

List of Figures

1	OS overview	2
2	Motherboard chipsets	3
3	Motherboard chipsets (bw version)	3
4	CPU's working cycle	3
5	Bootstrapping	4
6	Bootstrapping (bw version)	4
7	Process' virtual address space	5
8	UNIX view of a process	6
9	Process creation	6
10	Thread operations	6
11	Deadlock — Resource issues	7
12	Deadlock notions	7
13	Deadlock — Banker algorithm	7
14	Deadlock avoidance	8
15	Deadlock avoidance	8
16	Producers and consumers	9
17	Producers and consumers (bw version)	9
18	A circular array	10
19	A circular array (bw version)	10
20	Real mode memory layouts	11
21	Tool chain	12
22	Relocation	13
23	First fit, best fit, worst fit	14
24	Memory fragmentation	14
25	Two-level paging	15
26	Memory segmentation	16
27	Memory segmentation — Address translation	17
28	File system tables	18
29	File tables	18
30	File tables	19
31	VFS objects	19
32	VFS objects	20
33	VFS file copy	20
34	VFS file copy (bw version)	20
35	DMA handshaking	21
36	Handshaking	21

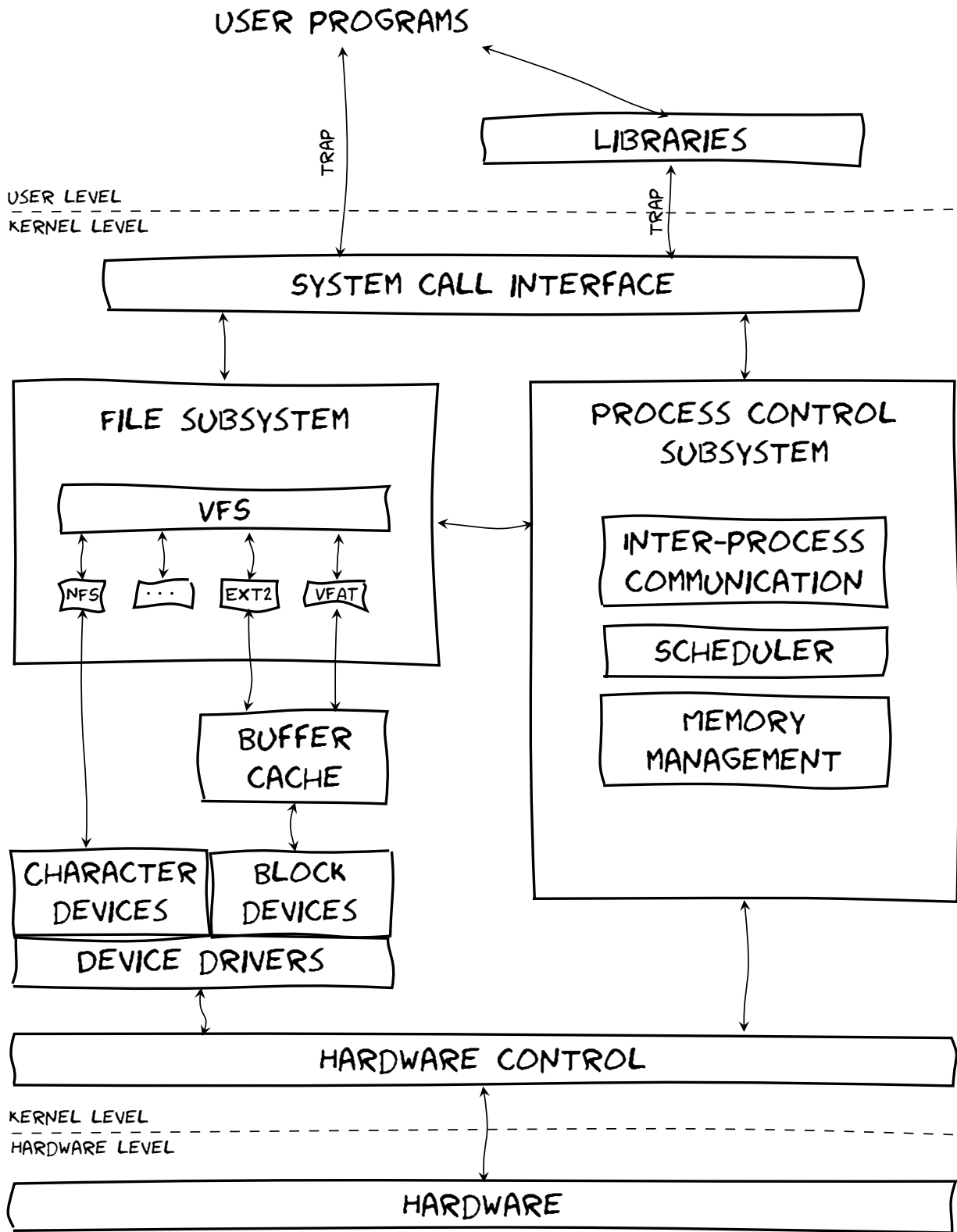


Fig. 1: OS overview

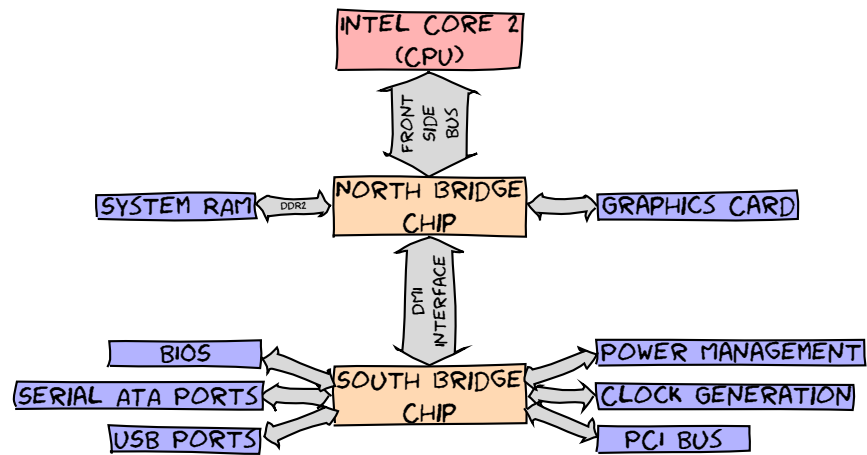


Fig. 2: Motherboard chipsets

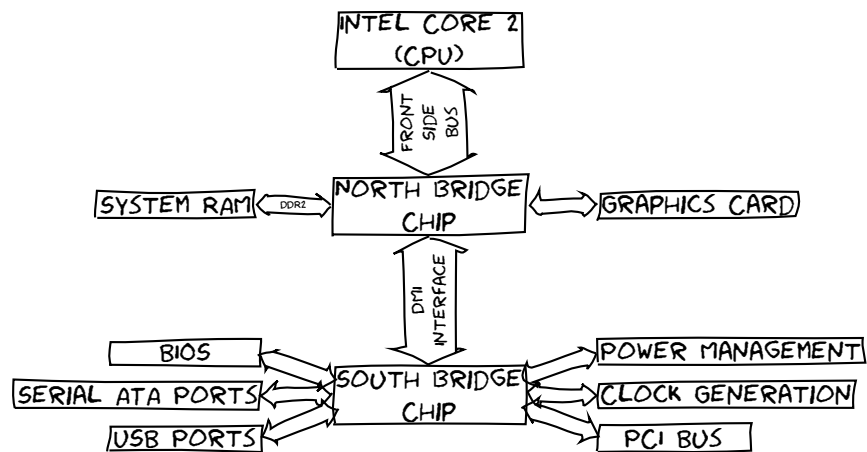


Fig. 3: Motherboard chipsets (bw version)



