

Asemblersko programiranje za x86_64 arhitekturu 3. deo

Mehanizmi pozivanja

potprograma

■ Prenos kontrole

- Na početak koda pozvane procedure
- Natrag u kod pozivaoca

■ Prenos podataka

- Parametri procedure
- Povratna vrednost funkcije

■ Upravljanje memorijom

- Dodela tokom izvršavanja potprograma (za lokalne promenljive)
- Dealokacija prilikom povratka iz potprograma

■ Ovi mehanizmi implementirani su mašinskim instrukcijama

```
P(...) {  
    •  
    •  
    y = Q(x);  
    print(y)  
    •  
}
```

```
int Q(int i)  
{  
    int t = 3*i;  
    int v[10];  
    •  
    •  
    return v[t];  
}
```

Mehanizmi pozivanja programa

P() {

Projektne odluke o ovim mehanizmima (koje se implementiraju mašinskim instrukcijama) sačinjavaju takozvani binarni interfejs aplikacije, engl.

Application Binary Interface (ABI).

Linux koristi System V ABI for AMD64
http://refspecs.linuxbase.org/elf/x86_64-abi-0.99.pdf

Windows koristi

<https://docs.microsoft.com/en-us/cpp/build/x64-calling-convention>

x86-64 Stek

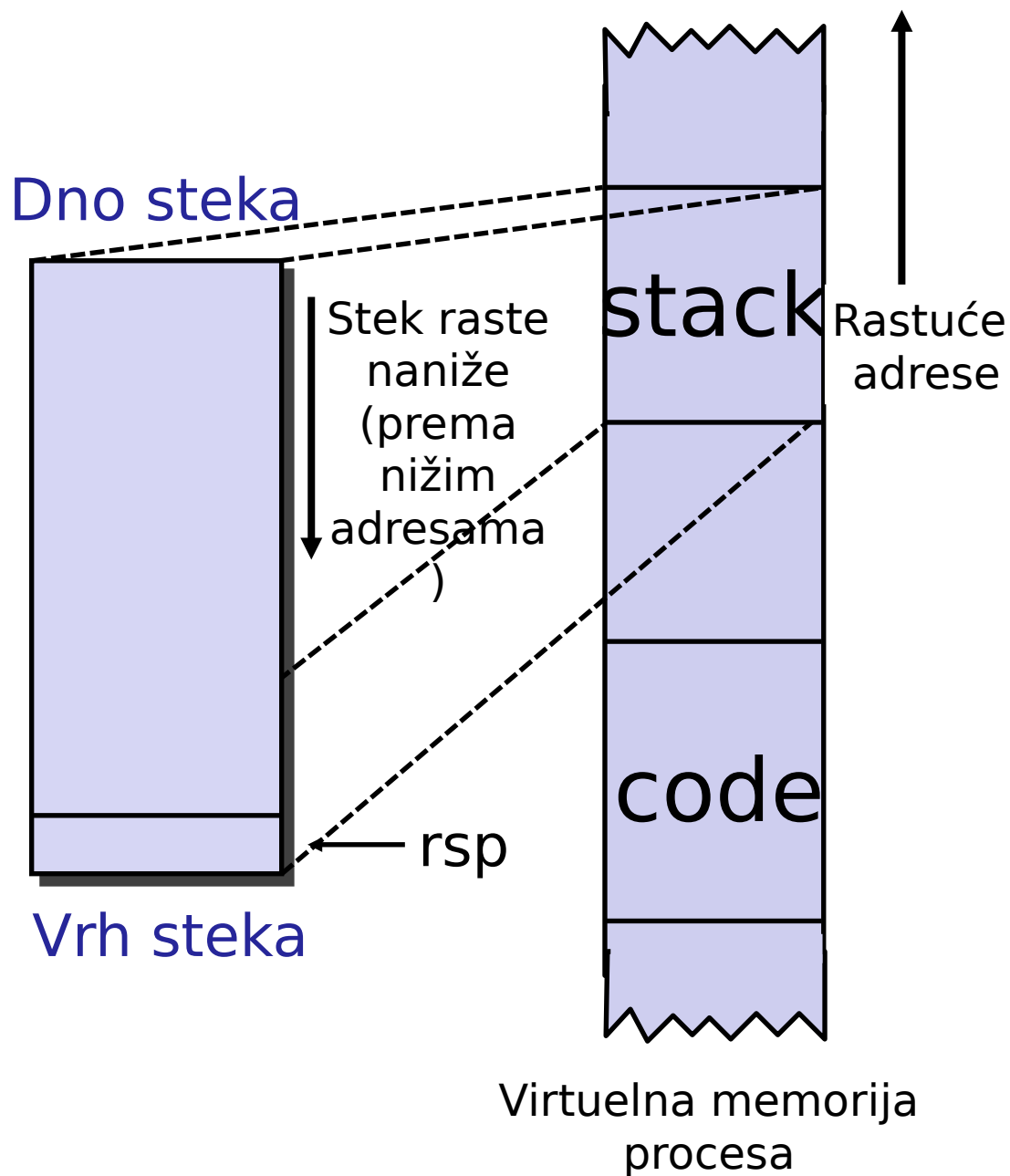
- **Region memorije kontrolisan po Last In First Out principu**

