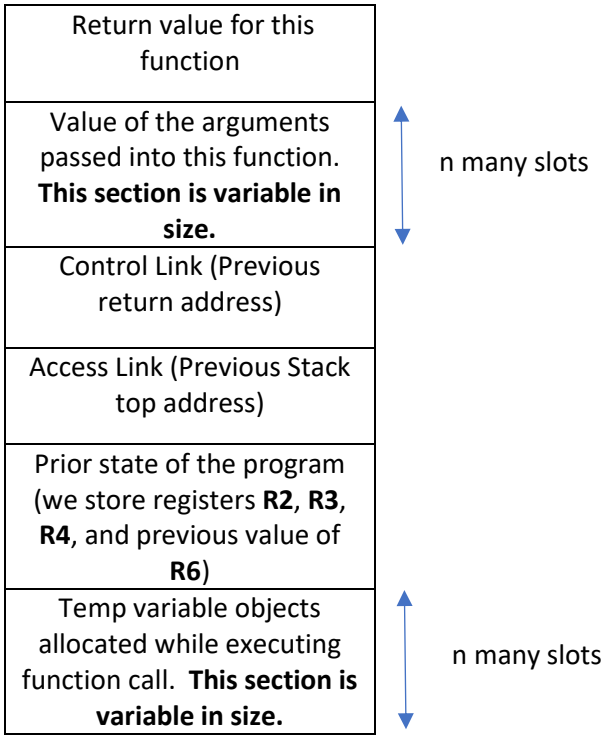
Registers Layout: Note that we frequently use **R1** as a temp register. It should always be considered locally temp, it does not persist across function calls. With this being said, we do use **R1** to store the current status pointer right before we add a new stack frame (which adjust the status pointer). By doing this, we prevent the old status pointer from being overwritten.

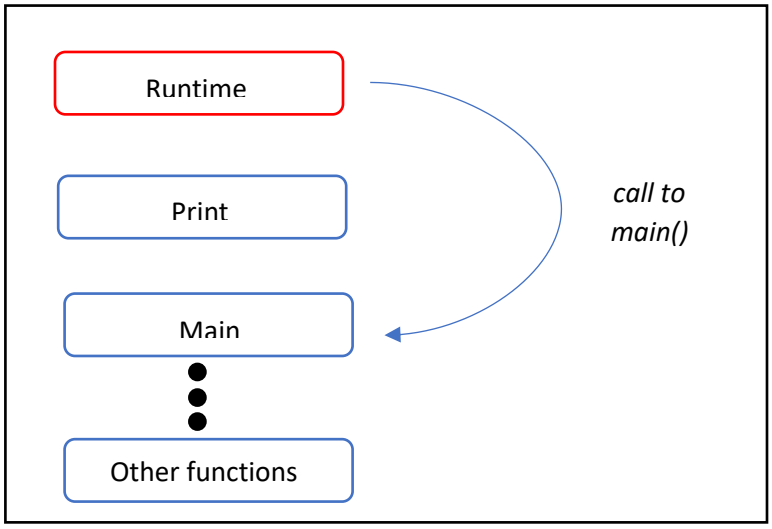
R0: Reserved for 0 value
R1: Temp register. Also used to pass current status pointer into function calls
R2: Final values of operations are stored here.
R3: Left value of register operations are stored here
R4: Right value of register operations are stored here
R5: Top of stack pointer
R6: Status Pointer
R7: Reserved for program counter

Stack Frame Layout: Stack frames are allocated and pushed onto the stack with each function call. This means that to start, the runtime environment pushes a stack frame on for main(). Similarly, print has a stack frame as well whenever it is called.



Runtime environment: The runtime environment sets up the code to call main and puts in a jump statement which will skip over the print statement. The print statement is a fixed size so we know this will work (see below for layout of IMEM). When adding the stack frame for main, we do things a bit differently then every other function call: The registers DMEM[1-n] in DMEM are used to store n-1 command line args, so we copy those arguments down 1 slot in DMEM to make room for the return value of main. Thus the first stack frame starts at DMEM[1]

Layout of IMEM



Function Calls: When a function is called, we first move all the temp args from the caller functions stack frame into the called functions stack frame. We move values from the last n temp variables of caller's stack frame into the (value of args) section of the called stack frame. This is how we pass parameters to functions (see below for a layout of DMEM). Then we store the current access link and the return address into the called functions stack frame so that they are saved. At this point we can store the current status pointer into temp **R1**, adjust the status pointer to the new stack frame, and change the program counter **R7.**

Inside the called function: Once called we copy all the registers we need to (**R2**, **R3**, **R4**, and the old **R6** ~ which is actually in temp **R1** ~). Then we execute code, restore the registers, restore the stack top variable (in the control link) and jump back to the return address (in the access link).

Layout of DMEM: