**Расширение системы команд RISC-V**

В базовом варианте система команд RISC-V действительно выглядит минималистичной и небольшой. Но! На этом уже можно писать программы, запускать проекты и пр.(вспомним набор команд, например, 8080 или Z80 – он тоже «небогатый», что не помешало их популярности).

Безусловно, современные приложения требуют более гибких подходов, чем использование лишь небольшого множества команд. Богатый практический опыт RISC-архитектур показал, что изначально заложенные ограничения на фиксированный набор команд, их фиксированную длину и пр. в итоге на реальных задачах приводил к «распуханию» кода, что породило в ряде архитектур «костыли» в виде сокращенных наборов инструкций (Thumb).

В архитектуре RISC-V данный аспект попытались учесть в виде опции расширения базовой системы команд в любом практически направлении, обязав лишь только в любом случае поддерживать базовый набор команд (I) – для стабильной работы и совместимости программного обеспечения (хотя бы на уровне исходных текстов).

По мере распространения архитектуры RISC-V и накопления практики ее эксплуатации была разработана базового ISA (RV32I или RV64I) и ряда стандартных расширений (IMAFD), получившее шутливое название «набор для жизни», или ISA «общего назначения». Сейчас этот набор обозначается литерой G – рис.1. Обратите внимание, что RV64I считается не расширением RV32I, а другой полной базовой кодировкой.

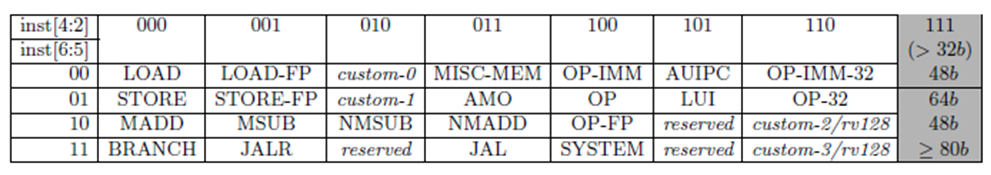


Рис.1 Карта основных кодов операций для RVG.

В дополнение к поддержке разработки стандартного программного обеспечения общего назначения, другой целью RISC-V является обеспечение основы для более специализированных расширений набора команд или более настраиваемых ускорителей.

Пробелы в кодировке команд и опциональное кодирование команд переменной длины предназначены для того, чтобы упростить использование возможностей разработки программного обеспечения для стандартной цепочки инструментов ISA при создании более специализированных процессоров. Для процессорных систем на базе FPGA этот пусть весьма интересен, плюс, естественно при разработке новых специализированных наборов инструкций.

Основные коды операций с установленными тремя или более младшими битами зарезервированы для команд длиной более 32 бит. Кодов операций, помеченных как зарезервированные (reserved), следует избегать для пользовательских расширений набора команд, поскольку они могут использоваться будущими стандартными расширениями.

**«Стандартные» и «нестандартное» расширения**

Любая реализация процессора RISC-V должна поддерживать базовое целое число ISA (RV32I или RV64I). Кроме того, реализация может поддерживать одно или несколько расширений.

Расширения делятся на две большие категории: стандартные и нестандартные.

**Стандартное расширение** - это то, которое в целом полезно и которое разработано таким образом, чтобы не конфликтовать с каким-либо другим стандартным расширением.

В настоящее время набор расширений MAFDQLCBTPV является либо завершенными, либо планируемыми стандартными расширениями, согласованными сообществом RISC-V.

**Нестандартное расширение** может быть (и с высокой долей вероятности будет) узкоспециализированным и может конфликтовать (пересекаться с кодировкой команд других расширений) с другими стандартными или нестандартно настроенными расширениями.

**Пробелы в кодировке команд и префиксы**

Пробел в кодировке команд - это некоторое количество битов команд, в пределах которых кодируется базовый ISA или ее расширение. RISC-V, в принципе, поддерживает различную длину команд, но даже в пределах одной длины команды доступны различные размеры пространства для кодирования.

