Краткая характеристика архитектуры Itanium

1 Связки команд и типы функциональных исполнительных устройств

2 Регистровая модель (программистская структура)

3 Краткая характеристика системы команд

4 Поддержка программной конвейеризации циклов

- **Знать:** формат команды и связки команд, основные типы функциональных устройств, назначение и формат основных типов регистров, предикатные регистры, регистр маркера текущего окна, особенности системы команд, спекуляция по управлению и данным; поддержка программной конвейеризации циклов: роль предикатных регистров, вращение регистров; структура процессора.

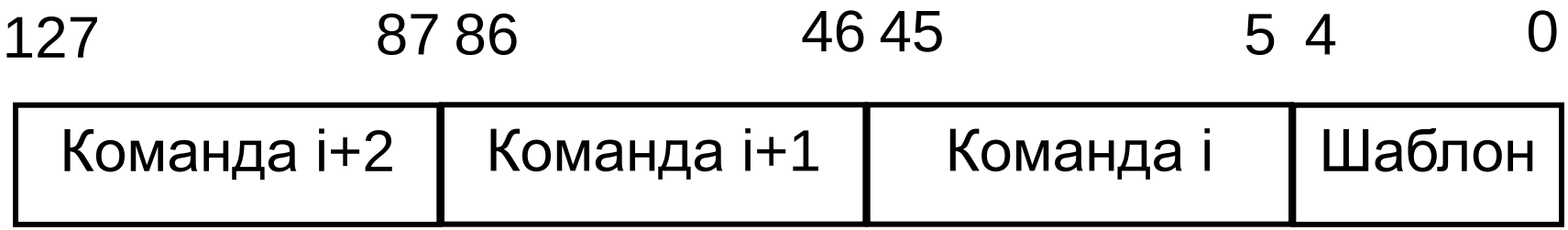
Литература:

- Цилькер Б.Я., Орлов С.А. Организация ЭВМ и систем. Учебник для вузов. – СПб.: Питер, 2004. – 668 с. (с. 413-446)

1 Связки команд и типы функциональных исполнительных устройств

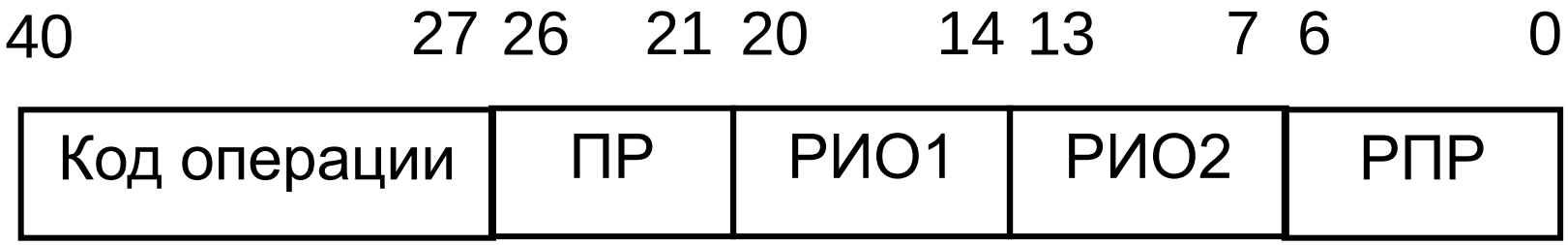
- Наиболее кардинальным нововведением IA-64 является явный параллелизм команд (EPIC), привносящих в IA-64 некоторые элементы, напоминающие архитектуру "сверхбольшого командного слова" (VLIW).
- В обоих архитектурах явный параллелизм представлен уже на уровне команд, управляющих одновременной работой функциональных исполнительных устройств (ФИУ).
- Соответствующие "широкие команды" HP/Intel назвали связками (bundle).
- Связка имеет длину 128 разрядов (рисунок 1).

1.1 Формат связки команд IA-64



- Связка включает 3 поля – слота для команд длиной 41 разряд каждая, и 5-разрядное поле шаблона.
- Предполагается, что команды связки могут выполняться параллельно разными ФИУ.
- Возможные взаимозависимости, препятствующие параллельному выполнению команд связки, отражаются в поле шаблона.
- Не утверждается, впрочем, что параллельно не могут выполняться и команды разных связок.

1.2 Формат команды IA-64



- Использованы следующие обозначения:

  ПР – предикатный регистр;

  РИО1 – регистр-источник первого операнда;

  РИО2 – регистр-источник второго операнда;

  РПР – регистр-приемник результата.

- В общем случае команды одного типа могут выполняться в более чем одном типе ФИУ (таблица 1).

