1. Zagadnienia

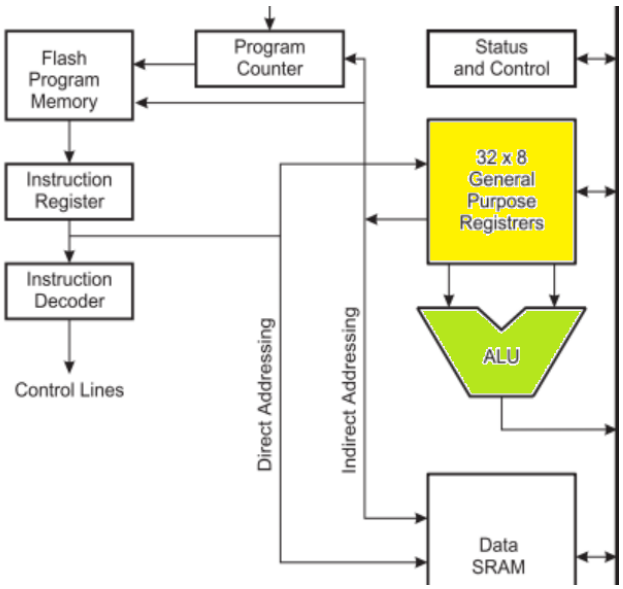
- Rejestry ogólnego przeznaczenia (rejestry robocze)
- Suma kontrolna, kontrola parzystości/nieparzystości
- Obliczanie cyfry kontrolnej numeru PESEL
- Instrukcje operujące na liczbach 16-bitowych

2. Rejestry robocze

ATmega16 zawiera 32 rejestry ogólnego przeznaczenia (*General Purpose Working Registers*). Są one umieszczone na początku pamięci danych od adresu 0x00 do 0x1F. Rejestry podzielone są na dwie grupy r0-r15 i r16-r31. Niektóre rejestry pełnią dodatkowe funkcje (np. w r0 i r1 umieszczany jest wynik instrukcji mnożenia MUL, r26-r31 tworzą tzw. rejestry wskaźnikowe). Rejestry robocze współpracują bezpośrednio z jednostką ALU. W procesorach tego typu rejestry nazywane są też akumulatorami; przykładowo procesor 8080/86 (procesor wykorzystywany w pierwszych komputerach PC) ma tylko1 akumulator i dodatkowo 6 rejestrów uniwersalnych, procesor 8051 ma 1 akumulator nazywany A i rejestr pomocniczy B, procesory z architekturą ARM mają 15 rejestrów roboczych (15 rejestr pełni rolę PC).

7	0	Addr.	
R0		$00	
R1		$01	
R2		$02	
...			
R13		$0D	
R14		$0E	
R15		$0F	
R16		$10	
R17		$11	
...			
R26		$1A	X-register Low Byte
R27		$1B	X-register High Byte
R28		$1C	Y-register Low Byte
R29		$1D	Y-register High Byte
R30		$1E	Z-register Low Byte
R31		$1F	Z-register High Byte

W ATmega16 wszystkie 32 rejestry robocze mogą służyć do wymiany danych z jednostką ALU (mogą więc pełnić rolę akumulatorów). Duża liczba rejestrów ułatwia operowanie danymi i jednocześnie przyspiesza działanie programów.



3. Suma kontrolna, wyznaczenie bitu parzystości (nieparzystości)

Suma kontrolna (*checksum*) to liczba obliczona według wybranego algorytmu, służąca do sprawdzenia poprawności przesyłania danych. Procesor wysyłający dane oblicza ich sumę kontrolną i dołącza ją do pakietu danych. Urządzenie odbierające również oblicza sumę kontrolną (na podstawie odebranych danych) i sprawdza czy obliczona wartość zgadza się z sumą kontrolną przysłaną z pakietem danych. Jeśli nie - to znaczy, że dane podczas transmisji zostały zmienione (uległy przekłamaniu). Szczególnym przypadkiem sumy kontrolnej jest *cyfra kontrolna* – zwykle ostatnia cyfra identyfikatora (np. w numerach PESEL, NIP, REGON). Innym przykładem sumy kontrolnej jest *bit kontroli parzystości* lub *nieparzystości* często stosowany w transmisji szeregowej. Bitem parzystości nazywa się bit kontrolny, który przyjmuje wartość 1, gdy liczba jedynek w przesyłanej wiadomości jest nieparzysta, lub 0 - gdy parzysta. Dołączenie do wiadomości bitu parzystości sprawia, że wiadomość ma zawsze parzystą liczbę jedynek.

