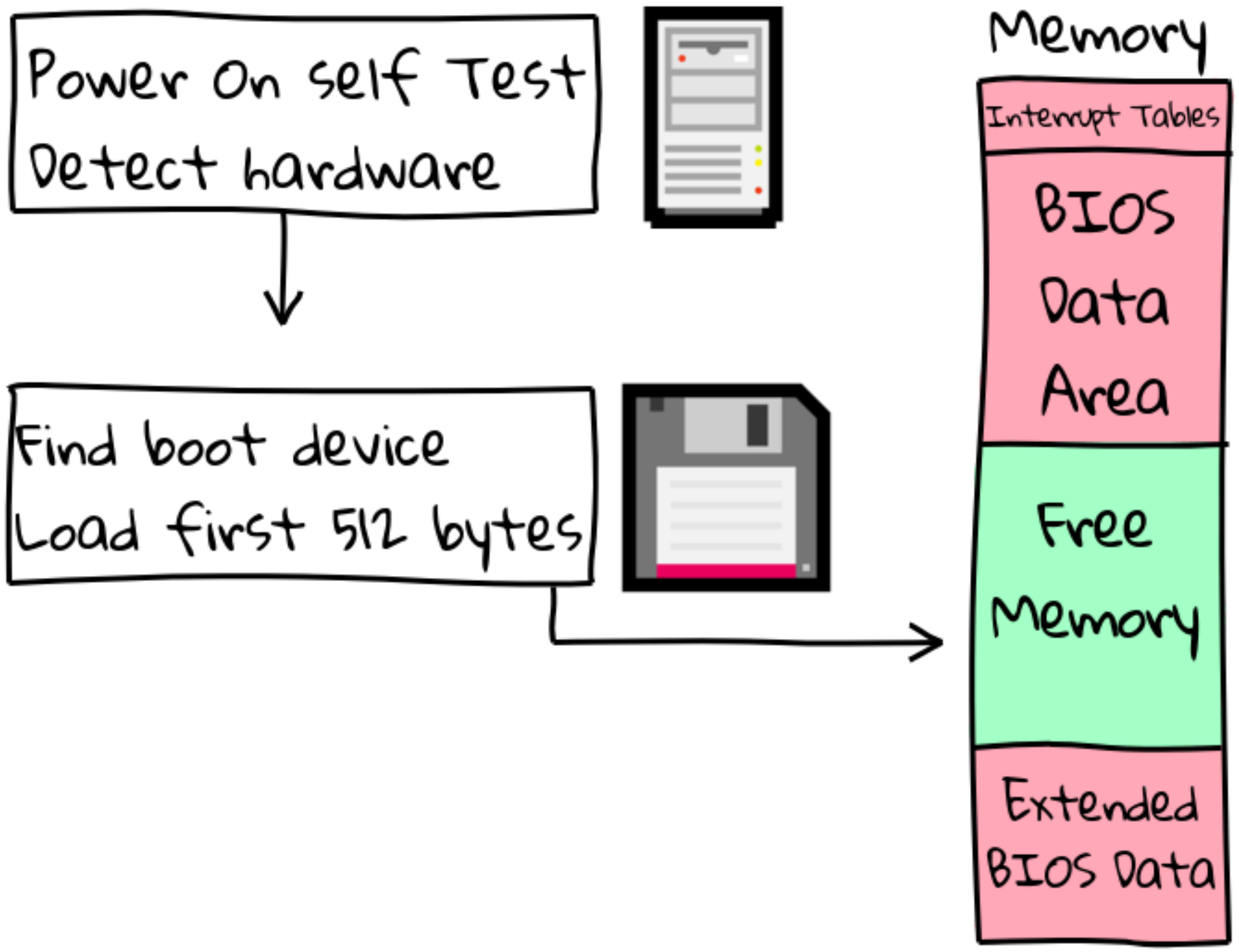
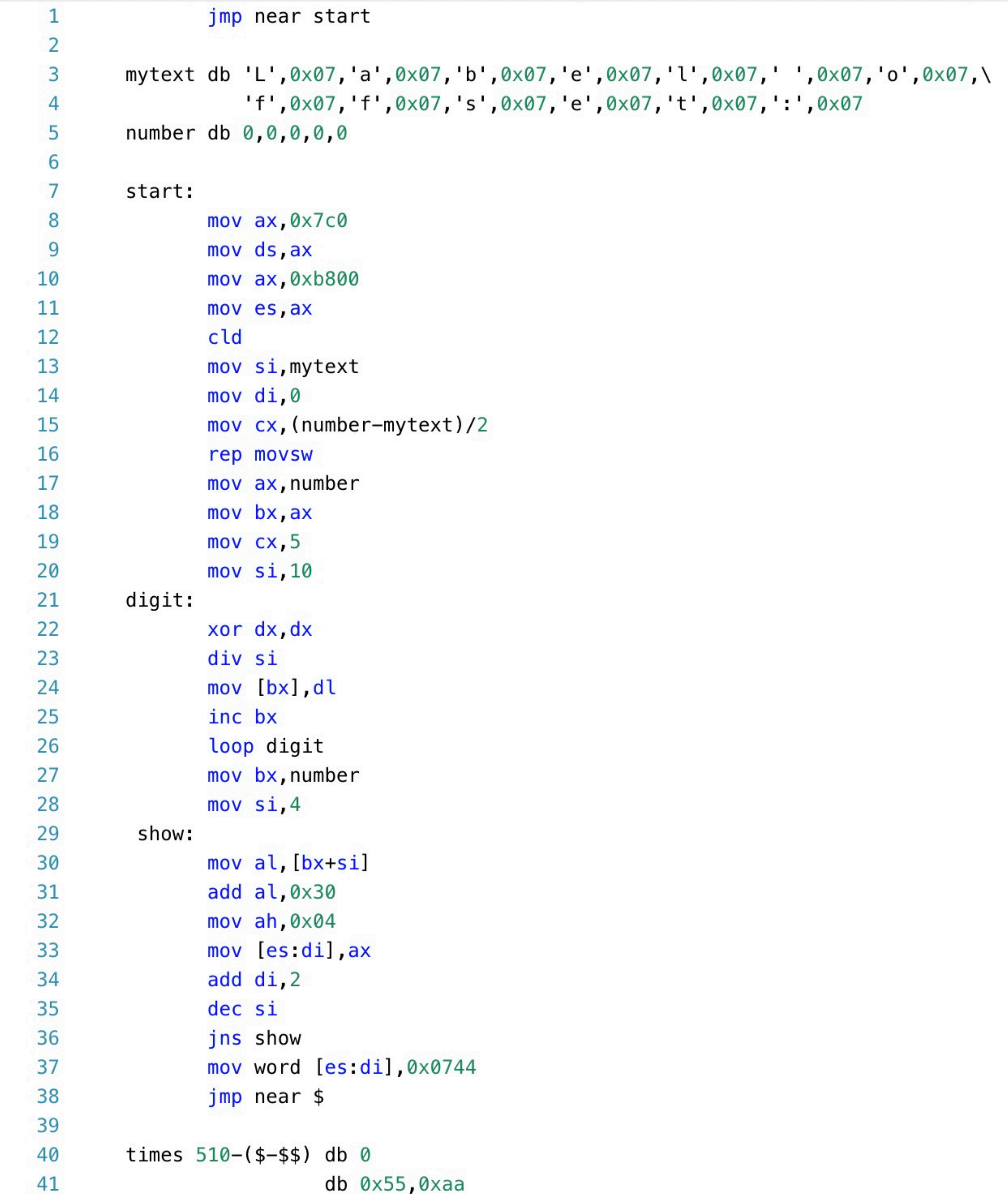
浅谈x86架构

翟增辉 2019.05.10



power-on -> bios -> mbr -> 0x7c00



```
       jmp near start

mytext db 'L',0x07,'a',0x07,'b',0x07,'e',0x07,'l',0x07,' ',0x07,'o',0x07,\
          'f',0x07,'f',0x07,'s',0x07,'e',0x07,'t',0x07,':',0x07
number db 0,0,0,0,0

start:
       mov ax,0x7c0
       mov ds,ax
       mov ax,0xb800
       mov es,ax
       cld
       mov si,mytext
       mov di,0
       mov cx,(number-mytext)/2
       rep movsw
       mov ax,number
       mov bx,ax
       mov cx,5
       mov si,10
digit:
       xor dx,dx
       div si
       mov [bx],dl
       inc bx
       loop digit
       mov bx,number
       mov si,4
 show:
       mov al,[bx+si]
       add al,0x30
       mov ah,0x04
       mov [es:di],ax
       add di,2
       dec si
       jns show
       mov word [es:di],0x0744
       jmp near $

times 510-($-$$) db 0
                 db 0x55,0xaa
```