1.3 Типы команд и исполнительных устройств

Тип команд	Тип исполнительного устройства	Описание команд
A	I или M	Целочисленные арифметические и логические
I	I	Целочисленные неарифметические
M	M	Обращение к памяти
F	F	С плавающей запятой
B	B	Переходы
L+X	I	Расширенные

1.4 Варианты составления связок

- В шаблоне указываются зависимости между командами (можно ли с командой К0 запустить параллельно К1, или же К1 должна выполняться только после К0), а также между другими связками (можно ли с командой К2 из связки С0 запустить параллельно команду К3 из связки С1).
- Имеются следующие варианты составления связки из трех команд:

  К0||К1||К2 – все команды исполняются параллельно;

  К0&К1||К2 – сначала К0, затем исполняются параллельно К1 и К2;

  К0||К1&К2 – параллельно обрабатываются К0 и К1, после них – К2;

  К0&К1&К2 – команды выполняются в последовательности К0, К1, К2.

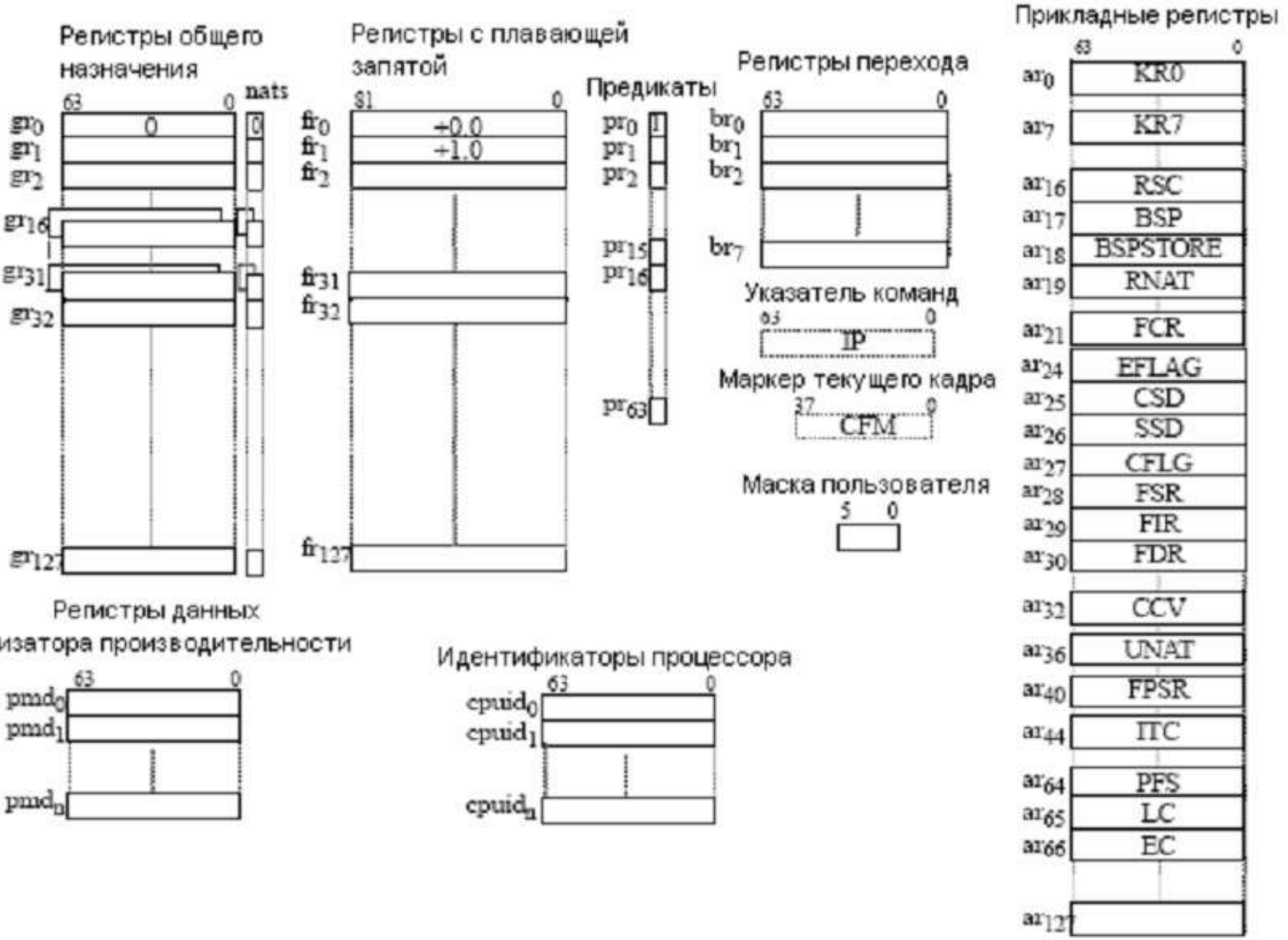
2 Регистровая модель (программистская структура)

2.1 Общие регистры

В регистровые файлы IA-64 входят:

- 128 регистров общего назначения GR;
- 128 регистров с плавающей запятой FR;
- 64 регистра предикатов PR;
- 8 регистров перехода BR;
- 128 прикладных регистров AR;
- не менее 4 регистров идентификатора процессора CPUID;
- регистр указателя команд IP, указывающий на адрес связки, содержащей исполняемую команду;
- регистр маркера текущего окна CFM, описывающий окно стека регистров и др.

