Overview of

RISC-V Bitmanip Extension

By SiYu Wu @ PLCT

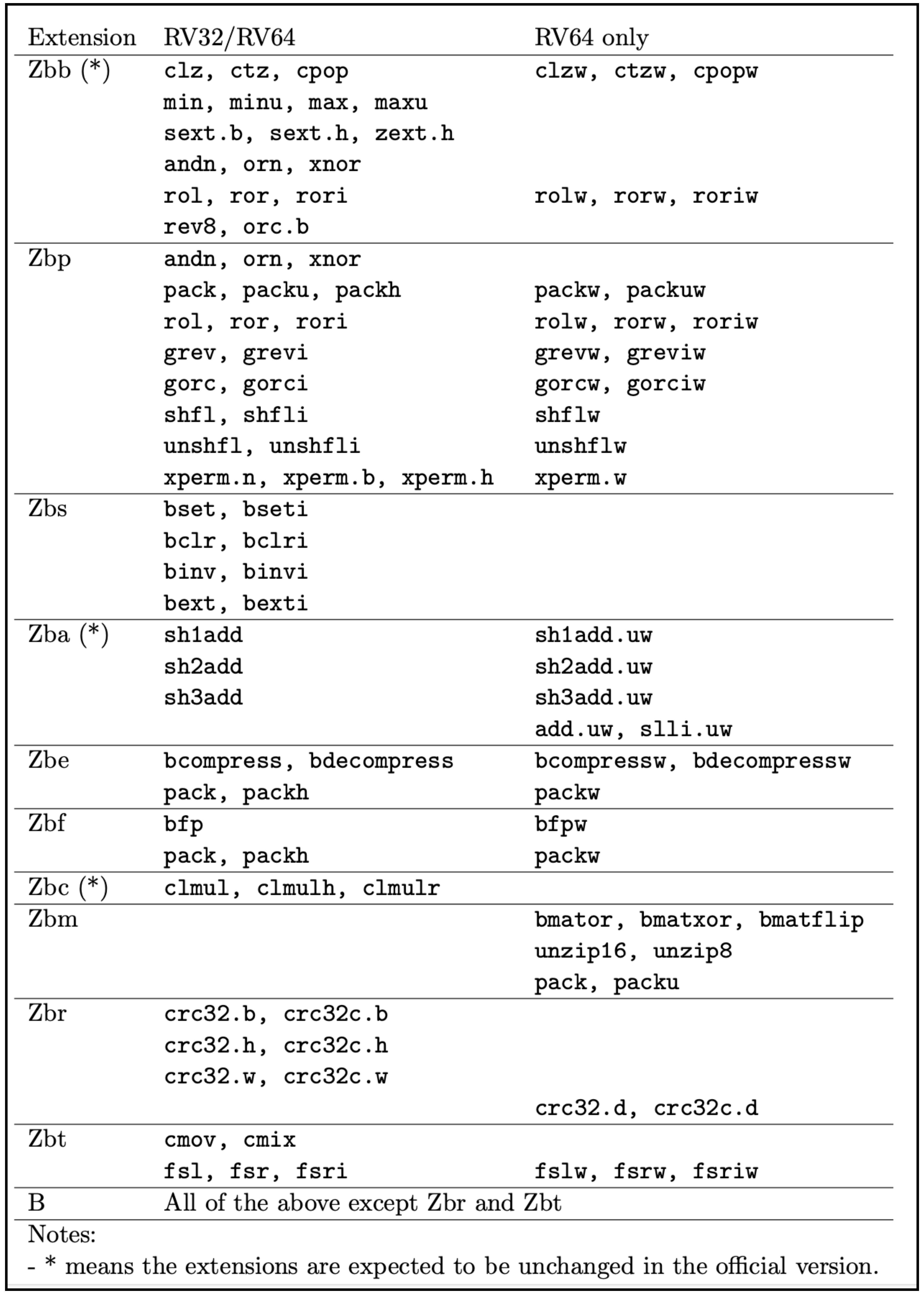
Jan 2021

INDEX

1. Introduction

Introduction

- Bitmanip == bit manipulation
  The extension to accelerate bit manipulation
- github.com/riscv/riscv-bitmanip
- Mainly include：
  - Basic bit manipulation
  - Bit permutation instructions
  - Misc / Other instructions (shows later)



Extension	RV32/RV64	RV64 only
Zbb (*)	`clz, ctz, cpop`	`clzw, ctzw, cpopw`
	`min, minu, max, maxu`	
	`sext.b, sext.h, zext.h`	
	`andn, orn, xnor`	
	`rol, ror, rori`	`rolw, rorw, roriw`
	`rev8, orc.b`	
Zbp	`andn, orn, xnor`	
	`pack, packu, packh`	`packw, packuw`
	`rol, ror, rori`	`rolw, rorw, roriw`
	`grev, grevi`	`grevw, greviw`
	`gorc, gorci`	`gorcw, gorciw`
	`shfl, shfli`	`shflw`
	`unshfl, unshfli`	`unshflw`
	`xperm.n, xperm.b, xperm.h`	`xperm.w`
Zbs	`bset, bseti`	
	`bclr, bclri`	
	`binv, binvi`	
	`bext, bexti`	
Zba (*)	`sh1add`	`sh1add.uw`
	`sh2add`	`sh2add.uw`
	`sh3add`	`sh3add.uw`
		`add.uw, slli.uw`
Zbe	`bcompress, bdecompress`	`bcompressw, bdecompressw`
	`pack, packh`	`packw`
Zbf	`bfp`	`bfpw`
	`pack, packh`	`packw`
Zbc (*)	`clmul, clmulh, clmulr`	
Zbm		`bmator, bmatxor, bmatflip`
		`unzip16, unzip8`
		`pack, packu`
Zbr	`crc32.b, crc32c.b`	
	`crc32.h, crc32c.h`	
	`crc32.w, crc32c.w`	
		`crc32.d, crc32c.d`
Zbt	`cmov, cmix`	
	`fsl, fsr, fsri`	`fslw, fsrw, fsriw`
B	All of the above except Zbr and Zbt	

Notes:
- * means the extensions are expected to be unchanged in the official version.

Design Criteria

- **Architecture Consistency**: Consistent with RISC-V philosophy.
- **Threshold Metric**: replace at least 3 instructions. (2 may be considered)
- **Data-Driven Value**: only have theoretical value will not be accepted, unless it is used in the real world.
- **Hardware Simplicity**: had better to be cheap and easy to implement
- **Compiler Support**: ISA changes that can be natively detected by the compiler, or are already used as intrinsics, will score higher than instructions which do not fit that criteria.

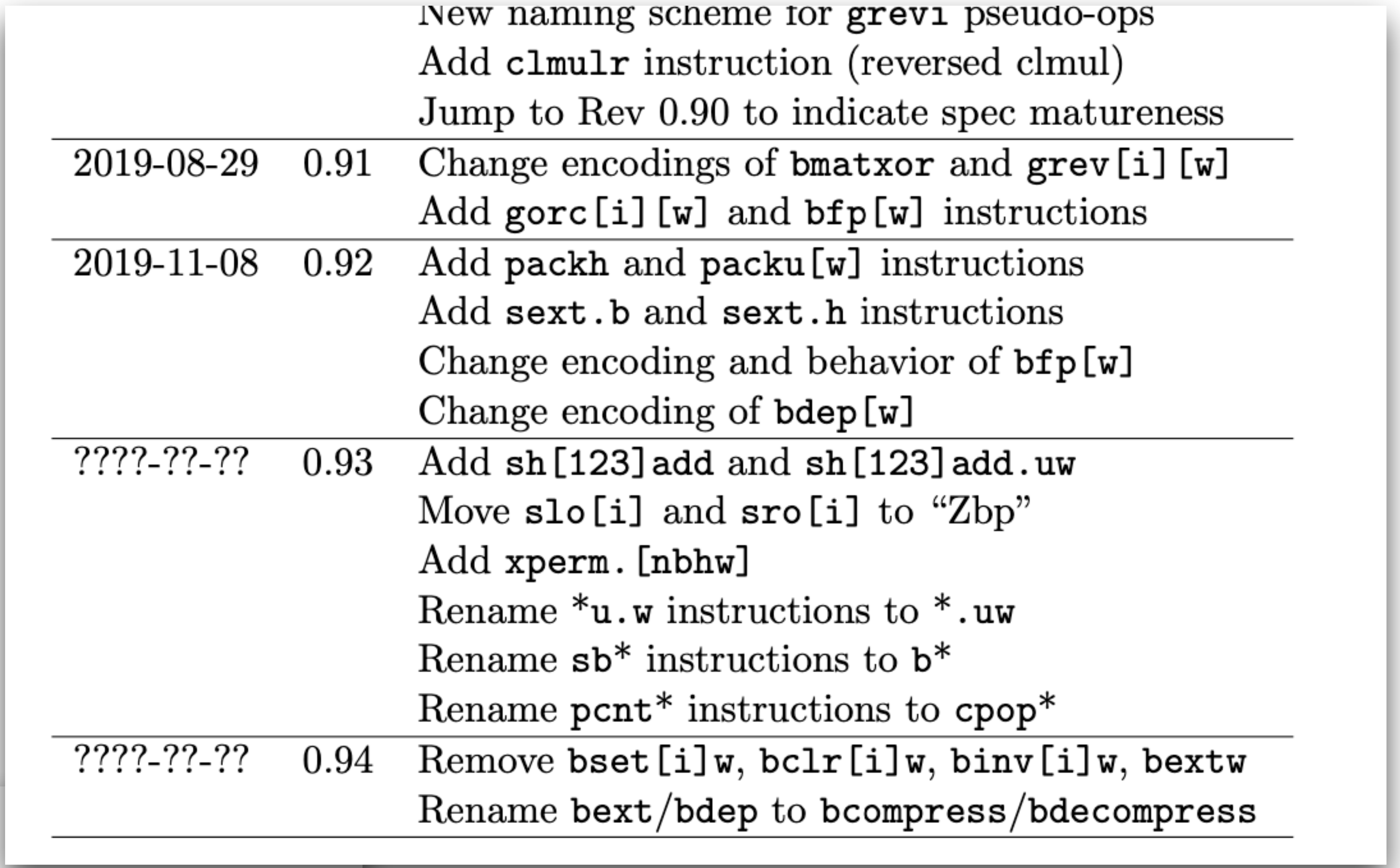
Common Usage

- 使用一条指令实现原本需要2～3条或更多指令的常用位操作
  - 基本位操作
    - 加载64位常数，位段操作，两整数平均，min/max，奇偶校验，检测整数溢出，bitmap操作，计算末尾非0字节数（优化strlen()，strcpy()）等
  - 向量包装（如：4个char压入1个4 byte word里）
  - Funnel shifts（两操作数前后链接，移位，取其upper或lower的一半）
  - 任意位排列组合（如：位翻转、大端序转小端序等）
  - 单bit操作、bitboards操作、位矩阵操作、Xorshift随机数生成、CRC计算

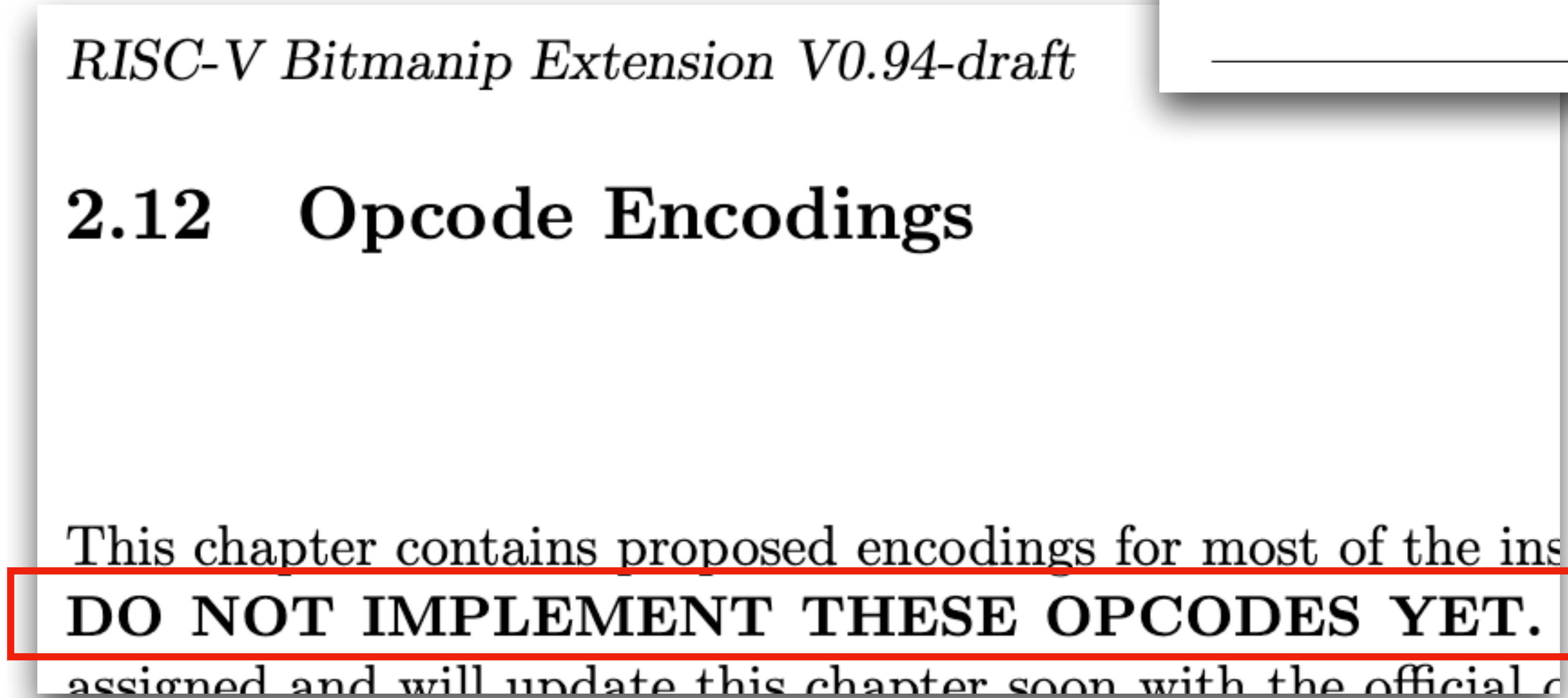
Current status

(for 2020/01/26)

