



Univerza v Ljubljani
Fakulteta *za računalništvo
in informatiko*

Univerzitetni študijski program, 3. letnik

Sistemska programska oprema

predavatelj: doc. Tomaž Dobravec

Nalagalnik

Nalagalnik

Osnovna naloga nalagalnika

- ▶ Nalagalnik naloži program v pomnilnik in poskrbi, da se program začne izvajati.

Vrste nalagalnikov

- ▶ Absolutni nalagalnik
Nalaganje na vnaprej določen naslov
- ▶ Nalagalnik s prenaslavljanjem
Nalaganje na naslov, ki ga pridobi od OS + popravljanje neposrednih naslovov
- ▶ Dinamični nalagalnik
Nalaganje v času izvajanja

Absolutni nalagalek

- ▶ Absolutni nalagalek naloži absolutne programe
- ▶ Nalagalek naslov absolutnih programov je za vse kontrolne sekcije določen v času pisanja programa.
- ▶ Do konca so razrešeni vsi naslovi.

Naloga absolutnega nalagaleka je, da

- ▶ prebere objektno datoteko,
- ▶ pretvori morebitne tekstovne zapise v binarne, 05 → 5
- ▶ objektno kodo naloži na predvidene lokacije in
- ▶ začne izvajati program.

Absolutni nalagalik - primer

```

HCOPY 00100000107A
T0010001E1410334820390010362810303010154820613C100300102A0C103900102D
T00101E150C10364820610810334C0000454F46000003000000
T0020391E041030001030E0205D30203FD8205D2810303020575490392C205E38203F
T0020571C1010364C0000F1001000041030E02079302064509039DC20792C1036
T002073073820644C000005
E001000

```

(a) Object program

Memory address	Contents			
0000	xxxxxxxx	xxxxxxxx	xxxxxxxx	xxxxxxxx
0010	xxxxxxxx	xxxxxxxx	xxxxxxxx	xxxxxxxx
⋮	⋮	⋮	⋮	⋮
0FF0	xxxxxxxx	xxxxxxxx	xxxxxxxx	xxxxxxxx
1000	14103348	20390010	36281030	30101548
1010	20613C10	0300102A	0C103900	102D0C10
1020	36482061	0810334C	0000454F	46000003
1030	000000xx	xxxxxxxx	xxxxxxxx	xxxxxxxx
⋮	⋮	⋮	⋮	⋮
2030	xxxxxxxx	xxxxxxxx	xx041030	001030E0
2040	205D3020	3FD8205D	28103030	20575490
2050	392C205E	38203F10	10364C00	00F10010
2060	00041030	E0207930	20645090	39DC2079
2070	2C103638	20644C00	0005xxxx	xxxxxxxx
2080	xxxxxxxx	xxxxxxxx	xxxxxxxx	xxxxxxxx
⋮	⋮	⋮	⋮	⋮

(b) Program loaded in memory

Nalaganje s prenaslavljanjem

Nalaganje s prenaslavljanjem uporabimo takrat, ko nalagamo prenaslovljiv modul.

Naloga nalagalnika je, da

1. od operacijskega sistema pridobi nalagalni naslov
2. prebere objektni modul in ga naloži na pravilni naslov;
naslov objektne kode izračuna po formuli

$$\text{naslov_kode} = \text{og_naslov_kode} + \text{nalagalni_naslov}$$

Nalaganje s prenaslavljanjem

3. popravi dele programske kode v skladu z navodili, ki so zapisana v prilagoditvenih zapisih; pri tem primerno prilagaja naslove podane v modifikacijskih zapisih.

(P) M 001061 05 + COPY₁₁
↓ nalagalni naslov = 0xA100
M 00B161 05 + A100

Nalaganje s prenaslavljanjem

4. izvajanje programa preusmeri na naslov, podan v E zapisu;
pri tem podanemu naslovu prišteje nalagalni naslov

