MIKRORAČUNARI - ISPIT - NOVEMBAR '09

1. (30 poena) Napisati *IA-32* asemblersku funkciju:

```
void fractions(int f1[], int f2[], int r[]);
```

koja sabira dva razlomka. Svaki razlomak dat je nizom celih brojeva dužine 2, pri čemu je element na poziciji 0 brojilac, a element na poziciji 1 imenilac razlomka. Razlomci koji se sabiraju dati su nizovima $f1$ i $f2$, dok rezultujući razlomak treba smestiti u niz r. Dobijeni zbir treba da bude "skraćen", tj. njegovi brojilac i imenilac treba da budu uzajamno prosti. Napisati potom i C-program koji učitava razlomke koji se sabiraju (najpre brojilac i imenilac prvog razlomka, a zatim brojilac i imenilac drugog razlomka), poziva funkciju i ispisuje njen rezultat na standardnom izlazu. Na primer, za ulaz:

```
4 15
7 20
```

izlaz treba da bude:

```
37 60
```

2. (30 poena) Napisati *IA-32* asemblersku funkciju:

```
int circles(int n, double * cr);
```

koja, koristeći paralelne *SSE2* instrukcije, za dati skup krugova u ravni ispituje koliko postoji parova krugova takvih da je jedan krug sadržan u drugom krugu. Krugovi su dati nizom na koji pokazuje cr, pri čemu se u nizu najpre nalaze x i y koordinata centra prvog kruga, a zatim i poluprečnik prvog kruga, nakon čega slede x i y koordinata centra drugog kruga, pa poluprečnik drugog kruga, itd. Broj krugova dat je parametrom n. Smatra se da je krug sadržan u drugom krugu čak i ako ga dodiruje sa unutrašnje strane. Napisati potom i C-program koji sa standardnog ulaza učitava broj krugova, a zatim podatke o krugovima (u gore opisanom redosledu), zatim poziva funkciju i na kraju ispisuje rezultat na standardni izlaz. Na primer, za ulaz:

```
5
1.0 1.0 2.0
0.1 0.1 1.0
2.0 1.0 1.0
2.5 3.5 3.0
1.0 4.5 5.0
```

izlaz treba da bude:

```
3
```

3. (40 poena) Napisati *ARM* asemblersku funkciju:

```
int unicode2utf8(unsigned unicode, unsigned char * utf8);
```

koja za datu vrednost *Unicode* karaktera formira niz *UTF-8* bajtova kojima se taj karakter kodira. Savremeni *Unicode* karakteri uzimaju kodne vrednosti u intervalu `0x000000-0x10FFFF` (najviše 21 bit u binarnom zapisu), a *UTF-8* kodiranje je jedno od najzastupljenijih vidova kodiranja koje svaku *Unicode* kodnu vrednost kodira sekvencom od 1, 2, 3 ili 4 bajta, u zavisnosti od opsega kome kodna vrednost pripada. Detalji načina kodiranja dati su u sledećoj tabeli:

Opseg *Unicode* kodova	Šema bitova *Unicode* koda	*UTF-8* kodiranje *Unicode* koda
`0x000000-0x00007F`	`0xxxxxxx`	`0xxxxxxx`
`0x00080-0x0007FF`	`0yyyyyxxxxxx`	`110yyyyy 10xxxxxx`
`0x00800-0x00FFFF`	`0zzzzyyyyyyxxxxxx`	`1110zzzz 10yyyyyy 10xxxxxx`
`0x010000-0x10FFFF`	`0wwwzzzzzzyyyyyyxxxxxx`	`11110www 10zzzzzz 10yyyyyy 10xxxxxx`

Vrednost *Unicode* karaktera data je parametrom *unicode* kao neoznačeni ceo broj, a rezultujuća sekvenca bajtova se smešta u niz neoznačenih *char*-ova na koji pokazuje $utf8$. Funkcija vraća dužinu sekvence bajtova. Napisati potom i *C*-program koji učitava vrednost *Unicode* karaktera (kao neoznačeni heksadekadni broj), a zatim poziva funkciju i ispisuje njen rezultat (opet u heksadekadnoj notaciji) na ekranu. Na primer, za ulaz:

```
0x1FAB
```

izlaz treba da bude:

```
0xE1 0xBE 0xAB
```