Общие регистры



2.1.1 Регистры общего назначения

- Являются основным ресурсом, используемым для целочисленных и мультимедиа вычислений.
- Нумеруются от GR0 до GR127 и доступны всем программам со всеми уровнями привилегий.
- Каждый регистр имеет кроме 64 разрядов для хранения данных еще один дополнительный разряд для бита NaT (Not a Thing (не факт)), который используется для прослеживания отложенных спекулятивных исключений.
- Регистры GR8 – GR31 в режиме IA-32 используются также под целочисленные регистры и регистры селекторов и дескрипторов сегментов IA-32.
- Регистры GR0 – GR31 называются статическими (GR0 всегда содержит 0), а GR32 – GR127 – стекируемыми.
- Статические регистры "видны" всем программам.
- Стекируемые регистры становятся доступными в программной единице через окно стека регистров, в них может быть организовано вращение регистров.

2.1.2 Регистры с плавающей запятой

- 82-разрядные регистры с плавающей запятой FR0-FR127 также подразделяются на статические (FR0-FR31, причем, всегда FR0=0.0, FR1=1.0) и вращаемые (FR32-FR127).
- Регистры FR8-FR31 в режиме IA-32 содержат числа с плавающей запятой и мультимедийные данные.

2.1.3 Регистры предикатов

- Регистры предикатов (PR0-PR63) являются одноразрядными; в них помещаются результаты команд сравнения.
- Обычно эти команды устанавливают сразу два регистра PR в зависимости от условия – соответственно истинность условия и его отрицания. Такая избыточность обеспечивает дополнительную гибкость.

2.1.4 Регистры переходов

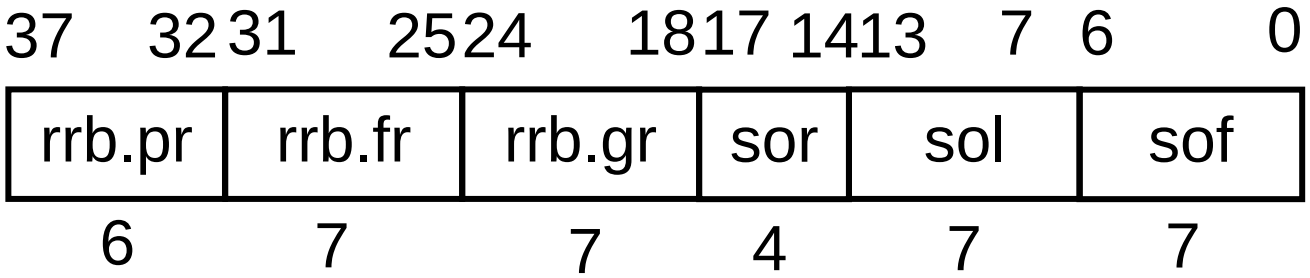
- 64-разрядные регистры переходов BR0-BR7 применяются для указания адреса перехода в соответствующих командах перехода (если адрес перехода не кодируется в команде явно).

2.1.5 Регистр указателя команды

- Регистр указателя команды (IP) содержит адрес связки, в которой находится выполняемая команда.
- Из регистра можно непосредственно считывать информацию, используя команду mov ip.
- В регистр нельзя непосредственно записывать, но его содержимое увеличивается после выполнения команд и в регистр может быть занесено новое значение при переходах.
- Поскольку связки команд занимают 16 байт и выровнены по 16-байтной границе, то по крайней мере 4 значащих бита в программном счетчике всегда равны нулю.
- При выполнении команд процессора с IA-32 архитектурой программный счетчик содержит 32-разрядный виртуальный линейный адрес текущей команды, дополненный нулями.

2.1.6 Регистр маркера текущего окна (CFM)

- Каждое окно (фрейм) стека регистров ассоциируется с маркером окна. Регистр маркера текущего окна содержит описание текущего состояния окна регистров стека.



Формат маркера окна в IA-64

Поле	Описание
sof	Размер окна стека
sol	Размер локальной области окна стека
sor	Размер вращаемой области окна стека (чило вращаемых регитсров определяется как 8*sor)
rrb.gr	База переименования регистров для общих регистров
rrb.fr	База переименования регистров для регистров с плавающей запятой
rrb.pr	База переименования регистров для предикатных регистров

2.1.7 Прикладные регистры

- Это специализированные в основном 64-разрядные регистры.
- Регистры AR0-AR127 применяются в IA-64 и IA-32.
- Регистры AR0-AR7 называются регистрами ядра;
- Запись в регистры ядра привилегирована, но они доступны для чтения в любом приложении и используются для передачи приложению сообщений от операционной системы.
- Ряд AR-регистров являются фактически регистрами IA-32 (дескриптор сегмента кодов, дескриптор сегмента стека и др.)

