд – второй. После выполнения команды синхронизации тег рабочей ячейки переходит в исходное состояние – **wr.** Рассмотрим пример. Допустим, что подвыражения в выражении

(M+N)*(P-10)

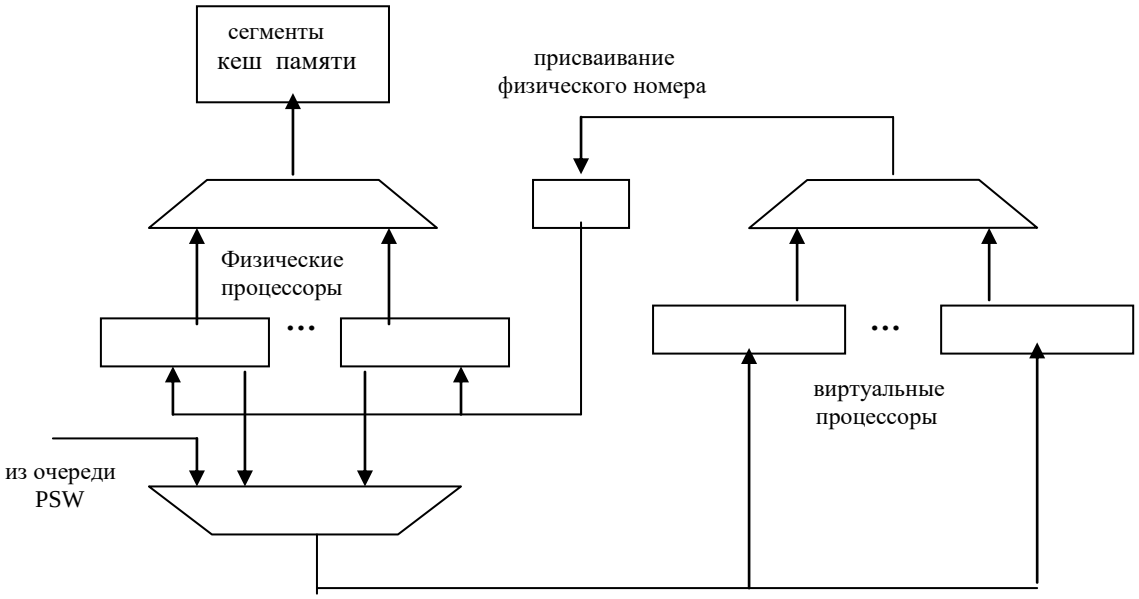
вычисляются разными трассами. Пусть первый процессор начинает считать с точки First, а второй с точки Second.

```
First   rdv   a   no  M
        add   a   no  N
        fork  a   no  Mlt
Second  rdv   a   no  P
        sub   l   S   10
Mlt     mult  w   no  Wa
```

2. Реализация компьютера.

Рассмотрим несколько важных на наш взгляд аспектов реализации предлагаемой архитектуры. Описание достаточно конспективно и дает лишь общее представление, не вдаваясь в некоторые, порой существенные, подробности.

Конечно, виртуальные процессоры сами по себе не выполняют составляющих трассу команд. Для этой цели используются физические процессоры. Ниже приведена схема, демонстрирующая связь виртуальных и физических процессоров.



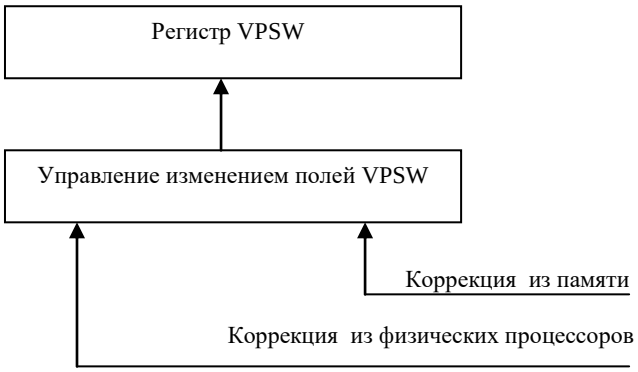
На схеме в виде трапеций обозначены устройства “выборки с привилегией”, позволяющие разрешать конфликтные ситуации, возникающие при одновременном обращении к одному устройству нескольких претендентов, как, например, из нескольких процессоров к одному и тому же сегменту памяти. Более подробно это устройство будет рассмотрено ниже.

