Operativni sistemi: interna struktura i principi dizajna

Poglavlje 7 Upravljanje memorijom

Sedmo izdanje

William Stallings

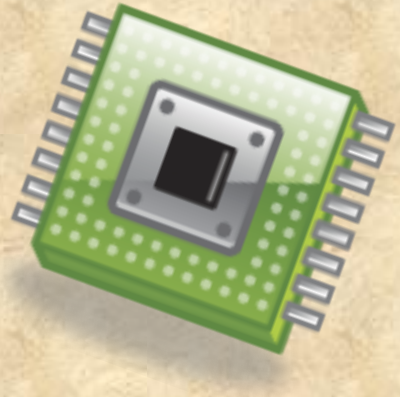
Definicija

- Upravljanje memorijom je proces
  - dodeljivanja primarne memorije korisničkim programima
  - oslobađanja te memorije kada više nije potrebna
  - zaštite memorijskog prostora svakog korisnika od drugih korisničkih programa; tj. obezbeđivanja da svaki program pristupa samo memorijskim lokacijama koje su mu dodeljene.

Zahtevi

- Da bi efikasno upravljao memorijom, OS mora imati
  - Politike dodele memorije
  - Metode praćenja stanja memorijskih lokacija (slobodne ili dodeljene)
  - Politike oduzimanja memorije jednom procesu radi dodele drugom

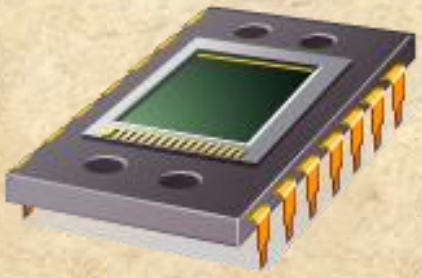
Pojmovi upravljanja memorijom



Okvir	Blok glavne memorije fiksne dužine.
Stranica	Blok podataka fiksne dužine koji se nalazi u sekundarnoj memoriji (npr. na disku). Stranica podataka može privremeno biti kopirana u okvir glavne memorije.
Segment	Blok podataka promenljive dužine koji se nalazi u sekundarnoj memoriji. Ceo segment može privremeno biti kopiran u slobodnu oblast glavne memorije (segmentacija), ili segment može biti podeljen na stranice koje se pojedinačno kopiraju u glavnu memoriju (kombinacija segmentacije i straničenja).

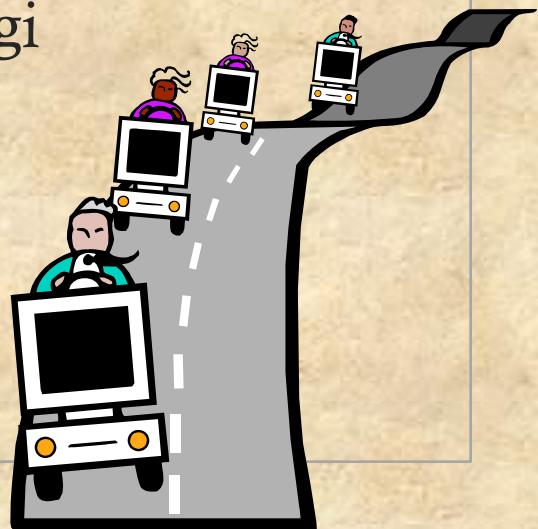
Zahtevi upravljanja memorijom

- Upravljanje memorijom treba da zadovolji sledeće zahteve:
  - Relokacija
  - Zaštita
  - Deljenje
  - Logička organizacija
  - Fizička organizacija



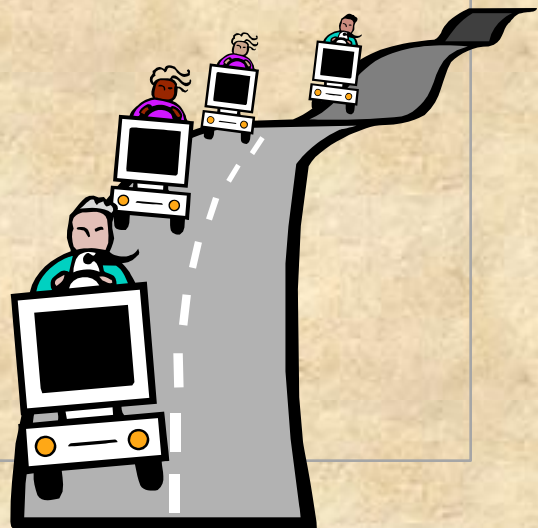
Relokacija

- Relokacija je proces prilagođavanja adresa programa stvarnim fizičkim adresama na kojima se program nalazi dok se izvršava
- Zašto je potrebna relokacija?
  - Programer/prevodilac ne zna koji će drugi programi biti u memoriji kada se program bude izvršavao



Relokacija

- Zašto je potrebna relokacija? (nastavak)
  - Aktivni procesi moraju moći da se izbacuju iz glavne memorije i vraćaju u nju, kako bi iskorišćenost procesora bila maksimalna
  - Zahtev da proces, kada se ponovo vrati u memoriju, mora biti smešten u istu oblast memorije bio bi ograničavajući
- Zato mora biti moguće prilagoditi adrese svaki put kada se program učita.



Zahtevi adresiranja

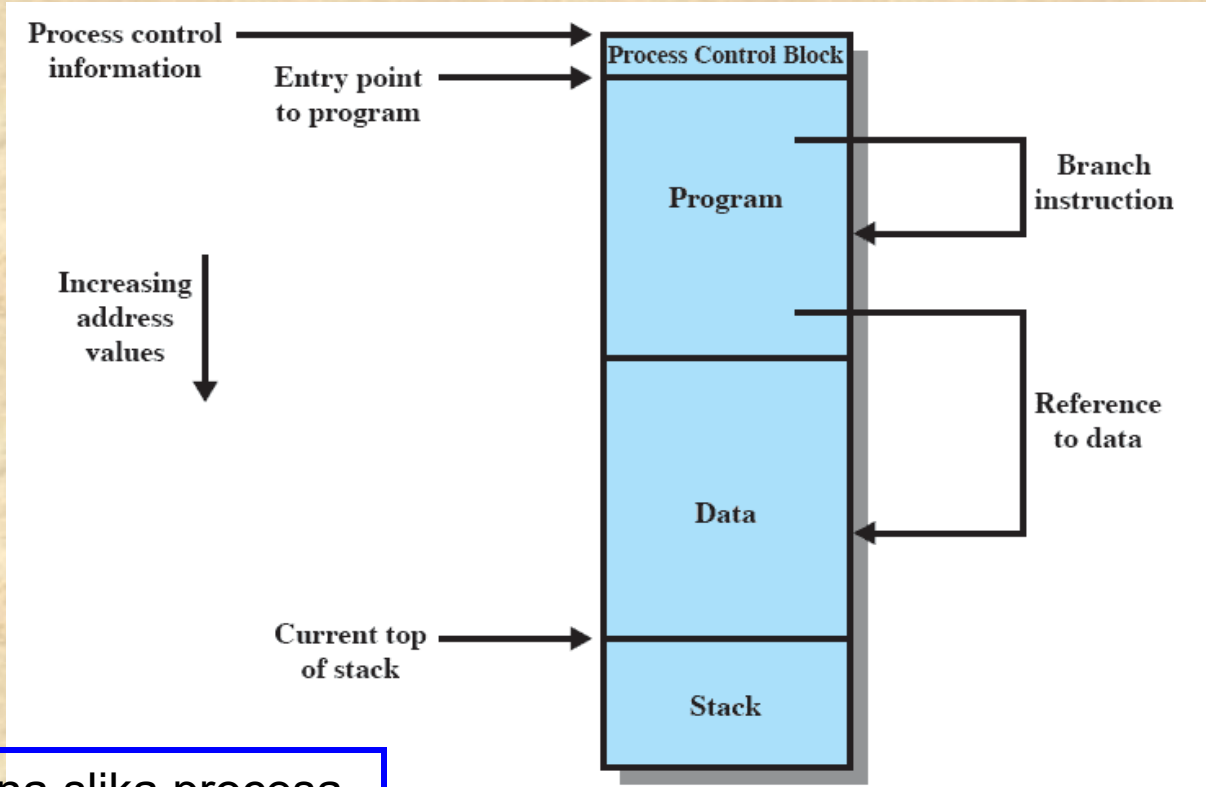


Figure 7.1 Addressing Requirements for a Process

Uprošćena slika procesa

Zaštita

- Procesi moraju dobiti dozvolu da pristupaju memorijskim lokacijama radi čitanja ili upisa
- Lokacija programa u glavnoj memoriji je nepredvidiva
- Memorijske reference procesa moraju se proveravati pri izvršavanju
- Mehanizmi koji podržavaju relokaciju podržavaju i zaštitu



