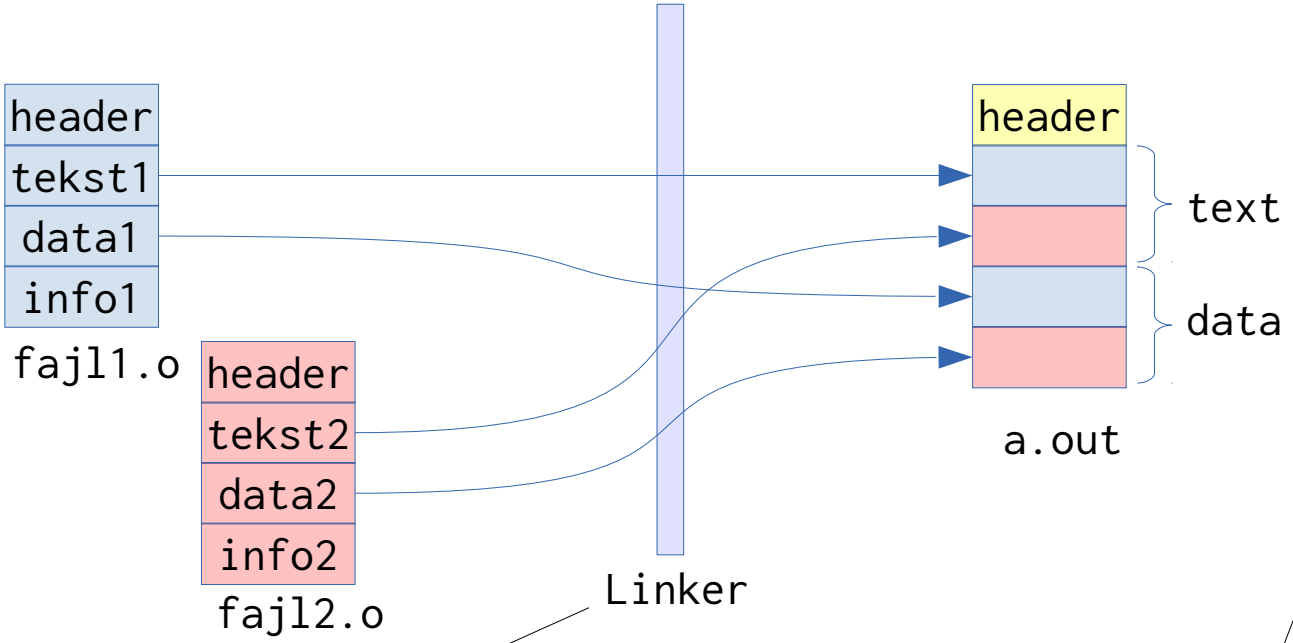
Operativni sistemi

dr.sc. Amer Hasanović




```
OBJS = \
	bio.o\
	console.o\
	exec.o\
	file.o\
	fs.o\
	ide.o\
	ioapic.o\
	kalloc.o\
	kbd.o\
	lapic.o\
	log.o\
	main.o\
	mp.o\
	picirq.o\
	pipe.o\
	proc.o\
	spinlock.o\
	string.o\
	swtch.o\
	syscall.o\
	sysfile.o\
	sysproc.o\
	timer.o\
	trapasm.o\
	trap.o\
	uart.o\
	vectors.o\
	vm.o\
```

```
ld -m elf_i386 -T kernel.ld -o kernel entry.o $(OBJS)
```