Необходимость введения виртуальных процессоров обусловлено возникающими при вычислении трассы прерываниями, вызываемыми, например, обращением к памяти. Их отсутствие привело бы к простою физических процессоров и, следовательно, к замедлению вычислений. Следует отметить, реализация виртуального процессора на порядок проще физического.

Компьютер работает следующим образом. Получив номер свободного виртуального процессора, PSW из очереди PSW переписывается в этот виртуальный процессор. Процессор получает признак готовности. Готовый виртуальный процессор при наличии свободного физического захватывает его номер, и его VPSW переписывается в физический. Начинается процесс вычисления трассы. В случае возникновения прерывания VPSW переносится в соответствующий ему виртуальный процессор. Признак готовности снимается. Признак восстанавливается после выполнения условий прерывания, например, приходит результат чтения из памяти. VPSW готового виртуального процессора вновь передается свободному физическому, и выполнение команд трассы продолжается. Окончание вычислений приводит к освобождению как физического, так и виртуального процессоров.

2.1 Виртуальный процессор.

Виртуальный процессор состоит из двух частей – регистра, содержащего VPSW, и схемы управления, позволяющей изменять содержимое полей VPSW. Информация для изменения полей поступает как из памяти, так и из физических процессоров.

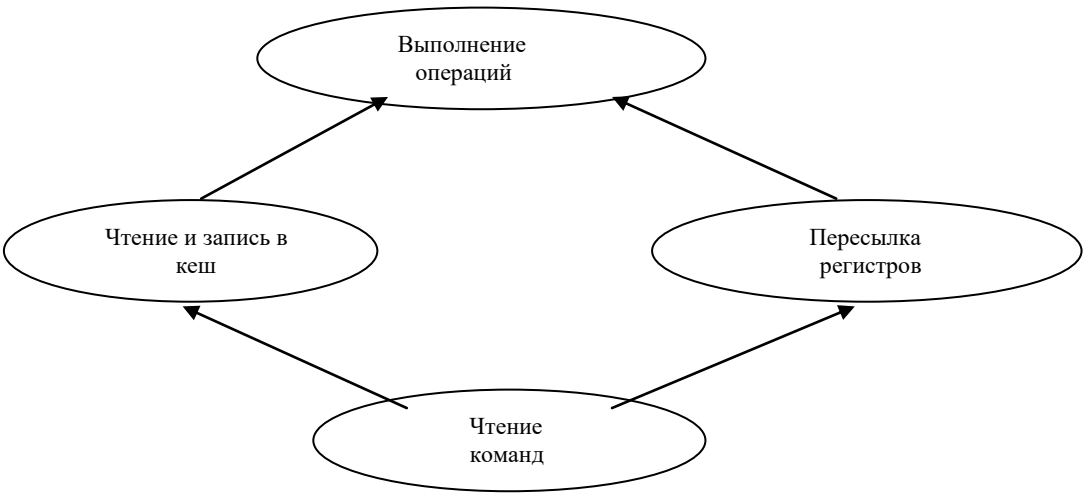


В качестве одного из полей VPSW содержит PSW. Кроме того, в WPSV входят поля, определяющие состояния автоматов – компонент физического процессора, и ряд специальных признаков. При реализации управления VPSW решались три задачи.

1. Выполнение необходимых временных зависимостей.
2. Недопущение возникновения тупиковых ситуаций.
3. Недопущение “засыпания” процессоров.

2.2 Физический процессор.

Физический процессор представляет собой совокупность из четырех автоматов, выполняющих чтение команды, чтение и запись в кеш, пересылку регистров и выполнение операций. Автоматы работают независимо друг от друга при наличии переданной информации и при условии, что используемые регистры не используются другим автоматом. Такой способ выполнения команды предполагает больший параллелизм при выполнении команд, чем конвейерная обработка, конечно, при условии записи команд в последовательности, оптимальной для выполнения.



