УДК 004.27 DOI: 10.31114/2078-7707-2020-3-165-171

Система команд процессоров архитектуры R^{2}T

В.В.Ерохин

АО «НИИМА» ПРОГРЕСС», г. Москва, vladimir.v.erokhin@gmail.com

Аннотация — Предлагается система команд процессорных архитектур большой разрядности. Система команд состоит из набора команд обычного процессора и набора для процессоров встраиваемых приложений.

***Ключевые слова* — архитектура, процессор, система команд.**

I. Введение

Автор не ставил перед собой задачи разработки «еще одного процессора», это задача несложная. Разработать архитектуру действительно новую, которая может быть использована в дальнейших разработках процессоров большой разрядности – цель данной работы.

В работе сделана попытка определить систему команд, которая с минимальными изменениями может быть использована в архитектурах как 32-разрядных (даже 16- и 8-битных), так в процессорах с очень высокой разрядностью – до 1024 бит и много выше. Обобщенное наименование всех архитектур этого класса - RISC**2 (R^{2}T).

Основой для построения системы команд были использованы следующие принципы:

- **Константы можно только загружать, в том числе и непосредственно (за исключением небольшого числа операций).**
- **Отказ от понятия слова в плане выделения данных некоторой размерности в операциях.**
- **Набор инструкций (базовый и расширения системы команд) одинаков для всех вариантов процессоров от 32 (16 или даже 8) битных до максимума и, желательно, всех расширений архитектуры.**

В табл. 1 приведены используемые в работе обозначения.

Таблица 1

Используемые обозначения

w8, w16, w32, w64, w128, w256, w512, w1024, w2048…	Размеры операнда, соответственно: байт, 16-битное слово, 32-битное слово, 64-битное слово, 128-битное слово, 256-битное слово, 512-битное слово, 1024-битное слово, 2048-битное слово.
qualifier	Параметр команды, определяющий размер операнда, к которому относится операция. Допустимые значения {**w8, w16, w32, w64, w128, w256, w512, w1024, w2048, etc.**}
offset**xx**	Смещение адреса в виде знаковой константы длиной **xx** бит, содержащейся непосредственно в командах
imm**xx**	Непосредственные данные со знаком или без знака (определяется командой), содержащиеся в коде команды длиной **xx** бит.
<<	Сдвиг влево на указанное далее число разрядов
c(xlen-1)	бит переноса из разряда (xlen-1)
succ	Следующий элемент, в данном тексте подразумевается следующий по номеру регистр

II. Архитектура процессоров RISC**2 (R2T)

Разумеется, в работе использовались некоторые идеи из [1] при выборе некоторых команд, а также работа [2] в определении набора регистров процессоров для встраиваемых приложений.

Формат команд и их мнемоника процессоров R^{2}T приведено в таблице 2.

В таблице 3 дано полное описание системы команд процессоров R^{2}T.

Система команд состоит из основного набора команд и нескольких расширений системы команд, описание которых выделено голубым цветом в табл.3. Зеленым цветом в этой же таблице выделено подмножество системы команд для встраиваемых приложений. Следует учесть, что процессоры для встраиваемых приложений имеют набор регистров, состоящий из 16 регистров общего назначения в отличие от обычных

процессоров архитектуры R^{2}T, у которых число регистров общего назначения равно 32.

