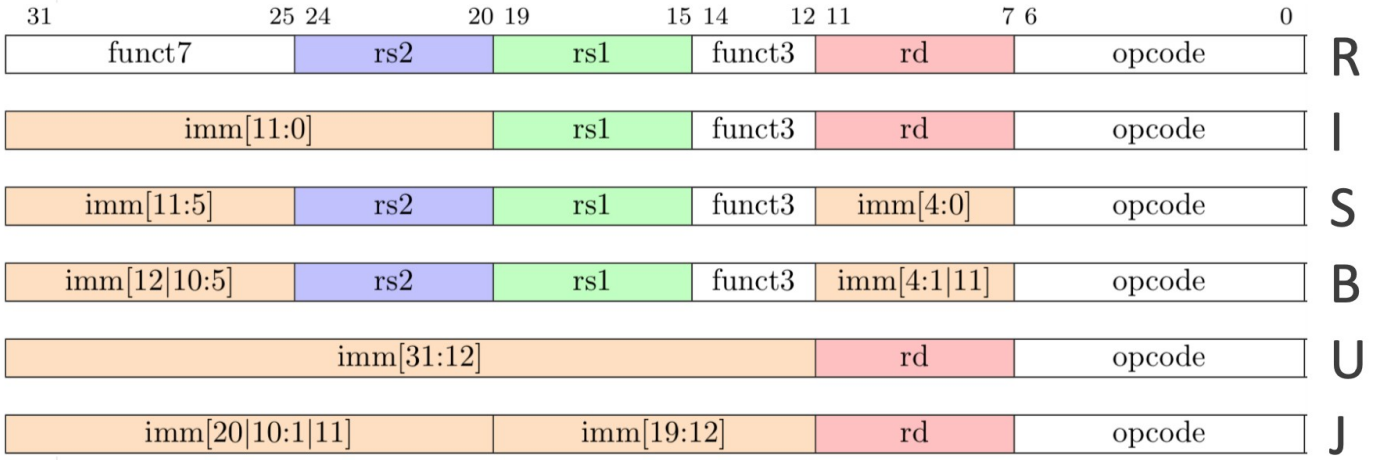
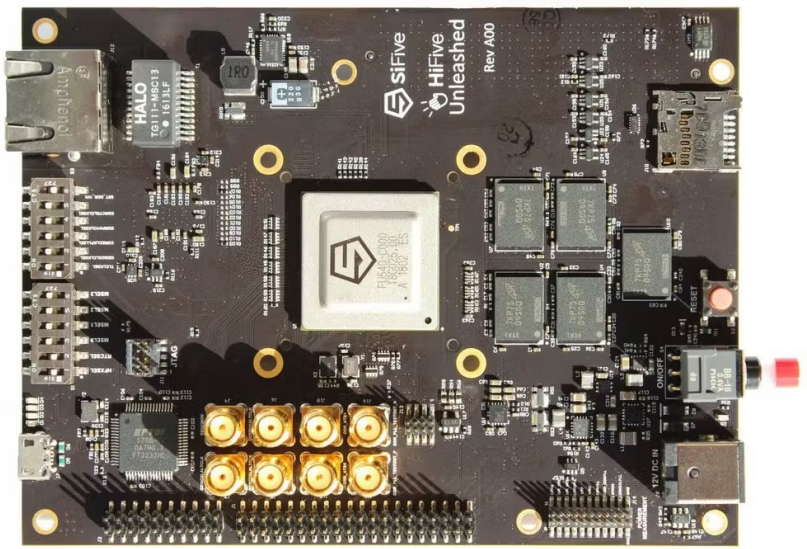
Архитектура процессоров RISC-V





Особенности

RISC-V — расширяемая открытая и свободная система команд и процессорная архитектура на основе концепции RISC для управления микропроцессорами и микроконтроллерами.

Спецификации архитектурных описаний RISC-V свободно доступны и бесплатны для любого использования, включая коммерческие реализации непосредственно в кремнии или для конфигурирования ПЛИС.

Участие в проектировании и обсуждении спецификаций архитектурных описаний открытое.

Система команд имеет зарезервированные в спецификации биты для кодирования расширений без ограничения области применения.