Fig. 4: CPU's working cycle

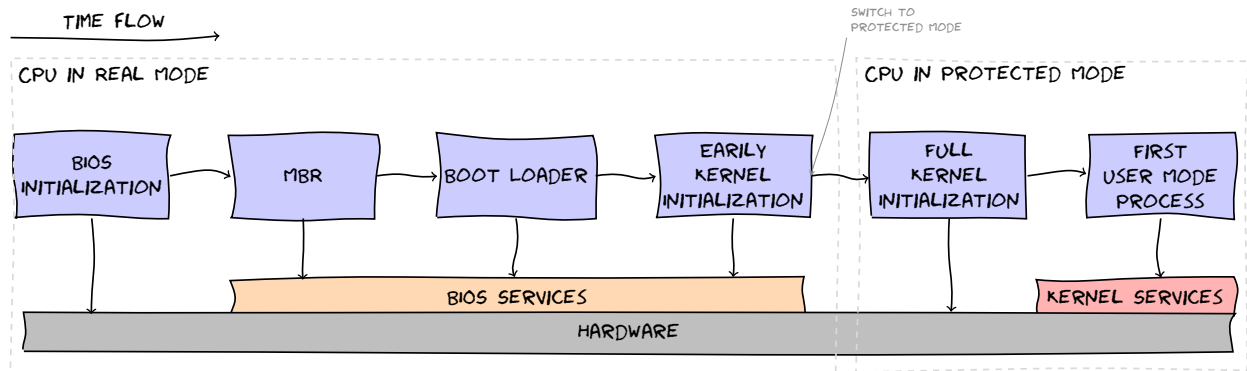


Fig. 5: Bootstrapping

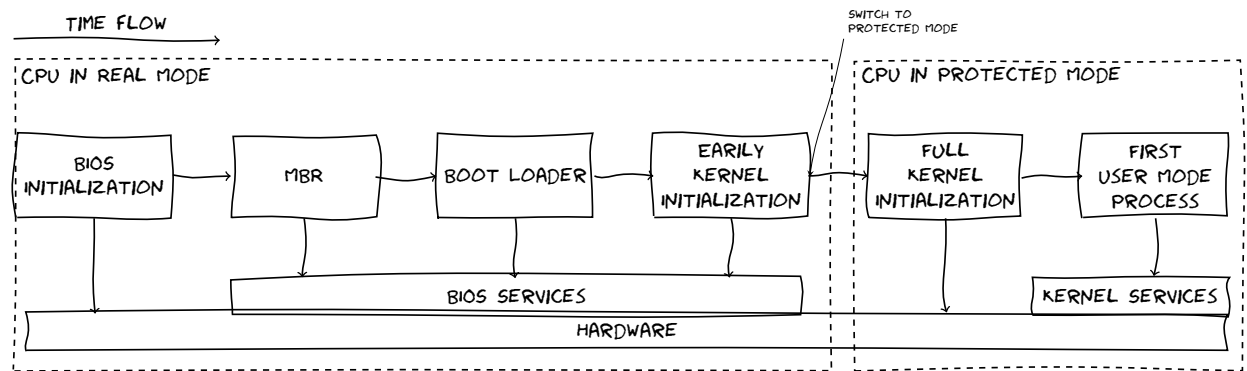


Fig. 6: Bootstrapping (bw version)

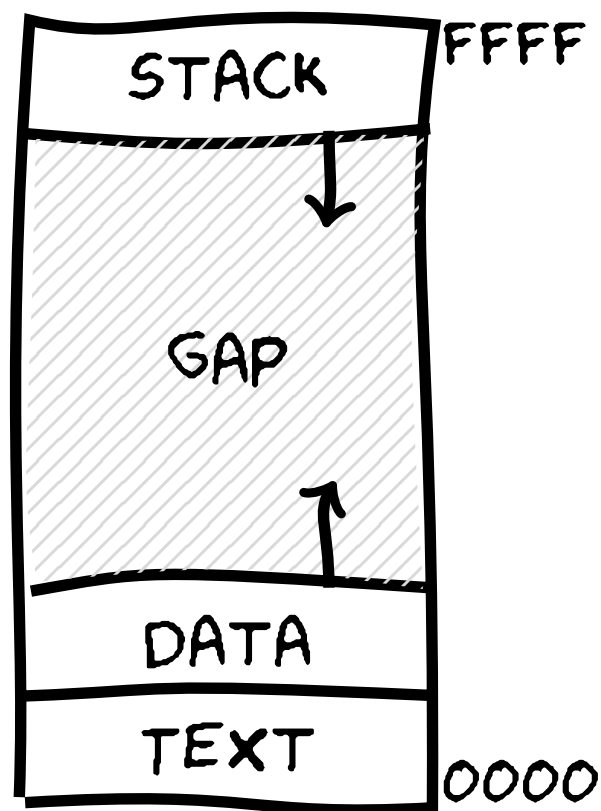
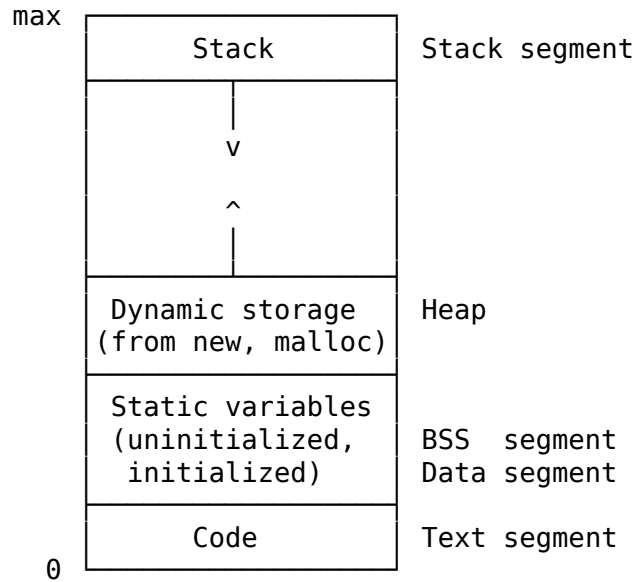


Fig. 7: Process' virtual address space



THE SIZE OF A PROCESS
(TEXT + DATA + BSS) IS
ESTABLISHED AT COMPILE TIME

Fig. 8: UNIX view of a process

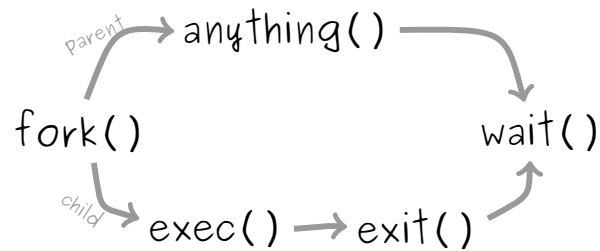


Fig. 9: Process creation

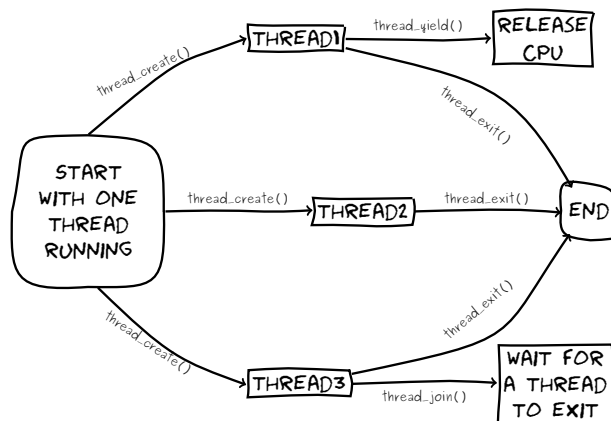


Fig. 10: Thread operations

```

typedef int semaphore;
semaphore resource_1;
semaphore resource_2;

void process_A(void) {
    down(&resource_1);
    down(&resource_2);
    use_both_resources( );
    up(&resource_2);
    up(&resource_1);
}

void process_B(void) {
    down(&resource_1);
    down(&resource_2);
    use_both_resources( );
    up(&resource_2);
    up(&resource_1);
}
    
```