qemu simulation

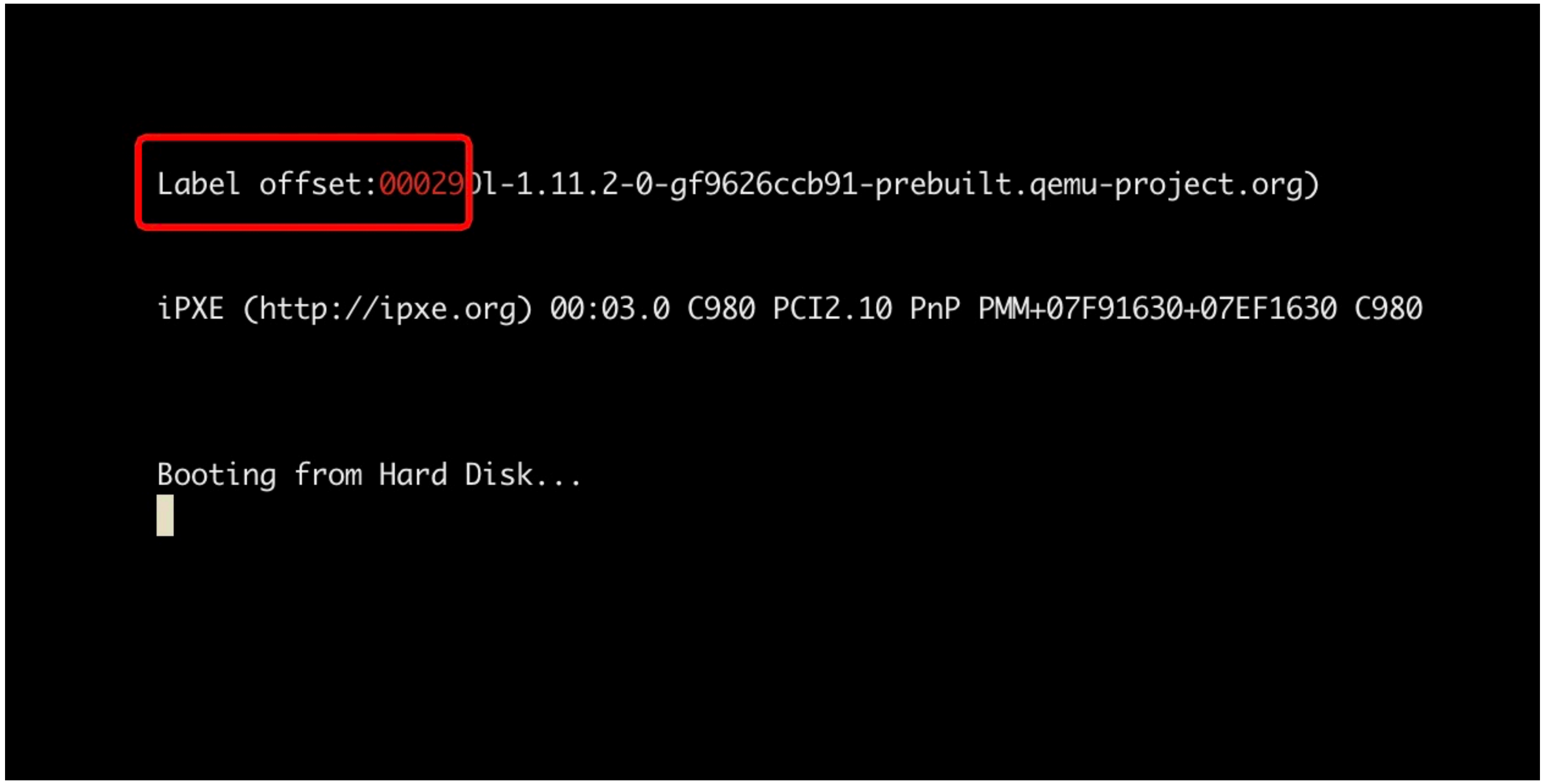
- nasm -f bin mbr.asm -o mbr.bin

- qemu-img create -f vpc -o subformat=fixed singchia.vhd 64M

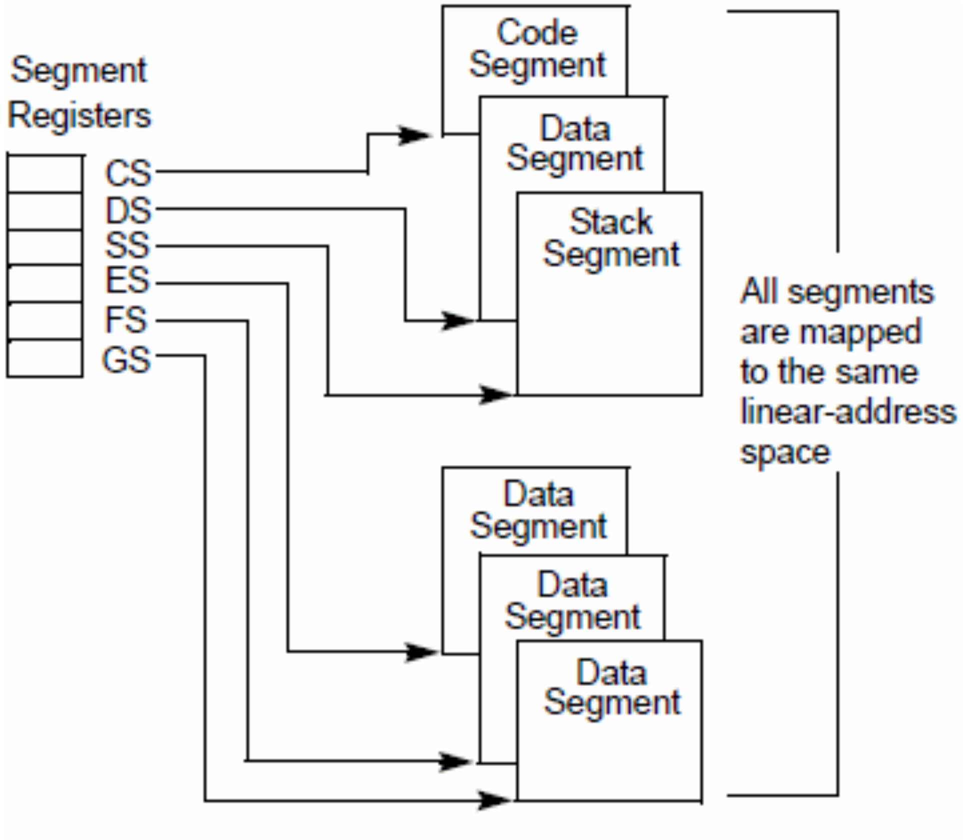
- dd bs=512 count=1 conv=notrunc if=mbr.bin of=singchia.vhd

- qemu-img convert -f vpc -O raw singchia.vhd singchia.img

- qemu-system-x86_64 -drive file=singchia.img,index=0,media=disk,format=raw -curses



```
Label offset:000290l-1.11.2-0-gf9626ccb91-prebuilt.qemu-project.org)

iPXE (http://ipxe.org) 00:03.0 C980 PCI2.10 PnP PMM+07F91630+07EF1630 C980


Booting from Hard Disk...
```

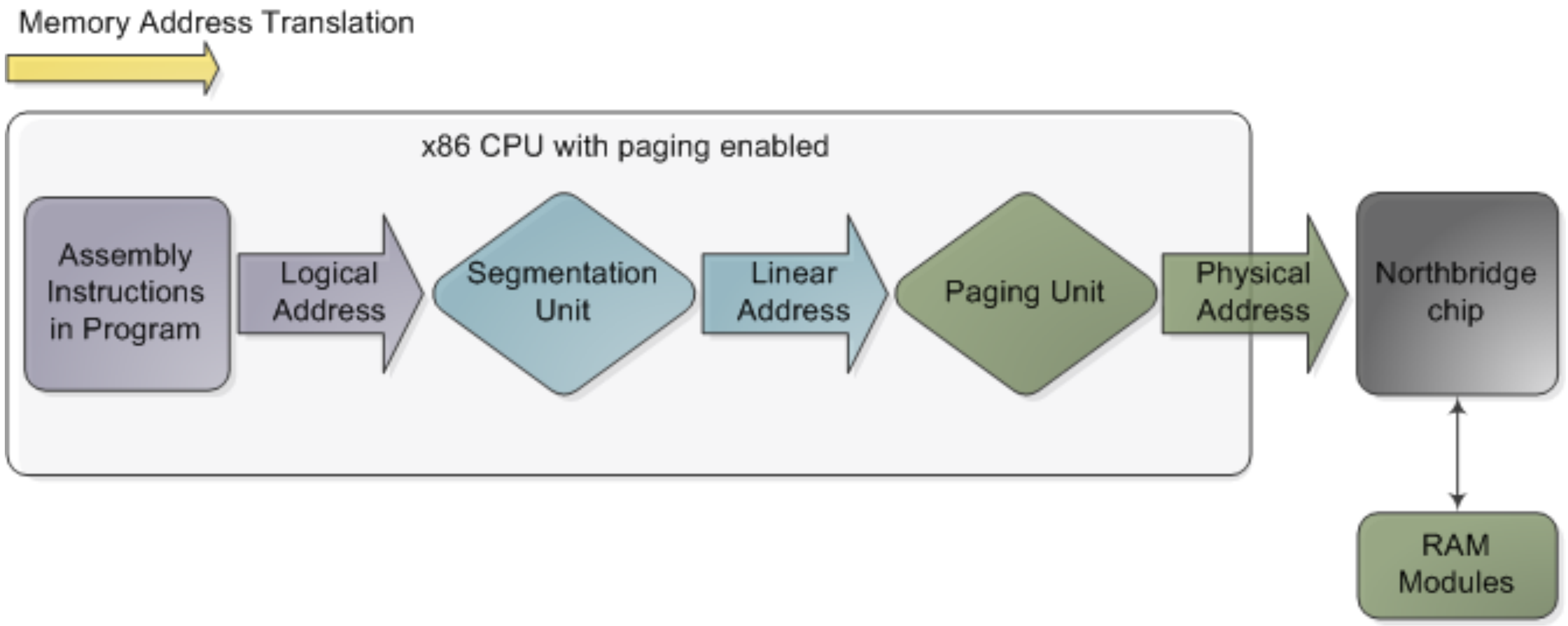


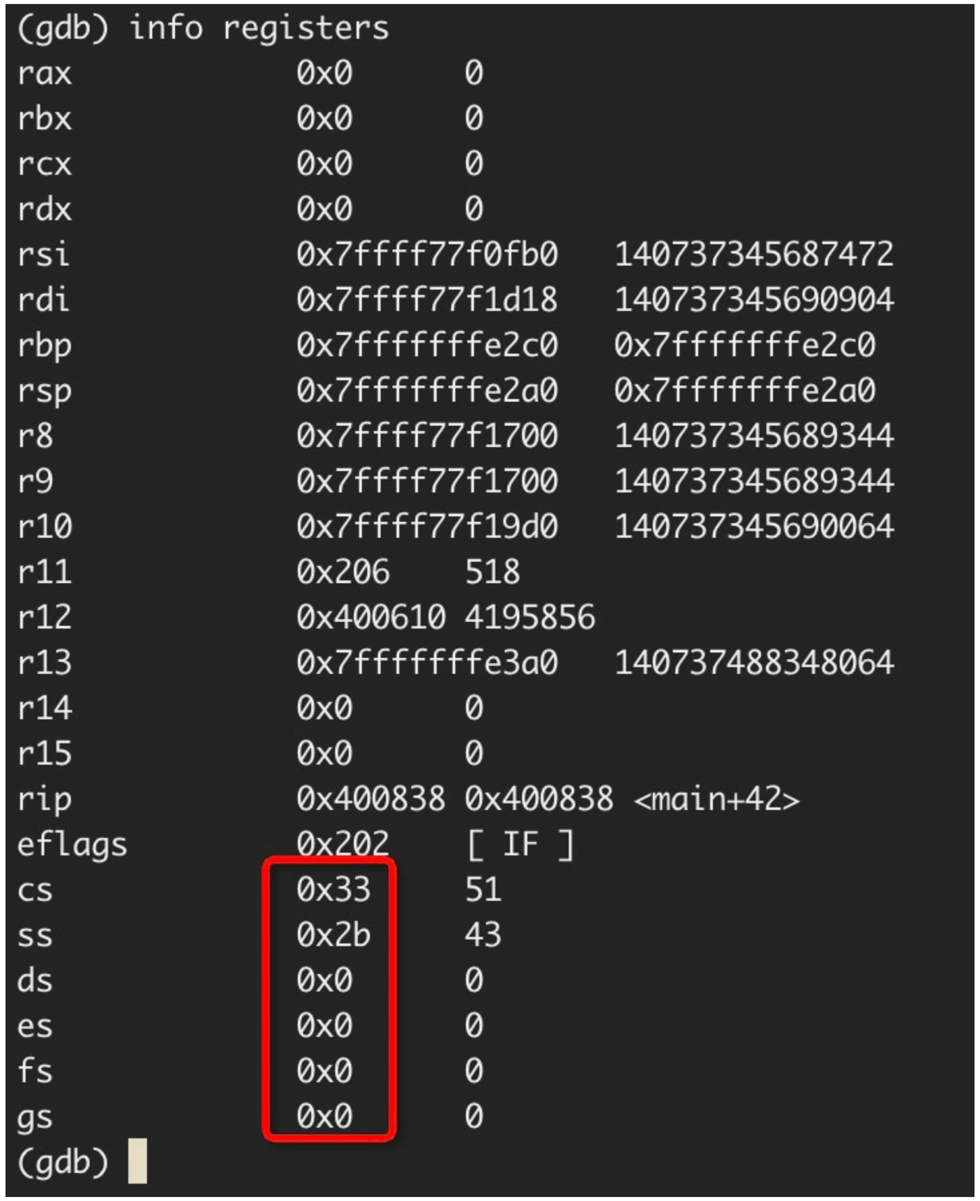
Use of Segment Registers in Segmented Memory Model