- **push Src**

- Dohvata izvorišni (*Src*) operand
- Umanjuje *rsp* za 8
- Upisuje operand na mem. adresu datu u *rsp*

- **pop Dest**

- Čita vrednost sa adrese koju daje *rsp*
- Uvećava *rsp* za 8
- Upisuje vrednost na



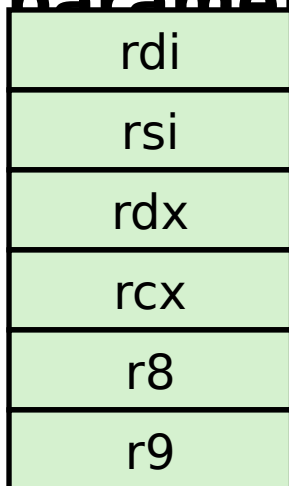
Tok kontrole kod potprograma

- Za podršku pozivanju i povratku iz potprograma koristi se stek
- **Pozivanje potprograma:** `call label`
 - Stavljanje (Push) povratne adrese na stek
 - Skok na labelu
- **Povratna adresa je:**
 - Adresa sledeće instrukcije neposredno posle `call` instrukcije
- **Povratak iz potprograma:** `ret`
 - Skida (Pop) adresu sa steka
 - Skok na adresu

Tok podataka kod potprograma

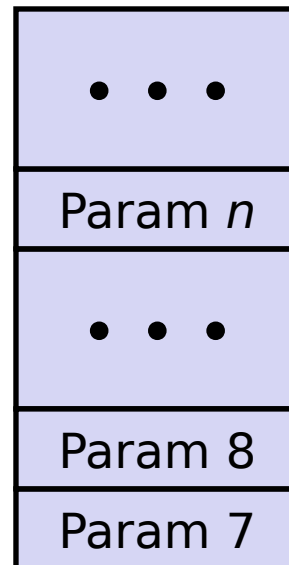
Registri

- Prenos prvih 6 parametara



-  a vrednost

Stek



- Stek se koristi samo po potrebi (npr. Više od 6 parametara)

Primer pren parametara

```
void multstore
(long x, long y, long *dest)
{
    long t = mult2(x, y);
    *dest = t;
}
```

```
00000000000400540 <multstore>:
    # x in rdi, y in rsi, dest in rdx
    . . .
400541: mov    rbx,rdx    # Save dest
400544: call   400550 <mult2>    # mult2(x,y)
    # t in %rax
400549: mov    QWORD PTR [rbx],rax # Save at dest
    . . .
```

```
long mult2
(long a, long b)
{
    long s = a * b;
    return s;
}
```

```
00000000000400550 <mult2>:
    # a in rdi, b in rsi
400550: mov    rax,rdi    # a
400553: imul   rax,rsi    # a * b
    # s in rax
400557: ret             # Return
```

