ARM Stack

- BOF/RTL/ROP 이해를 위한 -

조 진 성

경희대학교 컴퓨터공학과

Mobile & Embedded System Lab.

Computer Engineering in KyungHee University
Mobile & Embedded System Lab.

ARM Registers

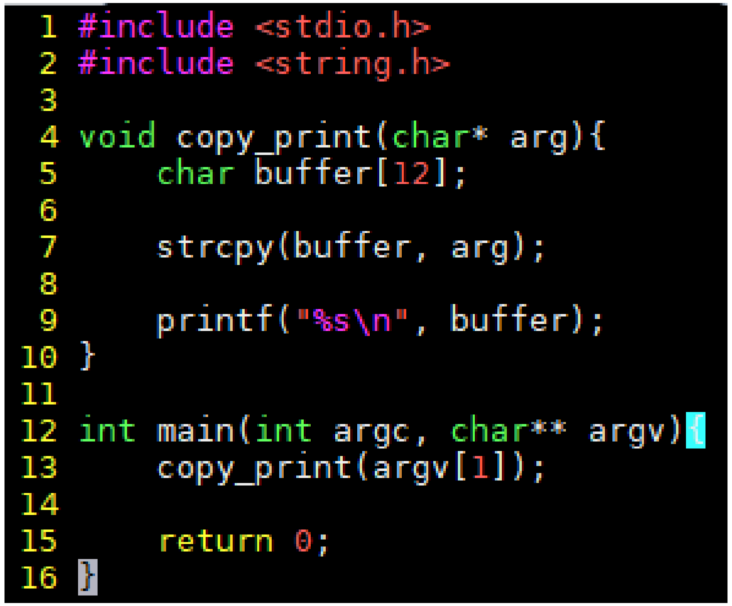
- pc : program counter
- sp : stack pointer
- lr : linked register (return address)
- r0~r12 : general purpose registers

ARM Instructions

- add r11, sp, #4
  - sp에서 4를 더해서 r11에 저장
- sub sp, sp, #8
  - sp에서 8을 빼서 sp에 저장
- str r0, [r11, #-8]
  - r0의 값을 r11에서 8을 뺀 메모리 주소에 저장 (store)
- ldr r3, [r3]
  - r3 값의 메모리 주소에서 데이터를 읽어 r3에 저장 (load)
- mov r0, r3
  - r3의 값을 r0에 저장
- push {r11, lr}
  - Lr과 r11의 값을 stack에 저장 (뒤부터 push)
- pop {r11, pc}
  - Stack에서 pop하여 r11과 pc에 저장 (앞부터 pop)
- bl 0x10cfc <copy_print>
  - 0x10cfc 번지로 branch (with link)

ARM Stack

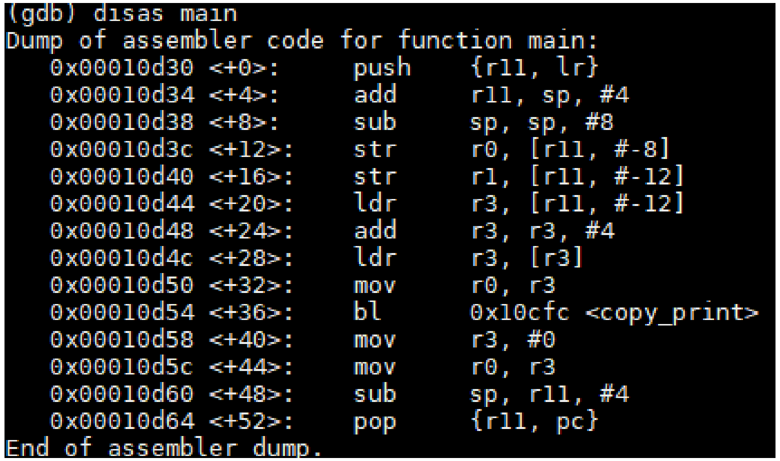
- Sample code
  - gcc –g –o armstack armstack.c
  - ./armstack AAAABBBBCCC
  - argc = 2
  - argv[1] = “AAAABBBBCCC\0”



```c
#include <stdio.h>
#include <string.h>

void copy_print(char* arg){
    char buffer[12];

    strcpy(buffer, arg);

    printf("%s\n", buffer);
}

int main(int argc, char** argv){
    copy_print(argv[1]);

    return 0;
}
```

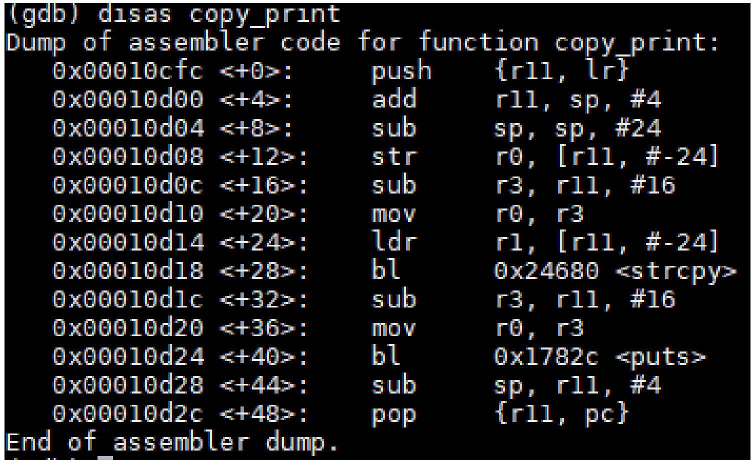
ARM Stack

- Disassemble
  - gdb armstack
  - main



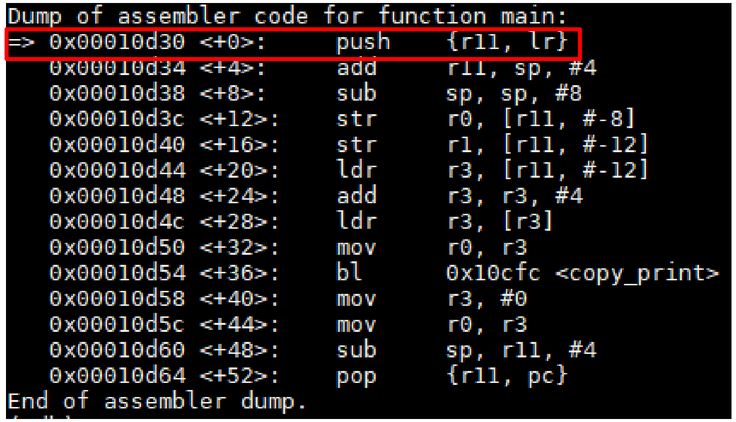
```
(gdb) disas main
Dump of assembler code for function main:
   0x00010d30 <+0>:     push    {r11, lr}
   0x00010d34 <+4>:     add     r11, sp, #4
   0x00010d38 <+8>:     sub     sp, sp, #8
   0x00010d3c <+12>:    str     r0, [r11, #-8]
   0x00010d40 <+16>:    str     r1, [r11, #-12]
   0x00010d44 <+20>:    ldr     r3, [r11, #-12]
   0x00010d48 <+24>:    add     r3, r3, #4
   0x00010d4c <+28>:    ldr     r3, [r3]
   0x00010d50 <+32>:    mov     r0, r3
   0x00010d54 <+36>:    bl      0x10cfc <copy_print>
   0x00010d58 <+40>:    mov     r3, #0
   0x00010d5c <+44>:    mov     r0, r3
   0x00010d60 <+48>:    sub     sp, r11, #4
   0x00010d64 <+52>:    pop     {r11, pc}
End of assembler dump.
```

  - copy_print

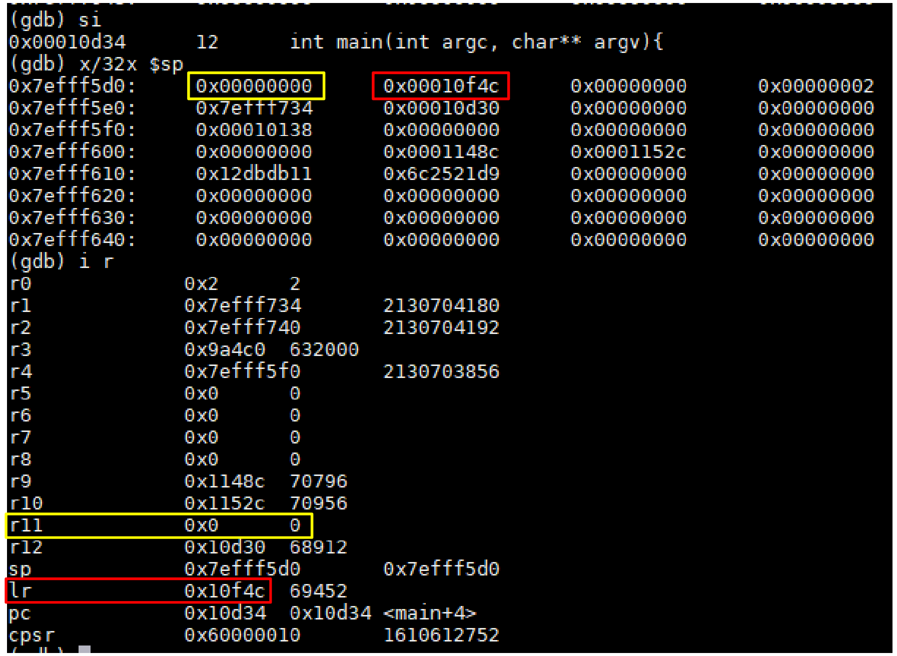


```
(gdb) disas copy_print
Dump of assembler code for function copy_print:
   0x00010cfc <+0>:     push    {r11, lr}
   0x00010d00 <+4>:     add     r11, sp, #4
   0x00010d04 <+8>:     sub     sp, sp, #24
   0x00010d08 <+12>:    str     r0, [r11, #-24]
   0x00010d0c <+16>:    sub     r3, r11, #16
   0x00010d10 <+20>:    mov     r0, r3
   0x00010d14 <+24>:    ldr     r1, [r11, #-24]
   0x00010d18 <+28>:    bl      0x24680 <strcpy>
   0x00010d1c <+32>:    sub     r3, r11, #16
   0x00010d20 <+36>:    mov     r0, r3
   0x00010d24 <+40>:    bl      0x1782c <puts>
   0x00010d28 <+44>:    sub     sp, r11, #4
   0x00010d2c <+48>:    pop     {r11, pc}
End of assembler dump.
```

