

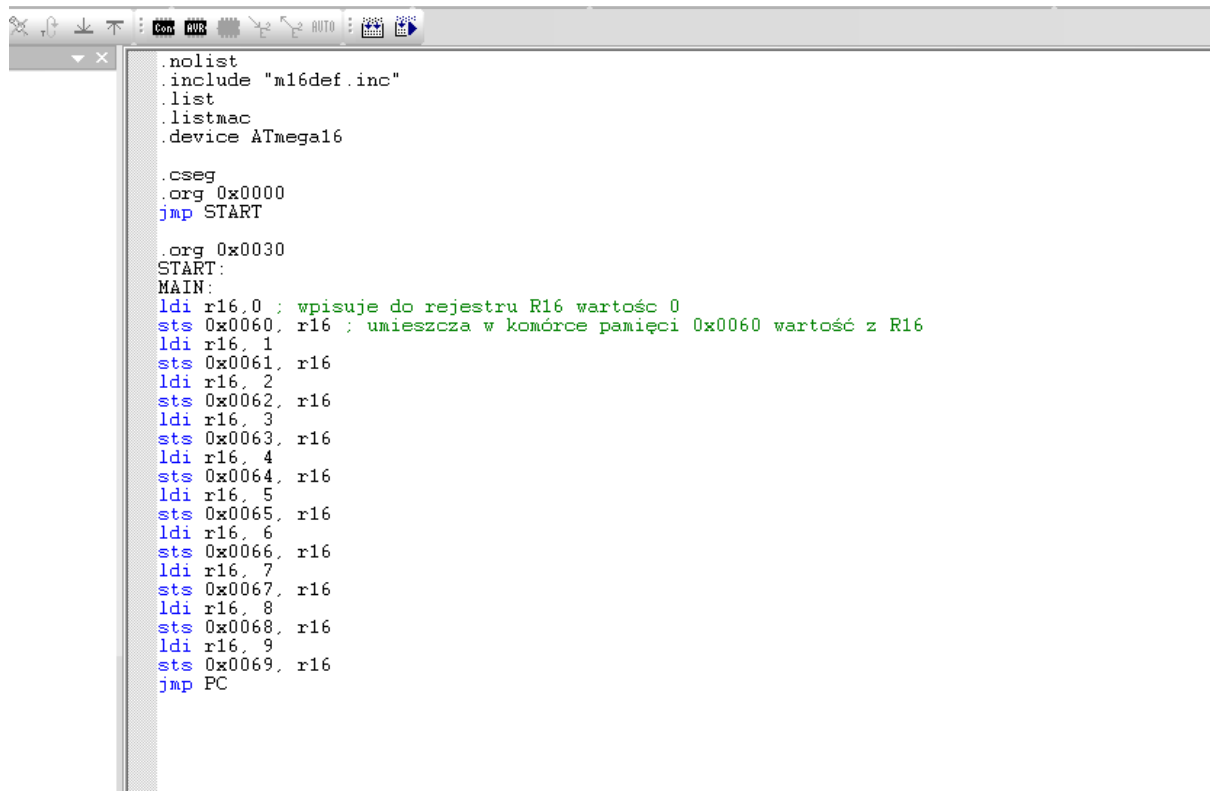
8: Zapis i odczyt pamięci SRAM, rejestry wskaźnikowe

Imię nazwisko, indeks :
Marcel Garczyk, 147935
Paweł Hatka 147952,

Data wykonania : 15.12.2022 r.

Grupa : T-2, czwartek 15:10-16:40

Program 1:



```
.nolist
.include "m16def.inc"
.list
.listmac
.device ATmega16

.cseg
.org 0x0000
jmp START

.org 0x0030
START:
MAIN:
ldi r16, 0 ; wpisuje do rejestru R16 wartość 0
sts 0x0060, r16 ; umieszcza w komórce pamięci 0x0060 wartość z R16
ldi r16, 1
sts 0x0061, r16
ldi r16, 2
sts 0x0062, r16
ldi r16, 3
sts 0x0063, r16
ldi r16, 4
sts 0x0064, r16
ldi r16, 5
sts 0x0065, r16
ldi r16, 6
sts 0x0066, r16
ldi r16, 7
sts 0x0067, r16
ldi r16, 8
sts 0x0068, r16
ldi r16, 9
sts 0x0069, r16
jmp PC
```

Za pomocą polecenia sts zapisujemy do odpowiedniej komórki pamięci wartość zawartą w rejestrze R16.

