|  |  |  |
| --- | --- | --- |
|  | **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 A

## Hyrje

Kodi i mëposhtëm shërben për sortimin e elementeve të një vektori në C++. Funksioni populloVektorin shërben për marrjen e numrit të anëtarëve të vektorit, dhe pastaj me anë të një unaze for, merr elementet e vektorit nga user-i. Ky funksion kthen madhësinë e vektorit në fund. Funksioni unazaKalimit pranon si parametra vektorin a dhe madhësinë n. Në këtë funksion deklarohet variabla min që merr vlerën e anëtarëve të vektorit, (fillimisht do marr anëtarin e parë si vlerë) dhe variabla loc që merr vlerën e iteruesit. Pastaj e thërrasim funksionin unazaVlerave. Ky funksion pranon si parametra vektorin,n, p, min dhe loc. Dy variablat min dhe loc i dërgohen si referenca në mënyrë që ndryshimet që bëhen në to brenda funksionit unazaVlerave, t’i përcillen edhe brenda f-ionit unazaKalimit. Ky f-ion gjen elementin më të vogël. Fillimisht në min gjendet anëtari i parë i vektorit, ky do krahasohet me të gjithë anëtarët tjerë (përmes unazës for krahasohet me të gjithë anëtarët), dhe nëse ndonjëri nga ta është më i vogël, atëherë atë vlerë ia japim variablës min, kurse vlerën e iteruesit ia asociojmë variablës loc. Te unazaKalimit, pasi e kemi gjetur elementin më të vogël, me anë të unazaVlerave, fillimisht anëtarin e parë e vendosim në një variabël temporary. Pastaj në anëtarin e parë e vendosim minimumin (a[loc]), kurse vlerën e ndodhur në temporary e vendosim te a[loc]. Ky proces do të përsëritet përderisa iteruesi është më I vogël se n-1 (për shkak se elementin e fundit do të jetë sortuar). Pasi të dalim nga kjo unazë, brenda f-ionit unazaKalimit me anë të një unaze, I shfaqim elementet e sortuara të vektorit. Në funksionin main do t’i thërrasim f-ionet populloVektorin dhe unazaKalimit. Deklarojmë një vektor dhe variabël n (vlerën e merr nga funksioni populloVektorin) dhe pastaj ja dërgojmë si parametra f-ionit unazaKalimit.

