Praktische opgave 1: Assembly

David N. Jansen

ICT Infrastructuren 2013/14

Je gaat een sorteeralgoritme, te weten comb sort, implementeren in assembly. Je gebruikt x86-assembly omdat je zeer waarschijnlijk zelf een x86-processor in je computer hebt. Algoritme 1 laat het comb-sort-algoritme zien. Het idee erachter is: Vergelijk telkens twee elementen in de array, en als het eerste element groter is dan het tweede, verwissel ze. In het begin worden elementen vergeleken die verder van elkaar af staan, en later worden elementen vergeleken die direct bij elkaar staan. Het is niet erg efficiënt, maar simpel genoeg om in assembly te implementeren. De array $array[0], array[1], \ldots, array[length-1]$ wordt gesorteerd. Deze variant geeft als resultaat het aantal keer terug dat twee elementen zijn verwisseld.

Algorithm 1 Comb sort

```
1.1:  function comb_sort(array, length)
1.2:      result := 0
1.3:      gap := length, swapped := false
1.4:      while (gap > 1) ∨ swapped do
1.5:          if gap > 1 then
1.6:              gap := 100 · gap / 128, naar beneden afgerond
1.7:          end if
1.8:          swapped := false
1.9:          for i := 0, 1, ..., length − gap − 1 do
1.10:             if array[i] > array[i + gap] then
1.11:                 Verwissel array[i] met array[i + gap]
1.12:                 result := result + 1
1.13:                 swapped := true
1.14:             end if
1.15:         end for
1.16:     end while
1.17: return result
```

Op de website vind je een *.tar.gz* bestand dat een kader bevat voor het implementeren. Het bevat drie bestanden:

comb_sort.c is het hoofdprogramma dat dient om jouw implementatie te testen;

my_comb_sort.s is het nog (bijna) lege bestand waarin jouw implementatie komt te staan.

Makefile wordt gebruikt door het tool make en maakt de compilatie makkelijker. Onder Linux is make normaal gesproken al geïnstalleerd; bij Windows heel soms ook.

Compileer het programma eerst eens door in een commandovenster te typen:

```
$ make
gcc -Os -m32 my_comb_sort.s comb_sort.c -o my_comb_sort
$ ./my_comb_sort
Test met 2 elementen
```

```
Voor sorteren: 4322  520
Na sorteren:   4322  520
Resultaat van de functie: 1171
Verstreken tijd: 2e-06 seconden
Fout: de array is niet goed gesorteerd!
...
$ _
```

Zoals je ziet wordt de array niet gesorteerd, en is het aantal verwisselingen een schijnbaar willekeurig getal. De opdracht is om het comb-sort-algoritme in het bestand my_comb_sort.s in x86-assembly te implementeren.

Een paar tips

- De parameters van de functie zijn het *adres* van de array en de lengte. De eigenlijke elementen staan dus in het RAM, op adres *array*, *array* + 4 etc. Dat betekent dus dat je eerst *array* in een register moet laden en dan die registerwaarde als adres moet gebruiken om het eigenlijke element van de array aan te spreken.

- Je kunt beschrijvingen van alle assembly-instructies op het web vinden, b.v. het Intel 80386 Programmer's Reference Manual 1986; op http://en.wikipedia.org/wiki/Intel%5F80386 staat ook een algemene beschrijving. De x86-processoren hebben heel veel instructies, waarvan je slechts een paar weinige nodig hebt. Lees de documentatie over de instructies mov, reken-operaties als add, sub, inc, dec, tests als cmp, en (conditional) jumps zoals jz, jle.

  Bij de meeste rekenoperaties moet één van de operanden een register zijn en mag de andere een register of RAM-cel zijn. De RAM-cel identificeer je met „[*label* + *register1* + *constante factor***register2*]". (Elk van deze delen is optioneel; de constante factor moet 1, 2, 4 of 8 zijn; de volgorde maakt niet uit.) Voorbeelden van adressen: [array], [ebp+8], [eax*4+edi–4], [8*ecx+label], [esi+123456+ebx].

- Variabelen kun je in de registers eax, ebx, ecx, edx, esi en edi opslaan. (esp en ebp zijn al in gebruik.) Elke keer als je een nieuwe variabele tegenkomt, moet je een register kiezen. Bij ingewikkelde programma's moet je sommige variabelen in het RAM opslaan, maar deze functie is eenvoudig genoeg dat dat niet nodig is. Alleen de array is meestal zo groot dat je uitsluitend zijn adres in een register kunt plaatsen.

  Zet een duidelijk commentaar in je programma dat aangeeft welke variabele je in welk register opslaat.

- Implementeer eerst een paar regels van de functie, b.v. zorg ervoor dat *result* wordt teruggegeven (d.w.z. aan het einde van de functie in eax staat). Compileer en test je functie. Daarna voeg je het begin en einde van de **while**-loop aan de functie toe en compileer en test je hem opnieuw. Daarna de rest van de functie...

- Gebruik voor multiplicatie de instructie imul *doelregister, bronregister, constante*. Vervang divisie door shift-naar-rechts: sar *register, constante* werkt als een divisie door $2^{constante}$.

Inleveren

Je levert uiterlijk **maandag 9 december 2013,** voor het hoorcollege begint, een werkend assembly-programma in dat het comb-sort-algoritme implementeert (het bestand my_comb_sort.s dus). In dit bestand geef je uitleg welk deel van je assembly-programma overeenkomt met welk deel van het algoritme hierboven. Je geeft ook aan welke variabele in welk register staat.

Maluspunt. Als je de deadline met 1 minuut tot 2 uur overschrijdt krijg je –0,5 maluspunt, bij overschrijding van 2 tot 24 uur –1 maluspunt op je eindcijfer. Bij verdere overschrijding reken ik de opgave als niet ingeleverd en mag je niet aan het tentamen deelnemen.