Таблица 2

Формат команд и их мнемоника процессоров R2T

31															16	15					`									0	№ бита / мнемоника
rd					rs1					rs2											0	0	0	0	0	0	1	0	0	0	0	add rd, rs1, rs2
rd					rs1					rs2											0	0	0	0	0	1	0	0	0	0	0	sub rd, rs1, rs2
rd					rs1					rs2											0	0	0	0	0	1	1	0	0	0	0	and rd, rs1, rs2
rd					rs1					rs2											0	0	0	0	1	0	0	0	0	0	0	or rd, rs1, rs2
rd					rs1					rs2											0	0	0	0	1	0	1	0	0	0	0	xor rd, rs1, rs2
rd					rs1					rs2											0	0	0	0	1	1	0	0	0	0	0	shl rd, rs1, rs2
rd					rs1					rs2											0	0	0	0	1	1	1	0	0	0	0	shr rd, rs1, rs2
rd					rs1					rs2											0	0	0	1	0	0	0	0	0	0	0	sar rd, rs1, rs2
rd					rs					x0											0	0	0	1	0	0	1	0	0	0	0	not rd, rs
rd					rs					qualifier											0	0	0	1	0	1	0	0	0	0	0	sext qualifier, rd, rs
0					0					qualifier											0	0	0	1	0	1	1	0	0	0	0	oflchck qualifier
rd					rs					offset[10:0]											0	0	0	1	1	0	0	0	0	0	0	jalr rd, rs, offset11
offset[15:11]					rs					offset[10:0]											0	0	0	1	1	0	1	0	0	0	0	bz rs, offset16
offset[15:11]					rs					offset[10:0]											0	0	0	1	1	1	0	0	0	0	0	bg rs, offset16
offset[15:11]					rs					offset[10:0]											0	0	0	1	1	1	1	0	0	0	0	bgu rs, offset16
rd					imm[15:0]																0	0	1	0	0	0	0	0	0	0	0	mvs rd, imm16
rd					rs					imm[10:0]											0	0	1	0	0	0	1	0	0	0	0	addi rd, rs, imm11
rd					rs					imm[10:0]											0	0	1	0	0	1	0	0	0	0	0	shli rd, rs, imm11
rd					rs					imm[10:0]											0	0	1	0	0	1	1	0	0	0	0	shri rd, rs, imm11
rd					rs					imm[10:0]											0	0	1	0	1	0	0	0	0	0	0	sari rd, rs, imm11
rd					rs					0											0	0	1	0	1	0	1	0	0	0	0	clo rd, rs
rd					rs1					rs2											0	0	1	0	1	1	0	0	0	0	0	muls rd, rs1, rs2
rd					rs1					rs2											0	0	1	0	1	1	1	0	0	0	0	mulu rd, rs1, rs2
rd					rs1					rs2											0	0	1	1	0	0	0	0	0	0	0	addc rd, rs1, rs2
rd					rs1					rs2											0	0	1	1	0	0	1	0	0	0	0	subb rd, rs1, rs2
rd					rs1					rs2											0	0	1	1	0	1	0	0	0	0	0	divs rd, rs1, rs2
rd					rs1					rs2											0	0	1	1	0	1	1	0	0	0	0	divu rd, rs1, rs2
rd					rs					qualifier											0	0	1	1	1	0	0	0	0	0	0	csrwr qualifier, rd, rs
rd					rs					qualifier											0	0	1	1	1	0	1	0	0	0	0	csrrd qualifier, rd, rs
															rd						rs1					rs2					0	0	0	1	add rd, rs1, rs2
															rd						rs1					rs2					0	0	1	0	sub rd, rs1, rs2
															rd						rs1					rs2					0	0	1	1	and rd, rs1, rs2
															rd						rs1					rs2					0	1	0	0	or rd, rs1, rs2
															rd						rs1					rs2					0	1	0	1	xor rd, rs1, rs2
															rd						rs1					rs2					0	1	1	0	shl rd, rs1, rs2
															rd						rs1					rs2					0	1	1	1	shr rd, rs1, rs2
															rd						rs1					rs2					1	0	0	0	sar rd, rs1, rs2
															rd						offset[7:0]										1	0	0	1	bz rs, offset8