- Среди прикладных регистров следует также указать:
  - AR16 (RSC) – регистр конфигурации стека регистров, используемый для управления работой "машины" стека регистров IA-64 (RSE);
  - AR17 (BSP) – регистр, в котором находится адрес памяти, где сохраняется положение GR32 в текущем окне стека;
  - AR40 (FPSR) – регистр состояния для команд с плавающей запятой IA-64;
  - AR44 (ITC) – интервальный таймер;
  - AR64 (PFS) – регистр предыдущего состояния функции, куда автоматически копируются некоторые другие регистры при вызове подпрограмм;
  - AR65 (LC) – регистр, используемый для организации циклов со счетчиком;
  - AR66 (EC) – 6-разрядный регистр эпилога.

2.1.8 Другие регистры

- ***Регистры контроля (мониторинга) обрабатываемых данных (PMDR).*** Используются на привилегированных уровнях.
- ***Регистр маски пользователя (UMR).*** Регистр является элементом регистра состояния процессора и доступен программам приложений. Маска пользователя управляет выравниванием данных в памяти, порядком расположения байтов в памяти и др.
- ***Регистры идентификации процессора (CPUIDR).*** Регистры CPUIDR являются 64-рязрядными. В регистрах CPUIDR0 и CPUIDR1 находится информация о производителе, в регистре 2 – серийный номер процессора, а в регистре 3 задается тип процессора (семейство, модель, версия архитектуры и т.п.) и число регистров CPUID. Разряды регистра 4 указывают на поддержку конкретных особенностей IA-64, реализованных в данном процессоре.
- ***Регистры таблицы адресов опережающей загрузки (ALATR).***

2.2 Системные регистры

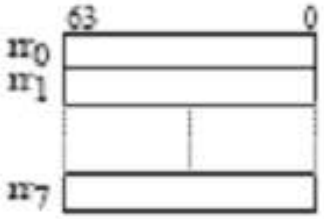
- Регистр состояния процессора (PSR) – 64-битный регистр, который содержит информацию о текущем выполняющемся процессе.
- Управляющие регистры (CR) – несколько 64-битных регистров, которые хранят состояние процессора при прерывании, служат для управления всей системой, и указывают общие параметры процессора для прерываний и управления памятью.
- Регистры прерываний – эти регистры обеспечивают возможность маскировать внешние прерывания, читать номера векторов внешних прерываний, номера векторов программных прерываний для внутренних асинхронных событий в процессоре и внешних источников прерываний.
- Счётчики интервалов времени – 64-битный счётчик интервалов времени служит основой для замеров производительности.
- Отладочные регистры контрольных точек – пары 64-битных регистров (один для данных, а другой для команд) (DBR, IBR).

2.2 Системные регистры (продолжение)

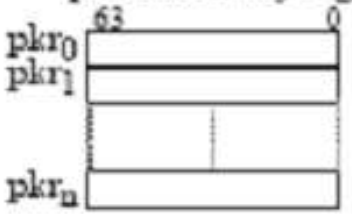
- Регистры конфигурации и данных анализатора производительности (PMC, PMD).
- Сгруппированные регистры общего назначения (GR16 – GR31) – набор из 16 сгруппированных регистров общего назначения.
- Регистры регионов (RR) – восемь 64-битных регистров для указания идентификаторов и предпочитаемых размеров страниц для адресных пространств.
- Регистры ключей защиты (PKR) – как минимум шестнадцать 64-битных регистров содержащих ключи защиты и разрешения чтения, записи и выполнения для доменов защиты виртуальной памяти;
- Буфер ассоциативной трансляции (TLB) – хранит последние использованные отображения виртуальных адресов в физические. Состоит из двух буферов: команд – ITLB и данных – DTLB. В свою очередь каждый из выделенных буферов делится на две секции: регистры трансляции (TR) и кэш-транслятор (CT).

