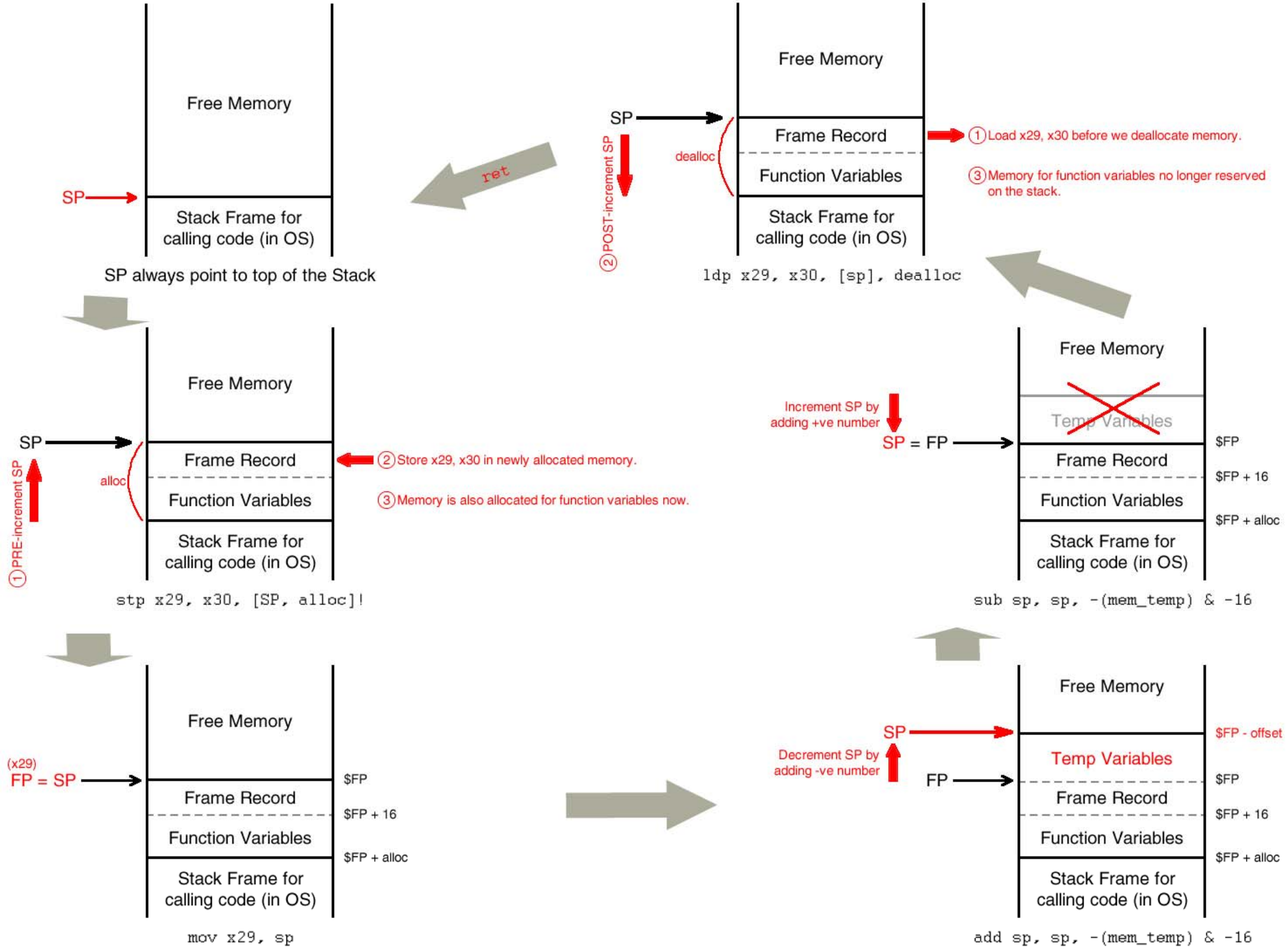
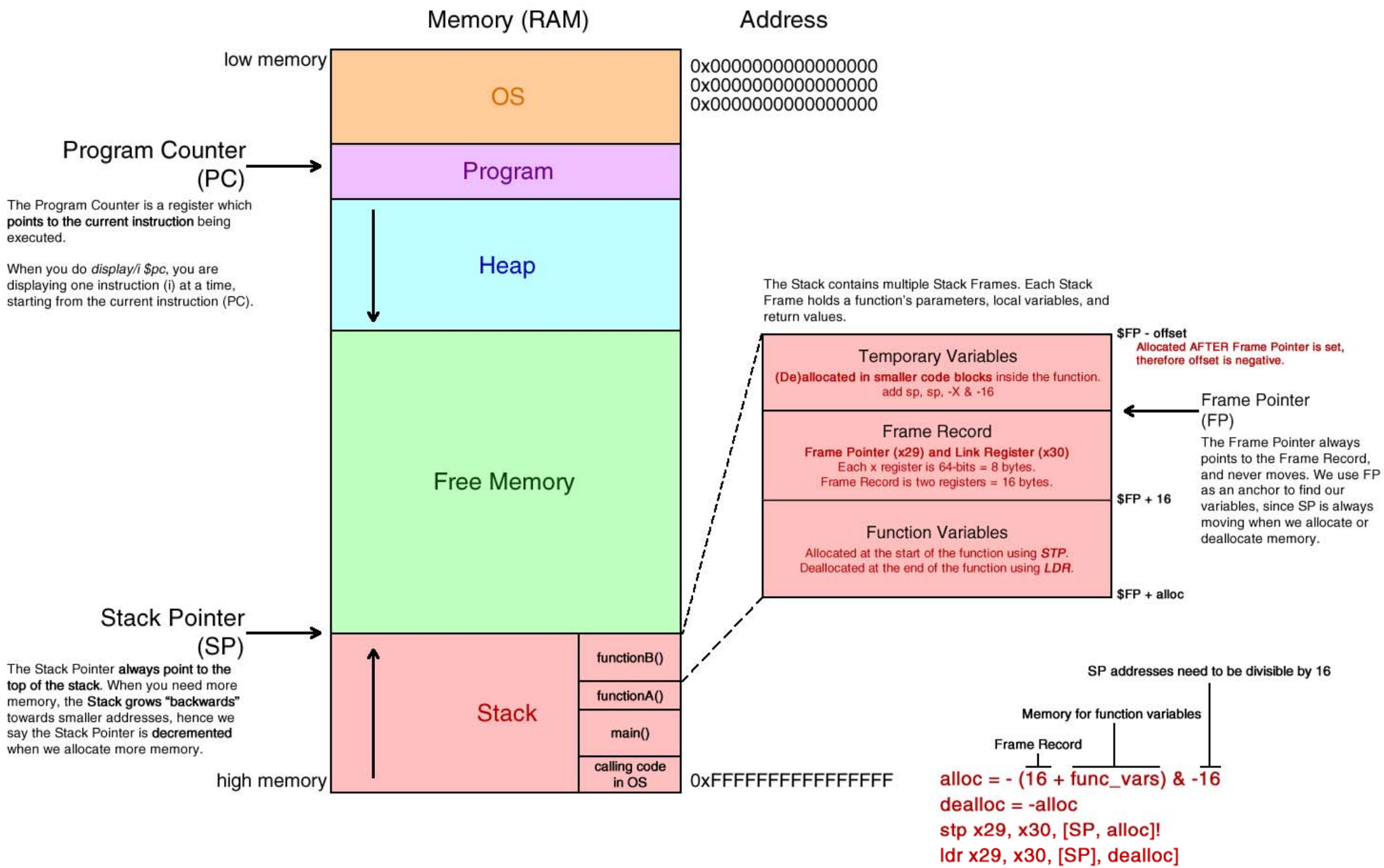

Free Memory
SP
Stack Frame for calling code (in OS)
SP always point to top of the Stack
ret
Free Memory
SP
Frame Record
dealloc
Function Variables
Stack Frame for calling code (in OS)
② POST-increment SP
① Load x29, x30 before we deallocate memory.
③ Memory for function variables no longer reserved on the stack.
ldp x29, x30, [sp], dealloc
Free Memory
SP
Frame Record
alloc
Function Variables
Stack Frame for calling code (in OS)
① PRE-increment SP
② Store x29, x30 in newly allocated memory.
③ Memory is also allocated for function variables now.
stp x29, x30, [SP, alloc]!
Free Memory
Increment SP by adding +ve number
SP = FP
Temp Variables
Frame Record
Function Variables
Stack Frame for calling code (in OS)
$FP
$FP + 16
$FP + alloc
sub sp, sp, -(mem_temp) & -16
Free Memory
(x29)
FP = SP
Frame Record
Function Variables
Stack Frame for calling code (in OS)
$FP
$FP + 16
$FP + alloc
mov x29, sp
Free Memory
SP
Decrement SP by adding -ve number
FP
Temp Variables
Frame Record
Function Variables
Stack Frame for calling code (in OS)
$FP - offset
$FP
$FP + 16
$FP + alloc
add sp, sp, -(mem_temp) & -16


Memory (RAM)
Address
low memory
OS
0x0000000000000000
0x0000000000000000
0x0000000000000000
Program Counter (PC)
Program
The Program Counter is a register which points to the current instruction being executed.
