UNIVERZITET U TUZLI
FAKULTET ELEKTROTEHNIKE

Arhitektura računara

Zadaća 1

Tuzla, mart/ožujak 2025.

Sadržaj

Zadatak 1

Napisati program u MIPS assembly-u koji učitava broj sa memorijske lokacije `number`, ispituje da li je broj paran ili neparan koristeći striktno bitwise operatore te rezultat smješta na memorijsku lokaciju `result`. Obje memorijske lokacije trebaju biti u `data` ili `rodata` sekciji programa.

Zadatak 2

Napisati program u MIPS assembly-u koji radi sljedeće operacije nad cijelim brojem.

1. Ispituje da li je broj paran ili neparan. Rezultat treba da bude `1` ili `0`.
2. `broj x 22`
3. `broj x 7`.
4. `broj % 8`
5. `broj % 32`

Broj se nalazi na memorijskoj lokaciji `number`, a rezultat smjestiti na memorijsku lokaciju `result`. Obje vrijednosti trebaju biti u `data` ili `rodata` sekciji programa. Dovoljno je koristiti samo bitwise operatore.

Zadatak 3

Napisati program koji resetuje bite `0, 14, 27 i 31` registra `t0` te rezultat upisuje u povratnu vrijednost programa.

Zadatak 4

Napisati program u MIPS assembly-u koji će naći median niza 16 bitnih cijelih brojeva. Ukoliko niz ima neparan broj elemenata, median je srednji element. U suprotnom median izračunati kao aritmetičku sredinu dva srednja elementa (naravno koristeći cjelobrojno djeljenje) i rezultat smjestiti na memorijsku lokaciju `result`. Prije računanja provjeriti da li je niz sortiran. Ukoliko nije na memorijsku lokaciju result upisati vrijednost: `0xDEADFA11`. Veličinu niza učitati sa lokacije `size`. Niz i veličina niza trebaju biti u `data` ili `rodata` sekciji programa.

Na primjer, za sljedeći niz: `1, 19, 350, 740, 1522` median je `350`
Za niz: `1, 19, 350, 740, 1522, 2020` median je `545`.

Zadatak 5

Napisati program u MIPS assembly-u koji učitava null terminirani niz karaktera sa memorijske lokacije str te provjerava da li je niz palindrom. Rezultat (`1` ili `0`) upisati na memorijsku lokaciju result.

Zadatak 6

Napisati `exchange` funkciju u MIPS assembly-u čiji potpis treba da izgleda kao:

```
int16_t exchange(int16_t* a, int16_t b);
```

Funkcija treba da na memorijsku lokaciju na koju pokazuje prvi argument upiše vrijednost drugog argumenta. Povratna vrijednost funkcije treba da bude stara vrijednost na memorijskoj lokaciji prvog argumenta.

Testirati poziv funkcije iz main-a napisanog u C-u, nakon toga iz main-a napisanog u asembleru.

Zadatak 7

Napisati swap funkciju u MIPS assembly-u čiji potpis treba da izgleda kao:

```
void swap(void*, void*,int32_t);
```

Funkcija treba da zamjeni vrijednosti na memorijskim lokacijama na koju pokazuju prva dva argumenta. Količina podataka (u byte-ima) je data trećim argumentom. Pretpostaviti da su date memorijske lokacije iste dužine.

Za dva niza:
```
int32_t n1[10] = {1,3,5,7,9,-9,-7,-5,-3,-1};
int32_t n2[10] = {0,2,4,6,8,-8,-6,-4,-2,0};
```

Poziv funkcije treba da izgleda kao:

```
swap(n1, n2, 10 * sizeof(int32_t));
```

Nakon poziva funkcije nizovi n1 i n2 trebaju zamijeniti vrijednosti potpuno.

Testirati poziv funkcije iz main-a napisanog u C-u, nakon toga iz main-a napisanog u asembleru.

Zadatak 8

Napisati funkciju `toBinary` u MIPS assembly-u koja uzima pointer na `buffer` te 32 bitni cijeli `broj`. Funkcija treba da `broj` pretvori u njegovu binarnu reprezentaciju i upiše u `buffer` (u vidu niza karaktera: `‘1’, ‘0’,’1’,’1’...`). Osim toga, funkcija treba da ispiše na ekran ovaj buffer nakon što je konverzija izvršena te koliko je iteracija petlje bilo potrebno da se broj pretvori u binarni. Funkcija kao povratnu vrijednost nazad vraća broj iteracija koji je petlja odradila. (Obratiti pažnju da buffer mora biti `null` terminiran). Dovoljnu veličinu buffera treba da osigura caller funkcije.

Testirati poziv funkcije iz main-a napisanog u C-u, nakon toga iz main-a napisanog u asembleru.

Zadatak 9

Napisati transform funkciju u MIPS asembleru čiji potpis treba da izgleda kao:

```
void transform(char* buff, int32_t size, char (*predicate)(char));
```

Funkcija treba da modifikuje (transformira) proslijeđeni buffer tako da nad svakim članom pozove predicate funkciju i njen rezultat smjesti nazad na istu lokaciju unutar buffer-a.

Primjer poziva funkcije:

```
char c1[12] = “Hello world”;

void transform(char* buff, int32_t size, char (*predicate)(char));

char mytoupper(char c) {
  if (c > 96 && c < 127) return c - 32;
  return c;
}

int main(int argc, char* argv[]) {
  transform(c1, 12, mytoupper);
  printf("C1: %s\n", c1);
  return 0;
}
```