Программная архитектура SPARC-V9

(Рафиков М.Р., P3217)

Список источников:

* David Weaver, Tom Germond. The SPARC Architecture Manual Version 9  
  (официальная документация к архитектуре, используемая при составлении документа)
* [ru.wikipedia.org/wiki/SPARC](https://ru.wikipedia.org/wiki/SPARC)  
  (общая информация)

*Дисклеймер! Перевод вольный и без знания контекста на момент написания*

*Также, в документ включены только те тексты и их части, которые имеют ценность для составления модели .pdsl*

SPARC (Scalable Processor ARChitecture — масштабируемая архитектура процессора) — открытая архитектура микропроцессоров на базе RISC, разработанная в 1985 году в компании Sun Microsystems.

Существует три основные версии SPARC:

1. SPARC-V7 — первая версия, разработанная в 1985 году и описывающая 32-разрядный микропроцессор
2. SPARC-V8 — вторая версия, разработанная в 1992 году
3. SPARC-V9 — третья и последняя на данный момент версия, разработанная в 1994 году. В отличие от двух предыдущих, эта версия описывает 64-разрядный микропроцессор

Популярные реализации архитектуры SPARC-V9 — UltraSPARC (Sun Microsystems), SPARC64 (Fujitsu), R1000 (МЦСТ).

Операционные системы, работающие на SPARC:

* SunOS
* Solaris
* Linux и т.д.

Обзор архитектуры SPARC-V9

Корневыми типами данных SPARC-V9 являются:

* целые числа (32 и 64 бита)
* числа с плавающей точкой (32, 64 и 128 бит)

Логически SPARC-V9 состоит из арифметико-логического устройства (АЛУ, или ALU) и математического сопроцессора (МС, или FPU), в каждом из которых существует свой набор регистров. Размер регистров АЛУ составляет 64 бита, а МС — 32, 64 и 128 бит.

Операндами инструкций могут быть один регистр, пара регистров, четыре регистра и константы.

Процессор, основанный на SPARC-V9, может работать в двух режимах: привилегированный (privileged), открывающий доступ к привилегированным инструкциям, и непривилегированный.

Инструкции в архитектуре SPARC-V9 подразделяются на следующие типы:

* доступа к памяти
* целочисленной операции
* операции с плавающей точкой
* передачи управления
* условной передачи значений между регистрами
* арифметические, логические и сдвиговые
* доступа к регистрам состояния
* управления состоянием регистров

Регистры

1. общее представление

Регистры SPARC-V9 подразделяются на регистры общего назначения и регистры управления и статуса.

К регистрам общего назначения относятся:

* целочисленые РОН (r-регистры)
* РОН для чисел с плавающей точкой (f-регистры)

К регистрам управления и статуса относятся (список приведен без перевода):

* Program Counter register (PC)
* Next Program Counter register (nPC)
* Processor State register (PSTATE)
* Trap Base Address register (TBA)
* Y register (Y)
* Processor Interrupt Level register (PIL)
* Current Window Pointer register (CWP)
* Trap Type register (TT)
* Condition Codes Register (CCR)
* Address Space Identifier register (ASI)
* Trap Level register (TL)
* Trap Program Counter register (TPC)
* Trap Next Program Counter register (TNPC)
* Trap State register (TSTATE)
* Hardware clock-tick counter register (TICK)
* Savable windows register (CANSAVE)
* Restorable windows register (CANRESTORE)
* Other windows register (OTHERWIN)
* Clean windows register (CLEANWIN)
* Window State register (WSTATE) 30 5 Registers
* Version register (VER)
* Implementation-dependent Ancillary State Registers (ASRs)
* Implementation-dependent IU Deferred-Trap Queue
* Floating-Point State Register (FSR)
* Floating-Point Registers State register (FPRS)
* Implementation-dependent Floating-Point Deferred-Trap Queue

2. регистры непривилегированного режима

Регистры, представленные в этом разделе, видны любой непривилегированной программе (обычные приложения).

