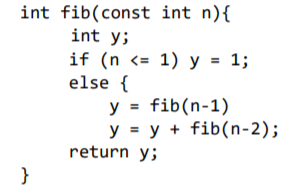
Fibonacci

The original test code that was given in the specification:



Registers required:

* n, determines the element of the series (based on the code
* y, the returned value
* a register to hold
* a register holding the stack pointer

Stack implementation:

* A separate register that will always contain the address pointing to the top of the stack
* A separated region of the RAM, that will become the stack.
* Stack size???
* Stack overflow???
* **STORER RN RS instruction:** Stores the contents of RN to the address contained in RS (stack register). Increments RS by one. (We have 6 free bits, that will be occupied by cond.; We can choose to use one bit that determines whether RS should increment. This means STORER could be a special case of STR which can be more general purpose)
* **LOADR RN RS instruction:** Load the value stored at the address contained in RS to the register RN. Decrementing RS by one. (Same applies here as well)

**CODE IN ASSEMBLY:**

R7=stack register, R6=function return address, R1=n, R2=function return value

**Initializing:**

0SET R7 X //X will be the initial address for the stack pointer

//R8 = stack register

**Main:**

1 SET R1 n //n is stored in R1  
2 CALL R6 4 //call fib, store 3 in R6  
3 STP // R2 will hold the end result

**Fib():**

*if:*4 if n > 1 jump to line 6  
5 SET R2 0x1 //R2 = 1  
6 JUMPR R6 //return from function  
*else:  
 y=fib(n-1);*  
7 STORER R1 RS //save the current value of n for later  
8 sub 1 from R1 //R1 = n-1, now we can call fib(n-1)  
9 STORER R6 RS //save the return address, since R6 will be overwritten  
10 CALL R6 4 //calling fib(n-1), R6 changes to 0x11  
11 LOADR R6 RS //reloading the original return address; (R2=fib(n-1))  
12 LOADR R1 RS //reloading n to R1, fib(n-1) probably changed R1  
13 MOV R3 R2 //R3=R2, R2 will change and we need fib(n-1)  
 *y=y+fib(n-2);*  
14 sub 2 from R1 //R1 = n-2 now we can call fib(n-2)  
15 STORER R6 RS //save the return address, since R6 will be overwritten  
16 CALL R6 4 //calling fib(n-2), R6 changes to 0x17  
17 LOADR R6 RS //reloading the original return address; (R2=fib(n-2))  
17 ADD R2 R3 //R2=R2+R3, R2=fib(n-2) + fib(n-1); overflow???  
*return y;*  
18 JUMPR R6 //return from function; R2=fib(n)

Sources:

<http://www.keil.com/support/man/docs/armasm/armasm_dom1359731124840.htm>

<https://www.youtube.com/watch?v=TnESZpRcE_U&list=LLMXjgFctyn79UnvoKPGdtDQ&index=4&t=0s>