- draft - still in active update
- Opcode not fully confirmed yet



		New naming scheme for **grevi** pseudo-ops
		Add `clmulr` instruction (reversed clmul)
		Jump to Rev 0.90 to indicate spec matureness
2019-08-29	0.91	Change encodings of `bmatxor` and `grev[i][w]`
		Add `gorc[i][w]` and `bfp[w]` instructions
2019-11-08	0.92	Add `packh` and `packu[w]` instructions
		Add `sext.b` and `sext.h` instructions
		Change encoding and behavior of `bfp[w]`
		Change encoding of `bdep[w]`
????-??-??	0.93	Add `sh[123]add` and `sh[123]add.uw`
		Move `slo[i]` and `sro[i]` to "Zbp"
		Add `xperm.[nbhw]`
		Rename *`u.w` instructions to *`.uw`
		Rename `sb`* instructions to `b`*
		Rename `pcnt`* instructions to `cpop`*
????-??-??	0.94	Remove `bset[i]w`, `bclr[i]w`, `binv[i]w`, `bextw`
		Rename `bext`/`bdep` to `bcompress`/`bdecompress`



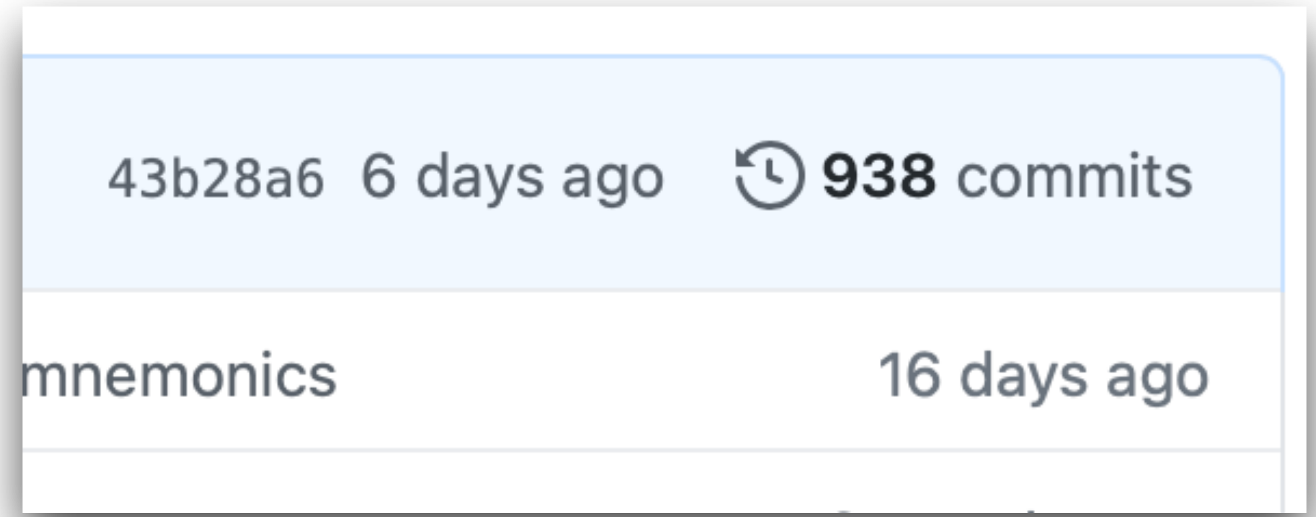
RISC-V Bitmanip Extension V0.94-draft

2.12 Opcode Encodings

This chapter contains proposed encodings for most of the ins

DO NOT IMPLEMENT THESE OPCODES YET.

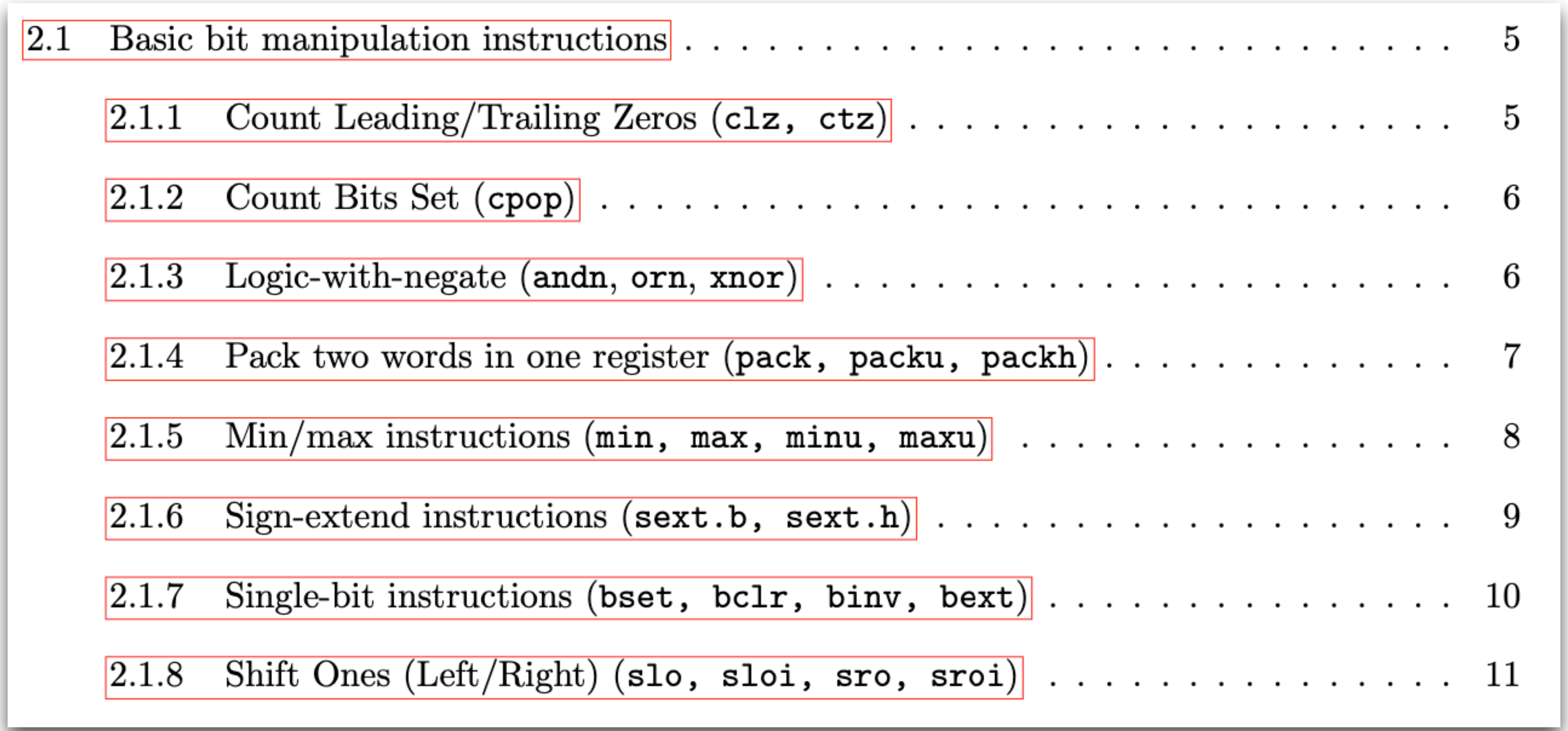
assigned and will update this chapter soon with the official c