Okviri (aktivacioni zapisi) na steku

- Tokom pozivanja potprograma, na steku se formira **okvir (eng. frame)**

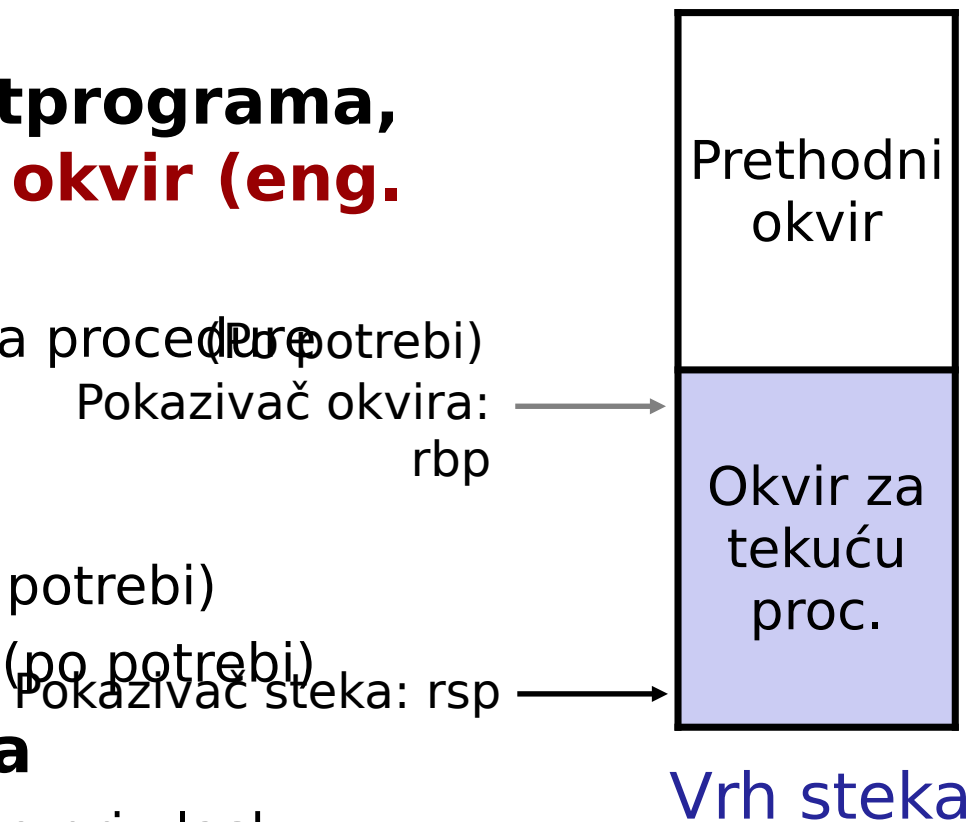
- Stanje za instancu poziva procedure (po potrebi)

- **Sadržaj okvira**


- Povratna adresa
- Lokalne promenljive (po potrebi)
- Privremene promenljive (po potrebi)

- **Upravljanje okvirima**

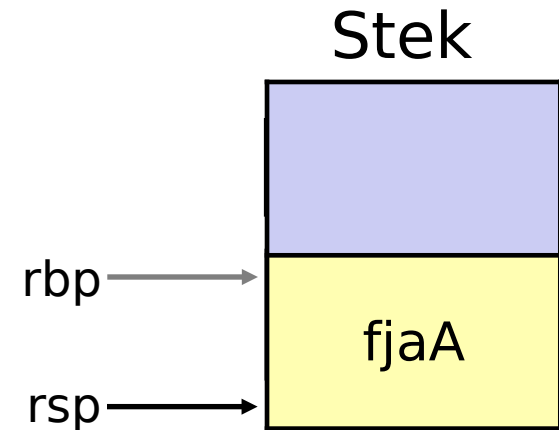
- Memorija mu se dodeljuje pri ulasku u proceduru
 - “Set-up” kod
 - Uključuje push povratne adrese instrukcijom **call**
- Memorija mu se dealocira pri povratku iz procedure



