嵌入式系统联谊会
www.esbf.org

IAR SYSTEMS

Professional development tools for RISC-V

Ryan Sheng, IAR Systems (China)

Future-proof software tools and services for embedded development

- Dedicated team of support, sales and service worldwide
- 46,000 customers
- 32% of revenue invested in R&D

36 years in the industry
Listed on NASDAQ Stockholm

Uppsala	Shanghai	+ Distributor representation in 40+ countries
Munich	Dallas	
Paris	Boston	
Tokyo	Los Angeles	
Seoul	San Francisco	

2018
- Sales SEK 385M
- Operating profit SEK 116M
- Net cash SEK 114M

The world’s most widely used development tools for embedded applications

Be free! Build what you want in the platform of your choice.

IAR Embedded Workbench

150,000
USERS
WORLDWIDE

12,000+
SUPPORTED
DEVICES

36
YEARS OF
EXPERIENCE

IAR Embedded Workbench

Complete C/C++ compiler and debugger toolchain

IAR SYSTEMS

Widest device support

Industry-leading code optimization technology

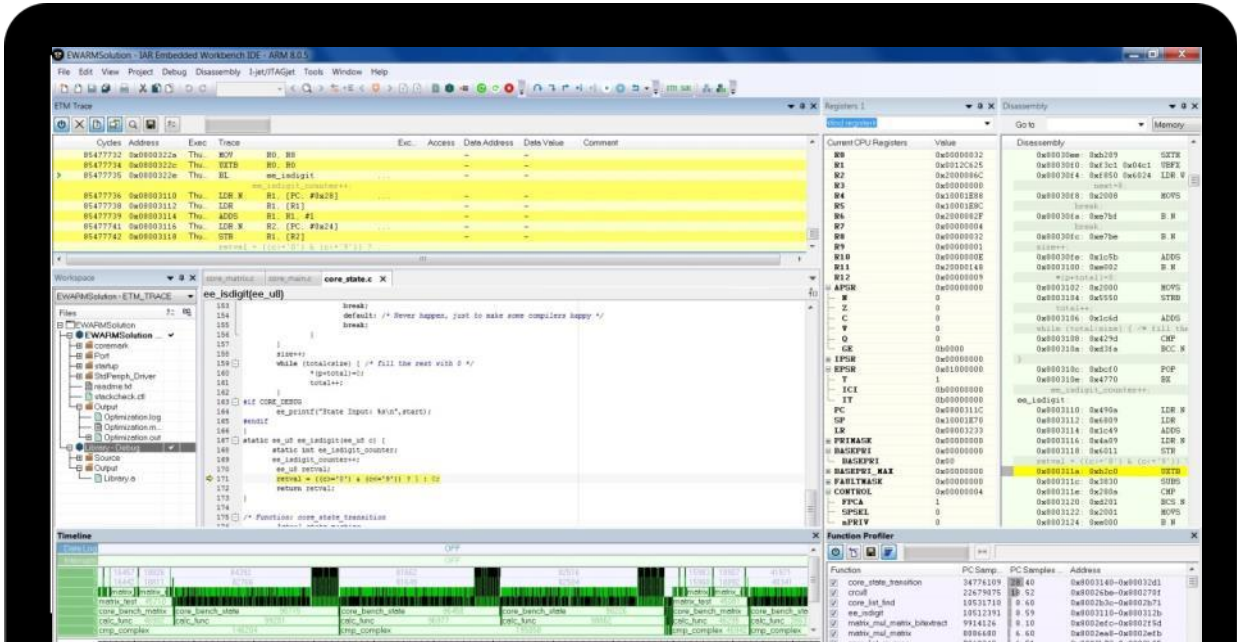
Comprehensive debugger

Integrated static and runtime code analysis

Functional safety certified versions

ISO/ANSI C/C++ compliance with C11 and C++14

Global support and training



Safety tested · Production monitored · TÜV SÜD · Functional Safety

CMSIS COMPLIANT
ARM® Cortex™ Microcontroller Software Interface Standard

Support for 12,000+ devices

40+ architectures

All available 8-,16- and 32-bit MCUs

Cortex-M0	Cortex-R8	AVR	H8
Cortex-M0+	Cortex-A5	AVR32	STM8
Cortex-M1	Cortex-A7	RX	ColdFire
Cortex-M3	Cortex-A8	RL78	HCS12
Cortex-M4	Cortex-A9	RH850	S08
Cortex-M7	Cortex-A15	78K	MAXQ
Cortex-M23	ARM11	SuperH	CR16C
Cortex-M33	ARM9	V850	SAM8
Cortex-R4	ARM7	R32C	RISC-V
Cortex-R5	SecurCore	M32C	
Cortex-R52	8051	M16C	
Cortex-R7	MSP430	R8C	

