

UNIVERZITET U TUZLI
FAKULTET ELEKTROTEHNIKE



Arhitektura računara

Zadaća 2

Tuzla, april/travanj 2024.

Sadržaj

Sadržaj	2
Zadatak 1	2
Zadatak 2	3
Zadatak 3	4
Zadatak 4	5
Zadatak 5	6
Zadatak 6	6
Zadatak 7	7
Zadatak 8	7

Zadatak 1

Napisati funkciju `foo` i `main` u MIPS assembleru.

```
short foo(short a, short b) {
    if (a < 2) {
        return 0;
    } else if (b < 3) {
        return 1;
    }
    if (a < b) {
        return a + foo(a - 1, b - 3);
    } else {
        return a - b + foo(a - 2, b - 1);
    }
}

int main(int argc, char* argv[]) {
    int i, j;
    for (i = 2; i < 6; ++i) {
        for (j = 3; j < 7; ++j) {
            printf("foo(%d, %d), %hd\n", i, j, foo(i, j));
        }
    }
    return 0;
}
```

Output:

```
|foo(2, 3) = 2|foo(2, 4) = 2|foo(2, 5) = 2|foo(2, 6) = 2|foo(3, 3) =
0|foo(3, 4) = 4|foo(3, 5) = 4|foo(3, 6) = 5|foo(4, 3) = 2|foo(4, 4)
= 2|foo(4, 5) = 5|foo(4, 6) = 4|foo(5, 3) = 3|foo(5, 4) = 1|foo(5,
5) = 4|foo(5, 6) = 7
```

Zadatak 2

Napisati funkciju `combinations` koja uzima set karaktera (`chars`), veličinu seta (`size`) te broj ponavljanja (`r`). Funkcija treba da ispiše nizove dužine `r` koji predstavljaju sve kombinacije datog karaktera seta. Poziv funkcije iz `main`-a:

```
int main(int argc, char const* argv[]) {
    char set[] = {'a', 'b'};
    combinations(set, 2, 2);
    return 0; }
```

Za karakter set { 'a', 'b' } i $r = 3$, funkcija treba da ispiše:

aaa
aab
aba
abb
baa
bab
bba
bbb

Za karakter set { 'a', 'b', 'c' } i $r = 1$, funkcija treba da ispiše:

a
b
c

Implementacija funkcije combinations u fajlu combinations.S:

```
static char* d;

void combinations(char* chars, uint32_t size, uint32_t r) {
    d = malloc(r);
    combinations_impl(chars, r, size, r);
}

static void combinations_impl(char* chars, uint32_t maxr,
                              uint32_t size, uint32_t r) {
    if (r == 0) {
        print_comb(maxr);
        return;
    }
    for (int i = 0; i < size; ++i) {
        d[maxr - r] = chars[i];
        combinations_impl(chars, maxr, size, r - 1);
    }
}

static void print_comb(uint32_t maxr) {
    for (int i = 0; i < maxr; ++i) {
        printf("%c", d[i]);
    }
    printf("\n");
}
```

Zadatak 3

Početkom devedesetih 3D video-igre su bile u svojim začecima. U to vrijeme hardverska ograničenja računara su bila najveća prepreka do kreiranja koliko toliko interesantnih igara. U toku igranja neke 3D video-igre, računanje udaljenosti se obavljalo na milijarde puta svake sekunde, samim time potrebno je milijarde puta u sekundi odraditi normalizaciju vektora što je u suštini:

$$\frac{1}{\sqrt{x}}$$

Operacija korjenovanja i djeljenja je bila poprilično skupa za hardver u to doba. Programer po imenu John Carmack je napravio naizgled magičnu funkciju koja radi ovu operaciju bez dijeljenja i korjenovanja. Naravno, rješenje predstavlja dovoljno blisku aproksimaciju ove operacije. Više detalja možete pročitati na [Wikipedia stranici](#).

