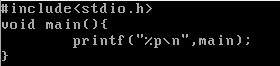
hw 7) Make a system call, sys\_get\_phyloc(), which will display the physical address of main().

1) Write a simple program that prints the address of main().

printmain.c





출력결과, main의 주소는 0x80483b4라는걸 확인할 수 있었다.

2) Call sys\_get\_phyloc(main) in this program which passes the address of main.

3) sys\_get\_phyloc(addr) is a system call that performs following steps in order:

step0: print the value of PGDIR\_SHIFT, PTRS\_PER\_PGD, PAGE\_SHIFT, PTRS\_PER\_PTE

PGDIR\_SHIFT=22: number of shifting to extract directory number from a logical address. Logical address 0x080484a4 = (dir 20h, page 48h, offset 4a4h)

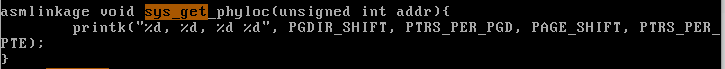
pgd\_index=20h, pte\_index=48h

PAGE\_SHIFT=12: number of shifting to extract page number from a logical address.

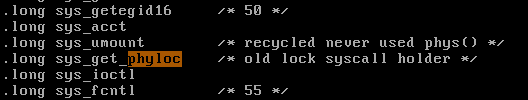
PTRS\_PER\_PGD=1024: number of directory entries in a directory table

PTRS\_PER\_PTE=1024: number of frame pointer entries in a directory

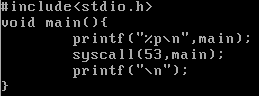
arch/x86/mm/mmap.c



arch/x86/kernel/syscall\_table\_32.S



printmain.c

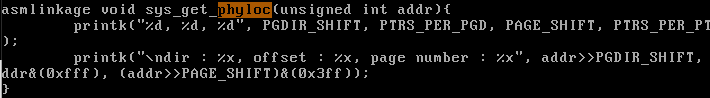


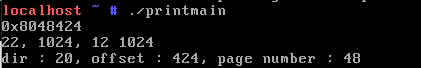


arch/x86/mm/mmap.c에서 sys\_get\_phyloc 함수를 만들어주고, 그것을 syscall\_table\_32.S에서 53번에 명시해주고 printmain.c에서 불렀다. 그 결과, 각각의 결과가 다 맞게 나왔다.

step1: extract directory number (dir), page number(pg), and offset(off) from addr, and display them.

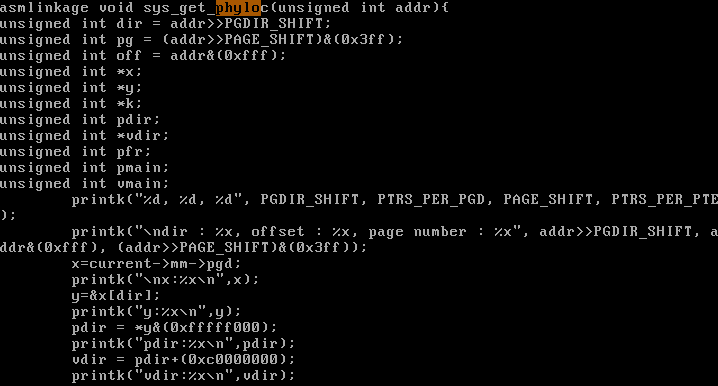
arch/x86/mm/mmap.c

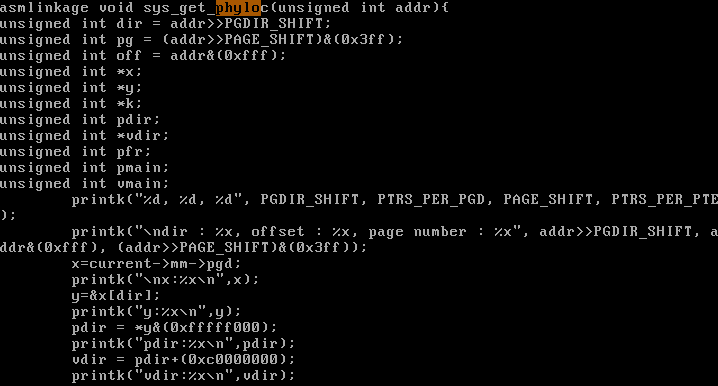




8048424를 2진수로 바꿨을 때 10바이트씩 끊어서 각각 dir, pg, off가 된다. dir은 addr에서 PGDIR\_SHIFT만큼 shift해주면 되고, offset은 addr의 마지막 12비트이기 때문에 addr에 0xfff를 and연산 시켜주면 된다. 마지막으로 pg는 PAGE\_SHIFT가 12이기 때문에 addr>>PAGE\_SHFIT를 해주면 12비트만큼 오른쪽으로 shift해주고 거기에 0x3ff와 and연산 시켜주면 뒤에서 10비트만 가져올 수 있게 된다. 출력결과, dir은 20, pg는 48, offset은 424가 나왔다.