(a)

```

semaphore resource_1;
semaphore resource_2;

void process_A(void) {
    down(&resource_1);
    down(&resource_2);
    use_both_resources( );
    up(&resource_2);
    up(&resource_1);
}

void process_B(void) {
    down(&resource_2);
    down(&resource_1);
    use_both_resources( );
    up(&resource_1);
    up(&resource_2);
}
    
```

(b)

Fig. 11: Deadlock — Resource issues

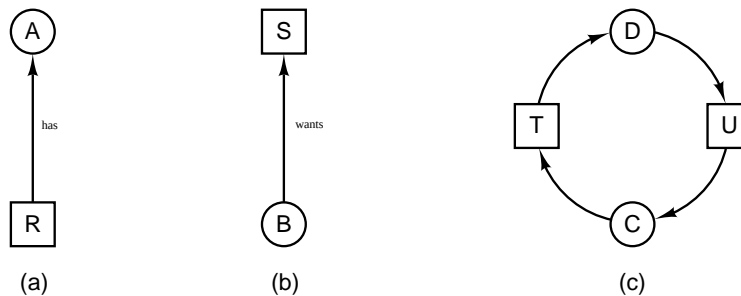


Fig. 12: Deadlock notions

Has Max		
A	0	6
B	0	5
C	0	4
D	0	7

Free: 10

(a)

Has Max		
A	1	6
B	1	5
C	2	4
D	4	7

Free: 2

(b)

Has Max		
A	1	6
B	2	5
C	2	4
D	4	7

Free: 1

(c)

Fig. 13: Deadlock — Banker algorithm

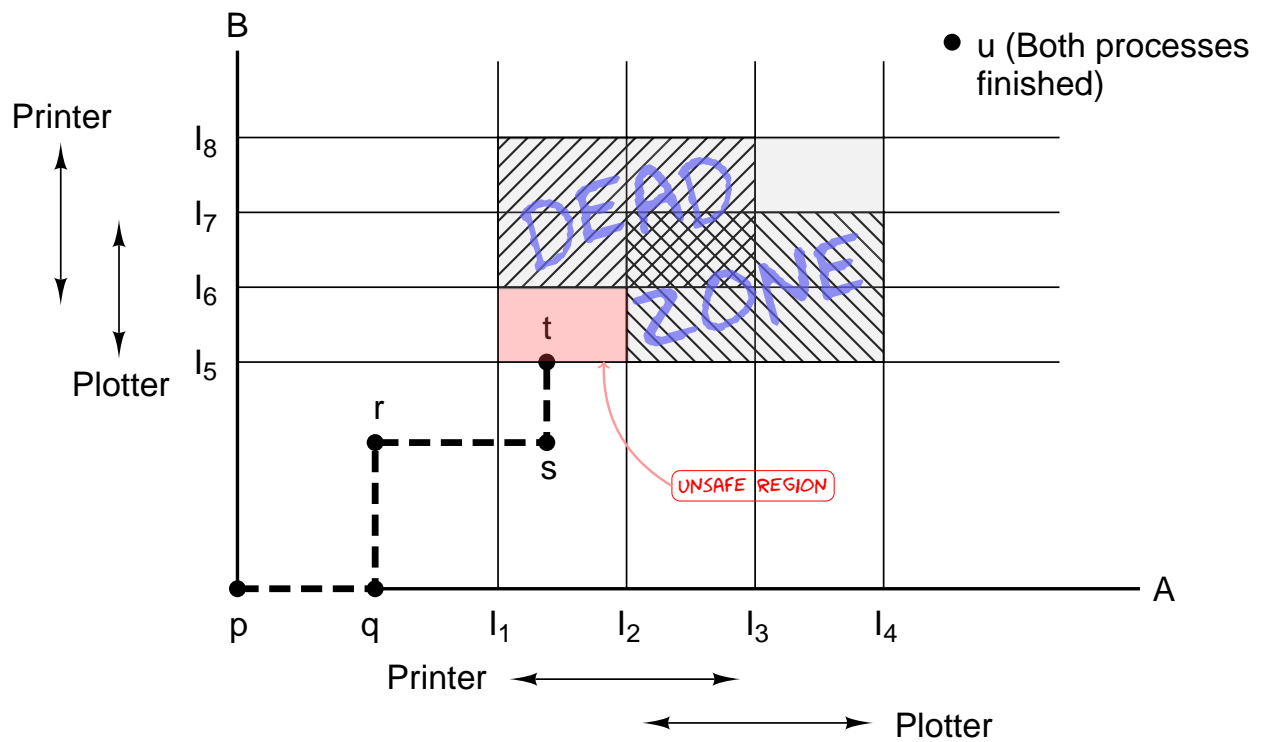


Fig. 14: Deadlock avoidance

Has Max		
A	3	9
B	2	4
C	2	7
Free: 3		
(a)		

Has Max		
A	4	9
B	2	4
C	2	7
Free: 2		
(b)		

Has Max		
A	4	9
B	4	4
C	2	7
Free: 0		
(c)		

Has Max		
A	4	9
B	—	—
C	2	7
Free: 4		
(d)		

Fig. 15: Deadlock avoidance

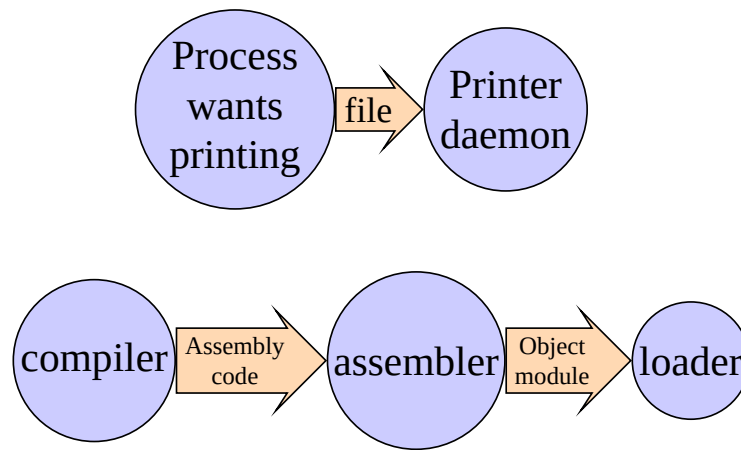


Fig. 16: Producers and consumers

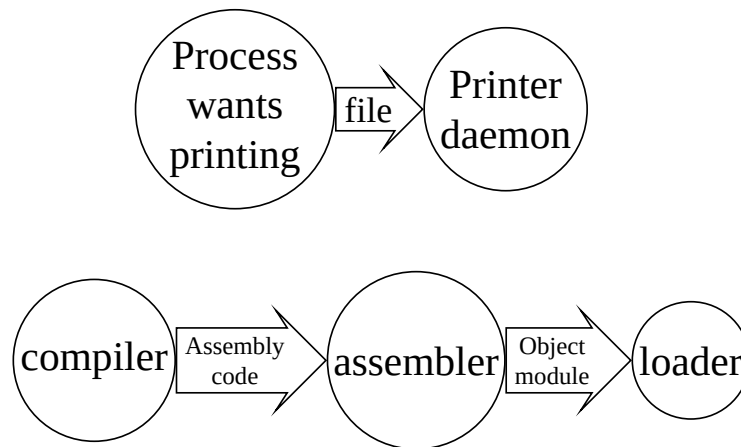


Fig. 17: Producers and consumers (bw version)

