// main.c, 159

// real kernel, use timer and trapframe

// your team name and team member names here

Also include "entry.h" which defines constant TIMER\_INTR

and TimerEntry that are needed in the following code.

Declare a new data member: the idt\_ptr learned

from the timer lab. We need to locate IDT with this.

int main() {

int pid;

call InitKernelData() (same as in Simulated)

call InitKernelControl() (see below)

create IdleProc (same as in Simulated)

LoadRun(pcb[0].TF\_ptr); // load/run IdleProc

return 0; // this will never be executed

}

Add SetEntry() here, learned from timer lab.

void InitKernelData() is the same as in Simulated

void InitKernelControl() { // learned from timer lab, remember to modify main.h

locate IDT 1st

call SetEntry() to plant TimerEntry jump point

program the mask of PIC

(but NO "sti" which is built into the process trapframe)

}

void Scheduler() is the same as in Simulated

void KernelMain(TF\_t \*TF\_ptr) {

save TF\_ptr to PCB of running process

switch on TF\_ptr->intr\_id {

if it's TIMER\_INTR:

call TimerISR()

dismiss timer event: send PIC with a code

break;

default:

show msg: cons\_printf("Panic: unknown intr ID (%d)!\n", TF\_ptr->intr\_id);

breakpoint(); // fallback to GDB

}

// same as in Simulated:

// poll key and handle keystroke simulated events (s/e/b/q, but no 't' key)

call Scheduler() to chose process to load/run if needed

call LoadRun(pcb[running\_pid].TF\_ptr) to load/run selected proc

}