Deljenje

- Korisno je omogućiti svim procesima pristup istoj kopiji programa, umesto da svaki ima svoju zasebnu kopiju
- Upravljanje memorijom mora omogućiti kontrolisan pristup deljenim oblastima memorije bez narušavanja zaštite
- Mehanizmi koji podržavaju relokaciju podržavaju i mogućnost deljenja



Logička organizacija

- Glavna memorija je organizovana kao linearni (1-D) adresni prostor koji se sastoji od niza bajtova ili reči.
- Programi nisu nužno organizovani na ovaj način
- Straničenje na bazi segmentacije

Programi se pišu u obliku modula

- moduli se mogu pisati i prevoditi nezavisno
- modulima se daju različiti stepeni zaštite (samo čitanje, samo izvršavanje)
- deljenje na nivou modula odgovara načinu na koji korisnik vidi problem

Fizička organizacija

- Memorija na dva nivoa za čuvanje programa:
  - Disk (spor i jeftin) i RAM (brz i skuplji)
  - Glavna memorija je nepostojana, disk nije
- Korisnik ne bi trebalo da bude odgovoran za organizovanje prenosa koda/podataka između ta dva nivoa.

Fizička organizacija

Programeru se ne može prepustiti odgovornost za upravljanje memorijom

Memorija raspoloživa za program i njegove podatke može biti nedovoljna

preklapanje (overlay) omogućava da se različitim modulima dodeli ista oblast memorije, ali je programiranje dugotrajno

Programer ne zna koliko će prostora biti na raspolaganju

Particionisanje memorije

- Upravljanje virtuelnom memorijom dovodi procese u glavnu memoriju radi izvršavanja na procesoru
  - uključuje virtuelnu memoriju
  - zasnovano na segmentaciji i straničenju
- Upravljanje memorijom particionisanjem
  - korišćeno u više varijanti u nekim danas zastarelim operativnim sistemima
  - ne uključuje virtuelnu memoriju

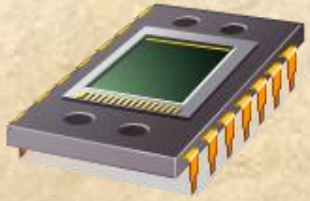
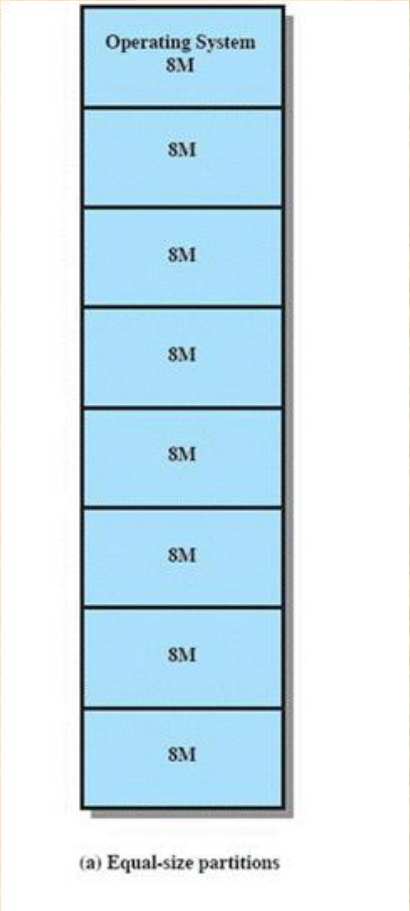


Tabela 7.2 Tehnike upravljanja memorijom

Tehnika	Opis	Prednosti	Nedostaci
Fiksno particionisanje	Glavna memorija se pri generisanju sistema deli na određeni broj statičkih particija. Proces se može učitati u particiju iste ili veće veličine.	Jednostavna implementacija; mali dodatni troškovi za operativni sistem.	Neefikasno korišćenje memorije zbog unutrašnje fragmentacije; najveći broj aktivnih procesa je fiksan.
Dinamičko particionisanje	Particije se kreiraju dinamički, tako da se svaki proces učitava u particiju tačno svoje veličine.	Nema unutrašnje fragmentacije; efikasnije korišćenje glavne memorije.	Neefikasno korišćenje procesora zbog potrebe za sažimanjem radi suzbijanja spoljašnje fragmentacije.
Jednostavno straničenje	Glavna memorija je podeljena na okvire jednake veličine. Svaki proces je podeljen na stranice jednake veličine, iste dužine kao okviri. Proces se učitava tako što se sve njegove stranice učitaju u slobodne okvire, koji ne moraju biti susedni.	Nema spoljašnje fragmentacije.	Mala unutrašnja fragmentacija.
Jednostavna segmentacija	Svaki proces je podeljen na određeni broj segmenata. Proces se učitava tako što se svi njegovi segmenti učitaju u dinamičke particije, koje ne moraju biti susedne.	Nema unutrašnje fragmentacije; bolje iskorišćenje memorije i manji dodatni troškovi nego kod dinamičkog particionisanja.	Spoljašnja fragmentacija.
Straničenje virtuelne memorije	Kao kod jednostavnog straničenja, osim što nije potrebno učitati sve stranice procesa. Potrebne stranice koje nisu u memoriji kasnije se automatski učitavaju.	Nema spoljašnje fragmentacije; viši stepen multiprogramiranja; veliki virtuelni adresni prostor.	Dodatni troškovi složenog upravljanja memorijom.
Segmentacija virtuelne memorije	Kao kod jednostavne segmentacije, osim što nije potrebno učitati sve segmente procesa. Potrebni segmenti koji nisu u memoriji kasnije se automatski učitavaju.	Nema unutrašnje fragmentacije; viši stepen multiprogramiranja; veliki virtuelni adresni prostor; podrška za zaštitu i deljenje.	Dodatni troškovi složenog upravljanja memorijom.

Fiksno particionisanje

- Particije jednake veličine
  - svaki proces čija veličina nije veća od veličine particije može se učitati u slobodnu particiju
- OS može izbaciti proces iz memorije ako su sve particije zauzete i nijedan proces nije u stanju Spreman ili Izvršava se



(a) Equal-size partitions

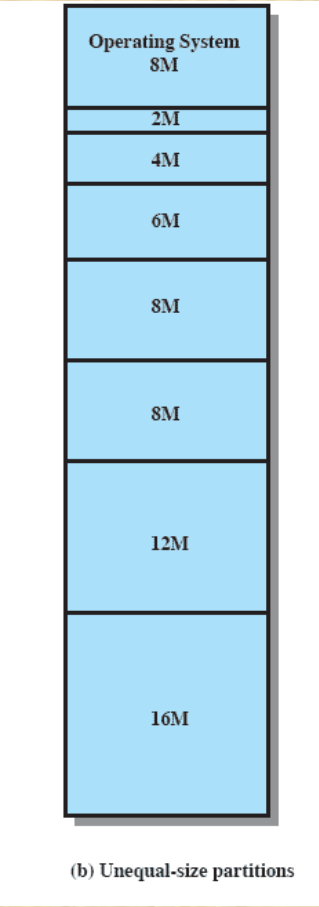
Nedostaci

- Program može biti prevelik da stane u particiju
  - program se mora projektovati uz korišćenje preklapanja
- Glavna memorija se koristi neefikasno
  - svaki program, bez obzira na veličinu, zauzima celu particiju
  - ***unutrašnja fragmentacija***
    - neiskorišćen prostor jer je učitani blok podataka manji od particije



Particije nejednake veličine

- Particije nejednake veličine ublažavaju ove probleme
  - programi do 16M mogu se smestiti bez preklapanja
  - particije manje od 8M omogućavaju smeštanje manjih programa uz manju unutrašnju fragmentaciju



