DCV č. 1

- upravte příklad č. 2 tak, že hodnoty do pole2 budou kopírovány od konce, tedy

  `pole2[N-i] = pole1[i] + pole1[2], i = 0,1, ... N`

```
            .include    "m32def.inc"
// Konstanty
.equ        N           = 10
.equ        PRVNI       = 22

// Přejmenování registrů
.def        count       = r16
.def        temp        = r17
.def        pole1_2     = r18

// Proměnné
            .DSEG
pole1:      .BYTE       N
pole2:      .BYTE       N

            .CSEG
            // Pointer na pole1 do X
            ldi         XL, LOW(pole1)
            ldi         XH, HIGH(pole1)
            // Inicializace řídící proměnné cyklu
            ldi         count, N
            // Inicializace proměnné pro plnění pole
            ldi         temp, PRVNI
// Cyklus pro naplnění pole1 PRVNI, PRVNI+1, ...
cyklus:     st          X+, temp    // pole1[i] = temp
            // temp se o 1 zvýší -> hodnoty v pole1 budou o 1 vyšší
            inc         temp
            // count se o 1 sníží -> změna řídící proměnné
            dec         count
            // Pokud se count != 0 skok na cyklus
            brne        cyklus

            // Inicializace řídící proměnné cyklu
            ldi         count, N
            // Ukazatel na pole1 do Y
            ldi         YL, LOW(pole1)
            ldi         YH, HIGH(pole1)
            // První adresa za posledním prvkem v pole2 do X
            ldi         XL, LOW(pole2+N)
            ldi         XH, HIGH(pole2+N)
            // Pomocí adresování s offsetem uložíme do pole1_2 = Y[2] (= pole1[2])
            ldd         pole1_2, Y+2

// Cyklus pro kopie do pole2
kopiePole2:
            ld          temp, Y+                // temp = pole1[i]
            add         temp, pole1_2           // temp += pole1[2]
            st          -X, temp                // pole2[N-i-1] = temp -> pole2[N-i-1] = pole[i]
+ pole1[2]
            dec         count                   // count se o 1 sníží
            brne        kopiePole2              // Pokud se count != 0 skok na kopiePole2

end:        rjmp        end                     // Nekonečná smyčka
```

DCV č. 3

- Do předchozího programu doplňte výpočet součtu všech prvků obou polí dohromady. Výsledek uložte do zvláštní proměnné.

```
            .include   "m32def.inc"

// Konstanty
.equ        N           = 10
.equ        PRVNI       = 3

// Přejmenování registrů
.def        count       = r16
.def        temp        = r17
.def        pole1_2     = r18
.def        rSoucet     = r19

// Proměnné
            .DSEG
pole1:      .BYTE       N
pole2:      .BYTE       N
soucet:     .BYTE       1

            .CSEG
            // Pointer na pole1 do X
            ldi         XL, LOW(pole1)
            ldi         XH, HIGH(pole1)
            // Inicializace řídící proměnné cyklu
            ldi         count, N
            // Inicializace proměnné pro plnění pole
            ldi         temp, PRVNI
            ldi         rSoucet, 0            // Inicializace soucet
// Cyklus pro naplnění pole1 PRVNI, PRVNI+1, ....
cyklus:
            st          X+, temp              // pole1[i] = temp
            add         rSoucet, temp         // soucet += pole1[i]
            // temp se o 1 zvýší -> hodnoty v pole1 budou o 1 vyšší
            inc         temp
            // count se o 1 sníží -> změna řídící proměnné
            dec         count
            // Pokud se count != 0 skok na cyklus
            brne        cyklus

            // Inicializace řídící proměnné cyklu
            ldi         count, N
            // Ukazatel na pole1 do Y
            ldi         YL, LOW(pole1)
            ldi         YH, HIGH(pole1)
            // První adresa za posledním prvkem v pole2 do X
            ldi         XL, LOW(pole2+N)
            ldi         XH, HIGH(pole2+N)
            // Pomocí adresování s offsetem uložíme do pole1_2 = Y[2] (= pole1[2])
            ldd         pole1_2, Y+2

// Cyklus pro kopie do pole2
kopiePole2:
            ld          temp, Y+   // temp = pole1[i]
```

```
            add         temp, pole1_2         // temp += pole1[2]
            add         rSoucet, temp         // soucet += pole2[N-i-1]
            st          -X, temp              // pole2[N-i-1] = temp -> pole2[N-i-1] = pole[i]
+ pole1[2]
            dec         count                 // count se o 1 sníží
            brne        kopiePole2            // Pokud se count != 0 skok na kopiePole2

            sts         soucet, rSoucet       // Uložení výsledného součtu do proměnné soucet

end:        rjmp        end         // Nekonečná smyčka
```

Příklad č. 3

Upravte kód v předchozím příkladu tak, aby meze intervalu byly `prom` ϵ $\langle A; B \rangle$.

```
            .include    "m32def.inc"

// Registry
.def        rTemp       = r16
.def        rTemp1      = r17

// Konstanty
.equ        A           = 5
.equ        B           = 10

// Proměnné
            .DSEG
prom:       .byte       1
error:      .byte       1

// Vlastní program
            .CSEG
            .ORG        0x0000
            lds         rTemp, prom           // rTemp = prom

            cpi         rTemp, A              // prom COMPARE A
            brlo        PrictiB               // if (prom < A) -> PrictiB
TestNaB:
            cpi         rTemp, B              // prom COMPARE B
            breq        PrictiA               // if (prom == B) -> PrictiA
            brcc        PrictiB               // if (prom > B) -> PrictiB
PrictiA:
            ldi         rTemp1, A             // rTemp1 = A
            add         rTemp, rTemp1         // rTemp += A (prom += A)
            rjmp        TestC                 // -> TestC
PrictiB: // (else)
            ldi         rTemp1, B             // rTemp1 = B
            add         rTemp, rTemp1         // rTemp += B (prom += B)
TestC:      clr         rTemp1                // error = 0
            brcc        konec                 // if (C==0) -> konec
            ldi         rTemp1, 1             // if (C==1) { error = 1; }
konec:      sts         error, rTemp1         // error = rTemp1
end:        rjmp        end                   // nekonečná smyčka
```