Milestones


Q3 2017
Small pre-dev on compiler for RISC-V
Q2 2018
Official start of the development project
June 2019
First release
TBD
Functional safety release
2016
Starts exploring RISC-V
Q1 2018
Joins the RISC-V foundation
Q1 2019
First beta (limited for partners and demos)

Compiler

- Proprietary design based on over 36 years of experience
- Based on a platform that are common between different targets to handle global optimizations, language compliance, etc.
  - Different architecture, one solution
  - Easy source code migration between different MCU targets
- Target unique backend for specific adaptations and optimizations
- RISC-V specifics
  - Primary target will be supporting standard extensions
  - Investigation in process for how to support customized extensions

Challenges on optimization

- Comparing to more complex instruction sets, RISC-V has some challenges when it comes to code size optimization
  - Arithmetic's with higher resolution than the natural data size yields larger code;
  - Absence of carry flags and instructions to save and restore multiple registers are other examples;
- IAR’s initial optimization target is reducing the code size
  - Main focus of IAR has always been to supply the best code size and speed on the market.

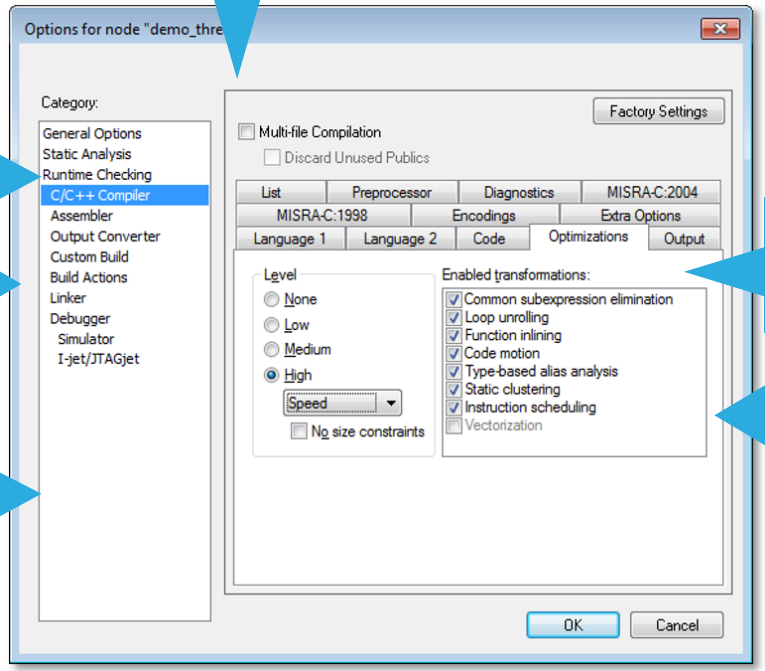
IAR C/C++ Compiler

Multi-file compilation allows the optimizer to operate on a larger set of code

Multiple levels of optimizations for code size and execution speed

The linker can remove unused code

Option to maximize speed with no size constraints



Major functions of the optimizer can be controlled individually

Balance between size and speed by setting different optimizations for different parts of the code

Language standards

- ISO/IEC 14882:2015 (C++14, C++17)
- ISO/IEC 9899:2012 (C11)
- ANSI X3.159-1989 (C89)

