

COD LAB3 实验报告

姓名：刘芷辰

学号：PB21111728

日期：2023.4.18

1. 实验题目

汇编程序设计

2. 实验目标

- 理解RISC-V常用32位整数指令功能
- 熟悉RISC-V汇编仿真软件RARS，掌握程序调试的基本方法
- 掌握RISC-V简单汇编程序设计以及存储器初始化文件（COE)的生成方法
- 理解CPU调试模块PDU的使用方法

3. 实验内容

3.1 PART 1（设计汇编程序）

必做部分（只是用10种指令及其伪指令）

- 代码

```
1  .data
2  save: .word 0x0001,0x0001
3
4  .text
5  # 将n加载到寄存器t0中
6  addi t0,zero,40
7
8  la a0, save
9  # 将第一项和第二项加载到寄存器t1和t2中
10 lw t1, 0(a0)
11 lw t2, 4(a0)
```

```

12 # 将第一项保存到内存中
13 sw t1, 0(a0)
14 # 将第二项保存到内存中
15 sw t2, 4(a0)
16 # 初始化计数器i为2
17 li t3, 2
18 loop:
19     # 如果i = n, 跳出循环
20     beq t3, t0, end
21     # 计算下一项的值
22     add t4, t1, t2
23     # 保存下一项的值到内存中
24     add t5, t3, t3
25     add t5, t5, t3
26     add t5, t5, t3          #4*i
27     add t5, t5, a0
28     sw t4, 0(t5)
29     # 更新t1和t2
30     add t1, zero, t2
31     add t2, zero, t4
32     # 更新计数器i
33     addi t3, t3, 1
34     # 继续循环
35     jal zero loop
36 end:
37

```

n的加载通过addi指令改变，设置计数器t3，每更新一次加1，在小于n时，将储存在t1的数和存储在t2的数相加得到下一项，然后更新t1和t2中的内容

实验结果

Data Segment										
Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)		
0x00000000	1	1	2	3	5	8	13	21		
0x00000004	55	89	144	233	377	610	987			
0x00000008	1597	2584	4181	6765	10946	17711	28657	46368		
0x0000000c	75025	121393	196418	317811	514229	832040	1346209	2178309		
0x00000010	3524518	5702887	9227465	14930352	24157817	39088169	63245986	102334155		
0x00000014	0	0	0	0	0	0	0	0		
0x00000018	0	0	0	0	0	0	0	0		
0x0000001c	0	0	0	0	0	0	0	0		
0x00000020	0	0	0	0	0	0	0	0		
0x00000024	0	0	0	0	0	0	0	0		
0x00000028	0	0	0	0	0	0	0	0		
0x0000002c	0	0	0	0	0	0	0	0		
0x00000030	0	0	0	0	0	0	0	0		
0x00000034	0	0	0	0	0	0	0	0		
0x00000038	0	0	0	0	0	0	0	0		
0x0000003c	0	0	0	0	0	0	0	0		
0x00000040	0	0	0	0	0	0	0	0		

选做部分（实现n的输入）

代码

```

1  .data
2  save: .word 0x0001,0x0001
3  kbsr: .word 0x7f00
4  kbdr: .word 0x7f04
5  .text
6  addi t2,zero,1
7  addi t3,zero,32          #空格作为终止符
8  addi x11,zero,2          #计数,判断是否两位数
9  cal:
10 beq x11,zero,doubdigit   #一位数
11 addi t5,t4,-48
12 jal zero start
13 doubdigit:               #两位数
14 addi t6,t4,-48
15 slli x12,t5,3
16 add x12,x12,t5
17 add t5,x12,t5
18 add t5,t5,t6
19 start:
20 lw t1, kbsr
21 lw t1, 0(t1)
22 bne t1,t2,start          #判断kbsr
23 lw t4, kbdr
24 lw t4, 0(t4)
25 addi x11,x11,-1
26 bne t4, t3,cal           #判断是否为终止符
27
28 # 将n加载到寄存器t0中
29
30 add t0, t5,zero
31
32 la a0, save
33 # 将第一项和第二项加载到寄存器t1和t2中
34 lw t1, 0(a0)
35 lw t2, 4(a0)
36 # 将第一项保存到内存中
37 sw t1, 0(a0)
38 # 将第二项保存到内存中
39 sw t2, 4(a0)
40 # 初始化计数器i为2
41 li t3, 2
42 loop:
43     # 如果i >= n, 跳出循环
44     bge t3, t0, end
45     # 计算下一项的值