rd	offset[7:0]					1	0	1	0	bg	rs, offset8	
rd	offset[7:0]					1	0	1	1	bgu	rs, offset8	
rd	imm[7:0]					1	1	0	0	mvs	rd, imm8	
0	0	x	x	0	0	1	1	0	1	oflchck	qualifier	
rd	rs	x	x	0	1	1	1	0	1	sext	qualifier, rd, rs	
rd	rs	x	x	1	0	1	1	0	1	ld	qualifier, rd, rs	
rd	rs	x	x	1	1	1	1	0	1	st	qualifier, rd, rs	
rd	rs	0	0	0	0	1	1	1	0	not	rd, rs	
rd	rs	0	0	1	0	1	1	1	0	jalr	rd, rs	
rd	rs	0	1	0	0	1	1	1	0	clo	rd, rs	
0	0	0	1	1	0	1	1	1	0	ecall		
0	0	1	0	0	0	1	1	1	0	ebreak		
0	0	1	0	1	0	1	1	1	0	pusha		
0	0	1	1	0	0	1	1	1	0	popa		
rd	offset[19:0]					0	0	1	1	1	1	0	jal	rd, offset20
rd	offset[19:0]					0	1	1	1	1	1	0	auipc	rd, offset20
rd	rs	qualifier	offset[10:0]			0	0	1	1	1	1	ld	qualifier, rd, rs, offset11	
rd	rs	qualifier	offset[10:0]			0	1	1	1	1	1	st	qualifier, rd, rs, offset11	
rd	rs	qualifier	offset[10:0]			1	0	1	1	1	1	ll	qualifier, rd, rs, offset11	
rd	rs	qualifier	offset[10:0]			1	1	1	1	1	1	sc	qualifier, rd, rs, offset11	

Примечание 1. Цветом в графе «мнемоника» выделены команды процессора для встраиваемых приложений.

Примечание 2. Неокрашенные битовые поля имеют значение 0.

Таблица 3

Описание системы команд процессоров R2T

мнемоника	Описание
add rd, rs1, rs2	rd = rs1 + rs2 Регистр rd получает значение суммы регистров rs1 и rs2. Суммируются регистры целиком, никакого деления на части (байт, слово, двойное слово…) не делается. Исключение по переполнению не выполняется. Переполнение может фиксироваться командой oflchck, следующей за данной операцией, по результатам выполнения которой выполняется исключение по переполнению.
sub rd, rs1, rs2	rd = rs1 - rs2 Регистр rd получает значение разности регистров rs1 и rs2. Операция выполнятся над полными регистрами, никакого деления на части (байт, слово, двойное слово…) не делается. Исключение по переполнению не выполняется. Переполнение может фиксироваться командой oflchck, следующей за данной операцией, по результатам выполнения которой выполняется исключение по переполнению.
and rd, rs1, rs2	Операции побитового логического «И» над операндами rs1, rs2, результат фиксируется в регистре rd.
or rd, rs1, rs2	Операции побитового логического «ИЛИ» над операндами rs1, rs2, результат фиксируется в регистре rd.
xor rd, rs1, rs2	Операции исключающего «ИЛИ» над операндами rs1, rs2, результат фиксируется в регистре rd.
shl rd, rs1, rs2	Операция сдвига влево операнда в регистре rs1 на величину, записанную в регистре rs2. Результат фиксируется в регистре rd.