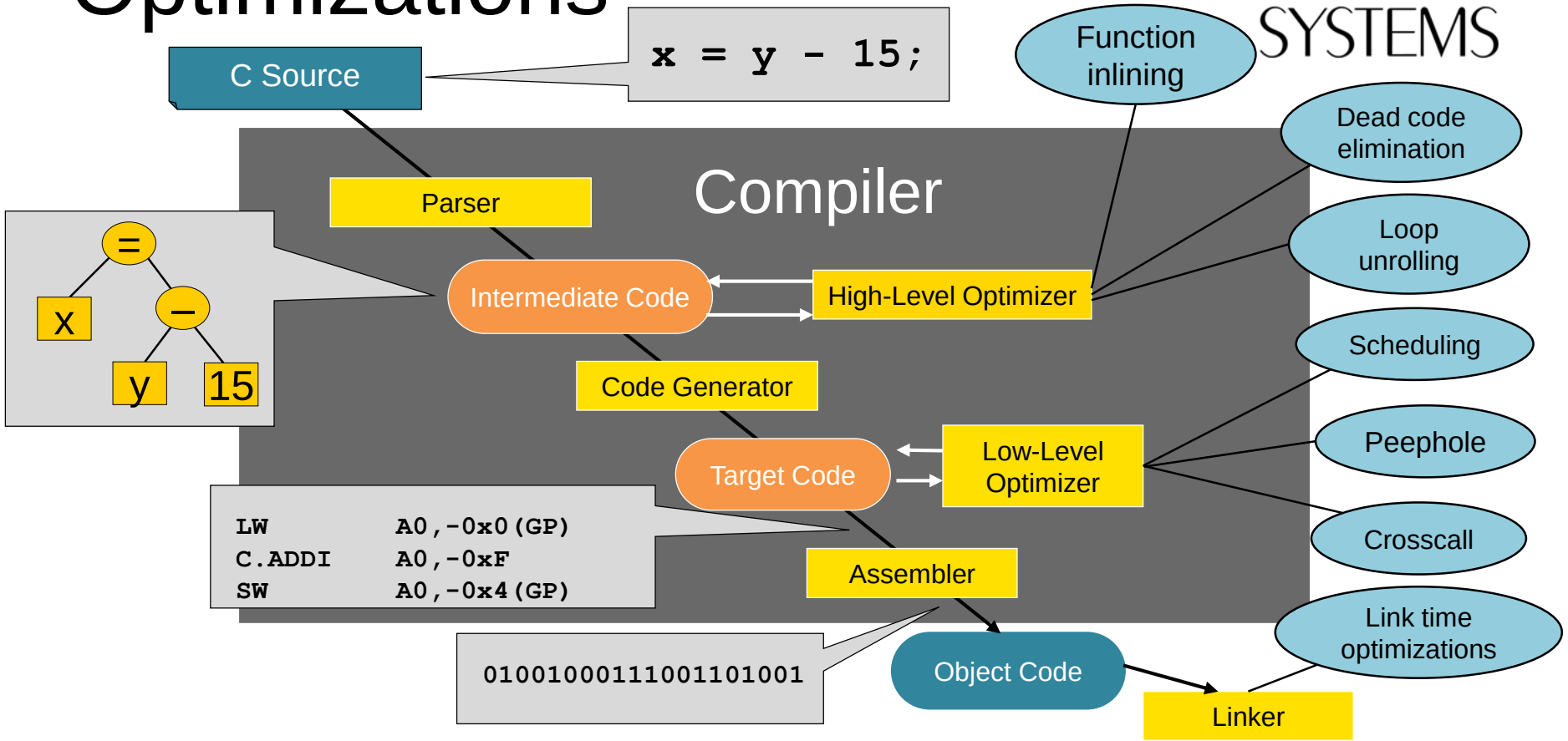
- IEEE 754 standard for floating-point arithmetic

Well-tested

Commercial test suites
- Plum-Hall Validation test suite
- Perennial EC++VS
- Dinkum C++ Proofer

In-house developed test suite >500,000 lines of C/C++ test code run multiple times
- Processor modes
- Memory models
- Optimization levels

Optimizations



Speed, size or both?

Optimization	Effect	
Common sub-expressions	Speed ↑	Size ↓
Loop unrolling	Speed ↑	Size ↑
Function inlining	Speed ↑	Size ↑
Code motion	Speed ↑	Size →
Dead code elimination	Speed →	Size ↓
Static clustering	Speed ↑	Size →
Instruction scheduling	Speed ↑	Size →
Peephole	Speed ↑	Size ↓
Cross call	Speed ↓	Size ↓

To-do list in code optimizations before the formal release 1.10

- Cross call and cross jump
- Hoisting
- Common sub-expressions elimination
- Additional link time optimizations
- Optimized use of compact instructions
- Numerous peephole optimizations