43b28a6 6 days ago **938** commits

mnemonics 16 days ago

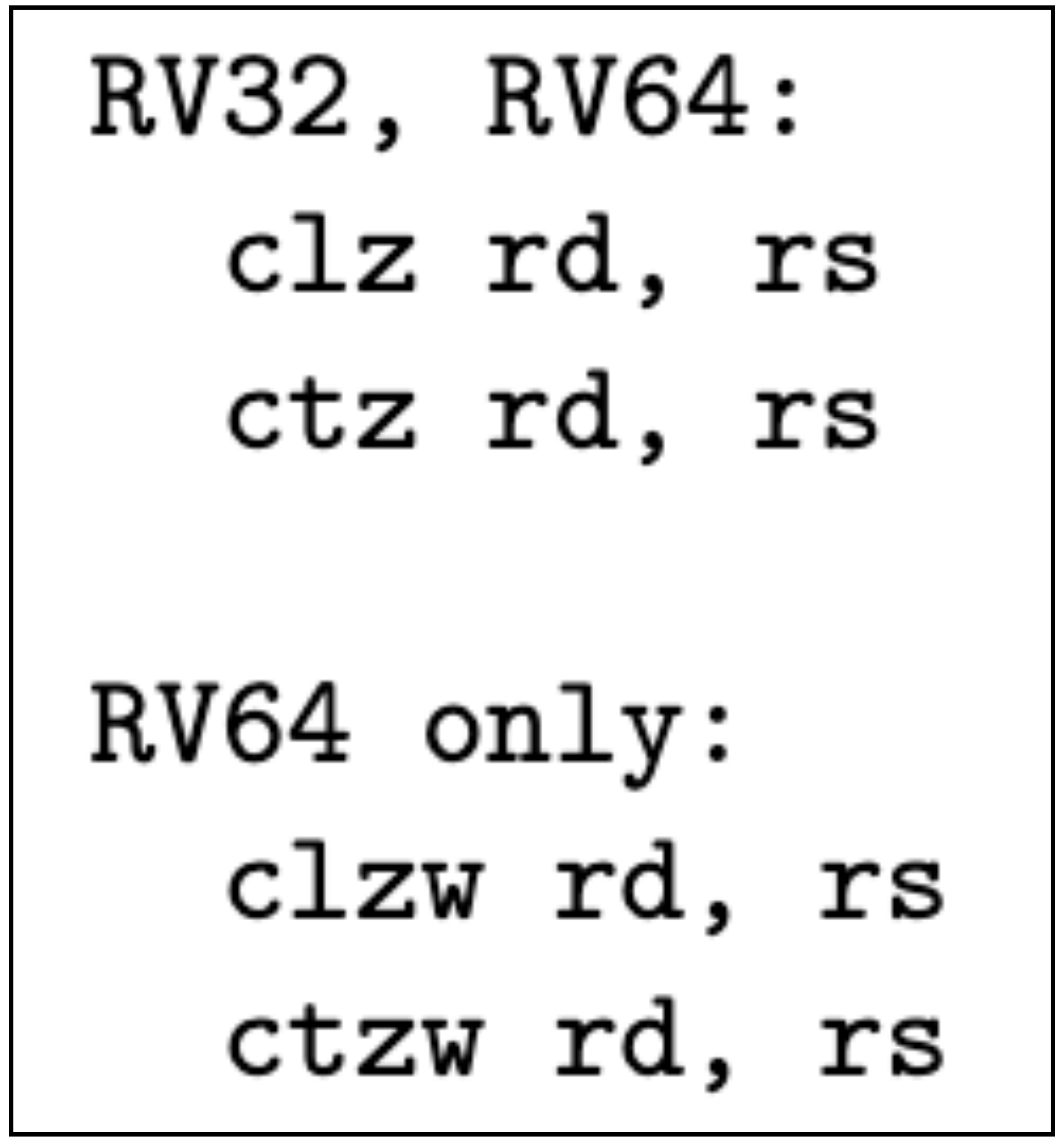
2.1. Basic bit manipulation instructions



Count Leading/Trailing Zeros

`clz, ctz`

- clz：前导0位数
- ctz：后导0位数



```
RV32, RV64:
    clz rd, rs
    ctz rd, rs

RV64 only:
    clzw rd, rs
    ctzw rd, rs
```

Count Bits Set

cpop

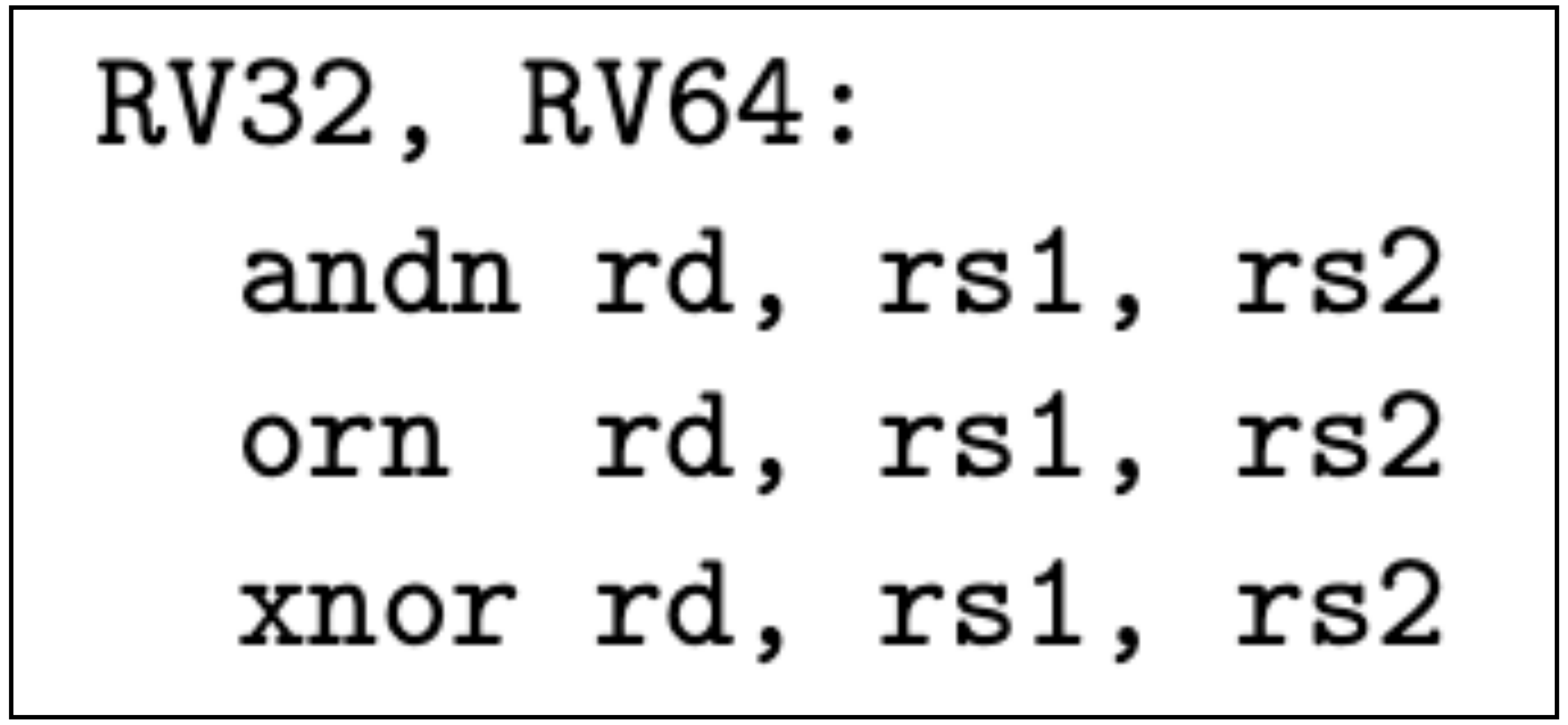
- 统计为1的位数

```
RV32, RV64:
  cpop rd, rs

RV64 only:
  cpopw rd, rs
```