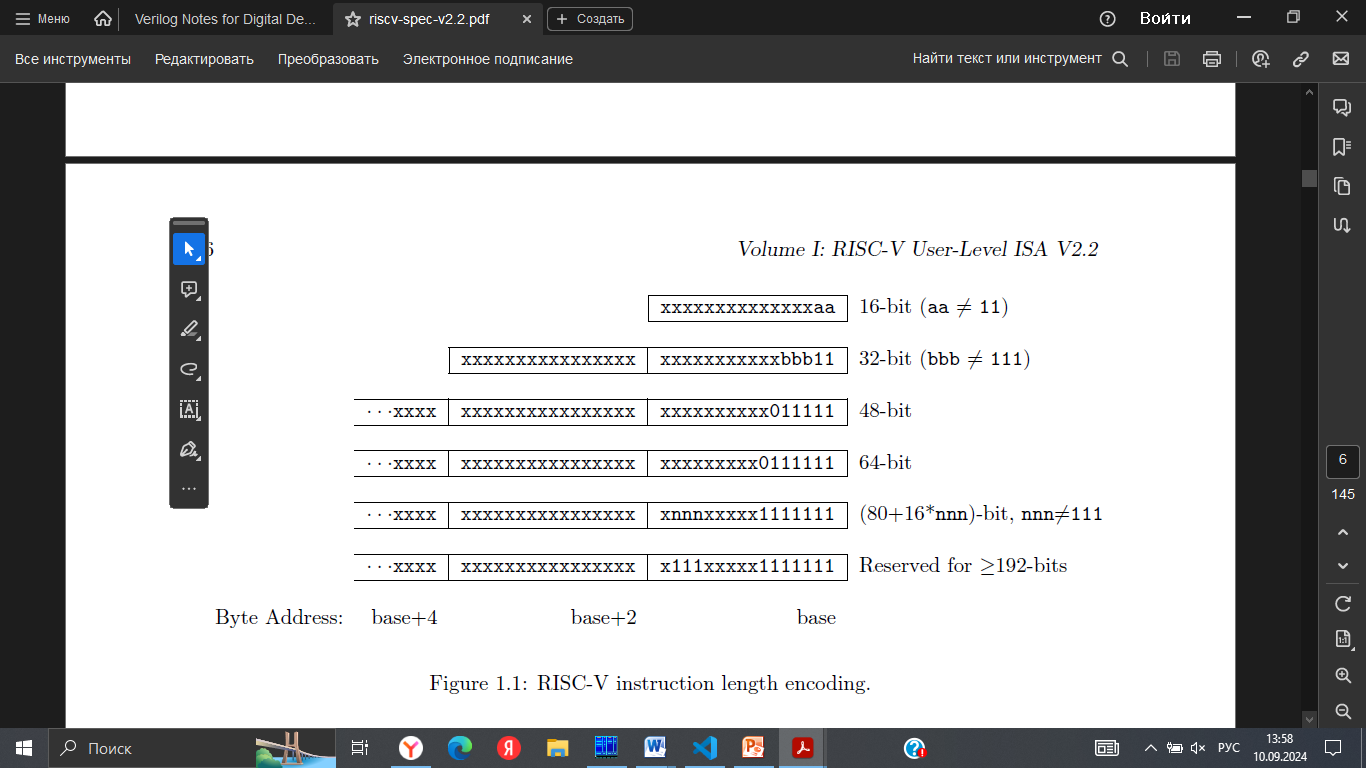


Рис.2 Кодирование длина инструкций в архитектуре RISC-V.

Префикс в базовой ISA – это двухбитное поле 11, содержащееся в битах 1--0 32-разрядного слова инструкции, в то время как префикс для стандартного атомарного расширения A является семибитным полем 0101111, содержащимся в битах 6—0.. Особенность формата кодирования заключается в том, что 3-разрядное поле funct3 используется для кодирования второстепенного кода операции и не является смежным с основными битами кода операции в 32-разрядном формате команд, но считается частью префикса для 22-разрядного пространства команд.

***Если смотреть на базовый набор команд (I), то «пробелом» в кодировке, допускающим расширения можно считать поле funct7 в инструкциях R-типа – почти полных 7-битный диапазон значение (за исключением funct7=0x00, 0x20) в кодировании операций не используется.***

***Также есть свободные значения поля fuct3 в инструкциях типа load/store.***

***Аналогично инструкция JARL занимает только одно значение из возможных вариантов поля funct3.***

**Глобальные кодировки, совместимые со стандартами**

Полное или глобальное кодирование ISA для реальной реализации RISC-V должно выделять уникальный неконфликтный префикс для каждого включенного пространства кодирования команд. Базовым и каждому стандартному расширению был выделен стандартный префикс, чтобы гарантировать, что все они могут сосуществовать в едином глобальном пространстве.