Системные регистры

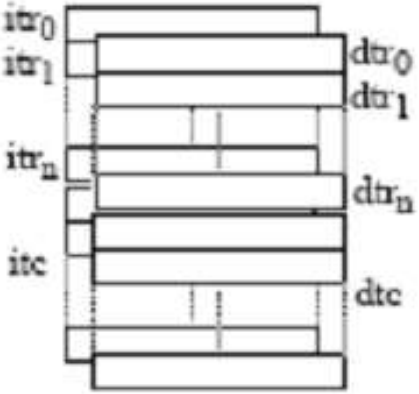
Регистры регионов



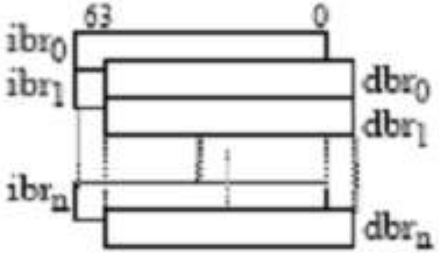
protection key regs



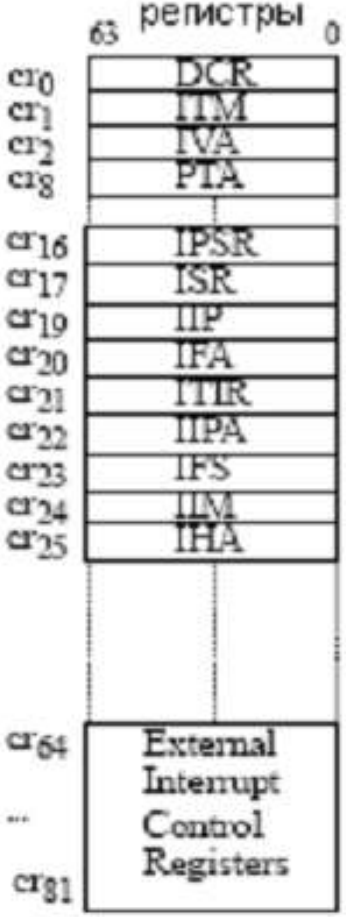
Буффер ассациативной трансляции



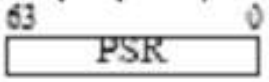
Отладочные регистры точек останова



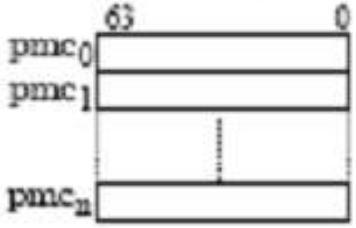
Управляющие регистры



Регистр состояния процессора



Регистры конфигурации анализатора производительности



3. Краткая характеристика системы команд

Все рассматриваемые команды можно подразделить на следующие группы:

- команды работы со стеком регистров (например, alloc);
- целочисленные команды;
- команды сравнения и работы с предикатами;
- команды доступа в память;
- команды перехода;
- мультимедийные команды;
- команды пересылок между регистрами;
- "разные" (операции над строками и подсчет числа единиц в слове);
- команды работы с данными в форме с плавающей запятой.

3.1. Целочисленные команды

- Целочисленные команды IA-64 включают арифметические операции (add, sub и др.), логические операции (and, or, xor и др.), операции над битами и сдвиги, а также 32-разрядные операции.
- Большинство этих команд трехадресные, а их аргументы находятся в регистрах, однако встречается и литеральное представление аргументов.
- Имеются также модификации команд add и sub, которые являются четырехадресными: в них к сумме/разности регистров прибавляется/вычитается 1.
- Следует отметить, что команда умножения целых чисел в регистрах GR отсутствует; для перемножения необходима пересылка целых в регистры FR и применение операции умножения, выполняемой в ФИУ вещественного типа. Некоторые специалисты считают это "наименее удачной" чертой системы команд IA-64.

3.2. Команды сравнения и работа с предикатами

- Типичными примерами команд этой группы являются:
  - команда cmp сравнивает два регистра GR (или регистр GR и литерал), проверяя выполнение одного из 10 возможных условий (больше, меньше или равно и т.п.);
  - команда tbit тестирует заданный бит;
  - команда fcmp сравнивает два числа с плавающей запятой.
- Однако результатом сравнения является не единственный код условия, что типично для обычных процессоров. Логический результат сравнения (1 – истина, 0 – ложь) записывается обычно в пару предикатных регистров (во второй пишется отрицание первого).
- Эти значения предикатных регистров используются затем не только в командах условного перехода, как в обычных микропроцессорах. Почти все команды IA-64 выполнимы "под предикатами".

Пример команд под предикатами

Фрагмент программы (на Фортране):

```
IF(IEQJ) THEN
K=K+1
ELSE L=L+1
ENDIF
```

Фрагмент программы на ассемблере:

```
cmp.eq p3,p4=r4,r5;;
(p3) add   r6=r6,r0,1
(p4) add   r7=r7,r0,1
```

- Предполагается, что значения I, J, K, L уже находятся в регистрах r4, r5, r6, r7 соответственно (так обозначаются регистры RG в ассемблере IA-64).
- Здесь команды сложения использованы в четырехадресной форме: они помещают в регистр результат (r6 и r7 соответственно), старое значение этого регистра плюс 1 (формально еще плюс значение в r0, которое равно нулю).
- Команды add используются с так называемыми квалифицирующими предикатами p3 и p4. Если значения I и J совпадают, то в предикатных регистрах формируются следующие значения: (p3)=1, а (p4)=0. Тогда первая команда add выполняется, а вторая – нет.

3.3. Команды доступа в память

- Прежде всего это команды загрузки регистров и записи из них в оперативную память.
- Команда ld загружает в GR 1-, 2-,4- и 8-байтные целочисленные величины.
- Аналогично ldf загружает в FR числа с плавающей запятой размером 4, 8, 10 байт, а также пары 4-байтных чисел.
- В этих командах можно указать также тонкие особенности работы с оперативной памятью и кэшем. Имеются и специальные команды работы с кэшем.
- Принципиальной является возможность кодирования указанных команд загрузки в специальных спекулятивных формах. Различают загрузку спекулятивную по управлению и спекулятивную по данным.