x86开机启动为16位实地址模式
段、通用寄存器16位，段大小为64k
地址总线20位，可寻址大小为1M

保护模式

- 多任务 -> task state segment
- 内存隔离 -> 分段? 分页? 段页?
- 特权隔离 -> 系统调用



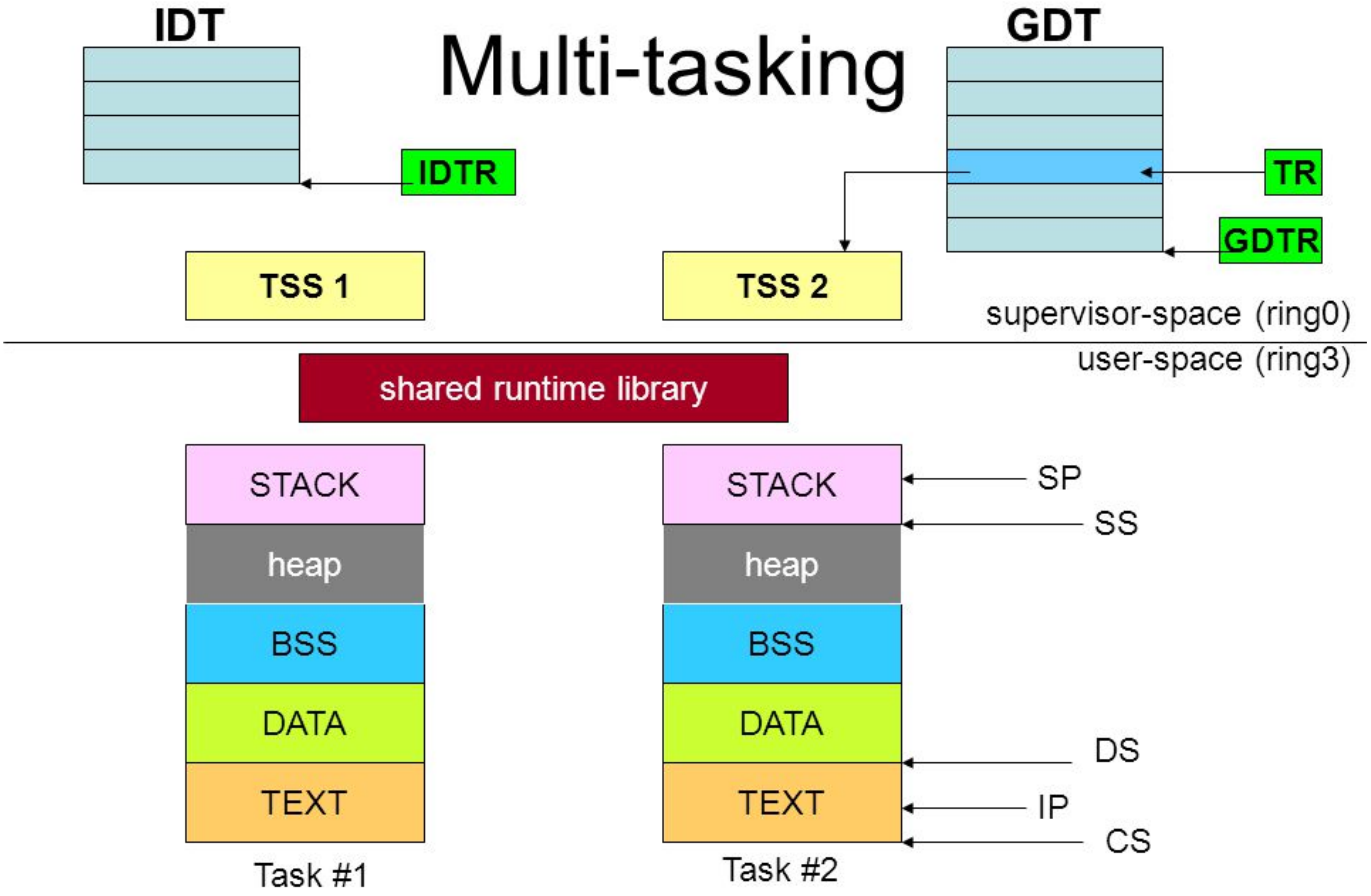


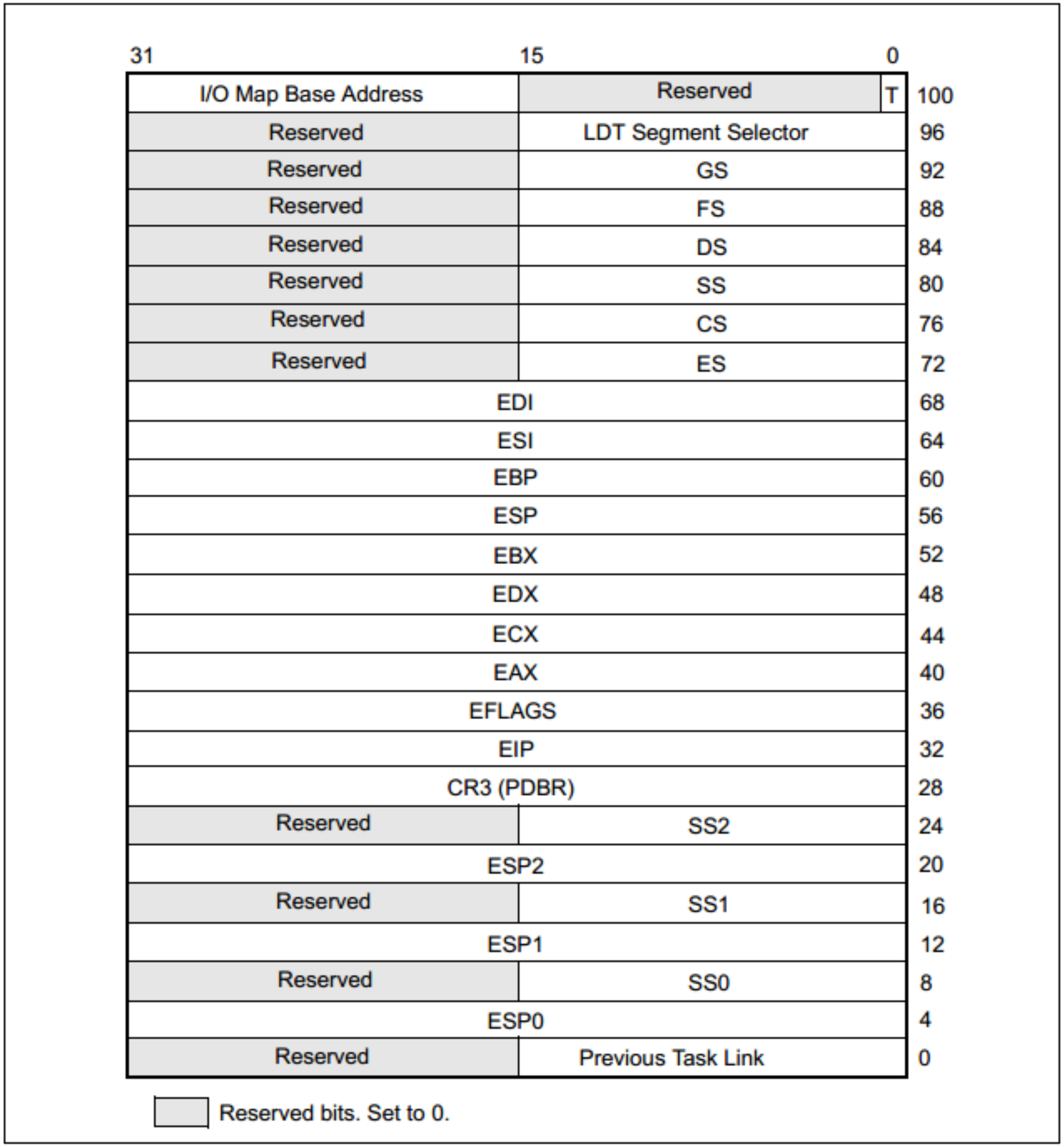
```
(gdb) info registers
rax            0x0      0
rbx            0x0      0
rcx            0x0      0
rdx            0x0      0
rsi            0x7ffff77f0fb0   140737345687472
rdi            0x7ffff77f1d18   140737345690904
rbp            0x7fffffffe2c0   0x7fffffffe2c0
rsp            0x7fffffffe2a0   0x7fffffffe2a0
r8             0x7ffff77f1700   140737345689344
r9             0x7ffff77f1700   140737345689344
r10            0x7ffff77f19d0   140737345690064
r11            0x206    518
r12            0x400610 4195856
r13            0x7fffffffe3a0   140737488348064
r14            0x0      0
r15            0x0      0
rip            0x400838 0x400838 <main+42>
eflags         0x202    [ IF ]
cs             0x33     51
ss             0x2b     43
ds             0x0      0
es             0x0      0
fs             0x0      0
gs             0x0      0
(gdb) 
```

The 0x33 Segment Selector

https://www.malwaretech.com/2014/02/the-0x33-segment-selector-heavens-gate.html

Multi-tasking





31	15	0	
I/O Map Base Address	Reserved	T	100
Reserved	LDT Segment Selector		96
Reserved	GS		92
Reserved	FS		88
Reserved	DS		84
Reserved	SS		80
Reserved	CS		76
Reserved	ES		72
EDI			68
ESI			64
EBP			60
ESP			56
EBX			52
EDX			48
ECX			44
EAX			40
EFLAGS			36
EIP			32
CR3 (PDBR)			28
Reserved	SS2		24
ESP2			20
Reserved	SS1		16
ESP1			12
Reserved	SS0		8
ESP0			4
Reserved	Previous Task Link		0

▨ Reserved bits. Set to 0.

