Universiteti i Prishtinës
Fakulteti i Inxhinierisë Elektrike dhe Kompjuterike
Arkitektura e Kompjuterëve, 2022/2023
Prof. Valon Raça & Synim Selimi

Emri Mbiemri: Florian Saqipi
ID: 210756100117

Detyra e parë: Opsioni A:

Hyrje

Opsioni i cili kam zgjedhur për këtë detyrë është opsioni A. Unë kam zgjedhur të punoj këtë detyrë, sepse implementimi i saj prej C++ në MIPS do të jetë mjaft interesant, për arsye se funksioni është implementuar në mënyre rekurzive, që na sjell disa sfida në MIPS. Pra, duhet kujdesur që parametrat të ruhen përgjatë thirrjeve rekurzive të funksionit po ashtu edhe adresat kthyese.

```cpp
// Factorial of n = 1*2*3*...*n

#include <iostream>
using namespace std;

int factorial(int, int);

int main() {
    int n, result, a = 1;

    cout << "Enter a non-negative number: ";
    cin >> n;

    result = factorial(a, n);
    cout << "Factorial of " << n << " = " << result;
    return 0;
}

int factorial(int a, int n) {
    if (n > a) {
        return a * n * factorial(a, n - 1);
    } else {
        return a;
    }
}
```

Figura 1. Kodi i opsionit A në C++

Kodi më lartë paraqet një funksion **factorial,** që gjen faktorielin në C++ me dy parametra **a** dhe **n**. Pra, fillimisht kemi deklaruar funksionin me dy parametrat e tij pastaj tri variablat tjera, ku variablën **a** e kemi inicializuar menjëherë me vlerën 1. Kërkojmë parametrin **n** nga përdoruesi. Pastaj thërrasim funksionin **factorial** me këta dy parametra i cili gjen vlerën e faktorielit rekurzivisht dhe pastaj printojmë rezultatin në ekran. Funksioni faktoriel do të thërras veten, përderisa **n > a**. Në momentin që ky kusht nuk vlen, funksioni kthen variablën **a** me vlerën **1**. Pastaj, ndodh shumëzimi në fazën kthyese të funksioneve rekurzive me varibalat **a** dhe **n**. Pra, në fund kthehet rezultati i faktorielit të kërkuar dhe printohet në ekran. Për implementimin nga një gjuhë e lartë si C++, në gjuhë të ultë si MIPS, shumë procese në C++, që janë të abstrahuara, në MIPS duhet kujdesur që të implementohen në mënyrë efiçiente.

Realizimi i kodit në MIPS

Te figura më poshtë është paraqitur implementimi në **MIPS** i pjesës së funskionit **main** në C++. Pra, shihet se shumica e veprimeve në C++ kërkojnë dy apo më shumë instruksione në MIPS.

```
main:
    li  $v0, 4          #  Vendosim 4 ne regjistrin $v0
    la  $a0, prompt     #  Vendosim ne regjistrin $a0 adresen prompt te stringut ne memorje statike qe
    syscall             #  System call qe ben printimin e nje stringu "Enter a non-negative number: "

    li $v0, 5           #  Vendosim 5 ne regjistrin $v0
    syscall             #  System call qe mer nje input nga tastiera
    move $t0, $v0       # Kopjojme vleren e regjistrit $v0 ne $t0 per perdorim te mevonshem, ($t0 = input) ($t0 = n)

    lw $a0, a           # Vendosim ne $a0 vleren ne adresen a pra numrin 1 ($a0 = a) (int a = 1)
    move $a1, $t0       # kopjojme ne regjistrin $a1 vleren e $t0 pra inputin nga useri ($a1 = n)

    jal factorial       # Pasi kemi pergatitur parametrat tani kercejmë ne etiketen factorial
                        # pra vendoset adresa e instruksionit te radhes ne $ra
    move $t1 , $v0      # Vendosim ne $t1 rezultatin e fituar nga funksioni qe gjendet ne $v0

    li  $v0, 4          # Printimi i stringut me adresen res "Factorial of "
    la  $a0, res
    syscall

    li  $v0, 1          # Printimi i vleres input nga perdoruesi ($t0 = input)($t0 = n)
    move $a0, $t0
    syscall

    li $v0, 4           # Printimi i stringut barazim " = "
    la $a0, equals
    syscall
    li $v0, 1           # Printimi i rezulatit qe kthen funksioni ne $t1 ($t1 =
    move $a0, $t1
    syscall
    li $v0, 4           # printo "\n" pra new line ne konsole
    la $a0, nl
    syscall
    li $v0, 10          # Exit program
    syscall
```