```

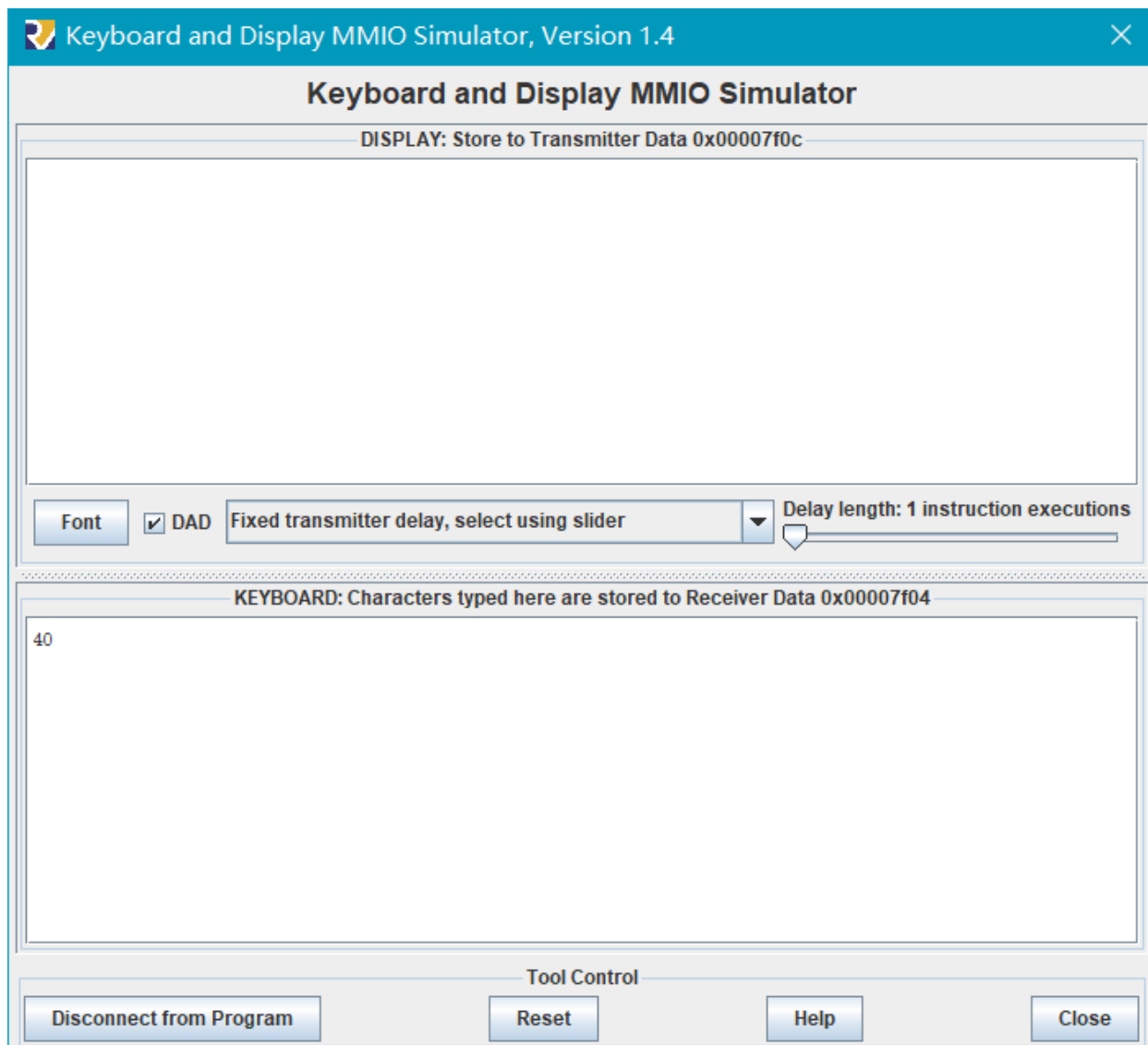
```

46     add t4, t1, t2
47     # 保存下一项的值到内存中
48     slli t5, t3, 2                # 4*i
49     add t5, t5, a0
50     sw t4, 0(t5)
51     # 更新t1和t2
52     add t1, zero, t2
53     add t2, zero, t4
54     # 更新计数器i
55     addi t3, t3, 1
56     # 继续循环
57     jal zero loop
58 end:
59

