Sprawozdanie z laboratorium OSK

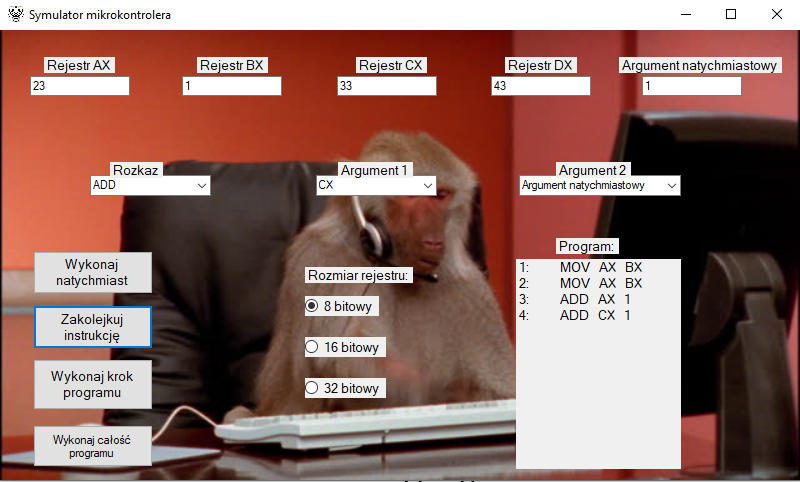
Projekt V

Emulator mikrokontrolera

1. Opis projektu

Nasz emulator zakłada istnienie 4 rejestrów roboczych + możliwości wpisania argumentu natychmiastowego. Możliwe jest wykonanie rozkazów MOV, ADD oraz SUB. Rejestry możemy ustawić jako 8, 16 lub 32 bitowe, co zmieni dopuszczalną maksymalną wartość w rejestrze.

1. Interfejs

* rys 1. Zrzut widoku aplikacji*

Możliwe do wykonania rozkazy dobieramy z elementów *ComboBox*, aby zapobiec błędom w składni.

Rozkazy są przechowywane w pamięci komputera jako specjalne struktury w kolejce <*Queue>*. Wbudowany typ *Queue* jest rejestrem FIFO, więc pierwszy załadowany rozkaz zostanie wykonany jako pierwszy.

Rozkazy można wykonać od razu, lub zakolejkować do programu.

Tak wygląda kod wykonywania rozkazu:

private void wykonaj\_rozkaz(rozkaz roz)

{

UInt64 temp1 = Convert.ToUInt64(nazwa\_na\_rejestr(roz.arg1).Text);

UInt64 temp2 = 0;

if (roz.arg2 == "AX" || roz.arg2 == "BX" || roz.arg2 == "CX" || roz.arg2 == "DX")

temp2 = Convert.ToUInt64(nazwa\_na\_rejestr(roz.arg2).Text);

else

temp2 = Convert.ToUInt64(roz.arg2);

switch (roz.roz)

{

case "MOV":

nazwa\_na\_rejestr(roz.arg1).Text = nazwa\_na\_rejestr(roz.arg2).Text;

return;

case "ADD":

if(temp1+temp2>this.wielkosc\_rej)

nazwa\_na\_rejestr(roz.arg1).Text = Convert.ToString(temp1 + temp2 - 255);

else

nazwa\_na\_rejestr(roz.arg1).Text = Convert.ToString(temp1 + temp2);

return;

case "SUB":

if (temp1 >= temp2)

nazwa\_na\_rejestr(roz.arg1).Text = Convert.ToString(temp1 - temp2);

else

nazwa\_na\_rejestr(roz.arg1).Text = Convert.ToString(255 + temp1 - temp2);

return;

default:

break;

}

}

1. Podsumowanie

Przyjęta struktura programu pozwala na rozszerzenie jego funkcjonalności. Jednak jako że nie operujemy na bitach, implementacja np. rejestru flag mogła by przysporzyć kłopotów.