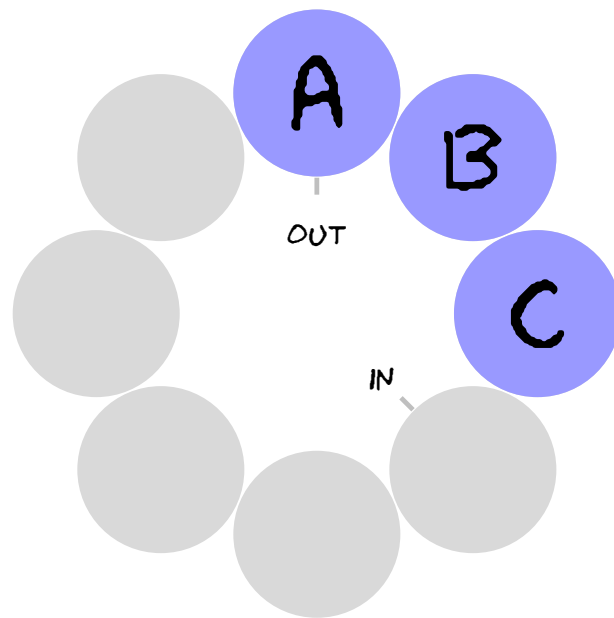


Fig. 18: A circular array

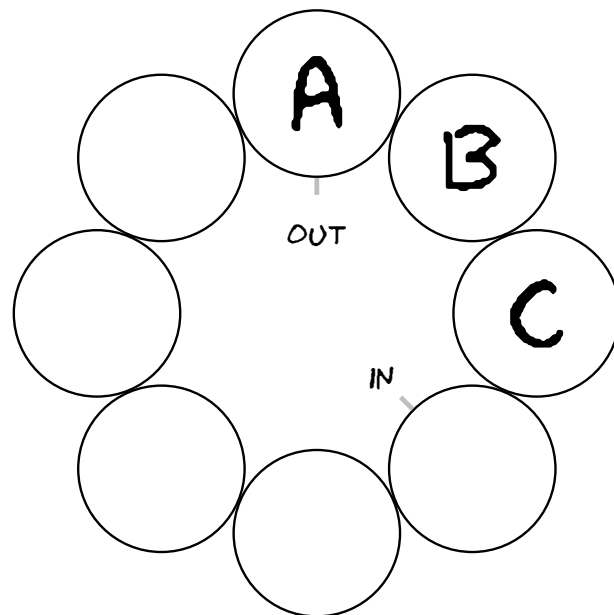


Fig. 19: A circular array (bw version)