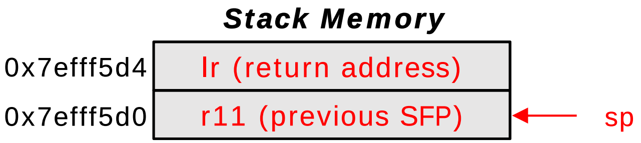
ARM Stack



```
Dump of assembler code for function main:
=> 0x00010d30 <+0>:     push    {r11, lr}
   0x00010d34 <+4>:     add     r11, sp, #4
   0x00010d38 <+8>:     sub     sp, sp, #8
   0x00010d3c <+12>:    str     r0, [r11, #-8]
   0x00010d40 <+16>:    str     r1, [r11, #-12]
   0x00010d44 <+20>:    ldr     r3, [r11, #-12]
   0x00010d48 <+24>:    add     r3, r3, #4
   0x00010d4c <+28>:    ldr     r3, [r3]
   0x00010d50 <+32>:    mov     r0, r3
   0x00010d54 <+36>:    bl      0x10cfc <copy_print>
   0x00010d58 <+40>:    mov     r3, #0
   0x00010d5c <+44>:    mov     r0, r3
   0x00010d60 <+48>:    sub     sp, r11, #4
   0x00010d64 <+52>:    pop     {r11, pc}
End of assembler dump.
```

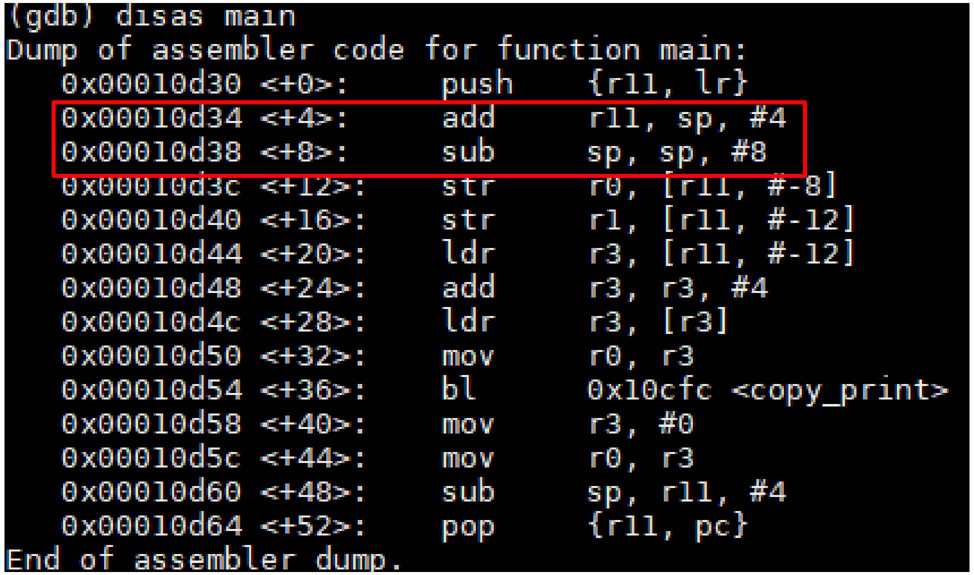


```
(gdb) si
0x00010d34      12      int main(int argc, char** argv){
(gdb) x/32x $sp
0x7efff5d0:     0x00000000      0x00010f4c      0x00000000      0x00000002
0x7efff5e0:     0x7efff734      0x00010d30      0x00000000      0x00000000
0x7efff5f0:     0x00010138      0x00000000      0x00000000      0x00000000
0x7efff600:     0x00000000      0x0001148c      0x0001152c      0x00000000
0x7efff610:     0x12dbdb11      0x6c2521d9      0x00000000      0x00000000
0x7efff620:     0x00000000      0x00000000      0x00000000      0x00000000
0x7efff630:     0x00000000      0x00000000      0x00000000      0x00000000
0x7efff640:     0x00000000      0x00000000      0x00000000      0x00000000
(gdb) i r
r0             0x2      2
r1             0x7efff734       2130704180
r2             0x7efff740       2130704192
r3             0x9a4c0  632000
r4             0x7efff5f0       2130703856
r5             0x0      0
r6             0x0      0
r7             0x0      0
r8             0x0      0
r9             0x1148c  70796
r10            0x1152c  70956
r11            0x0      0
r12            0x10d30  68912
sp             0x7efff5d0       0x7efff5d0
lr             0x10f4c  69452
pc             0x10d34  0x10d34 <main+4>
cpsr           0x60000010       1610612752
```

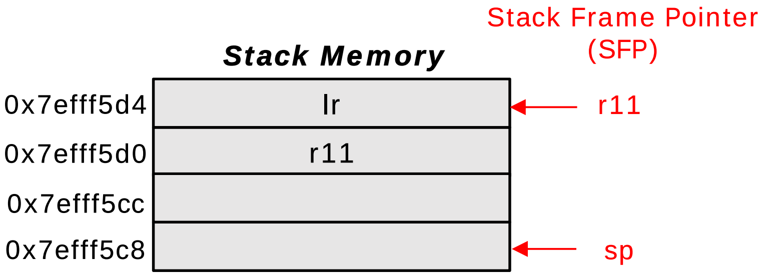


Stack Memory

ARM Stack



```
(gdb) disas main
Dump of assembler code for function main:
   0x00010d30 <+0>:     push    {r11, lr}
   0x00010d34 <+4>:     add     r11, sp, #4
   0x00010d38 <+8>:     sub     sp, sp, #8
   0x00010d3c <+12>:    str     r0, [r11, #-8]
   0x00010d40 <+16>:    str     r1, [r11, #-12]
   0x00010d44 <+20>:    ldr     r3, [r11, #-12]
   0x00010d48 <+24>:    add     r3, r3, #4
   0x00010d4c <+28>:    ldr     r3, [r3]
   0x00010d50 <+32>:    mov     r0, r3
   0x00010d54 <+36>:    bl      0x10cfc <copy_print>
   0x00010d58 <+40>:    mov     r3, #0
   0x00010d5c <+44>:    mov     r0, r3
   0x00010d60 <+48>:    sub     sp, r11, #4
   0x00010d64 <+52>:    pop     {r11, pc}
End of assembler dump.
```

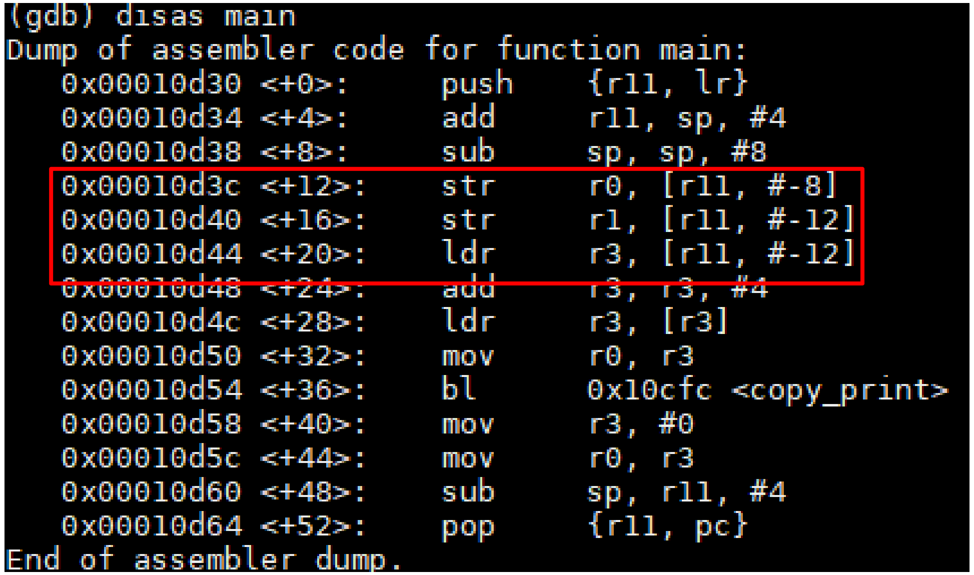


Stack Frame Pointer (SFP)