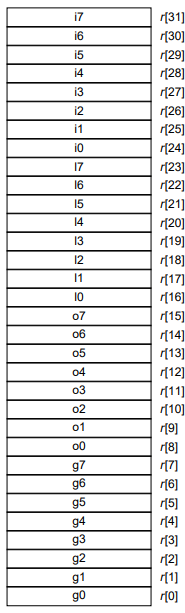
2.1.1. r-регистры общего назначения

Конкретная реализация архитектуры может содержать от 64 до 528 регистров общего назначения размером в 64 бита. Все они содержат 8 глобальных регистров, 8 альтернативных глобальных регистров и наборы по 16 регистров, число которых зависит от реализации.

2.1.1.1. глобальные r-регистры

Диапазон регистров r[0]..r[7] называется диапазоном глобальных регистров g0..g7. Этот диапазон может указывать как стандартный набор глобальных регистров, так и альтернативный. В любой момент времени к работе доступен только один такой набор (какой именно — определяется в поле AG регистра PSTATE).

Альтернативный набор глобальных регистров необходим, чтобы предоставить trap-handlers регистровое пространство, не зависящее от исполняемой программы и не имеющее какого-либо влияния на ход ее исполнения.

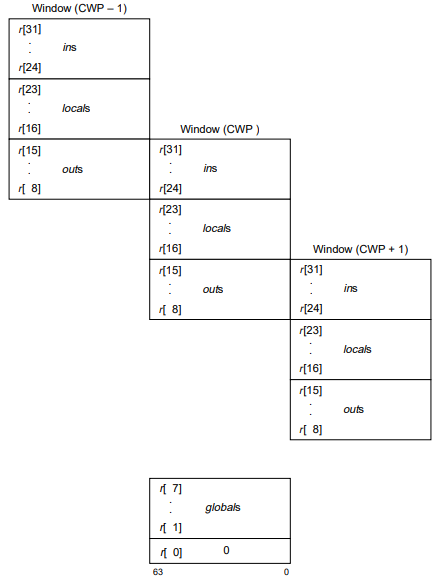


(Набор доступных к использованию регистров. Объяснение приставок: g — глобальные регистры (global), o — выходные регистры регистрового окна (out), l — локальные регистры регистрового окна (local), I — входные регистры регистрового окна (in).)

Регистр g0 всегда будет считываться как 0, а запись в него значения не будет иметь никакого эффекта на выполняемую программу.

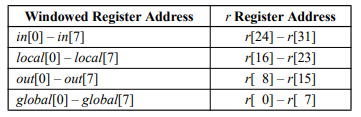
2.1.1.2. оконные r-регистры

Любая инструкция программы может обращаться к 8 глобальным регистрам и 24 оконным регистрам. Регистровое окно (с индексом CWP) включает в себя 8 входных, 8 локальных и 8 выходных регистров, указывающих на входные регистры другого, соседнего окна (с индексом CWP+1).



(Демонстрация устройства регистровых окон. Видно, что к использованию доступны 8 глобальных регистров и 24 регистра текущего окна с индексом CWP.)

Число регистровых окон NWINDOWS зависит от конкретной реализации архитектуры и находится в диапазоне от 3 до 32. Таким образом, общее количество r-регистров общего назначения равняется сумме 8 глобальных регистров, 8 альтернативных глобальных регистров и 16 регистрам окна, помноженным на количество окон NWINDOWS. Так как число окон находится в диапазоне от 3 до 32, общее количество r-регистров архитектуры может так же находится в диапазоне от 64 до 528.