(b) Unequal-size partitions

Dodela memorije

Fiksne particije

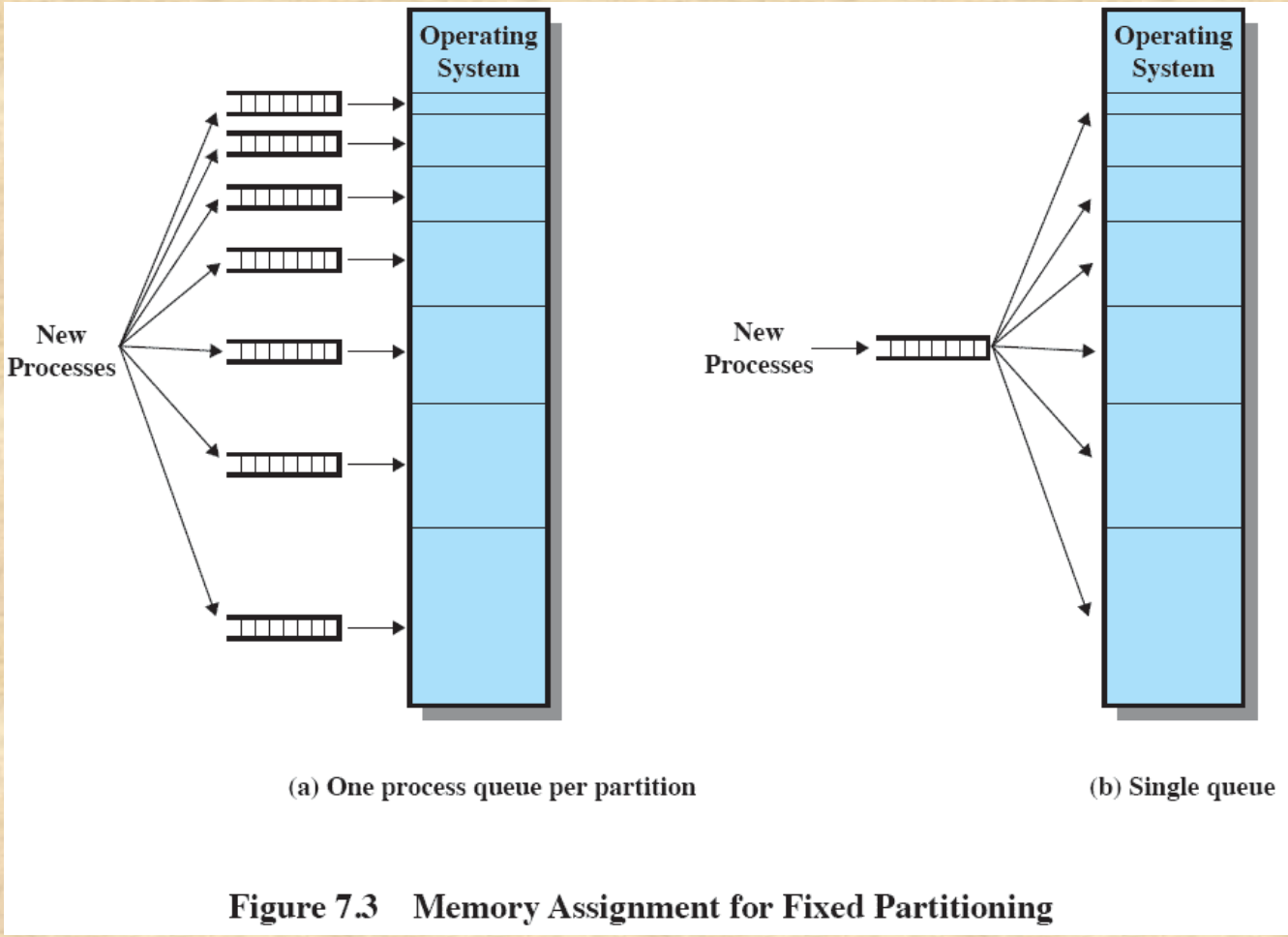


Figure 7.3 Memory Assignment for Fixed Partitioning

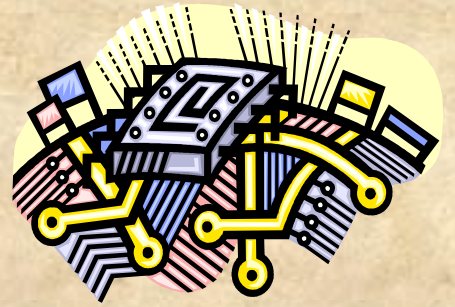
Nedostaci

- Broj particija zadat pri generisanju sistema ograničava broj aktivnih procesa u sistemu
- Mali poslovi neće efikasno koristiti prostor particije



Dinamičko particionisanje

- Particije su promenljive dužine i broja
- Procesu se dodeljuje tačno onoliko memorije koliko mu je potrebno
- Ovu tehniku je koristio IBM-ov operativni sistem za mejnfrejm računare, OS/MVT



Efekat dinamičkih particija

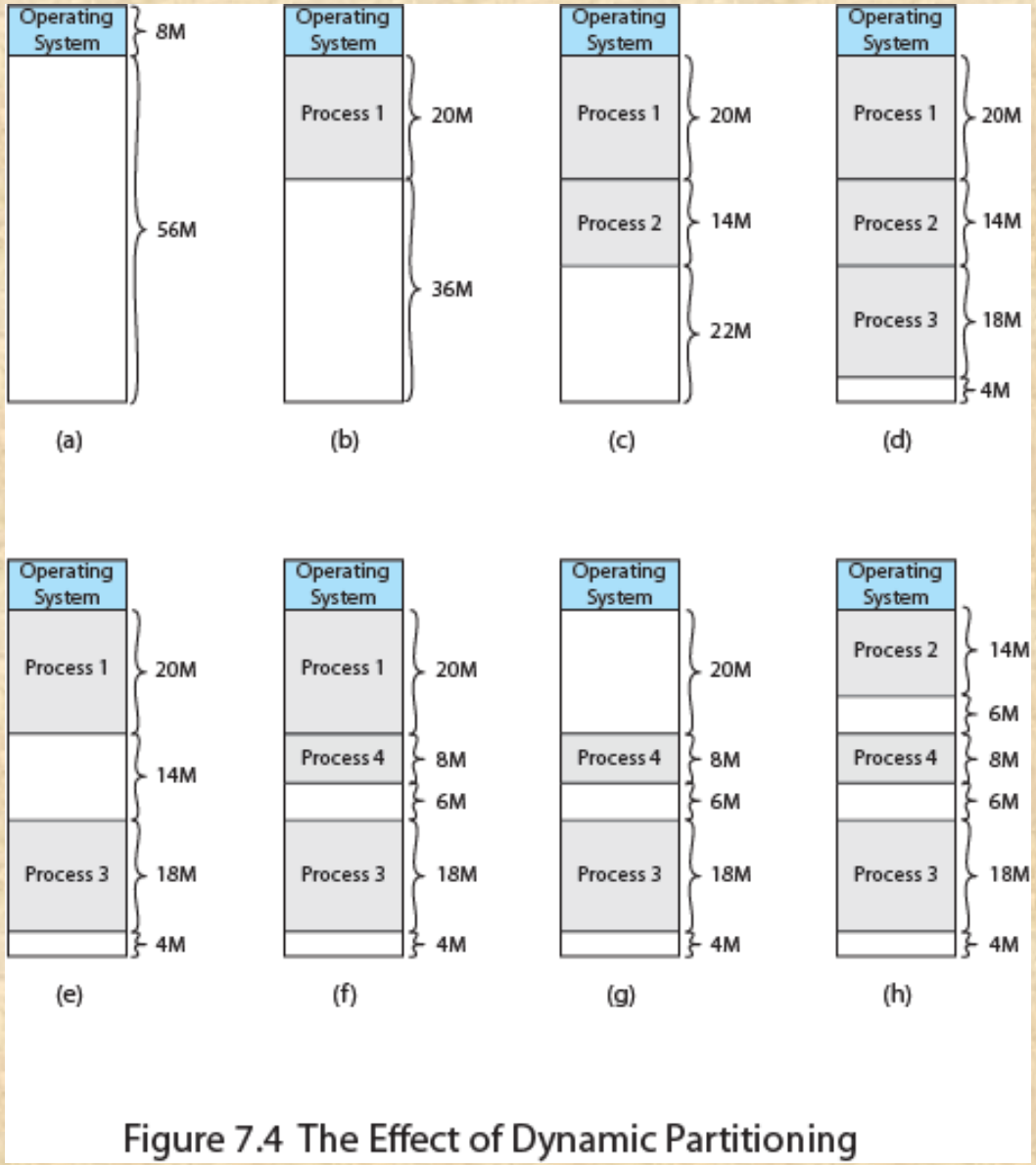


Figure 7.4 The Effect of Dynamic Partitioning

Dinamičko particionisanje

Spoljašnja fragmentacija

- memorija postaje sve više fragmentisana
- iskorišćenost memorije opada

Sažimanje

- tehnika za prevazilaženje spoljašnje fragmentacije
- OS pomera procese tako da budu jedan do drugog
- sva slobodna memorija je u jednom bloku
- dugotrajno je i troši vreme CPU-a

