# BI-SAP

## 4. proseminář

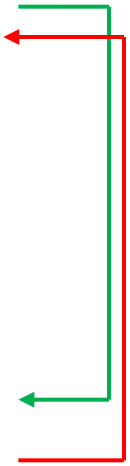
### Instrukční soubor AVR

Podprogramy

Miroslav Skrbek, Martin Novotný

# Volání podprogramu CALL, RET

```
        ldi     R16,0x42
        ldi     R17,0x69
        call    secti
        ldi     R17,0x12
        call    secti
        ldi     R17,0x34
        call    secti
konec:  jmp     konec

secti:  add     R16,R17
        ret
```

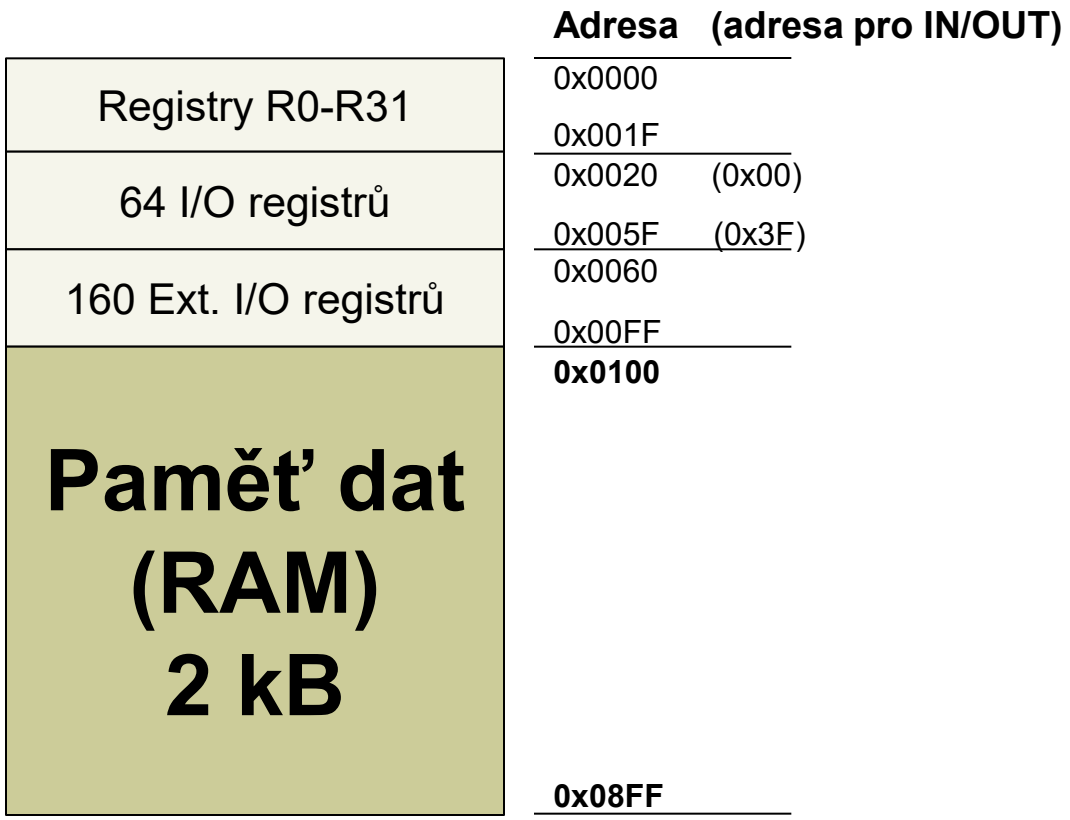


# Volání podprogramu CALL, RET

VolaniPodprogramu.LSS