Совместимая со стандартами глобальная кодировка - это такая, в которой база и каждое включенное стандартное расширение имеют свои стандартные предустановки. Глобальная кодировка, совместимая со стандартом, может включать нестандартные расширения, которые не противоречат включенным стандартным расширениям.

Совместимая со стандартом глобальная кодировка может также использовать стандартные предустановки для нестандартных расширений, если соответствующие стандартные расширения не включены в глобальную кодировку. Другими словами, стандартное расширение должно использовать свой стандартный префикс, если оно включено в совместимую со стандартом глобальную кодировку, но в противном случае его префикс можно свободно перераспределять.

Эти ограничения позволяют общей цепочке инструментов ориентироваться на стандартное подмножество любой глобальной кодировки, совместимой со стандартом RISC-V.

**Гарантированное нестандартное пространство кодирования**

Для поддержки разработки собственных пользовательских расширений выделено части пространства кодирования инструкций никогда не будут использоваться стандартными расширениями.

Основные коды операций, помеченные как ***custom-0*** и ***custom-1*** (рис.1), будут исключены в будущих стандартных расширениях и рекомендованы для использования расширениями пользовательского набора команд в базовом 32-разрядном формате команд. Коды операций, помеченные custom-2/rv128 и custom-3/rv128, зарезервированы для дальнейшего использования RV 128, и поэтому также может использоваться для пользовательских расширений набора команд в RV32 и RV64.

