uARM: a simple ARM virtual machine

Marco Melletti

2 dicembre 2014

Genealogia delle VM

- Chip: PDP-11, device ancora utilizzati (1983)
- MPS: MIPS, memoria virtuale sempre attiva (2004)
- uMPS: MIPS, memoria virtuale opzionale (2007)
- uMPS2: MIPS, supporto multicore, interfaccia grafica ristrutturata (2011)
- uARM

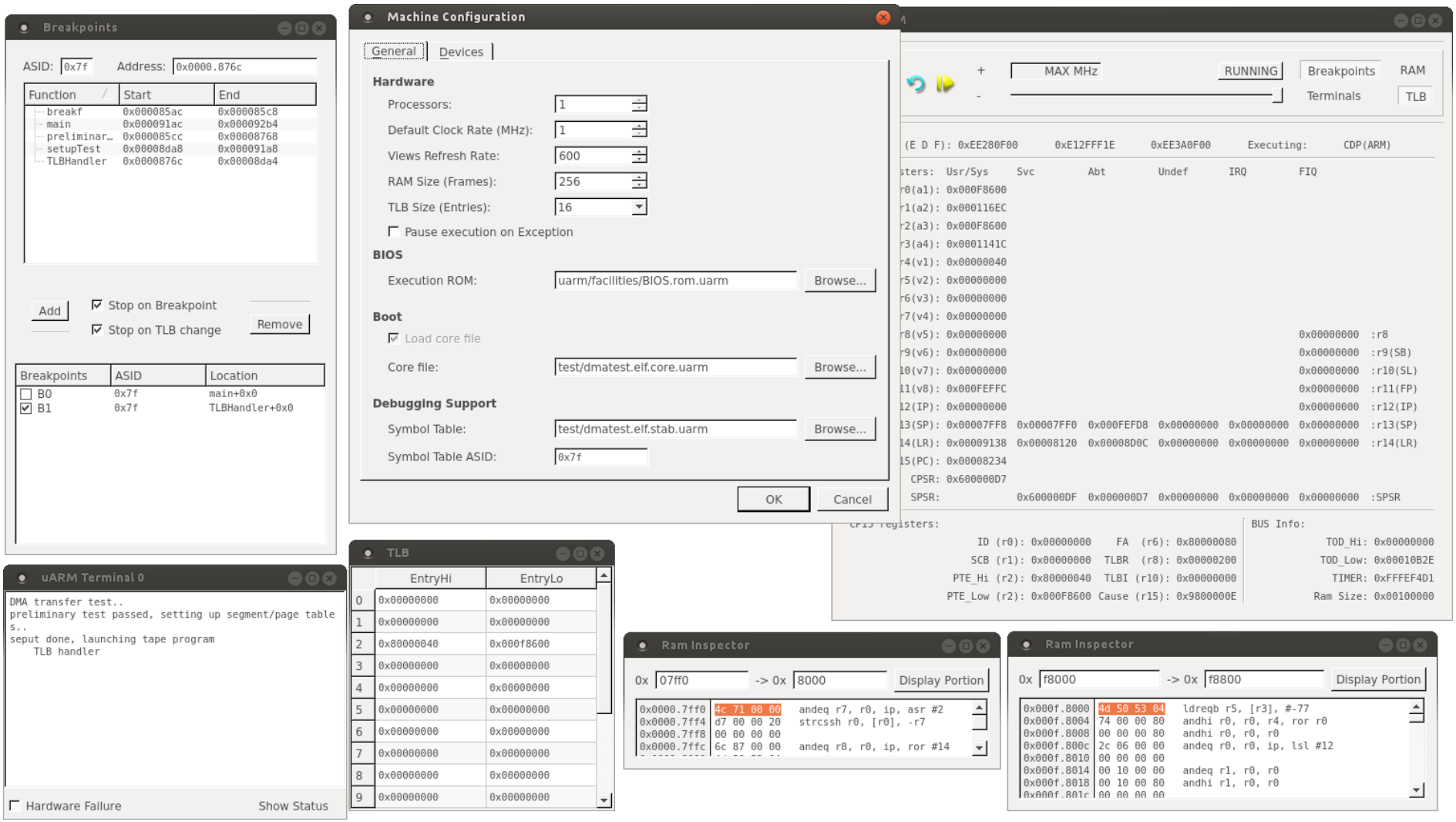
ARM: Advanced RISC Machine

- Architettura RISC: Reduced Instruction Set Computer
- Attuale e largamente utilizzata:
  - Embedded Systems
  - Smartphones
  - Nintendo DS
  - Raspberry Pi
  - Game Boy Advance
  - iPod
  - ...