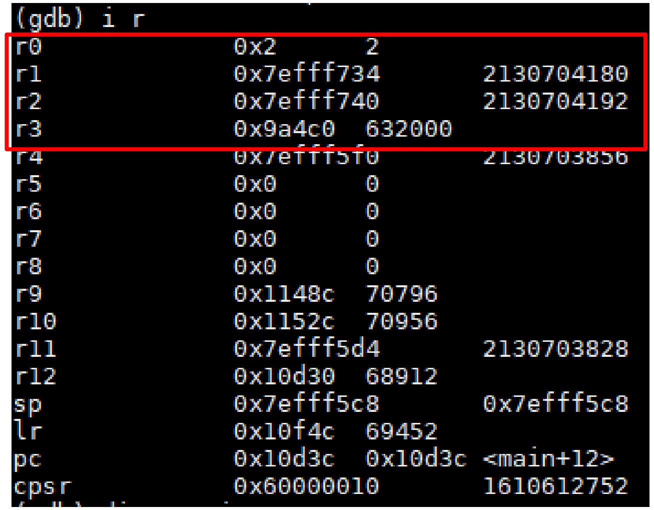
Stack Memory

0x7efff5d4	lr
0x7efff5d0	r11
0x7efff5cc	
0x7efff5c8	

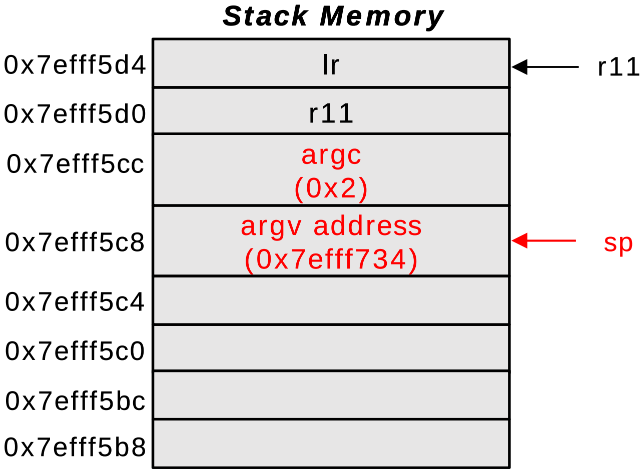
ARM Stack



```
(gdb) disas main
Dump of assembler code for function main:
   0x00010d30 <+0>:     push    {r11, lr}
   0x00010d34 <+4>:     add     r11, sp, #4
   0x00010d38 <+8>:     sub     sp, sp, #8
   0x00010d3c <+12>:    str     r0, [r11, #-8]
   0x00010d40 <+16>:    str     r1, [r11, #-12]
   0x00010d44 <+20>:    ldr     r3, [r11, #-12]
   0x00010d48 <+24>:    add     r3, r3, #4
   0x00010d4c <+28>:    ldr     r3, [r3]
   0x00010d50 <+32>:    mov     r0, r3
   0x00010d54 <+36>:    bl      0x10cfc <copy_print>
   0x00010d58 <+40>:    mov     r3, #0
   0x00010d5c <+44>:    mov     r0, r3
   0x00010d60 <+48>:    sub     sp, r11, #4
   0x00010d64 <+52>:    pop     {r11, pc}
End of assembler dump.
```



```
(gdb) i r
r0             0x2      2
r1             0x7efff734       2130704180
r2             0x7efff740       2130704192
r3             0x9a4c0  632000
r4             0x7efff5f0       2130703856
r5             0x0      0
r6             0x0      0
r7             0x0      0
r8             0x0      0
r9             0x1148c  70796
r10            0x1152c  70956
r11            0x7efff5d4       2130703828
r12            0x10d30  68912
sp             0x7efff5c8       0x7efff5c8
lr             0x10f4c  69452
pc             0x10d3c  0x10d3c <main+12>
cpsr           0x60000010       1610612752
```

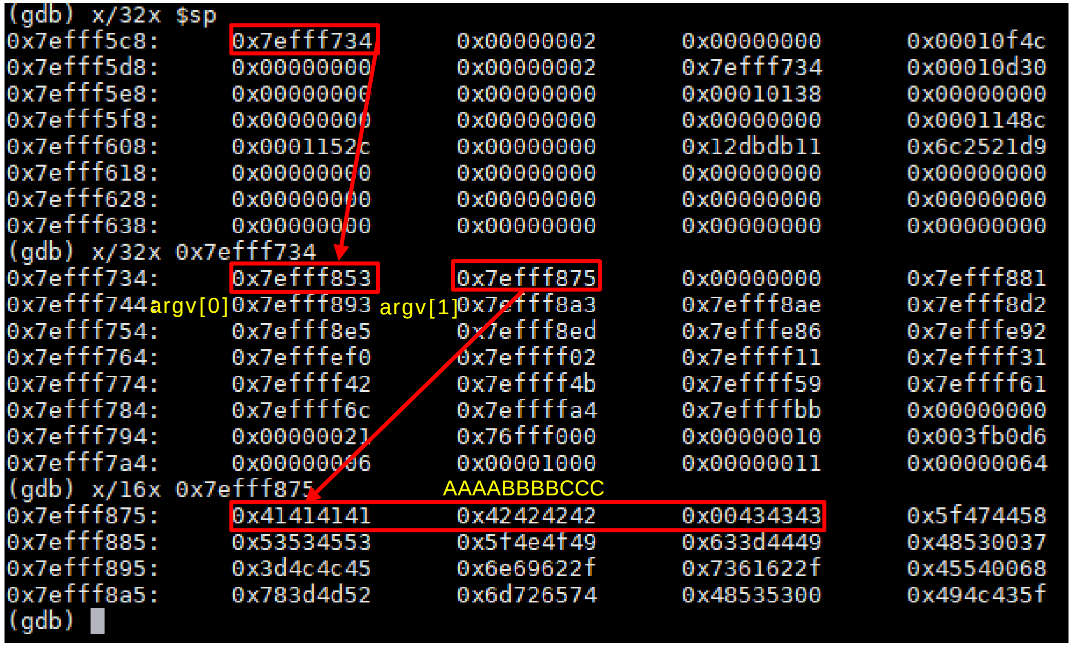


Stack Memory

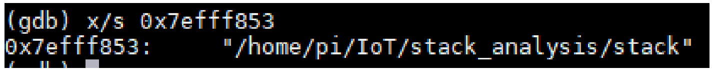
0x7efff5d4	lr	← r11
0x7efff5d0	r11	
0x7efff5cc	argc (0x2)	
0x7efff5c8	argv address (0x7efff734)	← sp
0x7efff5c4		
0x7efff5c0		
0x7efff5bc		
0x7efff5b8		

r3 = 0x7efff734(argv address)

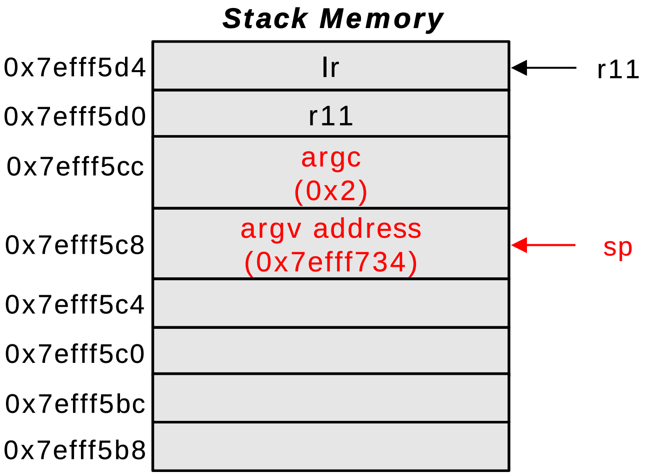
ARM Stack



```
(gdb) x/32x $sp
0x7efff5c8:     0x7efff734      0x00000002      0x00000000      0x00010f4c
0x7efff5d8:     0x00000000      0x00000002      0x7efff734      0x00010d30
0x7efff5e8:     0x00000000      0x00000000      0x00010138      0x00000000
0x7efff5f8:     0x00000000      0x00000000      0x00000000      0x0001148c
0x7efff608:     0x0001152c      0x00000000      0x12dbdb11      0x6c2521d9
0x7efff618:     0x00000000      0x00000000      0x00000000      0x00000000
0x7efff628:     0x00000000      0x00000000      0x00000000      0x00000000
0x7efff638:     0x00000000      0x00000000      0x00000000      0x00000000
(gdb) x/32x 0x7efff734
0x7efff734:     0x7efff853      0x7efff875      0x00000000      0x7efff881
0x7efff744:     0x7efff893      0x7efff8a3      0x7efff8ae      0x7efff8d2
0x7efff754:     0x7efff8e5      0x7efff8ed      0x7efffe86      0x7efffe92
0x7efff764:     0x7efffef0      0x7effff02      0x7effff11      0x7effff31
0x7efff774:     0x7effff42      0x7effff4b      0x7effff59      0x7effff61
0x7efff784:     0x7effff6c      0x7effffa4      0x7effffbb      0x00000000
0x7efff794:     0x00000021      0x76fff000      0x00000010      0x003fb0d6
0x7efff7a4:     0x00000006      0x00001000      0x00000011      0x00000064
(gdb) x/16x 0x7efff875
0x7efff875:     0x41414141      0x42424242      0x00434343      0x5f474458
0x7efff885:     0x53534553      0x5f4e4f49      0x633d4449      0x48530037
0x7efff895:     0x3d4c4c45      0x6e69622f      0x7361622f      0x45540068
0x7efff8a5:     0x783d4d52      0x6d726574      0x48535300      0x494c435f
(gdb)
```



