Interrupt Programming

Young Won Lim
10/31/21

Based on

ARM System-on-Chip Architecture, 2nd ed, Steve Furber

Introduction to ARM Cortex-M Microcontrollers
– Embedded Systems, Jonathan W. Valvano

Digital Design and Computer Architecture,
D. M. Harris and S. L. Harris

ARM assembler in Raspberry Pi
Roger Ferrer Ibáñez

https://thinkingeek.com/arm-assembler-raspberry-pi/

C Interrupt Handlers

```
__irq void IRQHandler (void)
{
    volatile unsigned int *base = (unsigned int *) 0x80000000;     // base = 0x80000000;
    if (*base == 1)                                                // which interrupt was it?
    {
        C_int_handler();                                           // process the interrupt
    }
    *(base+1) = 0;                                                 // clear the interrupt
}
```

https://developer.arm.com/documentation/dui0056/d/handling-processor-exceptions/interrupt-handlers/simple-interrupt-handlers-in-c

C Interrupt Handlers

```
IRQHandler PROC
STMFD    sp!,  {r0-r4,r12,lr}
MOV      r4,   #0x80000000
LDR      r0,   [r4,#0]
SUB      sp,   sp, #4
CMP      r0,   #1
BLEQ     C_int_handler
MOV      r0,   #0
STR      r0,   [r4,#4]
ADD      sp,   sp, #4
LDMFD    sp!,  {r0-r4,r12,lr}
SUBS     pc,   lr, #4
ENDP
```

https://developer.arm.com/documentation/dui0056/d/handling-processor-exceptions/interrupt-handlers/simple-interrupt-handlers-in-c

C Interrupt Handlers

__irq keyword is not used:

```
IRQHandler PROC
STMFD    sp!,  {r4,lr}
MOV      r4,   #0x80000000
LDR      r0,   [r4,#0]
CMP      r0,   #1
BLEQ     C_int_handler
MOV      r0,   #0
STR      r0,   [r4,#4]
LDMFD    sp!,  {r4,pc}
ENDP
```

https://developer.arm.com/documentation/dui0056/d/handling-processor-exceptions/interrupt-handlers/simple-interrupt-handlers-in-c

Dual-channel DMA transfer

```
LDR     r11,  [r8, #IOData]   ; Load port data from the IO device.
STR     r11,  [r9], #4        ; Store it to memory: update the pointer.
CMP     r9,   r10             ; Reached the end ?
SUBLSS  pc,   lr, #4          ; No, so return.
                              ; Insert transfer complete code here.
```

https://developer.arm.com/documentation/dui0056/d/handling-processor-exceptions/interrupt-handlers/example-interrupt-handlers-in-assembly-language

Dual-channel DMA transfer

```
LDR      r13,  [r8, #IOStat]       ; Load status register to find which port caused the interrupt.
TST      r13,  #IOPort1Active
LDREQ    r13,  [r8, #IOPort1]      ; Load port 1 data.
LDRNE    r13,  [r8, #IOPort2]      ; Load port 2 data.
STREQ    r13,  [r9], #4            ; Store to buffer 1.
STRNE    r13,  [r10], #4           ; Store to buffer 2.
CMP      r9,   r11                 ; Reached the end?
CMPLE    r10,  r12                 ; On either channel?
SUBNES   pc,   lr, #4              ; Return
                                   ; Insert transfer complete code here.
```

https://developer.arm.com/documentation/dui0056/d/handling-processor-exceptions/interrupt-handlers/example-interrupt-handlers-in-assembly-language

Prioritization

```
; first save the critical state
SUB        lr,   lr, #4             ; Adjust the return address before we save it.
STMFD      sp!,  {lr}               ; Stack return address
MRS        r14,  SPSR               ; get the SPSR ...
STMFD      sp!,  {r12, r14}         ; ... and stack that plus a
                                    ; working register too.
                                    ; Now get the priority level of the
                                    ; highest priority active interrupt.
MOV        r12,  #IntBase           ; Get the interrupt controller's base address.
LDR        r12,  [r12, #IntLevel]   ; Get the interrupt level (0 to 31).
```

https://developer.arm.com/documentation/dui0056/d/handling-processor-exceptions/interrupt-handlers/example-interrupt-handlers-in-assembly-language

Prioritization

```
; Now read-modify-write the CPSR to enable interrupts.
MRS      r14,      CPSR                ; Read the status register.
BIC      r14,      r14, #0x80          ; Clear the I bit
                                       ; (use 0x40 for the F bit).
MSR      CPSR_c,   r14                 ; Write it back to re-enable
                                       ; interrupts and
LDR      PC,       [PC, r12, LSL #2]   ; jump to the correct handler.
                                       ; PC base address points to this instruction + 8
NOP                                    ; pad so the PC indexes this table.
                                       ; Table of handler start addresses
DCD    Priority0Handler
DCD    Priority1Handler
DCD    Priority2Handler
```

https://developer.arm.com/documentation/dui0056/d/handling-processor-exceptions/interrupt-handlers/example-interrupt-handlers-in-assembly-language

