ARM to C calling convention, registers to save

It's been a while since I last coded arm assembler and I'm a little rusty on the details. If I call a C function from arm, I only have to worry about saving r0-r3 and lr, right? If the C function uses any other registers, is it responsible for saving those on the stack and restoring them? In other words, the compiler would generate code to do this for C functions. For example if I use r10 in an assembler function, I don't have to push its value on the stack, or to memory, and pop/restore it after a C call, do I?

This is for arm-eabi-gcc 4.3.0.

I realise I could read the whole EABI, but then shortcutting RTFM is what SO is for, right? :-)

c | assembly | arm | function-calls

edited Nov 16 '08 at 1:38
SoapBox
8,643 10 30

asked Nov 4 '08 at 10:16
richq
11.3k 2 25 48
93% accept rate

feedback

2 Answers

It depends on the ABI for the platform you are compiling for. On Linux, there are two ARM ABIs; the old one and the new one. AFAIK, the new one (EABI) is in fact ARM's AAPCS. I have a bookmark pointing to http://www.arm.com/pdfs/bsabi.zip as a place to get the ARM ABI specification, but that link seems to be stale.

Briefly:

- r0-r3 are the argument and scratch registers; r0-r1 are also the result registers
- r4-r8 are callee-save registers
- r9 might be a callee-save register or not (on some variants of AAPCS it is a special register)
- r10-r11 are callee-save registers
- r12-r15 are special registers

A callee-save register must be saved by the callee (in opposition to a caller-save register, where the caller saves the register); so, *if* this is the ABI you are using, you do not have to save r10 before calling another function (the other function is responsible for saving it).

Edit: Which compiler you are using makes no difference; gcc in particular can be configured for several different ABIs, and it can even be changed on the command line. Looking at the prologue/epilogue code it generates is not that useful, since it is tailored for each function *and* the compiler can use other ways of saving a register (for instance, saving it in the middle of a function).

edited Apr 29 '09 at 6:15
Matt J
6,236 15 32

answered Nov 4 '08 at 10:46
CesarB
9,325 1 11 31

Thanks, this seems to ring some bells. I think the first "r0-r4" in your list is a typo, right? +1 (and probably best answer unless there's a radical turn around) – richq Nov 4 '08 at 10:52

Yes, it was a typo (and not the only one, but I fixed the other ones before hitting submit the first time - or so I hope). – CesarB Nov 4 '08 at 11:02

"You can download the whole ABI specification and its supporting documents and example code as a ZIP archive from this page." Zip Archive: infocenter.arm.com/help/topic/com.arm.doc.ihi0036b/bsabi.zip – noloader Jun 23 at 3:58

feedback

To add up missing info on NEON registers:

From the AAPCS, §5.1.2.1: * s16–s31 (d8–d15, q4–q7) must be preserved * s0–s15 (d0–d7, q0–q3) and d16–d31 (q8–q15)

Original post: arm-to-c-calling-convention-neon-registers-to-save

answered **Mar 29 at 4:44**

Pavel

125 5

feedback

Not the answer you're looking for? Browse other questions tagged c | assembly | arm | function-calls **or ask your own question.**

question feed