bit kontrolny
1

bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0
1	0	0	1	1	0	1	1

Program_01

```
// ~ nagłówek (dyrektywy)
.cseg
.org 0x0000
jmp START  ;skok do etykiety
           ;START (adres 0x0030)
// wektory przerwań
.org 0x0030
START:
;ustawienie stosu
ldi r16, high(RAMEND)       ;0x04
out SPH, r16
ldi r16, low(RAMEND)        ;0x5f
out SPL, r16

MAIN:
ldi r16, 0b10011011
ldi r20, 0       ;wynik sumowania
ldi r21, 9       ;licznik bitów

DALEJ:
dec r21
breq KONIEC

ror r16
brcc DALEJ       ;jest 0
                 ;jest 1
inc r20          ;dodaj 1 do r20
jmp DALEJ

KONIEC:
jmp PC
```

nie zapomnij o wpisaniu dyrektyw nagłówka

bajt do analizy

zmniejsza licznik bitów, jeżeli różny od 0 (Z=0), wykonuje następną instrukcję

zmniejsza licznik bitów, jeżeli 0 (Z=1), wykonuje skok do KONIEC - kończymy analizę

jeżeli C=0 wykonuje skok do DALEJ

jeżeli C=1 wykonuje kolejną instrukcję, (inc r20)

Bajt do analizy został zapisany do rejestru r16. W r20 będzie wynik zliczanych jedynek (na początku wpisano 0), w r21 pomocniczo liczba przesunięć. Program 8 razy przesunie zawartość r16 w prawo (ROR r16), za każdym razem najmłodszy bit wysuwany jest do znacznika C. Znacznik C jest sprawdzany i jeżeli bit jest zerem wykonujemy kolejne przesunięcie (BRCC DALEJ), jeżeli bit jest 1 zwiększamy licznik "jedynek" (INC r20) i wracamy do kolejnego przesunięcia (JMP DALEJ). Za każdym razem zmniejszamy (DEC r21) i sprawdzamy licznik bitów (chcemy przesunąć 8 bitów), jeżeli licznik doliczy do zera kończymy analizę (BREQ KONIEC).

Na podobnej zasadzie wyznacza się *bit nieparzystości*. Bitem nieparzystości nazywa się bit kontrolny, który jeśli jest ustawiony na 1, to oznacza, że liczba jedynek w wiadomości jest parzysta. W ustawieniach transmisji szeregowej możemy wybrać czy przesyłamy bajt bez bitu kontrolnego, czy przesyłamy bit parzystości, czy przesyłamy bit nieparzystości.

4. Obliczanie cyfry kontrolnej numeru PESEL

Numer PESEL jest to 11-cyfrowy stały symbol numeryczny jednoznacznie identyfikujący określoną osobę fizyczną.

Zbudowany jest z następujących elementów:

- zakodowanej daty urodzenia
- liczby porządkowej
- zakodowanej płci
- cyfry kontrolnej

a	b	c	d	e	f	g	h	i	j	k
data urodzenia						liczba porządkowa				
9	7	0	5	2	5	1	2	3	4	?

płeć

cyfra kontrolna

Cyfrę kontrolną można wyznaczyć następująco:

- obliczyć sumę

$$suma = 9 \times a + 7 \times b + 3 \times c + 1 \times d + 9 \times e + 7 \times f + 3 \times g + 1 \times h + 9 \times i + 7 \times j$$

- obliczyć resztę dzielenia *suma* przez 10 (modulo 10)

$$\frac{suma}{10} = wynik + reszta$$

cyfra kontrolna

Program_02

```
// ~ fragment programu

MAIN:
ldi r16, 9 ;a
ldi r17, 7 ;b
ldi r18, 0 ;c
ldi r19, 5 ;d
ldi r20, 2 ;e
ldi r21, 5 ;f
ldi r22, 1 ;g
ldi r23, 2 ;h
ldi r24, 3 ;i
ldi r25, 4 ;j

ldi r26, 3
ldi r27, 7
ldi r28, 9

ldi r30,0
ldi r31,0       ;wynik

mul r16, r28    ;a*9
add r30, r0

mul r17, r27    ;b*7
add r30, r0
adc r31, r1

mul r18, r26    ;c*3
add r30, r0
adc r31, r1

add r30, r19    ;d
adc r31, r1

mul r20, r28    ;e*9
add r30, r0
adc r31, r1

mul r21, r27    ;f*7
add r30, r0
adc r31, r1

mul r22, r26    ;g*3
add r30, r0
adc r31, r1

add r30, r23    ;h
adc r31, r1
```

wpisanie cyfr numeru PESEL do rejestrów roboczych

współczynniki 3, 7, 9 pomocne przy mnożeniu

Wynik obliczenia sumy (może być liczbą 16-bitową, więc przeznaczono dwa rejestry)