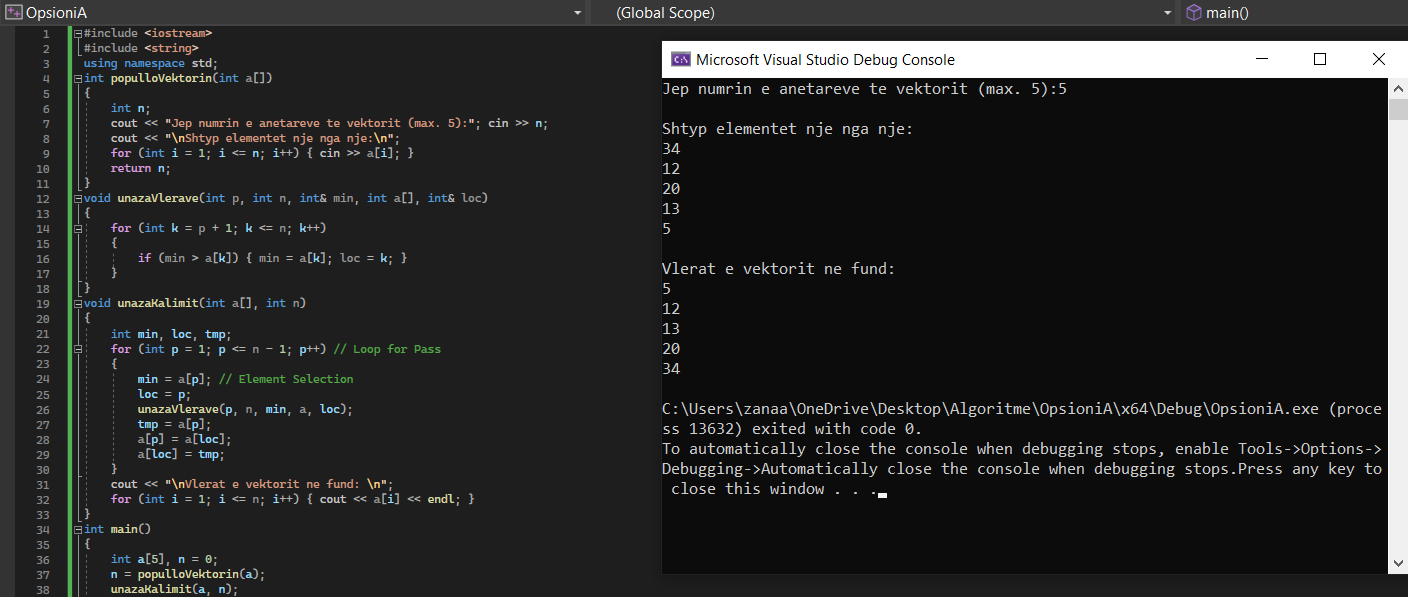


Figura 1. Kodi i shkruar në C++ dhe rezultati i shfaqur në console pas ekzekutimit të programit

## Realizimi i kodit në MIPS

Nën direktivën .data kam rezervuar hapësirë 20 bytes për 5 elementet e vektorit, tri mesazhe, dhe një karakter qw krijon hapësirë mes elementeve kur t’i paraqes në console. Nën pjesën .text gjendet i tërë kodit. Etiketa .globl main shërben për t’i treguar programit që funksioni kryesor është main. Tek main me anë të instruksionit jal (jump and link, e përdorim këtë instruksion meqenëse jump vetëm kërcen te një etiketë e caktuar, kurse jump and link përpos që kërcen te një etiketë e caktuar, e ruan edhe adresën se ku do kthehet pasi të përfundojë instruksionet përkatëse) e kam thirrur procedurën populloVektorin dhe unazaKalimit.

