**README**

**Engine-V contest entry for the RISC-V Soft CPU Contest 2018**

**Antti Lukats, MicroFPGA UG**

**November 26, 22:24 PST**

Engine-V is designed to pass all rules and requirements set in the original contest announcement. Instruction emulation by trap handler is not used, all 55 compliance tests are passing, philosophers and synchronization examples are running in both verilator test benches simulating boot either from SPI Flash or eSRAM in the MSS pre-loaded by the Cortex-M3 (M3 Assisted boot method) as in actual hardware specified by the contest rules.

Complete development was done on single Windows PC, Linux was not used at all.

Installation, requirements and dependencies

* Python 3.7
* Libero 11.9
* Icecube2 release 2017.08.27940
* Radiant release 1.0.0.350.6 (only used for programming)
* riscv-none-embedded-gcc from <https://gnu-mcu-eclipse.github.io/toolchain/riscv/>
* “ninja” stuff - explained how to install by zephyr project
* mingw (only needed if recompiling verilator testbenches)
* verilator 3.8.1 – included in the engine-V github repo
* 7zip (some files are compressed in github)
* MF8A18 compiler, executable included in engine-V github repo

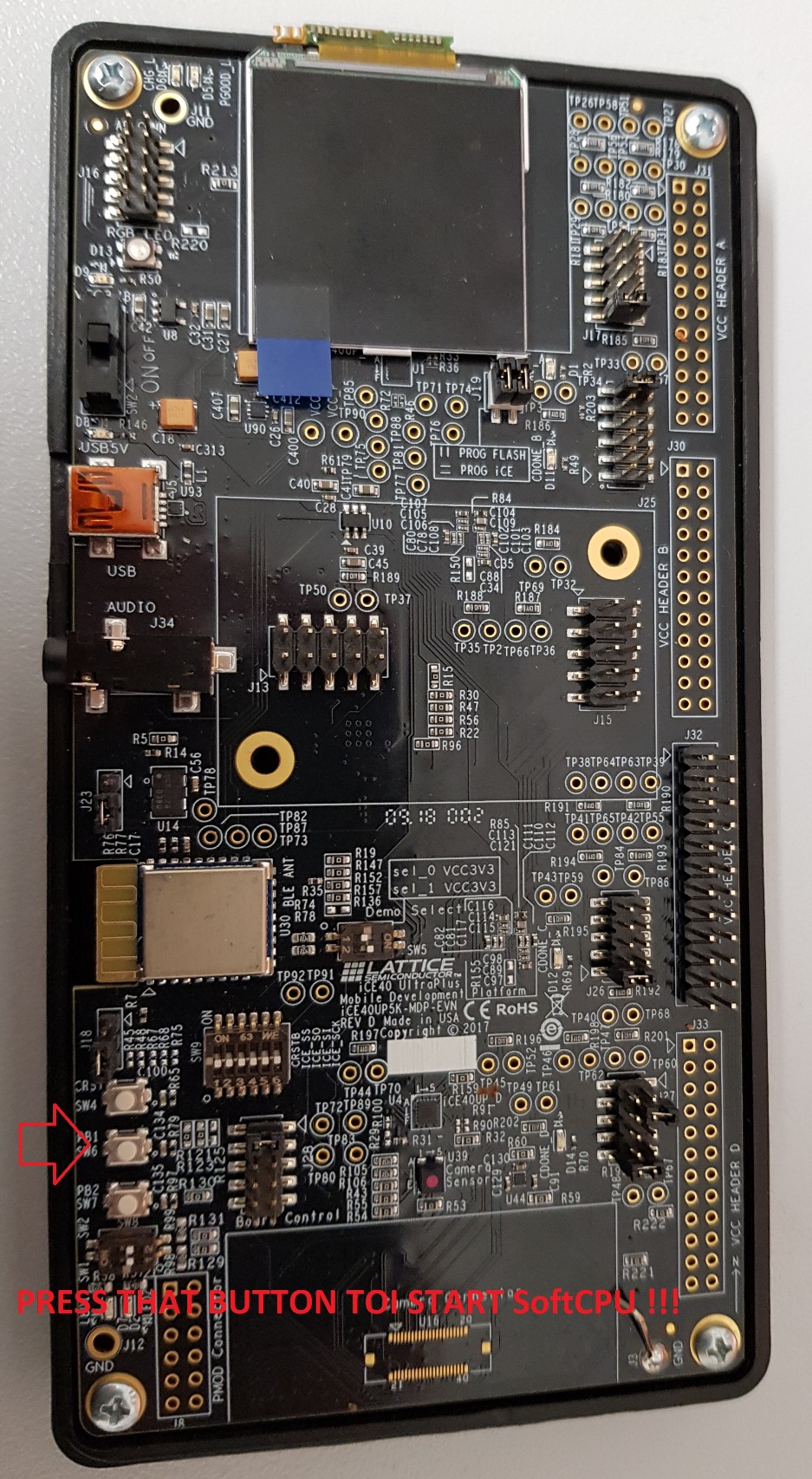
# Verilator test benches

Batch files to run RV32I tests and Zephyr apps are included in run and run\_mss folders, the first one runs simulation for engine-V configured for Lattice MDP board booting from SPI Flash, the other uses eSRAM over AHB inside MSS simulation model where eSRAM is preloaded by the verilator testbench simulating the preload by Cortex-M3 in actual hardware – Creative Board with SmartFusion2.

There is short readme about the verilator in the github repo too.

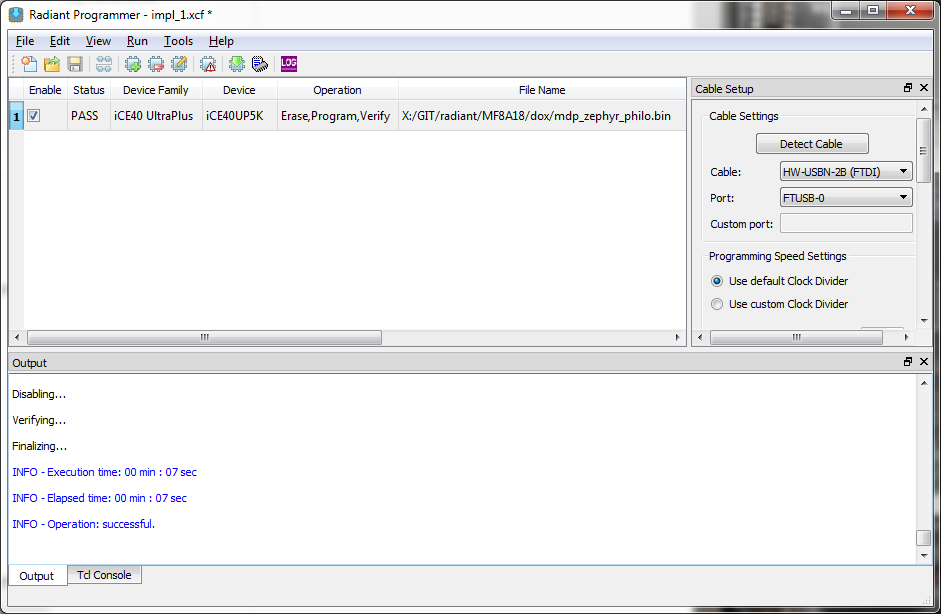
# Lattice MDP Board

IMPORTANT, it is mandatory to manually press PB1 pushbutton marked with arrow to start the SoftCPU, this is not malfunction or defect the logic to safely reset from power up was removed deliberately to save LUT resources – the need to press a button is intended functionality!

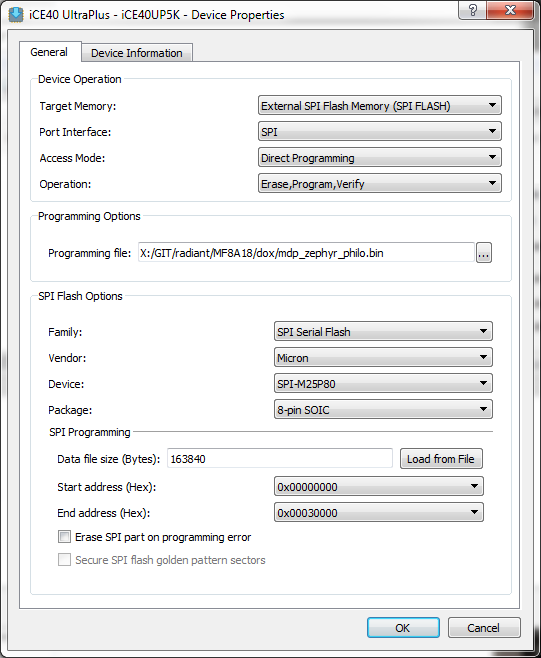


Serial port speed 115200, switches and jumpers as pictured. FPGA “C” (U3) is used by the bistreams.