izlazni format

Linker skripta

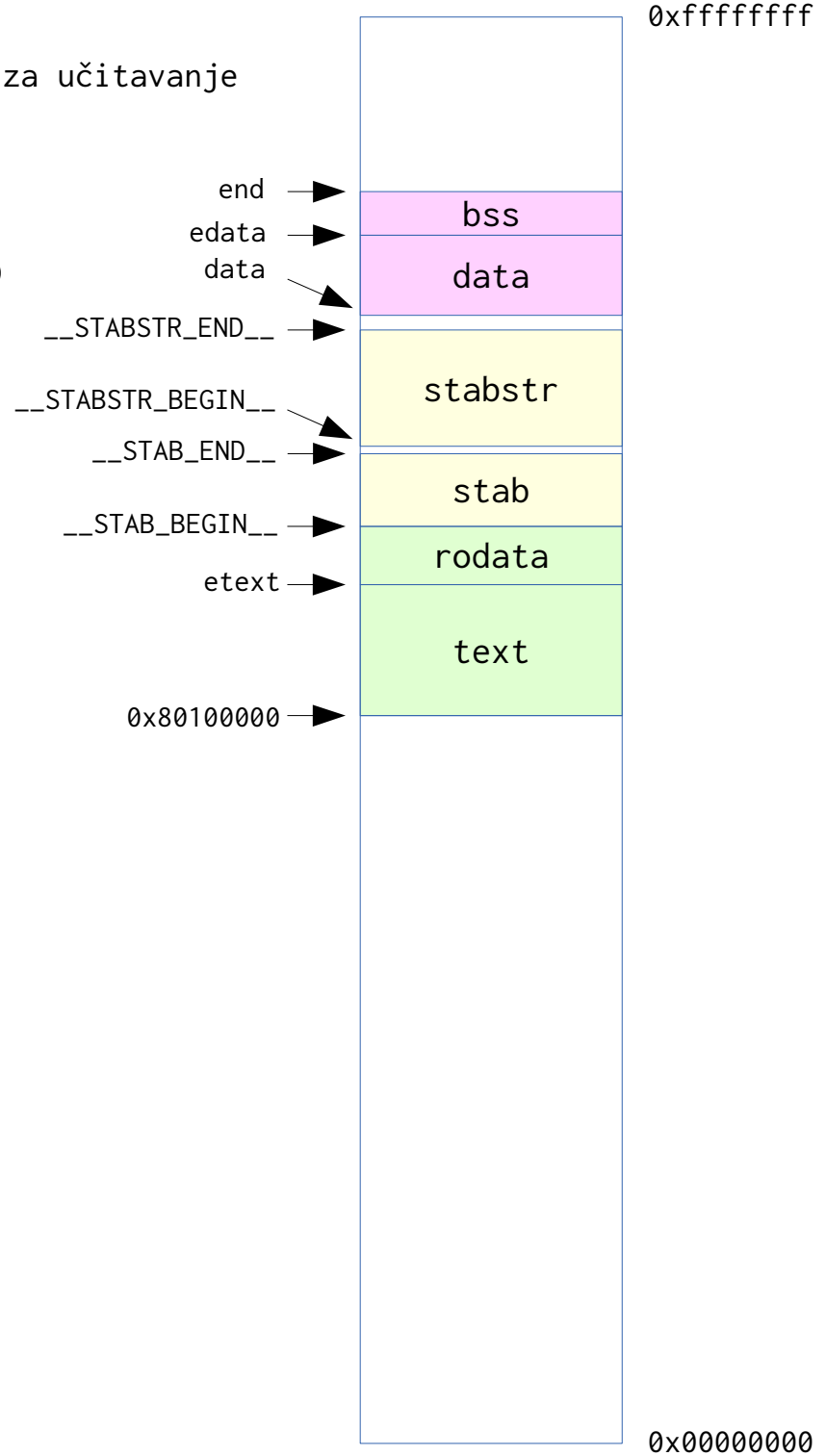
Izlazni fajl

Lista ulaznih objektnih fajlova

kernel.ld

```
OUTPUT_ARCH(i386)
ENTRY(_start)
SECTIONS
{
  . = 0x80100000;
  .text : AT(0x100000) {
     *(.text .stub .text.* .gnu.linkonce.t.*)
  }
  PROVIDE(etext = .);
  .rodata : {
     *(.rodata .rodata.* .gnu.linkonce.r.*)
  }
  .stab : {
     PROVIDE(__STAB_BEGIN__ = .);
     *(.stab);
     PROVIDE(__STAB_END__ = .);
     BYTE(0)
  }
  .stabstr : {
     PROVIDE(__STABSTR_BEGIN__ = .);
     *(.stabstr);
     PROVIDE(__STABSTR_END__ = .);
     BYTE(0)
  }
  . = ALIGN(0x1000);
  PROVIDE(data = .);
  .data : {
     *(.data)
  }
  PROVIDE(edata = .);
  .bss : {
     *(.bss)
  }
  PROVIDE(end = .);
  /DISCARD/ : {
     *(.eh_frame .note.GNU-stack)
  }
}
```

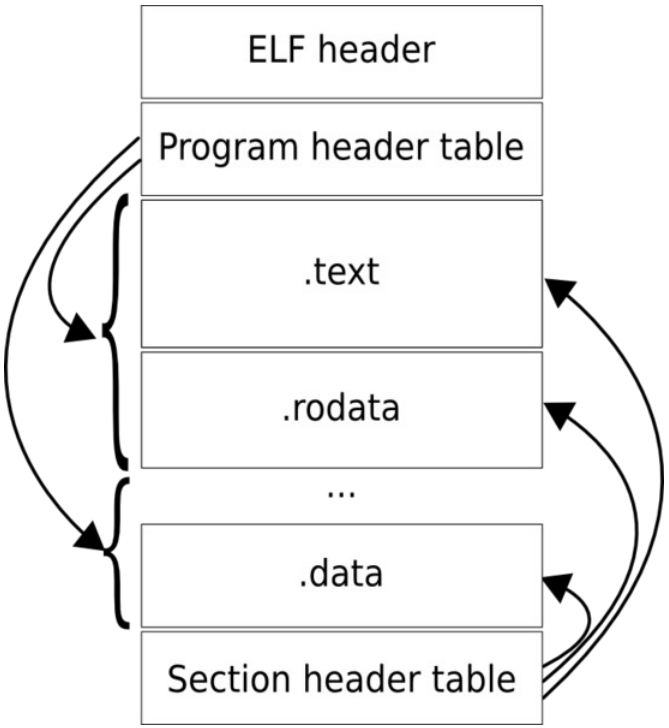
Fizička adresa za učitavanje



objdump -h kernel

```
kernel:     file format elf32-i386

Sections:
Idx Name          Size      VMA       LMA       File off  Algn
  0 .text         0000839a  80100000  00100000  00001000  2**2
