计算机组成原理实验3报告

PB20000156
徐亦昶

Task1 10条指令的自动化测试

首先在0号位置存放0xffff并手动测试sw,lw和beq指令：

```
.data
data0:.word 0xffff
.text
sw x0,0(x0) #Test sw
lw t0,0(x0) #Test lw
beq t0,x0,CONTINUE1 #Test beq
beq x0,x0,EXIT
CONTINUE1:
```

随后使用已经测试好的跳转和数据转移指令测试其他指令。当结果符合预期时跳到下一条测试，否则程序结束。最后将寄存器s0置为1并结束程序。
完整代码：

```
.data
data0:.word 0xffff
data1:.word 0x001a
data2:.word 0x0001
test_add:.word 0x001b
test_addi:.word 0x001e
test_sub:.word 0x001d
test_auipc:.word 0x4068
.text
sw x0,0(x0) #Test sw
lw t0,0(x0) #Test lw
beq t0,x0,CONTINUE1 #Test beq
beq x0,x0,EXIT
CONTINUE1:
lw t0,4(x0)
blt x0,t0,CONTINUE2 #Test blt
beq x0,x0,EXIT
CONTINUE2:
lw t0,data1
lw t1,data2
add t0,t0,t1 #Test add
lw t2,test_add
beq t0,t2,PASS1
beq x0,x0,EXIT
PASS1:
addi t0,t0,0x0003 #Test addi
```

```
lw t2,test_addi
beq t0,t2,PASS2
beq x0,x0,EXIT
PASS2:
sub t0,t0,t1 #Test sub
lw t2,test_sub
beq t0,t2,PASS3
beq x0,x0,EXIT
PASS3:
auipc t0,0x0001 #Test auipc
lw t2,test_auipc
beq t0,t2,PASS4
beq x0,x0,EXIT
PASS4:
addi ra,x0,0
jal ra,CONTINUE3 #Test jal
addi s0,x0,1
beq x0,x0,EXIT
CONTINUE3:
addi t0,ra,-4
jalr ra,4(t0) #Test jalr
addi t0,x0,0xff
EXIT:
```

测试lw、sw、beq时先肉眼观察，发现正常进入CONTINUE1，随后进行自动化测试，运行后寄存器如图所示：

zero	0	0x00000000
ra	1	0x00003094
sp	2	0x00002ffc
gp	3	0x00001800
tp	4	0x00000000
t0	5	0x00003080
t1	6	0x00000001
t2	7	0x00004068
s0	8	0x00000001
s1	9	0x00000000
a0	10	0x00000000
a1	11	0x00000000
a2	12	0x00000000
a3	13	0x00000000
a4	14	0x00000000
a5	15	0x00000000
a6	16	0x00000000
a7	17	0x00000000
s2	18	0x00000000
s3	19	0x00000000
s4	20	0x00000000
s5	21	0x00000000
s6	22	0x00000000
s7	23	0x00000000
s8	24	0x00000000
s9	25	0x00000000
s10	26	0x00000000
s11	27	0x00000000
t3	28	0x00000000
t4	29	0x00000000
t5	30	0x00000000
t6	31	0x00000000
pc		0x0000309c

s0的值是1，可见测试通过。

Task2 排序

程序分模块编写：

sort：排序的核心，a0表示数组基址，a1表示数组长度。

```
sort: #a0 stores the base address,a1 stores the legnth of the array
addi t0,a0,0
```

```
add a1,a1,a1 #a1*=4
add a1,a1,a1
add t1,t0,a1
addi t2,t0,0 #i
OUTER_FOR:
beq t2,t1,END_OUTER
addi t3,t2,4 #j
INNER_FOR:
beq t3,t1,END_INNER
lw t4,(t2)
lw t5,(t3)
ble t4,t5,NEXT_INNER #no need to swap
addi t6,t4,0 #swap
addi t4,t5,0
addi t5,t6,0
sw t4,(t2) #restore
sw t5,(t3)
NEXT_INNER:
addi t3,t3,4 #j moves on
jal x0,INNER_FOR
END_INNER:
addi t2,t2,4 #i moves on
jal x0,OUTER_FOR
END_OUTER:
jalr x0,0(ra)
```

使用了冒泡排序。
getchar：读入一个字符，并返回给a0。

```
getchar:
getchar_LOOP:
lw t0,keybd_st
lw t0,(t0)
beq t0,x0,getchar_LOOP
lw a0,keybd_data
lw a0,(a0)
jalr x0,0(ra)
```