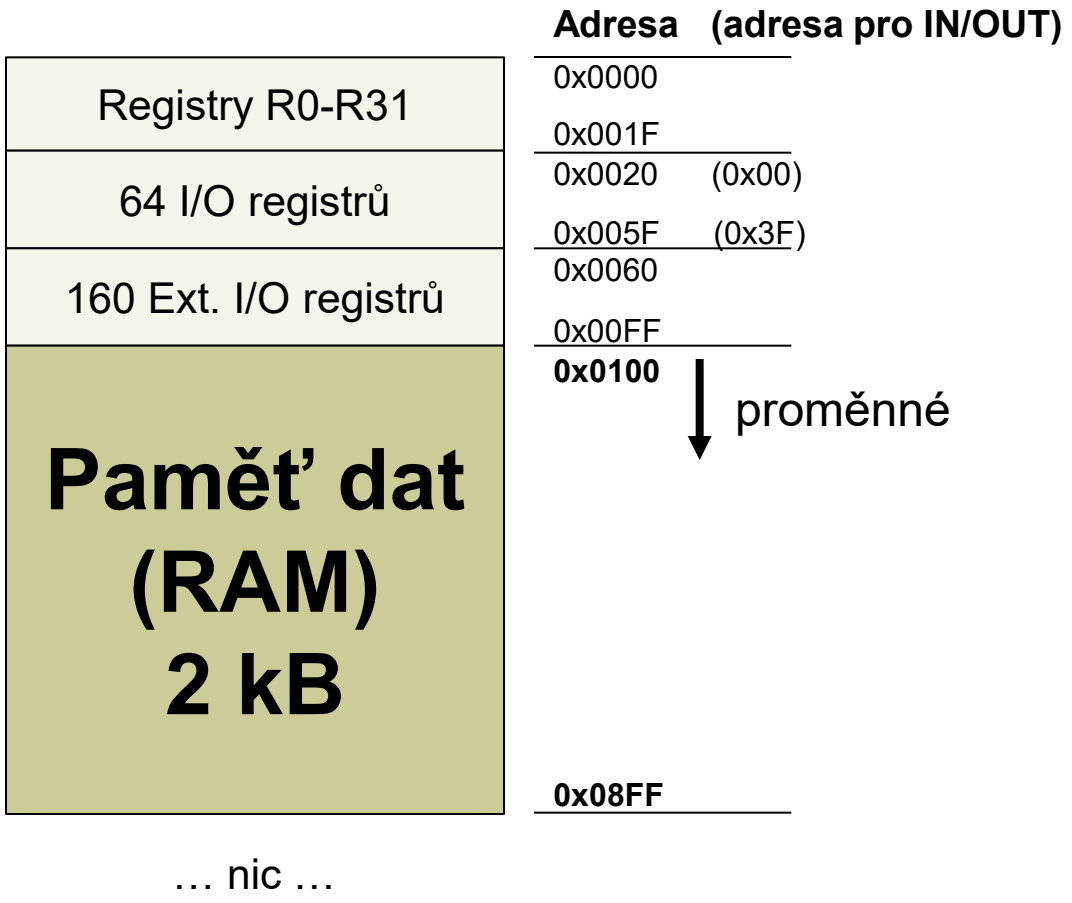
```
000000 e402                ldi    R16,0x42
000001 e619                ldi    R17,0x69
000002 940e 000c           call   secti
000004 e112                ldi    R17,0x12
000005 940e 000c           call   secti
000007 e314                ldi    R17,0x34
000008 940e 000c           call   secti
00000a 940c 000a    konec: jmp    konec

00000c 0f01         secti: add    R16,R17
00000d 9508                ret
```