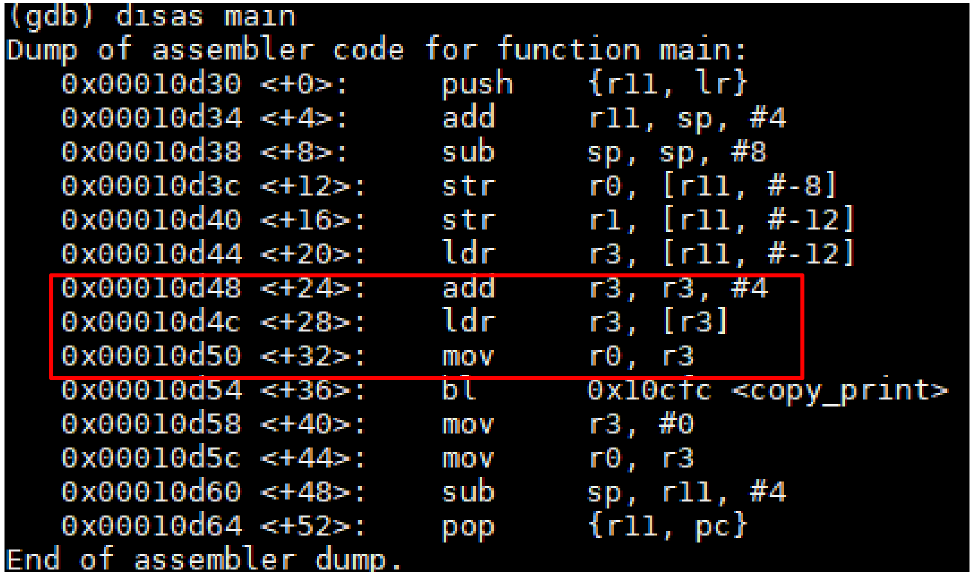
```
(gdb) x/s 0x7efff853
0x7efff853:     "/home/pi/IoT/stack_analysis/stack"
```



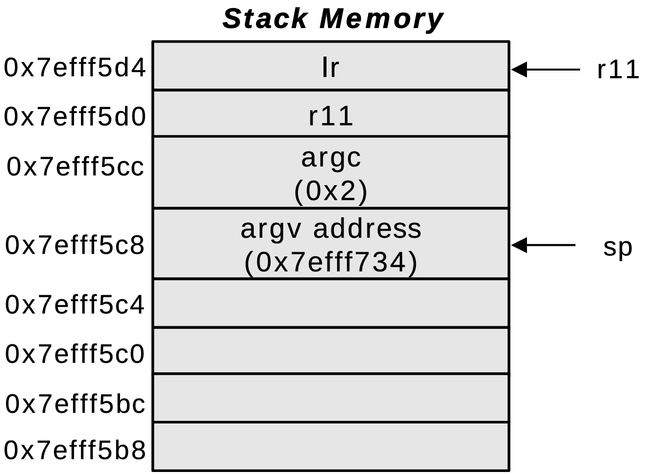
	Stack Memory	
0x7efff5d4	lr	r11
0x7efff5d0	r11	
0x7efff5cc	argc (0x2)	
0x7efff5c8	argv address (0x7efff734)	sp
0x7efff5c4		
0x7efff5c0		
0x7efff5bc		
0x7efff5b8		

r3 = 0x7efff734(argv address)

ARM Stack



```
(gdb) disas main
Dump of assembler code for function main:
   0x00010d30 <+0>:     push    {r11, lr}
   0x00010d34 <+4>:     add     r11, sp, #4
   0x00010d38 <+8>:     sub     sp, sp, #8
   0x00010d3c <+12>:    str     r0, [r11, #-8]
   0x00010d40 <+16>:    str     r1, [r11, #-12]
   0x00010d44 <+20>:    ldr     r3, [r11, #-12]
   0x00010d48 <+24>:    add     r3, r3, #4
   0x00010d4c <+28>:    ldr     r3, [r3]
   0x00010d50 <+32>:    mov     r0, r3
   0x00010d54 <+36>:    bl      0x10cfc <copy_print>
   0x00010d58 <+40>:    mov     r3, #0
   0x00010d5c <+44>:    mov     r0, r3
   0x00010d60 <+48>:    sub     sp, r11, #4
   0x00010d64 <+52>:    pop     {r11, pc}
End of assembler dump.
```



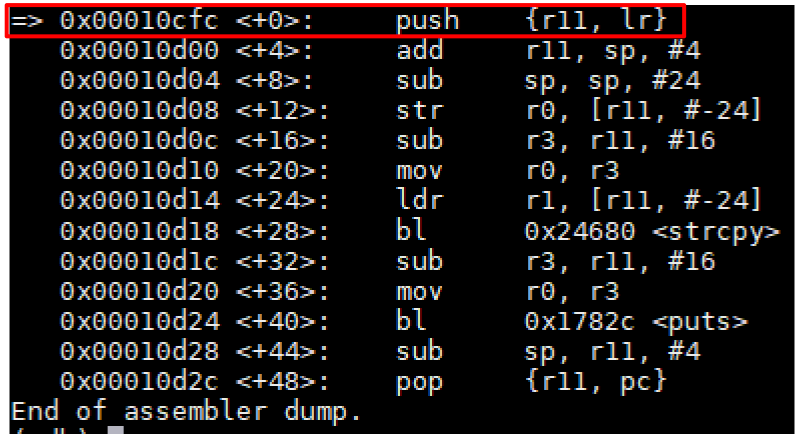
add r3, r3, #4
        r3 = 0x7efff738 (argv[1] address)
ldr r3, [r3]
        r3 = stack[0x7efff738] = 0x7efff875

r0 = address of argv[1]

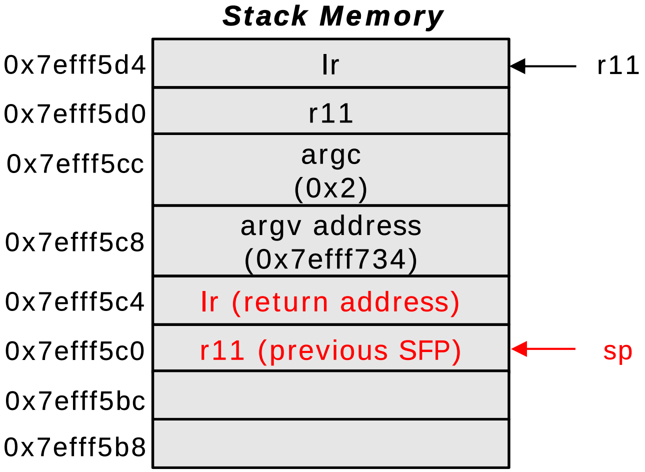
⬇

copy_print(argv[1])

ARM Stack



```
=> 0x00010cfc <+0>:     push    {r11, lr}
   0x00010d00 <+4>:     add     r11, sp, #4
   0x00010d04 <+8>:     sub     sp, sp, #24
   0x00010d08 <+12>:    str     r0, [r11, #-24]
   0x00010d0c <+16>:    sub     r3, r11, #16
   0x00010d10 <+20>:    mov     r0, r3
   0x00010d14 <+24>:    ldr     r1, [r11, #-24]
   0x00010d18 <+28>:    bl      0x24680 <strcpy>
   0x00010d1c <+32>:    sub     r3, r11, #16
   0x00010d20 <+36>:    mov     r0, r3
   0x00010d24 <+40>:    bl      0x1782c <puts>
   0x00010d28 <+44>:    sub     sp, r11, #4
   0x00010d2c <+48>:    pop     {r11, pc}
End of assembler dump.
```

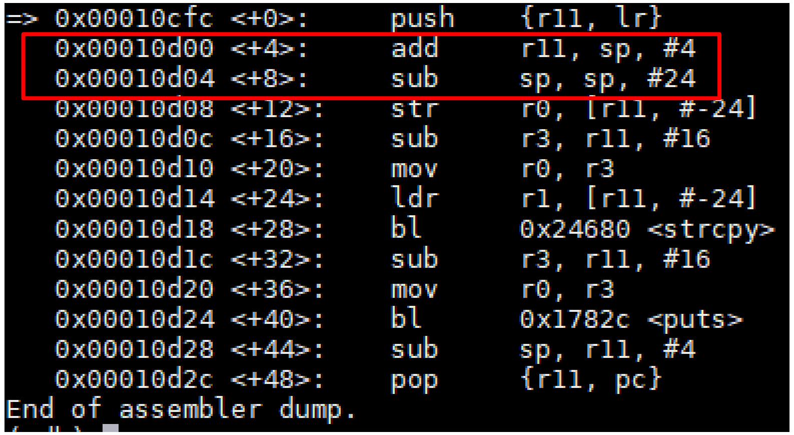


Stack Memory

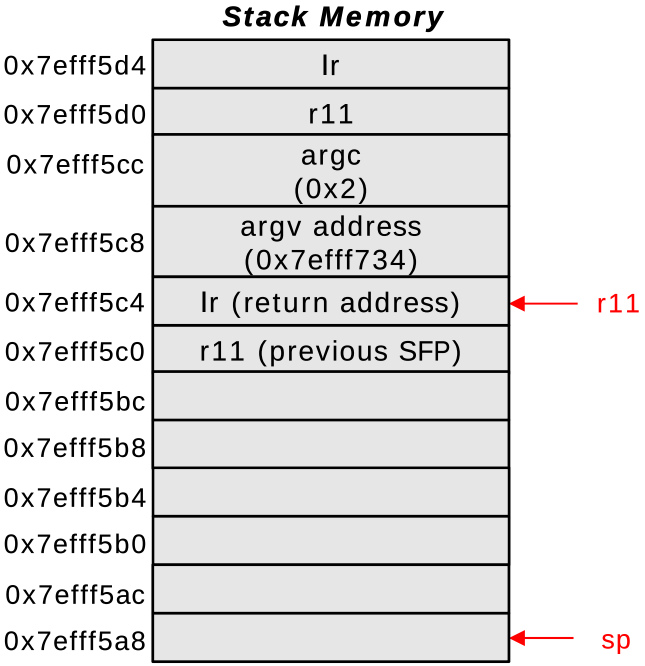
0x7efff5d4	lr	r11
0x7efff5d0	r11	
0x7efff5cc	argc (0x2)	
0x7efff5c8	argv address (0x7efff734)	
0x7efff5c4	lr (return address)	
0x7efff5c0	r11 (previous SFP)	sp
0x7efff5bc		
0x7efff5b8		

r0 = address of argv[1]

