Fakulta elektrotechniky a informatiky
Univerzita Pardubice

Algoritmus řazení SelectSort

Semestrální práce z předmětu Mikroprocesorová technika

Jméno: Jiří Paar

Datum: 25. 5. 2008

Zadání

Napište program v jazyce symbolických adres, který pomocí algoritmu Select Sort seřadí blok 8b. čísel se znaménkem. Data budou seřazena sestupně podle velikosti. Řazení bude napsáno jako podprogram se dvěma parametry – adresa začátku bloku dat a jeho délka.

Popis algoritmů

Obecný popis algoritmu Select Sort

Princip:

1. V poli nalezneme prvek s největší hodnotou a zapamatujeme si jeho pořadí
2. Vyměníme tento prvek s prvkem na prvním místě
3. Postupně opakujeme body 1. a 2. nad zbývající, nesetříděnou částí pole.

Implementace:

- Celý algoritmus probíhá pomocí dvou vnořených cyklů.
- Vnější cyklus probíhá od začátku bloku dat do konce.
- Vnitřní cyklus prohledává od pozice ve vnějším cyklu do konce bloku dat. Vnitřní cyklus se tedy po každém průchodu vnějším cyklem o jednu zkrátí.
- Ve vnitřním cyklu se hledá maximum. Když je nalezeno zapamatuje se jeho pozice v poli (zapamatuje se hodnota vnitřního cyklu).
- Za vnitřním cyklem se prohodí prvek na začátku prohledávaného bloku dat s nalezeným maximem.

Popis algoritmu Select Sort na platformě AVR

Před voláním podprogramu se do pointeru X uloží adresa bloku dat a do registru `pocetN` se uloží počet prvků.

V průběhu podprogramu se hodnota pointeru X změní (bude ukazovat na první prvek za polem dat), registr `pocetN` se nezmění.

- Provede se záloha používaných registrů na zásobník. To pro případ, že by se registry používali před voláním podprogramu a jejich hodnoty by se i nadále používali po volání podprogramu.
- Vynulují se řídící proměnné cyklů (`vnitrniCykl`, `vnejsiCykl`) ⇒ inicializace cyklů.
- Prohledá se celý blok dat. Vyhledá se maximum a toto hledání se vždy posune o 1 prvek ke konci bloku dat. Toto nalezené maximum se prohodí s prvkem na začátku prohledávaného bloku dat.
    - Do registru `rMaxPozice` se uloží hodnota registru `vnejsiCykl` (předpokládá se, že hodnota v `vnejsiCykl` je pozice maxima, resp. první prvek v prohledávaném bloku dat je považován za maximum).
    - Do `vnitrniCykl` se zkopíruje `vnejsiCykl`. Hledání od pozice v registru `vnejsiCykl` ke konci. `vnitrniCykl` se inkrementuje, aby se hledání provádělo o další prvek dále. Prohledávání bude nad dalšími prvky.
    - Do pointeru Z se zkopíruje X. Z se používá jako ukazatel na prvek ve vnitřním cyklu.
    - Do `rMax` se uloží hodnota, kam ukazuje Z. V `rMaxPozice` je pozice v poli právě tohoto prvku. Z se inkrementuje (posun v poli na další prvek).
    - Provede se hledání maxima. V cyklu se prohledávají prvky od `pole[vnejsiCykl]` do `pole[pocetN-1]`.