Prioritization

```
; ...
Priority0Handler
STMFD     sp!,          {r0 – r11}     ; Save other working registers.
                                       ; Insert handler code here.
; ...
LDMFD     sp!,          {r0 – r11}     ; Restore working registers (not r12).

; Now read-modify-write the CPSR to disable interrupts.
MRS       r12,          CPSR           ; Read the status register.
ORR       r12,          r12, #0x80     ; Set the I bit
                                       ; (use 0x40 for the F bit).
MSR       CPSR_c,       r12            ; Write it back to disable interrupts.

; Now that interrupt disabled, can safely restore SPSR then return.
LDMFD     sp!,          {r12, r14}     ; Restore r12 and get SPSR.
MSR       SPSR_csxf,    r14            ; Restore status register from r14.
LDMFD     sp!,          {pc}^          ; Return from handler.

Priority1Handler
; ...
```

https://developer.arm.com/documentation/dui0056/d/handling-processor-exceptions/interrupt-handlers/example-interrupt-handlers-in-assembly-language

Context Switch

```
STMIA      r13,         {r0 – r14}^    ; Dump user registers above r13.
MRS        r0,          SPSR           ; Pick up the user status
STMDB      r13,         {r0, lr}       ; and dump with return address below.
LDR        r13,         [r12], #4      ; Load next process info pointer.
CMP        r13,         #0             ; If it is zero, it is invalid
LDMNEDB    r13,         {r0, lr}       ; Pick up status and return address.
MSRNE      SPSR_cxsf,   r0             ; Restore the status.
LDMNEIA    r13,         {r0 – r14}^    ; Get the rest of the registers
NOP
SUBNES     pc,          lr, #4         ; and return and restore CPSR.
                                       ; Insert "no next process code" here.
```

https://developer.arm.com/documentation/dui0056/d/handling-processor-exceptions/interrupt-handlers/example-interrupt-handlers-in-assembly-language

Example

```
    .section .isr_vector
    .align 2
    .globl __isr_vector
__isr_vector:
    .long   __StackTop                   /* Top of Stack */
    .long   Reset_Handler
    .long   NMI_Handler
    .long   HardFault_Handler
    .long   MemoryManagement_Handler
    .long   BusFault_Handler
    .long   UsageFault_Handler
    .long   0                            /*Reserved */
    .long   0                            /*Reserved */
    .long   0                            /*Reserved */
    .long   0                            /*Reserved */
    .long   SVC_Handler
    .long   DebugMon_Handler
    .long   0                            /*Reserved */
    .long   PendSV_Handler
    .long   SysTick_Handler
```

https://interrupt.memfault.com/blog/arm-cortex-m-exceptions-and-nvic

Example

```
 /* External Interrupts */
.long   POWER_CLOCK_IRQHandler
.long   RADIO_IRQHandler
.long   UARTE0_UART0_IRQHandler
.long   SPIM0_SPIS0_TWIM0_TWIS0_SPI0_TWI0_IRQHandler
.long   SPIM1_SPIS1_TWIM1_TWIS1_SPI1_TWI1_IRQHandler
.long   NFCT_IRQHandler
.long   GPIOTE_IRQHandler
.long   SAADC_IRQHandler
.long   TIMER0_IRQHandler
.long   TIMER1_IRQHandler
.long   TIMER2_IRQHandler
.long   RTC0_IRQHandler
.long   TEMP_IRQHandler
.long   RNG_IRQHandler
[...]
```

https://interrupt.memfault.com/blog/arm-cortex-m-exceptions-and-nvic

Example

```
typedef enum {
[...]
  POWER_CLOCK_IRQn      =  0,     /*!< 0  POWER_CLOCK                              */
  RADIO_IRQn            =  1,     /*!< 1  RADIO                                    */
  UARTE0_UART0_IRQn     =  2,     /*!< 2  UARTE0_UART0                             */
  SPIM0_SPIS0_TWIM0_TWIS0_SPI0_TWI0_IRQn
                        =  3,     /*!< 3  SPIM0_SPIS0_TWIM0_TWIS0_SPI0_TWI0        */
  SPIM1_SPIS1_TWIM1_TWIS1_SPI1_TWI1_IRQn
                        =  4,     /*!< 4  SPIM1_SPIS1_TWIM1_TWIS1_SPI1_TWI1        */
  NFCT_IRQn             =  5,     /*!< 5  NFCT                                     */
  GPIOTE_IRQn           =  6,     /*!< 6  GPIOTE                                   */
  SAADC_IRQn            =  7,     /*!< 7  SAADC                                    */
  TIMER0_IRQn           =  8,     /*!< 8  TIMER0                                   */
  TIMER1_IRQn           =  9,     /*!< 9  TIMER1                                   */
  TIMER2_IRQn           = 10,     /*!< 10 TIMER2                                   */
  RTC0_IRQn             = 11,     /*!< 11 RTC0                                     */
[...]
  } IRQn_Type;
```