shr rd, rs1, rs2	Операция логического сдвига вправо операнда в регистре rs1 на величину, записанную в регистре rs2. Результат фиксируется в регистре rd.
sar rd, rs1, rs2	Операция арифметического сдвига вправо (знаковый разряд операнда дублируется, т. е. знак числа сохраняется) операнда в регистре rs1 на величину, записанную в регистре rs2. Результат фиксируется в регистре rd.
not rd, rs	Операция побитового отрицания операнда в регистре rs1. Результат фиксируется в регистре rd.
sext qualifier, rd, rs	Операция расширения знака операнда, расположенного в регистре rs, результат записывается в регистр rd. Величина операнда определяется параметром qualifier и может принимать значения {w8, w16, w32, w64, w128…}. Если значение qualifier = w8, то расширяется знак байта, если w16 – знак 16-разрядного слова…
oflchck qualifier	Проверка возникновения переполнения по результату операций сложения и вычитания. Параметр qualifier определяет размер результата, который следует проверять на переполнение. Если ожидаемый результат операции сложения/вычитания qualifier = w8, то проверяется переполнение при операции на байте, если w16 – 16-разрядного слова и т. д. В случае, если было переполнение, возникает исключение переполнения и переход к соответствующей процедуре.
jalr rd, rs, offset11	Переход по адресу rs + offset11, где rs- содержимое регистра rs, а offset11 – знаковое число. Адрес возврата сохраняется в регистре rd.
bz rs, offset16	Условный переход. Если rs = 0, то осуществляется переход по адресу PC + offset16, где offset16 – целое со знаком.
bg rs, offset16	Условный переход. Если rs > 0, то осуществляется переход по адресу PC + offset16, где rs – целое со знаком, offset16 – целое со знаком.
bgu rs, offset16	Условный переход. Если rs > 0, то осуществляется переход по адресу PC + offset16, где rs – целое без знака, offset16 – целое со знаком.
mvs rd, imm16	Операция сдвига регистра rd влево с одновременным вдвиганием в регистр непосредственного 16-разрядного операнда. Используется для задания начальных значений путем одно- или неоднократного выполнения команды.
addi rd, rs, imm11	rd = rs1 + imm11 Регистр rd получает значение суммы регистра rs1 и знакового числа imm11. Исключение по переполнению не выполняется. Переполнение может фиксироваться командой oflchck, следующей за данной операцией, по результатам выполнения которой выполняется исключение по переполнению. Используется для изменения параметров циклов и т. п. простых случаев. Команда из расширения «0» системы команд.
shli rd, rs, imm11	Операция сдвига влево операнда в регистре rs на величину, записанную в поле imm11. Результат фиксируется в регистре rd. Если значение imm11 больше разрядности регистра, rd получает значение 0. Команда из расширения «0» системы команд.
shri rd, rs, imm11	Операция логического сдвига вправо операнда в регистре rs на величину, записанную в поле imm11. Результат фиксируется в регистре rd. Если значение imm11 больше разрядности регистра, rd получает значение 0. Команда из расширения «0» системы команд.
sari rd, rs, imm11	Операция арифметического сдвига вправо операнда в регистре rs на величину, записанную в поле imm11. Результат фиксируется в регистре rd. Если значение imm11 больше разрядности регистра, rd получает значение знака числа, записанного в rs. Команда из расширения «0» системы команд.
clo rd, rs	Подсчет числа лидирующих единиц в регистре rs с записью результата в rd.

	Команда из расширения «B» системы команд.
muls rd, rs1, rs2	Умножение знаковое операндов в регистрах rs1, rs2 и записью результата в пару (rd, succ(rd)). Команда из расширения «M» системы команд.
mulu rd, rs1, rs2	Умножение беззнаковое операндов в регистрах rs1, rs2 и записью результата в пару (rd, succ(rd)). Команда из расширения «M» системы команд.
addc rd, rs1, rs2	Сложение с переносом rd = rs1 + rs2 + cf Команда из расширения «M» системы команд.
subb rd, rs1, rs2	Вычитание с заемом rd = rs1 - rs2 – cf (Команда из расширения «M» системы команд)
divs rd, rs1, rs2	Деление знаковое операндов в регистрах rs1, rs2 и записью результата в пару (rd, succ(rd)). В rd записывается частное, остаток - в регистр со следующим номером. Команда из расширения «D» системы команд.
divu rd, rs1, rs2	Деление беззнаковое операндов в регистрах rs1, rs2 и записью результата в пару (rd, succ(rd)). В rd записывается частное, остаток - в регистр со следующим номером. Команда из расширения «D» системы команд.
csrwr qualifier, rd, rs	Запись значения из регистра rs в регистр управления/статуса rd
csrrd qualifier, rd, rs	Чтение из регистра управления/статуса rs в регистр rd
add rd, rs1, rs2	rd = rs1 + rs2 Регистр rd получает значение суммы регистров rs1 и rs2. Суммируются регистры целиком, никакого деления на части (байт, слово, двойное слово…) не делается. Исключение по переполнению не выполняется. Переполнение может фиксироваться командой oflchck, следующей за данной операцией, по результатам выполнения которой выполняется исключение по переполнению.
sub rd, rs1, rs2	rd = rs1 - rs2 Регистр rd получает значение разности регистров rs1 и rs2. Операция выполнятся над полными регистрами, никакого деления на части (байт, слово, двойное слово…) не делается. Исключение по переполнению не выполняется. Переполнение может фиксироваться командой oflchck, следующей за данной операцией, по результатам выполнения которой выполняется исключение по переполнению.
and rd, rs1, rs2	Операции побитового логического «И» над операндами rs1, rs2, результат фиксируется в регистре rd.
or rd, rs1, rs2	Операции побитового логического «ИЛИ» над операндами rs1, rs2, результат фиксируется в регистре rd.
xor rd, rs1, rs2	Операции исключающего «ИЛИ» над операндами rs1, rs2, результат фиксируется в регистре rd.
shl rd, rs1, rs2	Операция сдвига влево операнда в регистре rs1 на величину, записанную в регистре rs2. Результат фиксируется в регистре rd.
shr rd, rs1, rs2	Операция логического сдвига вправо операнда в регистре rs1 на величину, записанную в регистре rs2. Результат фиксируется в регистре rd.
sar rd, rs1, rs2	Операция арифметического сдвига вправо (знаковый разряд операнда дублируется, т. е. знак числа сохраняется) операнда в регистре rs1 на величину, записанную в регистре rs2. Результат фиксируется в регистре rd.
bz rs, offset8	Условный переход. Если rs = 0, то осуществляется переход по адресу PC + offset8, где offset8 – целое со знаком.