1.3 Выбор с привилегиями.

Ниже представлена схема, реализующая выбор с привилегиями.

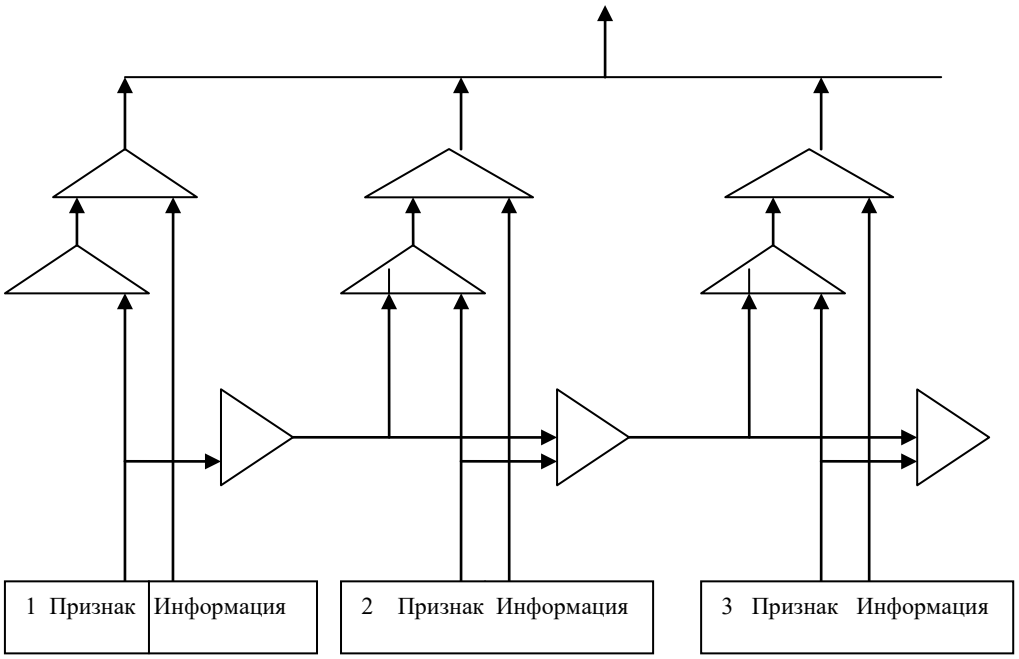
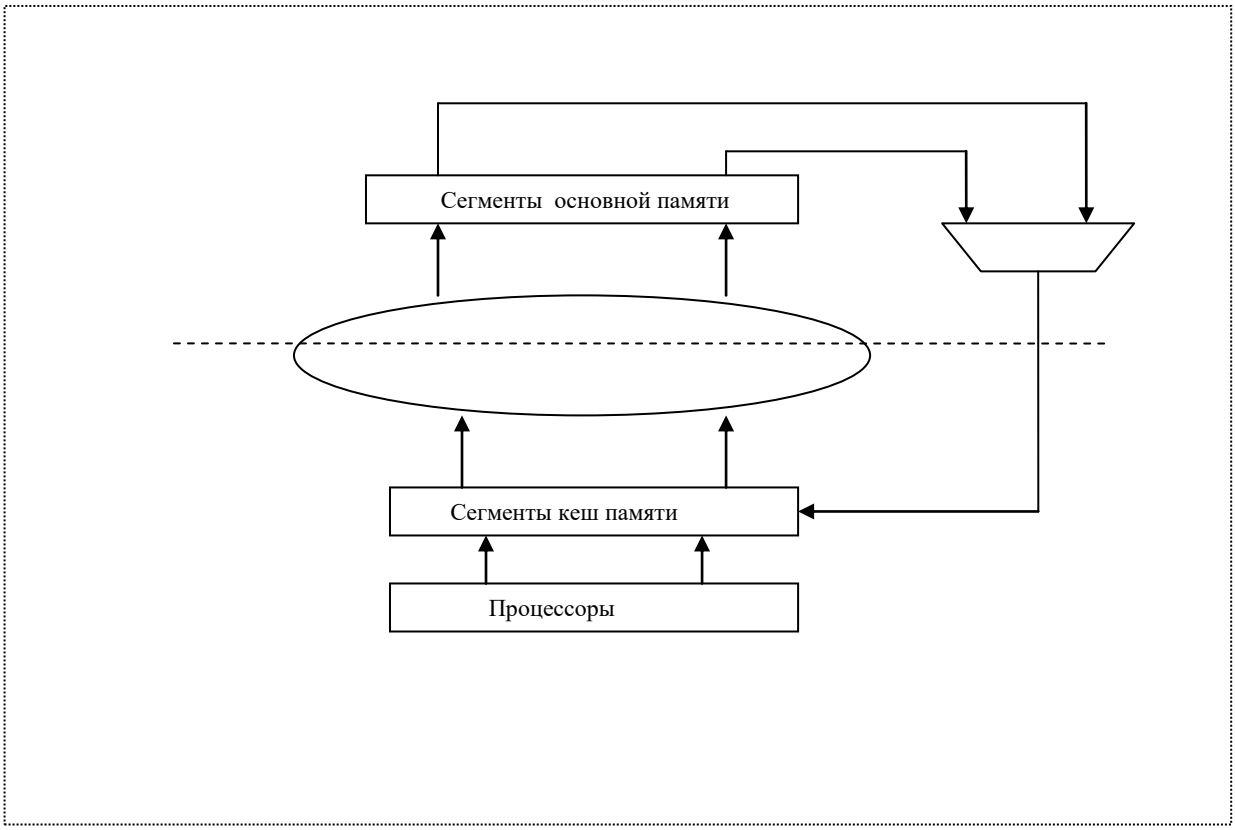


