Abstract

A method for extension of a virtual machine dedicated for single-core environment on multi-core environment is presented. The cores execute control projects of the IEC 61131-3 standard involving mutual exchange of data. The virtual machine is understood as a processor implemented by software which executes certain universal intermediate code generated by a compiler of source program. Implementation by software and universal intermediate code enable installation of the virtual machine on various processors. CPDev developed in RUT is an example of single-core environment involving the virtual machine, and used by industrial controllers with 16- and 32-bit microprocessors, as well as by x86-64. The CPDev with unique VMASM intermediate language is the basis for this work.

It is assumed that each core of the processor involves the CPDev virtual machine extended by data exchange with the other cores through shared memory. Therefore the method for extension of the single-core environment on several cores consists of the following rules: 1) the same declarations of the exchanged global variables in projects for the cores, 2) additional attributes of these variables to define reading or writing from/to the shared memory, 3) exchange of the variables at the beginning and end of the core control cycle, 4) elimination of the core conflicts in the shared memory access. It is important that usage of a RTOS real-time operating system is not required.

Denotational model of the virtual machine operation is presented, as well as the models and C/C++ realizations of selected instructions of the VMASM intermediate language, together with the model of conflict-free access to the shared memory. The denotational models can be implemented in any programming language. In addition, it is shown how efficiency of the virtual machine and size of the code memory depend on memory access methods such as byte access, system *memcpy*, and pointer-based direct access.

The first of the presented examples concerns execution of two related IEC 61131-3 projects by two cores of the STM32 microcontroller in a development board. The projects are written in FBD and ST languages, with conflicts in the shared memory access eliminated by a hardware semaphore. Due to different execution cycles of the projects, the multi-threaded WinController is applied for simulation of the cooperating cores.

The laboratory system for identification of RFID transponders is the second example. The system consists of a small mobile robot with dual-core ESP32 microcontroller and quad-core Raspberry Pi microcomputer as a master unit. Three virtual machines assigned to isolated cores operate in the system, exchanging the data through the shared memory or Wi-Fi. Mutex mechanism prevents memory access conflicts. Linux (Raspberry Pi) and FreeRTOS (ESP32)

operating systems are excluded from scheduling the tasks to which the virtual machines are assigned. After simulation of the common project for the whole system, it has been partitioned into tasks for the cores.