

Re: Recursive Call

Source: <http://coding.derkeiler.com/Archive/Cobol/comp.lang.cobol/2005-11/msg00215.html>

- *From:* "charles hottel" <jghottel@xxxxxxxxxx>
 - *Date:* Fri, 11 Nov 2005 19:08:20 -0500
-

On IBM mainframes there was/is a standard linkage convention to be followed that uses a standard 18 fullword save area and certain standard register usage conventions. Register 15 contains the "branch to" address and Register 14 contains the "Return address". So to "call" another program you would do something like:

```
L R15,=V(MYSUBRTN)
BALR R14,R15
```

Another part of this linkage convention is that the calling program provide within its storage an 18 fullword save area that the subroutine can use to save the callers registers. The standard is that Register 13 will point to this save area at entry to the subroutine. The save areas themselves contain standard offsets where the register contents are saved plus there are forward/backward pointers that chain the save areas together into a linked list stack. Usually the first thing a subroutine does is save the caller's registers and set register 13 to point to it's save area. When an abend occurs a save area trace is produced in the dump and shows the call chain. In assembly language there are SAVE and RETURN macros and the RETURN macros has an option that can be used to set a bit in the save area to indicate to the dump program that a save area is no longer active. When a subroutine ends it restores the callers registers and branches to the return address.

Of course all of the above was the original convention. As IBM hardware and addressing modes have evolved there are more/different linkage instructions and branching instructions. Some of these use a linkage stack area and there are instructions that are used to access and manipulate it. Some work with 31-bit addressing and others work with 64-bit addressing. With 64 bit addressing the save area layout had to be modified to accommodate the larger addresses and larger/wider registers.

This is a pretty standard process for all the machines that I have worked on and each has it standard entry/exit magic instruction sequences to accomplish this. Also not mentioned, there is usually a standard way to pass parameters to the subroutine. I believe that on the IBM mainframe COBOL follows the convention when subprograms are CALLED.

"Michael Wojcik" <mwojcik@xxxxxxxxxxxx> wrote in message

Re: Recursive Call

news:dl2hl00kku@xxxxxxxxxxxxxxxxxxxxxx

>
> In article <dktc9l\$169\$1@xxxxxxxxxxxxxxxxxxxx>, docdwarf@xxxxxxxx () writes:
>>
>> The fellow with whom I was conversing mentioned use of the stack-trace in
>> debugging; I said that this was not, in my experience, a common method
>> applied to IBM-architecture mainframes. I based this on my experience as
>> a mere apps-jockey; when something blows up I look for the PSW, find the
>> address of the previous statement and proceed from there.
>>
>> This ignores – in the radical sense of 'in (not) gnosis' – what folks at
>> the systems level do. Is there something akin to stack tracing done on
>> Big Blue Big Iron running 'classic' (COBOL/CICS/DB2) systems?
>
> I was curious to see answers to this, but it doesn't appear that you've
> had any takers.
>
> I've not done much debugging of mainframe systems-level code, but the
> folks I've seen do it generally start with a dump, and do sometimes
> manually backtrack through subroutine calls. These programs were
> mostly written in assembly, and they used some variant of BAL (branch
> and link) for subroutine calls, and maintained their own scratch
> areas for return addresses, parameters, and so forth. So there was
> no contiguous "stack" as there is on x86 and the like, but they did
> sometimes trace the flow of execution back using the dump.
>
> --
> Michael Wojcik michael.wojcik@xxxxxxxxxxxxxxxx
>
> Auden often writes like Disney. Like Disney, he knows the shape of
> beasts --
> (& incidently he, too, might have a company of artists producing his
> lines) --
> unlike Lawrence, he does not know what shapes or motivates these beasts.
> -- Dylan Thomas

• *References:*

- ◆ **Recursive Call**
 ◇ From: Paolo
- ◆ **Re: Recursive Call**
 ◇ From: Joe Zitzelberger
- ◆ **Re: Recursive Call**
 ◇ From: Arnold Trembley
- ◆ **Re: Recursive Call**
 ◇ From: Michael Wojcik
- ◆ **Re: Recursive Call**

Re: Recursive Call

◇ *From:*

◆ ***Re: Recursive Call***

◇ *From:* Michael Wojcik

- Prev by Date: ***Re: IBM, ANIS/ISO, 68 / 74 file types (was: Cobol language dictionary page -- assignment complete?)***
- Next by Date: ***Re: Cobol work?***
- Previous by thread: ***Re: Recursive Call***
- Next by thread: ***Re: Recursive Call***
- Index(es):
 - ◆ ***Date***
 - ◆ ***Thread***