Figure 7-2. 32-Bit Task-State Segment (TSS)

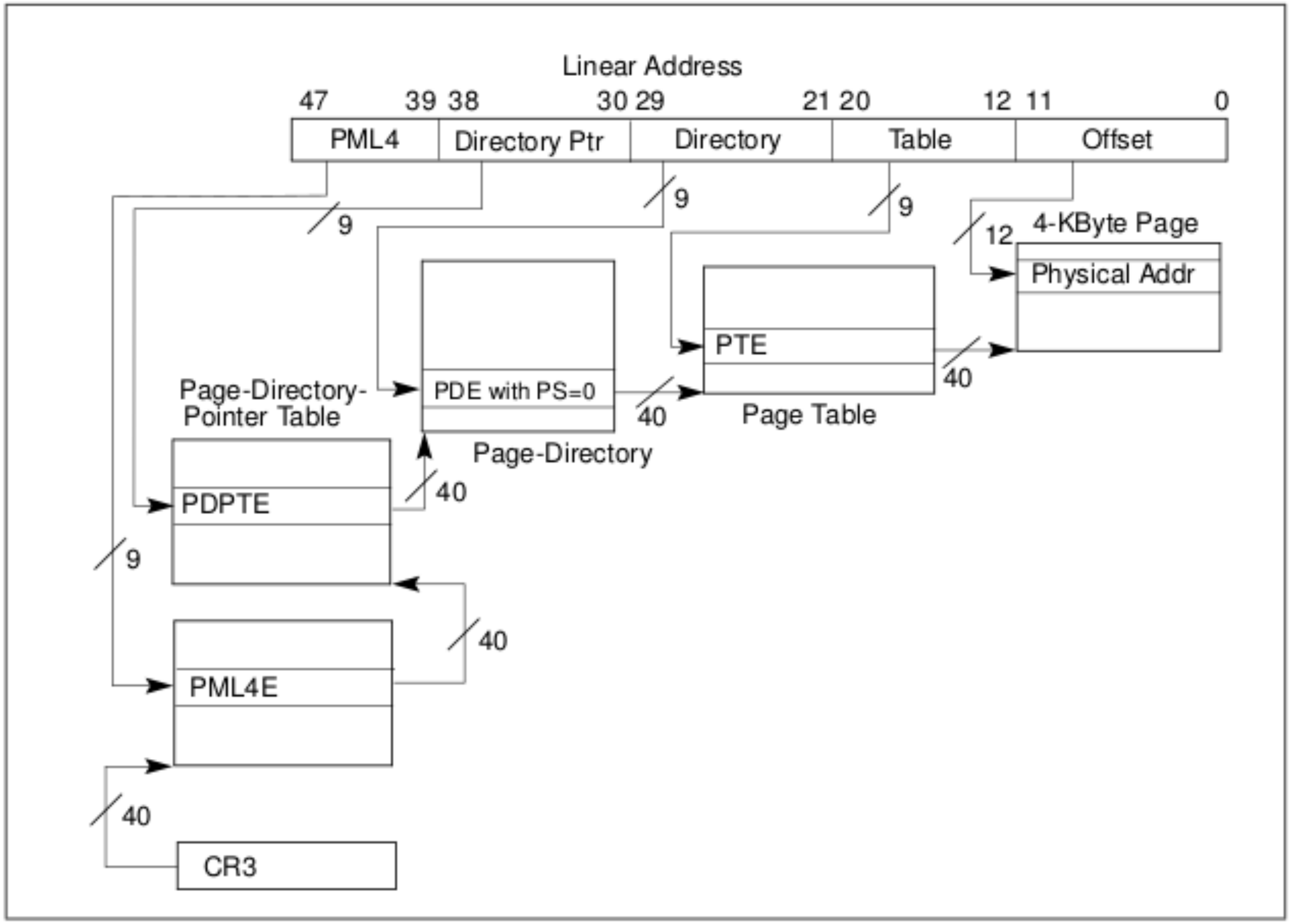
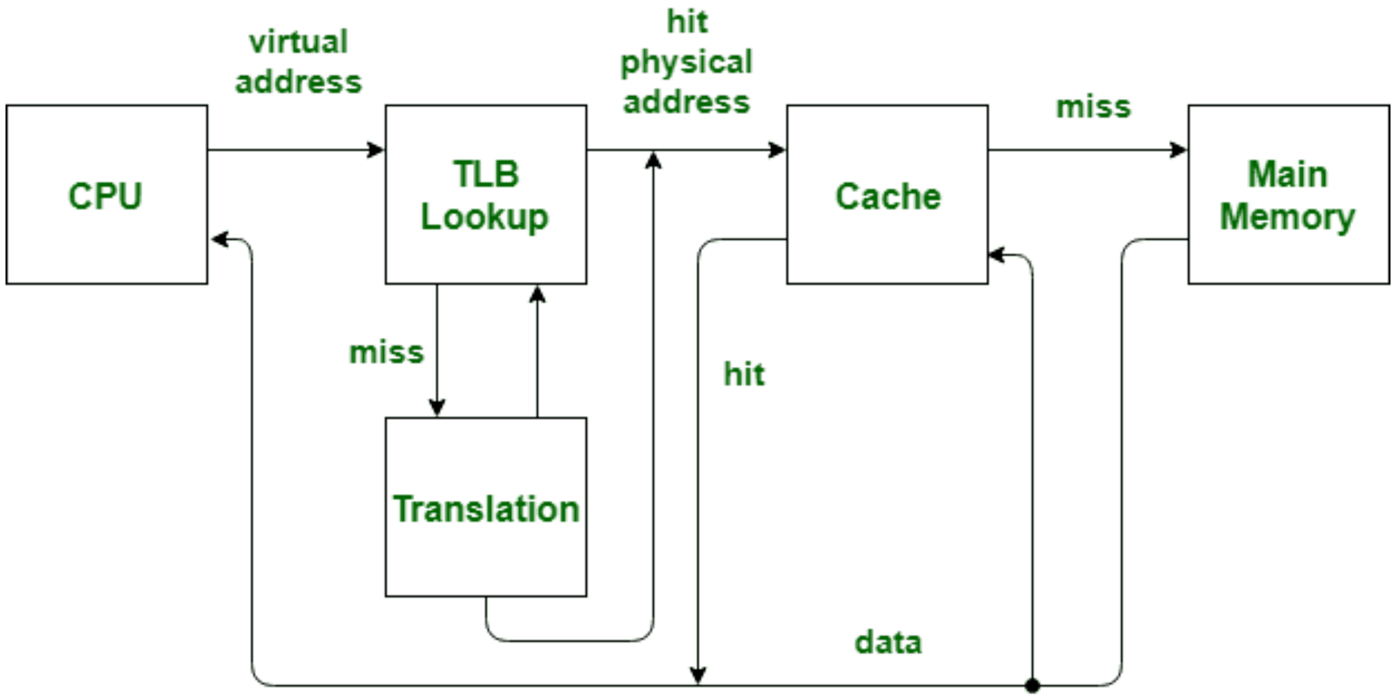


Figure 4-8. Linear-Address Translation to a 4-KByte Page using IA-32e Paging

4级页表



```
> cpuid
cache and TLB information (2):
     0x63: data TLB: 1G pages, 4-way, 4 entries
     0x03: data TLB: 4K pages, 4-way, 64 entries
     0x76: instruction TLB: 2M/4M pages, fully, 8 entries
     0xff: cache data is in CPUID 4
     0xb5: instruction TLB: 4K, 8-way, 64 entries
     0xf0: 64 byte prefetching
     0xc3: L2 TLB: 4K/2M pages, 6-way, 1536 entries
```

Translation: Paging

TLB: Translation Lookaside Buffer

Cache: L1 & L2 Cache

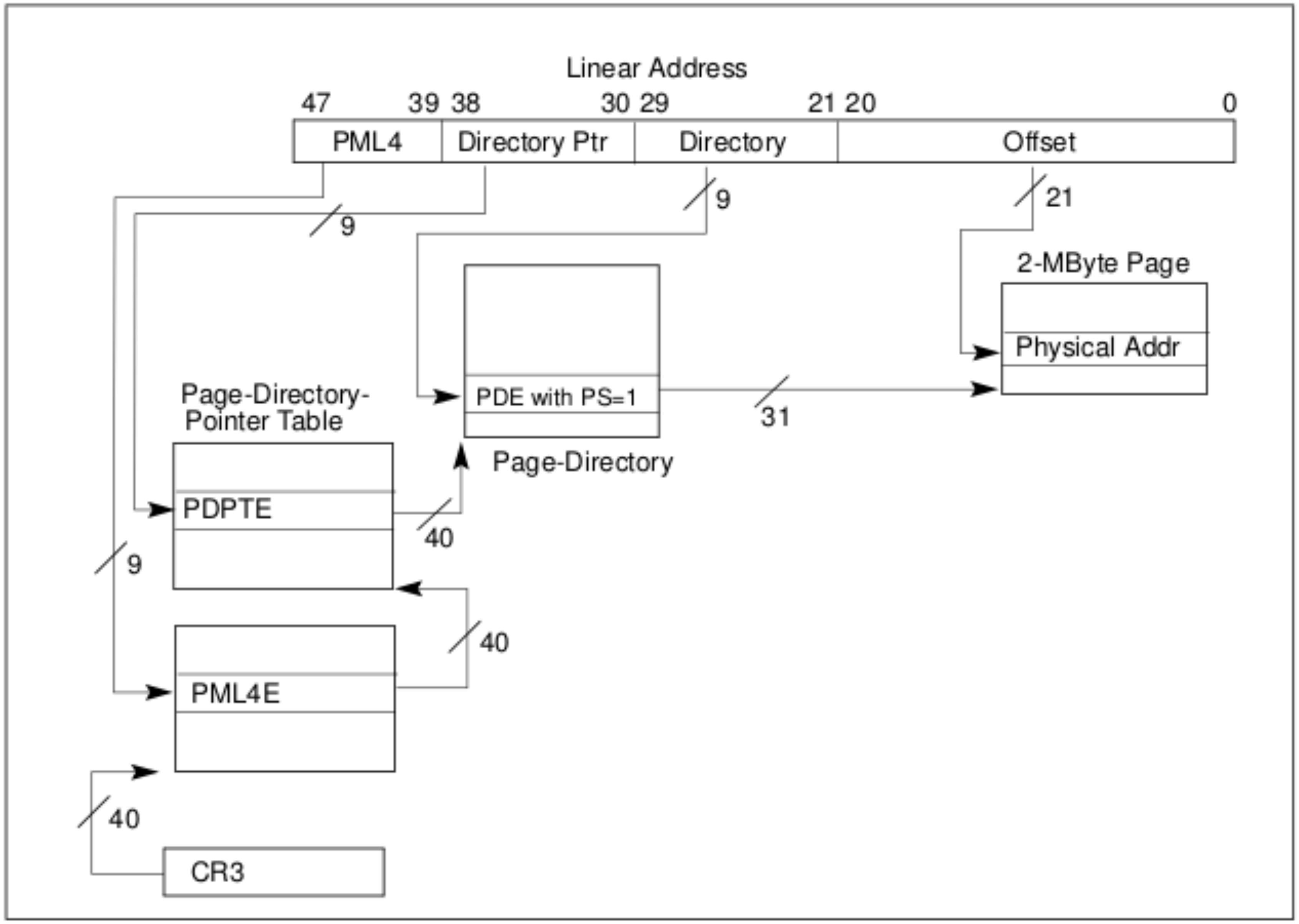
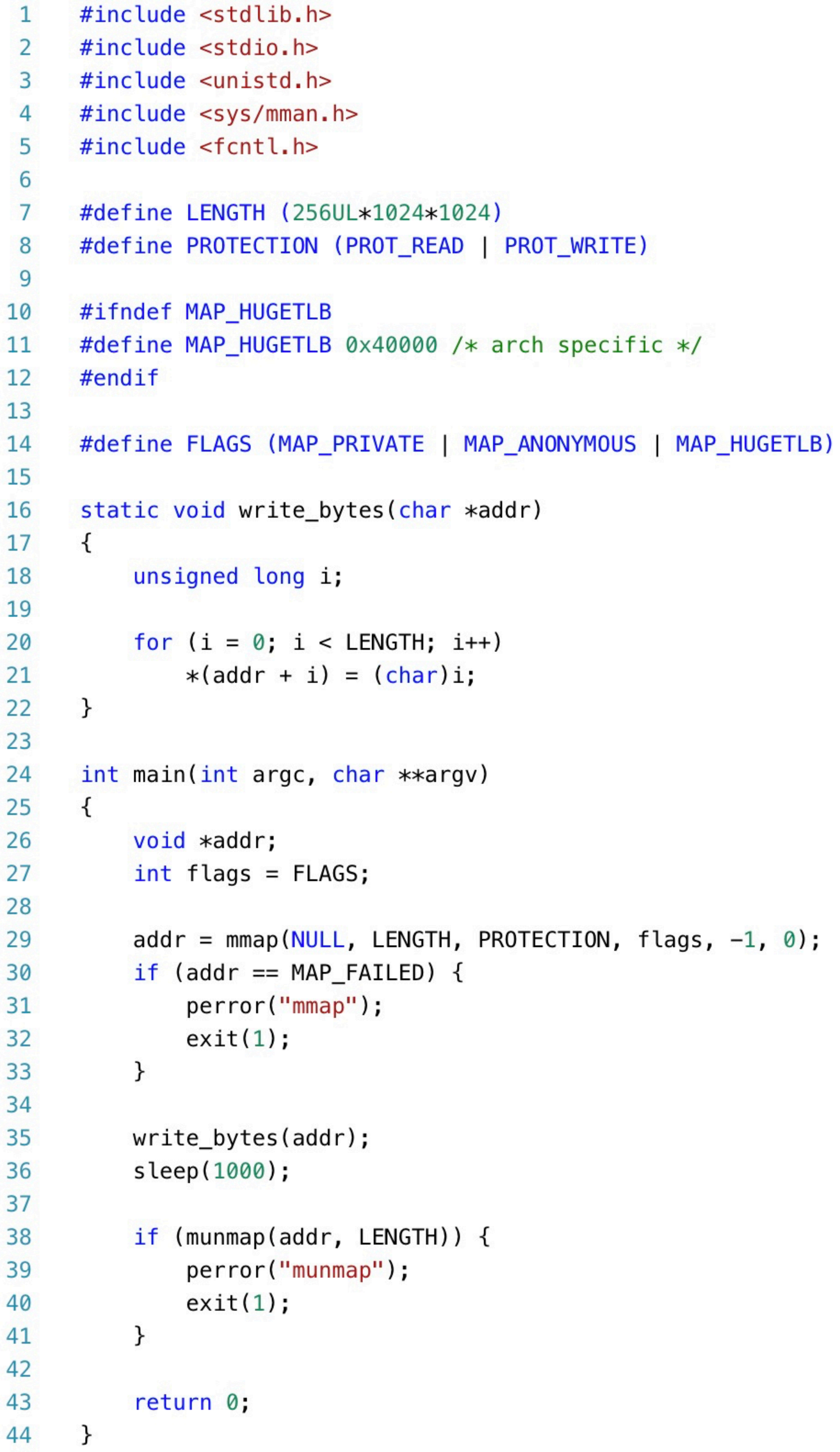