SPI Flash Programming should be done with Radiant Programmer:



Device properties to be set as:



Important files size must be set manually to 163840, radiant wants to set it to smaller number when changing programming file.

After programming please press the middle button PB1 in MDP board to start engine-V.

Readymade binary programming files are in \boards\lattice\iCE40-UltraPlus-MDP\demos\zephyr

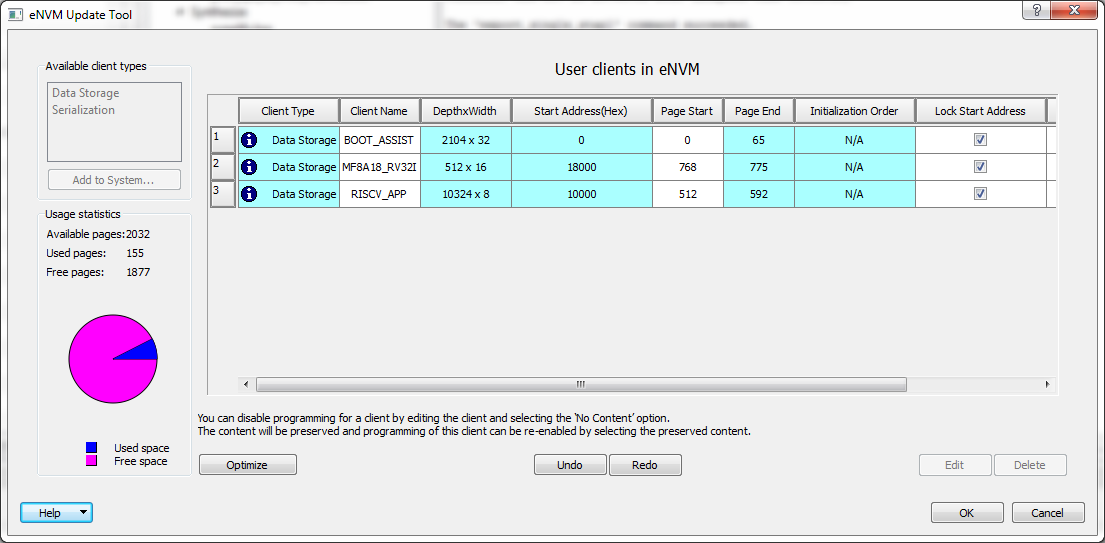
Full icecube 2 project is included in github this project will generate the FPGA bitstream without RISC-V application image that needs to be appended to the bitstream, python scripts that perform are included in the verilator folder, starting verilator test bench creates such combined bit files on the fly, the same script should be used to prepare programming files for Radiant.

To ROM512K16.v python script mem2v.py in \tools\ should be used.

# Creative Board SmartFusion2

Path boards\microchip\CreativeBoard-SF2

Full libero archive is included in repo, eNVM init files are in separate folder eNVM, they have to be manually assigned (libero uses absolute path!) with “Update eNVM Memory Content”



BOOT\_ASSIST is Cortex-M3 assisted boot application code

MF8A18 is microcode for MF8A18 engine, it uses two flash pages

RISCV\_APP is application to run

Readymade programming files are included in \bitstreams folder

Programming from Libero, RISC-V applications start automatically.

