```
Lista de instrucoes:


Nao faz nada:
NOP
00000000000000000000000000000000

Tipo R
31:26   25:21   20:16   15:11   10:0
DESCRICAO: rC = rA OPER rB

opcode  rA      rB      rC      funct
ADD
000001  00001   00001   00001   00000000001
SUB
000001  00001   00001   00001   00000000010
AND
000001  00001   00001   00001   00000000100
MUL
000001  00001   00001   00001   00000001000
SLT
000001  00001   00001   00001   00000010000

Tipo Imm
DESCRICAO: rB = rA OPER imm
31:26   25:21   20:16   15:0
opcode  rA      rB      imm
ADDI
000101  00001   00001   0000000000000000

Desvio

opcode  rA      rB      offset
BEQ
if(rA == rB) PC += offset*4
010000  00000   00000   0000000000000000
if(rA == rB) PC -= offset*4
010000  00000   00000   1000000000000000
BLT
if(rA < rB) PC += offset*4
110000  00010   00000   0000000000000000
if(rA < rB) PC -= offset*4
110000  00010   00000   1000000000000000

Memoria
DESCRICAO: rB = MEM[offset + rA]
opcode  rA      rB          offset
LW
001001  00010   00100   0000000000000000
MEM[offset + rA] = rB
SW
001000  00010   00010   0000000000000000


--testar load store
        inst_mem[0] <= 32'b00000000000000000000000000000000; // nop
        inst_mem[1] <= 32'b001001_00000_00010_0000000000000010; // lw 2(r0) -> r2
        inst_mem[2] <= 32'b00000000000000000000000000000000; // nop
        inst_mem[3] <= 32'b00000000000000000000000000000000; // nop
        inst_mem[4] <= 32'b00000000000000000000000000000000; // nop
        inst_mem[5] <= 32'b00000000000000000000000000000000; // nop
        inst_mem[6] <= 32'b00000000000000000000000000000000; // nop
        inst_mem[7] <= 32'b001001_00000_00011_0000000000000011; // lw 3(r0) -> r3
        inst_mem[8] <= 32'b00000000000000000000000000000000; // nop
        inst_mem[9] <= 32'b00000000000000000000000000000000; // nop
        inst_mem[10] <= 32'b00000000000000000000000000000000; // nop
        inst_mem[11] <= 32'b00000000000000000000000000000000; // nop
        inst_mem[12] <= 32'b00000000000000000000000000000000; // nop
        inst_mem[13] <= 32'b000001_00010_00011_00100_00000000001; // add r2 r3 -> r4
        inst_mem[14] <= 32'b00000000000000000000000000000000; // nop
        inst_mem[15] <= 32'b00000000000000000000000000000000; // nop
        inst_mem[17] <= 32'b00000000000000000000000000000000; // nop
```

```
        inst_mem[18] <= 32'b00000000000000000000000000000000; // nop
        inst_mem[19] <= 32'b001000_00000_00100_0000000000000010; // sw 2(r0) <- r4
        inst_mem[20] <= 32'b00000000000000000000000000000000; // nop


--testar tipo r e branch
        inst_mem[0] <= 32'b00000000000000000000000000000000; // nop
        inst_mem[1] <= 32'b000101_00010_00011_0000000000000011; // addi r2  3 -> r3
        inst_mem[2] <= 32'b00000000000000000000000000000000; // nop
        inst_mem[3] <= 32'b000101_00011_00011_0000000000000001; // addi r3  1 -> r3
        inst_mem[4] <= 32'b00000000000000000000000000000000; // nop
        inst_mem[5] <= 32'b000001_00011_00011_00100_00000000001; // add r3  r3 -> r4
        inst_mem[6] <= 32'b000110_00010_00101_0000000000000001; // subi r2 1 -> r5
        inst_mem[7] <= 32'b00000000000000000000000000000000; // nop
        inst_mem[8] <= 32'b00000000000000000000000000000000; // nop
        inst_mem[9] <= 32'b000001_00101_00011_00110_00000000010; // sub r5 r3 -> r6
        inst_mem[10] <= 32'b100000_00000_00000_0000000000000000; // beq r0 r0 -> inst 0
        inst_mem[11] <= 32'b00000000000000000000000000000000; // nop

--testar mull
        inst_mem[0] <= 32'b00000000000000000000000000000000; // nop
        inst_mem[0] <= 32'b000001_00010_00011_00100_00000000001; // mul r2 r3 -> r4
        inst_mem[1] <= 32'b000001_00010_00011_00100_00000000001; // mul r2 r3 -> r4




r0 -> constante zero
r1 -> registradores livres
r2
r3
r4
r5
r6
r7
r8
r9
r10
r11
r12
r13
r14
r15
r16
r17
r18
r19
r20
r21
r22
r23
r24
r25
r26
r27
r28
r29
r30
r31
```