# Datový prostor a paměť dat

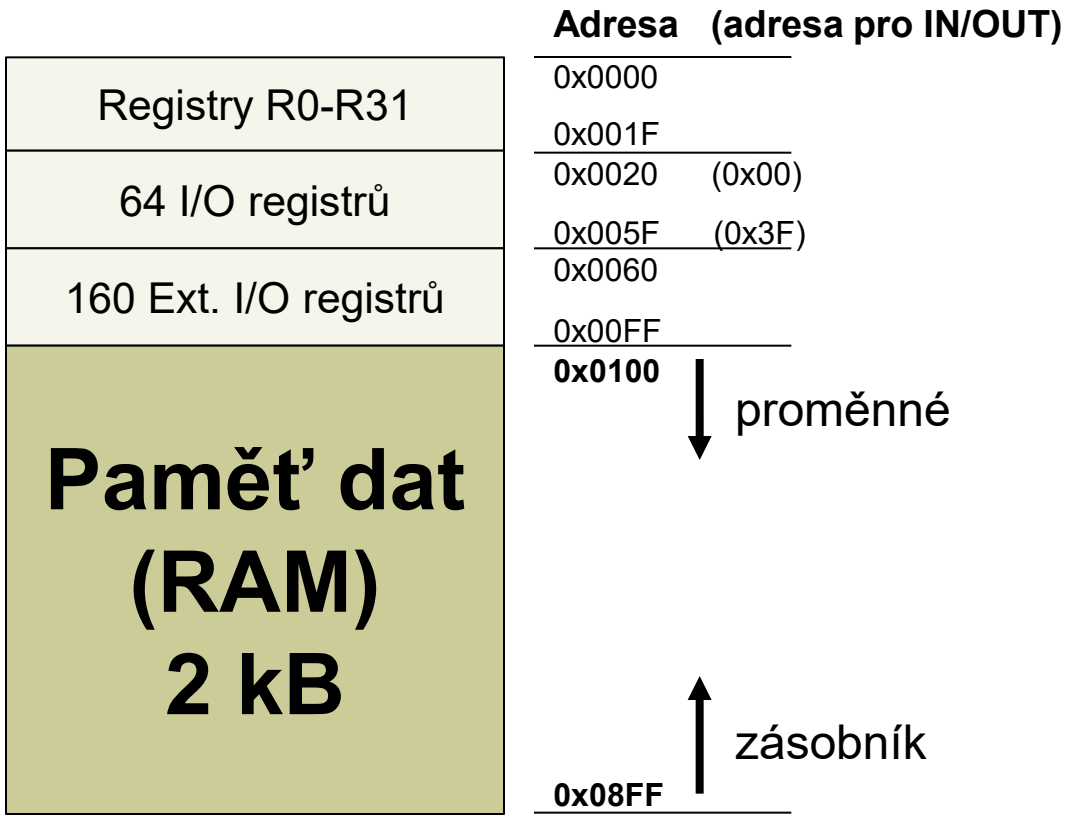



… nic …

# Datový prostor a paměť dat

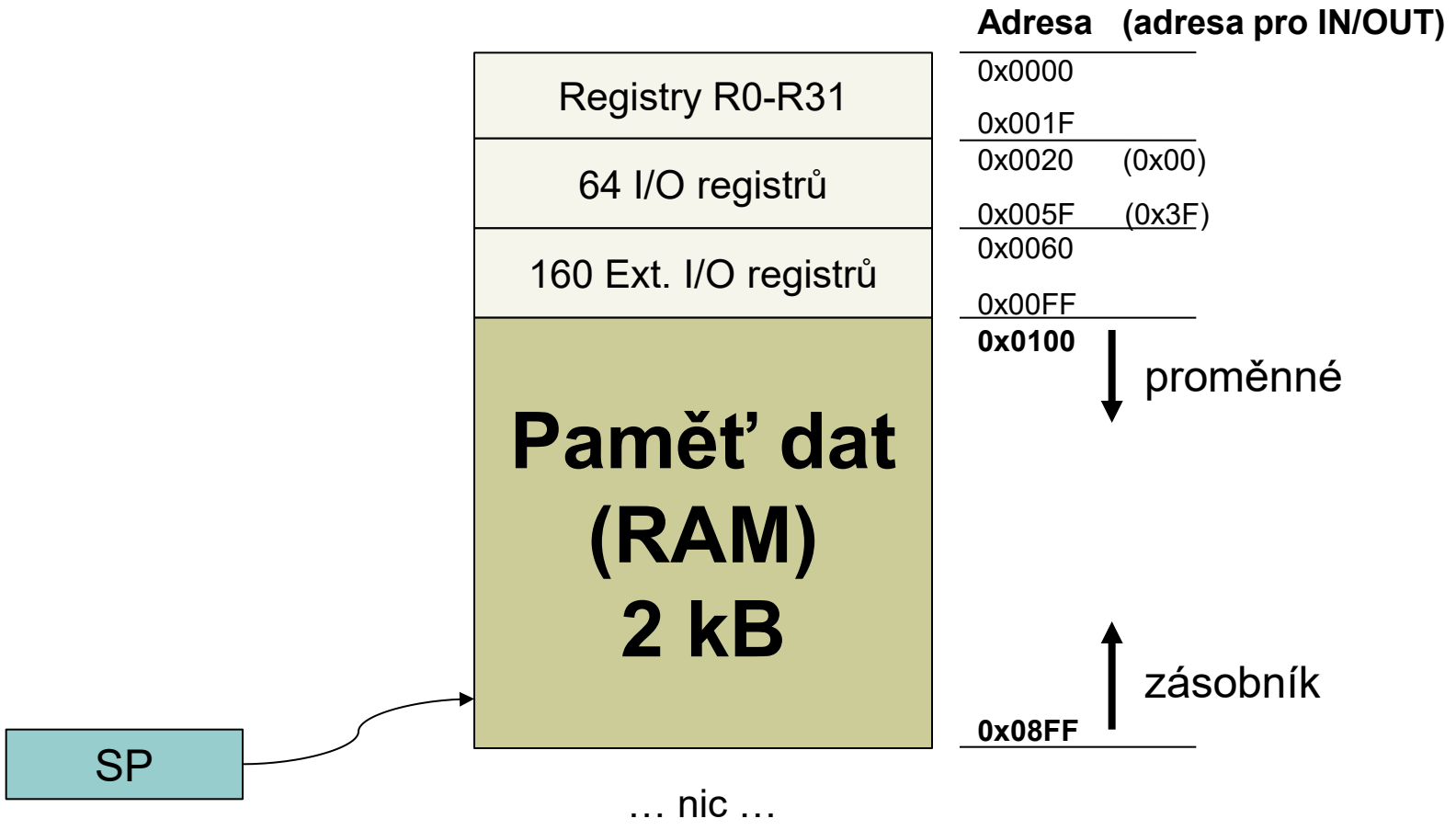


# Datový prostor a paměť dat

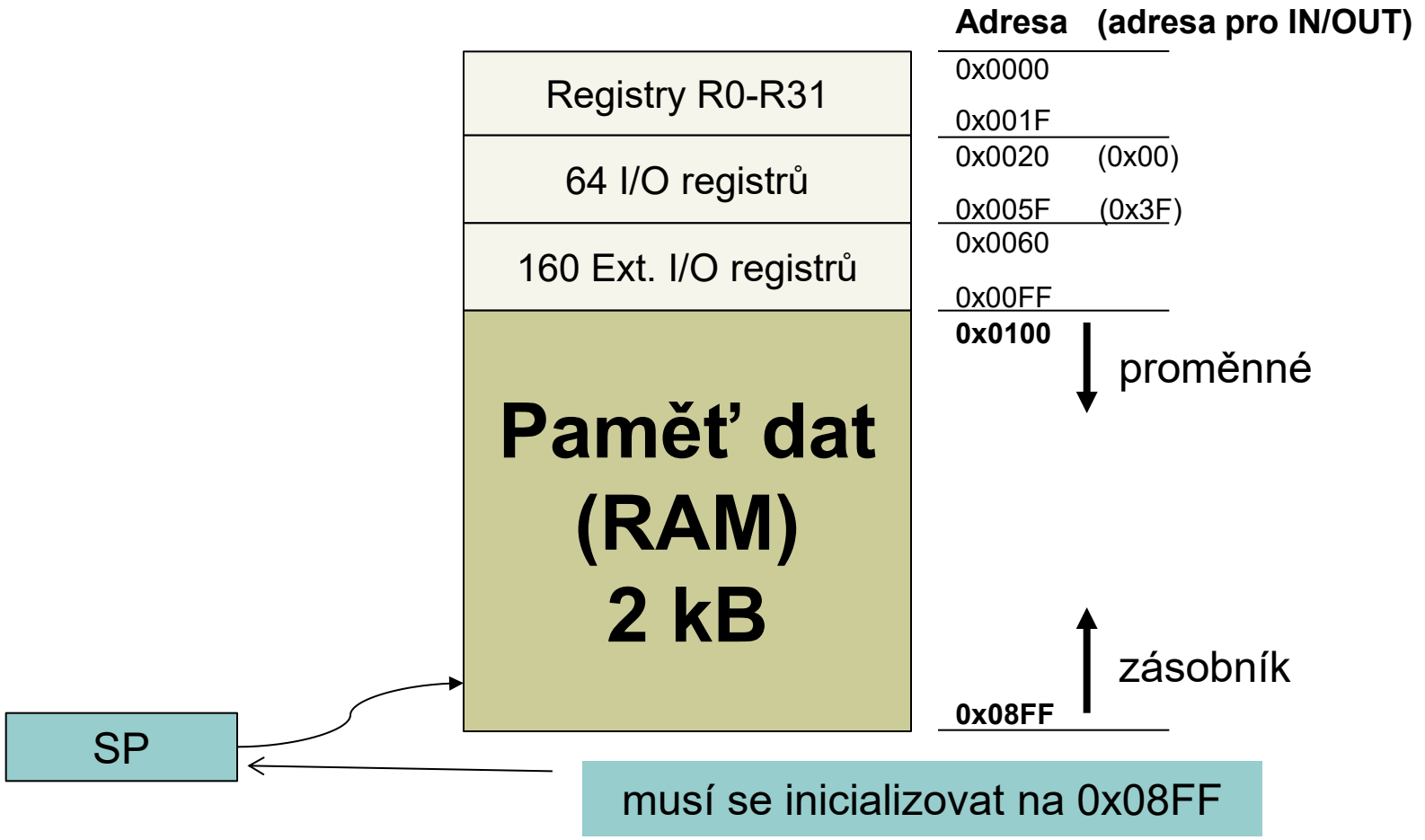