bg rs, offset8	Условный переход. Если rs > 0, то осуществляется переход по адресу PC + offset8, где rs – целое со знаком, offset8 – целое со знаком.
bgu rs, offset8	Условный переход. Если rs > 0, то осуществляется переход по адресу PC + offset8, где rs – целое без знака, offset8 – целое со знаком.
mvs rd, imm8	Операция сдвига регистра rd влево с одновременным вдвиганием в регистр непосредственного 8-разрядного операнда. Используется для задания начальных значений путем одно- или неоднократного выполнения команды.
oflchck qualifier	Проверка возникновения переполнения по результату операций сложения и вычитания. Параметр qualifier определяет размер результата, который следует проверять на переполнение. Если ожидаемый результат операции сложения/вычитания qualifier = w8, то проверяется переполнение при операции на байте, если w16 – 16-разрядного слова и т. д. В случае, если было переполнение, возникает исключение переполнения и переход к соответствующей процедуре.
sext qualifier, rd, rs	Операция расширения знака операнда, расположенного в регистре rs, результат записывается в регистр rd. Величина операнда определяется параметром qualifier и может принимать значения {w8, w16, w32, w64, w128…}. Если значение qualifier = w8, то расширяется знак байта, если w16 – знак 16-разрядного слова…
ld qualifier, rd, rs	Загрузка данных в регистр rd из памяти по адресу rs + offset12, где rs – содержимое регистра rs (база), offset12 – смещение без знака. Параметр qualifier определяет тип загружаемых данных и выравнивание адреса. Если qualifier = w8, то адрес может быть произвольным, если w16 – кратен 2, w32 – 4, w64 – 8.
st qualifier, rd, rs	Загрузка данных из регистра rd в память по адресу rs + offset12, где rs – содержимое регистра rs (база), offset12 – смещение без знака. Параметр qualifier определяет тип загружаемых данных и выравнивание адреса. Если qualifier = w8, то адрес может быть произвольным, если w16 – кратен 2, w32 – 4, w64 – 8.
not rd, rs	Операция побитового отрицания операнда в регистре rs. Результат фиксируется в регистре rd.
jalr rd, rs	Переход по адресу rs, где rs- содержимое регистра rs. Адрес возврата сохраняется в регистре rd.
clo rd, rs	Подсчет числа лидирующих единиц в регистре rs с записью результата в rd.
ecall	Инструкция используется, чтобы сделать запрос к поддерживающей среде выполнения (ОС)
ebreak	Инструкция используется отладчиками, чтобы вернуть управление обратно в среду отладки.
pusha	Загрузка регистрового файла в стек
popa	Выгрузка регистрового файла из стека
jal rd, offset20	Переход по адресу pc + offset20, где offset20 – знаковое число. Адрес возврата pc сохраняется в регистре rd.
auipc rd, offset20	Операция загрузки в rd значения pc + offset20<<12.
ld qualifier, rd, rs, offset12	Загрузка данных в регистр rd с расширением знака из памяти по адресу rs + offset12, где rs – содержимое регистра rs (база), offset12 – смещение без знака. Параметр qualifier определяет тип загружаемых данных и выравнивание адреса. Если qualifier = w8, то адрес может быть произвольным, если w16 – кратен 2, w32 – 4, w64 – 8 и т. д.
st qualifier, rd, rs, offset12	Загрузка данных из регистра rd в память по адресу rs + offset12, где rs – содержимое регистра rs (база), offset12 – смещение без знака.