Primer

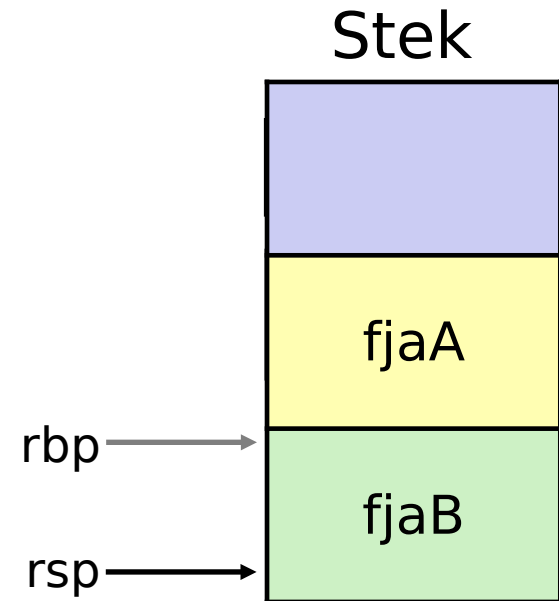


```
fjaA(...) {  
    .  
    .  
    fjaB();  
    .  
    .  
}  
  
fjaB() {  
    .  
    .  
    fjaC();  
    .  
    .  
}  
  
fjaC {  
    .  
    .  
}
```



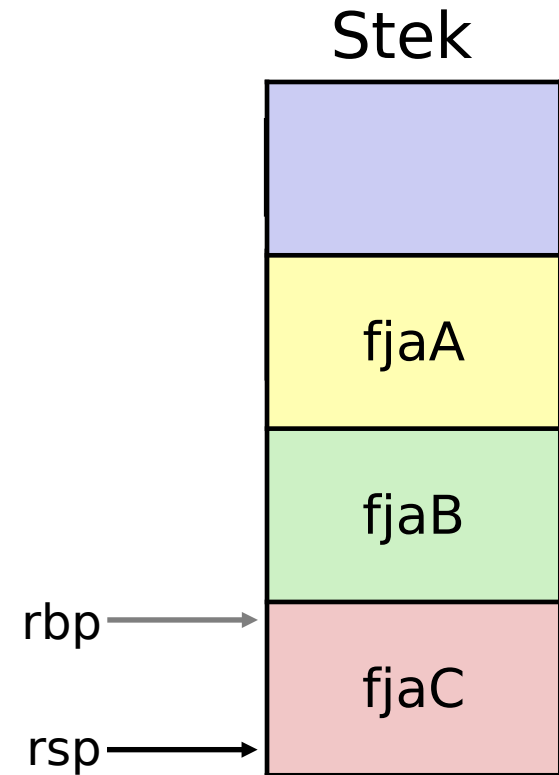
Primer

```
fjaA(...) {  
    .  
    .  
    fjaB();  
    .  
    .  
}  
  
fjaB() {  
    .  
    .  
    fjaC();  
    .  
    .  
}  
  
fjaC {  
    .  
    .  
}
```



