1 BBsort

pred citanim si pozrite znovu instrukcie RAM-u. tiez sa mozno oplati zopakovat si, ako postupuje bubblesort pri triedeni.

program sa velmi jemne lisi od toho, ktory som pisal na tabulu na cviceni .... z pedagogickych dovodov pribudli dva riadky.

program ***ma*** *vela zbytocnych riadkov. su tam na to, aby bolo jasnejsie co, kde a s cim aktualne robime, a tiez aby bolo lepsie vidno, co mame v ktorych registroch.*

ak najdete niekde chybu, mozete mi napisat na fduris@dcs.fmph.uniba.sk

zaciname. vstup vyzera nasledovne:

$$n,\ a_1,\ a_2,\ ...,\ a_n$$

aby sme s nim mohli rozumne pracovat, nahrame ho do registrov. dalo by sa nahrat tieto cisla od prveho registra R_1 dalej. da sa ale ocakavat, ze pocas triedenia budeme potrebovat nejaky ten register na pomocne operacie, preto by bolo dobre nechat si prvych par registrov v zalohe. takych 9 by mohlo stacit.

ok, cislo n teda nacitame do 10teho registra a potom cisla $a_1,\ a_2,\ ...$ do registrov $R_{11},\ R_{12},\ ...$ atd. posledne cislo a_n bude v registri R_{10+n} (premyslite si to).

tato cast programu vyzera nasledovne:

```
1 READ   1
2 LOAD   =10
3 STORE  2
4 LOAD   1
5 STORE  *2
6 LOAD   =11
7 STORE  2
```

v prvom riadku sa nacitali prve cislo (n) zo vstupu a ulozili ho do (pomocneho) registra R_1. toto cislo n ale chceme mat aj v 10tom registri.

ok, najpr si nastavime adresu, kam ho chceme ulozit ... 10ty register ... adresa je 10. tuto adresu si ulozime do druheho registra R_2. uvedomte si, ze nemame instrukciu na to, aby sme do R_2 ulozili nejake cislo len tak ... musime ist cez register R_0[1] ... teda najprv nahrame do R_0 cislo 10 (druhy riadok) a potom R_0 ulozime do R_2 ... treti riadok.

tymto sme si nastavili spravnu adresu, kam chceme ulozit cislo n (ktore je zatial v prvom registri). ako som uz spominal, vsetko kopirovanie medzi registrami musi ist cez R_0 (lebo mame take instrukcie). preto najprv skopirujeme R_1 do R_0 a potom ho ulozime na tu adresu, ktora je v druhom registri (**nepriame** adresovanie) ... to su riadky 4 a 5.

$$R_0 := R_1$$
$$R_{R_2} := R_0$$

tymto mame v R_{10} ulozene cislo n (a samozrejme je tiez stale ulozene v R_1).

aby sme mohli nacitat ostatne cisla zo vstupu, budeme potrebovat cyklus ... teda budeme potrebovat nejake pocitadlo kolko krokov treba este spravit (aby sme sa vedeli zastavit) a tiez budeme potrebovat vediet adresy, kam budeme citane prvky ukladat.

na pocitadlo nam bude sluzit R_1 ... momentalne tam je ulozene cislo n a ked po kazdom nacitanom cisle zo vstupu odcitame od R_1 jednotku, vstup budeme docitany ked v R_1 bude nula.

na spravne adresovanie budeme pouzivat register R_2. kedze v R_{10} je ulozene cislo n, dalsi volny register je R_{11} ... jeho adresa je teda 11 ... toto cislo si chceme ulozit do R_2 (riadky 6 a 7). sme pripraveni na cyklus :)

ak mate pocit, ze tato uvodna cast je zbytocne prekomplikovana, a ze by sa dala spravit jednoduchsie, tak je to dobre ... jej zamerom bolo hlavne precvicit si veci, ktore budeme dalej potrebovat.