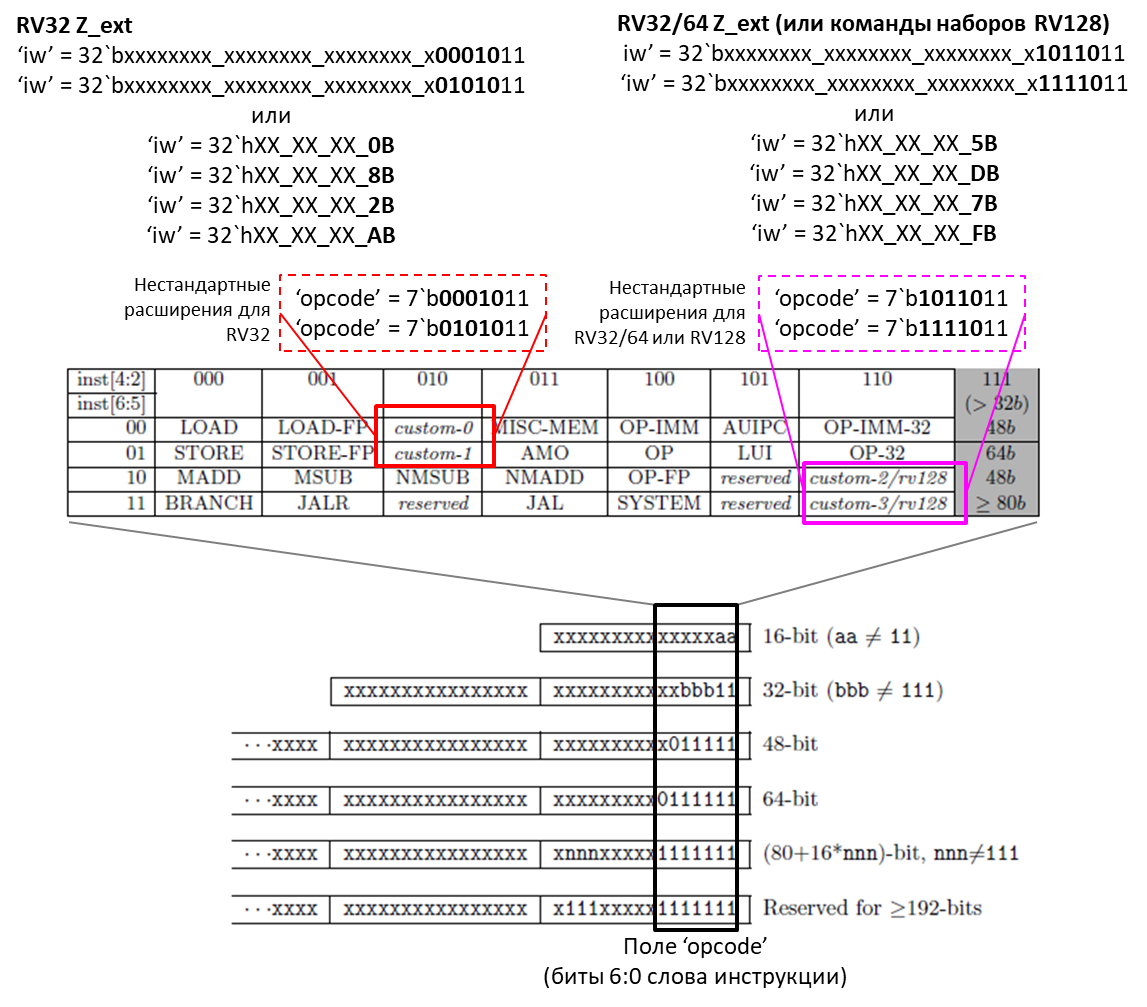


Рис. 3 Кодирование расширенных нестандартных наборов команд.

**Философия разработки расширений RISC-V**

Философия разработки расширений RISC-V нацелена на поддержку большого количество независимо разработанных расширений, поощряя, тем самым, разработчиков работать в рамках пространства кодировок команд и предоставляя инструменты для их упаковки в совместимую со стандартом глобальную кодировку путем выделения уникальных предварительных условий. Некоторые расширения более естественно реализованы в виде дополнений к существующим расширениям и будут совместно использовать любое значение префикса назначенное их родительскому расширению. Стандартное расширение префиксов позволяет избежать ложных несовместимостей в кодировании основных функциональных возможностей, в то же время, позволяя настраивать более экзотические расширения.

Эта возможность переупаковки расширений RISC-V в различные глобальные кодировки, совместимые со стандартами, может быть использована несколькими способами.

Одним из вариантов использования является разработка узкоспециализированных пользовательских ускорителей, предназначенных решения специфических задач. Можно, к примеру, отказаться от всего, кроме базового набора I, и добавить только расширения, необходимые для текущей задачи.

Другим вариантом использования является создание исследовательского прототипа для нового типа расширения набора команд. Исследователи, возможно, не захотят тратить время на реализацию модуля выборки команд переменной длины, и поэтому хотели бы создать прототип своего расширения, используя простую 32-битную кодировку команд фиксированной ширины. Однако это новое расширение может оказаться слишком большим, чтобы сосуществовать со стандартными расширениями в 32-разрядном пространстве. Если исследовательским экспериментам не нужны все стандартные расширения, «стандартно-совместимое» глобальное кодирование может исключить неиспользуемые стандартные расширения и повторно использовать их предварительные исправления, чтобы поместить предлагаемое расширение в нестандартное местоположение для упрощения разработки исследовательского прототипа.

Стандартные инструменты по-прежнему смогут ориентироваться на базу и любые стандартные расширения, которые присутствуют, чтобы сократить время разработки. Как только расширение набора команд будет оценено и изменено, оно может быть доступно для упаковки в большее пространство кодирования переменной длины, чтобы избежать конфликтов со всеми стандартными расширениями.

Если рассматривать карту опкодов команд (рис.1), то можно увидеть, что даже следуя соглашениям, и не занимая пространство зарезервированных расширений остаётся «место для манёвра», позволяющее реализовывать собственные подмножества команд, сохраняя совместимость с привычным или с наиболее распространённым инструментальным программным обеспечением.

**Расширения в пределах 32-битного формата инструкций**

В стандартном кодировании три из доступных 30-разрядных пробелов в кодировке команд (те, которые имеют 2-разрядные префиксы 00, 01 и 10) и по стандарту используются для включения дополнительного расширения сжатого формата инструкций. Однако, если расширение сжатого набора команд («С») не требуется, то данной пространство становятся доступным для использования (30 бит на кодировку – почти неограниченная свобода!))

**Доступные 25-разрядные пробелы в кодировке команд**

25-разрядный пробел в кодировке команд соответствует основному коду операции в базовой кодировке и стандартной кодировке расширения.

Существует четыре основных кода операции, специально зарезервированных для пользовательских расширений, каждый из которых представляет собой 25-битное пространство кодировки. Два из них зарезервированы для возможного использования в базовой кодировке RV128 (будут OP-IMM-64 и OP-64-OP-64), но могут использоваться для стандартных или нестандартных расширений для RV32 и RV64.

Два кода операций, зарезервированных для RV64 (OP-IMM-32 и OP-32-OP-32), также могут использоваться для стандартных и нестандартные расширения только для RV32.