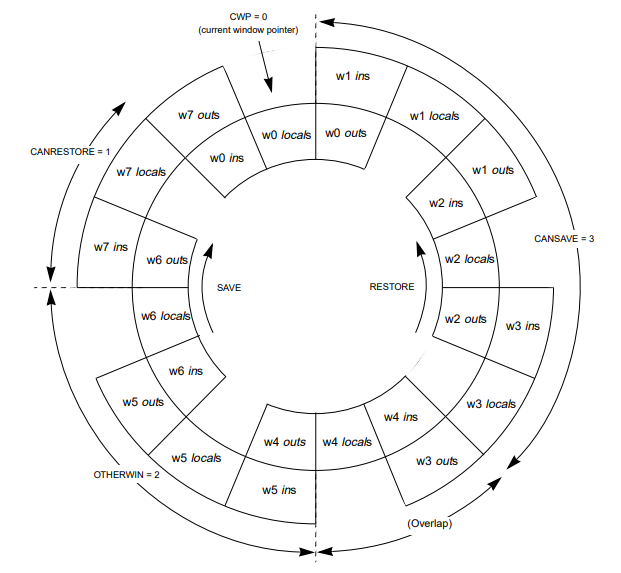


(Оконная адресация регистров.)

Номер текущего используемого окна указывается в регистре CWP. При выполнении инструкции RESTORE значение CWP декрементируется, а при выполнении инструкции SAVE, наоборот, инкрементируется.

2.1.1.3. пересечения окон

Так как арифметика регистра CWP происходит в пределах значения NWINDOWS, окно под наибольшим возможным порядковым номером, т.е. под номером NWINDOWS – 1, пересекается с окном под номером 0. Таким образом, выходные регистры этого окна NWINDOWS – 1 являются в то же время входными для окна 0.



(Демонстрация кругового логического расположения регистровых окон и операций над окнами.)

2.1.2. r-регистры особого назначения

Два регистра из набора r-регистров имеют особое назначение. Так, регистр r[0] всегда содержит ноль (о чем было сказано ранее), а регистр r[15] используется для хранения адреса выполненной инструкции CALL.

2.1.3. регистры управления/статуса блока целочисленной арифметики (Integer Unit)

Непривилегированные регистры управления/статуса блока целочисленной арифметики включают в себя регистры PC (Program Counter) и nPC (next Program Counter), регистр Y 32-битного умножения/деления и регистры ASRs (Ancillary State Registers), опциональные и зависящие от конкретной реализации архитектуры.

2.1.3.1. cчетчики операций (PC и nPC)

Регистр PC содержит в себе адрес текущей выполняемой инструкции, а nPC — адрес следующей (если не случился Trap). Два бита с наименьшими индексами обоих регистров всегда содержат нули.

TODO

Регистр PC напрямую используется инструкциями CALL, Bicc, BPcc, BPr, FBfcc, FBPfcc, JMPL и RETURN. Значение регистра считывается инструкцией RDPC.

2.1.3.2. регистр Y 32-битного умножения/деления



В SPARC-V9 регистр Y присутствует только в целях сохранения обратной совместимости с более ранними версиями и его использование не рекомендуется. Команды, использующие этот регистр, рекомендуется так же не использовать (SMUL, SMULcc, UMUL, UMULcc, MULScc, SDIV, SDIVcc, UDIV, UDIVcc, RDY и WRY).

TODO

2.1.3.3. вспомогательные регистры состояния (Ancillary State Registers, ASR)

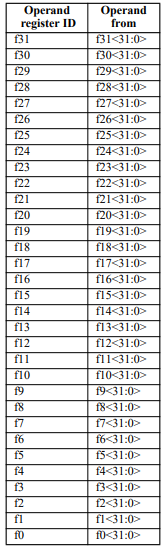
SPARC-V9 предоставляет набор опциональных ASR. Доступ к этим регистрам может быть как привилегированным, так и непривилегированным. Подробнее в части 3.10.