                  CONTENTS, ALLOC, LOAD, READONLY, CODE
  1 .rodata       00000682  8010839c  0010839c  0000939c  2**2
                  CONTENTS, ALLOC, LOAD, READONLY, DATA
  2 .stab         00000001  80108a1e  00108a1e  00009a1e  2**0
                  CONTENTS, ALLOC, LOAD, DATA
  3 .stabstr      00000001  80108a1f  00108a1f  00009a1f  2**0
                  CONTENTS, ALLOC, LOAD, DATA
  4 .data         00002596  80109000  00109000  0000a000  2**12
                  CONTENTS, ALLOC, LOAD, DATA
  5 .bss          00009b9c  8010b5a0  0010b5a0  0000c596  2**5
```



readelf -l kernel

```
Elf file type is EXEC (Executable file)
Entry point 0x10000c
There are 2 program headers, starting at offset 52

Program Headers:
  Type           Offset   VirtAddr   PhysAddr   FileSiz MemSiz  Flg Align
  LOAD           0x001000 0x80100000 0x00100000 0x0b596 0x1513c RWE 0x1000
  GNU_STACK      0x000000 0x00000000 0x00000000 0x00000 0x00000 RWE 0x10

 Section to Segment mapping:
  Segment Sections...
   00     .text .rodata .stab .stabstr .data .bss
   01
```

