数字逻辑与处理器基础实验
32 位 MIPS 处理器设计

孙伟艺*
白钦博†
王敏虎 ‡

2017 年 7 月 24 日

*无 52 2015011010
†无 52 2015010996
‡无 52 2015011003

1 实验目的

- 熟悉现代处理器的基本工作原理
- 掌握单周期和流水线处理器的设计方法

2 设计方案

2.1 ALU

(孙伟艺)

2.2 单周期数据通路

(孙伟艺)

2.3 流水线数据通路

(白钦博)

2.4 外设

本次实验需要使用 LED 灯、七段数码管、串口等外设参与 CPU 工作，为 CPU 提供运算所需要的数据并将 CPU 的运算结果表示出来。外设不是 CPU 的硬件组成部分，对于 CPU，外设被看作内存中的一个普通地址，CPU 不需要了解外设工作的具体细节，只需要执行程序员编写好的程序，将数据存入对应的地址即可，后续工作应当由外设电路独立完成。

本次实验中，地址0x40000000 到0x40000020 被用于外设地址。在 CPU 的连接中，这些地址被连接到专门的电路而非内存中。

地址如下表被分配给外设。

地址	功能
0x40000000	定时器 TH
0x40000004	定时器 TL
0x40000008	定时器控制 TCON
0x4000000C	LED
0x40000010	Switch
0x40000014	七段数码管
0x40000018	UART 发送数据
0x4000001C	UART 接收数据
0x40000020	串口状态

写出外设 verilog 描述文件

```
always@(*) begin
        if(rd) begin
                case(addr)
                        32'h40000000: rdata <= TH;
                        32'h40000004: rdata <= TL;
                        32'h40000008: rdata <= {29'b0,TCON};
                        32'h4000000C: rdata <= {24'b0,led};
                        32'h40000010: rdata <= {24'b0,switch};
                        32'h40000014: rdata <= {20'b0,digi};
                        32'h40000018: rdata <= {24'b0,TX_DATA};
                        32'h4000001C: begin
                                rdata <= {24'b0,RX_DATA};
                                RX_GET <= 0;
                        end
                        32'h40000020: rdata <= {30'b0,RX_GET,TX_STATUS};
                        default: rdata <= 32'b0;
                endcase
        end
        else
                rdata <= 32'b0;
        if (RX_STATUS == 1) RX_GET<=1;
end

always@(negedge reset or posedge clk) begin
        if(~reset) begin
                TH <= 32'b0;
                TL <= 32'b0;
                TCON <= 3'b0;
                TX_SEND <= 0;
                led <= 8'b0;
        end
        else begin
                if(TCON[0]) begin        //timer is enabled
                        if(TL==32'hffffffff) begin
                                TL <= TH;
                                if(TCON[1]) TCON[2] <= 1'b1;                //irq is enabled
                        end
                        else TL <= TL + 1;
```

```
                end

                if(wr) begin
                        case(addr)
                                32'h40000000: TH <= wdata;
                                32'h40000004: TL <= wdata;
                                32'h40000008: TCON <= wdata[2:0];
                                32'h4000000C: led <= wdata[7:0];
                                32'h40000014: digi <= wdata[11:0];
                                32'h40000018: begin
                                        TX_DATA <= wdata[7:0];
                                        TX_SEND <= 1;
                                end
                                default:;
                        endcase
                end
                if (TX_SEND == 1) TX_SEND <= 0;
        end
end
```

以下为实验要求编写的各外设的说明

2.4.1 LED 灯

LED 灯是本次实验中比较简单的外设。在 CPU 访问0x4000000C 地址时将对应位数赋给相应管脚即可。

2.4.2 七段数码管

本次实验使用四个七段数码管，由于要求使用软件译码，硬件部分较为简单，七段数码管使用 12 位进行控制，前 4 位表示当前点亮的数码管位置，后八位表示七段数码管各管脚的电平，与 LED 灯类似的是，当 CPU 访问0x40000014 时，将对应位数赋给对应接线即可。复杂的译码和控制部分将在汇编程序中说明。

