Design Name: AVS

Instruction	**Functionality**	**Opcode**
init *Rx, imm*	Rx = imm	000 xx ii
base *Rx, [Mem]*	Rx = [Mem]?	001 xx yy
load *imm*	$R1 = imm	100 11 ii
store *Rx, imm*	Mem[imm] = Rx	011 xx ii
shl *Rx*	Rx shift left one bit, 0 shifted into LSB	100 00 xx
sll *Rx, Ry*	Rx = Rx * (2^Ry)	100 xx yy
slt *Rx, Ry*	$R0 = 1 if Rx < Ry	101 xx yy
BezDec *imm*	If $R0 == 0, then PC = PC + imm, else $R0 = $R0 – 1, PC = PC + 1	100 01 ii
BnezDec *imm*	If $R0 != 0, then PC = PC + imm, else $R0 = $R0 – 1, PC = PC + 1	100 10 ii
xori *Rx, imm*	$R0 = Rx (EXCL) with imm	110 imm
andi *Rx, imm*	$R0 = Rx (AND) with imm	111 xx ii
jump *'branch'*	PC = PC + imm	010 iiii
addi *Rx, imm*	Rx = Rx + imm	001 xx yy
halt	Stop	000 00 00

Machine Code for Program 1:

```
#Assume everything is equal to zero at first
#$t1 = 00
#$t4 = 01
#$t5 = 10
#$t6 = 11
#$t7 = 5
#$t9 = 6
#$s0 = 7

addi $t6, $0, 1                 001 11 10
lw $t1, P($0)                   100 11 00

loop:
beq $t1, $0, exit               100 01 ??   #end program
addi $t7, $0, -1                001 ?? 11   #-1 in 2s complement
addi $t5, $0, 5                 001 10 ??   #5 value?
addi $t9, $0, 17                001 ?? ??    #need 4 bits of 17
next:
beq $t5, $0, next2              100 01 ??    #jump to loop
```

Design Name: AVS

add $t6, $t6, $t4	001 11 01
add $t5, $t5, $t7	001 10 ??
j next	010 ????
next2:	
slt $s0, $t6, $t9	101 11 ??
bne $s0, $0, down	100 10 ??
subi $t6, $t6, 17	001 11 ??
j next2	010 ????
down:	
add $t5, $0, 5	001 10 ??
add $t1, $t1, $t7	001 00 ??
add $t4, $0, $t6	001 01 11
j loop	010 ?? ??
exit: sw $t6, R($0)	011 11 00
	000 00 00