2.1.4. f-регистры

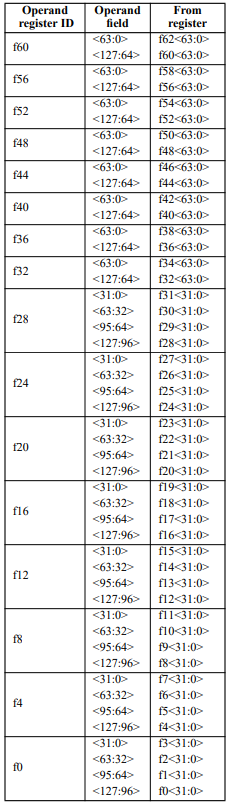
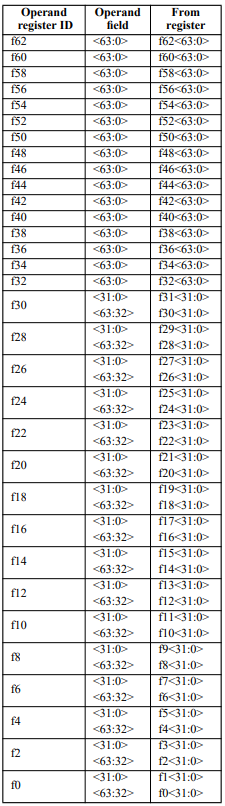
Математический сопроцессор (МС, или FPU) включает в себя несколько наборов регистров для чисел с плавающей точкой:

* 32 f-регистра одинарной точности (32 бита), нумерованные f[0], f[1]..f[31]
* 32 f-регистра двойной точности (64 бита), нумерованные f[0], f[2]..f[62]
* 16 f-регистра четверной точности (128 бит), нумерованные f[0], f[4]..f[60]

В отличие от оконных r-регистров f-регистры доступны выполняющейся программе всегда. f-регистры организованы так, что некоторые из них пересекаются, и потому являются псевдонимизированными (aliased). F-регистры считываются и записываются командой FPop, а также инструкциями нецелочисленных загрузки и сохранения. Демонстрация способа хранения f-регистров приведена на следующих фигурах:



(f-регистры одинарной точности)



(f-регистры двойной и четверной точности)

2.1.4.1.

TODO

3. регистры привилегированного режима

Регистры, представленные в этом разделе, видны только программам, запущенным в привилегированном режиме, то есть тогда, когда PSTATE.PRIV = 1. Привилегированные регистры доступны для записи инструкции WRPR и для чтения RDPR.

3.1. регистр состояния процессора (Processor State Register, PSTATE)



(поля регистра PSTATE)

Уникальный регистр PSTATE содержит информацию о текущем состоянии процессора. Запись значений в этот регистр мгновенна, поэтому состояние этого регистра обновляются сразу во время выполнения текущей инструкции.

3.1.1. PSTATE\_impldep (PID1, PID0)

Наличие и семантика полей PID1 и PID0 зависят от конкретной реализации архитектуры. Программы, предназначенные для запуска на многих возможных реализациях, должны записывать в эти поля только их предыдущие значения или нули.

3.1.2. PSTATE\_current\_little\_endian (CLE)

Чтение инструкций из памяти всегда производится от старшего к младешму (big-endian). Вид чтения данных с использованием ASI зависит от поля CLE:

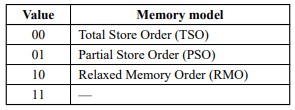
* Если CLE = 1, используется little-endian
* Если CLE = 0, используется big-endian

3.1.3. PSTATE\_trap\_little\_endian (TLE)

При возникновении trap, текущее значение PSTATE помещается в trap-стек, а бит TLE помещается в CLE нового PSTATE.

3.1.4. PSTATE\_mem\_model (MM)

Два бита поля MM определяют модель памяти, используемую процессором.



(возможные значения MM)

Любая реализация архитектуры SPARC-V9 должна включать в себя TSO, более строгую модель, тогда как две другие необязательны к наличию.

3.1.5. PSTATE\_RED\_state (RED)

Если PSTATE.RED = 1, процессор работает в режиме RED (Reset, Error and Debug). Целочисленный блок устанавливает в это поле единицу, когда возникает hardware reset. Такое же значение устанавливается, если возникает TRAP в то время, когда TL = (MAXTL – 1).