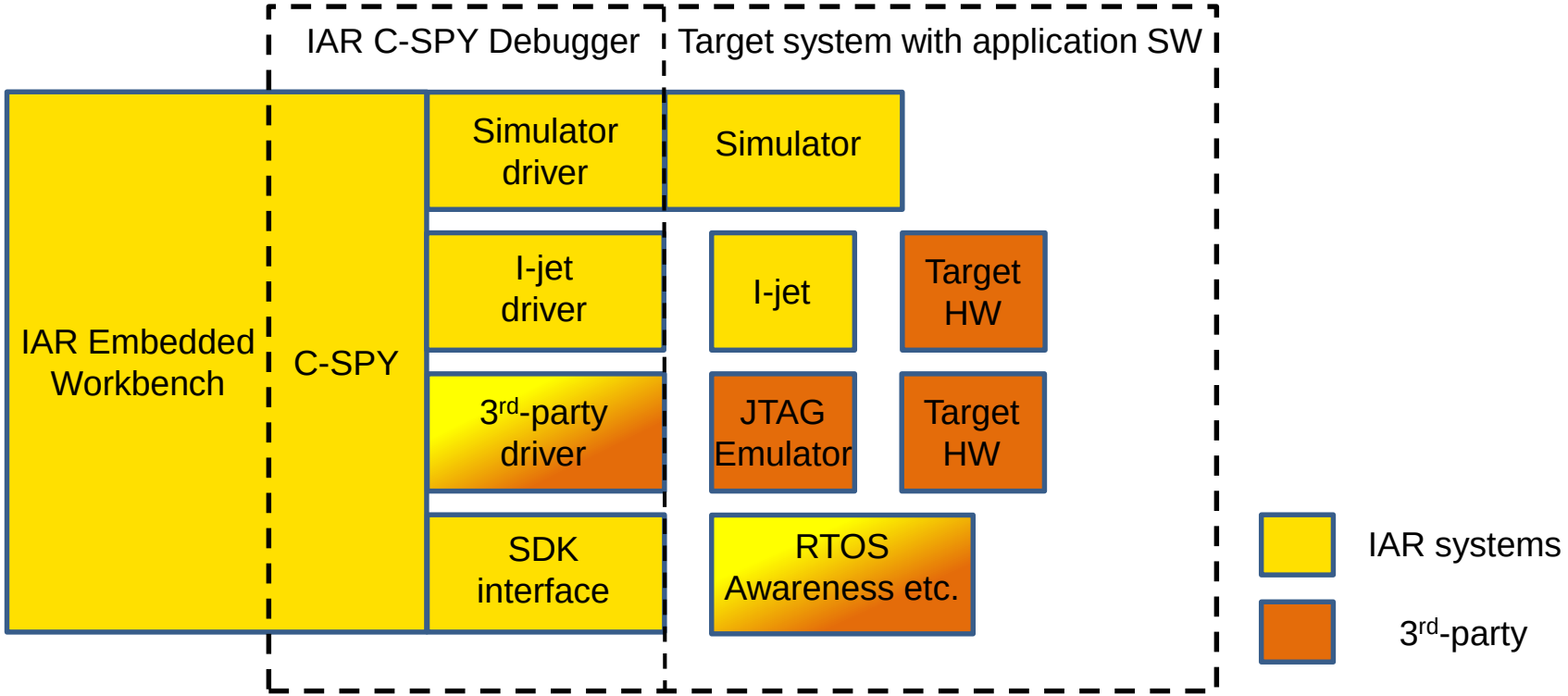
Debugging

- IAR Embedded Workbench for RISC-V will have a fully integrated debugging solution which follows the debug standard of RISC-V Foundation (version 0.13 so far).

- It will reuse the C-SPY debugger interface that have been used in many of other IAR Embedded Workbench products.
  - Rich debug macro language, etc.

- Probes
  - IAR I-jet will support RISC-V.
  - Support for major 3rd-party probes will be added later.

