CS 3468 – Homework 5

Stuart Olsen

November 12, 2014

Chapter 8

What is the difference between an architecture-specific device driver and a generic device driver?

An *architecture-specific* driver manages the hardware integrated into the master processor, while a *generic* driver manages hardware located on the board and not integrated into the master processor.

Give two examples of each.

MMU and on-chip memory drivers are architecture-specific; PCI and USB drivers are generic.

Define at least ten types of device drivers that would be needed based on the block diagram shown in Figure 8-47.

- SDRAM controller driver
- Flash storage driver
- EEPROM driver
- JTAG driver
- IR driver
- UART driver
- Ethernet driver
- Display driver
- HDLC driver
- CAN driver

List and describe five types of device driver functions.

- *Interrupt-Handling Startup* involves initialization of interrupt hardware.
- *Interrupt-Handling Shutdown* involves configuring interrupt hardware into a power-off state.
- *Interrupt-Handling Disable* allows other software to disable active interrupts.
- *Interrupt-Handling Enable* allows other software to enable inactive interrupts.
- *Interrupt-Handler Servicing* involves executing interrupt-handling code.

What are the three main types of interrupts? List examples in which each type is triggered.

- *Software* interrupts are triggered explicitly by code in the instruction stream being executed.
- *Internal hardware* interrupts are generated by hardware in response to conditions in the instruction stream.
- *External hardware* interrupts are initiated by hardware other than the master CPU.

What is the difference between an auto-vectored and an interrupt-vectored scheme?

An interrupt-vectored system provides for the ability to use different code to handle different interrupts; in an auto-vectored scheme, it is the responsibility of the single interrupt handler to discriminate between interrupts.

Name and describe four examples of device driver functions that can be implemented for I/O.

- *I/O Acquire* allows software to gain singular access to I/O.
- *I/O Release* allows software to free or unlock I/O.
- *I/O Read* allows software to read data from I/O.
- *I/O Write* allows software to write data to I/O.

Additional Questions

In ATmega128, assume a 32-bit integer A is stored in `r4:r7`, and another 32-bit integer B is stored in `r8:r11`.

Make an AVR function that implements $C = A + B$. The function stores the return in `r12:r15`.

```
add32:
    ; C = A
    mov r12, r4
    mov r13, r5
    mov r14, r6
    mov r15, r7
    ; C += B
    add r12, r8
    adc r13, r9
    adc r14, r10
    adc r15, r11
    ; Done
    ret
```

Make an AVR program that implements $A{+}{=}B$. The program shall call the function $C = A + B$ for the computation.

```
add32a:
    ; C = A + B
    call add32
    ; A = C
    mov r4, r12
    mov r5, r13
    mov r6, r14
    mov r7, r15
```

The MICAz mote has $4352\ \mathrm{B}$ of data RAM. Make an AVR function to clean the data memory. The function should complete the following steps:

- Clean the `SREG` register;
- Set the stack pointer to the bottom of data RAM;
- Clear the whole data RAM to 0.

```
    .include "m128def.inc"

start:
    ; Set up the stack
    ldi r16, low(RAMEND)
    ldi r17, high(RAMEND)
    out SPL, r16
    out SPH, r17
    ; Clear data memory
    ldi XL, low(SRAM_START)
    ldi XH, high(SRAM_START)
    ldi r18, 0x00
clrloop:
```

```
    cp XL, r16
    cpc XH, r17
    breq clrdone
    st X+, r18
    rjmp clrloop
clrdone:
    ; Clear SREG
    out SREG, r18
```