Uso degli strumenti cross compiler ed emulatore per ARM sotto LINUX x86

Il documento fa vedere come configurare una macchina con LINUX (ubuntu 18) e processore Intel/AMD (x86) in modo da riuscire a compilare codice C in assembler ARM v7, a compilare programmi in assembler ARM v7 e ad eseguirli sia come programmi che attraverso debugger GDB.

Primo passo: installazione dei tool necessari

Occorre installare 3 pacchetti, mediante i comandi

- apt install gcc-arm-linux-gnueabihf
- apt install qemu
- apt install gdb-multiarch

Secondo passo: utilizzo dei tool

Compilazione codice C

Per vedere come il compilatore gnu compila codice c (hello.c) generando codice ARM:

- arm-linux-gnueabihf-gcc hello.c -o hello_arm_static -static

Il flag static è importante, altrimenti nell'esecuzione del programma (vedi sotto) ci troviamo ad avere un errore di caricamento delle librerie dinamiche, che non stanno nel posto dove dovrebbero essere (lì ci sono quelle per Intel …)

Esecuzione codice ARM

Per eseguire il programma risultante

- qemu-arm ./hello

Compilazione codice assembler

Per compilare un programma assembler si usa la stessa procedura

- arm-linux-gnueabihf-gcc hello.s -o hello_arm_static -static

anche se potremmo al posto del wrapper gcc utilizzare il wrapper assembler che corrisponde al comando:

- arm-linux-gnueabihf-as ello.s -o hello_arm_static -static

Esecuzione codice mediante debugger sotto qemu

Occorre aprire due finestre shell.
Nella **prima finestra**, lanciamo l'esecuzione del programma in qemu con il debugger interno, redirigendo il controllo di quel debugger su una porta:

- qemu-arm -g 12345 ./hello

ATTENZIONE: perchè si possa adoperare correttamente il debugger il programma deve essere stato generato utilizzando l'opzione **-ggdb3** ovvero deve essere compilato con un comando

- arm-linux-gnueabihf-gcc hello.s -o hello_arm_static -static -ggdb3

Nella **seconda finestra** occorre lanciare l'istanza del debugger che si collega al debugger aperto sulla porta 12345 dal qemu con il comando:

- gdb-multiarch -q --nh -ex 'set architecture arm' -ex 'file *hello'* -ex ' target remote localhost:*12345'* ;

Attenzione ad usare il nome giusto dell'eseguibile, in questo caso hello, e della porta utilizzata per lanciare il debugger, in questo caso 12345.

A questo punto nella seconda finestra avete il prompt del debbuger. Potete per esempio mettere un breakpoint sul main:

- b main

continuare l'esecuzione (gdb sotto qemu non supporta il run)

- c

e quindi cominciare a utilizzare i comandi usuali del gbd, per esempio "n" per eseguire la prossima istruzione o "info registers" per vedere il contenuto dei registri. Può risultare particolarmente utile la vista TUI (vedi manuali) del debugger, che permette di vedere registri, codice e prompt dei comandi GDB in modalità testo sul terminale.