

1 Cross-compiler

- 1.1 Installare crosstool-ng (<https://crosstool-ng.github.io/>)
- 1.2 Eseguire "ct-ng menuconfig"
- 1.3 Eseguire "ct-ng build"
- 1.4 In alternativa e' possibile scaricare un cross-compiler gia' pronto. Ad esempio per BBB si puo' usare un compilatore Linaro:
<https://releases.linaro.org/>
(/opt/x-tools/gcc-linaro-7.4.1-2019.02-x86_64_arm-linux-gnueabi/bin)
- 1.5 [Facoltativo] Aggiungere alla variabile d'ambiente PATH la cartella di installazione del cross-compiler

2 Il processo di compilazione

2.1 Scrivere noop.c

```
+-----+
| void _reset(void)
| {
|     for(;;);
| }
+-----+
```

2.1 Compilare noop.c

2.1.1 Compilazione:

```
+-----+
| $ ${PREFIX}gcc -Wall -Wextra -O0 -ffreestanding -mcpu=cortex-a8 \
| -march=armv7-a -mfloat-abi=hard -mfpu=vfpv3 -marm -c noop.c
+-----+
```

2.1.2 Collegamento:

```
+-----+
| $ ${PREFIX}ld -nostdlib -e _reset -o sert.elf noop.o
+-----+
```

2.2 Estrazione del codice macchina:

```
+-----+
| $ ${PREFIX}objcopy -S -O binary sert.elf sert.bin
+-----+
```

2.3 Controllo del file ELF e del file binario

```
+-----+
| $ ${PREFIX}objdump -d sert.elf
|
| sert.elf:          file format elf32-littlearm
|
| Disassembly of section .text:
|
| 00010074 <_reset>:
|   10074:          eafffffe          b          10074 <_reset>
+-----+
```

3 Caricamento dell'eseguibile ed esecuzione

3.1 Lanciare minicom

3.2 Caricare in RAM il programma con "loadb 0x80000000"

3.2.1 Avviare il trasferimento di sert.bin con <Ctrl-A>+S ("kermit")

3.2.2 La linea comando per il trasferimento via kermit e'

```
$ kermit -i -l %l -b %b -B -s
```

ove %l e' il device file della seriale e %b e' la velocita'

3.3 Eseguire con "go 0x80000000"

4 Makefile

4.1 Creare il file Makefile

```
+-----+
| CROSSPATH?=/opt/x-tools/gcc-linaro-7.4.1-2019.02-x86_64_arm\
| -linux-gnueabi/bin-linux-gnueabi/bin
| CROSSPFX=$(CROSSPATH)/arm-linux-gnueabi-
| CC=$(CROSSPFX)gcc
| AS=$(CROSSPFX)as
| LD=$(CROSSPFX)ld
| NM=$(CROSSPFX)nm
| OBJCOPY=$(CROSSPFX)objcopy
| OBJDUMP=$(CROSSPFX)objdump
| # Alternatively: -O2 || -O3 -fno-tree-vectorize
| CFLAGS=-Wall -Wextra -O2 -ffreestanding -ffast-math
| ARCHFLAGS=-mcpu=cortex-a8 -march=armv7-a -mfloat-abi=hard \
| -mfpu=vfpv3
| CCARCHFLAGS=$(ARCHFLAGS) -marm
| CFILES:=$(shell ls *.c 2>/dev/null)
| SFILES:=$(shell ls *.S 2>/dev/null)
| HFILES:=$(shell ls *.h 2>/dev/null)
| AOBJS:=$(SFILES:%.S=%.o)
| COBJS:=$(CFILES:%.c=%.o)
| TARGET=sert
+-----+
```

```

all: $(TARGET).bin $(TARGET).lst $(TARGET).sym

%.bin: %.elf
    $(OBJCOPY) -S -O binary $< $@

$(TARGET).elf: $(A0BJS) $(COBJS) $(TARGET).lds
    $(LD) -nostdlib -T $(TARGET).lds -o $@ $(A0BJS) $(COBJS)

$(COBJS): $(HFILES)

%.o: %.S
    $(CC) $(CCARCHFLAGS) -c $<

%.o: %.c
    $(CC) $(CFLAGS) $(CCARCHFLAGS) -c $<

%.s: %.c
    $(CC) $(CFLAGS) $(CCARCHFLAGS) -S $<

%.lst: %.elf
    $(OBJDUMP) -d $^ > $@

%.sym: %.elf
    $(NM) $^ | sort > $@

.PRECIOUS: %.elf

.PHONY: clean

clean:
    rm -f *~ *.o *.s *.bin *.elf *.lst *.sym

```

5 Script del linker

5.1 Creare il file sert.lds

```

+-----+
|ENTRY(_reset)
|
|mem_start = 0x80000000;
|mem_length = 512*0x100000; /* 512 MiB */
|mem_end    = mem_start + mem_length;
|
|MEMORY
|{
|  ram (rwx) : ORIGIN = mem_start, LENGTH = mem_length
|}
|
|SECTIONS
|{
|  .text : {
|    *(.text)
|    . = ALIGN(4);
|  } > ram
|  .rodata : {
|    *(.rodata)
|    . = ALIGN(4);
|  } > ram
|  .data : {
|    *(.data)
|    . = ALIGN(4);
|  } > ram
|  .bss : {
|    _bss_start = .;
|    *(.bss)
|    *(COMMON)
|    . = ALIGN(4);
|    _bss_end = .;
|  } > ram
|}
+-----+

```

6 Accensione dei LED

6.1 Consultare il manuale BBB (Sec. 6.6)

6.2 Consultare il manuale ARM335x (Sec. 25, Sec. 2.1)

6.3 Scrivere le istruzioni per accendere gli user LED:

```

+-----+
|void _reset(void)
|{
|  int *gpio1 = (int *) 0x4804c000;
|
|  gpio1[0x194/4] |= (1<<21)|(1<<22)|(1<<23)|(1<<24);
|}
+-----+

```

```
| for(;;) |
| ; |
|} |
+-----+
```

Funziona? NO!

6.4 Consultare il manuale ARM335x (Sec. 25.3.4.3)

6.5 Aggiungere l'istruzione per impostare i pin come output:

```
+-----+
|gpio1[0x134/4] &= ~((1<<21)|(1<<22)|(1<<23)|(1<<24));|
+-----+
```

Funziona? Si!

6.6 Consultare ancora il manuale ARM335x (Sec. 8.1.12.1.29, Sec. 2.1)

6.7 Aggiungere le istruzioni per abilitare il modulo GPI01:

```
+-----+
|int *cm_per = (int *) 0x44e00000; |
|cm_per[0xac/4] = 0x40002; |
+-----+
```

6.8 Provare a disabilitare il modulo GPI01: l'accesso ai pin provoca il reset della CPU. Spiegare il motivo consultando ancora Sec. 8.1.12.1.29

=====