IAR C-SPY debugger overview



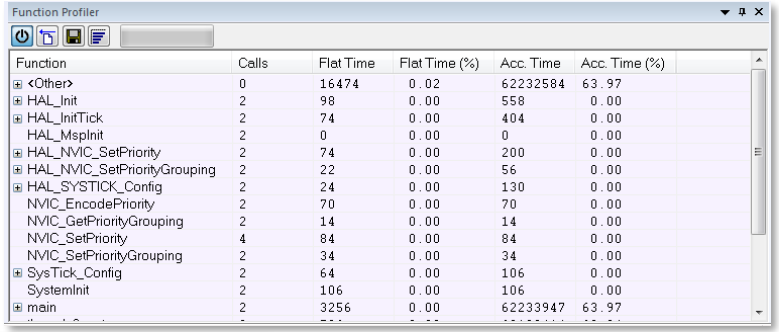
I-jet in-circuit debug probe

- Supports RISC-V, ARM7/ARM9/ARM11 and Cortex-M/R/A cores
- USB interface
- Target power of up to 400mA can be supplied from I-jet with overload protection
- Target power consumption can be measured with ~200µA resolution at 200kHz
- JTAG and Serial Wire Debug (SWD) clocks up to 32MHz
  (no limit on the MCU clock speed)
- Support for SWO speeds of up to 60MHz
- Unlimited flash breakpoints
  (to be added for RISC-V)

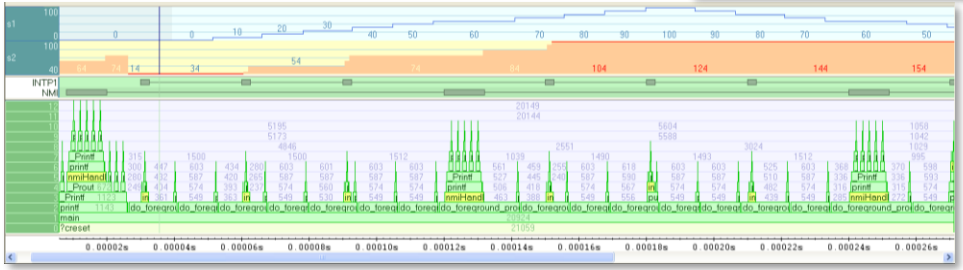
Integrated profiling tools

Function profiling

- Based on simulator, sampled trace or full trace
- Execution time per function
- Select time interval



Function	Calls	Flat Time	Flat Time (%)	Acc. Time	Acc. Time (%)
<Other>	0	16474	0.02	62232584	63.97
HAL_Init	2	98	0.00	558	0.00
HAL_InitTick	2	74	0.00	404	0.00
HAL_MspInit	2	0	0.00	0	0.00
HAL_NVIC_SetPriority	2	74	0.00	200	0.00
HAL_NVIC_SetPriorityGrouping	2	22	0.00	56	0.00
HAL_SYSTICK_Config	2	24	0.00	130	0.00
NVIC_EncodePriority	2	70	0.00	70	0.00
NVIC_GetPriorityGrouping	2	14	0.00	14	0.00
NVIC_SetPriority	4	84	0.00	84	0.00
NVIC_SetPriorityGrouping	2	34	0.00	34	0.00
SysTick_Config	2	64	0.00	106	0.00
SystemInit	2	106	0.00	106	0.00
main	2	3256	0.00	62233947	63.97

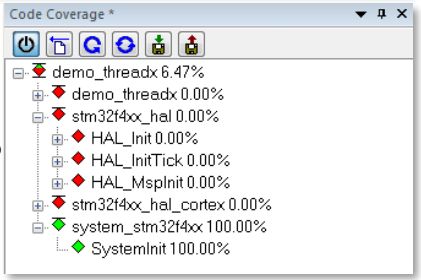


Timeline window shows the application's profile

Interrupt log, Data log, Event log, Call stack

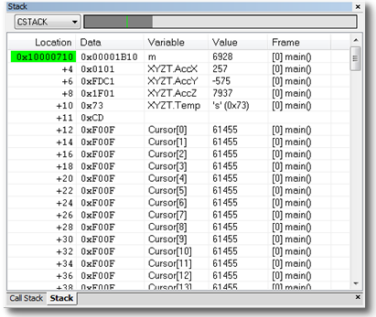
Code coverage analysis

Which code has been executed?