Logic-with-negate

andn, orn, xnor

- 先对rs2取反，再and、or、xor



```
RV32, RV64:
   andn rd, rs1, rs2
   orn  rd, rs1, rs2
   xnor rd, rs1, rs2
```

Pack two words in one register

pack，packu，packh

- pack：分别截取rs1和rs2的**低**1/2，拼接为结果（rs2在高位，rs1在低位）
- packu：分别截取rs1和rs2的**高**1/2，拼接为结果（rs2在高位，rs1在低位）
- packh：分别截取rs1和rs2的**低8位**
  拼接为结果的低16位（rs2在高位，rs1在低位）
  其余为0

```
RV32, RV64:
  pack  rd, rs1, rs2
  packu rd, rs1, rs2
  packh rd, rs1, rs2

RV64 only:
  packw  rd, rs1, rs2
  packuw rd, rs1, rs2
```

Min/max instructions

`min, max, minu, maxu`

- 结果为两数中较大/较小的（u：视为无符号数）
- 可以避免产生分支影响流水线性能

```
RV32, RV64:
  min  rd, rs1, rs2
  max  rd, rs1, rs2
  minu rd, rs1, rs2
  maxu rd, rs1, rs2
```

Single-bit instructions

`bset, bclr, binv, bext`

- 对rs1对第<rs2/imm>位进行操作，结果写入rd
  - bset：单bit置位
  - bclr：单bit清除
  - binv：单bit翻转
  - bext：单bit读取

```
RV32, RV64:
  bset  rd, rs1, rs2
  bclr  rd, rs1, rs2
  binv  rd, rs1, rs2
  bext  rd, rs1, rs2
  bseti rd, rs1, imm
  bclri rd, rs1, imm
  binvi rd, rs1, imm
  bexti rd, rs1, imm

RV64:
  bsetw  rd, rs1, rs2
  bclrw  rd, rs1, rs2
  binvw  rd, rs1, rs2
  bextw  rd, rs1, rs2
  bsetiw rd, rs1, imm
  bclriw rd, rs1, imm
  binviw rd, rs1, imm
```

Shift Ones (Left/Right)

`slo, sloi, sro, sroi`

- 补1移位
  - 类似基础的逻辑移位指令，但移位时补1，而非补0

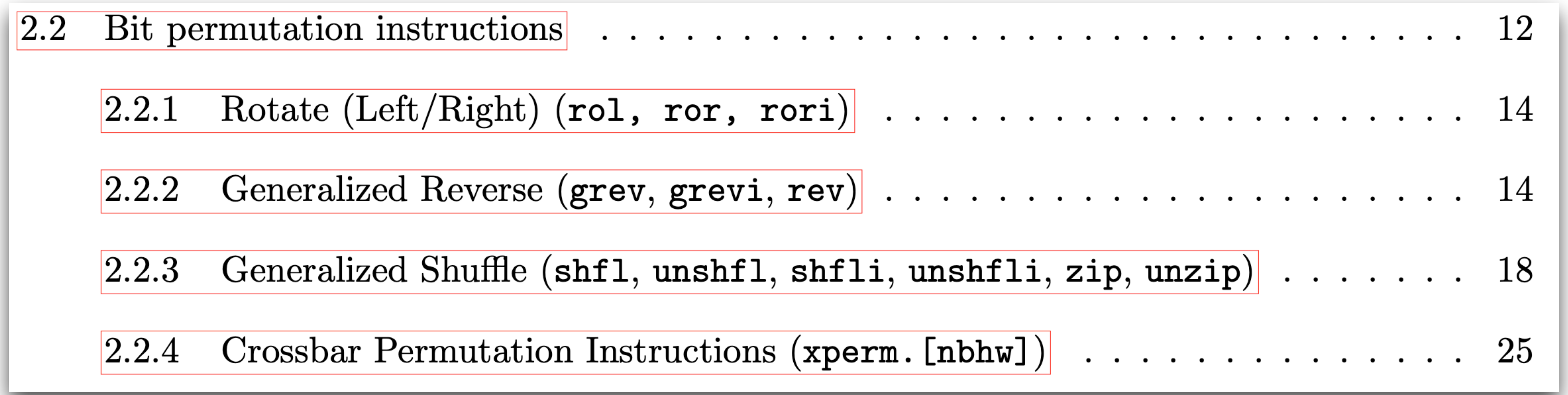
```
RV32, RV64:
  slo  rd, rs1, rs2
  sro  rd, rs1, rs2
  sloi rd, rs1, imm
  sroi rd, rs1, imm

RV64 only:
  slow  rd, rs1, rs2
  srow  rd, rs1, rs2
  sloiw rd, rs1, imm
  sroiw rd, rs1, imm
```

2.2. Bit permutation instructions

Bit permutation

位“重排列”