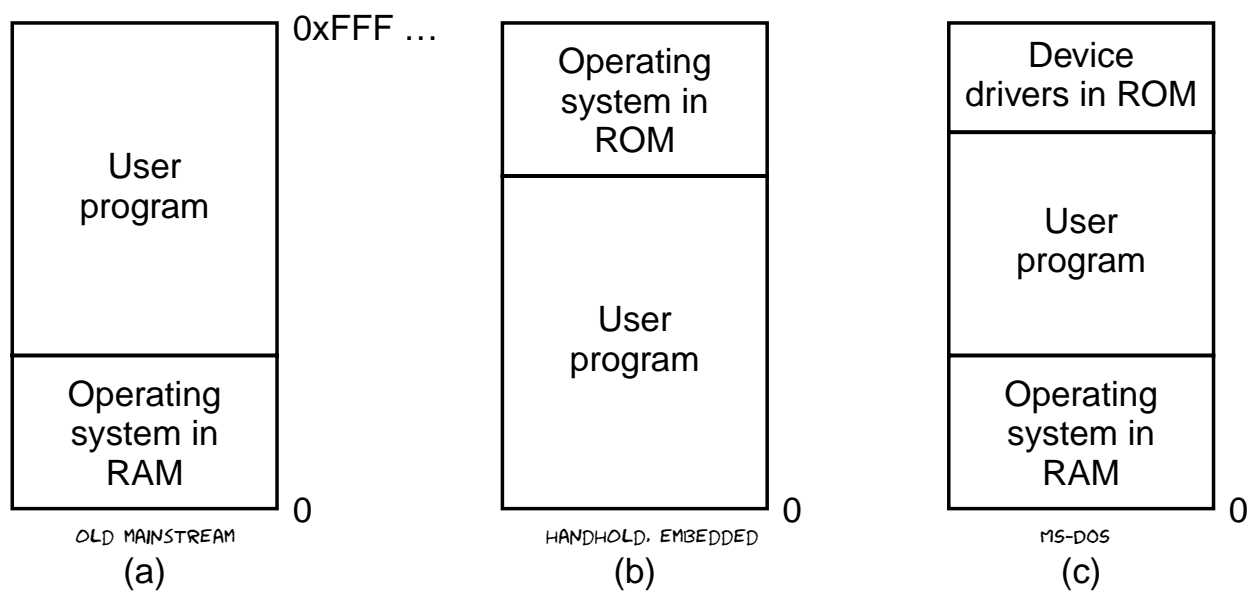


Fig. 20: Real mode memory layouts

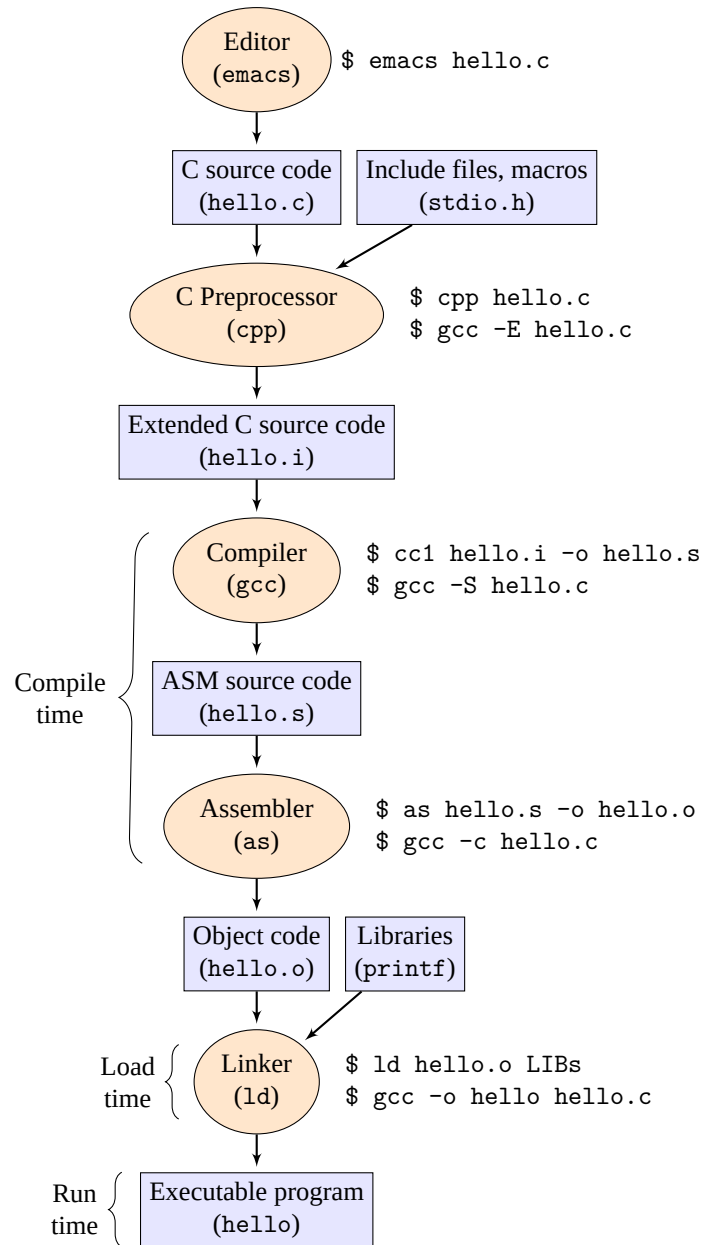


Fig. 21: Tool chain

EXPOSING PHYSICAL MEMORY TO PROCESSES IS NOT A GOOD IDEA

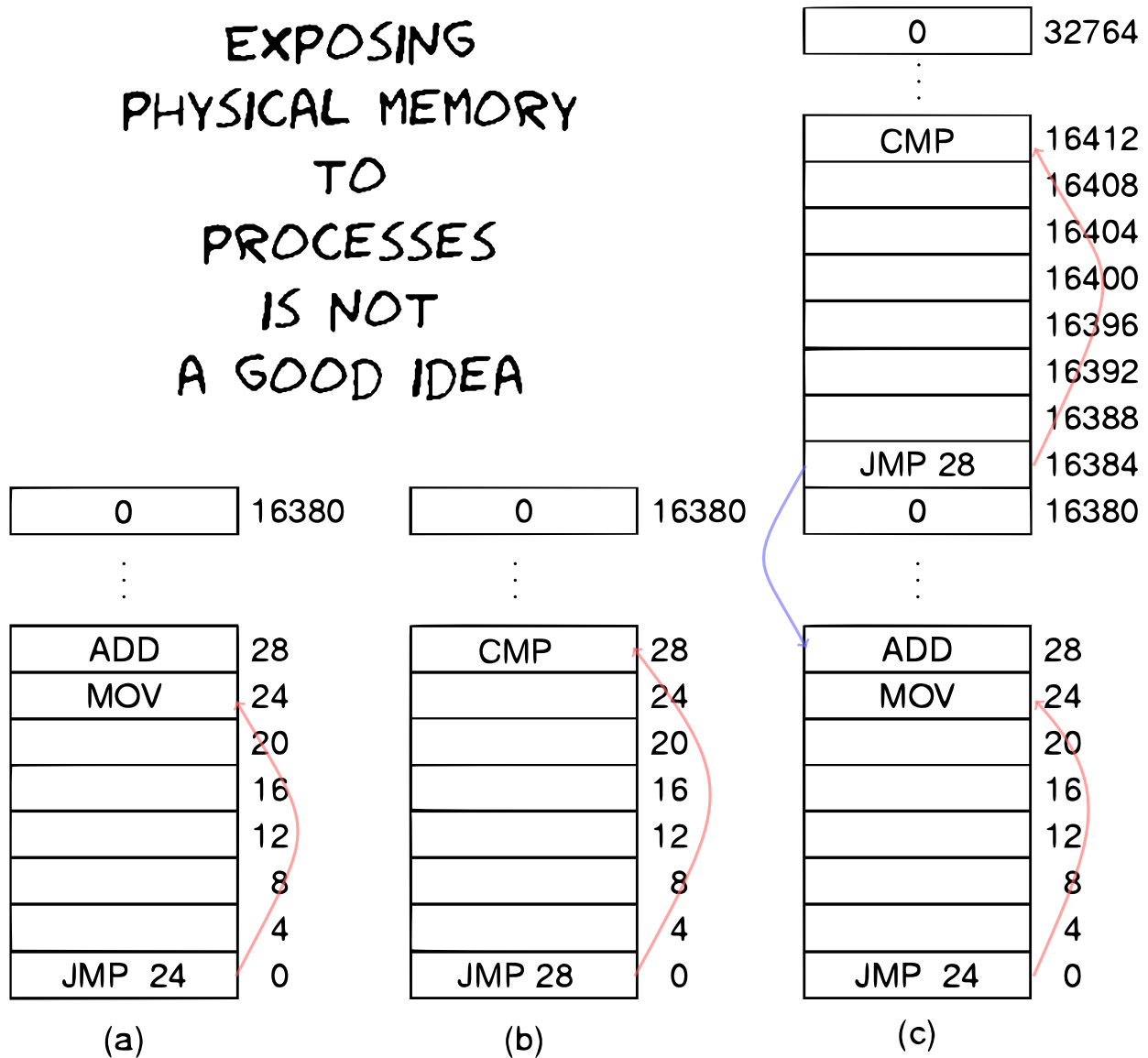


Fig. 22: Relocation

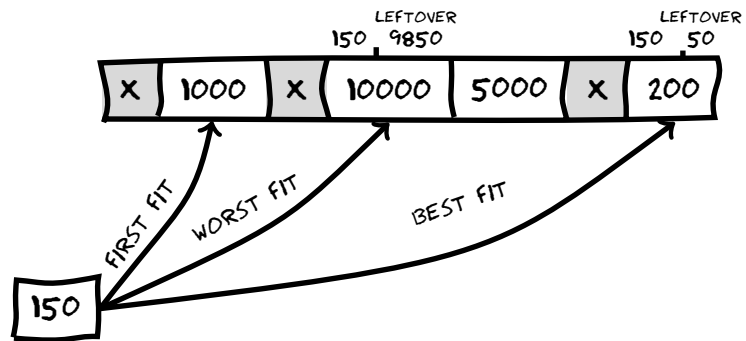