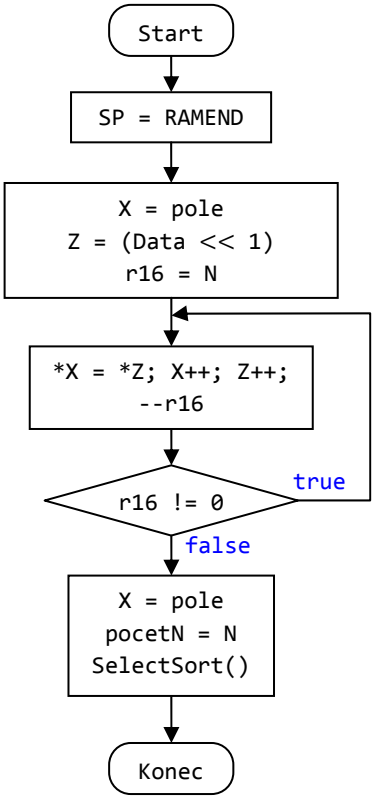
        - Porovná se `vnejsiCykl` s `pocetN`. Jsou-li stejné → konec cyklu.
        - Do `rTemp` se vloží hodnota, kam ukazuje Z. Z se inkrementuje (posun v poli na další prvek).
        - Porovná se `rMax` s `rTemp`. Je-li `rTemp` ≥ `rMax`, uloží se do `rMax rTemp` a do `rMaxPozice` se uloží hodnota `vnitrniCykl`.
        - `vnitrniCykl` se inkrementuje a skok na začátek cyklu.
    - Vypočte se rozdíl `pocetN – rMaxPozice` a tento rozdíl se opět odečte od hodnoty Z. Ve vnitřním cyklu se prošli všechny prvky až na konec a Z tedy ukazuje na první prvek za blokem dat. Těmito výpočty se dostaneme na pozici maxima v poli dat. Např. `pole = 0x60`, `pocetN = 10`, `rMaxPozice = 4`, `Z = 0x6A`, rozdíl `10 – 4 = 6` a `0x6A – 6 = 0x64`, tedy čtvrtý prvek v poli.
    - Do `rTemp` se uloží prvek na začátku prohledávaného bloku dat. Na začátek prohledávaného bloku dat se uloží `rMax` a X se inkrementuje, počátek prohledávaného bloku dat se posune na další prvek. `rTemp` se uloží na adresu, kam ukazuje pointer Z. Tímto postupem se provede prohození prvků v poli, záměna maxima s prvkem na začátku prohledávaného bloku dat.
    - `vnejsiCykl` se inkrementuje a porovná se s `pocetN`. Nejsou-li stejné skok na začátek vnějšího cyklu.