3.1.6. PSTATE\_enable\_floating\_point (PEF)

Если PSTATE.PEF = 1, в работу будет включен блок операций с плавающей точкой, позволяя привилегированной программе управлять этим блоком. Чтобы блок можно было использовать, и PSTATE.PEF, и FPRS.FEF должны содержать единицу. В ином случае выполнение любой инструкции, ссылающейся на FPU, приведет к возникновению fp\_disabled trap.

3.1.7. PSTATE\_address\_mask (AM)

Если PSTATE.AM = 1, адреса инструкций и данных будут интерпретированы так, как если бы их старшие 32 бита были нулями перед отправкой в MMU или память. 32-битные приложения должны работать в режиме PSTATE.AM = 1.

3.1.8. PSTATE\_privileged\_mode (PRIV)

Если PSTATE.PRIV = 1, процессор работает в привилегированном режиме.

3.1.9. PSTATE\_interrupt\_enable (IE)

Если PSTATE.IE = 1, процессор будет принимать прерывания.

3.1.10. PSTATE\_alternate\_global (AG)

Если PSTATE.AG = 0, набор регистров r[0]..r[7] будет считаться стандартным набором глобальных регистров, и альтернативным, если PSTATE.AG = 1.

3.2. Регистр trap-уровня (Trap Level Register, TL)



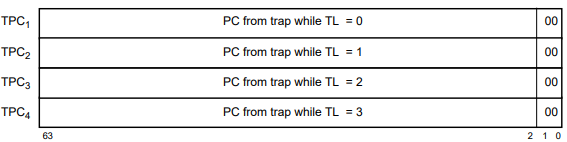
Регистр TL указывает текущий trap-уровень. TL = 0 является обычным уровнем операций (trap не происходят). При значении TL > 0 будут обрабатываться 1 или больше trap. Максимальное значение TL указывается в MAXTL. Реализация архитектуры должна поддерживать хотя бы 4 уровня trap помимо нулевого.

3.3. Регистр уровня прерываний (Processor Interrupt Level, PIL)



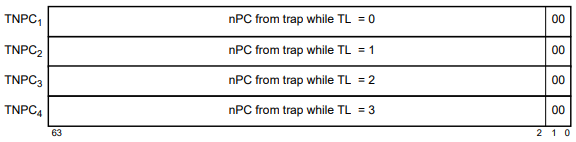
PIL указывает на уровень прерываний, начиная с которого процессор будет принимать то или иное прерывание. Большему числу соответствует больший уровень.

3.4. Траповый счетчик операций (Trap Program Counter, TPC)



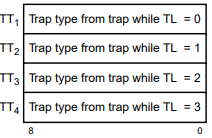
Какой-либо регистр TPC содержит в себе PC предыдущего trap-уровня. Количество регистров TPC = MAXTL, но лишь один доступен к работе в момент времени. Текущее значение TL определяет, какой из регистров TPC доступен. Попытка чтения или записи регистра TPC при TL = 0 приведет к исключению illegal\_instruction.

3.5. Траповый счетчик следующей операций (Trap Next Program Counter, TNPC)



Данный набор регистров аналогичен набору TPC, но содержит nPC вместо PC.

3.6. Регистр типа трапа (Trap Type Register, TT)



Регистр TT содержит тип trap, который вызвал вхождение в текущий trap-уровень.

Количество регистров TT = MAXTL, но лишь один доступен к работе в момент времени. Текущее значение TL определяет, какой из регистров TT доступен. Попытка чтения или записи регистра TT при TL = 0 приведет к исключению illegal\_instruction.

3.7. Регистр базового адреса трапов (Trap Base Address, TBA)



Регистр TBA предоставляет верхние 49 битов адреса, который используется для выбора вектора трапов. Нижние 15 битов всегда читаются как нули, и записи в них игнорируются.

Полный адрес вектора специфицируется в TBA, TL, TT[TL], а также пятью нулями.



Поле TL > 0 заполняется нулем, если при взятии трапа TL = 0, и наоборот.

