AGH

AKADEMIA GÓRNICZO-HUTNICZA
IM. STANISŁAWA STASZICA W KRAKOWIE

Wydział Informatyki, Elektroniki i Telekomunikacji

Prosty mikrokontroler w logice programowalnej - sprawozdanie z laboratorium.

Rok akademicki 2018/2019

Mateusz Szwed
Piotr Dulęba

05.03.2019

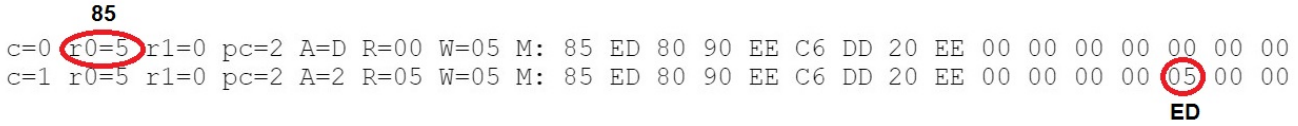
1 Pisanie własnego programu (zadanie 3.5)

W pliku `mem_init_start.coe` zapisaliśmy wymagane instrukcje, które dawały następujące rezultaty:

1.1 Wpisanie liczby 0x05 do komórki o adresie 13.

Opis	OPCODE	MNEMONIC	BYTE CODE
wpisuje liczbę 0x05 do rejestru R0	85	MOV 0x05	10000101
wpisuje wartość spod rejestru R0 pod adres 13	ED	STR 0, 0xD	11101101

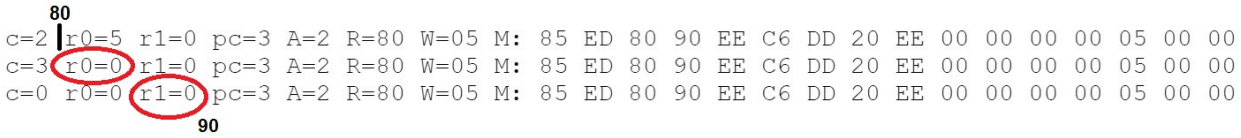
Rezultat:



1.2 Wyczyścić zawartość rejestrów R0 i R1, wpisując do nich 0.

Opis	OPCODE	MNEMONIC	BYTE CODE
wpisuje liczbę 0x00 do rejestru R0	80	MOV 0x0	10000000
wpisuje liczbę 0x00 do rejestru R1	90	MOV 0x0	10010000

Rezultat:



1.3 Wyczyścić zawartość komórki o adresie 14, wpisując do niej 0.

Opis	OPCODE	MNEMONIC	BYTE CODE
wpisuje liczbę z rejestru R0 do komórki o adresie 14	EE	STR 0, 0xE	11101110

Rezultat:

1.4 Do rejestru R0 wpisać liczbę 0x6

Opis	OPCODE	MNEMONIC	BYTE CODE
wpisuje liczbę 0x6 do rejestru R0	86	MOV 0x6	1000 0110

Rezultat:

```
                86
c=0 r0=6 r1=0 pc=6 A=5 R=86 W=00 M: 85 ED 80 90 EE 86 DD 20 EE 00 00 00 00 05 00 00
```

1.5 Do rejestru R0 dodać wartość z pamięci pod adresem 13.

Opis	OPCODE	MNEMONIC	BYTE CODE
do R1 wpisz wartość z adresu 13	DD	LDR 1, 0xD	11011101
do R0 zapisz wynik R0+R1	20	ADD R0, R1	0010 0000

Rezultat:

```
                DD
c=3 r0=6 r1=0 pc=7 A=D R=DD W=00 M: 85 ED 80 90 EE 86 DD 20 EE 00 00 00 00 05 00 00
c=0 r0=6 r1=5 pc=7 A=D R=05 W=00 M: 85 ED 80 90 EE 86 DD 20 EE 00 00 00 00 05 00 00
c=1 r0=6 r1=5 pc=7 A=7 R=05 W=00 M: 85 ED 80 90 EE 86 DD 20 EE 00 00 00 00 05 00 00
c=2 r0=6 r1=5 pc=8 A=7 R=20 W=00 M: 85 ED 80 90 EE 86 DD 20 EE 00 00 00 00 05 00 00
c=3 r0=B r1=5 pc=8 A=7 R=20 W=00 M: 85 ED 80 90 EE 86 DD 20 EE 00 00 00 00 05 00 00
           20
```

1.6 Umieścić wartość rejestru R0 w pamięci w komórce o adresie 14

Opis	OPCODE	MNEMONIC	BYTE CODE
zawartość R0 wpisz do adresu 14	EE	STR 0, 0xE	1110 1110

