ECE 598 – Advanced Operating Systems
Lecture 7

Vince Weaver

http://www.eece.maine.edu/~vweaver

vincent.weaver@maine.edu

9 February 2016

Announcements

- Homework #3 was assigned, due Thursday

HW#2 Review

- Code problems: mostly delay too short or second delay asm branching into first.
- Wasted lots of time tracking down obscure bug, where the BSS wasn't where I thought it would be. Literal pools and linker scripts. Turned out to be a missing -c in the Makefile.
- Size: C about 200 bytes, assembly 68 bytes?
  Can look at .dis files for disassembly
  C: 60 bytes of initialization, asm: 12 bytes for delay loop,

on C is 56 bytes (due to pessimization from volatile, etc) also saves/restores LR and registers to maintain calling convention. can't explain some of it

- volatile – have C compiler not optimize away stores
- C array of 32-bit ints vs actually byte-wise access
- SPI1_CEN_2. Bonus SPI ports

Integer to String Conversion

This it the algorithm I use, there are other ways to do it that don't involve the backwards step (starting off by dividing by 1 billion and dividing the divisor by 10 each time).

- Repeatedly divide by 10.
- Digit is the remainder. Repeat until quotient 0.
- Make sure handle 0 case.
- Convert each digit to ASCII by adding 48 ('0')
- Why does the number end up backwards?

Division by 10

- ARM1176 in Pi has no divide routine, why isn't this a problem?
- Generic x=y/z division is not possible without fancy work (iterative subtraction? Newton approximation?)
- Dividing by a constant is easier
- C compiler cheats, for /10 it effectively multiplies by 1/10.

- Look at generated assembly, you'll see it multiply by `0x66666667`

- Why is it not a problem when dividing by 16?

What are interrupts?

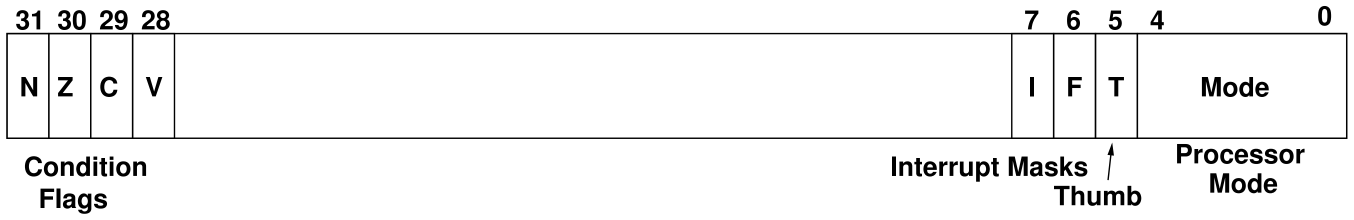
- What types of hardware generate interrupts?
  Keyboard, timers, I/O, etc.

- What can an OS use interrupts for?
  Avoiding polling. Also context switching.

Exceptions and Interrupts

- All architectures are different
- ARM does it a little differently from others.

ARM CPSR Register



- Current Program Status Register
- Contains flags in addition to processor mode
- Six privileged modes: abort, fast interrupt, interrupt, supervisor, system, undefined
- One non-privileged: user (cannot write CPSR)
- Interrupts and exceptions will automatically switch modes

ARM Interrupt Registers

User/Sys	Fast	IRQ	Supervisor	Undefined	Abort
r0					
r1					
r2					
r3					
r4					
r5					
r6					
r7					
r8	r8_fiq				
r9	r9_fiq				
r10	r10_fiq				
r11	r11_fiq				
r12	r12_fiq				
r13/sp	r13_fiq	r13_irq	r13_svc	r13_undef	r13_abt
r14/lr	r14_fiq	r13_irq	r14_svc	r14_undef	r14_abt
r15/pc					
cpsr	spsr_fiq	spsr_irq	spsr_svc	spsrc_undef	spsr_abt

Unlike other architectures, when switching modes the ARM hardware will preserve the status register, PC and stack and give you mode-specific versions (register bank switching). Also for Fast Interrupts r8-r12 are saved as well, allowing fast handlers that do not have to save registers to the stack.

ARM Interrupt Handling

- ARM core saves CPSR to the proper SPSR
- ARM core saves PC to the banked LR (possibly with an offset)
- ARM core sets CPSR to exception mode (disables interrupts)
- ARM core jumps to appropriate offset in vector table

Vector Table

Type	Type	Offset	LR	Priority
Reset	SVC	0x0	–	1
Undefined Instruction	UND	0x04	lr	6
Software Interrupt	SVC	0x08	lr	6
Prefetch Abort	ABT	0x0c	lr-4	5
Data Abort	ABT	0x10	lr-8	2
UNUSED	–	0x14	–	–
IRQ	IRQ	0x18	lr-4	4
FIQ	FIQ	0x1c	lr-4	3

- See ARM ARM ARMv6 documentation for details.
- Defaults to 0x000000. On some ARM you can move to any 32-byte aligned address.
- Interrupts: IRQ = general purpose hardware, FIQ = fast interrupt for really fast response (only 1), SWI = syscalls, talk to OS
- FIQ mode auto-saves r8-r12.
- Different stacks? IRQ mode, SVC mode (boots into), user-mode stack

Ways to return from IRQ

- `subs pc,r14,#4`
  Sneakily branches and gets the right status register (due to S in SUBS)

- `sub r14,r14,#4`
  `...`
  `movs pc,r14`

- Another stores lr and other things to stack, then restores
  `sub r14,r14,#4`

```
stmbd r13!,{r0-r12,r14}
...
ldmfd r13!,{r0-r12,pc}^
```

The caret means to loast cpsr from spsr

Exclamation point means to update r13 after popping.

IRQ Handlers in C

In gcc for ARM, you can specify the interrupt type with an attribute. Automatically restores to right address.

```
void function () __attribute__ ((interrupt ("IRQ")));

/* Can be IRQ, FIQ, SWI, ABORT and UNDEF */

void __attribute__((interrupt("UNDEF"))) undefined_instruction_vector(void) {

    while(1) {
        /* Do Nothing */
    }
}
```