Stack analysis

Calculates maximum stack usage, helps find the optimal stack size, and checks stack integrity at runtime to detect overflow



Solution for safety critical applications

Certified toolchain

- A special functional safety edition of IAR Embedded Workbench

Simplified validation

- Functional safety certificate from TÜV SÜD
- Safety report from TÜV SÜD
- Safety guide

Guaranteed support through the product life cycle

- Prioritized support
- Validated service packs
- Regular reports of known problems

Standards:

IEC 61508
ISO 26262
EN 50128
IEC 62304

High level timeplan

First release

- RV32IMFC
- Initial compiler optimizations, focus on code size
- IAR simulator
- I-jet debugger
- C11 (no RISC-V ABI compliant), C++17
- C-STAT (static code analysis)
- Official support for a set of CPU cores

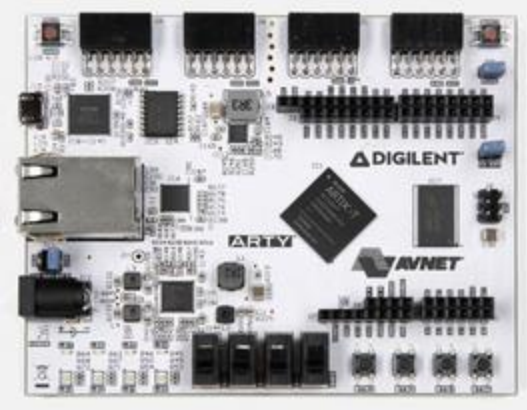
High level timeplan

Future releases

- Improved optimizations for code size and speed
- Atomics
- RV32E
- RV64I
- C RISC-V ABI compliance
- C-RUN (runtime code analysis)
- Trace
- Functional safety certification
- 3rd-party debug probes

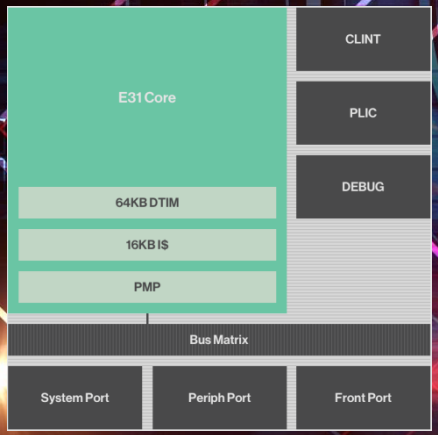
Demonstration

Artix-7
Arty
FPGA
Kit

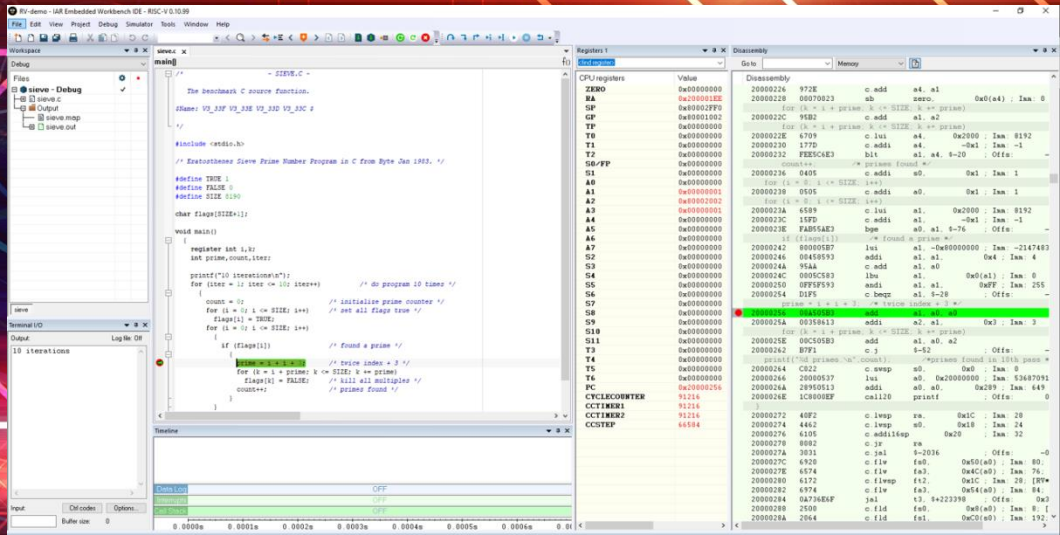


+

SiFive
E31
Core



IAR Embedded
Workbench for
RISC-V + I-jet




IAR
SYSTEMS
Thanks for your attention!
www.iar.com/riscv