Mikroprocesorová technika

Semestrální práce

Zadání č.2

Jan Jaroš
2.ročník KMT

Příklad č. 2

V paměti dat je uložen blok 8 b čísel se znaménkem. Sestavte program, který tato čísla srovná vzestupně podle velikosti. Pro řazení použijte algoritmus Insert Sort. Řazení bude napsáno jako podprogram se dvěma parametry – adresa začátku bloku dat a jeho délka.

Popis algoritmu Insert Sort

1. První prvek pole ponecháme na svém místě.
2. Vezmeme druhý prvek a porovnáme jej s prvním. Je-li menší, zařadíme ho na první místo a první prvek posuneme, jinak jej ponecháme na místě.
3. Vezmeme třetí prvek a porovnáme jej s prvními dvěma prvky. Je-li menší než některý z nich, zařadíme jej na odpovídající pozici a následující prvky podle potřeby posuneme. Jinak je ponecháme na původních místech.
4. Obdobně postupujeme i s ostatními prvky v poli.

Neboli můj program provede:

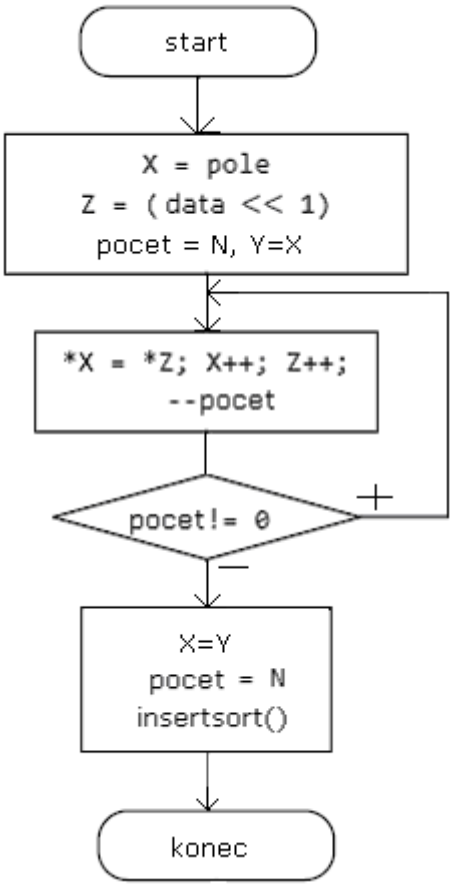
- Program nejdřív prověří sousedící prvky pole, jestli jsou seřazené (1<2,2<3,3<4,4<5,……)
  - Tím je zjištěno zda je pole seřazené. Když je zjištěno, že prvek potřebuje správně zařadit,hned se hledá nová pozice. Tím je tato funkce rychlejší, než kdyby se měla hledat pozice u setříděného prvku. Nevýhoda tohoto princi je, že když se přerovnává většina prvků pole, trvá celé třídění o trochu déle než bez tohoto kroku.
  - Toto porovnání je děláno pomocí BRLT, protože je pracováno se znaménkem.
  - V "pocitadlo" se počítá, kolik seřazených čísel je zkontrolováno.
- Jestli zjistí, že číslo není seřazené, hledá novou pozici pro ten prvek.
  - Hledá správnou pozici jen v seřazené části pole.
  - V průběhu hledání se v "odecitdadlo" uloží nově nalezená pozice tohoto čísla.
- Po nalezení vyhovující pozice se příslušné číslo zapíše, tak aby bylo pole seřazená.
  - Vezmou se hodnoty od pozice "odecitadla" až "pocitadlo"a odsunou se o jednu doprava, tím vzniklo místo pro zapsání čísla, kterému byla hledaná pozice.
- Dále se pokračuje v porovnávání sousedních čísel.