Popis hlavního programu

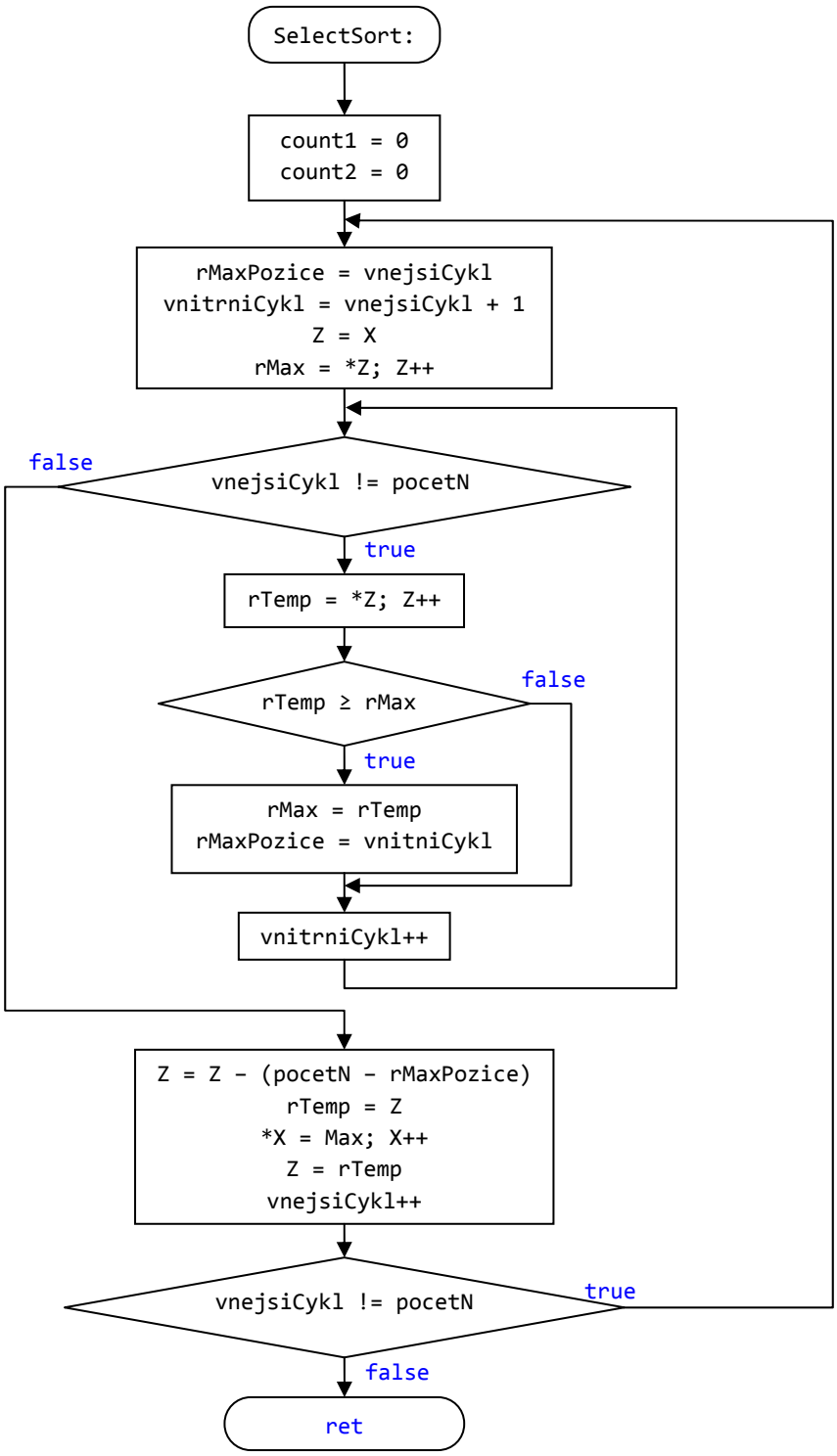
- Nastavení zásobníku na nejvyšší adresu paměti RAM
    - Zásobník při uložení dat svůj ukazatel snižuje.
- Načtení dat z paměti programu do pole proběhne v cyklu.
    - Do `r16` se uloží počet prvků a do ukazatele X se uloží počáteční adresa pole. Do ukazatele Z se uloží adresa dat (uložených v paměti programu), adresa je o 1 posunuta doleva.
    - Postupně se do `r17` uloží data z adresy, kam ukazuje pointer Z a Z se inkrementuje. Poté se do `r17` uloží na adresu, kam ukazuje pointer X a X se inkrementuje.
    - `r16` se dekrementuje a zkontroluje se zda-li není rovna nule, není-li provede se skok na začátek cyklu.

