



Napredni C kurs


Namenski računarski sistemi

Čas 03, 2021/2022

Big i Little Endian

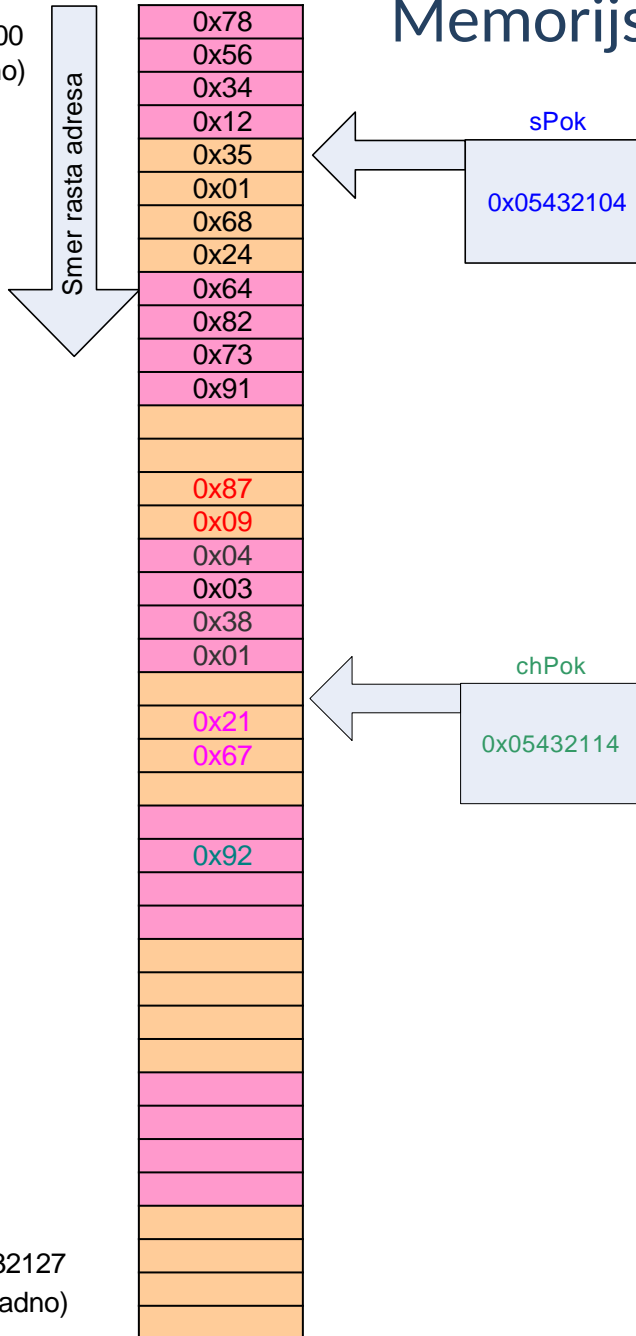
- Redosled slaganja byte-ova u memoriji
- BE: najviši zadnji
- LE: najniži zadnji
- Bi-Endian: obe mogućnosti
- Zavisno od HW platforme
 - Intel je LE
 - MIPS je BE (ili LE – opciono)
- Problem je prisutan i kod
 - Čitanja binarnih datoteka
 - Komunikacione razmene podataka
- Ne postoji sistemsko rešenje, tj mora se programirati konverzija podataka
- Ograničava prenosivost koda

```
char buf[4];  
int k = 0x01020304;  
memcpy( buf, &k, sizeof(k) );
```

	Little Endian		Big Endian
buf[0]	04		01
buf[1]	03		02
buf[2]	02		03
buf[3]	01		04
	Intel		MIPS

Memorijska slika – Little Endian

Adresa : 0x05432100
(88285440 dakadno)



long Niz[10];

Niz[0] = 0x12345678;

Niz[1] = 0x24680135;

Niz[2] = 0x91738264;

short *sPok = (short*)&Niz[1];

sPok[5] = 0x0987;

char *chPok = (char*)(sPok + 8);

chPok[1] = 0x21;

chPok[2] = 0x67;

*(chPok+5) = 0x92;

*(Niz + 4) = 0x01020304;