Algoritmi smeštanja

Najbolje uklapanje (best-fit)	Prvo uklapanje (first-fit)	Sledeće uklapanje (next-fit)
• bira blok čija je veličina najbliža traženoj	• pretražuje memoriju od početka i bira prvi slobodan blok koji je dovoljno velik	• pretražuje memoriju od mesta poslednjeg smeštanja i bira sledeći slobodan blok koji je dovoljno velik

Primer konfiguracije memorije

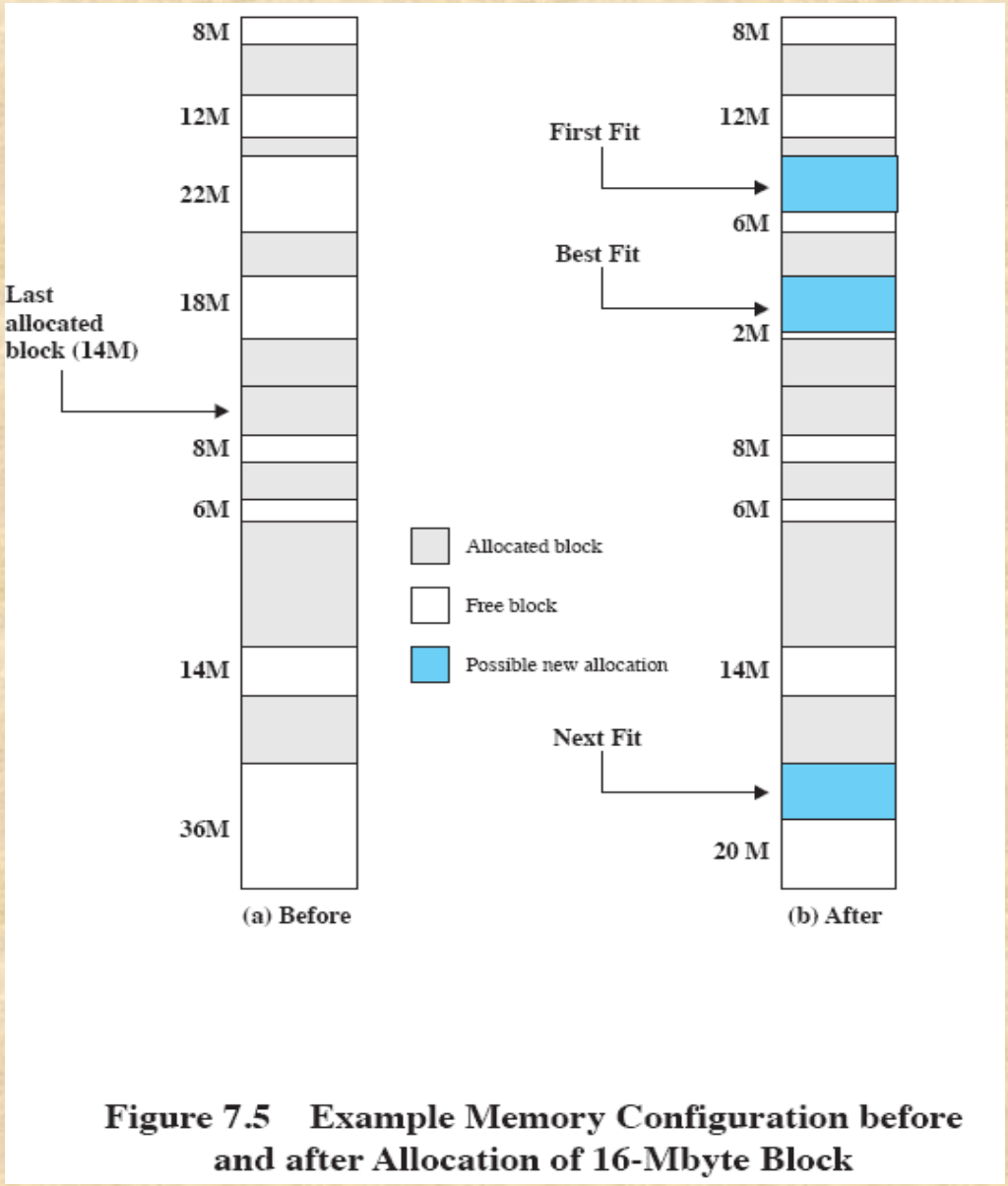
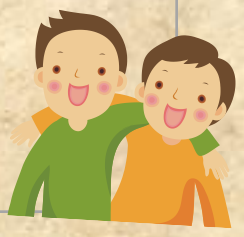


Figure 7.5 Example Memory Configuration before and after Allocation of 16-Mbyte Block

Drugarski sistem (buddy system)

- Kombinuje fiksno i dinamičko particionisanje
- Prostor raspoloživ za dodelu tretira se kao jedan blok
- Raspoloživi su memorijski blokovi veličine 2^K *reči,* $L \leq K \leq U$, gde je
  - 2^L = *veličina najmanjeg bloka koji se dodeljuje*
  - 2^U = veličina najvećeg bloka koji se dodeljuje; obično je 2^U veličina cele memorije raspoložive za dodelu



