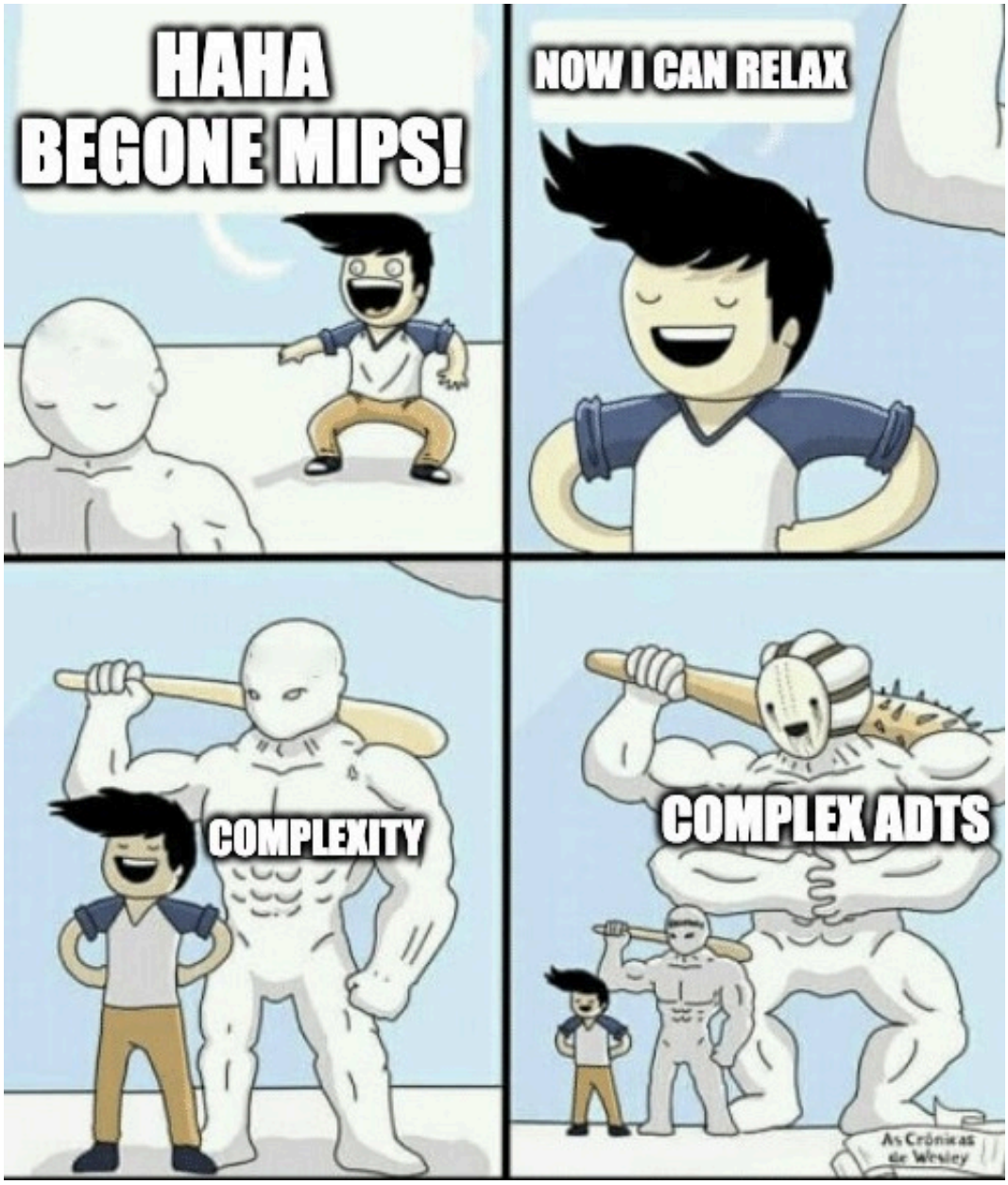
4.1 - Week 4 - Applied - Theory

Final Week for MIPS



We are going to wrap up the MIPS concepts this week and move on to bigger and (arguably) better things from Week 5 onwards.

The following are the objectives for today's applied session:

- To understand memory diagrams.
- To test your understanding of function aspects in MIPS
- To understand the function calling and returning convention in MIPS.
- To be able to write simple MIPS functions and translate them from a higher-level language.

