

r0	15 2 15
r1	15 2 15 2 3
r2	???
r3	15 1

r13(sp) ~~0x120~~ ~~0x11C~~ ~~0x114~~ ~~0x110~~ ~~0x10F~~ ~~0x110~~
 r14(lr) ~~0x100010~~ ~~loop+CS~~ mod+32 0x114
 r15(pc) ~~isprime~~ ~~mod~~ ~~divmod~~ loop+28

```

loop:
→   ldr r1, [sp, #4]
→   add r1, #1
→   str r1, [sp, #4]

```

```
→ ldr r0, [sp] ←
→ cmp r1, r0
→ bge prime ← could have moved to r0, r1
→ bl mod
→ cmp r0, #0
→ beq notprime
```

→ b loop

```
prime:
    mov r0, #1
    b end
notprime:
    mov r0, #0
end:
    add sp, #8
    pop {pc}
```

Memory layout diagram showing stack frames for 'div mod' and 'main' functions. The 'div mod' frame contains local variables 'z' (0x104), '15' (0x108), and '???' (0x10c). The 'main' frame contains '15' (0x114), '2' (0x118), and '0x100010' (0x11c).

```
int mod(int num, int den) {
    return num % den;
}
```

```
$ gcc -o isprime isprime.o main.c
```

```
mod:
push    {fp, lr}
add     fp, sp, #4
sub     sp, sp, #8
str     r0, [fp, #-8]
str     r1, [fp, #-12]
ldr     r3, [fp, #-8]
mov     r0, r3
ldr     r1, [fp, #-12]
bl      10690 <__aeabi_idivmod>
mov     r3, r1
mov     sp, r3
sub     fp, fp, #4
pop     {fp, pc}
```

$$f_p = 511$$

Memory

The diagram illustrates a stack frame structure. On the left, a vertical dashed box represents the stack. Inside this box, from top to bottom, are the following labels: *locals* (in red), *fp* (in red), *ret addr* (in red), *2*, *15*, *fp*, and *return addr*. To the right of the stack, a horizontal rectangle represents the frame pointer. An arrow labeled *fp* points from this rectangle to the *fp* entry in the stack. Another arrow points from the *return addr* entry in the stack to the right, indicating the return path.

Memory

Memory