ESCOLA POLITÉCNICA DA UNIVERSIDADE DE SÃO PAULO
Graduação em Engenharia Computação

PCS3732 - Laboratório de Processadores

Professor Jorge Kinoshita

Grupo 10 - Planejamento E10

Arthur Pires da Fonseca NUSP: 10773096

Sumário

Perguntas

- Estude partes do código em http://www.pcs.usp.br/~jkinoshi/2012/exp-int-versatile.pdf
- Junte o código em irq.s e compile (códigos e como rodar no apêndice).
- Em um pdf :

Comente as partes em que o código está dividido e o que faz cada parte.

O código é dividido em 6 partes:

1. Vetor de interrupção: local aonde o kernel recorre quando ocorre algum tipo de interrupção.
2. Constantes: definição das labels que codificam os endereços usados no código.
3. Tratamento das interrupções: descrição das rotinas que são chamadas no vetor de interrupção.
4. Tratamento da interrupção de timer: descrição de como proceder após uma interrupção causada pelo temporizador.
5. Inicialização do timer: valores iniciais personalizados para o timer.
6. Programa principal: chamamento da rotina “timer_init”.

Como o timer é programado?

É necessário mexer em alguns registradores do coprocessador da arquitetura ARM para programar-se o timer.

Isso é feito em 3 passos:

- Bit clear (BIC) do bit número 7 do registrador CPSR: isso ativa interrupções IRQ.
- Mudança do registrador (parte da memória) INTEN para 4: habilitação das interrupções do timer.
- Mudança do valor do registrador TIMER0V: valor inicial do timer.

Como se define o intervalo entre interrupções?

A rotina “handler_timer” define o que acontece entre uma interrupção e outra do timer.

Essa função reseta o pedido de interrupção que foi feito anteriormente, possibilitando que o programa que estava sendo executado antes volte a rodar.

Para que servem os registradores do timer e onde são utilizados no programa (inicialização no programa principal ou interrupt handler)?

TIMER0L - load

Especifica o endereço de load do timer. Não foi especificado no relatório qual o propósito desse registrador.

TIMER0C - control:

Registrador de controle do timer. Dependendo do valor que é colocado nele, o timer irá se comportar de uma forma diferente (definição de modo de operação).

TIMER0X - clear:

Registrador de clear do timer. Identifica se está ocorrendo um pedido de interrupção ou não.

TIMER0V - value:

Serve para especificar o número inicial associado à contagem que o timer faz.

Esse registrador é usado na rotina “timer_init”, que faz parte da inicialização do programa principal.

Para que servem os registradores da controladora de interrupção e onde são utilizados no programa (inicialização no programa principal ou interrupt handler)?

INTEN - enable:

Permite que o timer gere ou não interrupções.

INTPND - status

Registrador de status da interrupção. Informa qual o tipo da interrupção. Especificamente no código em questão, uma interrupção de timer é identificada caso o valor do registrador seja 0x10 = 16_{10}.

INTSEL - select (FIQ, IRQ)

Registrador de seleção. Não foi usado no código, mas serve para alternar entre a geração de FIQs ou IRQs.

O que significa? :

```
LDR r0, INTEN
LDR r1,=0x10 @bit 4 for timer 0 interrupt enable
STR r1,[r0]
```

O trecho de código acima serve para habilitar as interrupções do timer.

Apêndice

Como rodar

```
eabi-as codigo.s codigo.o
eabi-as codigo.s -o codigo.o
eabi-ld -T irqld.ld codigo.o -o program.elf
eabi-bin program.elf program.bin
qemu program.bin
```

Como entrar no qemu gerado

```
eabi-qemu -se program.elf

[gdb] break c_entry
[gdb] continue
[gdb] ...
[gdb] quit
pkill qemu
```

codigo.s

```
.global _start
.text


main:
        bl timer_init @initialize interrupts and timer 0
        stop: b stop

_Reset:
        bl main
        b .
undefined_instruction:
        b .
software_interrupt:
        b do_software_interrupt @vai para o handler de interrupções de software

prefetch_abort:
        b .
data_abort:
        b .
not_used:
        b .
irq:
        b do_irq_interrupt @vai para o handler de interrupções IRQ
```

```
fiq:
        b .
do_software_interrupt: @Rotina de Interrupçãode software
        add r1, r2, r3 @r1 = r2 + r3
        mov pc, r14 @volta p/ o endereço armazenado em r14

do_irq_interrupt: @Rotina de interrupções IRQ
        STMFD sp!, {r0 - r3, LR} @Empilha os registradores
        LDR r0, INTPND @Carrega o registrador de status de interrupção
        LDR r0, [r0]
        TST r0, #0x0010 @verifica se é uma interupção de timer
        BNE handler_timer @vai para o rotina de tratamento da interupção de timer
        LDMFD sp!, {r0 - r3,lr} @retorna
        mov pc, r14

handler_timer:
        LDR r0, TIMER0X
        MOV r1, #0x0
        STR r1, [r0] @Escreve no registrador TIMER0X para limpar o pedido de
interrupção
        @ Inserir código que sera executado na interrupção de timer aqui (chaveamento
de processos, ou alternar LED por exemplo)
        LDMFD sp!, {r0 - r3,lr}
        mov pc, r14 @retorna


timer_init:
        mrs r0, cpsr
        bic r0,r0,#0x80
        msr cpsr_c,r0 @enabling interrupts in the cpsr
        LDR r0, INTEN
        LDR r1,=0x10 @bit 4 for timer 0 interrupt enable
        STR r1,[r0]
        LDR r0, TIMER0C
        LDR r1, [r0]
        MOV r1, #0xA0 @enable timer module
        STR r1, [r0]
        LDR r0, TIMER0V
        MOV r1, #0xff @setting timer value
        STR r1,[r0]
        mov pc, lr

_start:
        b _Reset @posição 0x00 - Reset
        ldr pc, _undefined_instruction @posição 0x04 - Intrução não-definida
        ldr pc, _software_interrupt @posição 0x08 - Interrupção de Software
        ldr pc, _prefetch_abort @posição 0x0C - Prefetch Abort
        ldr pc, _data_abort @posição 0x10 - Data Abort
        ldr pc, _not_used @posição 0x14 - Não utilizado
        ldr pc, _irq @posição 0x18 - Interrupção (IRQ)
        ldr pc, _fiq @posição 0x1C - Interrupção(FIQ)
        _undefined_instruction: .word undefined_instruction
        _software_interrupt: .word software_interrupt
        _prefetch_abort: .word prefetch_abort
```

```
_data_abort: .word data_abort
_not_used: .word not_used

_irq: .word irq
_fiq: .word fiq

INTPND: .word 0x10140000 @Interrupt status register
INTSEL: .word 0x1014000C @interrupt select register( 0 = irq, 1 = fiq)
INTEN: .word 0x10140010 @interrupt enable register
TIMER0L: .word 0x101E2000 @Timer 0 load register
TIMER0V: .word 0x101E2004 @Timer 0 value registers
TIMER0C: .word 0x101E2008 @timer 0 control register
TIMER0X: .word 0x101E200c @timer 0 interrupt clear register
```

irqld.ld

```
ENTRY(_start)
SECTIONS
{
 . = 0x0;
 .text : { * (.text); }
}
```