|  |  |  |
| --- | --- | --- |
|  | **Universiteti i Prishtinës**  Fakulteti i Inxhinierisë Elektrike dhe Kompjuterike  Arkitektura e Kompjuterëve, 2022/2023  Prof. Valon Raça & Synim Selimi | Zana Ademi  220756100008 |

# Detyra e parë: Opsioni C

## Hyrje

Programi i mëposhtëm llogarit dy shuma sipas logjikës në vijim: është deklaruar dhe inicializuar vektori me 7 anëtarë, variabla sum\_pre që fillimisht mban vlerën e anëtarit të parë të vektorit, dhe variabla sum\_post me vlerën fillestare zero. Me anë të një unaze for kalojmë nëpër anëtarët e vektorit. Në sum\_pre e vendosim shumën e anëtarëve të vektorit. Ndërsa elementit në indeksin i-1 ia asociojme shumën mes: elementit në pozitën i dhe indeksit i (dmth vektori[i]+i). Pastaj në sum\_post e vendosim shumën e elementeve në pozitën i-1. Kjo unazë përsëritet derisa të përmbushet kushti që iteratori të jetë më i vogël se madhësia e vektorit. Pasi të dalim nga unaza, në sum\_post e vendosim shumën mes vlerës që ndodhet aktualisht në sum\_post dhe elementit të fundit të vektorit. Në fund i printojmë vlerat e llogaritura.

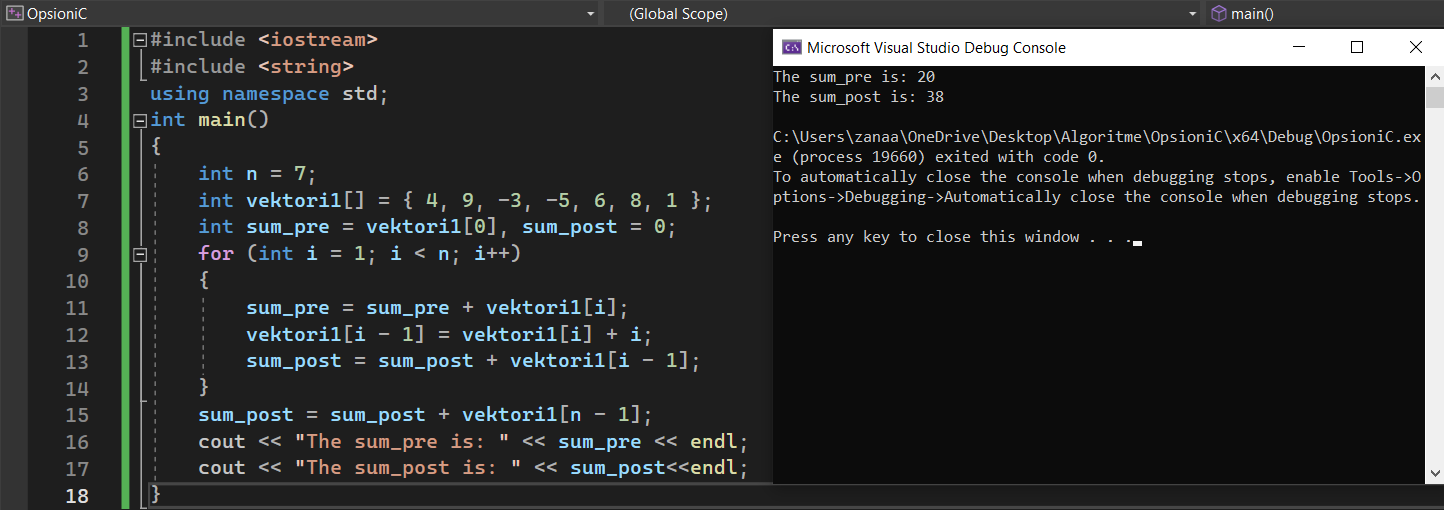


Figura 1. Kodi i shkruar në C++ dhe rezultati i shfaqur në console pas ekzekutimit të programit

## Realizimi i kodit në MIPS

Në pjesën .data kam deklaruar vektorin, si dhe dy mesazhe e një rend i ri që do të përdoren në fund kur të paraqesim rezultatin në ekran. Në direktivën .text gjendet i tërë kodi i programit. Nën etiketën main, në regjistrin $t0 me anë të instruksionit la (load address) kam vendosur adresën bazë te vektorit. Me anë të instruksionit addi (add-immediate që i takon I-formatit, mund ta përdorim për të mbledhur vlerën në një regjistër me një konstantë-vlerë imediate), kam vendosur vlerën 1 në $t1(për inicializimin e iteratorit), vlerën 7 në $t2(madhësia e vektorit), si dhe vlerën e sum\_post në $s3. Ndërsa me anë të instruksionit lw (load word-shërben për marrjen e të dhënave nga memoria), kam vendosur vlerën e anëtarit të parë të vektorit në $s1 (inicializimi i sum\_pre), dhe vlerën e anëtarit të fundit në $s3.