# Zephyr

On windows:

CD philosophers (or synchronization)

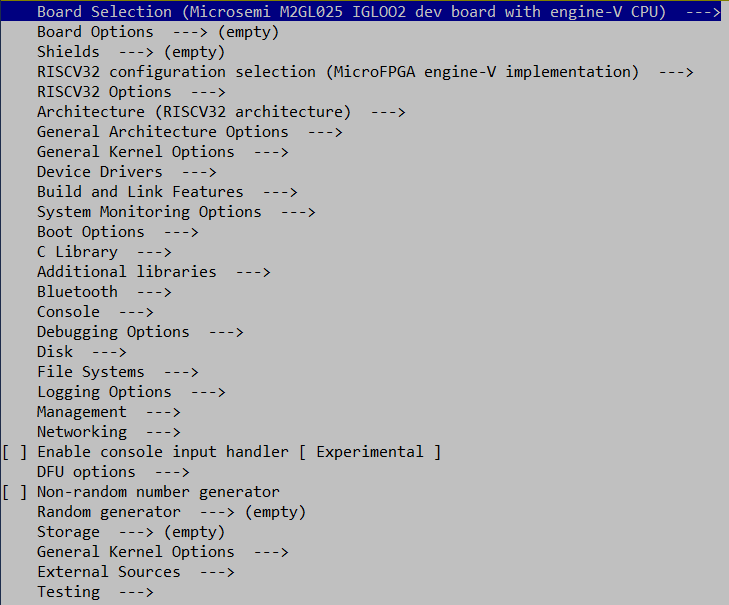
MKDIR build

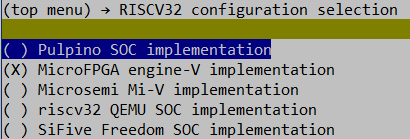
CD build

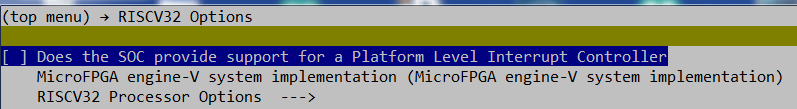
Then create make.bat file with following content:

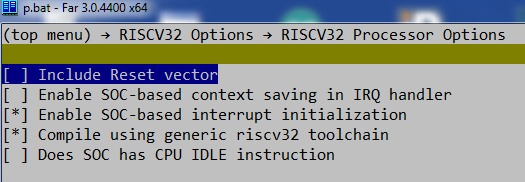
set ZEPHYR\_BASE=X:\GIT\riscv-contest\zephyr\1.13\zephyr  
set BOARD\_DIR=X:\GIT\riscv-contest\zephyr\1.13\zephyr\boards  
set BOARD=m2gl025\_ev  
set ARCH=riscv  
set TOOLCHAIN\_VENDOR=none  
set ZEPHYR\_TOOLCHAIN\_VARIANT=zephyr  
cmake -GNinja ..  
 ninja  
 ninja menuconfig

