|  |  |  |
| --- | --- | --- |
|  | **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 B

## Hyrje

Në programin e mëposhtëm me anë të një funksioni (me tri parametra) janë kryer disa operacione, kurse në funksionin kryesor main janë deklaruar dhe inicializuar dy vektorë me nga 6 anëtarë dhe është thirrur funksioni për llogaritjen e operacioneve përkatëse me vektorë, me ç’rast dhe është printuar rezultati i operacionit. Funksioni operacioniMeVektore ka tre parametra (dy parametra si vektorë, dhe një variabël n, që paraqet madhësinë e vektorëve). Brenda funksionit janë deklaruar dy variabla temp1 dhe temp2, si dhe një variabël për llogaritje të shumës (fillimisht ia kemi asociuar vlerën 0). Me anë të një unaze for mund të kalohet nëpër anëtarë të vektorëve. Në variablat temp1 e temp2 ruhen vlera e anëtarit të vektorit1 në pozitën i dhe vektorit2 në pozitën (n-1-i), respektivisht. Pastaj, në vektorin1 në pozitën i, vendoset shuma e temp1 e temp2, kurse në vektorin2 në pozitën (n-1-i) vendoset zbritja e tyre. Në vazhdim llogaritet shuma mes vlerës aktuale që ndodhet në variablën sum, me vlerën që e ruajtëm në vektorin1 dhe në vektorin2. Ky proces do përsëritet përderisa vlen kushti i<n. Në rastin kur ky kusht nuk plotësohet, dalim nga unaza dhe e kthejmë si rezultat shumën e llogaritur, e cila do printohet si rezultati në funksionin kryesor main.

Hapat që duhet t’i ndjekim për realizimin e kodit në MIPS janë:

1. Në pjesën .data deklarohen dy vektorë, dhe një tekst për paraqitjen e rezultatit.
2. Adresat fillestare të vektorëve dhe madhësinë e tyre i ruajmë në ndonjë regjistrat për argumente, sepse do t’i dërgohen procedurës si parametra.
3. Kërcejmë te procedura ku do kryhen operacionet përkatëse dhe si rezultat do kthehet shuma të cilën e vendosim në njërën nga regjistrat për kthim të rezultateve. Pastaj kthehemi te funksioni kryesor main (caller).
4. Pasi kthehemi në main, shkruajmë instruksionet për paraqitje të rezultatit.