Primer drugarskog sistema

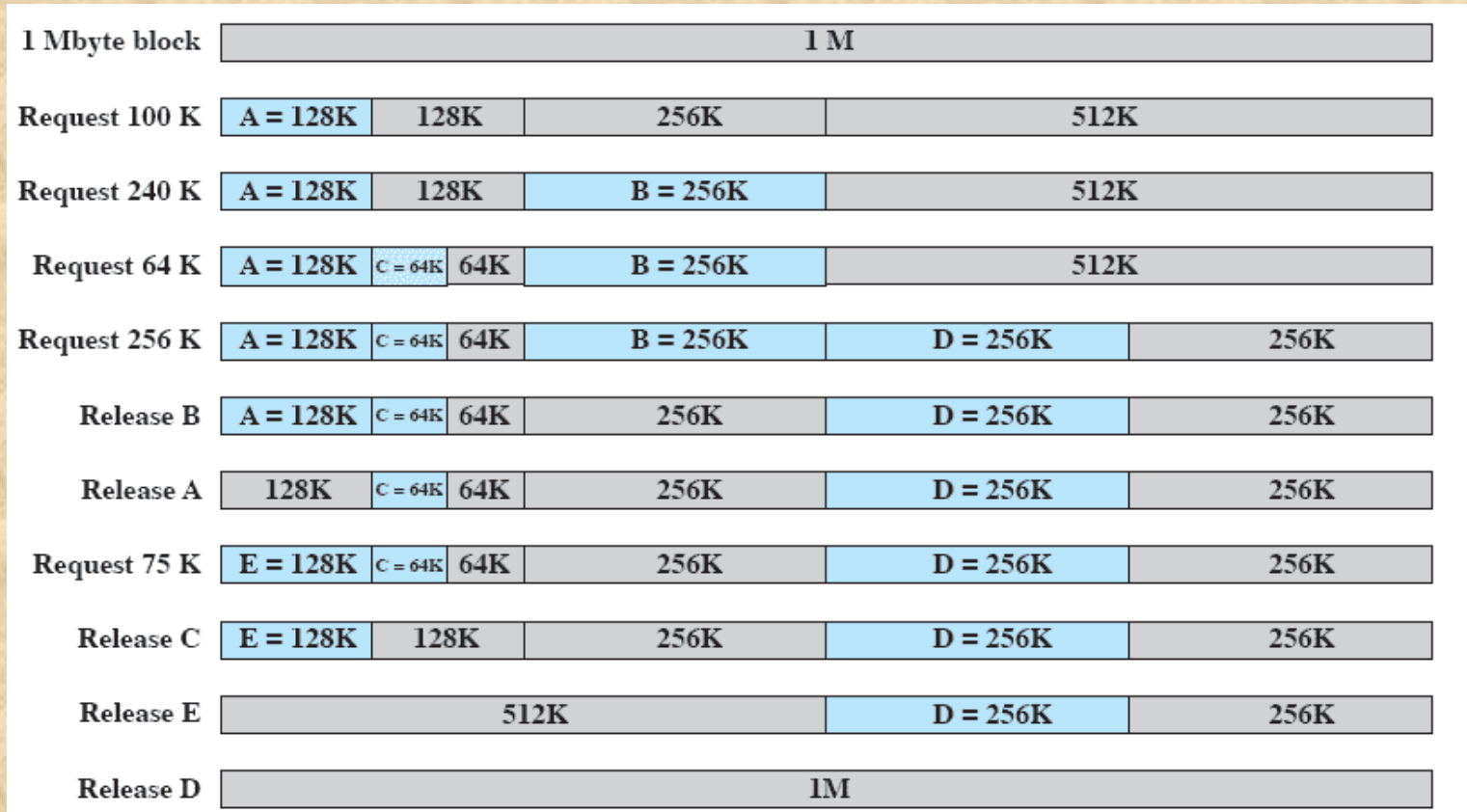


Figure 7.6 Example of Buddy System

Prikaz stablom

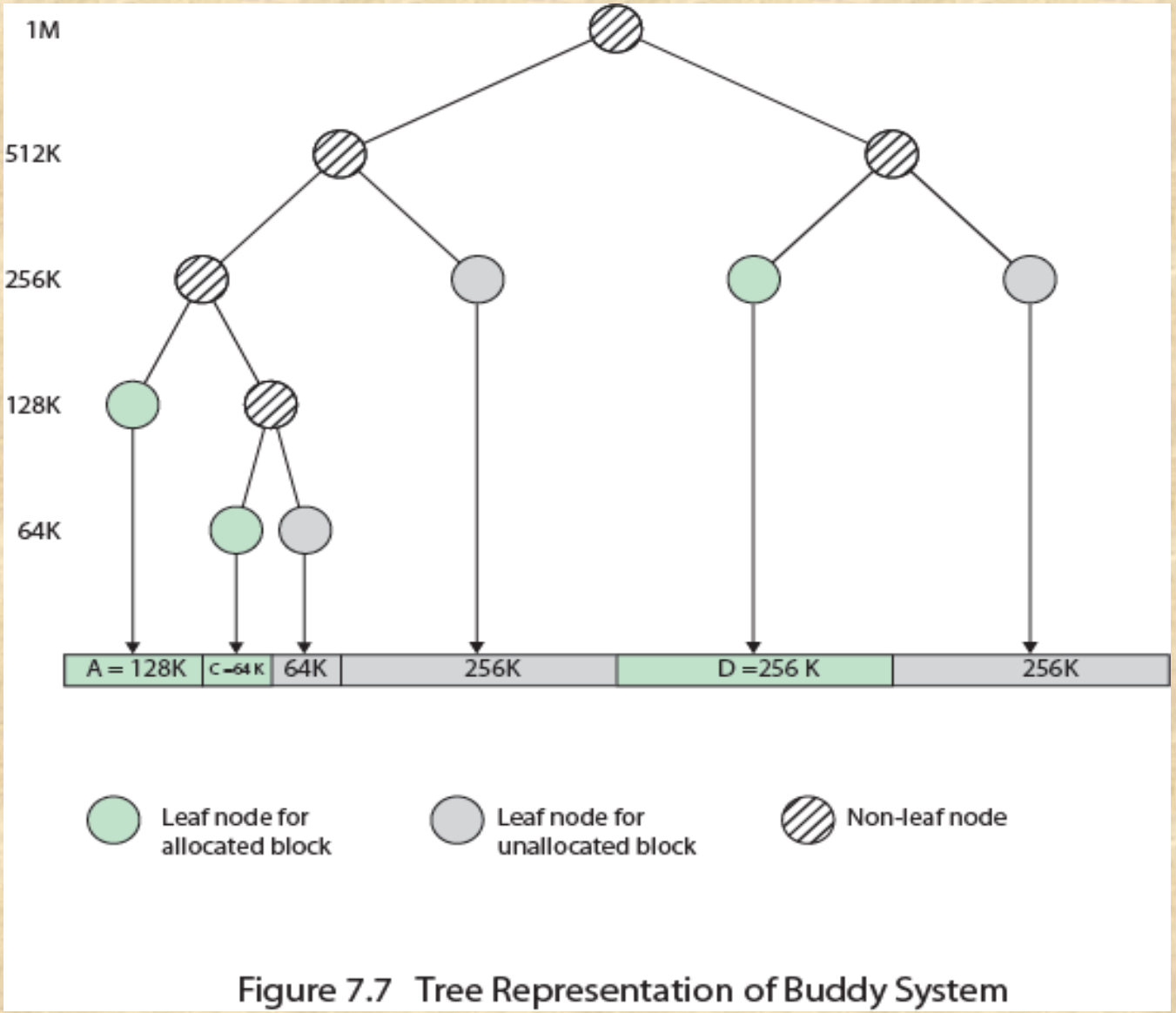


Figure 7.7 Tree Representation of Buddy System

Adrese

Logička

- referenca na memorijsku lokaciju, nezavisna od trenutnog rasporeda podataka u memoriji

Relativna

- adresa se izražava kao lokacija u odnosu na neku poznatu tačku

Fizička ili apsolutna

- stvarna lokacija u glavnoj memoriji

Relokacija

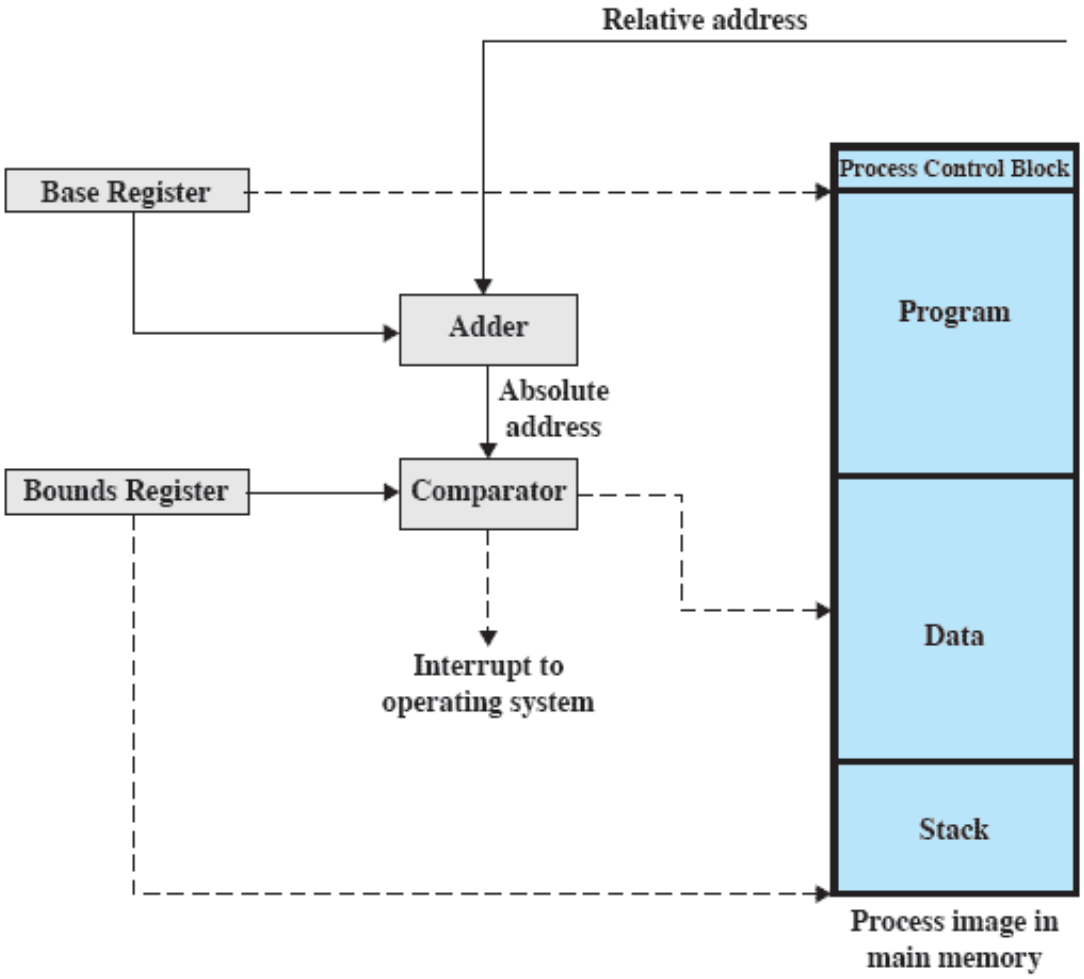


Figure 7.8 Hardware Support for Relocation

Straničenje

- Memorija se deli na relativno male delove jednake, fiksne veličine
- I proces se deli na male delove iste, fiksne veličine