Nën etiketën begin\_loop, me anë të instruksionit sll (shift left logical-mund të përdoret për shumëzim me 2^i) kam shumëzuar vlerën e i me 4, në mënyrë që të kërcejmë tek elementi i rradhës në vektor. Me anë të instruksionit add (R-format) kam mbledhur 4\*i me adresën fillestare të vektorit (që gjendet në $t0), që të lëvizim për 4\*i nga adresa bazë e vektorit, dhe këtë vlerë e kam vendosur në regjistrin $t3. Në regjistrin $t4 kam vendosur vlerën që ndodhet në $t3 me një offset 0 (sepse dy instruksionet e mëparshme përkujdesen për kalim nëpër anëtarë të vektorit), përkatësisht kam vendosur vlerën e anëtarit në pozitën i(vekori[i]) në $t4. Me anë të slt (set less than) kam krahasuar vlerën e i me n, në rastin kur i<n, vlera ne $t5=1, përndryshe $t5=0. Me anë të beq (branch if equal) kam krahasuar vlerën në $t5 me zero, dhe nëse këto dy vlera janë të barabarta, programi kërcen te etiketa End. Përndryshe programi vazhdon dhe llogarit shumën sum\_pre (shumën e elementeve të vektorit). Pastaj e llogarit offset-in (i-1) për të kaluar nëpër anëtarët e vektorit në pozitën (i-1). Kjo pjesë poashtu realizohet me ndihmën e instruksioneve sll, add dhe lw (për ta vendosur vlerën e vektori[i-1] në regjistrin $t8 në rastin tonë). Me anë të add në $t8 (ku ndodhet vektori[i-1]) vendoset shuma e $t4 (vektori[i]) dhe $t1 (i). Pastaj llogaritet sum\_post si shumë mes vlerës në sum\_post dhe vlerës aktuale në vektori[i-1]. Pas kësaj duhet siguruar që ta rrisim vlerën e i-së dhe të kërcejmë te fillimi i unazës (me anë të jump, kërcejmë te etiketa Loop), në mënyrë që të përsëritet ky proces përserisa i është më e vogël se n.

