Uporaba urejevalnika texta Visual Studio Code za programiranje ARM zbirnika

Andrej Sušnik

October 16, 2022

1 Predpogoji za namestitev

1.1 Operacijski sistem Linux

Za uspešno programiranje in razhroščevanje ARM zbirnika moramo najprej namestiti nekaj programov.

1.1.1 GCC for ARM embedded

Da lahko prevajamo in razhroščujemo programe za vgrajene sisteme moramo namestiti GCC za arm vgrajene sisteme. To storimo z naslednjimi ukazi:

```
$ cd ~/Downloads
$ wget https://developer.arm.com/-/media/Files/downloads/gnu-rm
    /10.3-2021.10/gcc-arm-none-eabi-10.3-2021.10-x86_64-linux.tar.bz2

$ sudo mv gcc-arm-none-eabi-10.3-2021.10 /usr/share

$ sudo ln -s /usr/share/gcc-arm-none-eabi-10.3-2021.10/bin/arm-none-eabi-
    gdb /usr/bin/arm-none-eabi-gdb
$ sudo ln -s /usr/share/gcc-arm-none-eabi-10.3-2021.10/bin/arm-none-eabi-
    ld /usr/bin/arm-none-eabi-ld
$ sudo ln -s /usr/share/gcc-arm-none-eabi-10.3-2021.10/bin/arm-none-eabi-
    objcopy /usr/bin/arm-none-eabi-objcopy
$ sudo ln -s /usr/share/gcc-arm-none-eabi-10.3-2021.10/bin/arm-none-eabi-
    gcc /usr/bin/arm-none-eabi-gcc
$ sudo ln -s /usr/share/gcc-arm-none-eabi-10.3-2021.10/bin/arm-none-eabi-
    g++ /usr/bin/arm-none-eabi-g++
$ sudo ln -s /usr/share/gcc-arm-none-eabi-10.3-2021.10/bin/arm-none-eabi-
    objdump /usr/bin/arm-none-eabi-objdump
$ sudo ln -s /usr/share/gcc-arm-none-eabi-10.3-2021.10/bin/arm-none-eabi-
    size /usr/bin/arm-none-eabi-size
$ sudo ln -s /usr/share/gcc-arm-none-eabi-10.3-2021.10/bin/arm-none-eabi-
    nm /usr/bin/arm-none-eabi-nm
```

1.1.2 Visual Studio Code

Program za urejanje texta VSCode si prenesemo z spodnje povezave in ga namestimo povezava.
Ko prenesemo in namestimo VSCode. Ga odpremo in namestimo še 3 razširitve.

1. Arm Assembly - Barvanje kode za arm-ov asembler
2. Cortex-Debug - Razhroščevanje programov
3. Memory Viewer - Pregled spomina