2.4.3 switch 开关

switch 开关是输入设备，不能被写入，当 CPU 试图读0x40000010 时，将对应连线上的电平返回。

2.4.4 串口

串口使用轮询方式编写，三位0x40000018,0x4000001C,0x40000020 控制，其中0x40000020 是串口状态位，其最后两位中的第一位用来标示是否收到新的数据，第二位用来表示目前串口的发送状态。当数据被写入0x40000018 时，串口将自动发送其后八位并修改串口发送状态，当0x4000001C 访问后，串口将修改串口状态位，将接收标志位置为低以等待下一个数据。

编写汇编程序时，应当首先访问串口状态位，确定串口已经接收到数据，再访问串口接收数据地址。使用轮询方式的一个问题即是，如果轮询时间过长，串口中的数据可能会丢失，但是在本实验的条件下，9600 波特率的串口接收一次数据的时间足以 CPU 完成上万个时钟周期的运算，可以认为串口数据能够被及时访问。

2.5 汇编程序与汇编器

2.5.1 汇编程序

本次实验中的汇编程序由两部分组成，一部分是从串口读入数据并运行算法计算最大公约数，另一部分是数码管的译码和显示，由于实验中定时器的中断只用来完成数码管的扫描，因此数码管的译码和显示代码就是中断处理代码。

第一部分代码如下。首先启动定时器，而后检查串口标志位，当串口标志位有效时读串口数据。待输入的两个数据均读取完成后，计算结果并访问外设以显示结果。当完成任务后，CPU 将进入无限循环的状态以便观察结果。

第二部分的代码如首先将处理与定时器相关的中断，而后保护现场，由于程序比较简单，主程序与中断处理程序未使用相同的寄存器，因此这一步在代码中未体现。中断处理代码首先检查数码管对应外设数据，并移动扫描位，而后进行软件译码，软件译码的过程即 case 块语法转换成汇编语法，较为繁琐，因此在报告中删去部分。软件译码完成后，修改数码管外设对应地址值，重新启动定时器，回到主程序继续执行。

两部分代码见于附录。

2.5.2 汇编器

汇编器使用 python 语言编写。汇编器依照以下步骤执行工作：

1. 遍历代码，计算 Label 名称对应的地址值
2. 遍历代码，将 Label 名称表示的地址转换成相对地址或绝对地址
3. 遍历代码，将汇编程序转换成机器码

匹配和替换工作主要使用正则表达式完成，主要代码见于附录。

3 关键代码与文件清单

\ \ assemble

```
\       assemble.py  汇编器程序
\       data.txt 实验使用测试程序
\       encode.py 机器码转换为verilog文件结构程序
\       m_code.txt 汇编器转换机器码
\       verilog.txt 直接贴入verilog rom.v 文件程序
\
\ \ OneCycle
\   \   Adder.v 全加器
\   \   ALU.v 单周期ALU
\   \   Control.v 单周期控制信号生成文件
\   \   CPU.v 单周期CPU结构文件
\   \   DataMem.v 单周期Data Memory
\   \   digitube_scan.v
\   \   divclk.v 分频模块
\   \   Peripheral.v 外设模块
\   \   regfile.v 寄存器
\   \   rom.v 指令存储器
\   \   UART.v 串口相关电路实现
\   \
\   \ \ output_files
\ \ Pipeline
\   \   Adder.v 全加器
\   \   ALU.v 流水线ALU
\   \   Control.v 流水线控制信号生成文件
\   \   CPU.v 流水线CPU结构文件
\   \   DataMem.v 流水线Data Memory
\   \   digitube_scan.v
\   \   Peripheral.v 外设模块
\   \   regfile.v 寄存器
\   \   rom.v 指令存储器
\   \   UART.v 串口相关电路实现
\   \
\   \ \ output_files
```

4 仿真结果与分析

5 综合情况

6 硬件调试情况

7 思想体会