MMU straničenje (paging)

Mapiranje adresnih prostora

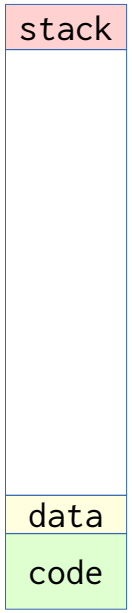
- Neka je data funkcija f:
  - y = f(x)
    - x → adresa byte-a u nekom virtuelnom adresnom prostoru
    - y → adresa u fizičkom prostoru gdje se nalazi željeni podatak adresiran u virtuelnom prostoru
- Funkcija f mapira virtuelni prostor u fizički prostor i može se implementirati procesom straničenja (paging)

Straničenje

- Mehanizam koji omogućava:
  - multipleksiranje više virtuelnih adresnih prostora u jedan fizički;
  - protekciju dijelova fizičke memorije;
  - mapiranje dijelova fizičke memorije na više lokacija u jednom virtuelnom adresnom prostoru, i/ili u više različitih adresnih prostora
- Realizira se particiranjem memorije na stranice (pages) → komade memorije istog broja byte-a:

Fizička memorija

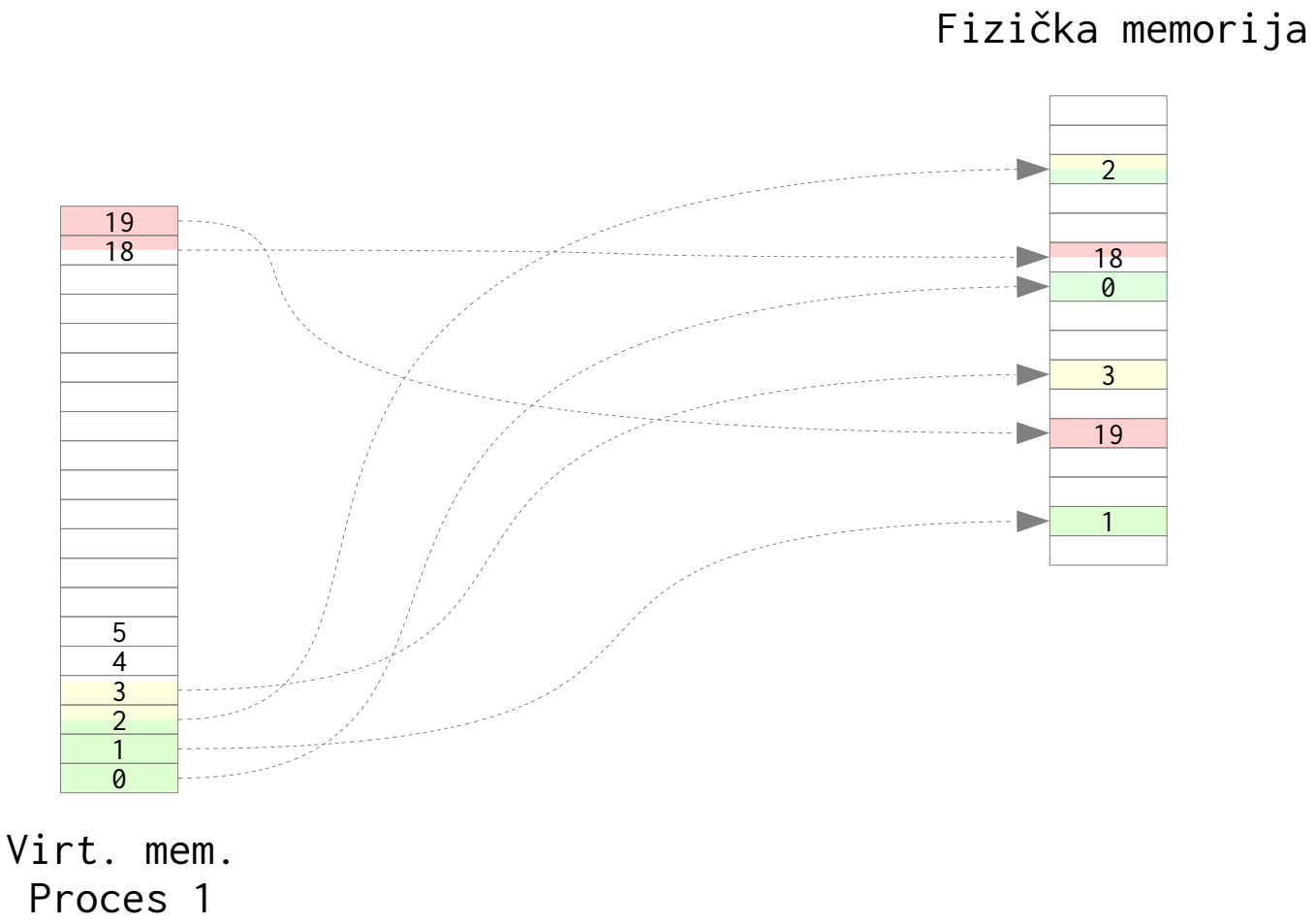