Figura 2. Implementimi në MIPS i pjesës së funksionit main në C++

Fillimisht kemi vendosur etiketën **main** ku do të kryejmë funksionalitetin që kryen funksioni main në kodin në C++. Së pari, kam bërë printimin e kërkesës nga përdoruesi (“Enter a non-negative number”). Pra këtë mesazh të kërkesës e kam vendosur në pjesën **.data** të kodit, pra në pjesën **static data** të memorjes. Pastaj, marim nga tastiera vlerën që vendos përdoruesi dhe e vendosim në regjistrin **$t0**, për përdorim të mëvonshëm, në C++ kjo do jetë variabla **n**.

Pastaj, për të përgatitur regjistrat e parametrave para thirrjes së funksionit, do vendosim në regjistrin për parametra **$a0,** vlerën që kemi në etiketën **a** (kam vendosur **a** në pjesen **.data**, pra **static data** pasi në detyrë në C++ është deklaruar dhe incializuar në të njëjtën kohë). Pra, **$a0** mer vlerën **1**, pra të variablës **a** në C++. Regjistri tjetër **$a1** do të merr vlerën e dhënë nga përdoruesi, që gjindet në regjistrin **$t0**, në C++ paraqet variablën **n**. Pasi kemi vendosur vlerat e parametrave në regjistrat e parametrave tani kërcejmë tek etiketa **factorial** ku llogaritet rezultati dhe e vendosim në regjistrat për vlerat kthyese të funksioneve, pra në **$v0** kemi rezultatin.

Pastaj, vendosim në regjistrin **$t1** rezultatin, që mos të humbet. Pastaj, do të shtypin pjesët e stringut që nevojiten për shfaqjen e rezultateve të funksionit dhe rezultatin e funskionit në regjisitrin **$t1.** Pjesa e fundit e kodit bën terminimin e programit.

```
        .data                                   # Static data nw memorje
prompt:     .asciiz "Enter a non-negative number: ";  # Pjeset e mesazheve tw nevojshme nw konzole
res:   .asciiz "Factorial of "                  # Keto vlera string kan etiketen e tyre dhe vendosen
equals:   .asciiz " = "                         # ne regjistrat ne printim me la (load adress) kur nevojiten
nl:     .asciiz "\n"
a:   .word 1                                    # kam vendosur a ne static data pasi ne detyre deklarohet dhe
                                                # incializohet nw tw njejtwn kohw pra => (int a = 1) vendoset ne $a0
```

Figura 3. Pjesa e vlerave të vendosura në **static data**

Këtu janë vendosur vlerat string të mesazheve të nevojshme për printim në konsolë. Ato kan etiketat apo adresat e tyre të cilat i vendosim në regjistra kur printojmë. Si dhe kam vendosur **a** me vlerë **1**. Pasi, ajo deklarohet dhe inicializohet njëkohësisht në C++.

```
factorial:
    slt $t2, $a0, $a1   # kontrollojme per $a0 < $a1, apo ne detyre (a < n)
    beq $t2, $zero, else # Nese $t2 = 0 atehere $a0 nuk eshte me e vogel $a1 bejme kercim ne else
                         # pra nuk kemi thirrje funksioni rekurzive
    addi $sp, $sp, -12  # alokojme 12 bytes ne stack per parametrat dhe adresen kthyese
    sw  $a0, 0($sp)     # vendosim $a0 ne stack (mund edhe te mos vendoset pasi nuk ndryshon)  ($a0 = a)
    sw  $a1, 4($sp)     # vendosim $a1 ne stack pasi ndryshon me thirrje funksionesh ($a1 = n)
    sw  $ra, 8($sp)     # vendosim $ra ne stack per te ruajtur adresen kthyese pas thirrjeve rekurzive
    addi $a1, $a1, -1   # dektremetojme parametrin ne regjistrin $a1 ($a1 = n), n=n-1

    jal factorial       # therrasim ne menyre rekurzive funskionin pasi kemi ruajtuar parametrat e adresat kthyese
    lw $a0, 0($sp)      # vendosim ne $a0 vleren e parametrit te rezervuar ne stack
    lw $a1, 4($sp)      # vendosim ne $a1 vleren e parametrit te rezervuar ne stack
    lw $ra, 8($sp)      # vendosim ne adresen kthyese qe kemi ruajtur ne stack
                        # pra vlerat para thirrjes rekurzive gjenden ne stack
    mult $v0, $a0       # shumezojme $v0 me $a0 pra (a * rezultati)
    mflo $v0            # e vendosim vleren nga regjistri special ne $v0
    mult $v0, $a1       # pastaj shumezojme me $a1  (n * rezultati)
    mflo $v0            # e vendosim ne $v0

    addi $sp, $sp, 12   # dealokojme haspiren ne stack
    jr $ra              # dhe kthehemi tek adresa kthyese ne $ra

else:                   # Ne etiketen else hyjme nese $a0 nuk eshte me e vogel se $a1
    move $v0, $a0       # atehere ne $v0 ruajme vleren e $a0 ( $a0 = a)
    jr $ra              # dhe kthehemi tek adresa kthyese
```