Если реализация не требует плавающей точки, то семь основных кодов операций, зарезервированных для стандартных расширений с плавающей точкой (LOAD-FP, STORE-FP, MADD, MSUB, NMSUB, NMADD, OP-FP), могут быть повторно использованы для нестандартных расширений. Аналогично, основной код операции AMO может быть использован повторно, если стандартные атомарные расширения не требуются.

Если реализация не требует инструкций длиной более 32 бит, тогда доступны дополнительные четыре основных кода операции.

Базовая кодировка RV32I использует только 11 основных кодов операций плюс 3 зарезервированных кода операций, оставляя до 18 доступных для расширений.

Базовая кодировка RV64I использует только 13 основных кодов операций плюс 3 зарезервированных кода операций, оставляя до 16 доступных для расширений.

**Доступные 22-разрядные пробелы в кодировке команд**

22-разрядный пробел в кодировке соответствует второстепенному коду операции funct3 в базовой кодировке и стандартной кодировке расширения. Несколько основных кодов операций содержат второстепенный код операции funct3, который занят не полностью, оставляя доступными несколько 22-битных пробелов кодировки.

Обычно основной код операции выбирает формат, используемый для кодирования операндов в оставшихся битах инструкции, и в идеале расширение должно соответствовать формату операнда основного кода операции, чтобы упростить аппаратное декодирование.

**Другие пробелы**

Меньшие пробелы доступны под некоторыми основными кодами операций, и не все второстепенные коды операций заполнены полностью (поля funct3, funct7).

**Добавление выровненных 64-разрядных расширений команд**

Простейший подход к предоставлению места для расширений, которые слишком велики для базового 32-разрядного формата команд фиксированной ширины, заключается в добавлении естественно выровненных 64-разрядных команд. Реализация по-прежнему должна поддерживать 32-разрядный базовый формат команд, но может потребовать, чтобы 64-разрядные инструкции были выровнены по 64-разрядным границам для упрощения выборки команд, при этом 32-разрядная инструкция NOP используется в качестве дополнения выравнивания там, где это необходимо.

Чтобы упростить использование стандартных инструментов, 64-разрядные инструкции должны быть закодированы так, как описано на рис.2. Однако реализация может выбрать нестандартную кодировку длины инструкции для 64-разрядных инструкций, сохранив стандартную кодировку для 32-разрядных инструкций. Например, если сжатые инструкции не требуются, то 64-разрядная инструкция может быть закодирована с использованием одного или нескольких нулевых битов в первых двух битах инструкции.

**Поддержка кодировок VLIW**

Хотя RISC-V не был разработан в качестве основы для чисто VLIW-машины, кодировки VLIW могут быть добавлены в качестве расширений с использованием нескольких альтернативных подходов. Во всех случаях должна поддерживаться базовая 32-разрядная кодировка, позволяющая использовать любые стандартные программные средства.

**Группа команд фиксированного размера**

Простейший подход заключается в определении единого большого естественно выровненного формата команд (например, 128 бит) в рамках которого кодируются операции VLIW. В обычном VLIW такой подход привел бы к растрате памяти команд для хранения NOPS, но RISC-V-совместимая реализация должна была бы также поддерживать базовые 32-разрядные инструкции, ограничивая расширение размера кода VLIW функциями, ускоряемыми VLIW.

**Группы кодированной длины**

Другой подход заключается в использовании стандартной кодировки длины из рис.2 для параллельного кодирования в группы структур, позволяющие сжимать NOPs из инструкции VLIW. Например, 64-разрядная инструкция может содержать две 28-разрядные операции, в то время как 96-разрядная инструкция может содержать три 28-разрядные операции и так далее. В качестве альтернативы, 48-разрядная инструкция может содержать одну 42-разрядную операцию, в то время как 96-разрядная инструкция может содержать две 42-разрядные операции и так далее.

Этот подход имеет преимущество в сохранении базовой кодировки ISA для инструкций, выполняющих одну операцию, но имеет недостаток в том, что требуется новая 28-разрядная или 42-разрядная кодировка для операций внутри инструкций VLIW и смещенная выборка команд для больших групп. Одно из упрощений заключается в том, чтобы не позволять инструкциям VLIW пересекать определенные микроархитектурно значимые границы (например, строки кэша или страницы виртуальной памяти).

Основным недостатком этого подхода является то, что в базовом ISA отсутствует поддержка сложной предикации, обычно требуемая в агрессивной системе VLIW, и трудно добавить пространство для указания большего количества регистров предикатов в стандартном 30-битном пространстве кодирования.