When you do display/i $pc, you are displaying one instruction (i) at a time, starting from the current instruction (PC).
Heap
Free Memory
Stack Pointer (SP)
The Stack Pointer always point to the top of the stack. When you need more memory, the Stack grows "backwards" towards smaller addresses, hence we say the Stack Pointer is decremented when we allocate more memory.
Stack
functionB()
functionA()
main()
calling code in OS
high memory
0xFFFFFFFFFFFFFFFF
The Stack contains multiple Stack Frames. Each Stack Frame holds a function's parameters, local variables, and return values.
$FP - offset
Allocated AFTER Frame Pointer is set, therefore offset is negative.
Temporary Variables
(De)allocated in smaller code blocks inside the function.
add sp, sp, -X & -16
Frame Pointer (FP)
The Frame Pointer always points to the Frame Record, and never moves. We use FP as an anchor to find our variables, since SP is always moving when we allocate or deallocate memory.
Frame Record
Frame Pointer (x29) and Link Register (x30)
Each x register is 64-bits = 8 bytes.
Frame Record is two registers = 16 bytes.
$FP + 16
Function Variables
Allocated at the start of the function using STP.
Deallocated at the end of the function using LDR.
$FP + alloc
SP addresses need to be divisible by 16
Memory for function variables
Frame Record
alloc = - (16 + func_vars) & -16
dealloc = -alloc
stp x29, x30, [SP, alloc]!
ldr x29, x30, [SP], dealloc]