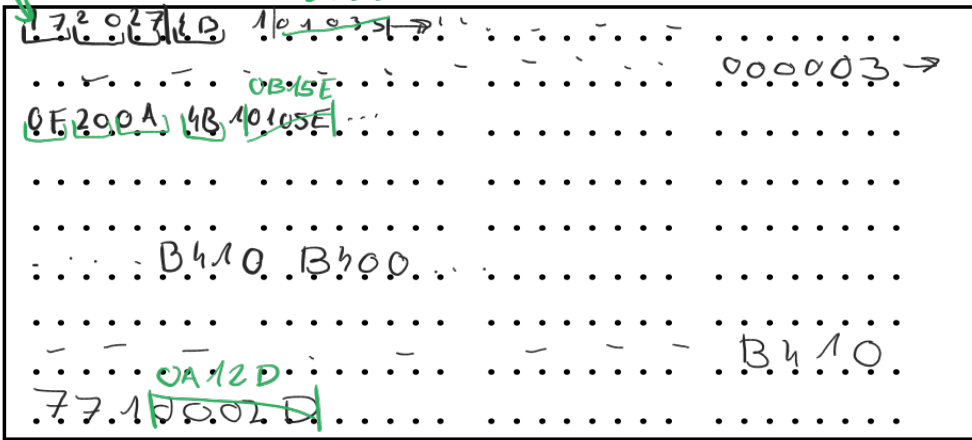
(P) E 000000
↓ + nalagalni naslov
E 00A100

Nalaganje s prenaslavljanjem - primer

Prenaslovljiv program COPY “naloži” v pomnilnik na lokacijo 0x0A100.

```
HCOPY 000000 00107A
T 000000 1D 172027 4B101033 032023 ...
T 00001D 0D 000003 0f200A 4B10105E ...
...
T 001033 1D B410 B400 ...
T 001050 0E 3B2FE9 13100000 4F0000 F1 001000
T 00105E 1C B410 7710002D E32012 ...
E 000000
```

```
M 000004 05 +COPY
M 000024 05 +COPY
M 001061 05 +COPY
```

Naslov	Vsebina			
0000	xxxxxxxx	xxxxxxxx	xxxxxxxx	xxxxxxxx
...	PC
A0F0	xxxxxxxx	xxxxxxxx	xxxxxxxx	xxxxxxxx
A100	<div></div>			
A110				
A120				
:				
B130				
:				
B150				
B160				
...



Dinamični nalagalek

Kako ravnati v primeru, ko je velikost programa večja od razpoložljivega pomnilnika?

- a) Uporaba navideznega pomnilnika
 - ▶ potrebujemo podporo OS
 - ▶ izvede se odstranjevanje ali segmentacija

- b) Dinamično nalaganje
 - ▶ hkrati so v pomnilniku le medseboj odvisni deli programa
 - ▶ neodvisni deli programa lahko zasedejo ISTO pomnilniško lokacijo

```

A() {
  B();
  :
  E();
}

```

```

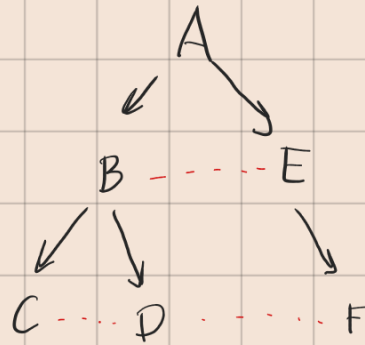
B() {
  :
  C();
  :
  D();
}

```

```

E() {
  F();
  :
}

```



A	B	C	
	E	D	
		F	

Dinamični nalagalniki

Naloge prevajalnika

- ▶ detektira neodvisne dele kode,
- ▶ namesto klicev prekrivajočih delov programa vključi klic dinamičnega nalagalnika

Naloga povezovalnika

- ▶ če dinamičen nalagalec ni del OSa, mora povezovalnik kodo dinamičnega nalaganja vključiti v kodo prevedenega programa.

Naloge dinamičnega nalagalnika:

- ▶ naloži del kode na predvideno mesto,
- ▶ popravi morebitne neposredne naslove,
- ▶ kliče dinamično naloženo kodo.

Začetni nalagalek

Kaj se zgodi, ko se računalnik “zbudi”?

- ▶ Načeloma: ob zagonu je računalnik prost (idle) brez naloženih programov v pomnilniku.
- ▶ Kdo potemtako sproži začetek vsega dogajanja (prvo nalaganje)?

Kje je začetek?

- ▶ Kdo naloži nalagalek?
- ▶ Kdo pa naloži OS?
- ▶ Kdo pa naloži nalagalek za OS?
- ▶ Kdo pa naloži nalagalek za nalagalek za OS?