Figura 4. Implementimi në MIPS i funksionit factorial në C++

Këtu gjendet implmentimi në MIPS i kodit të funksionit factorial. Së pari, kemi krahasimin ekuivalent me krahismin **if (n > a)** në C++. Kam vendosur të kontrolloj nëse **$a0 < $a1** pra **a < n => n > a**, dhe pastaj **$t2** do merr vlerën **1**, kodi do vazhdojë pa kërcim deri te thirrja rekurzive. Përndryshe, nëse ka vlerën **0** do kercejmë tek etiketa **else** sepse **$a0** $\not<$ **$a1** pra **a** $\not<$ **n => n** $\not>$ **a**. Pra nëse nuk kemi kërcim te etiketa else, vazhdon ekzekutimi i kodit.

Rezervimin e hapsirës në stack e kam bërë pas kushtit dhe jo para tij, sepse nuk kemi nevojë të fusim në stack parametrat përveç nëse kemi thirrje rekurzive, kemi bwrw pra optimizim. Pra, kam rezervuar 12 bytes në stack. Nga 4 bytes për **$a0**(në C++ **a**), **$a1**(në C++ **n**) dhe **$ra**. Parametrat **$a0**, **$a1** dhe adresa

kthyese në **$ra** para thirrjeve rekurzive duhet të futen në stack, për arsye se përgjatë thirrjeve rekurzive ndryshojnë vlerat e tyre. **Megjithatë regjistri $a0 (në C++ a) nuk ndryshon përgjatë thirrjeve rekurzive, prandaj $a0 për optimizim mund të mos futet në stack**. Unë e kam futur në stack për arsye se kemi thirrje rekurzive dhe qëllimi i detyrës mendoj se është ruajta e parametrave në stack përgjatë thirrjeve rekurzive. Pastaj, dekrementojmë parametrin në **$a1** (në C++ **n**) pra **n-1**.

Pastaj, pasi kemi futur parametrat dhe adresën kthyese në stack bëjmë thirrjen rekurzive deri sa mbrrijmë në kushtin kur kërcejmë në etiketën **else,** ku në **$v0**, ku ruhet rezultati i funksioneve, vendosim vlerën e **$a0** (në C++ **a**) dhe bëjmë kërcimin në adresën në **$ra**. Pastaj, pas thirrjeve rekurzive marim parametrat **$a0** dhe **$a1** si dhe adresën në **$ra** nga stack. Bëjmë shumëzimin me parametrat në **$a0** dhe me rezultatin në **$v0**. Nga regjistri special **LOW**, vendosim vlerën rezultuese në **$v0**, të njëjtin veprim e bëjmë për **$a1**. Pastaj, pasi kemi nxjerr parametrat e nevojshëm dhe adresën kthyese, dealokojmë hapsirën e rezervuar në stack për 12bytes dhe kërcejmë në adresën në regjistrin **$ra**. Rezultatin e faktorielit e kemi në regjistrin **$v0**.

