Uvod u organizaciju i arhitekturu računara 2 - I smer - JUN 1

Na Desktop-u se nalazi direktorijum **ar_PrezimeIme_alasNalog_grupa**. Preimenujte taj direktorijum tako što ćete umesto `PrezimeIme` i `alasNalog` navesti svoje prezime, ime i korisničko ime na studentskom serveru Alas, tim redom, za grupu stavite g2. Npr. student Marko Marković sa nalogom mi21123 preimenovaće direktorijum u ar_MarkovicMarko_mi20123_g2
U tom direktorijumu treba da smestite sve programe koje predajete. Potpis funkcija koje pišete moraju biti isti kao što je navedeno u zadatku! Ne treba praviti poddirektorijume za zadatke. Kada završite sa radom, zatvorite sve otvorene aplikacije i pozovite dežurnog asistenta da preuzme rad. Predviđeno vreme za rad je 3 sata.

1. **INLINE** [10p]
   Prilikom obrade tekstova različitim metodama iz oblasti obrade prirodnog jezika vrlo često reči prolaze kroz razne transformacije. Jedna od čestih transformacija je podela reči na n-grame (podnizove od n karaktera) i prebrojavanje koliko se puta n-gram nalazi u nekoj reči. U fajlu **1.c** data je `main` funkcija kao i veći deo implementacije funkcije **count_n_gram** koja broji pojavljivanja n-grama u reči. Potrebno je u inline asembleru implementirati zakomentarisani deo funkcije.
   Primeri korišćenja:

   Primer 1
   ```
   Ulaz:
     aaa
     a
   Izlaz:
     3
   ```

   Primer 2
   ```
   Ulaz:
     abcdabcdab
     abc
   Izlaz:
     2
   ```

   Primer 3
   ```
   Ulaz:
     abababa
     aba
   Izlaz:
     3
   ```

2. Na asemblerskom jeziku za **ARM_32** arhitekturu napisati:

   a) [10p]
   Funkciju `int argmax(unsigned *a, int n)` koja u nizu neoznačenih celih brojeva `a` dužine `n` pronalazi indeks maksimalnog elementa. U slučaju da više brojeva ima istu vrednost prednost dati manjem indeksu (pogledajte primer 2). Asemblerski kod sačuvati u datoteci **2a.s**

   Primeri korišćenja:

   Primer 1
   ```
   Ulaz:
     5
     1 2 3 4 5
   Izlaz:
     4
   ```

   Primer 2
   ```
   Ulaz:
     5
     1 2 5 5 5
   Izlaz:
     2
   ```

   Primer 3
   ```
   Ulaz:
     6
     3 5 2 4 6 1
   Izlaz:
     4
   ```

   b) [10p]
   Funkciju `void argmaxes(unsigned **A, int n, int m, int *args)` koja za svaki niz dužine `m` u nizu nizova `A` dužine `n` pronalazi indeks maksimalnog elementa i upisuje ga u niz `args`. Asemblerski kod sačuvati u datoteci **2b.s**

   Pomoć: koristite funkciju koju ste implementirali u delu a).

   Primeri korišćenja:
   Prvo se unosi broj nizova (`n`) zatim dužina jednog niza (`m`) i nakon toga elementi nizova. Izlaz su izračunati indeksi maksimalnih elemenata.

   Primer 1
   ```
   Ulaz:
     2 3
     1 2 3
     3 2 1
   Izlaz:
     2 0
   ```

   Primer 2
   ```
   Ulaz:
     3 4
     8 2 7 0
     4 3 5 2
     1 6 6 2
   Izlaz:
     0 2 1
   ```

   Primer 3
   ```
   Ulaz:
     4 4
     1 2 3 4
     4 1 2 3
     3 4 1 2
     2 3 4 1
   Izlaz:
     3 0 1 2
   ```

3. Na asemblerskom jeziku za **X86_64** arhitekturu napisati

a) [10p]
Funkciju `int power(int n, int m)` koja računa n^m za ceo broj `n` i prirodan broj `m`. Pretpostaviti da rezulata staje u tip `int` bez prekoračenja. Asemblerski kod sačuvati u datoteci **3a.s**

Primeri korišćenja:

Primer 1
```
Ulaz:
 5 2
Izlaz:
 25
```

Primer 2
```
Ulaz:
 -17 3
Izlaz:
 -4913
```

Primer 3
```
Ulaz:
 11111 0
Izlaz:
 1
```

b) [10p]
Funkciju `int find_power(int n, int k, int *ms, int nm)` koja u niz `ms` pronalazi indeks elementa m za koji važi $k = n^m + 1$. Ukoliko takav element ne postoji funkcija vraća -1. Asemblerski kod sačuvati u datoteci **3b.s**

Pomoć: koristite funkciju koju ste implementirali u delu a).

Primeri korišćenja:
Prvo se unose `n` i `k`, zatim broj elemenata niza `ms` i na kraju elementi tog niza. Izlaz je rezultat implementirane funkcije.

Primer 1
```
Ulaz:
 5 26
 3
 4 1 2
Izlaz:
 2
```

Primer 2
```
Ulaz:
 -5 -124
 5
 8 5 2 3 4
Izlaz:
 3
```

Primer 3
```
Ulaz:
 2 4
 5
 7 2 1 3 4
Izlaz:
 -1
```