Program 2:

```
.listmac
.device ATmega16

.cseg
.org 0x0000
jmp START

.org 0x0030
START:
ldi R17, high(ramend)
out sph, R17
ldi R17, low(ramend)
out spl, R17
MAIN:
ldi r16, 0 ; wpisuje do rejestru R16 wartość 0
sts 0x0060, r16 ; umieszcza w komórce pamięci 0x0060 wartość z R16
ldi r16, 1
sts 0x0061, r16
ldi r16, 2
sts 0x0062, r16
ldi r16, 3
sts 0x0063, r16
ldi r16, 4
sts 0x0064, r16
ldi r16, 5
sts 0x0065, r16
ldi r16, 6
sts 0x0066, r16
ldi r16, 7
sts 0x0067, r16
ldi r16, 8
sts 0x0068, r16
ldi r16, 9
sts 0x0069, r16

call odczyt

jmp PC
odczyt:
lds r16, 0x0060
lds r16, 0x0061
lds r16, 0x0062
lds r16, 0x0063
lds r16, 0x0064
lds r16, 0x0065
lds r16, 0x0066
lds r16, 0x0067
lds r16, 0x0068
lds r16, 0x0069
```

Jest to rozszerzenie programu 1 o podprogram służący do odczytu danych za pomocą polecenia lds z odpowiednich komórek pamięci i zapisanie ich do rejestru R16, w przypadku powyższego programu, po wywołaniu podprogramu "odczyt" w rejestrze R16 będą kolejno nadpisywane wartości od 0-9.

Program 3:

```
.nolist
.include "m16def.inc"
.list
.listmac
.device ATmega16

.cseg
.org 0x0000
jmp START

.org 0x0030
START:
MAIN:
ldi r26, low(0x0090) ; wskaźnik X
ldi r27, high(0x0090)
ldi r16, 0
st X+, r16
ldi r16, 1
st X+, r16
ldi r16, 2
st X+, r16
ldi r16, 3
st X+, r16
ldi r16, 4
st X+, r16
ldi r16, 5
st X+, r16
ldi r16, 6
st X+, r16
ldi r16, 7
st X+, r16
ldi r16, 8
st X+, r16
ldi r16, 9
st X+, r16
```

Adresacje z Programu 1 jesteśmy w stanie zastąpić za pomocą rejestru wskaźnikowego X, aby go aktywować musimy umieścić w rejestrach R26 i R27 młodszą i starszą część tego adresu. Za pomocą polecenia `st X+,` wpisujemy do odpowiedniej komórki adresu wartość z R16. Następnie następuje zwiększenie wskaźnika X.

Program 3 jesteśmy w stanie zapisać również jako:

```
.nolist
#include "m16def.inc"
.list
.listmac
.device ATmega16

.cseg
.org 0x0000
jmp START

.org 0x0030
START:
MAIN:
ldi r26, low(0x0090) ;wskaźnik X
ldi r27, high(0x0090)
ldi r16,0
ldi r17,10
SRAM_loop:
st X+, r16
inc r16
dec r17
brne SRAM_loop
nop

JMP PC
```

,gdzie poprzez zastosowanie pętli wpisujemy do kolejnych adresów, wartości od 0 - 9.

