Arhitektura računara

Stringovi i sistemski pozivi

Stringovi

- Nizovi znakova
- Kodiranje: ASCII
- Tip podataka: **.ascii**
- String se završava sa **NULL** znakom “**\0**”
  - C konvencija
- Primer definisanja stringa:

  string: .ascii “neki tekst\0”
- Prikaz sadržaja stringa u dibageru
  - *Data/Memory* dijalog

Primer: Izbacivanje razmaka sa kraja i početka stringa - C program

```
char s[] = “  neki tekst   \0”;
char *p = s;

// izbacivanje razmaka sa kraja stringa
while(*p)
    p++;
while(*(p-1) == ‘ ’)
    p--;
*p = 0;

// izbacivanje razmaka sa početka stringa
while(*s == ‘ ’) {
    p = s;
    while(*(p+1)) {
        *p = *(p+1);
        p++;
    }
    *p = 0;
}
```

Primer: Izbacivanje razmaka sa kraja stringa - asemblerski program

```
string: .ascii "  abcd   \0"
…
    movl $string, %eax
kraj_s:                 # nalaženje kraja stringa
    cmpb $0, (%eax)
    je razmaci
    incl %eax
    jmp kraj_s
razmaci:                # izbacivanje razmaka sa kraja
    cmpb $' ', -1(%eax)
    jne izbaci
    decl %eax
    jmp razmaci
izbaci:
    movb $0, (%eax)
kraj:
    movl $1, %eax
    movl $0, %ebx
    int $0x80
```

Sistemski pozivi

- Omogućuju pozivanje operacija operativnog sistema
- Pristup korišćenjem naredbe softverskog prekida, **int**, sa vektorom **0x80**
- Prenos parametara ide preko registara

Sistemski pozivi

- U registar **eax** treba postaviti **broj sistemskog poziva**
- U registre **ebx, ecx, edx, esi** i **edi** treba postaviti (eventualne) **argumente poziva**
- Nakon završetka poziva, registar **eax** sadrži **informaciju o završetku poziva**, pri čemu su vrednosti ostalih registara nepromenjene

Sistemski poziv za ulaz (read)

eax ← **3** (broj sistemskog poziva za ulaz)
ebx ← **0** (deskriptor fajla za stdin)
ecx ← **adresa ulaznog bafera**
edx ← **veličina bafera** (maksimalan broj karaktera koji korisnik može uneti)

- Uneti string se završava kodom za Enter, odnosno novi red (10 = 0xA)
  - pod uslovom da može da stane u ulazni bafer
- Nakon izvršenog sistemskog poziva, registar **eax** sadrži tačan broj unetih karaktera, računajući i karakter za novi red
  - taj broj će biti manji od **edx** ukoliko se unese manje od maksimalnog broja karaktera, ili će biti jednak **edx** ukoliko se unese broj karaktera veći ili jednak maksimalnom

Sistemski poziv za izlaz (write)

eax ← **4** (broj sistemskog poziva za izlaz)
ebx ← **1** (deskriptor fajla za stdout: za stderr ova vrednost je 2)
ecx ← **adresa izlaznog bafera**
edx ← **veličina bafera**

- Ovaj poziv ne raspoznaje “\0” i ispisaće onoliko znakova koliko se upiše u edx!

Sistemski poziv za završetak programa (exit)

eax ← **1** (broj sistemskog poziva za završetak programa)
ebx ← **izlazni kod (kod greške)**

- Kod greške je vrednost koja se prosleđuje operativnom sistemu kada program završi sa radom
  - iz bash-a se može dobiti komandom: **echo $?**

Primer 1: Unos stringa

```
str_max = 40
str: .fill str_max, 1, 42
…
    movl $3, %eax
    movl $0, %ebx
    leal str, %ecx
    movl $str_max, %edx
    int $0x80

    movl $1, %eax
    movl $0, %ebx
    int $0x80
```

leal

- Smešta adresu izvornog operanda u odredišni

```
leal str, %eax   # izvršavanje
movl $str, %eax  # kompajliranje

# adresa trećeg znaka
movl $2, %esi
leal str(, %esi, 1), %eax

#GREŠKA - ne postoji na x86
movl $str(, %esi, 1), %eax
```