Спекуляция по управлению

- Означает возможность заранее выполнить команды, расположенные за командой условного перехода, до перехода на соответствующую ветвь программы.
- Наличие большого числа ресурсов процессора позволяет запускать на выполнение команды, которые начнут выполняться одновременно с уже начавшими выполняться другими командами.
- Однако позднее может выясниться, что эти спекулятивно выполненные команды оказались выполненными напрасно (перехода не было) и нужно произвести "откат".
- Кроме обычных не спекулятивных команд (ld, ldf, …) в IA-64 имеются их спекулятивные модификации (ld.s, ldf.s, …).
- В точке программы, где надо использовать результат спекулятивного выполнения, применяют спекулятивную команду chk.s, проверяющую признак отложенного прерывания. При прерывании команда передает управление по указанному в ней адресу, там программист должен расположить коды обработки ситуации.

Спекуляция по данным

- Данный тип спекулятивного выполнения команд может иметь место, когда вслед за записью в память идет команда загрузки регистра, и невозможно заранее определить, не будут ли перекрываться в памяти используемые этими командами данные.
- В IA-64 имеются спекулятивные команды загрузки (ld.a, ldf.a,…), которые называются "усовершенствованными" командами загрузки.
- Аналогично взаимозависимости между командами по управлению, "расшиваемой" применением спекулятивных команд с "постфиксом" .s, модифицированные команды загрузки вместе с соответствующей командой проверки chk.a (аналог команды chk.s) позволяют исключить задержки выполнения взаимозависимости по данным.

3.4. Команды перехода

- В этих командах адрес перехода всегда выравнивается на границу связки, т.е. управление передается на ее слот 0.
- Имеется команда перехода относительно счетчика команд, в которой явно кодируется 21-разрядное смещение. Эти переходы осуществимы в пределах +/- 16 Мбайт относительно счетчика.
- В непрямых переходах адрес перехода задается в регистре BR.
- Обычный условный переход br.cond, или просто br, использует значение кодируемого в команде предикатного регистра PR для определения истинности условия.
- Указав в команде регистр PR0, в котором всегда находится 1, можно получить безусловный переход. Регистр PR0 кодируется также в командах вызова процедур/возврата (br.call/br.ret).

3.5. Организация циклов

- Имеется 5 типов команд перехода, применяемых для организации циклов.
- Команда br.loop применяется для организации циклов со счетчиком, в которой адрес перехода кодируется относительно IP. В команде используется регистр LC: если он не равен 0, то его содержимое уменьшается на 1, и выполняется переход; если LC=0, то перехода не будет.
- В расширении кода операции перехода можно закодировать подсказку для процессора о стратегии динамического или статического предсказания этого перехода.

3.6. Команды операций с плавающей запятой

- Формат регистров FR включает 64-разрядную мантиссу, 17-разрядный порядок и 1 бит под знак числа. На уровне подпрограмм поддерживается четверная точность.
- В 64-разрядном регистре FPSR указываются признаки деления на ноль, переполнение порядка, исчезновение порядка, потери значимости, формат данных и другая информация о состоянии.
- FP-команды загрузки имеют модификации, соответствующие всем аппаратно поддерживаемым типам данных, которые в ассемблере задаются последним символом мнемокода (ldfs для SP, ldfd для DP и т.д.).
- Арифметические команды включают операции типа "умножить и сложить", и "умножить и вычесть", команды вычисления максимума/минимума, а также команды расчета обратной величины и обратного квадратного корня.
- Реализация двух последних команд упрощает работу с конвейерами.

3.7. Пример работы с плавающей запятой

Фрагмент программы (на Фортране):

```
DO I=1,N
C(I)=A(I)+B(I)
ENDDO
```

Фрагмент программы на ассемблере:

```
Lb1: ldfd f6=[r6],8     //Загрузка в f6 A(I)
     ldfd f7=[r7],8;;   //Загрузка в f7 B(I)
     fadd f8=f6,f7;;    //Сложение f6 и f7
     stfd [r8]=f8,8     //Запись C(I)
     br.cloop Lb1;;     //Переход на метку
```

- В ассемблерном представлении приведено только собственно тело цикла.
- Предполагается, что в регистре r6 находится начальный адрес массива A, а в r7 – массива B, а в r8 – массива C.
- После выполнения каждой команды ldfd и команды stfd содержимое регистров r6-r8 увеличивается на 8 (размер элемента массива в байтах), что указывается в последнем аргументе этих команд.
- Команда fadd складывает содержимое регистров f6 и f7, помещая результат в регистр f8.
- Команда br.cloop обеспечивает переход на начало тела цикла.

4. Поддержка программной конвейеризации циклов

- Эти циклы аналогичны обычным аппаратным конвейерам.
- В таком цикле также имеется три фазы.
- *В фазе заполнения конвейера* (пролог) новая итерация цикла запускается на выполнение на каждой стадии.
- *В фазе ядра* на каждой стадии запускается одна итерация, и одна итерация завершается (конвейер заполнен).
- *В фазе эпилога* новых итераций не запускается, а завершается выполнение ранее запущенных итераций (см. пример).