Vývojové diagramy

Hlavní program



Algoritmus řazení



Výpis programu

```
        .include        "m32def.inc"

.equ    N       = 10    // Konstanta s počtem prvků

.def    rMax            = r0    // Obsahuje aktuální maximum
.def    rMaxPozice      = r1    // Obsahuje pozici v poli aktuálního maxima
.def    rTemp           = r16   // Pomocný registr
.def    vnejsiCykl      = r17   // Řídící proměnná pro vnější cyklus
.def    vnitrniCykl     = r18   // Řídící proměnná pro vnitřní cyklus
.def    pocetN          = r19   // Registr s velikostí pole

        .dseg
pole:   .byte   N       // Pole o velikosti N

        .cseg
        .org    0x0000
// ****************************************
// Hlavní program
// ****************************************
        // Inicializase zásobníku
        ldi     r16, LOW(RAMEND)
        out     SPL, r16
        ldi     r16, HIGH(RAMEND)
        out     SPH, r16

        // Naplnění pole daty z ROM
        // Adresa pole
        ldi     XL, LOW(pole)
        ldi     XH, HIGH(pole)
        // Adresa dat
        ldi     ZL, LOW(Data << 1)
        ldi     ZH, HIGH(Data << 1)
        // Počet prvků
        ldi     r16, N
IniPole:
        lpm     r17, Z+ // Z ROM do r17
        st      X+, r17 // pole[i] = r17
        dec     r16     // i--
        brne    IniPole // if (r16 != 0) -> IniPole

        // Inicializace parametrů podprogramu pro řazení
        // Adresa pole
        ldi     XL, LOW(pole)
        ldi     XH, HIGH(pole)
        // Počet prvků
        ldi     pocetN, N

        rcall   SelectSort      // Volání podprogramu pro řazení

end:    rjmp    end     // Nekonečná smyčka
```

```
// ******************************************************************************
// Algoritmus pro řezení 8b. čísel se znaménkem pomocí algoritmu SelectSort
// Vstupní parametry:
//      pointer X = začátek pole dat
//      pocetN = počet prvků v poli
// ******************************************************************************
SelectSort:
        // Záloha registrů na zásobník
        push    rMax
        push    rMaxPozice
        push    vnitrniCykl
        push    vnejsiCykl
        push    rTemp
        push    ZL
        push    ZH

        // Vynulování řídících proměnných cyklů
        clr     vnitrniCykl
        clr     vnejsiCykl
VnejsiSmycka:
        mov     rMaxPozice, vnejsiCykl  // rMaxPozice = vnejsiCykl
        mov     vnitrniCykl, vnejsiCykl // vnitrniCykl = vnejsiCykl
        inc     vnitrniCykl             // vnitrniCykl = vnejsiCykl + 1
        // Z = X, vnitřní cyklus bude prohledávat od prvku v X
        movw    Z, X
        ld      rMax, Z+                // rMax = *Z, Z++
VnitrniSmycka:
        cp      vnitrniCykl, pocetN
        breq    EndVnitrniSmycka        // if (vnitniCykl == pocetN) -> EndVnitrniSmycka
        ld      rTemp, Z+               // rTemp = *Z, Z++
        cp      rTemp, rMax
        brlt    DalsiPrvek              // if (rTemp < rMax) -> DalsiPrvek
        // Nalezeno nové maximum
        mov     rMax, rTemp             // rMax = rTemp
        mov     rMaxPozice, vnitrniCykl // rMaxPozice = vnitrniCykl
DalsiPrvek:
        inc     vnitrniCykl             // vnitrniCykl++
        rjmp    VnitrniSmycka           // -> VnitrniSmycka
EndVnitrniSmycka:
        // Výpočet prvku kam se má uložit prvek na jehož pozici patří rMax
        // Z -= (pocetN - rMaxPozice)
        mov     rTemp, pocetN           // rTemp = pocetN
        sub     rTemp, rMaxPozice       // rTemp = pocetN - rMaxPozice

        sub     ZL, rTemp               // ZL = ZL - (pocetN - rMaxPozice)
        sbci    ZH, 0                   // ZH = ZH - C

        // Prohození prvků
        ld      rTemp, X                // rTemp = *X
        st      X+, rMax                // *X = rMax, X++
        st      Z, rTemp                // *Z = rTemp

        // Administrava vnejšího cyklu
        inc     vnejsiCykl              // vnejsiCykl++
        cp      vnejsiCykl, pocetN
        brne    VnejsiSmycka            // if (vnejsiCykl != pocetN) -> VnejsiSmycka
```

```
        // Obnova registrů ze zásobníku
        pop     ZH
        pop     ZL
        pop     rTemp
        pop     vnejsiCykl
        pop     vnitrniCykl
        pop     rMaxPozice
        pop     rMax

        ret                             // Návrat z podprogramu

Data:
        .db     5, -10, -67, 89, 120, 67, -100, 0, 2, 5 // Data pro naplnění pole
```