Začetni nalagalnik

- ▶ Da celoten postopek nalaganja sploh začne teči, potrebujemo preprost nalagalnik, ki naloži prvi program.
- ▶ Temu nalagalniku rečemo **začetni nalagalnik** (angl. *bootstrap loader*).
- ▶ Začetni nalagalnik poskrbi, da se začne večfazni postopek začetnega nalaganja. Pri tem se v pomnilnik drug za drugim vnašajo pomožni (vedno bolj kompleksni) nalagalniki, ki prevzamejo nadaljne naloge.

Začetni nalagalek

Osnovno vprašanje ostaja: Kdo naloži **začetni** nalagalek?

V praksi se pojavljajo tri rešitve:

- a) Ob zagonu operater z uporabo prilagojene strojne opreme vnese prve ukaze
 - ▶ zgodovina
 - ▶ lahko pride do napak
- b) Celoten začetni nalagalek je shranjen v ROMu; ob zagonu računalnika se koda prepíše v RAM in izvajanje se prenese tja
 - ▶ slabost: začetnega nalagaleka ne moremo spreminjati



Začetni nalagalek

- c) Začetni nalagalek je sestavni del interne procesorjeve logike izveden kot poseben ukaz ali zelo kratek program v ROMu
 - ▶ Izvajanje tega ukaza oziroma tega programa iz ROMa povzroči branje programske kode **vnaprej določene dolžine iz vnaprej določene naprave na vnaprej določen naslov v pomnilniku**,
 - ▶ po branju se izvajanje prenese na prebrano kodo.
 - ▶ Običajno se nalagalniki nalagajo postopoma: zelo preprost nalagalek naloži malo bolj kompleksen nalagalek, ta naloži še kompleksnejšega, ...

Začetno nalaganje pri Intelovih procesorjih

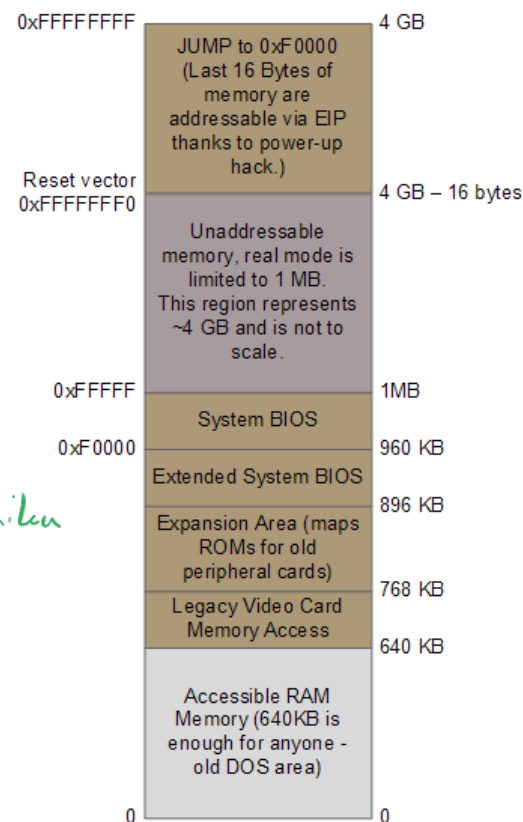
- ▶ Pri Intelovih procesorjih (od i80286 naprej) se zgornjih 64K naslovnega prostora (pod 1MB) imenuje **sistemska inicializacijsko področje (SIP)**
- ▶ v SIP se v postavi stalni pomnilnik z
 - ▶ zagonskimi in diagnostičnimi funkcijami (POST) ter
 - ▶ osnovni vhodno-izhodni sistem (BIOS).
- ▶ ob zagonu računalnika:

$PC := \text{NAJVIŠJI_NASLOV} - 16$

▶ $0xFFFFFFFF - 0x00000010 = 0xFFFFFFF0$ ← v 32-bit pomnilniku

na tem naslovu je zapisan ukaz za skok SIP - najprej se izvede POST nato pa se s pomočjo prekinitve 0x19 sproži osnovno začetno nalaganje.