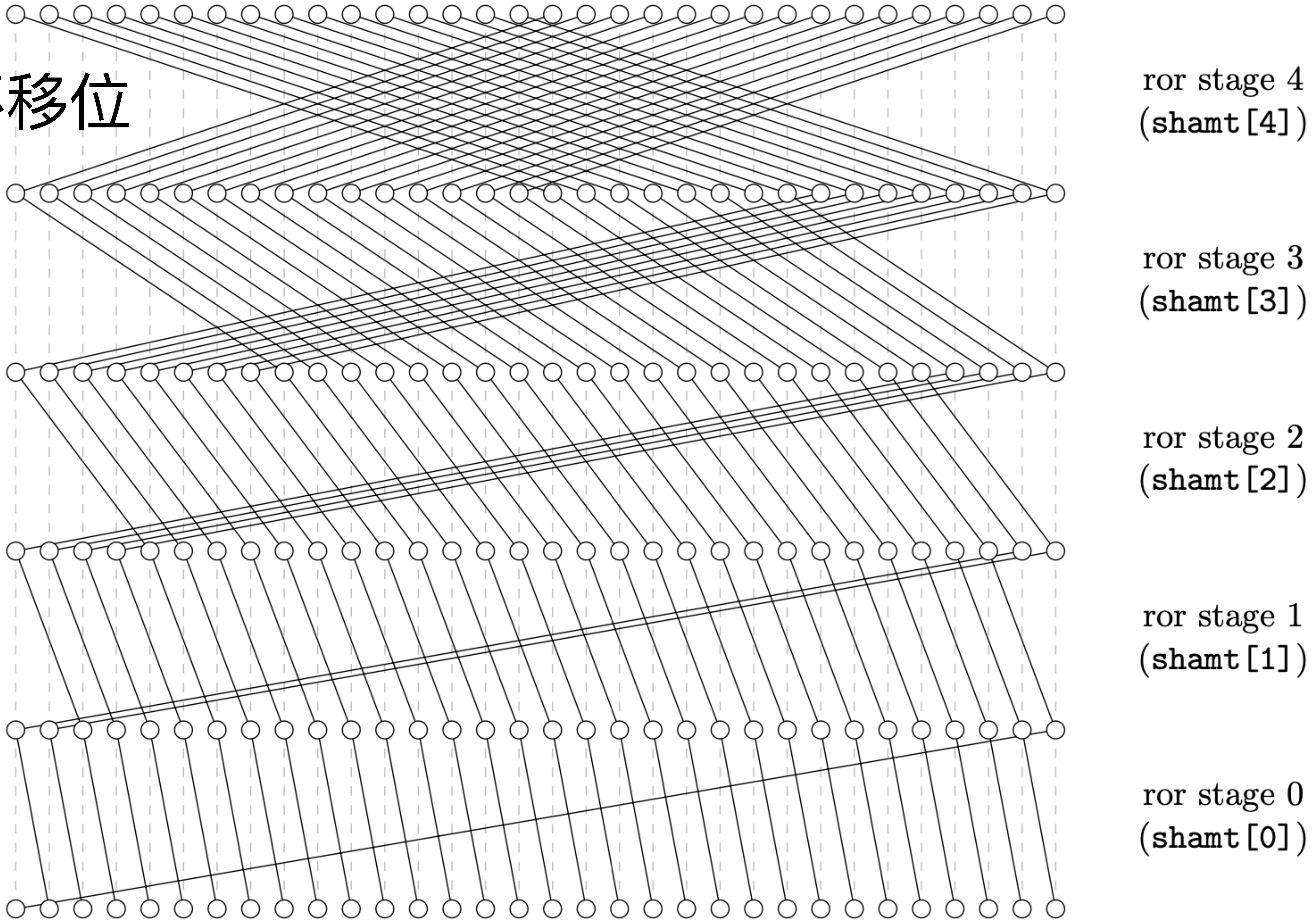
- 对于每位的“地址（位置）”施加一个可逆函数，在新的地址上放入原本的值
- 例如：循环移位： $i'_{\text{rot}} := i \pm k \mod \text{XLEN}$



Rotate (Left/Right)

`rol, ror, rori`

- 循环移位

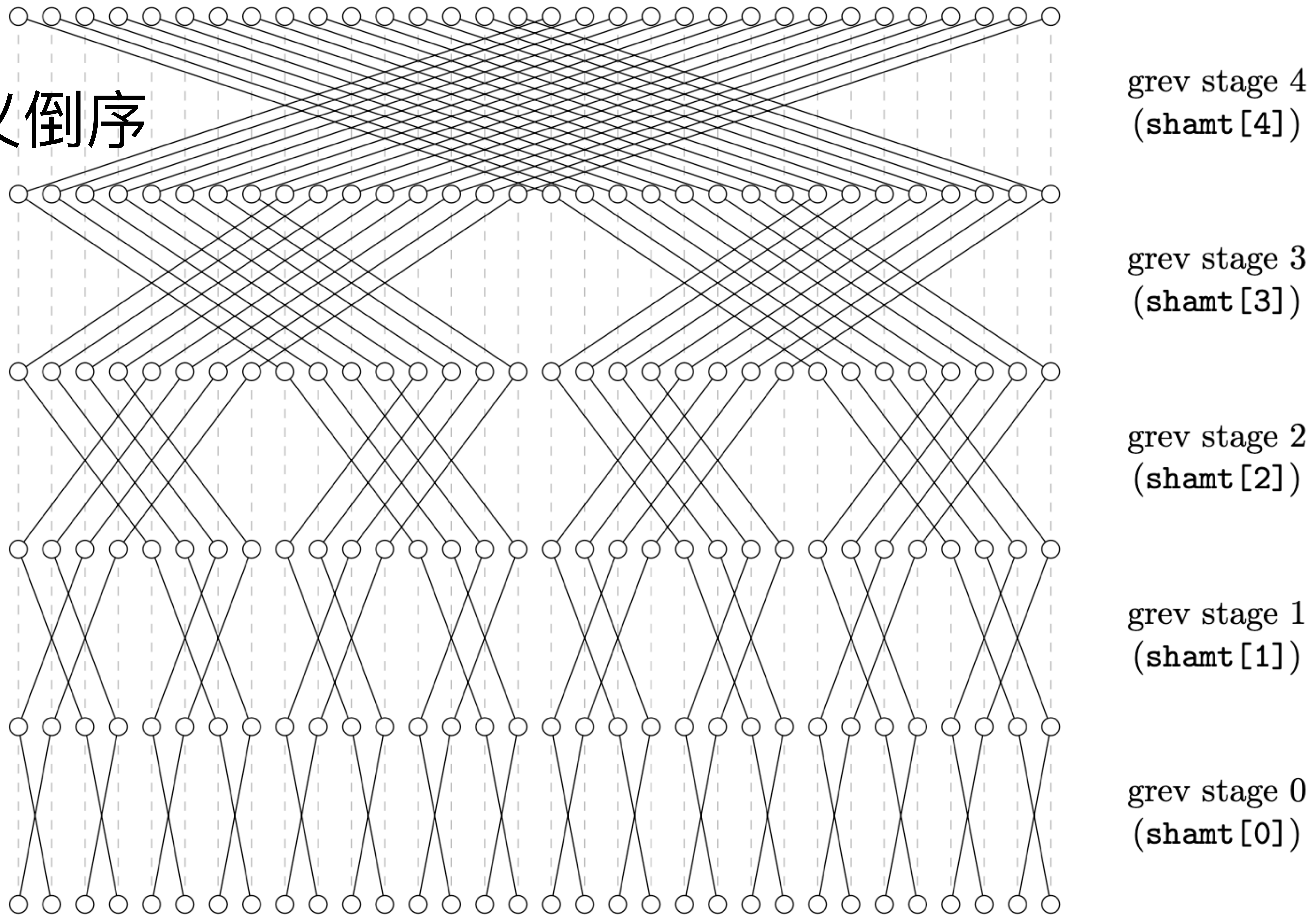


```
RV32, RV64:
  ror  rd, rs1, rs2
  rol  rd, rs1, rs2
  rori rd, rs1, imm

RV64 only:
  rorw  rd, rs1, rs2
  rolw  rd, rs1, rs2
  roriw rd, rs1, imm
```