Testimet me QtSpim

Testimet në QtSpim i kam bërë për vlerat negative dhe vlerën 0, për të cilat duhet kthyer vlera 1. Pastaj, edhe për vlerat pozitive, për të cilat pritet që të shfaqet faktorieli i atij numri të cilin përdoruesi e ka dhënë përmes tastierës në konsolë.

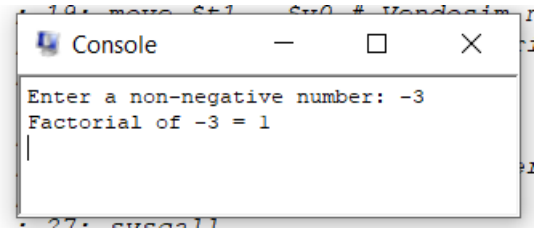

Figura 4. Rezultati i paraqitur në konsolë kur e japim një vlerë negative

Pra në figurën më lartë shihet se për çfardo vlere negative që japim përmes tastierës atëherë do kërcejmë në etiketën **else** dhe marim direkt vlerën e cila gjendet në regjistrin **$a0,** që është **1** (në C++ **a = 1**) dhe e vendosim si rezultat në regjistrin **$v0**. Andaj, pra na shfaqet neve vlera **1** si rezultat në konsolë.

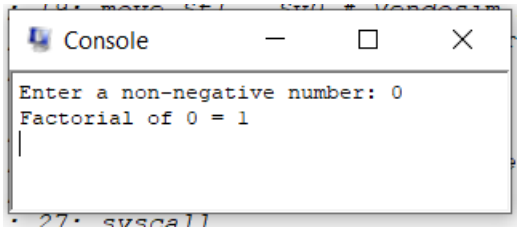

Figura 5. Rezultati i paraqitur në konsolë kur e japim vlerën 0

Pra në figurën më lartë shihet se për vlerën **0**, që e japim nga tastiera, në bazë të kushtit të vendsour te faktorieli do vazhdoj te etiketa **else** dhe ngjashëm si në rastin e kaluar, vendosim vlerën e regjistrit **$a0**, që është **1**, në regjistrin **$v0**. Pra, në regjistrin **$v0** kemi rezultatin. Andaj, shfaqet vlera **1** si rezultat në konsolë.

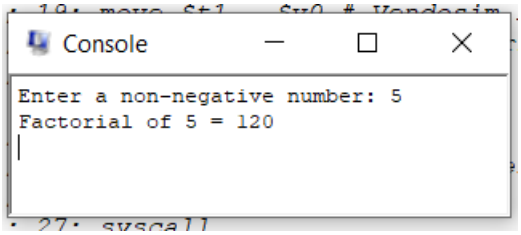

Figura 6. Rezultati i paraqitur në konsolë kur e japim një vlerë pozitive

Pra, në figurën më lartë shihet se për vlerën **5** të dhënë nga tastiera, rekurzivisht do të gjendet rezultati i faktorielit të atij numri me anë të shumëzimit. Pra, rezultati final do të gjendet në **$v0**. Në këtë rast për vlerën **5** në hyrje kemi fituar vlerën **120**.

Përfundimi

Si konkluzion, kjo detyrë ka qenë mjaft interesante si implementim në MIPS. Pasi shumica e hapave në C++ janë veq se të abstrahuar, kurse në MIPS kemi mundësinë, që secili prej këtyre hapave të kuptohet në detaje, pasi pra, MIPS është gjuhë e ulët. Me anë të kësaj detyre kemi përvetësuar njohuri të shumta në MIPS, nga formimi i kushteve, deri tek implementimi i një funksioni rekurziv. Pjesa e rekurzionit ka qenë shumë atraktive për mua, pasi parametrat dhe adresa kthyese duhet të futen në stack për arsye se duhet të ruhen vlerat përgjatë thirrjeve. Kjo na ka mundësuar që të kemi të qartë se si mund të punojmë me stack në memorje.

Implementimin e kodit në MIPS jam munduar ta bëj në mënyrë sa më të saktë me kodin e dhënë në C++, ka optimizime të mundshme të cilat nuk i kam zbatuar për qëllim të saktësisë. Pra, detyra ka qenë shumë atraktive, interesante dhe një mundësi shumë e mirë për aftësim të mëtejshëm në MIPS.