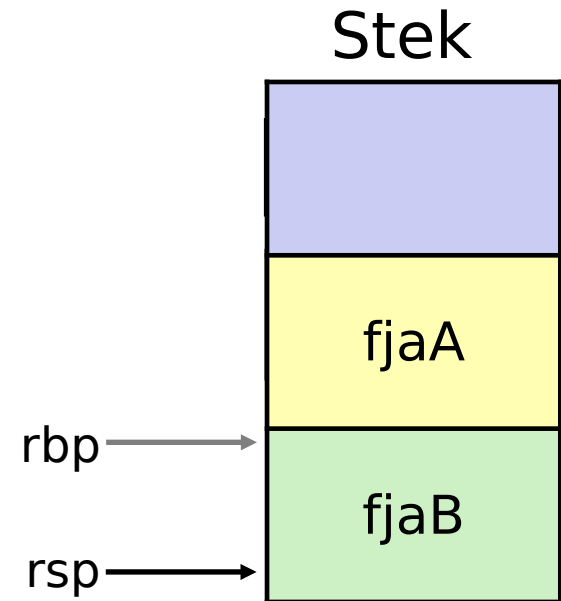
Primer

```
fjaA(...) {  
    .  
    .  
    fjaB();  
    .  
    .  
}  
  
fjaB() {  
    .  
    .  
    fjaC();  
    .  
    .  
}  
  
fjaC {  
    .  
    .  
}
```



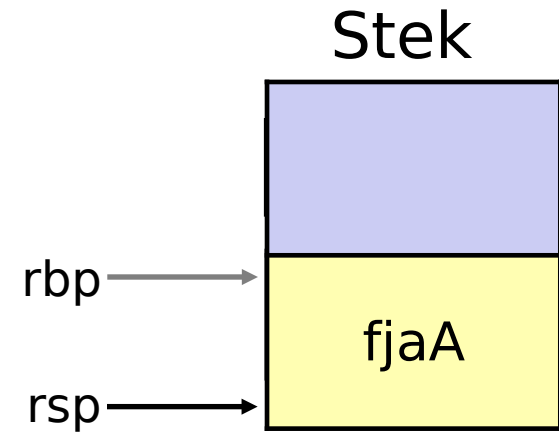
Primer

```
fjaA(...) {  
    .  
    .  
    fjaB();  
    .  
    .  
}  
  
fjaB() {  
    .  
    .  
    fjaC();  
    .  
    .  
}  
  
fjaC {  
    .  
    .  
}
```



Primer

```
fjaA(...) {  
    .  
    .  
    fjaB();  
    .  
    .  
}  
  
fjaB() {  
    .  
    .  
    fjaC();  
    .  
    .  
}  
  
fjaC {  
    .  
    .  
}
```