Wykonujemy pierwsze mnożenie a*9 i dodajemy do rejestru wyniku

wynik mnożenia nie przekroczy 255

kolejne mnożenie b*7 i dodajemy do rejestru wyniku

tu wynik dodawania może być już liczbą 16-bitową

kolejne mnożenie c*3 i dodajemy do rejestru wyniku

d*1 nie ma potrzeby mnożenia, tylko dodajemy do rejestru wyniku
itd..
e*9
f*7
g*3
h
i*9
j*7

```
mul r24, r28     ;i*9
add r30, r0
adc r31, r1

mul r25, r27     ;j*7
add r30, r0
adc r31, r1

call DZIELENIE_10

jmp PC


// podprogram dzielenia
.org 0x0100
DZIELENIE_10:
ldi r29, 255

LICZ_DALEJ:
inc r29                 ;r29 -> wynik dzielenia
sbiw r31:r30, 10
brpl LICZ_DALEJ
adiw r31:r30, 10        ;r31 -> reszta
ret
```

wywołanie podprogramu DZIELENIE_10

dzielenie przez 10 i obliczenie reszty

W podprogramie dzielenia odejmujemy 10 od liczby którą dzielimy i sprawdzamy, czy nie otrzymaliśmy liczby ujemnej, jeżeli nie wracamy do kolejnego odejmowania - za każdym razem zliczamy ile razy odjęliśmy 10 (INC r29). Jeżeli wynik odejmowania jest ujemny (przekroczenie zakresu liczb 8-bitowych w dół) kończymy odejmowanie (BRPL LICZ_DALEJ), ale musimy dodać odjęte nadmiarowo 10. W ten sposób uzyskujemy resztę. Wynik dzielenia jest liczbą odejmowań zliczoną w rejestrze r29 (ponieważ wykonaliśmy o jedno odjęcie za dużo, należało by na końcu obliczeń odjąć od r29 jeden, można też zacząć zliczanie nie od 0, lecz liczby o jeden mniejszej (czyli 255). Stąd też na początku podprogramu LDI r29, 255.

W podprogramie DZIELENIE_10 wykorzystano możliwość łączenia w pary rejestrów i wykonywania niektórych obliczeń matematycznych na liczbach 16-bitowych (mimo że procesor jest 8-bitowy). Instrukcja SBIW r31:r30, 10 odejmuje od 16-bitowej liczby zapisanej w rejestrach r31 i r30 stałą 10. Podobnie działa instrukcja dodawania ADIW r31:r30, 10. Jak się okazuje cały podprogram dzielenia przez 10 to tylko 5 instrukcji asemblera.

Uruchom Program_02 w pracy krokowej i sprawdź jego działanie. Możesz dla sprawdzenia wprowadzić znany numer PESEL.

Uwaga: w sprawozdaniu pisemnym proszę nie umieszczać swoich danych - proszę wprowadzić fikcyjny numer PESEL

5. Zadania do samodzielnej realizacji

1. Analiza instrukcji ADIW i SBIW

ADIW	Rd,K	Add Immediate To Word	Rd+1:Rd,K
SBIW	Rdl,K	Subtract Immediate from Word	Rdh:Rdl = Rdh:Rdl - K

- Zapoznaj się z działaniem i ograniczeniami stosowania instrukcji ADIW i SBIW (*Help > Assembler Help*).
- Jak sądzisz, dlaczego nie można operować na stałych większych niż 63?

2. Programy

- **Zadanie 1**: W przykładzie Program_02 zastąp podprogram dzielenia przez dziesięć swoim podprogramem (program z poprzednich ćwiczeń).
- **Zadanie 2**: Napisz program zawierający podprogram dzielenia liczby 16-bitowej przez dowolną liczbę z zakresu 1-63. Efekt końcowy to uzyskanie wyniku dzielenia i reszty.
- **Zadanie 3**:Napisz program obliczający cyfrę kontrolną numeru ISBN (13-cyfr). Zastanów się, czy w programie można wykorzystać tylko instrukcje arytmetyczne działające na liczbach 8-bitowych? W programie możesz wykonać najpierw sumowanie cyfr o odpowiednich wagach a potem mnożenia.
- **Zadanie 4**: Narysuj algorytmy do w/w programów.
- **Zadanie 5**: Zadanie podane przez prowadzącego.

3. Zadania dodatkowe (dla chętnych)

- **Zadanie A**: Napisz program obliczający cyfrą kontrolną numeru REGON.
- **Zadanie B**: Napisz program obliczający cyfrą kontrolną numeru kodów kreskowych EAN13. Czy konieczne są obliczenia na liczbach 16-bitowych?
- **Zadanie C**: Zadanie podane przez prowadzącego.

6. Sprawozdanie

- W sprawozdaniu należy umieścić algorytmy oraz kody programów z odpowiednim wyjaśnieniem działania zastosowanych dyrektyw i instrukcji.
- Na podstawie noty katalogowej ATmega16 opisać przeznaczenie i alternatywne funkcje rejestrów roboczych. Pod jakim adresem i w której pamięci umieszczone są te rejestry? (strona 11).