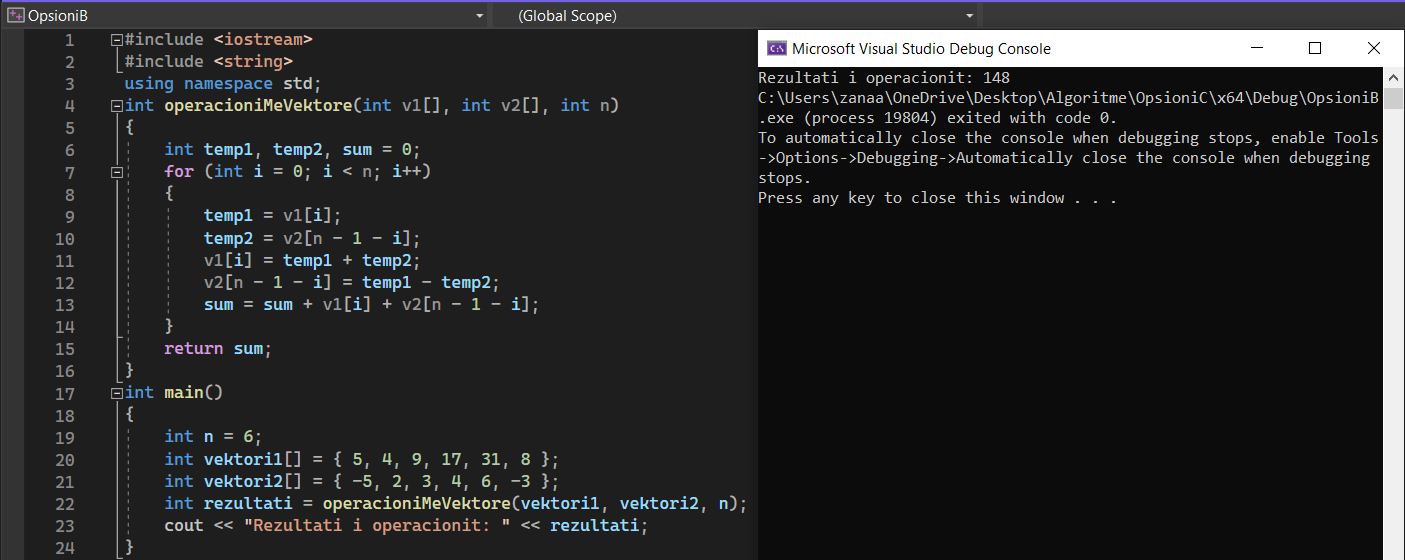


Figura 1. Kodi i shkruar në C++ dhe rezultati i shfaqur në console pas ekzekutimit të programit

## Realizimi i kodit në MIPS

Në pjesën .data kam deklaruar dy vektorë, si dhe një mesazh që do të përdoret kur të paraqesim rezultatin në console. Në direktivën .text gjendet i tërë kodi i programit. Nën etiketën main, me anë të instruksionit la (load address) kam vendosur adresën bazë te vektorit1 në regjistrin $a1, dhe adresën bazë të vektorit2 në regjistrin $a2. 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 6 në $a3 (madhësia e vektorëve). I kam përdorur këto regjistra ($a1, $a2, $a3) meqenëse këta regjistra i përdorim kur dëshirojmë t’i dërgojmë parametra procedurave. Pastaj 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 ktheht pasi të përfundojë instruksionet përkatëse) e kam thirrur procedurën operacioniMeVektore. Procedura i kryen instruksionet përkatëse dhe në fund e vendos rezultatin në regjistrin $v1 (regjistër për kthim të rezultateve nga procedura). Nën etiketën main me anë të instruksioneve përkatëse për printimin e teksteve dhe numrave të plotë (dmth duke shikuar system call codes për printimin e stringjeve dhe numrave të plotë nga libri, shtojca A), e kam printuar mesazhin e deklaruar në .data dhe rezultatin e llogaritur nga procedura. Në fund, nën etiketën Exit kam paraqitur instruksionet që shënojnë fundin e programit (poashtu duke shikuar system call code per exit nga libri-shtojca A).