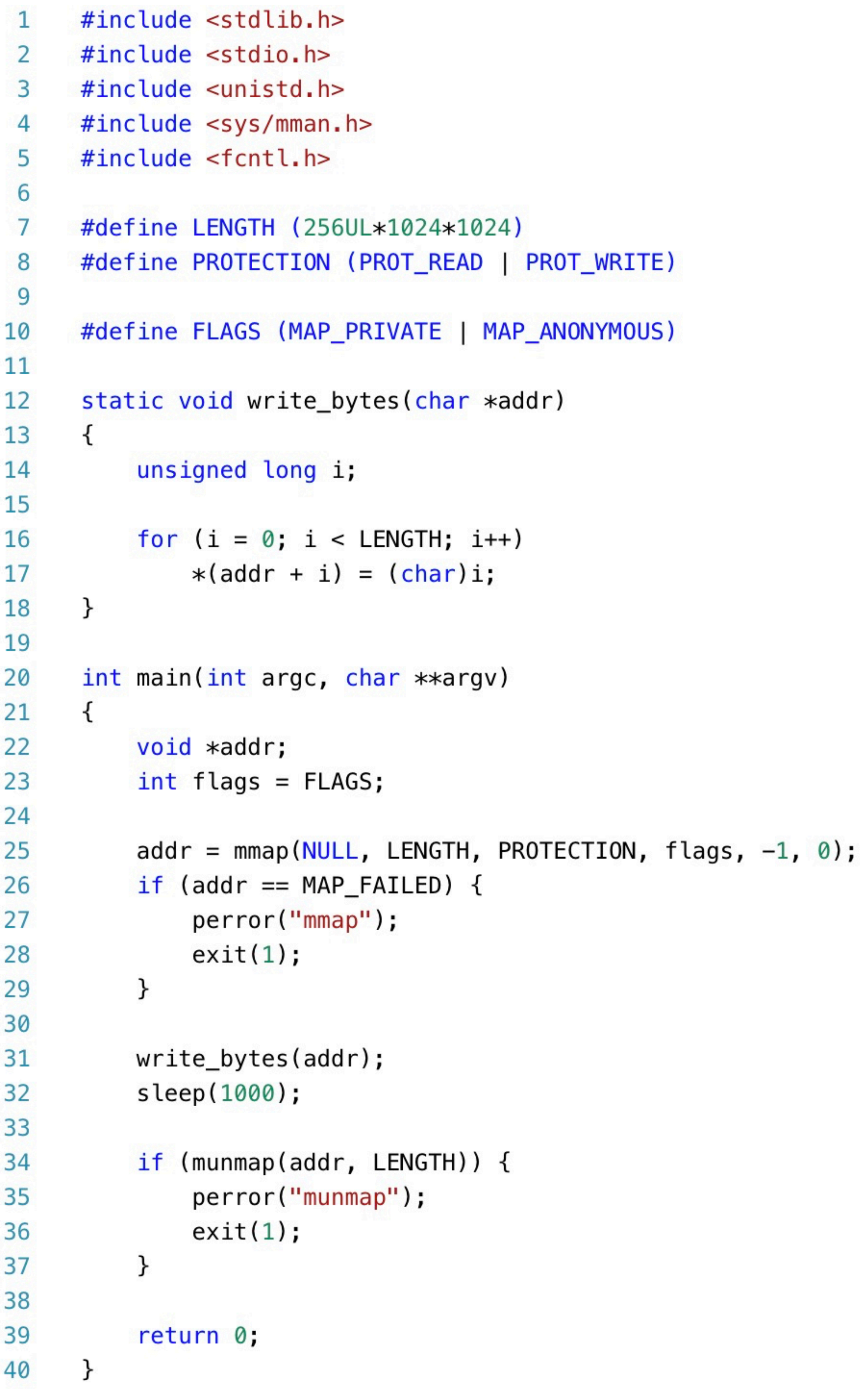
Figure 4-9. Linear-Address Translation to a 2-MByte Page using IA-32e Paging

减少tlb-cache misses

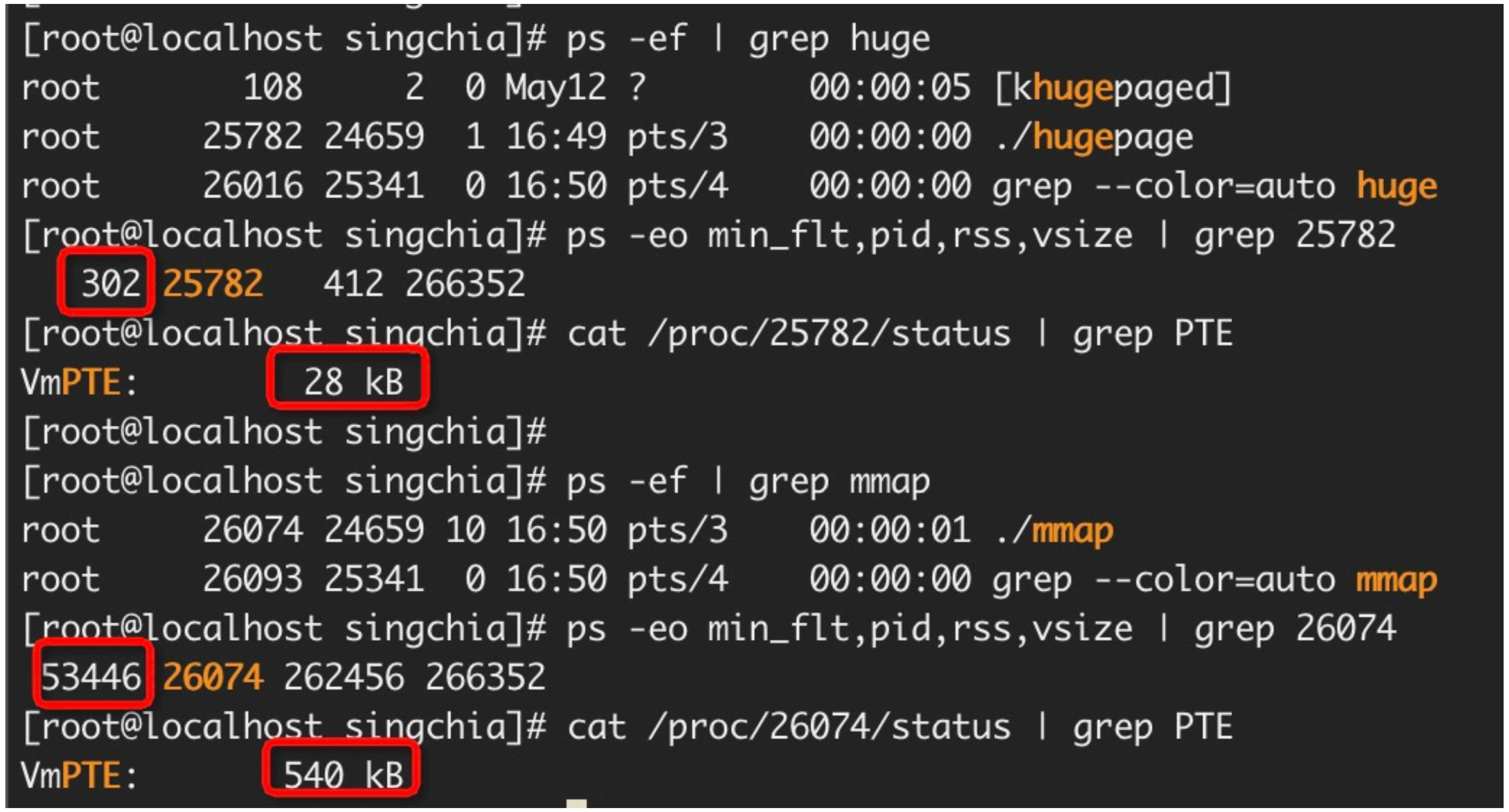
```
perf stat -e dTLB-load-missed,iTLB-load-missed
```



