Streszczenie

Przedstawiono metodę rozszerzenia maszyny wirtualnej przeznaczonej dla środowiska jednordzeniowego na środowisko wielordzeniowe, w którym rdzenie procesora wykonują projekty sterowania normy IEC 61131-3 wymieniając między sobą dane. Jako maszynę wirtualną należy rozumieć procesor zrealizowany programowo wykonujący pewien uniwersalny kod pośredni generowany przez kompilator programu źródłowego. Programowa realizacja i uniwersalny kod pośredni umożliwiają implementację maszyny na różnych procesorach. Przykładem jednordzeniowego środowiska z maszyną wirtualną jest CPDev opracowany w PRz i stosowany w sterownikach z procesorami 16- i 32-bitowymi oraz x86-64. CPDev z unikalnym językiem pośrednim VMASM stanowi bazę niniejszej pracy.

Przyjęto, że w każdym rdzeniu procesora funkcjonuje osobna maszyna wirtualna CPDev rozszerzona o wymianę danych z innymi rdzeniami za pośrednictwem pamięci współdzielonej. W związku z tym na metodę rozszerzenia środowiska jednordzeniowego na więcej rdzeni składają się następujące zasady: 1) jednakowe deklaracje wymienianych zmiennych globalnych w projektach dla rdzeni, 2) dodatkowe atrybuty tych zmiennych określające zapis lub odczyt do/z pamięci współdzielonej, 3) wymiana zmiennych na początku i końcu cyklu sterowania wykonywanego przez rdzeń, 4) eliminacja konfliktów rdzeni w dostępie do pamięci współdzielonej. Istotne jest, że nie wymaga się wykorzystania systemu operacyjnego czasu rzeczywistego RTOS.

Przedstawiono model denotacyjny funkcjonowania maszyny wirtualnej, modele i realizacje C/C++ wybranych instrukcji języka pośredniego VMASM, jak również model bezkolizyjnego dostępu do pamięci współdzielonej. Modele denotacyjne wskazują wprost na rozwiązania programowe, które można implementować w dowolnym języku. Pokazano ponadto jak na wydajność maszyny oraz na rozmiar pamięci kodu wpływają trzy metody dostępu do pamięci, tzn. dostęp bajtowy, systemowe *memcpy* oraz bezpośredni dostęp wskaźnikowy.

Pierwszy z przedstawionych przykładów dotyczy wykonywania dwóch powiązanych projektów IEC 61131-3 przez dwa rdzenie mikrokontrolera STM32 na płycie uruchomieniowej. Projekty napisano w językach FBD i ST, a do eliminacji konfliktów w dostępie do pamięci współdzielonej wykorzystano semafor sprzętowy. Ze względu na odmienne cykle wykonywania projektów, do symulacji współpracy rdzeni wykorzystano wielowątkowy WinController.

Drugim przykładem jest laboratoryjny system identyfikacji znaczników RFID, w skład którego wchodzi mały robot mobilny z dwurdzeniowym mikrokontrolerem ESP32 oraz

czterordzeniowy mikrokomputer Raspberry Pi jako jednostka nadrzędna. W systemie funkcjonują trzy maszyny wirtualne przypisane do izolowanych rdzeni, które wymieniają dane poprzez pamięć współdzieloną lub przez Wi-Fi. Mechanizm muteks nie dopuszcza do kolizji w dostępie do pamięci. Systemy operacyjne Linux (Raspberry Pi) i FreeRTOS (ESP32) zostały wykluczone z harmonogramowania zadań, do których przypisano maszyny wirtualne. Eksploracja jest półautomatyczna ze względu na brak bezprzewodowej lokalizacji. Po symulacji wspólnego projektu dla całego systemu rozdzielono go na zadania dla rdzeni.