Схема позволяет выбрать из нескольких, поступающих одновременно на схему объектов, один. Конфликт разрешается в пользу объекта с наивысшим приоритетом. С этой целью каждый объект снабжается признаком готовности, который позволяет выбрать информацию из собственного объекта, но запрещает выборку для объектов с меньшим приоритетом (большими номерами).

1.4 Кеш память.

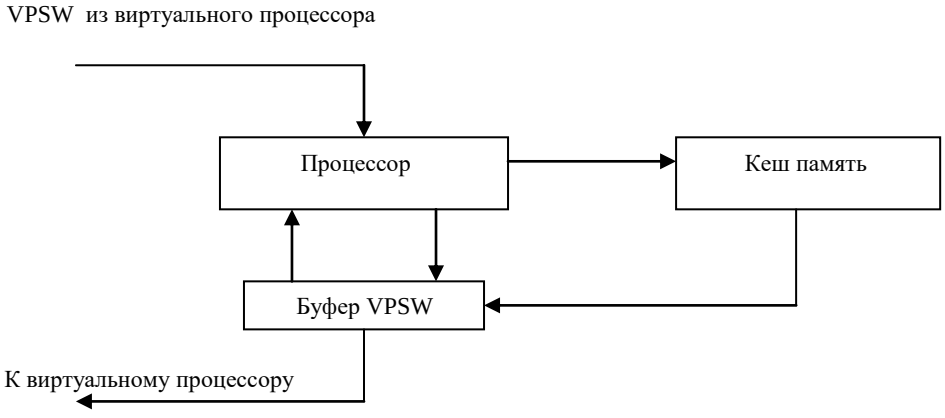
Авторы многопроцессорной архитектуры Tera MTA [3] утверждают, что для подобного рода систем кеш память не нужна, поскольку можно обойтись переключениями между трассами (тредами), естественно, при их достаточном количестве. Однако на наш взгляд кеш память полезна по крайней мере по трем причинам.

Во-первых, наличие кеш памяти позволяет отделить процессоры от основной памяти, например, используя кольцевую схему, объединяющую сегменты основной памяти и выходы запросов к основной памяти от кеш. Это позволяет уменьшить число связей между кристаллом и памятью до обычной величины. что дает возможность реализовать процессоры и кеш в одном кристалле. Ниже на схеме овал – это кольцевой буфер, а пунктирная линия – это граница, отделяющая кристалл от основной памяти.