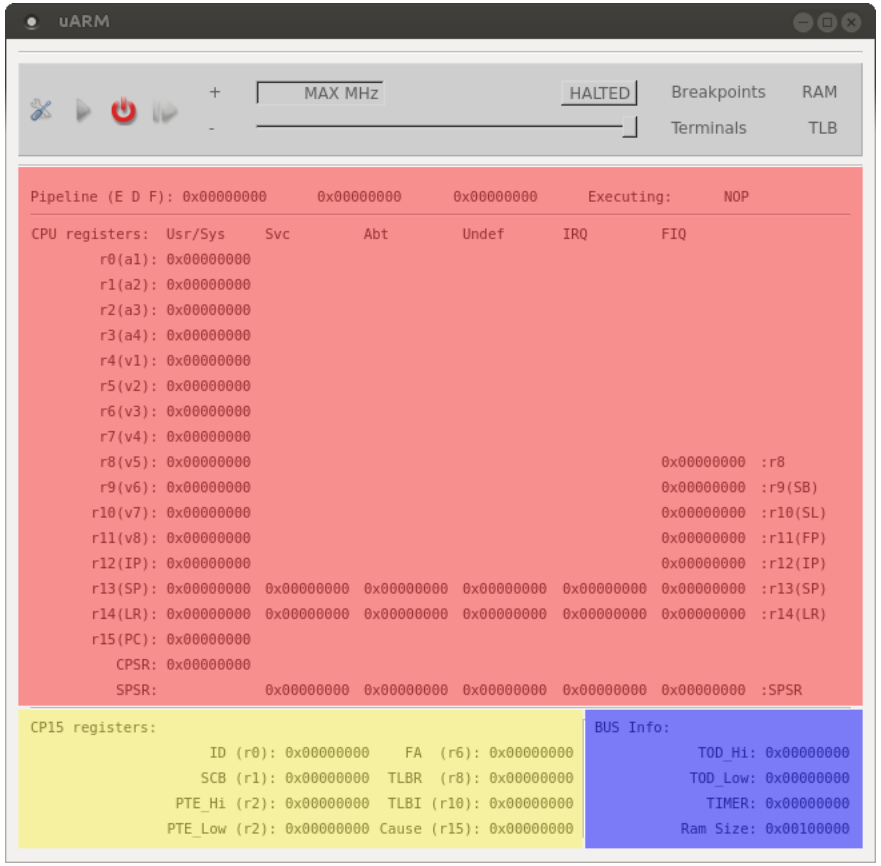
uARM

- Processore ARM7TDMI
- Memoria Little-Endian a dimensione variabile
- MMU con TLB a dimensione variabile
- 8 Device per tipo:
  - terminali
  - stampanti
  - schede di rete (VDE)
  - dischi fissi
  - nastri (dischi ottici)