В отличие от других академических проектов, которые обычно сосредоточены на простоте и образовательных целях, системы команд RISC-V проектируется для широкого круга компьютерных применений.

Особенности

Архитектура RISC-V формировалась из желания разработчиков следовать четырем простым принципам:

1) **для простоты придерживайтесь единообразия** (единообразный формат команд);
2) **типичный сценарий должен быть быстрым** (несколько простых инструкций для выполнения более сложных операций);
3) **чем меньше, тем быстрее** (относительно небольшое число регистров);
4) **хорошая разработка требует хороших компромиссов** (четыре формата команд – не один, не много форматов).

(Паттерсон и Хеннесси)

Даты

2010 – исследовательский проект при непосредственном участии Дэвида Паттерсона в Калифорнийском университете Беркли в США.
В настоящее время многие нынешние участники проектов развития RISC-V являются добровольцами, не связанными с университетом.

2015 – создан международный фонд RISC-V и ассоциация со штаб-квартирой в Цюрихе в Швейцарии (*для устойчивого развития, стандартизации и продвижения RISC-V*).

2018 – партнёрство с The Linux Foundation.
В руководство и технические комитеты входят две российские компании разработчики процессорных ядер — Syntacore и CloudBEAR, разработчики Альт Линукс, Астра Линукс.

2022 – корпорация Intel объявила об инвестировании в развитие RISC-V одного миллиарда долларов и вошла в состав руководства RISC-V.

2022 – в РФ образован Альянс RISC-V.

В совете директоров RISC-V ведущие роли занимают китайские компании и организации, включая Китайскую академия наук.

Базовая спецификация «RV32I»

RV — RISC-V, 32-разрядная, I - Integer (целочисленная арифметика)

- 32 регистра
- 39 инструкций.
- 6 форматов инструкций.

Базовые расширения:

- M — целочисленное умножение/деление
- A — атомарные операции с памятью
- F и D — вычисления с плавающей точкой с дополнительным набором регистров (одинарной Float и двойной Double точности)
- C — сжатый формат команд (подмножество RV32I), для удвоения плотности упаковки в машинном слове наиболее востребованных стандартных инструкций

Базовый набор RV32E для встраиваемых систем совпадает по кодированию и набору инструкции с RV32I, но содержит только 16 регистров. Применяется в основном в недорогих микроконтроллерах.

Ряд особенностей системы команд RISC-V

- Обязательное для реализации подмножество инструкций I — (Integer).
- Несколько стандартных опциональных расширений.
- Операции ветвления не используют флагов, а непосредственно сравнивают свои регистровые операнды.
- Минимальный набор регистров: регистр x0 (zero), 31 целочисленный регистр (x1 — x31), регистр счётчика команд (PC), множество CSR (Control and Status Registers, до 4096).
- RV32E (Embedded): 16 регистров общего назначения. Экономия аппаратные ресурсы, сокращение затрат на память, и на время сохранения/восстановления регистров при ***переключениях контекста***.
- При одинаковой кодировке инструкций предусмотрены реализации архитектур с 32-, 64- и 128-битными регистрами и операциями (RV32I, RV64I и RV128I).
- Нет операций над частями регистров, нет выделенных «регистровых пар».
- Операции не сохраняют биты переноса или переполнения (как в Си). Не генерируются исключения по переполнению и делению на 0.
- Целочисленная арифметика расширенной точности должна отдельно обрабатывать старшие разряды результата (умножение – спец. инструкции).

Ряд особенностей системы команд RISC-V

- Инструкции базового набора имеют длину 32 бита с выравниванием на границу 32-битного слова
- В общем формате предусмотрены инструкции различной длины (от 16 до 192 бит с шагом в 16 бит) с выравниванием на границу 16-битного слова. Полная длина инструкции декодируется унифицированным способом из её первого 16-битного слова.
- Для часто используемых инструкций стандартизовано применение их аналогов в 16-битной кодировке (C — Compressed extension).
- Так как кодировка базового набора инструкций не зависит от разрядности архитектуры, то один и тот же код потенциально может запускаться на различных RISC-V архитектурах, определять разрядность и другие параметры текущей архитектуры, наличие расширений системы инструкций, а потом автоконфигурироваться для целевой среды выполнения.
- Спецификацией RISC-V предусмотрено несколько областей в пространстве кодировок инструкций для пользовательских «X-расширений» архитектуры, которые поддерживаются на уровне ассемблера, как группы инструкций custom0 и custom1.

