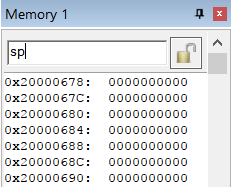
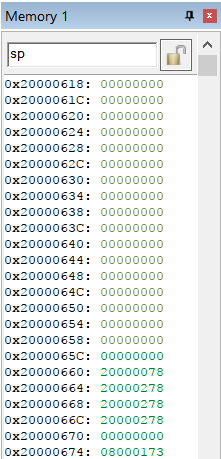
При запуске программы ничего интересного в стеке нет. 

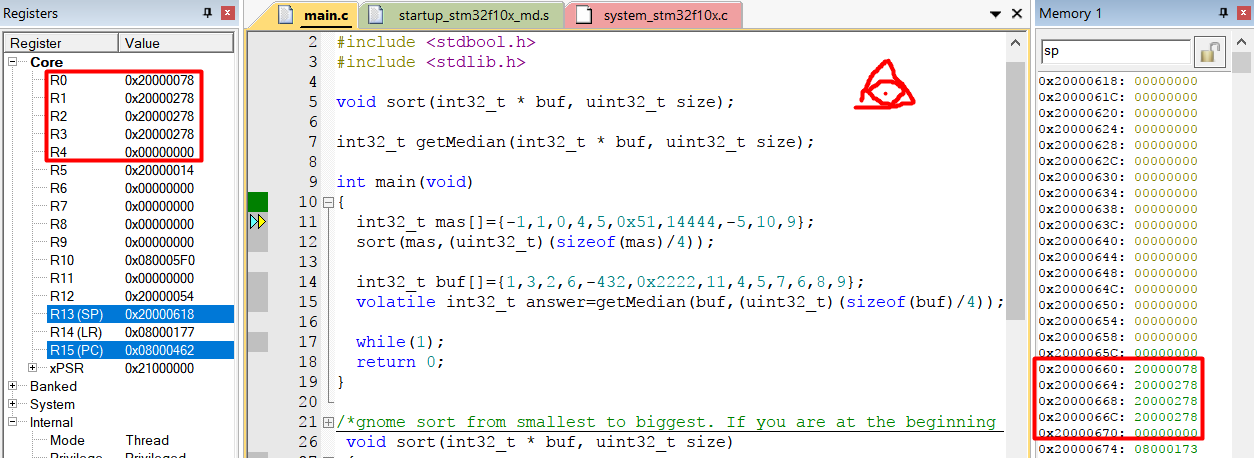
0x08000460 B098 SUB sp,sp,#0x60

Затем в стеке выделяется место под 6\*16 /4=24 четырёхбайтовых значений.

При этом мы можем видеть, что в глубине стека уже есть какие-то значения. Разберёмся:



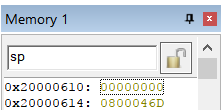
Самое нижнее-0x08000176 F000F9E7 BL.W exit (0x08000548). Предполагаю, что это выход из main как и из любой другой функции.



Почему забиваются ещё числа-не представляю. На лекции было что-то о том, что валяется в регистрах при инициализации. Но всё забывается) Ещё эти значения в стеке удивительно совпадают со значениями в регистре (нули могут быть случайностью, но 2...78-едва ли).

0x08000494 B510 PUSH {r4,lr}

Вызывается функция инициализации массива. При этом в стек закидывается два значения из двух регистров. Они оказываются в вершине стека. Это контекст вызова.



0x080004C6 E8BD4010 POP {r4,lr}

После того, как функция сделала свои дела (создала и инициализировала массив), значения регистров возвращаются из вершины стека в родные ячейки (возвращение контекста). Вершина стека при этом поднимается (стек уменьшается) на 2\*4байта.