ARM Stack



```
=> 0x00010cfc <+0>:     push    {r11, lr}
   0x00010d00 <+4>:     add     r11, sp, #4
   0x00010d04 <+8>:     sub     sp, sp, #24
   0x00010d08 <+12>:    str     r0, [r11, #-24]
   0x00010d0c <+16>:    sub     r3, r11, #16
   0x00010d10 <+20>:    mov     r0, r3
   0x00010d14 <+24>:    ldr     r1, [r11, #-24]
   0x00010d18 <+28>:    bl      0x24680 <strcpy>
   0x00010d1c <+32>:    sub     r3, r11, #16
   0x00010d20 <+36>:    mov     r0, r3
   0x00010d24 <+40>:    bl      0x1782c <puts>
   0x00010d28 <+44>:    sub     sp, r11, #4
   0x00010d2c <+48>:    pop     {r11, pc}
End of assembler dump.
```

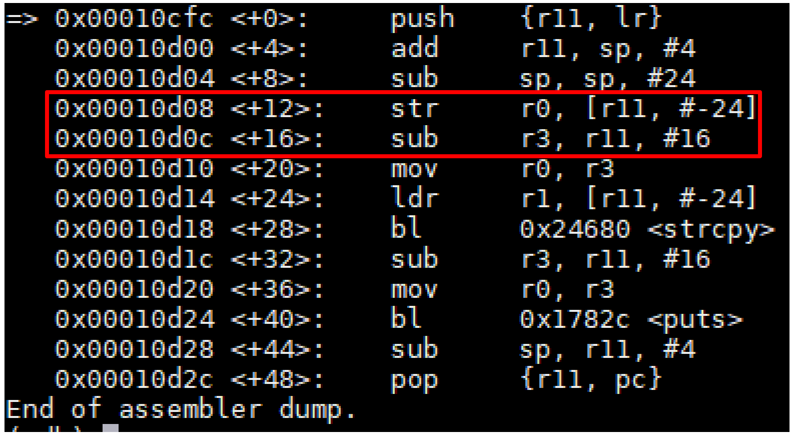


Stack Memory

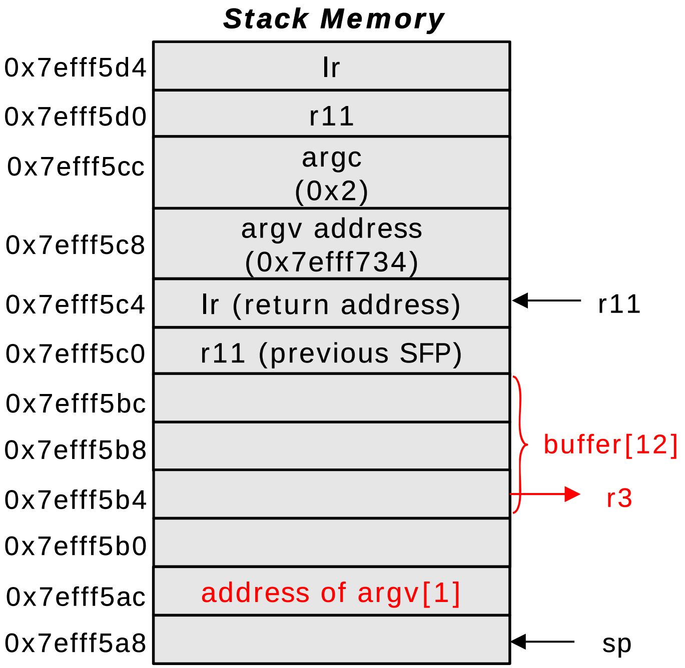
0x7efff5d4	lr	
0x7efff5d0	r11	
0x7efff5cc	argc (0x2)	
0x7efff5c8	argv address (0x7efff734)	
0x7efff5c4	lr (return address)	r11
0x7efff5c0	r11 (previous SFP)	
0x7efff5bc		
0x7efff5b8		
0x7efff5b4		
0x7efff5b0		
0x7efff5ac		
0x7efff5a8		sp

r0 = address of argv[1]

ARM Stack



```
=> 0x00010cfc <+0>:     push    {r11, lr}
   0x00010d00 <+4>:     add     r11, sp, #4
   0x00010d04 <+8>:     sub     sp, sp, #24
   0x00010d08 <+12>:    str     r0, [r11, #-24]
   0x00010d0c <+16>:    sub     r3, r11, #16
   0x00010d10 <+20>:    mov     r0, r3
   0x00010d14 <+24>:    ldr     r1, [r11, #-24]
   0x00010d18 <+28>:    bl      0x24680 <strcpy>
   0x00010d1c <+32>:    sub     r3, r11, #16
   0x00010d20 <+36>:    mov     r0, r3
   0x00010d24 <+40>:    bl      0x1782c <puts>
   0x00010d28 <+44>:    sub     sp, r11, #4
   0x00010d2c <+48>:    pop     {r11, pc}
End of assembler dump.
```

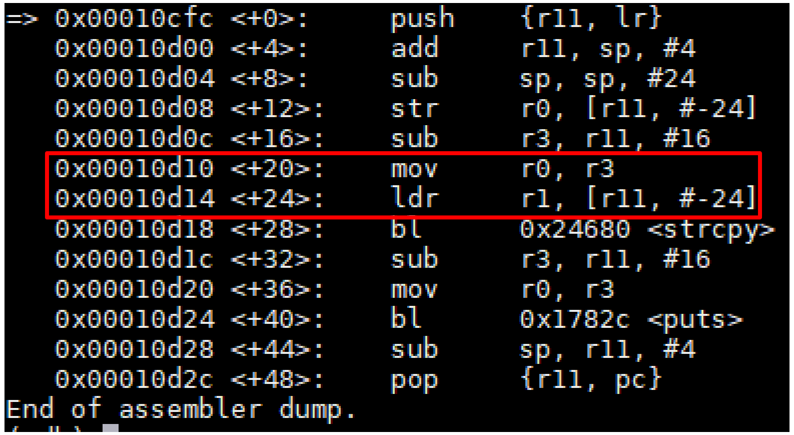


Stack Memory

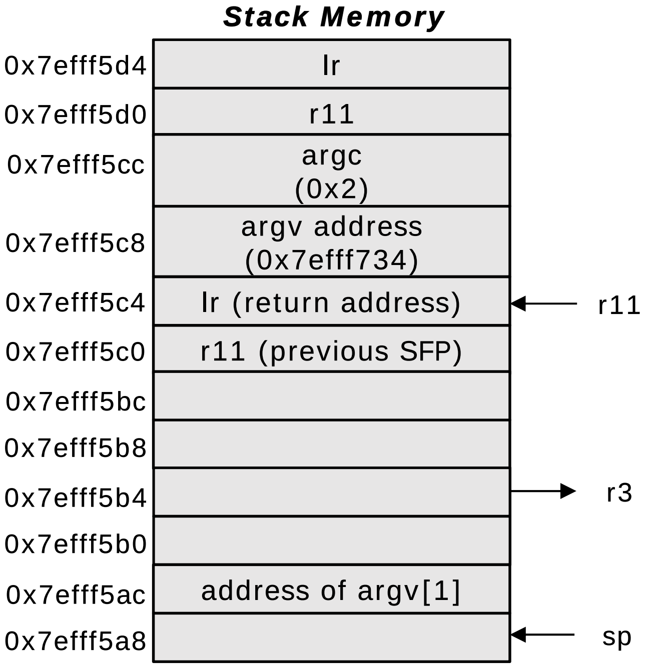
0x7efff5d4	lr
0x7efff5d0	r11
0x7efff5cc	argc (0x2)
0x7efff5c8	argv address (0x7efff734)
0x7efff5c4	lr (return address)
0x7efff5c0	r11 (previous SFP)
0x7efff5bc	
0x7efff5b8	
0x7efff5b4	
0x7efff5b0	
0x7efff5ac	address of argv[1]
0x7efff5a8	

r0 = address of argv[1]

ARM Stack



```
=> 0x00010cfc <+0>:     push    {r11, lr}
   0x00010d00 <+4>:     add     r11, sp, #4
   0x00010d04 <+8>:     sub     sp, sp, #24
   0x00010d08 <+12>:    str     r0, [r11, #-24]
   0x00010d0c <+16>:    sub     r3, r11, #16
   0x00010d10 <+20>:    mov     r0, r3
   0x00010d14 <+24>:    ldr     r1, [r11, #-24]
   0x00010d18 <+28>:    bl      0x24680 <strcpy>
   0x00010d1c <+32>:    sub     r3, r11, #16
   0x00010d20 <+36>:    mov     r0, r3
   0x00010d24 <+40>:    bl      0x1782c <puts>
   0x00010d28 <+44>:    sub     sp, r11, #4
   0x00010d2c <+48>:    pop     {r11, pc}
End of assembler dump.
```