Список различных наборов (систем) команд

Сокращение	Наименование	Версия	Статус
Базовые наборы			
RVWMO	Базовая модель согласованности памяти	2.0	Ratified
RV32I	Базовый набор с целочисленными операциями, 32-битный	2.1	Ratified
RV64I	Базовый набор с целочисленными операциями, 64-битный	2.1	Ratified
RV32E	Базовый набор с целочисленными операциями для встраиваемых систем, 32-битный, 16 регистров	1.9	Draft
RV128I	Базовый набор с целочисленными операциями, 128-битный	1.7	Draft
Часть 1 Стандартные непривилегированные наборы команд			
M	Целочисленное умножение и деление (Integer Multiplication and Division)	2.0	Ratified
A	Атомарные операции (Atomic Instructions)	2.1	Ratified
F	Арифметические операции с плавающей запятой над числами одинарной точности (Single-Precision Floating-Point)	2.2	Ratified
D	Арифметические операции с плавающей запятой над числами двойной точности (Double-Precision Floating-Point)	2.2	Ratified
Q	Арифметические операции с плавающей запятой над числами четверной точности	2.2	Ratified
C	Сокращённые имена для команд (Compressed Instructions)	2.2	Ratified

Список различных наборов (систем) команд

Сокращение	Наименование	Версия	Статус
Часть 1 Стандартные непривилегированные наборы команд			
Counters	Инструкции для счетчиков производительности и таймеров — наборы **Zicntr** и **Zihpm**	2.0	Draft
L	Арифметические операции над десятичными числами с плавающей запятой (Decimal Floating-Point)	0.0	Open
B	Битовые операции (Bit Manipulation)	0.36	Open
J	Двоичная трансляция и поддержка динамической компиляции (Dynamically Translated Languages)	0.0	Open
T	Транзакционная память (Transactional Memory)	0.0	Open
P	Короткие SIMD-операции (Packed-SIMD Instructions)	0.1	Open
V	Векторные расширения (Vector Operations)	1.0	Frozen
Zicsr	Инструкции для работы с контрольными и статусными регистрами (Control and Status Register (CSR) Instructions)	2.0	Ratified
Zifencei	Инструкции синхронизации пототоков команд и данных (Instruction-Fetch Fence)	2.0	Ratified
Zihintpause	Pause Hint	2.0	Ratified
Zihintntl	Non-Temporal Locality Hints	0.2	Draft
Zam	Расширение для смещённых атомарных операций (Extension for Misaligned Atomics)	0.1	Draft

Список различных наборов (систем) команд

Сокращение	Наименование	Версия	Статус
Часть 1 Стандартные непривилегированные наборы команд			
Zfh	Extensions for Half-Precision Floating-Point	1.0	Ratified
Zfhmin	Extensions for Half-Precision Floating-Point	1.0	Ratified
Zfinx	Standard Extensions for Floating-Point in Integer Registers	1.0	Ratified
Zdinx	Standard Extensions for Floating-Point in Integer Registers	1.0	Ratified
Zhinx	Standard Extensions for Floating-Point in Integer Registers	1.0	Ratified
Zhinxmin	Standard Extensions for Floating-Point in Integer Registers	1.0	Ratified
Ztso	Расширение для модели согласованности памяти RVTSO (Extension for Total Store Ordering)	0.1	Frozen
G	**= IMAFD Zicsr Zifencei** Обобщенное/сокращёное обозначение для набора расширений	н/д	н/д

Список различных наборов (систем) команд

Сокращение	Наименование	Версия	Статус
Часть 2 Стандартные наборы команд для привилегированных режимов			
Machine ISA	Инструкции аппаратного уровня	1.12	Ratified
Supervisor ISA	Инструкции уровня супервизора	1.12	Ratified
Svnapot Extension	(Extension for NAPOT Translation Contiguity)	1.0	Ratified
Svpbmt Extension	(Extension for Page-Based Memory Types)	1.0	Ratified
Svinval Extension	(Extension for Fine-Grained Address-Translation Cache Invalidation)	1.0	Ratified
Hypervisor ISA	Инструкции уровня гипервизора	1.0	Ratified