cyklus:

```
8  LOAD    1
9  JZERO   18
```

[1] nie je celkom pravda, instrukcia READ nam umoznuje ulozit vstup do hocijakeho registra ... je vsak jasne, ze teraz tuto instrukciu pouzit nechceme.

```
10  READ    *2
11  LOAD    2
12  ADD     =1
13  STORE   2
14  LOAD    1
15  SUB     =1
16  STORE   1
17  JUMP    8
```

uz som spominal, ze obsah nejakeho registra vieme zmenit len tak, ze donho nahrame obsah R_0. rovnako, ked chceme zistit, ci je v nejakom registri ulozena nula, vieme to spravit len tak, ze tento register nahrame do R_0 a pozrieme sa tam.

na zaciatku kazdeho cyklu je podmienka, ktora, ak treba, cyklus zastavi. povedali sme si, ze nase pocitadlo bude v registri R_1 a ze cyklus nacitavania skonci, ked R_1 klesne na nulu ... toto teda chceme overit. do R_1 sa ale pozriet nevieme, takze ho najprv nahrame do R_0 a pozrieme sa tam (8mi a 9ty riadok).

v pripade, ze v R_1 je uz nula, vieme, ze sme precitali vsetky cisla zo vstupu, a teda cele telo cyklu preskocime (JZERO 18).

v pripade, ze v R_1 este nula nie je, spravime nasledovne: precitame cislo zo vstupu a ulozime ho na adresu v R_2 (**nepriame** adresovanie). spomente si, ze pred tymto cyklom sme si do R_2 nastavili spravnu adresu pre prve cislo a_1, a preto mozeme prve cislo rovno citat a ulozit. teraz ale musime tuto adresu zmenit tak, aby sme dalsie cislo zapisali do dalsieho registra v poradi. to spravime jednoducho tak, ze k R_2 pripocitame jednotku (aby mal adresu nasledujuceho registra).

opat ale narazame na to, ze obsah R_2 priamo zmenit nemozeme. preto musime ist cez R_0 ... nahrame donho R_2, tam pripocitame jednotku, a potom vysledok opat nahrame do R_2 (11ty, 12ty a 13ty riadok).

no a kedze sme teraz precitali dalsie cislo zo vstupu, znizime nase pocitadlo v R_1 o jednotku ... opat cez register R_0 (14ty, 15ty a 16ty riadok).

toto je koniec tela cyklu, skocime naspat na podmienku, ci sme uz precitali vsetky cisla.

teraz mame vsetky cisla nacitane v registroch R_{11}, R_{12}, ..., R_{n+10}. rozmyslite si, preco je v R_1 nula a v R_2 $(n + 10)$.

dostali sme sa teda k samotnemu triedeniu. ako postupuje bubblesort? ak si predstavime, ze vstupne cisla su ulozene v poli, potom BBS zoberie prve dve susedne cisla, porovna ich, a ak je vacsie cislo pred mensim, tak ich vymeni. potom zoberie dalsie dva prvky a urobi to iste atd.

jeden takyto prechod cez pole vsak vo vseobecnosti toto pole neutriedi. pre jednoduchost teda spravime tychto prechodov n (asi by sme to vedeli spravit aj s mensim poctom ale teraz je nam to jedno).

teda budeme mat dva cykly ... vnutorny zakazdym prejde cez pole (registre $R_{11}...R_{10+n}$), porovna susedne prvky a pripadne ich vymeni. vonkajsi cyklus bude n krat opakovat ten vnutorny. na toto budeme potrebovat dve pocitadla. pre vnutorny cyklus budeme mat pocitadlo v R_1 a pre vonkajsi napr. v R_9.

```
18 LOAD    10
19 STORE   9
20 LOAD    9
21 JZERO   55
```

18ty a 19ty riadok nastavi pocitadlo vonk. cyklu R_9 na n (cislo n mame stale ulozene v 10tom registri; do R_9 ho odtial cez register R_0 nakopirujeme). 20ty a 21vy riadok je podmienka, ci uz vonk. cyklus skoncil.