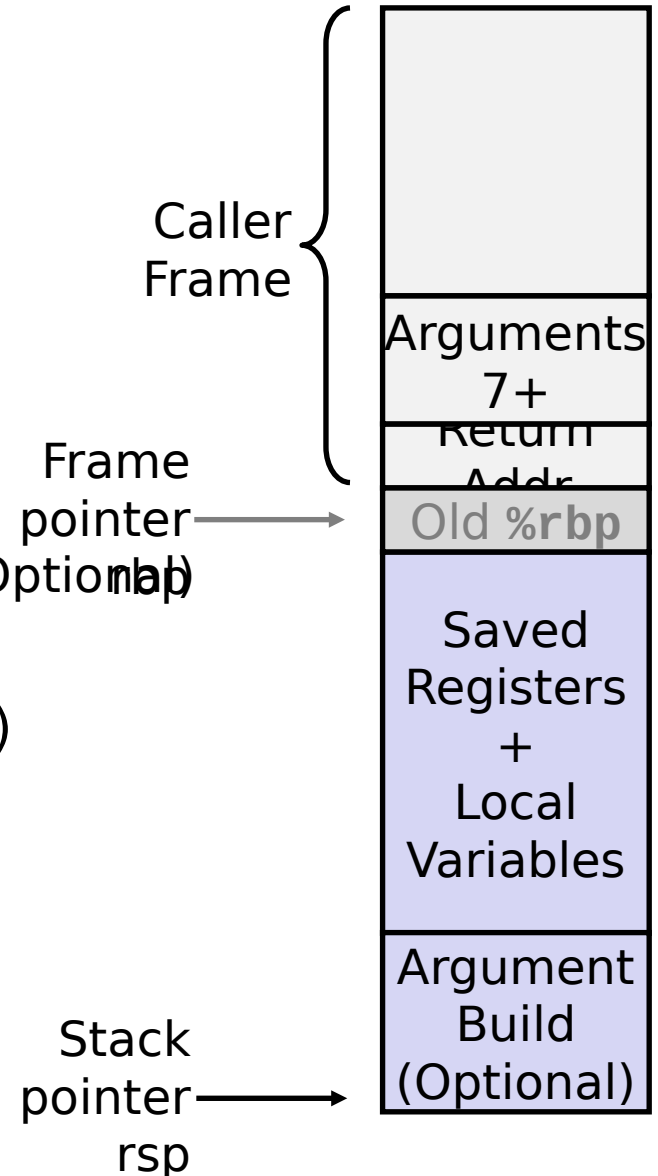
Okvir steka za x86-64/Linux

■ Tekući okvir (od “vrha” ka dnu)

- “Argument build:”
Stvari paramtri za funkciju koja će biti sledeća pozvana
- Lokalne promenljive
Ako ne mogu da stanu u registre
- Sačuvane vrednosti nekih registara (Optional)
- Stari pokazivač okvira (neobavezno)

■ Okvir pozivaoca

- Povratna adresa
 - Sačuvana instrukcijom call
- Stvarni parametri za tekući poziv



Primer: incr

```
long incr(long *p, long val) {  
    long x = *p;  
    long y = x + val;  
    *p = y;  
    return x;  
}
```

```
incr:  
    mov     rax, QWORD PTR [rdi]  
    add     rsi, rax  
    mov     QWORD PTR [rdi], rsi  
    ret
```

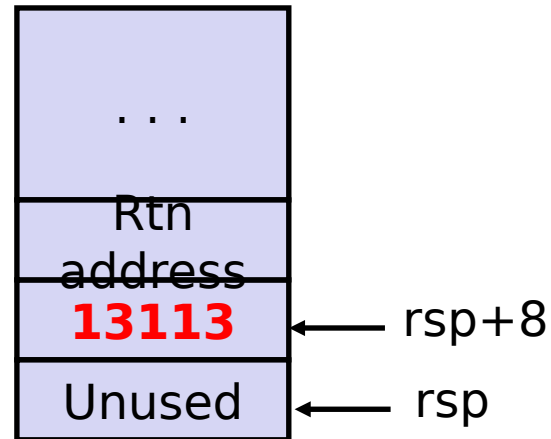
| Registar | Upotreba |
|----------|------------------------------------|
| rdi | Parametar p |
| rsi | Parametar val , y |
| rax | x , Povratna vrednost |

Primer: Pozivanje incr

```
long call_incr() {  
    long v1 = 13113;  
    long v2 = incr(&v1, 3000);  
    return v1+v2;  
}
```

```
call_incr:  
    sub    rsp,16  
    mov    QWORD PTR [rsp+8],13113  
    mov    esi,3000  
    lea    rdi,[rsp+8]  
    call   incr  
    add    rax,QWORD PTR [rsp+8]  
    add    rsp,16  
    ret
```

Stanje steka pre call incr



| Regista r | Upotreba |
|--------------|--------------|
| rax | v2, Rezultat |
| rdi | &v1 |
| rsi | 3000 |