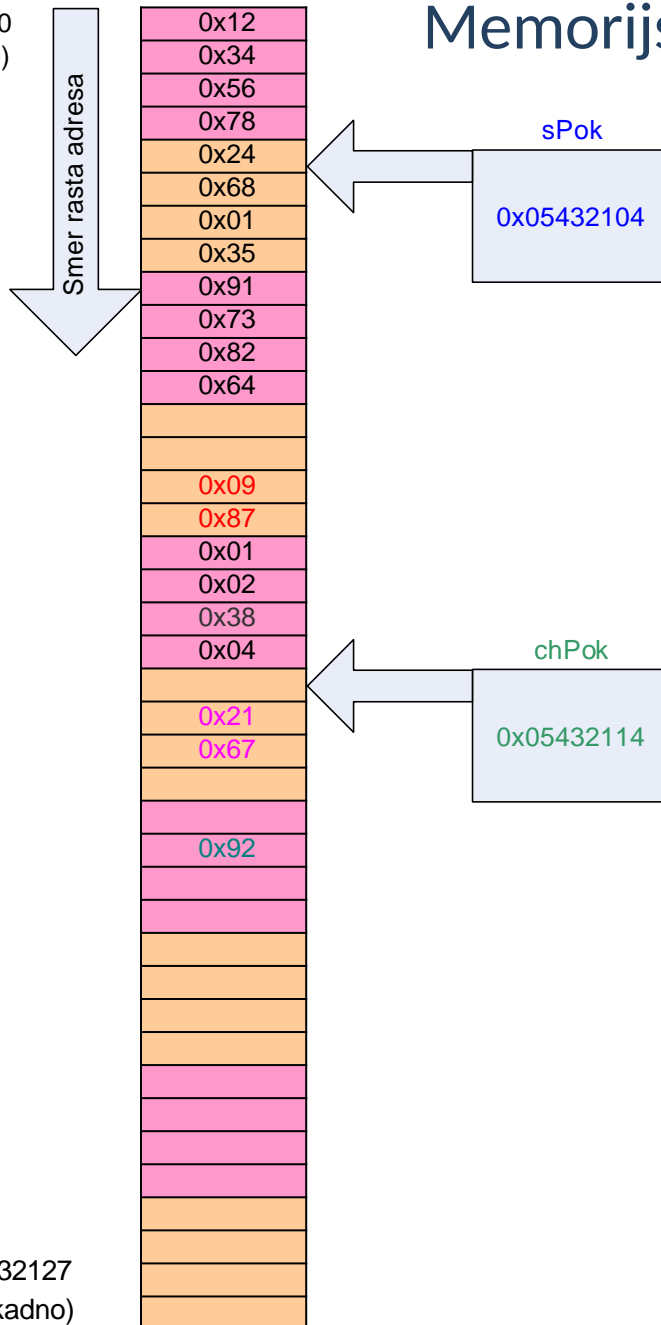
((char)Niz + 18) = 0x38;

Niz[4] == 0x01380304

Adresa : 0x05432127
(88285479 dakadno)

Memorijska slika – Big Endian

Adresa : 0x05432100
(88285440 dakadno)



Ugradjene funkcije za rad sa LE i BE

- Programski jezik C podržava velik broj funkcija za konvertovanje iz jednog zapisa u drugi. Na primer:
 - Funkcija `htonl()` pretvara broj tipa `long` u format za korišljenje u TCP-IP protokolu
 - Funkcija `htonf()` pretvara broj tipa `float` u format za korišljenje u TCP-IP protokolu
 - Funkcija `ntohl()` pretvara broj tipa `long` u format obrnut od formata za korišljenje u TCP-IP protokolu
 - Postoje još i `htond()`, `htonll`, `htohs()` i druge

Zadaci za vežbu

1. Napraviti sledeće funkcije:

`void copyLEToLE(int, void*)` – koja kopira broj dat u LE formatu na datu adresu u LE formatu

`void copyLEToBE(int, void*)` – koja kopira broj dat u LE formatu na datu adresu u BE format

Pitanje: Da li je za kopiranje iz BE u LE potrebno kreirati nove funkcije?

Odgovor: Ne. Prebacivanje iz BE u BE je isto kao i prebacivanje iz LE u LE, jer ne menja raspored bajtova. Prebacivanje iz BE u LE je isto kao i prebacivanje iz LE u BE, jer samo promeni raspored bajtova

Pitanje: Da li je neophodno praviti funkciju za tip podataka float?

Odgovor: Ne. Potrebno je samo sadržaj promenljive tipa float tretirati kao ceo broj, tj. njenu adresu kao adresu celobrojne promenljive

Primer: `copyLEToBE(*(int*) &float1), &float2);`

Zadaci za vežbu

2. Koristeći kreirane funkcije, kreirati funkciju Calculate, koja prima bafer od 10 bajtova, od čega
- a) nulti bajt predstavlja slovo 'L' ili 'B', koji označava da li je broj dat u LE ili BE formatu
 - b) prvi bajt predstavlja operaciju (+, -, *, -)
 - c) preostalih 8 bajtova predstavljaju dva broja tipa integer, nad kojim je potrebno realizovati operaciju.

U slučaju da je nulti bajt 'L', brojevi su dati u LE formatu, a ako je nulti bajt 'B', broj je dat u BE formatu. Povratna vrednost funkcije je tip podataka float koji sadrži rezultat u istom formatu.

Kreirati primer koji demonstrira rad kreiranih funkcija.

Zaglavlje funkcije Caclulate dato je sa:

```
float Calculate(char* buffer);
```

Zadatak za samostalnu vežbu

1. Napisati funkciju `Count()` koja prima bafer u kojem se nalaze dva niza od po `n` brojeva tipa `double` i broji koliko brojeva prvog niza je veće od broja u drugom nizu koji se nalazi na istoj poziciji. Nulti bajt bafera ima vrednost 'L' ukoliko su svi brojevi dati u LE formatu, a vrednost 'B' ukoliko su dati u BE formatu. Sledeća dva bajta su brojeve elemenata u svakom od dva niza, a zatim idu elementi prvaog niza, nakon čega idu elementi drugog niza. Kao rezultat fukcije vratiti broj tipa `int` u istom formatu kao i dati brojevi.

Zaglavlje funkcije `Count` dato je sa:

```
int Count(char* buffer);
```