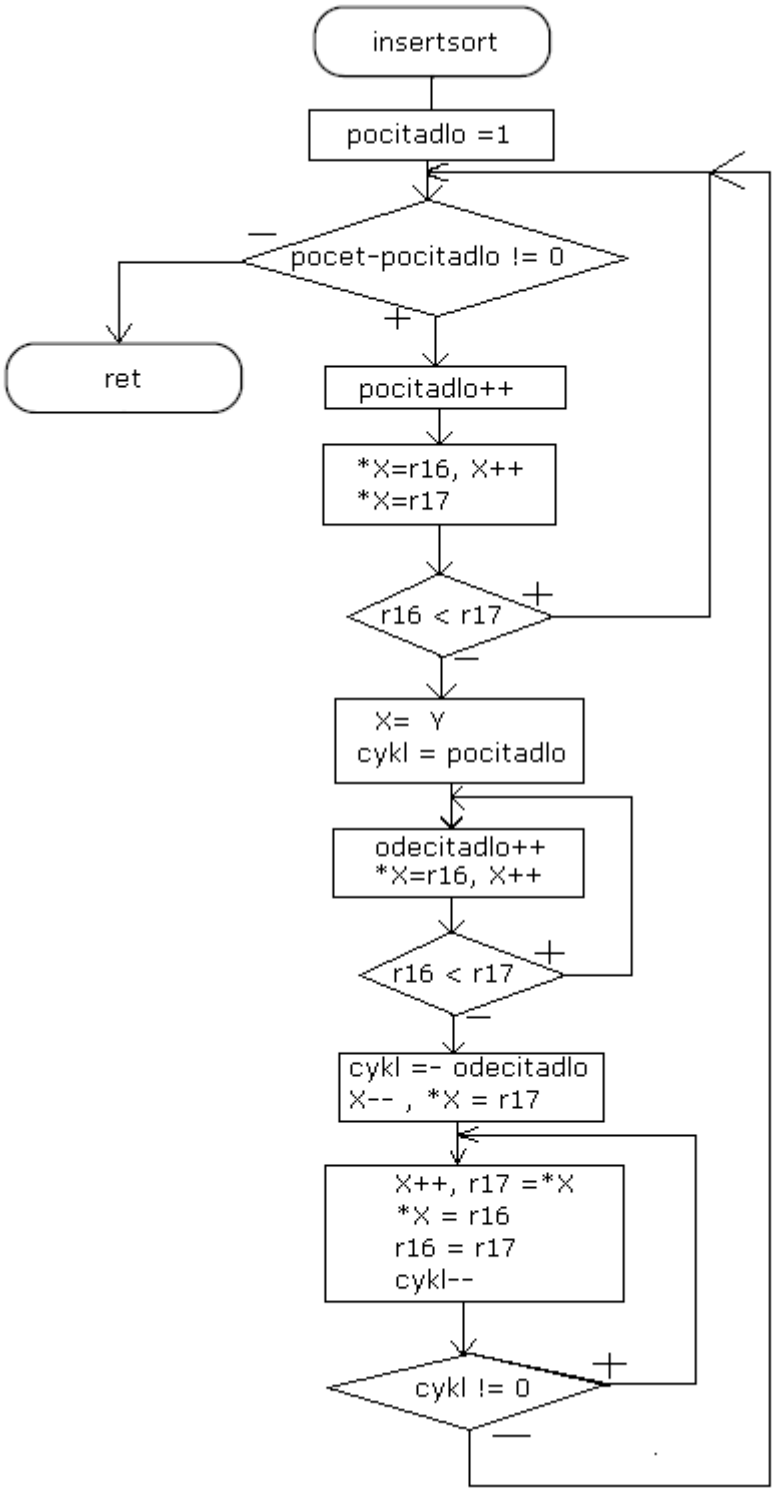
Popis hlavního programu

1. Načtení dat z paměti programu do pole proběhne v cyklu.
2. Do "pocet" se uloží počet prvků a do ukazatele X se uloží počáteční adresa pole. Do ukazatele Z se uloží adresa dat (uložených v paměti programu), adresa je o jedno místo posunuta doleva.
3. Postupně se do r16 uloží data z adresy, kam ukazuje pointer Z, Z se inkrementuje. Poté se do r16 uloží na adresu, kam ukazuje pointer X a následně se X inkrementuje.
4. "pocet" se dekrementuje a zkontroluje se zda-li není rovna nule, nyní se provede skok na začátek cyklu.
5. Pak se zase do "pocet" uloží počet prvků.

Popis podprogramu

1. Provede se záloha používaných registrů na zásobník.
2. Nastaví se počáteční hodnoty potřebné pro tuto funkci.
    (a) Pointer X za zálohuje(počáteční adresa pole) do pointeru Y.
    (b) Pocitadlo se nastavím do 1
3. If (Pocet == Pocitadlo) ,tak pole je seřazené a skočí na KONEC
4. Kontrola, zda je pole čísel seřazené:
    (a) Pocitadlo se inkrementuje.
    (b) Pomocí pointeru X je zkopírován prvek pole (na který ukazuje) do r16 a X se inkrementuje. Dále do r17 se zkopíruje prvek, na který ukazuje (po inkrementaci) X.
    (c) Porovná se, jestli prvek v r16 je menší než v r17. To je provedeno přes BRLT, protože porovnávám čísla se znaménkem. Jestli ano, skočím na začátek cyklu (3.bod), jestli ne, pokračuje dál.
5. Zjištění nové pozice čísla, aby bylo seřazené:
    (a) Odecitadlo se vynuluje
    (b) Pointer X se nastaví na první hodnotu pole
    (c) Inkrementuje se odecitadlo
    (d) Hodnota se nahraje do r16 a X se inkrementuje.
    (e) Znova je porovnáváno, jestli prvek v r16 je menší než v r17. Jestli ano, skočím na začátek cyklu (bod 5.c), jestliže ne, pokračuje dál.
6. Vytvoření místa pro neseřazené číslo a zapsání.
    (a) Odečtu od sebe pocet a odecitadlo a výsledek je uložen v cykl (kolik prvků pole se bude odsouvat)
    (b) X se dekrementuje a zapíše se hodnota z r17 na příslušnou pozici
    (c) X inkrementuje a do r17 se nahraje hodnota X aby se udělalo místo pro hodnotu z r16 dále se z r17 zkopíruje do r16
    (d) dekrementuje se cykl
    (e) cykl !=0 skoč na začátek cyklu (bod 6.c.) jestli = 0 tak pokračuj
    (f) If (Pocet == Pocitadlo) ,tak pokračuj v programu, jestli ne tak skoč na začátek cyklu(bod 3.)
    (g) ret