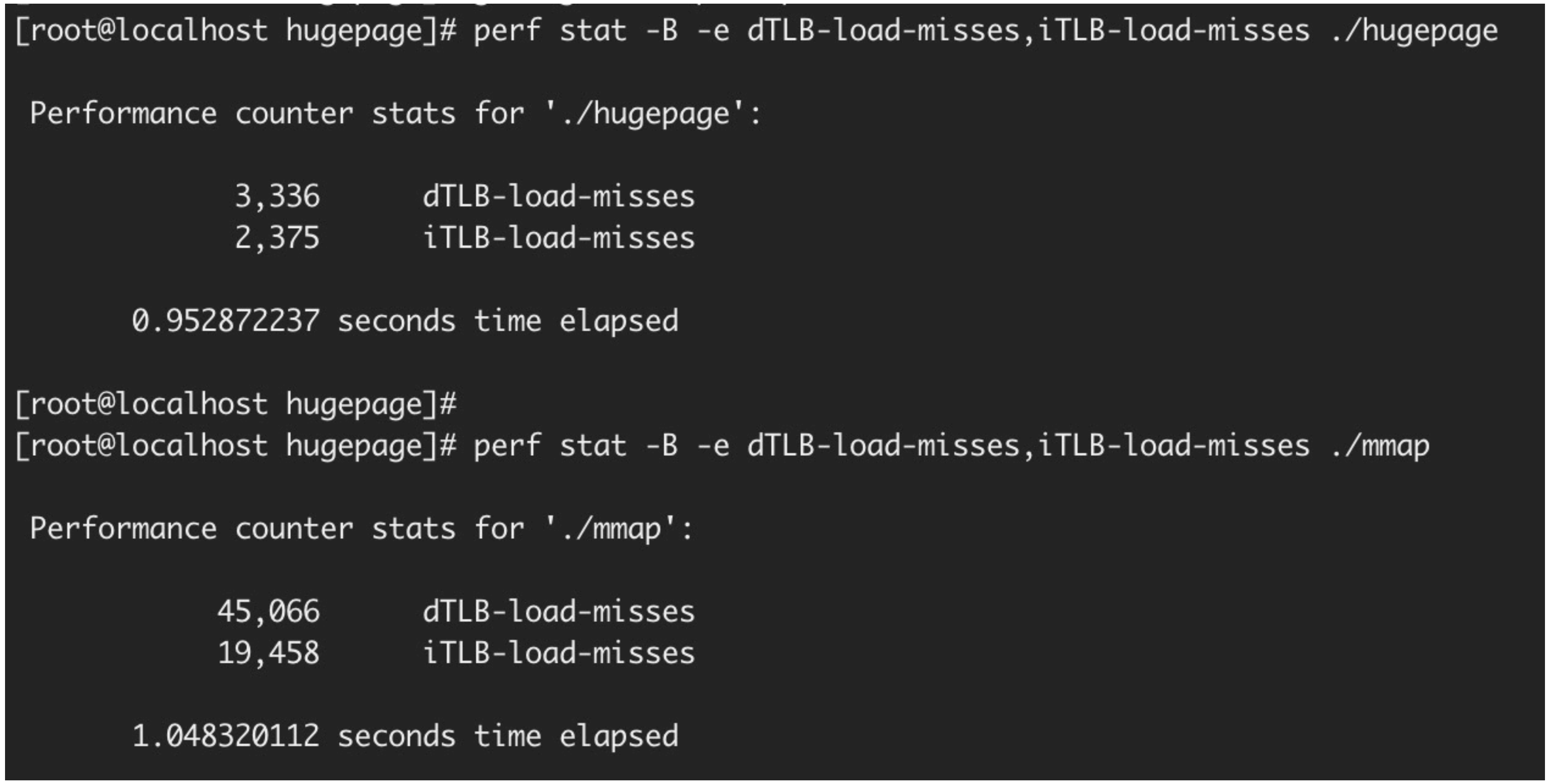
```c
#include <stdlib.h>
#include <stdio.h>
#include <unistd.h>
#include <sys/mman.h>
#include <fcntl.h>

#define LENGTH (256UL*1024*1024)
#define PROTECTION (PROT_READ | PROT_WRITE)

#ifndef MAP_HUGETLB
#define MAP_HUGETLB 0x40000 /* arch specific */
#endif

#define FLAGS (MAP_PRIVATE | MAP_ANONYMOUS | MAP_HUGETLB)

static void write_bytes(char *addr)
{
    unsigned long i;

    for (i = 0; i < LENGTH; i++)
        *(addr + i) = (char)i;
}

int main(int argc, char **argv)
{
    void *addr;
    int flags = FLAGS;

    addr = mmap(NULL, LENGTH, PROTECTION, flags, -1, 0);
    if (addr == MAP_FAILED) {
        perror("mmap");
        exit(1);
    }

    write_bytes(addr);
    sleep(1000);

    if (munmap(addr, LENGTH)) {
        perror("munmap");
        exit(1);
    }

    return 0;
}
```



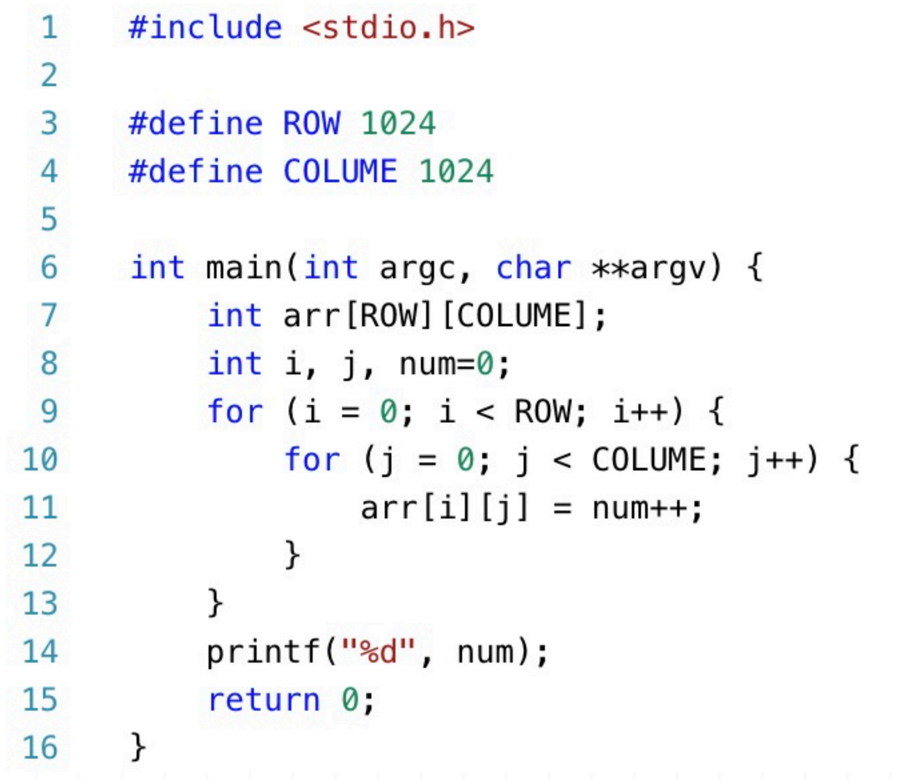
```c
#include <stdlib.h>
#include <stdio.h>
#include <unistd.h>
#include <sys/mman.h>
#include <fcntl.h>

#define LENGTH (256UL*1024*1024)
#define PROTECTION (PROT_READ | PROT_WRITE)

#define FLAGS (MAP_PRIVATE | MAP_ANONYMOUS)

static void write_bytes(char *addr)
{
    unsigned long i;

    for (i = 0; i < LENGTH; i++)
        *(addr + i) = (char)i;
}

int main(int argc, char **argv)
{
    void *addr;
    int flags = FLAGS;

    addr = mmap(NULL, LENGTH, PROTECTION, flags, -1, 0);
    if (addr == MAP_FAILED) {
        perror("mmap");
        exit(1);
    }

    write_bytes(addr);
    sleep(1000);

    if (munmap(addr, LENGTH)) {
        perror("munmap");
        exit(1);
    }

    return 0;
}
```



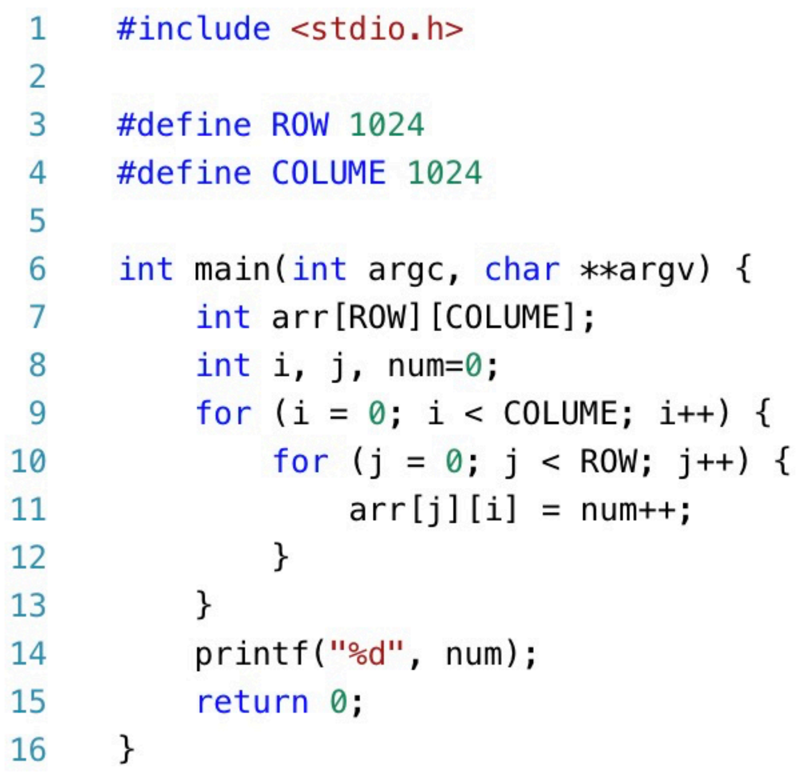
```
[root@localhost singchia]# ps -ef | grep huge
root       108      2  0 May12 ?        00:00:05 [khugepaged]
root     25782 24659  1 16:49 pts/3    00:00:00 ./hugepage
root     26016 25341  0 16:50 pts/4    00:00:00 grep --color=auto huge
[root@localhost singchia]# ps -eo min_flt,pid,rss,vsize | grep 25782
  302 25782   412 266352
[root@localhost singchia]# cat /proc/25782/status | grep PTE
VmPTE:        28 kB
[root@localhost singchia]#
[root@localhost singchia]# ps -ef | grep mmap
root     26074 24659 10 16:50 pts/3    00:00:01 ./mmap
root     26093 25341  0 16:50 pts/4    00:00:00 grep --color=auto mmap
[root@localhost singchia]# ps -eo min_flt,pid,rss,vsize | grep 26074
53446 26074 262456 266352
[root@localhost singchia]# cat /proc/26074/status | grep PTE
VmPTE:       540 kB
```