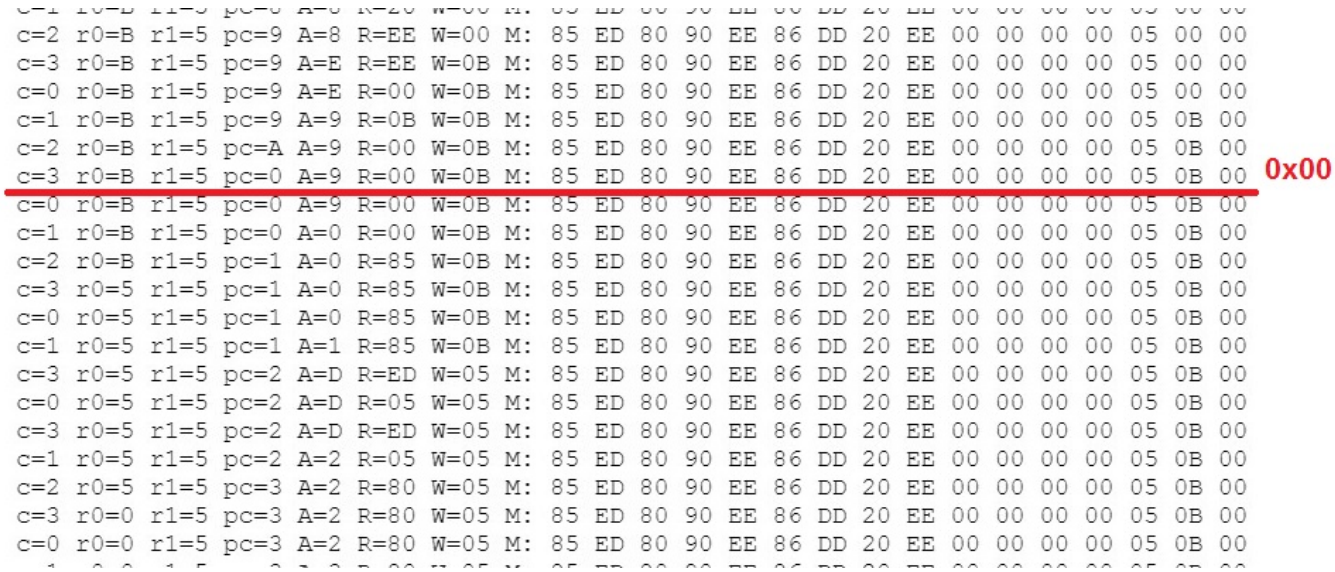
Rezultat:

```
c=0 r0=B r1=5 pc=9 A=E R=00 W=0B M: 85 ED 80 90 EE 86 DD 20 EE 00 00 00 00 05 00 00
c=1 r0=B r1=5 pc=9 A=9 R=0B W=0B M: 85 ED 80 90 EE 86 DD 20 EE 00 00 00 00 05 0B 00
                                                                          EE
```

1.7 Zapętlić wykonywanie programu.

Opis	OPCODE	MNEMONIC	BYTE CODE
skok do adresu nr. 0	0	JMP 0x0	00000000

Rezultat:



2 Utworzenie nowej instrukcji mikroprocesora.

W pliku `cpu.v` w odpowiednim miejscu uzupełniliśmy kod o żądane instrukcje.

2.1 Do rejestru R1 zapisz wynik dodawania wartości z rejestrów Ro i R1.

```
92      case(opcode[6:4])               //analizujemy bity 6, 5 i 4...
93          3'b000:                     //jesli wszystkie sa ustawione na 000...
94          begin
95                                      //wykonujemy skok pod adres dany stala natychmiastowa
96              pc <= opcode[3:0];      //zaladuj do PC wartosc znaleziona w bitach 3:0 kodu instrukcji
97          end
98
99          3'b010:                     //jesli bity 6:4 maja wartosc 010...
100         begin
101             r0 <= r0 + r1;          //wykonujemy dodawanie: add r0, r1 (r0 = r0 + r1)
102         end
103
104         3'b011:                     //jesli bity 6:4 maja wartosc 011...
105         begin
106             r1 <= r0 + r1;          //wykonujemy dodawanie: add r0, r1 (r1 = r0 + r1)
107         end
108
```

2.2 Na rejestrach R0 i R1 zapisz (8bitowy) wynik mnożenia wartości z rejestrów R0 i R1

```
92      case(opcode[6:4])               //analizujemy bity 6, 5 i 4...
93          3'b000:                     //jesli wszystkie sa ustawione na 000...
94          begin
95                                      //wykonujemy skok pod adres dany stala natychmiastowa
96              pc <= opcode[3:0];      //zaladuj do PC wartosc znaleziona w bitach 3:0 kodu instrukcji
97          end
98
99          3'b010:                     //jesli bity 6:4 maja wartosc 010...
100         begin
101             r0 <= r0 + r1;          //wykonujemy dodawanie: add r0, r1 (r0 = r0 + r1)
102         end
103
104         3'b011:                     //jesli bity 6:4 maja wartosc 011...
105         begin
106             r1 <= r0 + r1;          //wykonujemy dodawanie: add r0, r1 (r1 = r0 + r1)
107         end
108
109         3'b111:                      //jesli bity 6:4 maja wartosc 111...
110         begin
111             temp := r0*r1
112             r0 <= r0*r1;             //do r0 wpiszą się 4 najbardziej znaczące bity wyniku
113             r1 <= temp mod 'b1000   // a do r1 4 najmniej znaczące bity wyniku
114         end     //wykonujemy mnożenie: mul r0, r1 (r0:r1 = r0 * r1)
115
116
117
```