4.1. Пример программы конвейеризации цикла

Фрагмент программы на Фортране:

```
DO I=1,N
IND(I)=JND(I)+K
ENDDO
```

Фрагмент программы на ассемблере:

```
Lb1: ld8 r8=[r5],8;;   //Загрузка JND(I)
     add r9=r8,r7;;    //r9=r8+r7
     st8 [r6]=r9,8     //Запись IND(I)
     br.cloop Lb1;;    //Переход по
                         счетчику
```

Программная конвейеризация цикла:

```
     mov lc=99                  //Установка LC
     mov ec=4                   //Установка EC
     mov pr.rot=1<<16           //Установка PR
Lb1: (p16) ld8 r32=[r5],8;;     //Загрузка JND(I)
     (p18) add r35=r34,r7;;     //r35=r34+r7
     (p19) st8 [r6]=r36,8       //Запись IND(I)
     br.ctop Lb1;;              //Переход по счетчику
```

4.2. Раскрутка цикла

- Число тактов T между запуском последовательных итераций цикла называется интервалом инициации.
- Далее рассматривается только случай постоянного числа тактов. В простейшем случае T равно 1, очередная итерация цикла может запускаться на каждом такте.
- Каждой стадии цикла должен быть выделен PR-регистр из области вращения, определяющий, следует ли выполнять команды данной стадии.
- В простейшем случае стадия цикла включает одну команду.
- При использовании раскрутки (unrolling) циклов, которую часто необходимо применять для эффективного использования ресурсов процессора, каждая стадия включает несколько однотипных команд (например, для четных и нечетных итераций – при двукратной раскрутке). Тогда для каждой команды стадии применяется один и тот же PR-регистр.

4.3. Поддерживаемые циклы и предикатные регистры

- Программная конвейеризация циклов поддерживается как для исходных циклов со счетчиком, использующих команду br.cloop, так и для циклов while.
- Для циклов со счетчиком на первой стадии цикла должен применяться регистр PR16, а для циклов while – любой PR-регистр из области вращения.
- Предикаты последующих стадий должны иметь более высокие номера PR. Инициализация предикативных стадий возлагается на программиста.
- Применение PR-регистров, вращаемых GR и FR и специальных команд перехода позволяет не увеличивать размер кода цикла, что характерно для оптимизации циклов в RISC-процессорах.

4.4. Вращение регистров

- Вращение на одну регистровую позицию регистров GR и FR. осуществляется аппаратно при выполнении специальной команды перехода (br.ctop для циклов со счетчиком при расположении команды перехода в конце тела цикла).
- При выполнении такой команды вращение регистров осуществляется путем их переименования благодаря увеличению на 1 базовых значений в регистре маркера текущего окна CFM – соответственно полей CFM.rrb.gr, CFM.rrb.fr и CFM.rrb.pr.
- Вообще изменение полей rrb, приводящее к переименованию регистров, происходит при выполнении команд очистки rrb (clrrb, clrrb.pr) и переходов типа вызова процедур/возврата (br.call/br.ret) и специальных переходов, в том числе br.ctop.
- Файлы вращаемых регистров с точки зрения их переименования зациклены: регистр с максимальным номером после вращения переходит в регистр с минимальным номером из области вращения. Так, PR63 после вращения становится PR16.

4.5. Завершение цикла

- При организации программно конвейеризированного цикла со счетчиком в регистр LC помешается значение N-1 (N число итераций цикла), а в регистр EC – число стадий в теле цикла.
- Пока LC больше нуля, команда br.ctop продолжает выполнение цикла, уменьшая LC на 1 и вращая регистры на 1 путем увеличения rrb.
- Команда, при выполнении которой принимается решение о продолжении цикла, обеспечивает установку в 1 регистра PR63, который после вращения становится PR16.
- Когда LC становится равным нулю, начинается фаза эпилога. В ней br.ctop будет продолжать цикл, уменьшая EC на 1 – до тех пор, пока в регистре EC не окажется 0. Тогда br.ctop завершит цикл, передав управление на следующую команду.

4.6. Пример исходной программы

```
Lb1: ld8 r8=[r5],8;;   //Загрузка JND(I)
     add r9=r8,r7;;    //r9=r8+r7
     st8 [r6]=r9,8;;   //Запись IND(I)
     br.cloop Lb1;;    //Переход по счетчику
```

- Все целые величины – 64-разрядные, в регистре GR5 находится адрес начала массива IND, в GR6 – адрес начала массива JND, величина K – загружена в регистр GR7.
- Первая команда ld8 загружает 8 байт из ячейки памяти, адрес которой задан в регистре GR5, в регистр GR8. В конце ее выполнения к содержимому регистра GR5 прибавляется 8.
- Команда add складывает содержимое регистров GR8 и GR7, помещая результат в регистр GR9.
- Команда st8 записывает содержимое этого регистра в ячейку памяти, адрес которой находится в GR6, и в конце выполнения увеличивает его на 8.