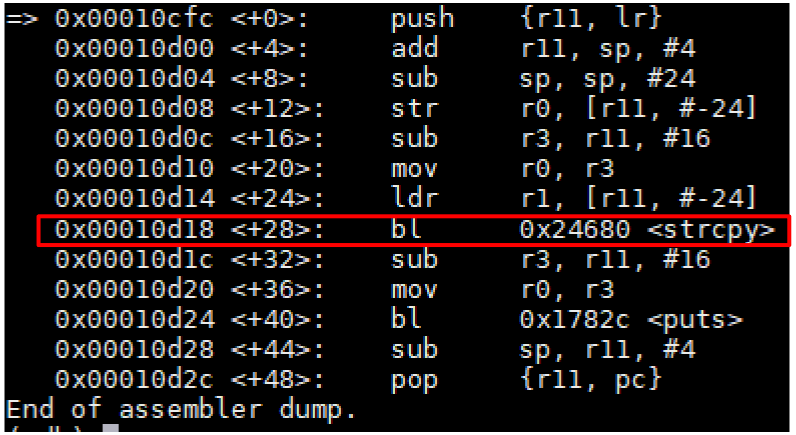
Stack Memory

0x7efff5d4	lr	
0x7efff5d0	r11	
0x7efff5cc	argc (0x2)	
0x7efff5c8	argv address (0x7efff734)	
0x7efff5c4	lr (return address)	← r11
0x7efff5c0	r11 (previous SFP)	
0x7efff5bc		
0x7efff5b8		
0x7efff5b4		→ r3
0x7efff5b0		
0x7efff5ac	address of argv[1]	
0x7efff5a8		← sp

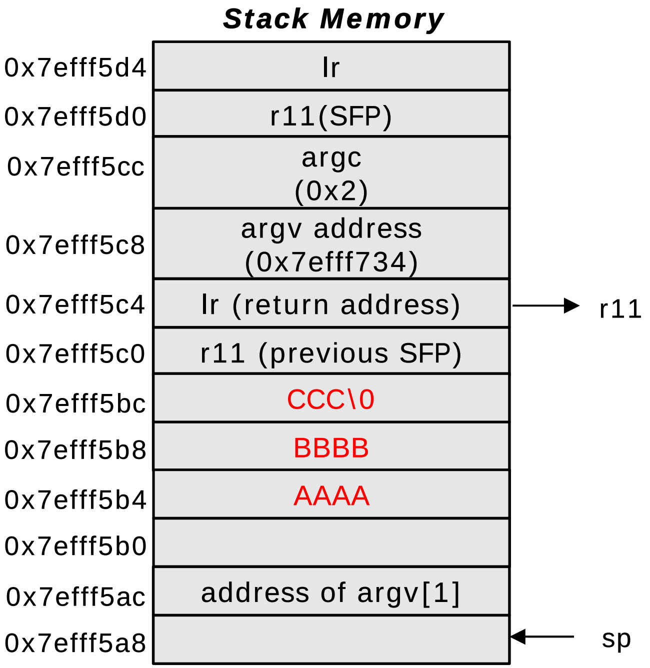
r0 = address of buffer
r1 = address of argv[1]

strcpy(buffer, argv[1])

ARM Stack



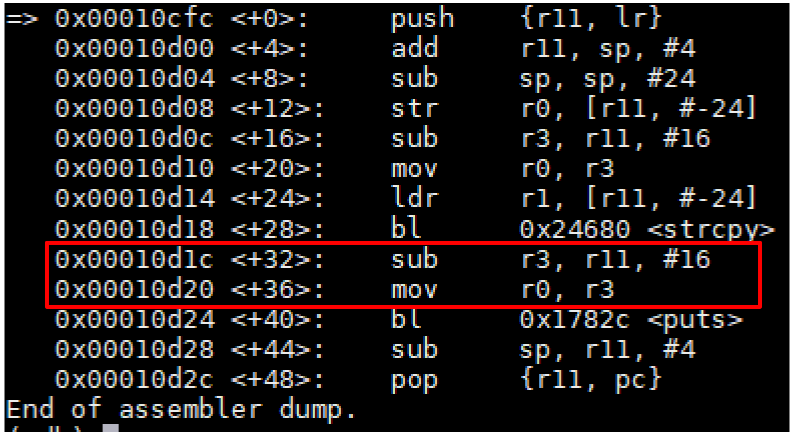
```
=> 0x00010cfc <+0>:     push    {r11, lr}
   0x00010d00 <+4>:     add     r11, sp, #4
   0x00010d04 <+8>:     sub     sp, sp, #24
   0x00010d08 <+12>:    str     r0, [r11, #-24]
   0x00010d0c <+16>:    sub     r3, r11, #16
   0x00010d10 <+20>:    mov     r0, r3
   0x00010d14 <+24>:    ldr     r1, [r11, #-24]
   0x00010d18 <+28>:    bl      0x24680 <strcpy>
   0x00010d1c <+32>:    sub     r3, r11, #16
   0x00010d20 <+36>:    mov     r0, r3
   0x00010d24 <+40>:    bl      0x1782c <puts>
   0x00010d28 <+44>:    sub     sp, r11, #4
   0x00010d2c <+48>:    pop     {r11, pc}
End of assembler dump.
```



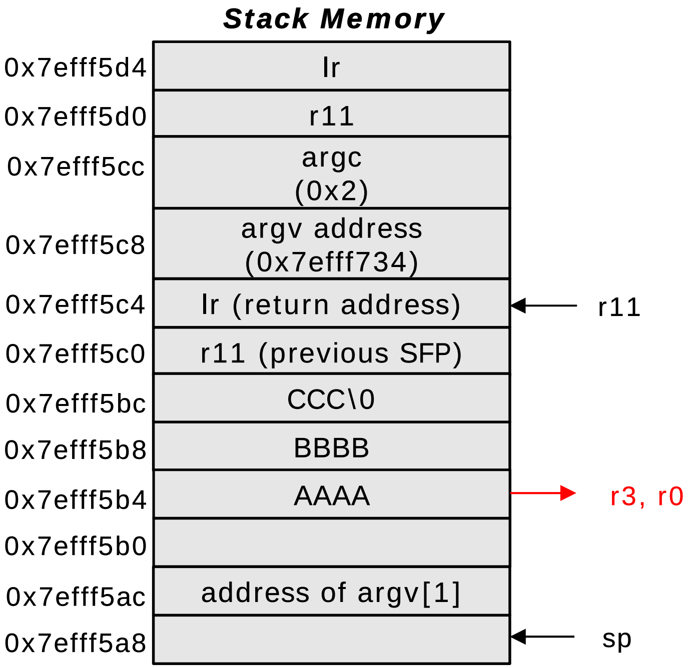
Stack Memory

0x7efff5d4	lr	
0x7efff5d0	r11(SFP)	
0x7efff5cc	argc (0x2)	
0x7efff5c8	argv address (0x7efff734)	
0x7efff5c4	lr (return address)	→ r11
0x7efff5c0	r11 (previous SFP)	
0x7efff5bc	CCC\0	
0x7efff5b8	BBBB	
0x7efff5b4	AAAA	
0x7efff5b0		
0x7efff5ac	address of argv[1]	
0x7efff5a8		← sp

ARM Stack



```
=> 0x00010cfc <+0>:     push    {r11, lr}
   0x00010d00 <+4>:     add     r11, sp, #4
   0x00010d04 <+8>:     sub     sp, sp, #24
   0x00010d08 <+12>:    str     r0, [r11, #-24]
   0x00010d0c <+16>:    sub     r3, r11, #16
   0x00010d10 <+20>:    mov     r0, r3
   0x00010d14 <+24>:    ldr     r1, [r11, #-24]
   0x00010d18 <+28>:    bl      0x24680 <strcpy>
   0x00010d1c <+32>:    sub     r3, r11, #16
   0x00010d20 <+36>:    mov     r0, r3
   0x00010d24 <+40>:    bl      0x1782c <puts>
   0x00010d28 <+44>:    sub     sp, r11, #4
   0x00010d2c <+48>:    pop     {r11, pc}
End of assembler dump.
```