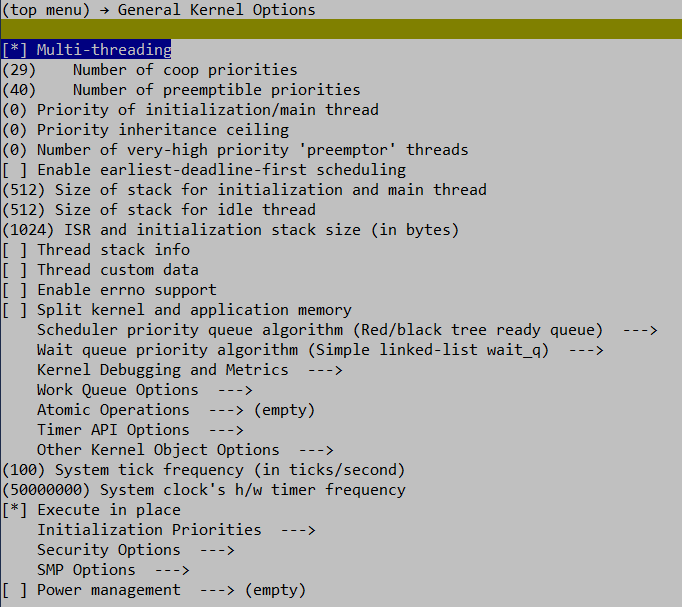
And execute make.bat

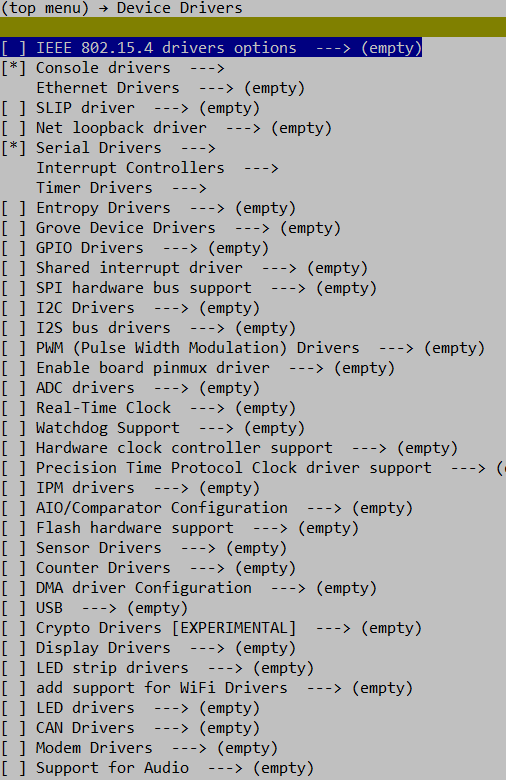


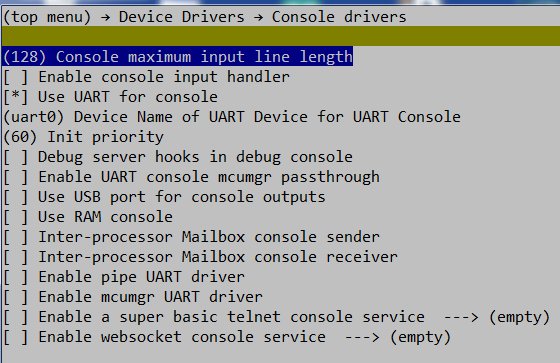


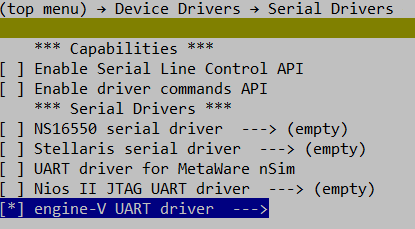


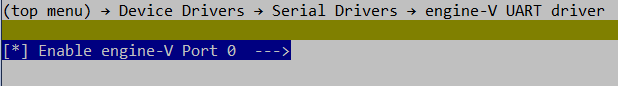


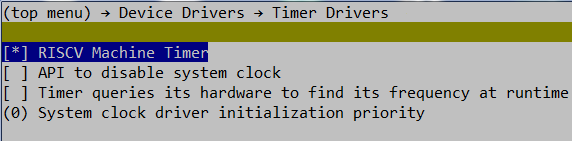


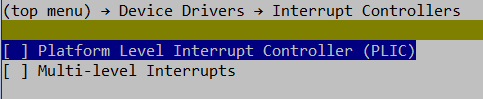












After those settings the images should be recompiled. For linux it is most likely needed to add:

**config COMPILER\_OPT  
 string  
 default "-march=RV32I"**

into file \arch\risdcv32\soc\riscv-privledge\ev\Kconfig.defconfig.series

otherwise zephyr SDK GCC would emit some mul-div instruction into the code. Both zephyr and synchronization examples have been recompiled using the zephyr github from engine-V and Zephyr SDK 0.93 on remote linux machine by a friend of mine, the binaries compiled on that remote linux PC have been tested to work on engine-V.

Zephyr images included in the engine-V repo are compiled on windows PC.

# RV32I Compliance Tests

In folder riscv-compliance/riscv-test-suite/rv32i

MKDIR build

CD build

Create file compile.bat with

**set TOOLCHAIN\_PATH=X:\GIT\riscv-contest\riscv\bin**

**set TEST\_ENV=absmin**

**set TEST\_TARGET=absmin**

**set CFLAGS=-march=rv32im -mabi=ilp32 -DNO\_IO\_ASSERT**

**set TEST=%1**

**del startfiles**

**%TOOLCHAIN\_PATH%\riscv-none-embed-gcc.exe %CFLAGS% -I..\..\..\riscv-target\%TEST\_TARGET% -I..\..\..\riscv-test-env -I..\..\..\riscv-test-env\%TEST\_ENV% -DPREALLOCATE=1 -mcmodel=medany -static -nostdlib --std=gnu99 -O3 -ffast-math -fno-common -fno-builtin-printf -Wl,-static,-nostdlib,-nostartfiles,-lm,-lgcc,-T ..\..\..\riscv-test-env\%TEST\_ENV%\link.ld ..\src\%TEST%.S**