… nic …

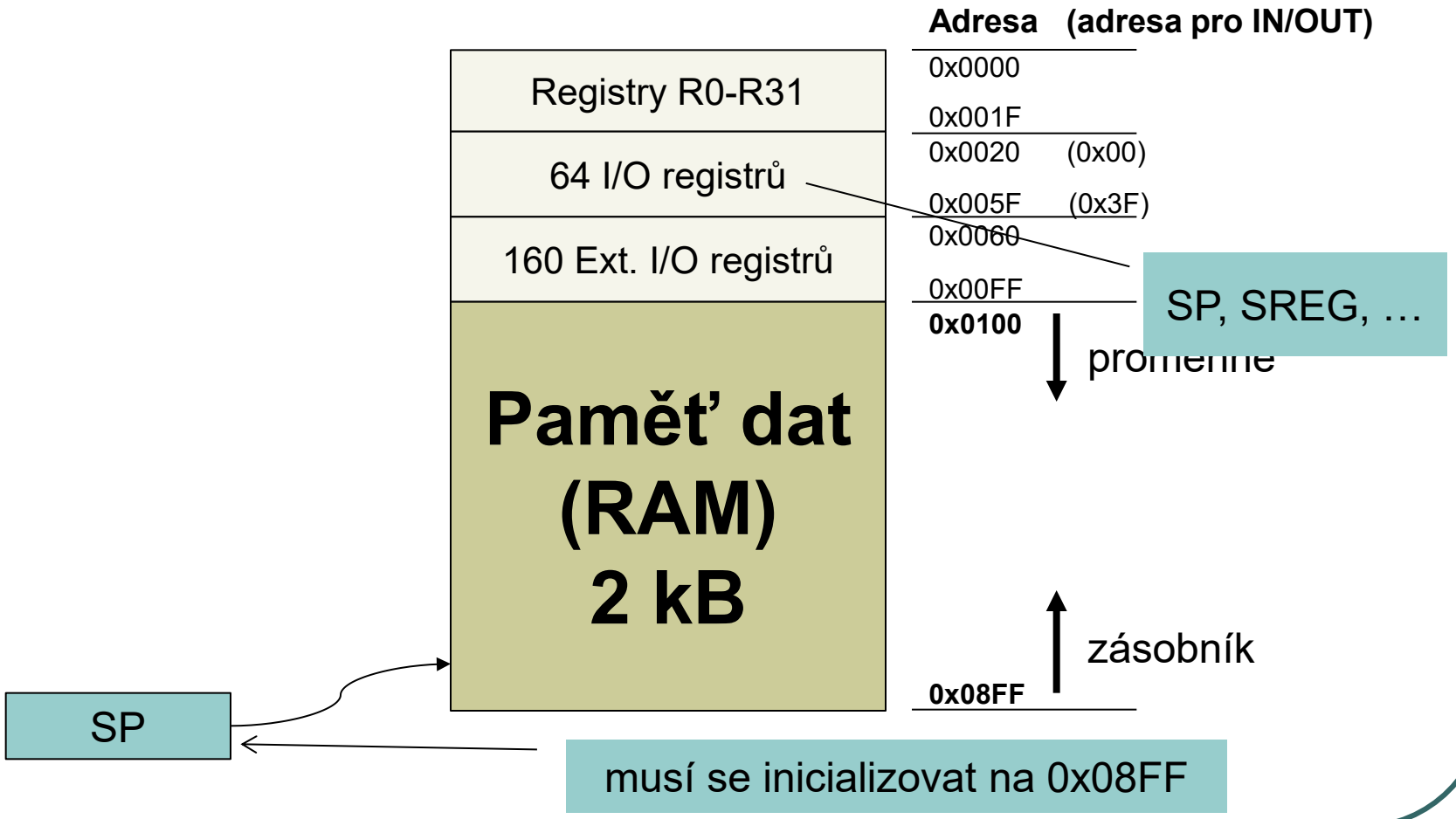
# Datový prostor a paměť dat



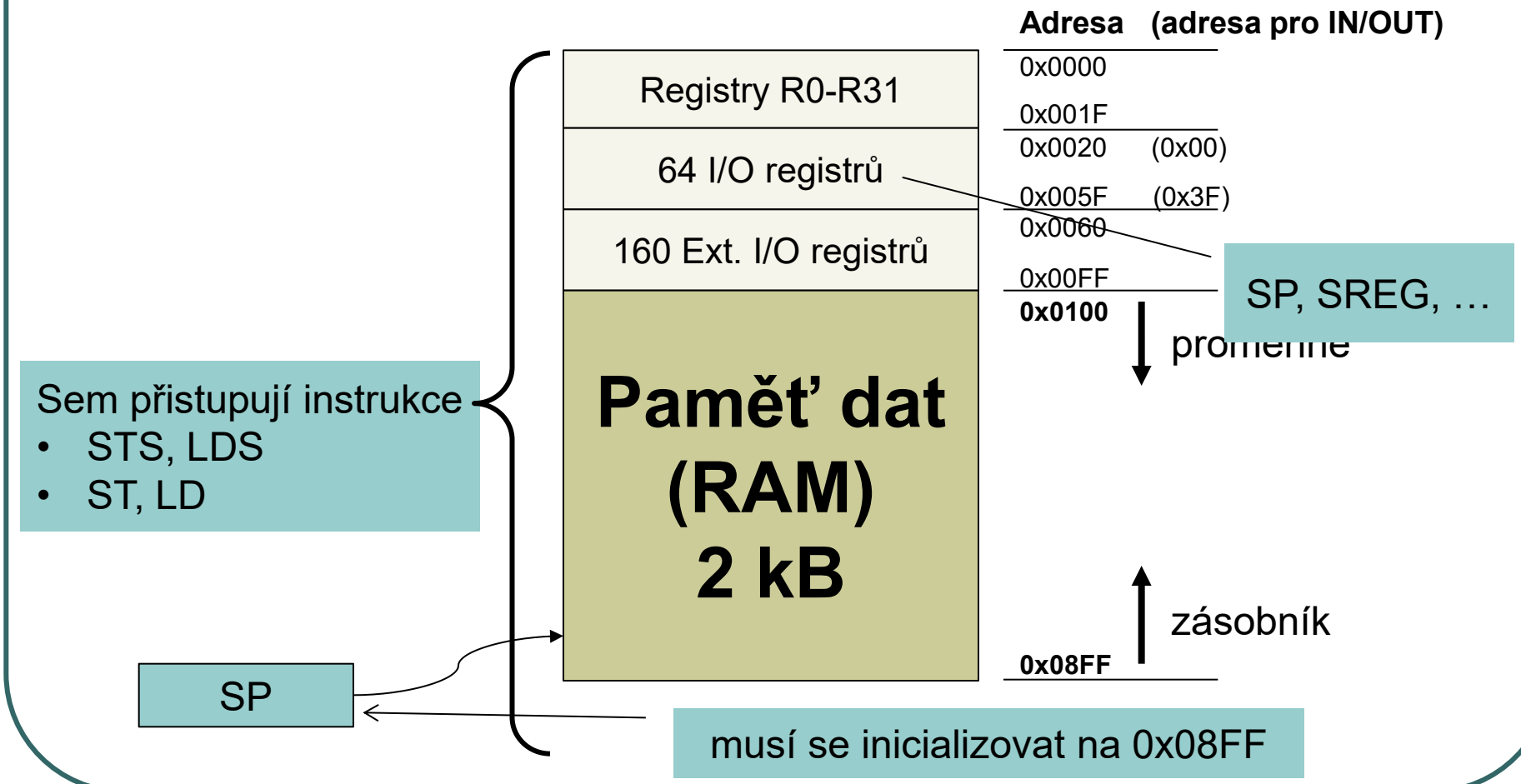
# Datový prostor a paměť dat



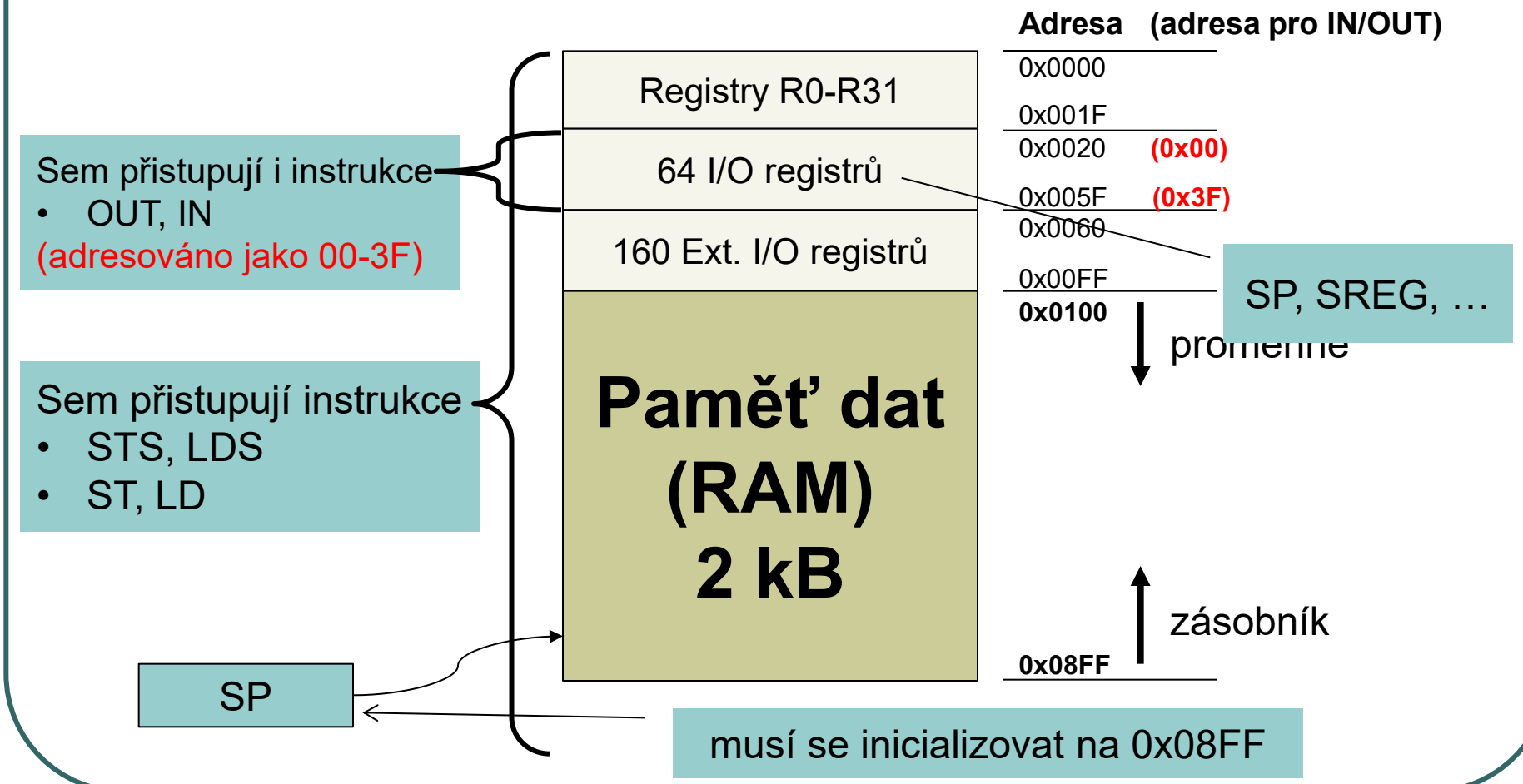
# Datový prostor a paměť dat



# Datový prostor a paměť dat



# Datový prostor a paměť dat



# Inicializace zásobníku na začátku každého programu

```
ldi     r16, 0xFF
out     SPL, r16
ldi     r16, 0x08
out     SPH, r16
```

Zápis hodnoty 08FFh do SP

V assembleru jsou adresy dolní a horní poloviny SP registru definovány takto:

```
.EQU  SPL = 0000003d        ;dolní polovina SP
.EQU  SPH = 0000003e        ;horní polovina SP
```