start
X = pole
Z = (data << 1)
pocet = N, Y=X
*X = *Z; X++; Z++;
--pocet
pocet!= 0
+
—
X=Y
pocet = N
insertsort()
konec


insertsort
pocitadlo =1
pocet-pocitadlo != 0
ret
pocitadlo++
*X=r16, X++
*X=r17
r16 < r17
X= Y
cykl = pocitadlo
odecitadlo++
*X=r16, X++
r16 < r17
cykl =- odecitadlo
X-- , *X = r17
X++, r17 =*X
*X = r16
r16 = r17
cykl--
cykl != 0

Program:

```
.include "m32def.inc"
// konstanty
.equ            N               = 10                    //nastavení délky pole čísel
//přejmenování registrů
.def            pocet           = r18
.def            pocitadlo       = r19
.def            odecitadlo      = r20
.def            cykl            = r21
// proměné v SRAM
                .DSEG
pole:           .BYTE           N

//program
                .CSEG
                ldi     r16, LOW(RAMEND)        // Nastavení zásobníku
                out     SPL, r16
                ldi     r16, HIGH(RAMEND)
                out     SPH, r16
                ldi pocet, N                    // počet čísel

                ldi     XL, LOW(pole)           // nastavení adresy pole do pointeru X
                ldi     XH, HIGH(pole)

                ldi ZL, LOW(data<<1)
                ldi ZH, HIGH(data<<1)

                movw Y,X                        // načtení hodnot
NACTENI:        lpm r16,Z+
                st X+,r16
                dec pocet
                brne NACTENI
                movw X,Y

                ldi pocet, N                    // počet čísel
                rcall INSERTSORT

END:            rjmp END

INSERTSORT:
                push r16                        // ochrana dat v registru
                push r17
                push pocet
                push pocitadlo
                push odecitadlo
                push cykl
                movw Y, X                       // záloha adresy pole prvků
                ldi pocitadlo, 1
```

```
DOBRE:
        cp pocet, pocitadlo     // když jsou všechny prvky pole seřazené ukončí podprogram
        breq KONEC
        inc pocitadlo                   // kolik hodnot bylo seřazeno
        ld r16, X+                      // načítání čísel
        ld r17, X
        cp r16, r17                     // jeli první číslo menší tak skoč
        brlt DOBRE

        movw X, Y                       // nastavení adresy pro úpravu pole
        mov cykl, pocitadlo             // pro mezi výpočet
        ldi odecitadlo, 0               // nastavení nuly

ZNOVA:  inc odecitadlo          // ZJIŠTĚNÍ KAM SE MÁ ČÍSLO DÁT ABY BYLO SEŘAZENÉ
        ld r16, X+                      // načtení prvního až n. prvku postupně
        cp r16, r17                     // zjištění kam neseřazené číslo patří
        brlt ZNOVA
        sub cykl, odecitadlo            // kolik prvků se má posunout
        st -X, r17                      // vložení čísla které nebylo seřazené

HOP:    adiw X, 1                       // =>posunutí všech ostatních prvků
        ld r17, X       // jedna hodnota se musí dát do meziregistru, aby se to tam dalo poskládat
        st X, r16
        mov r16, r17                    // vrácení hodnoty z meziregistru
        dec cykl                        //zjištění jestli se všechny čísla posunuly
        brne HOP
        cp pocet, pocitadlo         // když jsou všechny prvky pole seřazené ukončí podprogram
        brne DOBRE

KONEC:  pop cykl                        //vrácení dat registru z zásobníku
        pop odecitadlo
        pop pocitadlo
        pop pocet
        pop r17
        pop r16
        ret                             // vrácení z podprogramu

data:   .DB 1,10,3,-8,4,6,12,45,2,1
```