3.8. Регистр версии (Version Register, VER)



Регистр версии уточняет фиксированные параметры, относящиеся к конкретной реализации процессора и набора масок. Этот регистр доступен только для чтения.

3.9. Регистры состояния регистрового окна (Register-Window State Registers)

Состояния регистровых окон определяется набором привилегированных регистров. Они могут считываться и записываться инструкциями RDPR/WRPR. Также эти регистры модифицируются инструкциями, связанными с регистровыми окнами, и используются при генерации трапов, позволяющих отслеживающим программам заполнять, проливать и очищать регистровые окна.

3.9.1. Указатель на текущее окно (Current Window Pointer, CWP)



Регистром CWP определяется, регистры какого окна будут представлены регистрами r[8]..r[31].

3.9.2. Регистр cохраняемых окон (Savable Windows Register, CANSAVE)



В регистре CANSAVE содержится число регистровых окон, следующих за CWP. Этим число определяется количество окон, которые не используются программой и которые можно аллоцировать командой SAVE, не получим исключение пролития.

3.9.3. Регистр восстанавливаемых окон (Restorable Windows Register, CANRESTORE)



В регистре CANRESTORE содержится число регистровых окон, предшествующих CWP. Этим числом определяется количество окон, которые используются программой и которые можно восстановить инструкцией RESTORE, не получив исключение заполнения.

3.9.4. Регистр других окон (Other Windows Register, OTHERWIN)



TODO

3.9.5. Регистр состояния окон (Window State Register, WSTATE)



TODO

3.9.6. Регистр чистых окон (Clean Windows Register, CLEANWIN)



В регистре CLEANWIN содержится количество окон, которые можно использовать посредством команды SAVE, не получив исключение clean\_window. К таким окнам относятся окна, содержащие нули, валидные адреса и валидные данные. Регистры таких окон не требуется очищать перед использованием.

3.10. Вспомогательные регистры состояния (Ancillary State Registers, ASRs)

Архитектура SPARC-V9 предоставляет до 25 ASR, пронумерованных от 7 до 31. При этом, регистры 7..15 зарезервированы архитектурой на будущее и не должны быть использованы никакими программами. Регистры 16..31 доступны к использованию и могут использоваться как счетчики, диагностические регистры, таймеры и т.п. Привилегированность доступа к таким регистрам, их количество и семантика обращения к ним зависят от конкретной реализации.

Регистры ASR считываются и записываются инструкциями RDASR и WRASR соответственно. Эти инструкции привилегированны, если привилегированны оговоренные регистры.

3.11. Очередь отложенных нецелочисленных трапов (Floating-Point Deferred-Trap Queue, FQ)

TODO

3.11. Очередь отложенных целочисленных трапов (IU Deferred-Trap Queue)

TODO

Инструкции

1. общее представление

Инструкции SPARC-V9 извлекаются процессором из памяти и затем выполняются, отменяются или приводят к ошибке (trap). Инструкции кодируются в четыре главных формата и подразделены на следующие 11 категорий:

* Доступа к памяти
* Синхронизации памяти
* Целочисленной арифметики
* Передачи управления (Control Transfer, CTI)
* Условных переходов
* Управления регистровыми окнами
* Доступа к регистрам состояния
* Доступа к привилегированным регистрам
* Оперирования числами с плавающей точкой
* Инструкции, зависящие от реализации
* Зарезервированные

1.1. выполнение инструкций

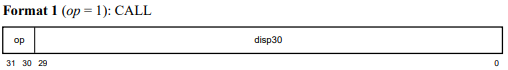
Инструкции извлекаются из участка памяти, который указан регистром PC. Любая инструкция может изменить состояния процессора и/или памяти. Побочным эффектом выполнения инструкции являются изменения значений в регистрах PC и nPC.