Generalized Reverse

`grev, grevi, rev`

- 广义倒序



```
RV32, RV64:
  grev  rd, rs1, rs2
  grevi rd, rs1, imm

RV64 only:
  grevw  rd, rs1, rs2
  greviw rd, rs1, imm
```

Generalized OR-Combine

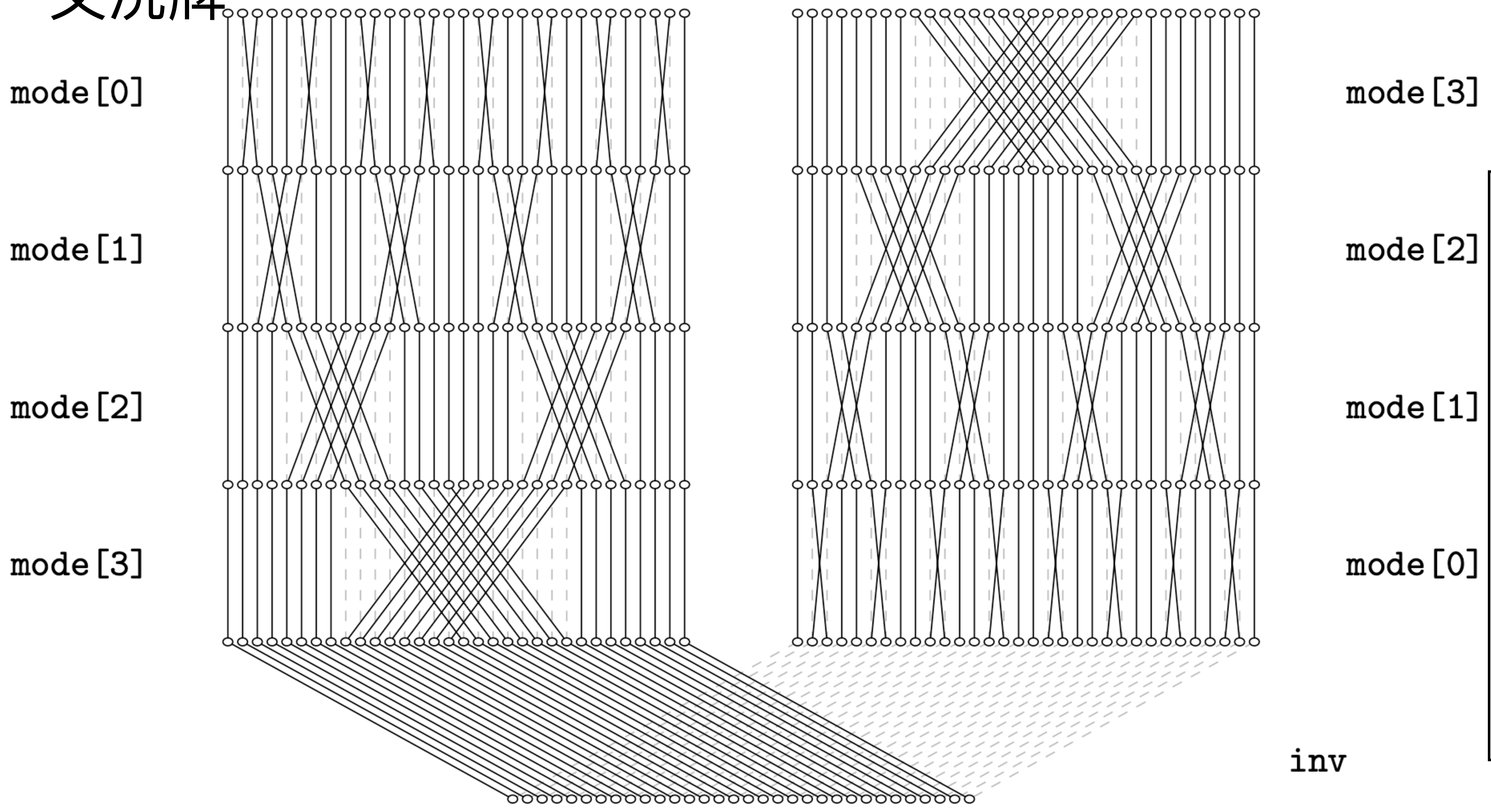
`gorc, gorci`

- 类似广义倒序，但对每对bit进行或操作，而非交换
  - 可用来检测某些位置上的bit是否都为0

Generalized Shuffle

shfl, unshfl, shfli, unshfli, zip, unzip

- 广义洗牌

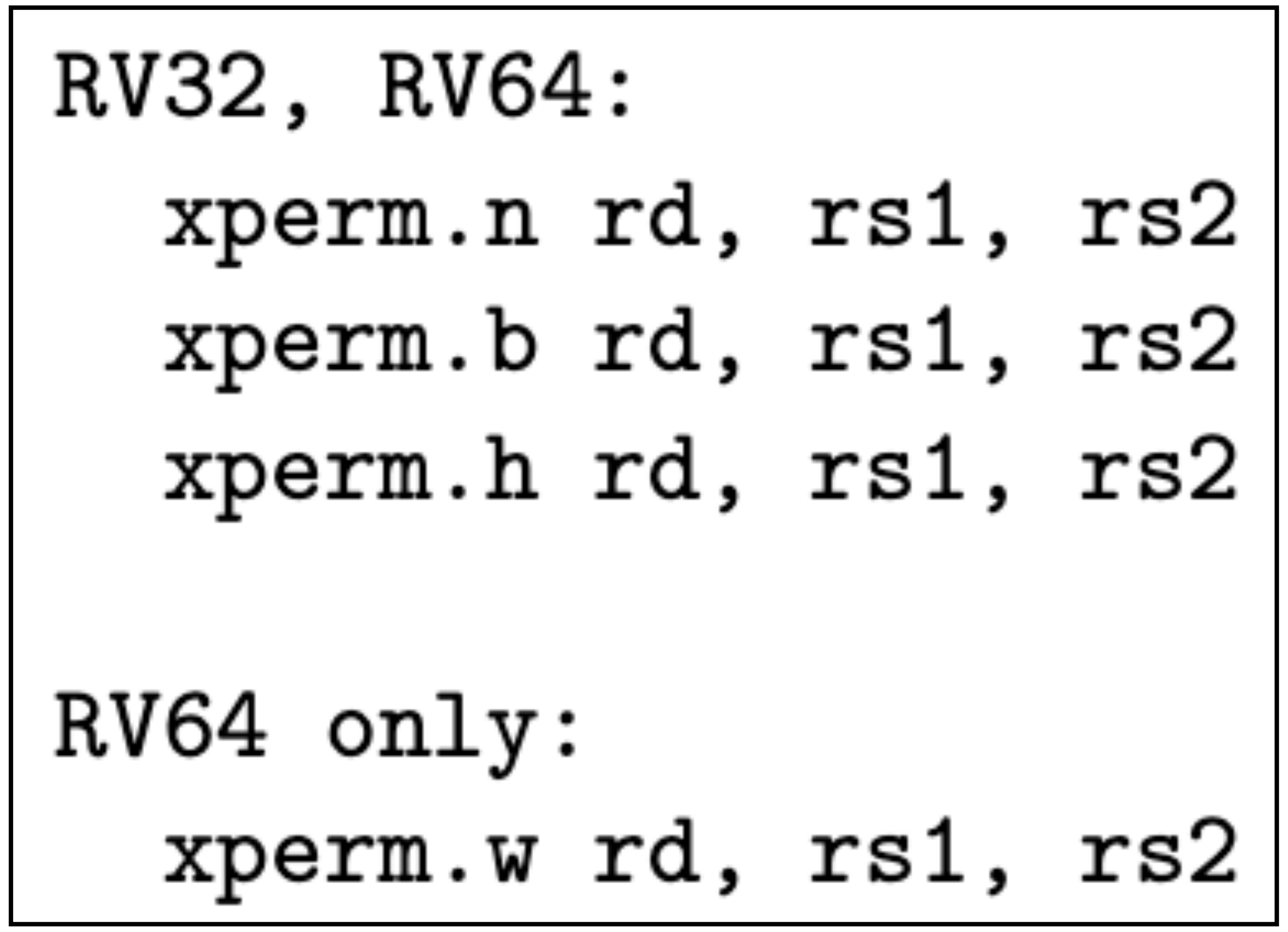


```
RV32, RV64:
  shfl    rd, rs1, rs2
  unshfl  rd, rs1, rs2
  shfli   rd, rs1, imm
  unshfli rd, rs1, imm

RV64 only:
  shflw    rd, rs1, rs2
  unshflw  rd, rs1, rs2
```

