

## **func1**

- `sub sp, sp, #16`

This decrements the stack pointer by 16 bytes

- `str w0, [sp, 12]`

This stores the value of register w0 into memory at 12 bytes offset from the current location of the stack pointer

- `str w1, [sp, 8]`

This stores the value of register w1 into memory at 8 bytes offset from the current location of the stack pointer

- `ldr w1, [sp, 12]`

This loads the value from the memory location 12 bytes offset from the stack pointer and loads it into w1. The value of w1 is thus the value of w0

- `ldr w0, [sp, 8]`

This loads the value from the memory location 8 bytes offset from the stack pointer and loads it into w0. The value of w0 is thus the value of w1.

- `cmp w1, w0`

This checks whether the values of w1 and w0 are equal or not

- `bls .L2`

This runs the .L2 function if the values are less than or equal to each other

- `ldr w0, [sp, 12]`

This loads the value from the memory location 12 bytes offset from the stack pointer and loads it into w0. The value of w0 is the value of... w0?

## **L2**

- `ldr w0, [sp, 8]`

This loads the value from the memory location 8 bytes offset from the stack pointer and loads it into w0.

## **L3**

- `add sp, sp, 16`

This adds 16 to the stack pointer

## main

- `stp x29, x30, [sp, -48]!`

This stores the values of the registers x29 and x30 onto the stack pointer, and decrements the value of the stack pointer by 48.

- `add x29, sp, 0`

This initializes the x29 register to be equal to the value of sp

- `str x19, [sp, 16]`

This stores the value of the register x19 into a location 16 bytes offset from the stack pointer

- `str w0, [x29, 44]`

This stores the value of the register w0 into a location 44 bytes offset from the stack pointer

- `str x1, [x29, 32]`

This stores the value of the register x1 into a location 32 bytes offset from the stack pointer

- `ldr x0, [x29, 32]`

This loads the value from the memory location 32 bytes offset from the stack pointer and loads it into x0. The value of x0 is thus the value of x1.

- `add x0, x0, 8`

This adds 8 to the value of x0

- `ldr x0, [x0]`

This takes the value in x0 as a memory address, fetches the data from that memory address, and stores the data from that memory address back into x0.

- `bl atoi`

This changes the value stored in x0 to an integer and stores it inside w0

- `mov w19, w0`

This simply copies the value in w0 to w19.

- `ldr x0, [x29, 32]`

This loads the value from the memor location 32 bytes offset from the stack pointer and loads it into x0. The value of x0 is thus the value x1.

- `bl atoi`

This changes the value stored in x0 to an integer and stores it inside w0.

- `mov w1, w0`

This copies the value in w0 to w1.

- `mov w0, w19`

This copies the value in w19 to w0.

- `bl func1`

This executes func1

- `mov w1, w0`

This copies the value in w0 to w1.

- `adrp x0, .LC0`
- `add x0, x0, :lo12:.LC0`

This loads the high bits of LC0 into x0 This adds the low bits of LC0 into x0

- `bl printf`

This runs the printf function

- `mov w0, 0`

This sets the value in w0 to 0.

- `ldr x19, [sp, 16]`

This loads the value 16 bytes offset from the stack pointer into x19

- `ldp x29, x30, [sp], 48`

This cleans up the function and returns the stack pointer to its original position and allows it to return.

- `ret`

This returns the value