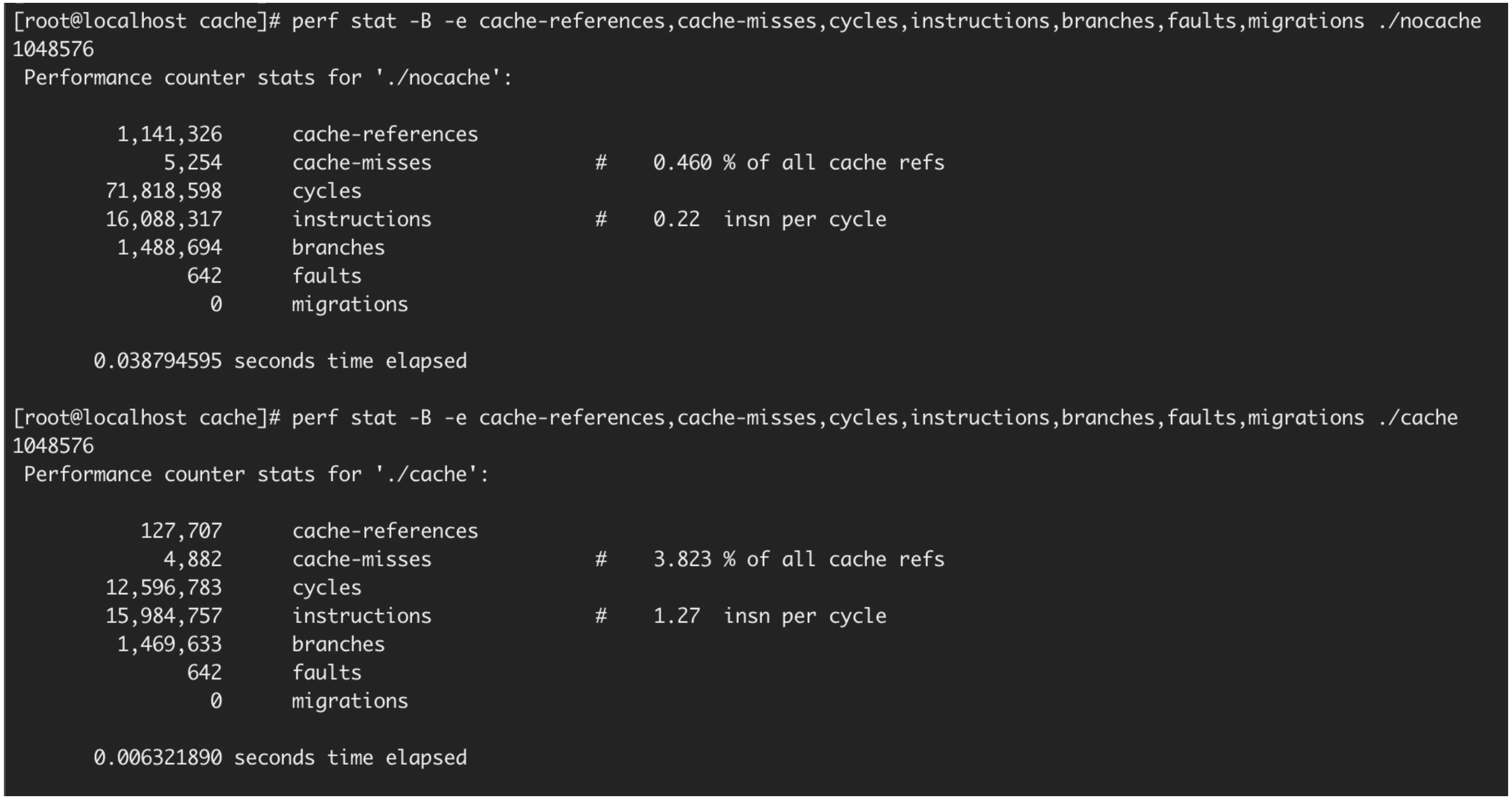
```
[root@localhost hugepage]# perf stat -B -e dTLB-load-misses,iTLB-load-misses ./hugepage

 Performance counter stats for './hugepage':

             3,336      dTLB-load-misses
             2,375      iTLB-load-misses

       0.952872237 seconds time elapsed

[root@localhost hugepage]#
[root@localhost hugepage]# perf stat -B -e dTLB-load-misses,iTLB-load-misses ./mmap

 Performance counter stats for './mmap':

            45,066      dTLB-load-misses
            19,458      iTLB-load-misses

       1.048320112 seconds time elapsed
```



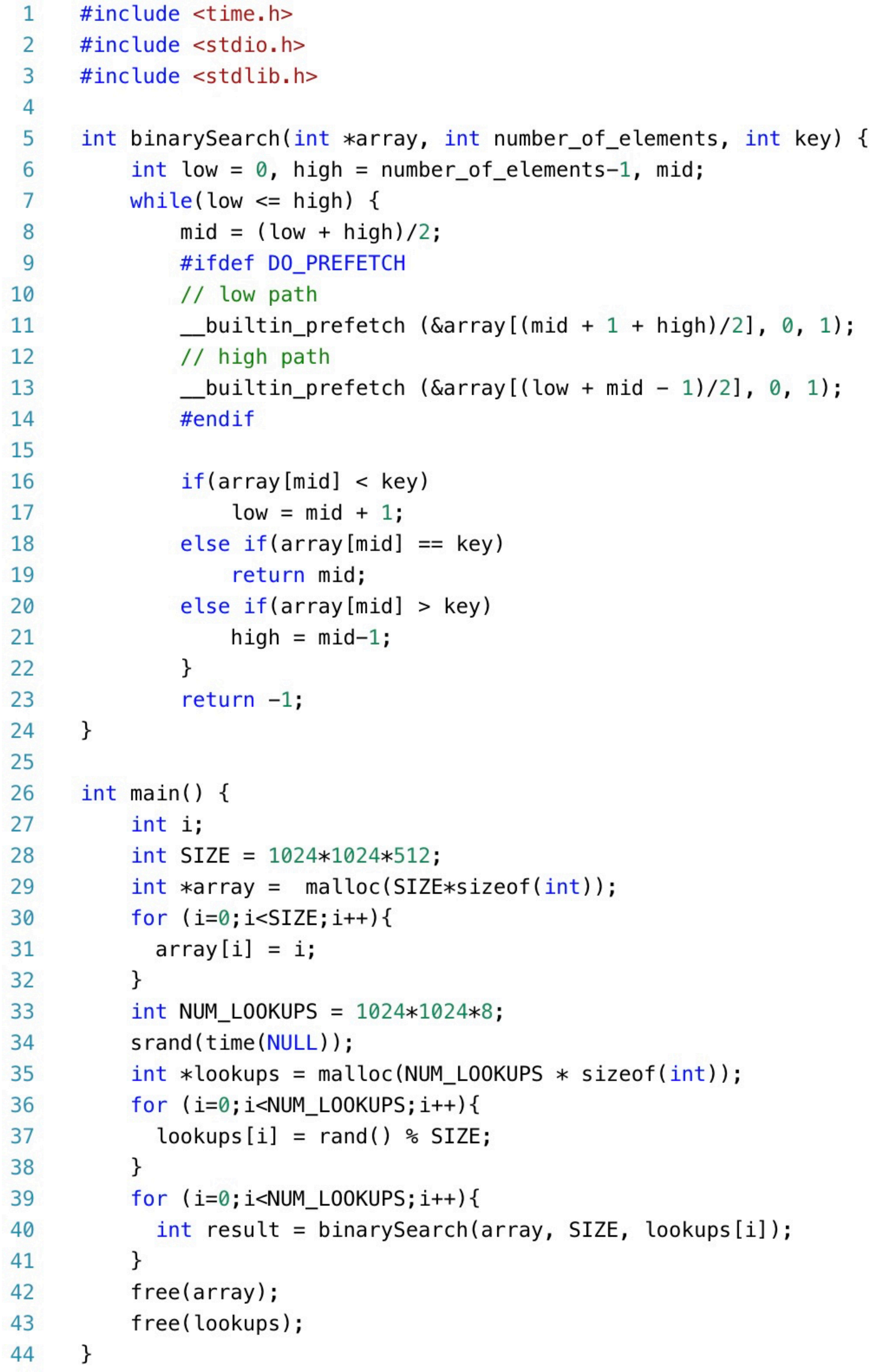
```c
#include <stdio.h>

#define ROW 1024
#define COLUME 1024

int main(int argc, char **argv) {
    int arr[ROW][COLUME];
    int i, j, num=0;
    for (i = 0; i < ROW; i++) {
        for (j = 0; j < COLUME; j++) {
            arr[i][j] = num++;
        }
    }
    printf("%d", num);
    return 0;
}
```



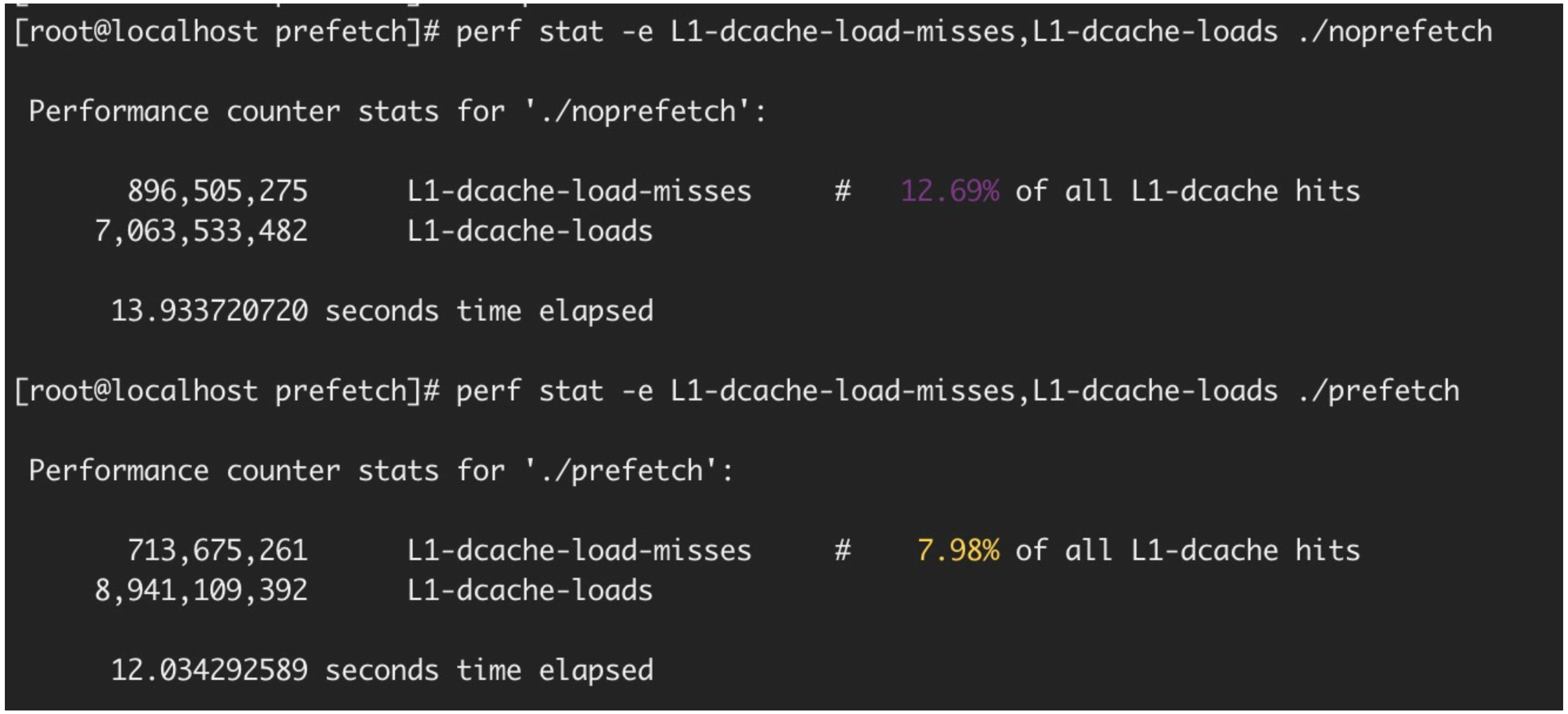
```c
#include <stdio.h>

#define ROW 1024
#define COLUME 1024

int main(int argc, char **argv) {
    int arr[ROW][COLUME];
    int i, j, num=0;
    for (i = 0; i < COLUME; i++) {
        for (j = 0; j < ROW; j++) {
            arr[j][i] = num++;
        }
    }
    printf("%d", num);
    return 0;
}
```



```
[root@localhost cache]# perf stat -B -e cache-references,cache-misses,cycles,instructions,branches,faults,migrations ./nocache
1048576
 Performance counter stats for './nocache':

         1,141,326      cache-references
             5,254      cache-misses              #    0.460 % of all cache refs
        71,818,598      cycles
        16,088,317      instructions              #    0.22  insn per cycle
         1,488,694      branches
               642      faults
                 0      migrations

       0.038794595 seconds time elapsed

[root@localhost cache]# perf stat -B -e cache-references,cache-misses,cycles,instructions,branches,faults,migrations ./cache
1048576
 Performance counter stats for './cache':

           127,707      cache-references
             4,882      cache-misses              #    3.823 % of all cache refs
        12,596,783      cycles
        15,984,757      instructions              #    1.27  insn per cycle
         1,469,633      branches
               642      faults
                 0      migrations

       0.006321890 seconds time elapsed
```



