1). 偏移量寻址
2). 寄存器间接寻址
3). 立即数寻址
4). 寄存器直接寻址
5). 偏移量寻址

3.

.21

nop ⇒ addi x0, x0, 0 无操作

ret ⇒ jalr x0 x1 0 跳转到 Reg[x1]
(调用函数的返回地址)

把 offset[31:12] 左移12位

call offset ⇒ auipc x6 offset[31:12]   ($x6 = pc + offset[31:12]$)
jalr x1 x6 offset[11:0]   ($x_1 = pc+4$, $pc = x6 + offset[11:0]$)

mv rd, rs ⇒ add rd, x0, rs

rdcycle rd ⇒ csrrs rd, cycle, x0
(cycle: cycle counter CSR，存入 rd；x0：不对CSR中任何一位设置)

sext.w rd rs ⇒ addiw rd, rs, 0
(RV64I)   有符号扩展到64位

7/1). 如果和的最高位和进位不同则溢出

并且只有2个同符号数相加才可能溢出

slti t3 , t2 , 0 ① 正+正=负：和<一个加数，另一加数

slt t4 , t0 , t1 ② 负+负=正：和>一个加数，另一加数

2). 两个无符号正数相加，溢出后变负数

addu t0 t1 t2

bltu t0 t1 overflow

3). ARM：用CPSR的溢出标志控制

MIPS：用计算指令触发中断产生溢出信号

x86：由硬件电路检测到溢出并引发中断

8 1). 填表：

$2^{xlen}-1$ x -1 x

这样设计是为了简化除法电路。对无符号除法返回全1是自然的结果？（riscv-spec-v2.2 的 natural value to return 没懂），有符号除法是用无符号除法电路实现的，这么做可以简化硬件（返回全1）

2). NV : Invalid Operation

DZ : Divide by Zero

OF : Overflow

UF : Underflow

NX : Inexact

会陷入系统调用（产生 instruction trap 引起异常）

x86 除以0 时会使当前程序暂停并报错。如果异
没有被屏蔽，则不存储结果；否则存储一个
法异常的符号 (?)

ARM ?

12. 1). S
2). ?
3). M
4). ?
5). U

17. 返回 $a_0 = 2^{30}$, $a_1 = 30$