Crossbar Permutation Instructions

`xperm.[nbhw]`

- “交叉开关”排列指令
  - 可实现多种位排列组合



```
RV32, RV64:
  xperm.n rd, rs1, rs2
  xperm.b rd, rs1, rs2
  xperm.h rd, rs1, rs2

RV64 only:
  xperm.w rd, rs1, rs2
```

2.3. Other / Misc Instructions

位域装入

取出/放入特定bit

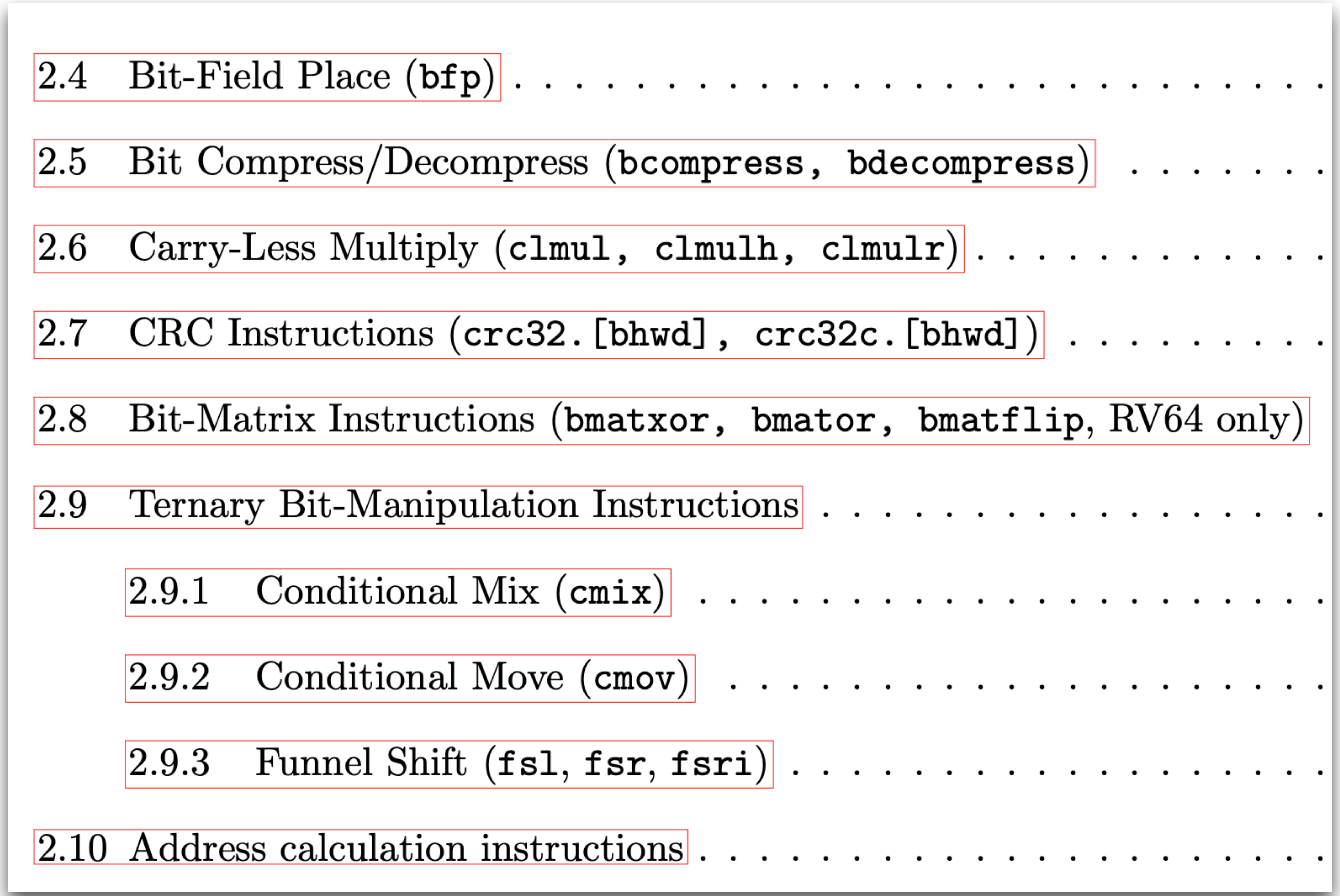
无进位乘法

CRC指令

位矩阵运算

三元位运算

地址计算（移位相加）



3. GNU/LLVM/Spike/QEMU Support status

GNU toolchain Support Status

- GNU upstream: No
- riscv-gnu-toolchain(master branch): No
- riscv-bitmanip/tools: Yes
  - It using the “riscv-bitmanip” branch of riscv-[gcc|binutils|opcodes] repos

LLVM Support Status

- Upstream (llvm/llvm-project): Yes
  - Update to Bitmanip Spec 0.93
    - since tag: llvmorg-12.0.0-rc1

Spike Support Status

- Upstream: Yes
  - Updated to Bitmanip Spec 0.92

QEMU Support Status

- Upstream: No
- patch: subsets of bitmanip extension
    - spec 0.93 - Zbb, Zbs and Zba subset instructions
    - https://patchew.org/QEMU/20210113071350.24852-1-frank.chang@sifive.com/

EOF