Podrobneje: <http://duartes.org/gustavo/blog/post/how-computers-boot-up>



Glavni nalagalni sektor

- ▶ BIOS omogoča le fizičen (ne pa tudi logičnega) dostop do zunanjih enot;
- ▶ začetnega nalagalnika zato ne moremo prebrati iz datoteke;
- ▶ začetni nalagalnik se nahaja v vnaprej določenem sektorju posamezne enote;
- ▶ pri osebnih računalnikih se kot stalni sektor uporablja prvi sektor na enoti. Imenuje se glavni nalagalni sektor ali MBR (master boot record).

Glavni nalagalni sektor

V MBR so poleg programske kode, ki sproži postopek nalaganja, shranjene tudi naslednje informacije:

- ▶ dopolnilni program za izbiro operacijskega sistema (opsijsko),
- ▶ podatki o lastnostih nalagalne naprave oziroma BIOSov blok parametrov (BIOS parameter block, BPB),
- ▶ informacije o particijah naprave.

Glavni nalagalni sektor

0000	EB 52 90 4E 54 46 53 20 20 20 20 00 02 08 00 00	äRINTFS
0010	00 00 00 00 00 00 F8 00 00 3F 00 FF 00 3F 00 00 00?..?...
0020	00 00 00 00 00 80 00 80 00 FD 25 9C 00 00 00 00 00€..€..?*
0030	04 00 00 00 00 00 00 00 00 5F C2 09 00 00 00 00 00&.....
0040	F6 00 00 00 00 01 00 00 00 5E EE 3A D8 12 3B D8 98	ö.....^i:Ø.;Ø-
0050	00 00 00 00 00 FA 33 C0 8E D0 BC 00 7C FB B8 C0 07ú3ÀØBh.1ú.À.
0060	8E D8 E8 16 00 B8 00 0D 8E C0 33 DB C6 06 0E 00	00è.....0À30E...
0070	10 E8 53 00 68 00 0D 68 6A 02 CB 8A 16 24 00 B4	.èS.h..hj.E8.¿.
0080	08 CD 13 73 05 B9 FF FF 8A F1 66 0F B6 C6 40 66	.f.s..1..8Kf.qE@f
0090	0F B6 D1 80 E2 3F F7 E2 86 CD C0 ED 06 41 66 0F	.qN€â?+â+íÀí.Af.
00A0	B7 C9 66 F7 E1 66 A3 20 00 C3 B4 41 BB AA 55 8A	·Ef+áfE .X·A»²U8
00B0	16 24 00 CD 13 72 0F 81 FB 55 AA 75 09 F6 C1 01	.¿.f.r.DGUV²u.8Á.
00C0	74 04 FE 06 14 00 C3 66 60 1E 06 66 A1 10 00 66	t.p...Xf`..f;..f
00D0	03 06 1C 00 66 3B 06 20 00 0F 82 3A 00 1E 66 6Af;..:..fj
00E0	00 66 50 06 53 66 68 10 00 01 00 80 3E 14 00 00	.fP.Sfh....€>...
00F0	0F 85 0C 00 E8 B3 FF 80 3E 14 00 00 0F 84 61 00è³.€>.....a.
0100	B4 42 8A 16 24 00 16 1F 8B F4 CD 13 66 58 5B 07	·B8.¿....<8f.fX[.
0110	66 58 66 58 1F EB 2D 66 33 D2 66 0F B7 0E 18 00	fXfX.è-f30f.
0120	66 F7 F1 FE C2 8A CA 66 8B D0 66 C1 EA 10 F7 36	f+RpÀ8Ef<BfÁè.+6
0130	1A 00 86 D6 8A 16 24 00 8A E8 C0 E4 06 0A CC B8	..+08.¿.8èAä...ì.
0140	01 02 CD 13 0F 82 19 00 8C C0 05 20 00 8E C0 66	..f...OEÀ. .0Àf
0150	FF 06 10 00 FF 0E 0E 00 0F 85 6F FF 07 1F 66 61mo...fa
0160	C3 A0 F8 01 E8 09 00 A0 FB 01 E8 03 00 FB EB FE	À ~.è...û.è..ûép
0170	B4 01 8B F0 AC 3C 00 74 09 B4 0E BB 07 00 CD 10	·.<8-<t..»...f.
0180	EB F2 C3 0D 0A 41 20 64 69 73 6B 20 72 65 61 64	è8X..A disk read
0190	20 65 72 72 6F 72 20 6F 63 63 75 72 72 65 64 00	error occurred.
01A0	0D 0A 4E 54 4C 44 52 20 69 73 20 6D 69 73 73 69	..NTLDR is missi
01B0	6E 67 00 0D 0A 4E 54 4C 44 52 20 69 73 20 63 6F	ng...NTLDR is co
01C0	6D 70 72 65 73 73 65 64 00 0D 0A 50 72 65 73 73	mpressed...Press
01D0	20 43 74 72 6C 2B 41 6C 74 2B 44 65 6C 20 74 6F	Ctrl+Alt+Del to
01E0	20 72 65 73 74 61 72 74 0D 0A 00 00 00 00 00	restart.....
01F0	00 00 00 00 00 00 00 00 83 A0 B3 C9 00 00 55 AAf ³É...U²