Fig. 23: First fit, best fit, worst fit

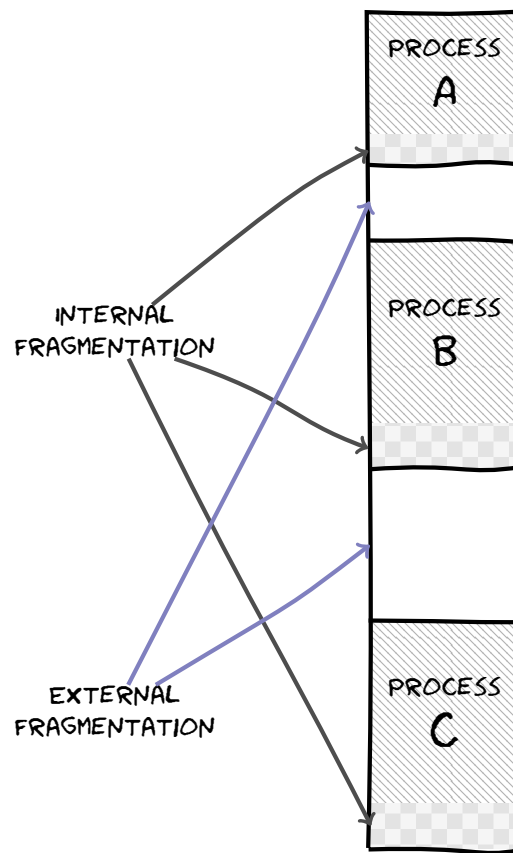


Fig. 24: Memory fragmentation

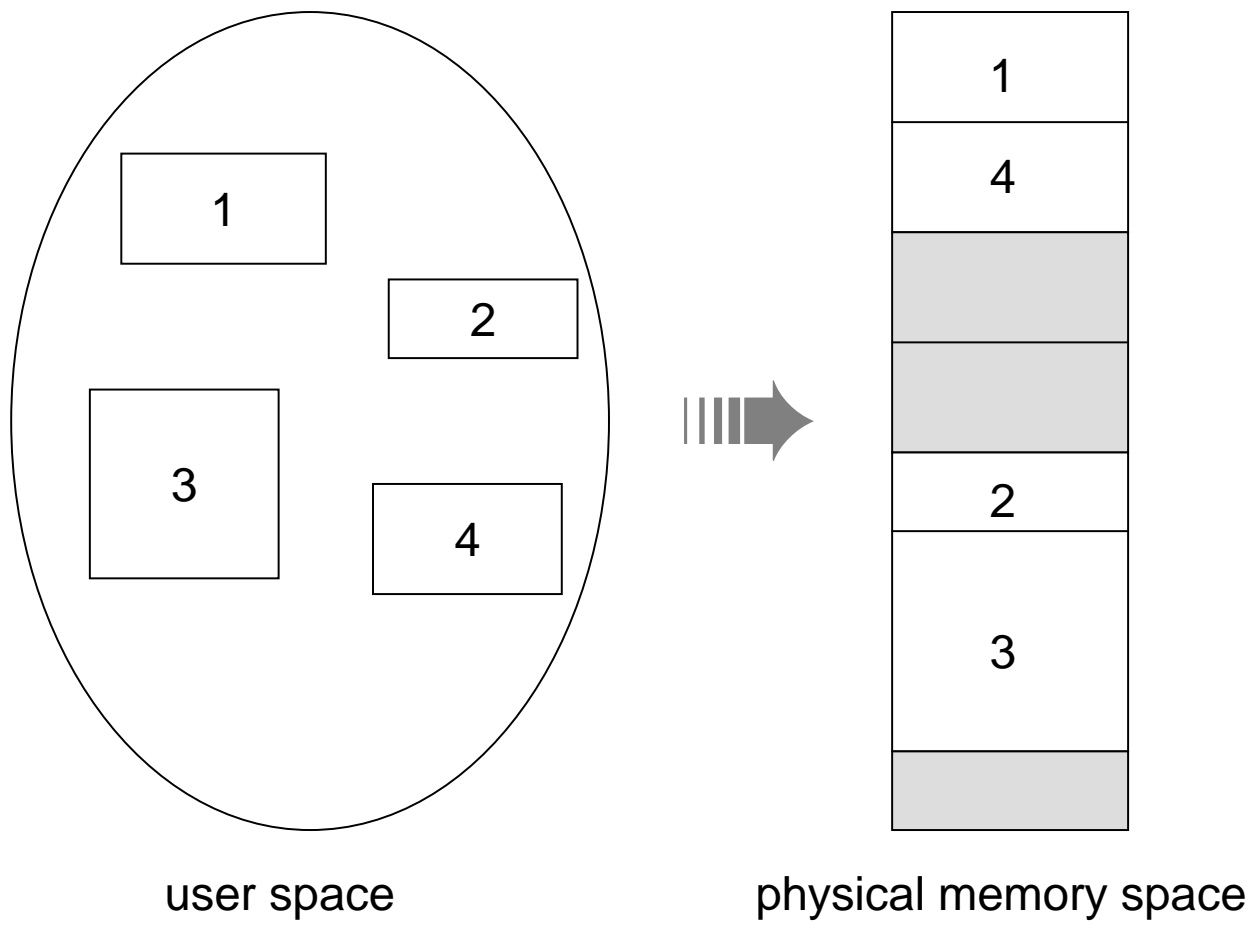


Fig. 26: Memory segmentation

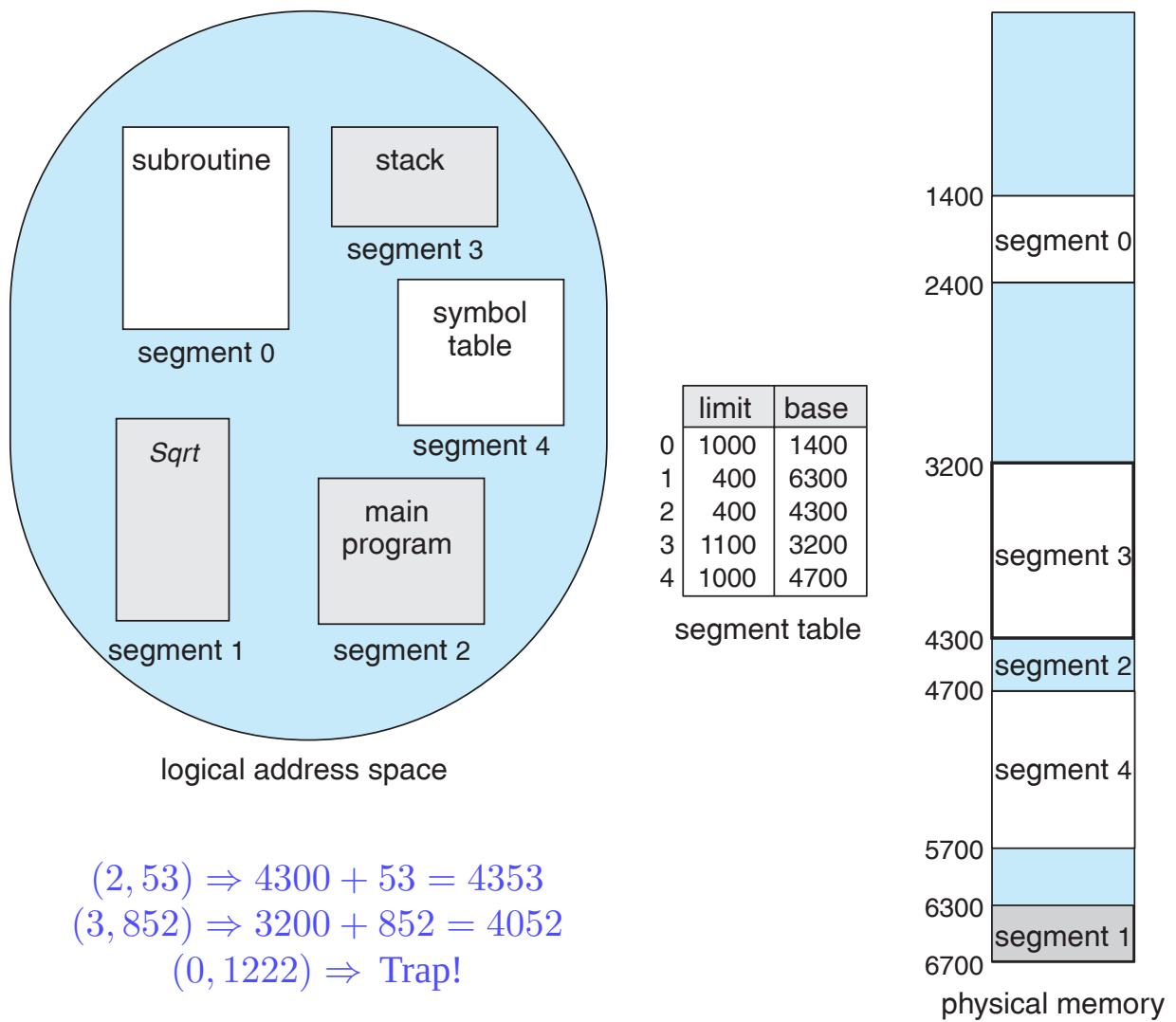


Fig. 27: Memory segmentation — Address translation