Function Call Convention

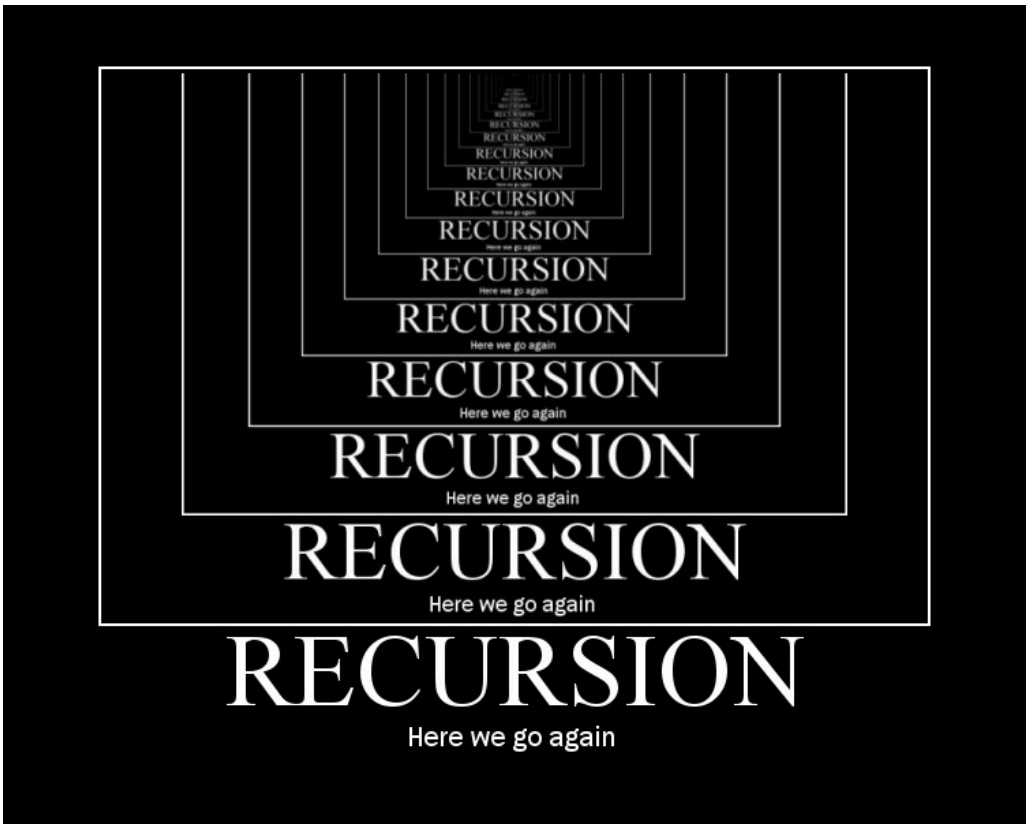
Lets move on to the first topic of the week - Function call convention in MIPS.

Functions are the building blocks for any efficient program. They help cut down on the lines of code and make the code more modularised.
We need to understand how functions work in the back-end in order to translate them from Python into MIPS.

Lets go through the following concepts:

- Caller vs Callee
- The jump and link instruction (`jal`)
- The Frame Pointer (`$fp`), the return address (`$ra`) and the stack pointer (`$sp`)
- The memory diagram
- The order of events in a function call