	Параметр qualifier определяет тип загружаемых данных и выравнивание адреса. Если qualifier = w8, то адрес может быть произвольным, если w16 – кратен 2, w32 – 4, w64 – 8 и т. д.
ll qualifier, rd, rs, offset12	Атомарное чтение данных в регистр rd с расширением знака из памяти по адресу rs + offset12, где rs – содержимое регистра rs (база), offset12 – смещение без знака. Параметр qualifier определяет тип загружаемых данных и выравнивание адреса. Если qualifier = w8, то адрес может быть произвольным, если w16 – кратен 2, w32 – 4, w64 – 8 и т. д.
sc qualifier, rd, rs, offset12	Атомарная запись данных из регистра rd в память по адресу rs + offset12, где rs – содержимое регистра rs (база), offset12 – смещение без знака. Если запись успешна, в регистр rd записывается 1 и 0 в противном случае. Параметр qualifier определяет тип загружаемых данных и выравнивание адреса. Если qualifier = w8, то адрес может быть произвольным, если w16 – кратен 2, w32 – 4, w64 – 8 и т. д.

*Если номер регистра rd равен максимальному, запоминается только первая часть результата. При умножении это старшее слово, при делении – частное. Если номер регистра rd равен нулю, запоминается только младшая часть результата, при умножении – младшее слово, при делении – остаток.

Примечание 1. Цветом выделены команды процессора для встраиваемых приложений.

III. Выводы

Предложена система команд для построения процессоров высокой размерности, независимая от длины операндов длиной от 8 до 2048 бит и более и включающая набор команд для встраиваемых процессоров.

Разумеется, в работе не ставилась цель создать «самую лучшую архитектуру», «самый лучший процессор», но данная разработка, возможно, завоюет свою нишу, особенно в приложениях, требующих целочисленных процессоров большой разрядности, для чего, собственно, она и создавалась.

За рамками обсуждения осталось довольно много вопросов, в частности, использование сопроцессоров с плавающей точкой, использование архитектур R^2T в процессорах классов SIMD, VLIW, оптимальная утилизация ресурсов процессоров большой разрядности и другие вопросы, которые будут обсуждены в дальнейшем.

В заключение отметим, что несмотря на довольно большой опыт в разработке процессоров, автор вполне сознает его ограниченность и предлагает сообществу развивать и уточнять систему команд и конфигурации процессоров R^2T.

Литература

[1] The RISC-V Instruction Set Manual. Volume I: User-Level ISA. https://riscv.org/wp-content/uploads/2017/05/riscv-spec-v2.2.pdf. [Electronic resource]. – Mode of access: https://riscv.org. – Date of access: 24.03.2020.

[2] Diamond Standard Processor Cores // ip.cadence.com [Electronic resource]. – Mode of access: https://ip.cadence.com. – Date of access: 24.03.2020.

Instruction Set for Architecture R^2T

V.V. Erokhin

NIIMA PROGRESS JSC, Moscow, vladimir.v.erokhin@gmail.com

Abstract — The instruction set for multi-bit processor architecture (ISA) is described. The ISA set consists of set for "usual" processor and set for embedded applications.

***Keywords* — architecture, processor, instruction set.**

References

1. The RISC-V Instruction Set Manual. Volume I: User-Level ISA. https://riscv.org/wp-content/uploads/2017/05/riscv-spec-v2.2.pdf. [Electronic resource]. – Mode of access: https://riscv.org. – Date of access: 24.03.2020.
2. Diamond Standard Processor Cores // ip.cadence.com [Electronic resource]. – Mode of access: https://ip.cadence.com. – Date of access: 24.03.2020.