Upotreba registara u x86-64 Linuxu

■ **rax**

- Povratna vrednost
- Čuva ga pozivalac
- Potprogram može da ga promeni

■ **rdi, ..., r9**

- Stvarni parametri
- Čuva pozivalac
- Potprogram može da ih promeni

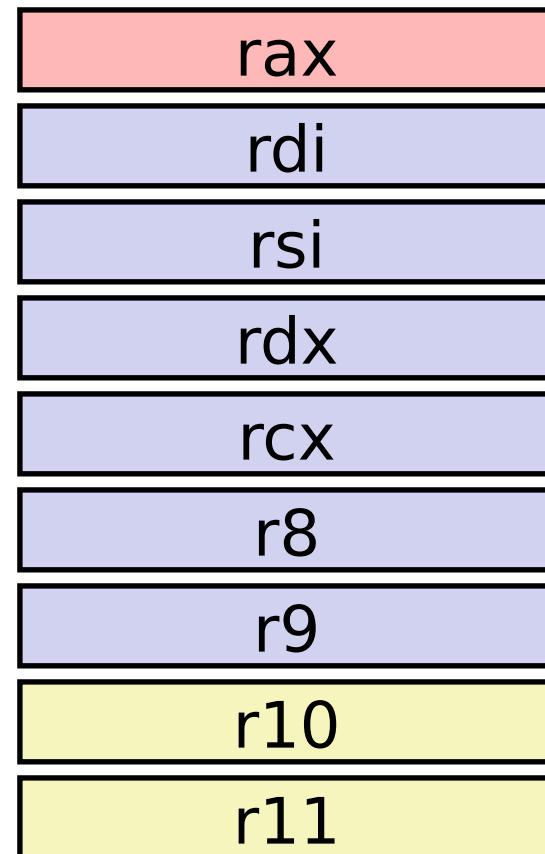
■ **r10, r11**

- Čuva pozivalac
- Potprogram može da ih promeni

Caller-saved
Return value

Caller-saved
Arguments

Caller-saved
Temporaries



Upotreba registara u x86-64 Linuxu (nastavak)

■ **rbx, r12, r13, r14**

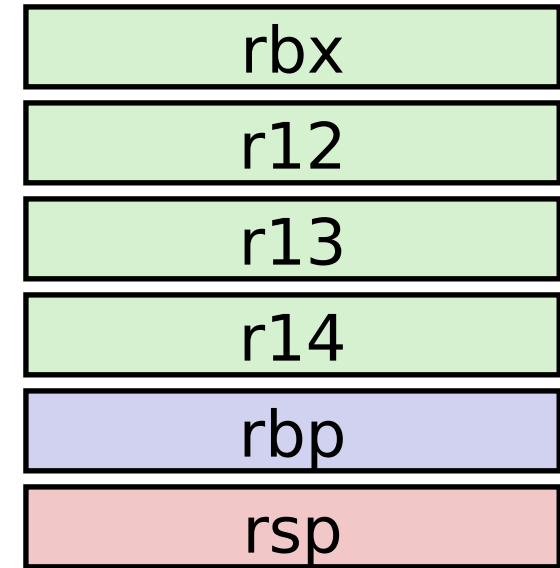
- Čuva ih pozvani potprogram
- Ako ih koristi, mora prethodno sačuvati sadržaj i na kraju korišćenja vratiti stari sadržaj

Callee-saved
Temporaries

■ **rbp**

- Čuva pozvani
- Pozvani mora snimiti & restaurirati
- Može biti korišćen kao pokazivač okvira
- O upotrebi rbp odlučuje kompajler

Special



■ **rsp**

- Specijalni slučaj registra koji čuva

Rekurzivne funkcije su automatski podržane

Okviri na steku omogućavaju svakom **pozivu** funkcije da ima privatne podatke:

- Sačuvane registre & lokalne promenljive
- Sačuvanu povratnu adresu

```
/* Recursive popcount */
long pcount_r(unsigned long x) {
    if (x == 0)
        return 0;
    else
        return (x & 1)
            + pcount_r(x >> 1);
}
```

```
pcount_r:
    test    rdi, rdi
    jne     .L1
    mov     eax, 0x0
    ret
.L1: push   rbx
    mov     rbx, rdi
    and     ebx, 0x1
    shr     rdi, 1
    call    pcount_r
    add     rax, rbx
    pop     rbx
    ret
```

| Regist ar | Upotreba | Vrsta |
|-----------|----------|----------|
| rdi | x | Argument |
| rax | Povratna | Povratna |