kedze vnutorny cyklus budeme velakrat opakovat, musime si pred jeho zacatim vzdy nastavit jeho parametre. aby sme vedeli susedne prvky porovnavat, budeme si v registroch R_2 a R_3 drzat adresy aktualnej dvojice .... teda na zaciatku bude v $R_2 = 11$ a $R_3 = 12$ (lebo to su nase prve dva prvky), potom $R_2 = 12$ a $R_3 = 13$ atd az nakoniec $R_2 = 10 + (n - 1)$ a $R_3 = 10 + n$ (premyslite si to).

pred samotnym vnut. cyklom si teda najpr nastavime adresy v R_2 a R_3 na prve dva prvky "pola".

este si musime vhodne nastavit pocitadlo R_1. vsimnite si, ze adresy v R_2 idu od $10+1$ az po $10+(n-1)$ (podobne adresy v R_3 idu od $10+2$ az po $10 + n$). teda vnutorny cyklus bude mat len $(n - 1)$ opakovani (premyslite si to).

```
22 LOAD    10
23 SUB     =1
24 STORE   1
```

```
25 LOAD    =11
26 STORE   2
27 LOAD    =12
28 STORE   3
29 LOAD    1
30 JZERO   51
```

spomente si, ze v R_{10} mame stale ulozene cislo n. 22hy, 23ti a 24ty riadok preto nahra R_{10} do R_0, tam od neho odcitame jednotku a vysledok ulozime do R_1 ... pocitadlo mame nastavene.

dvojica riadkov 25, 26 nahra do R_0 cislo 11 a ulozi ho do R_2 ... adresa prveho prvku porovnavanej dvojice (opat pripominame : robime to takto, lebo inak tam tu 11tku dostat nevieme).

dvojica riadkov 27, 28 nahra do R_3 cislo 12 ... tymto mame aj adresy spravne nastavene.

riadky 29 a 30 testuju, ci uz vnutorny cyklus skoncil (ci je v R_1 nula) .... po nich nasleduje telo vnutorneho cyklu.

dostavame sa k samotnemu porovnavaniu prvkov. ako to ale spravit, ked nemame nijaku instrukciu na porovnanie dvoch cisel?

majme dve cisla a, b. to co s nimi vieme spravit, je ich od seba odcitat, povedzme $b - a$ (na toto mame instrukciu SUB). ak bolo $a \geq b$, potom bude $b - a \leq 0$.

```
31 LOAD    *3
32 SUB     *2
33 JZERO   41
34 JPOS    41
```

31vy prvy riadok nahra do registra R_0 cislo v R_{R_3} (cize R_{i+1} ... nepriama adresacia). nasledne 32hy riadok odpocita od R_0 cislo v registri R_{R_2} (cize R_i). ak nam vyjde, ze tento rozdiel je nula, prvky su rovnake, a teda spravne usporiadane a nemusime ich vymienat (JZERO preskoci vymienanie prvkov).

podobne, ak nam vyjde rozdiel kladny, to bude znamenat, ze cislo v registri R_{i+1} je vacsie ako cislo v R_i, a teda su opat v dobrom usporiadani a znovu skocime prec (JPOS preskoci vymienanie prvkov).

jedine v pripade, ze rozdiel bude zaporny ($R_i > R_{i+1}$) bude treba prvky

vymenit (teda nepreskocime vymienanie prvkov, ktore bude nasledovat).