Stack Memory

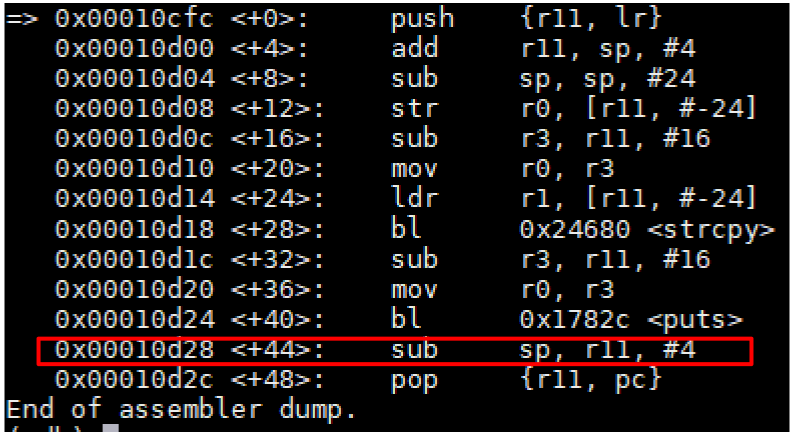
0x7efff5d4	lr	
0x7efff5d0	r11	
0x7efff5cc	argc (0x2)	
0x7efff5c8	argv address (0x7efff734)	
0x7efff5c4	lr (return address)	← r11
0x7efff5c0	r11 (previous SFP)	
0x7efff5bc	CCC\0	
0x7efff5b8	BBBB	
0x7efff5b4	AAAA	→ r3, r0
0x7efff5b0		
0x7efff5ac	address of argv[1]	
0x7efff5a8		← sp

r0 = address of buffer

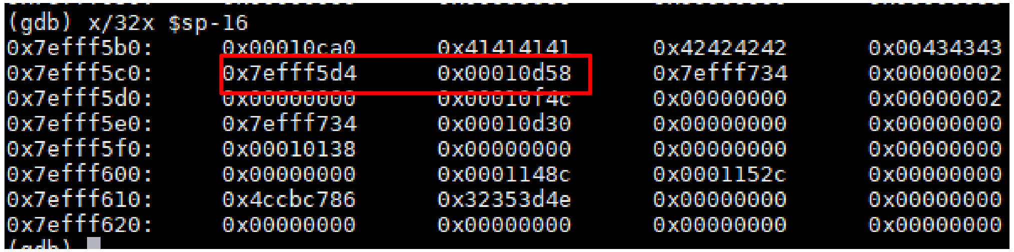
⬇

puts(buffer)

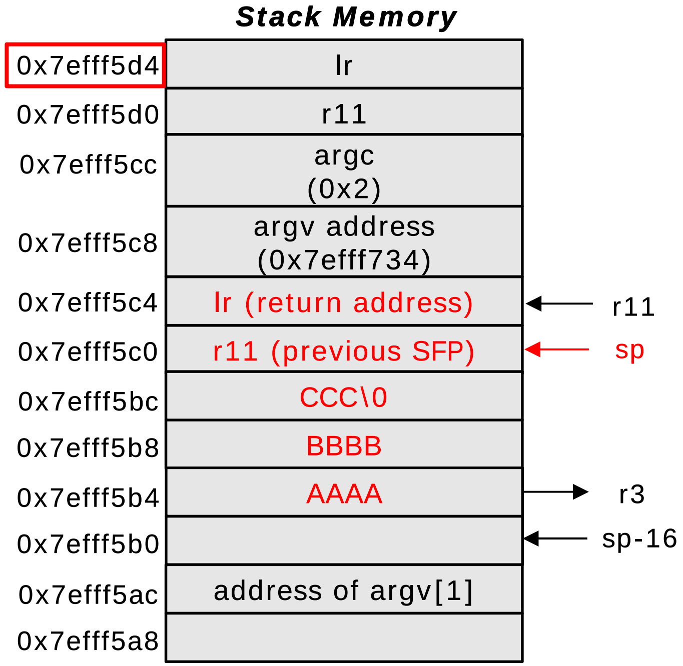
ARM Stack



```
=> 0x00010cfc <+0>:     push    {r11, lr}
   0x00010d00 <+4>:     add     r11, sp, #4
   0x00010d04 <+8>:     sub     sp, sp, #24
   0x00010d08 <+12>:    str     r0, [r11, #-24]
   0x00010d0c <+16>:    sub     r3, r11, #16
   0x00010d10 <+20>:    mov     r0, r3
   0x00010d14 <+24>:    ldr     r1, [r11, #-24]
   0x00010d18 <+28>:    bl      0x24680 <strcpy>
   0x00010d1c <+32>:    sub     r3, r11, #16
   0x00010d20 <+36>:    mov     r0, r3
   0x00010d24 <+40>:    bl      0x1782c <puts>
   0x00010d28 <+44>:    sub     sp, r11, #4
   0x00010d2c <+48>:    pop     {r11, pc}
End of assembler dump.
```



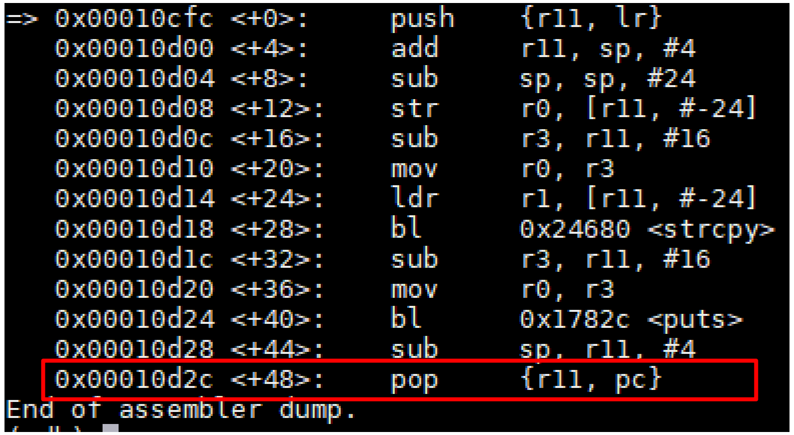
```
(gdb) x/32x $sp-16
0x7efff5b0:     0x00010ca0      0x41414141      0x42424242      0x00434343
0x7efff5c0:     0x7efff5d4      0x00010d58      0x7efff734      0x00000002
0x7efff5d0:     0x00000000      0x00010f4c      0x00000000      0x00000002
0x7efff5e0:     0x7efff734      0x00010d30      0x00000000      0x00000000
0x7efff5f0:     0x00010138      0x00000000      0x00000000      0x00000000
0x7efff600:     0x00000000      0x0001148c      0x0001152c      0x00000000
0x7efff610:     0x4ccbc786      0x32353d4e      0x00000000      0x00000000
0x7efff620:     0x00000000      0x00000000      0x00000000      0x00000000
```



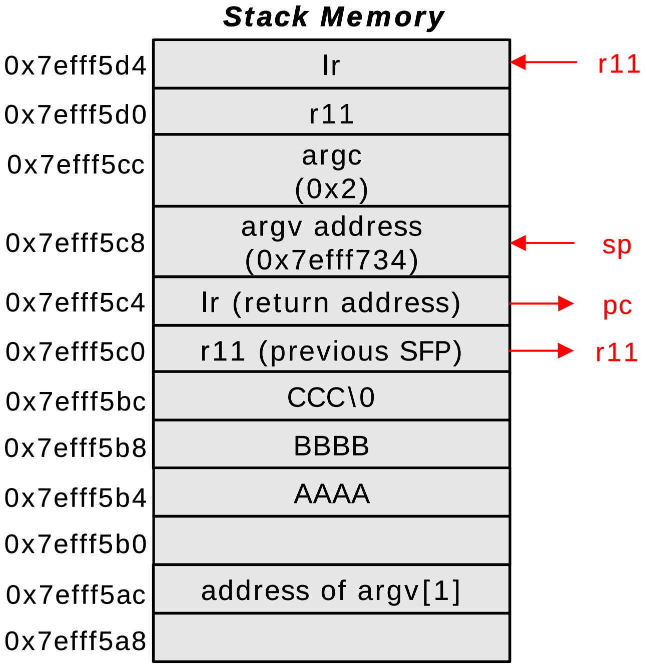
Stack Memory

0x7efff5d4	lr	
0x7efff5d0	r11	
0x7efff5cc	argc (0x2)	
0x7efff5c8	argv address (0x7efff734)	
0x7efff5c4	lr (return address)	← r11
0x7efff5c0	r11 (previous SFP)	← sp
0x7efff5bc	CCC\0	
0x7efff5b8	BBBB	
0x7efff5b4	AAAA	→ r3
0x7efff5b0		← sp-16
0x7efff5ac	address of argv[1]	
0x7efff5a8		

ARM Stack

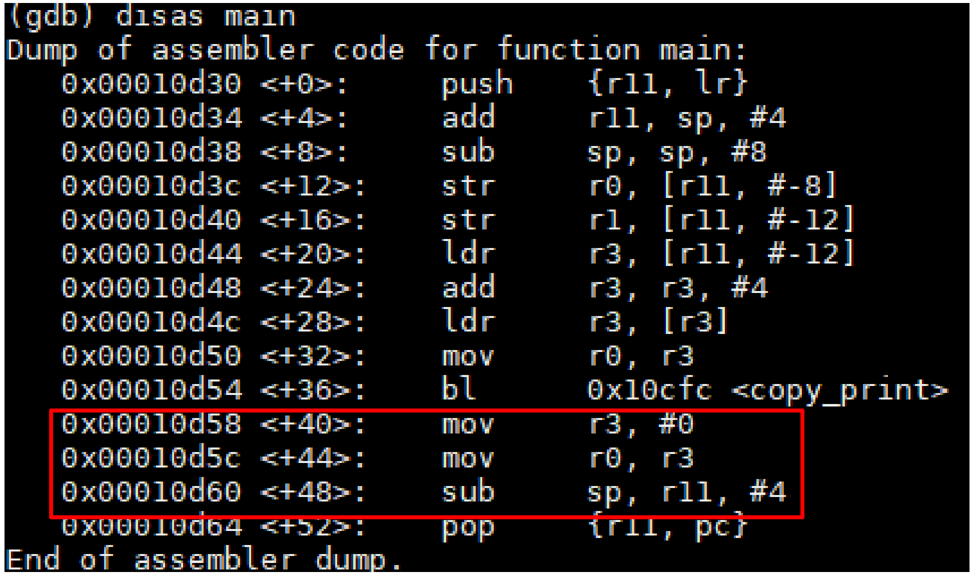


```
=> 0x00010cfc <+0>:     push    {r11, lr}
   0x00010d00 <+4>:     add     r11, sp, #4
   0x00010d04 <+8>:     sub     sp, sp, #24
   0x00010d08 <+12>:    str     r0, [r11, #-24]
   0x00010d0c <+16>:    sub     r3, r11, #16
   0x00010d10 <+20>:    mov     r0, r3
   0x00010d14 <+24>:    ldr     r1, [r11, #-24]
   0x00010d18 <+28>:    bl      0x24680 <strcpy>
   0x00010d1c <+32>:    sub     r3, r11, #16
   0x00010d20 <+36>:    mov     r0, r3
   0x00010d24 <+40>:    bl      0x1782c <puts>
   0x00010d28 <+44>:    sub     sp, r11, #4
   0x00010d2c <+48>:    pop     {r11, pc}
End of assembler dump.
```

