2 MIPS Instruction Set

We translate the binary code to assembly code ar first:
0: 000000 00100 00000 00011 00000 100101
00 04(4) 00(0) 03(3) 00 25
or $v1, $a0, $zero

4 : 001001 00100 00100 0000000000101000
09 04(4) 04(4) 0028
addiu $a0, $a0, 40

8 : 000000 00000 00000 00010 00000 100101
00 00 00 02(2) 00 25
or $v0, $zero, $zero

C : 100011 00011 00101 0000000000000000
23 03(3) 05(5) 0000
lw $a1, $v1, 00

10: 001001 00011 00011 0000000000000100
09 03(3) 03(3) 0004
addiu $v1, $v1, 4

14: 000101 00011 00100 1111111111111101
05 03(3) 04(4) fffd
bne $a0, $v1, 65533

18: 000000 00010 00101 00010 00000 100001
00 02 05 02 00 21
addu $v0, $v0, $a1

1C: 000000 11111 00000 00000 00000 001000
00 1f 00 00 00 08
jr $ra

20: 000000 00000 00010 00010 00000 100011
00 00 02(2) 02(2) 00 23
subu $v0, 00, $v0

The assmbly code is as follow

```
0: or $v1, $a0, $zero
4: addiu $a0, $a0, 40
8: or $v0, $zero, $zero
C: lw $a1, $v1, 00
10: addiu $v1, $v1, 4
14: bne $a0, $v1, 65533
18: addu $v0, $v0, $a1
1C: jr $ra
20: subu $v0, 00, $v0
```

The 0x14 instruction implements a branch jump. It can be translated as follow: If $a0 is not equal to $v1, the processor jumps to the introduction 65533 * 4 + 4 + PC. 65533 can be seen as a negative number -3, because this number is 16 bits long. Thus, the processor will jump to 0x0C. If $a0 is equal to $v1, the processor will not jump. This process corresponds to a do-while loop.
In addition, the return value is hold in $v0.

A more C like code

```
void func(int *a0){//may be array a[]
        int *v1 = a0; //one int is 4 bytes
        a0 = a0 + 10;
        int v0 = 0;
        do{
                a1 = *v1;
                v1++;
        }while(a0 != v1);
        v0 = v0 + a1;//v0 = a[9]
        return;
        v0 = -v0;//return -v0
}
```

This function recive a int array a and return -a[9]