Fizička memorija



Virt. mem.
Proces 1

Fizička memorija

Virt. mem.
Proces 1


Fizička memorija
19
18
5
4
3
2
1
0
2
18
0
3
19
1
Virt. mem.
Proces 1

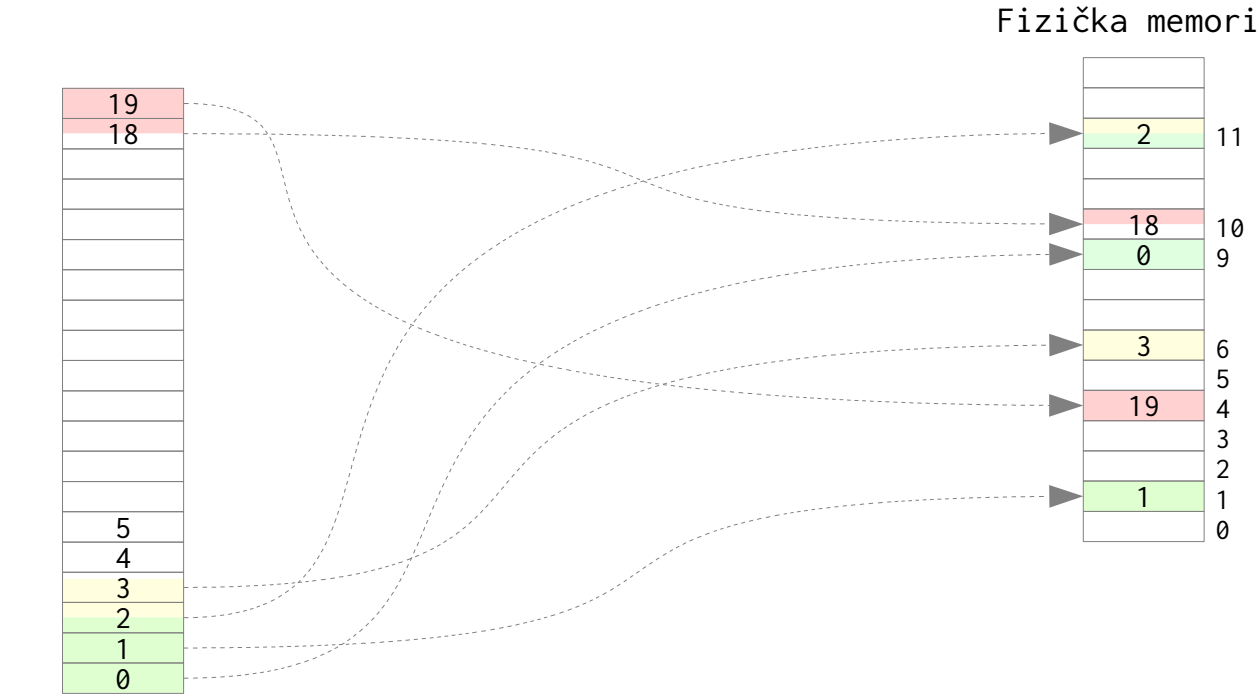
Tabele stranica

- Translacija jednog virtuelnog prostora u fizički vrši se putem tabele stranica (Page Table → PT)
  - PT se nalazi u memoriji;
  - PT ima onoliko redova (Page Table Entry → PTE) koliko ima stranica u virtuelnoj memoriji;
    - PTE
      - sadrži broj fizičke stranice (frame) u koju je virtuelna stranica mapirana
      - sadrži i dodatne bite koji govore o bitnim karakteristikama stranice (prisutna, zaštita od pisanja, …)

- Svaki proces dobije svoj PT
- Procesor u određenom registru čuva adresu PT-a procesa kojeg trenutno izvršava.

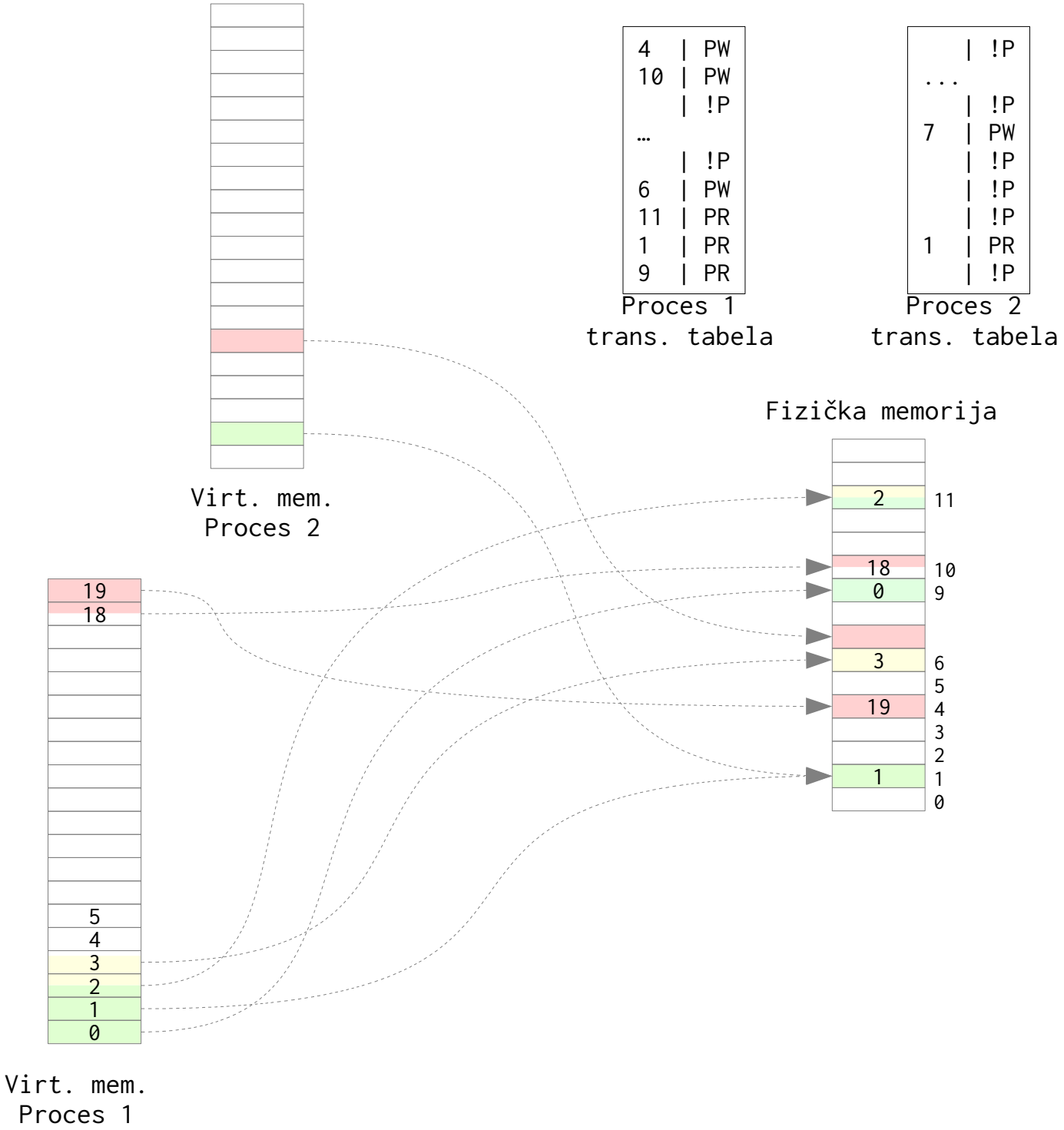
4	PW
10	PW
	!P
…	
	!P
6	PW
11	PR
1	PR
9	PR