uARM: GUI


Breakpoints
ASID: 0x7f
Address: 0x0000.876c
Function
Start
End
breakf 0x000085ac 0x000085c8
main 0x000091ac 0x000092b4
preliminar… 0x000085cc 0x00008768
setupTest 0x00008da8 0x000091a8
TLBHandler 0x0000876c 0x00008da4
Add
Stop on Breakpoint
Stop on TLB change
Remove
Breakpoints
ASID
Location
B0 0x7f main+0x0
B1 0x7f TLBHandler+0x0
Machine Configuration
General
Devices
Hardware
Processors: 1
Default Clock Rate (MHz): 1
Views Refresh Rate: 600
RAM Size (Frames): 256
TLB Size (Entries): 16
Pause execution on Exception
BIOS
Execution ROM: uarm/facilities/BIOS.rom.uarm
Browse...
Boot
Load core file
Core file: test/dmatest.elf.core.uarm
Browse...
Debugging Support
Symbol Table: test/dmatest.elf.stab.uarm
Browse...
Symbol Table ASID: 0x7f
OK
Cancel
MAX MHz
RUNNING
Breakpoints
RAM
Terminals
TLB
(E D F): 0xEE280F00 0xE12FFF1E 0xEE3A0F00 Executing: CDP(ARM)
Usr/Sys Svc Abt Undef IRQ FIQ
r0(a1): 0x000F8600
r1(a2): 0x000116EC
r2(a3): 0x000F8600
r3(a4): 0x0001141C
r4(v1): 0x00000040
r5(v2): 0x00000000
r6(v3): 0x00000000
r7(v4): 0x00000000
r8(v5): 0x00000000 :r8
r9(v6): 0x00000000 :r9(SB)
r10(v7): 0x00000000 :r10(SL)
r11(v8): 0x000FEFFC :r11(FP)
r12(IP): 0x00000000 :r12(IP)
r13(SP): 0x00007FF8 0x00007FF0 0x000FEFD8 0x00000000 0x00000000 0x00000000 :r13(SP)
r14(LR): 0x00009138 0x00008120 0x00008D0C 0x00000000 0x00000000 0x00000000 :r14(LR)
r15(PC): 0x00008234
CPSR: 0x600000D7
SPSR: 0x600000DF 0x000000D7 0x00000000 0x00000000 0x00000000 :SPSR
CP15 registers:
ID (r0): 0x00000000 FA (r6): 0x80000080
SCB (r1): 0x00000000 TLBR (r8): 0x00000200
PTE_Hi (r2): 0x80000040 TLBI (r10): 0x00000000
PTE_Low (r2): 0x000F8600 Cause (r15): 0x9800000E
BUS Info:
TOD_Hi: 0x00000000
TOD_Low: 0x00010B2E
TIMER: 0xFFFEF4D1
Ram Size: 0x00100000
uARM Terminal 0
DMA transfer test..
preliminary test passed, setting up segment/page tables..
seput done, launching tape program
TLB handler
Hardware Failure
Show Status
TLB
EntryHi
EntryLo
0 0x00000000 0x00000000
1 0x00000000 0x00000000
2 0x80000040 0x000f8600
3 0x00000000 0x00000000
4 0x00000000 0x00000000
5 0x00000000 0x00000000
6 0x00000000 0x00000000
7 0x00000000 0x00000000
8 0x00000000 0x00000000
9 0x00000000 0x00000000
Ram Inspector
0x 07ff0 -> 0x 8000 Display Portion
0x0000.7ff0 4c 71 00 00 andeq r7, r0, ip, asr #2
0x0000.7ff4 d7 00 00 20 strcssh r0, [r0], -r7
0x0000.7ff8 00 00 00 00
0x0000.7ffc 6c 87 00 00 andeq r8, r0, ip, ror #14
Ram Inspector
0x f8000 -> 0x f8800 Display Portion
0x000f.8000 4d 50 53 04 ldreqb r5, [r3], #-77
0x000f.8004 74 00 00 80 andhi r0, r0, r4, ror r0
0x000f.8008 00 00 00 80 andhi r0, r0, r0
0x000f.800c 2c 06 00 00 andeq r0, r0, ip, lsl #12
0x000f.8010 00 00 00 00
0x000f.8014 00 10 00 00 andeq r1, r0, r0
0x000f.8018 00 10 00 80 andhi r1, r0, r0

uARM: GUI



- Barra di controllo
- Stato Processore
- Stato Coprocessore
- Informazioni di sistema

uARM: Barra di controllo

- Configurazioni
- Controllo esecuzione
- Terminali
- Funzioni di debug

uARM: un esempio

Sorgente: `foo/esempio.c`

```
#include /usr/include/uarm/libuarm.h

int main(){
    tprint("Hello World\n\0");
    HALT();
    tprint("");
    return 0;
}
```

Compilazione: un esempio

Sorgente: `foo/esempio.c`

Compiliamo il file:

```
$> arm-none-eabi-gcc -mcpu=arm7tdmi -c -o foo/esempio.o \
foo/esempio.c
```

Linkiamo il file oggetto con la libreria di uARM e il file di inzializzazione:

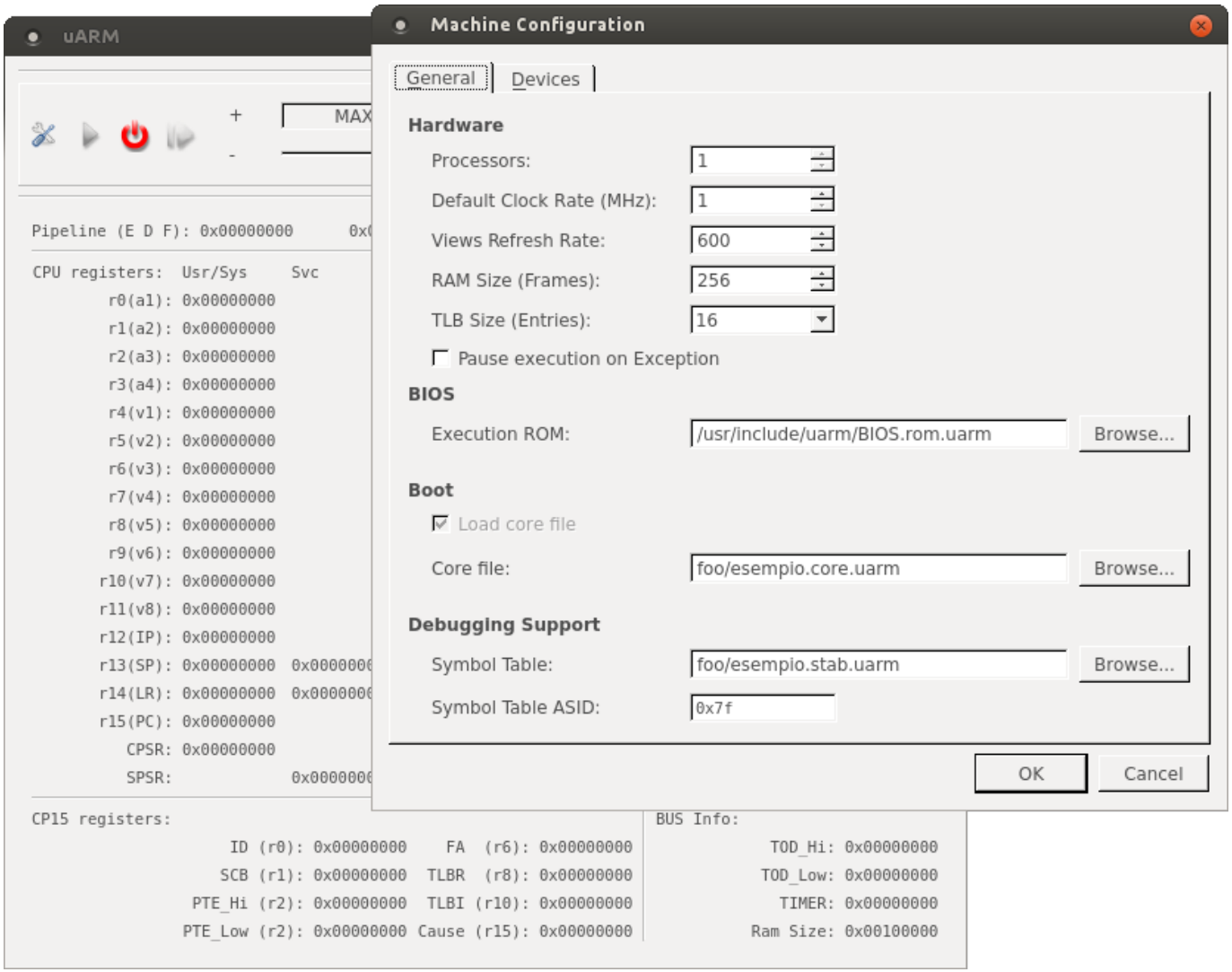
```
$> arm-none-eabi-ld -T \
/usr/include/uarm/ldscripts/elf32ltsarm.h.uarmcore.x \
-o esempio.elf /usr/include/uarm/crtso.o \
/usr/include/uarm/libuarm.o esempio.o
```

Convertiamo l'eseguibile nel formato di uARM:

```
$> elf2uarm -k foo/esempio.elf
```

