

Debug e verifica delle funzionalità

- **Compilazione**

È necessario eseguire `./compile.sh`

```
calcolatori@calcolatori-VirtualBox: ~/riscv-asm-quicksort
calcolatori@calcolatori-VirtualBox:~/riscv-asm-quicksort$ ./compile.sh
calcolatori@calcolatori-VirtualBox:~/riscv-asm-quicksort$
```

- **Esecuzione**

È obbligatorio eseguirlo con il debugger attivo (altrimenti non sarà possibile vedere il risultato) scrivendo `qemu-riscv64 -g 2323 ./quicksort`

```
calcolatori@calcolatori-VirtualBox: ~/riscv-asm-quicksort (ssh)
calcolatori@calcolatori-VirtualBox:~/riscv-asm-quicksort$ qemu-riscv64 -g 2323 ./quicksort
```

- **Debug**

Per debuggare basterà avviare `./debug.sh`;
Basterà poi scrivere `display/7b &nomevar`; per visualizzare `&nomevar` a ogni step.
Ad esempio: `display/7b &testarray`

```
1: x/7xb &testarray
0x11285:      0x01      0x03      0x05      0x06      0x07      0x08      0x0a
(gdb)
[Inferior 1 (Remote target) exited with code 06]
(gdb)
```

```
calcolatori@calcolatori-VirtualBox: ~/riscv-asm-quicksort (ssh)
calcolatori@calcolatori-VirtualBox:~/riscv-asm-quicksort$ ./debug.sh
GNU gdb (GDB) 8.0.50.20170724-git
Copyright (C) 2017 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law. Type "show copying"
and "show warranty" for details.
This GDB was configured as "--host=x86_64-pc-linux-gnu --target=riscv64-unknown-elf".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
<http://www.gnu.org/software/gdb/documentation/>.
For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from quicksort...done.
Remote debugging using :2323
_start () at main.S:18
18      la a1, testarray
(gdb) display/7b &testarray
1: x/7xb &testarray
0x11285:      0x05      0x01      0x03      0x07      0x08      0x0a      0x06
(gdb)
```

```
calcolatori@calcolatori-VirtualBox: ~/riscv-asm-quicksort (ssh)
40      ld s11, 16(sp)
1: x/7xb &testarray
0x11285:      0x01      0x03      0x05      0x06      0x07      0x08      0x0a
(gdb)
41      ld s9, 24(sp)
1: x/7xb &testarray
0x11285:      0x01      0x03      0x05      0x06      0x07      0x08      0x0a
(gdb)
42      addi sp, sp, 32
1: x/7xb &testarray
0x11285:      0x01      0x03      0x05      0x06      0x07      0x08      0x0a
(gdb)
quicksort_exit () at quicksort.S:44
44      ret
1: x/7xb &testarray
0x11285:      0x01      0x03      0x05      0x06      0x07      0x08      0x0a
(gdb)
_start () at main.S:26
26      ecall
1: x/7xb &testarray
0x11285:      0x01      0x03      0x05      0x06      0x07      0x08      0x0a
(gdb)
[Inferior 1 (Remote target) exited with code 06]
(gdb)
```

Take-away principale

L'ISA RISC-V è semplice. Questo la rende molto veloce e permette ai compilatori di ottimizzare particolarmente il codice. Se però si vuole scrivere assembly direttamente, diventa necessario porre particolare attenzione a ogni istruzione che si scrive, sia per scrivere il proprio codice in modo che "sfrutti" la pipeline, sia per evitare che vi siano problemi logici.

Leggere e "tradurre" codice di riferimento scritto in un linguaggio di programmazione di alto livello semplifica decisamente questo compito.