3.

) nop，等效为 addi x0, x0, 0　不改变x0的值

) ret，等效为 jr , ra　将程序计数器(PC)设置为ra寄存器的值

) call offset : jal ra, offset

) mv rd, rs　addi rd, rs, 0　相当于加上0等于未改变.

) rdcycle rd　rdinstret rd

) sext.w rd rs　slli rd, rs, 0

7. (1) 对于有符号数的加法

~~blt t0, t1, overflow~~　sub t3, t0, t1

~~bge t0, t2, overflow~~　mv t4, t2

(2) 对于无符号数的加法

add t0, t1, t2

blt t0, t1, overflow

blt t0, t1, overflow

(3) 在x86指令集中，有专门的标志位用于指示加法运算的溢出情况。比如，add指令会将标志位OF设置为1表示发生了溢出，否则OF会被清零。可以使用jo或jno等条件分支指令来检查这个标志位.

在ARM指令集中，也有专门的标志位用于指示加法运算的溢出情况。比如，add指令会将标志位V(溢出标志位)设置为1表示发生了溢出，否则V会被清零。可以使用bvs或bvc等条件分支指令来检查这个标志位.

6. (1)

指令	rs1	rs2	Op=DIVU时 rd值	Op=REMU时 rd值	Op=DIV时 rd值	Op=REM rd值
Op rd, rs1, rs2	x	0	抛出异常	x	抛出异常	x

除法操作必须进行处理否则会产生错误的结果，余的结果没有定义，保持原结果不变可能对有些程序来说更方便实用.

2) EI：浮点异常中断使能位，用于控制浮点异常是否会导致中断，1表示浮点异常会导致中断，0表示不会。

OV：溢出标志位：用于指示浮点运算是否发生了溢出

UV：下溢标志位：用于指示浮点运算是否发生了下溢。

DZ：除以0标志位，用于指示浮点运算是否发生了除以0的情况。

NV：无效操作标志位，用于指示浮点运算是否发生了无效操作。

3) 在x86架构中，当整数除法指令除数为0时，会触发"#DE"异常，对于浮点除法指令，当除数为0时，x86会将结果设为特殊值，具体取决于浮点指令的类型。

在ARM架构中，当整数除法指令除数为0时，会触发"UDIVZERO"异常，对于浮点数，结果设为特殊值。ARM还提供了一些额外的指令来检查浮点除法的操作数是否有效，例如VDIV和VDIVF的第三个操作数用于检查除数是否为0。

~~B. vecMul: addi t3, x0, 100~~

~~sub t3, t3, 1~~

~~bne beq t3, x0, vecMul~~

```
B. li   t3, 0
loop:   bge  t3, 100, end_loop
        lw   t4, 0(t0)
        lw   t5, 0(t1)
        mul  t6, t4, t5
        sw   t6, 0(t0)
        addi t3, t3, 1
        addi t1, t1, 4
        addi t0, t0, 4
```

```
        j loop
end_loop:
        lw   t7, 0(t2)
        sll  t8, t0, 0
        lw   t9, 0(t8)
        mul  t0, t9, t7
        jr   ra
```

14.
```
bge  a0, a1, greater_than
sub  a2, a0, a1
j    end_if
greater_than:
    add  a2, a0, a1
end_if
```

15.
```
addi  t2, t0, 0
sw    t2, 0(t0)
addi  t2, t1, 0
sw    t2, 4(t0)
slli  t3, t1, 2
add   t4, t0, t3
sw    t2, 0(t4)
```

16.
```
lw  t2, 0(t0)
lw  t3, 0(t1)
sw  t3, 0(t0)
```