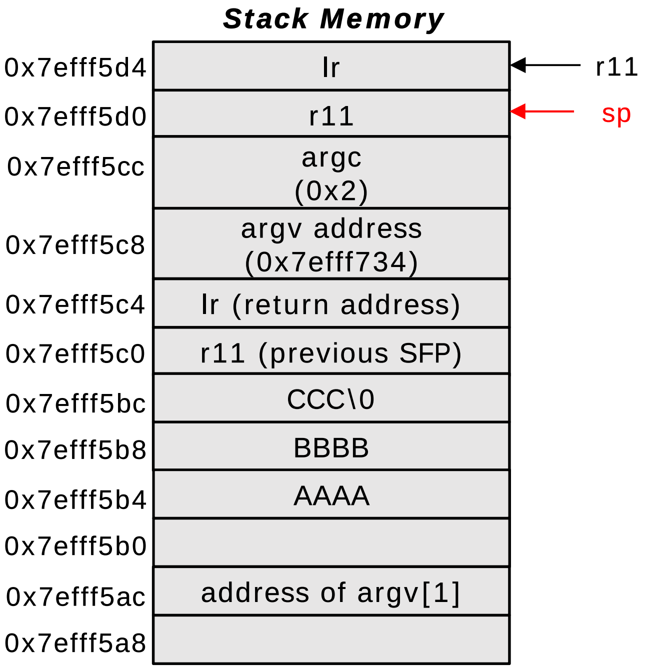


	Stack Memory	
0x7efff5d4	lr	r11
0x7efff5d0	r11	
0x7efff5cc	argc (0x2)	
0x7efff5c8	argv address (0x7efff734)	sp
0x7efff5c4	lr (return address)	pc
0x7efff5c0	r11 (previous SFP)	r11
0x7efff5bc	CCC\0	
0x7efff5b8	BBBB	
0x7efff5b4	AAAA	
0x7efff5b0		
0x7efff5ac	address of argv[1]	
0x7efff5a8		

ARM Stack



```
(gdb) disas main
Dump of assembler code for function main:
   0x00010d30 <+0>:     push    {r11, lr}
   0x00010d34 <+4>:     add     r11, sp, #4
   0x00010d38 <+8>:     sub     sp, sp, #8
   0x00010d3c <+12>:    str     r0, [r11, #-8]
   0x00010d40 <+16>:    str     r1, [r11, #-12]
   0x00010d44 <+20>:    ldr     r3, [r11, #-12]
   0x00010d48 <+24>:    add     r3, r3, #4
   0x00010d4c <+28>:    ldr     r3, [r3]
   0x00010d50 <+32>:    mov     r0, r3
   0x00010d54 <+36>:    bl      0x10cfc <copy_print>
   0x00010d58 <+40>:    mov     r3, #0
   0x00010d5c <+44>:    mov     r0, r3
   0x00010d60 <+48>:    sub     sp, r11, #4
   0x00010d64 <+52>:    pop     {r11, pc}
End of assembler dump.
```

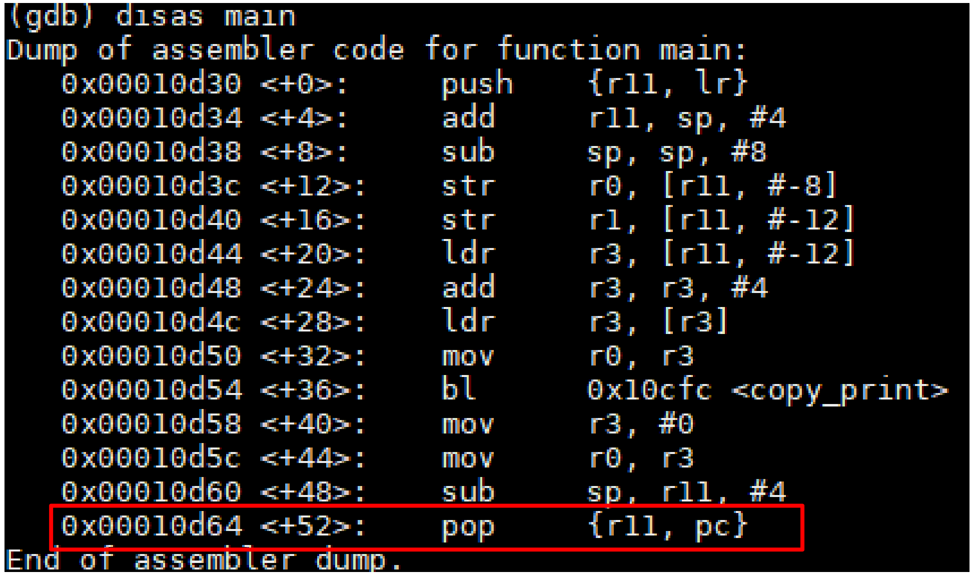


Stack Memory

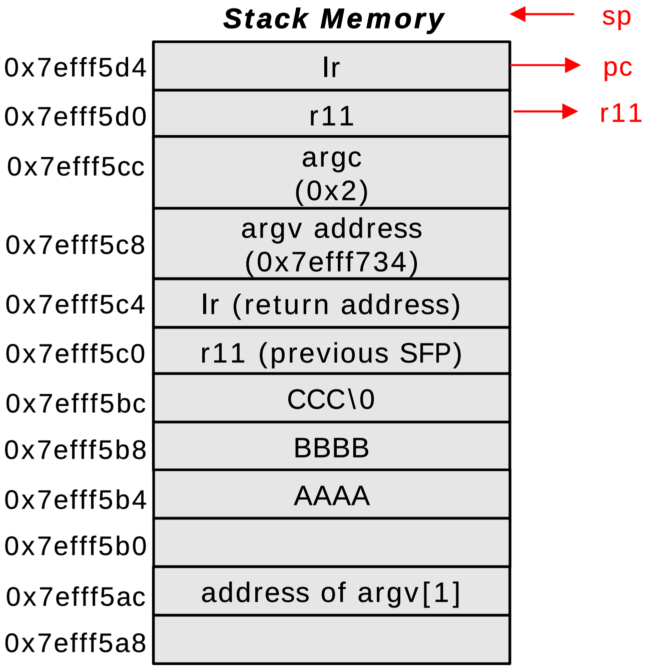
0x7efff5d4	lr	r11
0x7efff5d0	r11	sp
0x7efff5cc	argc (0x2)	
0x7efff5c8	argv address (0x7efff734)	
0x7efff5c4	lr (return address)	
0x7efff5c0	r11 (previous SFP)	
0x7efff5bc	CCC\0	
0x7efff5b8	BBBB	
0x7efff5b4	AAAA	
0x7efff5b0		
0x7efff5ac	address of argv[1]	
0x7efff5a8		

r3 = 0
r0 = 0

ARM Stack



```
(gdb) disas main
Dump of assembler code for function main:
   0x00010d30 <+0>:     push    {r11, lr}
   0x00010d34 <+4>:     add     r11, sp, #4
   0x00010d38 <+8>:     sub     sp, sp, #8
   0x00010d3c <+12>:    str     r0, [r11, #-8]
   0x00010d40 <+16>:    str     r1, [r11, #-12]
   0x00010d44 <+20>:    ldr     r3, [r11, #-12]
   0x00010d48 <+24>:    add     r3, r3, #4
   0x00010d4c <+28>:    ldr     r3, [r3]
   0x00010d50 <+32>:    mov     r0, r3
   0x00010d54 <+36>:    bl      0x10cfc <copy_print>
   0x00010d58 <+40>:    mov     r3, #0
   0x00010d5c <+44>:    mov     r0, r3
   0x00010d60 <+48>:    sub     sp, r11, #4
   0x00010d64 <+52>:    pop     {r11, pc}
End of assembler dump.
```



Stack Memory

	Stack Memory
0x7efff5d4	lr
0x7efff5d0	r11
0x7efff5cc	argc (0x2)
0x7efff5c8	argv address (0x7efff734)
0x7efff5c4	lr (return address)
0x7efff5c0	r11 (previous SFP)
0x7efff5bc	CCC\0
0x7efff5b8	BBBB
0x7efff5b4	AAAA
0x7efff5b0	
0x7efff5ac	address of argv[1]
0x7efff5a8	

r3 = 0
r0 = 0

Q & A

http://mesl.khu.ac.kr