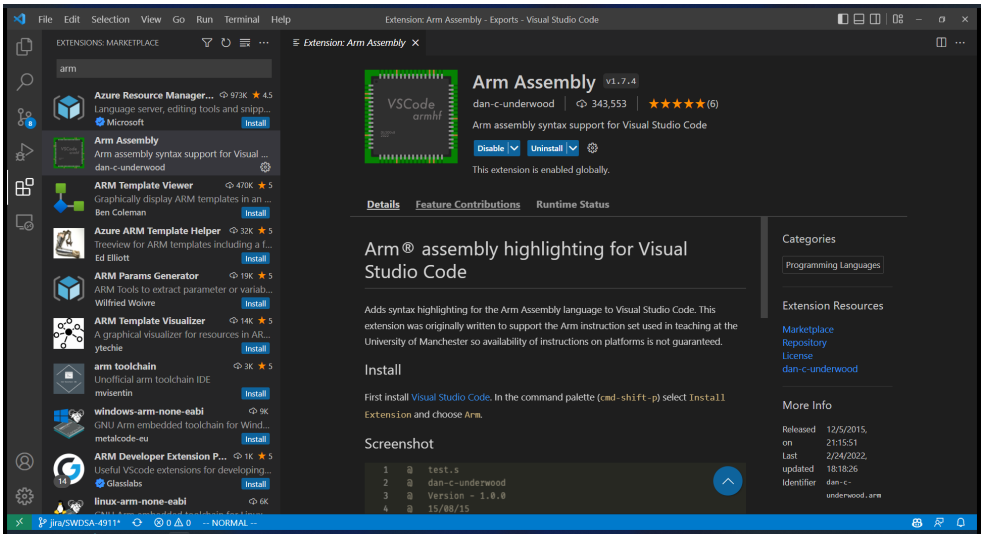

Figure 1: Razširitev Arm Assembly

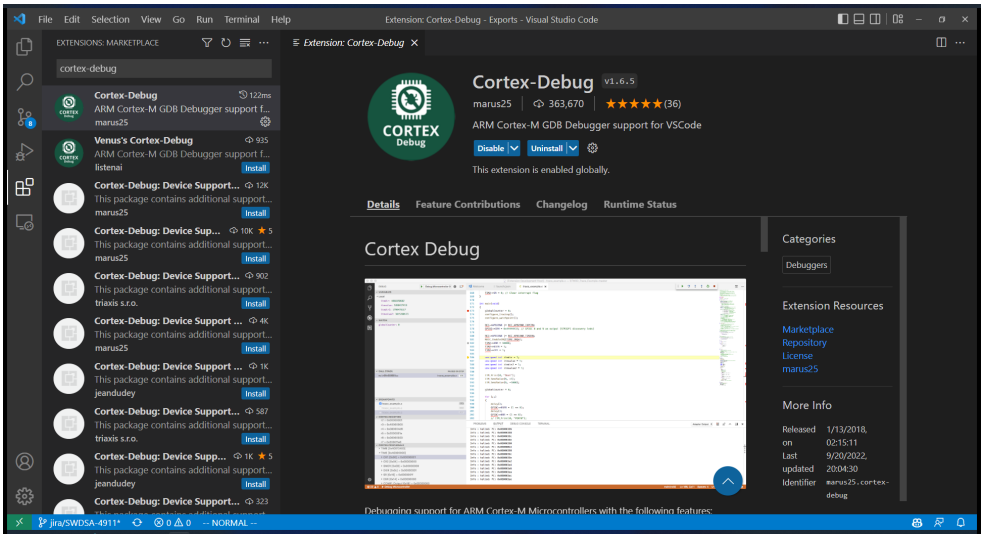

Figure 2: Razširitev Cortex-Debug

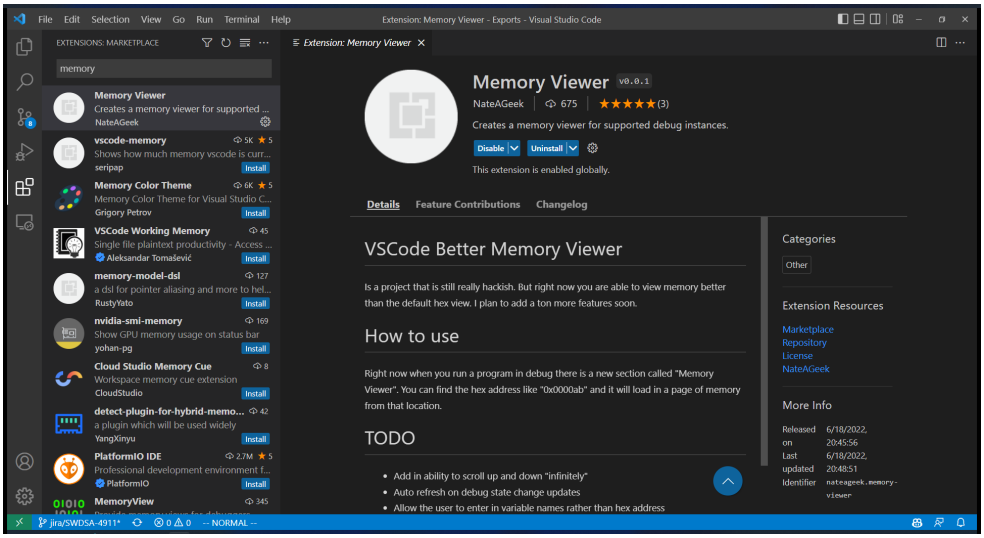

Figure 3: Razširitev Memory Viewer

1.1.3 xPack QEMU - ARM emulator

Da lahko poganjamo programe potrebujemo emulator procesorja ARM.
Dobimo ga tako, da poženemo naslednje ukaze:

```
$ cd ~/Downloadssimualtor
$ wget thttps://github.com/xpack-dev-tools/qemu-arm-xpack/releases/
    download/v7.1.0-1/xpack-qemu-arm-7.1.0-1-linux-x64.tar.gz

$ mkdir -p ~/.local/xPacks/qemu-arm
$ cd ~/.local/xPacks/qemu-arm

$ tar xvf ~/Downloads/xpack-qemu-arm-7.0.0-1-linux-x64.tar.gz
$ chmod -R -w xpack-qemu-arm-7.0.0-1

$ ln -s ~/.local/xPacks/qemu-arm/xpack-qemu-arm-7.0.0-1/bin/qemu-system-
    gnuarmeclipse /usr/bin/qemu-system-gnuarmeclipse
```

