

Maciej Byczko Bartosz Matysiak	Prowadzący: dr inż. Jacek Mazurkiewicz	Numer ćwiczenia 6
Cz 13:15 TN	Temat ćwiczenia: Transmisja portem szeregowym	Ocena:
Grupa: B	Data wykonania: 6 Maja 2021	

## 1 Polecenie

We wszystkich programach prezentowanych na zajęciach występowało "zjawisko" przekształcania danych przychodzących z terminala na komputerze tak by reprezentowały rzeczywiście kody ASCII znaków. W ten sposób udało się rozwiązać problem, a w zasadzie zaadaptować się do nieznanej przyczyny, stałego "zniekształcenia - przekłamania" danych odbieranych portem szeregowym. Zadanie polega na analizie tego fragmentu kodu, w którym owe przekształcenie danych jest realizowane i opisanie jakie działania są w nim podejmowane, na czym polega to "przekłamanie" danych i - ewentualnie - zaproponowanie innego - może lepszego / sprawniejszego - sposobu zrealizowania tej koniecznej korekty.

## 2 Analiza oryginalnego kodu

```

1 MOV R7, A      ; (1) zamiana odczytanego dziwoląga na ASCII
2 ANL 07H, #0FH  ; (2) // adres 07H to inaczej rejestr R7
3 CLR C          ; wyczyszczenie C aby przypadkiem nie przeszkodziło
4 RRC A          ; (3)
5 ANL A, #0F0H   ; (4)
6 ORL A, R7      ; (5)

```

W kolejnych instrukcjach zawartości rejestrów zmieniają się następująco:

(symbole  $A_7 - A_0$  oznaczają wartości bitów stanowiących początkową zawartość akumulatora, od najstarszego do najmłodszego)

$$\begin{array}{lcl}
 (1) & R_7 | A & = & A_7 & A_6 & A_5 & A_4 & A_3 & A_2 & A_1 & A_0 \\
 (2) & R_7 & = & 0 & 0 & 0 & 0 & A_3 & A_2 & A_1 & A_0 \\
 (3) & A & = & 0 & A_7 & A_6 & A_5 & A_4 & A_3 & A_2 & A_1 \\
 (4) & A & = & 0 & A_7 & A_6 & A_5 & 0 & 0 & 0 & 0 \\
 (5) & R_7 & = & 0 & A_7 & A_6 & A_5 & A_3 & A_2 & A_1 & A_0
 \end{array}$$

W wyniku podanej konwersji, młodszy półbajt akumulatora pozostał bez zmian, natomiast w starszym półbajcie trzy najstarsze bity uległy przesunięciu na młodszą pozycję. Na pozycji najstarszej pojawiło się 0, a wartość  $A_4$  uległa destrukcji.

### 3 Alternatywny kod

Kod można uprościć o jedną instrukcję i pozbyć się tym samym wykorzystania bitu carry, zamieniając kolejność instrukcji:

```

1 MOV R7, A      ; (1) zamiana odczytanego dziwolaga na ASCII
2 ANL 07H, #0FH  ; (2) // adres 07H to inaczej rejestr R7
3 ANL A, #0E0H   ; (4)
4 RR A          ; (3)
5 ORL A, R7      ; (5)

```

$$\begin{array}{rcll}
 (1) & R_7 | A & = & A_7 \ A_6 \ A_5 \ A_4 \ A_3 \ A_2 \ A_1 \ A_0 \\
 (2) & R_7 & = & 0 \ 0 \ 0 \ 0 \ A_3 \ A_2 \ A_1 \ A_0 \\
 (3) & A & = & A_7 \ A_6 \ A_5 \ 0 \ 0 \ 0 \ 0 \ 0 \\
 (4) & A & = & 0 \ A_7 \ A_6 \ A_5 \ 0 \ 0 \ 0 \ 0 \\
 (5) & R_7 & = & 0 \ A_7 \ A_6 \ A_5 \ A_3 \ A_2 \ A_1 \ A_0
 \end{array}$$

Kod ASCII wykorzystuje siedem młodszych bitów bajtu. Wszystkie bajty wykorzystane do reprezentacji mają na najstarszej pozycji bit o wartości 0. Kody powyżej realizują wyzerowanie najstarszego bitu, a także przesunięcie bitowe starszej części.

O ile przesunięcie bitowe można zaniedbać, najstarszy bit akumulatora można wyzerować jedną instrukcją:

```

1 CLR ACC.7

```