Форматы команд

Тип	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Регистр/регистр	funct7							rs2					rs1					funct3			rd					код операции					1	1
С операндом	±	imm[10:0]											rs1					funct3			rd					код операции					1	1
С длинным операндом	±	imm[30:12]																			rd					код операции					1	1
Сохранение	±	imm[10:5]						rs2					rs1					funct3			imm[4:0]					код операции					1	1
Ветвление	±	imm[10:5]						rs2					rs1					funct3			imm[4:1]				[11]	код операции					1	1
Переход	±	imm[10:1]										[11]	imm[19:12]								rd					код операции					1	1

Регистры. Соглашения о псевдонимах

Имя регистра в RISC-V	Имя в EABI	Имя в psABI	Описание в psABI	Кто сохраняет в psABI
32 целочисленных регистра				
x0	zero	zero	Всегда ноль	
x1	ra	ra	Адрес возврата (return address)	Вызывающий
x2	sp	sp	Указатель стека (stack pointer)	Вызываемый
x3	gp	gp	Глобальный указатель (global pointer)	
x4	tp	tp	Потоковый указатель (thread pointer)	
x5	t0	t0	Temporary / альтернативный адрес возврата	Вызывающий
x6	s3	t1	Temporary	Вызывающий
x7	s4	t2	Temporary	Вызывающий
x8	s0/fp	s0/fp	Saved register / frame pointer	Вызываемый
x9	s1	s1	Saved register	Вызываемый
x10	a0	a0	Аргумент (argument) / возвращаемое значение	Вызывающий
x11	a1	a1	Аргумент (argument) / возвращаемое значение	Вызывающий
x12	a2	a2	Аргумент (argument)	Вызывающий
x13	a3	a3	Аргумент (argument)	Вызывающий
x14	s2	a4	Аргумент (argument)	Вызывающий
x15	t1	a5	Аргумент (argument)	Вызывающий
x16	s5	a6	Аргумент (argument)	Вызывающий
x17	s6	a7	Аргумент (argument)	Вызывающий
x18-27	s7-16	s2-11	Saved register	Вызываемый
x28-31	s17-31	t3-6	Temporary	Вызывающий

32 дополнительных регистра с плавающей точкой				
f0-7		ft0-7	Floating-point temporaries	Вызывающий
f8-9		fs0-1	Floating-point saved registers	Вызываемый
f10-11		fa0-1	Floating-point arguments/return values	Вызывающий
f12-17		fa2-7	Floating-point arguments	Вызывающий
f18-27		fs2-11	Floating-point saved registers	Вызываемый
f28-31		ft8-11	Floating-point temporaries	Вызывающий

Регистры

Базовые регистры расширения I
(в расширении E только x0 – x15)

x0 Zero	x1 ra	x2 sp	x3 gp
x4 tp	x5 t0	x6 s3/t1	x7 s4/t2
x8 s0/fp	x9 s1	x10 a0	x11 a1
x12 a2	x13 a3	x14 s2/a4	x15 t1/a5
x16 s5/a6	x17 s6/a7	x18 s7/s2	x19 s8/s3
x20 s9/s4	x21 s10/s5	x22 s11/s6	x23 s12/s7
x24 s13/s8	x25 s14/s9	x26 s15/	x27 s16/
x28 s17/t3	x29 s18/t4	x30 s19/t5	x31 s20/t6

Вещественные регистры
расширения F, D, Q, L

f0 ft0	f1 ft1	f2 ft2	f3 ft3
f4 ft4	f5 ft5	f6 ft6	f7 ft7
f8 fs0	f9 fs1	f10 fa1-0	f11 fa1-1
f12 fa2	f13 fa3	f14 fa4	f15 fa5
f16 fa6	f17 fa7	f18 fs2	f19 fs3
f20 fs4	f21 fs5	f22 fs6	f23 fs7
f24 fs8	f25 fs9	f26 fs10	f27 fs11
f28 ft3	f29 tf4	f30 tf5	f31 ft6