期间程序会不断检测0x7f00(keybd_st，即MMIO键盘输入的ready bit)，如果为1说明有按键输入，对该按键数据进行读取(0x7f04,keybd_data)，读取后ready bit会自动置0。
scanf：输入一个十六进制数字，字符小写，输出返回给a0。

```
scanf:
addi sp,sp,-4
sw ra,(sp)
addi t2,x0,0 #result
scanf_LOOP:
jal ra,getchar
addi t0,x0,48 #'0'
```

```
addi t1,x0,57 #'9'
blt a0,t0,EXAM2
bgt a0,t1,EXAM2
jal x0,PROCESS_dig
EXAM2:
addi t0,x0,97 #'a'
addi t1,x0,102 #'f'
blt a0,t0,EXIT_scanf
bgt a0,t1,EXIT_scanf
addi a0,a0,-39
PROCESS_dig:
addi a0,a0,-48
add t2,t2,t2 #t2=t2<<4+int16(a0)
add t2,t2,t2
add t2,t2,t2
add t2,t2,t2
add t2,t2,a0
jal x0,scanf_LOOP
EXIT_scanf:
addi a0,t2,0
lw ra,(sp)
addi sp,sp,4
jalr x0,0(ra)
```

程序不断调用getchar读入字符，将其由ASCII码转化成相应的16进制数，并放到t2的个位。检测到非十六进制数字后循环终止，t2的数据被写入a0返回。程序在调用getchar后，临时寄存器t0被下一条语句重置因此两个模块不会出现临时寄存器的冲突。

putchar：输出一个存在a0中的字符。

```
putchar:
TRY:
lw t0,display_st
lw t0,(t0)
beq t0,x0,TRY
lw t0,display_data
sw a0,(t0)
jalr x0,0(ra)
```

程序不断读取0x7f08(display_st，显示器的ready bit)，读到1后说明显示器准备完毕，在0x7f0c中写入字符的ASCII码，随后0x7f08被自动置为0，字符成功显示后0x7f08又变回1，可以进行下一轮的显示。

printf：a0传入一个16进制数字，函数把它输出到屏幕上。

```
printf:
addi sp,sp,-24
sw ra,20(sp)
sw s3,16(sp)
sw s2,12(sp)
```

```
sw s1,8(sp)
sw s0,4(sp)
sw a0,0(sp)
lw gp,mask
addi s3,a0,0
PRINTF_FOR:
addi s0,x0,0 #upper most digit
GET_DIGIT:
lw s1,(gp)
beq s1,x0,END_PRINTF_FOR
bltu s3,s1,END_GET_DIGIT
sub s3,s3,s1
addi s0,s0,1
jal x0,GET_DIGIT
END_GET_DIGIT:
addi s2,x0,9
bge s2,s0,DIG
addi s0,s0,39
DIG:
addi s0,s0,48
addi a0,s0,0
addi gp,gp,4
jal ra,putchar
jal x0,PRINTF_FOR
END_PRINTF_FOR:
lw a0,0(sp)
lw s0,4(sp)
lw s1,8(sp)
lw s2,12(sp)
lw s3,16(sp)
lw ra,20(sp)
addi sp,sp,24
jalr x0,0(ra)
```

为了防止临时寄存器冲突，采用保存寄存器（这样可以在开头和末尾进行栈的操作，不容易出错），程序从最高位开始，每次读取一位数字，将其分情况转换成ASCII码后调用putchar输出。读取高位字符的方式是不断减去最高位为1、其他位都是0的mask，保证减完还是非负数。减了几次就说明最高位是几。由于减完以后的数会留到下一次使用，因此下一次的最高位就是这一次的次高位，以此类推会逐步获得所有位。

main：主函数，实现数据的输入、排序和输出。

```
main:
jal ra,scanf
addi s0,a0,0 #Length
addi s1,x0,0
lw s2,array #base address of the array
INPUT_ARRAY:
beq s1,s0,INPUT_ARRAY_END
jal ra,scanf
sw a0,(s2)
addi s2,s2,4
```

```
addi s1,s1,1
jal x0 INPUT_ARRAY
INPUT_ARRAY_END:
lw a0,array
addi a1,s0,0
jal ra,sort
lw s2,array #base address
addi s1,x0,0 #How many printed
PRINT_ARRAY:
beq s1,s0,END_PRINT_ARRAY
lw a0,(s2)
jal ra,printf
addi a0,x0,32
jal ra,putchar
addi s1,s1,1
addi s2,s2,4
jal x0,PRINT_ARRAY
END_PRINT_ARRAY:
EXIT:
```