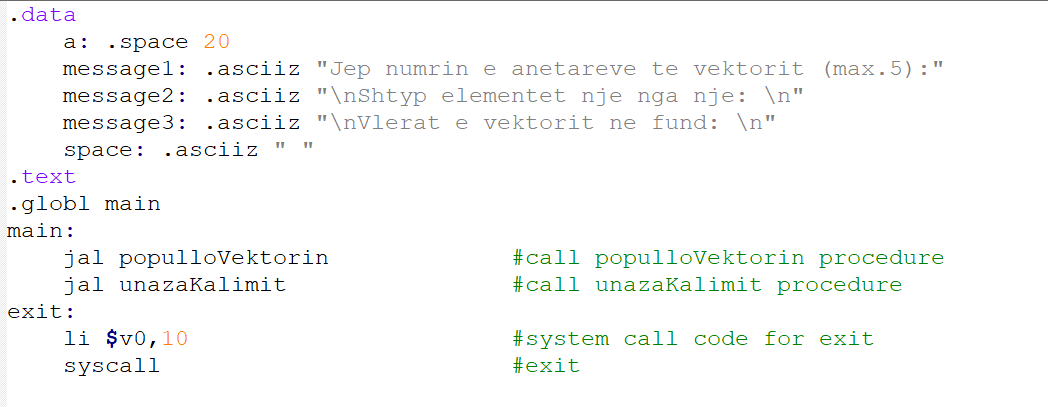


Figura 2. Pjesë e kodit e shkruar në MIPS assembler

Procedura populloVektorin përdoret për marrjen e madhësisë dhe elementeve të vektorit. Fillimisht kam shkruar instruksionet për shfaqjen e mesazhit të parë me etiketën message1, që kërkon nga user-i të jep madhësinë e vektorit. Këtë e kam bërë duke përdorur instruksionet përkatëse për printimin e teksteve (dmth duke shikuar system call codes për printimin e stringjeve nga libri, shtojca A). Pastaj kam shfaqur instruksionet për marrjen e input-it të userit, vlerë e cila ruhet në $v0, por unë e kam lëvizur në regjistrin $a1 ($a0-$a3 regjistra që i përdorim kur dëshirojmë t’i dërgojmë parametra procedurave.), sepse do ta përdori më vonë. Pastaj e kam shfaqur mesazhin e dytë që i kërkon user-it të shtyp elementet e vektorit një nga një. Meqë do të marrin në maksimum deri 5 elemente e kam përdorur një unazë. Në $t1 kam vendosur vlerën 1 (me anë të instruksionit add-immediate që i takon I-formatit, mund ta përdorim për të mbledhur vlerën në një regjistër me një konstantë-vlerë immediate), që do ta përdori si iterues. Instruksioni i radhës kërcen te fundi i unazës (te etiketa endLoop) në rastin kur iteruesi i bëhet më i madh se n. Përndryshe kur ky kusht nuk plotësohet programi vazhdon me instruksionet e rradhës për të marrur elementet e vektorit nga përdoruesi. Vlerat që i jep përdoruesi me anë të instruksionit sw (store word) vendosen në vektor në indeksin që e kam llogaritur si 4\*i (me anë të shift left logical-mund të përdoret për shumëzim me 2^i), për të kaluar nëpër secilin element të vektorit. Pastaj sigurohemi që të rrisim vlerën e iteruesit për 1, dhe të kërcejmë te fillimi i unazës (j Loop), në mënyrë që të marrim të gjitha elementet përderisa i është më e vogël dhe e barabartë me n. Kur ky kusht nuk plotësohet programit I kemi thënë të kërcejë në fund të unazës, me ç’rast aty gjendet instruksioni jr $ra për kthim te caller(main).

