Group 25: Alvin Zheng (dongxuez) Minghao Wang (minghaow) Yining Yang (yiningy) **Date:** 2013 – 11 – 08

Lab 3: IRQs and Hardware

18342 Fundamentals of Embedded Systems

1. Introduction

In this project, we have ported our SWI handler, exit, read and write syscalls from Lab 2 to this new infra-structure. We write a timer driver by setting up the OS Timers and maintaining a static variable of time. Also, we write the OS timer match IRQ management code for this new kernel. Finally, we write 3 test programs (2 mandatory and 1 extra) to successfully demonstrate our timer.

1.1 Overview

Our kernel implements 3 new features as follow:

- Timer Driver
  - Timer_Driver.c
- IRQ_Handler
  - IRQ_Handler.S: Assmebly half in a separate file
  - C_IRQ_Handler(void): C half in main.c
- Syscalls
  - time.c
  - sleep.c

And 2 mandatory and 1 extra test programs:

- splat.c
- typo.c
- hardestGame.c

2. Detailed Design

2.1 Timer Driver

static size_t timer

1) Elapsed time since kernel boots up in milliseconds.
2) Declared as static for encapsulation so other functions should use interface to access the timer.

initTimer(void)

1) Enable IRQ on the match between OSCR and OSMR0 by setting up OIER, ICMR and ICLR.
2) Configure OSMR to be 10ms (Determined by TIMER_RESOLUTION).
3) Initialize OSCR, OSSR and timer value.

freeTimer(void)

Restore the timer interrupt registers.

addTimer(void)

Increase the timer value every time the IRQ time match is satisfied and IRQ_Handler is called.

getTimer(void)

1) Get the current timer value in millisecond. Typically called by TIME_SWI and SLEEP_SWI.
2) Multiply the timer value by TIMER_RESOLUTION

2.2 IRQ_Handler

IRQ_Handler.S

1) Customized IRQ Handler (Assembly half).
2) Save the context of user mode, transfer control to C_IRQ_Handler.
3) Restore registers, subtract $lr with 4 and finally return to user application.

C_IRQ_Handler(void)

1) IRQ Handler (C half), called by IRQ Handler (Assembly half).
2) Handle OSMR0 timer interrupt only.
3) Increase the timer variable in Timer_Driver.

2.3 Syscalls

time.c

Return getTimer() from Time_Driver.

sleep.c

1) Use a while loop to have the system waiting
2) Unmask IRQ before while loop and mask again after the loop is passed through.

2.4 Test programs

splat.c

Display a spinning cursor using a carriage character (\r).

typo.c

Echo characters of what has been already typed along with the input time.

hardestGame.c

1) Car parking game utilizing time and sleep functions.
2) The car will enter a shaded area and the user should press Enter to stop the car right at the slot.

2.5 Other upgrades

read.c

Enable time match interrupts before waiting for user input, and disable IRQ again after the input is finished.

Call_UseApp.S

Initialize the IRQ stack pointer before transferring control to user application.

APPENDIX 1 – Memory Layout