**idea** : Mips assembly code not compiled or simulated , you have to do static analysis for it to get the flag .

**solve** :

when you open the file you’ll notice a very tall lapel called $ **LC0**

|  |
| --- |
| **$LC0:  .word -10  .word 51  .word 40  .word 22  .word 82  .word 40  .word 1  .word 79  .word -8  .word 78  .word -10  .word 12  .word 23  .word 40  .word 19  .word 79  .word 40  .word 1  .word 76  .word -2  .word 45  .word 40  .word 16  .word -9  .word -2  .word 80  .word 40  .word 72  .word 240  .word 26  .word 192  .word 64  .word 21  .word 15  .word 114  .word 22  .word 210  .word 73  .word 108  .word 26   .word 192  .word 64  .word 21  .word 62  .word 30  .word 26  .word 192  .word 43  .word 243  .word 64  .word 21  .word 75  .word 99  .word 16  .word 237  .word 64  .word 21  .word 36  .word 147  .word 100  .word 114  .word 58  .word 48  .word 64  .word 21  .word 22  .word 210  .word 49  .word 216  .word 27  .word 60  .word 64  .word 21  .word 12  .word 0  .word 49  .word 216  .word 16  .word 237  .word 40  .word 129  .section .text.startup,"ax",@progbits  .align 2  .globl main  .set nomips16  .set nomicromips  .ent main  .type main, @function** |

this is string part of the code which the value on it’ll be assigned to variables on the code ( and also this will be the flag ) .

if i’s not clear how to know the type of the assembly , you can find it on the label **$L15**

|  |
| --- |
| $L15:  lw $25,%call16(\_\_stack\_chk\_fail)($28)  nop  .reloc 1f,R\_MIPS\_JALR,\_\_stack\_chk\_fail |

analyse the code and search for important instructions

slt , bnz , bne ( or any branch instruction which do jumping , set less than which compares )

, xor , some of addiu .

and sure the register where the above values had loaded in .

…..

you’ll find :

load

|  |
| --- |
| lui $2,%hi($LC0) |

|  |
| --- |
| addiu $2,$2,%lo($LC0) |

loaded to **$2** using upper and lower parts of it as it’s more than 16 byte

and at the end it put it into $19

$L2 not interesting for us , the only important information on it is

|  |
| --- |
| lw $2,0($2)  move $18,$0  sw $2,0($3)  .option pic0  b $L5  .option pic2  li $16,81 # 0x51 |
|  |

put zero at **$18** as **$0** only carry zero

and jump to L5

81 is the size of the values loaded using **$2**

L2 looks like a loop

at **$L5**

|  |
| --- |
| **$L5:  slt $2,$18,27  bne $2,$0,$L13** |

will compare the value of **$18**  which is 0 now with 27 the write 1 or 0 to **$2**

then compare **$2**  with 0

then it jumps to **$L13**  if ture

this part looks like if statement >> if( $18 < 27 ){ jmp to L13 }

|  |
| --- |
| $L13:  lw $4,0($19)  nop  addiu $4,$4,12  xori $4,$4,0x6b |

in **L13** it load values from the place where $19 points to

then add 12 to it

then do xor woth 0x6b

which will get the first half of the flag .

then it’ll go to L4 to print the value then back to iterate again .

after $18 is greater than 27

|  |
| --- |
| andi $2,$18,0x1   bne $2,$0,$L14 |

it’ll continue in $L5

do and operation between $18 value and 0x1 , this operation is $18 % 2

then compare the result with 0 >> if(! $18 %2 ){ jump to L14 }

|  |
| --- |
| lw $4,0($19)  nop  addiu $4,$4,-12  xori $4,$4,0x6b |

it also load the value then add -12 and xor with 0x6b

this will get the rest of the flag .

else

|  |
| --- |
| li $4,32 # 0x20 |

put 32 on the value which is ‘ ‘

the flag is

**iT\_I5\_f0o1isH\_t0\_f3aR\_wha7\_We\_haVe\_Yet\_To\_s3E\_aNd\_kNow**