1.2 Uporaba razvojnega okolja

Z razvojem začnemo tako da prenesemo začetni projekt. To storimo z ukazom

```
git clone https://github.com/AndrejSusnik/ARM9Template
```

V visual studio code projekt odpremo tako da v glavnem meniju kliknemo File in Open folder v podmeniju, odpre se nam okno za izbiro direktorija v katerem izberemo začetni projekt.

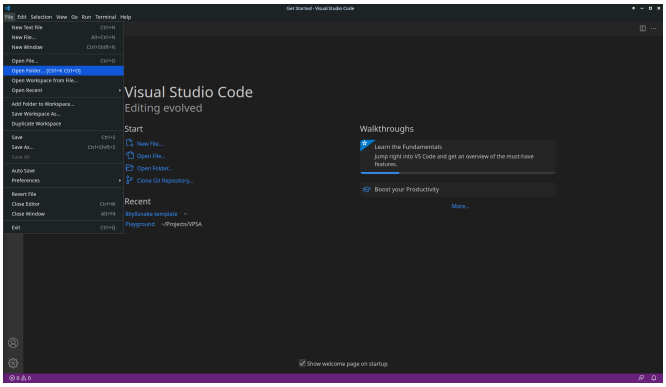

Figure 4: Odpiranje direktorija

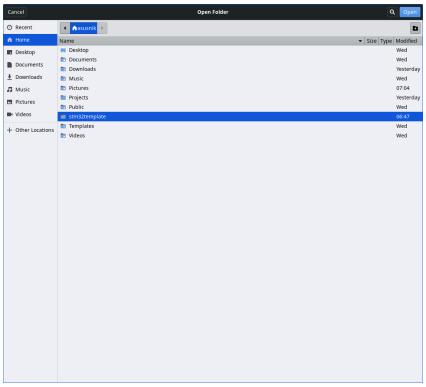

Figure 5: Odpiranje začetnega projekta

Nato odpremo zavihek debug and run

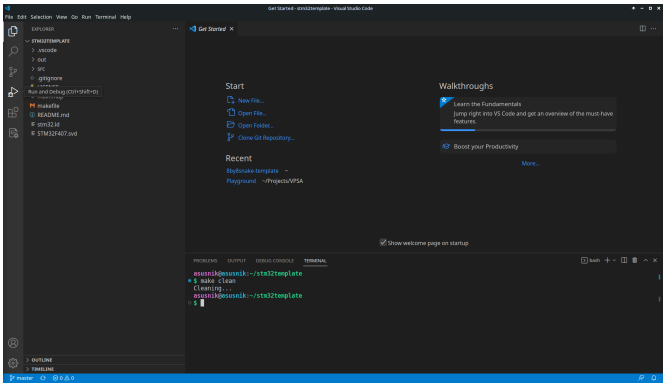

Figure 6: Debug and Run

Kjer v meniju izberemo simulator

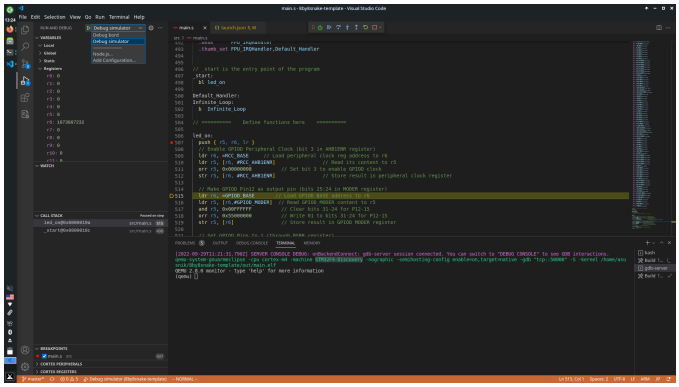

Figure 7: Izbira simulatorja

References

[1] https://code.visualstudio.com/

[2] https://www.st.com/en/microcontrollers-microprocessors/stm32-32-bit-arm-cortex-mcus.html

[3] https://www.st.com/en/development-tools/stm32cubeide.html

[4] https://developer.arm.com/downloads/-/gnu-rm

[5] https://xpack.github.io/qemu-arm/

[6] https://github.com/zrezke/8by8snake-template