Proces 1
trans. tabela



Virt. mem.
Proces 1

0


4 | PW
10 | PW
| !P
…
| !P
6 | PW
11 | PR
1 | PR
9 | PR
Proces 1
trans. tabela
| !P
...
| !P
7 | PW
| !P
| !P
| !P
1 | PR
| !P
Proces 2
trans. tabela
Fizička memorija
2
11
18
10
0
9
3
6
5
19
4
3
2
1
1
0
Virt. mem.
Proces 2
19
18
5
4
3
2
1
0
Virt. mem.
Proces 1

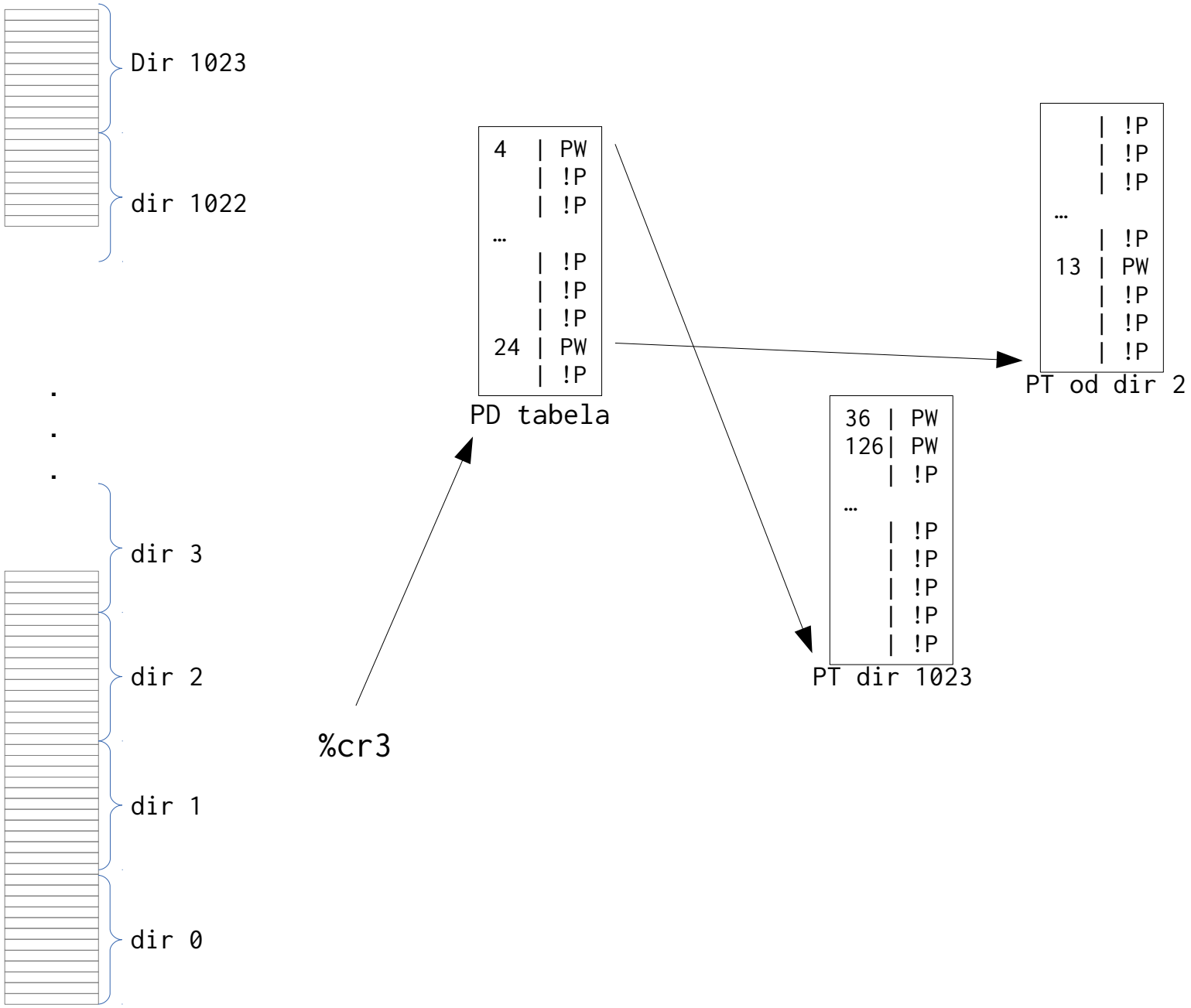
0

x86 i straničenje

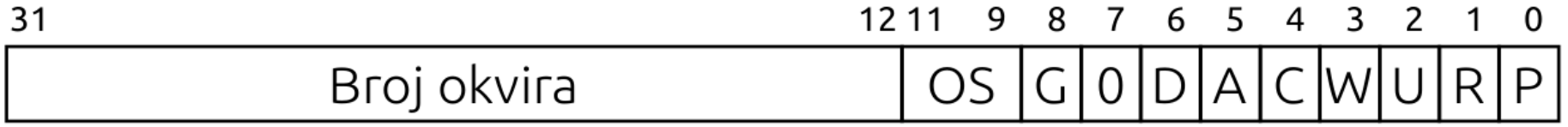
- Intel arhitektura:
  - omogućava aktiviranje straničenje promjenom bita `#32` u registru `cr0`;
  - funkcija straničenja operira nakon funkcije segmentiranja, tj straničenje radi na linearnim adresama;
  - istovremeno podržava stranice u dvije veličine stranica: `4KB` i `4MB`
    - bit `#5` registra `cr0` aktivira podršku za `4MB` stranice
  - jedan PTE zauzima 32 bita

X86 tranlacija adresa

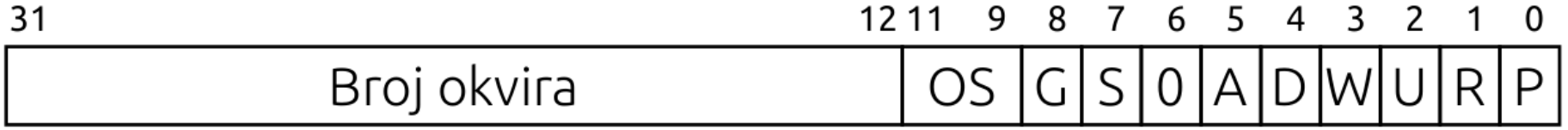
- Virtuelna memorija izdjeljena je na direktorije (1024 direktorija)
- Svaki direktorij podijeljen je na stranice (1024 stranice)
- Umjesto jednog globalnog PT-a svaki direktorij ima lokalni PT za translaciju stranica koje pripadaju tom direktoriju
  - ovakav PT zauzima 4 KB RAM-a (tj jednu fizičku stranicu)
- Svaki proces dobija jednu tabelu direktorija (Page Directory Table → PDT)
  - red u PDT-u (Page Directory Entry → PDE) zauzima 32 bita
  - PDT sadrži adresu stanice u kojoj se nalazi PT za mapiranje virtuelnih stranica koje pripadaju tom direktoriju, i dodatne bite koji opisuju karakteristike direktorija


Dir 1023
dir 1022
.
.
.
dir 3
dir 2
dir 1
dir 0
4 | PW
| !P
| !P
…
| !P
| !P
| !P
24 | PW
| !P
PD tabela
%cr3
| !P
| !P
| !P
…
| !P
13 | PW
| !P
| !P
| !P
PT od dir 2
36 | PW
126| PW
| !P
…
| !P
| !P
| !P
| !P
| !P
PT dir 1023