Recursive Functions in MIPS

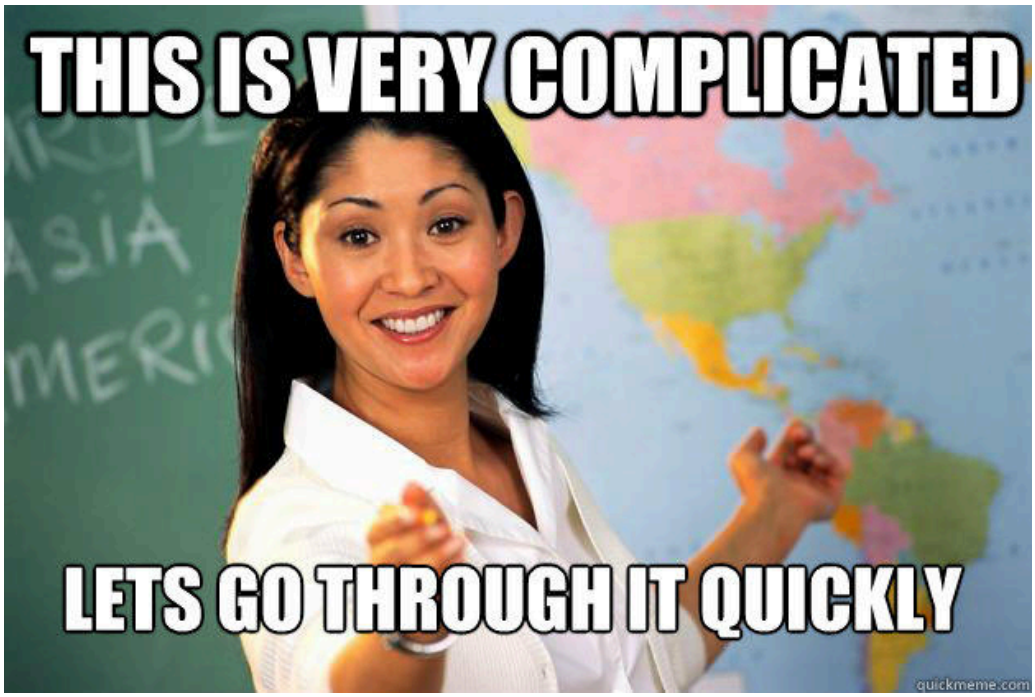


Let us go through some of the concepts of Recursive programming in MIPS.

The main topics to cover are:

- How do recursive programs actually work?
- What does the memory stack look like when running a recursive program in MIPS
- What happens in the base case?
- The information flow in a recursive program in MIPS

Passing lists as arguments to functions



Let's take it one step further, and figure out how to pass lists as arguments to functions in MIPS.

We will cover the following topics:

- Recap on how a list is created in MIPS
- Passing the list address as an argument to the function
- What NOT to do
- How to change the values of the list inside the function

Functions - All inclusive Quiz bonanza!

Alright! Let's go ahead and test our knowledge of function call basics.

Question 1 *Submitted Aug 16th 2022 at 8:20:21 am*

Arrange the events for the caller during a function's calling convention

- Caller Saves temporary registers by pushing them to the stack
- Caller prepares arguments to be added to the stack
- Caller calls the function using the jal instruction
- Caller clears the space created for the arguments
- Caller restores the temporary registers by clearing them off the stack
- Caller uses the return value returned by the function

Question 2 *Submitted Aug 16th 2022 at 8:21:42 am*

Arrange the events for the callee during a function's calling convention

- Callee saves original $fp and $ra to the stack
- Callee saves $sp to $fp
- Callee allocates space for local variables in the stack
- Callee performs the function's task and stores the return value in $v0
- Callee pops off the local variables from the stack
- Callee restores $fp and $ra by popping its saved value off the stack
- Callee returns using the jr $ra instruction

Question 3 *Submitted Aug 16th 2022 at 8:45:10 am*

Consider the following function:

```
def function(x:int, y:int) -> int:
    """ Dummy function to illustrate memory diagrams."""
    if x >= y:
        return x - y
    return 0

def main():
     print(function(27, 2), end = '')

main()
```

Draw the memory diagram for the function above as you enter the function.

Here is a template, copy and paste this in your answer and fill it in. You need to add where `fp`, `ra`, `x` and `y` are

```
    ###### Memory #####
    #     -  ($fp)    #
    #     - 4($fp)    #
    #     - 8($fp)    #
    #     - 12($fp)   #
    ###################
```

fp - ($sp)

ra - 4($sp)

x - 8($fp)

y - 12($fp)

Question 4 *Submitted Aug 16th 2022 at 8:22:01 am*

If the address of an array is in `$t0` then how would you reference the 3^{rd} element of that array?

- ($t0)
- 4($t0)
- 3($t0)
- **16($t0)**

Question 5 *Submitted Aug 16th 2022 at 8:22:15 am*

What register holds the return value when the caller gets control back?

- `$t0`
- `$s0`
- (selected) `$v0`
- `$a0`

Question 6 *Submitted Aug 16th 2022 at 8:22:43 am*

The function calling convention mentions allocating local variables on the stack. What is the definition of a local variable in a MIPS function?

Initialized in function

Question 7 *Submitted Aug 16th 2022 at 8:24:57 am*

How do you pass a list as an argument to a function in a translation from Python to MIPS?

Pass the first byte of the list as an argument which includes the length of the list

Question 8 *Submitted Aug 16th 2022 at 8:27:45 am*

In Week 1 we asked what the difference between recursion and iteration was. Does your knowledge of MIPS now show another (significant) difference between the two approaches? If so, describe it.

Recursion is an entire function stack in memory each time its called while iteration is just repeating the loop in function nth times.

Question 9 *Submitted Aug 16th 2022 at 8:25:43 am*

Recursive functions are always better in time complexity than iterative functions that perform the same task

- True

- [x] False

Question 10 *Submitted Aug 16th 2022 at 8:25:52 am*

Recursive functions are mostly worse in space complexity as compared to their iterative twin

- [x] True
- [ ] False