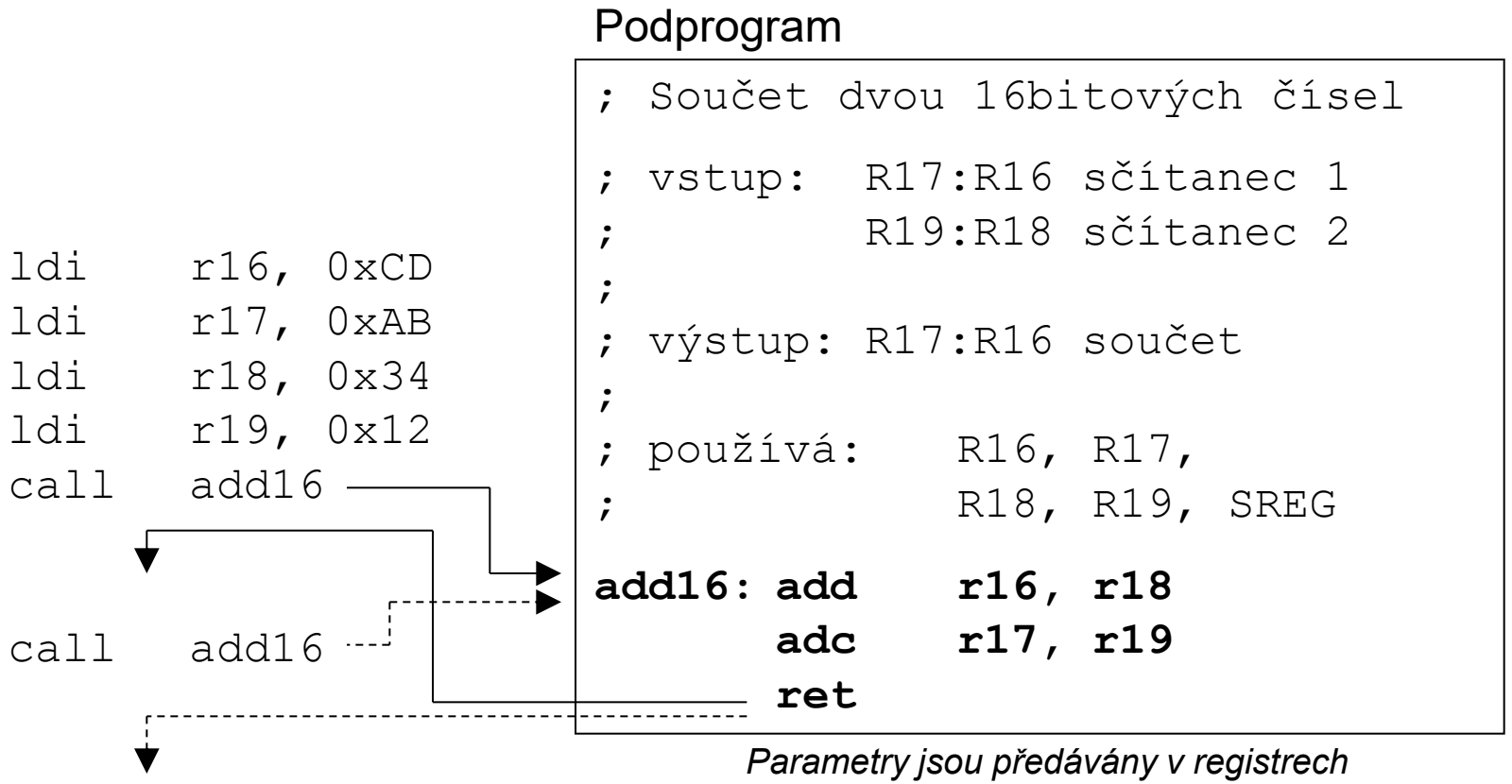
Zásobník inicializujeme na dostatečně vysokou adresu, protože roste směrem k nižším adresám. Maximální počáteční hodnota SP pro ATmega328P je 0x08FF.

# Použití zásobníku

- Ukládání návratových adres na zásobník při volání podprogramu
- Uložení návratové adresy při vyvolaném přerušení
- Předávání parametrů podprogramům
- Lokální proměnné podprogramů
- Dočasné uložení registrů pro zachování transparentnosti kódu. Typicky v obsluze přerušení.

# Úloha: Sečtěte ABCDh + 1234h + 1234h
# Výsledek bude v R17:R16

# Úloha: Sečtěte ABCDh + 1234h + 1234h
# Výsledek bude v R17:R16




*Poznámka: nezapomeňte přidat na začátek programu inicializaci zásobníku.*

# Úloha: napište podprogram, který zjistí délku řetězce uloženého v paměti dat

```
; Podprogram strlen
; vstup: Z (R31:R30) - adresa počátku řetězce
;
; výstup: R0 – délka řetězce
;
; používá:     R31, R30, … doplňte dle skutečnosti …
;
strlen:
        … zde doplňte instrukce …
        ret
```

*Poznámka: součástí řešení je i příklad volání podprogramu.*

Poznámka: abyste mohli příklad vyzkoušet, použijte předchozí příklad pro kopírování řetězce z paměti programu do paměti dat. Testovací řetězec pak definujte takto: mystr: .db „muj retezec“, 0

# Řešení

```
…
ldi     r30, low(str)
ldi     r31, high(str)
call    strlen
…
```

Poznámka: předpokládá se, že řetězec je již předem uložen v paměti dat.

```
; Podprogram strlen

; vstup: Z (R31:R30) –
;        adresa počátku řetězce
;
; výstup: R0 – délka řetězce
;
; používá:      R31, R30, R0, R1
;
strlen:         clr     r0
cycle:          ld      r1, Z+
                cpi     r1,0
                breq    return
                inc     r0
                jmp     cycle
return:         ret
```