**Именование расширений**

Стандартным расширениям ISA присваивается имя, состоящее из одной буквы. Например, первыми четырьмя стандартными расширениями целочисленных базисов являются: «M» для целочисленного умножения и деления, «A» для инструкций атомной памяти, «F» для инструкций с плавающей запятой одинарной точности и «D» для инструкций с плавающей запятой двойной точности. Любой вариант набора команд RISC-V может быть кратко описан путем объединения префикса базового целого числа с именами включенных расширений, например, «RV64IMAFD».

Стандартизованные сообществом RISC-V специализированные расширения начинаются с префикса Z.

Название нестандартных пользовательских расширений должно начинаться с префикса «Х».

**Выводы**

Опции расширения системы команд позволяют практически безболезненно вылетать в систему команд собственные операции, специфичные для каждой конкретной задачи и подержанные в FPGA-системе специализированными аппаратными блоками.

На самом деле, если присмотреться внимательнее, то можно увидеть, что расширение функциональности ядра RISC-V возможно даже в рамках базовой (или любой другой стандартной) системы команд. Это станет понятно при дальнейшем рассмотрении при построении декодера команд RV и микроархитектурных блоков процессора.

Это возможно сделать за счет блока регистров специального назначения (CSR) – да, ряд его адресов зарезервирован под «стандартные» функции, но тем не менее значительное количество адресов свободно и может быть задействовано с пользой для задачи. Под видом регистров специального назначения могут скрываться регистры управления или данных любых пользовательских вычислительных или исполнительных блоков или устройств. И разработчику достаточно будет лишь определить протокол взаимодействия с ними, и далее работать штатным средствами типового программного обеспечения.

Типовые «стандартные» CSR-регистры и их адреса приведены рисунках 4-6.