Program do samodzielnej realizacji:

Program porównujący wartość wpisaną do rejestru z wartością odczytaną, na tej podstawie jeżeli wartości są równe zapalana jest 1 dioda na Porcie B. Program został sprawdzony na rzeczywistym układzie. Program wykorzystuje rejestr wskaźnikowy Z.

```

.nolist
#include "m16def.inc"
.list
.listmac
.device ATmega16

.cseg
.org 0x0000
jmp START

.org 0x0030
START:

MAIN:
ldi r16, 0b11111111
out DDRE, r16

ldi r30, low(0x0060) ;wskaźnik Z
ldi r31, high(0x0060)
ldi r16, 85
ldi r17, 255
ldi r25, 0b01111111
ldi r26, 0b11111110

SRAM_loop:
st Z+, r16
ld R19, Z
cpc R16, R19
brcs PRAWA
out PORTB, R25
call delay_1s
JMP ET
PRAWA:
out PORTB, R26
call delay_1s
ET:
dec r17
brne SRAM_loop
nop

SRAM_loop1:
st Z+, r16
ld R19, Z
inc R19
cpc R16, R19
brcs PRAWA1
out PORTB, R25
call delay_1s
JMP ET1
PRAWA1:
out PORTB, R26
call delay_1s
ET1:
dec r17
brne SRAM_loop1
nop

jmp MAIN
|
DELAY_1s:
NOP
NOP
ldi R22, 2
ldi R23, 75
ldi R24, 188
OP_1s:
dec R24
brne op_1s
dec R23
brne op_1s
dec R22
brne op_1s
ret

```

Zapis imienia i nazwiska w kodzie ASCII:

```
.nolist
.include "m16def.inc"
.list
.listmac
.device ATmega16

.cseg
.org 0x0000
jmp START

.org 0x0030
START:

MAIN:
ldi r30, low(0x0060) ;wskaźnik Z
ldi r31, high(0x0060)

ldi R16,0101 0000 ; P w ascii
st Z+, r16

ldi R16,0100 0001 ; A
st Z+, r16

ldi R16,0101 0111 ; W
st Z+, r16
ldi R16,0100 0101 ;E
st Z+, r16

ldi R16,0100 1100 ;L
st Z+, r16

ldi R16,0010 0000 ; spacja
st Z+, r16

ldi R16,00100 1000 ;H
st Z+, r16

ldi R16,0100 0001 ; A
st Z+, r16

ldi R16,0101 0100 ;T
st Z+, r16

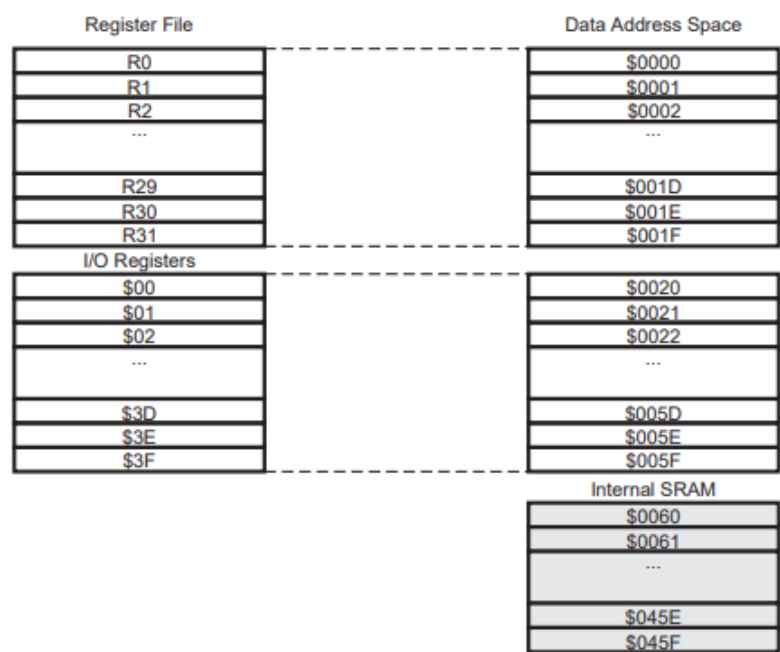
ldi R16,0100 1011 ; K
st Z+, r16

ldi R16,0100 0001 ; A
st Z+, r16
```

Program działa na takiej samej zasadzie jak poprzednie podprogramy.

Mapa pamięci SRAM:

Figure 9. Data Memory Map



EPROM:

Figure 8. Program Memory Map