Funkcija u kodu Quake engine-a se zvala `Q_rsqrt` (Quake Reverse Square Root) i njen prepisan transkript (zajedno sa komentarima) se nalazi ispod:

```
float Q_rsqrt( float number )
{
    long i;
    float x2, y;
    const float threehalfs = 1.5F;

    x2 = number * 0.5F;
    y  = number;
    i  = * ( long * ) &y;    // evil floating point bit level hacking
    i  = 0x5f3759df - ( i >> 1 );    // what the fuck?
    y  = * ( float * ) &i;
    y  = y * ( threehalfs - ( x2 * y * y ) );    // 1st iteration
    // y  = y * ( threehalfs - ( x2 * y * y ) );    // 2nd iteration,
                                                // this can be removed

    return y;
}
```

Napisati funkciju `Q_rsqrt` u MIPS assembly-u. Testirati njen poziv iz main-a te provjeriti koliko je aproksimacija bliska stvarnim vrijednostima koristeći kalkulator.

Zadatak 4

Implementirati funkciju `bar` u MIPS assembly-u. Koristiti `round`, `ceil`, `floor` i `trunc` instrukcije. Testirati njen poziv iz main-a napisanog u C-u.

```
float bar(float f1, float f2, double d1) {  
  
    if (f1 < f2)  
        return 2.f * ceil(f1) - round(f2) + trunc(d1);  
    else if (f1 == f2) {  
        return ceil(f1) - floor(f2) + trunc(d1);  
    }  
  
    return d1;  
}
```

Zadatak 5

Implementirati funkciju `foo` u MIPS assembly-u. Testirati njen poziv iz main-a napisanog u C-u.

```
int foo(double d1, float f1, float f2) {  
    if (d1 > f2) {  
        return (f1 + f2 - d1) / 3.;  
    }  
  
    if (d1 == f2 && d1 != f1 && d1 != 0.) {  
        return (f1 * 3.) / d1;  
    }  
  
    if (d1 < f2 && d1 == f1 && f1 != 0.) {  
        return abs(d1 - f2) / f1;  
    }  
  
    return f1 * f2 - 12. * d1;  
}
```

Zadatak 6

Implementirati funkciju baz u MIPS assembly-u. Testirati njen poziv iz main-a napisanog u C-u.

```
double baz(float f1, double* p1, float f2, int n) {
    if (*p1 == 0. || (f1 + f2 < n)) {
        return f1 + f2 * n;
    }

    if (*p1 != 0.) {
        return (f1 - f2 * n) / *p1;
    }

    return f1 * *p1 + n + f2;
}
```

Zadatak 7

Implementirati funkcije mymax a nakon toga i main u MIPS assembly-u.

```
double mymax(double start, const double* nums, uint32_t size) {
    for (int i = 0; i < size; ++i) {
        if (nums[i] > start)
            start = nums[i];
    }

    return start;
}

int main(int argc, char const* argv[]) {
    const double nums[] = {1., 5.7, 9.4, 14.8, 7.1, 8.2, 10.5};

    printf("MyMax: %f\n", mymax(nums[0], nums, 7));

    return 0;
}
```

Zadatak 8

Implementirati funkcije `roundToTick`, `getTick` i `calculateAmount` i MIPS assembleru.

```
void roundToTick(double* p, uint32_t precision) {
    uint32_t r = *p * precision;
    *p = (double)r / precision;
}

double getTick(double num) {
    if (num == 0)
        return 0.;
    if (num < 10)
        return 0.001;
    if (num < 100)
        return 0.01;
    if (num < 1000)
        return 0.1;
    return 1.;
}

double calculateAmount(double price, int quantity) {
    roundToTick(&price, 1 / getTick(price));
    return price * quantity;
}

int main(int argc, char const* argv[]) {
    for (double i = 3.451748; i < 4000.; i *= 10.) {
        printf("Amount(%f, 307) = %f\n", i, calculateAmount(i, 307));
    }
    return 0;
}
```