4.7. Пример конвейеризированной программы

- Предполагается, что выполнение команды ld8 занимает два такта, а остальные команды тела цикла (без br.cloop) выполняются за один такт.
- Тогда тело цикла будет состоять из четырех стадий, вторая из которых – пустая (ожидание завершения выполнения команды ld8 во втором такте).

```
        mov lc=99                   //Установка LC
        mov ec=4                    //Установка EC
        mov pr.rot=1<<16            //Установка PR
Lb1:  (p16) ld8 r32=[r5],8;;        //Загрузка JND(I)
      (p18) add r35=r34,r7          //r35=r34+r7
      (p19) st8 [r6]=r36,8;;              //Запись IND(I)
      br.ctop Lb1;;                 //Переход по счетчику
```

4.8. Описание конвейеризированной программы

- Конвейерный аналог программы из примера использует команду br.ctop.
- Соответственно первой стадии (команда ld8) выделяется предикатный регистр PR16, второй (add) – PR18, четвертой (st8) – PR19.
- Вместо статических регистров GR0-GR9, используются вращаемые регистры, начиная с регистра GR32.
- Перед началом выполнения цикла в регистр LC загружается значение N-1 (99), а в регистр EC – 4.
- Кроме того, производится установка вращаемых PR-регистров, что делается сразу для всех предикатных регистров командой mov pr.rot.

Заполнение конвейера (пролог)

Номер такта	Стадия конвейера			
	1	2	3	4
1	ld8 RG32=[r5],8;;			
2	ld8 r32=[r5],8;;	ld8 RG33=[r5],8;;		
3	ld8 r32=[r5],8;;	ld8 r33=[r5],8;;	add RG35=RG34,r7	
4	ld8 r32=[r5],8;;	ld8 r33=[r5],8;;	add r35=r34,r7	st8 [r6]=RG35,8
5	ld8 r32=[r5],8;;	ld8 r33=[r5],8;;	add r35=r34,r7	st8 [r6]=r36,8

- Вследствие вращения регистров величина JND(I), загружаемая в регистр GR32, двумя тактами позднее (по завершению ld8) уже оказывается в регистре GR34.
- Аналогично, значение IND(I)+K, помещаемое в регистр GR35, по завершению команды add (один такт) окажется в GR36.

Переименование регистров с помощью вращения

	Такт 1		Такт 2		Такт 3		Такт 4
RG124		RG125		RG126		RG127	
RG125		RG126		RG127		RG32	
RG126		RG127		RG32		RG33	
RG127		RG32		RG33		RG34	
RG32	RG32	**RG33**	RG32	**RG34**	RG32	**RG35**	RG32
RG33		RG34		RG35		RG36	
RG34		RG35		RG36		RG37	
RG35		RG36		RG37		RG38	
RG36		RG37		RG37		RG39	
RG37		RG38		RG38		RG40	

4.9. Прокрутка цикла

Номер Такта	Команды и порты				Значения регистров перед выполнением команды br.ctop					
	M	I	M	B	p16	p17	p18	p19	LC	EC
1	ld8			br.ctop	1	0	0	0	99	4
2	ld8			br.ctop	1	1	0	0	98	4
3	ld8	add		br.ctop	1	1	1	0	97	4
4	ld8	add	st8	br.ctop	1	1	1	1	96	4
...	...	...	...	...	...	...	...	...	...	...
99	ld8	add	st8	br.ctop	1	1	1	1	0	4
100		add	st8	br.ctop	0	1	1	1	0	3
101		add	st8	br.ctop	0	0	1	1	0	2
102			st8	br.ctop	0	0	0	1	0	1
...					0	0	0	0	0	0

Вследствие наличия большого числа ФИУ и соответствующих им портов, команды цикла могут выполняться одновременно, если бы работают с разными регистрами.

Это видно из таблицы, в которой не трудно рассмотреть аналогии с заполнением аппаратных конвейеров.

4.10. Пояснения к прокрутке цикла

- На тактах 1-3 происходит заполнение конвейера (пролог). Такты 4-99 относятся к фазе ядра. Такты 100-102 отвечают эпилогу.
- Если бы не было взаимозависимости, команды ld8, add и st8 могли бы работать параллельно в фазе ядра (предполагается, что имеется два порта памяти).
- Скажем, когда add начинает работу, ld8 могла бы начать новую загрузку, но уже в другой GR-регистр.
- В суперскалярных RISC-процессорах для достижения подобных целей приходится создавать отдельные коды программы для пролога и эпилога, раскручивать циклы, что приводит к увеличению длины программы. В IA-64 эти проблемы решены более эффективно.
- Нет необходимости вручную заниматься переименованием регистров, чтобы избавиться от взаимозависимости – для этого есть вращение регистров. Не нужно писать пролог и эпилог – все автоматизировано за счет применения PR-регистров и специальных команд.

Замечания

•При выполнении такой команды вращение регистров осуществляется путем их переименования благодаря уменьшению на 1 базовых значений в CFM – соответственно полей CFM.rrb.gr, CFM.rrb.fr и CFM.rrb.pr

•Вообще изменение полей rrb, приводящее к переименованию регистров, происходит при выполнении команд очистки rrb (clrrb, clrrb.pr) и переходов типа вызова процедур/возврата (br.call/br.ret) и специальных переходов, в том числе br.ctop.

•Файлы вращаемых регистров с точки зрения их переименования зациклены: регистр с максимальным номером после вращения переходит в регистр с минимальным номером из области вращения. Так, PR63 после вращения становится PR16.