Stranice	*Okviri*
• delovi procesa	• raspoloživi delovi memorije

Dodela slobodnih okvira procesu

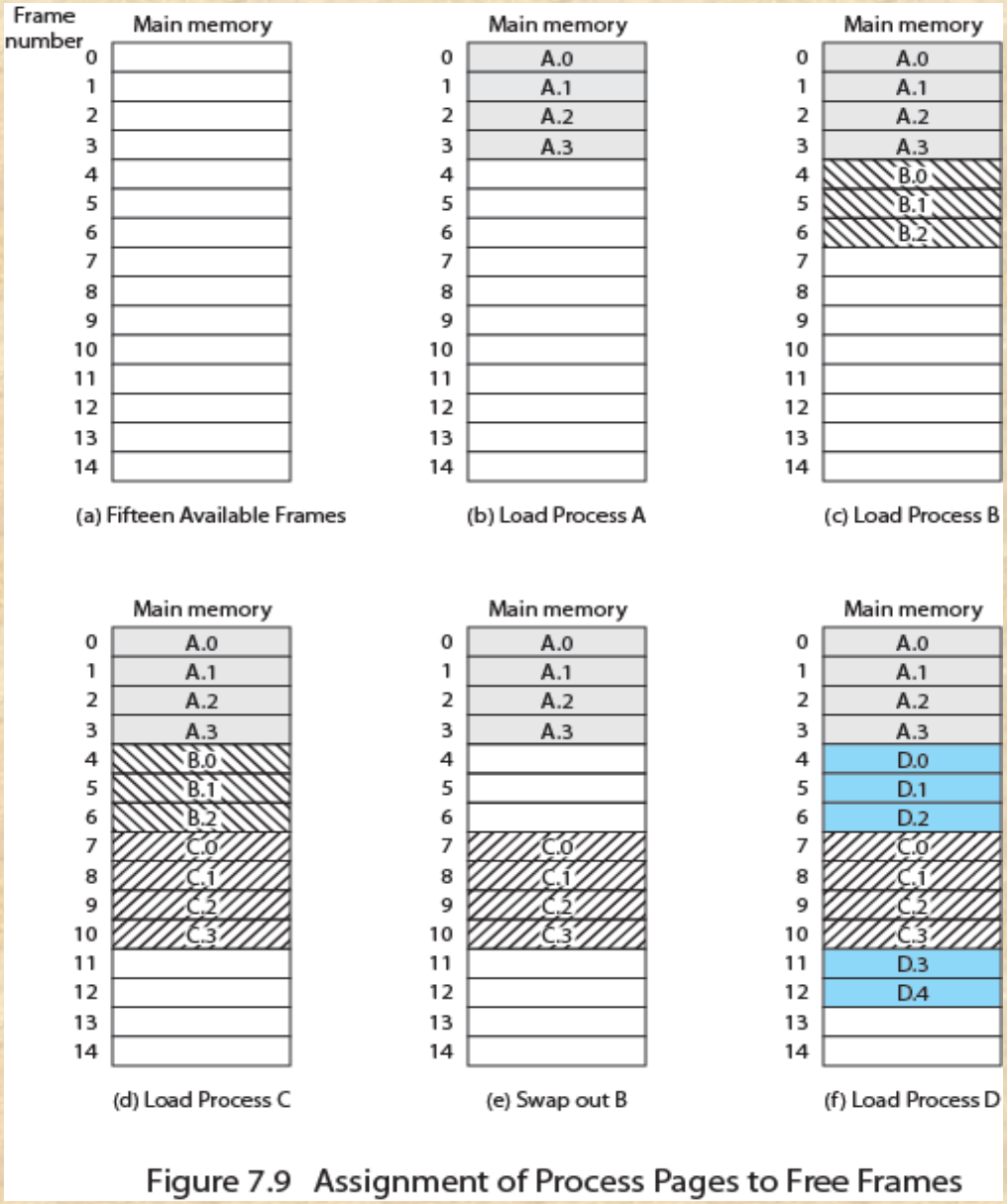


Figure 7.9 Assignment of Process Pages to Free Frames

Tabela stranica

- Operativni sistem je održava za svaki proces
- Sadrži lokaciju okvira za svaku stranicu procesa
- Procesor mora znati kako da pristupi tabeli stranica tekućeg procesa
- Procesor je koristi za formiranje fizičke adrese



Strukture podataka

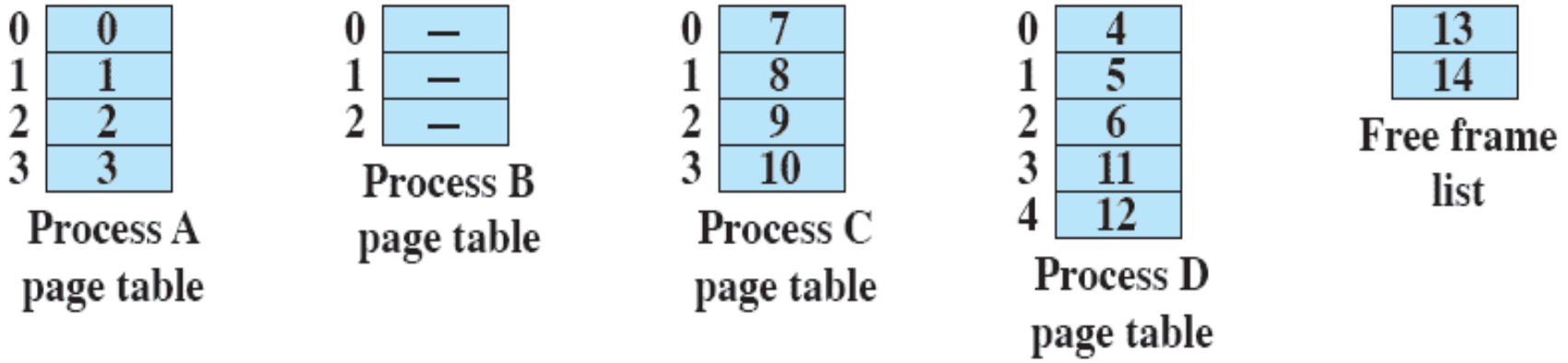


Figure 7.10 Data Structures for the Example of Figure 7.9 at Time Epoch (f)