Векторные регистры
расширение V

v0	v1	v2	v3
v4	v5	v6	v7
v8	v9	v10	v11
v12	v13	v14	v15
v16	v17	v18	v19
v20	v21	v22	v23
v24	v25	v26	v27
v28	v29	v30	v31

vl	vtype

Легенда цветов ячеек

x0	глобальный
x1	сохр. вызывающий
x2	сохр. вызываемый

Легенда содержания ячеек

ISA	числитель
EABI / psABI	знаменатель

32 (или 16 для встраиваемых систем) целочисленных регистра. При реализации вещественных групп команд 32 регистра.

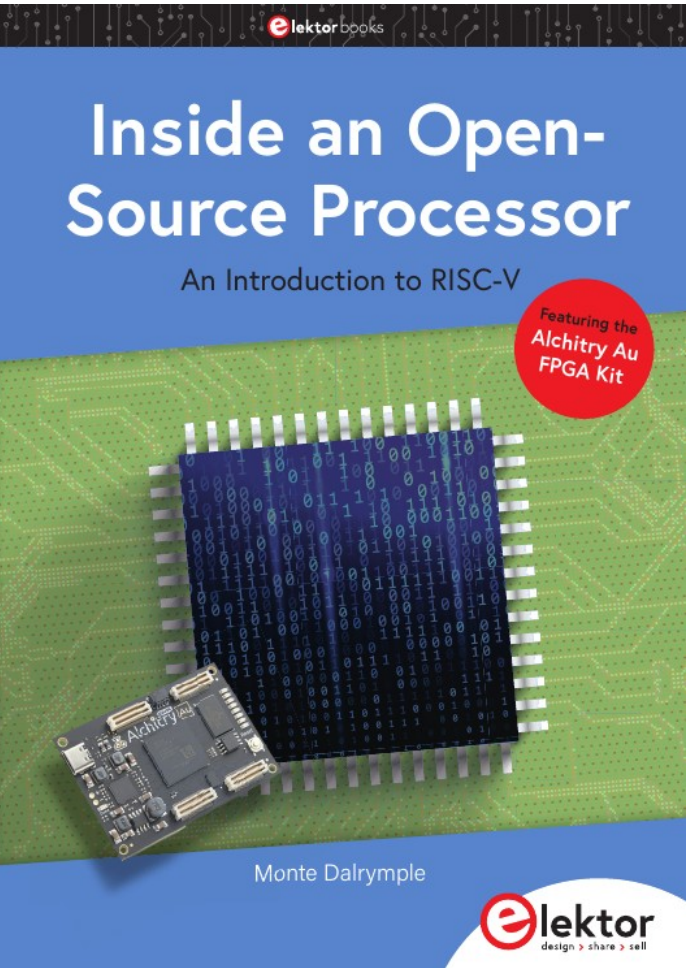
Для операций над числами с плавающей запятой 32 регистра FPU (Floating Point Unit), которые используются совместно разными расширениями базового набора инструкций для трёх вариантов точности: одинарной — 32 бита (F extension), двойной — 64 бита (D — Double precision extension), четверной — 128 бит (Q — Quadruple precision extension).