ok, vieme porovnat prvky. este ich treba vediet prehodit. na to vyuzijeme pomocny register napr. R_5. to co chceme spravit je iba

$$\begin{aligned} R_5 &:= R_i \\ R_i &:= R_{i+1} \\ R_{i+1} &:= R_5 \end{aligned}$$

to co nam dovoluju instrukcie je ale uplne ina vec ... vsetko musi ist cez R_0. v realite teda robime nieco taketo

$$\begin{aligned} R_0 &:= R_i \\ R_5 &:= R_0 \\ R_0 &:= R_{i+1} \\ R_i &:= R_0 \\ R_0 &:= R_5 \\ R_{i+1} &:= R_0 \end{aligned}$$

no a aby to bolo uplne matuce, my sa nevieme dostat k R_i len takto lahko ... musime pouzit nepriamu adresaciu (vieme, ze adresa i je ulozena v R_2 a adresa $i+1$ je ulozena v R_3). takze to co v skutocnosti robime je toto

$$\begin{aligned} R_0 &:= R_{R_2} \\ R_5 &:= R_0 \\ R_0 &:= R_{R_3} \\ R_{R_2} &:= R_0 \\ R_0 &:= R_5 \\ R_{R_3} &:= R_0 \end{aligned}$$

takze:

```
35  LOAD    *2
36  STORE   5
```

```
37 LOAD    *3
38 STORE   *2
39 LOAD    5
40 STORE   *3
```

tymto sme porovnali a, ak bolo treba, prehodili dva susedne prvky. teraz sa potrebujeme posunut na dalsie dva ... na to nam staci ku registrom R_2 a R_3 pripocitat jednotku

$$\begin{aligned} R_2 = i &\rightarrow R_2 = i + 1 \\ R_3 = i + 1 &\rightarrow R_3 = i + 1 + 1 = i + 2 \end{aligned}$$

opat musime ist cez R_0 ...

```
41 LOAD    2
42 ADD     =1
43 STORE   2
44 LOAD    3
45 ADD     =3
46 STORE   3
```

no a nakoniec tiez musime znizit pocitadlo R_1 o jednotku

```
47 LOAD    1
48 SUB     =1
49 STORE   1
```

toto je koniec vnutorneho cyklu, skocime naspat na podmienku, kde testujeme, ci uz ma vnutorny cyklus skoncit.

```
50 JUMP    29
```

skoncili sme vnutorny cyklus ale este stale sme vo vonkajsom. kedze ten vsak nerobi nic ine, nez opakuje n krat vnutorny cyklus, jedine, co treba spravit, je znizit pocitadlo vonkajsieho cyklu R_9 o jednotku. potom skocime na podmienku, ci uz vonkajsi cyklus skoncil.

```
51 LOAD    9
52 SUB     =1
53 STORE   9
54 JUMP    20
```

huraaa, mame nase pole utriedene. ostava nam uz iba tieto prvky vypisat na vystup. opat pouzijeme cyklus s pocitadlom v R_1. adresu aktualne vypisovaneho prvku budeme mat v R_2. na zaciatku teda treba nastavit $R_1 = n$ (cislo n mame stale bezpecne ulozene v R_{10}) a $R_2 = 11$ (adresa prveho prvku).

```
55 LOAD    10
56 STORE   1
57 LOAD    =11
58 STORE   2
```

dalej je samotny cyklus vypisovania. najpr testujeme na nulu register R_1 a ak je nenulovy, tak pokracujeme telom cyklu. v tele cyklu potom vypiseme prvok na adrese v R_2 (**nepriame** adresovanie), posunieme adresu na dalsi prvok (register) a odpocitame od pocitadla jednotku.

az bude pocitadlo R_1 nula, skocime na HALT.

```
59 LOAD    1
60 JZERO   69
61 WRITE   *2
62 LOAD    2
63 ADD     =1
64 STORE   2
65 LOAD    1
66 SUB     =1
67 STORE   1
68 JUMP    59
69 HALT
```

That's all Folks :)

koniec. toto je cely program. ak sa vam stale motaju jumpy a cykly, mozno pomoze napisat si vsetky riadky pod seba a nakreslit si sipky odkial a kam sa skace.

pripadne si skuste kreslit obrazky s registrami a sipkami, ktory register kam ukazuje.