*Poznámka: nezapomeňte přidat na začátek programu inicializaci zásobníku.*

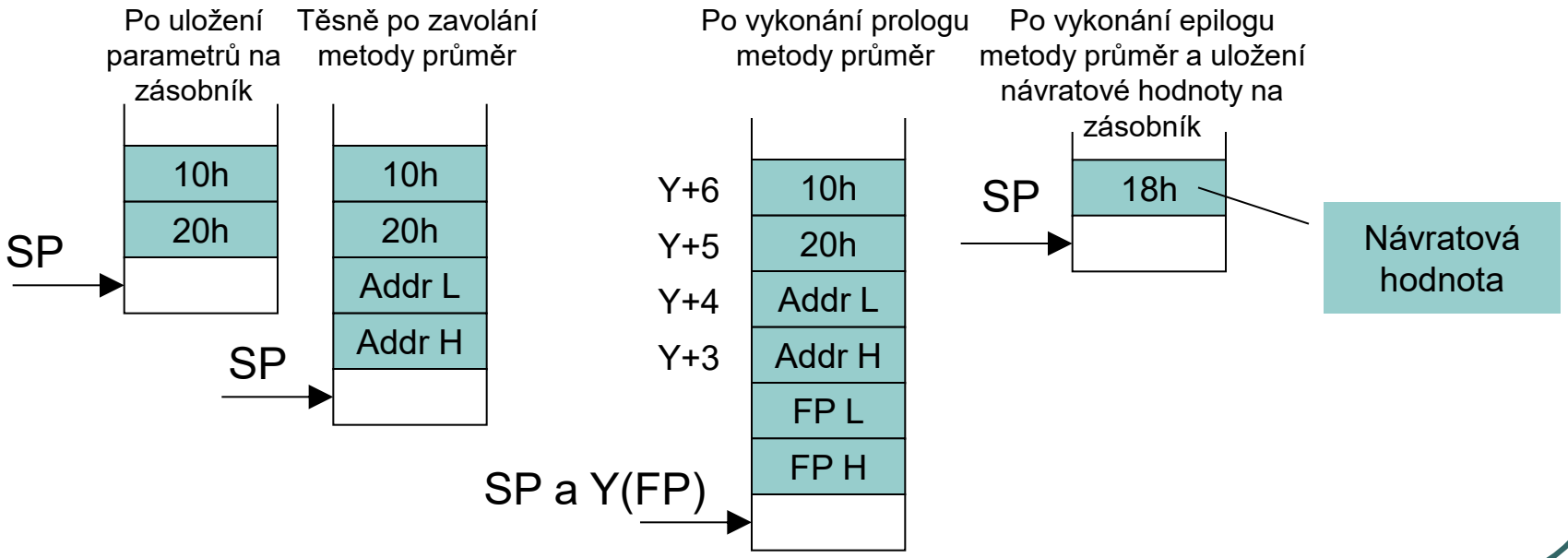
# Předávání parametrů přes zásobník

V jazyce C

```
p = prumer(0x10, 0x20);
```

V jazyce C

```
int prumer(byte a, byte b)
{  return (a + b) >> 1; }
```



# Příklad: podprogram průměr

Volání podprogramu

```
…
ldi     r16, 50   ; parametr a
push    r16
ldi     r16, 150  ; parametr b
push    r16
call    prumer
pop     r16
…
```

```
prumer:  ; entry code (prologue)
         push    r28
         push    r29
         in      r28, SPL
         in      r29, SPH
```

```
; body
ldd     r16, Y+6 ; parametr a
ldd     r17, Y+5 ; parametr b
add     r17, r16
lsr     r17

; exit code  (epilogue)
mov     r26, r28
mov     r27, r29
ldd     r30, Y+4
ldd     r31, Y+3
adiw    r26, 6
pop     r29
pop     r28
out     SPL, r26
out     SPH, r27
push    r17 ; push the return
              value on the stack

ijmp
;
```

*Poznámka: nezapomeňte přidat na začátek programu inicializaci zásobníku.*

# Úloha: rekurzivní výpočet faktoriálu

Matematicky

$$n! = n(n-1)! \quad \text{Jestliže } n > 1$$
$$1 \quad \text{Jestliže } n = 1$$

V Javě

```
int factorial(int n) {
        if (n==1) return 1;
        else return n*factorial(n-1);
}
```

*Program v Javě přepište do asembleru. Modifikujte předchozí program průměr, využijte již vámi napsaný podprogram pro násobení.*

# Faktoriál – prologue a tělo

```
fact:    ; entry code (prologue)
         push     r28
         push     r29
         in       r28, SPL
         in       r29, SPH
        ; body
         clr      r19
         ldd      r18, Y+5 ; par. n
         cpi      r18, 1
         breq     fa_exit
         ;
         ; spočítej (n-1)!
         ;
         dec      r18
         push     r18
         call     fact
         pop      r16
         pop      r17
         ;
         ;
         ;  vynásob n*(n-1)
         ;
         ldd      r18, Y+5 ; par. n
         clr      r19
         clr      r20
         ldi      r21, 9
         clc
fa_cycle: ror     r20
         ror      r19
         ror      r18
         dec      r21
         breq     fa_exit
         brcc     fa_cycle
         add      r19, r16
         adc      r20, r17
         jmp      fa_cycle
fa_end:
```

*Poznámka: nezapomeňte přidat na začátek programu inicializaci zásobníku.*

# Faktoriál – epilogue

```
        ; exit code  (epilogue)
fa_exit: mov     r26, r28
         mov     r27, r29
         ldd     r30, Y+4
         ldd     r31, Y+3
         adiw    r26, 5
         pop     r29
         pop     r28
         out     SPL, r26
         out     SPH, r27
         push    r19 ; ulož návratovou hodnotu na zásobník
         push    r18 ;
         ijmp
;
```

Volání podprogramu fact

```
ldi     r16, 8
push    r16     ; ulož parametr n
                    na zásobník
call    fact
pop     r16     ; vysledek
                    v r17:r16

pop     r17
```