Logičke adrese

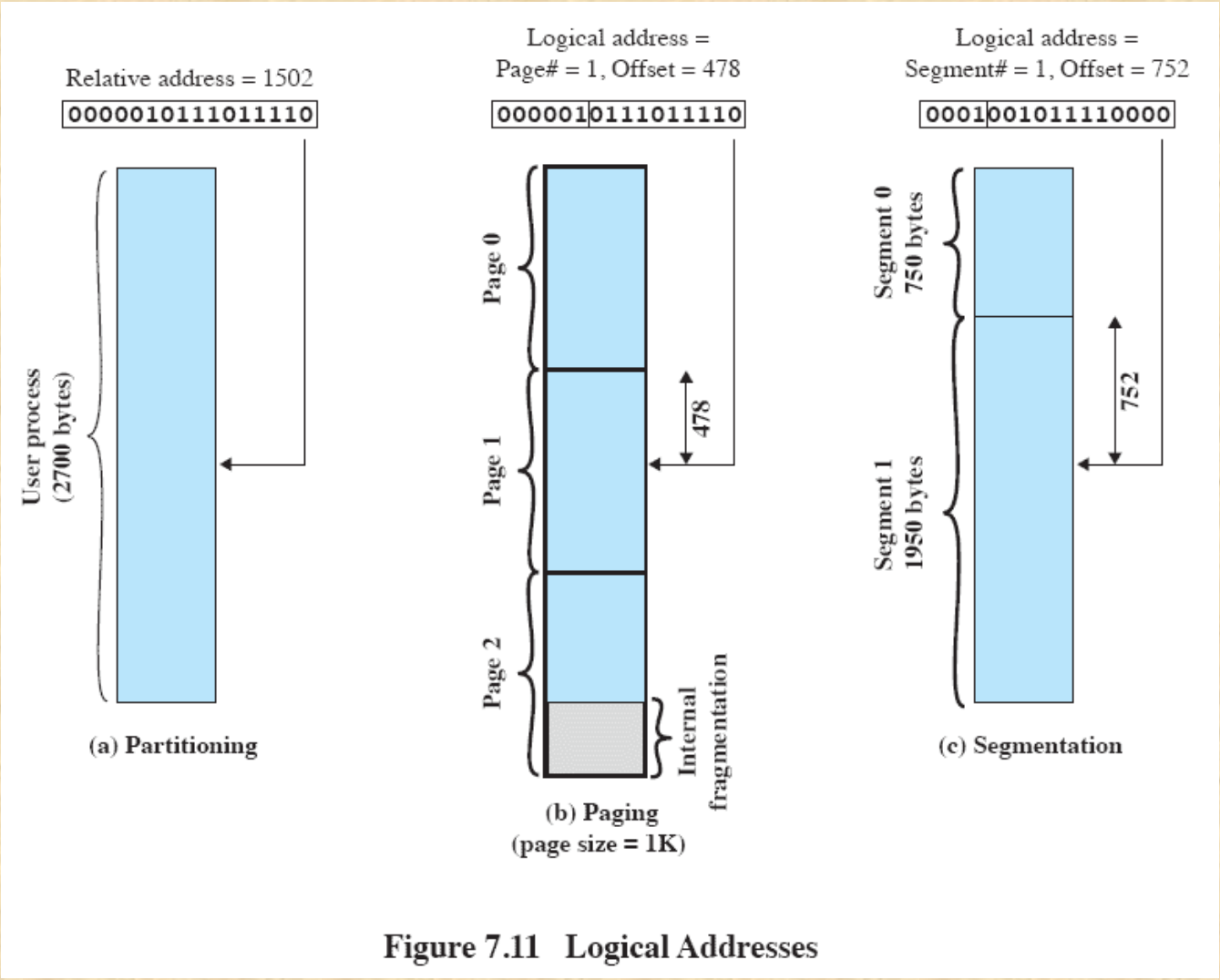
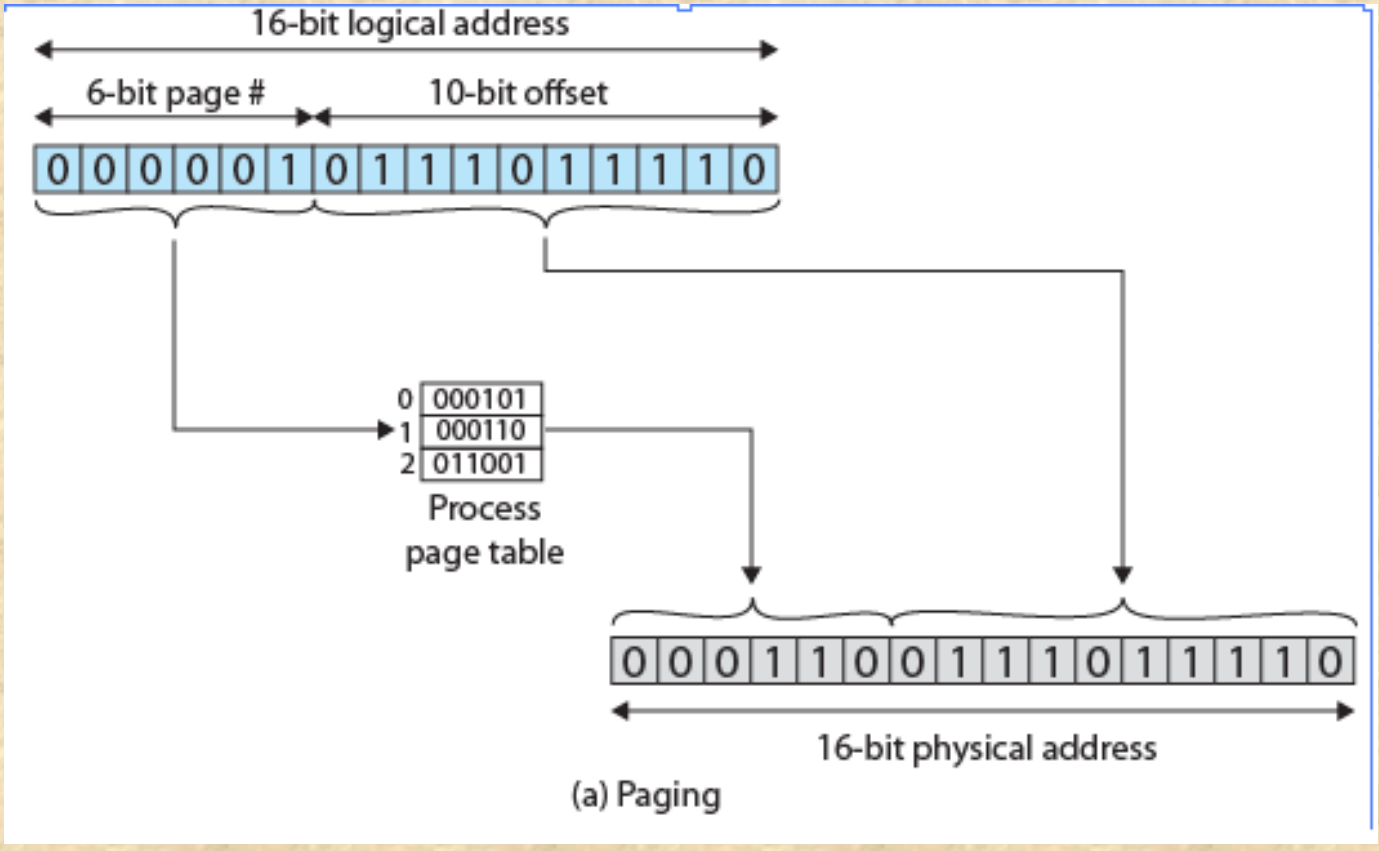