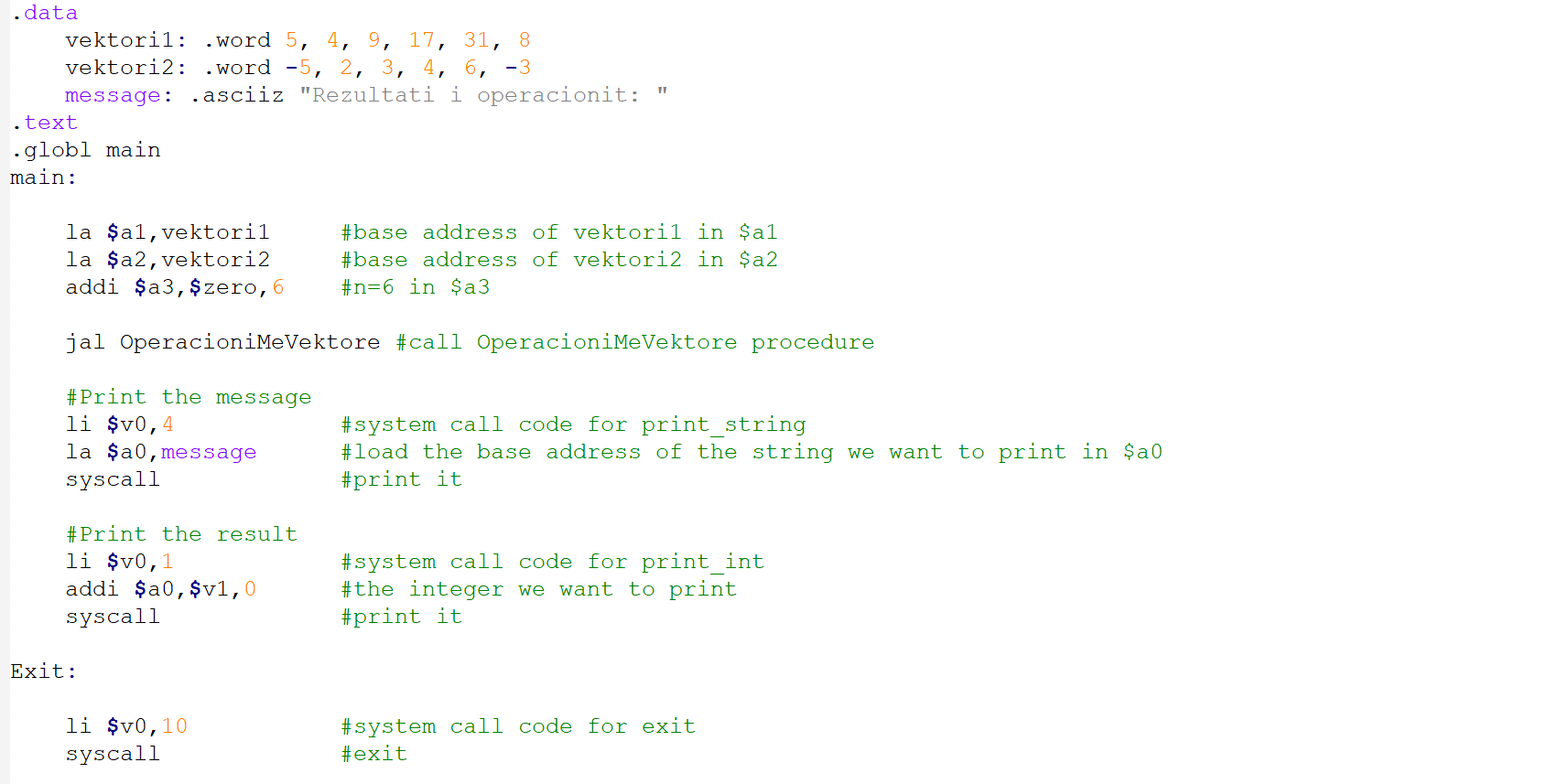


Figura 2. Pjesë e kodit e shkruar në MIPS assembler

Nën etiketën OperacioniMeVektore fillimisht me anë të addi në $t0 kam vendosur vlerën zero (që do ta përdori si iterator). 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 e kam përdorur regjistrin $s0 në main, por sidoqoftë në raste kur kodi është më i gjatë mund të mos kemi regjistra mjaftueshëm. Prandaj, fillimisht e vendosa në stack e pastaj bëra modifikimet në të. Për ta vendosur në stack, fillimisht stack pointer-in ($sp) duhet ta lëvizim për 4 byte, që të krijojmë hapësirë për vendosjen e $s0. Pastaj me anë të sw (store word-instruksion i formatit I që në këtë rast e merr vlerën në $s0 dhe e vendos në adresën e ku ndodhet $sp për momentin). Pas kësaj $s0 ja vendosa vlerën zero, që do ta përdori për vendosjen e shumës.

Nën etiketën Loop, me anë të slt (set less than) kam krahasuar vlerën e i me n. Në rastin kur i<n, vlera në $t8=1, përndryshe $t8=0. Me anë të beq (branch if equal) kam krahasuar vlerën në $t8 me zero, dhe nëse këto dy vlera janë të barabarta, programi kërcen te fundi i unazës, përkatësisht tek etiketa End. Përndryshe programi vazhdon me ekzekutimin e instruksioneve përkatëse me rradhë. 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 vektorin e parë. Me anë të instruksionit add (R-format) kam mbledhur 4\*i me adresën fillestare të vektorit1, që të lëvizim për 4\*i nga adresa bazë e vektorit1, dhe këtë vlerë e kam vendosur në regjistrin $t1. Duke përdorur instruksionin lw, kam vendosur vlerën që ndodhet në $t3 me një offset 0 (sepse dy instruksionet e mëparshme përkujdesen për kalim nëpër anëtarë të vektorit) në regjistrin $t2, përkatësisht kam vendosur vlerën e anëtarit në pozitën i(vekori[i]) në të.