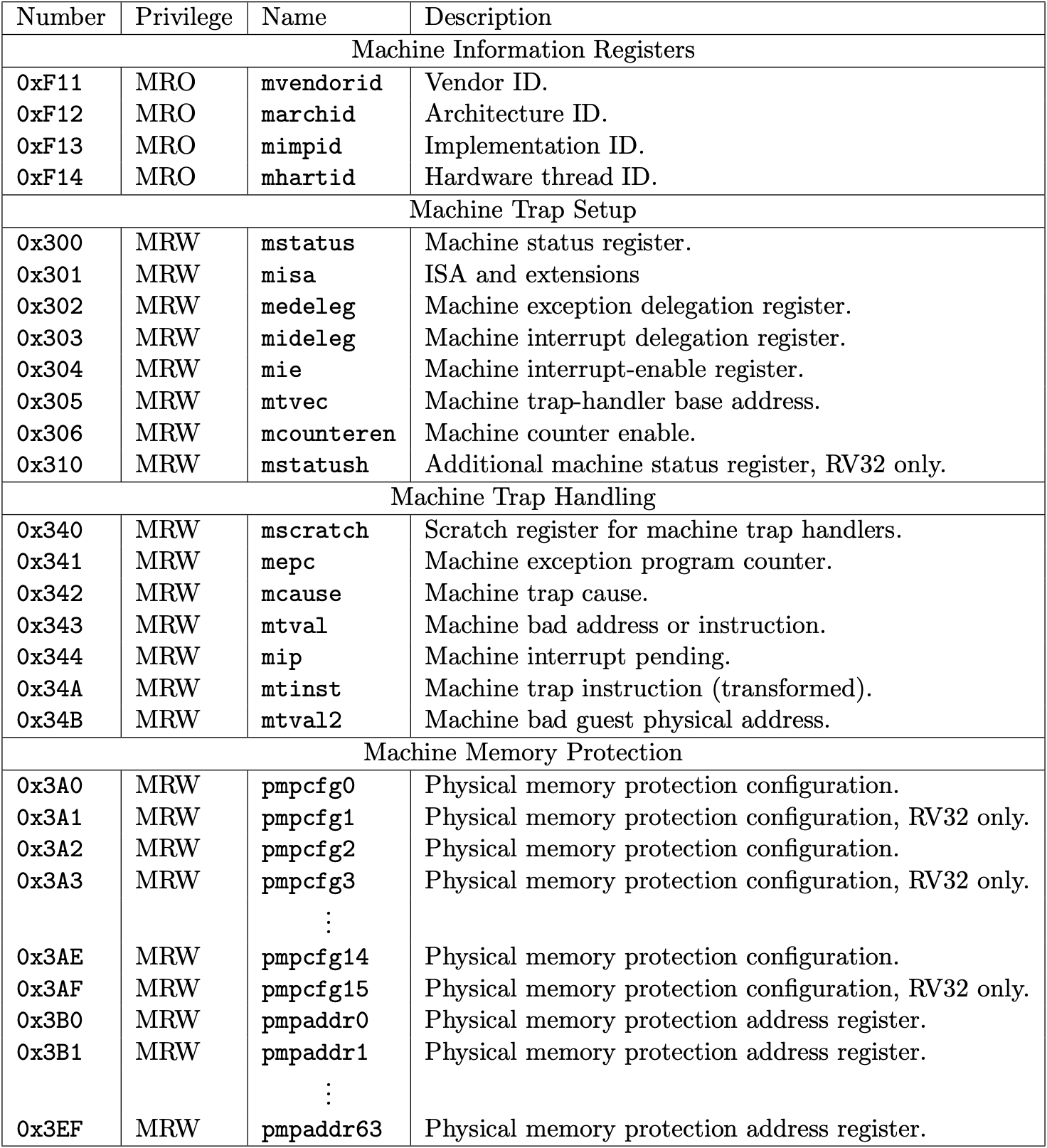


Рис.4 CSR регистры машинного уровня.

