

Forum:[Hack Forums](#)

Thread: [\[1/2\]\[Short\] Understanding functions and value parsing/returning in ARM64 ASM](#)

Biggy_Asian - October 8, 2024, 5:41 am

[Credited User link \(click me. OFFSITE CAREFUL\)](#)

Hope the admins dont mind me giving credit although it is off platform...

I saw this posted on another forum related to game hacking/cheats. I think I attempted to post another thread that had all the links however I think it may have broken offsite rules or I just didnt submit it either way I am going to post the hidden content of the post but also include the true OP on the othersites link to the same post as I am not gonna claim their work for my own.

This is going to be the 1/2 in the 2 part posts I will send here. Like I said this is from a forum related to game hacking (mainly unity games) for ios however you can still 100000% take things away from this as it should give you an understanding of how ASM on ARM64 devices works when interacting with the target.

The Juice:

Branch/Branch With Link

Want to put this here in case I forget and to teach others the basics on how function invocation as well as parsing variables into the function.

This tutorial will go over invocation of functions in ASM and how to parse values into the same function but how to also use any values returned.

Tried to make this easy as possible for you to understand and tried to make it as less messy to read too.

Understanding how a function is called:

A function is called in ASM by using the B/BL operator which is called "Branch" / "Branch With Link". This is what executes any function that is parsed to it.

Example from IDA:

As you can see we are calling the `PlayerManager$$get_CurrentData` function. After calling it we are then comparing with the `loc_3115650` function. (I wont go over CBZ. Only B/BL)

Those whom don't understand what is happening here will be wondering what value is X0. The answer to that is whatever the `PlayerManager$$get_CurrentData` function returns.

Below will be an example:

```
Code [Offset="0x1"] // We will use 0x1 as an example
int TestFunction(int input) // takes an input which is an int
{
    std::cout << "You entered: " << input << std::endl; // prints the input: "You entered: x"
    return input; // returns the number 1 when called
}
```

As you can see we have a function which returns a number which will be whatever the input argument is when parsed.

To run this, parse a value and to store the value returned we will do something like:

```
Code MOV X0, #10 ; Stores the value 10 into X0
B #0x1 ; Runs our function and parses X0 as the input argument
ADD X0, X0, 1 ; Adds 1 to whatever X0 is which in this case is 10 which the result is 11
Whatever X0 is will be parsed into the input argument as X0 is the first argument.
```

So if there was multiple arguments it would be X0, X1, X2, X3 etc (The number increments depending on the number of arguments the target function has).

The same is true for when a function returns multiple values too.

So that is how we parse and also use the returned data from the function we ran.

Now you might be confused by the fact the screenshot shows the program using BL (Branch with Link) and not just B (Branch).

The difference between Branch and Branch with Link:
Branch with link (BL) copies the address of the next instruction (after the BL) into the link register. The branch instruction doesn't.

BL would be used for a subroutine call, so when you want to return to where you were you can branch back to the link register.