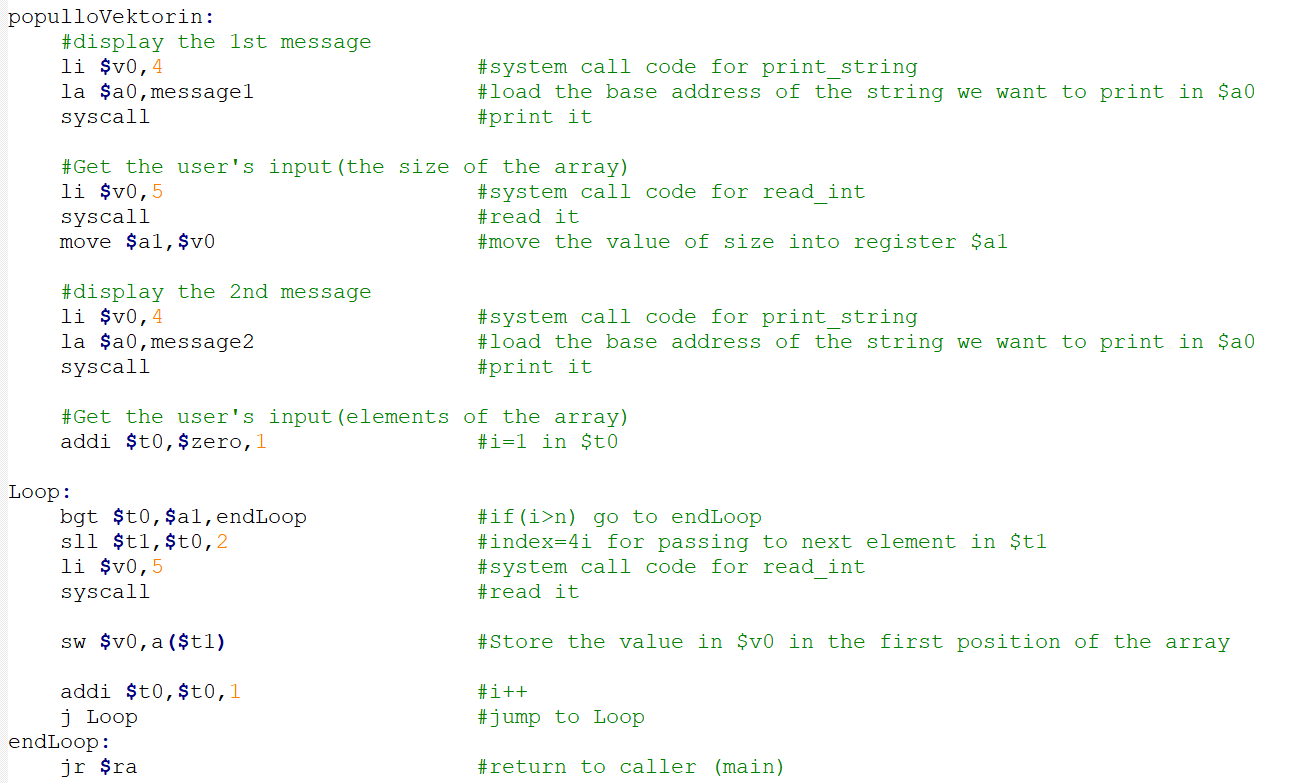


Figura 3. Vazhdimi i kodit të shkruar në MIPS assembler

Pasi jemi kthyer te main, ai ka ka thirrur procedurën unazaKalimit. Fillimisht këtu krijojmë hapësirë në stack për 5 vlera të ndodhura në regjistra përkatës. Return adresen do ta ruaj në stack, për shkak se këtu kam përdorur nested procedure, pra brenda kësaj procedure kam thirrur një procedurë tjetër. Në këtë rast duhet ta ruajmë vlerën e saj fillimisht në stack (që të mund të kthehemi te main në fund), si dhe pasi që të kryhen instruksionet në unazaVlerave, programi të kthehet te unazaKalimit përsëri. Siç e dimë, regjistrat $s0-$s7 janë callee save, që do të thotë nëse gjatë procedurës i përdorim këta regjistra, duhet fillimisht t’i vendosim në stack, ashtu që vlera e tyre të mos mbishkruhet nga callee (procedura), dhe në fund të rikthehen prapë vlerat e tyre në regjistrat fillestarë. Në këtë shembull unë nuk i kam përdorur këta regjistra në main, por sidoqoftë në raste kur kodi është edhe më i gjatë mund të mos kemi regjistra mjaftueshëm. Me anë të së i ruajmë vlerat e tyre në stack.

Pastaj në $t2 ruhet vlera e një iteruesi (p), kursë në $t3 ruhet vlera n-1. Pastaj kam shkruar instruksionin që në rastin kur iteruesi bëhet më I madh se n-1, programi të kërcejë në fund të unazës. Në të kundërtën programi vazhdon me instruksionet: në regjistrin $a1 vendoset elementi i vektorit në indeksin 4p (për të kaluar nëpër secilin element të vektorit), që fillimisht do ketë elementin e parë (pra, fillimisht