

# Vervangende Opgave 1: SPARC assembler

David N. Jansen

ICT Infrastructures 2007/08

Deze opgave kunnen studenten indienen die aan het vak ICT infrastructures 2007/08 hebben deelgenomen en huiswerkopgave 1 niet of niet voldoende hebben ingeleverd. Dit is een eenmalige actie; het is niet waarschijnlijk dat ik volgend jaar weer een vervangende opdracht aanbied.

Omdat deze opdracht ter vervanging van een andere dient, herhaal ik niet alle tekst die al in de andere opdracht stond.

We gaan een algoritme om priemgetallen te vinden, namelijk het *zeef van Eratosthenes*, implementeren in SPARC assembler. Dit is een efficiënt algoritme om alle priemgetallen tussen 1 en een van te voren gekozen bovengrens  $N$  te vinden. Het algoritme krijgt een pointer naar een array van bytes *priem* als parameter (alle elementen van het array zijn oorspronkelijk = 0); het algoritme zet *priem*[ $i$ ] op 1 als  $i$  priem is. Als resultaat levert de functie het aantal priemgetallen  $\leq N$ .

---

**Algorithm 1** Zeef van Eratosthenes

---

```
1.1: function priemgetallen(priem,  $N$ )
1.2:   if  $N < 2$  then
1.3:     return 0
1.4:   end if
1.5:   {2 is het enige even priemgetal en krijgt speciale behandeling;}
1.6:   priem[2] := 1
1.7:   aantal := 1
1.8:   {Eerst vermoeden we dat alle oneven getallen  $> 1$  priem zijn;}
1.9:   for  $i := 3, 5, 7, \dots, N$  do
1.10:    priem[ $i$ ] := 1
1.11:   end for
1.12:   {Daarna strepen we alle veelvouden weg;}
1.13:   for  $i := 3, 5, 7, \dots, N$  do
1.14:     for  $j := 2 \cdot i, 3 \cdot i, 4 \cdot i, \dots, N$  do
1.15:       { $j$  doorloopt alle veelvouden van  $i$ ; deze getallen zijn niet priem.}
1.16:       priem[ $j$ ] := 0
1.17:     end for
1.18:   end for
1.19:   {Nu tellen we de overgebleven priemgetallen;}
1.20:   for  $i := 3, 5, 7, \dots, N$  do
1.21:     if priem[ $i$ ]  $\neq 0$  then
1.22:       aantal := aantal + 1
1.23:     end if
1.24:   end for
1.25: return aantal
```

---

Op de website vind je een *.tar.gz* file dat een kader bevat voor het implementeren en testen van jouw implementatie van het zeef van Eratosthenes. De bediening van de SPARC-assembler heb ik al in de oorspronkelijke opgave 1 uitgelegd en herhaal ik hier niet.

Het bestand `eratosthenes.s` bevat een raamwerk waarin je jouw assembly-programma kunt invoegen. Door `make` in te typen wordt jouw programma vertaald; daarna kun je het programma `erato` starten om het te testen.

1. Vertaal het algoritme van het zeef van Eratosthenes naar SPARC-assembly en lever een werkend bestand `eratosthenes.s` in. Geef duidelijk commentaar in jouw programma.
2. Probeer het algoritme nog efficiënter te maken: je mag de bovenstaande pseudocode veranderen en/of jouw assembly-programma optimaliseren. Leg uit wat je doet. De snelste implementatie krijgt een bonuspunt.

Als je wilt deelnemen, moet je mij dat tot donderdag 20 maart 2008, 12.00 uur meedelen. Daarna heb je vier weken de tijd om jouw uitwerking te maken; je moet hem dus uiterlijk op donderdag 17 april 2008, 12.00 uur inleveren. Deze deadline wordt niet verlengd.

Voor eventuele vragen kun je langskomen in mijn nieuwe kamer, 02.612 in het Huygensgebouw, of mij e-mailen.