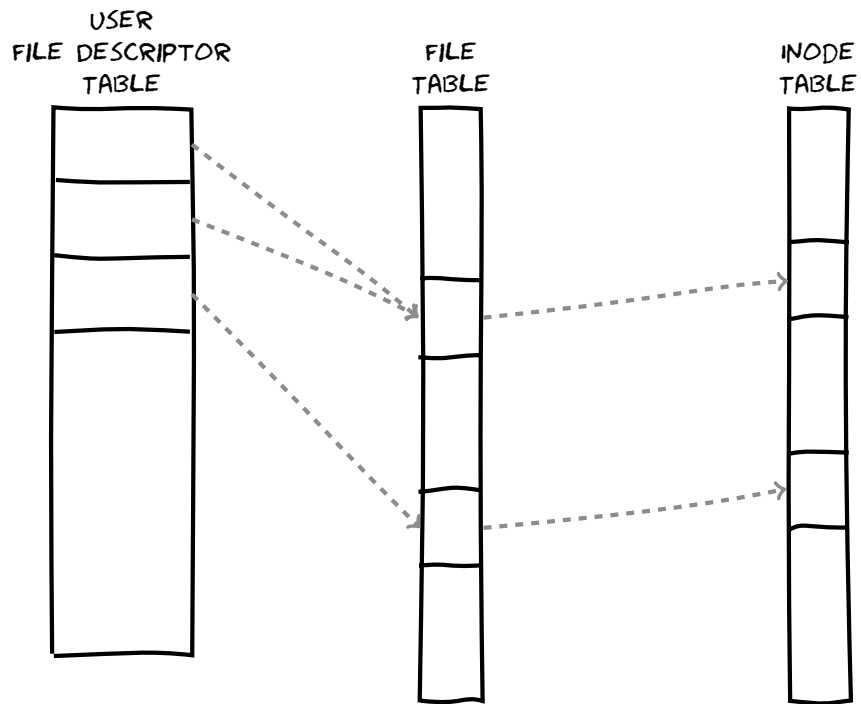


Fig. 28: File system tables

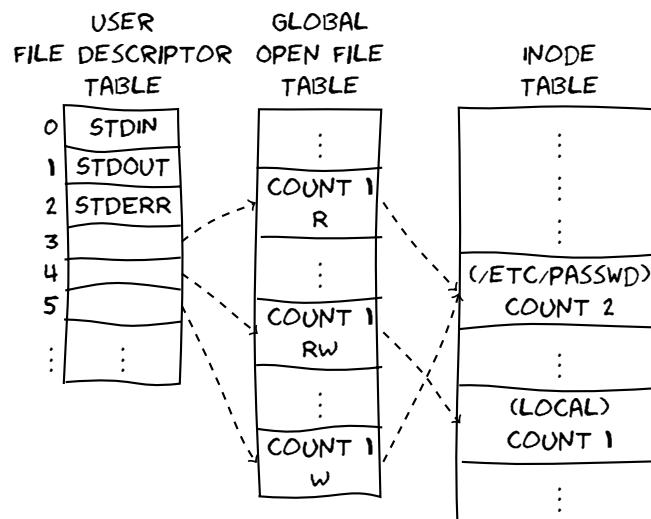


Fig. 29: File tables

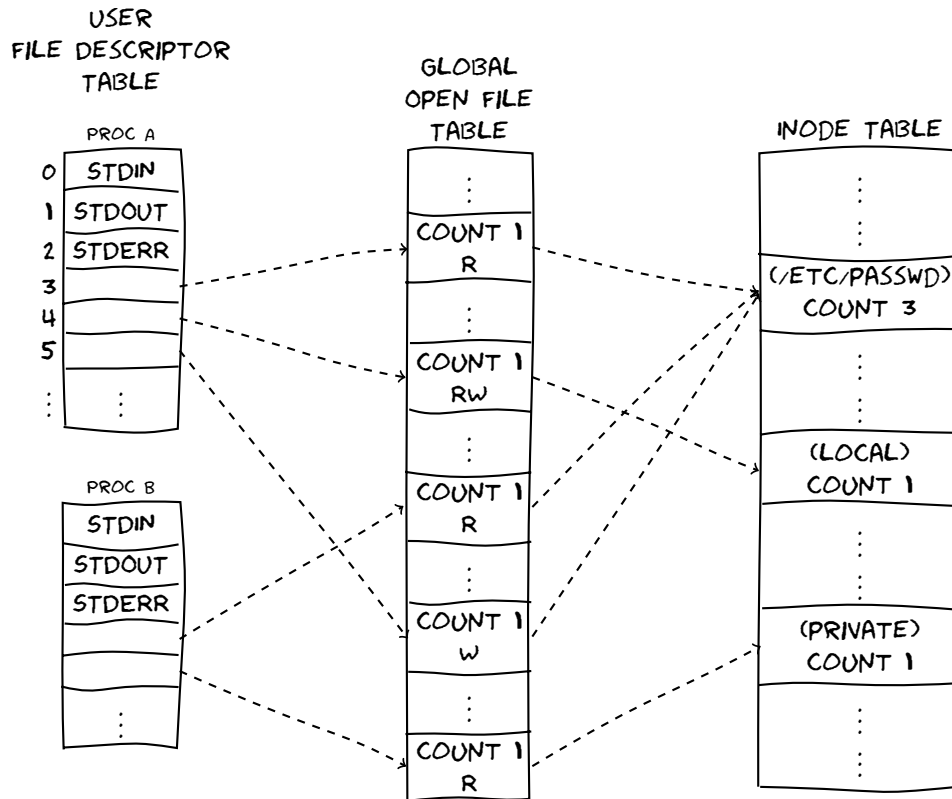


Fig. 30: File tables

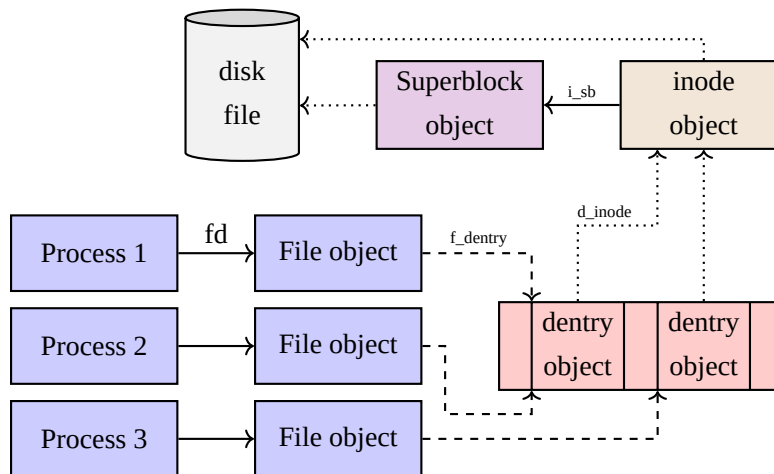


Fig. 31: VFS objects

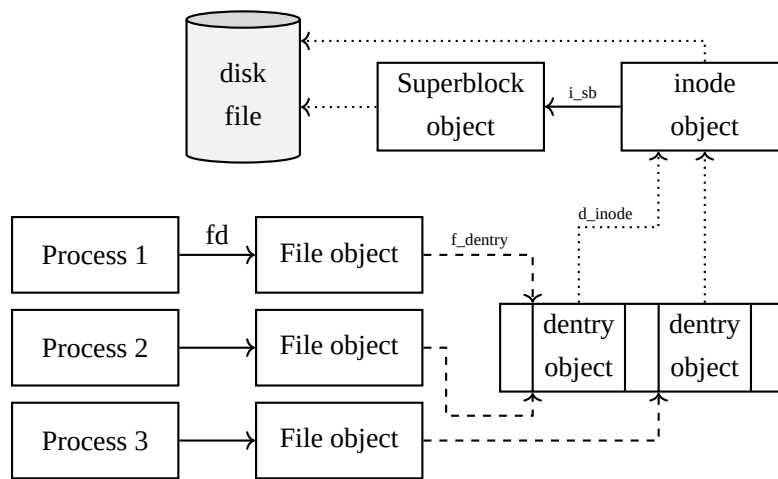