supozojmë që elementi I parë është min). Vlerën e iteruesit p e kam bartur në regjistrin $a3 (loc=p). Këta regjistra I përdora meqë do t’ia dërgojë si parametra procedurës unazaVlerave. Me anë të jal kam kaluar te unazaVlerave (tani jal e ruan adresën ku do të kthehet te procedura unazaKalimit).

Te unazaVlerave në regjistrin $t5 kam vendosur një iterues (k) që ka vlerën 2 (k=p+1). Pastaj brenda unazës kam llogaritur indexin si 4k që në fakt I bië se do fillojë nga anëtari i dytë (meqë do ta krahasojmë minimumin [(fillimisht me vlerën e anëtarit të parë) me të gjithë anëtarët tjerë]. Pastaj kemi kushtin në rast që k bëhet më e madhe se n, ne dalim nga unaza. Përndryshe vazhdojmë me instruksionet e rradhës. Në $t7 kam vendosur vlerën e anëtarit në indeksin 4k (a[k]). Nëse vlera në këtë regjistër është më e vogël se vlera në $a2 (min), atëherë regjistri $t8=1 else $t8=0(këto I kam krahasuar duke përdorur set less than). Nëse vlera në $t8 është zero, programi shkon te etiketa rriteIndeksin,me ç’rast e rrit iteruesin dhe kërcen te fillimi I unazës. Përndryshe nëse vlera në $t8 është 1, vlerën në a[k] ia asociojmë minimumit (me anë të lë), kurse vlerës në $a3 (loc) ia bëjmë vlerën sa e ka iteruesi k (me anë të pseudoinstruksionit move). Pastaj rritet iteruesi dhe kërcejmë të fillimi I unazës, që ta krahasojmë minimumin me secilin anëtarë të vektorit. Në rast kur nuk plotësohet kushti I deklaruar në fillim, programi shkon në fund të unazës, ku kemi instruksionin për kthim te caller (unazaKalimit).