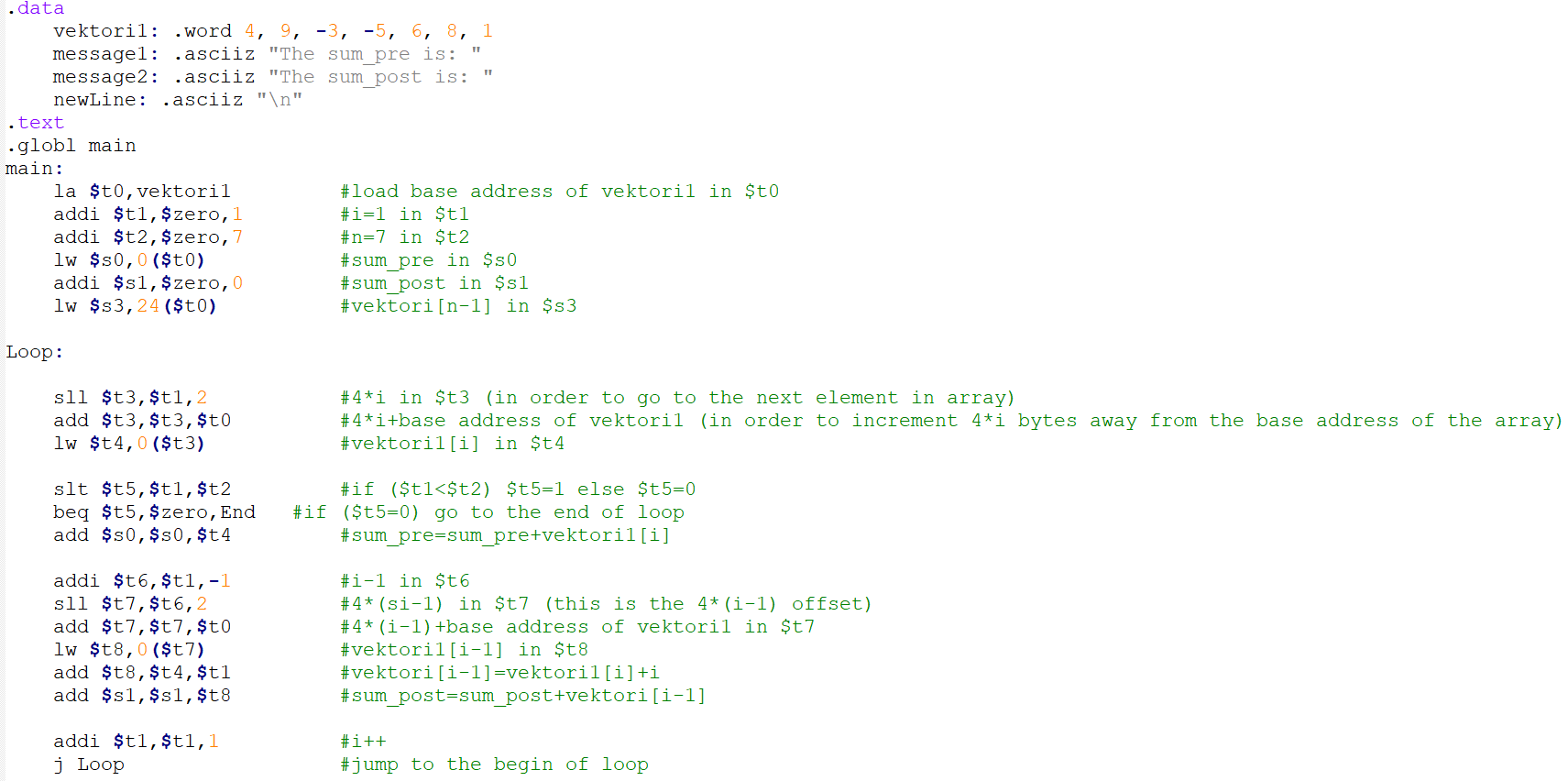


Figura 2. Pjesë e kodit e shkruar në MIPS assembler

Në rastin kur i bëhet e barabartë me n, programit i kemi thënë të kërcejë tek End, përkatësisht të dalë nga unaza. Kur bëhet ky hap në $s1 (ku ndodhet sum\_post) vendoset shuma mes vlerës që ndodhet aktualisht në $s1 dhe elementit të fundit të vektorit që e kemi vendosur në $s3.

Pastaj duke shikuar instruksionet përkatëse për printimin e stringjeve dhe numrave të plotë në libër, i kam shkruar instruksionet për printimin e dy mesazheve (të deklaruara te direktiva .data) dhe rezultatit të dy shumave. Në fund poashtu, nën etiketën Exit kam paraqitur instruksionet që shënojnë fundin e programit.