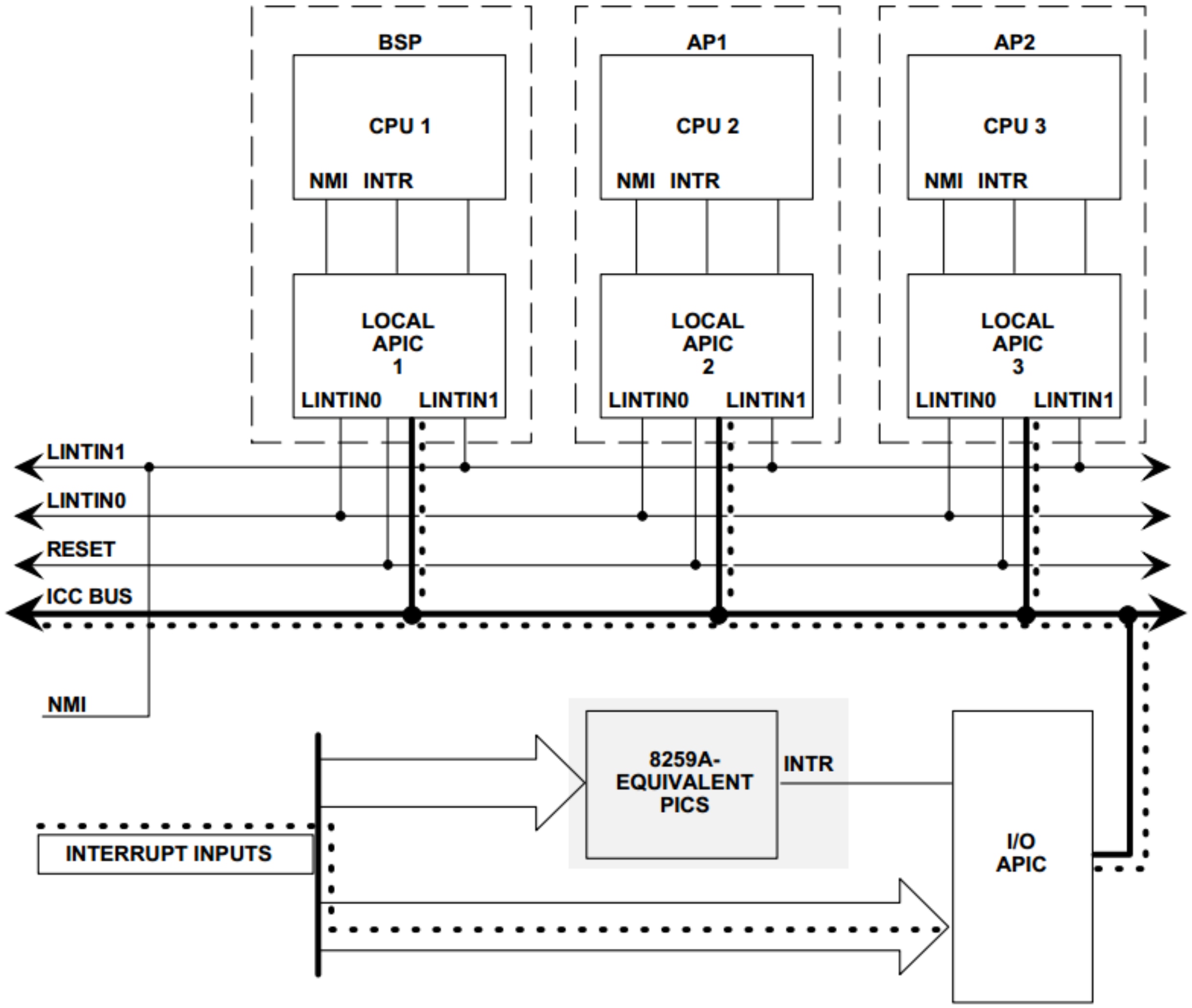
```c
#include <time.h>
#include <stdio.h>
#include <stdlib.h>

int binarySearch(int *array, int number_of_elements, int key) {
    int low = 0, high = number_of_elements-1, mid;
    while(low <= high) {
        mid = (low + high)/2;
        #ifdef DO_PREFETCH
        // low path
        __builtin_prefetch (&array[(mid + 1 + high)/2], 0, 1);
        // high path
        __builtin_prefetch (&array[(low + mid - 1)/2], 0, 1);
        #endif

        if(array[mid] < key)
            low = mid + 1;
        else if(array[mid] == key)
            return mid;
        else if(array[mid] > key)
            high = mid-1;
        }
        return -1;
}

int main() {
    int i;
    int SIZE = 1024*1024*512;
    int *array =  malloc(SIZE*sizeof(int));
    for (i=0;i<SIZE;i++){
      array[i] = i;
    }
    int NUM_LOOKUPS = 1024*1024*8;
    srand(time(NULL));
    int *lookups = malloc(NUM_LOOKUPS * sizeof(int));
    for (i=0;i<NUM_LOOKUPS;i++){
      lookups[i] = rand() % SIZE;
    }
    for (i=0;i<NUM_LOOKUPS;i++){
      int result = binarySearch(array, SIZE, lookups[i]);
    }
    free(array);
    free(lookups);
}
```



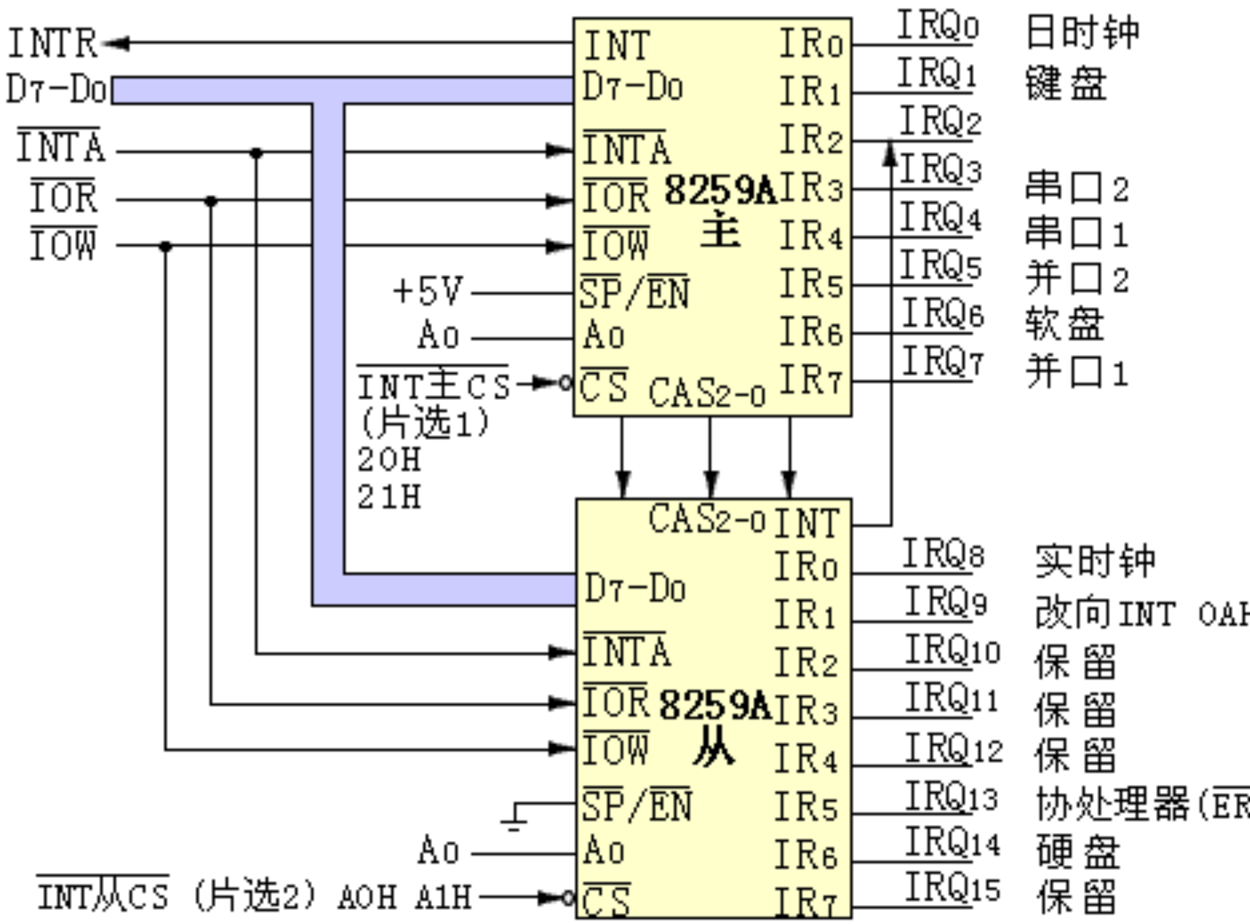
```
[root@localhost prefetch]# perf stat -e L1-dcache-load-misses,L1-dcache-loads ./noprefetch

 Performance counter stats for './noprefetch':

       896,505,275      L1-dcache-load-misses     #   12.69% of all L1-dcache hits
     7,063,533,482      L1-dcache-loads

      13.933720720 seconds time elapsed

[root@localhost prefetch]# perf stat -e L1-dcache-load-misses,L1-dcache-loads ./prefetch

 Performance counter stats for './prefetch':

       713,675,261      L1-dcache-load-misses     #    7.98% of all L1-dcache hits
     8,941,109,392      L1-dcache-loads

      12.034292589 seconds time elapsed
```

内存对齐

- 电气特性，内存访问以offset方式访问，非对齐的内存访问需要多次。
- cpu cache line大小约64B，非对齐可能导致cache misses



SHADED AREAS INDICATE UNUSED CIRCUITS. DOTTED LINE SHOWS INTERRUPT PATH.



- irq balance
- uio

后续

- 多核并发 -> 数据&计算本地化
- 内核bypass -> no kernel-user copy
- 无锁队列
- ddio
- …