Me anë të dy instruksioneve kam llogaritur vlerën (n-1-i), që do përdoret si offset për kalim nëpër anëtarë të vektorit2, dhe e kam vendosur në regjistrin $t3. Me sll, e kam shtyer për 2 (shumëzuar me 4) vlerën në $t3 dhe vendosur në $t4. Kam mbledhur 4\*(n-1-i) me adresën fillestare të vektorit2, që të lëvizim për 4\*(n-1-i) nga adresa bazë e vektorit2, dhe këtë vlerë e kam vendosur në regjistrin $t4. Duke përdorur lw, vlerën që ndodhet në $t4 me offset 0, e kam vendosur në $t5 (vektori[n-1-i]).

Duke përdorur pseudoinstruksionin move, vlerën në $t2 (vektori1[i]) e kam bartur në $t6, kurse vlerën në $t5 (vektori2[n-1-i]) e kam bartur në $t7. Mbledhjen e vlerave në regjistrat $t6 dhe $t7, e kam vendosur në $t2 (te vektori1[i]), kurse zbritjen (me anë të pseudoinstruksionit sub) e tyre në $t5 (te vektori2[n-1-i]). Shumën e kam llogaritur me anë të dy instruksioneve, fillimisht duke mbledhur vlerën aktuale që gjendej në të (në $s0) me vlerën në $t2 (vektori1[i] = temp1+temp2). Në instruksionin e dytë duke mbledhur vlerën në $s0 me vlerën në $t5 (vektori2[n-1-i] = temp1-temp2).

Së fundmi e rrisim vlerën e i-së dhe kërcejmë te fillimi i unazës (me anë të jump, kërcejmë te etiketa begin\_loop), në mënyrë që të përsëritet ky proces përserisa i<n. Në rastin kur i bëhet e barabartë me n, programit i kemi thënë të kërcejë tek End, përkatësisht të dalë nga unaza. Tek End, vlerën e shumës e bartim te regjistri $v1, regjistër ky që shërben për kthim të rezultateve. Në fund duhet të sigurohemi që vlerën që e kemi vendosur në stack pointer ta kthejmë në regjistrin përkatës, dhe stack pointer-in ta kthejmë në vendin e tij. Me anë të instruksionit jr $ra kthehmi te caller (te main).

