armv7 registers 32 bit registers

a1	r0		
a2	r1		
a3	r2		
a4	r3		
v1	r4		
v2	r5		
v3	r6		
v4	r7		
v5	r8		
v6	r9		
v7	r10	fp	frame pointer
r8	r11	ip	inter process scratch
	r13	sp	stack pointer
	r14	lr	link register(return addr)
	r15	pc	program counter
	cp sr		bit flags register

n	**z**	**c**	**v**				

N negative
Z zero
c carry
v oVerflow
Danger Zone:
The **program counter** is the address of the instruction to load next.
The **link register** stores the return address from a subroutine. It's your breadcrumb.

Stack in linux is full descending
stack pointer points to the last full address
subtract from the sp to add things to the 'bottom' if the stack (ppl may say top of the stack)

byte 3	byte 2	byte 1	byte 0	address
				0xffffffff
				above -4
				-4
				0x00000000

When you put things on the stack, you subtract from **stack pointer** first (sp points to last full) then add your thing.
To return to a register, move the thing, then add 4 (for 32 bits) to the sp
Often you will be pushing and popping registers to preserve them, which will all be 32 bit values, with a lot of +- 4 bytes
be careful when you store things on the stack that are not 32 bits that you are in the part of the stack you think you're in