R-type:整数计算里的所有 R-R 计算，CSR 指令
I-type:整数计算里的所有 I-R 计算， JALR， LOAD， 内存模型，环境调用与断点
S-type(B-type):STORE（条件分支指令）
U-type(J-type):整数计算里的 LUI, AUIPC 计算（函数调用指令 JAL）

整数计算

ADD **[**sign-extended **I**mmediate] addi 实现了伪指令 mv, sext.w
SUBtract

AND **[**sign-extended **I**mmediate]
OR **[**sign-extended **I**mmediate]
XOR **[**sign-extended **I**mmediate] xori 实现了伪指令 not

Shift**L**eft **L**ogical [unsigned **I**mmediate]
Shift**R**ight **L**ogical [unsigned **I**mmediate]
Shift**R**ight **A**rithmetic [unsigned **I**mmediate]

Load**U**pperUnsigned**I**mmediate 推测 LUI 和 ADDI 结合实现伪指令 li
Add**U**pperUnsigned**I**mmediateTo**PC** AUIPC 和其它指令结合，可实现伪指令 la, load，store，call, tail

Set**L**ess**T**han [**U**nsigned]
Set**L**ess**T**han**I**mmediate **[U**nsigned**]** sltiu 实现了伪指令 seqz

控制流

Jump**A**nd**L**ink[**R**egister] 子程序调用(JAL)与返回(JALR)

Branch**EQ**ual
Branch**N**ot**E**qual
Branch**L**ess**T**han[**U**nsigned] blt[u]实现了伪指令 bgt[u]
Branch**G**reater**E**qual[**U**nsigned] bge[u]实现了伪指令 ble[u]

内存读写(仅 Load 和 Store 指令可读写内存)

Load**W**ord
Load**H**alfword [**U**nsigned extend]
Load**B**yte [**U**nsigned extend]
Store**W**ord
Store**H**alfword
Store**B**yte

内存模型

FENCE Load & Store
FENCE.Instruction & Data

CSR 指令

CSRRead & **W**rite [unsigned **I**mmediate]
CSRRead & **S**et Bits [unsigned **I**mmediate]
CSRRead & **C**lear Bits [unsigned **I**mmediate]

Rea**DCYCLE** [**H**igh] 由 csrrs 实现的伪指令
Rea**DTIME** [**H**igh] 由 csrrs 实现的伪指令
Rea**DINSTRET** [**H**igh] 由 csrrs 实现的伪指令

环境调用与断点

Environment**CALL**
Environment**BREAK**

通用寄存器

ZERO	x0	提供常数或丢弃结果，实现了伪指令 nop, neg, negw, snez, sltz, sgtz, beqz, bnez, blez, bgez, bltz, bgtz, j, jr, ret, csrr, csrw[i], csrs[i], csrc[i]
Reture**A**ddress	x1	
Stack**P**ointer	x2	
Global**P**ointer	x3	
Thread**P**ointer	x4	
Frame**P**ointer	x8	
Temporary**0-6**		x5-7,x28-31
Saved**0-11**		x8-9,x18-27
Function**A**rguments**0-7**		x10-x17

CSR 寄存器

machine **vendor** **id**

machine **arch**itecture **id**

machine **imp**lementation **id**

machine **har**dware **t**hread **id**

[**u** | **s** | **m**] **status**

[**u** | **s** | **m**] **i**nterrupt **e**nable

[**u** | **s** | **m**] **t**rap **vec**tor base address

[**s** | **m**] **e**xception **deleg**ation

[**s** | **m**] **i**nterrupt **deleg**ation

[**s** | **m**] **counter** **en**able

[**u** | **s** | **m**] **scratch**

[**u** | **s** | **m**] **e**xception **p**rogram **c**ounter

[**u** | **s** | **m**] trap **cause**

[**u** | **s** | **m**] **t**rap **val**ue

[**u** | **s** | **m**] **i**nterrupt **p**ending

[**s**] **a**ddress **t**ranslation & **p**rotection

physical **m**emory **p**rotection **c**on**fig**uration**0-3** (M-mode)

physical **m**emory **p**rotection **addr**ess**0-15** (M-mode)

[| **m**] **cycle** counter [**h**igh] (U or M)

[| **m**] **time**r [**h**igh] (U or M)

[| **m**] **inst**ructions-**ret**ired counter [**h**igh] (U or M)

[| **m**] **h**ardware **p**erformance-**m**onitoring **counter****3-31** [**h**igh] (U or M)

machine **h**ardware **p**erformance-**m**onitoring **event** selector**3-31**

la rd, symbol	**auipc rd, symbol[31:12] ; addi rd, rd, symbol[11:0]**
l{b|h|w|d} rd, symbol	**auipc rd, symbol[31:12] ; l{b|h|w|d} rd, symbol[11:0](rd)**
s{b|h|w|d} rd, symbol, rt	**auipc rd, symbol[31:12] ; s{b|h|w|d} rd, symbol[11:0](rt)**
nop	**addi x0, x0, 0**
li rd, immediate	
mv rd, rs	**addi rd, rs, 0**
not rd, rs	**xori rd, rs, -1**
neg rd, rs	**sub rd, x0, rs**
negw rd, rs	**subw rd, x0, rs**
sext.w rd, rs	**addiw rd, rs, 0**
seqz rd, rs	**sltiu rd, rs, 1**
snez rd, rs	**sltu rd, x0, rs**
sltz rd, rs	**slt rd, rs, x0**
sgtz rd, rs	**slt rd, x0, rs**
beqz rs, offset	**beq rs, x0, offset**
bnez rs, offset	**bne rs, x0, offset**
blez rs, offset	**bge x0, rs, offset**
bgez rs, offset	**bge rs, x0, offset**
bltz rs, offset	**blt rs, x0, offset**
bgtz rs, offset	**blt x0, rs, offset**
bgt rs, rt, offset	**blt rt, rs, offset**
ble rs, rt, offset	**bge rt, rs, offset**
bgtu rs, rt, offset	**bltu rt, rs, offset**
bleu rs, rt, offset	**bgeu rt, rs, offset**
j offset	**jal x0, offset**
jal offset	**jal x1, offset**
jr rs	**jalr x0, rs, 0**
jalr rs	**jalr x1, rs, 0**
ret	**jalr x0, x1, 0**
call offset	**auipc x6, offset[31:12] ; jalr x1, x6, offset[11:0]**
tail offset	**auipc x6, offset[31:12] ; jalr x0, x6, offset[11:0]**
fence	**fence iorw, iorw**

rdinstret[h] rd	**csrrs rd, instret[h], x0**
rdcycle[h] rd	**csrrs rd, cycle[h], x0**
rdtime[h] rd	**csrrs rd, time[h], x0**
csrr rd, csr	**csrrs rd, csr, x0**
csrw csr, rs	**csrrw x0, csr, rs**
csrs csr, rs	**csrrs x0, csr, rs**
csrc csr, rs	**csrrc x0, csr, rs**
csrwi csr, imm	**csrrwi x0, csr, imm**
csrsi csr, imm	**csrrsi x0, csr, imm**
csrci csr, imm	**csrrci x0, csr, imm**