1.

a. This should be implemented within the ‘I’ instruction set, as it contains an immediate value within the instruction.

```
beq $t0, SignExtImm end
lw $t0, 0($t1)
addi $t1, $t1, 1
end:
```

b.

```
001000 00000 01101 0000010010101011
000001 00000000001000000000000010

000000 00000 01101 0110100001000011
001100 01101 01110 0000000000111100
000000 00100 01110 11001 00000 100000
101011 11001 01101 0000000000000100

000101 01101 00000 0000000000001100
000000 10111 000000000000000 001000
```

2.

a.

```
# sum = 0                 Use $t9 for sum
# i = 0                   Use $t8 for i
#
# array1[i]               Use $t0 for storage
# array2[i]               Use $t1 for storage
#
# array1[i]-array2[i]     Use $t2 for res.
# abs($t1)                Use $t3 for res.
#
# I have no idea what sat8 is so I’m just
# going to assume its a library or something
#-------------------------------------------------------#
```

B

```
sad:
li $t9, 0                 # sum = 0
li $t8, 0                 # int i in loop condition

j condition

loop:

lw $t0, 0($a0)            # load array1[i]
lw $t1, 0($a1)            # load array2[i]
addi $a0, $a0, 4          # increment array1 pointer by 4
addi $a1, $a1, 4          # increment array2 pointer by 4
addi $t8, $t8, 1          # increment i by 1

add $t2, $t0, $t1         # array1[i]-array2[i]
abs $t3, $t2              # abs($t2)
add $t9, $t9, $t3         # sum += $t3

condition:
slt $t4, $t8, $a2         # if (i < len)     $t4 = 1
bne $t4, $0, loop         # $t4 == 0?

add $v0, $t3, $0
```