```

实现轮询输入N：每次读取KBSR地址的数据，若不是1，则继续轮询等待输入，直到输入数据后，KBSR被置1，将KBDR中的数据判断是否为终止符，不是则第一位存入t5，第二位存入t6，并判断此时是否已经读入了两位，一位则继续轮询直到终止符，两位则将第一位存入t5的ascii码换算为十进制乘上10再加上第二位，实现了两位数n的输入，然后将得到的n存入t0中，后续和必做一样

- 实验结果



Data Segment									
Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)	
0x00000000	1	1	2	3	5	8	13	21	
0x00000020	34	55	89	144	233	377	610	987	
0x00000040	1597	2584	4181	6765	10946	17711	28657	46368	
0x00000060	75025	121393	196418	317811	514229	832040	1346269	2178309	
0x00000080	3524578	5702887	9227465	14930352	24157817	39088169	63245986	102334155	
0x000000a0	0	0	0	0	0	0	0	0	
0x000000c0	0	0	0	0	0	0	0	0	
0x000000e0	0	0	0	0	0	0	0	0	
0x00000100	0	0	0	0	0	0	0	0	
0x00000120	0	0	0	0	0	0	0	0	
0x00000140	0	0	0	0	0	0	0	0	
0x00000160	0	0	0	0	0	0	0	0	
0x00000180	0	0	0	0	0	0	0	0	
0x000001a0	0	0	0	0	0	0	0	0	

3.2 PART 2（设计32位移位寄存器）

32位移位寄存器代码

```

1  module Shift_reg(
2      input rst,
3      input clk,           // Work at 100MHz clock
4

```

```

5     input [31:0] din,    // Data input
6     input [3:0] hex,    // Hexadecimal code for the switches
7     input add,          // Add signal
8     input del,          // Delete signal
9     input set,          // Set signal
10
11     output reg [31:0] dout // Data output
12 );
13
14     reg [31:0] shift_reg;
15
16
17     always @(posedge clk or posedge rst) begin
18         if (rst) begin
19             shift_reg <= 32'b0;
20             dout <= 32'b0;
21         end
22         // Set
23         else if (set) begin
24             shift_reg <= din;
25         end
26         // Add
27         else if (add) begin
28             shift_reg <= {shift_reg[27:0], hex};
29         end
30         // Delete
31         else if (del) begin
32             shift_reg <= {4'b0, shift_reg[31:4]};
33         end
34
35         dout <= shift_reg[31:0];
36     end
37
38
39 endmodule
40

```

通过拼接的方式实现移位

FPGA烧写检查

已在线下检查

4. 总结

本次实验难度适中

更加熟悉了RISC-V的指令操作，并且学习了如何通过轮询的方式从键盘输入数据

本次有实验文档之后对实验理解起来容易一些了，希望之后的实验都能有实验文档