1. Instrucciones del Repertorio del procesador MIPS.

a) Aritmético Lógicas.

```
.text
add     $t1, $t2, $t3          #Addition (with over ow)
addi    $t1, $t2, 0x1000       #Addition Immediate (with over ow)
addu    $t1, $t2, $t3          #Addition (without over ow)
addiu   $t1, $t2, 0x1000       #Addition Immediate (without over ow)
and     $t1, $t2, $t3          #AND
andi    $t1, $t2, 0x1000       #AND Immediate
div     $t2, $t3               #Divide (signed)
divu    $t2, $t3               #Divide (unsigned)
mult    $t2, $t3               #Multiply
multu   $t2, $t3               #Unsigned Multiply
nor     $t1, $t2, $t3          #NOR
or      $t1, $t2, $t3          #OR
ori     $t1, $t2, 0x1000       #OR Immediate
sll     $t1, $t2, 31           #Shift Left Logical
sllv    $t1, $t2, $t3          #Shift Left Logical Variable
sra     $t1, $t2, 31           #Shift Right Arithmetic
srav    $t1, $t2, $t3          #Shift Right Arithmetic Variable
srl     $t1, $t2, 31           #Shift Right Logical
srlv    $t1, $t2, $t3          #Shift Right Logical Variable
sub     $t1, $t2, $t3          #Subtract (with over ow)
subu    $t1, $t2, $t3          #Subtract (without over ow)
xor     $t1, $t2, $t3          #XOR
xori    $t1, $t2, 0x1000       #XOR Immediate
lui     $t1, 0x1000            #Load Upper Immediate
slt     $t1, $t2, $t3          #Set Less Than
slti    $t1, $t2, 0x1000       #Set Less Than Immediate
sltu    $t1, $t2, $t3          #Set Less Than Unsigned
sltiu   $t1, $t2, 0x1000       #Set Less Than Unsigned Immediate
```

b) Bifurcaciones y Saltos

```
label:  beq     $t2, $t3, label         #Branch on Equal
        bgez    $t1, label              #Branch on Greater Than Equal Zero
        bgezal  $t1, label              #Branch on Greater Than Equal Zero And Link
        bgtz    $t1, label              #Branch on Greater Than Zero
        blez    $t1, label              #Branch on Less Than Equal Zero
        bgezal  $t1, label              #Branch on Greater Than Equal Zero And Link
        bltzal  $t1, label              #Branch on Less Than And Link
        bltz    $t1, label              #Branch on Less Than Zero
        bne     $t2, $t3, label         #Branch on Not Equal

        j       label                   #Jump
        jal     label                   #Jump and Link
        jalr    $t1                     #Jump and Link Register
        jr      $t1                     #Jump Register
```

c) Cargas y transferencias.

```
        lb      $t1, 0($t2)             #Load Byte
        lbu     $t1, 0($t2)             #Load Unsigned Byte
        lh      $t1, 0($t2)             #Load Halfword
        lhu     $t1, 0($t2)             #Load Unsigned Halfword
        lw      $t1, 0($t2)             #Load Word
        lwl     $t1, 0($t2)             #Load Word Left
        lwr     $t1, 0($t2)             #Load Word Right
        sb      $t1, 0($t2)             #Store Byte
        sh      $t1, 0($t2)             #Store Halfword
#Store the low halfword from register Rsrc at address.
        sw      $t1, 0($t2)             #Store Word
        swl     $t1, 0($t2)             #Store Word Left
        swr     $t1, 0($t2)             #Store Word Right
        mfhi    $t1                     #Move From hi
        mflo    $t1                     #Move From lo
        mthi    $t1                     #Move To hi
        mtlo    $t1                     #Move To lo
```

d) Control del procesador y coprocesadores.

```
        rfe                             #Return From Exception.  Restore the Status register.
        syscall                         #System Call. Register $v0 contains the number of
                                        #the system call (see Table 1) provided by SPIM.
        break   0                       #Break Cause exception n .
                                        #Exception 1 is reserved for the debugger.
        nop                             #No operation Do nothing.
El coprocesador de punto flotante es el número 1.
        lwcz    Reg, 0($t2)             #Load Word Coprocessor
#Load the word at address into register Reg of coprocessor z (0..3).
        swcz    Reg, 0($t2)             #Store Word Coprocessor
#Store the word from register Reg of coprocessor z at address.
        mfcz    CpuSrc, CPdest          #Move CPdest From Coprocessor z
#Move the contents of coprocessor z 's register CPdest to CPU register CpuSrc.
        mtcz    CpuSrc, CPdest          #Move To Coprocessor z
#Move the contents of CPU register CpuSrc to coprocessor z 's register CPdest.
        bczt    label                   #Branch to label if Coprocessor z flag is set (True)
        bczf    label                   #Branch to label if Coprocessor z flag not set (False)
```

e) Instrucciones de punto flotante.

```
abs.d    $f0, $f2          #Floating Point Absolute Value Double
abs.s    $f0, $f2          #Floating Point Absolute Value Single
#Compute the absolute value of the floating float double (single) in register $f2 and put it in register $f0.

add.d    $f0, $f2, $f4     #Floating Point Addition Double
add.s    $f0, $f2, $f4     #Floating Point Addition Single
#Compute the sum of the floating float doubles (singles) in registers $f2 and $f4 and put it in register $f0.

c.eq.d   $f0, $f2          #Compare Equal Double
c.eq.s   $f0, $f2          #Compare Equal Single
#Compare the floating point double (single) in register $f0 against the one in $f2 and set the floating point
#condition flag true if they are equal.

c.le.d   $f0, $f2          #Compare Less Than Equal Double
c.le.s   $f0, $f2          #Compare Less Than Equal Single
c.lt.d   $f0, $f2          #Compare Less Than Double
c.lt.s   $f0, $f2          #Compare Less Than Single
#Compare the floating point double in register $f0 against the one in $f2 and set the condition flag true
#if the first is less than the second.

cvt.d.s  $f0, $f4          #Convert Single  to Double
cvt.d.w  $f0, $f4          #Convert Integer to Double
#Convert the single precision floating point number or integer in register $f4 to a
#double precision number and put it in register $f0.

cvt.s.d  $f0, $f4          #Convert Double to Single
cvt.s.w  $f0, $f4          #Convert Integer to Single
#Convert the double precision floating point number or integer in register $f4 to a
#single precision number and put it in register $f0.

cvt.w.d  $f0, $f4          #Convert Double to Integer
cvt.w.s  $f0, $f4          #Convert Single to Integer
#Convert the double or single precision floating point number in register $f4 to
#an integer and put it in register $f0.

div.d    $f0, $f0, $f2     #Floating Point Divide Double
div.s    $f0, $f0, $f2     #Floating Point Divide Single

mov.d    $f0, $f4          #Move Floating Point Double
mov.s    $f0, $f4          #Move Floating Point Single
#Move the floating float double (single) from register $f4 to register $f0.

mul.d    $f0, $f2, $f4     #Floating Point Multiply Double
mul.s    $f0, $f2, $f4     #Floating Point Multiply Single
#Compute the product of the floating float doubles (singles)
#in registers $f2 and $f4 and put it in register $f0.

neg.d     $f0, $f4         #Negate Double
neg.s    $f0, $f4          #Negate Single
#Negate the floating point double (single) in register $f4 and put it in register $f0.

sub.d    $f0, $f2, $f4     #Floating Point Subtract Double
sub.s    $f0, $f2, $f4     #Floating Point Subtract Single
#Compute the difference of the floating float doubles (singles)
#in registers $f2 and $f4 and put it in register $f0.
```