.text段入口处会调用main函数。

完整代码：

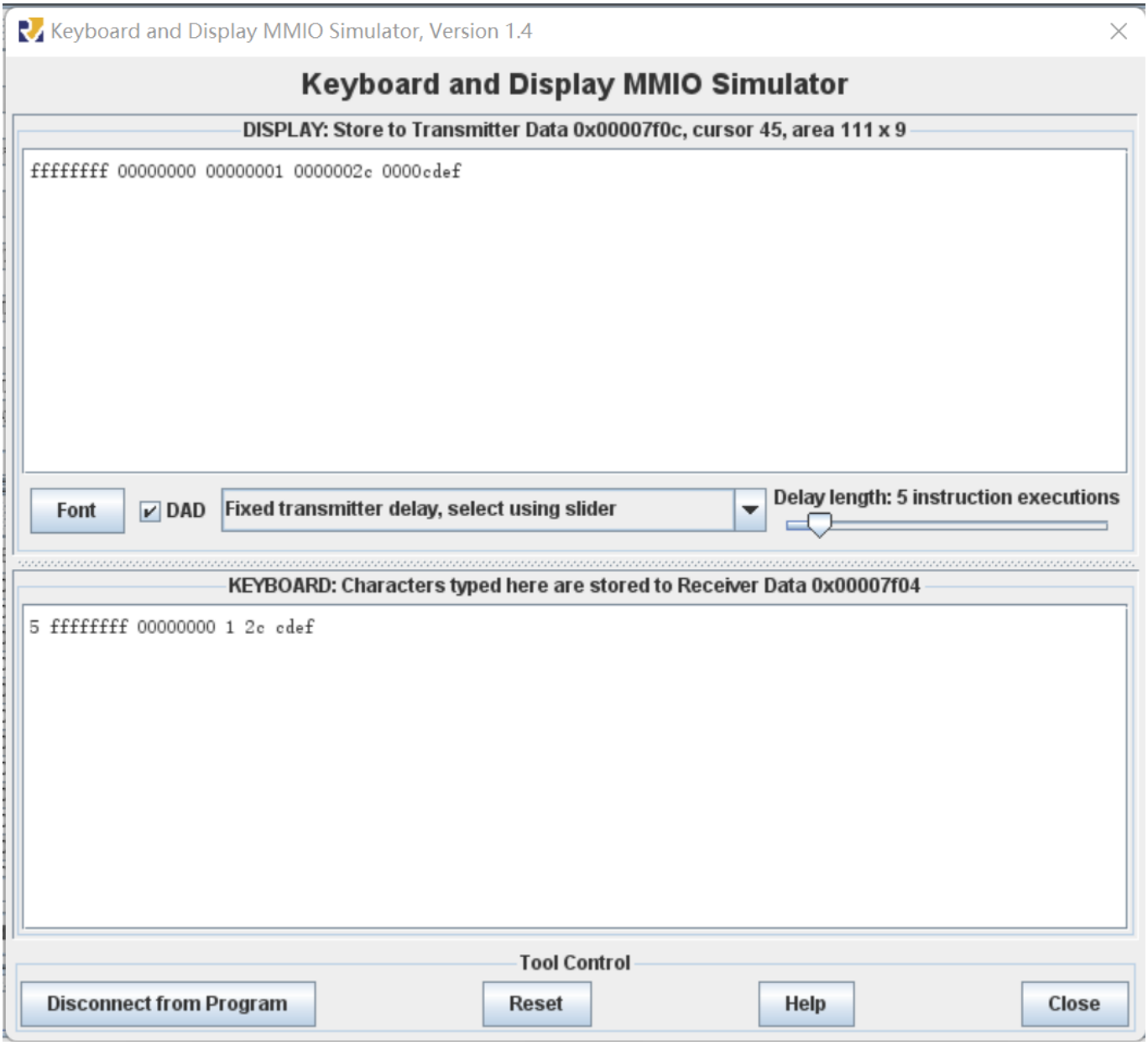
```
.data
array:.word 0x2000
mask:.word 0x0018
keybd_st:.word 0x7f00
keybd_data:.word 0x7f04
display_st:.word 0x7f08
display_data:.word 0x7f0c
mask1:.word 0x10000000
mask2:.word 0x01000000
mask3:.word 0x00100000
mask4:.word 0x00010000
mask5:.word 0x00001000
mask6:.word 0x00000100
mask7:.word 0x00000010
mask8:.word 0x00000001
.text
jal x0,main
#sort
sort: #a0 stores the base address,a1 stores the legnth of the array
addi t0,a0,0
add a1,a1,a1 #a1*=4
add a1,a1,a1
add t1,t0,a1
addi t2,t0,0 #i
OUTER_FOR:
beq t2,t1,END_OUTER
addi t3,t2,4 #j
INNER_FOR:
```

```
beq t3,t1,END_INNER
lw t4,(t2)
lw t5,(t3)
ble t4,t5,NEXT_INNER #no need to swap
addi t6,t4,0 #swap
addi t4,t5,0
addi t5,t6,0
sw t4,(t2) #restore
sw t5,(t3)
NEXT_INNER:
addi t3,t3,4 #j moves on
jal x0,INNER_FOR
END_INNER:
addi t2,t2,4 #i moves on
jal x0,OUTER_FOR
END_OUTER:
jalr x0,0(ra)
#getchar()
getchar:
getchar_LOOP:
lw t0,keybd_st
lw t0,(t0)
beq t0,x0,getchar_LOOP
lw a0,keybd_data
lw a0,(a0)
jalr x0,0(ra)
#input a integer
scanf:
addi sp,sp,-4
sw ra,(sp)
addi t2,x0,0 #result
scanf_LOOP:
jal ra,getchar
addi t0,x0,48 #'0'
addi t1,x0,57 #'9'
blt a0,t0,EXAM2
bgt a0,t1,EXAM2
jal x0,PROCESS_dig
EXAM2:
addi t0,x0,97 #'a'
addi t1,x0,102 #'f'
blt a0,t0,EXIT_scanf
bgt a0,t1,EXIT_scanf
addi a0,a0,-39
PROCESS_dig:
addi a0,a0,-48
add t2,t2,t2 #t2=t2<<4+int16(a0)
add t2,t2,t2
add t2,t2,t2
add t2,t2,t2
add t2,t2,a0
jal x0,scanf_LOOP
EXIT_scanf:
addi a0,t2,0
```

```
lw ra,(sp)
addi sp,sp,4
jalr x0,0(ra)
putchar:
TRY:
lw t0,display_st
lw t0,(t0)
beq t0,x0,TRY
lw t0,display_data
sw a0,(t0)
jalr x0,0(ra)
#Print a hexidecimal number
printf:
addi sp,sp,-24
sw ra,20(sp)
sw s3,16(sp)
sw s2,12(sp)
sw s1,8(sp)
sw s0,4(sp)
sw a0,0(sp)
lw gp,mask
addi s3,a0,0
PRINTF_FOR:
addi s0,x0,0 #upper most digit
GET_DIGIT:
lw s1,(gp)
beq s1,x0,END_PRINTF_FOR
bltu s3,s1,END_GET_DIGIT
sub s3,s3,s1
addi s0,s0,1
jal x0,GET_DIGIT
END_GET_DIGIT:
addi s2,x0,9
bge s2,s0,DIG
addi s0,s0,39
DIG:
addi s0,s0,48
addi a0,s0,0
addi gp,gp,4
jal ra,putchar
jal x0,PRINTF_FOR
END_PRINTF_FOR:
lw a0,0(sp)
lw s0,4(sp)
lw s1,8(sp)
lw s2,12(sp)
lw s3,16(sp)
lw ra,20(sp)
addi sp,sp,24
jalr x0,0(ra)
#main
main:
jal ra,scanf
addi s0,a0,0 #Length
```