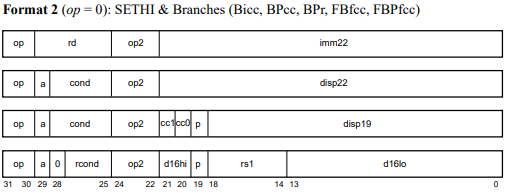
Если не возникает никаких ошибок и инструкция не является передачей управления, значение nPC копируется в PC, а сам nPC увеличивается на 4. Если же инструкция является передачей управления, nPC также копируется в PC, а новым значением nPC становится значение, указанное в выполняемой инструкции.

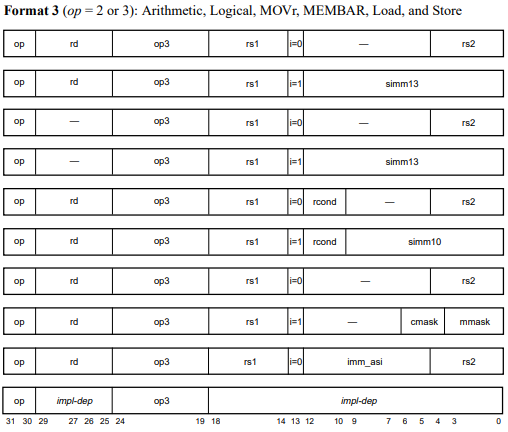
TODO

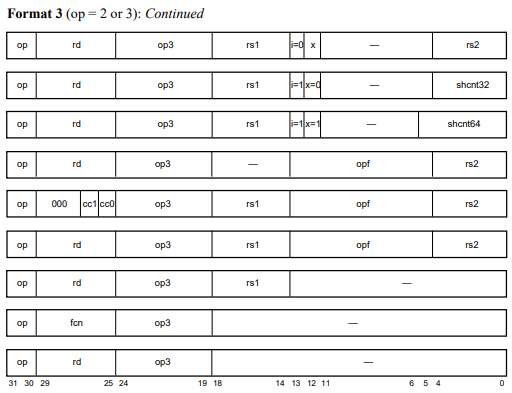
1.2. форматы инструкции

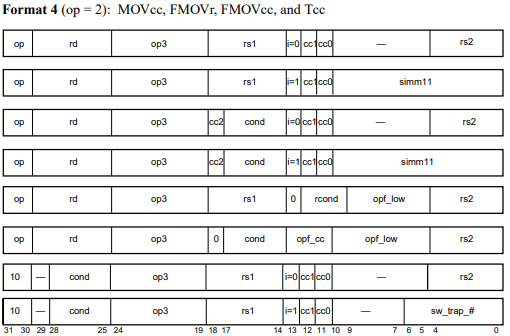
Инструкции кодируются в четыре главных 32-битных формата и несколько меньших.



