Příklad č. 1

Napište program, který obsah dva registrů chápaných jako čísla se znaménkem (dvojkový doplněk) sečte a vynásobí. Násobení proveďte dvěma způsoby – jednak pomocí příslušené instrukce násobení se znaménkem, a také jiným způsobem za předpokladu, že procesor nedisponuje instrukcí násobení se znaménkem, ale pouze bez znaménka. V tomto případě ponechte výsledek v přímém kódu.

U sčítání demonstrujte vhodnou volbou hodnot argumentů:

přetečení ALU (ale ne přetečení dvojkového doplňku)
přetečení dvojkového doplňku

Může k přetečení dvojkového doplňku dojít při násobení?

```
            .include   "m32def.inc"

// Registry
.def        rOp1       = r16
.def        rOp2       = r17

.def        rZn1       = r18
.def        rZn2       = r19

// Konstanty

//Proměnné
            .DSEG

// Vlastní program
            .CSEG
            .ORG       0x0000
            // nahrání záporného čísla
            ldi        rOp1, -10
            // Převod A -> D(A)
            ldi        rOp2, 76
            neg        rOp2

            // Součet
            add        rOp2, rOp1

            ldi        rOp2, 2

            // Násobení - přes MULS
            muls       rOp1, rOp2

            // Násobení - přes MUL
            mov        rZn1, rOp1
            mov        rZn2, rOp2          // kopie
            andi       rZn1, 0x80          // máme uložené znaménko rOp1
            andi       rZn2, 0x80          // máme uložené znaménko rOp2

            // Převod na absolutní hodnotu
            tst        rOp1                // nastaví příznaky v SREG
            brpl       skok1
            neg        rOp1
skok1:      tst        rOp2
```

```
            brpl        skok2
            neg         rOp2
skok2:      mul         rOp2, rOp1
            // Znaménko výsledku
            eor         rZn1, rZn2
            or          r1, rZn1    // Převod abs. součinu na přímý kód

end:        rjmp        end         // Nekonečná smyčka
```

Příklad č. 2

Deklarujte dvě proměnné: `prom` a `error` – obě 1 B čísla bez znaménka. Realizujte v JSA ekvivalent následujícího pseudokódu:

```
error = 0;
if (prom ϵ (A; B))
      prom += A;
else
      prom += B;
if (přetečení ALU)
      error = 1;
```

```
            .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                // rTemp COMPARE A
            breq        PrictiB                 // if (rTemp == A) -> PrictiB
            brlo        PrictiB                 // if (rTemp < A) -> PrictiB
TestNaB:
            cpi         rTemp, B                // rTemp COMPARE B
            brsh        PrictiB                 // if (rTemp >= B) -> PrictiB
PrictiA:
            ldi         rTemp1, A               // rTemp1 = A
            add         rTemp, rTemp1           // rTemp += rTemp1 (prom += A)
            rjmp        TestC                   // -> TestC
PrictiB:
            ldi         rTemp1, B               // rTemp1 = B
            add         rTemp, rTemp1           // tTemp += rTemp1 (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
```