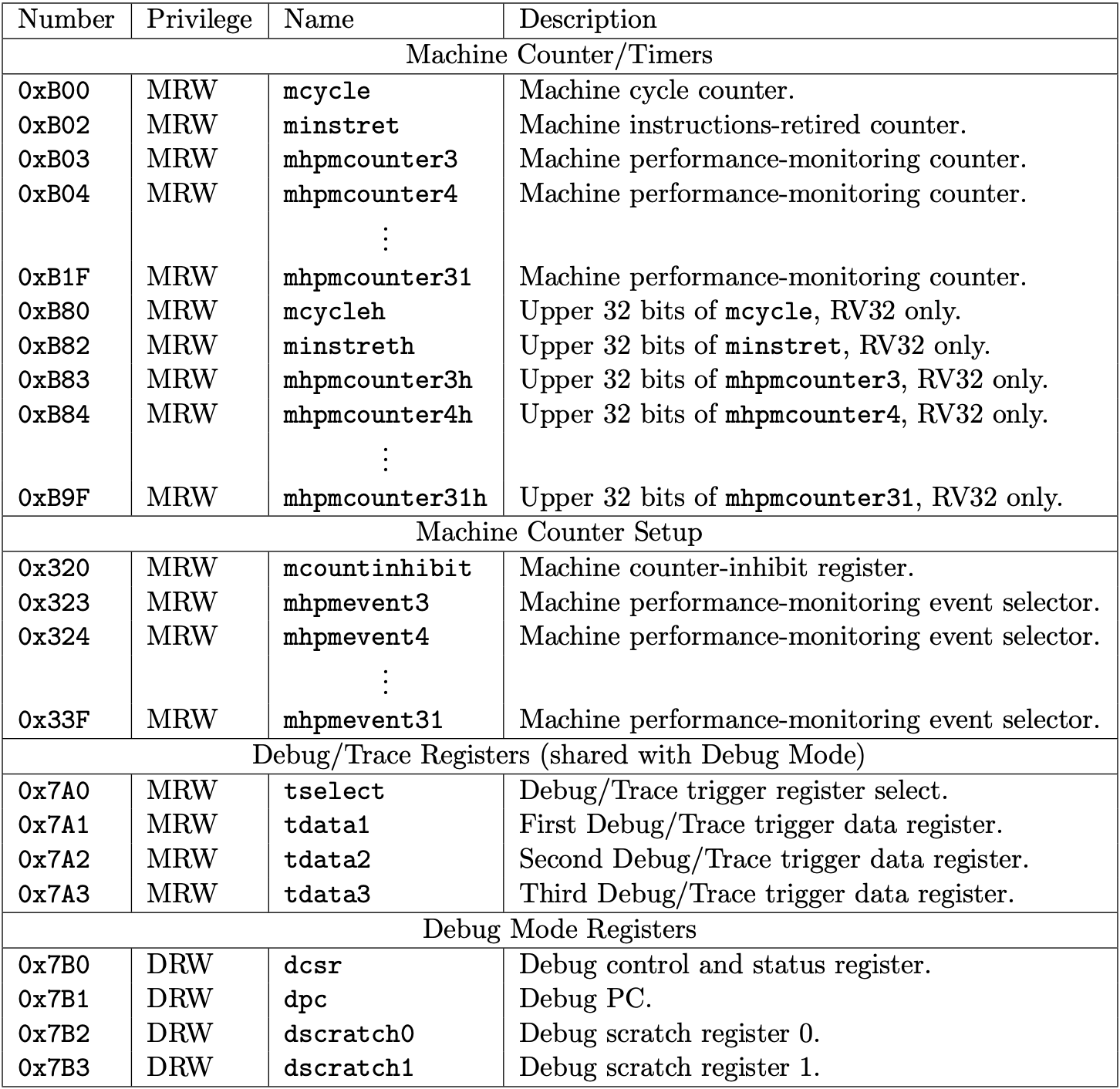


Рис.5 CSR регистры машинного уровня и регистры режима отладки.

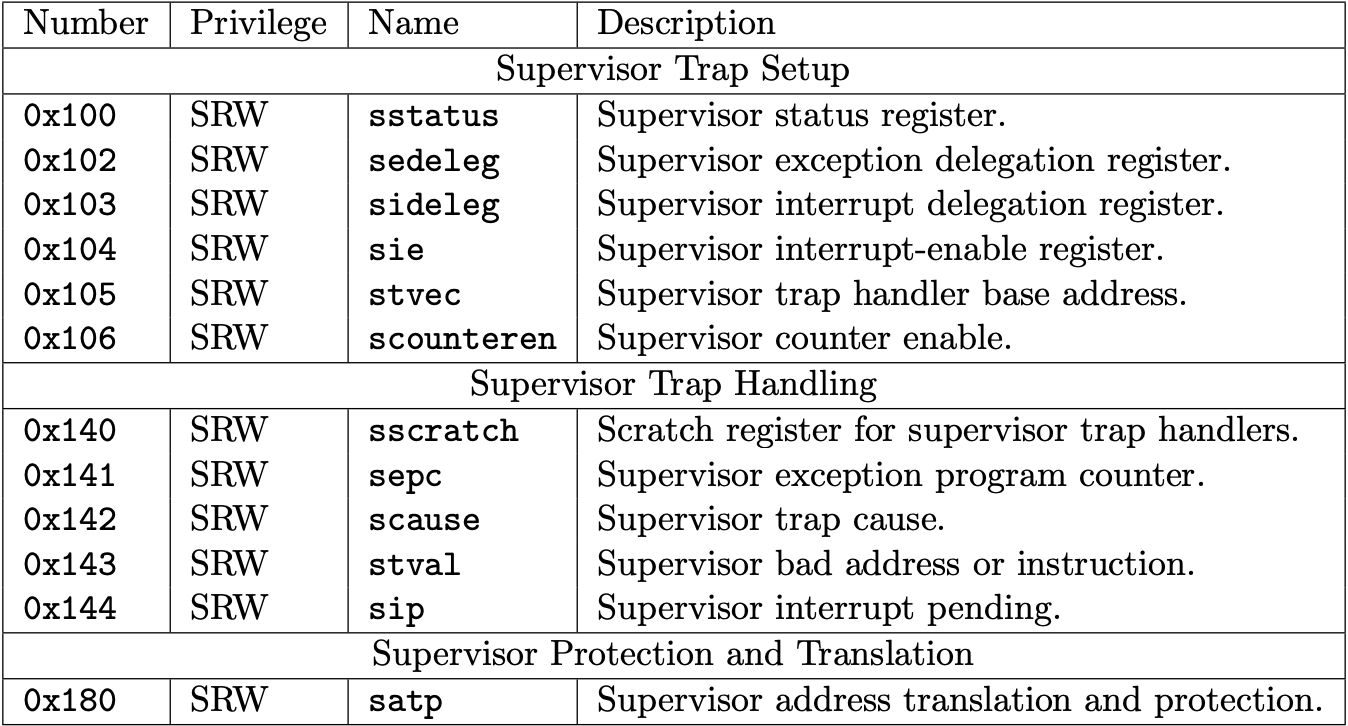


Рис. 6 CSR регистры уровня супервизора.

**Литература**

1. The RISC-V Instruction Set Manual. Volume I: User-Level ISA Document Version 2.2 // Editors: Andrew Waterman1, Krste Asanov
2. RISC-V Assembly Language Programmer Manual Part I developed by: SHAKTI Development Team @ iitm ’20 shakti.org.in