Во-вторых, расщепленная на чередующиеся сегменты память не спасает от очередей в случае обращения большого количества процессоров к одной и той же ячейке памяти, например, константе или глобальной переменной. В этом случае только быстрое рассасывание очереди позволяет избежать простоев процессоров. Именно это и достигается при помощи кеш памяти.

В-третьих, Кеш память позволяет уменьшить количество переписей VPSW из физического процессора в виртуальный. Это может быть достигнуто введением буфера VPSW. Действительно, после прерывания, связанного с обращением к кеш, процессор устанавливает VPSW на буфер и переходит к вычислению команд новой трассы. Если за время до прерывания этой новой трассы кеш память успевает выполнить заказ, то результат корректирует буфер, который и передается своевременно физическому процессору, минуя виртуальный.



Заключение.

Автор считает наиболее существенными следующие элементы предлагаемой архитектуры.

1.Виртуальные процессоры позволяют единообразным способом реализовать все прерывания, возникающие в процессе выполнения трассы, и упростить их обработку.

2. Аппаратура синхронизации берет на себя большую часть работ по синхронизации параллельного выполнения трасс.

3. Программа не зависит от конфигурации компьютера, например, количества виртуальных или физических процессоров.

4. Достаточно легко для трансляторов с языков высокого уровня генерировать команды разбиения программы на трассы.

Приложение 1. Индикаторы способа адресации.

a - признак автоматического базирования. Если это адрес в команде переходов или ветвлений, то в качестве базы берется регистр A, если адрес данных, то регистр B. Исполнительный адрес вычисляется следующим образом (База)+(Регистр)+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 Регистр используется как первый операнд, а исполнительный адрес вычисляется как (База)+Адрес.

c – исполнительный адрес вычисляется как (Регистр)+Адрес

d – косвенная адресация в командах, использующих дескриптор. Адрес самого дескриптора задается аналогично **a.** Дескриптор – три элемента памяти. В первом элементе задается адрес трассы, которой передаются данные, во втором – база той части программы, которой трасса 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 Регистр, второй – в адресной части команды. Результат остается в первом операнде. Команды с таким индикатором не изменяют регистр **S** как регистр результата. Однако, он может использоваться в качестве операнда.

Приложение 2. 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.

1. Распараллеливающие вычисления. Команды этой группы позволяют распараллеливать вычисления путем создания новой трассы

fork - разветвление Команда используется для создания трассы по указанному в команде адресу.

cfork - разветвление по условию.

gnr – генератор значений. Команда используется для параллельного выполнения команд тела цикла путем создания трасс с параметрами, изменяющимися в интервале – <значение первого операнда>... 0. Команда выполняется следующим образом. Если значение больше или равно нулю, то создается новая трасса, адрес которой в программе на единицу больше адреса команды gnr, и значение уменьшается на единицу. Когда значение становится отрицательным, то выполняется команда по адресу, заданному во втором операнде команды gnr.

3. Передающие управление. Команды выполняются внутри одной трассы.

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.

4. 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.

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

rdv – первому операнду присваивается значение второго.

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

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.

wrdr - запись с установкой тега wr.

wrtg - изменение значения тега на wr.

xchg – первый операнд меняется значением со вторым.

lbm - запись метки трассы.

6. Команды, прерывающие выполнение команд трассы.

stop - прекращение выполнения трассы.

brk - призыв к завершению всех трасс, помеченных меткой в команде lbm.

cut - завершение выполнения помеченной трассы после выполнения команды brk в другой трассе, но с той же меткой.

7. Пустая команда.

emp – не выполняет никаких действий.

Литература.

1. В.М. Михелев
   ПЕДАНТ Архитектура многопроцессорного комплекса,
   препринт ИПМ им. Келдыша РАН, 2001, N 81.

2. В.М. Михелев
   ВИКТОР Архитектура многопроцессорного комплекса,
   препринт ИПМ им. Келдыша РАН, 2002, N 74.

3. R. Alverson, D. Callaham, D. Cummings, B. Koblenz, A Portenfueld, B. Smith.
   The Tera computer system.
   Proceedings of the ACM International Conference on Supercomputing, June 1990.

Оглавление