step2: print the location of directory table of the current process: x

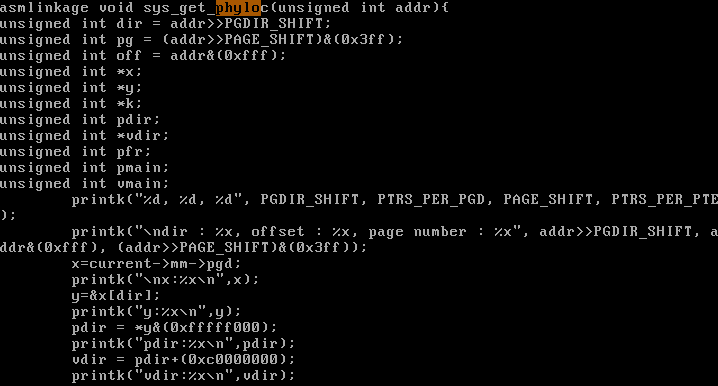




current->mm->pgd에서 x의 가상주소를 출력하도록 했다.

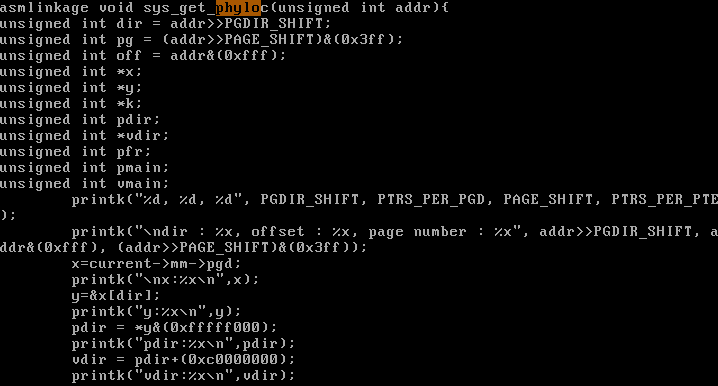
step3: print the location of directory table entry for main(): y

y= &x[dir];



step4: print the physical location of the directory (partial page table) for main():pdir

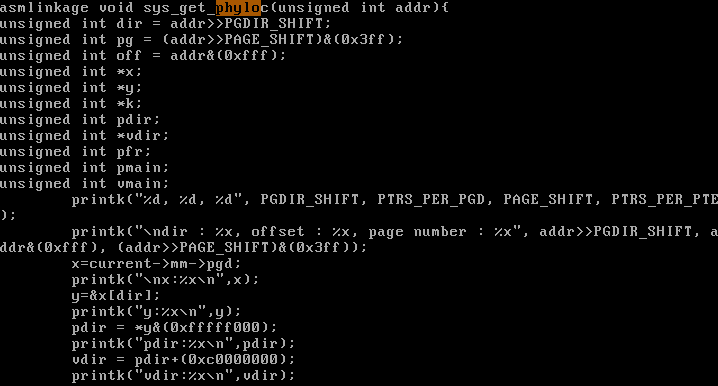
pdir= \*y & 0xfffff000; // the physical address should be at frame boundary



step5: print the virtual address of this directory: vdir

vdir = pdir + 0xc0000000; // physical to virtual mapping for kernel address

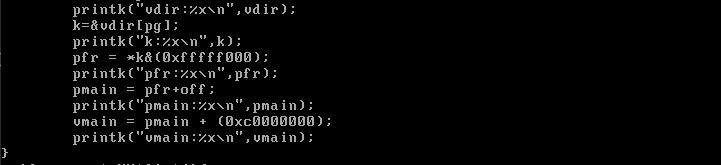
// read about kernel address space in Section 7.4.



물리주소와 가상 주소는 항상 0xc0000000만큼 차이가 난다.