0x080004CA EA5F7C82 LSLS r12,r2,#30

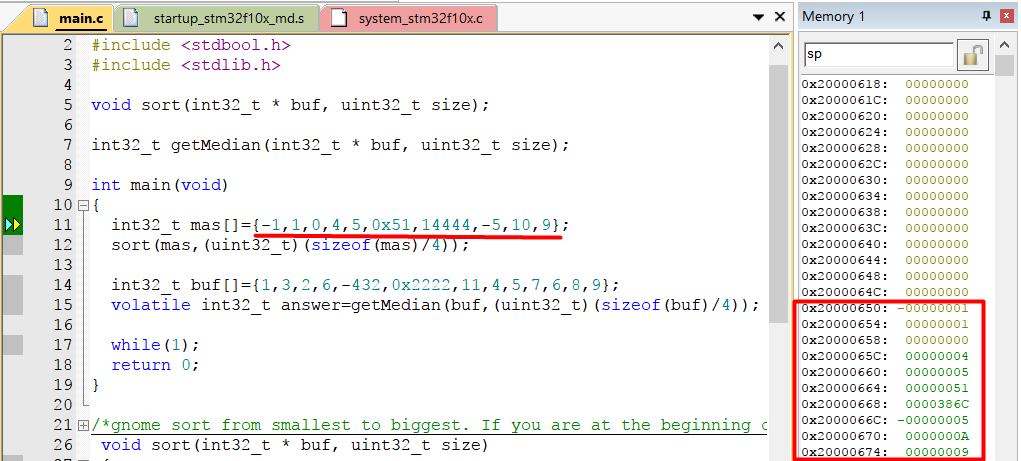
0x080004CE BF24 ITT CS

0x080004D0 F8513B04 LDRCS r3,[r1],#0x04

0x080004D4 F8403B04 STRCS r3,[r0],#0x04

0x080004D8 BF08 IT EQ

0x080004DA 4770 BXEQ lr

Происходят ещё какие-то махинации и только после этого возврат из функции. 

Но в стеке уже располагаются ожидаемые значения. Первый элемент ближе к вершине.

0x0800046C 210A MOVS r1,#0x0A

0x0800046E A80E ADD r0,sp,#0x38

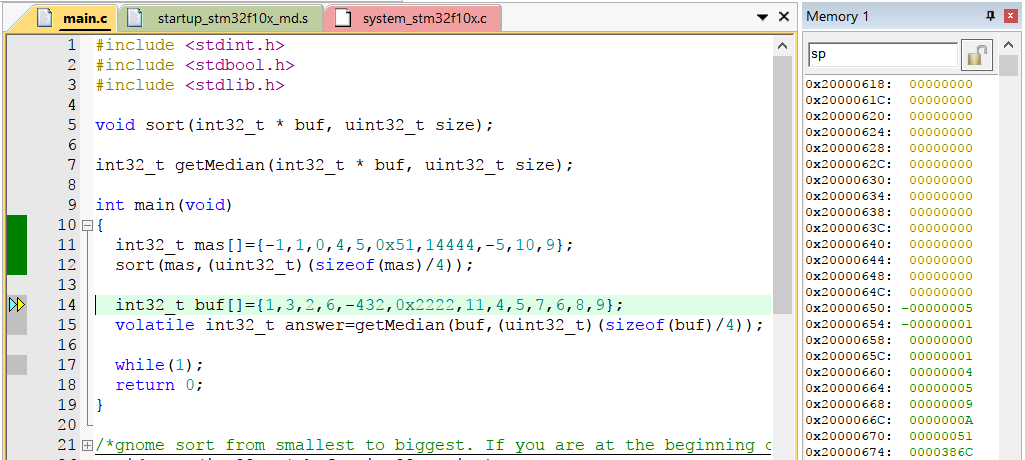
Перед вызовом функции сортировки в регистры заносятся параметры. Это длина массива и адрес его первого элемента.

0x08000428 B530 PUSH {r4-r5,lr}

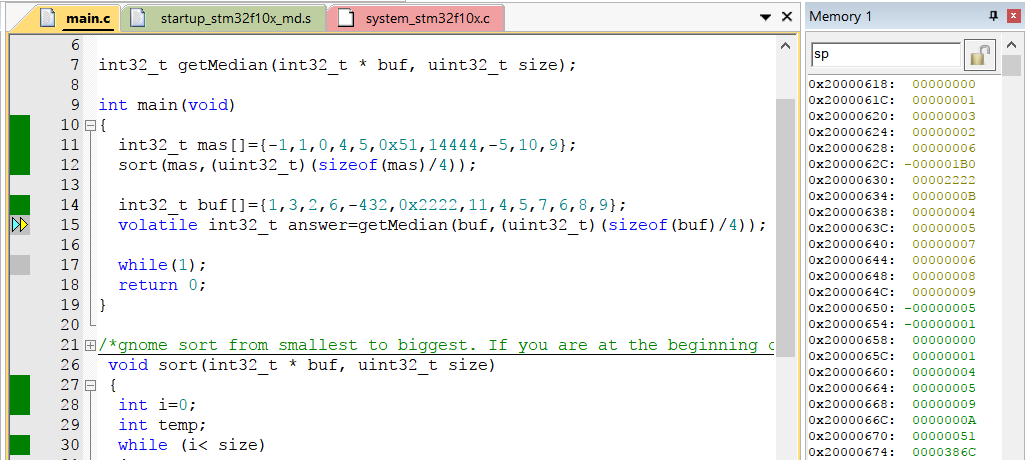
Вызывается функция сортировки. И в стек попадают уже три регистра. Создаётся впечатление, что регистры копируются в стек с наибольшего номера, то есть на вершине стека будет младший регистр. Соответственно первым заносится адрес возврата из функции. При этом отмечаем, что локальные переменные в данной функции хранятся в регистрах. Причиной этому их малое количество. Эти переменные составляют стековый кадр функции.