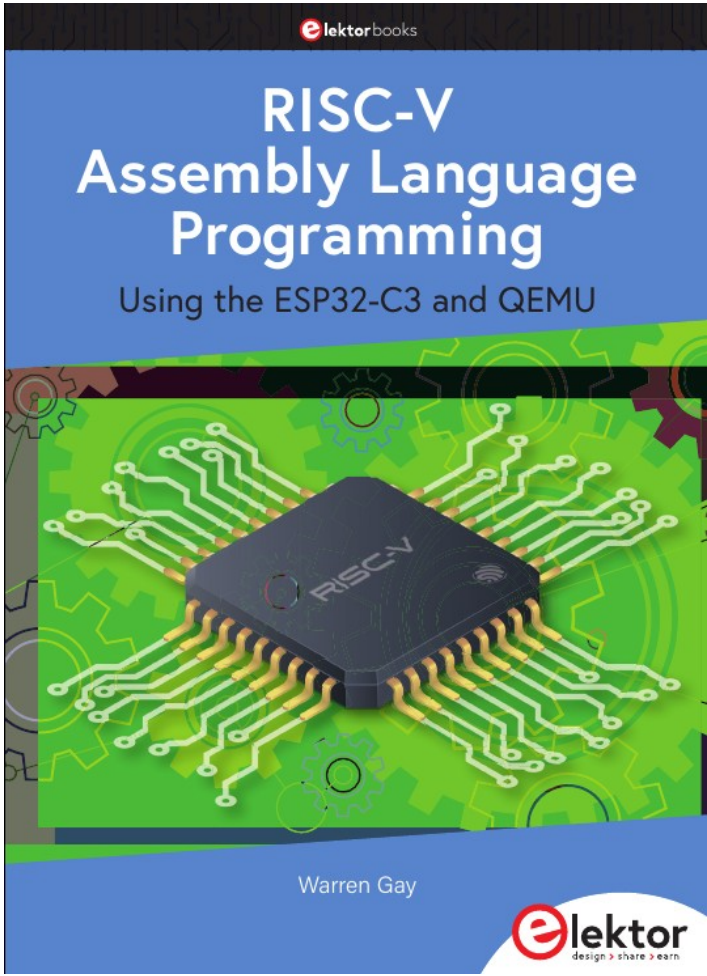
Дополнительно: 32 векторных регистра с векторами переменной длины, (указывается в регистре vlenb блока CSR).

Используемые источники

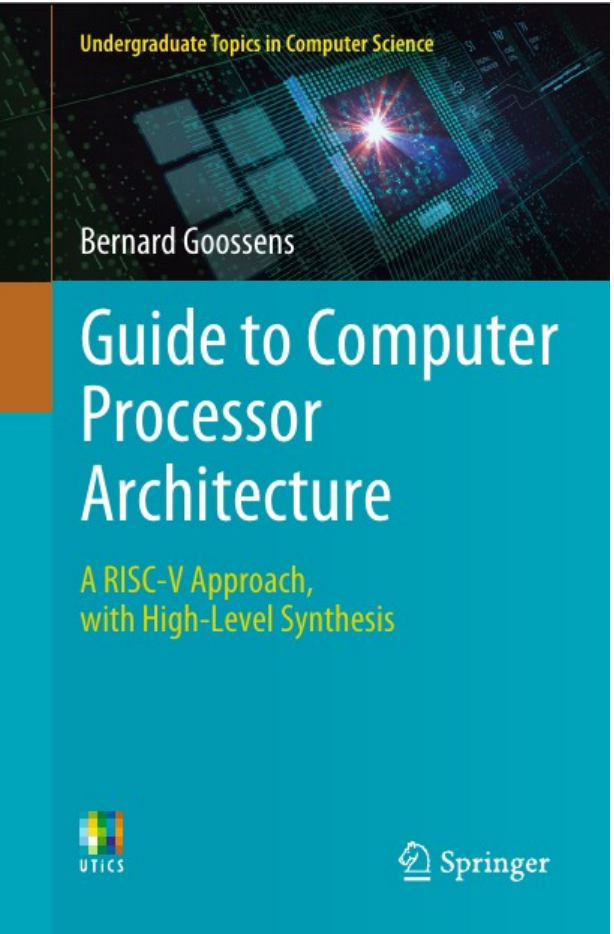
Сара Л. Харрис, Дэвид Харрис
Цифровая схемотехника и архитектура компьютера: RISC-V –
М.: ДМК Пресс, 2021. – 810 с.

