**SUBI Instructions**

lbi r1,3 // load 10 to r1

subi r2,r1,-2 // subtract -20 from 10

lbi r1,5 // load 5 to r1

subi r2,r1,10 // subtract 10 from 5

lbi r1,5 // load 5 to r1

subi r2,r1,3 // subtract 3 from 5

**JAL Instructions**

JAL 2 // Should jump to the target, PC = 4

Nop

Nop

Halt

// checks for negative jump

Nop

JAL 12 // Should jump to first target, PC = 16

Nop

Nop

Nop

Nop // PC = 10 second target

Halt

Nop

Nop // PC = 16 first target

JAL -10 // Should jump to second target, PC = 10

Nop

Nop

Halt

// checks for lack of jump

JAL 0 // Should increment to the next PC

Nop

Halt

// checks for correct values in R7 & PC from jump and link

JAL .GoHere

lbi r6, -1 // Should not enter here

.GoHere:

lbi r6, 1 // R6 should be 1

halt // R7 should have previous instruction

// checks basic JAL

lbi r1, 0x00

jal 2 // Jumps over next instruction

lbi r1, 0x01 // R1 should have 0

lbi r2, 0x00 // R2 should have 0

halt // R7 should have address of lbi $1, 0x01

// checks backward jump

lbi r1, 0x01

j 4 // j pc + 2 + 4

addi r1, r1, 1

j 2

jal -6 // jal pc + 2 -6

lbi r0, 0x01

halt // R7 should have address of 2nd lbi, r1 = 2, r0 = 1

// checks irregular & repeated use of jal

lbi r1, 0x01

jal 0

jal 0

jal 2

lbi r1, 0x02

halt // R1 = 1 and R7 should have address of lbi

// checks small positive, negative and zero jumps

jal 12

nop

nop

jal 0

halt

nop

nop

jal -10

// checks loop to earlier portion of program (R7 set to 0x8)

lbi r1, 0xfd

addi r1, r1, 0x01

bgez r1, .done // after 3 exec of add, go to halt

jal 0x7fa

.done:

Halt

// checks jump and link with most negative aligned immediate

jal -1024

// checks jump and link with largest aligned immediate

jal 1022

**JALR Instructions**

// checks if jalr can return to where jal jumped from

Nop

JAL 10 // Should jump to first target, PC = 14 & store PC = 4

Nop // PC = 4 second target

Halt

Nop

Nop

Nop

Nop // PC = 14 first target

JALR r7, 0 // Should jump to second target, PC = 4

Nop

Nop

Halt

// checks jump to registers except r7

LBI r3 26

Nop

Nop

JALR r3, 0 // Should jump to target, PC = 26

Nop

Nop

Nop

Halt

Nop

Nop

Nop

Nop

Nop

Nop // PC = 26 target

Nop

Halt

// checks if values can be added to jump register

JAL 4 // Jumps to TargetA, PC = 6

Nop

Nop

Nop // PC = 6 TargetA

JALR r7, 16 // Jumps to TargetB, PC = 18

Nop

Nop

Nop

Nop

Nop // TargetB

Halt

Nop

Nop

Halt

// checks jalr jump to half, R7 = 0x4

lbi r0, 0x4

jalr r0, 0x4

add r1, r1, r1

add r2, r2, r2

halt

// checks jalr jump to earlier part of program, R7 = 0xa

lbi r0, 0x0 // R0 used for jump address calculation

lbi r1, 0xfd // R1 acts as a loop counter

addi r1, r1, 0x01

bgez r1, .done // after 3 exec of add, go to halt

jalr r0, 0x4

.done:

halt

// checks jump and link

lbi r0, 0

jalr r0, 4 //jumps to A then B then C

lbi r0, 8 //target A

jalr r0, -6 //target B

nop

nop

halt //target C

// checks jump and link with large positive aligned jump

lbi r0, 127

jalr r0, 127

// checks jump and link with large negative jump

lbi r0, -128

jalr r0, -128