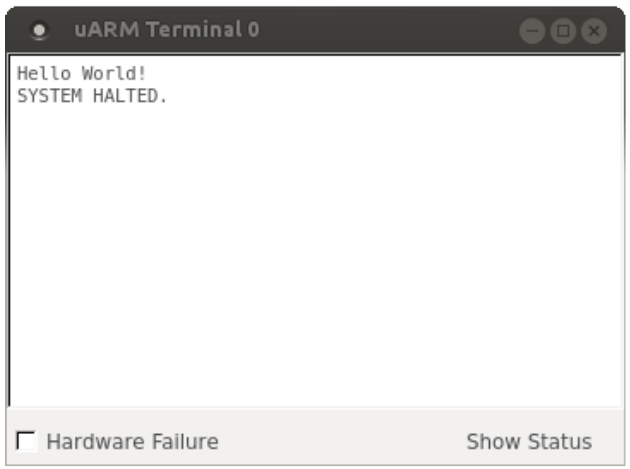
Esecuzione: un esempio

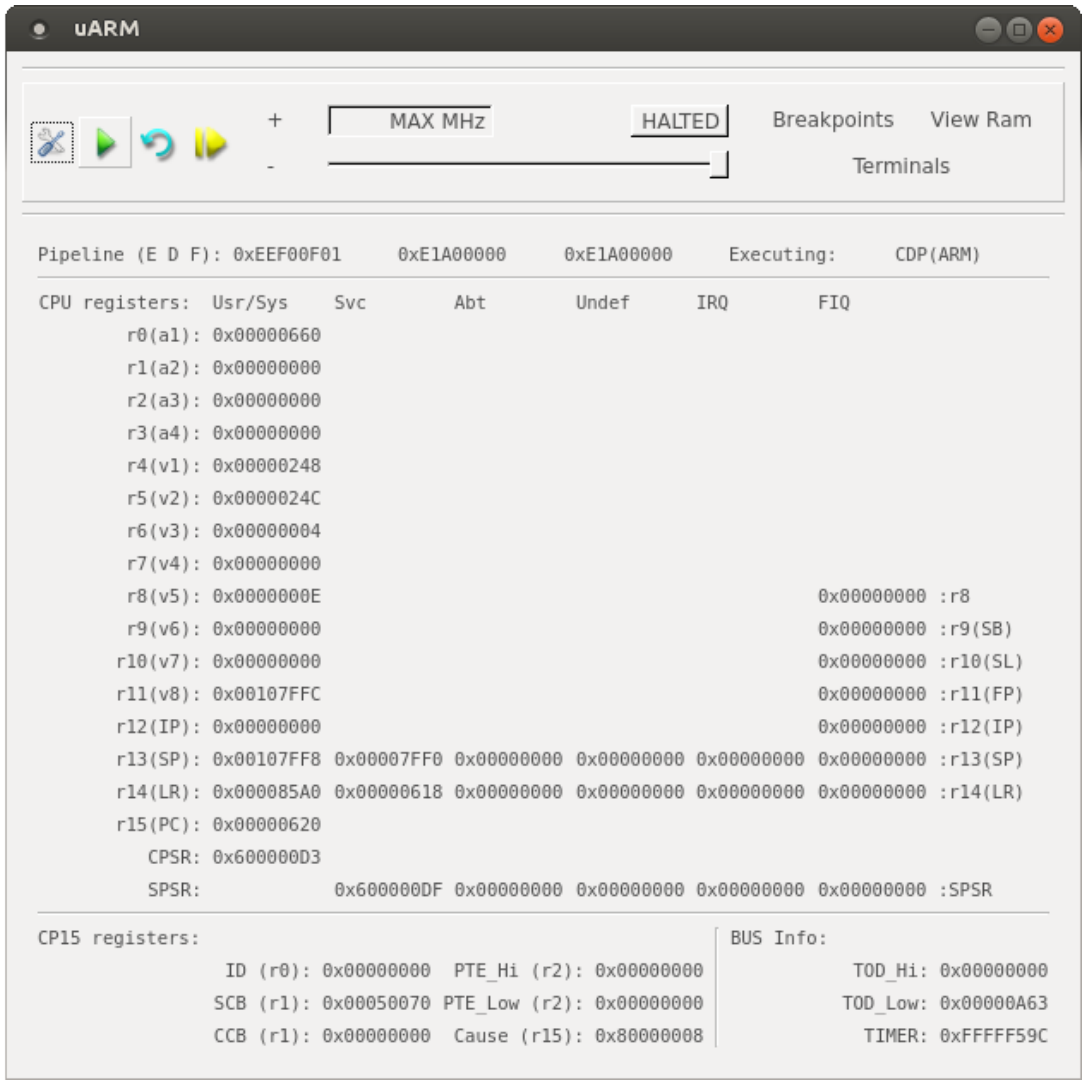
Impostiamo il core file generato per l'esecuzione:



Esecuzione: un esempio

Avviamo la macchina e lanciamo l'esecuzione, il terminale 0 mostrerà l'output:





uARM: un altro esempio

Proviamo davvero la macchina...

uARM

- Home page: `http://mellotanica.github.io/uARM/`
  - Repository ufficiale
  - Pacchetti per VirtLab
  - Questa introduzione
  - Specifiche della macchina

- Contattatemi per domande/problemi/richieste
  - marco.melletti@studio.unibo.it
  - mellotanica@hotmail.it

Grazie dell'attenzione

Domande?