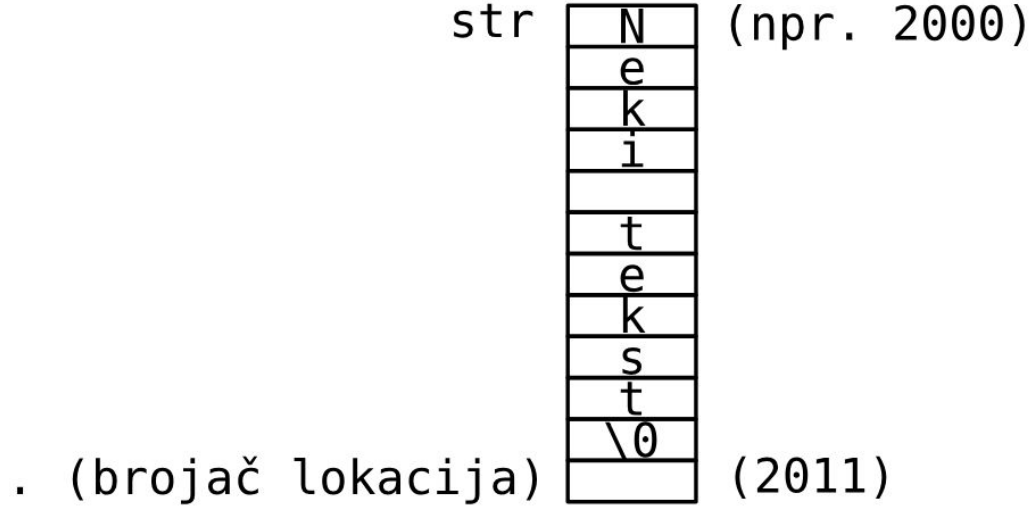
Primer 2: Ispis stringa

```
str: .ascii “Neki tekst\0”
str_len = 11
…
    movl $4, %eax
    movl $1, %ebx
    leal str, %ecx
    movl $str_len, %edx
    int $0x80

    movl $1, %eax
    movl $0, %ebx
    int $0x80
```

Primer 3: Ispis stringa

```
str: .ascii “Neki tekst\0”
str_len = . - str
…
    movl $4, %eax
    movl $1, %ebx
    leal str, %ecx
    movl $str_len, %edx
    int $0x80

    movl $1, %eax
    movl $0, %ebx
    int $0x80
```

Brojač lokacija



- Brojač lokacija postoji samo prilikom prevođenja (kompajliranja), pa se ovako može odrediti dužina samo unapred zadatih nizova (stringova)

Testiranje programa

- **Nije dovoljno isprobati program samo jednim test primerom!**
- Program treba testirati sa svim graničnim i više međugraničnih slučajeva, i to:
  - ulazne podatke koji treba da daju ispravan rezultat, i
  - ulazne podatke koji treba da izazovu grešku

Automatsko testiranje

- **./ testiraj.sh asm_program.S**
- Izvršiće seriju testova nad programom, i prikazati rezultate testiranja
- Nazive promenljivih koje već postoje u početnom .S fajlu ne treba menjati!
- Sadržaje stringova koji već postoje u početnom .S fajlu ne treba menjati!

Automatsko testiranje

- Automatsko testiranje ima smisla pokretati tek kada program proradi!
- Kod ovakvog testiranja treba obratiti pažnju na to da izlaz mora biti identičan traženom
  - u suprotnom neće biti prepoznat kao validan

Komentarisanje programa

- **Loše iskomentarisan kod**

```
main:                                   # početak
    movl $niz, %esi                     # adresa od niz u esi
    movl $NELEM-1, %edi                 # nelem-1 u edi
    movl $0, %eax                       # nula u eax

petlja:
    # dodaj (%esi, %edi, 4) na eax
    addl (%esi, %edi, 4), %eax
    decl %edi                           # smanje edi za 1
    jns petlja                          # skoci ako nije s

kraj:                                   # kraj programa
    movl %eax, suma                     # stavi eax u sumu
    movl $1, %eax                       # stavi 1 u eax
    movl $0, %ebx                       # stavi 0 u ebx
    int $0x80                           # pozovi int sa $0x80
```

Komentarisanje programa

- **Dobro iskomentarisan kod**

```
main:
    movl $niz, %esi       # registar za bazu
    movl $NELEM-1, %edi   # indeks poslednjeg elementa
    movl $0, %eax         # trenutna suma

petlja:    # od poslednjeg ka prvom elementu
    addl (%esi, %edi, 4), %eax
    decl %edi             # kada se desi edi<0, s postaje 1
    jns petlja

kraj:
    movl %eax, suma       # smeštanje sume u promenljivu
    movl $1, %eax         # exit sistemski poziv
    movl $0, %ebx
    int $0x80
```