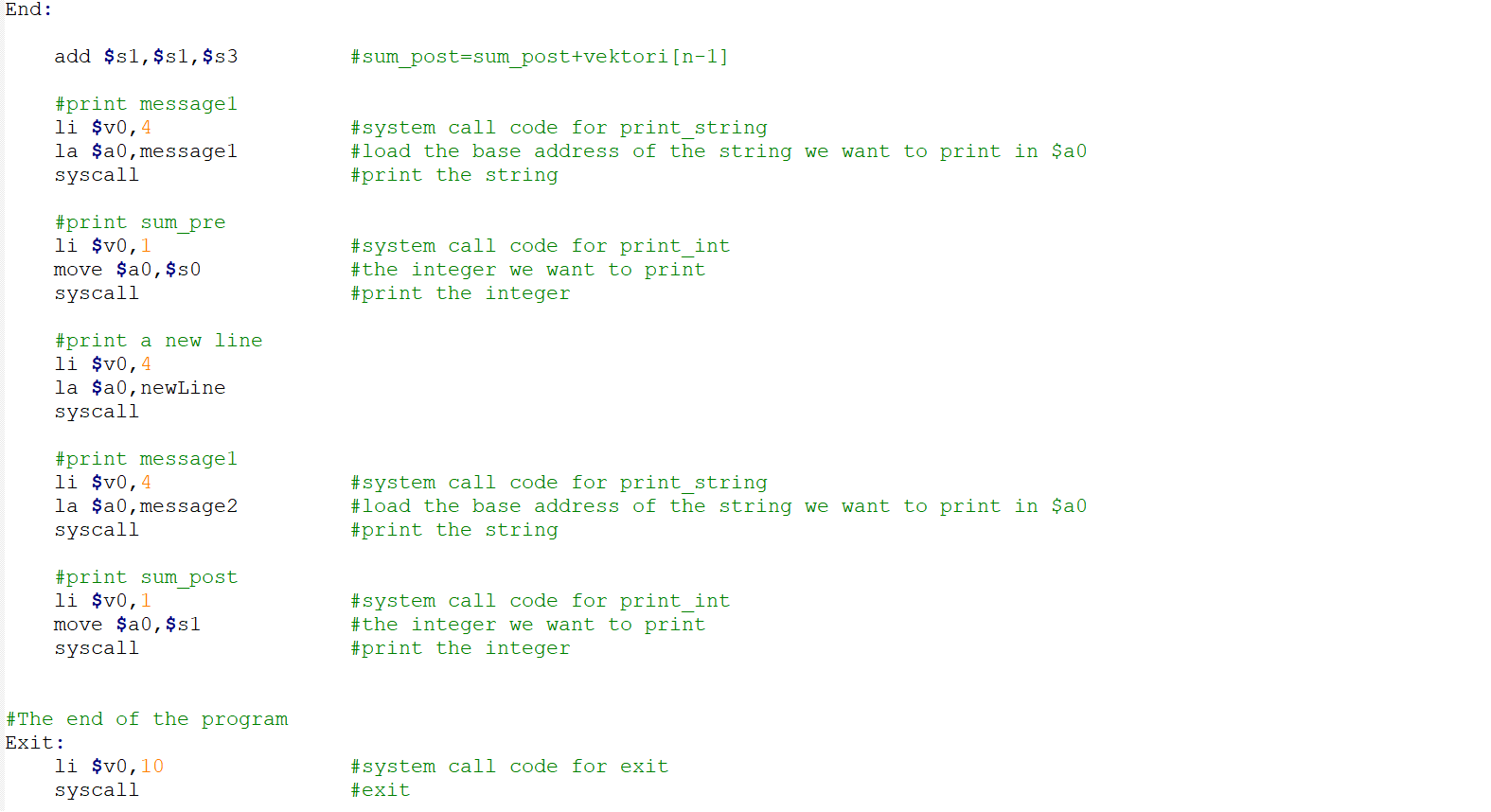


Figura 3. Vazhdimi i kodit të shkruar në MIPS assembler

## Testimet me QtSpim

Në këtë pjesë kam paraqitur një figurë të ekzekutimit të programit final me anë të simulatorit QtSpim. Rezultati i paraqitur në ekran, pas ekzekutimit të programit në QtSpim është i njëjti me rezultatin që është shfaqur kur e kam ekzekutuar kodin e shkruar ne C++.

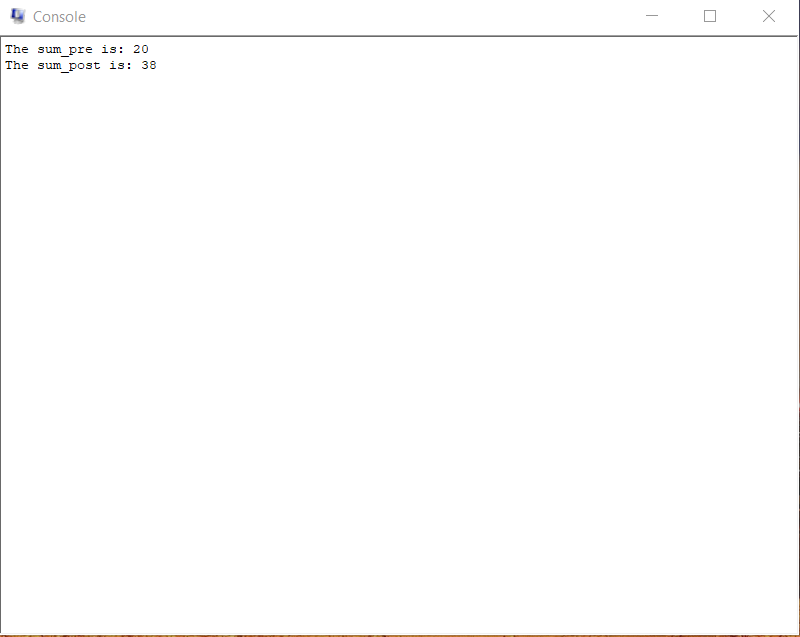


Figura 4. Rezultati i shfaqur në console pas ekzekutimit të programit