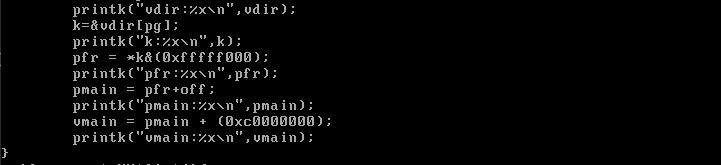
step6: print the location of the frame entry for main(): k

k = &vdir[pg];

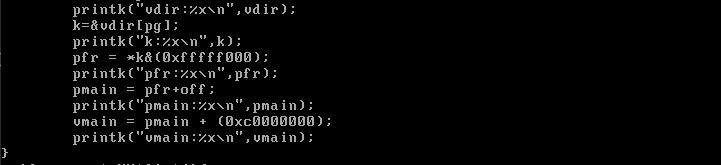


step7: print the physical location of frame for main(): pfr

pfr = \*k & 0xfffff000; // the physical address should be at frame boundary

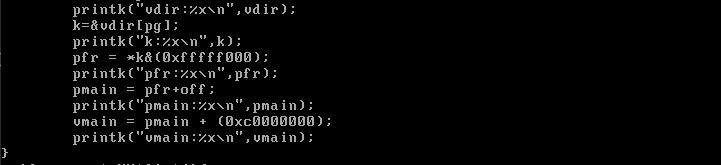


step8: print the physical address of main(): pmain



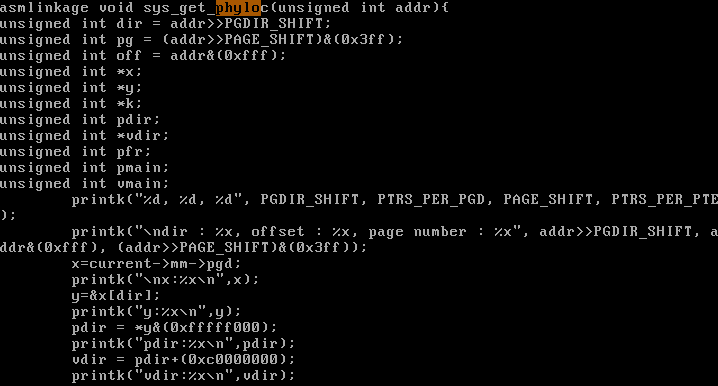
step7에서 구한 pfr에 offset을 더해주면 main의 물리적 주소가 나온다.

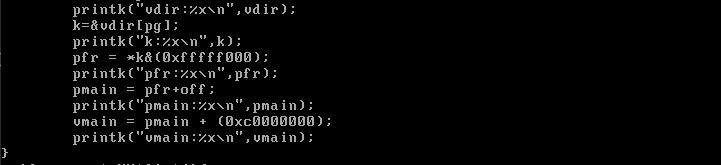
step9: print the virtual address for the physical address of main(): vmain

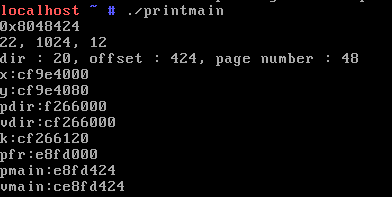


물리주소와 가상주소는 항상 0xc0000000만큼 차이가 나기 때문에, main의 물리주소에 0xc0000000을 더해줘서 가상주소를 구했다.

arch/x86/mm/mmap.c





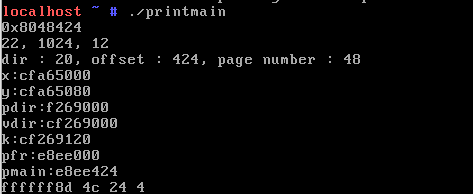


실행해본 결과, 모든게 다 정상적으로 출력됐다.

step10: display the first 4 bytes in it and compare them with the first 4 bytes of main in the original executable file(use "objdump -d program-name" to see the first 4 bytes of main in the original program). If they are same, you have the correct physical address of main.

arch/x86/mm/mmap.c







내가 구한 물리주소가 맞는 주소인지 확인하기 위해서 main의 가상주소를 캐릭터 포인터로 형변환 한 후 vmain의 첫 네바이트가 main을 기계어로 바꿨을 때 첫 네바이트와 같은지 확인해보았다. 그 결과, 둘다 똑같이 8d 4c 24 04였기 때문에 내가 찾은 main의 물리적 주소가 맞는 주소라는걸 확인할 수 있었다.