PDE i PTE format

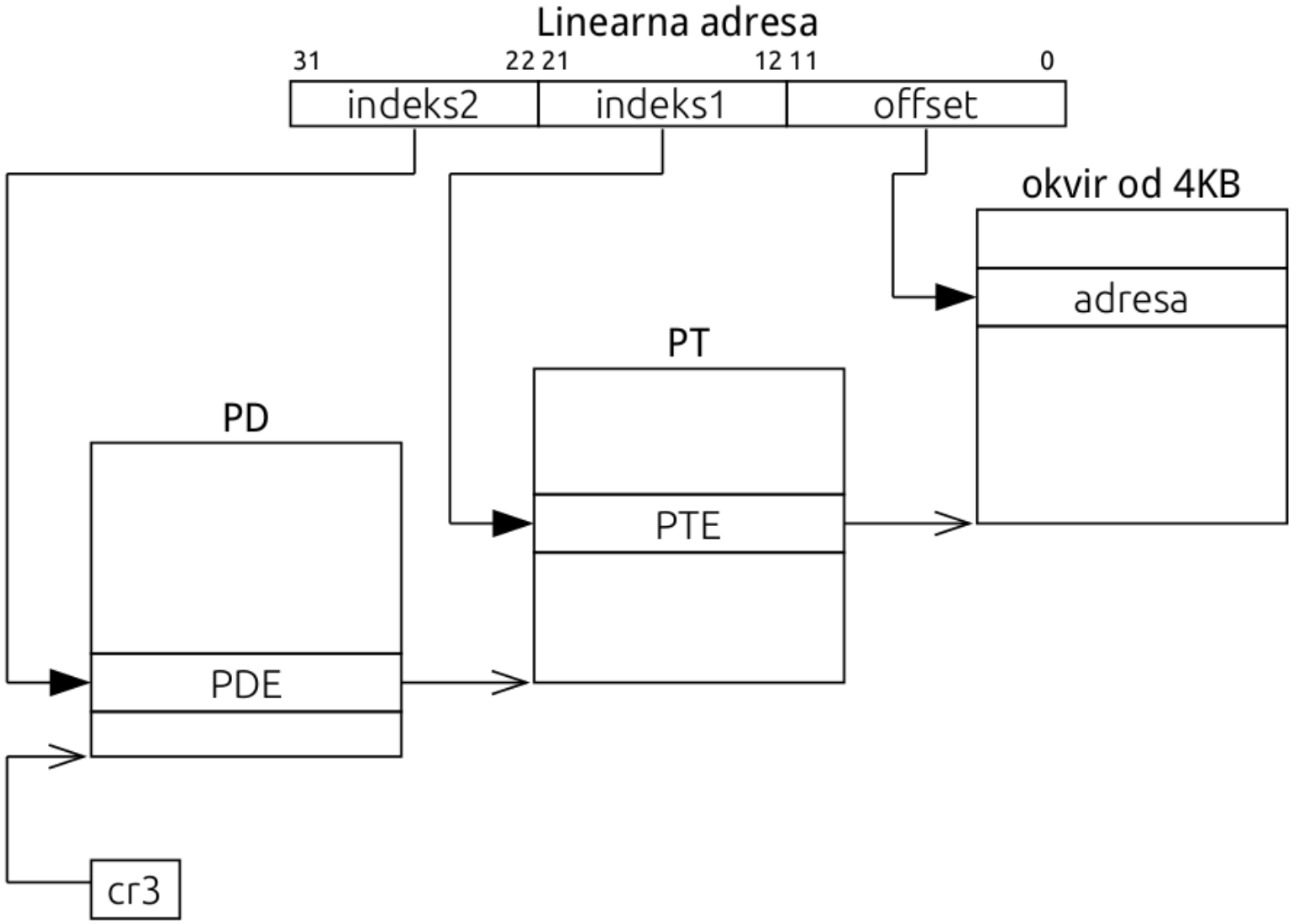


PTE

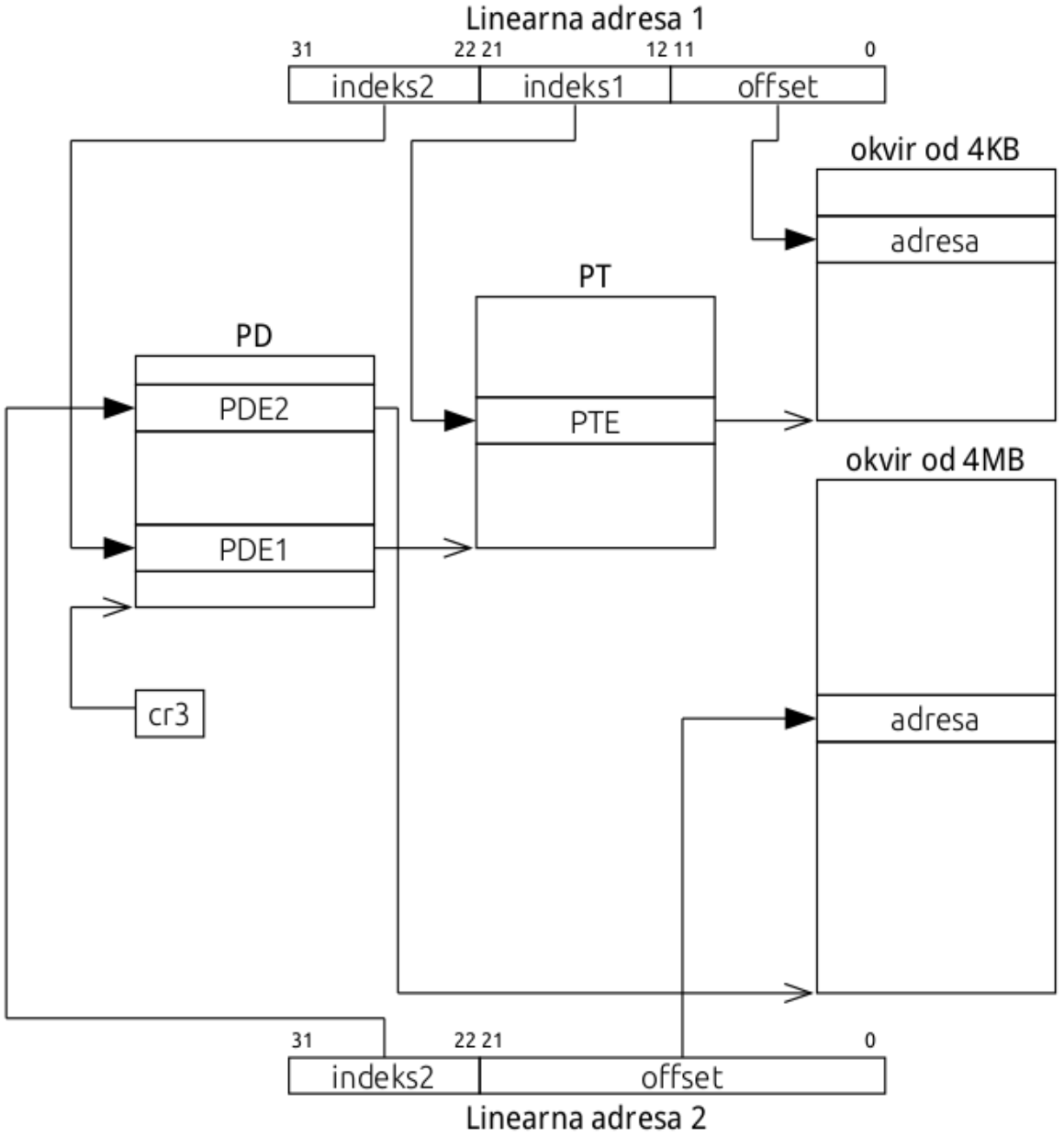


PDE

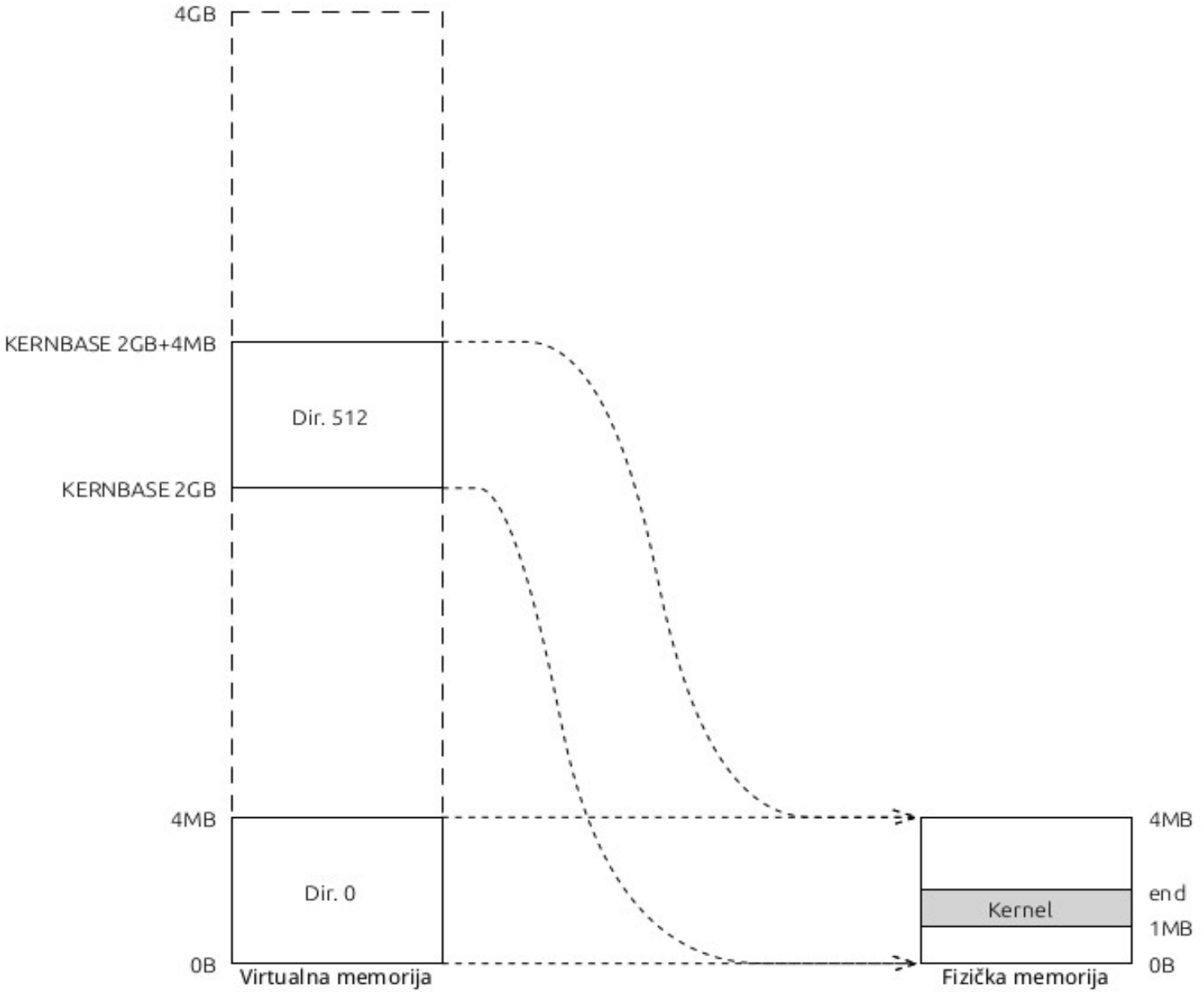
Translacija adresa sa 4KB stranicama



Translacija adresa sa 4KB i 4MB stranicama



Inicijalni virtuelni adresni prostor kernela



entry.S

```
#include "asm.h"
#include "memlayout.h"
#include "mmu.h"
#include "param.h"
.p2align 2
.text
.globl multiboot_header
multiboot_header:
  .long magic
  .long flags
  .long (-magic-flags)
.globl _start
_start = V2P_WO(entry)

.globl entry
entry:
  movl    %cr4, %eax
  orl     $(CR4_PSE), %eax
  movl    %eax, %cr4
  # Set page directory
  movl    $(V2P_WO(entrypgdir)), %eax
  movl    %eax, %cr3
  # Turn on paging.
  movl    %cr0, %eax
  orl     $(CR0_PG|CR0_WP), %eax
  movl    %eax, %cr0
  # Set up the stack pointer.
  movl $(stack + KSTACKSIZE), %esp
  mov $main, %eax
  jmp *%eax
.comm stack, KSTACKSIZE
```

```
#define KERNBASE 0x80000000

#define V2P_WO(x) ((x) - KERNBASE)
#define P2V_WO(x) ((x) + KERNBASE)
```

```
typedef uint pde_t;
```

```
pde_t entrypgdir[NPDENTRIES] = {
  [0] = (0) | PTE_P | PTE_W | PTE_PS,
  [KERNBASE>>PDXSHIFT] = (0) | PTE_P | PTE_W | PTE_PS,
};
```

```
#define NPDENTRIES      1024
#define PDXSHIFT        22
```

```
#define KSTACKSIZE 4096
```