Figure 7.11 Logical Addresses

Prevođenje logičke adrese u fizičku - straničenje

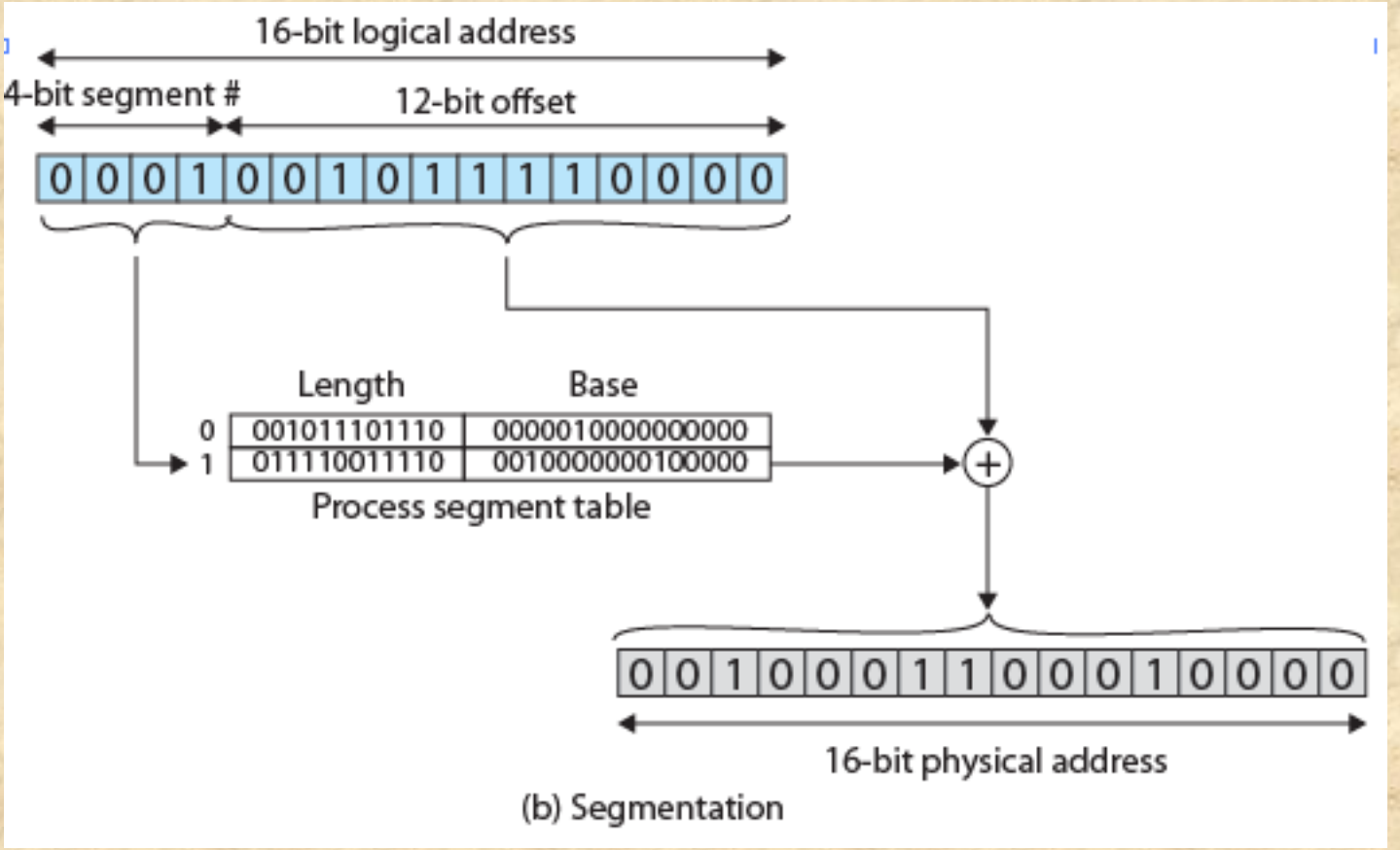


Segmentacija

- Program se može podeliti na segmente
  - mogu biti različite dužine
  - postoji maksimalna dužina
- Adresa se sastoji od dva dela:
  - broj segmenta
  - pomeraj
- Slično dinamičkom particionisanju
- Eliminiše unutrašnju fragmentaciju



Prevođenje logičke adrese u fizičku - segmentacija



Bezbednosna pitanja



Ako proces nije deo svoje memorije označio kao deljeni, nijedan drugi proces ne sme imati pristup sadržaju tog dela memorije

Ako proces označi da deo memorije mogu deliti drugi navedeni procesi, bezbednosni servis OS-a mora obezbediti da pristup imaju samo ti procesi



Napadi prekoračenjem bafera

- Bezbednosna pretnja vezana za upravljanje memorijom
- Naziva se i buffer overrun
- Može nastati kada proces pokuša da smesti podatke izvan granica bafera fiksne veličine
- Jedna od najčešćih i najopasnijih vrsta bezbednosnih napada



Primer prekoračenja bafera

```
int main(int argc, char *argv[]) {
    int valid = FALSE;
    char str1[8];
    char str2[8];

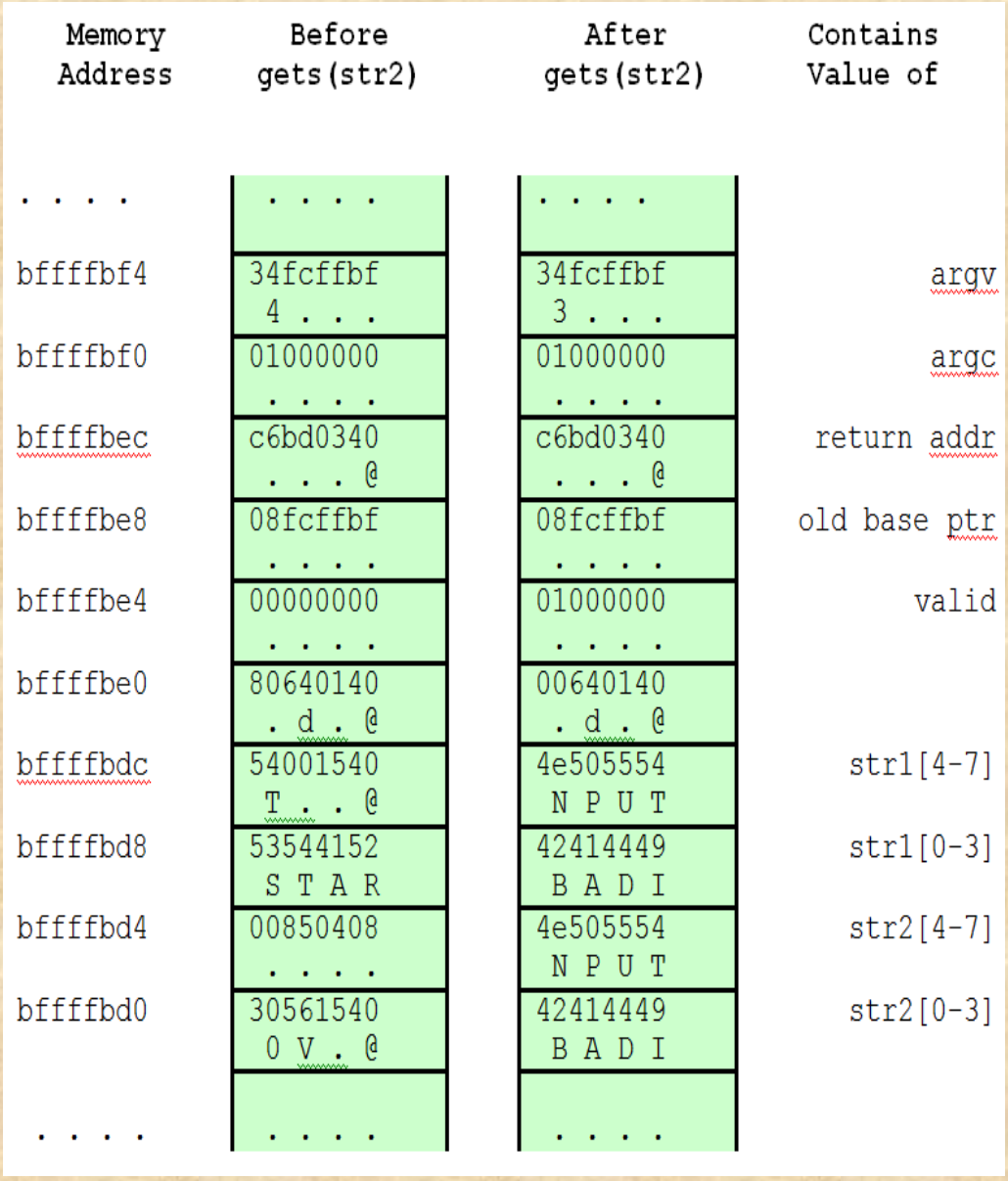
    next_tag(str1);
    gets(str2);
    if (strncmp(str1, str2, 8) == 0)
        valid = TRUE;
    printf("buffer1: str1(%s), str2(%s), valid(%d)\n", str1, str2, valid);
}
```

(a) Basic buffer overflow C code

```
$ cc -g -o buffer1 buffer1.c
$ ./buffer1
START
buffer1: str1(START), str2(START), valid(1)
$ ./buffer1
EVILINPUTVALUE
buffer1: str1(TVALUE), str2(EVILINPUTVALUE), valid(0)
$ ./buffer1
BADINPUTBADINPUT
buffer1: str1(BADINPUT), str2(BADINPUTBADINPUT), valid(1)
```

(b) Basic buffer overflow example runs

Stek pri prekoračenju bafera



Memory Address	Before gets(str2)	After gets(str2)	Contains Value of
. . . .	. . . .	. . . .	
bffffbf4	34fcffbf 4 . . .	34fcffbf 3 . . .	argv
bffffbf0	01000000	01000000	argc
bffffbec	c6bd0340 . . . @	c6bd0340 . . . @	return addr
bffffbe8	08fcffbf	08fcffbf	old base ptr
bffffbe4	00000000	01000000	valid
bffffbe0	80640140 . d . @	00640140 . d . @	
bffffbdc	54001540 T . . @	4e505554 N P U T	str1[4-7]
bffffbd8	53544152 S T A R	42414449 B A D I	str1[0-3]
bffffbd4	00850408	4e505554 N P U T	str2[4-7]
bffffbd0	30561540 0 V . @	42414449 B A D I	str2[0-3]
. . . .	. . . .	. . . .	

Odbrana od prekoračenja bafera

- Sprečavanje
- Otkrivanje i prekidanje
- Kategorije kontramera:



Odbrana pri prevođenju

- cilj je ojačati nove programe protiv napada

Odbrana pri izvršavanju

- cilj je otkriti i prekinuti napade u postojećim programima

Rezime

- Upravljanje memorijom
  - jedan od najvažnijih i najsloženijih zadataka operativnog sistema
  - memoriju treba tretirati kao resurs koji se dodeljuje i deli između više aktivnih procesa
  - poželjno je držati što više procesa u glavnoj memoriji
  - poželjno je osloboditi programere ograničenja veličine pri razvoju programa
  - osnovni alati su straničenje i segmentacija (mogu se kombinovati)
    - straničenje – male stranice fiksne veličine
    - segmentacija – delovi promenljive veličine