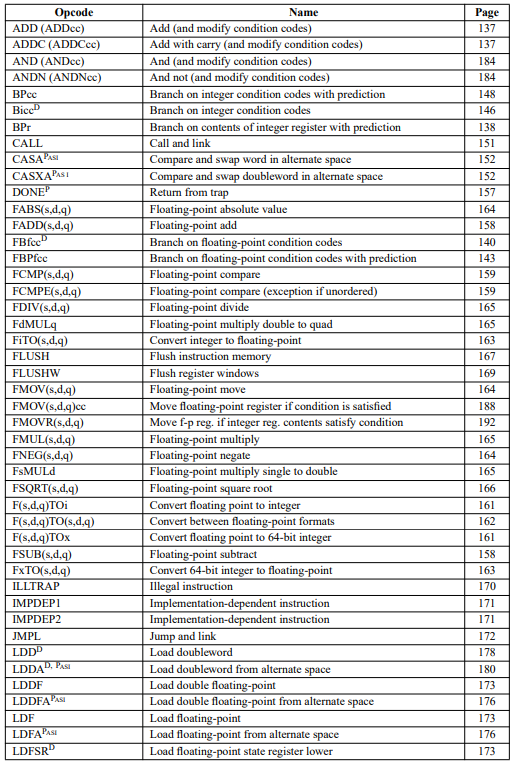


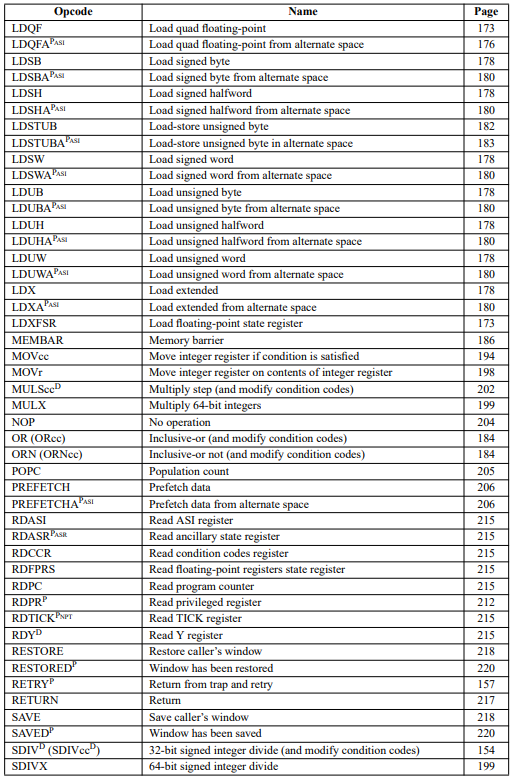


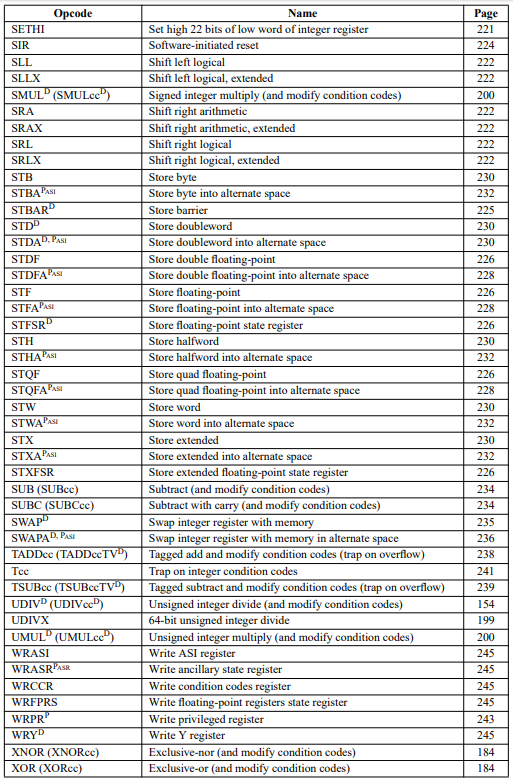




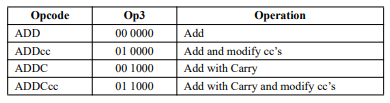
1.3. полный набор инструкций

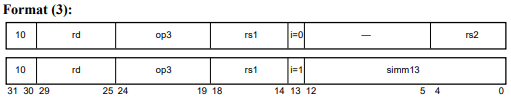






1.4. пример описания инструкции — инструкция ADD





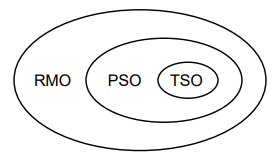
ADD и ADDcc высчитывают выражение “r[rs1] + r[rs2]”, если i = 0, или выражение “r[rs1] + sign\_ext(simm13)”, если I = 1, и записывают результат в r[rd].

ADDC и ADDCcc работают аналогично, но добавляют к результату значение поля c регистра CCR.

Модели памяти

Архитектура SPARC-V9 описывает три модели памяти:

* Total Store Order (TSO), обязательный к наличию по соображению обратной совместимости со SPARC-V8
* Partial Store Order (PSO)
* Relaxed Memory Order (RMO)



Программы, написанные для какой-либо из указанных моделей, будут также работать и на любой из включенных в нее моделей. При этом, ни одна из моделей не является приоритетной по стандарту.