Jump/Nop OEMString BIOS Parameter Block IPL Strings Signature



Glavni nalagalni sektor

BPB vsebuje naslednje podatke (primer za FAT32):

Pomen	Odmik	Dolžina
Oznaka OS	3	8
Število zlogov na sektor	11	2
Število sektorjev v gruči	13	1
Rezervirani sektorji	14	2
Število FAT	16	1
Število korenskih vstopov v kazalu	17	2
Število majhnih sektorje	19	2
Opis medija	21	1
Število sektorjev FAT	22	2
Število sektorjev na stezi	24	2
Število glav	26	2
Število skritih sektorjev	28	4
Število velikih sektorjev	32	4

več: http://en.wikipedia.org/wiki/BIOS_parameter_block

Glavni nalagalni sektor

Kako poteka nalaganje s pomočjo MBR?

- ▶ uporabnik v BIOS nastavi vrstni red nalagalnih enot
- ▶ po izvedbi POST se v BIOSu kliče funkcija (interrupt) 0x19, ki povzroči branje prvega sektorja s prve prisotne nalagalne enote;
- ▶ vsebina sektorja se prenese na naslov 0x0000:7C00, izvajanje se preusmeri na 0000:7C00
- ▶ če na izbrani nalagalni enoti ni operacijskega sistema,
 - ▶ koda v MBR izpiše sporočilo “Non-system disk or disk error”,
- ▶ sicer
 - ▶ aktivira se rutina za izbor operacijskega sistema (npr. LILO ali GRUB), katere osnovni del je v MBR.
- ▶ z izbiro OSa uporabnik določi particijo, iz katere se v nadaljevanju prebere BR (boot record); MBR prebere ta BR na naslov 0x0000:7C00 in izvajanje preseli nanj; koda tega BRja poskrbi za nalaganja OSa.

Glavni nalagalni sektor

Branje in prikaz MBR v Linux sistemih

- ▶ preberi MBR na svojem računalniku

```
dd if=/dev/ime_diska of=backup.mbr bs=512 count=1
```

- ▶ prikaži MBR v hex obliki

```
hexdump -C backup.mbr
```

- ▶ na podlagi podatkov iz BPB izračunaj število sektorjev na disku

- ▶ poglej, kaj izpiše program `file`

```
file backup.mbr
```

Glavni nalagalni sektor

Programska koda prebranega MBRja (Linux, FAT32)

```
00007C5A  FA      cli
00007C5B  31C0    xor ax,ax
00007C5D  8ED0    mov ss,ax
00007C5F  BC007C  mov sp,0x7c00
00007C62  FB      sti
00007C63  8ED8    mov ds,ax
00007C65  E80000  call 0x7c68    //push PC
00007C68  5E      pop si        //si=PC
00007C69  83C619  add si,byte +0x19 //si=si+0x19
00007C6C  BB0700  mov bx,0x7
00007C6F  FC      cld
00007C70  AC      lodsb        // v al da en bajt iz sp:si
00007C71  84C0    test al,al     // konec niza?
00007C73  7406    jz 0x7c7b
00007C75  B40E    mov ah,0xe
00007C77  CD10    int 0x10       // izpiše znak
00007C79  EBF5    jmp short 0x7c70 // ponovi
00007C7B  30E4    xor ah,ah
00007C7D  CD16    int 0x16       // read keyboard
00007C7F  CD19    int 0x19       // boot strap
```