Homework #7 M1522.000800 System Programming

Sample Solution

Question 1

Memory Mapping

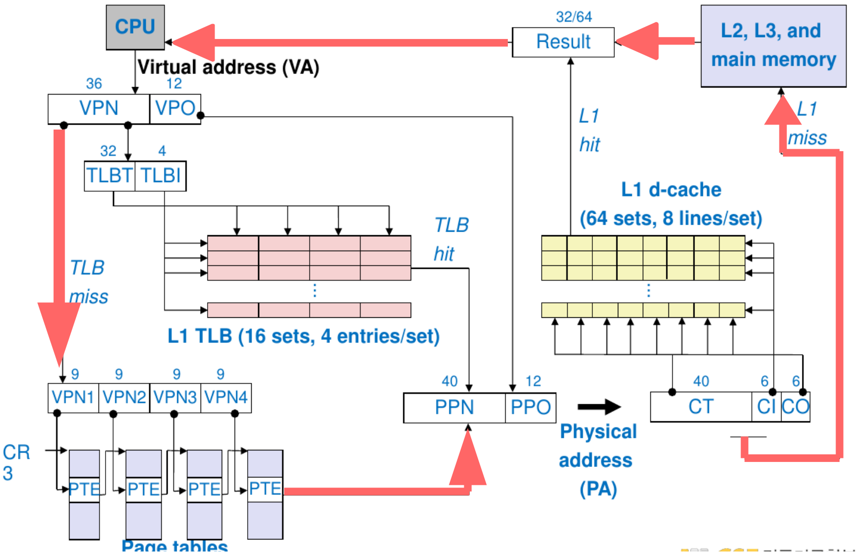
Given an input file `SP.txt` that consists of the string “`I hate System Programming!\n`”, write a C program that uses `mmap` to change the contents of `SP.txt` to “`I love System Programming!\n`”. Use fstat() to get the file size.

```
#include <fcntl.h>
#include <unistd.h>
#include <sys/mman.h>
#include <sys/stat.h>
#include <sys/types.h>

int main(void) {
    int fd;
    struct stat stat;
    char *mm;

    fd = open(“SP.txt”, O_RDWR);

    fstat(fd, &stat);

    mm = mmap(NULL, stat.st_size, PROT_READ | PROT_WRITE, MAP_SHARED, fd, 0);

    mm[2] = 'l';
    mm[3] = 'o';
    mm[4] = 'v';
    mm[5] = 'e';

    munmap(mm, stat.st_size);
    close(fd);

    return 0;
}
```

Question 2

Address Translation

The following picture is an **end-to-end address translation** mechanism on Intel i7 core.



a) How much large is the **L1 d-cache line size** in byte?

=> The Intel i7 L1 d-cache total size is 32KB(= 2^{15}). It has 64(= 2^6)sets, so a set has 2^9 Bytes. Also, a set has 8(= 2^3)lines, therefore each line has 64(=2^6)Btyes.

b) How much large is the **page size** in byte?

=> Page size is based on PPO (Physical Page Offset). It has 12 bits, so it has 4KB(=2^{12}).

c) If the **page entry size** for each page tables is same as **8B**, how much large is the **page table size** for each page tables in byte?

=> Each page table has 2^9 entries and its size is 8(=2^3)B, therefore the page table size is 4KB(=2^{12}).

d) If a program is to access a data, draw the address translation flow of the **worst case** in accessing time on this picture.

=> Drawn in the plot.

Question 3

Dynamic Memory Allocation

Determine the block sizes and header values that would result from the following sequence of `malloc` requests. Assumptions: (1) The allocator maintains **double-word** alignment, and uses an **implicit free list** with the block format from **the following plot**. (2) Block sizes are rounded up to the nearest multiple of **eight** bytes.

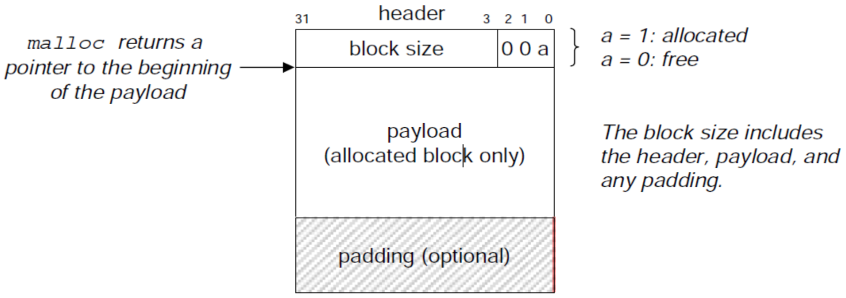

malloc returns a pointer to the beginning of the payload
31
header
3 2 1 0
block size
0 0 a
a = 1: allocated
a = 0: free
payload (allocated block only)
The block size includes the header, payload, and any padding.
padding (optional)

Figure 10.37: **Format of a simple heap block.**

Request	Block size (decimal bytes)	Block header (hex)
malloc(3)	8	0x9
malloc(11)	16	0x11
malloc(20)	24	0x19
malloc(21)	32	0x21

=> The allocator allocates 32 bytes for malloc(21), not 24 bytes because block size should involve payload, and header too, i.e. header always has 4bytes, and 21 + 4 = 25. Therefore, 25 is rounded up to 32. That's why malloc(21) is allocated 32 bytes for the block size.