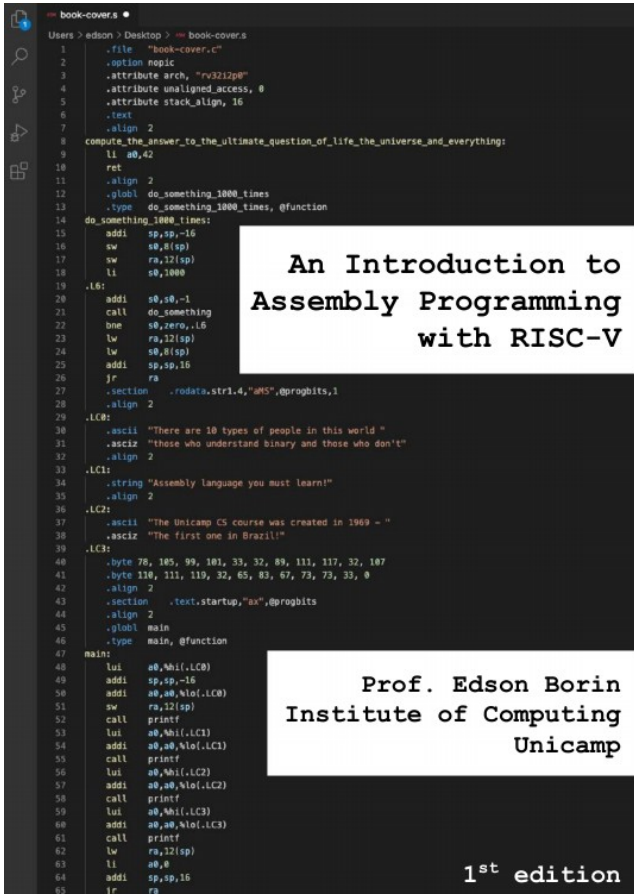
Используемые источники





Используемые источники





https://riscv-programming.org/

Используемые источники

Википедия. RISC-V:
https://ru.wikipedia.org/wiki/RISC-V

Эмуляция с использованием языков программирования C и C++

RISC-V GNU Compiler Toolchain:
https://github.com/riscv-collab/riscv-gnu-toolchain

Spike RISC-V ISA Simulator:
https://github.com/riscv-software-src/riscv-isa-sim

RISC-V Proxy Kernel and Boot Loader
https://github.com/riscv-software-src/riscv-pk

Примеры командной строки

```c
#include <stdio.h>

int func(int x)
{
  return x + 10;
}

int main()
{
  int r = func (10);
  printf("r = %d\n", r);
  return r+1;
}
```

Компиляция и запуск в 64-разрядном режиме

```
> riscv64-unknown-elf-gcc main.c -o main

> spike pk main
```

```
bbl loader
r = 20
```

Примеры командной строки

Получение ассемблерного кода в 64-разрядном режиме

```
> riscv64-unknown-elf-gcc -S main.c -o main.s
```

```c
#include <stdio.h>

int func(int x)
{
  return x + 10;
}

int main()
{
  int r = func (10);
  printf("r = %d\n", r);
  return r+1;
}
```

```
1   .file "main.c"
2   .option nopic
3   .attribute arch, "rv64i2p0_m2p0
4   .attribute unaligned_access, 0
5   .attribute stack_align, 16
6   .text
7   .align 1
8   .globl func
9   .type func, @function
10  func:
11  addi sp,sp,-32
12  sd s0,24(sp)
13  addi s0,sp,32
14  mv a5,a0
15  sw a5,-20(s0)
16  lw a5,-20(s0)
17  addiw a5,a5,10
18  sext.w a5,a5
19  mv a0,a5
20  ld s0,24(sp)
21  addi sp,sp,32
22  jr ra
23  .size func, .-func
24  .section .rodata
25  .align 3
26  .LC0:
27  .string "r = %d\n"
28  .text
29  .align 1
30  .globl main
31  .type main, @function
32  main:
33  addi sp,sp,-32
34  sd ra,24(sp)
35  sd s0,16(sp)
36  addi s0,sp,32
37  li a0,10
```

Примеры командной строки

Получение ассемблерного кода в 32-разрядном режиме

```
$ riscv64-unknown-elf-gcc -mabi=ilp32 -march=rv32i -S main.c -o main.s
$ riscv64-unknown-elf-as -mabi=ilp32 -march=rv32i main.s -o main.o
$ riscv64-unknown-elf-ld -m elf32lriscv main.o -o main
```

Преобразование ассемблерных RISC-V программ, полученных с использованием `riscv64-unknown-elf-gcc` в ассемблерные программы для эмулятора RARS