https://interrupt.memfault.com/blog/arm-cortex-m-exceptions-and-nvic

Example

```
(gdb) p/x *(uint32_t*)0xE000E400
$1 = 0x0
(gdb) set *(uint32_t*)0xE000E400=0xff
(gdb) p/x *(uint32_t*)0xE000E400
$2 = 0xe0
```

https://interrupt.memfault.com/blog/arm-cortex-m-exceptions-and-nvic

Example

```
void PendSV_Handler(void) {
  __asm("bkpt 1");
}

__attribute__((optimize("O0")))

static void trigger_pendsv(void) {
  volatile uint32_t *icsr = (void *)0xE000ED04;
  // Pend a PendSV exception using by writing 1 to PENDSVSET at bit 28
  *icsr = 0x1 << 28;
  // flush pipeline to ensure exception takes effect before we
  // return from this routine
  __asm("isb");
}
```

https://interrupt.memfault.com/blog/arm-cortex-m-exceptions-and-nvic

Example

```
(gdb) c
Continuing.

Program received signal SIGTRAP, Trace/breakpoint trap.
PendSV_Handler () at ../../../main.c:48
48      __asm("bkpt 1");
(gdb)
```

```
(gdb) p/x (*(uint32_t*)0xE000ED04)&0xff
$2 = 0xe
(gdb) p/x (*(uint32_t*)0xE000ED04)>>11&0x1
$3 = 0x1
(gdb) p/x (*(uint32_t*)0xE000ED04)>>12&0xff
$4 = 0x0
```

https://interrupt.memfault.com/blog/arm-cortex-m-exceptions-and-nvic

Example

```
__attribute__((optimize("O0")))
void POWER_CLOCK_IRQHandler(void) {
  __asm("bkpt 2");
  trigger_pendsv();
  __asm("bkpt 3");
}
```

https://interrupt.memfault.com/blog/arm-cortex-m-exceptions-and-nvic

Example

```
static void trigger_nvic_int0(void) {
  // Let's set the interrupt priority to be the
  // lowest possible for the NRF52. Note the default
  // NVIC priority is zero which would match our current pendsv
  // config so no pre-emption would take place if we didn't change this
  volatile uint32_t *nvic_ipr = (void *)0xE000E400;
  *nvic_ipr = 0xe0;

  // Enable the POWER_CLOCK_IRQ (External Interrupt 0)
  volatile uint32_t *nvic_iser = (void *)0xE000E100;
  *nvic_iser |= 0x1;

  // Pend an interrupt
  volatile uint32_t *nvic_ispr = (void *)0xE000E200;
  *nvic_ispr |= 0x1;

  // flush pipeline to ensure exception takes effect before we
  // return from this routine
  __asm("isb");
}
```

https://interrupt.memfault.com/blog/arm-cortex-m-exceptions-and-nvic

Example

```
Program received signal SIGTRAP, Trace/breakpoint trap.
POWER_CLOCK_IRQHandler () at ../../../main.c:53
53    __asm("bkpt 2");
(gdb) p/x *(uint32_t*)0xE000ED04
$1 = 0x810

(gdb) next
54    trigger_pendsv();
```

https://interrupt.memfault.com/blog/arm-cortex-m-exceptions-and-nvic

Example

```
(gdb) c
Continuing.

Program received signal SIGTRAP, Trace/breakpoint trap.
PendSV_Handler () at ../../../main.c:38
38    __asm("bkpt 1");
(gdb) bt
#0  PendSV_Handler () at ../../../main.c:38
#1  <signal handler called>
#2  0x00000306 in trigger_pendsv () at ../../../main.c:48
#3  0x00000322 in POWER_CLOCK_IRQHandler () at ../../../main.c:54
#4  <signal handler called>
#5  0x000003a8 in trigger_nvic_int0 () at ../../../main.c:76
#6  main (a=<optimized out>, argv=<optimized out>) at ../../../main.c:129
(gdb) p/x *(uint32_t*)0xE000ED04
$2 = 0xe
```

https://interrupt.memfault.com/blog/arm-cortex-m-exceptions-and-nvic

Example

```
(gdb) p/x *(uint32_t[16] *)0xE000E300
$3 = {0x1, 0x0 <repeats 15 times>}
```

```
(gdb) next
POWER_CLOCK_IRQHandler () at ../../../main.c:55
55      __asm("bkpt 3");
(gdb) p/x *(uint16_t[16] *)0xE000E300
$4 = {0x1, 0x0 <repeats 15 times>}
(gdb) p/x *(uint32_t*)0xE000ED04
$5 = 0x810
```

https://interrupt.memfault.com/blog/arm-cortex-m-exceptions-and-nvic

References

[1] http://wiki.osdev.org/ARM_RaspberryPi_Tutorial_C
[2] http://blog.bobuhiro11.net/2014/01-13-baremetal.html
[3] http://www.valvers.com/open-software/raspberry-pi/
[4] https://www.cl.cam.ac.uk/projects/raspberrypi/tutorials/os/downloads.html