0x0800045E BD30 POP {r4-r5,pc}

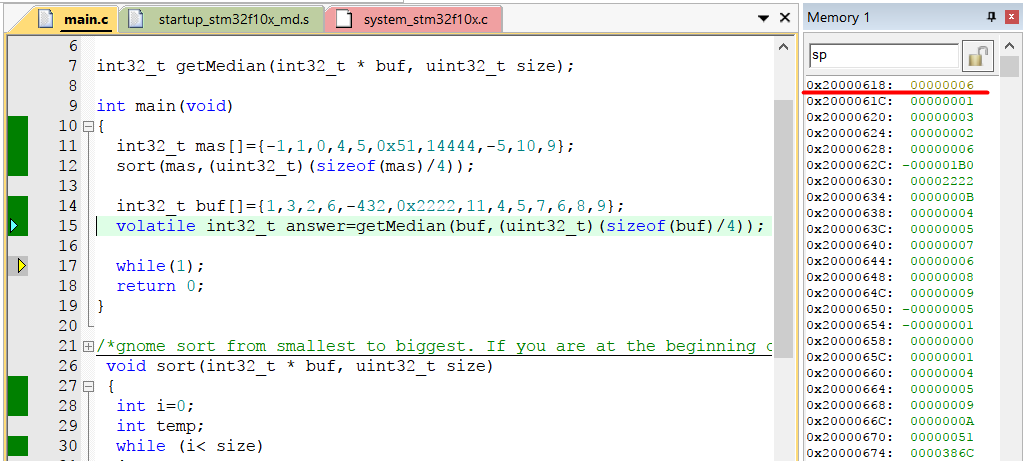
Возвращаем значения из стека в регистры. Последним забирается адрес возврата. Он сразу заносится в регистр pc (в отличие от функции создания массива), чтобы приступить к следующей инструкции.



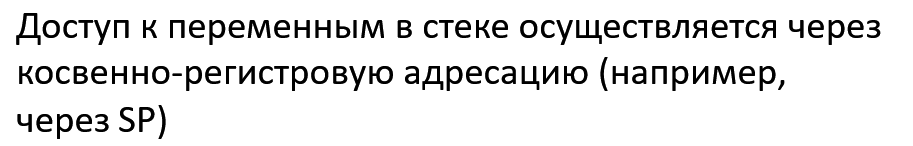
Массив лежит в стеке уже в отсортированном виде.

После инициализации второго массива стек увеличился. 

И напоследок вершина выделенного стекового кадра забивается переменной answer, которая хранит медиану массива buf.



Ужасно сумбурый ответ. И ужасно неточное понимание терминов «стековый кадр» и «контекст вызова». Но я попробую подытожить:

**а)** Покажите, где в стековом кадре функции расположены локальные объекты и что это за объекты.  
Локальные объекты расположены либо в регистрах (например, для функции sort), либо в самом стеке (функция main). Эти объекты-локальные переменные.  
    **б)** Покажите, как происходит выделение памяти для локальных объектов в стековом кадре.  
Выделение происходит простым передвижением указателя на вершину стека (в сторону убывания адреса). Тем самым между предыдущей и новой вершиной остаются незадействованные ячейки памяти, которые и используются для хранения локальных объектов.  
   **в)** Покажите, как происходит освобождение памяти, выделенной под локальные объекты.  
Освобождение памяти происходит передвижением указателя на вершину стека в сторону увеличения адреса. Делаем вид, что тех ячеей не было)  
    **г)** Как происходит обращение к локальным объектам в стековом кадре?  
Например, так. 0x0800046E A80E ADD r0,sp,#0x38. Здесь в нулевой регистр заносится адрес первого элемента массива.   
Собственно, это она и есть.  
   **д)** В каком месте стекового кадра будет расположен сохраняемый контекст вызова? В каком порядке будут сохранены регистры, в каком порядке они будут восстановлены? Какие команды для будут для этого использованы?  
Контекст вызова хранится близ вершины стека). На вершине располагается самый младший сохранённый регистр. То есть первым сохраняется и последним возвращается регистр lr, то есть регистр, хранящий адрес возврата из функции.  
    **е)** Покажите, как стек расположен в памяти, в какую сторону он растет.  
    Растёт в сторону убывания адреса.  
    **ж)** Определите, сколько вложенных вызовов функций происходит в вашей программе? Схематично изобразите последовательность стековых кадров для момента самого глубокого вложенного вызова.  
У меня есть только есть только один уровень вложенности. Это любая функция внутри main. Причём у всех этих функций (кроме main) стековый кадр хранится в регистрах. Попробую изобразить на примере main и sort:

r4  
r3  
r0  
r2

answer  
buf  
mas