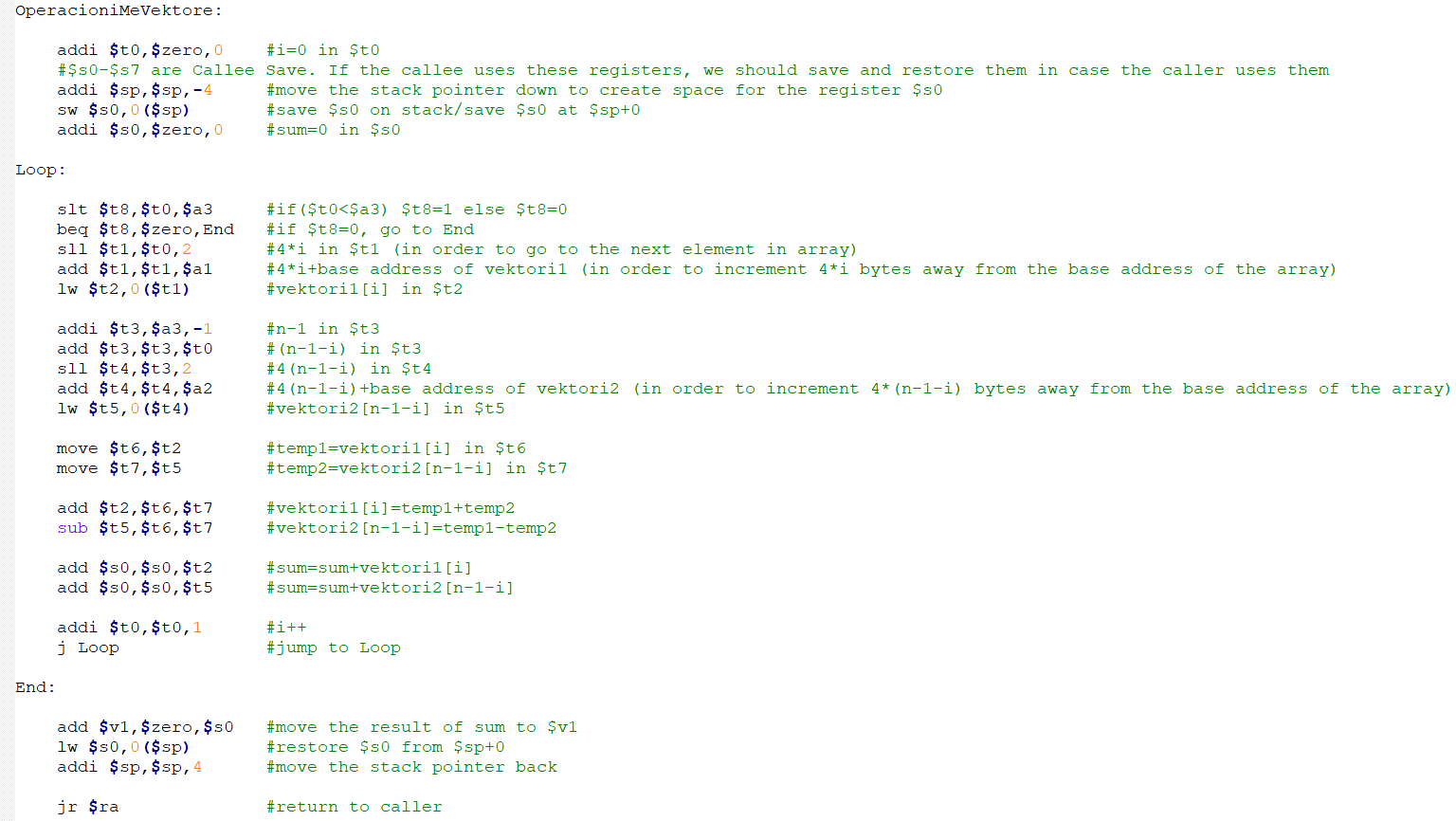


Figura 3. Vazhdimi i kodit të shkruar në MIPS assembler

## Testimet me QtSpim

Në këtë pjesë kam paraqitur një figurë të ekzekutimit të programit final me anë të simulatorit 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++.

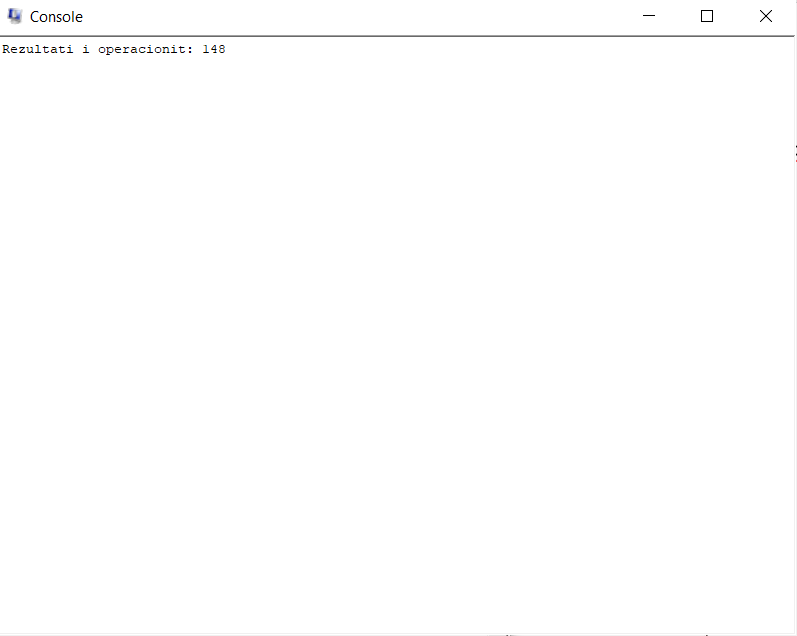


Figura 4. Rezultati i shfaqur në console pas ekzekutimit të programit