Pasi të kthehemi te procedura unazaKalimit kam llogaritur indeksin 4loc. Vlerën në a[p] fillimisht e kam vendosur në një regjistër $s0, kurse atë të vektorit në indeksin 4loc (ku në fakt do ndodhet elementi më i vogël i unazës) në regjistrin $s1. Pastaj vlerën e a[loc] e kam bartur me anë të store word në a[p], kurse vlerën që e ka pasur fillimisht a[p] (e ruajtur në $s0) e kam vendosur në a[loc]. Pastaj e kam rritur iteruesin (p++) dhe me anë të j Loop1 kam kërcyer te fillimi I unazës në mënyrë që ky proces të përsëritet për secilin element. Në rastin kur p bëhet më e madhe se n-1, programit I kemi thënë të kërcejë në fund të unazës. Aty e kam shfaqur mesazhin e fundit,që I tregon përdoruesit vlerat e vektorit në fund pas sortimit (këtë pjesë poashtu e kam bërë duke përdorur instruksionet përkatëse duke I shikuar system call code për printim të stringjeve nga libri-shtojcaA).

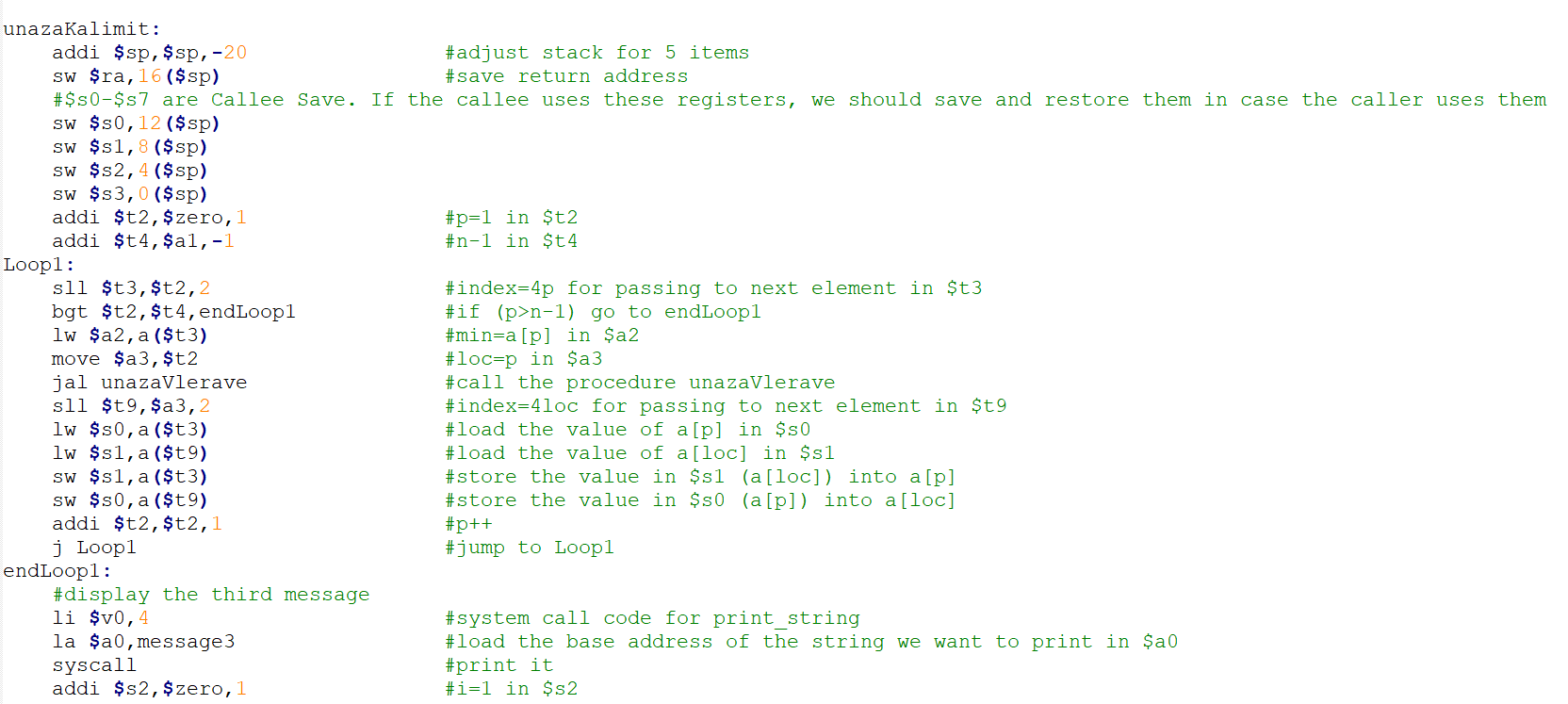


Figura 4. Vazhdimi i kodit të shkruar në MIPS assembler

Për t’I shfaqur elementet e vektorit duhet të përdorim një unazë (Loop3). Paraprakisht në regjistrin $s2 kam vendosur vlerën 1, që do ta përdori si iterues. Pastaj brenda unazës kam llogaritur një indeks që do na ndihmojë për kalim nëpër secilin anëtarë të vektorit (4i). Përderisa I nuk bëhet më e madhe se n (me anë të branch if greater than, nëse vlera i në $s2 bëhet më e madhe se n që është në $a1, kërcejmë në fund të unazës) do të shfaqen elementet e vektorit. Për shfaqjen e tyre kam përdorur instruksionet duke shikuar system call code për printimin e numrave të plotë nga libri. Poashtu kam përdorur edhe karakterin për krijimin e hapësirës mes elementeve të vektorit të paraqitur nën etiketën .space në pjesën e direktivës data. Pastaj kam rritur iteruesin dhe kam kërcyer në fillim të unazës(te etiketa Loop3) për të printuar secilin element në console. Kur kushti nuk plotësohet dalim nga unaza ku ia kthejmë regjistrave $s0-$s3 dhe $ra vlerën fillestare që e patëm ruajtur në stack. Në fund e kthejmë edhe stack pointer-in në vendin e tij paraprak. Pastaj me anë të jr $ra kthehemi te caller (main). Aty do ekzekutohen instruksionet për exit të programit. Pra, nën etiketën exit kam paraqitur instruksionet që shënojnë fundin e programit (poashtu duke shikuar system call code per exit nga libri-shtojca A).