**%TOOLCHAIN\_PATH%\riscv-none-embed-objdump.exe -D startfiles > %TEST%.txt**

**copy startfiles %TEST%.elf**

**%TOOLCHAIN\_PATH%\riscv-none-embed-objcopy.exe -Obinary %TEST%.elf %TEST%.bin**

**%TOOLCHAIN\_PATH%\riscv-none-embed-objcopy.exe -O verilog %TEST%.elf %test%.mem**

**%TOOLCHAIN\_PATH%\riscv-none-embed-objcopy.exe -O ihex %TEST%.elf %test%.hex**

**copy %test%.bin B:\RD\riscv\contest\Win32\Debug\images\%test%.bin**

**copy %test%.mem X:\GIT\verilator\engine-V\images\%test%.mem**

**del startfiles**

and a file do\_all.bat with

**call compile.bat I-IO-01**

**call compile.bat I-RF\_size-01**

**call compile.bat I-RF\_width-01**

**call compile.bat I-RF\_x0-01**

**call compile.bat I-DELAY\_SLOTS-01**

**call compile.bat I-ECALL-01**

**call compile.bat I-EBREAK-01**

**call compile.bat I-MISALIGN\_JMP-01**

**call compile.bat I-MISALIGN\_LDST-01**

**call compile.bat I-CSRRW-01**

**call compile.bat I-CSRRS-01**

**call compile.bat I-CSRRC-01**

**call compile.bat I-CSRRWI-01**

**call compile.bat I-CSRRSI-01**

**call compile.bat I-CSRRCI-01**

**call compile.bat I-ADD-01**

**call compile.bat I-ADDI-01**

**call compile.bat I-AND-01**

**call compile.bat I-ANDI-01**

**call compile.bat I-SUB-01**

**call compile.bat I-SLT-01**

**call compile.bat I-SLTU-01**

**call compile.bat I-SLTI-01**

**call compile.bat I-SLTIU-01**

**call compile.bat I-OR-01**

**call compile.bat I-ORI-01**

**call compile.bat I-XOR-01**

**call compile.bat I-XORI-01**

**call compile.bat I-AUIPC-01**

**call compile.bat I-LUI-01**

**call compile.bat I-BEQ-01**

**call compile.bat I-BNE-01**

**call compile.bat I-BLT-01**

**call compile.bat I-BGE-01**

**call compile.bat I-BLTU-01**

**call compile.bat I-BGEU-01**

**call compile.bat I-LB-01**

**call compile.bat I-LH-01**

**call compile.bat I-LW-01**

**call compile.bat I-LBU-01**

**call compile.bat I-LHU-01**

**call compile.bat I-SB-01**

**call compile.bat I-SH-01**

**call compile.bat I-SW-01**

**call compile.bat I-JAL-01**

**call compile.bat I-JALR-01**

**call compile.bat I-NOP-01**

**call compile.bat I-FENCE.I-01**

**call compile.bat I-DELAY-SLOTS-01**

**call compile.bat I-ENDIANESS-01**

**call compile.bat I-SLTI-01**

**call compile.bat I-SLTIU-01**

**call compile.bat I-SLLI-01**

**call compile.bat I-SRLI-01**

**call compile.bat I-SRAI-01**

**call compile.bat I-SLL-01**

**call compile.bat I-SLT-01**

**call compile.bat I-SLTU-01**

**call compile.bat I-SRL-01**

**call compile.bat I-SRA-01**

then execute do\_all.bat

sorry, those batch files should have been in the github repo!

# MF8A18

Special free to use “Contest Edition” of MF8A18 compiler is included in github

# Things that should be there but are not

Current implementation of engine-V on Microsemi actually includes full support for single stepping debugger and hardware assisted EBREAK processing, it was planned this to MF8A18 code, but this is still on the to-do list. The code would be less 30 words and still fit to 512 words (0.5 of one LSRAM). Cortex-M3 code to support debugger also missing. And Jupyter notebook to talk to the debugger also. Time did run out. One day missing.