Fig. 32: VFS objects

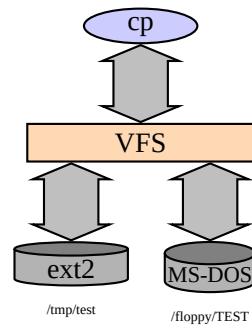


Fig. 33: VFS file copy

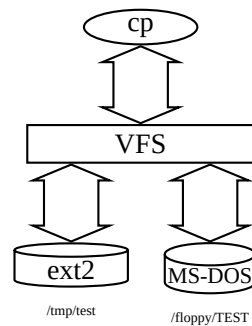


Fig. 34: VFS file copy (bw version)

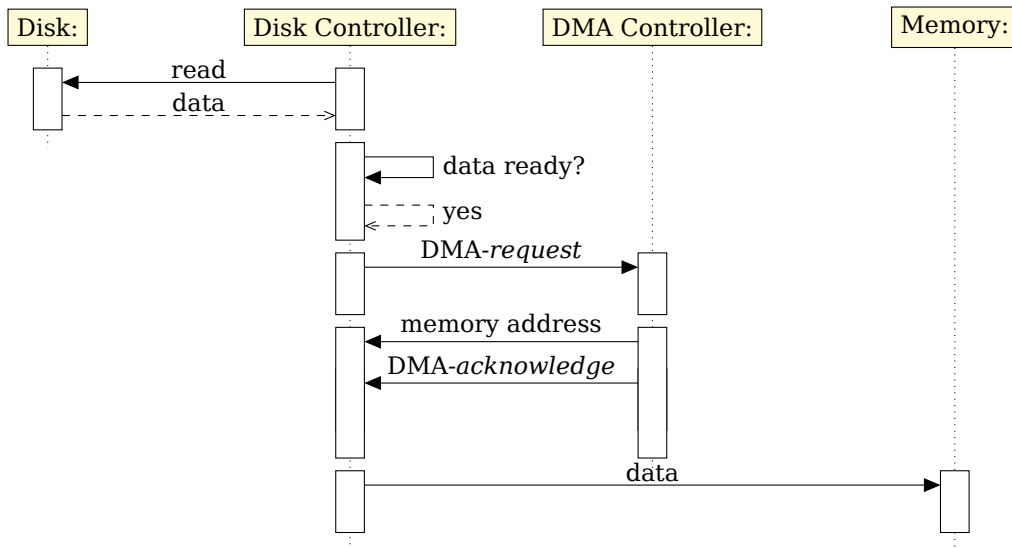


Fig. 35: DMA handshaking

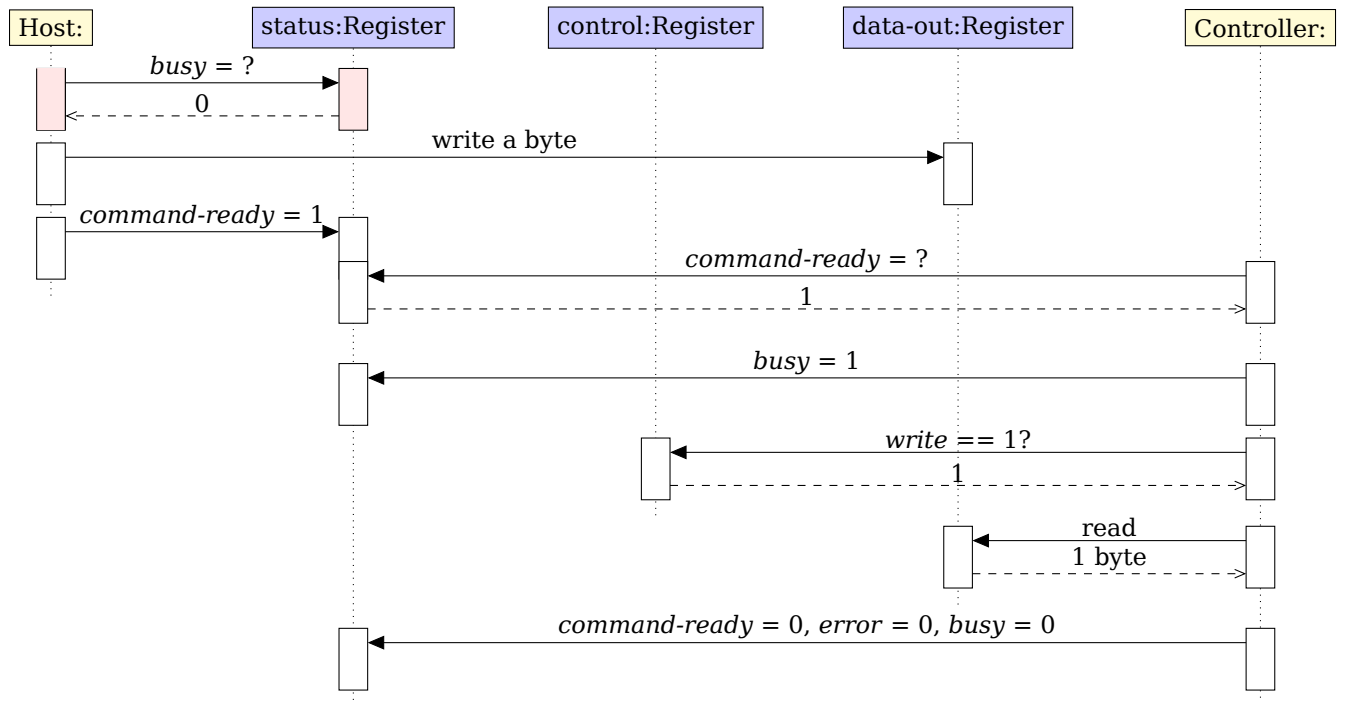


Fig. 36: Handshaking