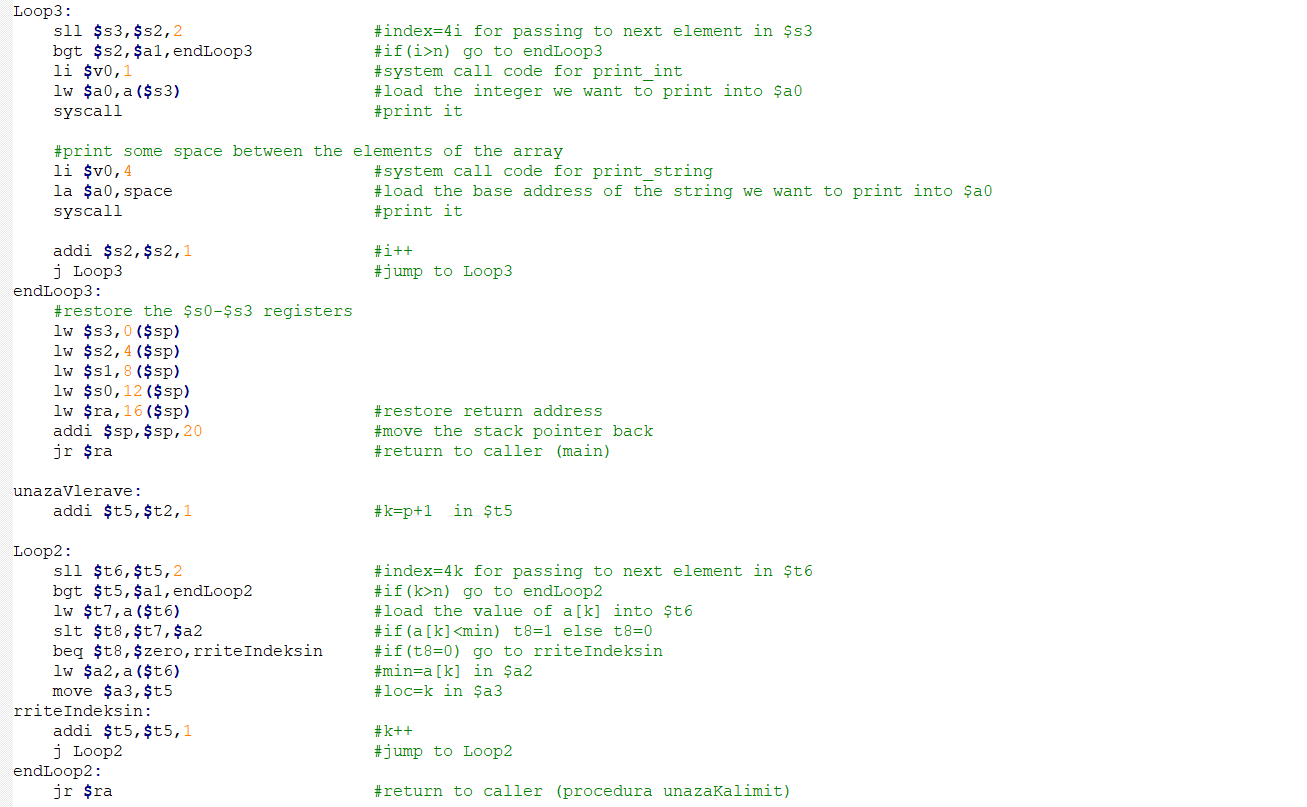


Figura 5. Vazhdimi i kodit të shkruar në MIPS assembler

## Testimet me QtSpim

Në këtë pjesë kam paraqitur një figurë të ekzekutimit të programit final me anë të simulatorit me QtSpim. Rezultati i paraqitur në ekran, pas ekzekutimit të programit në QtSpim është i njëjtë me rezultatin që është shfaqet kur ekzekutohet kodi i shkruar ne C++.

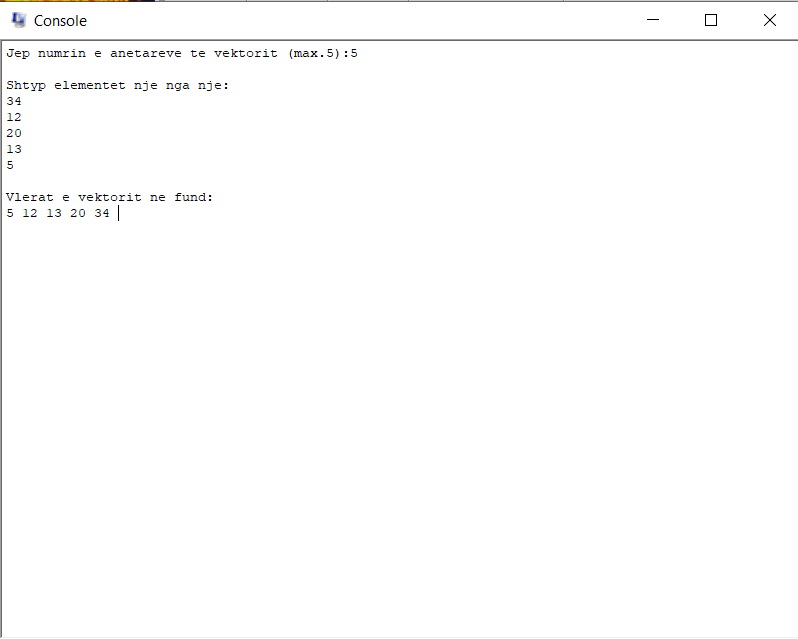


Figura 6. Rezultati i shfaqur në console pas ekzekutimit të programit