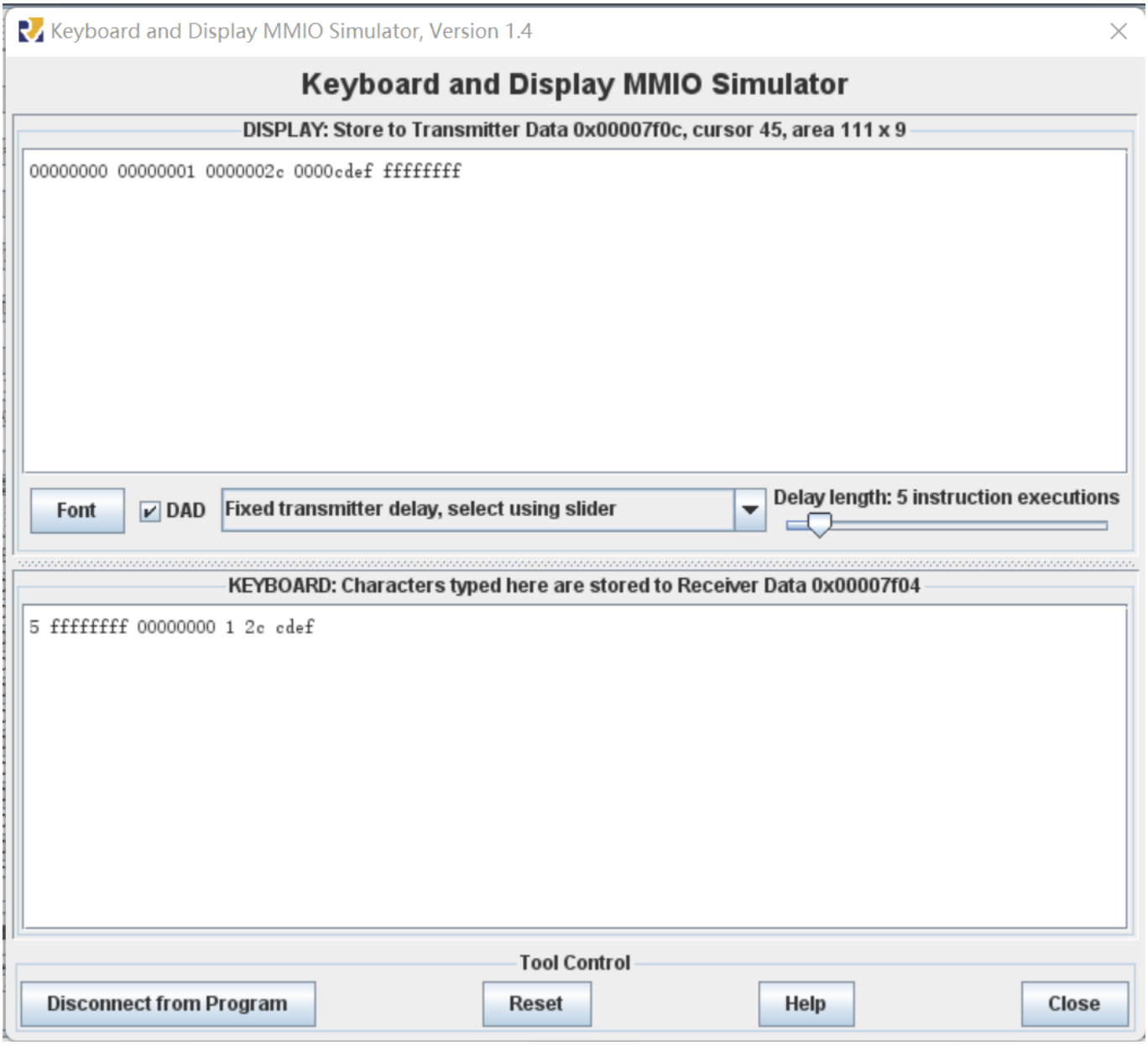
```
addi s1,x0,0
lw s2,array #base address of the array
INPUT_ARRAY:
beq s1,s0,INPUT_ARRAY_END
jal ra,scanf
sw a0,(s2)
addi s2,s2,4
addi s1,s1,1
jal x0 INPUT_ARRAY
INPUT_ARRAY_END:
lw a0,array
addi a1,s0,0
jal ra,sort
lw s2,array #base address
addi s1,x0,0 #How many printed
PRINT_ARRAY:
beq s1,s0,END_PRINT_ARRAY
lw a0,(s2)
jal ra,printf
addi a0,x0,32
jal ra,putchar
addi s1,s1,1
addi s2,s2,4
jal x0,PRINT_ARRAY
END_PRINT_ARRAY:
EXIT:
```

运行效果：



注意ffffffff（有符号-1）是最小的，原因是本程序默认有符号排序，如果是无符号排序，需要把sort的ble改成bleu。

无符号模式排序结果：



此时ffffffff变为最大。